EMNLP
2025

SlideCoder: Layout-aware RAG-enhanced Hierarchical Slide Generation from Design

Wenxin Tang*, Jingyu Xiao*, Wenxuan Jiang, Xi Xiao, Yuhang Wang,
Xuxin Tang, Qing Li, Yuehe Ma, Junliang liu, Shisong Tang, Michael R. Lyu

1 Background

- Natural language cannot fully express complex layouts
- Multimodal models struggle with dense visual structures
- Generated code often fails to execute correctly

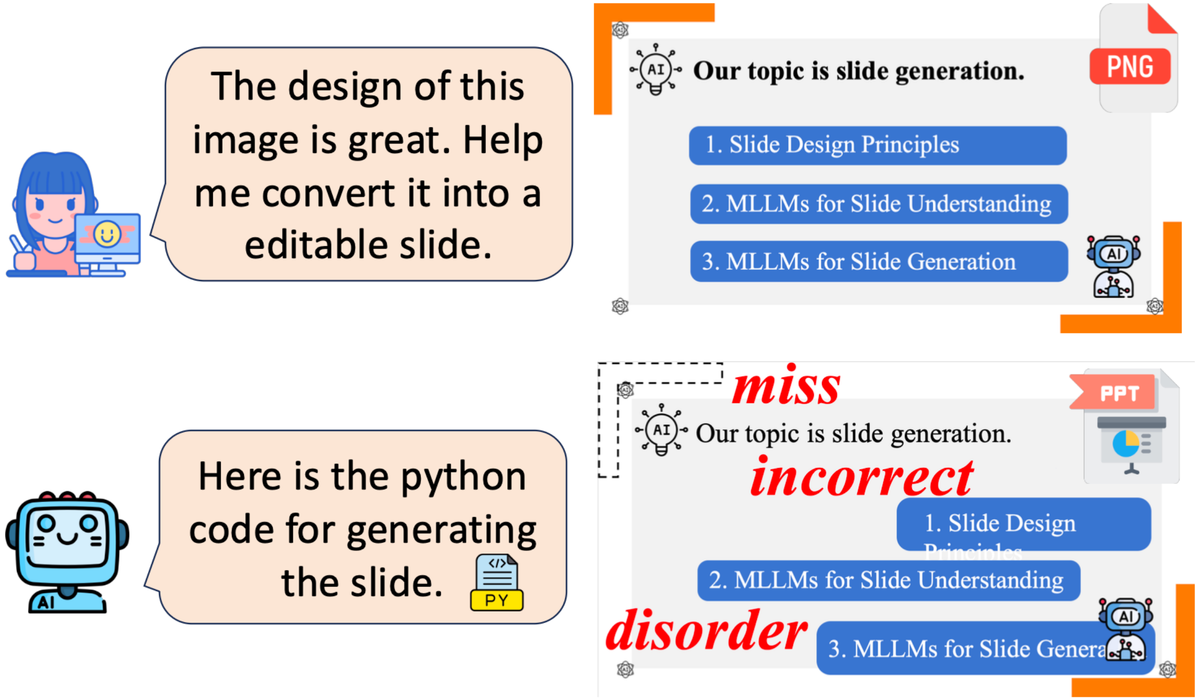

The design of this image is great. Help me convert it into a editable slide.
PNG
Our topic is slide generation.
1. Slide Design Principles
2. MLLMs for Slide Understanding
3. MLLMs for Slide Generation
AI
Here is the python code for generating the slide.
PY
miss
PPT
Our topic is slide generation.
incorrect
1. Slide Design Principles
2. MLLMs for Slide Understanding
disorder
3. MLLMs for Slide Genera
AI

Figure 1: Illustration of slide generation scenarios from design and mistakes made by MLLMs.

2 Slide2Code

EMNLP 2025

- A new benchmark for image-to-slide generation
- Categorized by Slide Complexity Metric (SCM)
- 900 samples across simple, medium, and complex layouts

