I-Design: Personalized LLM Interior Designer

摘要

室内设计让我们可以做自己，过自己想要的生活——每一种设计都像我们独特的个性一样独特。然而，对于非专业人士来说，表达和实现这一点并非易事，因为它需要将功能和视觉期望与物理空间的限制相结合；这使得室内设计成为一种奢侈。为了让它更容易被接受，我们推出了 I-Design，一个个性化的室内设计师，它允许用户通过自然语言交流生成和可视化他们的设计目标。I-Design 从一组大型语言模型代理开始，这些代理相互进行对话和逻辑推理，将文本用户输入转换为具有相对对象关系的可行场景图设计。随后，有效的放置算法确定场景中每个对象的最佳位置。然后通过从现有对象数据库中检索和集成资产，以 3D 形式构建最终设计。此外，我们提出了一种新的评估协议，该协议利用视觉语言模型并补充设计流程。大量定量和定性实验表明，I-Design 在提供高质量 3D 设计解决方案和与用户输入相匹配的抽象概念方面优于现有方法，展示了其在详细 3D 排列和概念保真度方面的优势。

关键词：大语言模型；文本转 3D；场景图；检索；室内设计；3D 室内场景合成

1 引言

1.1 研究背景

室内设计能够让我们展现自我并实现理想的生活方式。然而，寻找或创造完美的空间来满足我们的生活方式、愿望和需求对于非专业人士来说是一件极其困难的事情。因此，在本篇论文中，作者解决了具有挑战性的 3D 室内场景合成任务，结合用户对非结构化文本的描述，为没有专业设计背景的普通人提供了一种满足用户个性化需求的 3D 设计解决方案。

特定的生成模型 [1–3] 和数据驱动 [4–6] 的方法，展示了生成多样化和真实室内布局的显著能力。然而，它们的表现受限于用于训练的封闭集和有限数据集，这些数据集不可避免地是有偏且不完整的样本。此外，这些接受用户文本输入的方法只能利用具有预定义语法和规则的结构化文本。因此，引导这些模型生成符合用户偏好的，适用于现实世界中未见过的室内布局的实用设计具有挑战性。

然而，3D 室内场景合成任务需要超越任何专门的数据驱动模型的推理能力，在理解设计原则和概念，比如对象的选择、风格的适配以及空间的布局上，存在着很大的挑战。鉴于最近的技术进步，使用大语言模型 (LLMs) 进行 3D 室内场景合成有效的克服了这些障碍，例如，LayoutGPT [7] 通过大语言模型直接预测场景中物体的绝对位置来摆放物体。尽管这种技术在处理少量物体的小规模排列时有效，但在生成包含数十个以复杂方式相互关联的物体的逼真场景时，生成的效果会不尽人意。针对这个问题，在本篇论文中，作者利用多个大语言模型

Agents 分别负责不同的功能，来解决单个大语言模型 Agent 难以处理的 3D 室内场景合成任务中的挑战。

2 相关工作

2.1 基于大语言模型的 3D 场景合成

最初的场景合成方法集成了多个大语言模型，将用户文本输入编码为随后用于对象放置过程的向量表示 [4, 8–10]。[7] 通过专门使用 GPT 模型来生成 3D 室内场景表示，扩大了 LayoutGPT 的范围。它的功能就像一个检索系统，采用基于绝对坐标的策略来定位场景中的物体。[11] 过开发一种无数据集、开放词汇的方法来生成 3D 家居布局，进一步推进了这条研究路线。然而，这个研究方向经常遇到来自现有 GPT 模型在几何推理方面的局限性的挑战，导致场景中物体重叠或放置在场景边界之外。

2.2 基于生成模型的 3D 场景合成

3D 场景合成的一种实现方式涉及根据用户定义的规格生成多视角一致的图像集或全景图，并将这些 2D 表示转换为 3D 场景。[2] 和 [1] 使用虚拟相机在空间中导航，并通过图像修复、单目深度估计、3D 提升和拼接迭代生成图像。另一种方法是 [3]，利用对应感知的注意力模块在一次处理中生成多视角一致的全景图或图像集。

尽管利用了 2D 生成模型的能力，当前的方法在整合 3D 几何约束方面仍存在困难，而这些约束对于实际的室内设计应用（如平面图和墙壁）至关重要。[12] 试图解决这一挑战，但伪影问题仍然存在。此外，基于单目图像的深度估计 [13] 和 3D 提升 [2] 为最终的 3D 网格输出引入了不确定性，跨视图的松散语义耦合在室内场景合成中带来了挑战，导致不现实的场景，例如一个卧室中有多张床 [3, 12]。

