INFINIBENCH: INFINITE BENCHMARKING FOR VISUAL SPATIAL REASONING WITH CUSTOMIZABLE SCENE COMPLEXITY

Haoming Wang   Qiyao Xue   Wei Gao
Department of Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, PA 15261, USA
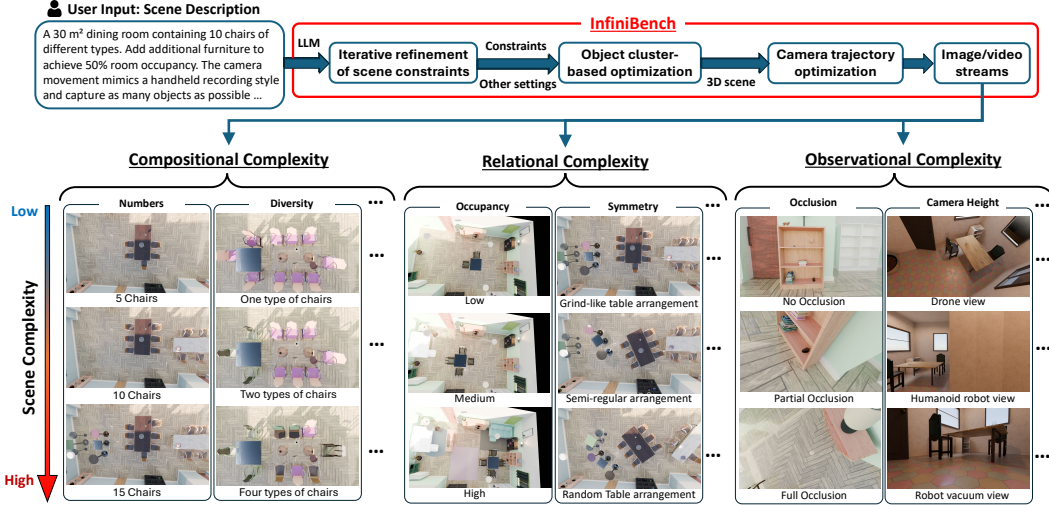`{haw200,qiyao_xue,weigao}@pitt.edu`

Figure 1: **InfiniBench**, as a fully automated, customizable and user-friendly benchmark generator, can generate theoretically unlimited variations of 3D scene benchmarks, according to the parameterized constraints of compositional, relational and observational scene complexities, to meet different benchmarking needs of analyzing the VLM's successes and failures in spatial reasoning tasks.

ABSTRACT

Modern vision-language models (VLMs) are expected to have abilities of spatial reasoning with diverse scene complexities, but evaluating such abilities is difficult due to the lack of benchmarks that are not only diverse and scalable but also fully customizable. Existing benchmarks offer limited customizability over the scene complexity and are incapable of isolating and analyzing specific VLM failure modes under distinct spatial conditions. To address this gap, instead of individually presenting benchmarks for different scene complexities, in this paper we present InfiniBench, a fully automated, customizable and user-friendly benchmark generator that can synthesize a theoretically infinite variety of 3D scenes with parameterized control on scene complexity. InfiniBench uniquely translates scene descriptions in natural language into photo-realistic videos with complex and physically plausible 3D layouts. This is achieved through three key innovations: 1) a LLM-based agentic framework that iteratively refines procedural scene constraints from scene descriptions; 2) a flexible cluster-based layout optimizer that generates dense and cluttered scenes previously intractable for procedural methods; and 3) a task-aware camera trajectory optimization method that renders scenes into videos with full object coverage as VLM input. Experiments demonstrate that InfiniBench outperforms state-of-the-art procedural and LLM-based 3D generation methods in prompt fidelity and physical plausibility, especially in high-complexity scenarios. We further showcased the usefulness of InfiniBench, by generating benchmarks for representative spatial reasoning tasks including measurement, perspective-taking and spatiotemporal tracking.

1

1 INTRODUCTION

Visual spatial reasoning Yang et al. (2025a), which comprehends the placement, orientation, and relationships of objects within a 3D space, is essential for AI to understand the real-world scenes from visual inputs such as images and video streams Bai et al. (2023); Chen et al. (2024b); Hurst et al. (2024). Unlike vision tasks on semantic perception of objects (e.g. image classification Devadason et al. (2025), segmentation Csurka et al. (2022); Barbosa and Osório (2023) and captioning Ghandi et al. (2023); Hu et al. (2022)), visual spatial reasoning requires abilities in geometric cognition, including spatial relation understanding, mental rotation and perspective taking Stogiannidis et al. (2025).

However, current vision-language models (VLMs) struggle to bridge the gap between 2D visual perception and 3D spatial reasoning, especially in scenes with high complexity Chen et al. (2025b). Such scene complexity is a multi-faceted concept that describes how the difficulty of understanding the scene's spatial arrangements varies, and it includes three dimensions: *1) Compositional complexity* about the scene content such as the number and variety of objects, which could result in visual clutter Chen et al. (2025b); *2) Relational complexity* about the intricacy of spatial relationships between objects, as more complicated spatial arrangement or higher ratio of scene occupancy challenges the VLM's capacity of accurately grounding the spatial language Chen et al. (2025a); *3) Observational complexity* about how the scene is presented, encompassing challenges introduced by extreme viewpoints and object occlusion Yu et al. (2025). These three dimensions disentangle different targets in human spatial perception, including the scene elements (compositional), the interconnections between these elements (relational), and the viewpoint-dependent nature of perception (observational) Greene (2023).

To thoroughly understand the VLM's abilities of spatial reasoning in these dimensions is a big challenge, because *they need benchmarks that are not only **diverse** and **scalable** but also **fully customizable***. Benchmarks should enable full isolation and flexible manipulation of scene properties, such as object density, relative positioning and levels of occlusion levels, through a set of **configurable parameters** within a large combinatorial space. For example, to evaluate VLM's robustness to irrelevant objects, the benchmark should be able to systematically increase the number of irrelevant objects, while retaining the task-related objects unchanged. This fine-grained customizabilility will allow us moving beyond the aggregate performance metrics (e.g., average task accuracy Yang et al. (2025a)), to uncover the specific failure modes of VLMs under distinct spatial conditions.

Existing benchmarks face significant limitations in customizablility, scalability, and semantic richness. Real-world datasets Yin et al. (2023); Hong et al. (2023); Jia et al. (2025); Wu et al. (2025); Chen et al. (2024a); Jia et al. (2025), although photorealistic, are costly to scale and lack parameterized control. A better alternative is to use synthetic data. Early methods use 3D rendering engines such as Blender Blender Foundation (2025) and IsaacSim isaac sim (n.d.) to procedurally generate 3D scenes, but lack realism and fail to capture the real-world functional and aesthetic constraints Wang et al. (2023). Recent techniques address these limitations by using 3D-aware diffusion models Meng et al. (2025); Liu et al. (2024); Go et al. (2025b;a); Engstler et al. (2025); Li et al. (2025); Xiang et al. (2025) with text Yang et al. (2025d); Go et al. (2025b) or image priors Engstler et al. (2025); Xia et al. (2025) for visual richness, but lack semantic annotations to generate QA pairs for reasoning tasks and fail to guarantee physical scene coherence. LLM-based scene generation Ran et al. (2025); Bucher and Armeni (2025); Sun et al. (2025); Zhang et al. (2025) can provide better semantic grounding, but falter on generating more complex scenes with correct layouts, due to LLMs' inherent limitations in spatial reasoning. Optimization-based procedural frameworks, such as Infinigen Raistrick et al. (2024) and ProcTHOR Deitke et al. (2022), similarly fail to generate complex and cluttered scenes, and depend on professional tuning in practical use.

To address these challenges, we advocate a fundamentally different approach: instead of individually presenting benchmarks for different scene complexities, we present a fully automated, customizable and user-friendly **benchmark generator**, namely ***InfiniBench***, which can generate a theoretically unlimited number of 3D scenes from parameterized constraints of scene complexity given by users. The basic rationale of InfiniBench design is to combine the strengths of procedural 3D generation in scalability and control with the expressive power of natural language. As shown in Figure 1, InfiniBench takes only a natural language description about the scenes as the input, and translates it into a physically plausible 3D scene that matches the scene complexity specified in the prompt.

To achieve this, we first developed a new LLM-based agentic framework that wraps an advanced optimizer to generate and refine the 3D scene constraints, according to the given prompt of scene

descriptions. The refined scene constraints are then transformed into 3D scenes by a custom optimization engine, which iteratively samples and places objects to satisfy all the semantic and physical rules in the scene constraints. In particular, we replace the conventional method of hierarchical optimization Xu et al. (2002); Yu et al. (2011); Merrell et al. (2011) with a more flexible cluster-based optimization strategy, to allow creating significantly more complex and dense scenes that were previously intractable. In this way, InfiniBench allows broad but effortless customization of the generated 3D scene from multiple perspectives (e.g., number and type of objects, spatial relationships, room layout, etc), without requiring professional knowledge or complicated tuning.

Another gap between 3D scene generation and VLM benchmarking is that outputs of current scene generation methods, typically 3D models in Blender files or point clouds, are not directly consumable by VLMs. To generate image sequences or video streams that VLMs can properly perceive, we developed a new method of camera trajectory optimization, ensuring that all task-relevant objects and their spatial relationship are clearly visible and covered in camera views. Following the principle of frontier-based exploration Yamauchi (1998); Lluvia et al. (2021); Yamauchi (1997), our algorithm incrementally builds the camera path by addressing each task-relevant object in sequence. For each object, it samples and evaluates multiple viewpoints and selects the best one that offers an unobstructed view after passing an occlusion check. A navigable path is then planned to link these viewpoints.

We validate the efficacy of InfiniBench by generating scenes with different scene complexities and comparing the generated scenes with procedural and LLM-based generation methods. Results showed that InfiniBench simultaneously ensures prompt fidelity and physical plausibility, and successfully generates complex and high-density 3D scenes required by user prompts, where other methods fail.

We further showed that customizable benchmarks generated by InfiniBench facilitate in-depth analysis of VLM's failure modes in various spatial reasoning tasks, including measurement (e.g. metric distance estimation), perspective-taking (e.g. object counting) and spatiotemporal tracking (e.g. identification of order identification) Brown et al. (2025). Such customizability in benchmarking would also be a key enabler of enhancing VLM's spatial reasoning capabilities, via spatially grounded training or prompt tuning.

