Text-to-Scene Generation

Presenter: Xiaohao Sun

CMPT-413

March 30, 2026

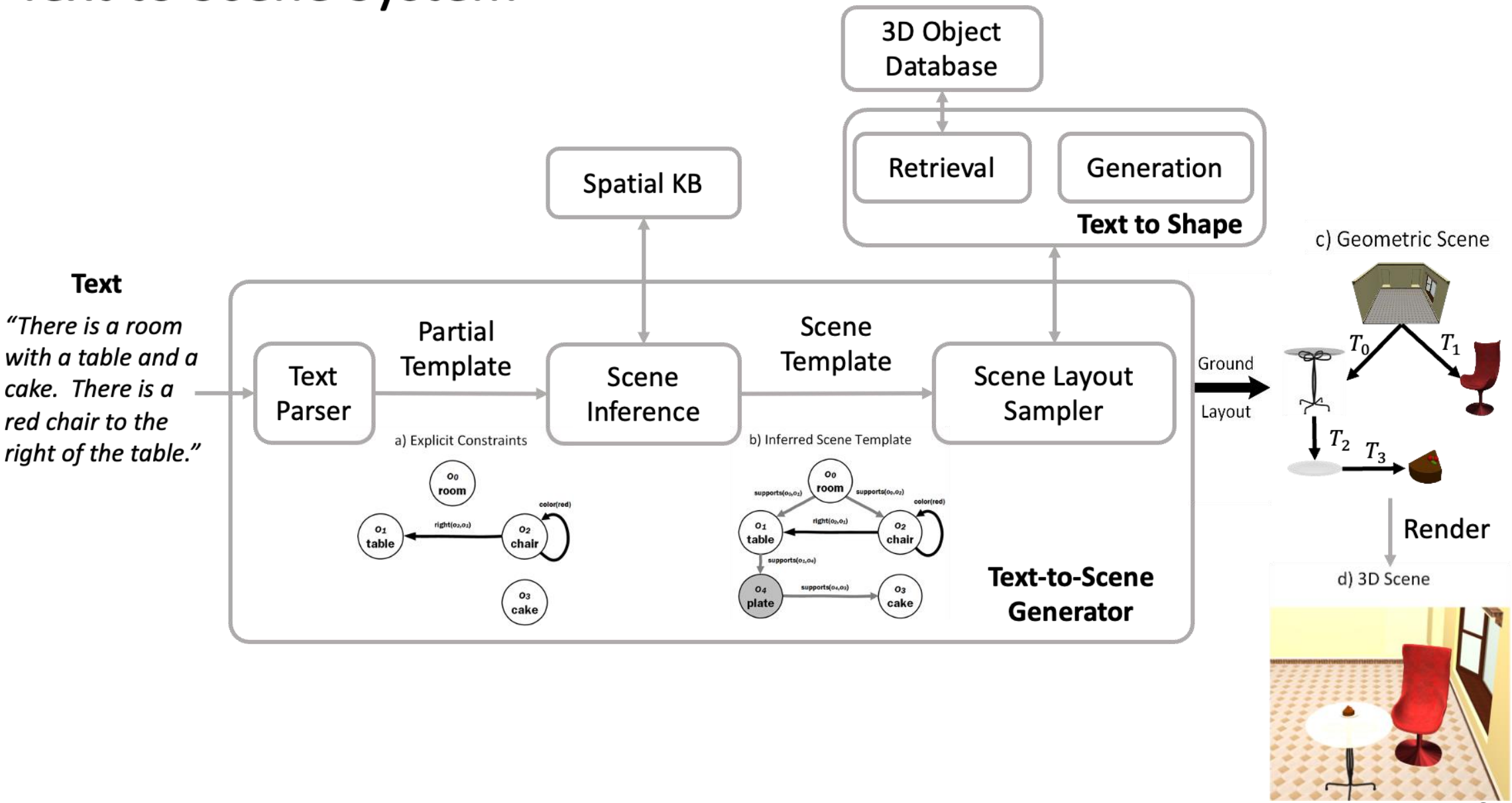
Generate 3D scenes from text descriptions

Three professors' office connected to a long hallway. The professor in office 1 is a fan of Star Wars.