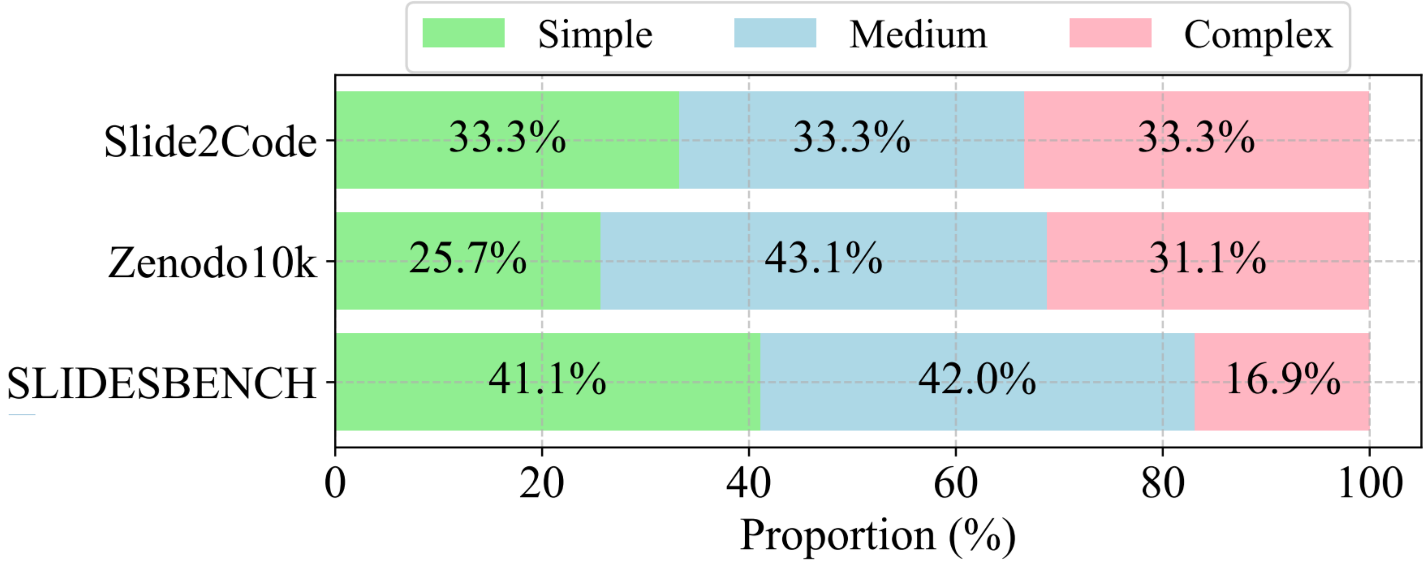
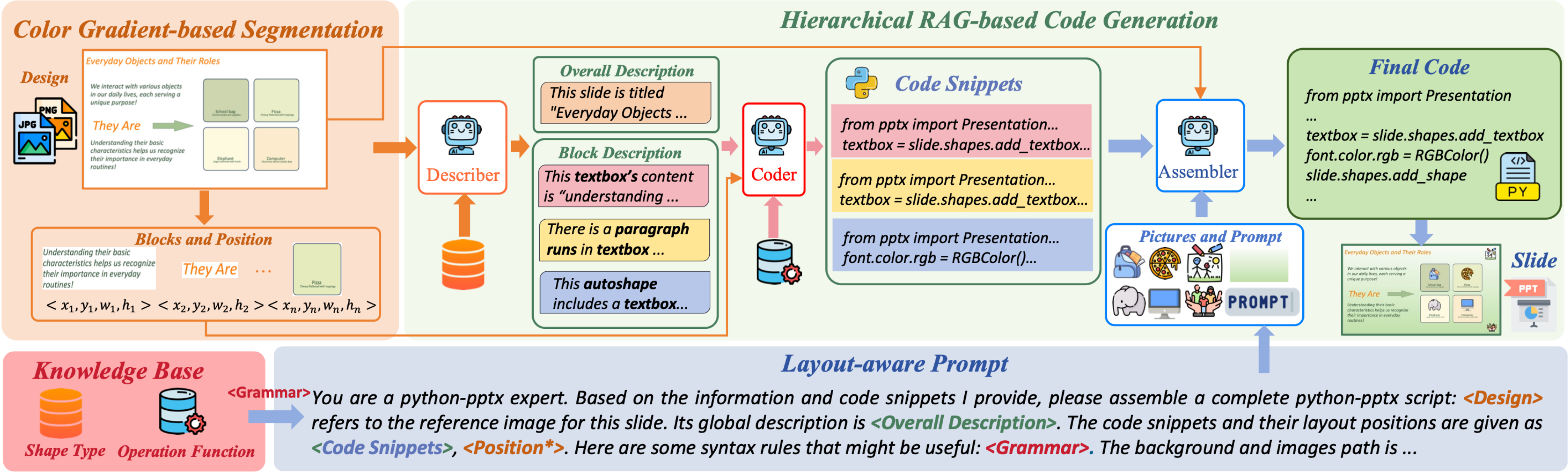


Figure 2: Proportion of samples across three levels in the Slide2Code, Zenodo10k, and SLIDEBENCH datasets.

