arXiv:2603.00905v1 [cs.CV] 1 Mar 2026

# PySpatial：用于零样本空间推理的三维视觉程序生成

**ZhanpengLuo**[1], [2*] **CeZhang**[1*] **Silong Yong**[1] **Cunxi Dai**[1] **Qianwei Wang**[1,3]
**Haoxi Ran**[1] **Guanya Shi**[1] **Katia Sycara**[1] **Yaqi Xie**[1][1]卡内基梅隆大学 [2]匹兹堡大学 [3]
密歇根大学

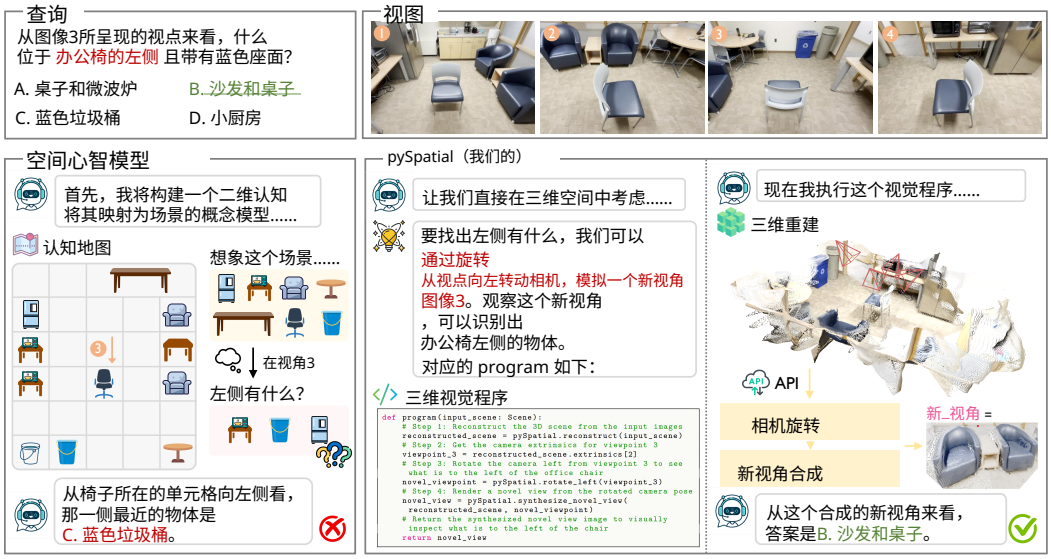


图1： **将我们的 pySpatial 与用于多视角空间推理任务的空间心理模型进行比较**。不同于依赖多模态大语言模型隐式想象来构建二维认知地图的空间心理模型（Yin 等人，2025），我们提出了pySpatial，这是一种视觉编程框架，可灵活组合空间工具（例如三维重建、相机运动和新视角合成），使多模态大语言模型能够在三维空间中进行显式推理，以完成各种空间推理任务。

## 摘要

多模态大语言模型（MLLMs）展现出强大的通用感知与推理能力，但在需要理解三维世界空间信息的任务上仍面临困难。为此，我们提出pySpatial，这是一个视觉编程框架，通过 Python 代码生成使多模态大语言模型能够调用空间工具。给定一段图像序列和一个自然语言查询，模型会组合调用空间工具的函数，包括三维重建、相机位姿恢复、新视角渲染、等。这些操作将原始二维输入转换为可供探索的三维场景，使多模态大语言模型能够基于结构化空间表示进行显式推理。值得注意的是，pySpatial 无需基于梯度的微调，可在完全零样本设置下运行。在具有挑战性的 MindCube 和 Omni3D-Bench 基准测试上的实验评估表明，我们的框架pySpatial始终优于强大的多模态大语言模型基线；例如，在 MindCube上，其表现比 GPT-4.1-mini 高出 12.94%。此外，我们还开展了真实世界的室内导航实验，机器人能够利用pySpatial生成的路线规划成功穿越复杂环境，彰显了我们方法的实用性。项目网站见 https://pySpatial.github.io。

*贡献相同。

## 1 引言

多模态大语言模型（MLLMs）在图像描述（Bucciarelli et al., 2024; Zhang et al., 2025b）、指代表达定位（Kazemzadeh et al., 2014）、视频理解（Zeng et al., 2025; Fu et al., 2025）和鲁棒推理（Mathew et al., 2021; Zhang et al., 2025a）等多种任务中取得了显著成功。然而，这些进展并未转化为稳健的三维空间推理能力：近期研究（Wu et al., 2025; Chen et al., 2024a; Chang et al., 2025）表明，多模态大语言模型仍难以应对从单张图像中判断物体相对位置或估计深度等基础任务（Liu et al., 2023; Cheng et al., 2024），到对自我中心运动和多视图关系进行更复杂推理等各类挑战（Yin et al., 2025; Yang et al., 2025）。这些局限严重阻碍了此类模型在机器人学、增强现实和具身智能等安全关键应用中的可靠部署；在这些应用中，导航、操作和人机交互等任务都依赖精准的空间理解（Li et al., 2024; Duan et al., 2024; Song et al., 2025; Qiao et al., 2025）。

尽管近期研究（Chen 等，2024a；Cheng 等，2024）主要致力于提升模型从单张图像中进行空间理解的能力（例如，“凳子在烤箱前面吗？”），但本文聚焦于更具挑战性的三维空间推理问题：环境仅通过有限视图被部分观测，模型必须跨视角进行推理，才能回答诸如“我应该如何从视图 *1* 移动到视图 *2*？”之类的问题。在这种情形下，最先进的多模态大语言模型表现仅略高于随机猜测（Yin 等人，2025）。近期研究（Chen 等，2024a；Ma 等，2025）表明，这一弱点很大程度上源于训练数据：尽管多模态大语言模型在互联网规模的图像-描述配对数据上进行预训练，但明确的三维监督既稀缺又代价高昂，导致模型难以学习语言与三维空间结构之间可靠的对应关系，进而限制了模型在三维空间中有效推理的能力。近期，Yin 等人 (2025) 探索了使用二维认知地图等数据结构的方法：模型在俯视图中编码物体的位置，以便在头脑中模拟空间布局，如图1所示。然而，这些方法仍依赖隐式的“想象”机制，效果有限。

这些局限促使我们提出核心研究问题：如何赋予多模态大语言模型在三维空间中进行显式推理的能力？实现这一目标的自然第一步，是获取一个明确的几何基础，让此类推理能够在其上进行。前馈式三维重建领域的最新进展（Wang 等人，2024年b；2025a）让这一目标成为可能：它们能够直接从稀疏二维视图中恢复场景几何，包括相机参数、深度图和场景级点云。这类表示将有限的二维视图转化为一个可探索的三维场景，模型可在其中执行空间变换（下文简称空间工具），例如相机平移、旋转和视点切换，以丰富视觉上下文并构建交互式推理链。例如，对于“如果我位于视角*3*，身后是什么”这一问题，模型可以在重建场景中指定的视点处将虚拟相机旋转180°，从而揭示此前被遮挡的区域，并以几何证据为其推理提供依据。

然而，如何使多模态大语言模型能够灵活组合空间工具，并以具备上下文感知能力的方式与三维环境无缝交互，仍是一项关键挑战。为此，受视觉编程领域开创性工作的启发（Gupta 和 Kembhavi，2023；Surís 等人，2023），我们提出了`pySpatial`，这是一个将 GPT-4o 等多模态大语言模型用作 Python 代码生成代理的框架，可调用函数来使用三维重建、动作的自然语言描述和新视角合成等工具。如图1所示，`pySpatial` 利用定义完善的应用程序接口，自动选择并组合适当的工具，以解决各种空间推理任务。值得注意的是，`pySpatial` 完全在零样本设置下运行，是一个适用于开源和闭源多模态大语言模型的即插即用框架，能够提供可解释的解决方案和可靠的响应，因此非常适合各种现实世界任务。

我们在 MINDCUBE 和 OMNI3D-BENCH 基准测试上评估了该方法的有效性。结果表明，pySpatial 始终大幅优于强大的多模态大语言模型基线（例如，在 MINDCUBE 上比 GPT-4.1-mini 提升 12.94%）。定性分析进一步验证，该方法能够生成高质量、可执行且可解释的视觉程序，从而以零样本方式有效解决复杂的空间推理任务。此外，我们还将 pySpatial 应用于现实世界的室内导航，使四足机器人能够利用生成的路线规划成功穿越复杂环境。

我们的贡献可概括如下：

• 我们提出了 pySpatial，这是一种新颖的零样本框架，可通过生成并执行视觉程序，让多模态大语言模型以结构化、组合式的方式运用各种空间工具，在三维空间中进行显式推理，从而解决多样的空间推理任务。

• 我们在 MINDCUBE 和 OMNI3D-BENCH 上评估了 pySpatial，其表现优于强大的多模态大语言模型基线。定性分析验证了 pySpatial 能够可靠地生成可执行、可解释的视觉程序，以完成多样的空间推理任务。

• 我们进一步评估了 pySpatial 在室内导航任务中的实际效果。结果表明，它能够生成路线规划，使四足机器人得以穿越复杂环境，展现出在实际应用场景中的巨大潜力。

## 2 相关工作

用于空间推理的多模态大语言模型。近期的多模态大语言模型在多模态任务上展现出卓越的性能（Wan 等，2025；Alayrac 等，2022；Zhang 等，2024；2026）。然而，研究表明，这些模型在理解空间关系方面存在显著局限（Yu 等，2024；Kamath 等，2023；Wang 等，2024a；Tong 等，2024），而空间关系理解是众多实际应用的关键前提，包括机器人操作（Huang 等，2022；Shridhar 等，2023）和具身导航（Qiao 等，2025；Huang 等，2023）。为解决这一问题，近期的研究（如 SpatialVLM（Chen 等，2024a）和 SpatialRGPT（Cheng 等，2024））通常提出可扩展的数据合成与数据整理流程，通过大规模预训练增强单视图空间推理能力。尽管取得了这些进展，近来的研究（Yin 等人（2025）；Huang 等，2024；Ma 等，2023）表明，当前的多模态大语言模型及此类方法在多视图场景下的几何理解和视角转换方面仍面临困难。在本研究中，我们是最早着手应对这一挑战的团队之一，并提出了一种名为 pySpatial 的新型零样本视觉编程框架。该框架系统地组合并应用各种空间工具，使模型能够显式地进行三维推理并解决多样的空间任务。

