EmbodiedScene: Towards Automated Generation of Diverse and Realistic Scenes for Embodied AI

Anonymous Authors[1]

Abstract

Collecting high-quality data within simulation environments has proven to be an effective strategy for addressing data challenges in embodied AI. As a critical step in data collection, the construction of simulation scenes heavily relies on expert knowledge, while automated approaches often struggle with ensuring sufficient diversity and realism. To address these limitations, we present EmbodiedScene, a hierarchical framework that leverages large language models (LLMs) to automate the generation of diverse and realistic simulation scenes, with a particular focus on multi-room indoor scene synthesis. To encourage diversity, we introduce a unified representation that encodes spatial configurations and layout semantics. This representation is populated with detailed content by LLMs and further diversified using evolutionary algorithms. It then serves as the foundation for downstream scene synthesis, guiding the generation of precise absolute parameters through a three-stage coarse-to-fine process: floor plan, region plan, and layout plan. To ensure reliability, we incorporate a vision-language model (VLM) as a stage-wise scene critic. The VLM provides feedback by comparing intended design objectives with the generated outputs and guides iterative refinement at each stage. Experimental results demonstrate that EmbodiedScene generates scenes with significantly greater realism and diversity than strong baselines. We further show that EmbodiedScene plays a key role in improving the performance of downstream embodied AI tasks.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

Recent advances in embodied AI (O'Neill et al., 2024; Kim et al., 2024; Black et al., 2024; Zhao et al., 2023) have demonstrated promising capabilities across a range of tasks, yet training robust agents still requires extensive interaction data from real-world settings. Simulation (Kolve et al., 2017; Savva et al., 2019; Gu et al., 2023; Makoviychuk et al., 2021) thus offers a scalable alternative for data collection. However, generating simulation scenes that accurately reflect human-like spatial layouts and furniture arrangements still relies heavily on expert knowledge and incurs substantial costs. The ability to automatically generate diverse and realistic simulation scenes is essential for developing embodied AI agents that generalize well across environments.

Existing simulated scenes can be broadly categorized into scan-based and synthetic environments. Scan-based scenes (Ramakrishnan et al., 2021; Xia et al., 2018) are reconstructed from real-world interiors and retain authentic spatial semantics, but they often suffer from low fidelity and scalability limitations due to the constraints of current reconstruction pipelines. In contrast, synthetic scenes provide greater flexibility and control. Manually authored scenes, created by artists using interactive tools (Khanna et al., 2024; Gu et al., 2023; Li et al., 2021; Deitke et al., 2020), typically exhibit high visual quality but require significant human effort and do not scale well. Other approaches treat scene generation as a combinatorial optimization problem (Yu et al., 2011; Wang et al., 2019; Raistrick et al., 2024), in which layout graphs are specified and resolved via heuristic search. However, it remains challenging to define a comprehensive constraint system for complex furniture categories and their combinations. Data-driven methods have also emerged, employing autoregressive models (Luo et al., 2020), transformers (Paschalidou et al., 2021), or diffusion models (Tang et al., 2024; Yang et al., 2024b;a) to learn spatial configurations from existing datasets and synthesize novel layouts. More recently, LLMs and VLMs have been integrated into the generation pipeline (Wei et al., 2022; Ma et al., 2023; Wang et al., 2023), enabling high-level semantic reasoning. Methods such as (Feng et al., 2023; Yang et al., 2024c; Öcal et al., 2024; Sun et al., 2024) leverage natural language prompts to guide layout generation, introducing a

Figure 1. Examples generated by EmbodiedScene. With just a simple prompt, EmbodiedScene is able to generate diverse styles and layouts that are physically plausible and approach human-level design quality.

more controllable and interpretable paradigm. Despite these advancements, existing methods often fall short in producing diverse and realistic environments and face limitations in generating coherent multi-room scenes.

To address these challenges, we propose EmbodiedScene, a hierarchical framework that automatically generates diverse and realistic simulation environments from natural language prompts. The framework consists of three core components: 1) structural representation evolution, 2) multi-stage scene generation, and 3) VLM-driven alignment and refinement. EmbodiedScene begins by constructing a unified structural representation of the scene, which is enriched using LLMs conditioned on the user prompt. To enhance diversity, this representation is further optimized via an evolutionary algorithm. The refined structure guides a multi-stage synthesis pipeline, with a VLM evaluating intermediate outputs and triggering refinements to maintain coherence and realism.

Scene diversity is essential for training agents that generalize beyond narrow environments. To address this, we enhance scene diversity from both spatial structural and visual perspectives. Specifically, we design an expressive structural representation that captures high-level spatial layout information, which is then populated by LLMs based on user prompts. An evolutionary algorithm is applied to this structure, which serves as the input to a downstream multi-stage scene generation pipeline. By evolving the structural input rather than the final scene output, we enable controllable and efficient diversity throughout the generation process.

Scene synthesis can be formulated as two closely related subproblems: room layout and furniture arrangement. Prior work (Yu et al., 2011; Tang et al., 2024; Wu et al., 2019; Shabani et al., 2023) often treats these components separately, seldom modeling the dependencies between them. For example, a dining table in the living room is typically placed near the kitchen to maintain functional continuity. To better capture such inter-room relationships, we propose a three-stage pipeline that follows a coarse-to-fine paradigm. Specifically, the *FloorPlan* stage solves the room layout, while the *LayoutPlan* stage addresses furniture arrangement. Bridging the two, the intermediate *RegionPlan* stage establishes connectivity by controlling the placement of functional regions, allowing the furniture layout in each room to be informed by the spatial context of neighboring rooms.

Since our framework involves multiple stages, errors such as LLM hallucinations and suboptimal decisions from intermediate solvers can accumulate if not properly addressed. To mitigate this, we introduce a VLM as a critic. Given the objective and output of each stage, the VLM provides detailed feedback to determine whether refinement is needed and how the current result should be refined.

We conduct extensive experiments to validate the effectiveness of our framework. First, in terms of diversity, we measure the generated scenes using both text similarity and image similarity metrics. For realism, we compare our generated scenes with real-world ones using FID and KID scores. Additionally, by utilizing VLM for automated as-

sessment, our generated scenes demonstrate significantly higher realism and semantic consistency compared to existing approaches. The effectiveness of our framework is further validated through object navigation experiments, where agents trained in our generated environments show improved performance. Through extensive ablation studies, we demonstrate that each component of our pipeline contributes to the overall quality of the generated scenes.

We summarize our contributions below: 1) We propose a structural representation for 3D scenes, populated by LLMs and diversified through evolutionary optimization. 2) We develop a three-stage coarse-to-fine generation pipeline with a strong capability of generating complex multi-room simulation scenes. 3) We propose a stage-level alignment and optimization mechanism to ensure efficient alignment between initial prompts and the generated scenes. 4) Experiments demonstrate the effectiveness of our method in improving scene diversity and realism and show the benefits of our generated scenes in downstream embodied AI tasks.

