Journal of Design Intelligence
https://www.ffspub.com/index.php/jdi
ISSN 3104-9001 (print)
E-mail: jdi.office@ffspub.com

Review

Language Models as 3D Layout Designers: a Survey

Cheng Wan[1], Yongsen Mao[2], Yuan Liu[2,†],
[1]*School of Statistics, Renmin University of China, Beijing 100872, China*
[2]*Division of Integrative Systems and Design, Hong Kong University of Science and Technology, Hong Kong, China*
[†]*E-mail: yuanly@ust.hk*

Received: November 10, 2025 / Revised: January 21, 2026 / Accepted: January 221, 2026 / Published online: February 24, 2026

Abstract: Scene design—spanning interior arrangement, virtual environment creation, and simulated asset generation for robotics—requires semantic understanding, physical plausibility, and aesthetic judgement. While traditional approaches rely on expert rules or optimization, Large Language Models (LLMs) promise more flexible, language-grounded reasoning and multimodal grounding. This survey provides a focused definition and task framework for layout agents and proposes a hierarchical categorization based on LLM layout agents' output representation. At the highest level, we distinguish Direct Methods, which predict asset poses (one-stage vs. multi-stage), from Indirect Methods, which emit intermediate representations (constraint-based or programmatic) that are converted to layouts via post-processing. Furthermore, we categorize the evaluation of layout agents into four dimensions: physical plausibility, prompt fidelity, semantic coherence, and aesthetics and realism, and summarize the commonly used evaluation methods. Finally, we review the current challenges faced by layout agents, including the lack of spatial reasoning ability, instability, poor generalization, and low efficiency. By organizing recent progress and identifying gaps, this survey aims to guide future research toward more capable, generalizable LLM layout agents.

Keywords: Layout Design; Large Language Model; LLM Agent
https://doi.org/10.64509/jdi.11.51

1 Introduction

3D scene design has broad applications across various domains, including interior and furniture layout design [1, 2], virtual environment [3] and game scene creation [4], and the construction of simulated assets for robotic training [5]. Designing a coherent and functional scene is an inherently complex process that involves multiple stages—selecting appropriate assets from a library, arranging them spatially within the environment, ensuring geometric and semantic consistency, and maintaining aesthetic and functional balance. A well-designed scene requires a combination of semantic coherence, physical plausibility, visual aesthetics, and functional usability. In the past, layout design has relied heavily on expert knowledge [6], manual adjustment [7], or handcrafted procedural rules and complex optimization-based algorithms [8]. These traditional approaches, while precise, are often labor-intensive, computationally expensive, and lack generalization across diverse scene types and styles [9].

The advent of sophisticated Artificial Intelligence is rapidly transforming this landscape [10]. An ideal AI layout designer should be capable of independently accomplishing every stage of the scene design process—from understanding abstract textual requirements [11], selecting appropriate assets [12], and arranging them spatially with semantic and physical consistency [13], to ensuring overall functional balance and aesthetic harmony within the scene. At present, several approaches can handle certain sub-tasks within this pipeline [14]. For example, rule-based methods [8] can position predefined objects according to manually specified constraints or heuristics, while generative model-based approaches [15] reformulate scene design as a generation problem, transforming natural language descriptions into 3D representations [16], rendered images [17], or even videos [18] that depict the intended environment. Despite the remarkable progress made by these AI-based scene design methods, they still face substantial limitations [14]. Most existing systems lack the holistic understanding and cross-stage reasoning ability required for coherent end-to-end design [19]. Furthermore, they often struggle to balance semantic accuracy, spatial realism, and asset-level constraints, especially when tasked with arranging fixed, real-world assets rather than generating objects freely [20].

† Corresponding author: Yuan Liu
*Academic Editor: Yan Hong

The emergence of Large Language Models (LLMs) [21, 22] and Vision Language Models (VLMs) [23] has opened up new possibilities for developing a truly universal AI layout designer. LLMs, such as GPT [24] and LLaMA [25] families, are trained on massive text corpora and exhibit remarkable abilities in semantic understanding, reasoning, and instruction following. They can interpret complex design requirements expressed in natural language, maintain contextual coherence, and generate structured, human-like responses—capabilities that are essential for high-level scene planning [26]. Building upon this foundation, VLMs extend these abilities to visual understanding and reasoning [27]. By jointly modeling textual and visual modalities, these models can analyze images, ground language in spatial layouts, and generate or modify visual content in accordance with linguistic instructions [28]. This multimodal reasoning capability makes them particularly promising for scene design, where tasks inherently involve interpreting spatial relationships, object affordances, and aesthetic composition. Therefore, the rise of LLMs and VLMs provides a powerful foundation for constructing an all-around "layout agent" [29–31]—one that can seamlessly connect linguistic intent with spatial imagination, and ultimately translate human instructions into coherent, physically realistic, and functionally meaningful 3D scenes.

In the past two years, research on LLM layout agents has grown rapidly. A review of publications from 2023 to 2025 shows a sharp upward trend in the number of papers related to "large language models," "3D scene layout," "object placement," and "LLM agent.", as is depicted in Figure 1. However, despite this rapid progress, a comprehensive survey dedicated to LLM-based layout agents remains absent. Most existing reviews [14, 19, 32] only briefly mention the emerging role of language models and lack a systematic comparison of the tasks, designs, and evaluations of LLMs' layout agents.

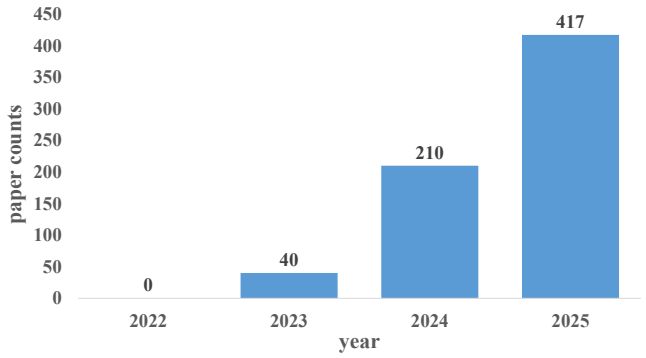

Figure 1: The number of papers on Google Scholar published between 2023 and 2025 that contain the keywords "large language models," "3D scene layout," "object placement," and "LLM agent", as of November 5, 2025.

Accordingly, this survey provides a focused and systematic review of recent progress in LLM layout agents, consolidating these fragmented studies, summarizing recent advances, and identifying open challenges in this evolving research direction. Our study is, to the best of our knowledge, the first to comprehensively analyze layout agents driven by LLMs and VLMs. We not only summarize the key research efforts in this emerging field but also propose a unified framework for understanding, categorizing, and evaluating these agents. We first provide a framework to define the tasks of layout agents. Then, based on the different forms of outputs, existing methods are distinguished between **Direct Methods** and **Indirect Methods** at the highest level: Direct methods generate asset layouts by directly predicting the positions and orientations of assets. These can be further divided into **One-stage Approaches**, which output all object poses in a single pass, and **Multi-Stage Approaches**, which iteratively refine placements through multiple reasoning steps or dialogues. Indirect methods, in contrast, do not directly output coordinates. Instead, they first generate intermediate representations—such as constraint sets or domain-specific programs—which are later translated into spatial layouts through post-optimization or external solvers. Based on this distinction, we further divide indirect methods into **Constraint-based** and **Programmetic** categories. This categorization provides a comprehensive and unified organizational framework for current LLM layout agent research. Besides the above categorization, we also systematically review the evaluation criteria used in existing works, summarizing how researchers assess the quality of generated layouts. Finally, we analyze the key limitations and open challenges that persist in this field, including 3D reasoning reliability, stability, efficiency, and generalization. By consolidating current knowledge and identifying these gaps, our survey aims to provide both a structured overview and a forward-looking roadmap for future research on LLM Layout Agents.

Our contribution can be summarized as follows:

- We present the first comprehensive survey dedicated to the field of LLM Layout Agents. We systematically review the recent progress in this emerging research area and provide a clear definition of the layout agent task, outlining its objectives and key components.
- We categorize existing layout agents according to the form of their output representations. This classification framework distinguishes between direct and indirect methods, further dividing them into subtypes, and offers a unified perspective that helps to better understand and compare current approaches.
- We conduct a comprehensive analysis of the evaluation metrics used in LLM layout agent research, highlighting the diversity and limitations of existing evaluation paradigms.
- We identify and discuss the key challenges and open problems that remain in this field, including a lack of spatial reasoning ability, stability, efficiency, and generalization. Our analysis provides insights and guidance for future work, aiming to promote the development of more capable and generalizable LLM layout agents.

The remainder of this paper is organized as follows. In Section 2, we present the preliminary background, introducing the foundations of AI-based layout generation, as well as key concepts related to LLMs, VLMs, and LLM Agents. Section 3 defines the layout generation problem, where we propose a general task framework and clarify the major subtasks involved in layout design. In Section 4, we provide a comprehensive categorization of current LLM layout agents based on their output forms and reasoning strategies. Then, Section 5 reviews and compares the evaluation methods used

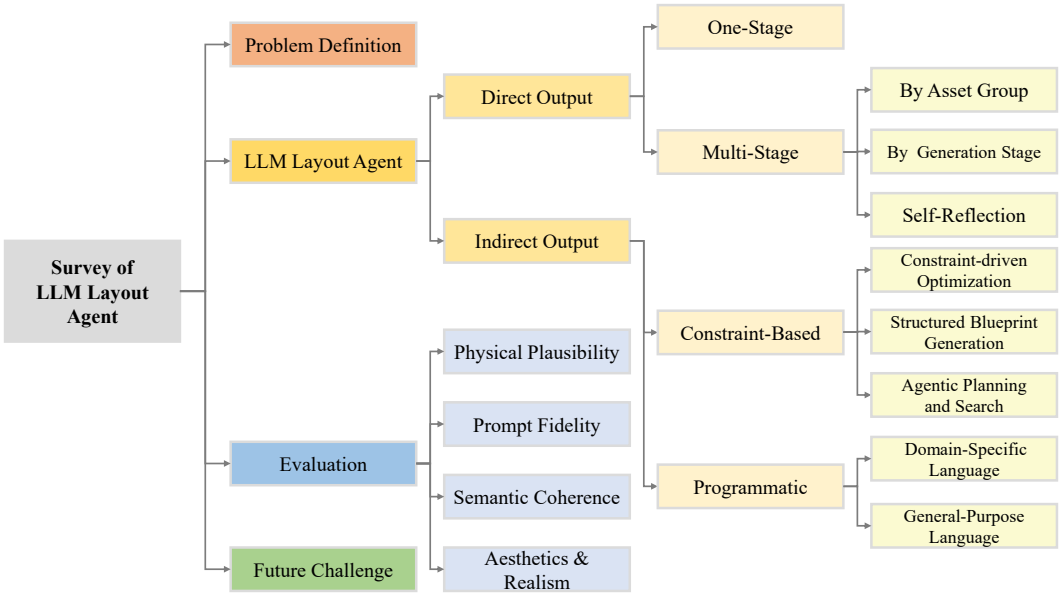

Figure 2: The logical framework of this survey

to assess layout agents. Section 6 discusses the existing limitations and open challenges faced by current approaches, offering insights into potential directions for future research, followed by our conclusion Section 7. An outline of our research is presented in Figure 2.

