VisualEDU: A Benchmark for Assessing Coding and Visual Comprehension through Educational Problem-Solving Video Generation

Hao Chen[1], **Tianyu Shi**[2,*], **Pengran Huang**[1], **Zeyuan Li**[1],
Jiahui Pan[1], **Qianglong Chen**[3], **Lewei He**[1,*]
[1]School of Artificial Intelligence, South China Normal University,
[2]University of Toronto, [3]Zhejiang University
helewei@m.scnu.edu.cn, tys@cs.toronto.edu

Abstract

Generating logically coherent video from text (T2V) for reasoning-intensive tasks like mathematical problem-solving presents a significant challenge for Vision-Language Models (VLMs). Therefore, we introduce VisualEDU, a benchmark based on Manim package to rigorously evaluate VLM capabilities in producing coherent, step-by-step video solutions for educational purposes, with a framework that integrates meta-prompt learning, visual and code feedback, and a modular drawing toolkit to enhance output quality. Novel metrics for temporal consistency, logical correctness, and visual clarity are proposed, and extensive experiments across nine VLMs reveal that while advanced proprietary models show promise, all struggle significantly with increasing task complexity (e.g., the performances of Claude-3.7-Sonnet and GPT-4o are below 56% on difficult tasks), highlighting limitations in code generation, visual feedback correction and precise tool invocation. VisualEDU offers a robust platform for systematic T2V assessment in reasoning-intensive domains and guides future VLM improvements in this area. All data related to VisualEDU is publicly available at https://github.com/UchihaIchigo/VisualEDU.

1 Introduction

Recent advances in generative AI have sparked growing interest in bridging the gap between natural language understanding and visual synthesis, giving rise to the challenging yet impactful task of text-to-video (T2V) generation. T2V holds significant promise for automated content creation, personalized education, enhanced accessibility, and creative expression (Wang et al., 2024b; FADIYA et al., 2025; Bar-Tal et al., 2024). The goal is to generate coherent sequences of video frames that faithfully and vividly depict a given textual prompt.

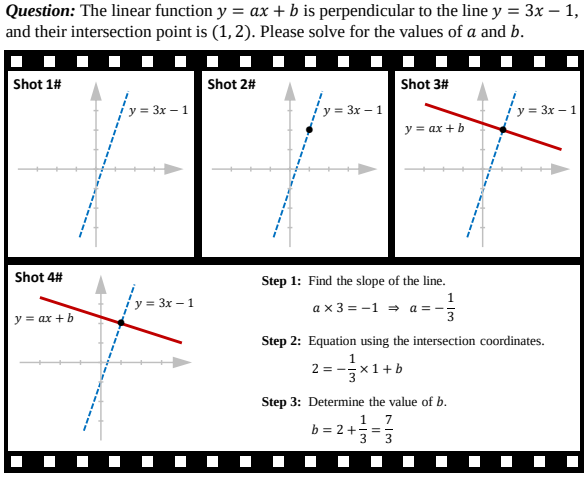

Figure 1: Problem-solving video for educational purposes powered by Manim package.

Achieving this involves multiple stages reflecting the complexity of modeling dynamic visual scenes from language. First, textual prompts are encoded via powerful pretrained language or Vision-Language Models (e.g., CLIP (Radford et al., 2021), CogVLM (Wang et al., 2024a), DeepSeek-VL (Joshi, 2025), PaliGemma 2 (Steiner et al., 2024)) to extract rich semantic representations. These are used to condition core generative models, typically latent diffusion models (LDMs) (Kim et al., 2023), which synthesize temporally coherent latent sequences. Architectures often include spatiotemporal attention or 3D convolutions (Zhang et al., 2025; Diba et al., 2023; Blattmann et al., 2023) to model dependencies across frames. A video decoder, usually derived from a VAE or a super-resolution module (Wang et al., 2025), converts latent codes into high-fidelity video frames. Beyond the mainstream text-to-video pipelines, researchers have also explored alternative strategies that require no additional training. Methods such as LCGD (Waseem et al., 2025) incorporate VLM into the diffusion process to improve semantic align-

*Corresponding authors. Emails: helewei@m.scnu.edu.cn, tys@cs.toronto.edu

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 16363–16394
November 4-9, 2025

ment and denoising, while ARTV (Weng et al., 2024) proposes an auto-regressive framework that sequentially generates frames to better handle temporal coherence and avoid drifting artifacts.

Despite progress from models like Lumiere (Bar-Tal et al., 2024), ModelScope (Wang et al., 2023), and Sora (FADIYA et al., 2025), current T2V systems face key limitations. Benchmarks such as T2vbench (Ji et al., 2024), FETV (Liu et al., 2023), Vbench (Huang et al., 2024b), Editboard (Chen et al., 2025), Evalcrafter (Liu et al., 2024), and Ve-bench (Sun et al., 2025) highlight persistent challenges, including temporal inconsistency, motion artifacts, object and background instability, poor alignment with textual descriptions, violations of fundamental physical principles, and difficulties in rendering textual elements accurately within videos.

Crucially, current evaluations focus on general content, overlooking domains that require structured logic and correctness—such as educational or instructional videos. Fine-grained control over camera motion, object trajectories, and symbolic reasoning has been largely underexplored, as existing models are primarily trained on unstructured internet-scale data (FADIYA et al., 2025; Huang et al., 2024b). To better address these challenges, we adopt a domain-specific solution—Manim (The Manim Community Developers, 2024; Sánchez et al., 2025), an open-source Python library widely used for creating mathematical animations and educational videos through programmatic control. Manim provides fine-grained specification of visual elements such as text, shapes, graphs, and equations, along with their corresponding animations, timings, and transitions. These capabilities make Manim particularly well-suited for generating videos that demand logical correctness, visual clarity, and consistent symbolic reasoning.

To address these gaps, we introduce VisualEDU, a benchmark tailored to evaluate VLMs' ability to generate logical and readable math problem-solving videos. Our contributions are fourfold:

- We introduce a novel benchmark designed to evaluate the coding and visual comprehension capabilities of VLMs through the task of educational problem-solving video generation.
- We propose a framework incorporating meta-prompt learning, Manim tool libraries, and visual feedback correction to improve structured video generation.
- Comprehensive evaluation metrics including Logical Pass Rate, Feedback Effectiveness and Aesthetic Score are designed for the multi-granular subtasks in the framework.
- We conduct extensive experiments across 9 VLMs (5 proprietary, 4 open-source), revealing two key findings: (i) all models degrade significantly as task complexity increases; (ii) models behave differently in code generation and visual feedback correction, exposing fundamental limitations in current VLM.

2 Related Works

2.1 VLM and Text-to-Video Agent

Recent T2V methods increasingly incorporate VLMs to improve temporal coherence, compositionality, and control. Dysen-VDM (Fei et al., 2024) leverages VLMs to construct dynamic scene graphs, enhancing temporal reasoning. GPT4Motion (Lv et al., 2024) employs GPT-4 to script Blender simulations, enabling physically grounded video generation. LCGD (Waseem et al., 2025) injects VLM guidance into diffusion denoising, improving semantic alignment. DriveDreamer-2 (Zhao et al., 2025) uses VLMs to synthesize structured driving scenes, while VideoTetris (Tian et al., 2024) proposes compositional attention mechanisms for multi-entity and long-form video generation. These approaches surpass early zero-shot methods like Text2Video-Zero (Khachatryan et al., 2023), demonstrating the growing role of VLMs in scalable, controllable T2V systems.

Despite these advances, a critical gap remains in evaluating VLMs' ability to generate videos requiring structured reasoning, procedural correctness, and symbolic clarity.

2.2 Coding and Text-to-Video Benchmarks

Evaluating text-to-video (T2V) models remains a critical and complex challenge, requiring metrics beyond pixel-level fidelity or basic classification scores. Early approaches often relied on proxy metrics computed on general-purpose datasets such as CLIP-based similarity (CLIPScore) (Radford et al., 2021). However, these metrics fail to capture essential attributes such as temporal coherence, action plausibility, and fine-grained text alignment.

To address these limitations, recent works have proposed specialized and comprehensive T2V

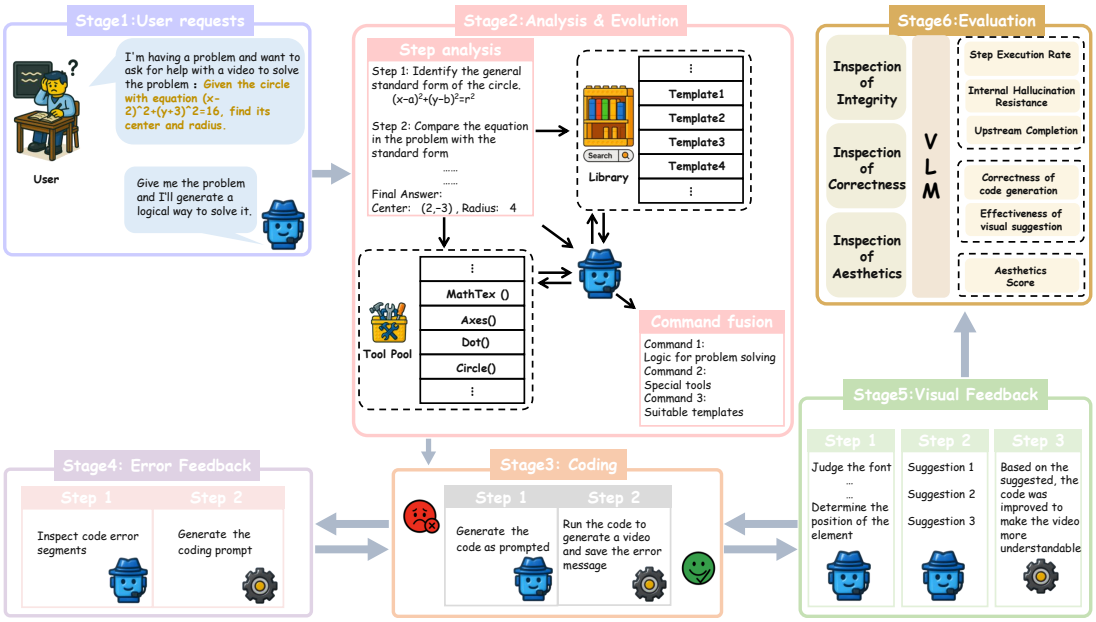

Figure 2: The framework of VisualEDU Benchmark.

benchmarks. VBench (Huang et al., 2024b) evaluates multi-dimensional aspects like subject/background consistency and temporal coherence via human ratings. FETV (Liu et al., 2023) focuses on fine-grained fidelity and action correctness using carefully designed prompts and manual assessments. EvalCrafter (Liu et al., 2024) integrates both automatic metrics and human preference judgments to assess motion quality, semantic alignment, and aesthetics. T2VBench (Ji et al., 2024) introduces a hierarchical framework with fine-grained temporal lexicons to systematically evaluate event, visual, and narrative dynamics. DEVIL (Liao et al., 2024) complements prior efforts by proposing dynamics-centered metrics and a dynamics-graded prompt benchmark, highlighting the critical role of vivid motion in faithful T2V generation.

While these benchmarks advance general-purpose T2V evaluation, they remain limited in domains requiring structured reasoning and logic, such as educational videos for mathematical problem-solving—where both semantic fidelity and procedural correctness are crucial.

3 VisualEDU Framework

We present the full VisualEDU framework (Figure 2), which consists of three core components: meta-prompt learning, feedback-enhanced flow, and a modular drawing tools library, designed to improve text-to-video generation for educational and reasoning-centric tasks.

3.1 Drawing Tools Library

To facilitate accurate code generation, we introduce a structured tool library that provides detailed documentation and usage guidelines for various Manim modules. This knowledge base allows the VLM to evaluate the solution process and selectively retrieve appropriate modules to visualize each step.

By grounding code generation in the tool library, we significantly reduce the likelihood of hallucinated or syntactically invalid code. Incorporating domain-specific knowledge from the library also improves the code execution success rate, as the retrieved templates align closely with task semantics and animation requirements.

3.2 Meta-Prompting with Tool-Integrated

Our method generates complex video editing prompts through a structured workflow (Figure 3). The system first decomposes high-level editing tasks into smaller, modular subtasks to reduce complexity. For each subtask, it retrieves the most relevant meta-prompts from a dynamic template library via semantic matching, optionally leveraging tool reasoning to refine sub-prompt generation.

Sub-prompts are then sequentially composed based on logical dependencies, with connective information inserted to form a coherent final prompt. A meta-learning feedback loop further enhances the system: upon successful execution, novel sub-prompt structures are generalized and integrated into the template library.

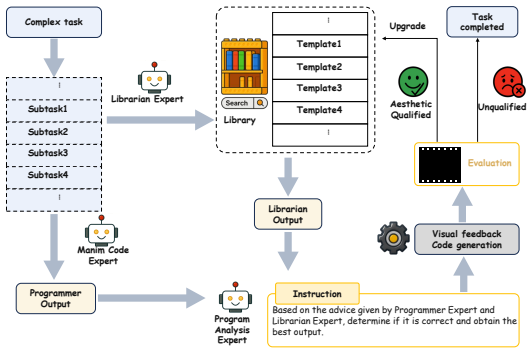

Figure 3: Meta-Prompting Flowchart. In Meta-Prompting, collaboration with experts from multiple domains ultimately resolves complex tasks.