2 RELATED WORK AND MOTIVATION

2.1 VARIABILITY OF SCENE COMPLEXITY

Existing spatial reasoning benchmarks Chen et al. (2024a); Wu et al. (2025); Jia et al. (2025) often categorize their data samples merely by task type or define scene complexity in simple terms, such as the number of rooms Chen et al. (2025b). Such coarse-grained variability conflates multiple dimensions of the scene complexity, and hence obscures the precise reasons of VLM failures in reasoning tasks, which may include an excessive number of distractors (compositional complexity), an unusually intricate object arrangement (relational complexity), or heavily occluded viewpoints (observational complexity). Instead, in InfiniBench, we allowed much more fine-grained variability of scene complexity in these dimensions, by using separate sets of scene parameters (specified by users in natural language) to explicitly represent complexity levels of these dimensions and generating scenes accordingly.

2.2 LAYOUT GENERATION IN 3D SCENE GENERATION

The main focus of InfiniBench design is the creation of scene layout, which defines the placement, orientation, and relationships of objects in the scene. One can leverage the reasoning capabilities of LLMs to directly produce a layout and then instruct a 3D engine to place assets in the scene Ran et al. (2025); Bucher and Armeni (2025); Sun et al. (2025); Zhang et al. (2025). To improve the coherence of generated layouts, some works guide LLMs with structural templates Feng et al. (2023) or represent scenes as graphs to facilitate compositional generation Çelen et al. (2024); Deng et al. (2025); Gao et al. (2024). However, the generated scene's complexity is constrained by current LLMs' flawed spatial logic. As shown in Figure 2, as the number of objects and the intricacy of constraints increase, these methods frequently produce physically implausible layouts, such as those with unrealistic object orientations, out-of-boundary objects or overlapped objects.
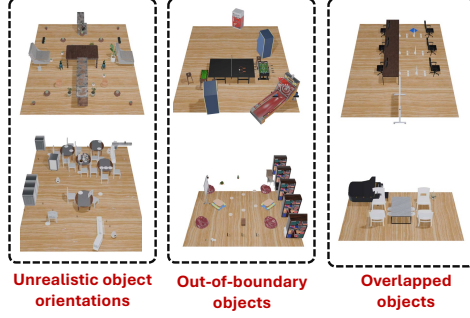
Figure 2: Limitations of LLM-based layout generation

To address this challenge and ensure the scalability of generating complex scenes, we proposed a more robust approach that separates high-level planning from low-level execution. Instead of tasking the LLM to generate the exact scene layout, we use LLM to generate high-level constraints about object arrangement. These constraints are then fed into a dedicated optimizer that generates a physically coherent 3D scene. By using optimization-based procedural generation frameworks Raistrick et al. (2024); Deitke et al. (2022) that are effective in 3D scene synthesis Zhao et al. (2021); Yu et al. (2011; 2015), we designed an improved optimizer for generating complex scenes and wrap this optimizer within an LLM-based agentic framework, making the entire pipeline both powerful and user-friendly.
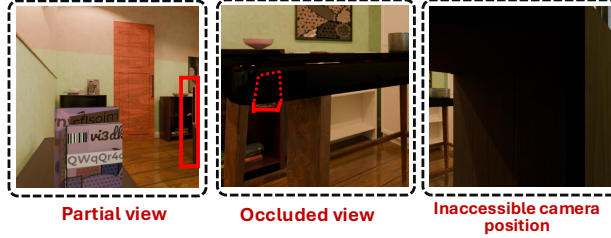

Figure 3: Poorly chosen viewpoints obscure critical information

2.3 Camera Trajectory Generation

Once a 3D scene is generated, creating the 2D visual input for the VLM requires a carefully planned camera trajectory in the scene. The quality of this trajectory is paramount, because a poorly chosen viewpoint can easily obscure critical objects or relationships, rendering a spatial reasoning question unanswerable, as shown in Figure 3.

Optimization-based methods sample numerous camera parameters with respect to a predefined objective, to maximize the object visibility Bares et al. (2000); Halper et al. (2001). While offering fine-grained control, this approach is expensive and struggles with defining an objective function that robustly captures the visual requirements for complex spatial reasoning tasks. Another alternative is to train a generative AI model Courant et al. (2024); Jiang et al. (2021); Wang et al. (2024). However, existing datasets for such training are primarily for aesthetically pleasing and object-centric photo shots, e.g., smooth orbits around a single object, rather than trajectories with controlled coverage of multiple objects. Some other methods Xie et al. (2023); Dehghanian et al. (2025) used reinforcement learning to train an agent for scene navigation without large training data, but needs to retrain the agent for each new scene.

Instead, our approach is inspired by frontier-based exploration Yamauchi (1997; 1998); Lluvia et al. (2021), a lightweight technique widely used in robotic navigation and mapping. In InfiniBench, we redefine frontier as object clusters, and sample viewpoints targeting these frontiers accordingly to strategically manage factors such as object occlusion.

3 Method

As shown in Figure 1, the core of InfiniBench design is a three-stage pipeline, which consists of (1) an LLM-based agentic framework that translates user prompts of scene descriptions into procedural
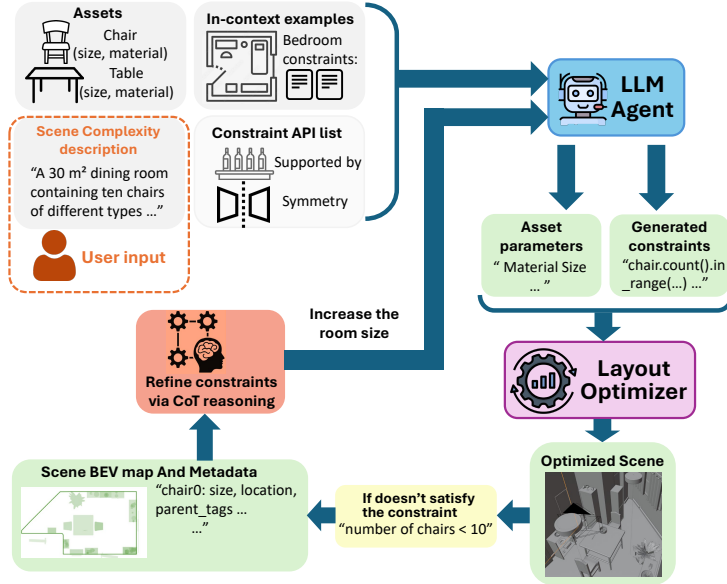
Figure 4: LLM-based agentic framework for iterative scene constraint generation, illustrated by an example of generating a scene of a 30 m^2 dining room with 10 chairs of different types.

constraints through iterative refinement; (2) a cluster-based layout engine that optimizes these constraints into physically plausible 3D scenes; and (3) a task-aware camera trajectory algorithm that renders the 3D scene into an informative video as VLM input.

3.1 AGENTIC GENERATION OF SCENE CONSTRAINTS

A primary barrier to traditional procedural generation of 3D scenes is the need of manually scripting complex scene constraints, a process that requires significant technical expertise, especially when the scene complexity grows. InfiniBench democratizes this process, as shown in Figure 4, by using a LLM agent to translate scene descriptions in natural language into procedural constraints. To ensure that the generated constraints are machine-readable, we teach the LLM agent with domain knowledge of 1) a comprehensive list of available procedural APIs and 2) a set of few-shot examples. The information about procedural APIs covers detailed syntax and functionality with respect to the procedural 3D generation framework being used (e.g., Infinigen Raistrick et al. (2024) and ProcTHOR Deitke et al. (2022)). The few-shot examples demonstrate the translation from high-level user requests (e.g., "a cluttered office desk") to constraint definitions (e.g., `set_object_count(monitor, 3)`, `on_top_of(keyboard, desk)`). More details about such in-context learning are in Appendix A.

A single pass of such scene constraint generation is often insufficient, as the LLM agent may generate constraints that are logically conflicting or physically impossible. For example, when asked to create a "bedroom with three monitors", it may generate constraints that place three monitors all on one desk asset with regular size, which is physically impossible due to the limited surface area of desk.

To address this challenge, InfiniBench iteratively refines the generated scene constraints via a critical feedback loop. In each iteration, we use the scene layout optimizer described in Section 3.2 to attempt realizing the generated constraints into a 3D scene. If the optimizer fails, the returned error report, which includes a bird's eye view (BEV) map showing object collisions and a textual summary describing the unmet constraints, is fed back to the LLM agent to refine the constraint generation. Such refinement enforces a Chain-of-Thought (CoT) reasoning process that prompts the LLM agent to first analyze the failure, propose a solution accordingly, and then implement the changes on scene constraints. For example, if the error report indicates that only one monitor is placed on the desk, the CoT reasoning process would infer the reason of failure as insufficient surface area, and the scene constraints will be accordingly revised to enforce larger parameters of the desk asset. More details about this CoT reasoning are in Appendix B.
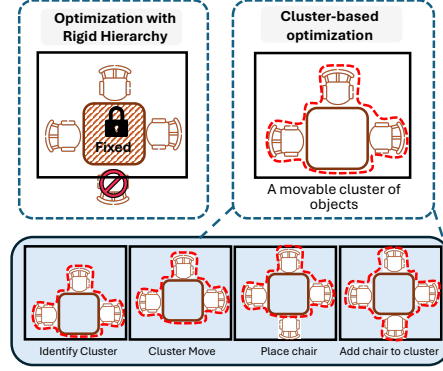
Figure 5: Traditional hierarchical optimization vs. our cluster-based optimization

3.2 Layout Optimization for Complex Scenes

When the scene complexity grows, hierarchical optimization methods used in the existing procedural generation frameworks such as Infinigen Raistrick et al. (2024) and ProcTHOR Deitke et al. (2022) would fail, because they first fix the positions of large objects (e.g. desks and beds) and usually result in unsolvable states where there is no space for smaller objects (e.g. chairs around the desk), as shown in Figure 6(a).

To ensure robust layout optimization with high scene complexity, we restructured the layout engine with a flexible cluster-based optimization strategy. This strategy, as shown in Figure 5, builds on a new concept of **movable cluster**, defined as a dynamic group of related objects (e.g., a table and chairs surrounding it) that are treated as a single entity in optimization. The optimization process, then, consists of the following three steps:

1. **Identifying clusters:** The scene's semantic graph is parsed to automatically group closely related objects into clusters, each of which consists of a large "parent" object and a number of small "child" objects.