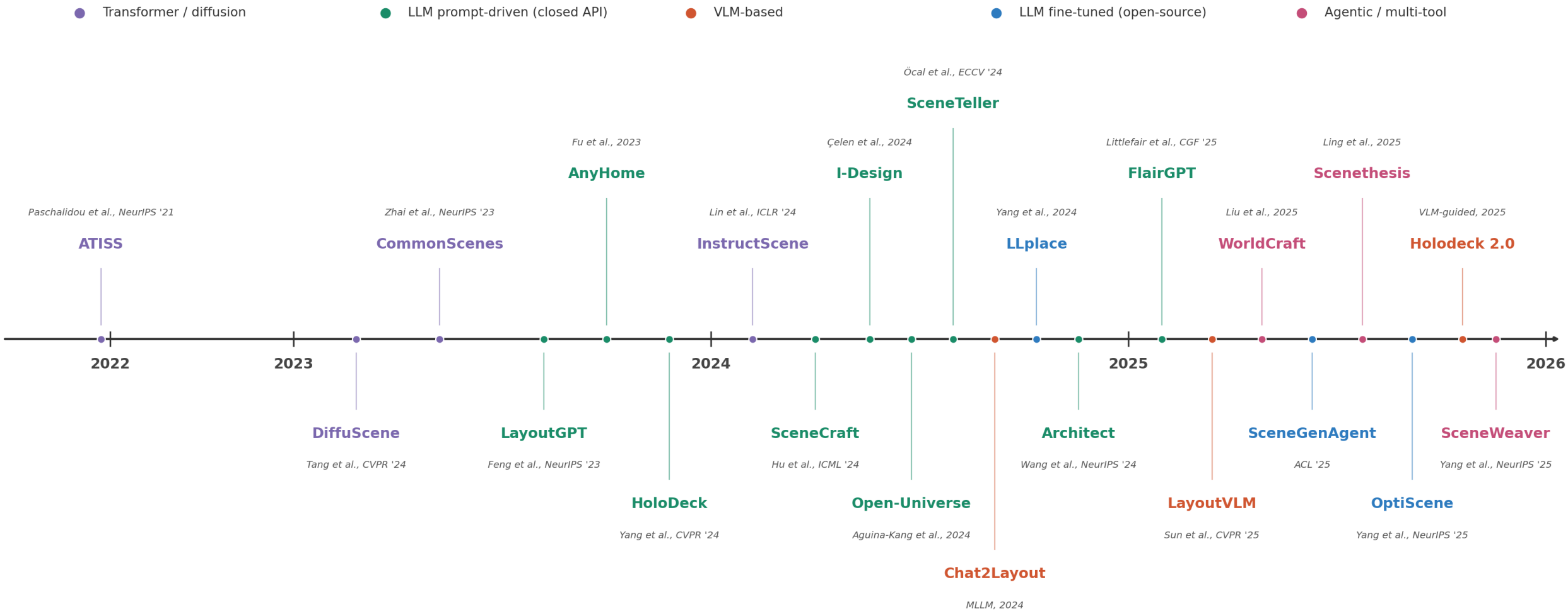
A 1b1b apartment of a researcher who has a cat.

Holodeck: Language Guided Generation of 3D Embodied AI Environments [Yang et al, 2023]

Text to Scene System



Timeline: Text-to-Indoor-Scene Generation Methods (2021–2025)



Trend: closed-API prompting → open-source fine-tuning → agentic multi-model systems

3D-FRONT

- 6,813 houses
- 14,629 rooms

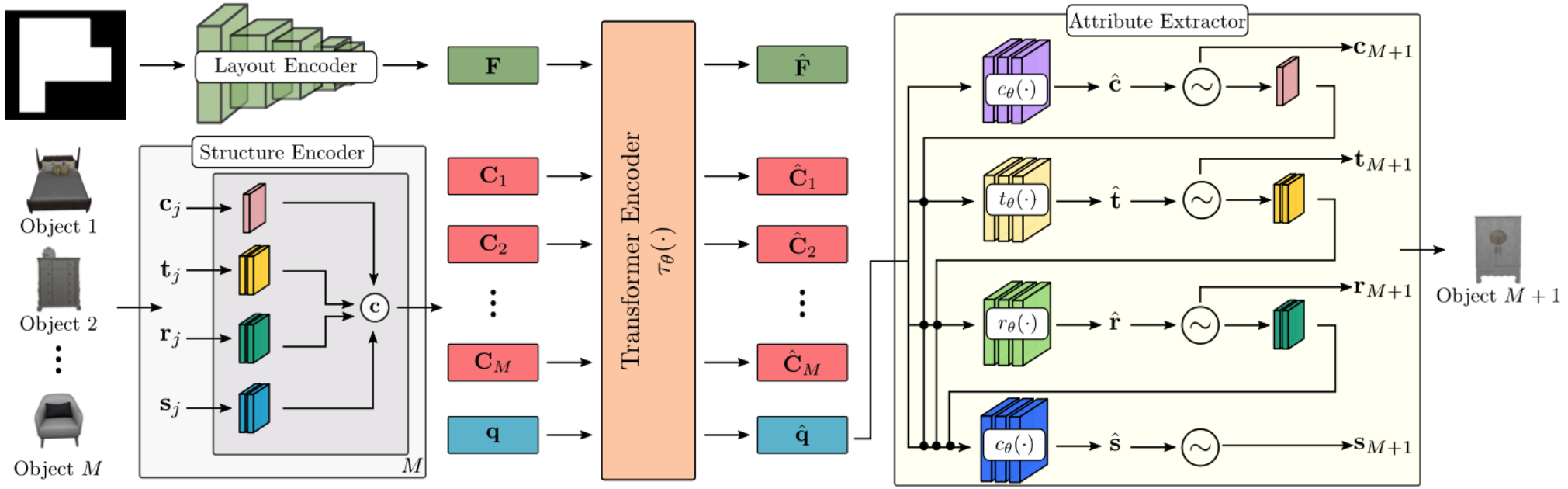
Room type	Number	Object categories
bedrooms	4000	21
living rooms	800	24
dining rooms	900	24

Approximate counts of each room type and object category in 3D-FRONT, as used by DiffuScene after post-processing [Tang et al, 2023]