This closed-loop process enables efficient task decomposition, improves prompt quality through adaptive retrieval, and continuously evolves by learning from successful outcomes, facilitating precise and scalable video creation.

3.3 Visual-Feedback

To enable VLMs to produce logically coherent and visually polished educational videos, we propose a multi-round visual feedback mechanism. The model provides lightweight suggestions—such as adjusting element positions, font sizes, and text-graphic alignment—based on intermediate renderings.

These suggestions guide iterative refinement of the generated Manim code, improving both functional correctness and visual quality. This feedback-driven process mitigates common issues like element overlap, inconsistent scaling, and disorganized layouts, ensuring that final videos achieve both pedagogical clarity and aesthetic appeal.

4 Data Construction

To facilitate the systematic evaluation of VLMs in mathematical reasoning, code generation, and visual communication, we construct VisualEDU, a multimodal dataset for generating and refining math visualizations from natural language problem prompts. The dataset spans diverse mathematical domains and encodes the full transformation pipeline from problem comprehension to aesthetic video rendering.

4.1 Problem-Type Guided Design

VisualEDU categorizes problems into six types: Functions, Plane Geometry, Analytic Geometry, Solid Geometry, Probability, and Calculus, each posing distinct reasoning and visualization challenges for evaluating VLMs' symbolic and spatial abilities. Function problems evaluate the ability of VLMs to generate dynamic scenes (e.g., Graph, Arrow, Transform) representing functional structures. Geometric problems assess spatial reasoning across 2D and 3D geometries through object placement, animation timing, and layout coherence. Probability and calculus problems test the depiction of uncertainty and continuous change using visualizations such as bar charts, point arrays, and area shading.

4.2 Stage-wise Construction Pipeline

Each VisualEDU sample is constructed through a structured multi-step pipeline, capturing intermediate reasoning, code, and visualization artifacts.

Our pipeline follows a structured workflow: VLMs first derive structured solutions from math problems, verified against expert-annotated ground truths. Based on the problem type, they select appropriate components and retrieve visualization templates through a meta-learning system. The models then synthesize executable Manim code, iteratively correcting errors from initial drafts to final renders. Rendered videos are segmented into keyframes, allowing VLMs to diagnose visual layout issues and propose refinements. Finally, the code is revised according to these diagnostics, and the resulting videos are evaluated by both human annotators and VLMs for clarity, visual appeal, and explanatory effectiveness.

Each data point in VisualEDU includes: original problem, step-by-step solution, Manim code (initial and final), rendered videos, keyframes, layout critiques, and multimodal quality scores.

4.3 Data Statistics

Table 1 shows task distribution across categories and difficulty levels. Each category includes balanced samples of varying complexity (Easy, Medium, Hard), determined by number of steps, required animation layers, and layout sophistication. Further breakdowns of code correction frequency, keyframe count, and scoring variance are provided in Appendix A.5.

5 Evaluation

In this section, we introduce the evaluation of the VisualEDU benchmark, including the design principles and metrics concerned. Unlike previous

Category	Difficulty level			Overall
	Easy	Middle	Hard	
Function Problems	12	11	11	34
Plane Geometry	12	11	11	34
Analytic Geometry	11	11	11	33
Solid Geometry	11	11	11	33
Probability	11	11	11	33
Calculus	11	11	11	33
Total	68	66	66	200

Table 1: Statistics of VisualEDU.

benchmarks, we employ novel metrics to assess the problem-solving video generation capabilities of VLMs, with an emphasis on the readability of video logic.

5.1 Design Principles

To comprehensively assess the video generation capabilities of VLMs for solving text-based problems, we evaluate them with the following principles:

- Breadth and Reality: Covering a wide range of problem types, domains, and difficulty levels to evaluate generalization and robustness.

- Depth and Logical Coherence: Multiple-round interaction to ensure the generated video adapts and refines its explanations based on additional contextual information or feedback. Emphasizing the need for VLMs to generate videos that clearly articulate the step-by-step solution to a given problem.

5.2 Metrics

Generating problem-solving videos requires a multi-dimensional evaluation, particularly when assessing both the overall structure and the detailed steps of the solution. To ensure a comprehensive evaluation, we propose a three-tiered framework that progressively evaluates the task from macro to micro, focusing on the following dimensions:

5.2.1 Logical Pass Rate

In the first stage of evaluation, we assess the overall performance of the model at a macro level by verifying whether key indicators necessary for video generation are successfully achieved. Subsequently, we decompose the video generation task into finer-grained subtasks and evaluate both the completion and the independence of each subtask. This ensures that the model not only produces a coherent final output but also demonstrates the ability to handle intermediate steps in a structured and modular manner.

Step Execution Rate. From a macro perspective, this phase evaluates the coverage of subtasks by the generated video. To more accurately reflect system efficiency, we extend the Step Execution Rate (SER) metric by incorporating retry penalties and enforcing dependency constraints between subtasks:

$$SER = \frac{\sum_{i=1}^{n_{st}} w_i \cdot c_i \cdot q_i \cdot p(r_i) \cdot \text{Dep}_i}{\sum_{i=1}^{n_{st}} w_i} \quad (1)$$

where n_{st} is the number of total subtasks; w_i denotes the importance weight of subtask i; c_i is a binary indicator of subtask completion (1 if completed, 0 otherwise); $q_i \in [0, 1]$ represents the quality score of subtask execution; $p(r_i) = a - b\ln(r_i + 1)$ applies a retry penalty, where a denotes the initial score and b controls the decay rate based on the number of retries r_i; $\text{Dep}_i \in \{0, 1\}$ indicates whether all prerequisite subtasks required for i have been successfully completed.

Internal Hallucination Resistance. In the evaluation of the generated videos, a primary focus is placed on the VLM's handling of task dependencies. To assess this capability, we introduce a metric called Internal Hallucination Resistance (IHR), which is defined as:

$$IHR = \sum_{i=1}^{m} (1 - \lambda_i \cdot (B_i \cdot (1 - s_i))) \cdot w_i \quad (2)$$

where n_{st} denotes the total number of subtasks; B is the Boolean set for the dependency; $B_i = 1$ indicates "hallucination"; $s_i \in [0, 1]$ indicate whether the preceding dependent task has failed ($s_i = 1$) or succeeded ($s_i = 0$). This term is introduced to disambiguate hallucination sources that stem from upstream execution errors. A score closer to 1 indicates higher semantic alignment. λ_i is the penalty factor for the i-th dependency, where a value of 1 indicates a completely incorrect dependency. w_i represents the weight of the i-th subtask, indicating its importance in the dependency task.

Combined evaluation of these dimensions determines whether a task is successfully completed or contains redundant steps due to incorrect dependencies. We compute the Logical Pass Rate (LPR) by multiplying the Step Execution Rate (SER) with Internal Hallucination Resistance (IHR), providing

a quantitative measure of whether the execution of the plan adheres to the intended task logic.

5.2.2 Feedback Effectiveness

In the second evaluation stage, we perform a detailed and fine-grained analysis of the accuracy of essential procedural steps. This assessment rigorously examines the effectiveness of the visual feedback recommendations produced by the Vision-Language Model (VLM), as well as the existence of any errors in the generated code that could potentially obstruct the successful execution of subsequent steps.

We define Feedback Effectiveness (FE) metrics:

$$FE = \alpha \cdot \sum_{v=1}^{|B|} w_v \cdot B_v + \beta \cdot (!B_c) \cdot (1 - \sum_{c=1}^{|N|} w_c \cdot \frac{N_c}{N_{max}}) \quad (3)$$

where $B_v \in \{0, 1\}$ are binary visual evaluation indicators representing issues such as Elements Overlap, Inconsistent Font, and Layout Misalignment, collectively denoted as $|B| = \{B_{EO}, B_{IF}, B_{LM}\}$. $N_c \geq 0$ are count-based code error indicators, including Syntax Errors and Parameter Errors, denoted as $|N| = \{N_{SE}, N_{PE}\}$.

The terms w_v and w_c refer to the normalized weights assigned to the visual and code components, respectively. Balancing coefficients $\alpha, \beta \in [0, 1]$ are used to weight the visual and code contributions, subject to the constraint $\alpha + \beta = 1$. Additionally, $!B_c$ is a binary indicator that evaluates to 0 if essential code is missing, and 1 otherwise. N_{max} defines the upper bound for the total number of code errors allowed.

5.2.3 Aesthetics Score

In the third evaluation stage, we define the overall aesthetic score based on a weighted combination of multiple positive and negative visual quality indicators. The Aesthetic Score (AS) is computed as:

$$AS = \frac{\sum_{i=1}^{m} w_i \cdot P_i + \sum_{j=1}^{n} v_j \cdot (1 - N_j)}{\sum_{i=1}^{m} w_i + \sum_{j=1}^{n} v_j} \quad (4)$$

where m and n denote the number of positive and negative indicators, respectively; P_i and N_j represent the i-th positive indicator and the j-th negative indicator, both taking binary values in $\{0, 1\}$; Specifically, w_i and v_j denote the respective weights assigned to each P_i and N_j. Positive indicators focus on desirable properties such as temporal logic coherence, answer accuracy, and readability, whereas negative indicators capture deficiencies including overlap, font inconsistency, layout error, and display incompleteness.

6 Experiments

6.1 Baselines

We evaluate 9 current mainstream VLMs, consisting of five proprietary models and four open-source models (detailed descriptions on these models can be found in Appendix B.3). The five proprietary models include Llama-3.2-11B-Vision-Instruct (Huang et al., 2024a), Llama-3.2-90B-Vision-Instruct (Huang et al., 2024a), Qwen2.5-VL-72B-Instruct, Qwen2.5-VL-7B-Instruct (Bai et al., 2025). For open-source models, we utilize GPT-4o-Mini (OpenAI, 2024a), GPT-4o-11-20 (OpenAI, 2024b), Grok-2-Vision-1212 (xAI, 2024), Claude-3-7-Sonnet-20250219 (Anthropic, 2025) and Gemini-2.5-Pro-Exp-03-25 (Google DeepMind, 2025). The performance results across these models are compared based on the capability to generate high-quality mathematical video explanations from textual problems.

6.2 Evaluation Settings

To ensure consistency and reproducibility, all models are accessed via their official APIs with standardized interactions and minimized inconsistencies. Keyframe images are provided via URLs rather than embedded as Base64 strings. The temperature is fixed at 0 across all models, with other parameters kept at default settings to ensure fair comparisons.

6.3 Main Results

As shown in Table 2, we present the overall performance of various VLMs on the VisualEDU benchmark. The detailed computation of the metrics listed in the table can be found in the Appendix B.4. In the comprehensive performance evaluation, the proprietary model Claude-3.7-Sonnet achieves the highest accuracy of 64.9%, outperforming all other models. Meanwhile, open-source models still lag significantly behind proprietary ones. We observe a clear trend that larger models exhibit stronger capabilities—for example, Llama-3.2-90B-Vision-Instruct outperforms Llama-3.2-11B-Vision-Instruct by approximately 10%.

Category	LPR			FE		AS	Overall
	SER	IHR	UC	CC	VA		
Open-weight							
Qwen2.5-VL-72B-Instruct	0.5817	0.6820	0.7317	0.4448	0.3800	0.6205	0.5120
Qwen2.5-VL-7B-Instruct	0.3004	0.4665	0.5433	0.2764	0.0750	0.6620	0.3903
Llama-3.2-90B-Vision-Instruct	0.5461	0.7390	0.8467	0.6664	0.2067	0.5160	0.4980
Llama-3.2-11B-Vision-Instruct	0.3720	0.5990	0.7033	0.4882	0.2967	0.5390	0.3951
Proprietary							
Claude-3-7-Sonnet-20250219	**0.6551**	**0.798**	0.855	0.7164	**0.7967**	0.7455	**0.6493**
Gemini-2.5-pro-exp-03-25	0.5422	0.7500	0.9117	0.7364	0.6583	**0.8210**	0.6207
GPT-4o-11-20	0.6226	0.7685	**0.9383**	**0.8598**	0.715	0.6875	0.6149
GPT-4o-mini	0.5772	0.7745	0.8750	0.7294	0.4917	0.6820	0.5953
Grok2-vision	0.5197	0.7480	0.8667	0.7076	0.5983	0.6630	0.5518

Table 2: Performance of various VLMs on the VisualEDU benchmark. Specifically, SER represents the Step Execution Rate, IHR denotes the Internal Hallucination Resistance, and UC measures Upstream Completion. CC and VA respectively represent Code Correctness and Visual Accuracy. The final score is computed through a weighted aggregation of Logic Pass Rate (LPR), Feedback Effectiveness (FE), and the Aesthetic Score (AS).

Claude-3.7-Sonnet demonstrates exceptional performance on SER and IHR metrics, achieving the highest pass rates in individual subtasks. At the local indicator level, GPT-4o shows notable advantages in reducing code errors and improving meta-template matching, while Claude-3.7-Sonnet maintains its lead in visual feedback accuracy. Additionally, content generated by Gemini-2.5-pro is more aligned with human aesthetic preferences, leading to higher visual appeal scores.