EMNLP 2025

3 SlideCoder



Color Gradient-based Segmentation

Design

Everyday Objects and Their Roles

We interact with various objects in our daily lives, each serving a unique purpose!

They Are

Understanding their basic characteristics helps us recognize their importance in everyday routines!

Blocks and Position

Understanding their basic characteristics helps us recognize their importance in everyday routines! *They Are* …

$< x_1, y_1, w_1, h_1 > < x_2, y_2, w_2, h_2 > < x_n, y_n, w_n, h_n >$

Hierarchical RAG-based Code Generation

Describer

Overall Description

This slide is titled "Everyday Objects ...

Block Description

*This **textbox's** content is "understanding ...*

*There is a **paragraph runs in textbox** ...*

*This **autoshape** includes a **textbox**...*

Coder

Code Snippets

from pptx import Presentation…
textbox = slide.shapes.add_textbox…

from pptx import Presentation…
textbox = slide.shapes.add_textbox…

from pptx import Presentation…
font.color.rgb = RGBColor()…

Assembler

Pictures and Prompt

PROMPT

Final Code

```
from pptx import Presentation
...
textbox = slide.shapes.add_textbox
font.color.rgb = RGBColor()
slide.shapes.add_shape
...
```

Slide

Knowledge Base

Shape Type *Operation Function*

<Grammar>

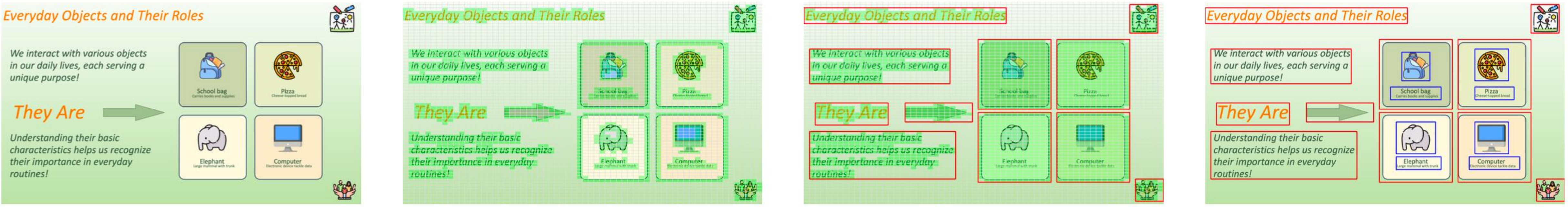
Layout-aware Prompt

You are a python-pptx expert. Based on the information and code snippets I provide, please assemble a complete python-pptx script: <Design> refers to the reference image for this slide. Its global description is <Overall Description>. The code snippets and their layout positions are given as <Code Snippets>, <Position>. Here are some syntax rules that might be useful: <Grammar>. The background and images path is ...*

- **CGSeg** — Color Gradient-based Segmentation
- **H-RAG** — Hierarchical Retrieval-Augmented Generation
- **LAP** — Layout-aware Prompting

4 CGSeg

EMNLP 2025



Everyday Objects and Their Roles

We interact with various objects in our daily lives, each serving a unique purpose!

They Are

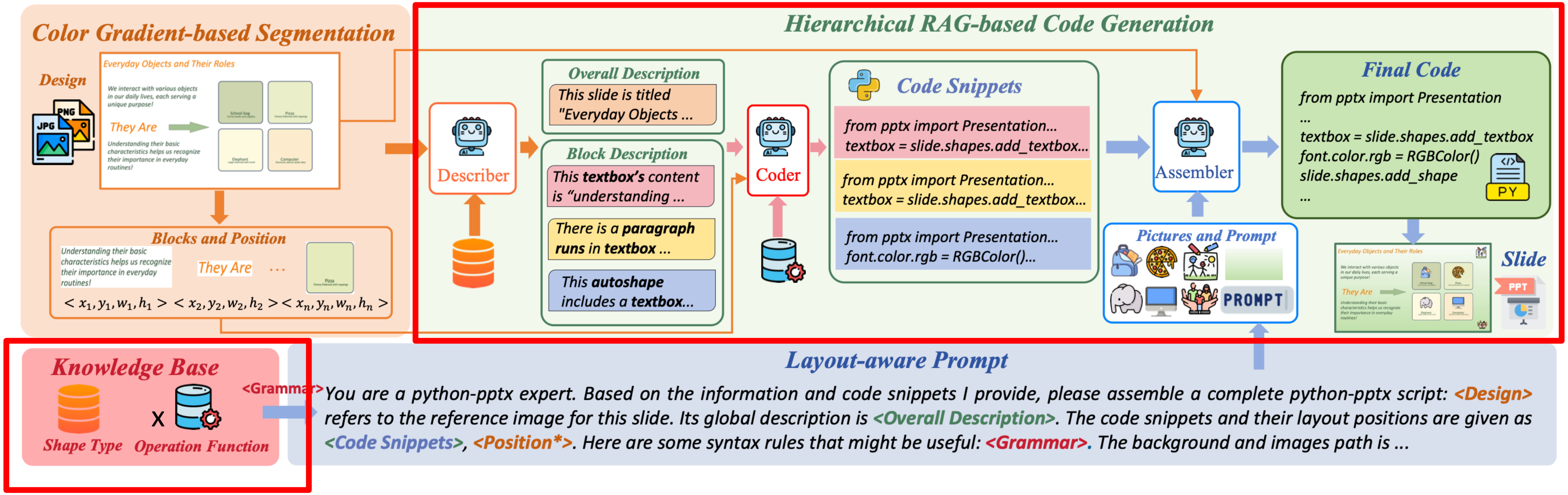
Understanding their basic characteristics helps us recognize their importance in everyday routines!