2. **Augmenting the action space:** The optimizer's action space is expanded with cluster-level operations, which allow an entire cluster to move to a better location without breaking its internal relationships among objects.

3. **Collision checking:** Collision checks are performed using each cluster's collective bounding box.

This flexible and cluster-based approach allows the optimizer to navigate a much larger solution space and reposition the movable cluster for a valid layout, making the generation of previously intractable high-density scenes feasible, as shown in Figure 6(b). More details about such cluster operations are in Appendix C, and more examples of InfiniBench's generated scenes are in Appendix D.
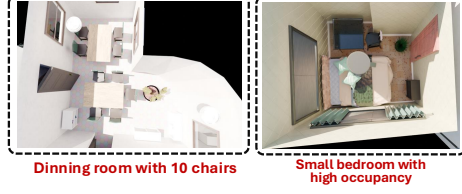
3.3 Camera Trajectory Optimization

Given a generated 3D scene and a spatial reasoning task, the objective of our camera trajectory optimization in InfiniBench is to generate the shortest possible camera path that provides a clear, full and unobstructed view of every task-relevant object. Our method is inspired by frontier-based exploration Yamauchi (1997; 1998); Lluvia et al. (2021), a classical robotic navigation technique. Unlike traditional approaches that explore frontiers using occupancy grids, we define "frontiers" as the set of unvisited target objects. We then develop a viewpoint sampling method to find optimal views for these targets while avoiding occlusion. As shown in Figure 7, this process consists of the following steps:

1. **Target selection:** From the camera's current position, we identify the closest "unvisited" target object.

2. **Viewpoint sampling and selection:** A set of candidate viewpoints is sampled around the target. Each viewpoint is scored based on three criteria: 1) if the camera location is valid, 2) if the object fully in the view? and 3) if there is occlusion between the camera location and the object. The viewpoint with the highest score is selected.
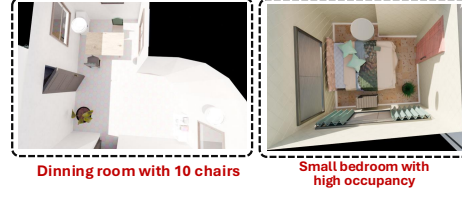
(a) Scenes generated with traditional hierarchical optimization

(b) Scenes generated by our cluster-based optimization

Figure 6: Comparison of generating highly complex scenes, with the same room shape and asset parameters
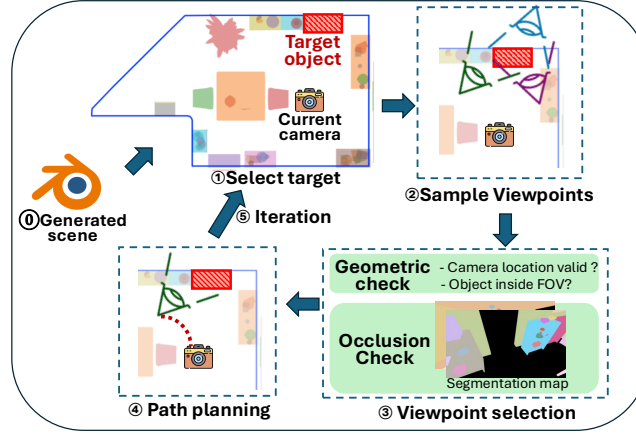
Figure 7: Camera trajectory optimization

3. **Path planning:** A collision-free path is computed from the current camera position to the selected viewpoint using Dijkstra's algorithm in 3D space.

4. **Iteration:** The camera moves to the new viewpoint, the object is marked "visited", and the optimization repeats until all target objects are visited.

After such optimization, we follow the optimized camera trajectory to render the scene into video frames, as exemplified in Figure 8. More details are in Appendix E.

Figure 8: An example of optimized camera trajectory and the corresponding video frames

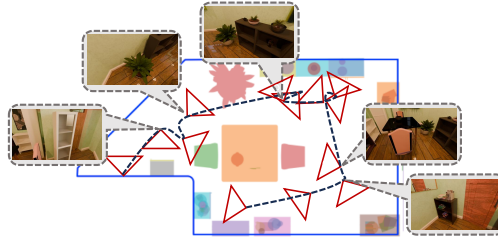|	Low amount of objects					Medium amount of objects					High amount of objects				
Method	Fidelity ↑	CLIP ↑	Realism ↑	#OB ↓	#CN ↓	Fidelity ↑	CLIP ↑	Realism ↑	#OB ↓	#CN ↓	Fidelity ↑	CLIP ↑	Realism ↑	#OB ↓	#CN ↓
I-Design Çelen et al. (2024)	**0.95**	28.4	0.72	1.6	0.4	0.92	27.9	0.68	3.4	5.1	0.90	27.1	0.61	6.9	10.3
Holodeck Yang et al. (2024)	0.97	30.1	0.81	1.9	2.1	**0.93**	29.5	0.77	4.6	3.1	0.88	28.8	0.71	7.7	9.4
LayoutGPT Feng et al. (2023)	0.96	29.8	0.79	1.3	2.4	**0.95**	29.1	0.74	3.7	4.6	0.93	28.3	0.67	4.5	13.5
Luminous Zhao et al. (2021)	0.87	27.5	0.75	**0.0**	**0.0**	0.84	26.8	0.70	**0.0**	0.1	0.42	26.2	0.63	**0.0**	**0.0**
Infinigen Raistrick et al. (2024)	0.92	31.0	0.88	**0.0**	0.1	0.87	30.4	0.84	**0.0**	0.2	0.64	29.7	0.79	0.0	0.2
InfiniBench	**0.98**	**31.8**	**0.93**	**0.0**	**0.0**	**0.95**	**31.5**	**0.89**	**0.0**	**0.0**	**0.98**	29.9	**0.81**	0.1	**0.0**

Table 1: Quantitative comparison of scene generation quality with different numbers of objects in the scene

	Low Occupancy					Medium Occupancy					High Occupancy				
Method	Fidelity ↑	CLIP ↑	Realism ↑	#OB ↓	#CN ↓	Fidelity ↑	CLIP ↑	Realism ↑	#OB ↓	#CN ↓	Fidelity ↑	CLIP ↑	Realism ↑	#OB ↓	#CN ↓
I-Design Çelen et al. (2024)	0.90	28.6	0.73	0.1	0.4	0.87	28.0	0.67	1.3	3.1	0.79	27.2	0.59	5.3	6.4
Holodeck Yang et al. (2024)	0.78	30.4	0.82	0.3	0.2	0.80	29.7	0.76	0.9	5.4	0.75	28.9	0.69	3.1	8.7
LayoutGPT Feng et al. (2023)	0.81	29.9	0.80	0.9	0.2	0.83	29.3	0.73	1.7	6.8	0.85	28.5	0.65	5.6	9.6
Luminous Zhao et al. (2021)	0.74	27.8	0.76	**0.0**	**0.0**	0.64	27.1	0.69	**0.0**	**0.0**	0.43	26.4	0.61	**0.0**	**0.0**
Infinigen Raistrick et al. (2024)	0.83	31.2	0.89	**0.0**	**0.0**	0.71	30.6	0.85	0.2	0.1	0.49	29.9	0.78	0.1	0.2
InfiniBench	**0.94**	**32.1**	**0.94**	**0.0**	**0.0**	**0.89**	**30.8**	**0.86**	**0.0**	0.1	**0.91**	**30.2**	**0.80**	**0.0**	0.1

Table 2: Quantitative comparison of scene generation quality with different levels of scene occupancy ratio

4 EVALUATION OF INFINIBENCH

We evaluate InfiniBench to answer two primary questions: 1) Given the required scene complexity, can InfiniBench generate physically plausible 3D scenes that are superior to those from existing baseline methods, in terms of correctness and quality? 2) How effective are different components of InfiniBench in such generation?

4.1 EXPERIMENTAL SETUP

Implementation. We use Gemini-2.5-Pro Comanici et al. (2025) in the agentic framework for scene constraint generation, and our layout optimizer is built upon Infinigen Raistrick et al. (2024), leveraging its rich and robust asset library. For camera trajectory generation and final 3D rendering, we enforce a pipeline with Trimesh Dawson-Haggerty et al. for geometric processing, PyRender Matl (2018) for fast occlusion checks, and Blender's Cycles engine Blender Foundation (2025) for high-fidelity video output. More details are in Appendix F.

Baselines. We compare InfiniBench with two categories of 3D scene generation methods. 1) *Procedural Frameworks* including Infinigen Raistrick et al. (2024) (using its original optimizer) and Luminous Zhao et al. (2021); and 2) *LLM-based Layout Generators* including LayoutGPT Feng et al. (2023), Holodeck Yang et al. (2024) and I-Design Çelen et al. (2024), which directly generate scene layouts from text. For fair comparisons, the same 3D asset library (provided by Infinigen) is used in all experiments.

Evaluation Metrics. We focus on two challenging aspects of scene complexity: the total number of objects and the scene occupancy ratio. We evaluate the generated scenes using the following metrics:

- *Prompt fidelity*, which measures the alignment between the generated scene and text prompt. It is calculated as the accuracy of the generated scene's object count and occupancy ratio against ground truth values in text prompt.
- *Text-image alignment*, which measure the semantic alignment as the CLIP score Radford et al. (2021) between the text prompt and the rendered top-down image of generated scene.
- *Layout realism*, scored by a GPT-5 evaluator as suggested by Scenethesis Ling et al. (2025), incorporating key factors such as spatial coherence and functional plausibility.
- *Physical plausibility*, measured by the number of physically impossible artifacts, especially the count of out-of-boundary objects (#OB) and pairs of collided objects (#CN), as suggested by SceneWeaver Yang et al. (2025c).

