图 学 学 报
JOURNAL OF GRAPHICS

大语言模型驱动的交互式建筑设计新范式——基于 Rhino7 的概念验证

蒋灿[1,2]，郑哲[2]，梁雄[1]，林佳瑞[2]，马智亮[2]，陆新征[2]

(1. 广联达科技股份有限公司，北京 100193；2. 清华大学土木工程系，北京 100084)

摘　　要：随着社会对建筑设计质量要求越来越高，建筑设计软件也变得越来越专业和复杂。现在的设计软件不仅学习成本高，而且交互模式复杂。大语言模型(LLM)的最新突破使计算机清晰地理解人类自然语言指令，并准确生成代码语言具有可行性，有望为人与软件的交互范式提供新思路。因此，本文提出了 LLM 驱动的交互式建筑设计新范式——将建筑师通过多次键鼠操作与设计软件交互转变为 LLM 根据建筑师自然语言指令生成并执行 API 调用脚本的方式；提出了技术路线并验证了其在建筑设计场景落地的可能性。该技术路线包括：①LLM 根据用户指令从 API 库中搜索与任务相关的 API；②LLM 基于指令和候选 API 摘要信息编写程序脚本并运行；③LLM 根据来自软件环境、用户等反馈改进优化所编写的程序脚本。通过 Rhino7 设计软件、GPT-4 和 CodeLlaMa 完成多个设计任务，测试当前 LLM 是否具备执行该技术路线各关键环节的能力。测试结果不仅证明了 LLM 驱动的交互式设计范式在建筑设计场景已初具落地前景，也为技术落地提供经验和建议。该设计范式的落地可以降低软件的使用门槛和学习成本，提高设计师工作效率；有望在未来的建筑设计软件中发挥重要作用。

关　键　词：建筑设计软件；软件交互；大语言模型；应用程序接口；GPT-4；Rhino7；Ladybug

中图分类号：TP 391　　**DOI：**10.11996/JG.j.2095-302X.2024030000

文献标识码：A　　**文 章 编 号：**2095-302X(2024)03-0000-00

A new interaction paradigm for architectural design driven by large language model: proof of concept with Rhino7

JIANG Can[1,2], ZHENG Zhe[2], LIANG Xiong[1], LIN Jiarui[2], MA Zhiliang[2], LU Xinzheng[2]

(1. Glodon Company Limited, Beijing 100193, China;

2. Department of Civil Engineering, Tsinghua University, Beijing 100084, China)

Abstract: As society places higher demands on the quality of architectural designs, design software has become more professional and complicated. Current design software not only incurs high learning costs but also features complex interaction modes. The recent breakthroughs in large language models (LLM) have enabled computers to clearly comprehend instructions based on human natural language and accurately generate code, which is expected to provide new ideas for the paradigm of human interaction with software. Therefore, this study designed a new paradigm of interactive architectural design driven by LLM, i.e., shifting from the architects interacting with the design software through multiple keyboard and mouse operations to LLMs writing scripts to invoke APIs according to

收稿日期：2023-09-25；定稿日期：2023-12-21

Received：25 September, 2023；**Finalized**：21 December, 2023

基金项目：国家自然科学基金项目(52378306)；北京市科委-中关村管委会项目(20220468132)

Foundation items: National Natural Science Foundation of China (52378306); Research Project of Beijing Municipal Science & Technology Commission, Administrative Commission of Zhongguancun Science Park (20220468132)

第一作者：蒋灿(1993–)，男，博士后，博士。主要研究方向为人工智能在智能建造领域的应用。E-mail：jiangc-l@glodon.com

First author：JIANG Can (1993-), postdoctoral, Ph.D. His main research interests cover application of artificial intelligence in intelligent construction. E-mail：jiangc-l@glodon.com

通信作者：林佳瑞(1987–)，男，副研究员，博士。主要研究方向为智能建造、数字孪生和知识图谱等。E-mail：lin611@tsinghua.edu.cn

Corresponding author：LIN Jia-rui (1987-), associate professor, Ph.D. His research interests are intelligent construction, digital twin and knowledge graph, etc. E-mail：lin611@tsinghua.edu.cn

architects' instructions. The methodology was proposed and its implementation feasibility in architecture design was validated. The methodology included: ①LLM retrieved task-related APIs from the API set according to user instructions; ②LLM wrote a program script based on instructions and the abstract of candidate APIs and ran it; ③ LLM revised the script written based on the feedback from the environment, users, etc. To validate the capabilities of current LLMs in executing the key steps of the methodology, multiple design tasks were completed with Rhino7 design software, GPT-4, and CodeLlaMa. The results not only demonstrated that the LLM-driven interactive design paradigm held initial prospects for implementation in architecture design, but also provided experiences and suggestions for its implementation. The implementation of this design paradigm could reduce the threshold and learning costs, improving the efficiency in many scenarios, and was expected to play a key role in future architectural design software.