Additionally, detailed performance metrics for all models are reported in Appendix B.4, while implementation specifics of the video generation pipeline are provided in Appendix B.2.

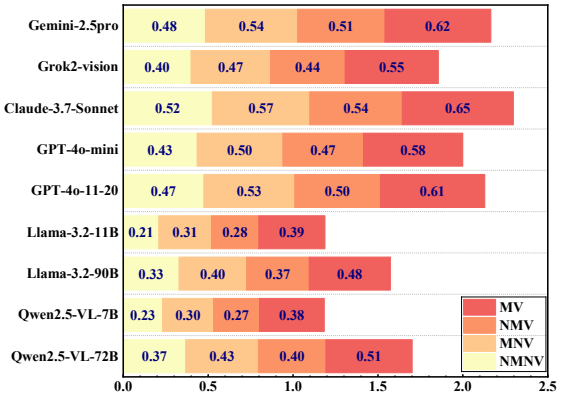

Figure 4: Ablation results. M: with Meta-Prompt, NM: without Meta-Prompt; V: with Visual Feedback, NV: without Visual Feedback.

6.4 Ablation Analysis

To evaluate the impact of our framework on video quality, we conduct an ablation study as shown in Figure 4. Several large models such as Qwen-2.5 and GPT-4o-mini show a sharp decline in performance in the absence of meta templates or visual feedback, highlighting the crucial role of these components in guiding VLMs to generate appropriate video content—particularly for complex reasoning tasks. For Claude-3.7-Sonnet, introducing meta-prompting results in a 5% overall accuracy improvement, while visual feedback contributes an additional 3% gain. Other models also exhibit substantial improvements, with combined performance increases ranging from 10% to 20% after incorporating these enhancements. Detailed results are provided in the Appendix B.4.

6.5 Error Analysis

6.5.1 Error Analysis by Task Difficulty Level

As shown in Figure 5, we conduct a comparative analysis of model performance across different difficulty levels in video generation tasks. The analysis leads to the following conclusions: text complexity significantly affects the generation of visually appealing and logically coherent videos. The Aesthetic Score is found to be negatively correlated with input task complexity, with the most pronounced performance degradation occurring in the "hard" difficulty group. The Aesthetic Score of most models drops significantly in this group, indicating that current VLMs still face limitations in handling complex reasoning and generation.

As task complexity increases, VLMs inevitably require invoking a larger number of Manim tool modules to complete video generation. Conse-

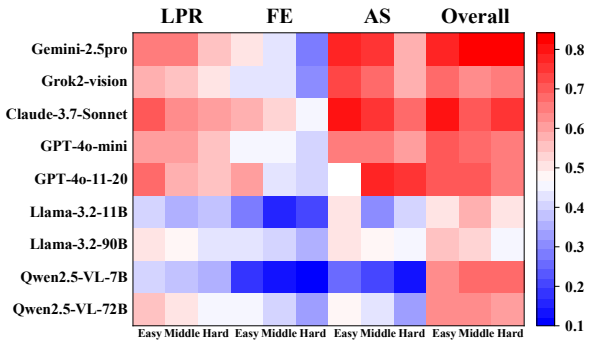

Figure 5: Comparative analysis of different models under various evaluation indicators, including Logic pass rate (LPR), Feedback Effectiveness (FE), Aesthetics Score (AS) and Overall.

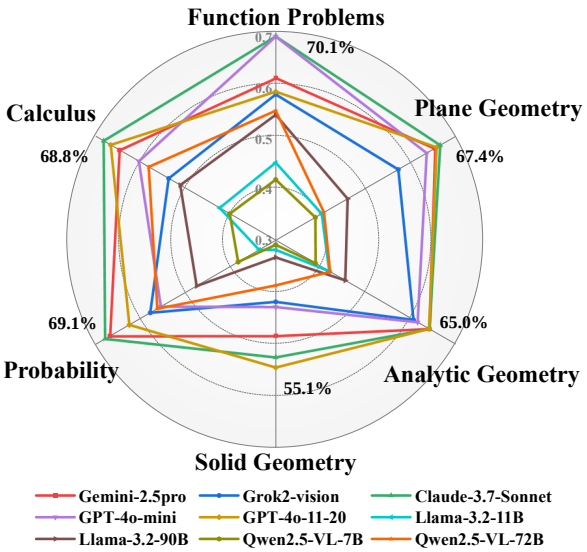

Figure 6: Comparison of the performance of different models in different scenarios. Percentage values indicate the success rate for each task.

quently, the models must accurately extract information from high-density sources such as task descriptions and keyframe specifications. This leads to a higher frequency of tool invocation errors, including incorrect parameter configurations and calls to nonexistent modules. These issues reflect the fragility of the current context-to-tool parameter mapping mechanisms in complex tasks.

6.5.2 Error Analysis by Application Scenario

As illustrated in Figure 6, we evaluate task completion performance across various problem scenarios. The results reveal that even the best-performing model, Claude-3.7-Sonnet, achieves only 55% accuracy in tool-intensive scenarios such as 3D geometry visualization, underscoring its limitations in practical task representation. Specifically, the model struggles to accurately interpret task requirements, generate content with correct logical and temporal sequencing, and frequently exhibits high parameter configuration error rates and insufficient graphical descriptions.

In other task scenarios, the top-performing models achieve similar levels of accuracy; however, a substantial gap remains between proprietary and open-source models. For example, proprietary models consistently reach over 60% accuracy in analytic and plane geometry tasks. In contrast, open-source models lag significantly behind. Furthermore, probability-related problems, which require the representation of various scenarios, result in much lower performance for smaller models, for example, Llama-3.2-11B only achieves 30% accuracy.

To summarize, current VLMs encounter significant challenges when handling tasks characterized by high module complexity, rich scenario-based descriptions, and intensive logical reasoning. These findings point to critical areas for improvement of VLMs, including robust tool invocation strategies, enhanced mathematical reasoning and code translation abilities, and better alignment with real-world application scenarios.

7 Conclusion

We introduce VisualEDU, a novel benchmark and dataset designed to evaluate VLMs' ability to generate structured educational videos. Alongside, we propose a specialized framework that integrates meta-prompt learning, feedback enhancement, video evaluation, and a modular drawing tool library.

Experiments show that while advanced models demonstrate promise, significant challenges persist, particularly for open-source VLMs. Key bottlenecks include handling complex tasks, long-context reasoning, precise tool invocation, and logical reasoning in mathematical domains. Ablation studies further confirm the critical role of meta-templates and visual feedback mechanisms in guiding VLMs during complex reasoning processes.

VisualEDU establishes a new platform for advancing text-to-video generation and systematically identifies key challenges for VLMs, including tool use, mathematical and code reasoning, and real-world alignment. Future work will explore reinforcement learning-enhanced frameworks to improve generalization and adaptability.

Limitations

Our research and resource release is limited to English, and is limited to generating math problem solving videos only. In the future, we hope to be able to generate videos in a variety of disciplines and gradually increase the difficulty of the questions. Our benchmarks need to be evaluated from a more diverse perspective and compared with domain-specific models. Additionally, future work will explore enhancing code generation capabilities via fine-tuning Vision-Language Models to improve their performance on our benchmark.

Ethics Statement

Misuse of our methods can produce harmful content that can have dependent side effects on teens' learning. This work and its assets are strictly for research purposes and against any harmful applications. Moderation and filtering mechanisms are recommended to curb suspicious content. All data used in this study are collected directly by the authors. The data used in this study have been thoroughly inspected to ensure that they contain no personally identifiable information or offensive content. Comprehensive human evaluation guidelines are provided, with explicit instructions to exclude any data involving illegal or criminal content.

Acknowledgements

This work was conducted as part of the research on educational applications of multimodal models within the Algorithm Team at Beijing Detianyuansheng Technology Co., Ltd., an education-focused startup in China. We sincerely appreciate the support from our colleagues and their constructive feedback , which contributed to the development of this study.

This work was also supported by the National Nature Science Foundation of China under Grant 52308250, the STI 2030-Major Projects under grant 2022ZD0208900, and the Guangdong Basic and Applied Basic Research Foundation under grant 2023A1515140100.

References

Anthropic. 2025. Claude 3.7 sonnet and claude code. Accessed: 2025-05-13.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. 2025. Qwen2.5-vl technical report. *Preprint*, arXiv:2502.13923.

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

Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. 2023. Align your latents: High-resolution video synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22563–22575.

Yupeng Chen, Penglin Chen, Xiaoyu Zhang, Yixian Huang, and Qian Xie. 2025. Editboard: Towards a comprehensive evaluation benchmark for text-based video editing models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 15975–15983.

Ali Diba, Vivek Sharma, Mohammad Arzani, Luc Van Gool, et al. 2023. Spatio-temporal convolution-attention video network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 859–869.

PE FADIYA, SUKRITH LAL PS, and KM SHEENA. 2025. Sora ai: The future of video generation. *Authorea Preprints*.

Hao Fei, Shengqiong Wu, Wei Ji, Hanwang Zhang, and Tat-Seng Chua. 2024. Dysen-vdm: Empowering dynamics-aware text-to-video diffusion with llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7641–7653.

Google DeepMind. 2025. Gemini 2.5: Our most intelligent ai model. Accessed: 2025-05-13.

Kunal Chawla Huang, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, A Lavender, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, et al. 2024a. The llama 3 herd of models. *Preprint*.

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

Pengliang Ji, Chuyang Xiao, Huilin Tai, and Mingxiao Huo. 2024. T2vbench: Benchmarking temporal dynamics for text-to-video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5325–5335.

Satyadhar Joshi. 2025. A comprehensive review of deepseek: Performance, architecture and capabilities.

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

Seung Wook Kim, Bradley Brown, Kangxue Yin, Karsten Kreis, Katja Schwarz, Daiqing Li, Robin Rombach, Antonio Torralba, and Sanja Fidler. 2023. Neuralfield-ldm: Scene generation with hierarchical latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8496–8506.

Mingxiang Liao, Qixiang Ye, Wangmeng Zuo, Fang Wan, Tianyu Wang, Yuzhong Zhao, Jingdong Wang, Xinyu Zhang, et al. 2024. Evaluation of text-to-video generation models: A dynamics perspective. *Advances in Neural Information Processing Systems*, 37:109790–109816.

Yaofang Liu, Xiaodong Cun, Xuebo Liu, Xintao Wang, Yong Zhang, Haoxin Chen, Yang Liu, Tieyong Zeng, Raymond Chan, and Ying Shan. 2024. Evalcrafter: Benchmarking and evaluating large video generation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22139–22149.

Yuanxin Liu, Lei Li, Shuhuai Ren, Rundong Gao, Shicheng Li, Sishuo Chen, Xu Sun, and Lu Hou. 2023. Fetv: A benchmark for fine-grained evaluation of open-domain text-to-video generation. *Advances in Neural Information Processing Systems*, 36:62352–62387.

Jiaxi Lv, Yi Huang, Mingfu Yan, Jiancheng Huang, Jianzhuang Liu, Yifan Liu, Yafei Wen, Xiaoxin Chen, and Shifeng Chen. 2024. Gpt4motion: Scripting physical motions in text-to-video generation via blender-oriented gpt planning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1430–1440.

OpenAI. 2024a. Gpt-4o mini: advancing cost-efficient intelligence. Accessed: 2025-05-13.

OpenAI. 2024b. Hello gpt-4o. Accessed: 2025-05-13.

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

Ivonne C Sánchez et al. 2025. Projeto am2: Animações matemáticas com manim. *Paradigma*, 46(1).

Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, Siyang Qin, Reeve Ingle, Emanuele Bugliarello, Sahar Kazemzadeh, Thomas Mesnard, Ibrahim Alabdulmohsin, Lucas Beyer, and Xiaohua Zhai. 2024. Paligemma 2: A family of versatile vlms for transfer. *Preprint*, arXiv:2412.03555.

Shangkun Sun, Xiaoyu Liang, Songlin Fan, Wenxu Gao, and Wei Gao. 2025. Ve-bench: Subjective-aligned benchmark suite for text-driven video editing quality assessment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 7105–7113.

The Manim Community Developers. 2024. Manim – Mathematical Animation Framework. `https://www.manim.community/`. Accessed: 2024-05-13.

Ye Tian, Ling Yang, Haotian Yang, Yuan Gao, Yufan Deng, Xintao Wang, Zhaochen Yu, Xin Tao, Pengfei Wan, Di ZHANG, et al. 2024. Videotetris: Towards compositional text-to-video generation. *Advances in Neural Information Processing Systems*, 37:29489–29513.

Fan Wang, Chaochao Chen, Weiming Liu, Minye Lei, Jintao Chen, Yuwen Liu, Xiaolin Zheng, and Jianwei Yin. 2025. Dr-vae: Debiased and representation-enhanced variational autoencoder for collaborative recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 12703–12711.

Jiuniu Wang, Hangjie Yuan, Dayou Chen, Yingya Zhang, Xiang Wang, and Shiwei Zhang. 2023. Modelscope text-to-video technical report. *arXiv preprint arXiv:2308.06571*.

Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Song XiXuan, et al. 2024a. Cogvlm: Visual expert for pretrained language models. *Advances in Neural Information Processing Systems*, 37:121475–121499.

Xiang Wang, Shiwei Zhang, Hangjie Yuan, Zhiwu Qing, Biao Gong, Yingya Zhang, Yujun Shen, Changxin Gao, and Nong Sang. 2024b. A recipe for scaling up text-to-video generation with text-free videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6572–6582.

Muhammad Waseem, Muhammad Usman Ghani Khan, and Syed Khaldoon Khurshid. 2025. Lcgd: Enhancing text-to-video generation via contextual llm guidance and u-net denoising. *IEEE Access*.

Wenming Weng, Ruoyu Feng, Yanhui Wang, Qi Dai, Chunyu Wang, Dacheng Yin, Zhiyuan Zhao, Kai Qiu, Jianmin Bao, Yuhui Yuan, et al. 2024. 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.

xAI. 2024. Bringing grok to everyone. Accessed: 2025-05-13.

Xiaorui Zhang, Qijian Xie, Wei Sun, and Ting Wang. 2025. Fall detection method based on spatio-temporal coordinate attention for high-resolution networks. *Complex & Intelligent Systems*, 11(1):1.

Guosheng Zhao, Xiaofeng Wang, Zheng Zhu, Xinze Chen, Guan Huang, Xiaoyi Bao, and Xingang Wang. 2025. Drivedreamer-2: Llm-enhanced world models for diverse driving video generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 10412–10420.

A Design

A.1 Procedure of VisualEDU

We propose a collaborative multi-agent architecture to automate the end-to-end pipeline for transforming mathematical problems into high-quality instructional videos. The overall workflow consists of a structured sequence of stages, wherein agents interact to progressively construct, refine, and validate visualized mathematical solutions. While each phase is guided by specialized prompts and agent roles, the process supports multi-turn interactions rather than a rigid Planner/Responder framework. The stages are summarized as follows:

- comprehending the mathematical problem and generating a step-by-step textual solution;
- selecting appropriate Manim visualization modules based on the solution structure and task type;
- optionally retrieving similar problems and reference solutions to inform the generation process;
- synthesizing initial Manim code by integrating the problem, derived solution, selected modules, and references;
- iteratively debugging and refining the code to resolve execution errors;
- performing visual inspection of the generated output (potentially using multi-modal inputs);
- further optimizing the code based on visual feedback to enhance the final animation quality.

Each stage is executed by a specialized agent with the following core responsibilities:

(1) **Solve Agent: Logical Reasoning Generator.** This agent parses the input math problem into a structured, step-by-step textual explanation through natural language understanding and symbolic inference. The resulting formal derivation and supporting narration serve as the foundational content for the instructional video and are passed to the next agent.

(2) **Scripting & Resource Allocation Agent.** This agent plans the visual rendering by selecting suitable `Manim` modules and reference templates from the Meta Prompt Library based on the logical structure provided by the Solve Agent. It composes a composite instruction prompt for code generation and maintains the Meta Prompt Library by updating it with high-quality exemplars.

(3) **Code Generation & Debugging Agent.** This agent uses a VLM to generate executable Python code from the instruction prompt and attempts to render the animation. In case of errors, it captures the logs and iteratively refines the code via revised prompts until successful rendering is achieved. The resulting video and code are forwarded for visual evaluation.

(4) **Visual Evaluation & Refinement Agent.** This agent assesses the visual quality of the output with respect to overlap, font consistency, layout coherence, and animation clarity. It generates structured feedback to improve code via further VLM interactions. This loop continues until the video meets predefined quality thresholds.

(5) **Dynamic Update Mechanism.** Validated videos and their associated code snippets—along with contextual metadata such as problem type and visual goals—are stored in the Meta Prompt Library. This repository supports prompt reuse and enhances future generation quality through cumulative learning.

This multi-agent system, guided by modular prompts and visual feedback, enables robust orchestration of VLM capabilities for instructional video generation. The prompts used at each stage are presented in Appendix B.4.

A.2 Manim tool pool

To enhance the fidelity of Manim library module utilization in code synthesized by VLMs, we introduce a methodology employing a JSON-formatted compendium of extant Manim modules. This structured resource constrains the VLM's selection to canonical library components, thereby mitigating the propensity for 'hallucination'—the erroneous generation of non-existent or user-defined modules—and improving the overall correctness of the generated code.

```
{
  "Middle School Mathematics Topics": {
    "Numbers and Algebra": {
      "Understanding Numbers and Operations": {
        "Used for constructing number lines, displaying digits and basic operation animations": [
          "NumberLine (Manim.mobject.numbers)",
          "DecimalNumber (Manim.mobject.numbers)",
          "Integer (Manim.mobject.numbers)"
        ],
        "Displaying arithmetic expressions, formulas and calculation steps": [
          "MathTex (Manim.mobject.tex_mobject)",
          "Tex (Manim.mobject.tex_mobject)"
        ],
        "Controlling the dynamic appearance and transformation of numbers and formulas": [
          "Write, FadeIn, Transform (Manim.animation)"
        ]
      },
      "Algebraic Expressions and Operations": {
        "Used for rendering fractions, algebraic expressions and factorization formulas": [
          "MathTex / Tex (Manim.mobject.tex_mobject)"
        ],
        "Demonstrating the transformation and simplification process of expressions": [
          "Transform, ReplacementTransform (Manim.animation.transform)"
        ]
      },
      "Equations and Inequalities": {
        "Used for displaying equations, inequalities and solution steps": [
          "MathTex / Tex (Manim.mobject.tex_mobject)"
        ],
        "Constructing coordinate systems to facilitate visual problem solving": [
          "Axes, NumberPlane (Manim.mobject.coordinate_systems)"
        ],
        "Displaying the graph of an equation and the intuitive location of its roots": [
          "FunctionGraph (Manim.mobject.graph)",
          "ParametricFunction (Manim.mobject.parametric_function)"
        ]
      }
    },
    "Shapes and Geometry": {
      "Basics of Plane Geometry": {
        "Constructing basic figures (points, lines, circles, polygons, arcs, angles)": [
          "Dot, Line, Arrow, Polygon, Circle, Arc, Sector, Angle (Manim.mobject.geometry)"
        ],
        "Displaying theorem proofs and formula annotations": [
          "MathTex / Tex (Manim.mobject.tex_mobject)"
        ],
        "Demonstrating the drawing of figures and dynamic proofs": [
          "ShowCreation, FadeIn, Transform (Manim.animation)"
        ]
      },
      "Triangles and Quadrilaterals": {
        "Drawing triangles and quadrilaterals": [
          "Polygon (Manim.mobject.geometry)"
        ],
        "Annotating interior/exterior angles, angle bisectors, etc.": [
          "Angle (Manim.mobject.geometry)"
        ],
        "Combining graphics to illustrate theorem proofs": [
          "MathTex / Tex (Manim.mobject.tex_mobject)"
        ]
      },
      "Circles and Related Concepts": {
        "Drawing circles, arcs, sectors and demonstrating tangents, inscribed angles, etc.": [
          "Circle, Arc, Sector, Angle (Manim.mobject.geometry)"
        ],
```

```
        "Displaying circle-related formulas and proof processes": [
          "MathTex / Tex (Manim.mobject.tex_mobject)"
        ]
      },
      "Geometric Transformations": {
        "Demonstrating transformations such as translation, rotation and reflection": [
          "Transform, ApplyMethod, FadeTransform (Manim.animation)"
        ],
        "Manipulating multiple objects simultaneously": [
          "VGroup (Manim.mobject.types.vectorized_mobject)"
        ]
      }
    },
    "Data and Statistics": {
      "Data Collection and Organization": {
        "Constructing coordinate backgrounds for statistical charts": [
          "NumberLine (Manim.mobject.numbers)",
          "Axes, NumberPlane (Manim.mobject.coordinate_systems)"
        ],
        "Creating bar charts and histograms": [
          "Rectangle, VGroup (Manim.mobject.geometry & Manim.mobject.types.vectorized_mobject)"
        ],
        "Annotating statistical data and formulas": [
          "MathTex / Tex (Manim.mobject.tex_mobject)"
        ]
      },
      "Statistical Measures": {
        "Dynamically displaying mean, median and mode": [
          "DecimalNumber, Integer (Manim.mobject.numbers)"
        ],
        "Showing formula derivations and explanations": [
          "MathTex / Tex (Manim.mobject.tex_mobject)"
        ]
      },
      "Basic Probability": {
        "Displaying probability formulas and event representations": [
          "MathTex / Tex (Manim.mobject.tex_mobject)"
        ],
        "Drawing probability distribution graphs": [
          "Axes (Manim.mobject.coordinate_systems)",
          "FunctionGraph (Manim.mobject.graph)"
        ]
      }
    }
  },
  "High School Mathematics Topics": {
    "Numbers and Expressions (Algebra Section)": {
      "Sets and Mappings": {
        "Displaying set symbols, intersections, unions, complements, etc.": [
          "MathTex / Tex (Manim.mobject.tex_mobject)"
        ],
        "Drawing Venn diagrams (if necessary)": [
          "SVGMobject (Manim.mobject.svg)"
        ],
        "Demonstrating dynamic changes in relationships between sets": [
          "FadeIn, Transform (Manim.animation)"
        ]
      },
      "Polynomials and Factorization": {
        "Displaying polynomial expressions and factorization formulas": [
          "MathTex / Tex (Manim.mobject.tex_mobject)"
        ],
        "Demonstrating factorization and simplification processes": [
          "Transform, ReplacementTransform (Manim.animation.transform)"
        ]
      },
      "Equations and Inequalities": {
        "Showing equation solving and inequality proofs": [
          "MathTex / Tex (Manim.mobject.tex_mobject)"
        ],
        "Drawing function graphs to intuitively display roots and regions": [
          "Axes, NumberPlane (Manim.mobject.coordinate_systems)",
          "FunctionGraph (Manim.mobject.graph)"
        ]
      }
    },
    "Functions and Analytic Geometry": {
      "Concepts of Functions and Graphs": {
        "Constructing coordinate backgrounds for function graphs
```

```
    ": [
          "Axes, NumberPlane (Manim
    .mobject.coordinate_systems)"
        ],
        "Drawing standard function
    graphs and parametric curves":
    [
          "FunctionGraph (Manim.
    mobject.graph)",
          "ParametricFunction (
    Manim.mobject.
    parametric_function)"
        ],
        "Annotating function
    expressions and transformation
    processes": [
          "MathTex / Tex (Manim.
    mobject.tex_mobject)"
        ]
      },
      "Composite Functions and
    Inverse Functions": {
        "Demonstrating the
    construction and inversion of
    functions": [
          "Transform (Manim.
    animation)",
          "FunctionGraph",
          "MathTex / Tex"
        ]
      },
      "Sequences": {
        "Displaying sequence terms,
     general terms and summation
    formulas": [
          "NumberLine (Manim.
    mobject.numbers)",
          "DecimalNumber (Manim.
    mobject.numbers)",
          "MathTex / Tex (Manim.
    mobject.tex_mobject)"
        ],
        "Dynamically demonstrating
    recursive processes": [
          "Transform, Write (Manim.
    animation)"
        ]
      },
      "Analytic Geometry": {
        "Drawing Cartesian
    coordinate systems": [
          "Axes, NumberPlane (Manim
    .mobject.coordinate_systems)"
        ],
        "Constructing figures such
    as lines, circles, ellipses and
     parabolas": [
          "Line, Dot, Circle,
    Polygon (Manim.mobject.geometry
    )"
        ],
        "Displaying equations and
    proofs of lines and circles": [
          "MathTex / Tex (Manim.
    mobject.tex_mobject)",
          "FunctionGraph,
    ParametricFunction"
        ]
      }
    },
    "Trigonometric Functions and
    Plane Vectors": {
      "Trigonometric Functions": {
        "Constructing coordinate
    backgrounds for trigonometric
    graphs": [
          "Axes, NumberPlane (Manim
    .mobject.coordinate_systems)"
        ],
        "Drawing graphs of sine,
    cosine and tangent functions":
    [
          "FunctionGraph,
    ParametricFunction (Manim.
    mobject.graph & Manim.mobject.
    parametric_function)"
        ],
        "Annotating angles and
    auxiliary proofs": [
          "Angle (Manim.mobject.
    geometry)",
          "MathTex / Tex (Manim.
    mobject.tex_mobject)"
        ]
      },
      "Plane Vectors": {
        "Drawing vectors and their
    operations": [
          "Vector (Manim.mobject.
    vector_space)"
        ],
        "Displaying vector formulas
     and computation processes": [
          "MathTex / Tex (Manim.
    mobject.tex_mobject)"
        ],
        "Dynamically demonstrating
    vector addition and
    decomposition": [
          "Transform, FadeIn (Manim
    .animation)"
        ]
      }
    },
    "Solid Geometry and Spatial
    Vectors": {
      "Spatial Geometry": {
        "Constructing 3D scenes and
     coordinate backgrounds": [
          "ThreeDScene (Manim.scene
    .three_dimensions)",
          "ThreeDAxes (Manim.
    mobject.three_dimensions)"
        ],
        "Drawing 3D surfaces (such
    as spheres and cones)": [
          "Surface,
    ParametricSurface (Manim.
    mobject.surface)"
        ],
        "Displaying spatial vectors
     and operations": [
          "Vector (suitable for 3D,
     Manim.mobject.vector_space)",
          "MathTex / Tex (Manim.
    mobject.tex_mobject)"
        ]
      }
    },
    "Probability and Statistics": {
```

```
    "Permutations, Combinations and Probability Theory": {
      "Displaying permutation, combination and probability formulas": [
        "MathTex / Tex (Manim.mobject.tex_mobject)"
      ],
      "Grouping objects for display": [
        "VGroup (Manim.mobject.types.vectorized_mobject)"
      ],
      "Drawing probability distribution graphs": [
        "Axes (Manim.mobject.coordinate_systems)",
        "FunctionGraph (Manim.mobject.graph)"
      ]
    },
    "Basics of Statistics": {
      "Displaying data distributions, mean, median, etc.": [
        "NumberLine, DecimalNumber (Manim.mobject.numbers)"
      ],
      "Showing statistical formulas and calculation steps": [
        "MathTex / Tex (Manim.mobject.tex_mobject)"
      ]
    }
  },
  "Calculus": {
    "Limits and Continuity": {
      "Constructing coordinate backgrounds for function graphs and demonstrating limit processes": [
        "Axes, NumberPlane (Manim.mobject.coordinate_systems)",
        "FunctionGraph, ParametricFunction (Manim.mobject.graph & Manim.mobject.parametric_function)"
      ],
      "Displaying limit definitions and expressions": [
        "MathTex / Tex (Manim.mobject.tex_mobject)"
      ]
    },
    "Derivatives and Differentiation": {
      "Displaying function graphs and tangent lines": [
        "Axes, FunctionGraph (Manim.mobject.coordinate_systems & Manim.mobject.graph)",
        "TangentLine (Manim.mobject.geometry, via get_tangent_line method)"
      ],
      "Showing derivative formulas and geometric interpretations": [
        "MathTex / Tex (Manim.mobject.tex_mobject)",
        "Vector, Arrow (Manim.mobject.vector_space & Manim.mobject.geometry)"
      ]
    },
    "Integrals": {
      "Displaying integral regions and function graphs": [
        "Axes, FunctionGraph (Manim.mobject.coordinate_systems & Manim.mobject.graph)"
      ],
      "Dynamically demonstrating definite integral regions": [
        "VMobject (used for area filling, customizable Area class)"
      ],
      "Displaying integral formulas and the fundamental theorem": [
        "MathTex / Tex (Manim.mobject.tex_mobject)"
      ]
    }
  }
}
```

