PreGenie: An Agentic Framework for High-quality Visual Presentation Generation

Xiaojie Xu[1], Xinli Xu[1], Sirui Chen[1],
Haoyu Chen[1], Fan Zhang[3], Ying-Cong Chen[1,2],

[1]The Hong Kong University of Science and Technology(Guangzhou),
[2]The Hong Kong University of Science and Technology, [3]Shanghai AI Laboratory

Abstract

Visual presentations are vital for effective communication. Early attempts to automate their creation using deep learning often faced issues such as poorly organized layouts, inaccurate text summarization, and a lack of image understanding, leading to mismatched visuals and text. These limitations restrict their application in formal contexts like business and scientific research. To address these challenges, we propose PreGenie, an agentic and modular framework powered by multimodal large language models (MLLMs) for generating high-quality visual presentations.

PreGenie is built on the Slidev presentation framework, where slides are rendered from Markdown code. It operates in two stages: (1) Analysis and Initial Generation, which summarizes multimodal input and generates initial code, and (2) Review and Re-generation, which iteratively reviews intermediate code and rendered slides to produce final, high-quality presentations. Each stage leverages multiple MLLMs that collaborate and share information. Comprehensive experiments demonstrate that PreGenie excels in multimodal understanding, outperforming existing models in both aesthetics and content consistency, while aligning more closely with human design preferences.

1 Introduction

Visual presentations play an important role in visual communication. They are frequently used in speeches, reports, and various forms of concept expression, helping to enhance audience understanding. Early visual presentations relied on manual creation, which was time-consuming and repetitive. In recent years, with the rapid development of generative models and MLLMs (Alayrac et al., 2022; Liu et al., 2023; Team et al., 2023), many efforts have been dedicated to automating the generation of visual presentations. However, the task of automatically generating visual presentations faces several challenges: (1) **Aesthetic and well-organized layouts**: From a design perspective, the generated slide content must be well-arranged and meet human aesthetic standards. (2) **Support for complex multimodal inputs and outputs**: The model needs to understand complex text-image inputs (e.g., a research paper or blog) and generate slides containing both text and images. (3) **Accuracy and relevance between text and images**: The generated text must closely align with the main ideas of the original content, and the images on each slide must be highly relevant to the accompanying text to aid audience comprehension.

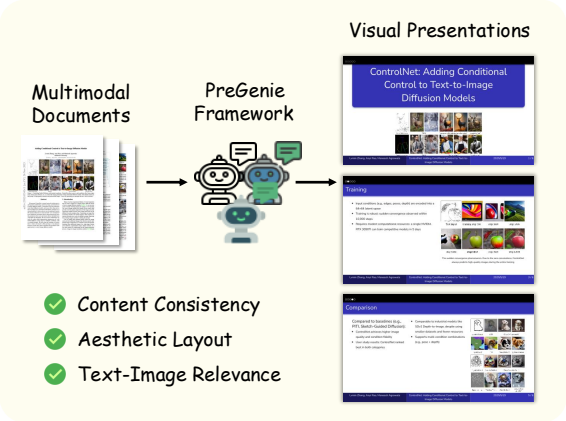

Figure 1: The PreGenie framework, powered by MLLMs, processes text-image inputs to generate high-quality visual presentations. Source: (Zhang et al., 2023)

Existing work on visual presentation generation can be divided into two categories: (1) Direct generation of text-image slides (Ma et al., 2025; Chen et al., 2025). These methods, often using generative models such as diffusion models, embed text directly into images. While the layouts generated by these methods are often visually appealing, they cannot handle complex text-image inputs. Additionally, the generated results are not editable, and content accuracy cannot be guaranteed. As a result,

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 3045–3063
November 4-9, 2025

these methods are unsuitable for fields with high requirements for professionalism and precision, such as scientific research. (2) Generation of intermediate code followed by rendering (Zheng et al., 2025; Ge et al., 2025; Cachola et al., 2024). Currently, this is the more widely adopted approach. These methods first generate intermediate code, which is then rendered into slides using predefined templates or rules. However, there is often a gap between the intermediate code and the final visual results, making it difficult to ensure the slides are harmonious and visually appealing. Moreover, while these methods generally handle text well, they struggle to understand images in the input document, and most of them only generate plain-text slides without images (Cachola et al., 2024; Bandyopadhyay et al., 2024).

This raises the question: **Can we develop a framework that supports multimodal inputs and generates visual presentations that are aesthetically pleasing with accurate and consistent content?**

In this work, we propose PreGenie, an agentic framework based on MLLMs that supports text-image document inputs and generates visually appealing, accurate, and coherent visual presentations. It leverages Slidev[1], a Markdown-based, flexible, and concise presentation framework. Compared to prior works that use intermediate code representations (Zheng et al., 2025; Ge et al., 2025) (e.g., HTML or Python's pptx library), Slidev provides a simpler structure, reducing the difficulty for LLMs to generate correct code and making subsequent editing easier. Based on the Slidev framework, our workflow consists of two stages: (1) **Analysis and Initial Generation**: This stage involves syntax analysis, text summarization, and image tagging and positioning to generate visual presentations that closely align with the input document. (2) **Review and Re-generation**: This includes code review and page review. Code review uses Large Language Models(LLMs) to check for formatting and content errors in the intermediate code. Building on this, page review uses Vision-language Models (VLMs) to examine the final visual results page by page, identifying errors that may be missed in the code (e.g., an image partially overflowing the slide). After these checks, problematic slides are regenerated.

While previous works on code generation have widely adopted code review mechanisms (Khan et al., 2024; Wang et al., 2024), the lack of a visual inspection mechanism has hindered the quality of the final slide outputs. By introducing a page review mechanism, we significantly improve the alignment and aesthetics of the layouts, effectively closing the gap between intermediate code and the final visual results. Extensive experiments and evaluations demonstrate that our method excels in design aesthetics, text coverage, and text-image consistency.

Overall, our contributions are as follows:

- We propose PreGenie, an agentic framework based on MLLMs, which supports multimodal document inputs and generates well-organized and accurate visual presentations.
- We introduce both intermediate code review and visual page review mechanisms into the presentation generation process, ensuring more reliable and visually appealing results.
- Our proposed framework supports a wide range of practical applications, providing a practical solution for the field of visual presentation generation.

2 Related Works

2.1 LLM-powered Autonomous Agents

LLMs (Radford et al., 2018, 2019; Devlin et al., 2019) have shown strong capabilities in language understanding and generation. With the introduction of multimodal LLMs (MLLMs) (Alayrac et al., 2022; Li et al., 2022; Liu et al., 2023; Team et al., 2023), these models can now process images, audio, and other inputs, enabling more comprehensive reasoning and content generation.

With decision-making and tool-use abilities, LLMs act as autonomous agents for perception, reasoning, and goal-directed actions. Early frameworks like ReAct (Yao et al., 2023) and Toolformer (Schick et al., 2023) combine reasoning with tools. Multi-agent systems, such as CAMEL (Li et al., 2023) and ChatDev (Qian et al., 2023), enable communication and role specialization, while AutoGPT (Yang et al., 2023) and MetaGPT (Hong et al., 2023) focus on task decomposition and planning. Inspired by these advances, our method adopts a multi-agent system that integrates multimodal understanding, task decomposition, and iterative review and feedback for structured presentation generation.