2. Related Work

2.1. 3D Indoor Scene Design

Existing scene generation approaches vary in terms of construction method and degree of automation. Some rely on handcrafted content (Khanna et al., 2024; Gu et al., 2023; Li et al., 2021; Deitke et al., 2020) or 3D scans of real environments (Ramakrishnan et al., 2021; Xia et al., 2018), achieving high realism but suffering from high manual costs and poor scalability. Others adopt procedural generation, combining assets based on predefined rules, such as rule-based combinatorial search (Deitke et al., 2022) or heuristic algorithms (Yu et al., 2011; Wang et al., 2019; Raistrick et al., 2024). While more scalable, these methods depend heavily on asset annotation and rule completeness, which limits both diversity and quality. More recently, data-driven models such as autoregressive models (Luo et al., 2020), Transformer-based architectures (Paschalidou et al., 2021), and diffusion models (Tang et al., 2024; Yang et al., 2024b;a) learn layout distributions from existing datasets to generate scenes without manual specification. LLMs have demonstrated remarkable capabilities in solving a variety of complex tasks (Wei et al., 2022; Wang et al., 2023; Ma et al., 2023; Yang et al., 2023; Hong et al., 2023), some approaches (Feng et al., 2023; Yang et al., 2024c; Çelen et al., 2024; Aguina-Kang et al., 2024) have attempted to leverage the broad knowledge and strong reasoning abilities of LLMs to assist in automated scene generation.

2.2. Embodied AI Simulation Environments

Building simulation environments is a key step in embodied AI training, as diverse and realistic scenes provide rich and high-quality tasks for embodied AI. Currently, most simulation environments (Kolve et al., 2017; Savva et al., 2019; Szot et al., 2021; Puig et al., 2023; Gu et al., 2023; Tao et al., 2025; Yu et al., 2020; Zhu et al., 2020; Nasiriany et al., 2024; Makoviychuk et al., 2021; Liu et al., 2023) for embodied AI rely on manual design, and scaling up these environments often requires significant labor costs. Although 3D scanning (Ramakrishnan et al., 2021) technology can reduce manual intervention, it still faces issues such as artifacts. While procedural generation methods (Deitke et al., 2022; Raistrick et al., 2024; Wang et al., 2023) can quickly produce a large number of simulation scenes, they are limited by manually defined rules, restricting scene diversity. The generative simulation framework (Authors, 2024), as an emerging concept, can automatically generate high-quality and diverse simulation scenes, thereby reducing the burden of manual design and offering more flexible scene variations.

3. EmbodiedScene

This section primarily introduces our framework, EmbodiedScene, designed for the automated generation of diverse and realistic simulation scenes. We begin by presenting an overview of EmbodiedScene to provide a comprehensive understanding of its key components, followed by a detailed description of each individual component.

3.1. Framework Overview

The diversity and realism of simulation scenes are critical factors in constructing high-quality datasets for embodied AI agents. However, achieving both objectives remains an open problem. To address these challenges, we propose EmbodiedScene, a general framework for automatic scene generation based on LLMs.

The core ideas of EmbodiedScene are to 1) leverage LLMs and VLMs to automate the generation and optimization processes, and 2) introduce a hierarchical approach for diversity and realism, which consists of three core components: high-level structual representation evolution, lower-level scene generation, and inter-level alignment and refinement.

The overall framework is illustrated in Figure 2. Specifically, EmbodiedScene comprises the following three key components. First, the high-level evolutionary mechanism expands the user's initial prompt into multiple structural representations. These representations then guide a multi-stage scene generation process at the lower level. Meanwhile, a inter-level alignment and refinement mechanism leverages VLMs to ensure consistency and robustness throughout the generation process.

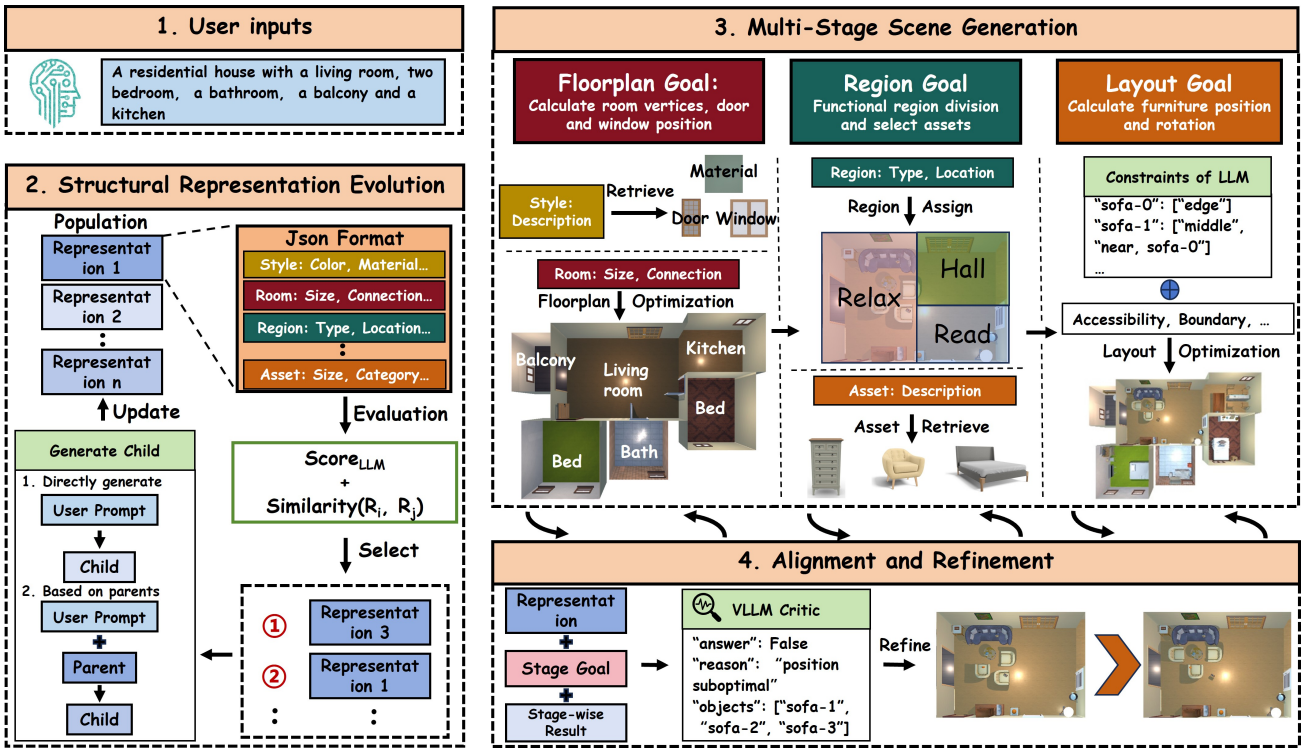

Figure 2. Overall framework of EmbodiedScene. After a user provides a prompt, EmbodiedScene first uses an LLM to populate the content based on our designed scene structural representation. Then, it enhances diversity through an evolutionary algorithm. The structural representation is used to guide a three-stage scene generation process, during which EmbodiedScene employs a VLM to evaluate and refine the goals and outputs at each stage.

3.2. High-level Input Diversity

The underlying motivation for this approach is to treat the structural representations as inputs to the generation framework. Directly using the final scene configurations as individuals for evolution leads to an excessively large optimization space, where randomly altering elements such as room or furniture positions makes it difficult to maintain physical plausibility. Moreover, there are sequential dependencies across different stages of scene generation. Modifications in earlier stages can render the outputs of subsequent stages invalid. For instance, changing the position of a room would necessitate readjusting the locations of doors, windows, and furniture, effectively incurring the cost of a near-complete regeneration. To address these challenges, we propose an evolution mechanism, which enriches and diversifies the initial user input. These enriched structural representations serve as the goals for subsequent scene generation, ensuring that every stage of scene creation and optimization is aimed at better achieving the goals.