2.3 基于先验学习的 3D 场景合成

另一种数据驱动的方法将解决 3D 室内场景合成（3DISS）的过程分为两个不同的步骤：3D 资产选择和布局合成，即确定房间的家具组合及其摆放位置。

在 3D 资产选择方面，主要有两种方法：(1) 基于数据集的检索和 (2) 通过生成模型进行合成。随着广泛且高质量的资产数据集的可用性，人们可以检索到合适的模型，例如使用语义嵌入 [14]，或者，生成模型可以根据文本或图像输入生成适当的 3D 资产 [15–18]。

在家具布置方面，经典方法通过应用特定规则来指导家具的摆放，构建用于程序化建模的语法，或通过人工交互进行编辑。随着大规模 3D 室内数据集的兴起，最近的研究转向通过生成模型从专家设计的布局中学习。[19] 和 [10] 采用基于 Transformer 的模型，以自回归的方式自主合成室内环境，依次选择和放置物体。[5] 通过学习人类对规则性的标准，通过基于 Transformer 的类似扩散的流水线来优化初始的粗略房间布局。最近的 [4] 和 [6] 利用扩散模型生成室内场景，将场景表示为场景图。

3 本文方法

如图1所示，用户以自然语言的形式指定设计偏好，I-Design 通过查询 LLM agent 生成房间物品，物品的属性以及以场景图形式表示的相对关系，通过使用回溯算法来解决场景图中物体的摆放关系。场景图确定后，根据房间的功能和风格从 3D Assets 中检索物品进行填充，并在 3D Composer 中组成最后的结果。

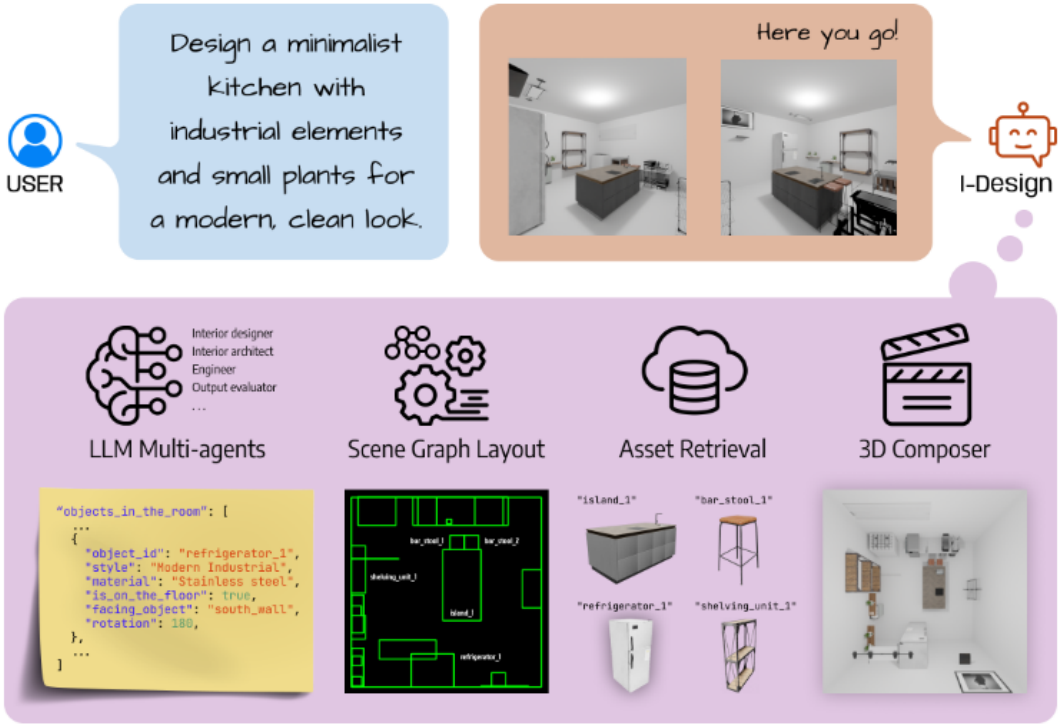

图 1. I-Design 结构图

3.1 输入输出

输入：用自然语言描述的设计需求的纯文本 prompt

输出：符合用户设计需求的 3D 室内设计图，包括场景图、平面图、预配置的渲染视图以及可解释的工件

3.2 LLM 多智能体代理

这篇论文采用多个大型语言模型 (LLMs) 进行代理，每个代理负责不同的职责和任务。如图2所示，系统将用户的输入解释并转换为功能性和个性化的五个不同的代理：Interior Designer、Interior Architect、Engineer、Layout Corrector 和 Layout Refiner。其中，Interior Designer 负责将用户的输入、房间的尺寸以及所需的物体的数量，并根据用户输入的偏好提出一组对象的选择，并确保它们的功能与房间类型相匹配。Interior Architect 负责建立对象与对象之间的连接，以及对象与房间布局的关系。Engineer 将相对场景图转换为根据制定模式结构化的 JSON 对象，并生成一个包含对象详细信息的 JSON 文件，智能体通过 JSON 模式验证器来评估生成文件的有效性，如果不符合要求，则需要修改。Layout Corrector 负责修复