[1]https://sli.dev/

2.2 Agents for Content Generation

Agent-based frameworks have become increasingly prominent in multimodal content generation. In visual domains, methods such as MM-StoryAgent (Xu et al., 2025), MovieAgent (Wu et al., 2025), and Video-LLM (Huang et al., 2024) decompose video generation into subtasks such as scene planning, narration, and rendering. LayoutDM (Inoue et al., 2023) and RenderAgent (Gonzalez-Morcillo et al., 2007) further emphasize structure-aware layout synthesis. For more structured outputs, POSTA (Chen et al., 2025), PosterLLaVa (Yang et al., 2024b), and VASCAR (Zhang et al., 2024b) generate layout-aware posters by combining saliency modeling, image tagging, and template-based rendering. Despite recent progress, existing agents still struggle with generating highly structured multimodal content.

2.3 Presentation Generation

Early approaches to presentation generation relied on extractive summarization, rule-based templates, or layout heuristics (Hu and Wan, 2014; Xu and Wan, 2022; Sun et al., 2021; Fu et al., 2022), which often lacked flexibility and multimodal support.

Recent methods powered by LLMs fall into two categories. The first directly synthesizes slide images using diffusion or image-conditioned models (Ma et al., 2025; Chen et al., 2025), producing visually rich outputs but offering limited structural control and editability. The second generates intermediate representations, such as Markdown or HTML, which are then rendered into slides (Zheng et al., 2025; Ge et al., 2025; Cachola et al., 2024; Yang et al., 2024b; Bandyopadhyay et al., 2024). This approach improves layout controllability and enables post-editing, but often lacks mechanisms to verify the rendered visual output.

Representative works like AutoPresent (Ge et al., 2025) emphasize layout-aware code synthesis, while PPTAgent (Zheng et al., 2025) simulates human editing workflows by iteratively modifying templates based on LLM feedback. While this staged refinement improves structure, it lacks visual validation and often misses issues like overflow, misalignment, or missing images. In contrast, PreGenie combines structured code generation with visual-level inspection, enabling reliable and semantically aligned presentation outputs.

3 Method

Inspired by human workflows, we decompose the task of presentation generation into multi-stages and a series of fine-grained steps. Our approach integrates five LLMs and VLMs that share a common context. As shown in Fig. 2, the framework starts with an input consisting of a webpage or a text-image document and processes it sequentially through the Text Summarizer, Image Captioner, Code Generator, Code Reviewer, and Page Reviewer, ultimately producing high-quality text-image presentations. Among these components, the Text Summarizer, Code Generator, and Code Reviewer are powered by LLMs, while the Image Captioner and Page Reviewer utilize VLMs.

The entire process is divided into two stages. The first stage, **Analysis and Initial Generation** (Fig. 2 top), focuses on performing a fundamental analysis of the input multimodal information and generating the initial version of the code for subsequent processing. The second stage, **Iterative Review and Re-generation** (Fig. 2 bottom), takes the output from the first stage and refines it through iterative checks of the code and visual elements of the rendered pages. This process ultimately results in refined and visually appealing slides.

3.1 Preliminary: Slidev Framework

We utilize Slidev, a simple and flexible slide generation framework, which offers several advantages compared to other commonly used intermediate code forms like HTML and the Python pptx library (Zheng et al., 2025; Ge et al., 2025). First, Slidev is Markdown-based, making it simple to use and easy for subsequent modifications by LLMs or users. Second, it is visually professional and aesthetically pleasing. Lastly, it supports multiple themes to fit different scenarios, such as business and academia, further enhancing its practicality. Fig. 3 shows an example Slidev code with the rendered slide page.

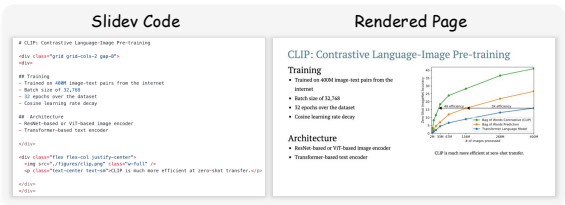

Figure 3: An example of Slidev Markdown code (left) and its rendered page (right). The code is well-structured, and easy to edit. Source: (Radford et al., 2021)

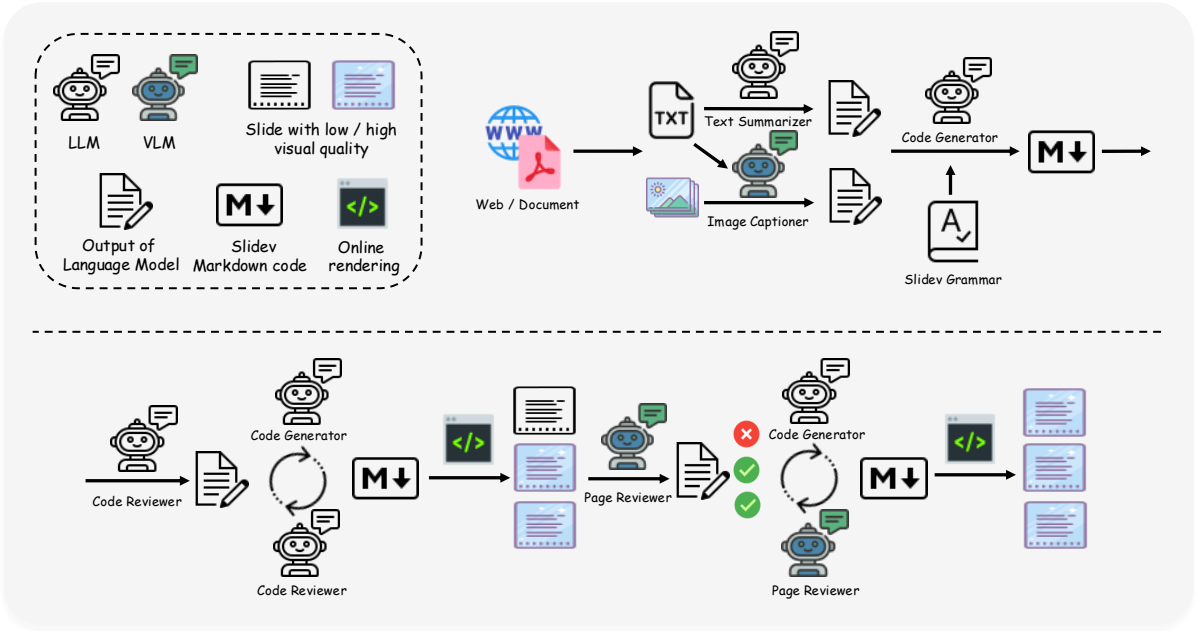

Figure 2: Our PreGenie framework is divided into two stages. The first stage (top) performs foundational analysis of the input multimodal information and generates the initial code. The second stage (bottom) iteratively reviews the code and the visual elements of the rendered slides, ultimately producing refined and aesthetically pleasing slides.

3.2 Analysis and Initial Generation

At this stage, we conduct input analysis and generate an initial version of the presentation. First, as shown in Fig. 2 (top), the input text is processed by a Text Summarizer, which analyzes the content to extract key information, including the article summary, title, authors, affiliations, and other essential details. Next, the input images, along with the input text, are passed to an Image Captioner. The Image Captioner assigns labels to the images, including titles, detailed descriptions, and their locations within the original text. As a result, we obtain two markdown files containing descriptions of the text and images.

