Holodeck 项目架构与运行指南

项目概述

Holodeck 是一个由 Allen Institute for AI 开发的框架，用于**通过自然语言指令自动生成 3D 具体化 AI 环境**。它基于 AI2-THOR 平台，支持 macOS 10.9+ 或 Ubuntu 14.04+。

核心功能：将文本描述（如"一个客厅"）转换为完整的 3D 室内场景，包括房间布局、家具、窗户、灯光等。

关键论文：Language Guided Generation of 3D Embodied AI Environments (CVPR 2024)

系统架构

整体流程

```
用户查询 (Natural Language)
        ↓
    LLM 推理 (GPT-4o)
        ↓
    生成场景配置
        ↓
    AI2-THOR 渲染
        ↓
    生成 3D 环境 (JSON + 图像/视频)
```

核心模块结构

```
ai2holodeck/
├── main.py                          # 入口程序
├── constants.py                     # 全局配置和路径
└── generation/
    ├── holodeck.py                  # 主生成引擎
    ├── rooms.py                     # 房间生成和地板平面规划
    ├── walls.py                     # 墙体生成
    ├── doors.py                     # 门窗框架生成
    ├── windows.py                   # 窗户生成
    ├── floor_objects.py             # 地面对象布置
    ├── wall_objects.py              # 墙面对象布置
    ├── ceiling_objects.py           # 天花板对象
    ├── small_objects.py             # 小型对象（装饰品）
    ├── lights.py                    # 灯光生成
    ├── skybox.py                    # 天空盒配置
    ├── object_selector.py           # LLM 对象选择引擎
    ├── objaverse_retriever.py       # 3D 资源检索（基于 CLIP）
    ├── layers.py                    # 资源分层管理
    ├── milp_utils.py                # 混合整数线性规划求解器
    ├── utils.py                     # 工具函数
    ├── prompts.py                   # LLM 提示词模板
    └── empty_house.json             # 空场景模板
```

关键组件详解

1. 主程序入口 (main.py)

功能：

- 解析命令行参数
- 初始化 Holodeck 引擎
- 支持三种运行模式：
  - `generate_single_scene`：生成单个场景
  - `generate_multi_scenes`：批量生成（从文件读取查询）
  - `generate_variants`：基于现有场景生成变体

关键参数：

参数	说明	默认值
`--query`	场景描述文本	"a living room"
`--openai_api_key`	OpenAI API 密钥	从环境变量读取

参数	说明	默认值
`--openai_api_base`	自定义 API 端点（支持本地模型）	无
`--llm_model_name`	使用的模型名称	gpt-4o-2024-05-13
`--save_dir`	输出目录	./data/scenes
`--use_milp`	是否使用 MILP 求解器	False
`--use_constraint`	是否启用约束满足	True
`--generate_image`	是否生成场景图像	True
`--generate_video`	是否生成场景视频	False
`--add_ceiling`	是否添加天花板	False
`--random_selection`	对象选择是否更随机	False

2. 核心生成引擎 (holodeck.py)

Holodeck 类初始化以下组件：

```
# LLM
self.llm = ChatOpenAI(model_name, max_tokens=4096)

# 多模态检索模型
self.clip_model             # CLIP ViT-L-14（文本-图像匹配）
self.sbert_model            # Sentence BERT（语义相似度）

# 生成器模块
self.floor_generator        # 地板平面生成
self.wall_generator         # 墙体生成
self.door_generator         # 门生成
self.window_generator       # 窗户生成
self.object_selector        # 对象选择
self.floor_object_generator     # 地面对象放置
self.wall_object_generator      # 墙面对象放置
self.ceiling_generator          # 天花板对象
self.small_object_generator     # 小对象生成
```

主生成流程 (generate_scene)：

1. **生成房间布局**：LLM 根据查询描述生成地板平面
2. **材料选择**：使用 CLIP 为房间选择合适的地板/墙面材料
3. **墙体和门窗**：生成墙体、门框、窗户的具体配置

4. **对象选择**：LLM 根据房间类型推荐合适的家具
5. **约束求解**：使用 DFS 或 MILP 求解器放置物体满足空间约束
6. **渲染与输出**：通过 AI2-THOR 渲染为 3D 场景，导出 JSON 配置文件

3. LLM 交互系统 (prompts.py)

系统使用精心设计的提示词指导 LLM 完成各个阶段的任务：

- **floor_plan_prompt**：生成矩形房间的坐标和材料配置
- **wall_height_prompt**：确定墙高
- **doorway_prompt**：设计房间间连接（门/开放空间）
- **window_prompt**：为每个房间设计窗户
- **object_selection_prompt**：推荐房间家具
- **object_constraints_prompt**：为对象分配空间约束（相对位置、对齐等）

提示词框架特点：

- 精确的输出格式（管道分隔）
- 约束条件明确（房间大小、容纳率等）
- 支持额外的自定义需求 (additional_requirements)

4. 资源检索系统 (objaverse_retriever.py)

