项目申请书

项目名称: 增强 Embodied-Reasoner：基于精细化三维感知与多轮对话的交互能力提升

项目主导师: 李鹏 (lipeng@iscas.ac.cn)

申请人: 刘嘉俊

日期: 2025.06.13

邮箱: jedward.jiajun@gmail.com

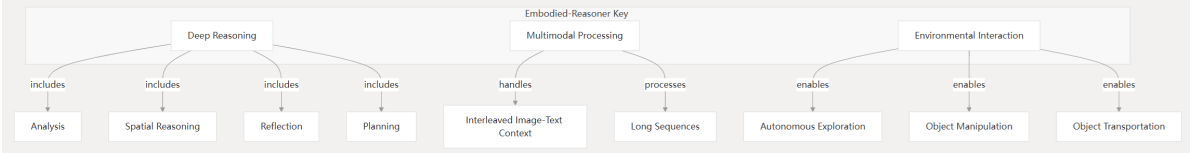
1. 项目背景与现状分析

Embodied-Reasoner 是一种将深度思考能力扩展到具身交互任务的新方法。其核心逻辑是：有效的具身推理不仅需要处理多模态输入的能力，还需要生成适应交互不同阶段的多样化思考过程（分析、规划、反思）。

然而，在 Embodied-Reasoner 迈向真实、复杂世界的道路上，仍面临两大核心挑战：

1. **同名物体混淆**： 在存在多个同名物体（如图书馆中的多个书架）时，仅依赖物体名称进行导航和交互会导致决策困难。
2. **大型物体交互精度不足**： 对于大型物体（如L型沙发），导航到其大致区域后，难以精确指定并操作其特定部位（如扶手、坐垫），限制了任务的成功率和精细度。

本项目旨在正面应对上述挑战，通过引入先进的**三维精细化感知**与**多轮澄清式对话**能力，将 Embodied-Reasoner 打造为能够在复杂现实环境中进行鲁棒、精确、自然交互的智能体。