Keywords: architectural design software; interaction with software; large language model; application programming interface; GPT-4; Rhino7; Ladybug

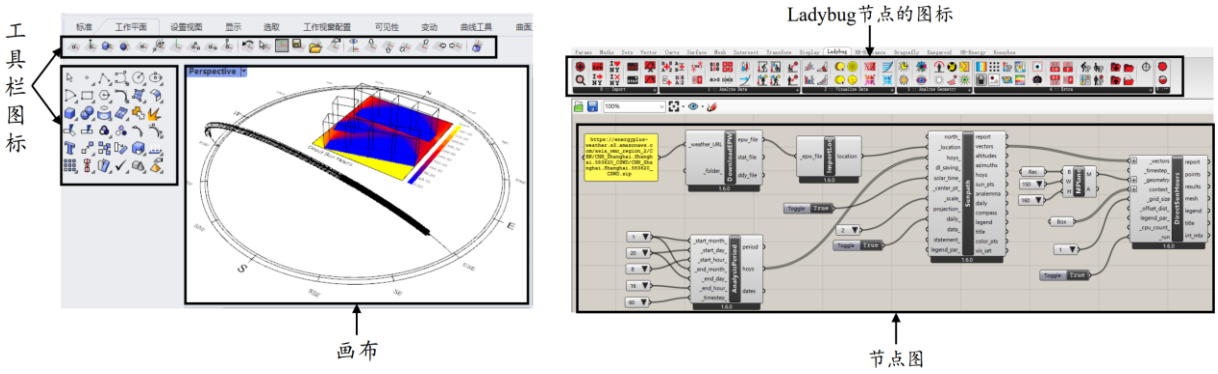
随着我国经济进入高质量发展阶段，社会各界对居住品质的要求也越来越高。为了更高效和严谨地设计和分析建筑方案，建筑师通常需要运用专业的建筑设计软件。随着建筑专业知识越来越丰富和复杂，建筑设计软件的使用门槛和学习成本在逐渐增高，建筑师与软件的交互也越来越复杂。

建筑设计软件提供大量的工具来支持建筑师完成设计任务，每个工具可以解决任务流程中的子任务；建筑师需要依据专业知识和对软件的理解分解设计任务，并选用合理的工具来完成子任务。以软件工程角度，各工具对应前端的图标和后端的应用程序接口(application program interface, API)，用户反复执行与图标交互并触发 API 运行的动作。以 Rhino7 为例(图 1(a))，进行建模时，需依次运用多个几何工具；进行建筑性能分析时，需将插件 Grasshopper/Ladybug[1]的多个节点组装为节点图(图 1(b))——节点图定义了各节点对应 API 的上下游关系。

因此，专业建筑设计软件的使用门槛和学习成本很高——用户需充分了解专业知识和相关设计软件，这样才能知道解决相关任务需要运用哪些工具并在软件中找到这些工具。另一方面，软件的交互模式复杂，完成设计任务不仅需要运用多个工具，且与每个工具交互时依赖若干次键鼠操作。这种传统交互范式在很多场景下非常低效，如在建筑性能分析任务中，搭建 Grasshopper/Ladybug 节点图通常需要花费几分钟。

在许多设计任务中，用文字表达任务意图已经比较容易；而在现今语音处理技术非常成熟的条件下，语音信息可以被非常精确地转换为文字信息，这使得人类可以更高效率地向软件表达意图。但是，传统软件难以理解人类自然语言；按照建筑师设计意图去运用相应工具完成设计任务则更不可能。

大型语言模型(large language models, LLM)的最新技术突破可以解决上述问题，进而带来全新的人与软件的交互范式。以 GPT-4 为首的 LLM 不仅具有出色的自然语言处理的能力，能理解软件用户用文字和语音表达的大部分需求；还具有良好的代码语言编写能力，可编写能调用自定义 API 的程序脚本。



(a) (b)