3D-FRONT: 3D Furnished Rooms with layOuts and semaNTics [Fu et al, ICCV 2021]

Objects can be generated autoregressively in a scene

- E.g., ATISS
- Conditioned on room type and floorplan



ATISS: Autoregressive Transformers for Indoor Scene Synthesis [Paschalidou et al, NeurIPS 2022]

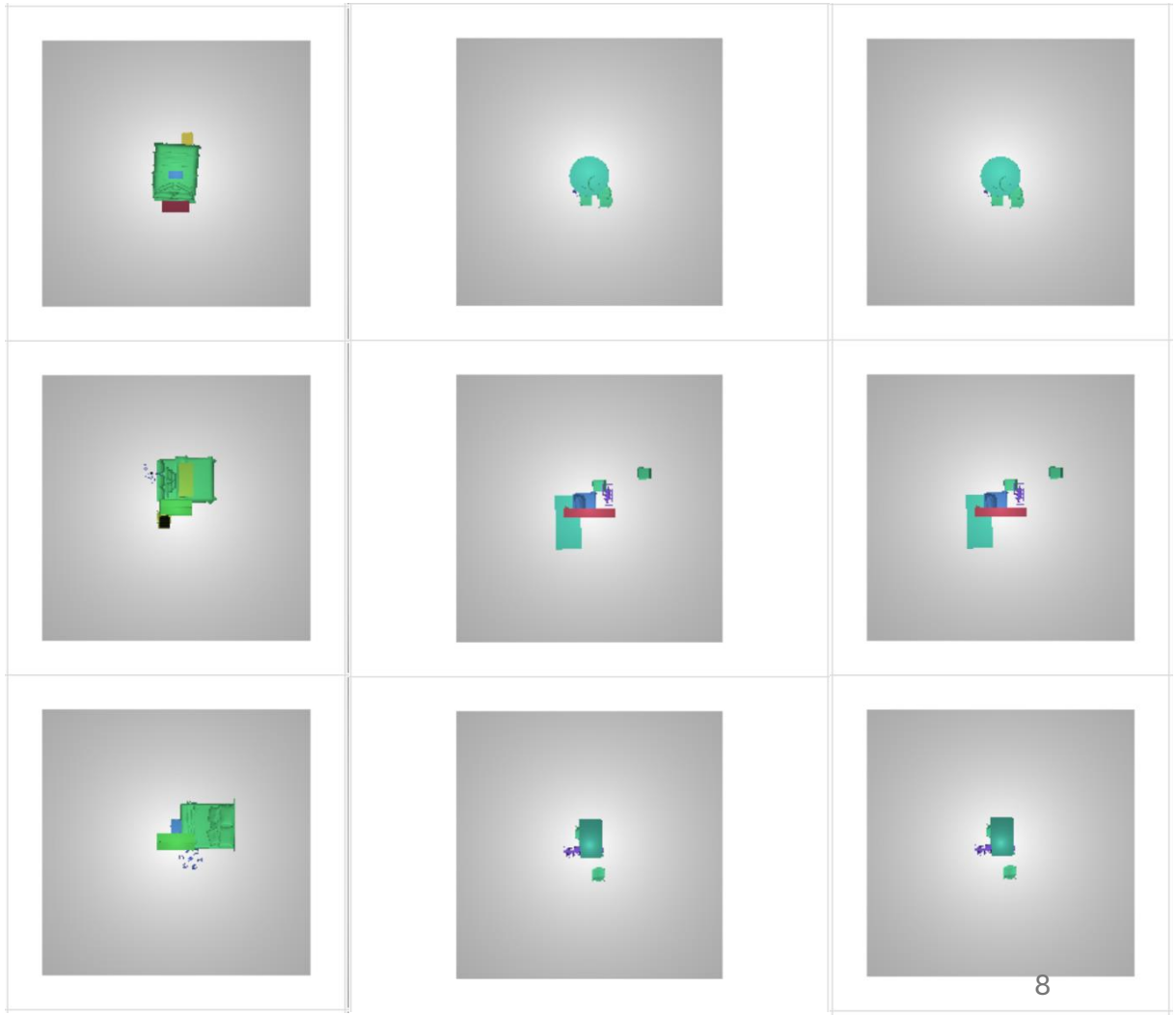
ATISS: Examples

Bedroom Living room Dining room

From Paper

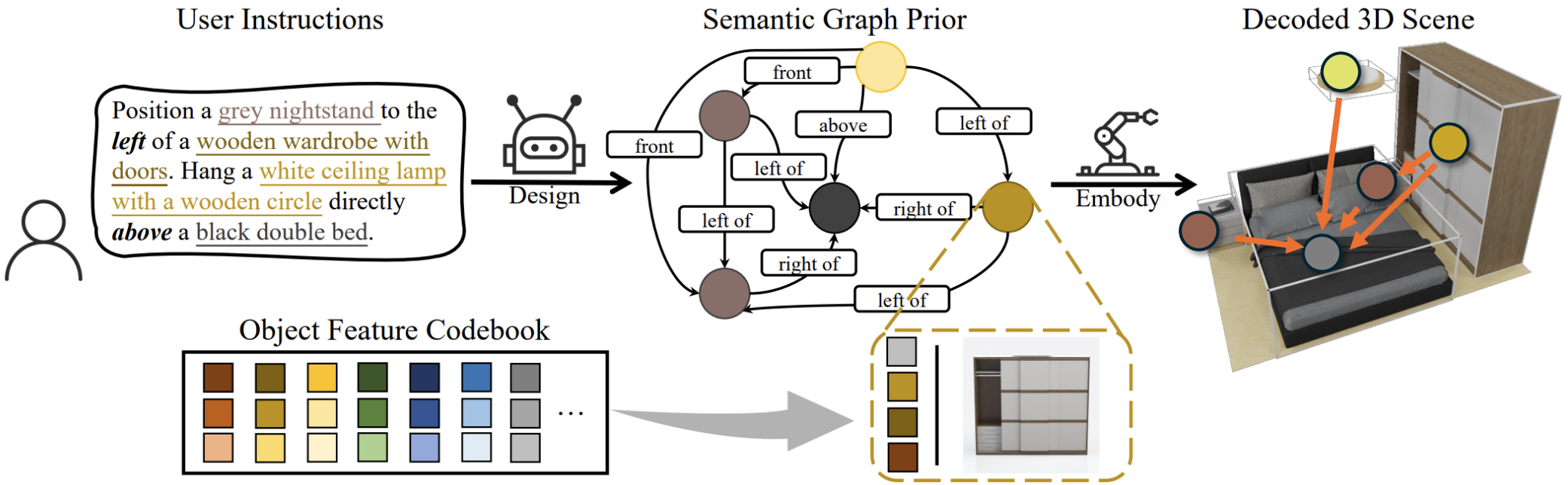
Bedroom Living room Dining room

Generated



Diffusion models can be used to generate scene graphs from text

- E.g., InstructScene, CommonScenes



InstructScene: Instruction-Driven 3D Indoor Scene Synthesis with Semantic Graph Prior
[Lin and Mu 2023]

LLM-based methods

Input

LLM / VLM

Intermediate Representation

Placement & Optimization (b, c, d)

Output

(a) **Direct generation**

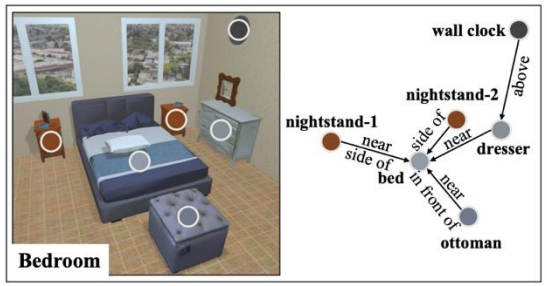
Structured Language

```
double_bed {
  length: 213px;
  width: 250px;
  height: 113px;
  left: 125px;
  ...
  orientation: 90°;
}
```

LayoutGPT [Feng et al. 2023]

(b) **Constraint guided / graph-structured**

Scene Graph



Holodeck [Yang et al. 2024]

(c) **Programmatic**

Structured Language

```
table.pos = [4.0, 2.5, table.size[2] / 2]
for chair in armchair.placements:
    solver.distance(chair, table)
    solver.point_towards(chair, table)
```

LayoutVLM [Sun et al. 2025]

(d) **Interactive**

Chat2Layout [Wang et al. 2024]

LLM to Structured Data

Method:

- Use LLM to generate JSON or CSS to represent a 3D scene

Advantages:

- Leverages LLM's broad knowledge for diverse outputs with a user-friendly natural language interface

Disadvantages:

- Limited visual understanding may lead to physically implausible or inconsistent outputs
- Rather limited scene types (bedroom, living room, dinning room)
- [LLplace] Does not support specifying relationships