2 Preliminary

In this section, we introduce the background knowledge required for this survey, including layout generation, large language models, vision language models, and LLM agents.

2.1 Layout Generation

Layout generation refers to the task of automatically producing a spatial arrangement of objects or assets within a scene [14]. Given an input—such as a textual description, a rough sketch, or a set of constraints—a layout generation system must identify or synthesize scene elements and determine their poses (positions, orientations) and scales [12] so that the final configuration fulfills the design intent while respecting spatial and physical constraints (e.g., non-collision, support relationships, and functional adjacencies) [33].

Following [14], scene generation approaches can be grouped into four broad paradigms, distinguished primarily by their internal representations and generative procedures. Procedural Generation [8] constructs layouts by applying explicit, rule-based algorithms or stochastic grammars that encode domain knowledge and spatial priors. Procedural systems generate layouts via deterministic or parameterized pipelines [9] that enforce constraints and produce repeatable structure. These methods are highly controllable but typically require manual rule design and may not capture the full variability of natural scenes [14]. The remaining paradigms rely on learned generative models [13, 34] operating over different data modalities. Neural 3D-based Generation represents scenes as neural 3D representations, e.g., NeRF [16] and 3DGS [35]. Methods in this category offer richer spatial fidelity [36] but generally demand substantial 3D supervision and heavier computation [37]. Besides neural 3D-based generation, Image-based methods frame layout as a problem in the two-dimensional visual domain, producing images or top-views that encode object presence and spatial relationships [38]; these 2D outputs can then be interpreted or lifted into 3D [39]. This paradigm leverages powerful 2D visual priors and excels in visual plausibility [17], though its implicit treatment of depth and 3D structure can complicate accurate spatial grounding [11]. Finally, as an extension to images, video-based generation models [18, 40] scenes as dynamic sequences, emphasizing temporal and multi-view coherence by generating frame sequences or view-consistent representations over time [41]. Such approaches are well suited to tasks requiring motion or changing viewpoints, but they introduce additional challenges in maintaining both spatial consistency across views and temporal continuity, often at increased representational and computational cost [42].

2.2 LLMs and VLMs

Large Language Models (LLMs) [24, 25, 43, 44] are neural networks pretrained on very large text corpora to model conditional text generation and high-level linguistic reasoning. The modern LLM paradigm is built on the Transformer architecture [21], which uses self-attention to produce scalable, parallelizable sequence representations; scaling model size and pretraining compute has been a dominant driver of capability gains (e.g., in-context learning and few-shot transfer [22]). The mainstream training pipeline typically involves large-scale unsupervised pretraining [45] followed by various forms of alignment or task specialization, including reinforcement learning and supervised fine-tuning [46] to improve utility and safety.

As an extension to LLM, Vision-Language Models (VLMs) [28, 47] connects visual encoders to language models so the system can accept images or video alongside text. Architecturally, common design patterns include (1) a frozen or finetuned visual encoder that extracts visual tokens or

features [48], (2) a lightweight cross-modal adapter, like a Q-former [49] or projection module [28] that maps visual features into the LLM token space, and (3) instruction or multimodal finetuning that teaches the combined model to follow visual-language prompts [28]. Other approaches train a unified multimodal transformer end-to-end [24, 50]. These pathways trade off compute, parameter efficiency, and the ability to leverage pretrained unimodal models.

Several algorithmic techniques are commonly applied to improve comprehension and alignment in both LLMs and VLMs. First, Reinforcement Learning from Human Feedback (RLHF) [46] is widely used to align model outputs with human preferences by training a reward model and applying RL optimization to the language policy. Another frequently used technique is Chain-of-Thought (CoT) [51], which elicits intermediate reasoning steps from LLMs and can substantially improve performance on compositional tasks. In multimodal systems, visual instruction tuning, i.e., generating or collecting image-text instruction pairs and finetuning [28] has become an effective way to teach grounded, instruction-following behavior.

Finally, by combining image understanding with geometric or embodied signal modalities, VLMs are increasingly applied to spatial and 3D reasoning tasks [52–55]. Embodied multimodal models incorporate continuous sensor or pose inputs to ground language in real-world geometry and actions, enabling planning and robot control [56]. More recent efforts explicitly augment VLMs with large-scale 3D spatial supervision or synthetic 3D-aware VQA data to improve metric spatial reasoning and view-consistent understanding [57], which is an important step toward layout and scene reasoning for design and robotics.

2.3 LLM agents

An LLM agent [58–60] is a goal-driven system that embeds a large language model within an interactive loop to perform complex, multi-step tasks. The workflow of an LLM agent involves perceiving inputs, reasoning about goals, issuing action plans or tool calls, executing those actions through external modules, observing outcomes, and iteratively refining behavior. Typical agent architectures combine an LLM-based planner that decomposes high-level objectives into subgoals, perception adapters [28] that convert images or 3D data into semantically grounded representations, tool and executor modules (e.g., search, simulators, geometry solvers, renderers)[29, 61] that realize concrete operations, and stateful memory and verification components [59] that record progress and check feasibility; a refiner module subsequently post-processes outputs [62] before finalization. Agents may operate in one-shot or iterative modes [30], vary in autonomy and human-in-the-loop interaction, and trade off controllability against autonomy depending on task requirements. When applied to layout design, the agentic paradigm reframes synthesis as planning and tool-assisted optimization—enabling compositional workflows and precise numeric control but also raising challenges in spatial grounding, tool integration, and reliable multi-step evaluation [63]. This survey focuses on such agentic approaches for layout and scene generation, emphasizing the interface between language reasoning, multimodal perception, and geometric tooling.

3 Problem Formulation

Before introducing LLM layout agents, We first provide a complete formal definition of the layout agent task. A layout agent can be viewed as a function Agent$(\cdot)$ that recieves the current scene information Scene, the user requirement U, and a set of N asset information entries $\{A_i\}_{i=1}^{N}$. It then outputs n pairs representing the selected assets A_j and their spatial attributes $p_j := (O_j, P_j, S_j)$:

$$(A_j, p_j)_{j=1}^{n} = \text{Agent}\big(\text{SCENE}, U, (A_i)_{i=1}^{N}\big), \quad N \geq n. \tag{1}$$

In this formulation Equation 1:

- Each asset information entry A_i may consist of a textual description, an image, a point cloud or mesh representation, or a combination of these modalities. Such information can be directly provided by the user (e.g., uploading a photo or describing an asset's material) or retrieved by an external retriever.
- Similarly, the current scene information SCENE can be represented as a textual or visual description, an image, or a point cloud; the user requirement U may also be provided as either text or image.
- Ideally, O_j, P_j, and S_j are 3D vectors corresponding to the orientation, position, and size of the j-th object.

Figure 3 offers an illustration of Equation 1. Under this unified definition, many existing layout-agent tasks can be described within the same framework:

- **Asset selection:** when $n < N$, the agent must select the most appropriate assets for the given scene.
- **Placement planning:** SCENE denotes an empty scene (e.g., a room's top-down view), and U is the design goal or scene requirement (e.g., "Please design a living room").
- **Asset addition:** SCENE represents a partially populated scene, and user prompt U specifies that A is the single asset to be inserted into the scene.
- **Asset removal:** the agent identifies, according to U, which assets in $\{A_i\}_{i=1}^{N}$ should be removed.

It is worth noting that, in practice, most current layout agents implement only a subset of this full task. For instance, many models cannot output complete SE(3) bounding-box information, providing only partial pose parameters such as asset position and yaw rotation. Besides, most agents focus on one task type (e.g., placement planning) without jointly supporting asset addition, deletion, or selection within a unified framework.

4 LLM Layout Agent

In this section, we introduce the existing LLM layout agent methods. We first categorize these models into direct and indirect methods according to their different forms of output. The two types of methods are then divided into categorizations of

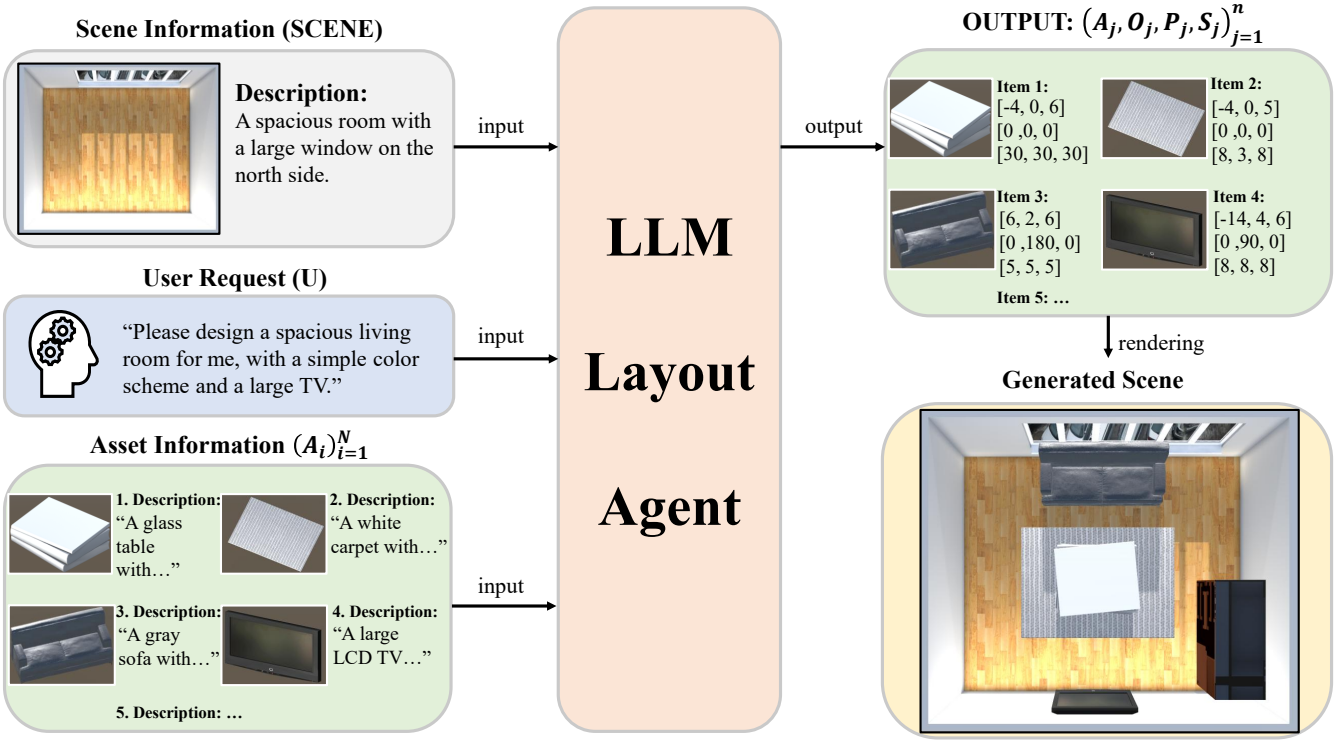

Figure 3: A task framework of LLM layout agents.

finer grains. In Table 1, we give a summary for existing LLM layout agents.