场景图中的无效连接，包括移除空间上不合理的边以及消除节点之间的歧义。最后，Layout Refiner 负责消除场景图中同一父节点下的子节点之间的歧义，确保子节点在场景图中的相对位置明确无误。

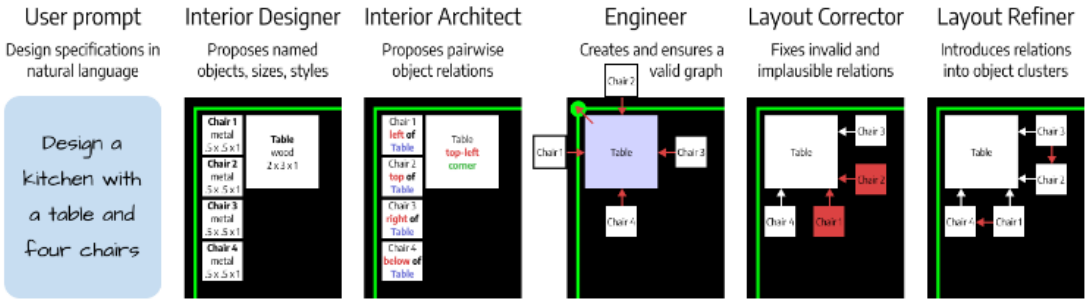

图 2. 多个 LLM 代理的场景图生成

3.3 场景图布局

在最终的后处理阶段，需要使用回溯算法对场景图中的对象的摆放位置进行定位。回溯算法用于将场景图 G 的相对表示转换为对象位置 P，房间最初填充了场景图 G 的根结点，这些根结点代表房间布局的基本元素，如墙壁和天花板。这些根结点的位置保持固定，而其他对象则围绕它们进行排列，图3展示了对场景图使用回溯算法进行处理的一个样例过程。

图 3. 基于回溯算法的场景图布局

3.4 3D 资产检索

使用对象的名称、风格和材质信息为每个对象生成文本描述，然后通过 CLIP [20] 文本编码器转换为文本嵌入。通过将 OpenShape [14] 编码与 CLIP 文本嵌入进行对齐，从而测量文本嵌入与所选数据库中学习到的对象表示之间的距离，这有助于检索与对象的文本描述最接近的 3D 资产，检索完成后，可以调整资产以适应场景图中为这些对象提供的边界框。

3.5 3D 合成器

采用一个现成的 3D 渲染器将生成最终的场景图并检索了 3D 资产后的房间内部以 3D 形式可视化。输出的结果包括 (1) 场景图,(2) 平面图,(3) 预配置的渲染视图, 以及 (4) 可解释性工件。用户可以对家具进行更换或从不同的视角去渲染场景。

4 复现细节

4.1 与已有开源代码对比

该论文代码已开源，链接如下：https://github.com/atcelen/IDesign/。与已经开源的代码相比，针对原文里提到的 3D 资产检索存在的物体质量不同等问题，采用基于文本的 3D 物体生成方法进行替代。

4.2 实验环境搭建

从 GitHub 上下载源代码到 Linux 服务器，切换到项目文件夹后，在 Linux 终端运行 conda create -n idesign python=3.9 命令安装项目的 conda 环境，成功安装完环境后运行 conda activate idesign 命令从 conda 的 base 环境切换到 idesign 环境。运行 pip install -r requirements.txt 命令安装 requirements.txt 里的包；运行 conda install pytorch==1.12.1 torchvision==0.13.1 torchaudio==0.12.1 cudatoolkit=11.3 -c pytorch 命令安装 pytorch 和 cuda；运行 pip install -U git+https://github.com/NVIDIA/MinkowskiEngine 命令安装 MinkowskiEngine 引擎；运行 conda install -c dglteam/label/cu113 dgl 安装 dgl 库。

(2) 创建”`OAI_CONFIG_LIST.json`” 文件夹，将 api-key 换成 OpenAI 的密钥，如图4所示。

```
[
    {
        "model": "gpt-4",
        "api_key": "sk-lvgvGXmqP9CnYBok98PwSLvL9ldQ4dygEILddJilCzQyF2xR"
    },
    {
        "model": "gpt-4-1106-preview",
        "api_key": "sk-lvgvGXmqP9CnYBok98PwSLvL9ldQ4dygEILddJilCzQyF2xR"
    },
    {
        "model": "gpt-3.5-turbo-1106",
        "api_key": "sk-lvgvGXmqP9CnYBok98PwSLvL9ldQ4dygEILddJilCzQyF2xR",
        "api_version": "2023-03-01-preview"
    }
]
```