Listing 1: Manim moudle json

A.3 Seed-Guided Multi-Stage Question Generation

The proposed new question generation mechanism builds a multi-stage production architecture designed to systematically evolve seed questions into diverse and quality-assured variants:

- **Target Definition-Agent:** The seed question and high-level generation directives (e.g., desired variation count, general difficulty trajectory, key knowledge points) will be received. While preserving the pedagogical intent of the seed question, specific, actionable generation targets—including the precise number of new questions, explicit difficulty controls, and defined knowledge point constraints—will be established. Its core function is to translate overarching generation goals into quantifiable parameters and constraints that guide the subsequent evolution process.
- **Prompt-Driven Generation-Agent:** It is responsible for receiving the seed question and the established generation targets, and subsequently invoking a LLM using carefully constructed prompts. These prompts are designed

to elicit new question variations that align with the seed question's content and the predefined targets for diversity, difficulty, and knowledge point coverage. The main role of this agent is to leverage the generative power of VLMs to produce a raw corpus of candidate new questions.

- **Initial Filtration-Agent:** The raw set of generated questions will be received, and this corpus will be systematically filtered based on a series of automated checks. These checks include ensuring grammatical correctness, assessing novelty through duplication detection (against the seed question and other generated questions), and verifying adherence to the specified knowledge point tags or constraints. The framework acts as a preliminary quality gate to eliminate syntactically flawed, redundant, or irrelevant question candidates.

- **Quality Assurance-Agent (Optional):** The filtered set of candidate questions will be received, and these questions will undergo a final quality assessment. This can involve an automated evaluation by a dedicated model to assess aspects like clarity, pedagogical soundness, and estimated difficulty, or it can entail preparing the questions for, and incorporating feedback from, a human review process. The evaluation process ensures that the finally selected new questions are of high quality, are unambiguous, and meet the specific pedagogical requirements, thus ensuring the integrity of the generated educational material.

This phrasing attempts to capture the more formal, passive-voice, and function-oriented style of your example.

A.4 Different Scenarios Standards for Mathematical Solution Videos

To ensure clarity, professionalism, and consistency across educational videos for six core types of mathematical problems, this appendix defines a comprehensive quality standard for video production. The criteria are structured around four core dimensions: **visual aesthetics**, **logical coherence**, **accuracy**, and **readability**, aiming to enhance the effectiveness and learning experience of instructional videos.

A.4.1 General Standards (Applicable to All Problem Types)

The following standards apply to all categories of mathematical solution videos to ensure baseline quality and consistency.

Visual Aesthetics

- **Video format:** Aspect ratio should be 16:9 with a minimum resolution of 1080p.

- **Background:** Use a clean blackboard or whiteboard-style background to minimize distractions.

- **Fonts and sizes:** A unified, legible font (e.g., Noto Serif SC, Microsoft YaHei, or handwritten-style fonts) should be used. Mathematical expressions must use font sizes no smaller than 24pt.

- **Layout and highlighting:** Solution steps should be clearly segmented. Key information (e.g., known conditions, target quantities, conclusions) should be highlighted using color where appropriate.

Logical Coherence

- **Reasoning flow:** Solutions should progress logically from known information to the final result.

- **Step transitions:** Each step should include connective language (e.g., "therefore," "it follows that") to clarify logical progression.

- **Diagram-text alignment:** Visual elements (e.g., figures, equations) must appear in sync with verbal or written explanations.

Accuracy

- **Mathematical rigor:** All formulas, symbols, definitions, and terminology must comply with standard curricula and official guidelines.

- **Error-free content:** Videos must be free from grammatical, computational, or graphical errors.

- **Diagram annotations:** Geometric diagrams must be clearly and correctly labeled (e.g., points, lines, angles, units).

Readability

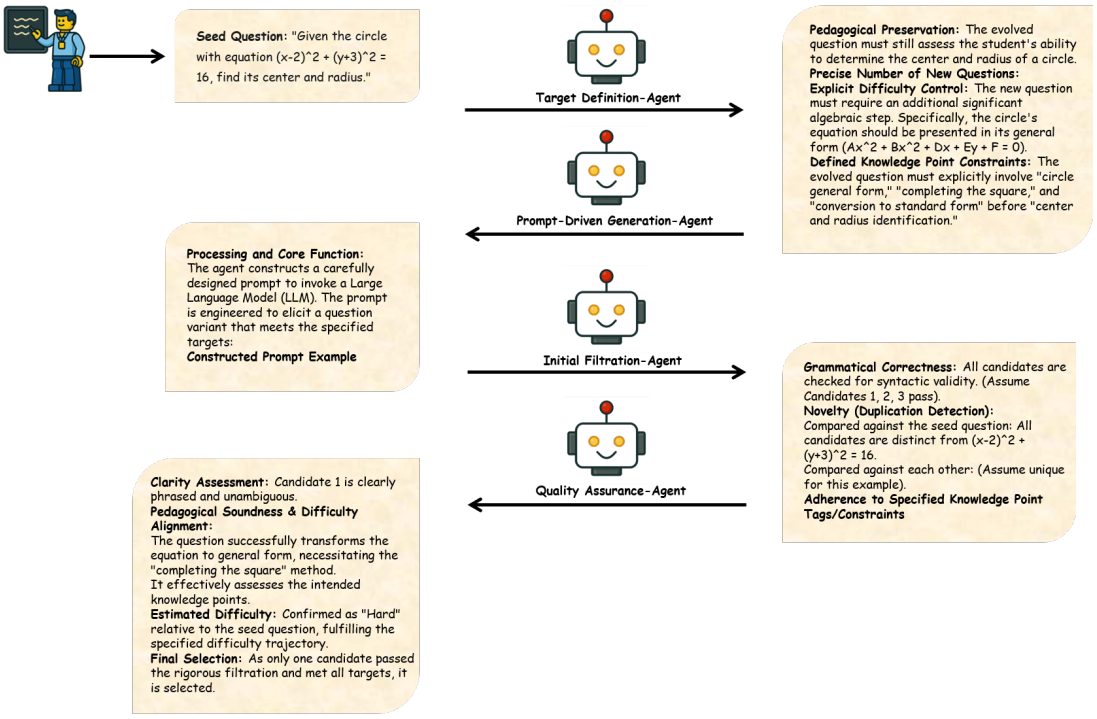

Figure 7: The framework of instructional question diversification from seed templates. The four-agent system evolves a simple circle problem (standard form) into a more complex one (general form) by defining transformation targets, generating candidates with an LLM, and then filtering and validating, while preserving the core geometric properties (center, radius).

- **Visual pacing:** Use clear split-screen layouts or controlled camera movements to guide attention.
- **Narration pace:** Recommended speech rate is 140–160 characters per minute in Mandarin.
- **Content flow:** Handwriting or typing should appear gradually and smoothly to ensure learners can follow along.

A.4.2 Task-Specific Requirements

In addition to the general standards, each problem type requires specific content elements and production considerations.

Function Problems (e.g., monotonicity, extrema, roots) Required Elements:

- Explicit function expressions.
- Domain and range analysis.
- Sketches or graphs illustrating function trends.
- Derivative or inequality-based analysis.
- Highlighted monotonic intervals and extrema.
- Written summary of conclusions (e.g., "the function is increasing on. . . ").

Special Notes: Use animation to illustrate monotonicity changes and formation of extrema where possible.

Analytic Geometry (e.g., lines, circles, ellipses, parabolas) Required Elements:

- Standard equations of geometric objects.
- Accurate Cartesian coordinate systems.
- Labeled key elements: intersections, symmetric points, lines.
- Algebraic calculations (e.g., distance formula, slope analysis).

Special Notes: Ensure diagrams reflect true spatial relations, not merely symbolic illustrations; dual verification via algebraic and geometric reasoning is encouraged.

Plane Geometry (e.g., angles, similarity, congruence, auxiliary lines) Required Elements:

- Complete, annotated geometric diagrams.
- Clear rationale for auxiliary constructions.
- Highlighted use of theorems (e.g., congruence conditions, angle bisector).

- Step-by-step synchronization between diagram updates and explanations.

Special Notes: Use dynamic geometry software (e.g., GeoGebra) to animate constructions and transformations.

Solid Geometry (e.g., surface area, volume, spatial angles) Required Elements:

- 3-view drawings or perspective diagrams of 3D structures.
- Clear labeling of key points, lines, and surfaces.
- Height, projection, and perpendicularity analysis.
- Demonstration of added lines or planes for construction.

Special Notes: Use 3D modeling tools (e.g., SketchUp) with rotation to visualize objects from multiple perspectives. Cross-sections may be needed for angle analysis.

Probability (e.g., classical models, conditional probability, expectation) Required Elements:

- Clear definition of the sample space.
- Proper notation and explanation of events.
- Detailed application of probability formulas.
- Visual aids (e.g., tree diagrams, tables) for complex cases.
- Verbal interpretation of probabilistic concepts.

Special Notes: Use animation to simulate random events (e.g., dice rolls); emphasize logical justification of probability values.

Calculus (e.g., limits, derivatives, integrals) Required Elements:

- Detailed derivation steps for limits, derivatives, and integrals.
- Graph-based analysis (e.g., inflection points, asymptotes).
- Annotated steps for differentiation/integration with brief justifications.
- Interpretation of results in geometric or physical terms (e.g., derivatives as velocity).

Special Notes: Use animation to show tangent slope changes, area accumulation, etc.; reinforce intuitive understanding of core concepts.

A.5 Details of Data Statistics

Table 3 quantifies the task complexity at each difficulty level in the VISUALEDU benchmark.

Difficulty Level	Token			Iterate
	Avg	Med	Range	
Easy	15184	15344	12869 - 19475	0.6
Middle	19173	17986	14169 - 24457	1.5
Hard	24543	24487	18195 - 36472	2.4

Table 3: Details of Difficulty Level.

B Experiments Details

B.1 Calculation Details

The recorded results are summarized in Table **??**. We report several subtask-specific indicators:

- **Preceding Task Execution Rate (PTER)**: Indicates whether the preceding dependent subtask produced any output.
- **Hallucination Rate (HR)**: Measures whether the current subtask output exhibits hallucinated content.
- **Average Human Rating (AHR)**: Human-annotated quality score for the subtask output.
- **Code Execution Rate (CER)**: Denotes whether the generated code is executable.
- **Code Defect Rate (CDR)**: Counts the number of errors due to incorrect parameters or syntax.
- **Visual Suggestion Effectiveness Rate (VSER)**: Reflects whether the VLM provided valid guidance based on video keyframes to improve code.
- **Aesthetic Subcomponent Rate (ASR)**: Boolean indicators evaluating visual aspects of the final video, including garbled characters, element overlap, inconsistent font size, layout misalignment, incomplete display, and logical coherence.