(a) Input Image (b) Activated Grid Blocks (c) Flood-filled Regions (d) Final result

Figure 4: An example of CGSeg applied to a slide reference image. The algorithm begins by computing color gradients (a-b), fills them (c), and recursively segments sub-regions (d).

- Recursive segmentation using color gradient
- Preserves spatial hierarchy and object boundaries
- Produces semantic sub-regions for code generation

5 H-RAG

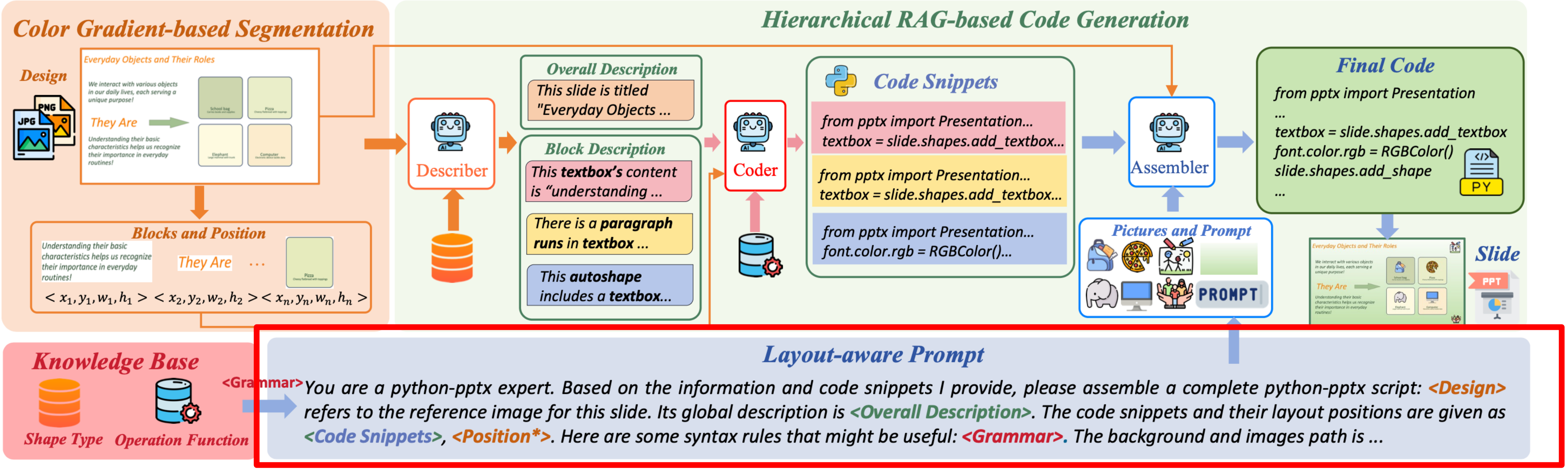

Color Gradient-based Segmentation
Design
PNG
JPG
Everyday Objects and Their Roles
We interact with various objects in our daily lives, each serving a unique purpose!
They Are
Understanding their basic characteristics helps us recognize their importance in everyday routines!
School bag
Pizza
Elephant
Computer
Blocks and Position
Understanding their basic characteristics helps us recognize their importance in everyday routines!
They Are
…
Pizza
$< x_1, y_1, w_1, h_1 > < x_2, y_2, w_2, h_2 > < x_n, y_n, w_n, h_n >$
Hierarchical RAG-based Code Generation
Describer
Overall Description
This slide is titled "Everyday Objects …
Block Description
This textbox's content is "understanding …
There is a paragraph runs in textbox …
This autoshape includes a textbox…
Coder
Code Snippets
from pptx import Presentation…
textbox = slide.shapes.add_textbox…
from pptx import Presentation…
textbox = slide.shapes.add_textbox…
from pptx import Presentation…
font.color.rgb = RGBColor()…
Assembler
Pictures and Prompt
PROMPT
Final Code
from pptx import Presentation
…
textbox = slide.shapes.add_textbox
font.color.rgb = RGBColor()
slide.shapes.add_shape
…
PY
Slide
PPT
Knowledge Base
x
Shape Type
Operation Function
<Grammar>
Layout-aware Prompt

