

# Paper Research: Scene reconstruction and decomposition

---

## Paper Research: Scene reconstruction and decomposition

1. Introduction
  - 1.1 Background
  - 1.2 Research Overview
2. Some details about selected papers
  - 2.1 3d gaussian splatting for real-time radiance field rendering
    - 2.1.1 Motivation
    - 2.1.2 Methods
      - 2.1.2.1 SfM 点云初始化:
      - 2.1.2.2 3D 高斯椭球集初始化:
      - 2.1.2.3 3D 椭球参数投影:
      - 2.1.2.4 基于光栅化的图像渲染:
      - 2.1.2.5 损失计算
      - 2.1.2.6 自适应密度控制
    - 2.1.3 Advantages and Disadvantages
    - 2.1.4 Personal Overview and some ideas
    - 2.1.5 My experiment
  - 2.2 Lerp: Language embedded radiance fields
    - 2.2.1 Motivation
    - 2.2.2 Methods
      - 2.2.2.1 体渲染
      - 2.2.2.2 多尺度监督
      - 2.2.2.3 Lerp的查询
    - 2.2.3 Advantages and Disadvantages
    - 2.2.4 My Experiment
  - 2.3 Langsplat: 3d language gaussian splatting
    - 2.3.1 Motivation
    - 2.3.2 Methods
      - 2.3.2.1 3D语言场的挑战
      - 2.3.2.2 使用SAM学习分层语义
      - 2.3.2.3 3D语言高斯
      - 2.3.2.4 开放语义查询任务
    - 2.3.3 Advantages and Disadvantages
    - 2.3.4 My Experiment
  - 2.4 Fastlgs: Speeding up language embedded gaussians with feature grid mapping
    - 2.4.1 Motivation
    - 2.4.2 Methods
      - 2.4.2.1 初始化
      - 2.4.2.2 语义网格网络
      - 2.4.2.3 映射策略
      - 2.4.2.4 查询和高斯训练
    - 2.4.3 Advantages and Disadvantages
  - 2.5 Vggt: Visual geometry grounded transformer
    - 2.5.1 Motivation
    - 2.5.2 Methods

- 2.5.2.1 问题定义
  - 2.5.2.2 主干网络 (用于处理输入图像并提取特征的Transformer网络)
  - 2.5.2.3 Prediction heads 预测头
  - 2.5.2.4 训练损失函数
  - 2.5.3 Experiment Result
  - 2.5.4 My Test
  - 2.5.5 Advantages and Disadvantages
  - 2.5.6 Presonal Overview and some ideas
3. Conclusion

# 1. Introduction

## 1.1 Background

先从科研热点的角度去分析这个词：**3D Scene reconstruction and decomposition**

先放一张计算机视觉顶会CVPR2025录用文章的相关的一些关键词图

关键词	翻译	频率
Multimodal	多模态	175
Diffusion Model	扩散模型	153
Large Language Model (LLM)	大语言模型	129
Gaussian Splatting	高斯泼溅	95
Transformer	Transformer	93
Vision-language Model	视觉-语言模型	65
Object Detection	目标检测	54
Video Generation	视频生成	54
Image Generation	图像生成	50
Text-to-Image	文生图	48
Mamba	Mamba	43
Self-Supervised	自监督	41
Semantic Segmentation	语义分割	37
Super-Resolution	超分辨率	35
Unsupervised	无监督	35
Semi-Supervised	半监督	31
CLIP	CLIP	29
Pose Estimation	姿态估计	28
Image Edit	图像编辑	23
3D Reconstruction	三维重建	21

图1：CVPR2025关键词图

结合这张图，我觉得三维场景重建与分解是一个非常有意义的调研方向，有如下原因：

1. **高斯溅射作为第四高频词，3D Gaussian Splatting(3DGS)**作为能全面取代 因为神经渲染方式带来高昂内存时间成本的 **NeRF**,是当前学术界的热点之一，3DGS是三维重建目前最火的方法，大家都在基于3DGS进行修改，实现更高效更低成本更精确的三维重建。(Tips:当然**VGST**作为CVPR Best Paper是否开启了下一个风口还不为所知)。