We design a unified representation $R = \{\text{Style}, \text{Room}, \text{Object}, \text{Door}, \text{Window}\}$ to capture the spatial configuration and layout semantics of indoor scenes. Specifically, *Style* describes the overall design style of the house, covering elements such as color schemes and material choices. *Room* includes attributes such as the size of each room, floor and wall materials, adjacent room relationships, and the functional zones within each room. *Object* refers to the set of furniture items selected for each functional zone, where each item is characterized by features such as size, category, and texture. *Door* represents the types of doors generated based on the established connectivity between rooms. *Window* denotes the windows included in each room. This structured representation is completed progressively in a staged manner and is used as the individual representation in the evolutionary algorithm. Details on the construction of scene representations can be found in the appendix F.

The structural representation evolution mechanism primarily consists of the following three stages:

Population Initialization. We take the initial user prompt and the structural representation templates as inputs to the LLM, guiding it to refine the representation r_i, which is formatted in JSON. This process is repeated n times to construct the initial representation population P.

Population Selection. We use a combined score that accounts for both generation quality and individual diversity as the fitness of each individual in the population, and iteratively select the best individuals based on this score. The detailed procedure of our selection method is presented in Algorithm 1. The computation of the combined score is as follows:

$$\text{fitness} = \arg\max_{r_i \in P \setminus S} \Big[\lambda \cdot Score_{LLM}(r_i) - (1-\lambda) \cdot \max_{r_j \in S} \text{Sim}(r_i, r_j) \Big] \quad (1)$$

The set S is initialized as an empty set and is used to store the selected individuals. $Score_{LLM}$ denotes the quality score assigned by a LLM acting as a critic to evaluate the generated outputs. The similarity score $\text{Sim}(r_i, r_j)$ is computed using cosine similarity between structural representations. Each representation is first encoded into a feature vector using pretrained Sentence-BERT (Reimers & Gurevych, 2019) model. The cosine similarity is calculated as follows:

$$\text{Sim}(r_i, r_j) = \frac{\phi(r_i) \cdot \phi(r_j)}{\|\phi(r_i)\| \, \|\phi(r_j)\|} \quad (2)$$

where $\phi(r)$ denotes the Sentence-BERT embedding of individual r.

Algorithm 1 Individual Selection Algorithm

Require: Population P, number of individuals to select k, LLM scoring function $Score_{LLM}(r)$, similarity function $Sim(r_i, r_j)$, trade-off parameter λ
Ensure: Selected set of individuals S

```
Initialize S ← ∅
while |S| < k do
    max_score ← −∞
    best_candidate ← None
    for each r_i ∈ P \ S do
        sim_max ← max_{r_j ∈ S} Sim(r_i, r_j) {If S = ∅, define sim_max = 0}
        score ← λ · Score_LLM(r_i) − (1 − λ) · sim_max
        if score > max_score then
            max_score ← score
            best_candidate ← r_i
        end if
    end for
    S ← S ∪ {best_candidate}
end while
return S
```

Population Evolution. We retain the top e individuals with the highest fitness scores in the population as the elite set E, from which a subset is randomly sampled as parents and provided to the LLM. The LLM then generates a new set of individuals C_1 that are maximally diverse from these parents. In parallel, we also employ a strategy where the LLM directly generates new individuals without any parents, forming the set C_2. The sets E, C_1, and C_2 together constitute the next generation of the population $P' = E \cup C_1 \cup C_2$, after which the selection and evolution process is repeated.

Through the iterative optimization of the above process, we can construct diverse enhanced prompts to guide subsequent scene generation. Further details of the evolutionary algorithm can be found in the appendix B.

3.3. Lower-level Scene Generation

After optimizing the structural representation population, we generate scenes based on each representation in the population. The representation r_i can be considered as the high-level goal of the scene generation, with each subsequent generation step aimed at more efficiently achieving this goal. To enable efficient scene generation, we propose a three-stage coarse-to-fine generation process.

Scene Floorplan Design. We first determine the room locations, door positions, and window positions based on the room connectivity information provided by the high-level representations. Previous work (Yang et al., 2024c) typically used LLMs to directly provide room positions. While LLMs can generate numerically reasonable room design information, the spatial relationships and connectivity of the rooms may not align with reality, as shown in Figure 3. In this generated house, reaching the kitchen from the outside requires passing through a path of the living room, bedroom, another bedroom, and a bathroom, which significantly deviates from realistic room layouts.

To solve the problem, we first use LLM to generate a constraint graph for the room layout, where each node represents a room, and edges indicate connectivity between rooms. Then, based on this constraint graph, we use the backtracking algorithm to determine the actual positions of the rooms. The specific implementation logic is shown in Algorithm 2.

Functional Area Division. After the floorplan is completed, we introduce a functional region partitioning mechanism to establish a coupling between room layout and furniture placement. For example, the dining table in the living room's dining area is intentionally positioned adjacent to the kitchen to enhance functional connectivity. We provide the LLM with the positions of all functional regions within the room, along with the locations of adjacent rooms, and task it with designing the layout of these functional zones. To avoid complex design challenges, we constrain the number of functional zones per room to a maximum of three. Therefore, we only need to enumerate the possible combinations of relative positions based on the number of zones and let the LLM select the most appropriate configuration.

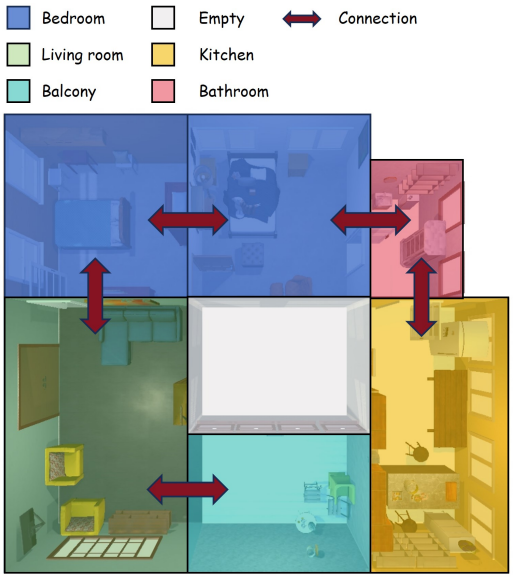

Figure 3. Scene generated by Holodeck. In the image, the connectivity of the rooms does not align with the semantics of a real-world scene, as it takes two bedrooms and a bathroom to get from the living room to the kitchen.

Once the positions of the functional zones are determined, we proceed to furniture asset selection. Following previous works(Feng et al., 2023; Yang et al., 2024c), we encode each asset as a vector and retrieve suitable assets from a predefined asset library based on similarity.