You are a python-pptx expert. Based on the information and code snippets I provide, please assemble a complete python-pptx script: <Design> refers to the reference image for this slide. Its global description is <Overall Description>. The code snippets and their layout positions are given as <Code Snippets>, <Position*>. Here are some syntax rules that might be useful: <Grammar>. The background and images path is ...

- Two-level knowledge bases:
  - **Shape Type KB:** Object definitions and templates
  - **Operation KB:** Functions and syntax patterns
- Three agents: Describer, Coder, Assembler

6 Layout-aware prompt

EMNLP 2025


Color Gradient-based Segmentation
Design
PNG
JPG
Everyday Objects and Their Roles
We interact with various objects in our daily lives, each serving a unique purpose!
They Are
Understanding their basic characteristics helps us recognize their importance in everyday routines!
School bag
Pizza
Elephant
Computer
Blocks and Position
They Are
…
Pizza
Knowledge Base
Shape Type
Operation Function
<Grammar>
Hierarchical RAG-based Code Generation
Describer
Overall Description
Block Description
Coder
Code Snippets
Assembler
Pictures and Prompt
PROMPT
Final Code
PY
Slide
PPT

Understanding their basic characteristics helps us recognize their importance in everyday routines!

$< x_1, y_1, w_1, h_1 > < x_2, y_2, w_2, h_2 > < x_n, y_n, w_n, h_n >$

This slide is titled "Everyday Objects ...

This textbox's content is "understanding ...

There is a **paragraph runs** in **textbox** ...

This **autoshape** includes a **textbox**...

```
from pptx import Presentation...
textbox = slide.shapes.add_textbox...
```

```
from pptx import Presentation...
textbox = slide.shapes.add_textbox...
```

```
from pptx import Presentation...
font.color.rgb = RGBColor()...
```

```
from pptx import Presentation
...
textbox = slide.shapes.add_textbox
font.color.rgb = RGBColor()
slide.shapes.add_shape
...
```

Layout-aware Prompt

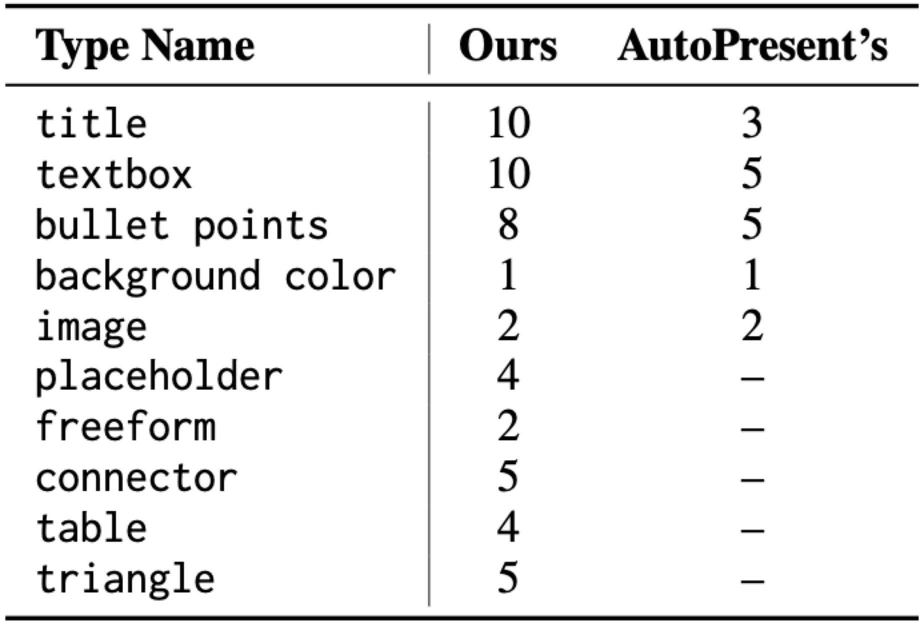
You are a python-pptx expert. Based on the information and code snippets I provide, please assemble a complete python-pptx script: <Design> refers to the reference image for this slide. Its global description is <Overall Description>. The code snippets and their layout positions are given as <Code Snippets>, <Position>. Here are some syntax rules that might be useful: <Grammar>. The background and images path is ...*