图 4. 配置 OpenAI 的 api-key

4.3 案例测试

在源代码里，作者在 test.py 文件里给了一个测试用例，生成一个长 4 米，宽 4 米，高 2.5 米的一个有创造性和活力的客厅，如图5所示。

```python
from IDesign import IDesign

i_design = IDesign(no_of_objects = 15,
                   user_input = "A creative vibrant living room",
                   room_dimensions = [4.0, 4.0, 2.5])

i_design.create_initial_design()
i_design.correct_design()
i_design.refine_design(verbose=True)
i_design.create_object_clusters(verbose=False)
i_design.backtrack(verbose=True)
i_design.to_json()
```

图 5. 案例测试

4.3.1 创建室内场景图

运行 python test.py 创建一个长 4 米，宽 4 米，高 2.5 米的文本为"A creative vibrant living room" 的室内场景图。

4.3.2 检索 3D 资产进行填充

运行 git clone https://huggingface.co/OpenShape/openshape-demo-support 命令从 huggingface 中下载 OpenShape，无法访问 huggingface 的话可以把库的压缩包下载到 Windows 本地，然后在 Windows PowerShell 终端通过 scp 命令将压缩包传到 Linux 服务器里进行解压缩。之后运行 cd openshape-demo-support 命令切换到 openshape 的文件夹，并运行 pip install –e . 命令安装 openshape。最后运行 python retrieve.py 对场景图进行 3D 资产的寻找和填充。

4.3.3 对场景图进行 3D 渲染

使用 pip install bpy 安装 blender 库，然后运行 python `place_in_blender.py` 对场景图和 3D 资产进行 3D 渲染。

4.3.4 运行结果

图 6. 运行结果

4.3.5 使用 GPT-4V 对生成的室内设计图进行评价

运行 python `gpt_v_as_evaluator.py` 命令，生成对设计图的评价，结果如图7所示。


```
{
  "realism_and_3d_geometric_consistency": {
    "grade": 7,
    "comment": "The render demonstrates reasonable 3D geometry with accurate proportions and perspective. However, some elements, like the chandelier and wall-mounted objects, lack detailed shadowing or texture depth, slightly reducing realism."
  },
  "functionality_and_activity_based_alignment": {
    "grade": 6,
    "comment": "The room features a couch, armchairs, and a central table, but their arrangement does not optimize usability for TV watching or collaborative activities. The wall-mounted TV is present but positioned too high for comfortable viewing."
  },
  "layout_and_furniture": {
    "grade": 6,
    "comment": "The layout provides basic functionality, but the central arrangement of the furniture feels cluttered. Additionally, the bookshelf and wall decorations appear more decorative than functional, which may limit practical usage."
  },
  "color_scheme_and_material_choices": {
    "grade": 8,
    "comment": "The room uses a pink and white color scheme that is cohesive and pleasant. While the color choice is subjective, the palette may not cater to all user preferences, particularly for neutral or professional tones."
  },
  "overall_aesthetic_and_atmosphere": {
    "grade": 7,
    "comment": "The room's aesthetic leans toward playful and vibrant, creating a lively atmosphere. However, it lacks the balance of warmth or sophistication that some users may prefer for a mixed-purpose space."
  }
}
```


图 7. 使用 gpt-4v 的评估结果图

4.4 创新点

I-Design 在 3D 家具填充上选择的方法是从现有大型 3D 资产存储库中进行检索得到的，这存在许多问题，比如（1）由于语义理解的局限性，检索到的 3D 资产可能在细节上与用户期望的描述不完全一致。(2) 从数据库中检索的 3D 资产质量可能参差不齐。(3) 许多 3D 资产具有默认的方向，可能与用户期望的方向不一样，且方向无法进行手动调整。(4) 3D 资产的实际尺寸可能与用户描述或场景图中指定的尺寸不完全匹配。(5) 尽管大型 3D 资产数据库提供了丰富的选择，但在某些特定设计需求下，可能难以找到完全符合要求的资产，限制了设计的灵活性和多样性。