Layout Optimization. After determining the functional regions of each room and the furniture assigned to each region, we perform layout optimization for each individual region. Specifically, we provide the LLM with a list of all furniture items within a region, and the LLM generates a set of layout constraints. We then solve for the positions of the furniture items by exhaustively searching over a discretized grid within the region. Due to the diversity in furniture types, sizes, and the increased complexity introduced by their combinations, layout optimization poses a significant challenge. Building upon prior work (Yu et al., 2011; Wang et al., 2019; Raistrick et al., 2024; Yang et al., 2024c), we design a comprehensive set of constraints to guide the layout process. These constraints include global position, orientation, relative distance, boundary conditions, collision avoidance, and accessibility. Detailed descriptions of these constraints are provided in Appendix D.

3.4. Inter-level alignment and refinement

Through the previous subsection, we can achieve realistic scene generation. However, the scene realism cannot be guaranteed. Although we guide the entire generation process with r_i, due to the hallucination problem of LLMs, biases can easily arise. These biases can ultimately cause a mismatch between the scene and the prompt description, leading to a loss of diversity in the generated scenes.

To address the problem, we propose a discriminative mechanism based on VLMs. For alignment, at each stage, we provide the VLM with the user's original prompt, previously generated context, and the current generation output. The VLM is then used to determine whether the newly generated content is consistent with the prior context. Additionally, in stages where suboptimal solutions may arise, we leverage the VLM to assess whether further refinement is needed. For example, during the room or furniture layout generation stage, we provide the VLM with a top-down view of the house or room and ask it to identify which rooms or objects require repositioning. In such cases, the positions of other elements are fixed during re-optimization. The prompt design for the VLM at different stages can be found in Appendix F.

4. Experiment

This section provides a comprehensive evaluation of EmbodiedScene from the perspectives of diversity and realism. Then, we construct datasets for navigation tasks using scenes generated by different methods to evaluate the impact of generated scenes on navigation performance. Finally, an ablation study is conducted to evaluate the contribution of each components.

4.1. Quality Eavluation

Diversity Evaluation. We evaluate the diversity of scenes generated by different method from two perspectives: textual similarity and image similarity. Specifically, we design 10 prompts with different requirements and generated 100 complete scenes for each prompt. We then select the corresponding generation stages across different methods, including room layout, doors, windows, and asset selection. For each corresponding stage, we empoly the Sentence-Bert model to encode the responses provided by the LLM and compute the cosine similarity between stages. The average similarity score is then calculated across all stages. Furthermore, we visualize the scenes generated by each method and use the CLIP (Radford et al., 2021; Ilharco et al., 2021) model to encode the images. We then compute the cosine similarity between different scenes generated under the same prompt and average the results across all prompts. The experimental results, shown in Table 1, demonstrate that EmbodiedScene significantly improves the diversity of scene generation compared to the baseline methods.

Realism Evaluation. Improving the realism of the scene can significantly improve the perceptual abilities of an agent. We assess the realism of our generated scenes from two perspectives.

Table 1. Comparison of Different Methods (Textual and Visual Similarity).We compare methods on three semantic facets—Style (overall design), Floor (room size and connectivity), and Layout (furniture types, attributes, and positions)—and measure image diversity at both single-room and whole-house scales with fixed room types and counts.

	Textual Similarity (↓)					Visual Similarity (↓)		
Methods	**Style**	**Floor**	**Layout**	**Overall**	**Average**	**Room**	**Overall**	**Average**
LayoutGPT	0.8299	0.8035	0.7335	0.7865	0.7884	0.8754	0.8647	0.8705
Holodeck	0.7782	0.8088	0.7183	0.7790	0.7711	0.8965	0.8847	0.8906
Ours	**0.5509**	**0.7038**	**0.6656**	**0.6933**	**0.6534**	**0.7608**	**0.7412**	**0.7510**

FID and KID Metrics. We compare the rendered images of generated scenes with images of real world scenes. Specifically, we import the generated scenes into the Habitat (Savva et al., 2019; Szot et al., 2021; Puig et al., 2023) engine and render them using its raster-based renderer. As shown in Table 2, the scenes produced by **EmbodiedScene** are visually closer to real scenes. It should be noted that the scores are generally high. Through further analysis of both generated and real scene images, we attribute this outcome to two main factors: the suboptimal quality of synthetic renderings and the presence of artifacts in 3D-scanned scenes.

Table 2. Comparison of different methods on HM3D and Gibson datasets.1emFID and KID scores are reported (lower is better).

	HM3D		Gibson	
Methods	FID↓	KID↓	FID↓	KID↓
LayoutGPT	115.2	103.8 ± 2.9	121.5	107.4 ± 3.1
Holodeck	97.9	74.1 ± 3.3	100.3	78.7 ± 3.4
Ours	**84.1**	**57.6±2.6**	**90.0**	**63.6±2.8**

VLM Preference. While FID and KID can measure the visual realism of the scenes, they are still subject to the limitations of the simulator's rendering capabilities. Therefore, we introduce a VLM-based scoring approach that evaluates both visual and semantic realism. We extract scene information at each generation stage, such as overall style, room types, room connectivity, and asset composition. Corresponding top-down views of either the entire scene or individual rooms are attached for each stage. We then prompt the VLM with generation results from different methods, and the VLM indicates its preference. As shown in Figure 4, the VLM exhibits a stronger inclination towards scenes generated by EmbodiedScene, suggesting that it perceives these scenes as more realistic.

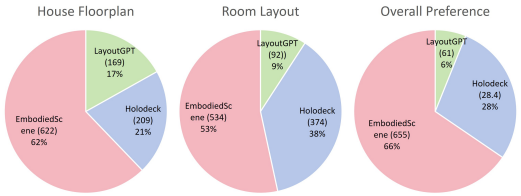

Figure 4. VLM Preference for different methods. Experimental design: 20 multi-room prompts generate 10 scenes each (200 sets), rated by five distinct VLMs. "House Floorplan" shows furniture-free floor plans, "Room Layout" presents top-down views of living room, bedroom, and kitchen, and "Overall" depicts a top-down view of the entire house.

4.2. Navigation Experiment

To validate the applicability of generated scenes, we utilize them to design a navigation experiment dataset. We select the ObjectNav (Batra et al., 2020) task for evaluation, where the objective is to locate a specified category of objects. Two well-established navigation benchmarks, RoboTHOR (Deitke et al., 2020) and HM3D (Ramakrishnan et al., 2021), are chosen as the primary test environments. The target object categories remain consistent with the benchmark settings, which are detailed in Appendix E. We follow (Khandelwal et al., 2022) and employ Decentralized Distributed Proximal Policy Optimization(Wijmans et al., 2019) (DDPPO) as the primary training framework for the agent's behavior policy.

The results of the models trained using different scene generation methods are presented in Table 3. Notably, the scenes generated by EmbodiedScene also achieve competitive test performance on Holodeck-generated environments, possibly due to the low visual variance between the two scene types, which benefits zero-shot generalization. EmbodiedScene outperforms Holodeck in both RoboTHOR and HM3D benchmarks but still underperforms in comparison to its own test performance. For RoboTHOR, a potential reason for this performance gap is that the scenes are not structured as realistic houses but instead consist of large rooms with various partitions. In the case of HM3D, the primary challenge lies in the significant visual disparity and increased complexity due to a greater number of rooms. The high computational cost of rendering more rooms in the simulation environment has currently limited our ability

Table 3. Navigation results across different train-test splits. Success (%) and Success weighted by Path Length (SPL) are reported for four evaluation datasets. The calculation formulas for Success and SPL can be found in the appendix E.