- Incorporates layout parameters (x, y, w, h)
- Uses consistent pptx coordinate units (inches)
- Ensures structural and visual alignment

7 SlideMaster

- Based on Qwen2.5-VL-7B
- Reverse-engineered data generation pipeline
- Expands object and style diversity (10 object types, 44 styles)

Table 2: Object Types and Corresponding Style count

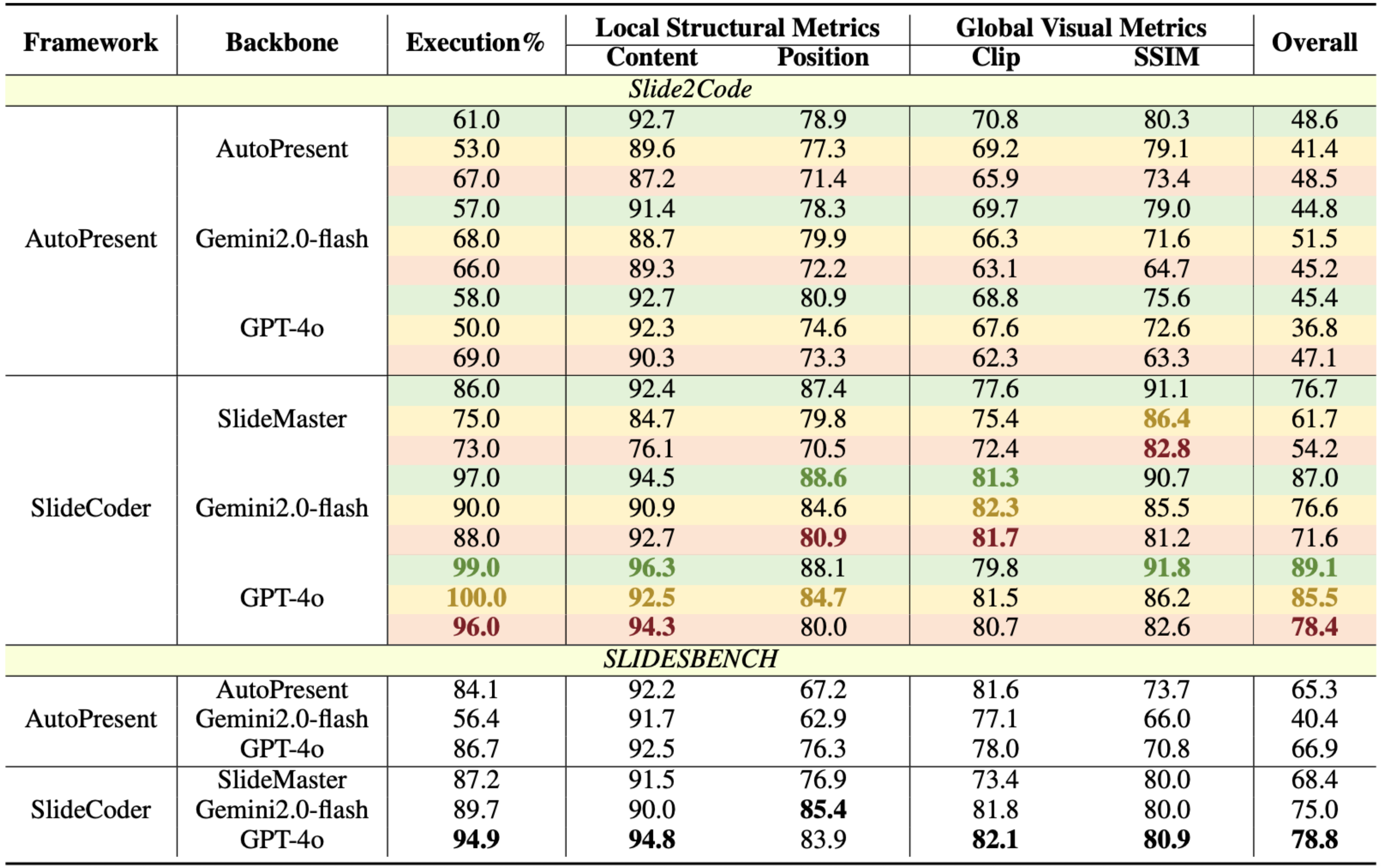


Type Name	Ours	AutoPresent’s
title	10	3
textbox	10	5
bullet points	8	5
background color	1	1
image	2	2
placeholder	4	–
freeform	2	–
connector	5	–
table	4	–
triangle	5	–