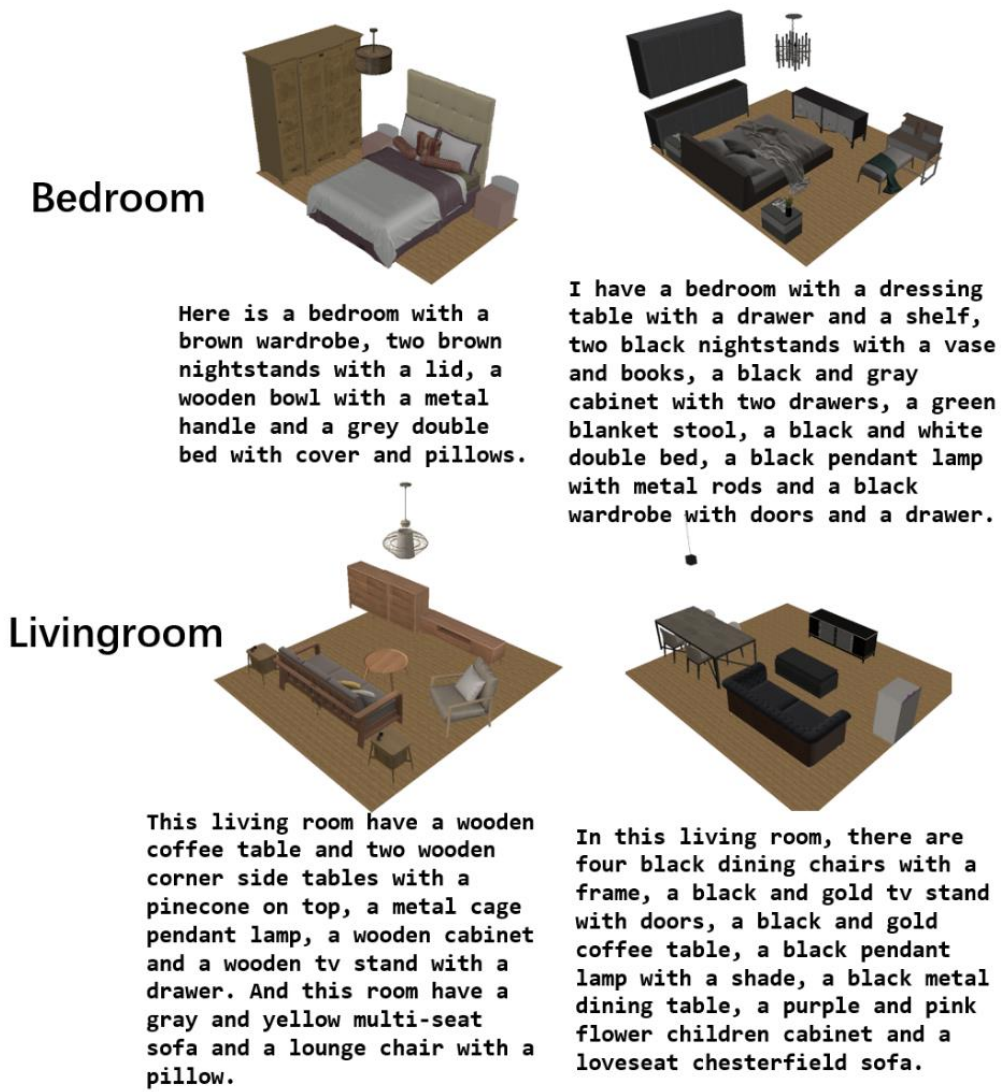
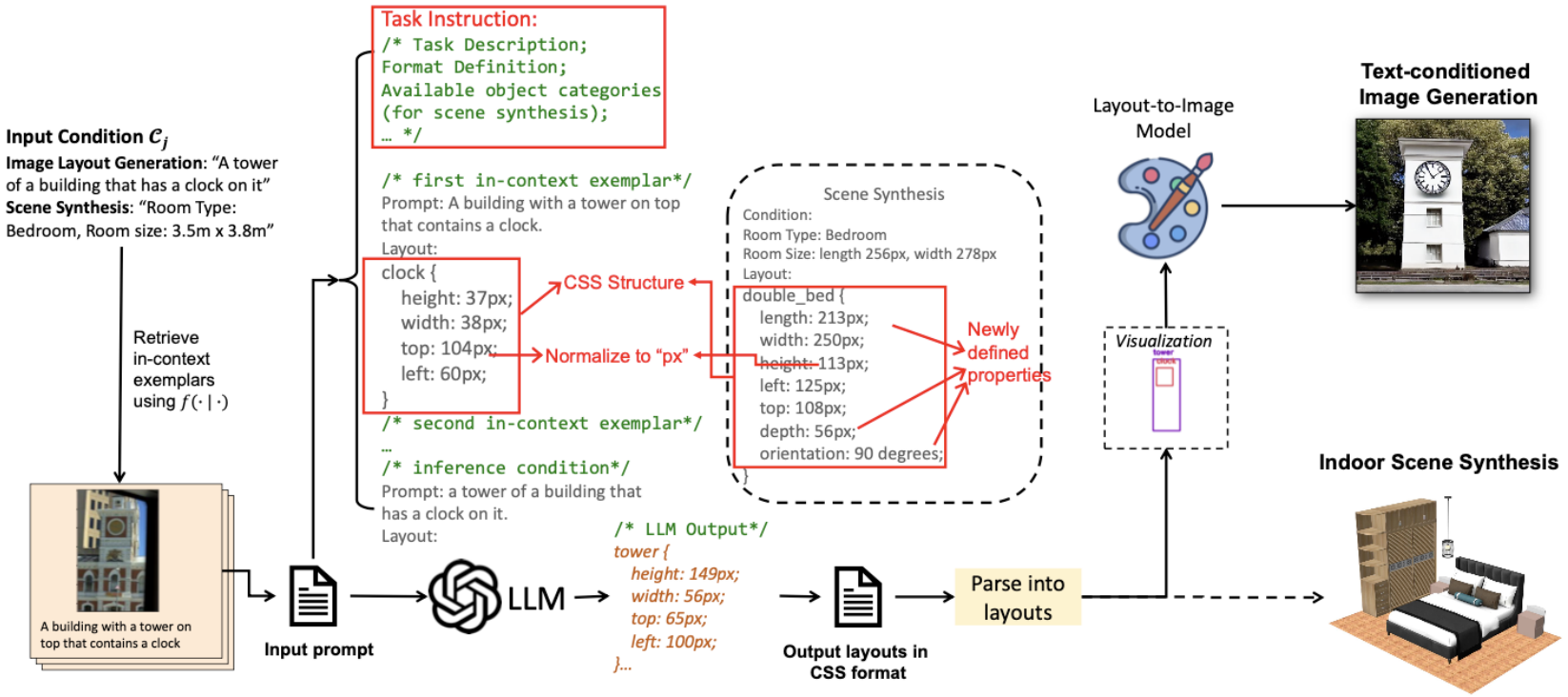