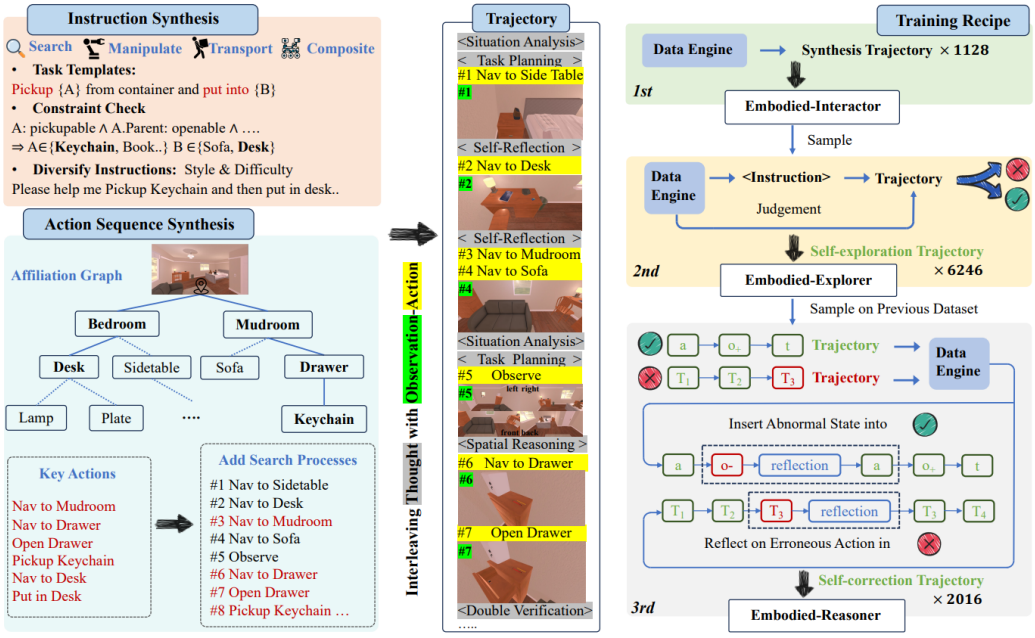


Figure 3. *Left*: Data Engine for <Instruction, Interactive Trajectory> synthesis. First, we synthesize instructions from task templates, and build an affiliation graph from scene's meta-data. It enables us to derive key actions needed for task. We add exploratory actions and insert thinking thoughts between observation and actions. *Right*: Three-stage training recipe. ①We fine-tune on synthesized trajectory to develop interaction skills. ②We sample multiple trajectories on novel tasks and evaluate their correctness. The successful ones are used for developing its exploring abilities. ③We continue to sample trajectories using updated model, injecting anomalous states and reflective thoughts in successful cases and correcting errors in failed ones. This self-correction training yields *Embodied-Reasoner*.

基于对当前 Embodied-Reasoner 代码库的了解与分析（基本框架见上图），以及协同视觉搜索、推理与行动的具身推理器Embodied Reasoner 和 讲座 | 具身推理模型Embodied-Reasoner让机器人学会思考与交互——浙江大学计算机学院博士生张文祺，一些简要的现状分析如下：