**三维重建**。传统的三维重建方法，例如运动恢复结构（Schon-berger & Frahm，2016），通常包含多个阶段，并且往往依赖耗时的优化流程。近年来，诸如 DUSt3R（Wang 等人，2024年b）、MASt3R（Leroy et al.，2024）、CUT3R（Wang 等人，2025b）和 VGGT（Wang 等人，2025a）等前馈式三维重建方法，利用大规模三维预训练和视觉变换器，直接预测像素对齐的三维点图。这些数据驱动方法展现出很强的泛化能力，即使在视图不重叠的场景中也是如此。在此进展基础上，后续研究将前馈式三维重建拓展至神经渲染（Charatanetal.，2024）、同步定位与建图（Maggio et al.，2025）和动态重建（Lin et al.，2025；Luo et al.，2025）。

**模块化视觉推理**。为增强组合式多模态理解，近期进展将视觉专家（如 GroundingDINO（Liu 等，2024）和 SAM（Ravi 等，2025））视为符号算子，并将它们组合起来以解决复杂的推理问题。代表性工作如 Visual ChatGPT（Wu 等，2023）、MM-REACT（Yang 等，2023）和 HuggingGPT（Shen 等，2023）沿此方向，将大语言模型与预定义工具链集成，以处理多模态输入。在此基础上，VisProg（Gupta 和 Kembhavi，2023）和 ViperGPT（Surís 等人，2023）引入了视觉编程，通过提示多模态大语言模型生成可执行的 Python 程序，经由预定义 API 调用一组视觉解析器，从而扩展了这一范式。最近，VADAR（Marsili 等，2025）将视觉编程范式引入单视图空间推理任务，并采用自适应 API 设计。相比之下，我们的`pySpatial` 提出了一个专为多视图空间推理设计的框架，为模型配备组合式三维工具，以应对多样而复杂的空间场景。

## 3 方法

本节介绍 pySpatial，这是一种视觉编程框架，可让多模态大语言模型通过生成并执行协调多种空间工具的视觉程序，在三维空间中进行显式推理，以应对各种空间推理任务。我们还将介绍该框架的设计，包括 pySpatial API 签名及其使用的空间工具。

### 3.1 问题形式化

我们考虑这样一种情形：向多模态大语言模型 $\mathcal{M}$ 提供图像序列 $\mathcal{I} = \{I_n\}_{n=1}^{N}$，其中每个视图的分辨率为 $H \times W$，并捕捉三维场景的局部观测；同时提供一个自然语言查询 $q$，询问物体之间的空间关系或相机运动。目标是从答案空间 $\mathcal{A}$中给出能够回答该查询的正确答案 $r^*$。

如第1节所述，我们通过前馈重建将有限的二维视图转换为可探索的三维场景。这会为每一帧生成一致的深度估计 $D$、相机内参 $K \in \mathbb{R}^{3\times3}$和外参$\mathbf{G} \in \mathrm{SE}(3)$。这些量共同定义了世界坐标系中的点云 $\mathcal{P}$，为下游推理提供几何基础。

此外，我们采用Surís 等人（2023）提出的程序合成视角。给定图像序列和查询（$\mathcal{I}$，$q$）作为输入，代码智能体 $\mathcal{F}$ 会生成一个 Python程序 $z$，通过定义明确的应用程序接口调用一组空间工具。解释器会执行该程序 $\mathcal{E}$ 以生成中间输出 $O$，其形式可根据程序而异，可以是文本、单张图像或图像列表 $z$。该输出提供直接的视觉证据，以支持回答查询。例如，当查询是“如果我位于视角*3*，身后是什么”时，程序会在指定视点将相机旋转180°，渲染出新的视图。最后，多模态大语言模型 $\mathcal{M}$ 会整合原始视觉输入和程序输出，生成最终答复 $r \in \mathcal{A}$。

### 3.2 空间工具与应用程序接口

为引导多模态大语言模型在三维空间中进行显式推理，我们引入了多种空间工具，例如三维重建、相机描述和新视角合成。我们提供了`pySpatial` 应用程序接口签名，见代码1，各工具的详细说明见以下各节。

**三维重建**。根据具体任务需求，我们采用两种前馈重建方法。对于度量尺度场景，我们使用 CUT3R（Wang 等人，2025b），它返回以真实世界单位表示的深度。当归一化单位空间中的相对距离足以满足需求时，我们采用 VGGT（Wang 等人，2025a），因为它具有良好的泛化能力。

```
class pySpatial: """用于三维视觉工具的pySptial接口。""" def
```

代码 1：`pySpatial` **应用程序接口签名**。

形式化地说，视图 $I_n$ 中的每个像素$\mathbf{p}_i \in \mathbb{R}^2$，其预测深度为 $D_n(\mathbf{p}_i)$，都会利用相机内参 $K$反投影到相机坐标系中，然后通过估计得到的位姿$\mathbf{G}_n \in \mathrm{SE(3)}$变换到世界坐标：

$$\mathbf{X}_i = \mathbf{G}_n^{-1}\,\pi^{-1}\big(\mathbf{p}_i, D_n(\mathbf{p}_i), K^{-1}\big), \quad (1)$$

其中 $\pi^{-1}$ 表示从图像坐标到相机坐标系中三维点的反投影。我们将所有帧中所有像素对应的$\mathbf{X}_i$ 拼接起来，从而得到世界空间中的点云 $\mathcal{P}$。

**相机描述**。我们将原始相机位姿矩阵转换为自然语言标签，使自我中心运动更易于理解传递给语言模型。每个位姿由一个外参矩阵 $\mathbf{G} = [\mathbf{R} \mid \mathbf{t}] \in \mathbb{R}^{3\times4}$，它将世界坐标中的点映射到相机坐标系中。对应的相机中心在世界坐标中为 $\mathrm{C} = -\mathrm{R}^{\top}\mathrm{t}$。给定两个位姿 ($\mathrm{R}_1$,$\mathrm{t}_1$) 和 ($\mathrm{R}_2$,$\mathrm{t}_2$)，世界坐标中的位移为 $\Delta\mathrm{C}_w = \mathrm{C}_2 - \mathrm{C}_1$。随后，我们将该位移表示在第一个相机的坐标系中，记为 $\Delta\mathrm{C}_1 = \mathrm{R}_1\,\Delta\mathrm{C}_w$。将位移限制在水平面内后，我们计算偏航角 $\theta = \mathrm{atan2}(d_x, d_z) \cdot 180/\pi$，其中 $(d_x, d_z)$ 分别是 $x$和 $z$分量，对应于 $\Delta\mathrm{C}_1$。该角度被离散化为八种典型运动类别（前进、后退、左侧、右侧以及四种对角方向），从而生成对自我中心运动的简洁自然语言描述。

**新视角合成**。为了便于对重建的三维场景进行主动探索，我们使智能体能够从任意相机位姿渲染新视图。给定一个点云 $\mathcal{P}$ 和一个

世界到相机变换 $\mathbf{G} = [\mathbf{R} \mid \mathbf{t}] \in \mathbb{R}^{3\times4}$，我们根据$\mathbf{G}$ 及相应的相机内参$\mathbf{K}$，将 $\mathcal{P}$ 光栅化为一幅 RGB图像。随后，智能体可以执行左转_和向后转_等高级动作，这些动作通过围绕世界 $y$轴旋转角度 $\phi$来实现。对相机到世界变换应用旋转并将其求逆，转换回世界到相机形式，即可得到更新后的相机位姿$\mathbf{G}'$。此操作可提供交互式视觉反馈，支持显式的空间推理。

### 3.3 3D视觉编程

**程序生成**。给定查询 $q$，代码智能体 $\mathcal{F}$ 合成一个 Python 程序 $z = \mathcal{F}(q)$，该程序由 `pySpatial` 应用程序接口中指定的函数调用组成。默认情况下，我们使用 GPT-4o 作为强大的多模态大语言模型基线；它经过互联网规模的 Python 代码数据训练，在代码生成方面表现出色。请注意，智能体仅与公共接口交互（例如，重建、旋转_右侧、合成_新颖_视图），无法访问模型权重、文件输入输出或渲染后端等内部实现细节。这种抽象将高层推理与底层执行分离。我们还为公共接口提供旋转和移动的默认参数，即旋转为 45，移动为 0.3，如代码所示：1。我们通过上下文示例引导程序合成，提示中包含接口文档和查询—代码对，但不包含标准答案。此外，我们利用结构化输出，先进行自由形式的自然语言推理，再合成 Python 代码。生成的 Python 代码（即视觉程序）充当显式的中间表示，用于编码一系列工具调用。它具有天然的可解释性，研究人员可以轻松检查、调试或修改生成的程序；同时也具有可组合性，能够与其他工具或下游推理模块无缝集成。程序构建完成后，由解释器执行，以产生具体的空间操作。

**程序执行**。在执行阶段，合成的程序 $z$ 由 Python 解释器 $\mathcal{E}$ 在输入图像序列 $\mathcal{I}$上执行，生成中间输出 $O = \mathcal{E}(z, \mathcal{I})$。根据查询内容，输出 $O$ 可以是文本、单张图像或一系列渲染视图。该中间输出将程序的推理步骤明确地锚定在可观测证据上。最后，多模态大语言模型 $\mathcal{M}$整合原始图像序列 $\mathcal{I}$、程序输出 $O$和自然语言查询 $q$，生成最终回答 $r = \mathcal{M}(\mathcal{I}, O, q)$。

## 4 实验

在本节中，我们评估 `pySpatial` 在 MindCube (Yin 等人，2025) 和 Omni3D-Bench (Marsili 等，2025) 上的有效性，并将其与现有最先进方法进行比较。

### 4.1 实验设置

**基准测试**。我们主要在 MINDCUBE（Yin 等人，2025）上评估我们的框架。该基准旨在探究多模态大语言模型在有限视图条件下的空间推理能力。具体而言，MINDCUBE 包含 21,000 多组空间问题与答案，基于 3,268 个多视图室内场景构建，涵盖三种典型的相机运动类型：旋转、环绕和穿行。我们还在 MINDCUBE-1k 上进行评估，这是 MINDCUBE 的一个子集，包含 1,050 个问题，专为评估而设计。此外，遵循先前的工作（Marsili 等，2025），我们也在 OMNI3D-BENCH 上评估框架。该基准是一个单视图空间推理基准，用于考察我们的视觉编程方法能否推广到多视图场景之外。