Each metric corresponds to one or more stages in the VisualEDU framework:

- **Solving Approach**, **Tool Localization**, and **Meta-Prompt Match** assess the quality of logical reasoning, visualization tool selection, and prompt design.

- **Code Generation**, **Visual Feedback**, and **Debugging Process** capture performance in code synthesis, refinement via multimodal feedback, and error correction.

We further record four specific execution checks:

- **Initial Code**: Whether the code runs upon first generation.
- **Initial w Debugging**: Whether debugging allows initially failed code to run.
- **Visual Feedback**: Whether visual-guided code is executable.
- **Visual w Debugging**: Whether visually revised code runs after debugging.

Additional statistics include:

- **Parameters Errors** and **Syntax Errors**: Count of errors in code logic and syntax.
- **Code Missing**: Binary flag indicating missing code output.
- **Total Iteration**: Total number of retry cycles for debugging.

During evaluation, we compute six aggregated metrics: **SER**, **IHR**, **UC**, **Visual Accuracy**, **Code Correctness**, and **Aesthetic Score**. Their detailed computation is provided as follows:

Step Execution Rate (SER). We assign weights of 0.1, 0.05, 0.05, 0.3, 0.2, and 0.3 to the six components: Solving Approach, Tool Localization, Meta-Prompt Match, Code Generation, Visual Feedback, and Debugging Process, respectively. The higher weights assigned to Code Generation, Visual Feedback, and Debugging reflect their critical roles in determining downstream quality. Notably, the initial code output from the VLM strongly conditions the effectiveness of subsequent visual rendering and error correction steps. Each component is scored based on human evaluation. We define the initial score $a = 1$ and the decay factor $b = 0.5$, ensuring that $p(r_i) \in [0, 1]$, where the retry count r_i corresponds to the iteration count of the code error rate. All non-debugging components are single-pass and thus assume a retry count of zero. The term Dep_i reflects checkpoint independence. Substituting these into the SER formula yields the final score.

Internal Hallucination Resistance (IHR). Let B_i be binary indicators of hallucination across six stages, with corresponding weights 0.1, 0.1, 0.1, 0.3, 0.2, and 0.2. Let $s_i \in \{0, 1\}$ indicate whether the preceding task failed (1 if failed). These are derived from the **Preceding Task Execution Rate**. We set $\lambda_i = 1$ to zero out the positive score if hallucination occurs. IHR is then computed as per the defined formulation.

Feedback Effectiveness (FE). This score incorporates both visual and code outputs. Visual Suggestion Effectiveness Rate and Code Defect Rate contribute key values. Let B_c denote code missing, with $N_{max} = 5$. Let N represent {Syntax Errors, Incorrect Parameters}, and B denote binary values from the visual suggestion indicators. We set $\alpha = 0.4$, $\beta = 0.6$, with normalized weights $w_v = 0.333$, and $w_c = \{0.4, 0.6\}$. The Feedback Effectiveness is derived accordingly.

Aesthetic Score (AS). We assign weights of 0.1, 0.2, 0.2, 0.2, 0.2, and 0.1 to: Logical Coherence (positive indicator), and the following negative visual factors—Garbled Characters, Elements Overlap, Inconsistent Font, Layout Misalignment, and Incomplete Display. These are aggregated to obtain the final aesthetic score.

B.2 Data details

As referenced in Tables 5 to 13, this section details the review data for the top-performing models evaluated on the VisualEDU benchmark, specifically GPT-4o-11-20, Gemini-2.5-pro, and Claude-3.7-Sonnet. The evaluation is conducted at both macro and micro levels. At the macro level, we assess the Step Execution Rate (SER). SER effectively quantifies the VLM's execution competence within the proposed framework, reflecting its ability to complete each step with minimal retries and strong alignment to expected behavior. At the micro level, the Internal Hallucination Resistance (IHR) evaluate the model's ability to avoid hallucinations. Upstream Completion (UC): examining the model's capability in identifying and processing inter-task dependencies.

Feedback Effectiveness is evaluated based on fine-grained error patterns. It consists of two key components: Code Correctness, which measures the model's ability to handle different categories of code-related errors; and Visual Accuracy, which captures whether the VLM provides effective guidance based on the visual keyframes extracted from

Metric	Qwen2.5-VL-72B	Qwen2.5-VL-7B	Llama-3.2-90B	Llama-3.2-11B	GPT-4o-11-20	GPT-4o-mini	Claude-3.7-Sonnet	Grok2-vision	Gemini-2.5-pro
Preceding Task Execution Rate (PTER)									
Solving Approach	1	1	1	1	1	1	1	1	1
Tool Localization	1	1	1	1	1	1	1	1	1
Meta-prompt Match	1	1	1	1	1	1	1	1	1
Code generation	1	1	1	1	1	1	1	1	1
Visual Feedback	0.6950	0.3250	0.8200	0.5900	0.9100	0.8350	0.8250	0.8400	0.9000
Debugging Process	0.5000	0.3050	0.7200	0.5200	0.9050	0.7900	0.7400	0.7600	0.8350
Hallucination Rate (HR)									
Solving Approach	0.0676	0	0	0	0	0	0	0	0
Tool Localization	0.0476	0.1948	0.3077	0.3723	0.1374	0.1658	0	0	0
Meta-prompt Match	0.0473	0	0.1000	0.1680	0	0	0	0	0
Code Generation	0.3850	0.8100	0.3300	0.5488	0.4650	0.4100	0.3200	0.3950	0.5250
Visual Feedback	0.2754	0.1077	0.2680	0.2797	0.1209	0.1875	0.1667	0.3781	0.1400
Debugging Process	0.1575	0.6684	0.1364	0.4634	0.1950	0.1692	0.0179	0.0061	0.2450
Average Human Rating (AHR)									
Solving Approach	0.8270	0.7960	0.8450	0.7960	0.8885	0.7885	0.8975	0.8885	0.8975
Tool Localization	0.7255	0.5170	0.6935	0.4790	0.6935	0.8485	0.8660	0.7395	0.6975
Meta-prompt Match	0.8645	0.9940	0.7050	0.58775	0.9950	0.8168	0.8480	0.7560	0.7248
Code Generation	0.5005	0.4910	0.5665	0.3825	0.5365	0.4975	0.6529	0.4883	0.5692
Visual Feedback	0.4338	0.2359	0.2464	0.2831	0.5253	0.4256	0.4577	0.2319	0.3879
Debugging Process	0.9231	0.3115	0.8500	0.5855	0.8175	0.9118	0.9313	0.7897	0.7809
Code Execution Rate (CER)									
Initial Code	0.5800	0.3050	0.7750	0.4950	0.6650	0.6900	0.6650	0.6500	0.4450
Initial w Debugging	0.6950	0.3250	0.8200	0.5900	0.9100	0.8434	0.8250	0.8400	0.9000
Visual Feedback	0.4575	0.3000	0.7050	0.4700	0.8850	0.7900	0.7300	0.7550	0.8300
Visual w Debugging	0.5000	0.3050	0.7200	0.5200	0.9050	0.7980	0.7400	0.7600	0.8350
Code Defect Rate (CDR)									
Parameters Errors	0.0759	1.3167	0.0511	0	0.4824	0.1667	0.0150	0	0
Syntax Errors	0.3267	1.6759	0.8429	1.8720	0.4500	0.4057	0.2650	0.8450	1.0302
Code Missing	0.2961	0.1212	0.1711	0.1628	0.0055	0.1092	0.1900	0.2200	0.1809
Total Iteration	0.3265	2.0408	0.8725	1.8842	0.9300	1.4873	0.3800	1.0302	1.1500
Visual Suggestion Effectiveness Rate (VSER)									
Elements Overlap	0.2446	0.0313	0.05455	0.2881	0.6319	0.4847	1	0.4063	0.3613
Inconsistent Font	0.5827	0.1406	0.1576	0.3475	0.8736	0.7055	0.8623	0.8375	0.7906
Layout Misalignment	0.8130	0.5313	0.5394	0.8729	0.8517	0.6196	1	1	0.9162
Aesthetic Subcomponent Rate (ASR)									
Garbled Characters	0.0916	0.0758	0.0457	0.0678	0.2912	0.2254	0.0100	0.2600	0.0750
Elements Overlap	0.8029	0.8769	0.9329	0.7373	0.5000	0.4970	0.5706	0.3681	0.1813
Inconsistent Font	0.8029	0.7846	0.7927	0.6949	0.2637	0.0778	0.0412	0.0123	0.0824
Layout Misalignment	0.4891	0.6000	0.4939	0.6780	0.3736	0.4611	0.3059	0.5092	0.2198
Incomplete Display	0.4234	0.7692	0.6037	0.8729	0.2912	0.3174	0.3118	0.5399	0.2912
Logical Coherence	0.8540	0.5539	0.7195	0.4576	0.8242	0.7246	0.8941	0.7423	0.8901

Table 4: Multi-Model Capability Evaluation Detail

Category	LPR			FE		AS	Overall
	SER	IHR	UC	CC	VA		
Overall	0.6551	0.7980	0.8550	0.7164	0.7967	0.7455	0.6493
Easy	0.6968	0.8284	0.8905	0.7749	0.8657	0.7940	0.7009
Middle	0.6516	0.8149	0.8657	0.7397	0.7960	0.6955	0.6344
Hard	0.6163	0.7500	0.8081	0.6333	0.7273	0.7470	0.6121
Function Problems							
Easy	0.8085	0.8917	0.9444	0.9167	1.0000	0.7167	0.7569
Middle	0.6805	0.8364	0.8788	0.7964	0.9091	0.6727	0.6538
Hard	0.6703	0.8182	0.8182	0.7164	0.8182	0.7636	0.6765
Plane Geometry							
Easy	0.7432	0.8909	0.9394	0.9782	1.0000	0.8727	0.7849
Middle	0.6251	0.6917	0.7222	0.5000	0.5833	0.8000	0.6120
Hard	0.6350	0.7909	0.8788	0.7055	0.7273	0.7091	0.6153
Analytic Geometry							
Easy	0.6847	0.8091	0.8788	0.7164	0.9091	0.8182	0.7849
Middle	0.6925	0.8909	0.9091	0.8073	1.0000	0.6636	0.6120
Hard	0.5925	0.6455	0.6970	0.4545	0.6364	0.6909	0.5383
Solid Geometry							
Easy	0.6104	0.7273	0.7879	0.5127	0.5455	0.7455	0.5761
Middle	0.5325	0.6818	0.7879	0.5927	0.4848	0.6818	0.5292
Hard	0.4947	0.5364	0.6364	0.3527	0.3636	0.7455	0.4795
Probability							
Easy	0.6061	0.7727	0.8788	0.7055	0.8182	0.8182	0.6631
Middle	0.6961	0.8818	0.9394	0.8873	0.8182	0.7091	0.6911
Hard	0.6548	0.8545	0.9091	0.7964	0.9091	0.8273	0.7004
Calculus							
Easy	0.7180	0.8727	0.9091	0.8073	0.9091	0.8000	0.7273
Middle	0.6851	0.9182	0.9697	0.8764	1.0000	0.6364	0.6612
Hard	0.6505	0.8545	0.9091	0.7745	0.9091	0.7455	0.6628
Struct (Meta/Visual Conditions)							
NMNV	0.5551	0.6480	0.7650	0.8164	0.6567	0.6255	0.5493
MNV	0.6151	0.7280	0.8250	0.7764	0.7167	0.7055	0.6093
NMV	0.5851	0.6880	0.7950	0.7964	0.6867	0.6655	0.5793
MV	0.6551	0.7980	0.8550	0.7164	0.7967	0.7455	0.6493

Table 5: Performance of Claude-3-7-sonnet-20250219 based on VisualEDU's various tests and metrics.