针对从大型 3D 数据库中检索 3D 资产存在的问题，本文使用一种基于文本的 3D 资产生成的方法进行尝试替代。通过参考 [21] 中提出的基于隐式函数的条件控制 3D 生成模型来基于文本进行 3D 资产的生成。具体来说，首先先训练一个编码器来产生隐式表示，然后在编码器产生的潜在表示上训练扩散模型。步骤如下：(1) 如图8，训练一个编码器将三维资产映射到隐函数的参数。编码器接收点云和多视角渲染图像作为输入，输出一个多层感知器（MLP）的参数，该参数表示隐函数。(2) 其次，在编码器的输出上训练一个条件扩散模型。扩散模型在潜在空间中进行高斯噪声的逐步去除，最终生成隐函数的参数。扩散模型的损失函数为：

$$L_{x_0} = \mathbb{E}_{x_0 \sim q(x_0), \epsilon \sim \mathcal{N}(0,I), t \sim U[1,T]} \left\| x_\theta(x_t, t) - x_0 \right\|_2^2 \tag{1}$$

其中，x_0 是去噪后的样本，ϵ 是随机噪声，t 是时间步，$x_\theta(x_t, t)$ 是扩散模型的预测值。

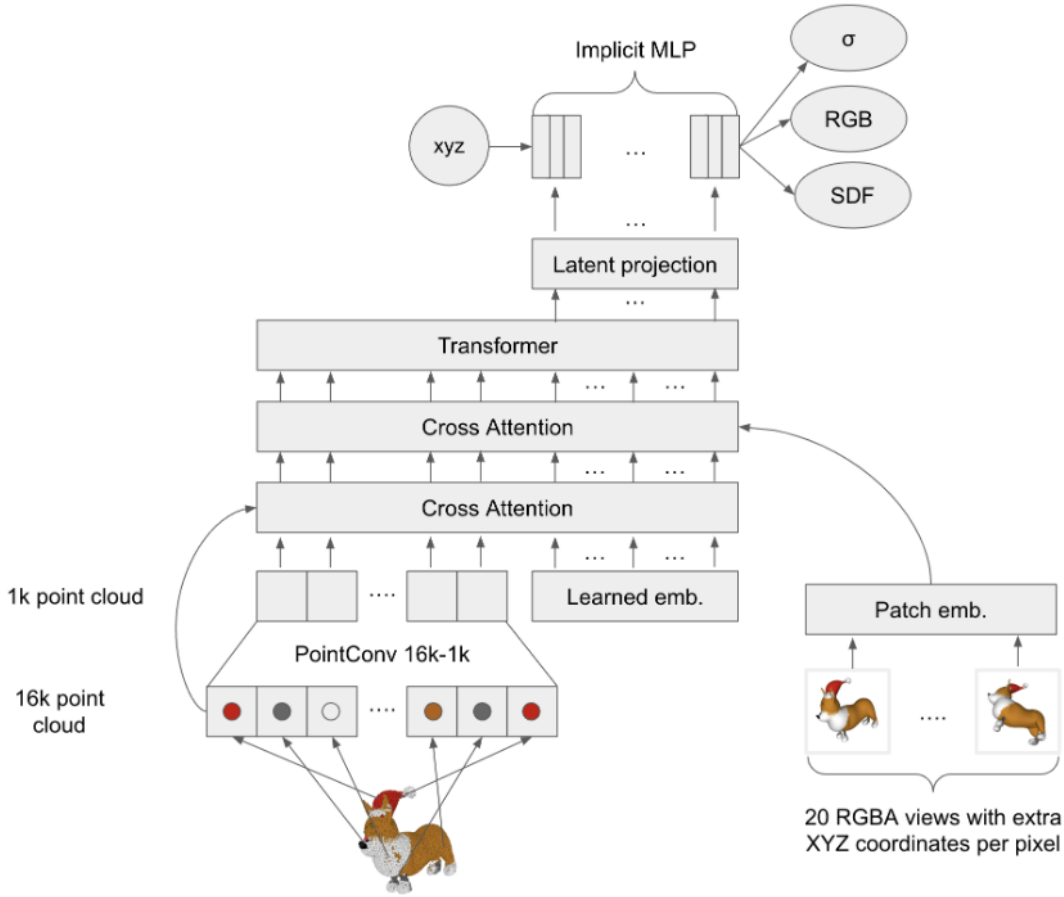

图 8. 编码器结构图

5 实验结果分析

论文使用微软的 AutoGen 框架来实现多智能体对话，每个智能体都配备了 GPT-4 模型。temperature 设置为 0.7，`top_p` 参数设置为 1.0。对于对象检索，论文依赖 OpenShape，利用文本嵌入从 Objaverse 中检索对象。场景随后使用 Blender 进行可视化和渲染。