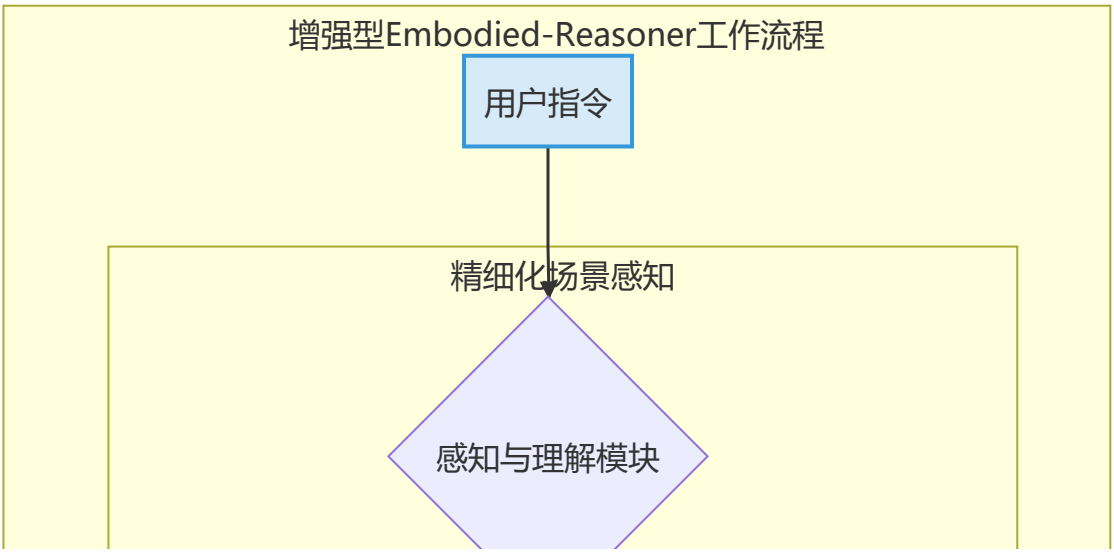
当前系统架构简析

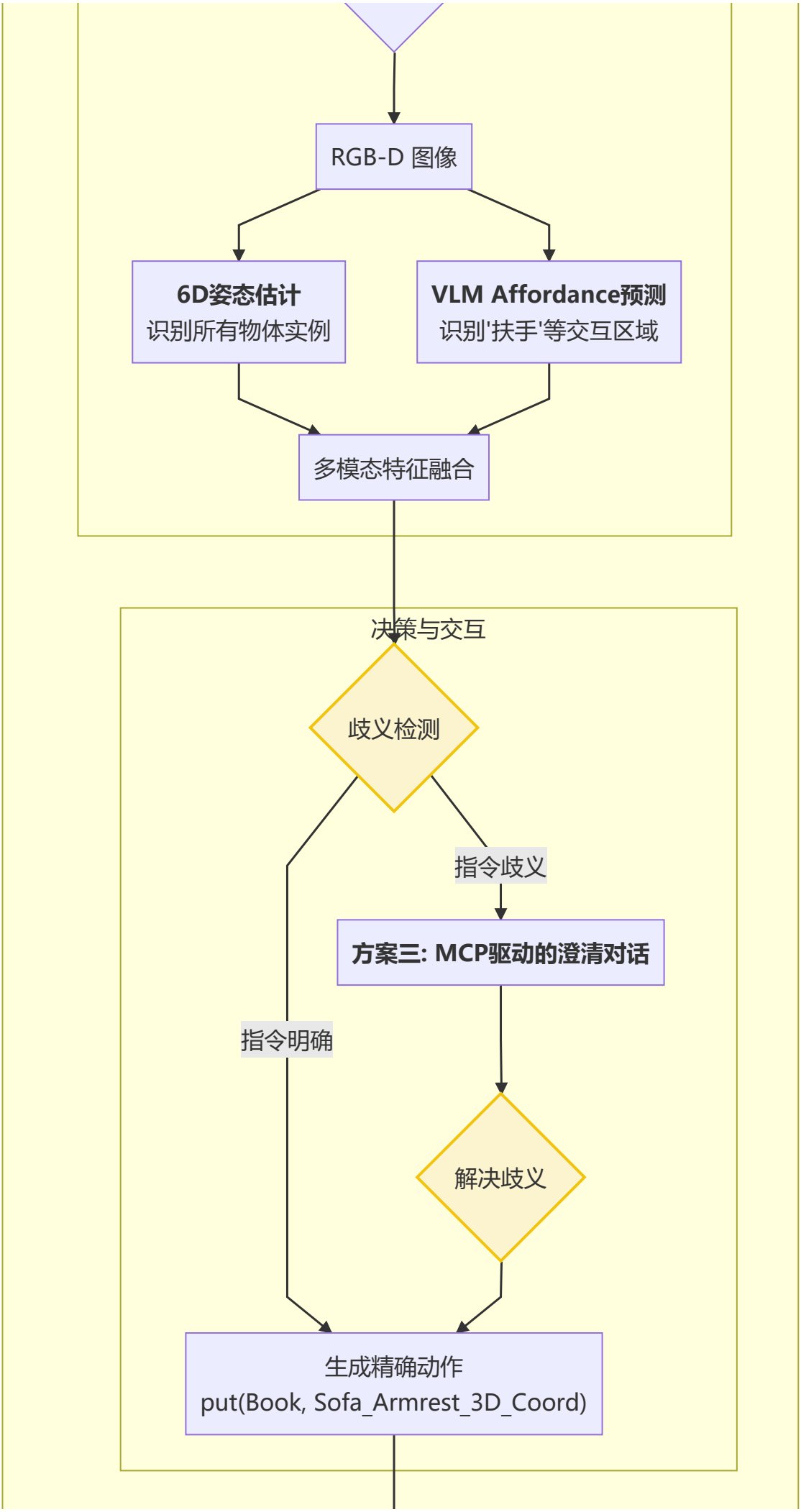
当前系统主要通过 `EventObject` 类管理物体状态和属性 [eventObject.py : 5-17]，通过 `BaseAgent` 类实现基础导航计算 [baseAgent.py : 67-109]，并由 `RocAgent` 类提供具体的移动和交互功能 [RocAgent.py : 5-14]。

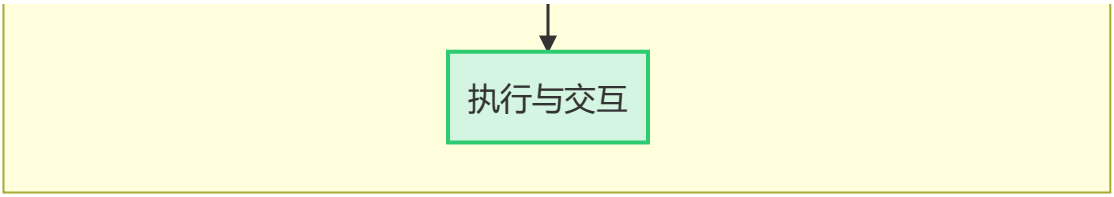
现有系统在物体定位方面主要依赖简单的几何角度计算和距离匹配（`calculate_best_view_angles` 函数），缺乏基于深度信息的精确6D姿态估计能力。物体管理通过 `EventObject` 类的 `item2object` 字典进行名称到物体实例的映射，但无法区分多个同名物体实例，导致在复杂场景中出现定位歧义。