**基线**。我们将`pySpatial` 与现有基线的四类方法进行比较：（1）开放权重的多图像多模态大语言模型，例如 LLaVA-OneVision-7B（Li 等，2025）和 Qwen2.5- VL-3B-Instruct（Bai 等，2025）；（2）专有多模态大语言模型，包括 GPT-4o、GPT-4.1-mini 和 Claude-4-Sonnet；（3）专用空间模型，例如 Space-Qwen（Chen 等，2024a）和 VLM-3R（Fan 等，2025），以及（4）先前的视觉编程方法，例如 ViperGPT（Surís 等人，2023）、VisProg（Gupta 和 Kembhavi，2023）和 VADAR（Marsili 等，2025）。

**实现细节**。默认情况下，我们遵循以往的视觉编程工作（Marsili 等，2025），利用 GPT-4o 作为代码智能体来生成 Python 程序，并针对查询生成最终答复。我们使用 VGGT（Wang 等人，2025a）作为 MindCube 和 Omni3D-Bench 基准测试中的三维重建模型。对于真实世界导航，我们使用 CUT3R（Wang 等人，2025b），它提供度量尺度重建，而非归一化输出。对于点云栅格化，

表 1：**在完整的 MindCube (Yin 等人，2025) 数据集上的性能比较**。最佳结果以**粗体**显示，次佳结果带下划线。请注意，由于预算限制，我们在此数据集上使用 GPT-4.1-mini 作为代码智能体来实现pySpatial。

| 方法 | 参考 | 总体 | 旋转 | 之间 | 周围 |
|---|---|---|---|---|---|
| 基线 | | | | | |
| 随机（机会） | - | 32.35 | 36.36 | 32.29 | 30.66 |
| 随机（频率） | - | 33.02 | 38.30 | 32.66 | 35.79 |
| 开放权重多图像模型 | | | | | |
| LLaVA-OneVision-7B | Li 等 (2025) | 47.43 | 36.45 | 48.42 | 44.09 |
| LLaVA-Video-Qwen-7B | Zhang 等(2025c) | 41.96 | 35.71 | 43.55 | 30.12 |
| mPLUG-Owl3-7B-241101 | Ye 等 (2025) | 44.85 | 37.84 | 47.11 | 26.91 |
| InternVL2.5-8B | Chen 等 (2024年b) | 18.68 | 36.45 | 18.20 | 13.11 |
| Qwen2.5-VL-7B-Instruct | Bai 等 (2025) | 29.26 | 38.76 | 29.50 | 21.35 |
| Qwen2.5-VL-3B-Instruct | Bai 等 (2025) | 33.21 | 37.37 | 33.26 | 30.34 |
| DeepSeek-VL2-Small | Lu 等人 (2024) | 47.62 | 37.00 | 50.38 | 26.91 |
| 专有模型 | | | | | |
| GPT-4o | OpenAI (2024) | 38.81 | 32.65 | 40.17 | 29.16 |
| GPT-4.1-mini | OpenAI （2025） | 45.62 | 37.84 | 47.22 | 34.56 |
| Claude-4-Sonnet | Anthropic (2025) | 44.75 | **48.42** | 44.21 | 47.62 |
| 专用空间模型 | | | | | |
| RoboBrain | Ji 等 (2025) | 37.38 | 35.80 | 38.28 | 29.53 |
| SpaceMantis | Chen 等 (2024a) | 22.81 | 37.65 | 21.26 | 29.32 |
| Spatial-MLLM | Wu 等 (2025) | 32.06 | 38.39 | 20.92 | 32.82 |
| Space-Qwen | Chen 等 (2024a) | 33.28 | 38.02 | 33.71 | 26.32 |
| VLM-3R | Fan 等 （2025） | 42.09 | 36.73 | 44.22 | 24.45 |
| pySpatial （本文方法） | - | **58.56** | 43.20 | **60.54** | **48.10** |

表 2：**在 MindCube-1k（Yin 等人，2025）数据集上的性能比较**。评估的心理模型（Yin 等人，2025）基于 Qwen2.5-VL-3B-Instruct（Bai 等，2025）。VADAR（含重建）表示我们使用自己的三维重建模块实现了 VADAR。最佳结果以**粗体**标出，次优结果以下划线标出。

| 方法 | 参考 | 总体 | 旋转 | 之间 | 周围 |
|---|---|---|---|---|---|
| 基线模型 | | | | | |
| Qwen2.5-VL-3B-Instruct | Bai 等 (2025) | 37.81 | 34.00 | 36.00 | 45.20 |
| GPT-4o | OpenAI (2024) | 42.29 | 35.00 | 43.00 | 46.40 |
| 空间心智模型 | | | | | |
| 思维链 | Yin 等人(2025) | 40.48 | 32.00 | 36.00 | 58.00 |
| 视图插值 | | 37.81 | 35.50 | 36.50 | 42.80 |
| 认知地图 | | 41.43 | 37.00 | 41.67 | 44.40 |
| 视觉编程方法 | | | | | |
| ViperGPT | Surís 等人 (2023) | 36.95 | 20.50 | 41.00 | 40.40 |
| VADAR | Marsili 等 (2025) | 40.76 | 33.50 | 40.67 | 46.80 |
| VADAR（含重建） | - | 35.62 | 31.00 | 36.83 | 36.40 |
| pySpatial **(我们的方法)** | - | **62.35** ± 1.18 | **41.83** ± 2.34 | **64.89** ± 2.60 | **72.67** ± 3.30 |

我们使用 Open3D（Zhou 等，2018）来渲染新视图。所有实验均在单个 NVIDIA A6000 Ada GPU 上进行。我们在附录 pySpatial 中提供了 pySpatial 的完整实现细节以及所用提示词，附录B 和C中亦有说明。代码将在论文被接收后公开。

### 4.2 定量结果

**在 MindCube 上的结果**。我们首先对 pySpatial 在具有挑战性的 MindCube 基准上进行全面评估，以严格评估其在多视角空间推理中的有效性。表 1 总结了与基线方法对比的结果。总体而言，pySpatial在所有类别的基线上均取得了明显的性能优势。具体而言，其总体准确率达到 58.56%，比最佳开放权重模型 DeepSeek-VL2-Small 高出 10.94%，并比最强专有模型 GPT-4.1-mini 高出 12.94%。在之间类别上，该类别要求对中心物体与所有周围物体的关系进行推理，pySpatial 达到 60.54%，大幅优于所有基线，且这些基线均未超过 50%。值得注意的是，pySpatial的表现也优于 VLM-3R（Fan 等人，2025），该方法利用

表3：**在 OMNI3D-BENCH上的性能比较。**遵循 VADAR（Marsili 等，2025），我们报告数值型（Yang 等，2025）类别的平均相对准确率，以及其他类别的准确率。最佳结果以**粗体**显示，次佳结果带下划线。

| 方法 | 参考 | 数值型（ct） | 数值型（其他） | y/n | 多项选择 | 总计 |
|---|---|---|---|---|---|---|
| 基线模型 | | | | | | |
| GPT-4o | OpenAI (2024) | **28.1** | 35.5 | 66.7 | 57.2 | 42.9 |
| Claude3.5-Sonnet | Anthropic (2024) | 22.4 | 20.6 | 62.2 | 50.6 | 32.2 |
| Llama-3.2 | Meta (2024) | 24.3 | 19.3 | 47.5 | 27.4 | 25.6 |
| Gemini1.5-Pro | Google (2024) | 25.2 | 28.1 | 46.2 | 37.6 | 32.0 |
| SpaceMantis | Chen 等 (2024a) | 20.0 | 21.7 | 50.6 | 48.2 | 30.3 |
| 视觉编程方法 | | | | | | |
| ViperGPT | Surís 等人(2023) | 20.0 | 15.4 | 56.0 | 42.4 | 26.7 |
| VisProg | Gupta 和 Kembhavi(2023) | 2.9 | 0.9 | 54.7 | 25.9 | 13.5 |
| VADAR | Marsili 等（2025） | 21.7 | 35.5 | 56.0 | **57.6** | 40.4 |
| pySpatial（我们的方法） | - | 22.9 | **38.6** | **72.0** | 54.7 | **44.2** |

CUT3R（Wang 等人，2025b）作为三维编码器，并在合成空间推理数据上进行了微调，高出 16.5%，尽管完全在零样本设置下运行。这些结果表明，pySpatial 在 MINDCUBE 的多样化任务类别上具有良好的泛化能力。通过将空间推理显式分解为模块化工具调用，我们的方法比开放权重和专有多模态大语言模型（包括专门用于空间推理的模型）提供了更强的归纳偏置。

**在 MINDCUBE-1k上的结果。**表 2比较了`pySpatial` 与基于隐式心理建模的方法（Yin 等人，2025）（例如，思维链推理、认知地图）以及以往视觉编程智能体（例如，ViperGPT、VADAR）在 MINDCUBE-1k 上的表现。我们有以下关键观察：（1）空间心智模型（Yin 等人，2025），依赖多模态大语言模型的隐式想象机制进行空间推理，因此性能提升有限；相比之下，`pySpatial` 的表现比每种方法都高出约 20%；（2）我们的 `pySpatial` 显著优于现有视觉编程方法，例如，性能比 VADAR 高出 21.9%。值得注意的是，`pySpatial` 也优于 VADAR（含重建）；我们使用自己的三维重建模块重新实现了 VADAR。该结果表明，即使配备三维信息，VADAR 的自适应 API 设计在三维空间推理方面仍不可靠且缺乏鲁棒性。这些结果验证了`pySpatial` 相较于现有基线的卓越效果，展现了通过启用显式三维推理来实现多视图空间推理的优势。

**在 OMNI三维-BENCH上的结果。**我们进一步在近期的单视图空间推理基准 OMNI3D-BENCH 上评估 pySpatial，结果表明，我们的框架能够有效泛化至单视图场景，并在各任务类别中带来一致的提升。表3 展示了 OMNI三维-BENCH上的结果；我们遵循 VADAR 的评估协议（Marsili 等，2025）：对数值型（其他）子任务报告平均相对准确率（MRA），其余类别则采用标准准确率。pySpatial 优于以往的视觉编程方法，成绩较 VADAR 提高 3.8%，较 ViperGPT 提高 17.5%，并在 OMNI3D-BENCH 上创下新的整体最先进水平。值得注意的是，pySpatial 的总分也超过了 GPT-4o，这凸显了我们的视觉编程框架甚至优于先进的专有多模态大语言模型。该结果彰显了 pySpatial 广泛的泛化能力：即使在几何线索不那么明显的单视图场景中，通过代码智能体显式调用三维函数，仍能提升空间推理能力。