4.1 Direct Method

In Equation 1, the core objective of a layout agent lies in determining the spatial poses $p_j = (O_j, P_j, S_j)$ of target assets within a given scene. The most straightforward approach to achieving this is to have the language model directly output the pose information of each asset, including its position, orientation, and size. We refer to such approaches as Direct Methods. Depending on the granularity of prediction, Direct Methods can be further divided into two subtypes: One-stage and Multi-Stage methods.

4.1.1 One-Stage Generation

One-stage methods represent the most straightforward approach in LLM layout agent design, where the model is prompted to generate all asset poses in a single inference step. LayoutGPT [26] demonstrated that LLMs can directly output spatial layouts for indoor scenes from textual instructions and enhances consistency in scene generation. Besides, LLplace [64] extended the approach to 3D indoor scene layout generation and editing, enabling users to interactively refine layouts through natural language. The one-stage paradigm has also been adapted to large-scale environments, such as in CityCraft [65], which applies direct LLM-based generation to 3D city layouts, showcasing its scalability and flexibility in complex spatial planning tasks.

To improve the accuracy and plausibility of one-stage outputs, several works have enhanced the LLM's spatial reasoning capabilities. SKE-Layout [66] leverages retrieval-augmented generation to inject spatial knowledge into the LLM, allowing it to better predict realistic coordinates and orientations. OptiScene [67] introduces a large-scale, human-aligned dataset for training, boosting the quality of generated layouts through supervised learning. Meanwhile, Cot2Scene [68] employs CoT prompting to guide the LLM through step-by-step spatial reasoning, resulting in more coherent and semantically meaningful scene compositions.

As the most straightforward approach, one-stage methods face several limitations, such as difficulty in handling complex constraints, limited fine-grained control, and potential inconsistencies in multi-object arrangements. However, their clear and interpretable outputs make them highly suitable as modular components in more sophisticated layout pipelines. For instance, LayoutGPT [26] can provide a strong initial layout prior and improve the asset count accuracy in diffusion-based layout image generation. Similarly, Sceneteller [69], GALA3D [70], and HOLODECK2.0 [39] have adopted one-stage layout agents as a first-stage planner to enhance the realism and coherence of their final 3D scene generations, demonstrating the enduring value of this approach in both research and practical applications.

4.1.2 Multi-Stage Generation

The core limitation of one-stage methods lies in the insufficient capability of the LLM backbone to produce comprehensive and well-reasoned placement plans for all assets. To address this issue, multi-stage approaches decompose layout design into multiple sub-planning problems, solving them either through the collaboration of multiple agents or by invoking a single agent multiple times. This hierarchical strategy enhances the agent's overall planning capability. We classify multi-stage approaches into two main paradigms: **task decomposition** and **self-reflection**. Moreover, task decomposition can be further divided into two groups. A comparison between different paradigms are presented in Figure 4.

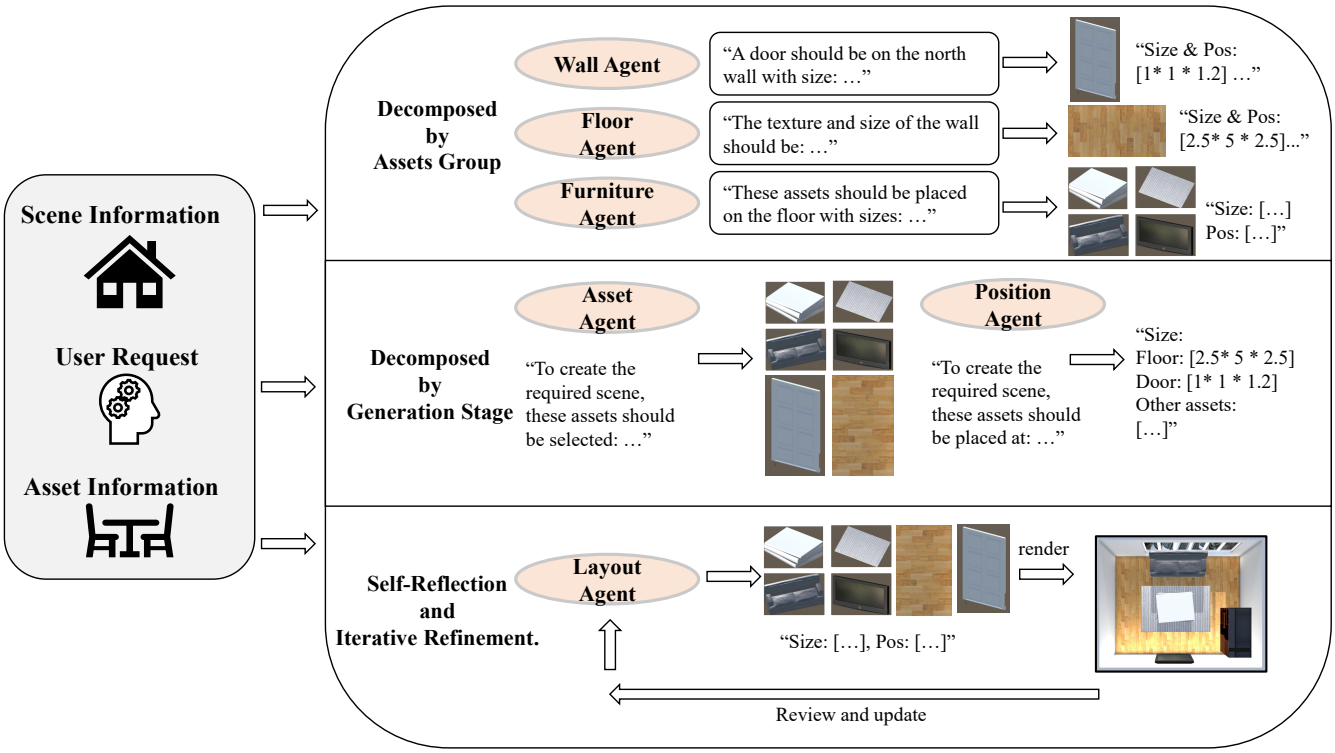

Figure 4: A comparison between three types of multi-stages methods.

Task Decomposition

The first paradigm, **task decomposition**, divides the overall layout problem into smaller, specialized sub-tasks. This can be performed in several ways:

Decomposition by Asset Group: This approach partitions the set of assets to be placed, assigning different groups to specialized agents. Following Equation 1, this approach can be formally represented as a sequence of generation steps where k agents $\text{Agent}_1, \text{Agent}_2, \cdots \text{Agent}_k$ handle disjoint subsets of the final assets:

$$\begin{aligned}(A_j, p_j)_{j=1}^{n_1} &= \text{Agent}_1\left(\text{SCENE}, U, \{A_i\}_{i=1}^{N}\right),\\ (A_j, p_j)_{j=n_1+1}^{n_2} &= \text{Agent}_2\left(\text{SCENE}, U, \{A_i\}_{i=1}^{N}\right),\\ &\vdots\\ (A_j, p_j)_{j=n_{k-1}+1}^{n} &= \text{Agent}_k\left(\text{SCENE}, U, \{A_i\}_{i=1}^{N}\right),\end{aligned} \tag{2}$$

and $n_1 < n_2 < \cdots < n_{k-1} < n$ denotes asset subset division. This method is exemplified by systems like Holodeck [71], which breaks down the creation of an environment into distinct modules for floors, doors, windows, and furniture. Each module is responsible for placing its designated category of assets, demonstrating a clear division of the task by asset type. Similarly, Zeng et al. [72] propose a method which operates on this principle by assigning different agents or processes to handle distinct components of a home, such as structural elements versus interior furnishings.

Decomposition by Generation Stage: Alternatively, the layout design task can be split into distinct functional stages. In this pipeline, different agents are responsible for different phases of the generation process. A typical decomposition strategy is to have one agent first select the required assets from the asset library, and then have another agent determine the poses of those assets. According to the definition in Equation 1, this approach can be formulated as:

$$\begin{aligned}\{A'_j\}_{j=1}^{n} &= \text{AssetAgent}\left(\text{SCENE}, U, \{A_i\}_{i=1}^{N}\right),\\ (p_j)_{j=1}^{n} &= \text{PoseAgent}\left(\text{SCENE}, U, \{A'_j\}_{j=1}^{n}\right).\end{aligned} \tag{3}$$

In this process, an "Asset Agent" first selects proper n assets $\{A'_j\}_{j=1}^{n}$ out of the given N assets, and a "Pose Agent" then decides the poses of each asset. Through this process, the second agent only needs to determine the poses of a subset of assets, thereby reducing its overall burden. It is worth noting that Equation 3 is not the only possible classification scheme; depending on the specific task, there exist other decomposition strategies with their own distinctive characteristics. An example is StageDesigner [73], which uses a three-stage pipeline for generating theater scenes from scripts: script analysis, foreground object generation, and background generation. Each module performs a distinct function in sequence, with the output of one stage feeding into the next. DirectLayout [74] also follows this paradigm by breaking the generation process into three distinct steps, sequentially handling different aspects of the layout problem. A similar staged approach is also adopted by Liu et al. [75], and the process is broken into phases like initial terrain sculpting, followed by the procedural placement of features.

Self-Reflection and Iterative Refinement.

The second major paradigm is **self-reflection**. In this approach, after an agent generates an initial output, the resulting scene is fed back into the same agent for evaluation and modification. This creates a feedback loop, allowing the agent to iteratively improve its own solution based on the state of the

scene. Following Equation 1, this process can be formally expressed as:

$$\begin{aligned}(A_j^{(1)}, p_j^{(1)})_{j=1}^{m_1} &= \text{Agent}\big(\text{SCENE}, U, \{A_i\}_{i=1}^{N}\big),\\ (A_j^{(2)}, p_j^{(2)})_{j=1}^{m_2} &= \text{Agent}\big(\text{SCENE}^{(1)}, U, \{A_i\}_{i=1}^{N}\big),\\ &\vdots\\ (A_j^{(k)}, p_j^{(k)})_{j=1}^{m_k} &= \text{Agent}\big(\text{SCENE}^{(k-1)}, U, \{A_i\}_{i=1}^{N}\big),\end{aligned} \tag{4}$$

where $\text{SCENE}^{(p)}$ represents the scene information after the p-th iteration. SceneWeaver [76] embodies this concept with a self-reflective agent that critiques its own placements and makes revisions to improve quality. Another framework proposed by Liu et al. [30] leverages a VLM to "see" and reflect upon the generated scene, enabling iterative corrections. Similarly, DisCo-Layout [77] uses a multi-agent framework where semantic and physical refinement agents work together, critiquing and correcting the layout in a loop until a satisfactory result is achieved.