5.1 定量评估

选择 LayoutGPT 作为一个 baseline 与 I-Design 进行对比实验，因为 LayoutGPT 仅支持客厅和卧室两个类型，为了公平起见，比较的范围仅限于这两种类型及其房间尺寸的变化。从 4 个指标将 I-Design 和 LayoutGPT 进行对比实验，如表 1 所示，其中，Average Number of Proposed Objects(NObj) 通过放置的对象数量来评估生成房间的多样性；Out-of-Boundary Rates (OOB) 通过检查物体超出房间边界的比率来评价生成房间布局的可行性；Bounding Box Loss (BBL) 通过评估所提出的家具边界框之间的重叠程度来评价生成房间布局的可行性；GPT-4V ratings 是通过 GPT-4V 模型基于渲染结果评估合成房间的视觉质量。评分标准涵盖功能性和基于活动的对齐、布局和家具、配色方案和材料选择，以及整体美学和氛围。

使用 GPT-4V 模型对生成的场景进行视觉质量评分，涵盖功能性、布局、颜色方案和整体氛围等方面，评估结果如表 2 所示。

Method	Room Type	NObj ↑	OOB ↓	BBL ↓	GPT-4V Criteria ↑				
					Func.	Layout.	Scheme.	Atmos.	Avg.
LayoutGPT	Bedroom	5.5	51.06	14.09	$4.8_{\pm 0.4}$	$4.8_{\pm 0.8}$	$4.6_{\pm 0.8}$	$4.9_{\pm 0.4}$	4.8
	LivingRoom	6.9	64.15	1.06	$4.8_{\pm 1.3}$	$4.8_{\pm 0.8}$	$4.6_{\pm 0.8}$	$4.6_{\pm 0.8}$	4.8
	Avg.	6.2	57.6	7.58	4.8	4.8	4.6	4.8	4.8
Ours (1-Design)	Bedroom	12.7	0.0	0.34	$5.2_{\pm 0.4}$	$5.5_{\pm 0.2}$	$5.3_{\pm 0.6}$	$5.5_{\pm 0.2}$	5.5
	LivingRoom	23.6	0.0	0.31	$5.8_{\pm 0.6}$	$5.8_{\pm 0.2}$	$5.9_{\pm 1.1}$	$5.7_{\pm 1.3}$	5.8
	Avg.	18.2	0.0	0.33	5.5	5.6	5.8	5.6	5.7

表 1. 与 Layout-GPT 的定量比较

Prompt type	**GPT-4V Criteria**				
	Func.	**Layout.**	**Schema.**	**Atmos.**	**Avg.**
Atmospheric	$4.9_{\pm 0.7}$	$4.9_{\pm 0.6}$	$4.3_{\pm 0.6}$	$4.4_{\pm 0.5}$	4.9
Scheme	$4.6_{\pm 0.5}$	$4.8_{\pm 0.6}$	$5.0_{\pm 0.5}$	$4.8_{\pm 0.5}$	5.0
Layout	$6.2_{\pm 0.\ 6}$	$6.0_{\pm 0.7}$	$5.2_{\pm 0.7}$	$5.6_{\pm 0.6}$	5.8
Functional	$7.0_{\pm 0.7}$	$6.5_{\pm 0.7}$	$5.8_{\pm 0.8}$	$6.2_{\pm 0.7}$	6.4

表 2. GPT-4V 评价的 I-Design 结果

5.2 用户实验

通过 I-Design 创建了 20 个场景，并通过 LayoutGPT 创建了另外 20 个场景。从每个样本的竞争对手汇聚中随机抽取了 5 个场景，从而组成了 100 对。每对场景描绘了一个卧室或客厅，每个场景包含 I-Design 的渲染图和 LayoutGPT 的渲染图。用户在每对场景中选择一个觉得最真实的场景，结果如图 3.1.2.1 所示，I-Design 的生成效果要优于 LayoutGPT 的生成效果。将最终收集的 1254 个卧室的结果和 660 个客厅的结果转换为 Bradley-Terry 偏好分数和概率，得到了图9和表 3 的结果。

(a) Bedroom Control Scenes

图 9. 用户评价

Method	Room Type	Bradley-Terry score ↑	Prob. ↑
LayoutGPT	Bedroom	0.12	0.42
	LivingRoom	0.17	0.38
Ours	Bedroom	**0.40**	**0.58**
	LivingRoom	**0.69**	**0.62**

表 3. 最终的用户评价结果

5.3 定性分析

如图10所示，每个合成的房间都巧妙地反映了用户文本输入的规格。当用户明确提到家具偏好和期望的位置时，如第四个样例所示，生成的房间与这些要求无缝对齐。同样，当规格是隐式的时，如第三个样例中展示的为有幼儿的家庭量身定制的卧室，合成房间的功能性满足了用户的偏好，婴儿床被放置在特大号床旁边。