2. 研究内容与技术方案

为应对 Embodied-Reasoner 在**同名物体混淆**与**大型物体交互精度不足**两大核心挑战，本项目将从**精细化感知**和**多轮对话**两个方面展开，并最终完成**系统集成与评估**。整体技术框架如下图所示：




RGB-D 图像
6D姿态估计
识别所有物体实例
VLM Affordance预测
识别'扶手'等交互区域
多模态特征融合
决策与交互
歧义检测
指令歧义
方案三: MCP驱动的澄清对话
指令明确
解决歧义
生成精确动作
put(Book, Sofa_Armrest_3D_Coord)



具体研究内容如下：

2.1 精细化场景感知：从"在哪"到"是哪个"与"放哪里"

为了让智能体不仅能找到物体，还能精确区分并与之进行精细交互，我们计划从两个层面增强其感知能力：

2.1.1 基于点云的6D姿态估计，实现同名物体区分

问题分析：当多个同名物体（如多个书架）存在时，仅靠类别和二维位置不足以进行唯一指代。引入三维空间信息是解决该问题的关键。

技术方案：我们将构建一个多级空间身份识别系统，为场景中的每个物体实例生成一个唯一的、稳定的空间签名。该系统将综合利用AI2THOR环境提供的深度信息。

1. **空间网格定位**：将场景划分为固定大小（如25cm × 25cm）的网格，为每个物体分配粗略的网格坐标，实现初步区域区分。
2. **点云质心精定位**：利用深度图生成物体点云，计算其三维几何质心，将定位精度提升至厘米级，用于区分邻近摆放的同类物体。
3. **姿态与尺寸编码**：计算物体的6D姿态和尺寸信息，形成一个更完备的描述子，以区分姿态或大小有差异的同类物体。