图 **1** 用户交互界面((a) Rhino7 主交互界面；(b) Grasshopper/Ladybug 插件交互界面)

Fig. 1 User interface ((a) Main interface of Rhino7; (b) Interface of Grasshopper/Ladybug plug-in)

因此，本文针对建筑设计场景提出 LLM 驱动的设计交互新范式。让 LLM 直接根据用户的自然语言指令编写程序脚本调用 API 完成设计任务，作为传统通过键鼠操作逐步驱动各 API 运行的替代方案。该新范式可以降低设计软件的使用门槛和学习成本，且在众多设计任务中比传统方式更高效。

本文研究进一步地提出了该范式的技术路线，其关键环节包括：①基于指令的 API 搜索；②基于指令和候选 API 的脚本编写；③基于反馈的脚本优化。通过运用 Rhino7，GPT-4 和 CodeLlaMa 完成多个设计任务表明，当前 LLM 已具备支撑该技术路线在建筑设计场景中落地的能力。

1 背景知识

1.1 自然语言处理在建筑设计的应用

面向自然语言处理(natural language processing, NLP)的语言模型经历了 5 个时代[2]：符号语言模型、统计语言模型(statistical language models, SLM)、神经语言模型(neural language models, NLM)、预训练语言模型(pre-trained language models, PLM)和 LLM。早在符号语言模型时段，就有学者利用本体论方法进行建筑知识管理[3]；在 SLM 时代以后，出现了更多 NLP 技术在建筑设计信息查询中的应用研究，如文献[4-5]都基于 Stanford Parser——包含多个 SLM[6]和 NLM[7]的 NLP 工具，分别实现了文字和语音驱动的 BIM 信息查询。在 PLM 时代，文献[8]提出了土建领域语言模型，文献[9-10] 基于 PLM 实现了自然语言驱动的 BIM 模型自动审查。但受限于模型能力，此时还少见用自然语言驱动建筑设计任务的相关研究。

进入 LLM 时代后，学者和开发者们一方面继续专注于信息查询任务，如 ZHENG 和 FISCHER[11]提出的 BIM-GPT；另一方面也在探索其他应用可能性，如基于 Stable Diffusion 模型[12]的自然语言驱动的概念设计图生成，和基于 GPT4[13]的 SketchUp 建模脚本编写[14]。但上述探索大多涉及意图模糊的开放性任务，难度偏低且与建筑专业知识关系较小；另一方面，上述探索也未对 LLM 在未来建筑设计软件中的作用提出系统性的观点和见解。

1.2 大语言模型

LLM 通过预训练从大量文本数据中学习人类自然语言逻辑和背景知识，获得理解和生成人类语言的能力。经过预训练的 LLM 可以完成基本的文本生成、机器翻译、问答系统、代码编写等任务，但其处理专业领域的复杂任务的能力不一。参数数量越高的 LLM，越擅长处理这类任务，但其训练成本也越高。如 GPT-4 的参数数量高达 1.8 万亿，其完成一次预训练预计花费 2 150 万美元。

在解决专业领域的复杂问题时，需要采用提示词工程(Prompt engineering)[15]来引导 LLM。提示词工程是指对 LLM 的输入——提示词(Prompt) 进行策略性设计的过程，通常需要在提示词中写明 LLM 的任务角色、任务条件以及任务相关的专业知识。

对于参数数量较少的 LLM，其处理专业领域的复杂问题能力较差，即使运用了提示词工程也很难完成任务。此时可进行微调，即让 LLM 继续学习特定任务相关文本数据，轻微调整其模型参数，以提高其解决特定任务的能力。但准备微调数据需要投入成本和精力；如果仅采用少量数据开展微调，可能会出现过拟合，降低 LLM 解决其他问题的能力。因此建议在条件允许时，宜采用能力更高的 LLM，结合提示词工程完成专业领域的复杂任务。

1.3 LLM 模型应用于代码编写

最近，LLM 的编程及 API 调用能力得到高度关注，并被众多研究证实。如 PATIL 等[16]通过微调得到 Gorilla 模型，根据用户意图生成 python 脚本，调用合适的深度学习模型，完成 AI 绘画、文本处理和科学计算等工作。LI 等[17]建立了具备银行账户管理、信息检索、财务管理、医疗管理等功能的 API 数据库；然后，GPT-3.5 一边从多轮对话中理解用户的指令，一边调用 API 执行，还能根据执行结果回复用户。WU 等[18]提出了 AutoGen 框架，并根据用户指令完成代码生成、数学、决策、问题查询等方面的任务。WANG 等[19]运用 GPT-3.5 和 GPT-4 编写调用底层 API 的程序脚本，驱动智能体自行游玩 MineCraft 游戏。