Category	LPR			FE		AS	Overall
	SER	IHR	UC	CC	VA		
Overall	0.5422	0.7500	0.9117	0.7364	0.6583	0.8210	0.6207
Easy	0.6055	0.8254	0.9353	0.8179	0.7015	0.7910	0.6602
Middle	0.5669	0.7791	0.9552	0.7982	0.6915	0.8313	0.6430
Hard	0.4528	0.6439	0.8434	0.5909	0.5808	0.8409	0.5581
Function Problems							
Easy	0.6348	0.8500	0.9167	0.8033	0.6667	0.7667	0.6664
Middle	0.5972	0.7636	0.9091	0.6727	0.8485	0.8818	0.6518
Hard	0.4190	0.5727	0.7576	0.4582	0.6667	0.8455	0.5142
Plane Geometry							
Easy	0.5910	0.7909	0.9394	0.8436	0.6667	0.8545	0.6744
Middle	0.6220	0.8667	1.0000	0.9300	0.7222	0.8167	0.6925
Hard	0.5218	0.6818	0.9091	0.7418	0.6364	0.8727	0.6155
Analytic Geometry							
Easy	0.6007	0.7818	0.9394	0.8436	0.6667	0.9091	0.6744
Middle	0.5562	0.7909	1.0000	0.8436	0.8182	0.8091	0.6925
Hard	0.4486	0.6000	1.0000	0.6000	0.6364	0.9455	0.5732
Solid Geometry							
Easy	0.5150	0.7364	0.9697	0.7673	0.5152	0.6727	0.5479
Middle	0.4062	0.6000	0.8788	0.5636	0.2424	0.8455	0.5197
Hard	0.2493	0.4364	0.5455	0.2073	0.1212	0.7727	0.3983
Probability							
Easy	0.6767	0.8273	0.8788	0.8073	0.7879	0.8455	0.7228
Middle	0.6227	0.8364	1.0000	0.9345	0.8788	0.7455	0.6626
Hard	0.5449	0.7727	0.8485	0.6836	0.6970	0.8545	0.6356
Calculus							
Easy	0.6122	0.9636	0.9697	0.8436	0.9091	0.7000	0.6548
Middle	0.5922	0.8091	0.9394	0.8327	0.6364	0.8909	0.6938
Hard	0.5328	0.8000	1.0000	0.8545	0.7273	0.7545	0.6115
Struct (Meta/Visual Conditions)							
NMNV	0.4322	0.5900	0.8117	0.8364	0.5183	0.6910	0.5107
MNV	0.4922	0.6700	0.8717	0.7964	0.5783	0.7710	0.5707
NMV	0.4622	0.6300	0.8417	0.8164	0.5483	0.7310	0.5407
MV	0.5422	0.7500	0.9117	0.7364	0.6583	0.8210	0.6207

Table 6: Performance of Gemini-2.5-pro-exp-03-25 based on VisualEDU's various tests and metrics.

Category	LPR			FE		AS	Overall
	SER	IHR	UC	CC	VA		
Overall	0.6226	0.7685	0.9383	0.8598	0.7150	0.6875	0.6149
Easy	0.7130	0.8567	0.9851	0.9451	0.7861	0.7030	0.6904
Middle	0.5867	0.7269	0.9204	0.8251	0.7065	0.6985	0.5893
Hard	0.5673	0.7212	0.9091	0.8085	0.6515	0.6606	0.5641
Function Problems							
Easy	0.7379	0.8583	1.0000	0.9633	0.7778	0.6333	0.6695
Middle	0.6097	0.7727	1.0000	0.8655	0.8788	0.6545	0.5742
Hard	0.4804	0.5727	0.8788	0.7091	0.6364	0.7273	0.5042
Plane Geometry							
Easy	0.8205	0.9273	1.0000	0.9782	0.9091	0.8727	0.8349
Middle	0.5211	0.6250	0.7778	0.6133	0.5000	0.8417	0.5981
Hard	0.5974	0.7000	0.9394	0.8291	0.6364	0.6364	0.5566
Analytic Geometry							
Easy	0.6714	0.8091	0.9394	0.8655	0.6970	0.8000	0.8349
Middle	0.5491	0.6091	0.8788	0.7964	0.5455	0.6364	0.5981
Hard	0.5026	0.6091	0.7576	0.6364	0.4848	0.6727	0.5162
Solid Geometry							
Easy	0.6262	0.8727	0.9697	0.8982	0.7273	0.4818	0.5632
Middle	0.5579	0.7636	1.0000	0.9455	0.8182	0.5818	0.5406
Hard	0.4979	0.7545	0.8788	0.7636	0.6667	0.6727	0.5506
Probability							
Easy	0.7153	0.7818	1.0000	0.9855	0.7879	0.6909	0.6614
Middle	0.5861	0.6455	0.8788	0.7927	0.7273	0.8091	0.6138
Hard	0.6744	0.9182	1.0000	1.0000	0.6970	0.5727	0.6397
Calculus							
Easy	0.7046	0.8909	1.0000	0.9782	0.8182	0.7455	0.7193
Middle	0.7023	0.9545	1.0000	0.9564	0.7879	0.6545	0.6951
Hard	0.6509	0.7727	1.0000	0.9127	0.7879	0.6818	0.6177
Struct (Meta/Visual Conditions)							
NMNV	0.4926	0.5885	0.8183	0.9598	0.5750	0.5375	0.4849
MNV	0.5526	0.6685	0.8783	0.9198	0.6350	0.6175	0.5449
NMV	0.5226	0.6285	0.8483	0.9398	0.6050	0.5775	0.5149
MV	0.6226	0.7685	0.9383	0.8598	0.7150	0.6875	0.6149

Table 7: Performance of GPT-4o-11-20 based on VisualEDU's various tests and metrics.

Category	LPR			FE		AS	Overall
	SER	IHR	UC	CC	VA		
Overall	0.5772	0.7745	0.8750	0.7294	0.4917	0.6820	0.5953
Easy	0.6052	0.7701	0.8955	0.7654	0.4726	0.7045	0.6123
Middle	0.5855	0.7955	0.8856	0.7445	0.5174	0.6791	0.6075
Hard	0.5404	0.7576	0.8434	0.6776	0.4848	0.6621	0.5655
Function Problems							
Easy	0.6456	0.8667	0.8889	0.7500	0.2222	0.8917	0.7253
Middle	0.6496	0.9091	0.9394	0.9018	0.3333	0.7727	0.7084
Hard	0.6921	1.0000	1.0000	1.0000	0.3636	0.7273	0.7370
Plane Geometry							
Easy	0.6950	0.9364	1.0000	0.9927	0.4242	0.7818	0.7388
Middle	0.6206	0.8833	0.9167	0.8333	0.3889	0.7000	0.6605
Hard	0.5365	0.7636	0.8182	0.7200	0.5152	0.6636	0.5758
Analytic Geometry							
Easy	0.5376	0.7545	0.8182	0.7127	0.5758	0.7364	0.7388
Middle	0.5496	0.7273	0.8182	0.7200	0.5455	0.6545	0.6605
Hard	0.4750	0.7000	0.7879	0.5927	0.4242	0.6455	0.5080
Solid Geometry							
Easy	0.5060	0.7000	0.9091	0.7855	0.2424	0.3909	0.4241
Middle	0.5330	0.7727	0.9394	0.8436	0.6061	0.5091	0.5011
Hard	0.3343	0.5273	0.6364	0.3309	0.3636	0.6000	0.3833
Probability							
Easy	0.7406	0.6364	1.0000	1.0000	0.9394	0.7000	0.6152
Middle	0.5462	0.6273	0.8182	0.7164	0.5758	0.7455	0.5720
Hard	0.5733	0.6545	0.9394	0.8764	0.5455	0.5636	0.5130
Calculus							
Easy	0.5026	0.7182	0.7576	0.3527	0.4545	0.7091	0.5594
Middle	0.6106	0.8455	0.8788	0.4436	0.6667	0.6909	0.6311
Hard	0.6314	0.9000	0.8788	0.5455	0.6970	0.7727	0.6759
Struct (Meta/Visual Conditions)							
NMNV	0.4372	0.5845	0.7450	0.8294	0.3517	0.5220	0.4553
MNV	0.4972	0.6645	0.8050	0.7894	0.4117	0.6020	0.5153
NMV	0.4672	0.6245	0.7750	0.8094	0.3817	0.5620	0.4853
MV	0.5772	0.7745	0.8750	0.7294	0.4917	0.6820	0.5953

Table 8: Performance of GPT-4o-mini based on VisualEDU's various tests and metrics.

Category	LPR			FE		AS	Overall
	SER	IHR	UC	CC	VA		
Overall	0.5197	0.7480	0.8667	0.7076	0.5983	0.6630	0.5518
Easy	0.5448	0.7761	0.8856	0.7809	0.6368	0.6821	0.5842
Middle	0.5415	0.7881	0.8955	0.7158	0.6169	0.6388	0.5634
Hard	0.4722	0.6788	0.8182	0.6248	0.5404	0.6682	0.5070
Function Problems							
Easy	0.5543	0.6833	0.8333	0.7400	0.6667	0.8000	0.6105
Middle	0.5667	0.7000	0.8182	0.4545	0.5152	0.7818	0.5975
Hard	0.5626	0.6091	0.7879	0.6145	0.6061	0.8182	0.5687
Plane Geometry							
Easy	0.5181	0.7273	0.9091	0.7418	0.7879	0.6727	0.5501
Middle	0.5735	0.8750	0.9444	0.8767	0.7500	0.7833	0.6683
Hard	0.4233	0.6818	0.8788	0.6509	0.6061	0.7273	0.5066
Analytic Geometry							
Easy	0.6161	0.8273	0.9394	0.9091	0.7576	0.5818	0.5501
Middle	0.5730	0.8545	0.9697	0.8764	0.8788	0.4182	0.6683
Hard	0.5891	0.8455	0.9091	0.8182	0.8182	0.6818	0.6158
Solid Geometry							
Easy	0.4703	0.7455	0.9091	0.7636	0.3030	0.5818	0.4970
Middle	0.3580	0.5182	0.7576	0.4691	0.2424	0.5273	0.3617
Hard	0.3844	0.6091	0.7576	0.5600	0.1818	0.6000	0.4310
Probability							
Easy	0.6669	0.9727	1.0000	0.9891	0.7576	0.7273	0.7164
Middle	0.5731	0.8909	0.9697	0.7855	0.6667	0.7091	0.6283
Hard	0.4016	0.5909	0.7879	0.4691	0.5455	0.6000	0.4184
Calculus							
Easy	0.4425	0.7091	0.7273	0.5455	0.5455	0.7182	0.5386
Middle	0.6017	0.8818	0.9091	0.8182	0.6364	0.6000	0.6038
Hard	0.4718	0.7364	0.7879	0.6364	0.4848	0.5818	0.5014
Struct (Meta/Visual Conditions)							
NMNV	0.3897	0.5680	0.7467	0.8076	0.4583	0.5130	0.4218
MNV	0.4497	0.6480	0.8067	0.7676	0.5183	0.5930	0.4818
NMV	0.4197	0.6080	0.7767	0.7876	0.4883	0.5530	0.4518
MV	0.5197	0.7480	0.8667	0.7076	0.5983	0.6630	0.5518

Table 9: Performance of Grok2-vision based on VisualEDU's various tests and metrics.

Category	LPR			FE		AS	Overall
	SER	IHR	UC	CC	VA		
Overall	0.5817	0.6820	0.7317	0.4448	0.3800	0.6205	0.5120
Easy	0.6290	0.7209	0.7811	0.5075	0.4577	0.6284	0.5496
Middle	0.5825	0.6881	0.7313	0.4507	0.4080	0.6224	0.5126
Hard	0.5330	0.6364	0.6818	0.3752	0.2727	0.6106	0.4734
Function Problems							
Easy	0.6970	0.7667	0.8611	0.4933	0.6667	0.6500	0.6007
Middle	0.6140	0.6818	0.7273	0.5455	0.3333	0.6818	0.5632
Hard	0.6352	0.7636	0.8182	0.5455	0.4242	0.5182	0.5194
Plane Geometry							
Easy	0.5677	0.6364	0.6667	0.3636	0.3636	0.5727	0.4574
Middle	0.5311	0.5167	0.6111	0.1567	0.4722	0.4750	0.3504
Hard	0.4952	0.4182	0.4545	0.0909	0.1818	0.7000	0.4044
Analytic Geometry							
Easy	0.6889	0.8000	0.8182	0.5455	0.6061	0.6636	0.4574
Middle	0.5902	0.8273	0.7879	0.4436	0.4242	0.6091	0.3504
Hard	0.5276	0.6818	0.7273	0.2727	0.3333	0.5455	0.4636
Solid Geometry							
Easy	0.5149	0.5000	0.6667	0.3891	0.2424	0.5273	0.3872
Middle	0.4820	0.4818	0.6061	0.2327	0.3030	0.5909	0.3822
Hard	0.4313	0.4545	0.5152	0.1709	0.1515	0.6636	0.3897
Probability							
Easy	0.6643	0.8455	0.8485	0.6182	0.4545	0.6818	0.6358
Middle	0.6047	0.7273	0.7576	0.5345	0.3939	0.7273	0.5773
Hard	0.5459	0.7455	0.8182	0.6255	0.2727	0.6000	0.5278
Calculus							
Easy	0.6348	0.7727	0.8182	0.6364	0.3939	0.6727	0.6018
Middle	0.6776	0.9091	0.9091	0.8182	0.5152	0.6636	0.6623
Hard	0.5627	0.7545	0.7576	0.5455	0.2727	0.6364	0.5354
Struct (Meta/Visual Conditions)							
NMNV	0.4417	0.4920	0.6017	0.5448	0.2400	0.4605	0.3720
MNV	0.5017	0.5720	0.6617	0.5048	0.3000	0.5405	0.4320
NMV	0.4717	0.5320	0.6317	0.5248	0.2700	0.5005	0.4020
MV	0.5817	0.6820	0.7317	0.4448	0.3800	0.6205	0.5120

Table 10: Performance of Qwen2.5-VL-72B-Instruct based on VisualEDU's various tests and metrics.