### 4.3 定性结果

为了进一步展示我们的`pySpatial` 框架的能力，我们在 MINDCUBE 基准测试中的代表性示例上进行了定性实验。如图2所示，每个查询都配有生成的三维视觉程序、重建的三维场景、程序输出，以及由`pySpatial`生成的最终回答。这些示例凸显了`pySpatial`如何使多模态大语言模型能够在由稀疏二维输入重建、可供探索的三维场景中进行显式推理。通过合成可执行且易于解释的视觉程序，执行相机平移、旋转和新视角合成等操作，该框架提供了易于解释的输出，使推理过程以几何证据为依据。在各种空间推理任务中，`pySpatial`生成的回答与真实标注高度一致，彰显了我们方法的有效性。值得注意的是，生成的三维视觉程序




图 2： **来自 MindCube 的四个代表性示例的定性结果。我们展示了pySpatial能够在重建的可探索三维场景中进行显式推理，并通过可解释且可执行的三维视觉程序有效处理多种空间推理任务。图 A1进一步说明，pySpatial 能够结合带有控制流结构的可执行三维视觉程序（例如，*for*循环），从而稳健地处理广泛的空间推理任务。放大查看效果更佳。**

包含结构良好的注释，以记录 pySpatial 的推理过程，从而提供透明性和可解释性，使研究人员能够轻松验证、调试或修改。

### 4.4 真实世界机器人导航

为测试仅使用多模态大语言模型进行真实世界部署的潜力，我们在一间 50 $m^2$ 的两室实验室中使用配备速度跟踪控制器的四足机器人。在该设置中，多模态大语言模型生成高级位置指令，这些指令经过人工转换为时间速度目标，由控制器进行跟踪，从而使机器人能够从初始位姿导航至目标物体（一个蘑菇玩具）。基于有限的二维视图，pySpatial 重建可探索的三维场景，通过视觉编程推断相机位姿，并生成供机器人执行的结构化运动规划。

如图 3所示，我们的 pySpatial 成功引导机器人穿过门口、正确转弯，并最终抵达正确的目标位置。值得注意的是，多模态大语言模型基线 GPT-4.1 难以判断左侧与右侧等相对方向，也无法提供绝对度量距离估计，因而导致导航错误。相比之下，我们的智能体能够输出精确的旋转和平移，与现实世界中的执行相吻合，从而可靠地完成任务。这项实验表明，我们的方法不仅能在问答基准测试中生成连贯的空间推理，还能有效迁移至实体机器人平台，以执行复杂的室内导航任务。




图3：**真实世界机器人导航的定性结果**。我们将 pySpatial 部署在 Unitree-Go1 机器人上，以有限视图作为输入，导航至目标物体（蘑菇玩具）。与 GPT-4.1 基线相比，后者因错误的初始转向而失败，并产生易发生碰撞的轨迹；pySpatial 则生成几何一致的规划，成功到达目标。

### 4.5 讨论

**代码智能体的消融研究**。为评估代码智能体的作用，我们在 MindCube-1k 基准测试上进行实验，比较各种多模态大语言模型基线在集成和未集成 `pySpatial` 时的表现。如桌子 4 所示，使用`pySpatial` 后，所有受测多模态大语言模型的表现均持续显著提升，包括 GPT-4o、GPT-4.1-mini 和 GPT-4.1。例如，GPT-4o 的准确率从 42.3% 提升至 62.7%，总体准确率表明，pySpatial 能够泛化至不同的多模态大语言模型，并有效增强空间推理能力。

表4：**代码智能体的消融研究**。我们报告了在 MINDCUBE-1k 数据集上的准确率。

| 方法 | 整体旋转 | | 之间 | 周围 |
|---|---|---|---|---|
| GPT-4o | 42.29 | 35.00 | 43.00 | 46.40 |
| + pySpatial | 62.67 | 41.00 | 65.00 | 66.33 |
| GPT-4.1-mini | 43.34 | 36.00 | 45.00 | 44.80 |
| + pySpatial | 58.19 | 37.50 | 62.00 | 65.60 |
| GPT-4.1 | 44.67 | 35.50 | 45.33 | 50.40 |
| + pySpatial | 62.35 | 41.83 | 64.89 | 72.67 |

**失败案例分析**。我们从 MindCube 基准测试中选取约 100 个具有代表性的样本，并进行人工分析，以确定最终回答错误时潜在的错误来源。如图 4所示，在 39% 的失败案例中，只有 6% 是由于生成的视觉程序不正确、无法回应查询所致，这验证了整体编程流程的有效性。此外，20% 的错误源于多模态大语言模型在最终推理步骤中的表现，而 13% 则源于三维重建模型的局限性：生成的视觉程序虽然正确，但 program 的输出未能提供有用信息。这些结果也表明，三维重建和代码生成模型的进步有望进一步提升我们的表现。

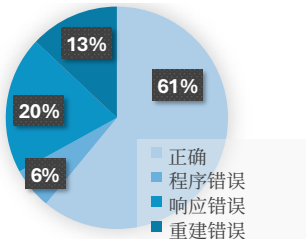


图4：**失败案例研究**。我们人工检查了 MINDCUBE 中约 100 个样本的错误来源。

**效率说明**。我们的`pySpatial` 在单线程、单个图形处理器上，用时 2.17 小时完成了 MindCube-1k 基准测试，处理 1,050 个查询，平均每个查询耗时 7.45 秒。具体而言，代码生成需要 2.41 秒，程序执行需要 2.14 秒，答案生成需要 2.90 秒。相比之下，VADAR（Marsili 等，2025）平均每个查询需要 17.25 秒。这些结果表明，我们的视觉编程框架在保持部署效率、无需过高成本的同时，提升了多模态大语言模型的空间推理能力。

## 5 结论

本文介绍了 pySpatial，这是一种通过零样本 Python 代码生成增强多模态大语言模型空间推理能力的视觉编程框架。通过组合三维重建和新视角合成等函数，pySpatial 将二维图像序列转换为可探索的三维场景，从而支持在三维空间中进行显式推理。在 MINDCUBE 和 OMNI3D-BENCH 基准测试上的实验表明，pySpatial 始终优于强大的多模态大语言模型基线；与 GPT-4.1-mini 相比，其在 MINDCUBE 上的性能提升最高达到 12.94%。除基准测试外，真实世界的室内导航实验进一步验证了其实用性：机器人能够利用 pySpatial 生成的路线规划成功穿越复杂环境。

## 致谢

本研究部分经费由美国陆军研究实验室（ARL）项目 W911NF-23-2-0007 和 W911QX-24-F-0049、美国国防高级研究计划局（DARPA）项目 FA8750-23-2-1015，以及美国海军研究办公室（ONR）项目 N00014-23-1-2840 资助。作者感谢岳翔博士在项目期间提供的宝贵讨论。

## 可复现性声明

我们致力于确保研究结果的可复现性。本研究使用的所有空间工具均已开源，所评估的基准数据集也均已公开。我们在第 4 节和附录中详细介绍了实验设置和实现细节，以便于复现。代码已公开发布于 https://github.com/Zhanpeng1202/pySpatial。

## 参考文献

Jean-Baptiste Alayrac、Jeff Donahue、Pauline Luc、Antoine Miech、Iain Barr、Yana Hasson、Karel Lenc、Arthur Mensch、Katherine Millican、Malcolm Reynolds 等。Flamingo：一种用于少样本学习的视觉语言模型。 神经信息处理系统进展，35:23716– 23736，2022。 3

Anthropic。Claude3.5sonnet，2024年10月。网址 https://www.anthropic.com/news/claude-3-5-sonnet。 7

Anthropic。Claude 4 Sonnet 系统卡，2025年5月。网址 https://www-cdn.anthropic.com/6be99a52cb68eb70eb9572b4cafad13df32ed995.pdf。 6

白帅、KeqinChen、XuejingLiu、王佳林、葛文斌、宋思博、党凯、王鹏、王世杰、唐军等。Qwen2.5-vl 技术报告。 *arXiv* 预印本 *arXiv:2502.13923*，2025。 5，6

Davide Bucciarelli、Nicholas Moratelli、Marcella Cornia、Lorenzo Baraldi 和 Rita Cucchiara。面向图像描述的多模态大语言模型个性化：一项实验分析。载于欧洲计算机视觉会议，第 351–368 页。施普林格，2024 年。2

Chun-Peng Chang、Alain Pagani 和 Didier Stricker。多模态大语言模型中的 3D 空间理解：消歧与评估。载于*IEEE*国际机器人与自动化会议，第 13537–13544 页。IEEE，2025 年。2

David Charatan、Sizhe Lester Li、Andrea Tagliasacchi 和 Vincent Sitzmann。pixelSplat：利用图像对生成三维高斯点泼溅，实现可扩展的通用三维重建。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 19457–19467 页，2024 年。3

Boyuan Chen、Zhuo Xu、Sean Kirmani、Brain Ichter、Dorsa Sadigh、Leonidas Guibas 和 Fei Xia。SpatialVLM：赋予视觉语言模型空间推理能力。在 *IEEE/CVF*计算机视觉与模式识别会议论文集中，第 14455–14465 页，2024a。 2，3、5、6、7

Zhe Chen、Jiannan Wu、Wenhai Wang、Weijie Su、Guo Chen、Sen Xing、Muyan Zhong、Qinglong Zhang、Xizhou Zhu、Lewei Lu 等。InternVL：扩展视觉基础模型并对齐