Figure from LLplace

Ex: LayoutGPT

- Tasks: image generation, scene generation
- Using LLM for scene layout planning, with CSS as the layout syntax



LayoutGPT: Compositional Visual Planning and Generation with Large Language Models
[Feng et al. 2023]

LLM with Specialized Modules

Method:
- Use multiple modules of specialized LLM to generate 3D scene

Advantages:
- Combines LLM's knowledge with domain-specific constraints and rules
- Can generate whole floor with multiple rooms

Disadvantages:
- Struggle with complex layout

Figure 1. Example outputs of HOLODECK—a large language model powered system, which can generate diverse types of environments (arcade, spa, museum), customize for styles (Victorian-style), and understand fine-grained requirements ("has a cat", "fan of Star Wars").

Figure from Holodeck

LLM to Programs

Method:

- Use LLM to generate executable programs to generate a 3D scene

Advantages:

- Highly flexible and expressive
- Able to generate more diverse and complex scenes
- Isolates parameters of layout for optimization

Disadvantages:

- Dependency on LLM's ability to generate correct, executable code
- Accurately representing and satisfying complex spatial relationships between objects can be challenging when working through code generation

Query: Three trees in a row besides a neighborhood

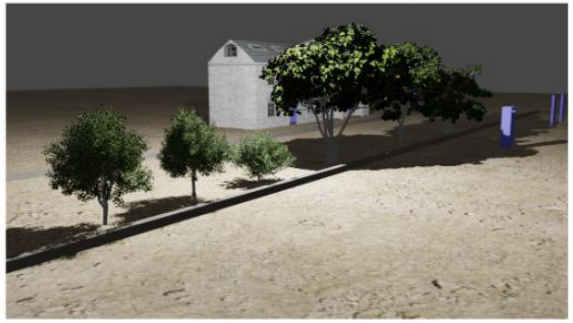
```
assets = [tree1, tree2, tree3, neighborhood]

# Alignment along x axis
x_align_score = alignment_score(assets, 'x')

# Perpendicularity between trees and neighborhood
perpendicularity_score(tree1, neighborhood)

# Parallelism between trees
tree_parallelism = parallelism_score([tree1, tree2, tree3])


constraints = [
    x_align_score, perpendicularity_score, tree_parallelism
]
```



Query: A new solar system with planets orbiting around a

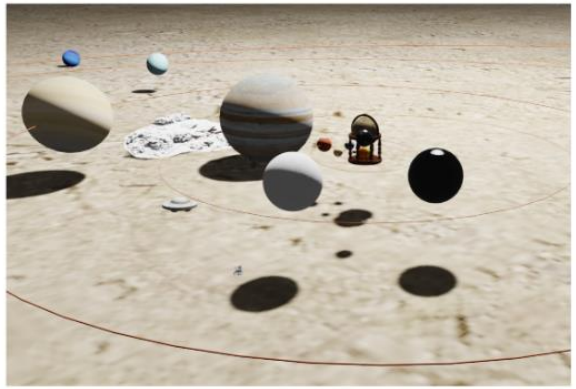
```
star_layout = import_obj("star.obj")[0]
planet_layouts = [import_obj(f"planet{i}.obj")[0] for i in range(1,4)]

assets = [star_layout] + planet_layouts

# Randomize planet positions
for planet in planet_layouts:
  r = random.uniform(10, 30)
  theta = random.uniform(0, 360) # angle in degrees
  x = r * math.cos(math.radians(theta))
  y = r * math.sin(math.radians(theta))
  planet.location = (x, y, 0)

# Orient planets towards star
for planet in planet_layouts:
  angle = vector_to_angle(planet.location, star_layout.location)
  planet.orientation = angle_to_euler(angle)

# Calculate rotational uniformity
uniformity = rotation_uniformity_score(planet_layouts, star_layout.location)
```