2. **三维重建**，自己作为**最后一个高频词**，也有21次的频率，是把一个从各种角度拍摄的一组二维图像通过处理得到一个可以实时交互的三维场景，本质实现了模态的一个转换，和**第一大关键词多模态**有密切关系。现在更多的可能考虑结合时间等维度，进行三维向四维的拓展。

3.三维重建的分解这个词，我自己理解为基于三维重建的语义分割等具体任务，**语义分割有37次的频率**，所以和比如将按照语义或者结构一个完整的3D场景拆分成多个不同的组成部分或元素：在一个室内场景中，将房间分解为墙壁、地板、天花板、家具、装饰品等不同的部分，或者将一个建筑物分解为基础、墙体、屋顶等结构部分，基本方法是借助**CLIP**与很多的**视觉语言模型**，或者借助**多模态大模型**，又是三个热点。

4.三维重建进行分解后，可以投入进一步的下游应用中，**图像生成，扩散模型**等关键词，可以进一步应用到分解后的场景中，实现了三维场景的改变，实现进一步的下游应用。

做科研绝对需要对热点的及时把握，特别是在CV这个比较卷的领域，你的idea想出来就必须很快去做，不然别人比你做出来了，你的工作就没有那么有原创性了，追求有开创性的论文一定是每一个科研工作者的目标！

---

学术化一点的文字开始正文：

三维重建指的是给定一个场景的多个视角图像，重建出这个场景的3维模型，是机器人感知模块中的关键技术之一。三维重建使机器人能够更好地感知、理解所处的环境并与之交互，在机器人领域的应用非常广泛。

传统的三维重建方法发展比较成熟，通常分为4个步骤：**运动结构恢复（SfM）**、**多视立体视觉（MVS）**、**表面重建以及纹理渲染**。

但传统重建方法在面对复杂场景时效率低且精度差，而**神经辐射场（NeRF）**的出现，给3维重建领域带来了革新和生命力。NeRF是一种利用稀疏视角图像重建场景的三维表示的**隐式学习**技术，通过对场景的光照和颜色信息进行编码，能够生成新的视角图像，实现高质量的3维重建和视角合成效果。

但是使用深度神经网络来建模一个连续体场景，对场景中的细节进行充分还原，实现逼真的视觉效果，需要付出巨大代价。NeRF存在很多问题需要改进。

在此背景下，**三维高斯溅射（3DGS）**技术应运而生。3DGS技术改变了NeRF技术的底层范式，彻底颠覆了NeRF的整个架构，从场景表示和渲染方式上作出了革新。3DGS技术一般使用数以万计的3D高斯基元来紧凑地表达某个场景，然后通过分块并行光栅化的方式实现高效的图像渲染。这种紧凑表达和高效渲染使得3DGS技术变得非常实用。在不影响重建质量和视觉效果的前提下，3DGS**实现了实时渲染**。另外，3DGS这种显式场景表达方式**提高了场景的可编辑性**，在三维重建和新视图合成领域达到了新的高度，基于3DGS的改进技术和下游应用也层出不穷。

## 1.2 Research Overview

由于NeRF与3DGS两大经典开山文章已经烂熟于心。这里将三维重建与分解理解为三维重建的语义分割。

我们挑选了关于三维重建和三维重建的语义分割的五篇文章，每一篇分别对应了我前面说的几个关键词，从这几篇文章的逐步变迁，也可以找到未来三维重建与分解领域的未来工作方向。

[1]Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4), 139-1.(三维重建最火也最经典的论文)

[2]Kerr, J., Kim, C. M., Goldberg, K., Kanazawa, A., & Tancik, M. (2023). Lerp: Language embedded radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 19729-19739).(首次引入预训练模型CLIP实现三维空间中的开放语义查询)

[3]Qin, M., Li, W., Zhou, J., Wang, H., & Pfister, H. (2024). Langsplat: 3d language gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 20051-20060).(由NeRF过渡到3DGS，开创用3DGS实现三维空间中的开放语义查询)

[4]Ji, Y., Zhu, H., Tang, J., Liu, W., Zhang, Z., Tan, X., & Xie, Y. (2025, April). Fastlgs: Speeding up language embedded gaussians with feature grid mapping. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Vol. 39, No. 4, pp. 3922-3930).(相对于LangSplat进一步追逐的工作，后续工作拓展到了时间维度，即4D)