Subsequently, we utilize a Code Generator to produce the code for the initial version of the presentation. Since we employ the Slidev framework, it is necessary to first understand its syntax. To facilitate this, we provide the Code Generator with a file containing the complete syntax and usage examples of Slidev, along with the previously generated text and image description files. Additionally, we specify requirements related to layout and aesthetics (e.g., limits on the number of lines per slide and the proportion of the slide occupied by images). Based on this input, the Code Generator produces an initial version of the Slidev code.

3.3 Iterative Review and Re-generation

The initial code may contain syntax errors, produce content that is inconsistent with the previous summaries, or fail to adhere to certain user-provided intentions, making it unable to function as expected. To address this, the output from the Code Generator, along with the relevant context, is passed to the Code Reviewer for auditing. As shown in Fig. 2 (bottom), the Code Reviewer examines the code, identifies issues, and provides feedback to the Code Generator, which then regenerates the code. This process is iterated until the system fully understands the user's intent and produces functional, error-free code.

However, some issues cannot be easily identified through code alone. For instance, images on slides might partially overlap the page boundaries, or the overall text-image layout might appear too crowded. While these problems are easily noticeable in the final slides, they may not be apparent in the raw code. To address this, the code that has been reviewed in the Code Reviewer loop is rendered into slide pages. Subsequently, the Page Reviewer inspects each slide to ensure that it adheres to layout rules and meets aesthetic standards. Feedback for problematic pages is sent back to the Code Generator for regeneration. This process also iterates until all issues are resolved.

Since only a small portion of problematic pages

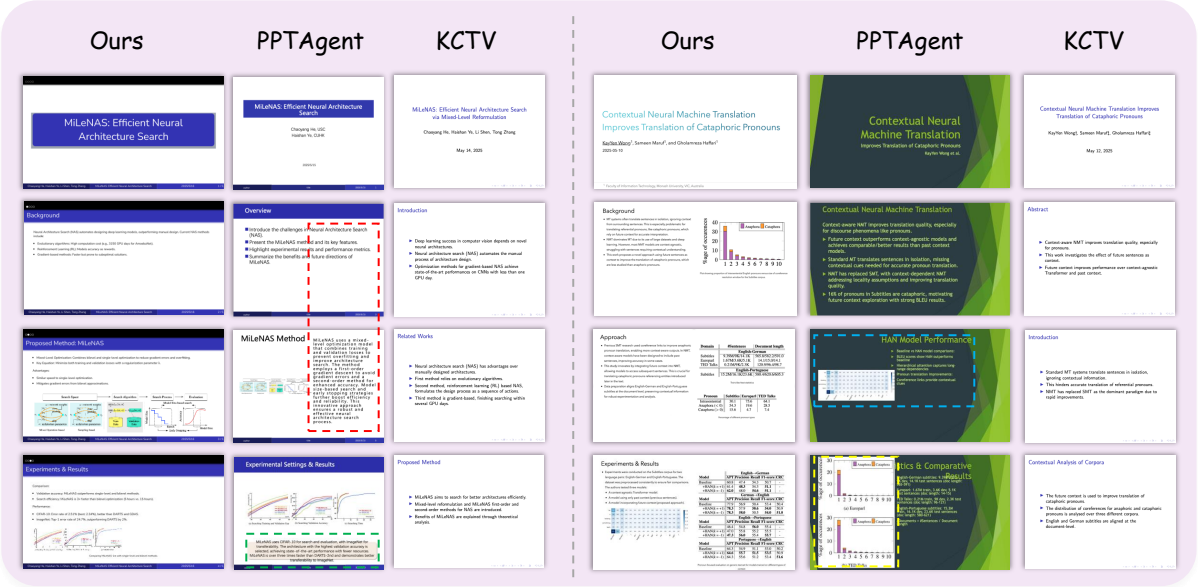

Figure 4: Qualitative comparisons among Ours, PPTAgent, and KCTV. Left: Slides with a similar theme. Right: Slides with a random theme. Representative pages are selected for comparison. Design and content errors are highlighted using colored boxes: inconsistent fonts across pages (red boxes), content overflow beyond page boundaries (green boxes), chaotic layouts (blue boxes), and repeated content (yellow boxes). Source: (He et al., 2020; Wong et al., 2020)

need to be modified at this stage, the likelihood of errors is significantly lower compared to earlier phases. As a result, the Code Reviewer is not reintroduced for further checks. As shown in the Fig. 2(bottom), among three slides, the first one fails to meet visual requirements. Under the iterative checks of the Page Reviewer, its code is regenerated and rendered repeatedly until it passes the review.

4 Experiment

We present the implementation details of our framework, along with its qualitative and quantitative results, as well as comparisons with other state-of-the-art models, including PPTAgent (Zheng et al., 2025), KCTV (Cachola et al., 2024), and AutoPresent (Ge et al., 2025).

4.1 Implementation Details

Our model is built upon LLM and VLM, specifically Qwen2.5-72B-Instruct (Yang et al., 2024a) for the LLM and Qwen2.5-VL-72B-Instruct (Bai et al., 2025) for the VLM. The Text Summarizer, Code Generator, and Code Reviewer components leverage the LLM, while the Image Captioner and Page Reviewer utilize the VLM. The dataset used in this study is sourced from the DOC2PPT dataset (Fu et al., 2022). From this dataset, we selected 200 samples rich in images, tables, and other content. We compare our method with For both our approach and PPTAgent, we employed locally deployed Qwen models. For KCTV and AutoPresent, we used the open-source code repositories provided by their authors and followed the best practices in their publications, using OpenAI's official API for computation.

4.2 Qualitative Results and Comparisons

To evaluate the effectiveness of our proposed method, we conducted a comprehensive comparison with other state-of-the-art models. Our method takes text-and-image documents as input and generates multi-page slides as output. To the best of our knowledge, PPTAgent is the only open-source solution that fully aligns with our setting. Additionally, we compared our method with KCTV and AutoPresent, which are state-of-the-art models for generating multi-page plain text and single-page text-and-image outputs, respectively.

For multi-page content generation, as illustrated in Fig. 4, our method significantly outperforms others in terms of aesthetic layout and content consistency. PPTAgent exhibits several issues in text-and-image layout, such as inconsistent fonts across pages (highlighted in red boxes), content overflow beyond page boundaries (green boxes), chaotic lay-

outs (blue boxes), and repeated content (yellow boxes). Our approach leverages the lightweight Slidev framework, resulting in stronger content consistency and better integration with code review, which allows for more effective detection of code errors. Furthermore, the page review mechanism detects visually uncoordinated elements on slides, which are sometimes overlooked at the code level alone. As for KCTV, it lacks support for generating images within slides and is limited to a small set of templates, leading to insufficient diversity in generated styles.

Figure 5: Qualitative comparisons between Ours and AutoPresent demonstrate our superior design quality. Source: (Wong et al., 2020; Izmailov et al., 2020)

For single-page content generation, we randomly selected two slides generated by our framework and asked GPT-4o (Hurst et al., 2024) to provide a detailed description of their layout and content. This description was then used as a prompt for AutoPresent. Since AutoPresent does not support image recognition, we focused solely on the design of the generated slides. As shown in Fig. 5, AutoPresent suffers from poor design quality and suboptimal layouts.