8 Experimental Results

- Compared models: AutoPresent, GPT-4o, Gemini 2.0, SlideMaster
- SlideCoder achieves top scores across all difficulty levels
- +40.5 points improvement over baselines

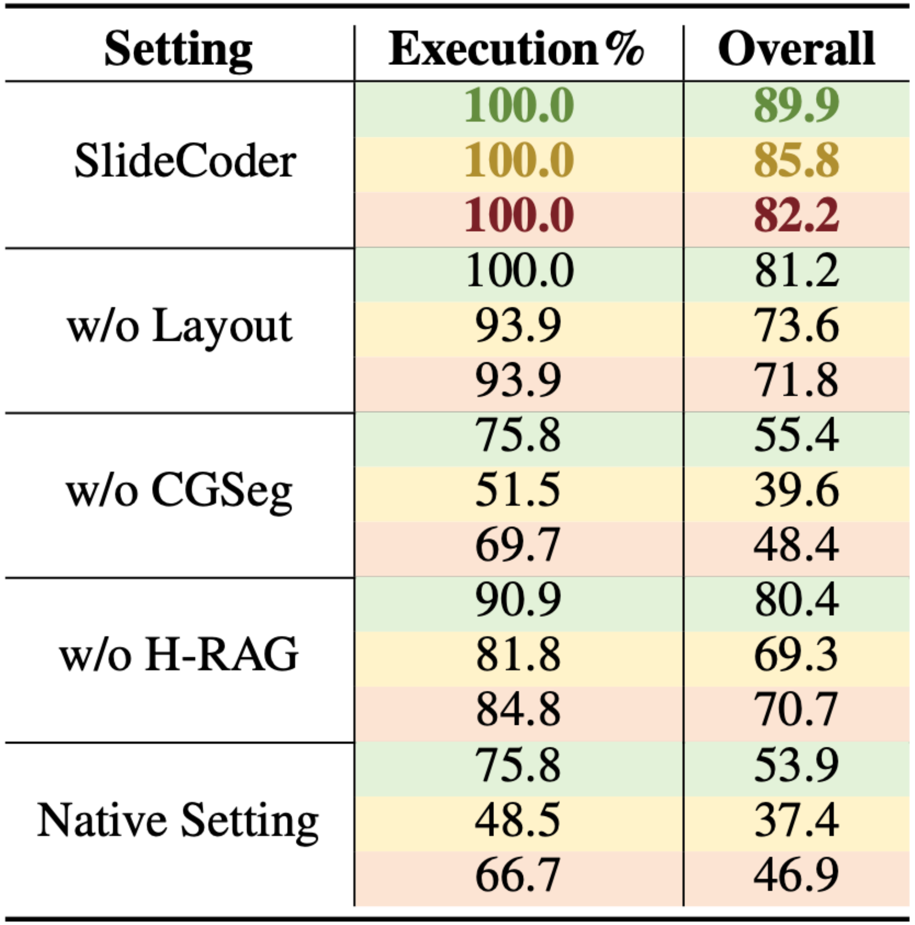
Table 1: Results on Slide2Code (top) and SLIDESBENCH (bottom) using SlideCoder and AutoPresent with different MLLMs. Green, yellow, and red indicate simple, medium, and complex levels in SlideCoder. **Bolded values** mark the best result per level.



Framework	Backbone	Execution%	Local Structural Metrics		Global Visual Metrics		Overall
			Content	Position	Clip	SSIM	
Slide2Code							
AutoPresent	AutoPresent	61.0	92.7	78.9	70.8	80.3	48.6
		53.0	89.6	77.3	69.2	79.1	41.4
		67.0	87.2	71.4	65.9	73.4	48.5
	Gemini2.0-flash	57.0	91.4	78.3	69.7	79.0	44.8
		68.0	88.7	79.9	66.3	71.6	51.5
		66.0	89.3	72.2	63.1	64.7	45.2
	GPT-4o	58.0	92.7	80.9	68.8	75.6	45.4
		50.0	92.3	74.6	67.6	72.6	36.8
		69.0	90.3	73.3	62.3	63.3	47.1
SlideCoder	SlideMaster	86.0	92.4	87.4	77.6	91.1	76.7
		75.0	84.7	79.8	75.4	**86.4**	61.7
		73.0	76.1	70.5	72.4	**82.8**	54.2
	Gemini2.0-flash	97.0	94.5	**88.6**	**81.3**	90.7	87.0
		90.0	90.9	84.6	**82.3**	85.5	76.6
		88.0	92.7	**80.9**	**81.7**	81.2	71.6
	GPT-4o	**99.0**	**96.3**	88.1	79.8	**91.8**	**89.1**
		100.0	**92.5**	**84.7**	81.5	86.2	**85.5**
		96.0	**94.3**	80.0	80.7	82.6	**78.4**
SLIDESBENCH							
AutoPresent	AutoPresent	84.1	92.2	67.2	81.6	73.7	65.3
	Gemini2.0-flash	56.4	91.7	62.9	77.1	66.0	40.4
	GPT-4o	86.7	92.5	76.3	78.0	70.8	66.9
SlideCoder	SlideMaster	87.2	91.5	76.9	73.4	80.0	68.4
	Gemini2.0-flash	89.7	90.0	**85.4**	81.8	80.0	75.0
	GPT-4o	**94.9**	**94.8**	83.9	**82.1**	**80.9**	**78.8**