通过这套机制，当智能体需要与特定物体交互时，它不再依赖模糊的名称，而是使用精确的空间签名进行定位，从而解决同名物体的混淆问题。

2.1.2 基于VLM的交互适宜区预测，实现大型物体精细操作

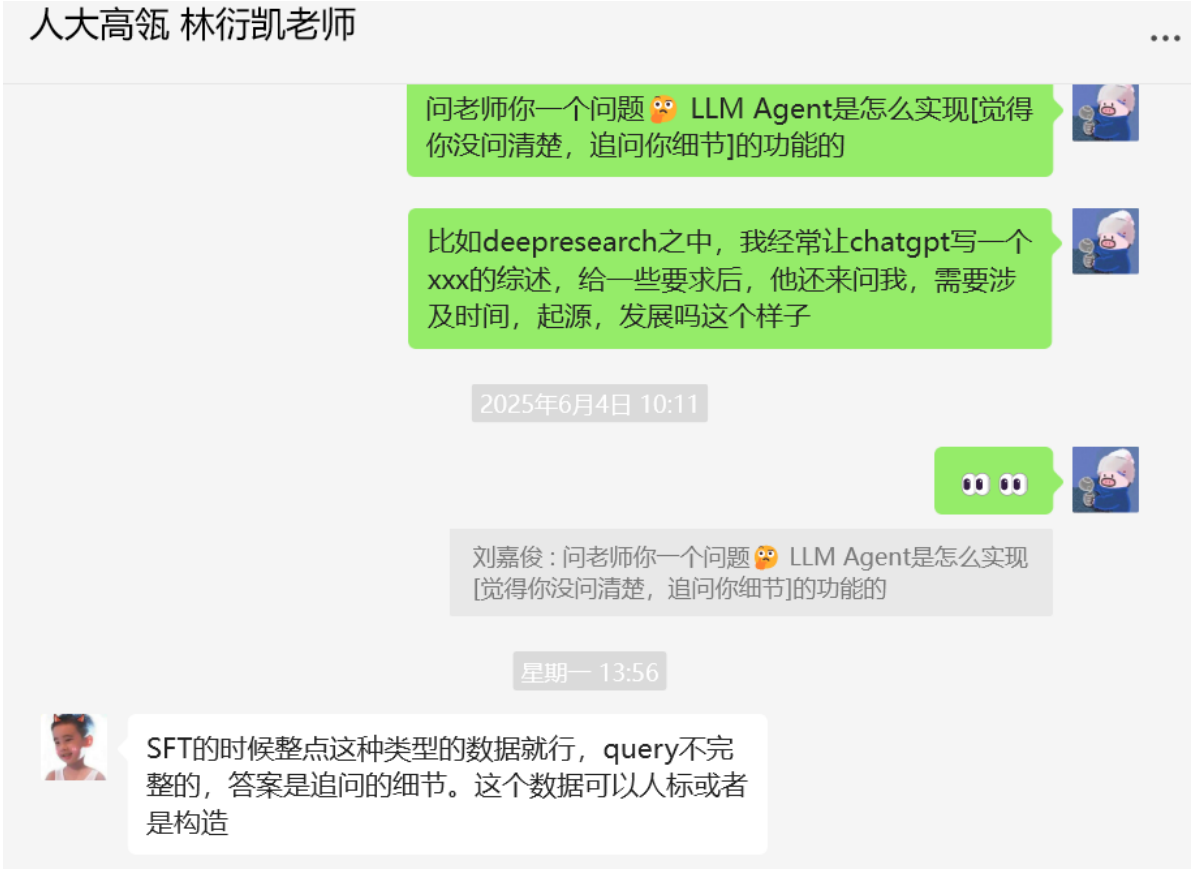
问题分析：对于大型物体（如沙发、床），用户指令往往涉及其特定区域（如"把遥控器放在沙发扶手上"）。传统的几何分割方法（如基于曲率、梯度的分割）对不同形态的家具泛化能力差，且无法理解"扶手"这类带有功能语义的区域。

核心思路：我们提出一个更根本的解决方案：利用VLM自身的视觉推理能力来理解和定位这些功能区域，预测交互适宜区（Affordance），即让模型根据任务需求，直接在图像上"画出"可以进行交互的区域。

技术方案：

1. **VLM驱动的Affordance预测**：当收到针对大型物体的指令时，系统将当前的第一视角图像和任务指令共同输入给VLM，并要求其输出一个"热力图"。这个输出直观地标示了图像中最适合执行该任务的区域。
2. **从2D Affordance到3D交互点**：获得2D热力图后，结合深度图，我们可以将热力值最高的区域投影到三维空间，计算出精确的3D交互坐标。
3. **数据与微调**：为实现此功能，需要对VLM进行SFT微调。我们将构建一个`(图像，指令，Affordance掩码)`格式的训练数据集。

   数据可通过半自动化方式构建：利用现有带部件标注的3D模型库，在AI2THOR中渲染，自动生成所需数据对。



方案优势：此方案将复杂的几何问题转化为VLM擅长的视觉-语言对齐问题，不仅更智能、泛化能力更强，也更符合Embodied-Reasoner项目的整体技术路线。

```
class AffordancePredictor:
    def __init__(self, vlm_model):
        self.vlm = vlm_model

    def predict(self, rgb_image, depth_map, instruction):
        """
        Use VLM to predict interaction regions and calculate 3D interaction
points.
        :param rgb_image: Current view RGB image
        :param depth_map: Corresponding depth map
        :param instruction: Task instruction
        :return: A dictionary containing 3D coordinates and confidence
        """
        # Step 1: Query VLM to get a 2D affordance heatmap
        prompt = f"Given the image, generate a heatmap indicating where one
could '{instruction}'. Highlight the most suitable area."
        heatmap_2d = self.vlm.generate_heatmap(rgb_image, prompt)

        # Step 2: Find the peak of the heatmap
        peak_yx = self.find_heatmap_peak(heatmap_2d)

        # Step 3: Project 2D peak to 3D space
        interaction_point_3d = self.project_2d_to_3d(peak_yx, depth_map)

        confidence = heatmap_2d[peak_yx[0], peak_yx[1]]
        return {
            'coordinates': interaction_point_3d,
            'confidence': confidence
        }
```