Eval Dataset	**Holodeck**		**EmbodiedScene**		**RoboTHOR**		**HM3DSem**	
Train Dataset	**Success**	**SPL**	**Success**	**SPL**	**Success**	**SPL**	**Success**	**SPL**
Holodeck	**58.34**	**42.35**	43.68	29.62	15.82	6.29	8.07	2.03
EmbodiedScene	52.72	30.89	**60.78**	**45.64**	**34.76**	**13.18**	**12.14**	**4.56**

to scale training to larger scenarios. We hypothesize that increasing the number of rooms in the training set could further enhance the model's performance. We will investigate this problem further in future work.

4.3. Ablation

We evaluate the importance of each module in our method. *w/o evolution* refers to removing the evolutionary mechanism for scene structural representation, instead generating multiple scenes using the same prompt without representation evolution. *w/o region* indicates that the multi-stage scene generation module no longer partitions the room into functional regions; instead, the LLM directly generates furniture for the entire room, followed by layout optimization. *w/o VLM critic* refers to removing the VLM from each stage, meaning that the generated outputs are passed directly to the next stage without alignment or refinement by a VLM.

We use 10 different user prompts for generation and evaluate the diversity and realism of the generated results under different experimental settings. The experimental results in Table 4 demonstrate that the structural representation evolution mechanism significantly improves the diversity of multiple scenes generated from the same prompt. Based on an analysis of the generated content, we attribute this to the evolutionary process encouraging more stylistic variations in the designs proposed by the LLM, which leads to greater differences in the selected furniture appearances and, consequently, improved scene diversity.

As for the functional region design, it effectively links layout optimization at the room and furniture levels, thereby improving scene realism. The use of a VLM as a critic brings limited improvements in realism. We hypothesize that this may be due to the fact that most of the images we provide are top-down views, which may prevent the VLM from fully understanding object and spatial information within the scene. We leave this issue for future work.

Table 4. Ablation results on different modules. The full model (EmbodiedScene) outperforms its ablations in terms of both diversity and realism metrics.

Method	Similarity ↓	FID ↓	KID ↓
EmbodiedScene	**0.76**	**84.1**	**57.6±2.6**
w/o evolution	0.88	87.5	60.6 ± 2.5
w/o region	0.76	92.8	65.9 ± 2.6
w/o VLM critic	0.76	86.4	59.8 ± 2.6

5. Conclusion

This paper introduces EmbodiedScene, an innovative scene generation framework focused on enhancing the diversity and realism of generated scenes. The framework, based on LLMs, consists of three main components: structural representation evolution, multi-stage scene generation, and alignment and refinement. Structural representation evolution mainly improve the input to the scene generation system, which increases the diversity of the input and, correspondingly, enhances the diversity of the output. This approach is more effective than randomly enhancing the diversity of the system's output. We design a more efficient scene generation pipeline. By breaking down the process into finer-grained steps, our generated scenes contain more realistic details. To further ensure the realism of the generated scenes and reduce hallucination issues in LLMs, an alignment and refinement mechanism based on VLMs is designed. At each stage of the generation, we strive to ensure the consistency between the generated results and the stage objectives. If inconsistencies are found, measures are taken to refine the output. Through various metrics, we demonstrate that our method significantly improves the diversity and realism of the generated scenes. Furthermore, navigation agents trained using these scenes exhibit enhanced generalization capabilities.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Aguina-Kang, R., Gumin, M., Han, D. H., Morris, S., Yoo, S. J., Ganeshan, A., Jones, R. K., Wei, Q. A., Fu, K., and Ritchie, D. Open-universe indoor scene generation using llm program synthesis and uncurated object databases. *arXiv preprint arXiv:2403.09675*, 2024.

Anthropic. Claude 3 opus. https://www.anthropic.com/news/claude-3-family, 2024. Accessed: 2025-05-16.

Authors, G. Genesis: A universal and generative physics engine for robotics and beyond, December 2024. URL https://github.com/Genesis-Embodied-AI/Genesis.

Batra, D., Gokaslan, A., Kembhavi, A., Maksymets, O., Mottaghi, R., Savva, M., Toshev, A., and Wijmans, E. Objectnav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*, 2020.

Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., et al. $\pi 0$: A vision-language-action flow model for general robot control, 2024. *URL https://arxiv.org/abs/2410.24164*, 2024.

Çelen, A., Han, G., Schindler, K., Van Gool, L., Armeni, I., Obukhov, A., and Wang, X. I-design: Personalized llm interior designer. *arXiv preprint arXiv:2404.02838*, 2024.

DeepMind, G. Gemini 2.0. https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/#project-mariner, 2024.

Deitke, M., Han, W., Herrasti, A., Kembhavi, A., Kolve, E., Mottaghi, R., Salvador, J., Schwenk, D., VanderBilt, E., Wallingford, M., et al. Robothor: An open simulation-to-real embodied ai platform. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3164–3174, 2020.

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

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

Gu, J., Xiang, F., Li, X., Ling, Z., Liu, X., Mu, T., Tang, Y., Tao, S., Wei, X., Yao, Y., Yuan, X., Xie, P., Huang, Z., Chen, R., and Su, H. Maniskill2: A unified benchmark for generalizable manipulation skills. In *International Conference on Learning Representations*, 2023.

Hong, S., Zheng, X., Chen, J., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S. K. S., Lin, Z., Zhou, L., et al. Metagpt: Meta programming for multi-agent collaborative framework. *arXiv preprint arXiv:2308.00352*, 3(4): 6, 2023.

Ilharco, G., Wortsman, M., Wightman, R., Gordon, C., Carlini, N., Taori, R., Dave, A., Shankar, V., Namkoong, H., Miller, J., Hajishirzi, H., Farhadi, A., and Schmidt, L. Openclip, July 2021.

Khandelwal, A., Weihs, L., Mottaghi, R., and Kembhavi, A. Simple but effective: Clip embeddings for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14829–14838, 2022.

Khanna, M., Mao, Y., Jiang, H., Haresh, S., Shacklett, B., Batra, D., Clegg, A., Undersander, E., Chang, A. X., and Savva, M. Habitat synthetic scenes dataset (hssd-200): An analysis of 3d scene scale and realism tradeoffs for objectgoal navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16384–16393, 2024.

Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P., et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Deitke, M., Ehsani, K., Gordon, D., Zhu, Y., et al. Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*, 2017.

Li, C., Xia, F., Martín-Martín, R., Lingelbach, M., Srivastava, S., Shen, B., Vainio, K., Gokmen, C., Dharan, G., Jain, T., et al. igibson 2.0: Object-centric simulation for robot learning of everyday household tasks. *arXiv preprint arXiv:2108.03272*, 2021.

Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., and Stone, P. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

Luo, A., Zhang, Z., Wu, J., and Tenenbaum, J. B. End-to-end optimization of scene layout. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3754–3763, 2020.

Ma, Y. J., Liang, W., Wang, G., Huang, D.-A., Bastani, O., Jayaraman, D., Zhu, Y., Fan, L., and Anandkumar, A. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

Makoviychuk, V., Wawrzyniak, L., Guo, Y., Lu, M., Storey, K., Macklin, M., Hoeller, D., Rudin, N., Allshire, A., Handa, A., et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021.