4.2 Indirect Method

While direct methods offer a straightforward approach by generating assets' numerical coordinates, they are often fraught with challenges. LLMs, at the core of layout agents, are text processors; tasking them with producing long vectors of precise floating-point numbers can lead to issues such as format violations, numerical instability, and hallucinated values that do not respect geometric or physical realities. To address these limitations, Indirect Methods propose a paradigm shift. Instead of requiring the LLM to function as a calculator that outputs final pose coordinates, indirect approach generate an intermediate, symbolic representation of the scene, which is then translated into a final layout by a separate, deterministic execution engine. This intermediate representation abstracts away the need for low-level numerical precision, allowing the LLM to focus on logic, relationships, and composition.

Broadly speaking, indirect methods can be broadly classified into two main categories, Constraint-Based Methods and Programming Language-Based Methods, based on the nature of the intermediate representation they produce. The comparison between two methods is presented in Figure 5. We next discuss these two kinds of methods in detail.

4.2.1 Constraint-Based Methods

In this paradigm, the LLM generates a logical, relational, or physical intermediate representation that abstracts the problem away from low-level coordinates. For instance, instead of outputting exact coordinates, the LLM might produce a set of symbolic rules such as: "*chair1* should face *desk1*," " *book1* must be placed on top of *desk1*," and "the distance between *chair1* and *desk1* should be less than 1 meter." These relationship constitutes a scene graph, which is then consumed by a deterministic backend to compute the final geometric layout. A classic approach is to formulate the task as an optimization problem to find the set of all object poses $P := \{p_j\}_{j=1}^{n}$ that minimizes a total energy function $E(P)$ [33, 78]. Commonly, the energy function is a weighted sum of individual loss terms, where each term L_j corresponds to a specific constraint:

$$P^* = \arg\min_P E(P) \quad \text{where} \quad E(P) = \sum_j w_j L_j(P). \tag{5}$$

For example, a distance constraint (e.g., "chair near desk") can be encoded as a hinge loss $L_{\text{dist}} = \max(0, \text{dist}(p_{\text{chair}}, p_{\text{desk}}) - d_{\max})$, which penalizes distance only when it exceeds a threshold. Similarly, a support constraint L_{support} penalizes misalignment of contact surfaces, and a collision loss $L_{\text{collision}}$ penalizes volumetric intersection. A numerical optimizer then adjusts the object poses to minimize this total energy.

By decoupling LLM's reasoning from calculation, indirect methods offer several key advantages. First, it allows the LLM to operate in its native symbolic domain of language and logic, where it excels, while offloading the complex, continuous-space geometric optimization to specialized and reliable algorithms. Furthermore, the intermediate constraints are human-readable, making the agent's "thought process" transparent. This allows for easier debugging, user editing of the plan before execution, and more predictable control over the final output. Furthermore, based on the *procedural role* this representation plays, the constraint-based methods can be further categorized into three approaches:

Constraint-driven Optimization: In this primary approach, the LLM's role is to produce a set of explicit rules or conditions that the final scene must satisfy. A separate optimization module then works to find a geometric configuration that minimizes the violation of these constraints. A prominent example is LayoutVLM [33], which translates user input into a set of differentiable constraints, allowing object poses to be directly optimized via gradient-based methods. Similarly, FlairGPT [79], methods by Sun et al. [80], and I-Design [81] all leverage an LLM to produce textual constraints or relations that a subsequent module interprets to place objects. This paradigm extends powerfully to physics, where physical laws act as implicit constraints. LayoutDreamer [62] pioneers this by regularizing the LLM's output with a physics engine to ensure stability, while RoomCraft [82] also relies on constraints to achieve complete, collision-free scenes.

Structured Blueprint Generation: Here, the LLM's primary task is to generate a comprehensive and structured data format—most commonly a scene graph—that serves as a holistic blueprint for the scene. In this formalism, objects are nodes and their relationships are edges. AnyHome [83] exemplifies this by first using an LLM to generate relational constraints which are then compiled into a structured scene graph. Graph Canvas [84] and GraphDreamer [85] place the graph at the center of their compositional synthesis pipeline. Other systems build sophisticated processes around this blueprint. Planner3D [86] uses the LLM-generated graph as a prior to regularize a numerical optimization process. DIScene [87] specializes in this by focusing on decoupling objects and modeling their interactions, which is the essence of constructing a detailed graph. Finally, Visual Harmony [88] and SceneLLM [89] demonstrate the versatility of this approach by generating dynamic scene graphs from varied inputs like images or text.

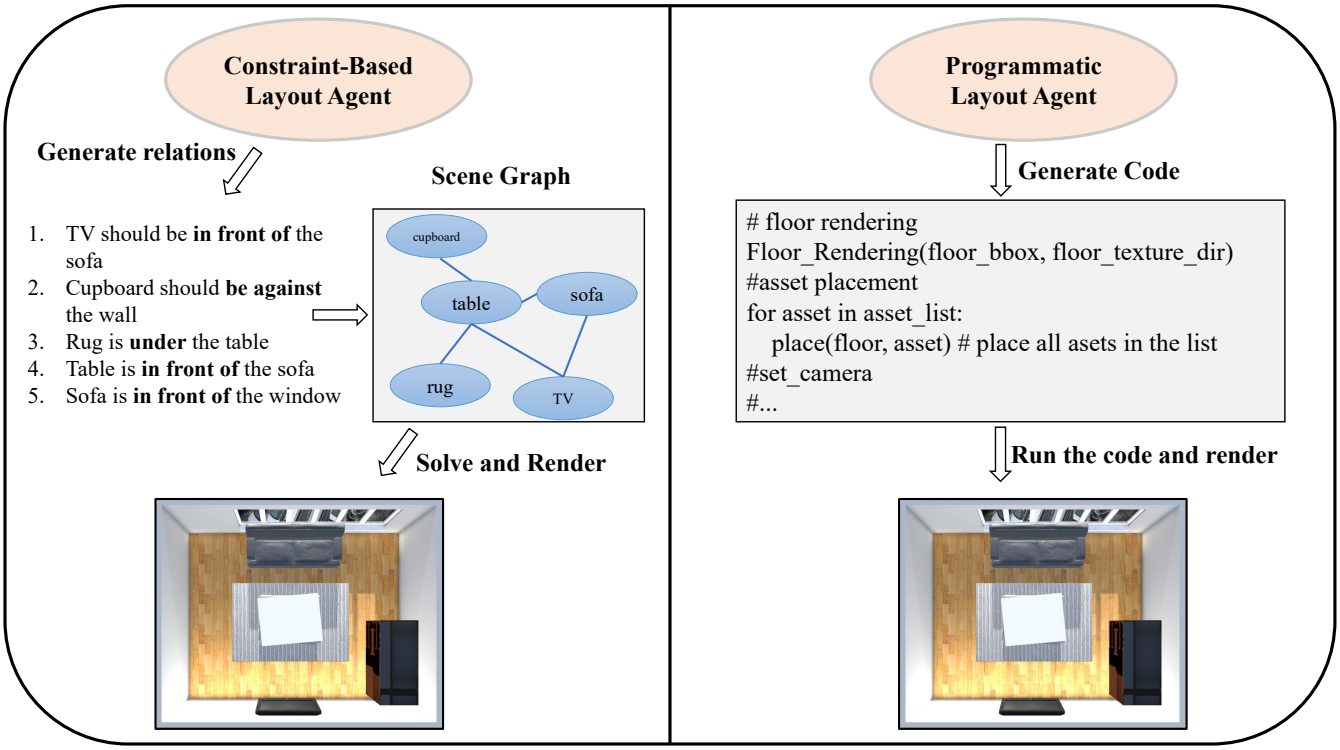

Figure 5: A comparison between the Constraint-Based Methods and Programming Language-Based Methods.

Agentic Planning and Search: In this most advanced paradigm, the LLM acts as an autonomous agent that actively uses constraints to guide a dynamic, multi-step process. The constraints are not just a static specification but a tool for reasoning and decision-making. For instance, Deng et al. [90] employs an LLM to generate spatial constraints that are then used to prune a tree search algorithm, allowing the agent to find valid placements efficiently and even backtrack to resolve conflicts. AutoLayout [91] establishes a "slow-fast collaborative reasoning" loop, where relational constraints are proposed and then iteratively refined by a separate process. At the highest level of abstraction, frameworks like Scenethesis [92] and WorldCraft [93] feature LLM agents that orchestrate a combination of tools—using constraints, generating procedural code, and executing multi-step reasoning—to construct vast and complex 3D worlds, showcasing how constraint-based reasoning becomes one of many skills in an agent's repertoire.

4.2.2 Programming Language-Based Methods.

This approach leverages the remarkable proficiency of LLMs as code generators. Instead of producing static coordinates or constraints, the LLM is tasked with writing an executable script that procedurally constructs the scene. This script is then run by an execution engine—often a 3D software environment like Blender [7] or a custom interpreter—to render the final layout. This methodology is powerful because it transforms the layout problem into a domain where LLMs excel, and the resulting code is structured, debuggable, and capable of encoding complex procedural logic such as loops, conditions, and variables. Zhang et al. [94] pioneered the creation of a dedicated, declarative Domain-Specific Language (DSL) for scenes, revealing the potential that an LLM could translate ambiguous natural language into a structured, unambiguous program that a custom interpreter could then execute to create a deterministic layout, effectively bridging the gap between human intent and machine execution. SceneMotifCoder [95] offers an innovative twist on this concept by learning a "visual program" from examples; the LLM is shown various visual motifs of object arrangements and synthesizes a programmatic script that can replicate those same spatial patterns, making it an example-driven approach to program learning. Meanwhile, Gumin et al. [96] provides a theoretical analysis of the trade-offs between generating step-by-step instructions (imperative code, like a Python script) versus defining the desired end state (declarative code, which specifies relationships).

Alternatively, many methods bypass custom languages and instead generate code in a General-Purpose Language (GPL) like Python, typically leveraging the API of a powerful 3D modeling tool. SceneCraft [29] is functions as an agent that directly translates a user's textual description into a complete Blender-compatible Python script. This script then procedurally creates, places, and modifies all objects within the Blender environment, harnessing the full power of a professional 3D suite. 3D-GPT [97] advances this architecture by using a team of chained agents; one agent parses the user's request, another conceptualizes the 3D model, and a final "coder" agent writes the procedural instructions for a tool like Blender to execute, showing how the complex task can be decomposed. For applications demanding high accuracy, SceneGenAgent [31] adapts this principle to industrial settings, where a dedicated coding agent generates scripts to place objects with exacting precision, a task for which programmatic control is uniquely suited. Finally, these programming-based methods culminate in sophisticated, hybrid agentic frameworks like WorldCraft [93]. This system employs LLM agents that can dynamically choose

the best tool for a task, using procedural code generation for large-scale world-building, defining constraints for fine-grained relationships, and employing multi-step reasoning to orchestrate the entire process, demonstrating the maturity of programming as a core component in a larger creative toolkit.