从上述研究中可以总结出可有效提高 LLM 编程成功率的 2 个经验：①API 搜索(API retrieve)，仅将与指令最相关的若干 API 的摘要信息写入提示词内；②环境反馈(environment feedback)，当 LLM 难以一次性地编写正确的程序脚本时，可以将该程序运行的报错信息或另一个 LLM 对此脚本不合理之处的分析，在下一轮对话中反馈给 LLM，

最终帮助其输出正确的脚本。

2 技术路线

为了确保 LLM 通过编程完成设计任务的成功率，应执行如图 2 所示的 3 个关键环节。首先需根据用户指令在 API 库中搜索与任务相关的 API，列为候选 API(2.1 节)；然后运用提示词工程，让 LLM 基于指令和候选 API 摘要信息编写程序脚本，并运行(2.2 节)；最后 LLM 根据环境反馈改进和优化程序脚本(2.3 节)，环境反馈可以来自软件运行环境、用户以及其他 LLM。

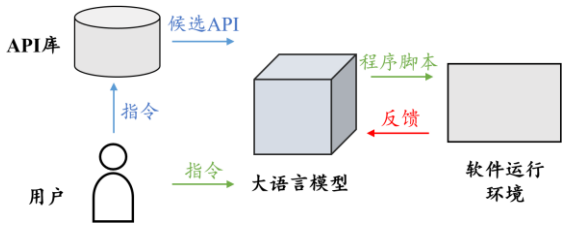


图 2 LLM 驱动的交互式设计范式技术路线

Fig. 2 Methodology of LLM-driven interactive design paradigm

2.1 API 搜索

建筑设计并非通用场景，LLM 的预训练数据中很可能不含该领域 API；因此需将这些 API 的摘要信息写入提示词中告知 LLM。设计软件的工具 API 众多，如果写入所有 API 的信息，会导致提示词过长；这不仅会大幅提高 LLM 忽略正确信息的可能性，还会降低其响应速度。

因此需执行 API 搜索，从众多 API 中筛选出与任务相关的 API。常见的 API 搜索方法是文本相似度匹配，即选取与任务指令相似度最高的若干个 API。具体的，首先利用词嵌入模型(如 M3E)将任务指令转换为词嵌入向量(word embedding vector)；然后将所有 API 的摘要信息也转换为词嵌入向量，API 摘要见表 1，通常只需转换 API 名称、功能和输入部分；接着计算语义相似度，其是向量空间中 2 个词嵌入向量夹角的余弦，相似度越高，2 个词嵌入向量所对应的文本在语义上越接近；最后将计算出的相似度排序并选取相似度最高的若干个 API。

表 1 API 摘要内容构成

Table 1 Formation of API abstract

内容	例子
API 名称	calc_sunpath(location, hoys, …)
API 功能	Calulate trajectory of sun according to location and time information
API 输入	Location information (latitude, longitude, etc.) of a city (location: Object); …
API 输出	A list of solar altitude (altitudes: list); A list of solar azimuth (azimuths: list); …
API 调用案例	altitudes, azimuths, datetimes, vectors = calc_sunpath(location, hoys)

2.2 LLM 编写程序脚本

在搜索得到候选 API 后，执行提示词工程将任务指令和候选 API 摘要整理为提示词。首先准备提示词模板，当 LLM 执行编写程序脚本调用自定义 API 的任务时，可采用图 3 所示的模板。该模板包含角色定义、背景信息和任务指令：角色定义部分明确指出 LLM 的角色是 Python 编程助手；背景信息部分介绍候选 API 摘要；任务指令部分说明希望 LLM 实现的目标。然后，将来自用户的任务指令和候选 API 摘要填入模板相关位置，即任务指令和背景信息部分。