2.1.3 技术整合：多模态特征融合

问题分析：智能体在决策时需要一个统一的、全面的场景表示。如何将上述的点云空间特征、VLM生成的Affordance特征以及原始的RGB视觉特征有效结合，这决定着系统性能的上限。

技术方案：我们将使用注意力机制来动态地权衡三种模态的重要性。

1. **特征编码**：为空间、视觉、Affordance三种模态设计专门的编码器，将其转换为维度统一的特征向量。
2. **注意力融合**：利用交叉注意力机制，让不同模态的特征相互"关注"，捕捉其内在关联。
3. **动态加权**：最终，系统会根据当前任务的性质动态调整各模态的权重。

2.2 多轮澄清式对话：解决指令歧义

问题分析：当用户的指令存在歧义时，现有系统无法处理，我们期望 Agent 能主动发现歧义，并通过对话澄清用户的意图。

2.2.1 歧义检测与澄清策略

技术方案：考虑以下三层级的歧义检测，设计相应的澄清策略。

1. **语义歧义检测**：用NLP分析指令本身的完整性。若指令缺少必要元素，则标记为歧义。
2. **场景歧义检测**：将指令与3.1节获得的场景感知结果进行比对。如果指令中的指代对象在场景中有多个实例，则标记为歧义。
3. **交互歧义检测**：当指令涉及大型物体的不同部位时，系统会检查Affordance预测是否返回了多个离散的候选区域，若是则标记为歧义。

澄清策略：一旦检测到歧义，系统会启动一个澄清对话。澄清问题的生成将是上下文感知的，例如，通过"您是指窗边的那个书架，还是门边的那个？"这样的方式，利用空间关系帮助用户区分。

2.2.2 MCP驱动的对话状态管理

问题分析: 简单的对话历史累积不足以支撑复杂的澄清逻辑。我们需要一个结构化的方法来管理澄清过程本身的状态。

技术方案: 为此，我们提出并设计一个**多轮澄清协议（Multi-turn Clarification Protocol, MCP）**。MCP不仅仅是记录对话，更是一个微型的状态机，专门用于追踪和管理从歧义检测到最终解决的全过程。