Query: Racoon is lying on a bed, listening to music,

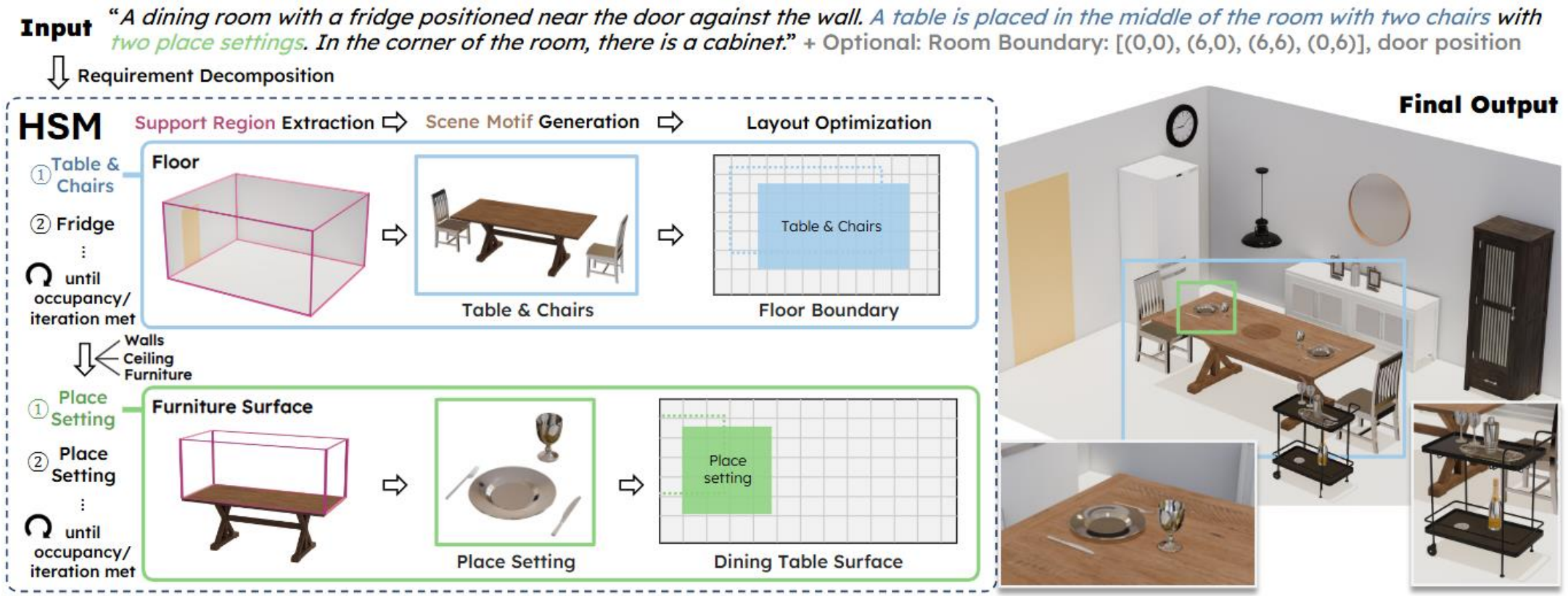
```
# Lie racoon down
racoon.orientation = (90, 0, 0)

# Initial headphone position
headphones.position = (racoon.position[0], racoon.position[1], racoon.position[2] + 1)

# Iteratively adjust headphones
proximity_threshold = 0.9
align_threshold = 0.9
for i in range(10):
  # Check proximity and alignment
  proximity_score = proximity_score(headphones, racoon, 0, 0.5)
  align_score = direction_score(headphones, racoon)
  # If constraints not met, adjust headphones
  if proximity_score < proximity_threshold or align_score < align_threshold:
    # Update position
    headphones.position = (
      racoon.position[0] + random.uniform(-0.1, 0.1),
      racoon.position[1] + random.uniform(-0.1, 0.1),
      racoon.position[2] + random.uniform(0.8, 1.2)
    )
```

Figure from SceneCraft

HSM leverages recurring “motifs” to arrange objects at all scales



HSM: Hierarchical Scene Motifs for Multi-Scale Indoor Scene Generation
[Pun et al. 2025]

HSM leverages recurring “motifs” to arrange objects at all scales

HSM: Hierarchical Scene Motifs for Multi-Scale Indoor Scene Generation
[Pun et al. 2025]

Interactive Scene Design with LLM-in-the-Loop

Advantages:

- User retains control over each modification
- Progressive refinement avoids regenerating from scratch

Disadvantages:

- Requires multiple interaction rounds — slower workflow
- Suggestion quality depends on LLM's spatial understanding