4.2 MAIN RESULTS

Quantitative Results. Tables 1 and 2 show that as the scene complexity increases, current generation methods exhibit distinct tradeoffs among different evaluation metrics. LLM-based generation methods reach the required complexity constraints and maintain relatively high Fidelity at the cost of lower physical plausibility, as evidenced by an increase in #OB and #CN. For instance, with a high amount of objects in the scene, LayoutGPT's collision count rises to 13.5, 5.6x higher than
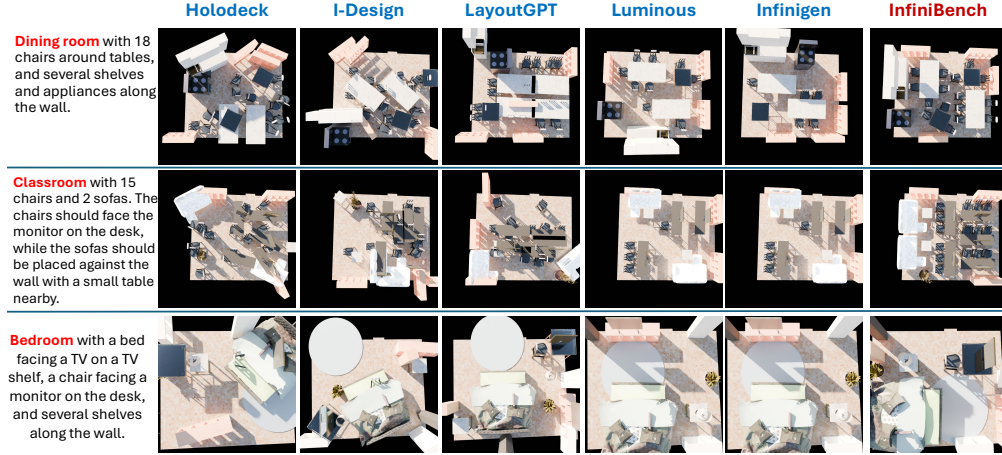
Figure 9: Qualitative comparisons of the scenes generated by InfiniBench and baseline methods

that with a low amount of objects. Conversely, procedural generation frameworks excel at maintaining physical plausibility, but have difficulty in adhering to the prompt's quantitative requirements of scene complexity. As such required scene complexity increases, their Fidelity scores drop drastically (e.g., 0.42 and 0.64 for Luminous and Infinigen, respectively), indicating a failure of generating the required number of objects.

InfiniBench successfully avoids such tradeoff. It consistently achieves high Fidelity comparable to the LLM-based approaches, and simultaneously ensures near-perfect physical plausibility that is on par with procedural methods. In the most challenging high-complexity settings, InfiniBench reaches top performance in all metrics, including the highest CLIP and Realism scores, demonstrating its superiority of generating both complex and physically coherent scenes.

Qualitative Results. Figure 9 provides qualitative comparisons of the scenes generated by InfiniBench and baseline methods. The scenes generated by baseline methods often fail in predictable ways when the required scene complexity high: the generated objects could be haphazardly placed, frequently intersecting, or fail to appear at all. In contrast, InfiniBench produces coherent, cluttered and physically plausible scenes that faithfully reflect the user's detailed request. This visual evidence underscores the effectiveness of our iterative refinement and cluster-based optimization pipeline. More examples are in Appendix G.

Ablation Study. To validate the contribution of each component in InfiniBench, we conduct an ablation study. We start with the base optimizer provided in InfiniGen, and compare it against with *constraint-refinement only*, *cluster-optimization only*, and the full InfiniBench that includes both agentic framework for constraint refinement and cluster-based layout optimizer. Results in Table 3 show that, even without the cluster-based layout optimizer, *constraint-refinement only* provides a moderate boost to Fidelity ($0.64 \rightarrow 0.71$), demonstrating its effectiveness in better following the text prompt. *Cluster-optimization only* yields a slighter improvement in Fidelity (to 0.68) but also marginally increases Realism. Combining both components results in a dramatic and synergistic improvement.

	Fidelity	Clip	Realism	#OB	#CN
Base	0.64	29.7	0.79	0.0	0.0
Constraint-refinement only	0.71	28.9	0.79	0.0	0.0
Cluster-optimization only	0.68	29.9	0.81	0.0	0.1
Full InfiniBench	0.92	29.9	0.81	0.2	0.1

Table 3: Ablation study of different components in InfiniBench

Further, we also evaluated how iterative refinement of scene constraints (Section 3.1) affects scene generation. As shown in Table 4, the refinement process typically converges within 5 iterations. While our method does require iterative reasoning, the resulting constraints are highly reusable. The layout optimizer can leverage these constraints to generate a large number of scenes, hence ensuring the high compute efficiency in layout generation.

9

	Fidelity	Clip	Realism	#OB	#CN
1 iteration	0.68	29.9	0.81	0.1	0.2
2 iteration	0.72	29.8	0.82	0.1	0.1
3 iteration	0.86	29.9	0.80	0.0	0.2
5 iterations	0.92	29.9	0.81	0.1	0.1
10 iterations	0.92	29.8	0.81	0.2	0.1

Table 4: Ablation study of how the number of iterations in constraint refinement affects the scene quality

Amount of irrelevant objects	Measurement task			Perspective-taking task			Spatiotemporal task		
	Low	Medium	High	Low	Medium	High	Low	Medium	High
Gemini-2.5-pro	69.2	68.9	66.4	70.6	67.2	66.9	87.9	70.1	56.2
GPT5	45.8	41.2	41.3	57.3	55.5	54.1	47.8	31.3	26.7
LLaVA-Video-7B	47.2	45.2	45.0	56.0	56.1	54.2	50.2	36.2	29.2
InternVL3.5-8B	48.4	48.2	47.8	53.0	52.9	52.1	79.8	64.9	47.0
Cambrian-S-7B	57.2	57.0	56.9	69.1	71.3	65.4	85.1	72.4	49.2

Table 5: Performance of Measurement, Perspective-taking and Spatiotemporal tasks with different numbers of irrelevant objects

5 USING INFINIBENCH TO DIAGNOSE VLMs' FAILURES IN SPATIAL REASONING

To further verify the usefulness of InfiniBench in diagnosing VLMs' abilities and failure modes in spatial reasoning with high complexity, we apply the benchmarks generated by InfiniBench with different levels of compositional, relational and observational complexities to the following set of representative spatial reasoning tasks Brown et al. (2025). We examine where and why VLM's spatial reasoning succeeds or fails, with two close-sourced models (Gemini-2.5-Pro Comanici et al. (2025), GPT-5 ope (2025)) and three open-sourced models (LLaVa-Video-7B Zhang et al. (2024), InternVL3.5-8B Wang et al. (2025), Cambrian-S-7B Yang et al. (2025b)).

Measurement tasks that probe VLM's ability of reasoning about metric properties and object scale. We generate tasks that require both dimensional comparison and fine-grained distance estimation based on complex referring expressions, such as *Q: What's the [Height] of the [Object_Name] with the texture of [Material_name] and [Relation to a nearby object]*. Performance will be measured by mean relative accuracy between numerical answer and ground truth Yang et al. (2025a).

Perspective-taking tasks that assess compositional understanding from a given viewpoint. We choose object counting tasks with various numbers of irrelevant objects, such as *Q: How many [Object_Name] are in the video*. Performance will be similarly measured by mean relative accuracy Yang et al. (2025a).

Spatiotemporal tasks that test the VLM's ability of reasoning over time and space. We generate tasks of identifying the appearance order of objects that demand temporal memory and ability of tracking objects across a dynamic camera trajectory, such as *Q: What's the appearance order of [Obj_A_description], [Obj_B_description], and [Obj_C_description]*. Tasks are multi-choice questions, and performance is measured by exact match accuracy.

5.1 VLM'S SPATIAL REASONING WITH VARYING COMPOSITIONAL SCENE COMPLEXITY

We constructed benchmarks with the number of objects from 5 to 50, as shown in Figure 10. Results in Figure 11 showed that the reasoning performance of VLMs declines markedly with more objects, revealing a shared difficulty in managing more complicated visual clutter. Gemini-2.5-Pro and Cambrian-S-7B achieve higher accuracy compared to other models, as the scene complexity grows. We hypothesize that the performance degradation with high compositional scene complexity stems from repetitive counting, since the same object could appears multiple times in the input video, an effect reflected in the models' tendency to overestimate counts relative to the ground truth.
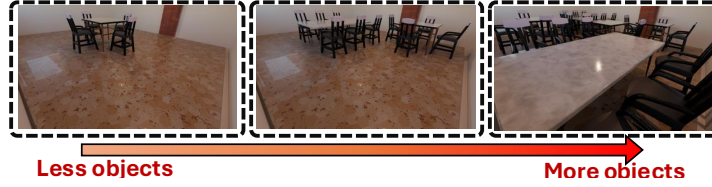
Figure 10: Examples of benchmark samples with varying compositional complexity (i.e., the number of objects)
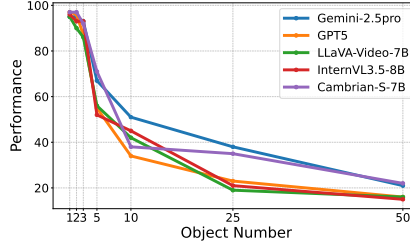

Figure 11: VLMs' performance of spatial reasoning with varying compositional scene complexity

Figure 12: Examples of benchmark samples with varying relational complexity (i.e., the amount of irrelevant objects)

5.2 VLM'S SPATIAL REASONING WITH VARYING RELATIONAL SCENE COMPLEXITY

We investigated how VLM's reasoning is influenced by irrelevant objects, which increase the complexity of spatial relations in the scene, as shown in Figure 12. Results in Table 5 show that increasing the number of irrelevant objects slightly reduces model performance. By examining the models' reasoning paths, we found that this degradation is likely caused by incorrect object referencing.

Figure 13: Examples of benchmark samples with varying observational complexity (i.e., different camera perspectives)

5.3 VLM'S SPATIAL REASONING WITH VARYING OBSERVATIONAL SCENE COMPLEXITY

We examined the impact of camera perspective on VLM performance. As shown in Figure 13, we generated two video variants for each scene: a bird's-eye view captured from a higher camera position (2.5 m) and an egocentric view from a lower perspective (1 m). Results in Table 6 show that, for Perspective-taking and Spatiotemporal tasks, the model performance improves substantially as the viewpoint becomes closer to the bird's-eye view. This enhancement may result from the fact that higher perspectives provide a clearer and more comprehensive view of the scene's spatial layout. On the other hand, negligible differences are noticed on Measurement tasks.