[5]Wang, J., Chen, M., Karaev, N., Vedaldi, A., Rupprecht, C., & Novotny, D. (2025). Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference* (pp. 5294-5306).(基于Transform的新架构，开创新蓝海？)

## 2. Some details about selected papers

### 2.1 3d gaussian splatting for real-time radiance field rendering

#### 2.1.1 Motivation

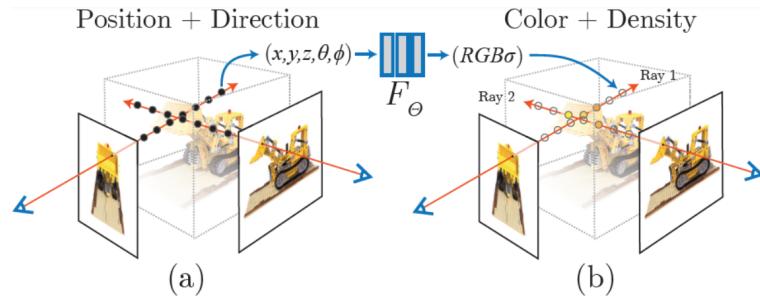


图2：NeRF核心方法架构图

这里简单介绍一下 NeRF，NeRF 是一种典型的隐式辐射场技术，主要使用隐式的基于坐标的模型将空间坐标映射到像素值，核心思想是用一个多层次感知机（MLP）来表示一个连续的场景，**输入为空间点的3D 坐标和观察方向，输出是空间点的颜色和密度信息。** NeRF 的渲染过程是基于**体渲染 (volume rendering)** 的，它将每条从相机发出的射线上的点的颜色和密度进行加权求和，得到该射线的最终颜色。NeRF 的训练过程是通过最小化渲染结果和真实图像之间的L2 损失来优化MLP 的参数。

很多follow NeRF 的工作，对 NeRF 进行改进的工作奔涌而出，而 NeRF 本质是使用神经网络表示场景，存在两个本质缺点：

**1.神经网络的训练需要高昂的渲染资源(算力，炼丹)和训练时间成本，无法实现实时渲染**

**2.作为隐式辐射场表达方式，很难对场景进行编辑，在下游应用严重受限**

而最近更快的方法不可避免地会牺牲速度来换取质量。

为了本质上解决NeRF存在的问题，“不破不立”，必须需要**底层范式和架构的创新**，在此背景下，3DGS应运而生。3DGS 技术从**场景表示和渲染方式**上作出了革新，实现了**紧凑的表达与高效的渲染**，可以实现**低成本的实时渲染**，也提高了**场景的可编辑性**。