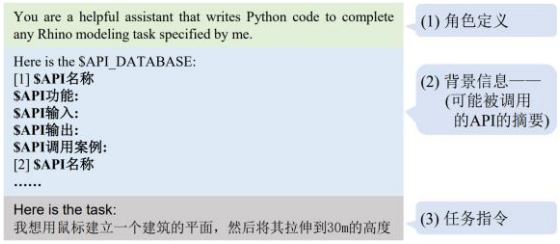


图 3 提示词模板

Fig. 3 Template of prompt

生成提示词后输入 LLM 中，得到 LLM 的回复，从回复中提取其编写的程序脚本并发送至设计软件运行环境中运行。只需将 LLM 的相关接口嵌入未来建筑设计软件中，即可自动化地执行上述所有步骤。

2.3 环境反馈

对于某些极其复杂的任务，LLM 难以一次性地编写正确的程序脚本；此时将环境反馈信息告知 LLM，可让其修正原代码并最终输出正确的脚本。错误代码在设计软件的运行中可能会报错，也可能会输出与用户意图不符的结果，此时需将报错信息或用户的意见反馈给 LLM；甚至可在程序运行前让另一 LLM 检查该程序脚本并给出评价。由于 LLM 支持多轮对话功能，通常只需将环境反馈写在对话历史末尾重新输入 LLM 即可；也可以设计新的提示词模板，执行提示词工程将旧代码、环境反馈等信息填入新的提示词中。

将 LLM 的相关接口嵌入未来建筑设计软件中，即可自动的运行 LLM 生成的源代码并获得运行环境反馈，再根据反馈合成新的提示词并重新运行。但少数情况下，LLM 生成的代码并没有语法错误，仅仅是运行结果不符合用户意图，而且很难

设计自动化机制检测出该问题。所以，未来的设计软件应考虑此情况，允许用户及时通过文本或语音反馈任务的完成情况。

3 概念验证

为了验证 LLM 驱动的交互式设计范式在建筑设计场景落地的可能性，本研究基于 Rhino7 建筑设计软件见表 2 共 12 类任务，以验证技术路线中各关键环节是否可以被有效执行。

执行这些任务依赖 RhinoScriptSyntax 模块或 Grasshopper/Ladybug 插件的相关 API。本研究首先基于相关文档与源代码建立 API 库，应用 M3E 模型计算语义相似度，证明了 API 搜索的可行性(3.1 节)；然后运用 GPT-4 和经过微调的 CodeLlaMa-7b-Instruct[20]，基于用户指令和候选 API 编写程序脚本并在 Rhino7 的 Python Script 工具中成功运行，证明了在 API 搜索条件下 LLM 具有完成上述任务的能力(3.2 节)；最后，以几何建模任务为例，证明了环境反馈机制可以优化 LLM 编写的脚本(3.3 节)。

表 2 用于概念验证的设计任务

Table 2 Design tasks for proof of the concept

需求	任务	API 数量
几何建模	生成矩形截面建筑模型	2
	生成不规则截面建筑模型	5
	生成多连立方体建筑模型	4
	多建筑模型随机排布	4
	多建筑模型按指定位姿排布	4
	考虑间距约束的多建筑模型排布	4
建筑性能分析	建筑日照分析	6
	建筑辐照度分析	6
可视化渲染	太阳路径计算与可视化	4
	天穹辐射密度计算与可视化	4
	视角与渲染模式转换	2
	模型颜色变化	1

3.1 验证：API 搜索

首先采用 Beautiful Soup 爬虫库爬取 Rhino 的官方文档；具体的，先自动下载文档所在网页的 HTML 源码文件，该文件结构化地记录了网页上的所有内容，通过正则匹配提取出 RhinoScriptSyntax 模块所有 API 的摘要信息。另一方面，将 Grasshopper/Ladybug 插件的节点封装为 API 并用相同的方法从文档中获得 API 摘要信息，由于部分节点源代码过于复杂且摘要信息过长，对其做了适当简化。

然后为表 2 中的各类任务分别设计 2 条不同的自然语言指令，利用 M3E 模型按照第 2.1 节所介绍的技术方案搜索与任务指令文本相似度最高的 10 个 API；搜索结果显示，召回率——任务相关 API 在搜索到的 API 中的比率达到了 81%。考虑到部分现有 API 的摘要信息存在缺失，信息的丰富程度和精确程度还有待提高；API 搜索的效果仍有提升空间。另一方面，测试发现 API 摘要的互相似性会对搜索造成不利影响，如平面拉伸 API (ExtrudeSurface) 与曲线拉伸 API (ExtrudeCurveStraight)摘要信息相似度高，有导致混淆的可能。在开发未来建筑设计软件时应尽量避免上述问题。

3.2 验证：LLM 编写程序脚本(图 4)