通用视觉-语言任务。载于《IEEE/CVF计算机视觉与模式识别会议论文集》，第24185–24198页，2024年b。6 An-Chieh Cheng、Hongxu Yin、Yang Fu、Qiushan Guo、Ruihan Yang、Jan Kautz、Xiaolong Wang 和 Sifei Liu。SpatialRGPT：视觉语言模型中的接地空间推理。《神经信息处理系统进展》，37:135062–135093，2024。2、3 Jianguo Duan、Liwen Zhuang、Qinglei Zhang、Ying Zhou 和 Jiyun Qin。制造业中的多模态感知-融合-控制与人机协作：综述。《国际先进制造技术期刊》，132(3):1071–1093，2024。2 Zhiwen Fan、Jian Zhang、Renjie Li、Junge Zhang、Runjin Chen、Hezhen Hu、Kevin Wang、Huaizhi Qu、Dilin Wang、Zhicheng Yan 等。VLM-3R：借助指令对齐的三维重建增强视觉语言模型。arXiv预印本 arXiv:2505.20279，2025。5、6 Chaoyou Fu、Yuhan Dai、Yongdong Luo、Lei Li、Shuhuai Ren、Renrui Zhang、Zihan Wang、Chenyu Zhou、Yunhang Shen、Mengdan Zhang 等。Video-MME：首个用于视频分析中多模态大语言模型的综合评测基准。载于《计算机视觉与模式识别会议论文集》，第24108–24118页，2025。Google。我们的下一代模型：Gemini 1.5，2024年2月。网址：https://blog.google/technology/ai/google-gemini-next-generation-model-february-2024/。7 Tanmay Gupta 和 Aniruddha Kembhavi。视觉编程：无需训练的组合式视觉推理。载于《IEEE/CVF计算机视觉与模式识别会议论文集》，第14953–14962页，2023。2、3、5、7 Chenguang Huang、Oier Mees、Andy Zeng 和 Wolfram Burgard。用于机器人导航的视觉语言地图。载于《IEEE国际机器人与自动化会议》，第10608–10615页。IEEE，2023。3 Jiangyong Huang、Silong Yong、Xiaojian Ma、Xiongkun Linghu、Puhao Li、Yan Wang、Qing Li、Song-Chun Zhu、Baoxiong Jia 和 Siyuan Huang。三维世界中的具身通用智能体。载于《国际机器学习会议》，第20413–20451页。PMLR，2024。3 Wenlong Huang、Pieter Abbeel、Deepak Pathak 和 Igor Mordatch。作为零样本规划器的语言模型：为具身智能体提取可执行知识。载于《国际机器学习会议》，第9118–9147页。PMLR，2022。3 Yuheng Ji、Huajie Tan、Jiayu Shi、Xiaoshuai Hao、Yuan Zhang、Hengyuan Zhang、Pengwei Wang、Mengdi Zhao、Yao Mu、Pengju An 等。RoboBrain：从抽象到具体的机器人操作统一大脑模型。载于《计算机视觉与模式识别会议论文集》，第1724–1734页，2025。6 Amita Kamath、Jack Hessel 和 Kai-Wei Chang。视觉语言模型究竟“上”哪儿了？探究其在空间推理方面的困境。载于《自然语言处理经验方法会议论文集》，第9161–9175页，2023。3 Sahar Kazemzadeh、Vicente Ordonez、Mark Matten 和 Tamara Berg。ReferItGame：在自然场景照片中指代物体。载于《自然语言处理经验方法会议论文集》，第787–798页，2014。2 Vincent Leroy、Yohann Cabon 和 Jérôme Revaud。在三维空间中利用MASt3R实现图像匹配接地。载于《欧洲计算机视觉会议》，第71–91页。施普林格，2024。3 Bo Li、Yuanhan Zhang、Dong Guo、Renrui Zhang、Feng Li、Hao Zhang、Kaichen Zhang、Peiyuan Zhang、Yanwei Li、Ziwei Liu 和 Chunyuan Li。LLaVA-OneVision：轻松实现视觉任务迁移。《机器学习研究汇刊》，2025。网址：https://openreview.net/forum? id=zKv8qULV6n。5、6 Xiaoqi Li、Mingxu Zhang、Yiran Geng、Haoran Geng、Yuxing Long、Yan Shen、Renrui Zhang、Jiaming Liu 和 Hao Dong。ManipLLM：面向以物体为中心的机器人操作的具身多模态大语言模型。载于《IEEE/CVF计算机视觉与模式识别会议论文集》，第18061–18070页，2024。2

Chenguo Lin、Yuchen Lin、Panwang Pan、Yifan Yu、Honglei Yan、Katerina Fragkiadaki 和 Yadong Mu。Movies：一秒内实现运动感知四维动态视图合成。arXiv 预印本 arXiv:2507.10065，2025。3 Fangyu Liu、Guy Emerson 和 Nigel Collier。视觉空间推理。《计算语言学协会学报》，11：635–651，2023。2 Shilong Liu、Zhaoyang Zeng、Tianhe Ren、Feng Li、Hao Zhang、Jie Yang、Qing Jiang、Chunyuan Li、Jianwei Yang、Hang Su 等。Grounding DINO：将 DINO 与基于定位的预训练相结合，实现开放集目标检测。载于《欧洲计算机视觉会议》，第 38–55 页。施普林格，2024。3 Haoyu Lu、Wen Liu、Bo Zhang、Bingxuan Wang、Kai Dong、Bo Liu、Jingxiang Sun、Tongzheng Ren、Zhuoshu Li、Hao Yang 等。DeepSeek-VL：迈向真实世界的视觉语言理解。arXiv 预印本 arXiv:2403.05525，2024。6 Zhanpeng Luo、Haoxi Ran 和 Li Lu。Instant4D：数分钟内实现四维高斯泼溅。载于《神经信息处理系统进展》，2025。URL https://openreview.net/forum?id=HTLa6Ao0jG。3 Wufei Ma、Yu-Cheng Chou、Qihao Liu、Xingrui Wang、Celso M de Melo、Jianwen Xie 和 Alan Yuille。SpatialReasoner：迈向显式且可泛化的三维空间推理。载于《神经信息处理系统进展》，2025。URL https://openreview.net/forum?id=hFaXVjRFHI。2 Xiaojian Ma、Silong Yong、Zilong Zheng、Qing Li、Yitao Liang、Song-Chun Zhu 和 Siyuan Huang。SQA3D：三维场景中的情境问答。载于《国际学习表征会议》，2023。URL https://openreview.net/forum?id=IDJx97BC38。3 Dominic Rosario Maggio、Hyungtae Lim 和 Luca Carlone。VGGT-SLAM：在 SL(4)流形上优化的稠密 RGB 同步定位与建图。载于《神经信息处理系统进展》，2025。URL https://openreview.net/forum?id=63ljkCGMhE。3 Damiano Marsili、Rohun Agrawal、Yisong Yue 和 Georgia Gkioxari。借助动态应用程序接口进行空间推理的视觉智能体 AI。载于《计算机视觉与模式识别会议论文集》，第 19446–19455 页，2025。3、5、6、7、9 Minesh Mathew、Dimosthenis Karatzas 和 CV Jawahar。DocVQA：用于文档图像视觉问答的数据集。载于《IEEE/CVF冬季计算机视觉应用会议》，第 2200–2209 页，2021。2 Meta。Llama 3.2：利用开放且可定制的模型革新边缘 AI 与视觉，2024 年 9 月。URL https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/。7 OpenAI。你好，GPT-4o，2024 年 8 月。URL https://openai.com/index/hello-gpt-4o/。6、7 OpenAI。在应用程序接口中推出 GPT-4.1，2025 年 4 月。URL https://openai.com/index/ gpt-4-1/。6 Yanyuan Qiao、Haodong Hong、Wenqi Lyu、Dong An、Siqi Zhang、Yutong Xie、Xinyu Wang 和 Qi Wu。NavBench：探测多模态大语言模型的具身导航能力。载于《神经信息处理系统进展》，2025。URL https://openreview.net/ forum?id=nf8PKQKtl2。2、3 Nikhila Ravi、Valentin Gabeur、Yuan-Ting Hu、Ronghang Hu、Chaitanya Ryali、Tengyu Ma、Haitham Khedr、Roman Rädle、Chloe Rolland、Laura Gustafson、Eric Mintun、Junting Pan、Kalyan Vasudev Alwala、Nicolas Carion、Chao-Yuan Wu、Ross Girshick、Piotr Dollar 和 Christoph Feichtenhofer。SAM 2：对图像和视频进行 Segment Anything。载于《国际学习表征会议》，2025。URL https://openreview.net/forum?id=Ha6RTeWMd0。3 Johannes L Schonberger 和 Jan-Michael Frahm。运动恢复结构再探。载于《IEEE计算机视觉与模式识别会议论文集》，第 4104–4113 页，2016。3

Yongliang Shen、Kaitao Song、Xu Tan、Dongsheng Li、Weiming Lu 和 Yueting Zhuang。Hugging-GPT：利用 Hugging Face 中的 ChatGPT 及其伙伴解决人工智能任务。神经信息处理系统进展，36：38154–38180，2023。3

Mohit Shridhar、Lucas Manuelli 和 Dieter Fox。Perceiver-Actor：一种用于机器人操作的多任务 Transformer。载于 机器人学习会议，第 785–799 页。PMLR，2023 年。 3

Chan Hee Song、Valts Blukis、Jonathan Tremblay、Stephen Tyree、Yu Su 和 Stan Birchfield。Ro-bospatial：为机器人学中的二维和三维视觉语言模型教授空间理解。载于 计算机视觉与模式识别会议论文集，第 15768–15780 页，2025 年。 2

Dídac Surís、Sachit Menon 和 Carl Vondrick。Vipergpt：通过 Python 执行进行视觉推理以实现推理。载于*IEEE/CVF* 国际计算机视觉会议论文集，第 11888–11898 页，2023 年。 2、3、4、5、6，7

Peter Tong、Ellis Brown、Penghao Wu、Sanghyun Woo、Adithya Jairam Vedagiri IYER、Sai Charitha Akula、Shusheng Yang、Jihan Yang、Manoj Middepogu、Ziteng Wang 等。Cambrian-1：以视觉为中心，对多模态大语言模型进行全面开放式探索。神经信息处理系统进展，37:87310–87356，2024。3

Zifu Wan、Ce Zhang、Silong Yong、Martin Q Ma、Simon Stepputtis、Louis-Philippe Morency、Deva Ramanan、Katia Sycara 和 Yaqi Xie。Only：单层干预即可充分缓解大型视觉语言模型中的幻觉。*arXiv* 预印本 *arXiv:2507.00898*，2025。3

Jianyuan Wang、Minghao Chen、Nikita Karaev、Andrea Vedaldi、Christian Rupprecht 和 David Novotny。VGGT：视觉几何基础 Transformer。载于计算机视觉与模式识别会议论文集，第 5294–5306 页，2025a。2，3，4，5