		Measure	Perspect	Spatiotemporal
Gemini-2.5	BEV	68.1	77.4	73.2
	Ego	68.9	67.2	70.1
GPT5	BEV	42.0	69.1	49.0
	Ego	41.2	55.5	31.3
LLaVA-Video-7B	BEV	45.0	68.3	47.1
	Ego	45.2	56.1	36.2
InternVL3.5-8B	BEV	47.6	62.7	68.1
	Ego	48.2	52.9	64.9
Cambrian-S-7B	BEV	57.5	76.2	78.8
	Ego	57.0	71.3	72.4

Table 6: VLM's reasoning performance with varying observational scene complexity

6 CONCLUSION

In this paper, we introduced InfiniBench, a fully automated benchmark generator that creates complex and physically plausible 3D scenes with customizable scene complexity. By enabling the systematic variation of scene properties, InfiniBench facilitates fine-grained and diagnostic analysis of VLM failures in spatial reasoning, paving the way for developing more robust and spatially-aware VLMs

REFERENCES

Technical report, OpenAI, August 2025. URL https://cdn.openai.com/gpt-5-system-card.pdf.

J. Bai, S. Bai, Y. Chu, Z. Cui, K. Dang, X. Deng, Y. Fan, W. Ge, Y. Han, F. Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

F. M. Barbosa and F. S. Osório. A threefold review on deep semantic segmentation: Efficiency-oriented, temporal and depth-aware design, 2023. URL https://arxiv.org/abs/2303.04315.

W. Bares, S. McDermott, C. Boudreaux, and S. Thainimit. Virtual 3d camera composition from frame constraints. In *Proceedings of the eighth ACM international conference on Multimedia*, pages 177–186, 2000.

Blender Foundation. Blender, 2025. URL https://www.blender.org. Open-source 3D creation suite.

E. Brown, A. Ray, R. Krishna, R. Girshick, R. Fergus, and S. Xie. Sims-v: Simulated instruction-tuning for spatial video understanding. *arXiv preprint arXiv:2511.04668*, 2025.

M. J. Bucher and I. Armeni. Respace: Text-driven 3d scene synthesis and editing with preference alignment. *arXiv preprint arXiv:2506.02459*, 2025.

A. Çelen, G. Han, K. Schindler, L. Van Gool, I. Armeni, A. Obukhov, and X. Wang. I-design: Personalized llm interior designer. In *European Conference on Computer Vision*, pages 217–234. Springer, 2024.

B. Chen, Z. Xu, S. Kirmani, B. Ichter, D. Sadigh, L. Guibas, and F. Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14455–14465, 2024a.

S. Chen, T. Zhu, R. Zhou, J. Zhang, S. Gao, J. C. Niebles, M. Geva, J. He, J. Wu, and M. Li. Why is spatial reasoning hard for vlms? an attention mechanism perspective on focus areas. *arXiv preprint arXiv:2503.01773*, 2025a.

X. Chen, T. Wang, Q. Li, T. Huang, J. Pang, and T. Xue. Gleam: Learning generalizable exploration policy for active mapping in complex 3d indoor scenes. *arXiv preprint arXiv:2505.20294*, 2025b.

Z. Chen, J. Wu, W. Wang, W. Su, G. Chen, S. Xing, M. Zhong, Q. Zhang, X. Zhu, L. Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24185–24198, 2024b.

G. Comanici, E. Bieber, M. Schaekermann, I. Pasupat, N. Sachdeva, I. Dhillon, M. Blistein, O. Ram, D. Zhang, E. Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

R. Courant, N. Dufour, X. Wang, M. Christie, and V. Kalogeiton. Et the exceptional trajectories: Text-to-camera-trajectory generation with character awareness. In *European Conference on Computer Vision*, pages 464–480. Springer, 2024.

G. Csurka, R. Volpi, B. Chidlovskii, et al. Semantic image segmentation: Two decades of research. *Foundations and Trends® in Computer Graphics and Vision*, 14(1-2):1–162, 2022.

Dawson-Haggerty et al. trimesh. URL https://trimesh.org/.

Z. Dehghanian, P. Ardekhani, A. Vahedi, H. Beigy, and H. R. Rabiee. Camera trajectory generation: A comprehensive survey of methods, metrics, and future directions. *arXiv preprint arXiv:2506.00974*, 2025.

M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, K. Ehsani, J. Salvador, W. Han, E. Kolve, A. Kembhavi, and R. Mottaghi. Procthor: Large-scale embodied ai using procedural generation. *Advances in Neural Information Processing Systems*, 35:5982–5994, 2022.

W. Deng, M. Qi, and H. Ma. Global-local tree search in vlms for 3d indoor scene generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 8975–8984, 2025.

J. R. Devadason et al. A comprehensive survey of image compression methods: from prediction models to advanced techniques. *Multimedia Tools and Applications*, pages 1–39, 2025.

P. Engstler, A. Shtedritski, I. Laina, C. Rupprecht, and A. Vedaldi. Syncity: Training-free generation of 3d worlds. *arXiv preprint arXiv:2503.16420*, 2025.

W. Feng, W. Zhu, T.-j. Fu, V. Jampani, A. Akula, X. He, S. Basu, X. E. Wang, and W. Y. Wang. Layoutgpt: Compositional visual planning and generation with large language models. *Advances in Neural Information Processing Systems*, 36:18225–18250, 2023.

G. Gao, W. Liu, A. Chen, A. Geiger, and B. Schölkopf. Graphdreamer: Compositional 3d scene synthesis from scene graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21295–21304, 2024.

T. Ghandi, H. Pourreza, and H. Mahyar. Deep learning approaches on image captioning: A review. *ACM Computing Surveys*, 56(3):1–39, 2023.

H. Go, B. Park, J. Jang, J.-Y. Kim, S. Kwon, and C. Kim. Splatflow: Multi-view rectified flow model for 3d gaussian splatting synthesis. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21524–21536, 2025a.

H. Go, B. Park, H. Nam, B.-H. Kim, H. Chung, and C. Kim. Videorfsplat: Direct scene-level text-to-3d gaussian splatting generation with flexible pose and multi-view joint modeling. *arXiv preprint arXiv:2503.15855*, 2025b.

M. R. Greene. Scene perception and understanding. In *Oxford Research Encyclopedia of Psychology*. 2023.

N. Halper, R. Helbing, and T. Strothotte. A camera engine for computer games: Managing the trade-off between constraint satisfaction and frame coherence. In *Computer Graphics Forum*, volume 20, pages 174–183. Wiley Online Library, 2001.

Y. Hong, C. Lin, Y. Du, Z. Chen, J. B. Tenenbaum, and C. Gan. 3d concept learning and reasoning from multi-view images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9202–9212, 2023.

X. Hu, Z. Gan, J. Wang, Z. Yang, Z. Liu, Y. Lu, and L. Wang. Scaling up vision-language pre-training for image captioning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 17980–17989, 2022.

A. Hurst, A. Lerer, A. P. Goucher, A. Perelman, A. Ramesh, A. Clark, A. Ostrow, A. Welihinda, A. Hayes, A. Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

isaac sim. Isaacsim, n.d. URL `https://github.com/isaac-sim/IsaacSim`. Open-source robotics simulation on NVIDIA Omniverse.

M. Jia, Z. Qi, S. Zhang, W. Zhang, X. Yu, J. He, H. Wang, and L. Yi. Omnispatial: Towards comprehensive spatial reasoning benchmark for vision language models. *arXiv preprint arXiv:2506.03135*, 2025.

H. Jiang, M. Christie, X. Wang, L. Liu, B. Wang, and B. Chen. Camera keyframing with style and control. *ACM Transactions on Graphics (TOG)*, 40(6):1–13, 2021.

S. Li, C. Yang, J. Fang, T. Yi, J. Lu, J. Cen, L. Xie, W. Shen, and Q. Tian. Worldgrow: Generating infinite 3d world. *arXiv preprint arXiv:2510.21682*, 2025.

L. Ling, C.-H. Lin, T.-Y. Lin, Y. Ding, Y. Zeng, Y. Sheng, Y. Ge, M.-Y. Liu, A. Bera, and Z. Li. Scenethesis: A language and vision agentic framework for 3d scene generation. *arXiv preprint arXiv:2505.02836*, 2025.

Y. Liu, X. Li, X. Li, L. Qi, C. Li, and M.-H. Yang. Pyramid diffusion for fine 3d large scene generation. In *European Conference on Computer Vision*, pages 71–87. Springer, 2024.

I. Lluvia, E. Lazkano, and A. Ansuategi. Active mapping and robot exploration: A survey. *Sensors*, 21(7):2445, 2021.

M. Matl. *Pyrender Documentation*. Read the Docs, 2018. URL `https://pyrender.readthedocs.io/en/latest/`. Version 0.1.45.

Q. Meng, L. Li, M. Nießner, and A. Dai. Lt3sd: Latent trees for 3d scene diffusion. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 650–660, 2025.

P. Merrell, E. Schkufza, Z. Li, M. Agrawala, and V. Koltun. Interactive furniture layout using interior design guidelines. *ACM transactions on graphics (TOG)*, 30(4):1–10, 2011.

A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

A. Raistrick, L. Mei, K. Kayan, D. Yan, Y. Zuo, B. Han, H. Wen, M. Parakh, S. Alexandropoulos, L. Lipson, et al. Infinigen indoors: Photorealistic indoor scenes using procedural generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21783–21794, 2024.

X. Ran, Y. Li, L. Xu, M. Yu, and B. Dai. Direct numerical layout generation for 3d indoor scene synthesis via spatial reasoning. *arXiv preprint arXiv:2506.05341*, 2025.

I. Stogiannidis, S. McDonagh, and S. A. Tsaftaris. Mind the gap: Benchmarking spatial reasoning in vision-language models. *arXiv preprint arXiv:2503.19707*, 2025.

F.-Y. Sun, W. Liu, S. Gu, D. Lim, G. Bhat, F. Tombari, M. Li, N. Haber, and J. Wu. Layoutvlm: Differentiable optimization of 3d layout via vision-language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 29469–29478, 2025.

W. Wang, Z. Gao, L. Gu, H. Pu, L. Cui, X. Wei, Z. Liu, L. Jing, S. Ye, J. Shao, et al. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

X. Wang, W. Ma, Z. Li, A. Kortylewski, and A. L. Yuille. 3d-aware visual question answering about parts, poses and occlusions. *Advances in Neural Information Processing Systems*, 36:58717–58735, 2023.

Z. Wang, Z. Yuan, X. Wang, Y. Li, T. Chen, M. Xia, P. Luo, and Y. Shan. Motionctrl: A unified and flexible motion controller for video generation. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024.

D. Wu, F. Liu, Y.-H. Hung, and Y. Duan. Spatial-mllm: Boosting mllm capabilities in visual-based spatial intelligence. *arXiv preprint arXiv:2505.23747*, 2025.