Additionally, the effectiveness of our proposed page review mechanism is demonstrated in Fig. 6. We aim for the PageReviewer to make minimal changes to the previously generated content, focusing solely on adjusting the positioning, size, and layout of design elements. In the example shown above, the image extended beyond the page boundaries, resulting in incomplete display. To resolve this, the PageReviewer adjusted the image size. In the example in the middle, the aspect ratio of the image was close to 1:1, which appeared too small in a single-column layout. Therefore, it was adjusted to a two-column layout. In the example below, the text crowded in the bottom-left corner of the slide and partially hidden has been reorganized into a bullet point format. These adjustments not only make the generated slides more visually appealing but also ensure that the information is presented more effectively, enhancing the overall user experience.

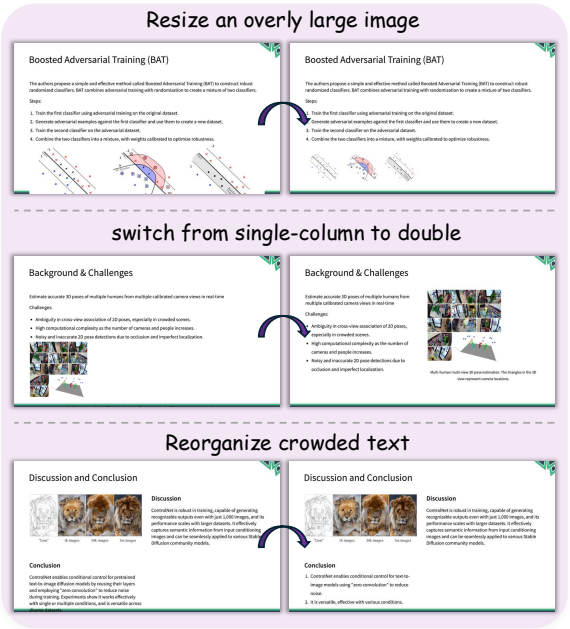

Figure 6: Typical design adjustments made by the Page Reviewer address a variety of issues. These adjustments ensure improved visual quality. Source: (Pang et al., 2020; Tang et al., 2014; Zhang et al., 2023)

4.3 Quantitative Results and Comparisons

We compared our method with multi-page content generation methods (PPTAgent, KCTV). Given the complexity of the generated content, we conducted evaluations using three approaches: traditional metrics, LLM scores, and human evaluations.

We adopt three categories of traditional metrics: Text Similarity, Text-Image Relevance, and Overall Score.

- Text Similarity: The similarity metrics evaluate the alignment between the reference text and the text on the generated slides. These metrics include the Rough-L score (Lin, 2004), based on the Longest Common Subsequence (LCS), and the Coverage score (Bandyopadhyay et al., 2024), which leverages cosine similarity between sentence embeddings. Due to the excessive word count in the original document and the typically concise nature of text in slides, similarity metrics

	Text Similarity (%)		Text-Image Relevance (%)		Overall Score (%)	
	Rough-L	Coverage	Clip	LongClip	Success Rate	Figure Proportion
Ours	21.95	27.70	**30.19**	**32.37**	91.26	88.35
Ours w/o Page Review	22.16	28.08	29.02	30.82	89.68	**88.61**
Ours w/o Code Review	18.47	27.21	28.74	29.49	58.72	81.59
PPTAgent	20.81	29.14	29.53	30.18	88.36	76.12
KCTV	**25.67**	**33.82**	/	/	**94.90**	/

Table 1: Quantitative comparison of three variations of our method, along with PPTAgent and KCTV, evaluated across different metrics. All metrics are percentage-based, with higher values indicating better performance. For each column, **bold** indicates the best performance, while underlined represents the second-best.

often become highly inaccurate when there is a significant length disparity. To address this, we utilize GPT-4 (Achiam et al., 2023) to abbreviate the original text while preserving its meaning as much as possible. The shortened version is then used as the reference text to calculate similarity with the text in the slides.

- Text-Image Relevance: For slides containing images, text-image relevance measures the alignment between each image and its corresponding page text. The final score is calculated as the average relevance across all such slides. Here the relevance is assessed using Clip (Radford et al., 2021) and LongClip (Zhang et al., 2024a), metrics specifically adapted for evaluating text-image similarity. Since some slides contain extensive text while others may only feature a title, we employ similarity metrics designed to handle texts of varying lengths effectively.

- Overall Score: Overall Score includes Success Rate and Figure Order. Success Rate represents the ratio of successful runs to the total number of runs, with each model being run three times for each data sample. Figure Proportion refers to the percentage of images from the original document that are included in the generated slides. These images are typically crucial, as they often encapsulate a significant portion of the document's content.

For all metrics, we calculate the average score on all successfully generated samples, with higher values indicate better performance.

As shown in the Table. 1, in the Text Similarity dimension, our method performs on par with the contemporary work PPTAgent, though the scores are slightly lower than KCTV. We believe this is because KCTV focuses specifically on text processing, while our approach handles multimodal inputs, taking into account layout design and text-image pairing, which may result in a slight loss of pure textual information. In the Text-Image Relevance dimension, our method slightly outperforms PPTAgent. This is due to our Image Captioner, which provides detailed descriptions for images, facilitating better matching between images and corresponding text. For cases involving long texts, where some text or images might be obscured or extend beyond the page boundaries, our Page Reviewer effectively mitigates such issues. As a result, our approach performs significantly better in long-clip evaluations compared to similar methods. In the Overall Score dimension, our Code Review mechanism significantly enhances the Success Rate of execution. This also applies to the Figure Proportion metric, as the Code Reviewer ensures that the generated code aligns with the prior summaries, allowing most of the images in the document to be effectively included.

Alongside traditional metircs, we compare our model with the concurrent work PPTAgent on LLM scores and human evaluations. Twenty users with experience in AI tools and research backgrounds, along with GPT-4o (Hurst et al., 2024), evaluated 10 sets of slides generated from the same input. The evaluation focused on four aspects: Page Design, Text Coherence, Text-Image Relevance and Page Consistency, with scores ranging from 1 to 10, higher the better. Users and GPT-4o share the same question template.

- Page Design evaluates whether the layout of individual slides is aesthetically pleasing and harmonious, with a proper balance between text and images, clarity, and readability.

- Text Coherence assesses whether the text in the slides aligns with the reference text. Similarly, we use an LLM (Achiam et al., 2023) to preprocess the original document.

- Text-Image Relevance measures the alignment between text and images on each page.
- Page Consistency examines whether the design style is consistent across all slides, including design elements like overall theme, font type and size.

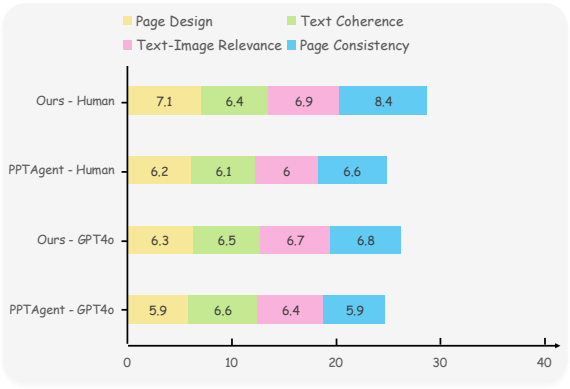

Figure 7: Human and GPT-4o evaluations comparing our approach with PPTAgent across four different aspects. Every score ranges from 1 to 10, higher the better.