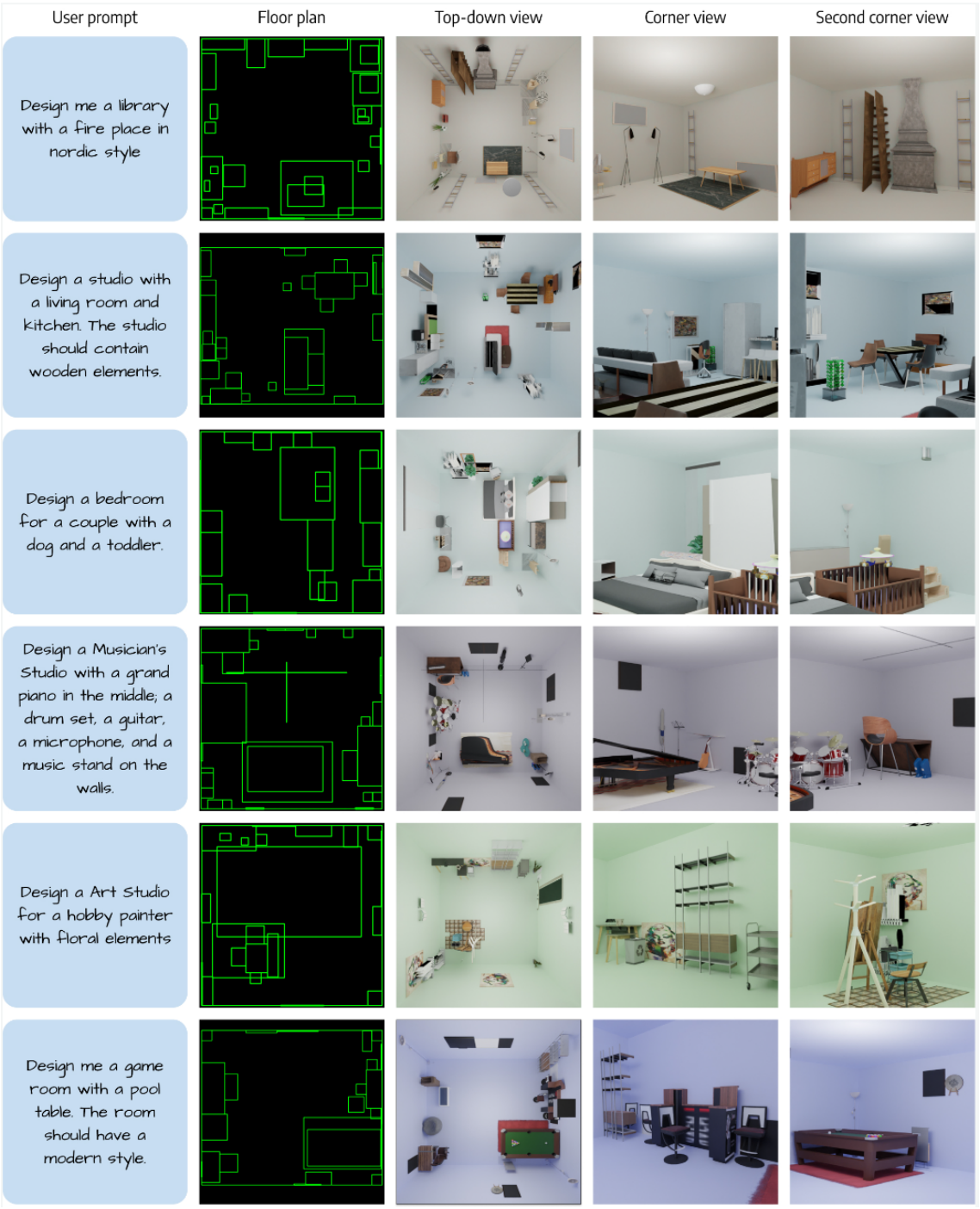

图 10. 结果示例图

5.4 3D 资产生成结果分析

Shap-E 的客户端代码已开源，链接如下：https://github.com/openai/shap-e。

在 Shap-E 的客户端界面的文本 Prompt 输入框里输入 “a black and wooden table”，结果如图11所示。

图 11. 一个黑色的木质桌子

输入 “a white and soft bed”，结果如图12所示。

图 12. 一个白色柔软的床

尽管 Shap-E 可以生成可识别的 3D 资产，但生成的样本通常看起来粗糙或缺乏细节。编码器有时会丢失详细的纹理信息，未来可以在编码器的结构上进行一些改进，使用更复杂的网络结构或增加网络深度，确保编码器能够充分捕捉输入 3D 资产的细节特征。

6 总结与展望

这篇论文通过引入多代理架构和场景图表示，提出了一种新的室内设计方法。I-Design 能够将用户的自然语言输入转化为高质量的 3D 设计解决方案，展示出在详细 3D 安排和概念保真度方面的优势。尽管如此，该方法仍面临一些局限性，如对象放置的终止问题和资产检索的质量问题。未来的研究方向包括探索基于学习的自动化对象放置方法和生成替代或补充资产检索过程的方法。