Jiayu Wang, Yifei Ming, Zhenmei Shi, Vibhav Vineet, Xin Wang, Sharon Li 和 Neel Joshi。图片是否胜过千言万语？深入探究视觉语言模型的空间推理。神经信息处理系统进展，37:75392–75421，2024a。3

Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros 和 Angjoo Kanazawa。具有持久状态的连续三维感知模型。载于计算机视觉与模式识别会议论文集，第 10510–10522 页，2025b。3，4，5，7

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii 和 Jerome Revaud。Dust3r：让几何三维视觉变得简单。载于*IEEE/CVF*计算机视觉与模式识别会议论文集，第 20697–20709 页，2024年b。2，3

Chenfei Wu、Shengming Yin、Weizhen Qi、Xiaodong Wang、Zecheng Tang 和 Nan Duan。Visual ChatGPT：借助视觉基础模型进行对话、绘图和编辑。 *arXiv* 预印本 *arXiv:2303.04671*，2023。 3

Diankun Wu、Fangfu Liu、Yi-Hsin Hung 和 Yueqi Duan。Spatial-MLLM：提升多模态大语言模型在基于视觉的空间智能方面的能力。见神经信息处理系统进展，2025。网址：https://openreview.net/forum?id=RnXS7aK4rK。 2，6

Jihan Yang、Shusheng Yang、Anjali W Gupta、Rilyn Han、李飞飞和 Saining Xie。空间思维：多模态大语言模型如何观察、记住并回忆空间。见 计算机视觉与模式识别会议论文集，第 10632–10643 页，2025。 2，7

Zhengyuan Yang、Linjie Li、Jianfeng Wang、Kevin Lin、Ehsan Azarnasab、Faisal Ahmed、Zicheng Liu、Ce Liu、Michael Zeng 和 Lijuan Wang。Mm-react：通过提示 ChatGPT 实现多模态推理与行动。 *arXiv* 预印本 *arXiv:2303.11381*，2023。 3Jiabo Ye、Haiyang Xu、Haowei Liu、Anwen Hu、Ming Yan、Qi Qian、Ji Zhang、Fei Huang 和 Jingren Zhou。mPLUG-owl3：迈向多模态大语言模型中的长图像序列理解。发表于 国际学习表征会议，2025。网址 https://openreview.net/forum?id=pr37sbuhVa。 6

Baiqiao Yin、Qineng Wang、Pingyue Zhang、Jianshu Zhang、Kangrui Wang、Zihan Wang、Jieyu Zhang、Keshigeyan Chandrasegaran、Han Liu、Ranjay Krishna 等。基于有限视图的空间心理建模。*arXiv* 预印本 *arXiv:2506.21458*，2025。1，2，3，5，6，7

Weihao Yu、Zhengyuan Yang、Linjie Li、Jianfeng Wang、Kevin Lin、Zicheng Liu、Xinchao Wang 和 Lijuan Wang。Mm-vet：评估大型多模态模型的综合能力。载于国际机器学习会议，第 57730–57754 页。PMLR，2024。3

Xiangyu Zeng、Kunchang Li、Chenting Wang、Xinhao Li、Tianxiang Jiang、Ziang Yan、Songze Li、Yansong Shi、Zhengrong Yue、Yi Wang、Yali Wang、Yu Qiao 和 Limin Wang。Time-suite：通过基于依据的微调提升多模态大语言模型的长视频理解能力。载于 国际学习表征会议，2025。网址 https://openreview.net/forum?id=nAVejJURqZ。2

Ce Zhang、Simon Stepputtis、Katia Sycara 和 Yaqi Xie。双原型演化：实现视觉语言模型的测试时泛化。神经信息处理系统进展，37：32111–32136，2024。3

Ce Zhang、Simon Stepputtis、Katia Sycara 和 Yaqi Xie。通过负学习增强视觉语言少样本适配。载于 *IEEE/CVF*冬季计算机视觉应用会议，第 5905–5915 页，2025a。2

Ce Zhang、Zifu Wan、Zhehan Kan、Martin Q. Ma、Simon Stepputtis、Deva Ramanan、Russ Salakhutdinov、Louis-Philippe Morency、Katia P. Sycara 和 Yaqi Xie。通过生成式反馈进行自我纠正解码，以减轻大型视觉语言模型中的幻觉。载于 国际学习表征会议，2025b。网址 https://openreview.net/forum?id=tTBXePRKSx。2

Ce Zhang、Kaixin Ma、Tianqing Fang、Wenhao Yu、Hongming Zhang、Zhisong Zhang、Haitao Mi 和 Dong Yu。VScan：重新思考视觉词元缩减，以提高大型视觉语言模型的效率。机器学习研究汇刊，2026。网址 https://openreview.net/forum?id=KZYhyilFnt。3

Yuanhan Zhang、Jinming Wu、Wei Li、Bo Li、Zejun MA、Ziwei Liu 和 Chunyuan Li。LLaVA-video：使用合成数据进行视频指令微调。《机器学习研究汇刊》，2025c。网址：https://openreview.net/forum?id=EElFGvt39K。6

Qian-Yi Zhou、Jaesik Park 和 Vladlen Koltun。Open3D：用于三维数据处理的现代库。*arXiv* 预印本 *arXiv:1801.09847*，2018。6

# 通过程序生成实现显式三维空间推理

## 附录

在附录中，我们提供了更多实验结果以及 pySpatial 框架的实现细节。附录组织如下：

- 章节A 提供了更多实验结果。
- 章节B 展示了 `pySpatial` 的应用程序接口规范。
- 章节 C 介绍了我们智能体的提示词实现。
- 第 D 节披露了大语言模型的使用情况。

更多实验结果

### A.1 MMSI-Bench 上的结果

我们进一步在 MMSI-Bench 上评估了我们的方法，并在表格 A1 中报告结果。我们观察到，pySpatial 将 MMSI-Bench 的总体表现平均提升了 6.4%，进一步证明了我们方法的有效性。

表格 A1：MMSI-Bench 上的评估结果。我们的 pySpatial 基于 GPT-4o。

| 模型 | 位置关系 | | | | | | 属性 | | 运动 | | MSR | Avg. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 摄像机–摄像机 | 物体–物体 | 区域–区域 | 摄像机–物体 | 物体–区域 | 摄像机–区域 | 测量 | 近似值 | Cam. | Obj. | – | |
| 专有 | | | | | | | | | | | | |
| GPT-5 | 43.0 | 35.1 | 32.1 | 48.8 | 42.4 | 51.8 | 60.9 | **36.4** | 32.4 | 36.8 | **42.0** | **41.9** |
| o3 | **45.2** | **39.4** | 37.0 | 44.2 | 47.1 | **62.6** | 54.7 | 28.8 | 31.1 | 32.9 | 34.9 | 41.0 |
| GPT-4.5 | 34.4 | 29.8 | **39.5** | **51.2** | 47.1 | 55.4 | 39.1 | 33.3 | **41.9** | **40.8** | 36.4 | 40.3 |
| GPT-4.1 | 36.6 | 26.6 | 27.2 | 29.1 | 36.5 | 27.7 | 37.5 | 24.2 | 36.5 | 32.9 | 28.8 | 30.9 |
| GPT-4o | 34.4 | 24.5 | 23.5 | 19.8 | 37.6 | 27.7 | 32.8 | 31.8 | 35.1 | 36.8 | 30.8 | 30.3 |
| Gemini-2.5-Pro | 39.7 | 31.9 | **39.5** | 45.3 | 35.2 | 43.3 | 51.5 | 21.2 | 36.4 | 30.2 | 34.3 | 36.9 |
| 开源 | | | | | | | | | | | | |
| InternVL3-78B | 34.4 | 23.4 | 32.1 | 12.8 | 37.6 | 26.5 | 37.5 | 19.7 | **28.4** | 31.6 | 29.3 | 28.5 |
| InternVL2.5-78B | 23.7 | 22.3 | **39.5** | 29.1 | 31.8 | **42.2** | 35.9 | 19.7 | 17.6 | 26.3 | 27.3 | 28.5 |
| Qwen2.5-VL-72B | 25.8 | **34.0** | 34.6 | 23.3 | 34.1 | 36.1 | **45.3** | 27.3 | 27.0 | 30.3 | 27.3 | **30.7** |
| LLaVA-OneVision-72B | **43.0** | 31.9 | 33.3 | 30.2 | 37.6 | 38.6 | 28.1 | 19.7 | 13.5 | 32.9 | 15.7 | 28.4 |
| 基线 | | | | | | | | | | | | |
| GPT-4o | 34.4 | 24.5 | 23.5 | 19.8 | 37.6 | 27.7 | 32.8 | 31.8 | 35.1 | 36.8 | 30.8 | 30.3 |
| +pySpatial（我们的方法） | 51.6 | 28.7 | 27.2 | 20.9 | 41.2 | 38.6 | 46.9 | 39.4 | 46.0 | 38.2 | 36.4 | 37.3 |

### A.2 更多消融实验

**代码智能体**。在表A2中，我们比较了`pySpatial`框架与不同代码智能体搭配时的性能。在所有类别中，无论用于代码生成的底层大语言模型是什么，`pySpatial` 的表现始终优于基础 GPT-4o 模型。在评估的智能体中，GPT-4o 的总体表现最强，准确率达到 62.67%；Qwen3-Coder 和 DeepSeek-v3 的结果相近，分别为 62.10% 和 61.05%。值得注意的是，Qwen3-Coder 在周围类别中表现最佳，而 GPT-4o 在所有任务类型上带来了最均衡的提升。这些发现表明，我们的框架对代码智能体的选择具有鲁棒性，且大部分性能提升源自三维视觉编程范式，而非所使用的特定代码大语言模型。

**三维重建骨干**。在表 A3中，我们比较了不同三维重建骨干对 `pySpatial` 框架性能的影响。包括 VGGT、Pi3 和 CUT3R 在内的三种骨干均显著优于 GPT-4o 基线，表明我们的三维视觉编程范式不受重建方法选择的影响。其中，Pi3 的总体表现最佳（63.33%），在 旋转 和 之间类别中均取得了显著提升。VGGT 的结果同样出色，而 CUT3R 的表现略逊一筹，但仍显著优于基础模型。这些发现表明，`pySpatial`能够有效利用多种现代重建骨干，其空间推理能力的提升并不依赖于特定的重建架构。