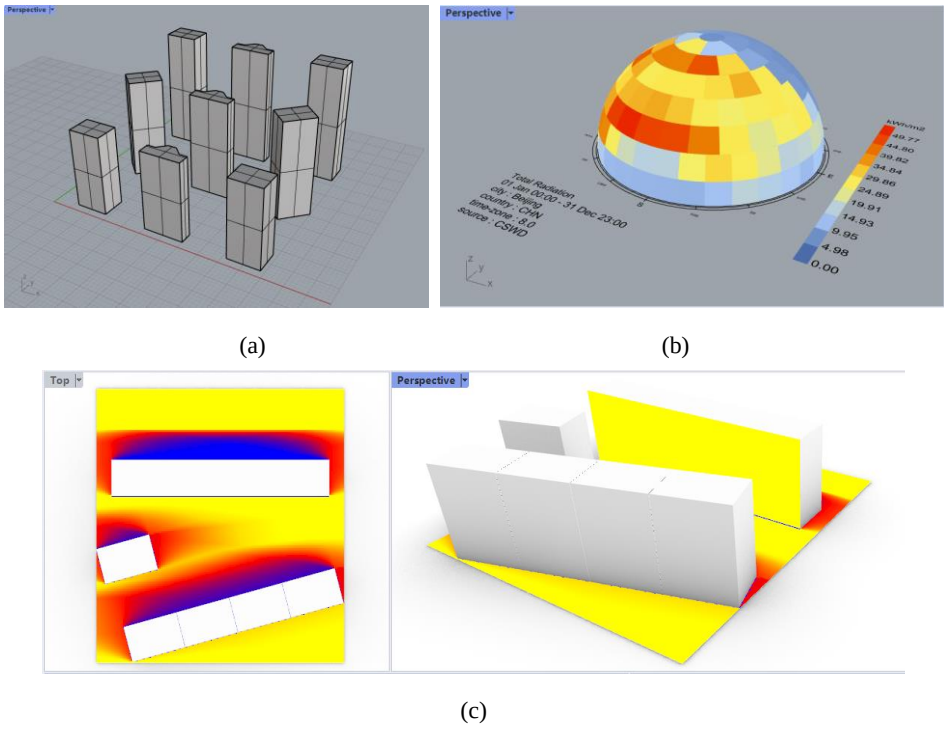
本研究通过表 2 中的任务，进一步验证 LLM 能否在 API 搜索的前提下成功生成程序脚本并执行设计任务。上述所有任务的难度都不低于文献[14]中的相关案例，部分任务还颇具挑战性，如在日照分析等任务中，用户指令未详细告知各子任务的上下游关系，需从相关 API 的摘要信息中了解；高效地完成考虑间距约束的多建筑模型排布任务需进行算法设计，不能简单地串联相关 API。

按照第 2.2 节所介绍的方法让 GPT-4 编写程序脚本完成任务。首先生成提示词，将任务指令和与任务相关的所有 API 摘要信息加入提示词中，同时适当添加不相关 API 作为干扰项以提高任务难度；然后让 GPT-4 编写程序脚本，并在 Rhino7 的 Python Script 工具中运行并可视化。测试结果表明，针对绝大多数任务，GPT-4 可一次性地输出正确的程序脚本；但在解决考虑间距约束的多建筑模型排布任务时，GPT-4 设计的算法较低效，生成的脚本未能在有限时间内完成运行。

然后，本文研究通过对比实验证明了 API 搜索的有效性。如执行生成不规则截面建筑模型任务时，如果不进行 API 搜索直接将 RhinoScriptSyntax 模块的全部 API 输入 GPT-4，会导致 GPT-4 混淆并选择错误的 API，生成楼体外表面模型而非楼体实体模型；如果搜索了与实体模型生成相关的 API，即使在提示词中加入干扰 API(与外表面模型生成相关)，GPT-4 也能一次性地生成正确的代码。

最后，本研究测试了 GPT-4 以外的 LLM 编写程序脚本完成建筑设计任务的能力。利用 CodeLlaMa-7b-Instruct 进行实验发现，虽然原始的 CodeLlaMa-7b-Instruct 不能直接完成任务，但以 GPT-4 生成的日照分析和辐照度分析程序脚本对

其进行微调后，CodeLlaMa 也能成功执行相似任务。



(a) (b)

(c)