C. Xia, S. Zhang, F. Liu, C. Liu, K. Hirunyaratsameewong, and Y. Duan. Scenepainter: Semantically consistent perpetual 3d scene generation with concept relation alignment. *arXiv preprint arXiv:2507.19058*, 2025.

J. Xiang, Z. Lv, S. Xu, Y. Deng, R. Wang, B. Zhang, D. Chen, X. Tong, and J. Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 21469–21480, 2025.

D. Xie, P. Hu, X. Sun, S. Pirk, J. Zhang, R. Mech, and A. E. Kaufman. Gait: Generating aesthetic indoor tours with deep reinforcement learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7409–7419, 2023.

K. Xu, J. Stewart, and E. Fiume. Constraint-based automatic placement for scene composition. In *Graphics Interface*, volume 2, pages 25–34, 2002.

B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pages 146–151. IEEE, 1997.

B. Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pages 47–53, 1998.

J. Yang, S. Yang, A. W. Gupta, R. Han, L. Fei-Fei, and S. Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 10632–10643, 2025a.

S. Yang, J. Yang, P. Huang, E. Brown, et al. Cambrian-s: Towards spatial supersensing in video. *arXiv preprint arXiv:2025*, 2025b.

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

Y. Yang, B. Jia, S. Zhang, and S. Huang. Sceneweaver: All-in-one 3d scene synthesis with an extensible and self-reflective agent. *arXiv preprint arXiv:2509.20414*, 2025c.

Y. Yang, J. Shao, X. Li, Y. Shen, A. Geiger, and Y. Liao. Prometheus: 3d-aware latent diffusion models for feed-forward text-to-3d scene generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 2857–2869, 2025d.

Z. Yin, J. Wang, J. Cao, Z. Shi, D. Liu, M. Li, X. Huang, Z. Wang, L. Sheng, L. Bai, et al. Lamm: Language-assisted multi-modal instruction-tuning dataset, framework, and benchmark. *Advances in Neural Information Processing Systems*, 36:26650–26685, 2023.

L.-F. Yu, S. K. Yeung, C.-K. Tang, D. Terzopoulos, T. F. Chan, and S. J. Osher. Make it home: automatic optimization of furniture arrangement. *ACM Trans. Graph.*, 30(4):86, 2011.

L.-F. Yu, S.-K. Yeung, and D. Terzopoulos. The clutterpalette: An interactive tool for detailing indoor scenes. *IEEE transactions on visualization and computer graphics*, 22(2):1138–1148, 2015.

S. Yu, Y. Chen, H. Ju, L. Jia, F. Zhang, S. Huang, Y. Wu, R. Cui, B. Ran, Z. Zhang, et al. How far are vlms from visual spatial intelligence? a benchmark-driven perspective. *arXiv preprint arXiv:2509.18905*, 2025.

Y. Zhang, J. Wu, W. Li, B. Li, Z. Ma, Z. Liu, and C. Li. Video instruction tuning with synthetic data, 2024. URL `https://arxiv.org/abs/2410.02713`.

Y. Zhang, Z. Li, M. Zhou, S. Wu, and J. Wu. The scene language: Representing scenes with programs, words, and embeddings. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 24625–24634, 2025.

Y. Zhao, K. Lin, Z. Jia, Q. Gao, G. Thattai, J. Thomason, and G. S. Sukhatme. Luminous: Indoor scene generation for embodied ai challenges. *arXiv preprint arXiv:2111.05527*, 2021.

Appendix: InfiniBench Supplementary Material

A   IN-CONTEXT LEARNING FOR SCENE CONSTRAINT GENERATION

In our design of InfiniBench, the LLM takes the following inputs: the user's scene description in natural language, the given descriptions about the list of available assets and constraint APIs (determined by the specific procedural generation framework being used). Besides, to ensure the generated scene constraints are machine-readable and physically plausible, in-context examples are also provided to the LLM to demonstrate the specific syntax of the underlying constraint language used in the procedural generation framework. In practice, we use the default home constraints which are manually crafted as the in-context examples. Some more specific examples, with respect to the constraint APIs and syntax being used in the Infinigen Raistrick et al. (2024) framework, are provided below.

A.1   EXAMPLE 1: DINING ROOMS

The following code block demonstrates how to define scene constraints for a dining room, including furniture relationships (chairs around tables) and clutter distribution (dishware and utensils).

```
 1  # region DININGROOMS
 2
 3  diningtables = furniture[Semantics.Table][tables.TableDiningFactory]
 4  diningchairs = furniture[Semantics.Chair][seating.ChairFactory]
 5  constraints["dining_chairs"] = rooms.all(
 6      lambda r: (
 7          diningtables.related_to(r).all(
 8              lambda t: (
 9                  diningchairs.related_to(r)
10                  .related_to(t, cu.front_against)
11                  .count()
12                  .in_range(3, 6)
13              )
14          )
15      )
16  )
17
18  score_terms["dining_chairs"] = rooms.all(
19      lambda r: (
20          diningchairs.related_to(r).count().maximize(weight=5)
21          + diningchairs.related_to(r)
22          .mean(lambda t: t.distance(diningchairs.related_to(r)))
23          .maximize(weight=3)
24          # cl.reflectional_asymmetry(diningchairs.related_to(r), diningtables.related_to(r)).
25      minimize(weight=1)
26          # cl.rotational_asymmetry(diningchairs.related_to(r)).minimize(weight=1)
27      )
28  )
29
30  constraints["dining_table_objects"] = rooms.all(
31      lambda r: (
32          diningtables.related_to(r).all(
33              lambda t: (
34                  obj[Semantics.TableDisplayItem]
35                  .related_to(t, cu.ontop)
36                  .count()
37                  .in_range(0, 2)
38                  * (obj[Semantics.Utensils].related_to(t, cu.ontop).count() >= 0)
39                  * (
40                      obj[Semantics.Dishware]
41                      .related_to(t, cu.ontop)
42                      .count()
43                      .in_range(0, 2)
44                  )
45              )
46          )
47      )
48  )
49
50  score_terms["dining_table_objects"] = rooms.mean(
51      lambda r: (
52          cl.center_stable_surface_dist(
53              obj[Semantics.TableDisplayItem].related_to(
54                  diningtables.related_to(r), cu.ontop
```

```
54              )
55          ).minimize(weight=1)
56      )
57 )
58
59 diningrooms = rooms[Semantics.DiningRoom].excludes(cu.room_types)
60 constraints["diningroom"] = diningrooms.all(
61     lambda r: (
62         (diningtables.related_to(r).count() == 1)
63         * storage.related_to(r).all(
64             lambda t: (
65                 (obj[Semantics.Dishware].related_to(t, cu.on).count() >= 0)
66                 * (
67                     obj[Semantics.OfficeShelfItem]
68                     .related_to(t, cu.on)
69                     .count()
70                     .in_range(0, 5)
71                 )
72             )
73         )
74     )
75 )
76 score_terms["diningroom"] = diningrooms.mean(
77     lambda r: (
78         diningtables.related_to(r).distance(r, cu.walltags).maximize(weight=10)
79         + cl.angle_alignment_cost(
80             diningtables.related_to(r), r, cu.walltags
81         ).minimize(weight=10)
82         + cl.center_stable_surface_dist(diningtables.related_to(r)).minimize(
83             weight=1
84         )
85     )
86 )
87 # endregion
```

Listing 1: An in-context example for Dining Room constraints provided to the LLM agent.

A.2   EXAMPLE 2: BATHROOMS

The following example illustrates the logical constraints for bathroom layouts, specifically enforcing the presence of essential hardware (toilets, sinks) and their spatial accessibility.

```
1 # region BATHROOMS
2 bathrooms = rooms[Semantics.Bathroom].excludes(cu.room_types)
3
4 toilet = wallfurn[bathroom.ToiletFactory]
5 bathtub = wallfurn[bathroom.BathtubFactory]
6 sink = wallfurn[bathroom.StandingSinkFactory]
7
8 hardware = obj[bathroom.HardwareFactory].related_to(bathrooms, cu.against_wall)
9
10 constraints["bathroom"] = bathrooms.all(
11     lambda r: (
12         mirror.related_to(r).related_to(r, cu.flush_wall).count().equals(1)
13         * sink.related_to(r).count().equals(1)
14         * toilet.related_to(r).count().equals(1)
15         * storage.related_to(r).all(
16             lambda t: (
17                 obj[Semantics.BathroomItem].related_to(t, cu.on).count() >= 0
18             )
19         )
20     )
21 )
22
23 score_terms["toilet"] = rooms.all(
24     lambda r: (
25         toilet.distance(doors).maximize(weight=1)
26         + toilet.distance(furniture).maximize(weight=1)
27         + toilet.distance(sink).maximize(weight=1)
28         + cl.accessibility_cost(toilet, furniture, dist=2).minimize(weight=10)
29     )
30 )
31
32 constraints["bathtub"] = bathrooms.all(
33     lambda r: (
34         bathtub.related_to(r).count().in_range(0, 1)
35         * hardware.related_to(r).count().in_range(1, 4)
36     )
37 )
38 score_terms["bathtub"] = bathrooms.all(
```

```
39    lambda r: (
40        bathtub.mean(lambda t: t.distance(hardware)).minimize(weight=0.2)
41        + sink.mean(lambda t: t.distance(hardware)).minimize(weight=0.2)
42        + hardware.mean(
43            lambda t: (
44                t.distance(rooms, cu.floortags).hinge(0.5, 1).minimize(weight=15)
45            )
46        )
47    )
48 )
49
50 score_terms["bathroom"] = (
51    mirror.related_to(bathrooms).distance(sink).minimize(weight=3)
52 ) + cl.accessibility_cost(mirror, furniture, cu.down_dir).maximize(weight=3)
53 # endregion
```

Listing 2: An in-context example for Bathroom constraints provided to the LLM agent.

B CoT Reasoning for Refinement of Scene Constraints

This section details the iterative feedback loop employed to resolve the conflicts in the generation of scene constraints. The refinement process operates as follows:

1. **Execution:** The layout optimizer attempts to generate the scene based on the current set of procedural constraints.

2. **Error Reporting:** Upon the completion of a fixed number of optimization steps, a structured error report is synthesized from the optimization metadata. This report comprises:
   - *Textual Summary:* A list of constraints that remained unsatisfied during the optimization process.
   - *Visual Metadata:* A semantic 2D BEV (Bird's-Eye View) segmentation map annotated with object labels to visualize spatial conflicts.