表 A2：我们的 pySpatial 框架使用不同代码智能体的性能比较 s.

| 方法 | 总体 | 旋转 | 之间 | 周围 |
|---|---|---|---|---|
| GPT-4o | 42.29 | 35.00 | 43.00 | 46.20 |
| pySpatial 使用 GPT-4o | 62.67 | 41.00 | 66.33 | 71.20 |
| pySpatial 使用 Qwen3-Coder | 62.10 | 40.00 | 64.50 | 74.00 |
| pySpatial 使用 DeepSeek-v3 | 61.05 | 40.50 | 65.33 | 64.80 |

Ta表 A3：使用不同三维重建骨干的 pySpatial 性能比较。

| 方法 | 总体 | 旋转 | 之间 | 周围 |
|---|---|---|---|---|
| GPT-4o | 42.29 | 35.00 | 43.00 | 46.20 |
| 使用 VGGT 的 pySpatial | 62.67 | 41.00 | 66.33 | 71.20 |
| 使用 Pi3 的 pySpatial | 63.33 | 43.50 | 65.66 | 72.33 |
| 使用 CUT3R 的 pySpatial | 61.05 | 40.50 | 64.33 | 69.60 |

表 A4：pySpatial 在不同数量上下文学习示例下的比较。

| 方法 | 总体 | 旋转 | 之间 | 周围 |
|---|---|---|---|---|
| GPT-4o | 42.29 | 35.00 | 43.00 | 46.20 |
| pySpatial（0 个示例） | 53.62 | 32.50 | 55.16 | 66.80 |
| pySpatial（2 个示例） | 62.67 | 41.00 | 66.33 | 71.20 |
| pySpatial（含 4 个示例） | 63.14 | 46.00 | 64.50 | 73.60 |

**代码智能体的上下文学习示例**。在表 A4中，我们考察了上下文学习示例的数量如何影响 `pySpatial` 框架的性能。我们观察到明显的上升趋势：即使只提供少量示例，也能显著提升所有类别的表现。使用 0 个示例时，性能就已大幅优于基础 GPT-4o 模型（53.62% 对 42.29%），这表明 `pySpatial` 即使没有示范引导也能有效运行。增加到 2 个示例后，性能进一步显著提升，总体准确率达到 62.67%。增加到 4 个示例时，性能达到最佳（63.14%），其中 旋转和 周围 类别的提升尤为显著。这些结果表明，`pySpatial` 能够从更多示例中受益，但即使只有极少量的上下文监督，也足以释放出强大的空间推理能力。

### A.3 补充定性结果

我们在图INDCUBE 中展示了更多定性结果A1。除了图2所示的线性程序流程外，`pySpatial` 还能够执行更具表现力的控制流操作，包括 for循环、条件语句和 lambda 风格函数组合，从而构建复杂的多步三维操作。这些更丰富的程序结构使`pySpatial`能够迭代探索候选视点、有选择地应用几何谓词，并根据依赖场景的约束调整推理方式。在所展示的示例中，我们的代码智能体生成的三维视觉程序始终能够引导`pySpatial` 给出与真实空间关系高度吻合的回答，进一步验证了我们的三维推理框架的正确性、可解释性和稳健性。

### A.4 补充真实场景实验

我们还在一个具有挑战性的动态场景中补充了一个定性示例，其中不同视图是在不同时间拍摄的，且有人在环境中移动，如图A2所示。尽管存在时间不一致性和动态元素，`pySpatial`仍然表现出色：重建模块能够稳健地整合多视角观测，并保留准确空间推理所需的稳定结构线索。相比之下，GPT-5 在这一场景中仍然表现不佳，无法生成安全或正确的导航轨迹，且经常给出与底层场景几何不符的指令。这些扩展的真实场景实验进一步验证了我们方法的稳健性和实际有效性。




图 A1：**更多来自 MindCube 的定性示例。我们展示了 pySpatial 如何使多模态大语言模型能够在重建的、可探索的三维场景中进行显式推理，让模型不仅能够理解空间结构，还能组合带有控制流的可执行三维视觉程序（例如 for 循环），从而稳健地解决各种空间推理任务。**




图 A2：具有挑战性的动态场景中的另一个真实世界定性示例。

## B I用于pySpatial的实现

API接口的完整说明请参阅代码2。

```
导入 os
导入 glob
从 typing 导入 List, Union
来自pathlib 导入Path
来自tool.recontruct 导入reconstruct_3d
来自tool.segment 导入segment_图像, 分割_自动
来自tool.estimate_深度导入estimate_深度
来自tool.camera_理解导入analyze_相机_轨迹
来自tool.novel_视角_合成导入novel_视角_合成, 向右旋转_, 向左旋转_,
    move_向前 , move_向后 , turn_周围
导入 正则表达式


类 重建:
    def __初始化__(自身, 点_云, 外参, 内参, 颜色 =无):
        自身. 点_云 = 点_云
        self. 外参 = 外参 # 4×4 NumPy 数组列表
        self. 内参 = 内参
        self. 颜色 = 颜色 # 添加颜色属性


类 场景:
    """保存图像数据的简单场景类。"""

    def __初始化__(self, 路径_到_图像: 联合 [字符串, 列表 [字符串]], 问题: 字符串 = "",
    场景_标识符: 字符串 = 无):
        self. 问题 = 问题
        self. 场景_标识符 = 场景 _id
        self. original_path = path_to_图像 # 为重建保存原始路径
        self. 图像 =self. _load_图像 ( path_to_图像 )
        self. reconstruction : 重建 = None
        self.代码: str = 无
        self.visual_clue = None

    def _加载_图像（self, 路径_到_图像: Union [str , 列表 [字符串 ]]) -> 列表 [str ]:
        """从目录或列表加载图像路径。"""
        如果 isinstance (path_到_图像, 字符串):
            如果 os.path.isdir( path_到_图像 ):
                # 检查这是否为演示目录 (其中包含 .glb 文件)
                演示_路径 =路径( 路径_到_图像 )
                glb_文件 =列表（演示_路径.glob("*. glb"))

                if glb_文件:
                    # 这是一个演示目录, 从 color/ 子目录加载图像
                    颜色_目录 = 演示_路径 /"颜色"
                    if 颜色_目录.exists():
                        图像 扩展名 = ['*. png', '*. jpg', '*. JPEG ']
                        图像 = []
                        对于 扩展名 在 图像_扩展名中:
                            图像 . 扩展 (glob.glob(os.path.join(str( 颜色_目录 ), ext)))
                        返回 sorted( 图像)
                    否则:
                        打印(f"警告: 检测到演示目录 但没有颜色/
    子目录 在 {path_到_图像 }")
                        返回 []
                否则:
                    # 普通目录, 从目录加载所有图像
                    图像_扩展名 = ['*. png', '*. jpg', '*. JPEG ']
                    图像 = []
                    对于扩展名在图像_扩展名:
                        图像 . extend (glob.glob(os.path.join( path_to_images , ext)))
                    返回排序后的( 图像 )
            否则:
                # 单个图像文件
                返回 [路径_至_图像 ]
        否则:
            # 图像路径列表
            返回(路径_指向_图像)


类 pySpatial:
    """ 简单 三维视觉工具的接口。"""
    # 我们目前禁用其他函数

    @staticmethod
```

```
def 重建（场景：Scene）：
    """根据场景图像进行三维重建。"""

    # 检查这是否为演示目录                    ( 包含          .glb 文件)
    如果是实例(场景.original_路径,                 字符串) 并且 os.path.isdir(scene.original_路径 ):
        演示_路径  =  Path(scene. original_路径 )
        glb_文件  =列表( 演示_路径.glob("*. glb"))

        if  glb_文件:
            # 这是一个演示目录，传入目录路径以加载演示数据
            结果  =重建_3的 (场景. 原始_路径 ,                    场景_标识 =场景. 场景_标识 )
        否则:
            # 使用图像路径进行常规重建
            结果  =重建_3的 (场景.图像 ,                    场景_标识 =场景. 场景_标识 )
    否则:
        # 使用图像路径进行常规重建
        结果  =重建_3了（场景.图像,                    场景_标识 =场景。场景_标识）

    # 将原始结果字典转换为重建对象
    点_云  =结果.get('点 ',                    无)
    相机  =result.get('相机 ',                 无)
    颜色  =result.get('颜色 ',                 无)      # 从结果中获取颜色

    # 将点云转换为 NumPy                    如果它是张量
    if  点_云 不为 None:
        如果 hasattr(点_云,                 'cpu'):    # PyTorch 张量
            点_云  =点_云 .cpu ().NumPy ()
        否则，如果具有(点_云,                 'NumPy'):     # 其他张量类型
            点_云  =点_云 .NumPy ()

    # 将颜色转换为 NumPy                 如果它是一个张量
    if  颜色 不为 None:
        如果 hasattr(颜色 ,          'CPU'):     # PyTorch 张量
            颜色  =颜色 .cpu ().numpy ()
        elif hasattr(颜色,                 'NumPy'):     # 其他张量类型
            颜色  =颜色 .NumPy ()

    # 如果可用，从相机中提取外参和内参
    外参  =None
    内参  =None

    如果相机不为None:
        # 假设 cameras 包含外参矩阵
        外参  =cameras.cpu ().numpy () 如果 hasattr(cameras ,                    'cpu')  否则cameras

    # 创建并返回包含颜色的重建对象
    重建  =重建（点_云,                                    外参,          内参,          颜色）

    # 存储原始结果以供调试
    重建。                _raw_结果  =结果

    返回 重建

@静态方法
def 描述_相机_运动（重建：Reconstructio                    n):
    """根据重建结果描述相机运动。
    参数:
    """
    外参  =  重建。外参
    返回 分析_相机_轨迹（外参）

@静态方法
定义 合成_新_视图（重建：重建 , 新_相机_位姿 , 宽度 =512, 高度=512 , 输出_路径 =无):

    """ 根据重建结果生成新视角合成                                                .
    参数:
        重建：包含以下内容的重建对象                点_云,          外参,          内参
        新_相机_位姿：新视点的 3x4 或 4x4 外参矩阵
        宽度：输出图像宽度（默认：512）
        高度：输出图像高度（默认：512）
        out_path：输出图像路径（默认值：None,                    如果未
已提供 )
    返回值:
        str 或图像：如果提供了 out_path，则为渲染图像的路径，否则为
图像对象
    """
    返回新颖_视图_合成（重建,                    新_相机_位姿,          宽度,      高度,      输出_路径）


@静态方法
定义 旋转_右侧(外参，角度 =无):
```

```
        """ 将相机位姿向右旋转"""
            返回 旋转_右侧（外参,                    角度）

    @staticmethod
    def 旋转_左侧（外参,                angle =None):
        """ 将相机位姿向左侧旋转"""
            返回 旋转_左侧（外参,                    角度)

    @静态方法
    定义 移动_向前（外参                ,  距离 =None):
        """向前移动相机位姿 """
            返回 移动_向前 (外参                    ,  距离 )

    @静态方法
    定义 移动_向后 (外参                ,  距离 =None):
        """向后移动相机位姿 """
            返回 移动_向后 (外参                    ,  距离 )

    @staticmethod
    定义 旋转_周围 ( 外参 ):
        """将相机位姿旋转 180 度 """
        返回 旋转_周围（外参）


类    智能体:
    def __初始化__(自身,    应用程序接口_密钥：字符串 = 无）：
        self. api_key    = api_key 或 os.getenv('OPENAI_应用程序接口_KEY ')

    def generate_代码 (self ,    场景·    场景):
        来自 智能体. codeAgent .查询 导入 生成_代码_来自_查询
        返回生成_代码_自_查询（场景,                    self. 应用程序接口_密钥）

    def parse_大语言模型_响应（self,    场景：Scene,    响应：字符串）：
        " " "
        提取第一个                Python 代码块（` ` `python                ...  ` ` `）中的文本。
        将代码作为字符串返回，或者                    如果未找到，则返回 ""。
        " " "
        来自 智能体。codeAgent 。执行 导入 解析_大语言模型_响应
        代码  =  解析_大语言模型_响应（响应）
        场景.代码  =  代码
        返回代码

    def 执行（self    ,  场景·    场景）：
        " " "
        执行包含场景的代码字符串，并返回视觉线索结果。
        " " "
        # try:
        #       from agent. codeAgent .execute import execute_代码
        #       program  =  execute_代码 (scene.code)

        #       visual_线索  =  program (scene)
        #       返回 visual    _clue
        # except Exception as e:
        #       import traceback
        #       错误_详情  =f" 执行失败：{str(e)}\ n回溯信息：{ traceback .
    format_exc ()}"
        #       # 存储错误以便进行详细报告
        #       self. last    _执行    _错误  =错误    _详情
        #       return f"代码生成期间发生错误，未提供视觉线索。
      错误：    {str(e)}"

        来自 智能体。codeAgent . 执行 导入 执行_代码
        program  =  执行_代码（场景.代码）

        视觉_线索  =  program（场景）
        返回 视觉    _clue

    def 答案（自身    ,  场景：场景    ,  视觉_线索 ):
        # 根据视觉线索回答问题
        从智能体.anwer 导入 答案

        # 在场景中设置视觉线索
        场景  .visual_clue  =  视觉  l_clue

        # 使用 API 密钥调用答案函数
        返回 答案（场景,                    self. 应用程序接口_密钥 )
```