图 4 LLM 编写的脚本运行效果((a)多建筑模型按指定位姿排布；(b)天穹辐射密度计算与可视化；(c)日照分析)

Fig. 4 Performance of execution of script written by LLM ((a) Site planning with multiple building models; (b) Calculation and visualization of radiance distribution of sky dome; (c) Sunlight analysis)

3.3 验证：环境反馈

在生成不规则截面建筑模型任务中，若不执行 API 搜索会导致 GPT-4 选择错误的 API，但通过环境反馈机制可以帮助 GPT-4 修正原有错误代码并输出正确的代码。由于 GPT-4 产生混淆，其第一次输出的代码只能生成楼体外表面模型而非实体模型；若在下一轮会话中告知此信息，GPT-4 便生成新的代码。不过第二次输出的代码出现了输入格式错误，再将报错信息在会话中告知 GPT-4，便能最终得到正确的代码。

在考虑间距约束的多建筑模型排布任务中，GPT-4 第一次生成的脚本采用了低效的算法——枚举建筑位置并逐栋检测是否满足间距约束条件，该脚本未能在有限时间内运行结束并得到正确的结果。在下一轮会话中向 GPT-4 提示更高效的算法思路——先计算各栋建筑位置的可行域，再在可行域内随机放置建筑模型，GPT-4 根据该思路生成新的代码并迅速地完成了该任务。

最后，本研究测试了 LLM 是否具备审查代码的能力。在 GPT-4 生成的任务脚本中人为添加错误，如变更变量名、删减步骤等，然后让 GPT-4 审查上述错误脚本并提供反馈。结果表明，GPT-4 可以准确地发现这些基本错误。但当脚本没有语法错误只是输出结果与任务意图不符，且需要具备相关专业知识才能识别此差异时，GPT-4 不能检查出此类问题。

4 结论与展望

由于 LLM 具有出色的自然语言处理能力和代码编程能力，LLM 有望对建筑师与设计软件的交互提供新范式——将设计师点击软件前端界面的图标触发后端 API 运行，变为 LLM 根据设计师的自然语言(文字或语音)指令基于 API 编写脚本并执行的方式。

本文进一步提出了让此新型交互范式落地的技术路线，其关键环节包括：①搜索与任务相关的 API；②基于任务指令和候选 API 的提示词工程及 LLM 编写程序脚本；③LLM 基于环境反馈修改与优化程序脚本。

通过在 Rhino7 中完成几何建模、建筑性能分析、可视化渲染等设计任务，本文证明了上述关键环节的可行性。首先在 API 库中搜索上述任务的

相关 API，达到 81%的召回率；然后将包含任务指令和候选 API 摘要信息的提示词输入 GPT-4 和微调后的 CodeLlaMa 中，生成程序脚本并在 Rhino7 的 Python Script 工具中运行，成功完成上述任务；最后通过几何建模任务证明了环境反馈有助于 LLM 修正错误代码。

这些试验不仅证明 LLM 驱动的交互式设计范式已经在建筑设计领域初具落地前景，而且为构建未来的建筑设计软件提供宝贵经验。基于试验经验提出以下建议：

1) 需保证 API 文档内容完整、精确，并尽可能降低各 API 摘要文本的互相似度，以提高 API 搜索的有效性；

2) 需建立和完善 API 搜索—提示词工程—脚本生成—脚本运行—报错反馈的自动化流程，也要同时提供用户反馈通道。

该新范式有助于降低建筑设计软件使用门槛和学习成本，减轻建筑师的操作负担并提高设计效率；对构建面向未来的智能设计工作流具有借鉴意义。但仍需注意到，对于任务指令过于冗长的情况，该新范式的交互效率优势不明显；因此未来的建筑设计软件宜保留传统交互方式。

鉴于 LLM 在本文研究中的卓越表现，建议未来进一步探索 LLM 应用于设计工作流的潜力，如探索 LLM 理解更宏观层级的设计意图，并规划设计方案的能力；同时，仍需要关注如何进一步的降低建筑师表达意图的成本，研究以语音和设计草图表达设计意图的解决方案。

参考文献（References)

[1] ROUDSARI M S, PAK M. Ladybug: a parametric environmental plugin for Grasshopper to help designers create an environmentally-conscious design[C]//The 13th International Conference of International Building Performance Simulation Association. New York: Curran Associates, Inc, 2014: 3128-3135.