As shown in Fig. 7, our method outperforms PPTAgent in terms of the overall scores from both humans and GPT. For the Text Coherence dimension, the difference between the two methods is negligible. In Text-Image Relevance, our scores are slightly higher than those of PPTAgent, which aligns with our previous experiments.

Our most significant advantage lies in Page Design and Page Consistency. This is due to the presence of review mechanisms that ensure high visual quality, both within individual pages and across multiple pages. Additionally, human evaluators tend to favor our approach in terms of design and consistency, which we believe is because humans are more sensitive to subtle differences in design. This indicates that our method aligns more closely with human aesthetic preferences.

	API Calls	Generation Time
Stage 1	3	14.6s
Stage 2: Code Review	4.4	8.9s
Stage 2: Visual Review	3.8	51.5s

Table 2: Computational costs and generation time. The costs primarily come from MLLM API calls.

We also calculate the average computational costs and generation time, shown in Table. 2. In Stage 1, the number of calls is fixed. In Stage 2, due to the iterative review mechanism, the number of API calls is not fixed, as each iteration involves two API calls (Review and Regenerate). Our generation time is longer than PPTAgent, because the VLM-based Page Review accounts for a significant portion of the computational time.

Figure 8: Application: presentation generation from long text. The images are generated using an image generation model (Hurst et al., 2024) based on contextual information. Source: (Carroll, 2024)

Figure 9: Application: presentation generation from existing slides. The top-left corner shows a poorly designed slide generated by AutoPresent. Without altering the content, we generate slides with various themes featuring excellent layout design. Source: (Ge et al., 2025)

5 Applications

With minimal effort in modifying LLM prompts, our framework can support a wide range of slide generation applications, extending beyond text-image documents. For instance, when generating slides from text-only input, Fig. 8 illustrates an example of storyboard creation. Additionally, our framework can extract content from poorly designed existing slides and reformat them with enhanced layouts, as demonstrated in Fig. 9.

6 Conclusion

In this paper, we present PreGenie, an agentic framework powered by MLLMs for generating high-quality visual presentations. Built on the Slidev framework, PreGenie summarizes multimodal input, generates Markdown code, and refines the code and slides iteratively. Experiments show that PreGenie outperforms existing models in aesthetics, content consistency, and alignment with human design preferences. PreGenie also supports diverse practical applications, offering an effective solution for visual presentation generation.

Limitations

The performance of our PreGenie framework is primarily limited by two factors: the Slidev framework and MLLMs.

Slidev Framework: As a Markdown-based framework, Slidev offers simple and error-resistant layout rules but lacks flexibility in some areas. For example, placing an image in a specific position is not as intuitive as directly dragging and dropping visual elements. Additionally, Slidev's compatibility with popular presentation tools, such as Microsoft PowerPoint (PPT), is not seamless. Users cannot directly import PPT templates into the Slidev framework, which may limit customization options for some users.

MLLMs: While current MLLMs excel at understanding general images, they struggle with complex visual elements like charts and graphs, which are both common and critical in documents. This limitation can impact the quality of the generated presentations. Furthermore, the hallucination issue prevalent in MLLMs can occasionally result in intermediate code containing content completely unrelated to the input text and images, further compromising the output quality.

We believe that in the future, more flexible and standardized presentation-generation frameworks will emerge, and as MLLMs continue to improve in multimodal understanding and generation capabilities, they will be able to handle more complex tasks. This progress will address these limitations and enable higher-quality, structured presentation generation.

Ethical Considerations

The PreGenie framework processes only the documents provided by users. If a document contains unsafe or harmful visual content, the model may directly include it in the generated presentation. Therefore, we strongly advise users to avoid providing harmful input content to the model.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, and 1 others. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Sambaran Bandyopadhyay, Himanshu Maheshwari, Anandhavelu Natarajan, and Apoorv Saxena. 2024. Enhancing presentation slide generation by llms with a multi-staged end-to-end approach. *arXiv preprint arXiv:2406.06556*.

Isabel Cachola, Silviu Cucerzan, Allen Herring, Vuksan Mijovic, Erik Oveson, and Sujay Kumar Jauhar. 2024. Knowledge-centric templatic views of documents. *arXiv preprint arXiv:2401.06945*.

Lewis Carroll. 2024. Alice in wonderland. *Drama & Theatre*, 2024(116):39–39.

Boli Chen, Xin Huang, Lin Xiao, and Liping Jing. 2020. Hyperbolic capsule networks for multi-label classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3115–3124.

Haoyu Chen, Xiaojie Xu, Wenbo Li, Jingjing Ren, Tian Ye, Songhua Liu, Ying-Cong Chen, Lei Zhu, and Xinchao Wang. 2025. Posta: A go-to framework for customized artistic poster generation. *arXiv preprint arXiv:2503.14908*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Tsu-Jui Fu, William Yang Wang, Daniel McDuff, and Yale Song. 2022. Doc2ppt: Automatic presentation slides generation from scientific documents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 634–642.

Jiaxin Ge, Zora Zhiruo Wang, Xuhui Zhou, Yi-Hao Peng, Sanjay Subramanian, Qinyue Tan, Maarten Sap, Alane Suhr, Daniel Fried, Graham Neubig, and 1 others. 2025. Autopresent: Designing structured visuals from scratch. *arXiv preprint arXiv:2501.00912*.

Carlos Gonzalez-Morcillo, Gerhard Weiss, Luis Jimenez, and David Vallejo. 2007. A multi-agent approach to distributed rendering optimization. In *Proceedings Of The National Conference On Artificial Intelligence*, volume 22, page 1775. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Chaoyang He, Haishan Ye, Li Shen, and Tong Zhang. 2020. Milenas: Efficient neural architecture search via mixed-level reformulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11993–12002.

Nico Herbig, Tim Düwel, Santanu Pal, Kalliopi Meladaki, Mahsa Monshizadeh, Antonio Krüger, and Josef van Genabith. 2020. Mmpe: A multi-modal interface for post-editing machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1691–1702.

Sirui Hong, Xiawu Zheng, Jonathan Chen, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, and 1 others. 2023. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 3(4):6.

Yue Hu and Xiaojun Wan. 2014. Ppsgen: Learning-based presentation slides generation for academic papers. *IEEE transactions on knowledge and data engineering*, 27(4):1085–1097.

Bin Huang, Xin Wang, Hong Chen, Zihan Song, and Wenwu Zhu. 2024. Vtimellm: Empower llm to grasp video moments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14271–14280.

Shaofei Huang, Tianrui Hui, Si Liu, Guanbin Li, Yunchao Wei, Jizhong Han, Luoqi Liu, and Bo Li. 2020. Referring image segmentation via cross-modal progressive comprehension. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10488–10497.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. 2023. Layoutdm: Discrete diffusion model for controllable layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10167–10176.

Pavel Izmailov, Polina Kirichenko, Marc Finzi, and Andrew Gordon Wilson. 2020. Semi-supervised learning with normalizing flows. In *International conference on machine learning*, pages 4615–4630. PMLR.