ObjathorRetriever 类负责：

- 使用 **CLIP** 进行文本-图像相似度匹配
- 从 OBJathor 数据集检索 3D 资产
- 基于**语义相似度**和**尺寸匹配**筛选最合适的模型
- 支持多进程检索加速

工作流程：

```
对象描述（文本）
    ↓
CLIP 文本编码器
    ↓
与资产图像特征比较
    ↓
选择相似度最高的资产
```

```
    ↓
验证尺寸和物理属性
```

5. 对象布置引擎 (object_selector.py)

ObjectSelector 类使用多阶段 LLM 调用：

1. **阶段 1**：LLM 推荐每个房间需要的主要对象类别
2. **阶段 2**：LLM 进一步细化描述和数量
3. **约束生成**：LLM 为每个对象生成空间约束（如"靠近墙边"、"面向沙发"）

布置策略：

- **DFS 求解器**（推荐）：深度优先搜索满足约束条件
- **MILP 求解器**（备选）：混合整数线性规划，精确但计算量大

空间容纳：

- 地面容纳率：区域面积 × 0.4
- 墙面容纳率：周长 × 0.5

6. AI2-THOR 集成 (utils.py)

- 初始化 Unity 控制器
- 向场景加载 3D 模型
- 生成俯视图和视频
- 管理物理模拟和碰撞检测

数据流和依赖

输入数据结构

场景 JSON 格式 (`scene`)：

```
{
    "query": "a modern living room",           # 用户查询
```

```
    "rooms": [                                      # 房间数组
        {
            "roomType": "living room",
            "vertices": [(x1, y1), ...],            # 房间顶点
            "floorMaterial": {"name": "...", "color": "..."},
            "wallMaterial": {...},
        },
        ...
    ],
    "walls": [...],                                 # 墙体配置
    "doors": [...],                                 # 门配置
    "windows": [...],                               # 窗户配置
    "selected_objects": {...},                      # 选中的对象
    "floor_plan": {...},                            # 布置信息
    ...
}
```

外部数据依赖

OBJathor 数据集 (必需):

```
~/.objathor-assets/
├── 2023_09_23/
│   ├── assets/                                     # 3D 模型库
│   ├── features/                                   # CLIP 特征（预计算）
│   ├── annotations.json.gz                         # 对象元数据
│   └── thor_object_data/                           # AI2-THOR 特定数据
└── holodeck/
    └── 2023_09_23/
        ├── scenes/                                 # 生成的场景
        └── thor_object_data/                       # 物体特征和注解
```

下载命令:

```
python -m objathor.dataset.download_holodeck_base_data --version 2023_09_23
python -m objathor.dataset.download_assets --version 2023_09_23
python -m objathor.dataset.download_annotations --version 2023_09_23
python -m objathor.dataset.download_features --version 2023_09_23
```

运行指南

环境配置

1. 创建 Python 环境

```
conda create --name holodeck python=3.10
conda activate holodeck
```

2. 安装依赖

```
cd /Volumes/DoggyChen/GitProjects/Holodeck
pip install -r requirements.txt
pip install --extra-index-url https://ai2thor-pypi.allenai.org
ai2thor==0+8524eadda94df0ab2dbb2ef5a577e4d37c712897
```

3. 下载数据资产

```
python -m objathor.dataset.download_holodeck_base_data --version 2023_09_23
--path /Volumes/DoggyChen/objathor-assets
python -m objathor.dataset.download_assets --version 2023_09_23 --path
/Volumes/DoggyChen/objathor-assets
python -m objathor.dataset.download_annotations --version 2023_09_23 --path
/Volumes/DoggyChen/objathor-assets
python -m objathor.dataset.download_features --version 2023_09_23 --path
/Volumes/DoggyChen/objathor-assets
```

4. 配置环境变量（可选）

```
export OBJAVERSE_ASSETS_DIR=/Volumes/DoggyChen/objathor-assets
export OPENAI_API_KEY=your-api-key
```

运行模式

单场景生成

```
python ai2holodeck/main.py \
    --query "a modern living room with floor-to-ceiling windows" \
    --openai_api_key your-key \
    --save_dir ./data/scenes
```

使用本地 LLM 模型

```
python ai2holodeck/main.py \
    --query "a cozy bedroom" \
    --openai_api_key sk-dummy \
    --openai_api_base http://localhost:8000/v1 \
    --llm_model_name ./qwen2.5-32b
```

批量生成

创建查询文件 `queries.txt`：

```
a living room
a bedroom
a kitchen
a bathroom
```

运行：

```
python ai2holodeck/main.py \
    --mode generate_multi_scenes \
    --query_file queries.txt \
    --openai_api_key your-key
```

场景变体生成

```
python ai2holodeck/main.py \
    --mode generate_variants \
    --original_scene ./data/scenes/my_scene/my_scene.json \
    --query "a luxury version" \
    --number_of_variants 5 \
    --openai_api_key your-key
```

在 Unity 中查看场景

1. **安装 Unity** (版本 2020.3.25f1)
2. **克隆 AI2-THOR 仓库**

```
git clone https://github.com/allenai/ai2thor.git
cd ai2thor
git checkout 07445be8e91ddeb5de2915c90935c4aef27a241d
```