9 Ablation

Table 3: Overall performance of ablation study.



Setting	Execution%	Overall
SlideCoder	**100.0**	**89.9**
	100.0	**85.8**
	100.0	**82.2**
w/o Layout	100.0	81.2
	93.9	73.6
	93.9	71.8
w/o CGSeg	75.8	55.4
	51.5	39.6
	69.7	48.4
w/o H-RAG	90.9	80.4
	81.8	69.3
	84.8	70.7
Native Setting	75.8	53.9
	48.5	37.4
	66.7	46.9

10 Case Study

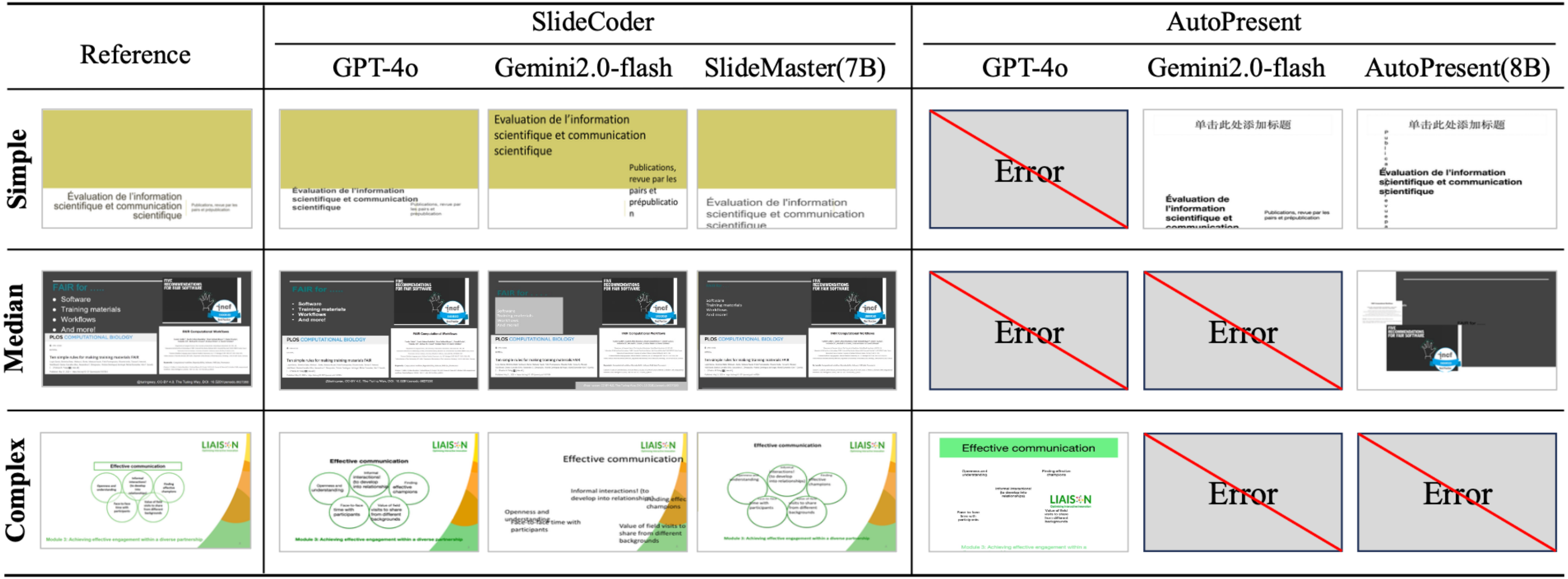


Figure 5: Examples of slides generated by different methods in three difficulty levels.

Thank you!

- **Speaker: Wenxin Tang**
- **Codes:** https://github.com/vinsontang1/SlideCoder
- **Email**: twx24@mails.tsinghua.edu.cn