Zaid Khan, Vijay Kumar BG, Samuel Schulter, Yun Fu, and Manmohan Chandraker. 2024. Self-training large language models for improved visual program synthesis with visual reinforcement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14344–14353.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, and 1 others. 2023. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026.

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

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

Jinlin Liu, Yuan Yao, Wendi Hou, Miaomiao Cui, Xuansong Xie, Changshui Zhang, and Xian-sheng Hua. 2020. Boosting semantic human matting with coarse annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8563–8572.

Andrew Luo, Zhoutong Zhang, Jiajun Wu, and Joshua B Tenenbaum. 2020. End-to-end optimization of scene layout. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3754–3763.

Jian Ma, Yonglin Deng, Chen Chen, Nanyang Du, Haonan Lu, and Zhenyu Yang. 2025. Glyphdraw2: Automatic generation of complex glyph posters with diffusion models and large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 5955–5963.

Tianyu Pang, Xiao Yang, Yinpeng Dong, Kun Xu, Jun Zhu, and Hang Su. 2020. Boosting adversarial training with hypersphere embedding. *Advances in Neural Information Processing Systems*, 33:7779–7792.

Wanli Peng, Hao Pan, He Liu, and Yi Sun. 2020. Ida-3d: Instance-depth-aware 3d object detection from stereo vision for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13015–13024.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, and 1 others. 2023. Chatdev: Communicative agents for software development. *arXiv preprint arXiv:2307.07924*.

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

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, and 1 others. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

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

Edward Sun, Yufang Hou, Dakuo Wang, Yunfeng Zhang, and Nancy XR Wang. 2021. D2s: Document-to-slide generation via query-based text summarization. *arXiv preprint arXiv:2105.03664*.

Siyu Tang, Mykhaylo Andriluka, and Bernt Schiele. 2014. Detection and tracking of occluded people. *International Journal of Computer Vision*, 110:58–69.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Ning Wang, Bingkun Yao, Jie Zhou, Xi Wang, Zhe Jiang, and Nan Guan. 2024. Large language model for verilog generation with golden code feedback. *arXiv preprint arXiv:2407.18271*.

KayYen Wong, Sameen Maruf, and Gholamreza Haffari. 2020. Contextual neural machine translation improves translation of cataphoric pronouns. *arXiv preprint arXiv:2004.09894*.

Weijia Wu, Zeyu Zhu, and Mike Zheng Shou. 2025. Automated movie generation via multi-agent cot planning. *arXiv preprint arXiv:2503.07314*.

Sheng Xu and Xiaojun Wan. 2022. Posterbot: A system for generating posters of scientific papers with neural models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 13233–13235.

Xuenan Xu, Jiahao Mei, Chenliang Li, Yuning Wu, Ming Yan, Shaopeng Lai, Ji Zhang, and Mengyue Wu. 2025. Mm-storyagent: Immersive narrated storybook video generation with a multi-agent paradigm across text, image and audio. *arXiv preprint arXiv:2503.05242*.

Hui Yang, Sifu Yue, and Yunzhong He. 2023. Auto-gpt for online decision making: Benchmarks and additional opinions. *arXiv preprint arXiv:2306.02224*.

Qwen An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxin Yang, Jingren Zhou, Junyang Lin, and 25 others. 2024a. Qwen2.5 technical report. *ArXiv*, abs/2412.15115.

Tao Yang, Yingmin Luo, Zhongang Qi, Yang Wu, Ying Shan, and Chang Wen Chen. 2024b. Posterllava: Constructing a unified multi-modal layout generator with llm. *arXiv preprint arXiv:2406.02884*.

Zhibo Yang, Lihan Huang, Yupei Chen, Zijun Wei, Seoyoung Ahn, Gregory Zelinsky, Dimitris Samaras, and Minh Hoai. 2020. Predicting goal-directed human attention using inverse reinforcement learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 193–202.

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

Beichen Zhang, Pan Zhang, Xiaoyi Dong, Yuhang Zang, and Jiaqi Wang. 2024a. Long-clip: Unlocking the long-text capability of clip. In *European Conference on Computer Vision*, pages 310–325. Springer.

Jiahao Zhang, Ryota Yoshihashi, Shunsuke Kitada, Atsuki Osanai, and Yuta Nakashima. 2024b. Vascar: Content-aware layout generation via visual-aware self-correction. *arXiv preprint arXiv:2412.04237*.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3836–3847.

Hao Zheng, Xinyan Guan, Hao Kong, Jia Zheng, Weixiang Zhou, Hongyu Lin, Yaojie Lu, Ben He, Xianpei Han, and Le Sun. 2025. Pptagent: Generating and evaluating presentations beyond text-to-slides. *arXiv preprint arXiv:2501.03936*.

A Prompt Details

A.1 PreGenie Framework

We present the prompts used in our PreGenie framework, including those for the Text Summarizer(Fig. 10), Image Captioner(Fig. 11), Code Generator(Fig. 12 and 13), Code Reviewer(Fig. 14), and Page Reviewer(Fig. 15).

Since the Code Generator operates in two stages, initial generation and subsequent regeneration after receiving review feedback, it has two distinct sets of prompts. The only difference is that the input for the latter includes the review feedback, and the prompt explicitly instructs the generator to consider this feedback. Fig. 12 and Fig. 13 illustrate the prompts used for the initial generation and the regeneration after review, respectively.

A.2 GPT-4o Evaluation

We present the GPT-4o prompts used for the calculation of quantitative metrics.

- Page Design: *You are a professional designer with very strict evaluation standards. Now please give a score of 1-10 based on the aesthetic quality and harmony of the layout of the slides, considering aspects like the balance between text and images, clarity, and readability. A score of 10 represents the best and 1 represents the worst.*
- Text Coherence: *You are a professional editor with very strict evaluation standards. Now please give a score of 1-10 based on how well the text in the slides aligns with the reference text, focusing on accuracy, clarity, and coherence. A score of 10 represents perfect alignment and 1 represents poor alignment.*
- Text-Image Relevance: *You are a professional visual content reviewer with very strict evaluation standards. Now please give a score of 1-10 based on how well the images on each slide align with the accompanying text, ensuring both relevance and effectiveness. A score of 10 represents perfect alignment and 1 represents no alignment.*
- Page Consistency: *You are a professional presentation designer with very strict evaluation standards. Now please give a score of 1-10 based on how consistent the design style is across all slides, including elements like overall theme, font type, font size, and other design elements. A score of 10 represents perfect consistency and 1 represents poor consistency.*

System Prompt:

```
You are a professional text summarization
assistant specializing in accurately condensing
written content while preserving key details and
important information. Your task is to extract and
present the most relevant points of a document,
including the title, author(s), affiliation(s),
and
other critical metadata, in a clear and concise
manner. Your output must be formatted as a
Markdown file.
```

User Prompt:

```
Here is a document for you to summarize:
<Document text>

Please summarize this document and generate a
Markdown file. Ensure the summary includes the
following:
1. The title of the document (if available).
2. The author's name(s) (if available).
3. The author's affiliation(s) (if mentioned).
4. A concise summary of the main content,
focusing on key points, findings, or conclusions.

Output the summary in a Markdown format as
<TextSummary.md>. Ensure the details are
accurate and well-organized.
```

Input:

```
<Document Text>
```

Output:

```
<TextSummary.md>
```

Figure 10: Prompts for Text Summarizer.