Nasiriany, S., Maddukuri, A., Zhang, L., Parikh, A., Lo, A., Joshi, A., Mandlekar, A., and Zhu, Y. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

Öcal, B. M., Tatarchenko, M., Karaoğlu, S., and Gevers, T. Sceneteller: Language-to-3d scene generation. In *European Conference on Computer Vision*, pp. 362–378. Springer, 2024.

OpenAI. Gpt-4o: Openai's omnimodal model. https://openai.com/index/hello-gpt-4o/, 2024. Accessed: 2025-05-16.

O'Neill, A., Rehman, A., Maddukuri, A., Gupta, A., Padalkar, A., Lee, A., Pooley, A., Gupta, A., Mandlekar, A., Jain, A., et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Paschalidou, D., Kar, A., Shugrina, M., Kreis, K., Geiger, A., and Fidler, S. Atiss: Autoregressive transformers for indoor scene synthesis. *Advances in Neural Information Processing Systems*, 34:12013–12026, 2021.

Puig, X., Undersander, E., Szot, A., Cote, M. D., Partsey, R., Yang, J., Desai, R., Clegg, A. W., Hlavac, M., Min, T., Gervet, T., Vondrus, V., Berges, V.-P., Turner, J., Maksymets, O., Kira, Z., Kalakrishnan, M., Malik, J., Chaplot, D. S., Jain, U., Batra, D., Rai, A., and Mottaghi, R. Habitat 3.0: A co-habitat for humans, avatars and robots, 2023.

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. PmLR, 2021.

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

Ramakrishnan, S. K., Gokaslan, A., Wijmans, E., Maksymets, O., Clegg, A., Turner, J., Undersander, E., Galuba, W., Westbury, A., Chang, A. X., et al. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv preprint arXiv:2109.08238*, 2021.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9339–9347, 2019.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shabani, M. A., Hosseini, S., and Furukawa, Y. Housediffusion: Vector floorplan generation via a diffusion model with discrete and continuous denoising. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5466–5475, 2023.

Sun, F.-Y., Liu, W., Gu, S., Lim, D., Bhat, G., Tombari, F., Li, M., Haber, N., and Wu, J. Layoutvlm: Differentiable optimization of 3d layout via vision-language models. *arXiv preprint arXiv:2412.02193*, 2024.

Szot, A., Clegg, A., Undersander, E., Wijmans, E., Zhao, Y., Turner, J., Maestre, N., Mukadam, M., Chaplot, D., Maksymets, O., Gokaslan, A., Vondrus, V., Dharur, S., Meier, F., Galuba, W., Chang, A., Kira, Z., Koltun, V., Malik, J., Savva, M., and Batra, D. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Tang, J., Nie, Y., Markhasin, L., Dai, A., Thies, J., and 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.

Tao, S., Xiang, F., Shukla, A., Qin, Y., Hinrichsen, X., Yuan, X., Bao, C., Lin, X., Liu, Y., kai Chan, T., Gao, Y., Li, X., Mu, T., Xiao, N., Gurha, A., Rajesh, V. N., Choi, Y. W., Chen, Y.-R., Huang, Z., Calandra, R., Chen, R., Luo, S., and Su, H. Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai. *Robotics: Science and Systems*, 2025.

Wang, K., Lin, Y.-A., Weissmann, B., Savva, M., Chang, A. X., and Ritchie, D. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior

networks. *ACM Transactions on Graphics (TOG)*, 38(4): 1–15, 2019.

Wang, Y., Xian, Z., Chen, F., Wang, T.-H., Wang, Y., Fragkiadaki, K., Erickson, Z., Held, D., and Gan, C. Robogen: Towards unleashing infinite data for automated robot learning via generative simulation. *arXiv preprint arXiv:2311.01455*, 2023.

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. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Wijmans, E., Kadian, A., Morcos, A., Lee, S., Essa, I., Parikh, D., Savva, M., and Batra, D. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019.

Wu, W., Fu, X.-M., Tang, R., Wang, Y., Qi, Y.-H., and Liu, L. Data-driven interior plan generation for residential buildings. *ACM Transactions on Graphics (TOG)*, 38(6): 1–12, 2019.

Xia, F., Zamir, A. R., He, Z., Sax, A., Malik, J., and Savarese, S. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9068–9079, 2018.

Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., and Chen, X. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.

Yang, X., Man, Y., Chen, J., and Wang, Y.-X. Scenecraft: Layout-guided 3d scene generation. *Advances in Neural Information Processing Systems*, 37:82060–82084, 2024a.

Yang, Y., Jia, B., Zhi, P., and Huang, S. Physcene: Physically interactable 3d scene synthesis for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16262–16272, 2024b.

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, 2024c.

Yu, L. F., Yeung, S. K., Tang, C. K., Terzopoulos, D., Chan, T. F., and Osher, S. J. Make it home: automatic optimization of furniture arrangement. *ACM Transactions on Graphics (TOG)-Proceedings of ACM SIGGRAPH 2011, v. 30,(4), July 2011, article no. 86*, 30(4), 2011.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Zhao, T. Z., Kumar, V., Levine, S., and Finn, C. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023.

Zhu, Y., Wong, J., Mandlekar, A., Martín-Martín, R., Joshi, A., Nasiriany, S., and Zhu, Y. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

A. Limitations & Future Work

Currently, our approach faces two main issues. First, the application scenarios are not yet broad enough, as it has only been tested in navigation experiments. Since the scene generation method is decoupled from the embodied simulation environment, it can also be applied to mobile manipulation tasks, thereby enhancing the value of scene generation. This is something we plan to focus on in the future. Secondly, during the experiments, we found that there is still a lack of high-quality simulation assets to support various simulation tasks, especially for manipulation tasks. The current assets at the part-level are very limited and cannot meet the demand for large-scale simulation training of embodied intelligent manipulation tasks. Therefore, we hope to introduce automation into the generation of simulation assets in the future, to provide a foundation for generating larger-scale and higher-quality scene generation methods.

B. Detials of evolutionary algorithm

In our experiments, the population size is set to $N = 10$, and the evolutionary algorithm runs for $T = 3$ iterations. In each iteration, we first select the top $k = 5$ elite individuals to form the set E. From E, we randomly sample $m = 3$ individuals as parents. These parents, along with the original user prompt, are provided to the LLM to generate a set of new individuals C_1 through contrastive prompting. Meanwhile, the LLM also generates an additional set C_2 of $n = 2$ individuals solely based on the original prompt, without any parent input. The next-generation population is then defined as $P' = E \cup C_1 \cup C_2$.

C. Backtracking algorithm

To solve the absolute positions of rooms given their sizes and adjacency relationships, we adopt a backtracking-based algorithm as shown in Algorithm 2. This algorithm incrementally places rooms while satisfying non-overlapping constraints and connectivity requirements.

Variable explanations:

- $\mathcal{A}$: the adjacency graph, where nodes are rooms and edges represent required connections between rooms.
- S: a mapping from room ID to its size (width and height).
- T_{max}: the maximum number of iterations allowed during the search process.
- R: the set of all room identifiers.
- r_0: the starting room, selected as the one with the highest degree in $\mathcal{A}$.
- solution: the current partial solution, mapping room IDs to their placed polygons (positions and orientations).
- visited: the set of rooms that have already been placed.
- $N_{\text{unvisited}}$: neighboring rooms of already placed rooms that are not yet visited.
- P_{cand}: candidate placements (position and rotation) for a room to be placed.