Notably, although we categorize layout agents into one-stage, multi-stage, constraint-based, and programmatic methods, some large-scale or multi-module agents can simultaneously incorporate multiple approaches. For example, in Holodeck, the door and floor agent employs a one-stage direct generation strategy, whereas the furniture module relies on constraint-based generation. Table 1 provides a more detailed categorization of the existing layout agents.

5 Evaluation of Layout Agents

A dedicated discussion on evaluation methodologies is crucial because, unlike traditional machine learning tasks with a single, clear "ground truth," 3D scene generation is an inherently ill-posed problem. For a prompt like "a cozy modern living room," there is no single correct answer; there exists a vast distribution of valid and aesthetically pleasing layouts. Consequently, simple metrics like comparing generated coordinates to a reference scene are insufficient and often misleading. Thus, a robust evaluation framework is necessary to systematically and reproducibly assess the quality of generated layouts, enabling fair comparison between models and guiding future research. In this section, we first introduce the necessary dimensions of LLM layout agent evaluation before discussing existing methodologies to quantify these aspects. A comprehensive comparison across existing models over performance evaluation is presented in Table 1.

5.1 Dimensions of Evaluation

The evaluation of a generated 3D layout is typically deconstructed into several key dimensions, each targeting a different aspect of scene quality.

Physical Plausibility: This is the most fundamental aspect of evaluation, assessing whether the scene adheres to basic physical laws. This dimension is typically broken down into three main criteria: inter-object intersections [99], which check if any two objects improperly occupy the same physical space; gravitational stability, which ensures that objects would not topple or fall in a real-world setting; and support hierarchy, which verifies that objects are logically supported by others (e.g., a laptop is on a table, not floating beside it). A physically impossible scene is immediately identifiable as low-quality, regardless of its other merits.

Prompt Fidelity: This is critical for text-to-scene models, as it measures how faithfully the generated scene reflects the user's input prompt. This evaluation encompasses two key areas: object and attribute presence, which is a direct check to ensure all requested objects (e.g., "a round table") and their properties (e.g., "blue chairs") are correctly instantiated in the scene; and relational alignment, which verifies that explicit spatial constraints mentioned in the prompt (e.g., "the painting above the sofa") are accurately fulfilled.

Semantic Coherence: This dimension evaluates the logical and functional consistency of the scene, answering the question: "Does this layout make sense?" As explored in benchmarks like SceneEval [100], this is assessed through two main lenses: object co-occurrence, which measures the likelihood of certain objects appearing together in a specific room type (e.g., a sofa and a television in a living room); and relational plausibility, which evaluates the sensibility of the spatial relationships between object pairs. For instance, a chair being next to a table is plausible, whereas a chair on top of a ceiling fan is not).

Aesthetics & Realism: This is the most subjective but arguably one of the most important dimensions, capturing the overall visual appeal, composition, and believability of the scene. This aspect considers factors like whether the layout is well-balanced and uncluttered (aesthetics) and whether the generated scene looks like a real-world photograph (realism).

5.2 Evaluation Metrics and Methodologies

To quantify the dimensions outlined above, a combination of automated geometric metrics, statistical measures, and human-in-the-loop evaluations are employed.

Metrics for Physical Plausibility: A wide range of models [33, 67, 69, 81] use geometric calculations to automatically assess physical violations. For example, Out-of-Boundary Rate (OOB) measures the percentage of scenes where one or more objects are placed partially or wholly outside the predefined room boundaries. Another common metric is Object-Overlap Rate [99], which calculates the fraction of generated scenes that contain at least one pair of intersecting objects, where intersection is typically determined by overlapping 3D bounding boxes.

Metrics for Prompt Fidelity: These metrics assess how well the generated content aligns with the user's textual instructions. One widely adopted metric is the CLIP Score [48], which measures the semantic similarity between the text prompt and rendered images of the generated scene [64, 70] [71] by encoding both into a shared embedding space and computing the cosine similarity between their vectors. In interactive settings, researchers often report an Editing Success Rate [64] to evaluate whether the model correctly executes a specific edit command (e.g., "make the table larger") by comparing the scene state before and after the command. A further quantitative check is the Average Number of Proposed Objects (NObj) [26], which verifies whether the model respects quantitative constraints. For example, if a prompt requests "a table and three chairs," this metric counts the number of objects generated to see if it matches the request.

Evaluation of Semantic Coherence, Aesthetics, and Realism: Due to their subjective nature, these dimensions often require human judgment or sophisticated AI-based proxies. A commonly used evaluation method is direct scoring, where evaluators, either human rater [69, 71, 82] or a powerful Vision-Language Model like GPT-4o [33, 81], are asked to score a generated scene on a predefined scale (e.g., 0/1 for binary correctness, or a 1-5 Likert scale) across several axes, such as "Is this layout functional?" or "Is this scene aesthetically pleasing?". To reduce cognitive load and rater bias, some researchers also use pairwise comparison. this method presents evaluators with two scenes generated from the same prompt by the baseline and the proposed model. The

Table 1: The classification of representative works on LLM-based layout agents and the evaluation perspectives they adopt.

Model	One Stage	Multi Stage	Direct Output	Constraint -Based	Physical Plausibility	Prompt Fidelity	Semantic Coherence	Aesthetics & Realism
LayoutGPT [26]	✓	×	✓	×	×	✓	—	—
IDesign [81]	×	✓	×	✓	✓	✓	M	M
SKE-Layout [66]	✓	×	✓	×	×	✓	—	—
OptiScene [67]	✓	×	✓	×	✓	✓	M	M
SceneTeller [69]	✓	×	✓	×	✓	✓	H	H
LLplace [64]	✓	×	✓	×	✓	✓	M	M
GALA3D [70]	✓	×	✓	×	✓	✓	H	H
Holodeck [71]	×	✓	✓	✓	×	✓	H	H
StageDesigner [73]	×	✓	✓	×	✓	✓	H	H
DirectLayout [74]	×	✓	✓	×	✓	✓	H	H
Liu et al. [30]	×	✓	×	✓	✓	✓	H & M	H & M
DisCo-Layout [77]	×	✓	✓	✓	✓	×	M	—
SceneWeaver [76]	×	✓	✓	×	✓	✓	M	M
LayoutVLM [33]	✓	×	×	✓	✓	✓	H & M	H & M
AnyHome [83]	×	✓	×	✓	✓	✓	—	—
Deng et al. [90]	×	✓	×	✓	×	✓	H	H
Liu et al. [84]	×	✓	×	✓	×	✓	H	H
DIScene [87]	✓	×	×	✓	×	✓	H	H
RoomCraft [82]	×	✓	×	✓	✓	✓	H	H
LayoutDreamer [62]	✓	×	×	✓	✓	✓	—	—
Scenethesis [92]	×	✓	×	✓	✓	✓	H & M	H & M
FlairGPT [79]	✓	×	×	✓	✓	✓	H & M	H & M
AutoLayout [91]	×	✓	×	✓	✓	✓	M	M
Visual Harmony [88]	✓	×	×	✓	×	×	H	H
Planner3D [86]	✓	×	×	✓	✓	✓	—	—
WorldCraft [93]	×	✓	×	✓	×	✓	H & M	H & M
SceneCraft [29]	×	✓	×	✓	×	✓	—	—
SceneGenAgent [31]	×	✓	×	✓	×	✓	H	H
SceneMotifCoder [95]	✓	×	✓	×	✓	✓	M	M
Chat2Layout [98]	×	✓	✓	×	✓	✓	—	—

For the metrics semantic coherence and aesthetics & realism, (1) M denotes evaluation by LLMs or VLMs, (2) H denotes evaluation by human participants, (3) H&M denotes both human and machine evaluation, (4) -indicates that the metric is not included. It is worth noting that many large-scale LLM agents are composed of multiple sub-agents, which may span across different paradigms. Therefore, multiple columns in this table can be checked simultaneously (for example, one part of an agent may adopt direct output, while another part is based on constraints).

evaluator's task is simply to choose which one they prefer ("Model A is better," "Model B is better," or "They are about equal"). This comparative judgment is often more reliable and consistent than assigning absolute scores.

6 Difficulty and Future Challenges

Despite notable advances in automated 3D scene generation, layout agents built on large language models (LLMs) still face four recurring challenges that limit their practical robustness and generality.

Limited spatial reasoning. These agents frequently exhibit gaps in spatial understanding that appear in several ways: many rely on text- or 2D image–centric visual-language backbones rather than representations that natively encode 3D geometry (e.g., point clouds or meshes); they tend to handle qualitative spatial language (such as "next to the sofa") better than quantitative relationships (precise distances, relative scales, and absolute orientations); and this shortfall often extends to physical common sense, producing artifacts such as floating objects, interpenetration, or unstable arrangements. As a result, downstream systems typically require ad-hoc post-processing rules or physics simulations to enforce semantic and physical plausibility.

Generation efficiency. Many state-of-the-art agents adopt iterative or multi-turn pipelines (for instance, stepwise optimization or repeated refinement dialogues) to improve controllability. While effective for steering outputs, these approaches increase computational cost and inference latency, which can make rapid generation of large-scale or high-fidelity scenes impractical in real-world settings.

Sensitivity and instability. Model outputs are often sensitive to prompt formulation and the details of the interaction loop. High-quality results frequently depend on careful prompt engineering, explicit constraints, or iterative user feedback. Consequently, modest variations in wording or dialog strategy can produce substantially different layouts, reducing robustness and making the user experience less predictable.

Limited generalization. Many models perform well within narrowly defined domains—particularly indoor home design, where most training data and benchmarks concentrate—but their performance degrades when applied to less familiar or open-world scenarios (for example, complex outdoor environments, industrial sites, or atypical room types). Addressing dataset bias and architectural bottlenecks is therefore important for moving from specialized "interior designers" toward more general "world builders."