3. **Refinement:** Based on the user's original scene description and the generated error report, the LLM evaluates whether the user's requirements have been fully met. If unsatisfied, the LLM employs Chain-of-Thought (CoT) reasoning to analyze the failure and output revised constraints. In practice, we use the thinking mode of Gemini-2.5-Pro to enable CoT reasoning.

A detailed end-to-end example of such CoT reasoning, with the constraint APIs and syntax provided by the Infinigen framework Raistrick et al. (2024), is provided in Figure 14 below:

C Details of Cluster-Based Optimization

To address the limitations of the "rigid hierarchy" that is prevalent in traditional procedural generation frameworks, in InfiniBench we implemented a robust cluster-based optimization strategy.

C.1 Cluster Definition and Identification

We define the **root** of a cluster as an object situated at the highest level of the hierarchy (typically large furniture, such as tables or desks). Objects at lower hierarchy levels are designated as **child** objects if they maintain a "stable against" spatial relationship with a parent object. Information regarding these clusters is dynamically retrieved and updated from the global "scene state" throughout the optimization process.

C.2 Augmented Action Space

In standard rigid-hierarchy optimization, the action space for individual objects comprises addition, resampling, deletion, translation, rotation, relation plane changes, relation target changes, and swapping. To facilitate cluster-based optimization, we augment this space by adding the following cluster-specific operations:
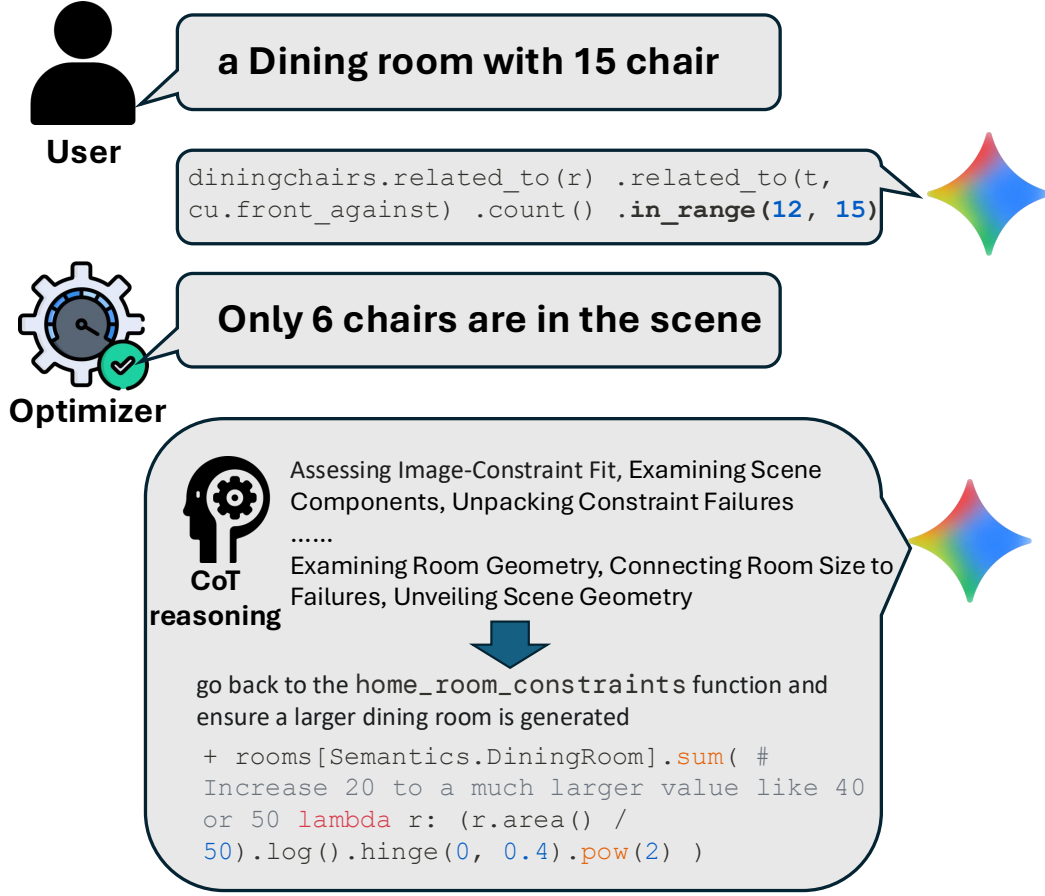
Figure 14: An example of scene constraint refinement via CoT reasoning

- `resample_cluster`: Resamples the parameters for the cluster's root object, triggering a cascade update for child objects.
- `translation_cluster`: Translates the entire cluster as a single entity while strictly preserving the relative positions of all constituent objects.
- `rotation_cluster`: Rotates the cluster around the geometric center of the root object, maintaining the relative orientation and position of all child objects.

C.3 COLLISION CHECKING

Following each operation, the optimizer performs a collision detection and checking routine. If a collision is detected, the operation is immediately reverted to maintain physical plausibility. For cluster-level operations, collision checks are performed using the cluster's collective bounding box (or the aggregate of all component meshes) to ensure the entire group is placed validly without intersecting other scene elements.

D GALLERY OF GENERATED 3D SCENES

We provide additional examples of the 3D scenes generated by InfiniBench, demonstrating its versatility across various room types and levels of scene complexity. The scenes with different levels of compositional complexity are shown in Figures 15-17, and the scenes with different levels of relational complexity are shown in Figures 18-20. In addition, in Figure 21 and Figure 22, we

Figure 15: Examples of generated **living rooms** with varying levels of **compositional complexity**, measured by the number of objects. 'Low' indicates seating for fewer than five people; 'Medium' indicates seating for five to ten people; and 'High' indicates seating for more than ten people. The user's prompt to the LLM agent is: "Generate a living room with seating for ** people."

Figure 16: Examples of generated **bedrooms** with varying levels of **compositional complexity**, measured by the number of objects. 'Low' indicates only one shelf; 'Medium' indicates 2-8 shelves; and 'High' indicates seating for more than 10 shelves. The user's prompt to the LLM agent is: "Generate a bedroom with ** shelves."

also provided examples of the optimized camera trajectories, along with the scene frames at the selected waypoints, with different levels of observational complexities. Note that, in addition to InfiniBench's native scene-generation pipeline, we also leverage the room-shape generation module from Infinigen to produce more realistic renderings of these example scenes.

E DETAILS OF THE CAMERA TRAJECTORY OPTIMIZATION ALGORITHM

To ensure that the generated camera trajectory simulates a natural, human-like perspective while maximizing the visual coverage of task-related objects, we constrain the optimization problem to a 4-DOF space. We fix the camera's vertical height (z) to approximate a standard eye level and restrict the roll angle (θ_x) to zero to maintain view stability. Consequently, the optimization is performed over the remaining four degrees of freedom: the horizontal position coordinates (x, y) and the rotational angles for yaw (θ_y) and pitch (θ_z). This configuration allows the camera to freely

Figure 17: Examples of generated **dining rooms** with varying levels of **compositional complexity**, measured by the number of objects. 'Low' indicates seating for fewer than five people; 'Medium' indicates seating for five to ten people; and 'High' indicates seating for more than ten people. The user's prompt to the LLM agent is: "Generate a dining room with seating for ** people."

Figure 18: Examples of generated **living rooms** with varying levels of **relational complexity**, measured by the room occupancy ratio. 'Low' indicates an occupancy rate below 10%; 'Medium' indicates 10–50%; and 'High' indicates an occupancy rate above 50%. The user's prompt to the LLM agent is: "Generate a living room with with occupancy rate of **."

navigate accessible areas on the horizontal plane and adjust its viewing angle to effectively frame relevant spatial features.

Optimization Workflow. As shown in Algorithm 1, our method follows an iterative workflow that progressively builds a path to view every task-related object in the scene. The process is as follows:

0. Scene Preprocessing: Before optimization, the generated scene file (e.g., a `.blend` file) is preprocessed to extract the necessary geometric and semantic metadata. Using Blender's `Bmesh` API, we extract the 3D mesh of every object. These meshes are then triangulated and converted into `Trimesh` objects, which facilitates efficient geometric queries. A 2D floor plan is also extracted to define the navigable area for the camera.

1. Target Selection (find the closest target object): The algorithm begins at a predefined starting point, typically a door. It maintains a list of "unvisited" objects. In each iteration, it identifies the object from this list that is closest to the camera's current position, designating it as the next target.

Figure 19: Examples of generated **bedrooms** with varying levels of **relational complexity**, measured by the room occupancy ratio. 'Low' indicates an occupancy rate below 10%; 'Medium' indicates 10–50%; and 'High' indicates an occupancy rate above 50%. The user's prompt to the LLM agent is: "Generate a bedroom with with occupancy rate of **."

Figure 20: Examples of generated **dining rooms** with varying levels of **relational complexity**, measured by the room occupancy ratio. 'Low' indicates an occupancy rate below 10%; 'Medium' indicates 10–50%; and 'High' indicates an occupancy rate above 50%. The user's prompt to the LLM agent is: "Generate a dining room with with occupancy rate of **."

2. Viewpoint sampling: A set of candidate viewpoints is randomly sampled in the vicinity of the current target object. Each viewpoint is a 4-DOF pose that is oriented towards the object.

3. Viewpoint selection: Each candidate viewpoint is evaluated against a series of physical constraints to determine its validity. The best viewpoint is one that satisfies all the following criteria:

- *Accessibility*: The camera's (x, y) location must be within the accessible area of the 2D floor plan (i.e., inside the room but outside the bounding boxes of all objects).

- *Field of View (FOV)*: The target object must be fully visible within the camera's viewing frustum from the candidate pose.