[1] SADEGHIPOUR ROUDSARI M, PAK M, VIOLA A. Ladybug: a parametric environmental plugin for grasshopper to help designers create an environmentally-conscious design[C]//Building Simulation Conference Proceedings", "Proceedings of Building Simulation 2013: 13th Conference of IBPSA. IBPSA, 2013: 3128-3135.

[2] ZHAO W X, ZHOU K, LI J Y, et al. A survey of large language models[EB/OL]. (2023-03-31) [2023-05-24].. http://arxiv.org/abs/2303.18223.pdf.

[3] ANUMBA C J, ISSA R R A, PAN J Y, et al. Ontology-based information and knowledge management in construction[J]. Construction Innovation, 2008, 8(3): 218-239.

[4] LIN J R, HU Z Z, ZHANG J P, et al. A natural-language-based approach to intelligent data retrieval and representation for cloud BIM[J]. Computer-Aided Civil and Infrastructure Engineering, 2016, 31(1): 18-33.

[5] SHIN S, ISSA R R A. BIMASR: framework for voice-based BIM information retrieval[J]. Journal of Construction Engineering and Management, 2021, 147(10): 04021124.

[6] SOCHER R, BAUER J, MANNING C D, et al. Parsing with compositional vector grammars[J]. ACL 2013 - 51st Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 2013, 1: 455-465.

[7] CHEN D Q, MANNING C. A fast and accurate dependency parser using neural networks[C]//The 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Stroudsburg, PA, USA: Association for Computational Linguistics, 2014: 740-750.

[8] ZHENG Z, LU X Z, CHEN K Y, et al. Pretrained domain-specific language model for natural language processing tasks in the AEC domain[J]. Computers in Industry, 2022, 142: 103733.

[9] ZHOU Y C, ZHENG Z, LIN J R, et al. Integrating NLP and context-free grammar for complex rule interpretation towards automated compliance checking[J]. Computers in Industry, 2022, 142: 103746.

[10] ZHENG Z, ZHOU Y C, LU X Z, et al. Knowledge-informed semantic alignment and rule interpretation for automated compliance checking[J]. Automation in Construction, 2022, 142: 104524.

[11] ZHENG J W, FISCHER M. BIM-GPT: a prompt-based virtual assistant framework for BIM information retrieval[EB/OL]. (2023-04-18) [2023-05-11]. http://arxiv.org/abs/2304.09333.pdf.

[12] ROMBACH R, BLATTMANN A, LORENZ D, et al. High-resolution image synthesis with latent diffusion models[C]//2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition. New York: IEEE Press, 2022: 10674-10685.

[13] OPENAI,: , ACHIAM J, et al. GPT-4 technical report[EB/OL]. (2023-05-24) [2023-06-05]. http://arxiv.org/abs/2303.08774.pdf.

[14] TutorialsUp. SketchUp + ChatGPT 4 different use cases [EB/OL]. (2023-05-04) [2023-06-11]. https://www.youtube.com/watch?v=IPoFA-XyWrc.

[15] WHITE J, FU Q C, HAYS S, et al. A prompt pattern catalog to enhance prompt engineering with ChatGPT[EB/OL]. (2023-02-21) [2023-05-14]. http://arxiv.org/abs/2302.11382.pdf.

[16] PATIL S G, ZHANG T J, WANG X, et al. Gorilla: large language model connected with massive APIs[EB/OL]. (2023-05-24) [2023-06-05]. http://arxiv.org/abs/2305.15334.pdf.

[17] LI M H, ZHAO Y X, YU B W, et al. API-bank: a comprehensive

benchmark for tool-augmented LLMs[EB/OL]. (2023-04-14) [2023-06-06]. http://arxiv.org/abs/2304.08244.pdf.

[18] WU Q Y, BANSAL G, ZHANG J Y, et al. AutoGen: enabling next-gen LLM applications via multi-agent conversation[EB/OL]. (2023-08-16) [2023-09-07]. http://arxiv.org/abs/2308.08155.pdf.

[19] WANG G Z, XIE Y Q, JIANG Y F, et al. Voyager: an open-ended embodied agent with large language models[EB/OL]. (2023-05-25) [2023-07-28]. http://arxiv.org/abs/2305.16291.pdf.

[20] ROZIÈRE B, GEHRING J, GLOECKLE F, et al. Code llama: open foundation models for code[EB/OL]. (2023-08-24) [2023-09-04]. http://arxiv.org/abs/2308.12950.pdf.