1. **MCP核心状态**:
   - `IDLE`: 系统处于待命状态，无正在处理的歧义。
   - `AMBIGUITY_DETECTED`: 系统检测到指令歧义，协议上下文中会记录所有候选对象及其特征。
   - `AWAITING_CLARIFICATION`: 系统已生成并发出澄清问题，正在等待用户响应。
   - `RESOLVED`: 用户响应已收到，歧义成功解决，协议上下文中记录了唯一确定的目标。
2. **MCP状态跟踪器**: 我们将实现一个`MCPStateTracker`模块，负责在对话过程中驱动状态转换。例如，当歧义被检测到，状态从`IDLE`变为`AMBIGUITY_DETECTED`；当问题发出后，变为`AWAITING_CLARIFICATION`。这使得系统的行为更加清晰和可预测。
3. **数据驱动**: 澄清模型的训练数据也将围绕MCP进行构造，每一条数据样本将包含完整的状态转换序列，让模型不仅学习生成问题，更能理解在特定澄清状态下的行为逻辑。

```
class MCPStateTracker:
    def __init__(self):
        self.state = 'IDLE'
        self.context = {}
```

```
    def transition_to(self, new_state, context=None):
        self.state = new_state
        if context:
            self.context.update(context)

class ClarificationDialogueManager:
    def __init__(self, scene_analyzer, vlm_model):
        self.scene_analyzer = scene_analyzer
        self.vlm = vlm_model
        self.mcp_tracker = MCPStateTracker()

    def process_command(self, user_input, conversation_history):
        """
        Process user commands based on MCP.
        """
        current_state = self.mcp_tracker.state

        if current_state == 'IDLE':
            ambiguity_info = self.scene_analyzer.detect_ambiguity(user_input)
            if not ambiguity_info['is_ambiguous']:
                return {'type': 'execution', 'action':
self.resolve_action(user_input)}
            else:
                self.mcp_tracker.transition_to('AMBIGUITY_DETECTED',
{'candidates': ambiguity_info['candidates']})

        if self.mcp_tracker.state == 'AMBIGUITY_DETECTED':
            question =
self.generate_clarification_question(self.mcp_tracker.context['candidates'])
            self.mcp_tracker.transition_to('AWAITING_CLARIFICATION',
{'question': question})
            return {'type': 'clarification_needed', 'question': question}

        if self.mcp_tracker.state == 'AWAITING_CLARIFICATION':
            resolved_target = self.process_clarification_response(user_input,
self.mcp_tracker.context)
            if resolved_target:
                self.mcp_tracker.transition_to('RESOLVED', {'target':
resolved_target})
                return {'type': 'execution', 'action':
self.resolve_action_with_target(resolved_target)}
            else:
                # Failed to resolve, try to ask again
                pass

    def generate_clarification_question(self, candidates):
        # ... (same as before)
        prompt = f"I see multiple {candidates[0]['name']}. Do you mean the one
near the {candidates[0]['relation']} or the one near the {candidates[1]
['relation']}?"
        question = self.vlm.ask(prompt)
        return question
```

3. 个人背景匹配

申请人：刘嘉俊
院校：中国人民大学 高瓴人工智能学院
学历：本科在读（2023年9月入学）
学业表现：GPA 3.84/4.00，第一、三学期专业排名第1
联系方式：jedward.jiajun@gmail.com | +86 186-7655-1927

基于本项目的技术需求，我的技术背景与项目要求高度匹配，同时具备快速学习能力和项目管理能力，而且暑假会一直待在北京，甚至可以线下与导师沟通交流：

核心技术能力

- **深度学习框架**：熟练使用PyTorch，具备模型训练、调优和部署经验
- **计算机视觉**：了解点云处理、3D姿态估计相关技术栈
- **自然语言处理**：具备扎实的NLP基础，在课程项目中多次实践大语言模型相关的应用开发。
- **Linux开发**：熟悉Linux开发与Shell脚本，具备扎实的系统级编程能力。

算法编程能力证明

- **ACM-ICPC国际大学生程序设计竞赛亚洲区决赛银牌**（2024年12月）
- **CCPC"小米杯"全国大学生程序设计竞赛定向邀请赛金牌**（2025年4月）
- **CCF CCSP大学生计算机系统与程序设计竞赛银牌**（2024年10月）

相关研究经验

- **导师**：李崇轩教授
- **研究领域**：深度生成模型，专注3D/4D生成模型
- **技术关联**：3D场景理解和生成技术与本项目的6D姿态估计需求相关

为什么选择这个项目?

我对此项目抱有浓厚兴趣，因为它完美契合我当前的研究方向与技术热情。

1. **技术背景高度匹配**：我的研究经验聚焦3D生成模型，这与项目所需的**6D姿态估计**和**三维场景理解**能力直接相关。同时，我在NLP课程中积累的**大语言模型应用经验**，也为开发本项目核心的**多轮澄清对话系统**和**VLM交互适宜区预测**打下了坚实基础。
2. **解决核心问题的能力**：我不仅理解项目旨在解决的**同名物体混淆**和**大型物体交互**两大痛点，更对申请书中提出的技术方案（如点云空间签名、Affordance预测）有深入思考。我的算法竞赛背景保证了我具备将复杂方案转化为高质量代码的实现能力。
3. **对开源与具身智能的热情**：我长期关注具身智能领域，并渴望能为 `Embodied-Reasoner` 这样的前沿开源项目贡献力量。参与"开源之夏"不仅是提升个人能力的机会，更是我践行开源精神、推动技术发展的绝佳平台。