- *Occlusion Check*: To ensure a clear line of sight, we perform an occlusion check. A segmentation map is rendered from the viewpoint using a fast off-screen renderer (`pyrenderer`). By projecting the 3D mesh of the target object and comparing it against the segmentation map's depth buffer, we calculate the percentage of the object that is oc-
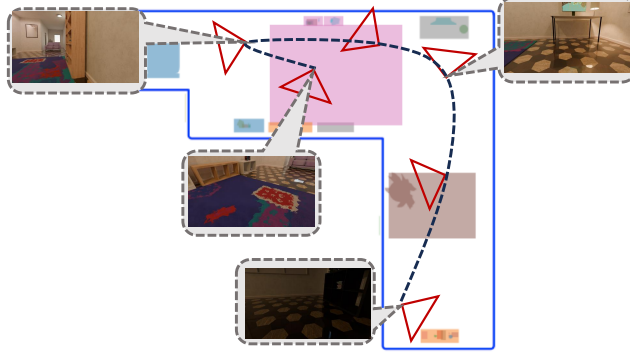
Figure 21: Visualization of the camera trajectory under a specific level of observational complexity (partial occlusion). The user's prompt to the LLM agent is: "Generate a typical living room. The camera should capture objects with partial occlusion."
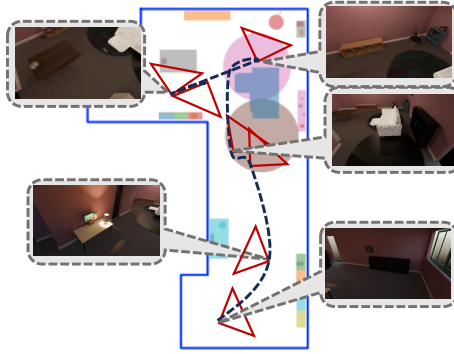

Figure 22: Visualization of the camera path under a specific level of observational complexity (bird-eye view, camera height is 2m). The user's prompt to the LLM agent is: "Generate a typical living room. The camera should capture height should be 2m"

cluded by other geometry. A viewpoint is only considered valid if the occlusion is below a predefined threshold.

The candidate that best satisfies these conditions (e.g., has the lowest occlusion) is selected as the next keyframe in the trajectory.

4. Path planning: Once a valid viewpoint is selected, we use Dijkstra's algorithm to compute the shortest, collision-free path on the 2D floor plan from the camera's current position to the new target viewpoint.

5. Iteration: The camera is moved along the planned path to the new viewpoint, the target object is marked as "visited," and the process repeats from Step 1. This loop continues until all objects in the scene have been successfully captured by at least one viewpoint. The final output is a complete, smooth camera trajectory composed of a sequence of optimal viewpoints connected by navigable paths, ensuring a comprehensive visual record of the scene.

F   DETAILS OF INFINIBENCH IMPLEMENTATION

As illustrated in Figure 23, we employ the Blender Cycles engine, which utilizes path tracing, to achieve photorealistic frame rendering. A critical parameter in this process is the number of path-tracing samples. While 8192 samples are typically recommended to produce noise-free images (as shown in Figure 24), this setting is computationally expensive, requiring approximately 5 minutes per frame. To optimize for efficiency without compromising quality, we adjust the sample count to

Algorithm 1 Camera Trajectory Optimization

Require: Scene, MaxSamplingTimes, MaxDistance, Threshold, RequiredRange

1: *CamParams* ← GETCAMERAPARAMETERS
2: *Objects* ← EXTRACTOBJECTMESHES(*Scene*)
3: *Targets* ← GETCAPTURELIST
4: *CurrentView* ← GETDOORPOSITION
5: *TotalPath* ← [ ]
6: *RemainingTargets* ← COPY(*Targets*)
7: **while** ¬ISEMPTY(*RemainingTargets*) **do**
8:     *target* ← FINDCLOSESTTARGET(*CurrentView*)   // 1. Target selection
9:     **for** $i = 1$ to *MaxSamplingTimes* **do**
10:         *view* ← SAMPLEVIEWPOINT(*target*, *MaxDistance*)   // 2. Viewpoint sampling
11:         **if** ¬CHECKCOLLISION(*view*, *Targets*) **then**   // 3. Viewpoint selection
12:             *FOV* ← COMPUTEFOV(*view*, *CamParams*)
13:             *FOVPercent* ← CHECKFOV(*FOV*, *target*)
14:             **if** *FOVPercent* > *Thr* **then**
15:                 *SegMap* ← RENDERER(*view*, *Scene*)
16:                 *Occ* ← COMPUTEOCCLUSIONRATE(*SegMap*, *target*)
17:                 **if** *Occ* ∈ *RequiredRange* **then**
18:                     *Path* ← PLANPATHDIJKSTRA(*CurrentView*, *view*, *scene*)   // 4. Path planning
19:                     ADD(*TotalPath*, *Path*)
20:                     *CurrentView* ← *view*
21:                     **break**
22:                 **end if**
23:             **end if**
24:         **end if**
25:     **end for**
26:     REMOVE(*RemainingTargets*, *target*)   // 5. Proceed to the next closest target
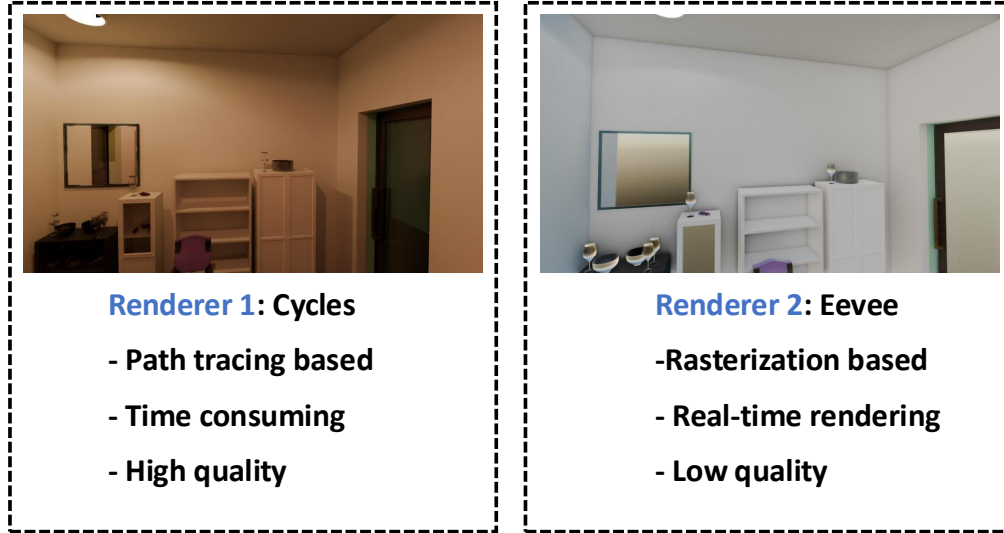27: **end while**
28: **return** *TotalPath*

Figure 23: Comparison of image quality across different renderers.

32 and enable denoising. As shown in Figure 23, after reducing the number of samples and enabling such denoising, the rendered frames appear nearly identical to the originals with 8192 samples, with only some fine textures slightly blurred. The specific rendering configuration is detailed below (unlisted parameters remain at default values).

- `num_samples`: 128
- `adaptive_threshold`: 0.01
- `denoise`: True
- `exposure`: 1

G ADDITIONAL QUALITATIVE COMPARISONS

As shown in Figure 25, we present additional examples comparing scenes generated by InfiniBench against baseline approaches. To ensure a fair comparison, all methods utilize the same 3D asset library and an identical square-shaped room plan. For baselines implemented in disparate engines, we exported the scene layouts and rendered them using Blender with identical settings to maintain visual consistency. For baselines incapable of processing natural language input directly, we employed a constraint generation method similar to Infinigen, iteratively refining the constraints until the program became executable.
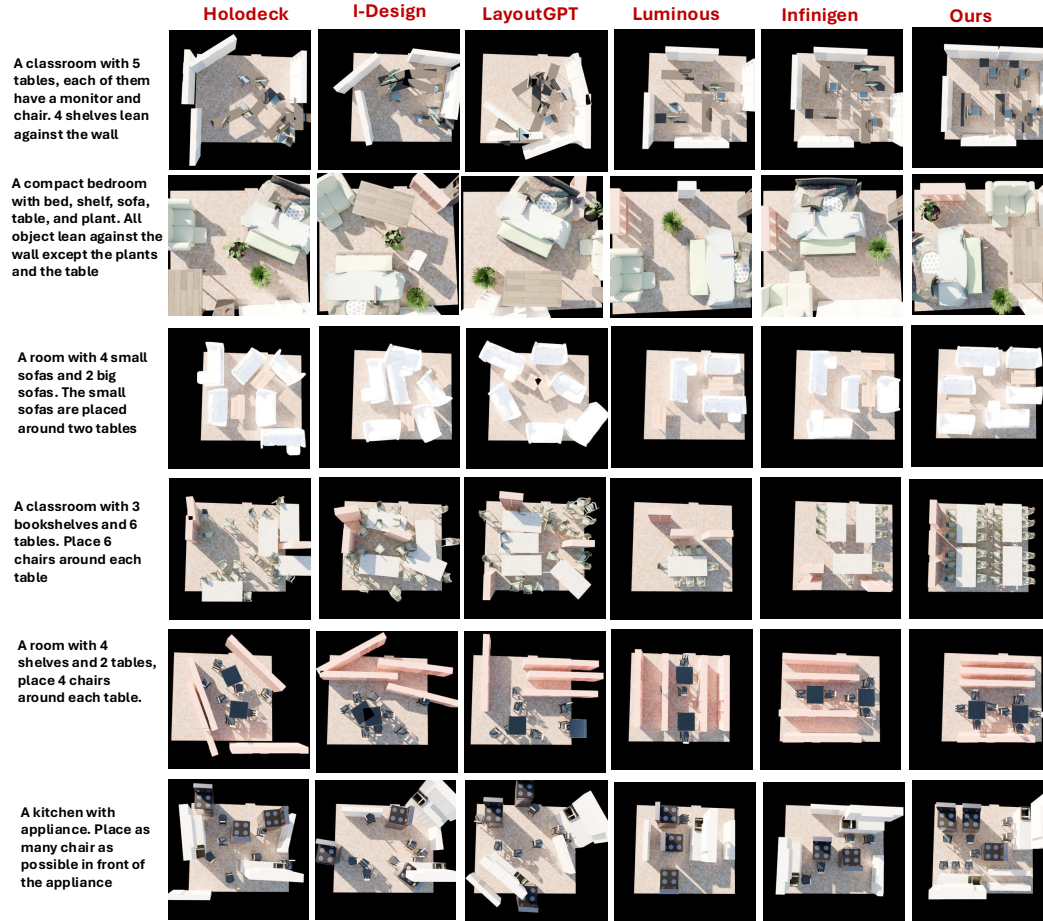
Figure 24: Impact of sample counts on image quality.

Figure 25: Comparing InfiniBench with baselines