At the beginning of the algorithm, we select the room r_0 with the largest number of connections and place it at the origin $(0, 0)$ with a fixed orientation. Then we iteratively expand the layout by selecting unvisited rooms that are adjacent to the currently placed ones.

For each such room r_i, we generate candidate placements P_{cand} using the following procedure:

- For each already placed room r_j connected to r_i, we consider all exterior edges of r_j (top, bottom, left, right).
- Along each edge, we discretize possible contact points using a fixed step size (e.g., 1 unit grid).
- For each point, we attempt to place r_i at four different orientations: $0°$, $90°$, $180°$, and $270°$.
- We ensure that the candidate placement does not cause overlap with existing rooms and satisfies connectivity.

The candidate placements P_{cand} are shuffled to introduce randomness. For each candidate p, we temporarily update the solution and continue the search recursively. If a complete placement is found, the process stops. If no valid placement is possible for any candidate, we backtrack by reverting the last step and trying the next candidate.

This approach effectively explores the layout search space while ensuring spatial feasibility and topological consistency.

Algorithm 2 Room Placement using Backtracking

Require: $\mathcal{A}, S, T_{max}$
Initialize $R \leftarrow$ keys($\mathcal{A}$)
Select $r_0 \leftarrow$ room with largest degree in $\mathcal{A}$
Place r_0 at $(0, 0)$
Initialize solution $\leftarrow \{r_0\}$ and visited $\leftarrow \{r_0\}$
Initialize $t \leftarrow 0$
while |visited| $< |R|$ and $t < T_{max}$ **do**
  $N_{unvisited} \leftarrow$ unvisited neighbors of visited
  for each r_i in $N_{unvisited}$ **do**
    $P_{cand} \leftarrow$ generate valid polygons for r_i
    if $P_{cand} = \emptyset$ **then**
      continue
    end if
    shuffle P_{cand}
    for each p in P_{cand} **do**
      update_solution(solution, visited, r_i, p)
      if backtrack(solution, visited) **then**
        break
      end if
      revert_solution(solution, visited)
    end for
  end for
end while

D. Constraints for layout optimization

Constraints for furniture arrange are below:

1. **Global Constraints**: Whether the furniture needs to be near the edge of the room.
2. **Orientation Constraints**: Orientation constraints mainly include consistent orientation and focused orientation.
3. **Distance Constraints**: Whether the center-to-center distance between objects falls within a specified range.
4. **Boundary Constraints**: Furniture in a region should not exceed the region's boundaries.
5. **Collision Constraints**: Furniture within a region should not collide with one another.
6. **Accessible Constraints**: Furniture is considered accessible if there exists at least one path from any door in the room to the front side of the furniture. We use the A* algorithm to compute the shortest such path.

E. Navigation Experiment Details

The navigation experiments are conducted on an Intel(R) Xeon(R) Gold 6326 CPU @ 2.90GHz, using 8 NVIDIA 3090 GPUs.

Task Setting. We select the ObjectNav (Batra et al., 2020) task for evaluation, where the objective is to locate a specified category of objects. Two well-established navigation benchmarks, RoboTHOR (Deitke et al., 2020) and HM3D (Ramakrishnan et al., 2021), are chosen as the primary test environments. The target object categories remain consistent

with the benchmark settings. Specifically, for the RoboTHOR benchmark, we designed scenes containing 12 object categories: `AlarmClock`, `Apple`, `BaseballBat`, `Basketball`, `Bowl`, `GarbageCan`, `HousePlant`, `Laptop`, `Mug`, `SprayBottle`, `Television`, and `Vase`. For the HM3D benchmark, we design scenes featuring 6 object categories: `Bed`, `Chair`, `Sofa`, `TV`, `Plant`, and `Toilet`. In both tasks, the agent follows the predefined action space consisting of `STOP`, `MOVE_FORWARD`, `TURN_LEFT`, `TURN_RIGHT`, `LOOK_UP`, and `LOOK_DOWN`. The movement step size is set to 0.25m, and the turning angle is fixed at 30 degrees. The success criterion for task completion is also consistent across the two benchmarks: the agent must determine that it is within a certain distance from the target object and can visually perceive it from a particular angle. At this point, executing the STOP action signifies task completion.

Algorithm Architecture. We follow (Khandelwal et al., 2022) and employ Decentralized Distributed Proximal Policy Optimization(Wijmans et al., 2019) (DDPPO) as the primary training framework for the agent's behavior policy. The Proximal Policy Optimization(Schulman et al., 2017) (PPO) algorithm takes multiple sensor data modalities and the agent's previous action as inputs. These inputs require specific preprocessing through dedicated networks. The RGB sensor data is processed using the CLIP model, while object category information, GPS, Compass, and previous action data are handled using Fully Connected (FC) layers. The processed features are then concatenated and fed into an RNN network, whose output is subsequently passed through a linear layer to generate the final state representation for PPO. The PPO algorithm then outputs a probability distribution over possible actions. For training, we utilize eight NVIDIA 3090 GPUs with the AI2-THOR simulation engine. Given the substantial memory consumption of individual environments, each GPU hosts ten environment workers.

Experiment Details. The generated scenes are split into train, validation, and test sets in a 60/20/20 ratio. The generated scenes are divided into training, validation, and test sets following a 60/20/20 split. For the RoboTHOR benchmark, each scene consist of at most three rooms. During training, each scene generates 1,000 episodes, while the validation and test scenes generate 150 and 500 episodes, respectively. For the HM3D benchmark, each scenario contains at most four rooms. Each training scene generates 2,000 episodes, with validation and test scenes generating 150 and 500 episodes, respectively. To assess the zero-shot generalization capability of the trained models, a learn-from-scratch approach is employed. The models are trained from scratch on the generated datasets, and the best performing models are subsequently evaluated on the RoboTHOR and HM3D environments.

Evaluation Metrics. In the ObjectNav task, agent performance is assessed using two widely adopted metrics: *Success* and *SPL*.

Success quantifies whether the agent completes the navigation task by reaching the target object and issuing the `STOP` action within a predefined distance threshold. It is defined as:

$$\text{Success} = \frac{1}{N}\sum_{i=1}^{N} S_i$$

where N is the total number of evaluation episodes, and $S_i \in \{0, 1\}$ indicates whether the i-th episode is successful.

SPL further incorporates path efficiency by penalizing unnecessarily long trajectories, even in successful episodes. It is computed as:

$$\text{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \cdot \frac{l_i}{\max(p_i, l_i)}$$

where l_i is the geodesic shortest distance from the start to the goal in episode i, and p_i is the actual path length taken by the agent. SPL values range from 0 to 1, with higher scores indicating more efficient and successful navigation.

F. Method Implementation Details

We use GPT-4o (OpenAI, 2024) as both the LLM and the VLM in our framework. We use five representative VLMs for evaluation: GPT-4, GPT-4o, GPT-4o-mini, Gemini 2.0 Flash (DeepMind, 2024), and Claude 3 Opus (Anthropic, 2024). The scene generation experiments are conducted on an Intel(R) Xeon(R) Platinum 8171M CPU @ 2.60GHz.