4. 规划

项目总工期：200小时，预计14周完成
项目周期：2025年7月1日 - 2025年9月30日

第1-2周（07月01日 - 07月14日）：环境搭建与方案细化
预计工时：30小时

- [ ] 深入研读Embodied-Reasoner代码库，理解现有架构，搭建开发环境。
- [ ] 分析`EventObject`，`BaseAgent`，`RocAgent`等核心模块与数据流。
- [ ] 结合代码库，完成最终技术方案的细节敲定和可行性验证。

第3-4周（07月15日 - 07月28日）：6D姿态估计模块开发
预计工时：35小时

- [ ] 实现基于点云的6D姿态估计算法，完成多级空间身份识别系统。

- [ ] 开发用于区分同名物体的空间签名生成与匹配逻辑。
- [ ] 单元测试该模块在不同场景下的定位与区分精度。

第5-6周（07月29日 - 08月11日）：VLM Affordance预测模块开发
预计工时：35小时

- [ ] 实现`AffordancePredictor`类，开发VLM驱动的交互适宜区预测功能。
- [ ] 完成2D热力图到3D交互点的投影计算。
- [ ] 并行开始多模态特征融合模块（`MultimodalFeatureFusion`）的初步开发，设计不同模态的编码器。

第7周（08月12日 - 08月18日）：MCP对话管理模块开发
预计工时：20小时

- [ ] 实现`MCPStateTracker`作为核心状态机。
- [ ] 开发`ClarificationDialogueManager`，集成歧义检测与上下文问题生成逻辑。
- [ ] 初步测试在模拟歧义场景下的澄清对话流程。

第8-9周（08月19日 - 09月01日）：系统集成与数据生成
预计工时：30小时

- [ ] 将感知模块（姿态估计、Affordance预测）与对话模块集成到Agent主逻辑中。
- [ ] 开发自动化脚本，利用PartNet等三维模型库在AI2THOR中生成`(图像, 指令, Affordance掩码)`格式的训练数据。
- [ ] 搭建VLM微调所需的基础设施。

第10-11周（09月02日 - 09月15日）：模型微调与评估框架
预计工时：25小时

- [ ] 在生成的数据集上对VLM进行微调，以优化Affordance预测的准确性。
- [ ] 设计并实现专门的评估框架，包含同名物体区分、大型物体精细操作、多轮澄清效率三大场景。
- [ ] 准备基线模型用于后续的性能对比。

第12-13周（09月16日 - 09月29日）：全面评估与迭代优化
预计工时：20小时

- [ ] 在评估框架中对增强后的系统进行全面测试，并与基线系统进行性能对比。
- [ ] 根据评估结果分析系统瓶颈，进行针对性的性能优化与Bug修复。
- [ ] 完善各模块的错误处理与降级策略。

第14周（09月30日）：文档完善与项目交付
预计工时：5小时

- [ ] 编写完整的API文档、使用指南和项目总结报告。
- [ ] 准备最终的项目成果结题。

质量保证

- 每周进行Code Review和单元测试。
- 定期与导师沟通同步项目进展，及时调整技术方案。
- 采用Git进行完善的版本控制。

3. 个人期待与承诺

我期望通过本项目的实施，达成以下目标：

- **项目成果**：成功增强Embodied-Reasoner，显著提升其在复杂场景中的任务成功率，为社区贡献一个高质量、可复现的具身智能解决方案。
- **个人成长**：在实践中深度掌握具身智能、多模态AI和系统工程的前沿技术，提升自己从定义问题到交付完整解决方案的全链路能力。

同时，我做出长期维护承诺，在项目周期结束后，我将继续以社区贡献者的身份，长期关注并维护本项目的代码，响应社区的issue和pull request，修复潜在的Bug。

我有信心在规定时间内交付高质量的项目成果，为Embodied-Reasoner的发展做出贡献。

刘嘉俊