代码 2：**完整** `pySpatial` **API规范**。

## C实现D细节A智能体P提示词pySpatial

我们在下方方框中展示了实验中使用的提示词。

**A智能体P提示词pySpatial**

```
任务_描述 = """
    现在请你解决一个与空间推理相关的问题。
    输入是图像和一个自然语言问题，
    该问题专门用于测试你的空间推理能力。
    直接将这些任务作为视觉任务来解决并非易事
    语言模型，直接解决这些任务并不简单。不过，你可以使用以下 Python API:
"""

api_规范 = """
    在 pySpatial API 中，我们明确引入了三维归纳偏置。
    我们提供一个场景类，其中包含图像和一个问题。
    此外，我们还提供一个三维重建过程，可用于
    生成三维点云和相机参数。

    类 Reconstruction:
        def __init__(self, point_cloud, 外参, 内参):
            self.point_云 = 点_云
            self.extrinsics = 外参
            self.intrinsics = 内参


    类 场景:
        "保存图像数据的简单场景类。"
        def __初始化__(self, 路径_到_图像: 联合[字符串, 列表[字符串]],
                                                    问题: 字符串 = ''):
            自身的问题 =问题
            自身图像 =自身._加载_图像（路径_到_图像）
            自身重建 : 重建 = 无

        def _加载_图像（自身，路径_到_图像：联合[字符串，列表[字符串]]）
                                                        -> 列表[字符串]:
            “从目录或列表中加载图像路径。”
            if isinstance(path_to_图像, str):
                if os.path.isdir(path_to_images):
                    # 从目录加载所有图像
                    图像_扩展名 = ['*.png',          '*.jpg',     '*.jpeg']
                    图像 = []
                    for ext in image_extensions:
                        图像.extend(glob.glob(os.path.join(
                                                path_to_图像, ext)))
                    return sorted(图像)
                else:
                    # 单个图像文件
                    返回 [路径_到_图像]
            否则:
                # 图像路径列表
                返回 list(路径_到_图像)

    类 pySpatial:
        "三维视觉工具的简单界面。"
        # 我们暂时禁用其他功能
```

```
        @staticmethod
        def reconstruct(scene: Scene):
            "从场景图像进行三维重建。"

            return reconstruct_3d(scene.images)

        @staticmethod
        def describe_相机_运动(recon: 重建):
            "根据重建结果描述相机运动。
            参数:
            "
            外参 =recon.extrinsics
            return describe_相机_运动(extrinsics)

        @静态方法
        def 合成_新_视角(recon: 重建,
                                        新_相机_位姿):
            "根据重建结果生成新视角合成。
            参数:
            "
            return 新_视角_合成(recon)

        # 用于操作相机位姿的方法
        定义旋转_右侧(extrinsic, 角度=np.pi/2):

        定义旋转_左侧(extrinsic, 角度=np.pi/2):

        定义向前移动_(extrinsic , 距离=0.1):

        定义向后移动_(extrinsic , 距离=0.1):

        def turn_周围(外参):


        @staticmethod
        def estimate_深度(图像):
            return estimate_深度(图像)
"""

# 上下文学习示例
示例_问题 ="""
    问题 1:
    问题: “根据这两个显示同一场景的视图:
    从第一个视图到
    第二个视图?
    A. 斜向前方和左侧
    B. 直接向右侧
    C. 正左侧
    D. 斜向前方和右侧"

    如何解决这个问题?
    步骤 1: 我们可以利用相机外参轻松找到答案。
    步骤 2: 因此, 我们应该先重建场景,
    然后利用相机外参找到答案。
    步骤 3: 仍然难以直接从
    外参矩阵中得到答案。
    步骤 4: 我们可以使用 pySpatial.describe_相机_运动
```

以获得答案。
接下来，在提供的 pySpatial API 中编写 Python 代码，
然后由智能体自动收集代码
我编写并执行了它。

```Python
定义 program(input_场景: 场景):
    重建三维 = pySpatial.reconstruct(input_场景)
    相机_运动 =pySpatial.describe_相机_运动(
                重建三维)
    返回相机 –运动
```
步骤 5：从执行中获取视觉线索后，我就能轻松匹配答案：

问题 2：
根据这四张图像（图像 1、2、3 和 4）
从不同视点（正面、左侧、背面，展示粉色瓶子，
和右侧），每台相机都与房间墙壁对齐，并且部分
拍摄到了周围环境：如果我站在同一个位置，并且
朝向与图像 1 中所示相同，然后我向右侧转身
并向前移动，我会离粉色毛绒玩具更近吗
还有床头板吗？

由于我们无法比较三维空间中的距离，
我们可以渲染两张图像，并将它们用作视觉线索。
```python
def program(input_场景: Scene):

    reconstructed_场景 = pySpatial.reconstruct(input_场景)
    base_视点 =reconstructed_场景.外参[0]
    # 图像 1 表示数组中的第 0 个索引

    视点_turn_右侧 = pySpatial.rotate_右侧
                        (base_视点)
    视点_向_前方移动 =pySpatial.move_向前
                        (视点_向_右侧转)

    图像_右侧 =pySpatial.synthesize_新颖_视图
                (重建的_场景，视点_向_右侧转)
    图像_前方 =pySpatial.synthesize_新颖_视图
                (重建的_场景，视点_向_前方移动)

    # 我们应该比较这两幅图像，检查物体
      是否存在，以及距离是否更近。
    视觉_线索 = [图像_右侧，图像_前方]
    返回视觉信息 _clue
```
"""

代码_生成_提示 = f"""
现在请使用 pySpatial API 编写一个 Python 函数
来解决问题。
请注意，你可以先进行推理，然后编写代码。
但代码应放在 ```python ``` 块中。
编写一个简洁的代码块

```
        此外，编写的函数应命名为 program
        并且输入参数应为 Scene对象。
        例如，
        ``python
        def program(input_场景: Scene):
            ...
            返回.....
        ```
        尽量在代码中添加简单的注释，解释你的逻辑。

        确保先进行推理，说明我们为什么要这样编写 program，
        因为我们有一个 pySpatial API，可以让我们探索三维
        空间，请先像这样进行推理（我想知道某物的
        右侧是什么，因此我只需渲染一个新视角
        由此）。
"""

# ReAct 提示模板：ReAct：语言模型中的推理与行动协同
https://arxiv.org/abs/2210.03629

答案_背景 = f"""
        我们现在正在解决一个空间推理问题。
        作为视觉语言模型，直接解决这些任务并非易事
        模型。
        然而，我们可以使用以下 PySpatial API：
        {api_specification}

        我们根据 PySpatial API 生成 Python 代码来解决
        这个问题。
"""

答案_提示 = """
        根据代码和执行中的视觉线索                    ，答案
        问题。
"""

# 不含视觉线索的答案提示
不含_视觉_线索_背景 = """
        根据问题解决这个空间推理问题
        以及图像输入。

        首先，分析问题，从中提取有用信息
        问题描述，然后尝试根据
        有用的视觉信息作答。

        如果找不到最佳答案，请尽可能给出你的最佳猜测。
"""
```

## D 大语言模型的使用

我们仅将大语言模型用作辅助工具，以润色本手稿的文字。它们用于改善语法、清晰度和可读性，但未参与构思、数据分析、实验设计或结果解读。
```