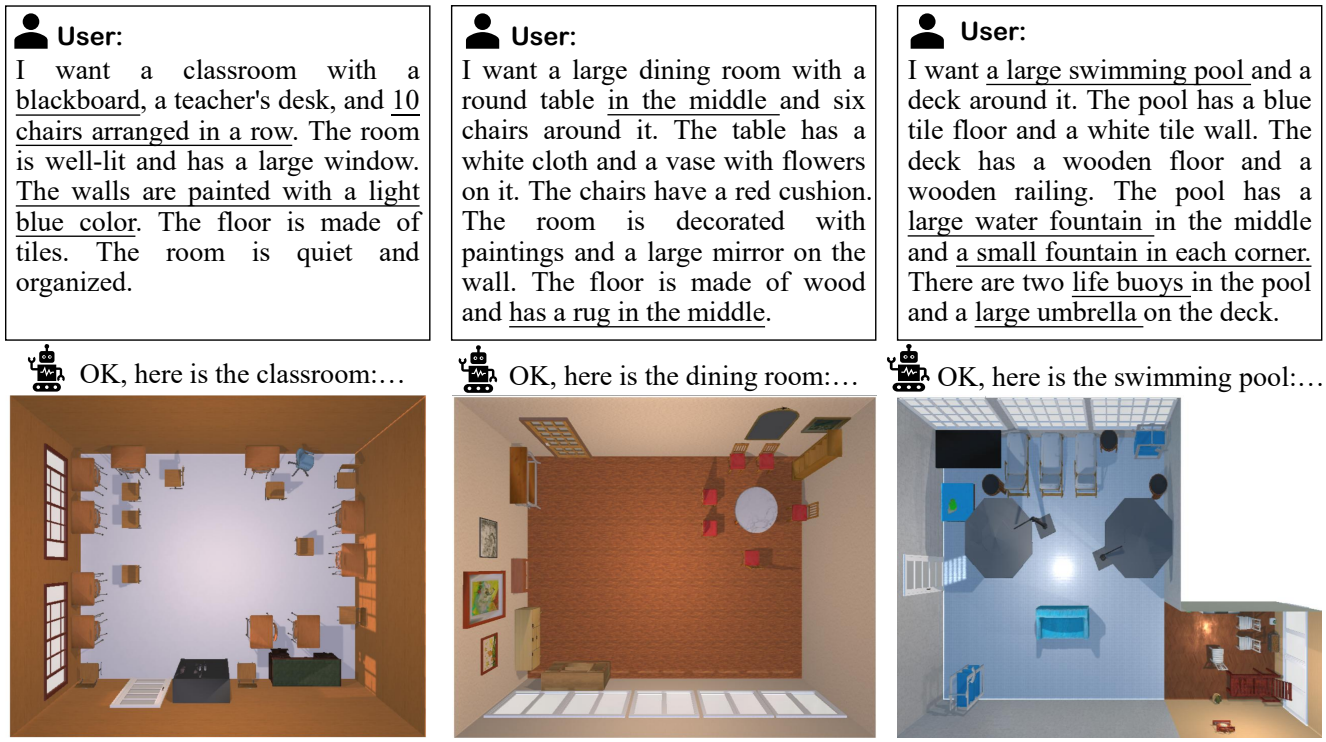

Figure 6: Three examples generated by Holodeck [71]. The inconsistencies between the scenes and the prompts are highlighted with underlines in the prompts.

In summary, improving spatial reasoning, efficiency, stability, and generalization is central to advancing LLM-based layout agents into dependable creative tools. To illustrate common failure modes, Figure 6 shows three examples generated by Holodeck [71]. In the classroom scene, the chairs are not arranged as expected (ten chairs forming a circle) and are not oriented toward the table. In the restaurant scene, the round table is not centered as intended and a central carpet is missing. In a request for a swimming-pool scene, the system fails to produce a coherent layout. Rather than reflecting a single deficiency, these failures expose several structural limitations shared by many current layout agents. First, most models are not explicitly trained or specialized for spatial perception tasks that require a global understanding of scene geometry (e.g., placing a table precisely at the center of a room). As a result, they often struggle with layout instructions that depend on holistic room-level awareness rather than local relational cues. Second, because most existing agents do not operate on detailed 3D representations such as dense point clouds or mesh-level geometry, but instead rely on coarse asset metadata or text/image-level abstractions, fine-grained placement details are frequently lost. This limitation manifests in subtle but important errors, such as chairs failing to face the table correctly or objects being misaligned despite satisfying high-level relational constraints. More fundamentally, the majority of layout agents are built upon a general-purpose LLM backbone. While effective in common indoor scenarios seen during training, such backbones generalize poorly to unfamiliar or open-domain environments. In atypical scenes—such as the swimming-pool example shown in Figure 6—the model may fail even at the asset retrieval stage, let alone at arranging appropriate objects in physically and semantically coherent configurations.

It is worth noticing that, these limitations have direct implications for embodied AI systems. When layout agents are used to generate environments for embodied navigation or manipulation, errors in global spatial structure, object orientation, or asset selection can severely hinder downstream perception, planning, and control. Consequently, the current shortcomings of LLM-based layout agents not only affect visual plausibility but also constrain their integration with embodied agents, highlighting the need for 3D-aware representations, domain-specialized training, and tighter coupling between symbolic reasoning, geometric execution, and physical interaction.

7 Conclusion

This survey systematically reviews the emerging field of LLM- and VLM-driven layout agents. We define the layout-agent task and present a unified framework that distinguishes direct methods (one-stage and multi-stage) from indirect methods (constraint-based and programmatic). Building on this taxonomy, we catalog representative systems, analyze their output representations and reasoning strategies, and synthesize evaluation protocols used across the literature. We further provide a structured comparison of evaluation metrics, highlighting strengths and gaps in existing benchmarks and automated measures. Finally, we identify four recurring challenges—spatial reasoning, generation efficiency, sensitivity/instability, and limited generalization—and discuss how these limitations shape current practice. Together, our contributions consolidate dispersed work into a coherent picture,

offer a practical lens for comparing methods, and supply researchers with an organized reference to guide the design and assessment of future layout agents.

Looking ahead, future research must focus on enhancing the 3D grounding and long-horizon planning capabilities of these agents. The development of more holistic and standardized benchmarks will also be crucial for driving meaningful progress. Ultimately, the continued advancement of LLM layout agents promises to democratize 3D content creation, building a powerful bridge between high-level human imagination and the automated generation of coherent, functional, and compelling digital worlds for applications ranging from robotics to the metaverse.

Funding

There is no funding support for this paper.

Author Contributions

The individual contributions are listed below:

- Conceptualization: Cheng Wan, Yuan Liu.
- Formal Analysis: Cheng Wan.
- Writing—original draft preparation: Cheng Wan.
- Writing—review and editing: Cheng Wan, Yuan Liu, and Yongsen Mao.
- Supervision: Yuan Liu.
- Project Administration: Yuan Liu.
- Funding Acquisition: Yuan Liu.

Conflict of Interest

All the authors declare that they have no conflict of interest.

Data Avaliable

As a review, there is no data or code materials used in this study.

References

[1] Yu, L.-F., Yeung, S.-K., Tang, C.-K., Terzopoulos, D., Chan, T.F., Osher, S.J.: Make it home: automatic optimization of furniture arrangement. ACM Transactions on Graphics **30**(4), 1–12 (2011) https://doi.org/10.1145/2010324.1964981

[2] Sun, X., Goel, D., Chang, A.X.: SemLayoutDiff: Semantic Layout Generation with Diffusion Model for Indoor Scene Synthesis. arXiv preprint arXiv:2508.18597 (2025) https://doi.org/10.48550/arXiv.2508.18597

[3] Raistrick, A., Mei, L., Kayan, K., Yan, D., Zuo, Y., Han, B., Wen, H., Parakh, M., Alexandropoulos, S., Lipson, L., Ma, Z., Deng, J.: Infinigen Indoors: Photorealistic Indoor Scenes using Procedural Generation. arXiv preprint arXiv:2406.11824 (2024) https://doi.org/10.48550/arXiv.2406.11824

[4] Peddiraju, P., Clark, C.: Procedural Image Generation Using Markov Wave Function Collapse in Advances in Artificial Intelligence and Applied Cognitive Computing, pp. 525–542. Springer, Cham, Switzerland (2021). https://doi.org/10.1007/978-3-030-70296-0_39

[5] Zhong, W., Cao, P., Jin, Y., Luo, L., Cai, W., Lin, J., Wang, H., Lyu, Z., Wang, T., Dai, B., Xu, X., Pang, J.: InternScenes: A Large-scale Simulatable Indoor Scene Dataset with Realistic Layouts. arXiv preprint arXiv:2509.10813 (2025) https://doi.org/10.48550/arXiv.2509.10813

[6] Alexander, C.: A pattern language: towns, buildings, construction. Oxford university press, Berkeley, California, USA (1977)

[7] Gschwandtner, M., Kwitt, R., Uhl, A., Pree, W.: BlenSor: Blender sensor simulation toolbox. In International Symposium on Visual Computing, pp. 199–208 (2011). https://doi.org/10.1007/978-3-642-24031-7_20

[8] Parish, Y.I., Müller, P.: Procedural modeling of cities. In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, pp. 301–308 (2001). https://doi.org/10.1145/383259.383292

[9] Merrell, P., Schkufza, E., Li, Z., Agrawala, M., Koltun, V.: Interactive furniture layout using interior design guidelines. In ACM SIGGRAPH 2011 Papers, pp. 49–86 (2011). https://doi.org/10.1145/1964921.1964982

[10] Li, Y., Chen, H., Yu, P., Yang, L.: A review of artificial intelligence in enhancing architectural design efficiency. Applied Sciences **15**(3), 1476 (2025) https://doi.org/10.3390/app15031476

[11] Poole, B., Jain, A., Barron, J.T., Mildenhall, B.: Dreamfusion: Text-to-3d using 2d diffusion. arXiv preprint arXiv:2209.14988 (2022) https://doi.org/10.48550/arXiv.2209.14988

[12] Paschalidou, D., Kar, A., Shugrina, M., Kreis, K., Geiger, A., Fidler, S.: Atiss: Autoregressive transformers for indoor scene synthesis. In Advances in Neural Information Processing Systems 34, NeurIPS 2021, pp. 12013–12026 (2021)

[13] Li, M., Patil, A.G., Xu, K., Chaudhuri, S., Khan, O., Shamir, A., Tu, C., Chen, B., Cohen-Or, D., Zhang, H.: Grains: Generative recursive autoencoders for indoor scenes. ACM Transactions on Graphics **38**(2), 1–16 (2019) https://doi.org/10.1145/3303766

[14] Wen, B., Xie, H., Chen, Z., Hong, F., Liu, Z.: 3D Scene Generation: A Survey. arXiv preprint arXiv:2505.05474 (2025) https://doi.org/10.48550/arXiv.2505.05474

[15] Wu, J., Zhang, C., Xue, T., Freeman, W.T., Tenenbaum, J.B.: Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In Proceedings of the 30th International Conference on Neural Information Processing Systems, pp. 82–90 (2016)

[16] Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021) https://doi.org/10.1145/3503250

[17] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10684–10695 (2022). https://doi.org/10.1109/CVPR52688.2022.01042

[18] Liu, Y., Zhang, K., Li, Y., Yan, Z., Gao, C., Chen, R., Yuan, Z., Huang, Y., Sun, H., Gao, J., *et al.*: Sora: A review on background, technology, limitations, and opportunities of large vision models. arXiv preprint arXiv:2402.17177 (2024) https://doi.org/10.48550/arXiv.2402.17177