Category	LPR			FE		AS	Overall
	SER	IHR	UC	CC	VA		
Overall	0.3004	0.4665	0.5433	0.2764	0.0750	0.6620	0.3903
Easy	0.3638	0.5433	0.6070	0.3379	0.1144	0.6269	0.4194
Middle	0.2902	0.4463	0.5323	0.2931	0.0796	0.6746	0.3859
Hard	0.2465	0.4091	0.4899	0.1970	0.0303	0.6848	0.3654
Function Problems							
Easy	0.4208	0.6500	0.6667	0.3333	0.1944	0.6333	0.4721
Middle	0.2984	0.4818	0.5152	0.2618	0.1212	0.7636	0.4277
Hard	0.2806	0.4455	0.5152	0.1818	0.0606	0.6818	0.3908
Plane Geometry							
Easy	0.3388	0.5273	0.5455	0.2727	0.1212	0.6909	0.4340
Middle	0.3316	0.4833	0.6111	0.3967	0.1111	0.6083	0.3862
Hard	0.2689	0.4182	0.5152	0.2727	0.0303	0.6909	0.3759
Analytic Geometry							
Easy	0.4220	0.6091	0.6667	0.4545	0.1212	0.5273	0.4340
Middle	0.2477	0.3636	0.4545	0.1818	0.0303	0.7091	0.3862
Hard	0.2341	0.4182	0.4242	0.0909	0.0000	0.7273	0.3713
Solid Geometry							
Easy	0.3587	0.4364	0.6667	0.4218	0.0606	0.4182	0.3001
Middle	0.2938	0.4455	0.5758	0.3636	0.1212	0.5455	0.3386
Hard	0.2874	0.4091	0.5758	0.2727	0.0606	0.5455	0.3242
Probability							
Easy	0.2620	0.4091	0.4545	0.1818	0.0303	0.8182	0.4212
Middle	0.3189	0.5000	0.5758	0.3636	0.0303	0.6727	0.4148
Hard	0.1318	0.3000	0.3333	0.0000	0.0000	0.8000	0.3398
Calculus							
Easy	0.3753	0.6182	0.6364	0.3636	0.1515	0.6727	0.4659
Middle	0.2473	0.4000	0.4545	0.1818	0.0606	0.7545	0.3903
Hard	0.2762	0.4636	0.5758	0.3636	0.0303	0.6636	0.3902
Struct (Meta/Visual Conditions)							
NMNV	0.1604	0.2765	0.4133	0.3764	0.0260	0.5020	0.2503
MNV	0.2204	0.3565	0.4733	0.3364	0.0450	0.5820	0.3103
NMV	0.1904	0.3165	0.4433	0.3564	0.0650	0.5420	0.2803
MV	0.3004	0.4665	0.5433	0.2764	0.0750	0.6620	0.3903

Table 11: Performance of Qwen2.5-VL-7B-Instruct based on VisualEDU's various tests and metrics.

Category	LPR			FE		AS	Overall
	SER	IHR	UC	CC	VA		
Overall	0.5461	0.7390	0.8467	0.6664	0.2067	0.5160	0.4980
Easy	0.5788	0.7657	0.8706	0.7063	0.2289	0.5522	0.5384
Middle	0.5414	0.7463	0.8408	0.6615	0.1891	0.5358	0.5054
Hard	0.5178	0.7045	0.8283	0.6309	0.2020	0.4591	0.4495
Function Problems							
Easy	0.6210	0.8083	0.9167	0.5000	0.0833	0.5500	0.5528
Middle	0.6608	0.9273	1.0000	0.8073	0.2121	0.5727	0.6300
Hard	0.6598	0.7364	0.9091	0.7273	0.2121	0.5909	0.5525
Plane Geometry							
Easy	0.5544	0.7455	0.8788	0.8182	0.1818	0.5727	0.5410
Middle	0.3838	0.5583	0.6389	0.4167	0.1111	0.6750	0.4506
Hard	0.3876	0.5000	0.5758	0.3636	0.2121	0.6818	0.4429
Analytic Geometry							
Easy	0.7156	0.9364	0.9697	0.9091	0.5152	0.5727	0.5410
Middle	0.6477	0.8273	0.8788	0.6364	0.2727	0.5182	0.4506
Hard	0.5630	0.7636	0.8788	0.5127	0.2121	0.3909	0.4410
Solid Geometry							
Easy	0.4577	0.6091	0.7879	0.5818	0.2424	0.3182	0.3418
Middle	0.3822	0.6545	0.7576	0.5855	0.1818	0.3818	0.3602
Hard	0.4020	0.6636	0.7879	0.5782	0.2121	0.2909	0.3286
Probability							
Easy	0.6203	0.7818	0.9091	0.8109	0.1515	0.6000	0.5701
Middle	0.5688	0.7091	0.9091	0.8182	0.1515	0.5091	0.4978
Hard	0.5597	0.7909	0.9697	0.8873	0.1515	0.3545	0.4497
Calculus							
Easy	0.5003	0.7091	0.7576	0.6364	0.2121	0.7000	0.5678
Middle	0.6194	0.8182	0.8788	0.7273	0.2121	0.5455	0.5585
Hard	0.5344	0.7727	0.8485	0.7164	0.2121	0.4455	0.4821
Struct (Meta/Visual Conditions)							
NMNV	0.4061	0.5490	0.7167	0.7664	0.0667	0.3560	0.3580
MNV	0.4661	0.6290	0.7767	0.7264	0.1267	0.4360	0.4180
NMV	0.4361	0.5890	0.7467	0.7464	0.0967	0.3960	0.3880
MV	0.5461	0.7390	0.8467	0.6664	0.2067	0.5160	0.4980

Table 12: Performance of Llama-3.2-90B-Vision-Instruct based on VisualEDU's various tests and metrics.

Category	LPR			FE		AS	Overall
	SER	IHR	UC	CC	VA		
Overall	0.3720	0.5990	0.7033	0.4882	0.2967	0.5390	0.3951
Easy	0.4406	0.6716	0.7811	0.5857	0.4080	0.5090	0.4246
Middle	0.3040	0.5522	0.6119	0.3516	0.2338	0.5881	0.3689
Hard	0.3714	0.5727	0.7172	0.5279	0.2475	0.5197	0.3917
Function Problems							
Easy	0.5875	0.7667	0.9167	0.7300	0.5000	0.5333	0.5248
Middle	0.4887	0.6364	0.7879	0.5927	0.5152	0.6091	0.4836
Hard	0.2995	0.3273	0.5758	0.3636	0.1212	0.6545	0.3662
Plane Geometry							
Easy	0.3754	0.6909	0.6970	0.4691	0.4848	0.6455	0.4558
Middle	0.2079	0.4750	0.4167	0.0833	0.0833	0.7417	0.3698
Hard	0.4168	0.7182	0.7879	0.5273	0.2727	0.4273	0.3925
Analytic Geometry							
Easy	0.3423	0.6364	0.6364	0.3309	0.2121	0.6182	0.4558
Middle	0.2464	0.5364	0.5455	0.2618	0.1212	0.6091	0.3698
Hard	0.3808	0.6273	0.6970	0.5455	0.1818	0.6091	0.4494
Solid Geometry							
Easy	0.4458	0.6273	0.8182	0.6836	0.6970	0.3273	0.3431
Middle	0.3164	0.5182	0.6667	0.4218	0.4545	0.4182	0.2962
Hard	0.3127	0.4000	0.6364	0.4545	0.4545	0.4818	0.3090
Probability							
Easy	0.4184	0.6273	0.8485	0.6509	0.2424	0.2818	0.3069
Middle	0.3050	0.6091	0.7273	0.5018	0.1818	0.5000	0.3505
Hard	0.3671	0.6455	0.7576	0.6145	0.2727	0.4364	0.3839
Calculus							
Easy	0.4611	0.6727	0.7576	0.6364	0.3030	0.6455	0.5024
Middle	0.2682	0.5455	0.5455	0.2727	0.0606	0.6364	0.3689
Hard	0.4517	0.7182	0.8485	0.6618	0.1818	0.5091	0.4495
Struct (Meta/Visual Conditions)							
NMNV	0.2320	0.4090	0.5733	0.5882	0.1567	0.3790	0.2551
MNV	0.2920	0.4890	0.6333	0.5482	0.2167	0.4590	0.3151
NMV	0.2620	0.4490	0.6033	0.4882	0.2967	0.4190	0.2851
MV	0.3720	0.5990	0.7033	0.4882	0.2967	0.5390	0.3951

Table 13: Performance of Llama-3.2-11B-Vision-Instruct based on VisualEDU's various tests and metrics.

the video. Together, these indicators reflect the system's robustness and reliability in multimodal reasoning tasks.

In essence, this section outlines a comprehensive evaluation framework for assessing the planning capabilities of VLMs, distinguishing between macro-level logical correctness and micro-level execution fidelity through a series of specific quantitative metrics.

B.3 Version for Tested VLMs

To ensure the reproducibility of our results, we report the exact versions of all VLMs evaluated in our experiments.

- GPT-4o: GPT-4o-2024-11-20
- GPT-4o-mini: GPT-4o-mini
- Claude-3.7: Claude-3-7-sonnet-20250219
- Gork-2: grok-2-vision-1212
- Gemini-2.5: Gemini-2.5-pro-exp-03-25
- Qwen-2.5: Qwen2.5-VL-72B-Instruct
- Qwen-2.5: Qwen2.5-VL-7B-Instruct
- Llama-3.2: meta/Llama-3.2-90B-Vision-Instruct
- Llama-3.2: meta/Llama3.2-11b-vision-instruct

B.4 Agent Prompt

Generate solution from problem

Provide a detailed, step−by−step solution process for the following math problem in English.
Example:
Math Problem: Find the vertex of the function $f(x) = x^2 - 4x + 3$ and draw its graph.
Solution Process:
1.Derive the Function: First, we compute the derivative of $f(x)$, which is $f'(x) = 2x - 4$.
2.Find the Zero of the Derivative: Set $f'(x) = 0$ to solve for x, yielding $x = 2$. This indicates that the function has an extremum at $x = 2$.
3.Compute the y-Coordinate of the Vertex: Substitute $x = 2$ into the original function to get $f(2) = 2^2 - 4 * 2 + 3 = -1$. Thus, the vertex is at $(2, -1)$.
4.Determine the Type of the Vertex: Since the coefficient of the quadratic term is positive, the vertex represents the minimum point of the function, i.e., the lowest point on the graph.
Math Problem: {math_problem}
Solution Process:

Figure 8: System prompt for generating solution from problem.

Generate Manim moudle from problem

```
Input:
Problem: {math_problem}
Solution Process: {math_solution}
I need to convert the above solution process into a visualized Manim video. Please identify the required internal Manim modules from the JSON formatted file below, and output them.
JSON content: {Manim_moudle}
Output:
Required internal Manim modules:
```

Figure 9: System prompt for generate Manim moudle from problem.

Generate Similarity from problem

```
Example:
Input Question: Question.
Instruction: Please select the problem with the most similar key solution steps and approach to the given problem, and estimate the degree of similarity by outputting a numerical result directly.
Comparison Problems: [Question1, Question2, ... QuestionN]
Output (in the following format): 1, 50%.
Application:
Input Question: {math_problem}
Instruction: Please select the problem with the most similar key solution steps and approach to the given problem, and estimate the degree of similarity by outputting a numerical result directly.
Comparison Problems: '{compare}'
(Only output the number and percentage, without any textual explanation.)
Output (if none, output 0, 0%):
```

Figure 10: System prompt for generate Similarity from problem.

Generate Manim code from solution meta

```
For generating Manim code, please refer to the following code format. Combine it with the selected internal Manim modules to create a Manim visualization video for the above solution process.
Example Process:
Example Input Problem: {similar_Q}
Example Input Solution Process: {similar_A}
Example Output Manim Code: {similar_C}
Application:
Math Problem: {problem}
Solution Process: {solution}
Selected Manim Internal Modules: {Manim_moudle}
Finally, output only the Manim code (only output the code portion):
```

Figure 11: System prompt for generate Manim code from solution meta.

Correct error code

```
Modify the original Manim code based on the error feedback and output the corrected version.
Input Original Code: {math_code}.
Error Message: {err}
Finally, output only the modified complete Manim code (only output the code portion):
```

Figure 12: System prompt for correct error code.

Visualize suggestion

```
{[
{"type": "text", "text": Determine whether there is overlapping content in the image},
{"type": "image_url", "image_url": {"url": image_url}}
],
[
{"type": "text", "text": Please check it carefully once again and reflect.}
]},
{[
{"type": "text", "text": Determine whether the font sizes in the image are uniform},
{"type": "image_url", "image_url": {"url": image_url}}
],
[
{"type": "text", "text": Please check it carefully once again and reflect.}
]},
{[
{"type": "text", "text": Determine whether the overall layout satisfies that the text only appears on the left half of the screen, and the analysis image only appears on the right half of the screen. If not, please suggest moving the location and scaling the size},
{"type": "image_url", "image_url": {"url": image_url}}
],
[
{"type": "text", "text": Please check it carefully once again and reflect.}
]}
```

Figure 13: System prompt for visualize suggestion.

visualize feedback

```
Modify the original Manim code based on the visual input suggestions and output the updated version.
Visual Input Suggestion: {suggestion}
Original Code: {code}
Adjust the text length in Text() and MathTex() to improve text overlap, and adjust the scale(), use to_corner() parameters to improve image layout and font size
Finally, output only the complete modified Manim code (only output the code portion):
```

Figure 14: System prompt for visualize feedback.