针对论文里使用从 3D 数据库中进行 3D 资产检索存在的问题，使用基于生成式的 3D 资产生成方法进行替代。由生成结果可知，目前的 3D 资产生成方法仍存在明显的缺陷，比如生成的家具的纹理很粗糙，且对颜色的理解有偏差。未来可以与基于优化的 3D 生成技术结合使用。例如，可以使用 Shap · E 生成的 NeRF 或网格来初始化一种基于优化的方法，例如采用论文 [18] 中描述的分数蒸馏采样方法，即通过在可微分图像参数化中进行优化来实现采样的方式确保生成高保真度的 3D 对象。

参考文献

[1] Jaeyoung Chung, Suyoung Lee, Hyeongjin Nam, Jaerin Lee, and Kyoung Mu Lee. LucidDreamer: Domain-free Generation of 3D Gaussian Splatting Scenes. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2023.

[2] Lukas Höllein, Ang Cao, Andrew Owens, Justin Johnson, and Matthias Nießner. Text2Room: Extracting Textured 3D Meshes from 2D Text-to-Image Models. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2023.

[3] Shitao Tang, Fuyang Zhang, Jiacheng Chen, Peng Wang, and Yasutaka Furukawa. MVDiffusion: Enabling Holistic Multi-view Image Generation with Correspondence-Aware Diffusion. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2023.

[4] Jiapeng Tang, Yinyu Nie, Lev Markhasin, Angela Dai, Justus Thies, and Matthias Nießner. DiffuScene: Scene Graph Denoising Diffusion Probabilistic Model for Generative Indoor Scene Synthesis. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2024.

[5] Qiuhong Anna Wei, Sijie Ding, Jeong Joon Park, Rahul Sajnani, Adrien Poulenard, Srinath Sridhar, and Leonidas Guibas. LEGO-Net: Learning Regular Rearrangements of Objects in Rooms. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2023.

[6] Guangyao Zhai, Shun-Cheng Wu Evin Pınar Örnek, Yan Di, Federico Tombari, Nassir Navab, and Benjamin Busam. CommonScenes: Generating Commonsense 3D Indoor Scenes with Scene Graphs Diffusion. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2023.

[7] Weixi Feng, Wanrong Zhu, Tsu jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. LayoutGPT: Compositional Visual Planning and Generation with Large Language Models. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2023.

[8] Weixi Feng, Wanrong Zhu, Tsu jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. InstructScene: Instruction-Driven 3D Indoor Scene Synthesis with Semantic Graph Prior. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2023.

[9] Jingyu Liu, Wenhan Xiong, Ian Jones, Yixin Nie, Anchit Gupta, and Barlas Oğuz. CLIP-Layout: Style-Consistent Indoor Scene Synthesis with Semantic Furniture Embedding. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2023.

[10] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. SceneFormer: Indoor Scene Generation with Transformers. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2020.

[11] Rao Fu, Zehao Wen, Zichen Liu, and Srinath Sridhar. AnyHome: Open-Vocabulary Generation of Structured and Textured 3D Homes. *Proc. Euro. Conf. on Computer Vision*, 2024.

[12] Chuan Fang, Yuan Dong, Kunming Luo, Xiaotao Hu, Rakesh Shrestha, and Ping Tan. Ctrl-Room: Controllable Text-to-3D Room Meshes Generation with Layout Constraints. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2023.

[13] Bingxin Ke, Anton Obukhov, Shengyu Huang, Nando Metzger, Rodrigo Caye Daudt, and Konrad Schindler. Repurposing Diffusion-Based Image Generators for Monocular Depth Estimation. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2023.

[14] Minghua Liu, Ruoxi Shi, Kaiming Kuang, Yinhao Zhu, Xuanlin Li, Shizhong Han, Hong Cai, Fatih Porikli, and Hao Su. OpenShape: Scaling Up 3D Shape Representation Towards Open-World Understanding. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2023.

[15] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: Large Reconstruction Model for Single Image to 3D. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2023.

[16] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any Single Image to 3D Mesh in 45 Seconds without Per-Shape Optimization. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2023.

[17] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, and Wenping Wang. Wonder3D: Single Image to 3D using Cross-Domain Diffusion. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2023.

[18] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2022.

[19] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. ATISS: Autoregressive Transformers for Indoor Scene Synthesis. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2021.

[20] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2021.

[21] Heewoo Jun and Alex Nichol. Shap-E: Generating Conditional 3D Implicit Functions. *Proc. IEEE/CVF Conf. on Computer Vision & Pattern Recognition*, 2023.