[19] Fime, A.A., Mahmud, S., Das, A., Islam, M.S., Kim, J.-H.: Automatic Scene Generation: State-of-the-Art Techniques, Models, Datasets, Challenges, and Future Prospects. arXiv preprint arXiv:2410.01816 (2024) https://doi.org/10.48550/arXiv.2410.01816

[20] Zha, J., Fan, Y., Yang, X., Gao, C., Chen, X.: How to Enable LLM with 3D Capacity? A Survey of Spatial Reasoning in LLM. arXiv preprint arXiv:2504.05786 (2025) https://doi.org/10.48550/arXiv.2504.05786

[21] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 6000–6010 (2017)

[22] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., *et al.*: Language models are few-shot learners. In Proceedings of the 34th International Conference on Neural Information Processing Systems, pp. 1877–1901 (2020)

[23] Yin, S., Fu, C., Zhao, S., Li, K., Sun, X., Xu, T., Chen, E.: A survey on multimodal large language models. In Proceedings of the 3rd International Conference on Computer, Artificial Intelligence and Control Engineering, pp. 405–409 (2024). https://doi.org/10.1145/3672758.3672824

[24] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., *et al.*: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023) https://doi.org/10.48550/arXiv.2303.08774

[25] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., *et al.*: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023) https://doi.org/10.48550/arXiv.2302.13971

[26] Feng, W., Zhu, W., Fu, T.-j., Jampani, V., Akula, A., He, X., Basu, S., Wang, X.E., Wang, W.Y.: Layoutgpt: Compositional visual planning and generation with large language models. In Proceedings of the 37th International Conference on Neural Information Processing Systems, pp. 18225–18250 (2023)

[27] Zhang, J., Huang, J., Jin, S., Lu, S.: Vision-language models for vision tasks: A survey. IEEE transactions on pattern analysis and machine intelligence **46**(8), 5625–5644 (2024) https://doi.org/10.1109/TPAMI.2024.3369699

[28] Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. In Proceedings of the 37th International Conference on Neural Information Processing Systems, pp. 34892–34916 (2023)

[29] Hu, Z., Iscen, A., Jain, A., Kipf, T., Yue, Y., Ross, D.A., Schmid, C., Fathi, A.: Scenecraft: An llm agent for synthesizing 3d scenes as blender code. In Proceedings of the 41st International Conference on Machine Learning, pp. 19252–19282 (2024)

[30] Liu, X., Tai, Y.-W., Tang, C.-K.: Agentic 3D Scene Generation with Spatially Contextualized VLMs. arXiv preprint arXiv:2505.20129 (2025) https://doi.org/10.48550/arXiv.2505.20129

[31] Xia, X., Zhang, D., Liao, Z., Hou, Z., Sun, T., Li, J., Fu, L., Dong, Y.: SceneGenAgent: Precise Industrial Scene Generation with Coding Agent. arXiv preprint arXiv:2410.21909 (2025) https://doi.org/10.48550/arXiv.2410.21909

[32] Shi, Z., Peng, S., Xu, Y., Geiger, A., Liao, Y., Shen, Y.: Deep generative models on 3d representations: A survey. arXiv preprint arXiv:2210.15663 (2022) https://doi.org/10.48550/arXiv.2210.15663

[33] Sun, F.-Y., Liu, W., Gu, S., Lim, D., Bhat, G., Tombari, F., Li, M., Haber, N., Wu, J.: Layoutvlm: Differentiable optimization of 3d layout via vision-language models. In Proceedings of the Computer Vision and Pattern Recognition Conference, pp. 29469–29478 (2025). https://doi.org/10.1109/CVPR52734.2025.02744

[34] Tang, J., Nie, Y., Markhasin, L., Dai, A., Thies, J., Nießner, M.: Diffuscene: Denoising diffusion models for generative indoor scene synthesis. In Proceedings

of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 20507–20518 (2024). https://doi.org/10.1109/CVPR52733.2024.01938

[35] Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3D Gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4), 1–14 (2023) https://doi.org/10.1145/3592433

[36] Tancik, M., Casser, V., Yan, X., Pradhan, S., Mildenhall, B., Srinivasan, P.P., Barron, J.T., Kretzschmar, H.: Block-nerf: Scalable large scene neural view synthesis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8248–8258 (2022). https://doi.org/10.1109/CVPR52688.2022.00807

[37] Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM transactions on graphics **41**(4), 1–15 (2022) https://doi.org/10.1145/3528223.3530127

[38] Lin, C.H., Lee, H.-Y., Menapace, W., Chai, M., Siarohin, A., Yang, M.-H., Tulyakov, S.: Infinicity: Infinite-scale city synthesis. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 22808–22818 (2023). https://doi.org/10.1109/ICCV51070.2023.02085

[39] Bian, Z., Ren, R., Yang, Y., Callison-Burch, C.: HOLODECK 2.0: Vision-Language-Guided 3D World Generation with Editing. arXiv preprint arXiv:2508.05899 (2025) https://doi.org/10.48550/arXiv.2508.05899

[40] Bar-Tal, O., Chefer, H., Tov, O., Herrmann, C., Paiss, R., Zada, S., Ephrat, A., Hur, J., Liu, G., Raj, A., *et al.*: Lumiere: A space-time diffusion model for video generation. In SIGGRAPH Asia 2024 Conference Papers, pp. 1–11 (2024). https://doi.org/10.1145/3680528.3687614

[41] Xing, Z., Feng, Q., Chen, H., Dai, Q., Hu, H., Xu, H., Wu, Z., Jiang, Y.-G.: A survey on video diffusion models. ACM Computing Surveys **57**(2), 1–42 (2024) https://doi.org/10.1145/3696415

[42] Liu, H., Yan, W., Zaharia, M., Abbeel, P.: World model on million-length video and language with blockwise ringattention. arXiv preprint arXiv:2402.08268 (2024) https://doi.org/10.48550/arXiv.2402.08268

[43] Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., *et al.*: Qwen technical report. arXiv preprint arXiv:2309.16609 (2023) https://doi.org/10.48550/arXiv.2309.16609

[44] Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., *et al.*: Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437 (2024) https://doi.org/10.48550/arXiv.2412.19437

[45] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training. https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf Accessed 2025-11-28

[46] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., *et al.*: Training language models to follow instructions with human feedback. In Advances in Neural Information Processing Systems 35 (NeurIPS 2022), pp. 27730–27744 (2022)

[47] Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., *et al.*: Flamingo: a visual language model for few-shot learning. Advances in neural information processing systems **35**, 23716–23736 (2022)

[48] Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., *et al.*: Learning transferable visual models from natural language supervision. In International Conference on Machine Learning, pp. 8748–8763 (2021)

[49] Li, J., Li, D., Savarese, S., Hoi, S.: Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In International Conference on Machine Learning, pp. 19730–19742 (2023)

[50] Wang, Z., Yu, J., Yu, A.W., Dai, Z., Tsvetkov, Y., Cao, Y.: Simvlm: Simple visual language model pretraining with weak supervision. arXiv preprint arXiv:2108.10904 (2021) https://doi.org/10.48550/arXiv.2108.10904

[51] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., *et al.*: Chain-of-thought prompting elicits reasoning in large language models. In Proceedings of the 36th International Conference on Neural Information Processing Systems, pp. 24824–24837 (2022)

[52] Hong, Y., Zhen, H., Chen, P., Zheng, S., Du, Y., Chen, Z., Gan, C.: 3D-LLM: Injecting the 3d world into large language models. In Proceedings of the 37th International Conference on Neural Information Processing Systems, pp. 20482–20494 (2023)

[53] Wang, J., Jiang, H., Liu, Y., Ma, C., Zhang, X., Pan, Y., Liu, M., Gu, P., Xia, S., Li, W., *et al.*: A comprehensive review of multimodal large language models: Performance and challenges across different tasks. arXiv preprint arXiv:2408.01319 (2024) https://doi.org/10.48550/arXiv.2408.01319

[54] Driess, D., Xia, F., Sajjadi, M.S., Lynch, C., Chowdhery, A., Wahid, A., Tompson, J., Vuong, Q., Yu, T., Huang, W., *et al.*: PaLM-E: An embodied multimodal language model. In Proceedings of the 40th International Conference on Machine Learning, pp. 8469–8488 (2023)

[55] Li, F., Zhang, R., Zhang, H., Zhang, Y., Li, B., Li, W., Ma, Z., Li, C.: Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models. arXiv preprint arXiv:2407.07895 (2024) https://doi.org/10.48550/arXiv.2407.07895

[56] Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F., Wu, J., Wohlhart, P., Welker, S., Wahid, A., *et al.*: Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Proceedings of The 7th Conference on Robot Learning, pp. 2165–2183 (2023)

[57] Qi, Z., Fang, Y., Sun, Z., Wu, X., Wu, T., Wang, J., Lin, D., Zhao, H.: Gpt4point: A unified framework for point-language understanding and generation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 26417–26427 (2024). https://doi.org/10.1109/CVPR52733.2024.02495

[58] Wang, L., Ma, C., Feng, X., Zhang, Z., Yang, H., Zhang, J., Chen, Z., Tang, J., Chen, X., Lin, Y., *et al.*: A survey on large language model based autonomous agents. Frontiers of Computer Science **18**(6), 186345 (2024) https://doi.org/10.1007/s11704-024-40231-1

[59] Park, J.S., O'Brien, J., Cai, C.J., Morris, M.R., Liang, P., Bernstein, M.S.: Generative agents: Interactive simulacra of human behavior. In Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, pp. 1–22 (2023). https://doi.org/10.1145/3586183.3606763

[60] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K.R., Cao, Y.: React: Synergizing reasoning and acting in language models. In The Eleventh International Conference on Learning Representations, pp. 1–33 (2022)

[61] Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., Scialom, T.: Toolformer: Language models can teach themselves to use tools. In Proceedings of the 37th International Conference on Neural Information Processing Systems, pp. 68539–68551 (2023)

[62] Zhou, Y., He, Z., Li, Q., Wang, C.: LAYOUTDREAMER: Physics-guided Layout for Text-to-3D Compositional Scene Generation. arXiv preprint arXiv:2502.01949 (2025) https://doi.org/10.48550/arXiv.2502.01949

[63] Rivera, C., Byrd, G., Paul, W., Feldman, T., Booker, M., Holmes, E., Handelman, D., Kemp, B., Badger, A., Schmidt, A., *et al.*: Conceptagent: Llm-driven precondition grounding and tree search for robust task planning and execution. In 2025 IEEE International Conference on Robotics and Automation (ICRA), pp. 8988–8995 (2025). https://doi.org/10.1109/ICRA55743.2025.11128414

[64] Yang, Y., Lu, J., Zhao, Z., Luo, Z., Yu, J.J.Q., Sanchez, V., Zheng, F.: LLplace: The 3D Indoor Scene Layout Generation and Editing via Large Language Model. arXiv preprint arXiv:2406.03866 (2024) https://doi.org/10.48550/arXiv.2406.03866