F.1. Propmts of EmbodiedScene

The method is mainly based on LLM implementation. Below are the important prompts in the system.

Prompt 1: initial prompt template

You are an experienced room designer. Now, please design a house according to my requirements.
Here are the overall requirements for the room design:
${input}

Prompt 2: style prompt template

I will ask you to provide design requirements in stages. Now, please only provide the overall design style, color scheme, and material selection for the rooms.
Please return your answer in the following JSON format.

```
{
    "style": "Design style, please answer in a complete sentence, return a string",
    "color": "Color scheme, please answer in a complete sentence, return a string",
    "material": "Material selection, please answer in a complete sentence, return a string",
    "wall_height": "Wall height, return a float"
}
```

Please strictly follow the JSON format, only include keys and values, do not add comments, and do not include any other content.

Prompt 3: room prompt template

I will ask you to provide design requirements in stages. You have already provided the overall design style, color scheme, and material selection for the house. Here is your previous answer.
${style}
Now, please provide the names, sizes, and style information for all rooms.
Please return your answer in the following JSON format. Each room name as the key, and room size and style as the value.

```
{
    "Room Name": {
        "style": "Room style, please answer in a complete sentence, return a string",
        "floor": "Floor material, please answer in a complete sentence, return a string",
        "wall": "Wall material, please answer in a complete sentence, return a string",
        "size": "Room size, return a list containing two elements representing the length and width of the room in meters",
        "adjacency": "Room connections, given as a list where each element is a string representing the name of a room connected to the current room"
    }
}
```

Please strictly follow the JSON format, only include keys and values, do not add comments, and do not include any other content.

Prompt 4: door prompt template

I will ask you to provide design requirements in stages. You have already provided the connections between all rooms. Here is your previous answer.
All room names are: ${rooms}
The connections between rooms are:
${connections}
Now, please provide the names, sizes, and style information for all doors based on the room connections.
The connections could be of three types: doorframe(no door), doorway (with a door), or open (no wall separating rooms). The sizes available for doorframes or doorways are single (1m wide) and double (2m wide).
Please return your answer in the following JSON format. Each door name as the key, and door information as the value.

```
{
    "door-0": {   # example of a double doorway
        "room0": "Connected room name, return a string",    # the names of the two connected rooms
        "room1": "Connected room name, return a string",
        "type": "doorway",
        "size": "double",
        "style": "dark brown metal door"
    },
    "door-1": {  # example of open
        "room0": room0's name,    # the names of the two connected rooms
        "room1": room1's name,
        "type": "open",
        "size": "N/A",
        "style": "N/A"
    },
    "door-2": {   # example of single doorframe
        "room0": room0's name,    # the names of the two connected rooms
        "room1": room1's name,
        "type": "doorframe",
        "size": "single",
        "style": "wooden door with white frames"
    }
}
```

Please strictly follow the JSON format, only include keys and values, do not add comments, and do not include any other content.

Prompt 5: window prompt template

I will ask you to provide design requirements in stages. You have already provided the sizes and styles of all rooms. Here is your previous answer.
${rooms}
Now, please provide the names, sizes, number and style information for all windows based on the room sizes and styles.
The window types are: bay, hung, and slider. The width and height of the windows are both between 0.5 and 2 meters.
Please return your answer in the following JSON format. Each window name as the key, and window size, number and style as the value.

```
{
    "Window Name": {
        "room": "A string representing the name of the room where the window is located",
        "type": "Window type, return a string, only three types available: 'bay', 'hung', 'slider'",
        "size": "Window size, return a list containing two elements representing the length and width of the window in meters",
        "quantity": "Window number, return a integer",
        "style": "Window style, please answer in a complete sentence, return a string",
        "height": "Window base height (cm from floor), return a float"
    }
}
```

Please strictly follow the JSON format, only include keys and values, do not add comments, and do not include any other content.

Prompt 6: region prompt template

I will ask you to provide design requirements in stages. You have already provided the sizes and styles of all rooms. Here is your previous answer.
${rooms}
Now, please provide the corresponding functional areas for all rooms based on their sizes and styles.
Please return your answer in the following JSON format. Each room name as the key, and a list containing all functional areas as the value.

```
{
    "Room Name": [
        "Functional Area Name",
        "Functional Area Name",
        ...
    ]
}
```

Please strictly follow the JSON format, only include keys and values, do not add comments, and do not include any other content.

Prompt 7: object prompt template

I will ask you to provide design requirements in stages. You have already provided the size and style of ${room_name} and its functional areas. Here is your previous answer.
${room_info}
Now, please provide the names, sizes, and style information for all furniture based on the room size and style. Furniture is divided into floor furniture and wall furniture. Please return your answer in the following JSON format. Each functional area name as the key, and included furniture as the value.

```
{
    "Functional Area Name": {
        "floor": [
            {
                "name": "Furniture name",
                "number": "Furniture number, return a integer",
                "size": "Furniture size, return a list containing three elements representing the length, width and height of the furniture in meters",
                "description": "Furniture style, please answer in a complete sentence, return a string",
                "children": [
                    {
                        "name": "Small furniture name",
                        "number": "Small furniture number, return a integer",
                        "size": "Small furniture size, return a list containing three elements representing the length, width and height of the furniture in meters",
                        "style": "Small furniture style, please answer in a complete sentence, return a string"
                    }
                ]
            }
        ],
        "wall": [
            {
                "name": "Wall furniture name",
                "number": "Wall furniture number, return a integer",
                "size": "Wall furniture size, return a list containing three elements representing the length, width and height of the furniture in meters",
                "style": "Wall furniture style, please answer in a complete sentence, return a string",
                "children": [
                    {
                        "name": "Small furniture name",
                        "number": "Small furniture number, return a integer",
                        "size": "Small furniture size, return a list containing three elements representing the length, width and height of the furniture in meters",
                        "style": "Small furniture style, please answer in a complete sentence, return a string"
                    }
                ]
            }
        ]
    }
}
```

Please strictly follow the JSON format, only include keys and values, do not add comments, and do not include any other content.

Prompt 8: score prompt template

You are an experienced interior room designer. Now you need to score the following house sketch, with a score range of 0-10, where higher scores indicate better design.
The initial requirements for the house design are:
${query}
The designed house information:
${scene}
Please directly return your score as a float, without any additional characters or explanations.

Prompt 9: critic prompt template

You are an experienced room planner and I want to design a reasonable room plan.
The stage being designed now is "${stage}". Please judge my design from the following perspectives:
${rules}
The overall instruction of the design is as follows:
${query}
The design you have created is as follows:
${design}
Your response should be returned in the following JSON format:

```
{
    "reason": Please provide your reasons for each point based on the criteria I
    have given,
    "answer": Give a result based on the reasons, true or false,
    "rules": A list of the violated rules
}
```

Note: The JSON string should not contain comments, as I will use ‘json.loads()‘ to load your answer.

G. Potential Social Impact

EmbodiedScene enables automated generation of 3D scenes using LLMs, which may benefit applications such as simulation, embodied AI training, and game design. However, generated environments may reflect biases from training data or produce unrealistic layouts, especially in safety-critical contexts. We recommend applying this framework with human oversight when used in sensitive or real-world applications.