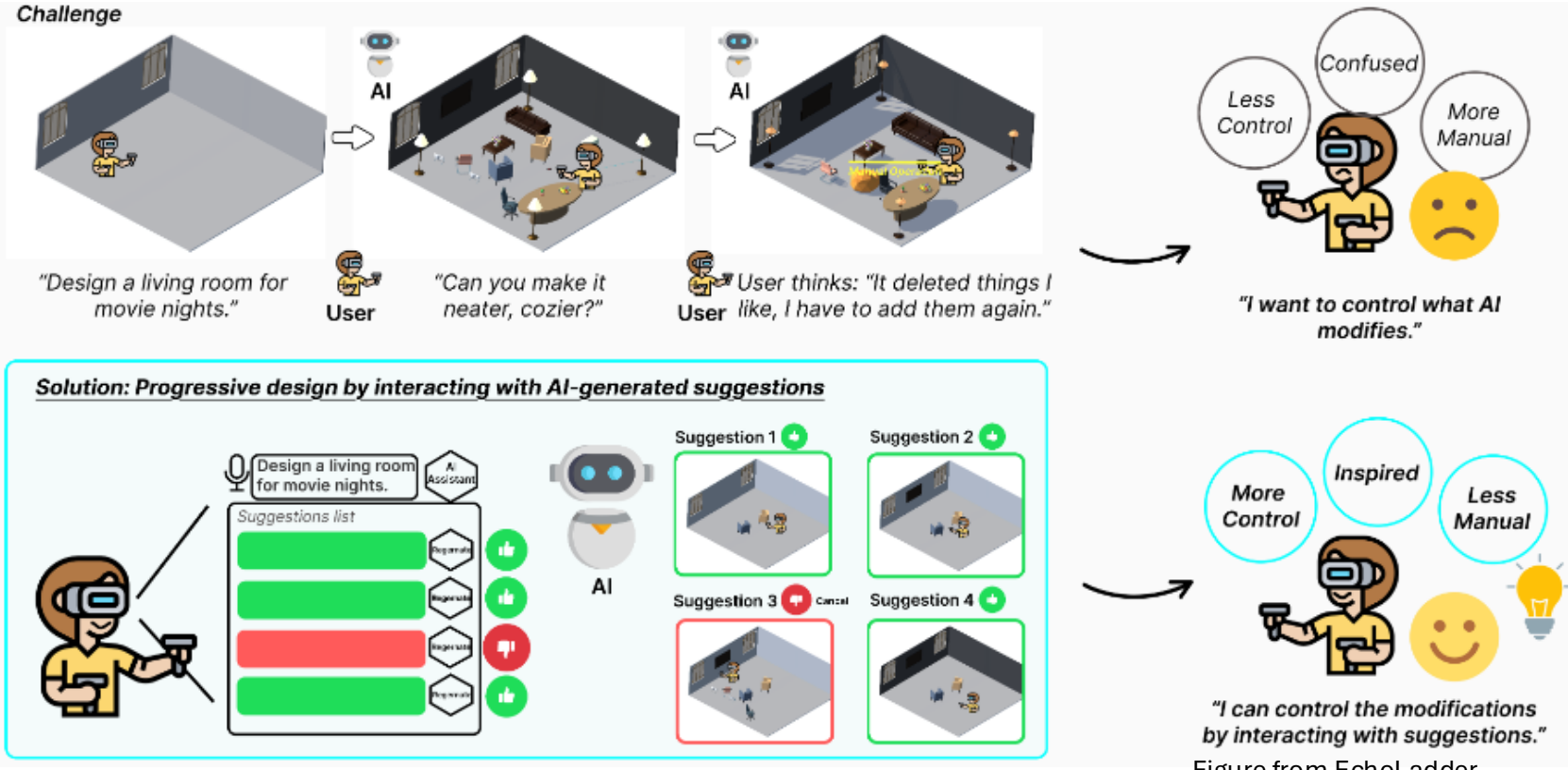


Figure from EchoLadder

LLM + image generation → 3D scene

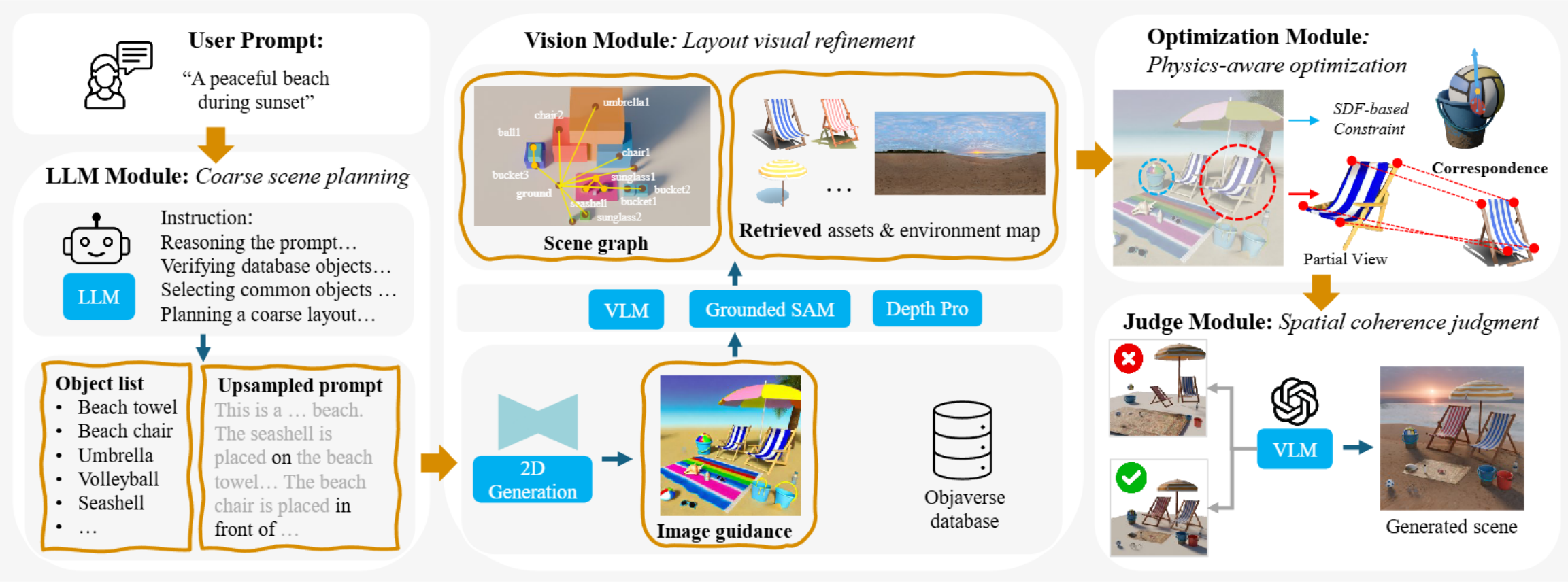


Figure from Scenethesis

Learn More!

SceneMotifCoder

HSM