System Prompt:

```
You are an expert image captioning assistant. Your
role is to generate meaningful captions for images
based on their content and the context provided in
a document. Ensure that your captions are accurate,
descriptive, and aligned with the references in
the document text. Present your output in a clear
and organized Markdown format.
```

User Prompt:

```
Here are the images and the corresponding
document text:
<Document Text>, <Document Images>

Please analyze the images and, based on their
content and the context of their references in the
document:
1. Assign a title to each image.
2. Provide a detailed explanation of what the
image shows and its relevance.
3. Indicate where the image is referenced in the
text.
4. Include the filename of each image.

Output your captions in a Markdown file named
<ImageCaption.md>. Ensure clarity, accuracy,
and proper formatting.
```

Input:

```
<Document Text>, <Document Images>
```

Output:

```
<ImageCaption.md>
```

Figure 11: Prompts for Image Captioner.

System Prompt:
You are a highly skilled code generation assistant. Your role is to generate high-quality, well-structured code based on the provided instructions, ensuring it adheres to the specified requirements and formatting conventions. Your outputs should be accurate, organized, and easy to use.

User Prompt:
Here is a Slidev grammar example file: <Slidev Grammar>
Additionally, here are two files:
Text Summary: <TextSummary.md>
Image Captions: <ImageCaption.md>

Please merge the content of <TextSummary.md> and <ImageCaption.md> into a single file formatted using the Slidev grammar provided. The merged file should meet the following requirements:
1. Use the image descriptions from <ImageCaption.md> as the definitive source for image content.
2. Design each page so that elements in the same column are not overcrowded. Split content into multiple columns if necessary to prevent overflow.
3. Avoid pages with too few elements. Expand content where needed to ensure an appropriate balance.
4. If certain columns contain only images without text, center the images on the page.
5. Consider the aspect ratio of images:
6. If the aspect ratio exceeds 2:1, the image should span multiple columns in multi-column layouts rather than appearing in a single column.
7. The first page should include the title and author information.
8. The last page should serve as a summary page.

Output the final code in Markdown format as <SlidevCode.md>. Ensure the code is clean, adheres to the Slidev grammar, and satisfies all specified layout requirements.

Input:
<Slidev Grammar>, <TextSummary.md>, <ImageCaption.md>

Output:
<SlidevCode.md>

Figure 12: Prompts for Code Generator, without review.

System Prompt:
You are a highly skilled code generation assistant…
% Same as the unreviewed version

User Prompt:
Here is a Slidev grammar example file: <Slidev Grammar>
Additionally, here are three files:
Text Summary: <TextSummary.md>
Image Captions: <ImageCaption.md>
Review Feedback: <CodeReview.md> or <PageReview.md>

Please merge the content of <TextSummary.md> and <ImageCaption.md> into a single file formatted using the Slidev grammar provided. While doing so, take into account the feedback provided in <CodeReview.md> or <PageReview.md> and address any relevant issues. The merged file should meet the following requirements:
1. Use the image descriptions from <ImageCaption.md>…
% Same as the unreviewed version

Input:
<Slidev Grammar>, <TextSummary.md>, <ImageCaption.md>, <CodeReview.md> or <PageReview.md>

Output:
<SlidevCode.md>

Figure 13: Prompts for Code Generator, with review.

System Prompt:

```
You are a highly skilled code reviewer. Your role
is to carefully analyze and evaluate code for
correctness, clarity, and adherence to the given
specifications. Your feedback should be precise,
constructive, and well-structured.
```

User Prompt:

```
Here is a Slidev grammar example file: <Slidev
Grammar>
Additionally, here are two files:
Text Summary: <TextSummary.md>
Image Captions: <ImageCaption.md>

Please review the code in <SlidevCode.md> to
ensure:
1. It adheres to the Slidev grammar described in
<Slidev Grammar>.
2. The content aligns with the information in
<TextSummary.md> and <ImageCaption.md>.
3. The code meets all content and layout
requirements, including handling of images, text,
and page structure as specified.
Output your review as a Markdown file named
<CodeReview.md>. Your review should clearly
identify any errors or inconsistencies in the code,
along with suggestions for improvement. If the
code is correct, confirm that it meets all
requirements.
```

Input:

```
<Slidev Grammar>, <TextSummary.md>,
<ImageCaption.md>, <SlidevCode.md>
```

Output:

```
<CodeReview.md>
```

Figure 14: Prompts for Code Reviewer.

System Prompt:

```
You are an expert visual page reviewer. Your role
is to evaluate the layout and design of slides,
ensuring they are visually appealing and properly
aligned. Your feedback should be clear,
actionable, and focused on improving the layout
without altering the core content.
```

User Prompt:

```
Here are some slides: <SlidevPages>
Additionally, here are two supporting files:
Image Information: <ImageCaption.md>
Original Images: <Document Images>

Please review each slide to check:
1. Whether any text or image exceeds the slide
boundaries.
2. Whether the layout ensures a proper balance
between text and images, avoiding overcrowding
or large empty spaces.
3. Whether the font sizes and styles are legible
and consistent throughout the slide, ensuring
readability without clashing with the visuals.
4. Whether the aspect ratios of images are
preserved, and whether wide or tall images are
placed appropriately without distorting the layout.

For each slide:
1. Indicate whether modifications are needed by
answering with "yes" or "no".
2. If "yes", provide specific suggestions to
adjust
the positions of existing images. Do not add or
remove any images.
Output your review as a Markdown file named
<PageReview.md>. Ensure your feedback is
concise and easy to follow.
```

Input:

```
<SlidevPages>, <Document Images>,
<ImageCaption.md>
```

Output:

```
<PageReview.md>
```

Figure 15: Prompts for Page Reviewer.

B Application Details

Presentation Generation from Long Text. Since we only have textual input, we pass the output of the Text Summarizer to an external LLM to generate prompts for a visual generation model. These prompts are then used to call the external visual generation model to create images. The resulting images can then be used as illustrations, which are subsequently provided to the Image Captioner and Code Generator for further use.

Presentation Generation from Existing Slides. After extracting information from the existing slides, we directly use it as the output of Text Summarizer and Image Captioner, which can then be passed to the subsequent Code Generator for further processing.

C Additional Results

Here we show additional presentation results generated by our PreGenie framework. For better demonstration, the number of slides is restricted to six.