3. **重装必要包**

```
pip uninstall Werkzeug Flask
pip install Werkzeug==2.0.1 Flask==2.0.1
```

4. **在 Unity 中打开项目**

- 在 Unity Hub 中打开 `ai2thor/unity` 文件夹
- 打开场景 `ai2thor/unity/Assets/Scenes/Procedural/Procedural.unity`

5. **连接场景**

```
python connect_to_unity.py --scene ./data/scenes/my_scene/my_scene.json
```

6. **按 Unity 中的播放按钮查看场景**

关键技术组件

使用的 AI 模型

模型	用途	来源
GPT-4o-2024-05-13	场景规划和对象选择	OpenAI API
CLIP ViT-L-14	文本-图像匹配，资产检索	OpenAI / LAION
Sentence-BERT (all-mpnet-base-v2)	语义相似度计算	Hugging Face

优化算法

求解器	用途	特点
DFS (推荐)	对象放置约束满足	快速，启发式，更好的布局
MILP	对象放置优化	精确，但计算开销大

外部依赖库

- **LangChain**：LLM 编排和提示管理
- **OpenAI**：API 访问
- **Open-CLIP**：CLIP 模型加载和推理
- **Sentence-Transformers**：语义编码
- **Shapely**：几何计算和碰撞检测
- **cvxpy**：MILP 求解基础
- **Gurobi**：商用 MILP 求解器（可选）
- **AI2-THOR**：3D 环境模拟和渲染
- **PIL/matplotlib**：图像处理和可视化

输出结果

生成的文件结构

```
./data/scenes/
└── <scene_name>/
    ├── <scene_name>.json              # 场景配置文件
    ├── <scene_name>_top_down.png      # 俯视图
    ├── <scene_name>_video.mp4         # 场景漫游视频（可选）
    ├── <scene_name>_raw_floor_plan    # LLM 生成的原始地板方案
    ├── <scene_name>_object_plan       # 对象选择方案
    └── metadata.json                   # 元数据（生成时间、参数等）
```

JSON 场景配置示例

```
{
    "query": "a modern living room",
    "rooms": [
```

```
        {
            "id": "room_0",
            "roomType": "living room",
            "floorPolygon": [[0, 0], [0, 5], [8, 5], [8, 0]],
            "vertices": [[0, 0], [0, 5], [8, 5], [8, 0]],
            "floorMaterial": {"name": "wood", "color": "#D2691E"},
            "walls": [...],
            "children": [...]  # 3D 对象
        }
    ],
    "walls": [...],
    "doors": [...],
    "windows": [...],
    "lights": [...]
}
```

高级配置

自定义提示词

在 `prompts.py` 中修改相应的提示词模板，以获得不同风格或功能的场景。

修改 LLM 参数

在 `holodeck.py` 中的 `ChatOpenAI` 初始化处调整：

- `temperature`：输出多样性
- `max_tokens`：最大生成长度
- `top_p`：核采样参数

调整物体布置策略

在 `object_selector.py` 中修改：

- `floor_capacity_ratio`：地面容纳率（默认 0.4）
- `wall_capacity_ratio`：墙面容纳率（默认 0.5）
- `similarity_threshold_floor/wall`：资产检索相似度阈值

故障排查

常见问题

问题	原因	解决方案
`FileNotFoundError: Path does not exist`	缺少 OBJathor 数据	运行数据下载命令
OpenAI API 错误	API 密钥无效或无网络	检查 API 密钥和网络连接
Unity 连接失败	AI2-THOR 版本不匹配	使用指定的 git commit hash
对象放置失败	房间太小或容纳率过低	增加 `floor_capacity_ratio`

调试模式

设置环境变量启用详细日志：

```
export DEBUGGING=1
python ai2holodeck/main.py --query "a living room" --openai_api_key your-key
```

引用

如果在您的工作中使用 Holodeck，请引用以下论文：

```
@InProceedings{Yang_2024_CVPR,
    author    = {Yang, Yue and Sun, Fan-Yun and Weihs, Luca and VanderBilt,
Eli and
                 Herrasti, Alvaro and Han, Winson and Wu, Jiajun and Haber,
Nick and
                 Krishna, Ranjay and Liu, Lingjie and Callison-Burch, Chris
and
                 Yatskar, Mark and Kembhavi, Aniruddha and Clark,
Christopher},
```

```
    title     = {Holodeck: Language Guided Generation of 3D Embodied AI
Environments},
    booktitle = {Proceedings of the IEEE/CVF Conference on Computer Vision
and
                 Pattern Recognition (CVPR)},
    month     = {June},
    year      = {2024},
    pages     = {16227-16237}
}
```

资源链接

- **GitHub 仓库**：https://github.com/allenai/Holodeck
- **项目主页**：https://yueyang1996.github.io/holodeck/
- **AI2-THOR 文档**：https://ai2thor.allenai.org/
- **OBJathor 仓库**：https://github.com/allenai/objathor
- **论文 (arXiv)**：https://arxiv.org/abs/2312.09067