[65] Deng, J., Chai, W., Huang, J., Zhao, Z., Huang, Q., Gao, M., Guo, J., Hao, S., Hu, W., Hwang, J.-N., *et al.*: Citycraft: A real crafter for 3d city generation. arXiv preprint arXiv:2406.04983 (2024) https://doi.org/10.48550/arXiv.2406.04983

[66] Wang, J., Cao, N., Ding, Y., Xie, M., Gu, F., Chen, C.: SKE-Layout: Spatial Knowledge Enhanced Layout Generation with LLMs. In 2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 19414–19423 (2025). https://doi.org/10.1109/CVPR52734.2025.01808

[67] Yang, Y., Luo, Z., Ding, T., Lu, J., Gao, M., Yang, J., Sanchez, V., Zheng, F.: OptiScene: LLM-driven Indoor Scene Layout Generation via Scaled Human-aligned Data Synthesis and Multi-Stage Preference Optimization. arXiv preprint arXiv:2506.07570 (2025) https://doi.org/10.48550/arXiv.2506.07570

[68] Jiang, B., Ma, R.: Cot2Scene: Generating 3D Indoor Scenes Using CoT-Enhanced LLMs. In 2025 6th International Conference on Computer Engineering and Application (ICCEA), pp. 1687–1690 (2025). https://doi.org/10.1109/ICCEA65460.2025.11103192

[69] Öcal, B.M., Tatarchenko, M., Karaoglu, S., Gevers, T.: SceneTeller: Language-to-3D Scene Generation. arXiv preprint arXiv:2407.20727 (2024) https://doi.org/10.48550/arXiv.2407.20727

[70] Zhou, X., Ran, X., Xiong, Y., He, J., Lin, Z., Wang, Y., Sun, D., Yang, M.-H.: GALA3D: Towards Text-to-3D Complex Scene Generation via Layout-guided Generative Gaussian Splatting. arXiv preprint arXiv:2402.07207 (2024) https://doi.org/10.48550/arXiv.2402.07207

[71] Yang, Y., Sun, F.-Y., Weihs, L., VanderBilt, E., Herrasti, A., Han, W., Wu, J., Haber, N., Krishna, R., Liu, L., *et al.*: Holodeck: Language guided generation of 3d embodied ai environments. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16227–16237 (2024). https://doi.org/10.1109/CVPR52733.2024.01536

[72] Zeng, P., Gao, W., Li, J., Yin, J., Chen, J., Lu, S.: Automated residential layout generation and editing using natural language and images. Automation in Construction **174**, 106133 (2025) https://doi.org/10.1016/j.autcon.2025.106133

[73] Gan, Z., Li, M., Chen, R., Ji, Z., Guo, S., Hu, H., Ye, G., Hu, Z.: StageDesigner: Artistic Stage Generation for Scenography via Theater Scripts. arXiv preprint arXiv:2503.02595 (2025) https://doi.org/10.48550/arXiv.2503.02595

[74] Ran, X., Li, Y., Xu, L., Yu, M., Dai, B.: Direct Numerical Layout Generation for 3D Indoor Scene Synthesis via Spatial Reasoning. arXiv preprint arXiv:2506.05341 (2025) https://doi.org/10.48550/arXiv.2506.05341

[75] Liu, J.-H., Zhang, S.-K., Zhang, C., Zhang, S.-H.: Controllable procedural generation of landscapes. In Proceedings of the 32nd ACM International Conference on Multimedia, pp. 6394–6403 (2024). https://doi.org/10.1145/3664647.3681129

[76] Yang, Y., Jia, B., Zhang, S., Huang, S.: SceneWeaver: All-in-One 3D Scene Synthesis with an Extensible and Self-Reflective Agent. arXiv preprint arXiv:2509.20414 (2025) https://doi.org/10.48550/arXiv.2509.20414

[77] Gao, J., Zhou, D., Liang, M., Liu, L., Fu, C.-W., Hu, X., Heng, P.-A.: DisCo-Layout: Disentangling and Coordinating Semantic and Physical Refinement in a Multi-Agent Framework for 3D Indoor Layout Synthesis. arXiv preprint arXiv:2510.02178 (2025) https://doi.org/10.48550/arXiv.2510.02178

[78] Fisher, M., Ritchie, D., Savva, M., Funkhouser, T., Hanrahan, P.: Example-based synthesis of 3D object arrangements. ACM Transactions on Graphics **31**(6), 1–11 (2012) https://doi.org/10.1145/2366145.2366154

[79] Littlefair, G., Dutt, N.S., Mitra, N.J.: FlairGPT: Repurposing LLMs for interior designs. Computer Graphics Forum **44**(2), 70036 (2025) https://doi.org/10.1111/cgf.70036

[80] Sun, W., Li, X., Li, M., Xu, K., Meng, X., Meng, L.: Hierarchically-Structured Open-Vocabulary Indoor Scene Synthesis with Pre-trained Large Language Model. In Proceedings of the AAAI Conference on Artificial Intelligence, pp. 7122–7130 (2025). https://doi.org/10.1609/aaai.v39i7.32765

[81] Çelen, A., Han, G., Schindler, K., Van Gool, L., Armeni, I., Obukhov, A., Wang, X.: I-design: Personalized llm interior designer, pp. 217–234 (2024). https://doi.org/10.1007/978-3-031-92387-6_17

[82] Zhou, M., Wang, X., Wang, Y., Zhang, Z.: RoomCraft: Controllable and Complete 3D Indoor Scene Generation. arXiv preprint arXiv:2506.22291 (2025) https://doi.org/10.48550/arXiv.2506.22291

[83] Fu, R., Wen, Z., Liu, Z., Sridhar, S.: AnyHome: Open-Vocabulary Generation of Structured and Textured 3D Homes (2024). https://doi.org/10.48550/arXiv.2312.06644

[84] Liu, L., Chen, S., Jia, S., Shi, J., Jiang, Z., Jin, C., Zongkai, W., Hwang, J.-N., Li, L.: Graph Canvas for Controllable 3D Scene Generation. arXiv preprint arXiv:2412.00091 (2024) https://doi.org/10.48550/arXiv.2412.00091

[85] Gao, G., Liu, W., Chen, A., Geiger, A., Schölkopf, B.: Graphdreamer: Compositional 3d scene synthesis from scene graphs. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 21295–21304 (2024). https://doi.org/10.1109/CVPR52733.2024.02012

[86] Wei, Y., Min, M.R., Vosselman, G., Li, L.E., Yang, M.Y.: Planner3D: LLM-enhanced graph prior meets 3D indoor scene explicit regularization. arXiv preprint arXiv:2403.12848 (2024) https://doi.org/10.48550/arXiv.2403.12848

[87] Li, X.-L., Li, H., Chen, H.-X., Mu, T.-J., Hu, S.-M.: DIScene: Object Decoupling and Interaction Modeling for Complex Scene Generation. In SIGGRAPH Asia 2024 Conference Papers, pp. 1–12 (2024). https://doi.org/10.1145/3680528.3687589

[88] Zhang, G., Wang, Y., Luo, C., Xu, S., Ming, Y., Peng, J., Zhang, M.: Visual Harmony: LLM's Power in Crafting Coherent Indoor Scenes from Images. In Lin, Z., Cheng, M.-M., He, R., Ubul, K., Silamu, W., Zha, H., Zhou, J., Liu, C.-L. (eds.) Pattern Recognition and Computer Vision, pp. 3–17. Springer, Singapore (2025)

[89] Fu, R., Liu, J., Chen, X., Nie, Y., Xiong, W.: Scene-LLM: Extending Language Model for 3D Visual Understanding and Reasoning. arXiv preprint arXiv:2403.11401 (2024) https://doi.org/10.48550/arXiv.2403.11401

[90] Deng, W., Qi, M., Ma, H.: Global-local tree search in vlms for 3D indoor scene generation. In Proceedings of the Computer Vision and Pattern Recognition Conference, pp. 8975–8984 (2025). https://doi.org/10.1109/CVPR52734.2025.00839

[91] Chen, W., Chi, D., Liu, Y., Yang, Y., Zhang, Y., Zhuang, Y., Quan, X., Hao, J., Li, G., Lin, L.: AutoLayout: Closed-Loop Layout Synthesis via Slow-Fast Collaborative Reasoning. arXiv preprint arXiv:2507.04293 (2025) https://doi.org/10.48550/arXiv.2507.04293

[92] Ling, L., Lin, C.-H., Lin, T.-Y., Ding, Y., Zeng, Y., Sheng, Y., Ge, Y., Liu, M.-Y., Bera, A., Li, Z.: Scenethesis: A Language and Vision Agentic Framework for 3D Scene Generation. arXiv preprint arXiv:2505.02836 (2025) https://doi.org/10.48550/arXiv.2505.02836

[93] Liu, X., Tang, C.-K., Tai, Y.-W.: WorldCraft: Photo-Realistic 3D World Creation and Customization via LLM Agents. arXiv preprint arXiv:2502.15601 (2025) https://doi.org/10.48550/arXiv.2502.15601

[94] Zhang, Y., Li, Z., Zhou, M., Wu, S., Wu, J.: The scene language: Representing scenes with programs, words, and embeddings. In Proceedings of the Computer Vision and Pattern Recognition Conference, pp. 24625–24634 (2025). https://doi.org/10.1109/CVPR52734.2025.0229

[95] Tam, H.I.I., Pun, H.I.D., Wang, A.T., Chang, A.X., Savva, M.: SceneMotifCoder: Example-driven Visual Program Learning for Generating 3D Object Arrangements. arXiv preprint arXiv:2408.02211 (2025) https://doi.org/10.48550/arXiv.2408.02211

[96] Gumin, M., Han, D.H., Yoo, S.J., Ganeshan, A., Jones, R.K., Aguina-Kang, R., Morris, S., Ritchie, D.: Imperative vs. Declarative Programming Paradigms for Open-Universe Scene Generation. arXiv preprint arXiv:2504.05482 (2025) https://doi.org/10.48550/arXiv.2504.05482

[97] Sun, C., Han, J., Deng, W., Wang, X., Qin, Z., Gould, S.: 3D-gpt: Procedural 3D modeling with large language models. In 2025 International Conference on 3D Vision (3DV), pp. 1253–1263 (2025). https://doi.org/10.1109/3DV66043.2025.00119

[98] Wang, C., Zhong, H., Chai, M., He, M., Chen, D., Liao, J.: Chat2Layout: Interactive 3D furniture layout with a multimodal LLM. arXiv preprint arXiv:2407.21333 (2024) https://doi.org/10.48550/arXiv.2407.21333

[99] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 658–666 (2019). https://doi.org/10.1109/CVPR.2019.00075

[100] Tam, H.I.I., Pun, H.I.D., Wang, A.T., Chang, A.X., Savva, M.: SceneEval: Evaluating semantic coherence in text-conditioned 3D indoor scene synthesis. arXiv preprint arXiv:2503.14756 (2025) https://doi.org/10.48550/arXiv.2503.14756