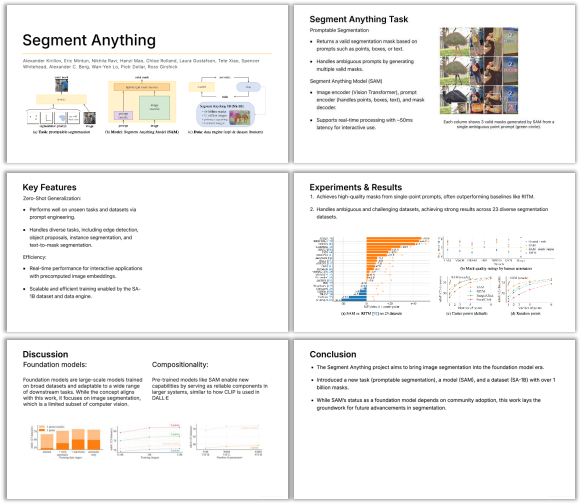

Figure 16: Additional result. Source: (Kirillov et al., 2023)

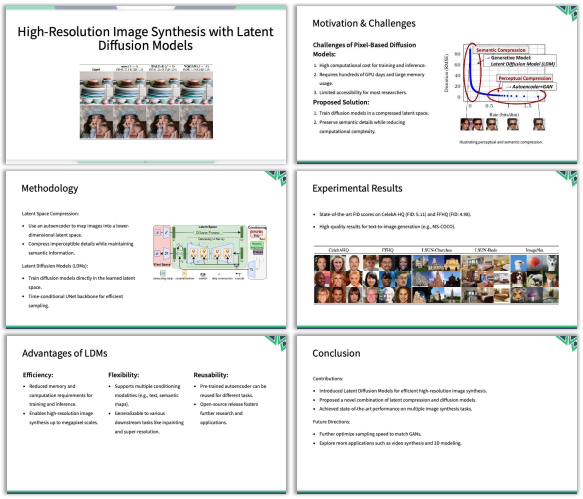

Figure 17: Additional result. Source: (Rombach et al., 2022)

Figure 18: Additional result. Source: (Liu et al., 2020)

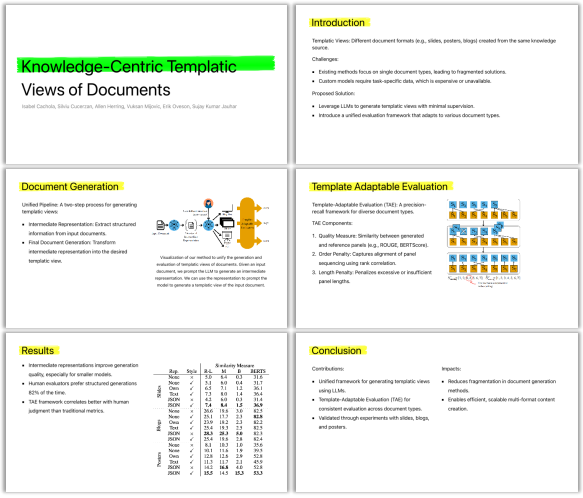

Figure 19: Additional result. Source: (Cachola et al., 2024)

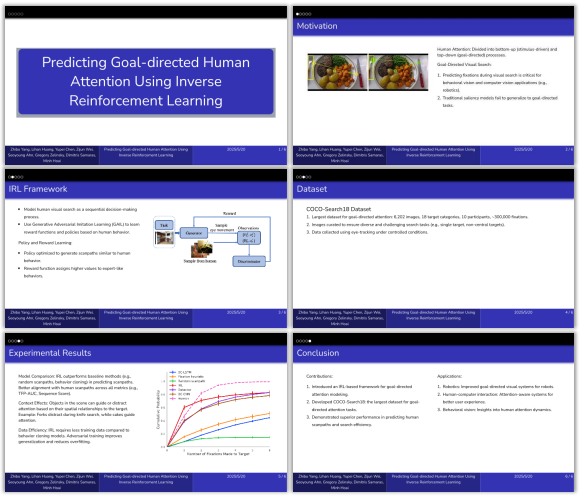

Figure 20: Additional result. Source: (Yang et al., 2020)

Figure 23: Additional result. Source: (Huang et al., 2020)

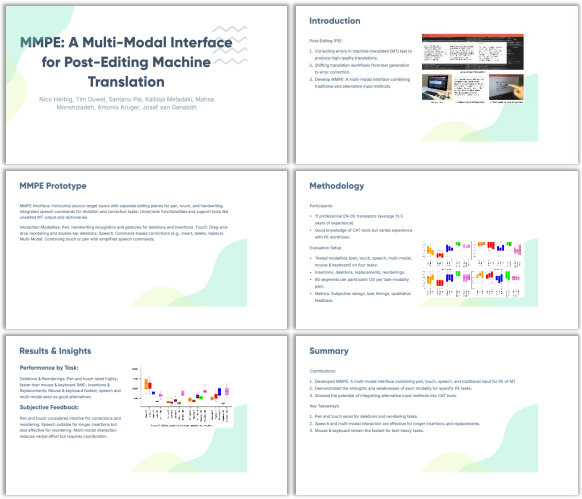

Figure 21: Additional result. Source: (Herbig et al., 2020)

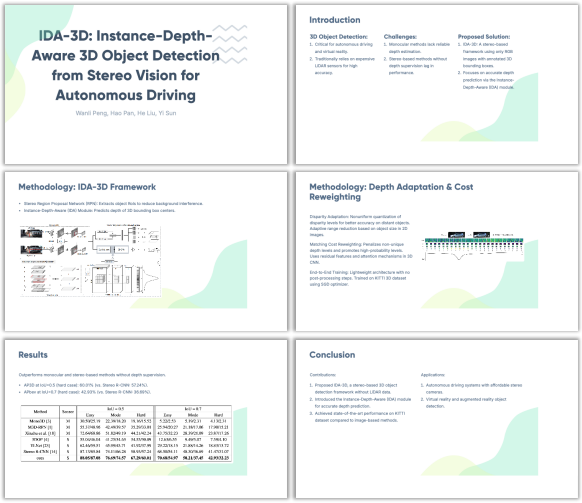

Figure 24: Additional result. Source: (Peng et al., 2020)

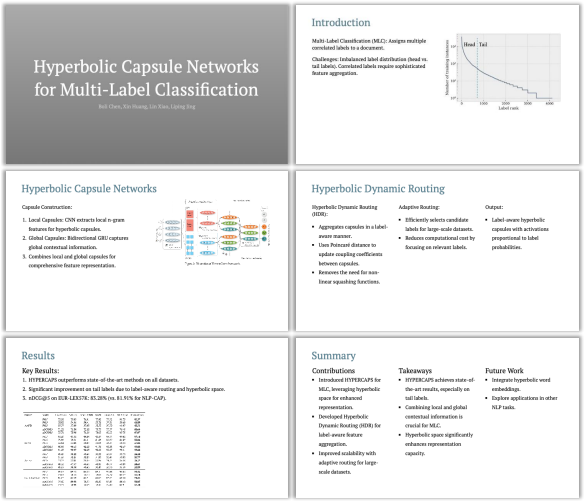

Figure 22: Additional result. Source: (Chen et al., 2020)

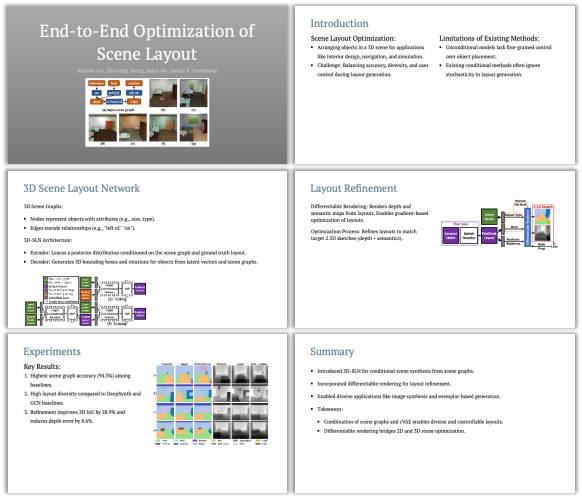

Figure 25: Additional result. Source: (Luo et al., 2020)