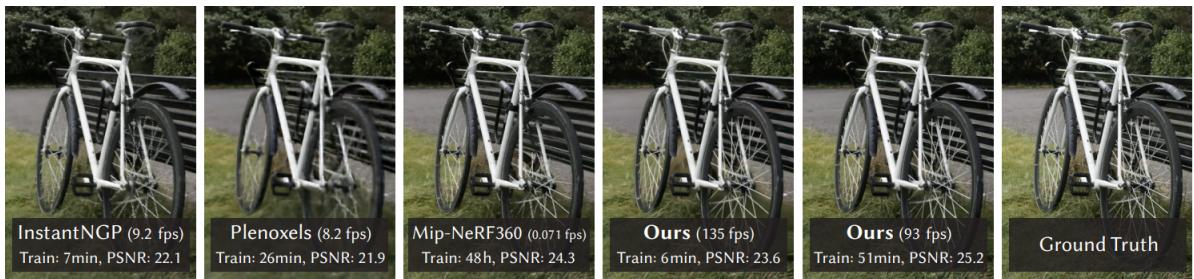


图3：3DGs效果对比图

### 2.1.2 Methods

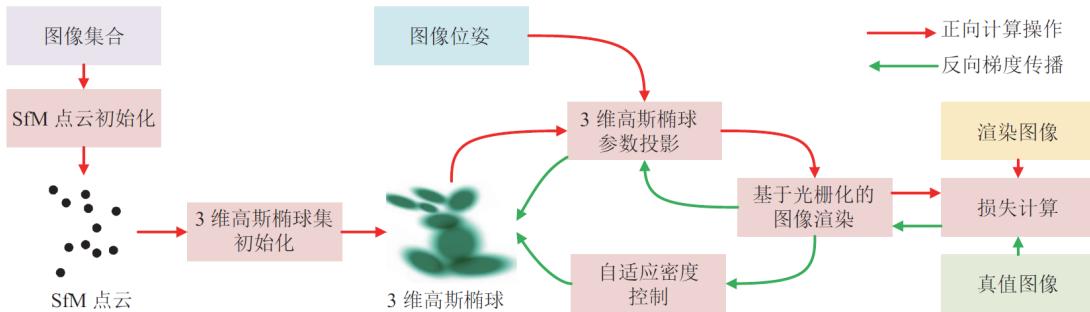


图4：3DGs技术整体框架

3DGs主要分为**正向误差计算**和**反向梯度传播**2个数据流，包含6个模块：**SfM 点云初始化、3D 高斯椭球集初始化、3D 椭球参数投影、光栅化图像渲染、损失计算以及自适应密度控制**，整体流程就是正向计算和反向优化的过程，接下来将详细介绍这6个模块。

#### 2.1.2.1 SfM 点云初始化：

在获取到场景的一组2D图像后，使用SfM技术从这一组2D图像中恢复出场景的稀疏3D点云。这里通过常用的COLMAP库来实现。

#### 2.1.2.2 3D 高斯椭球集初始化：

在SfM点云初始化步骤中得到的每个点云都将被初始化为一个**3D 高斯椭球**，具体来说，对于每个点云，用一个包含位置信息和形状信息的3D高斯函数来生成3D高斯椭球：

$$G(\mathbf{x}) = e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})}$$

位置信息通过3D高斯椭球的中心点即均值向量 $\boldsymbol{\mu}$ 表达。形状信息（包含旋转和缩放）通过3D高斯椭球的协方差矩阵 $\boldsymbol{\Sigma}$ 表达：

$$\boldsymbol{\Sigma} = \mathbf{R} \mathbf{S} \mathbf{S}^T \mathbf{R}^T$$

其他离散属性使用连续的球谐函数进行表示，球谐函数本质上是一组能表达球面上不同位置值的基本函数，阶数越高，表达能力越强。对于3D高斯椭球的不透明度信息，直接使用可学习优化的点云不透明度 $\alpha$ 表示。

### 2.1.2.3 3D 椭球参数投影：

得到3D 高斯椭球后，需要将其投影到2维图像平面进行后续渲染操作，根据计算机图形学中的投影关系，利用以下公式将空间中的3D 高斯椭球投影到图像平面上：

$$\Sigma' = JW \Sigma W^T J^T$$

其中，W 是视图变换矩阵（世界坐标系到相机坐标系的变换），J 是投影变换的仿射近似的雅可比矩阵。Σ' 是投影在图像平面后的2D 高斯椭球的协方差矩阵。

### 2.1.2.4 基于光栅化的图像渲染：

3D GS 使用基于有序点叠加的渲染方法，即通过点云中一定半径范围内能影响的有序点集合N 来计算一个像素的颜色C：

$$C = \sum_{i \in N} c_i \alpha'_i \prod_{j=1}^{i-1} (1 - \alpha'_j)$$

其中， $c_i$  表示点颜色， $\alpha'_i$  表示最终不透明度，用原始不透明度和高斯函数的乘积计算：

$$\alpha'_i = \alpha_i e^{-\frac{1}{2}(\mathbf{x}'_i - \mathbf{u}'_i)^T (\Sigma')^{-1} (\mathbf{x}'_i - \mathbf{u}'_i)}$$

公式中累乘项表示当前点受排序后前面点不透明度的影响权重， $\alpha'_j$  越大，不透明度越大， $1 - \alpha'_j$  越小，权重越小，当前点在颜色渲染时起到的作用越小，反之，作用越大。

具体的操作方法为：首先，将要渲染的整个图像分为 $16 \times 16$  大小的图像块，从每个图像块视锥内挑选可视的置信度

大于一定阈值的3D 高斯基元，并按3D 高斯基元到图像平面的深度值进行排序，为每个图像块单独开一个线程。

然后并行地在每个图像块上从近到远对3D 高斯基元进行溅射操作，对溅射留下的痕迹进行堆叠累积，然后作光栅化处理，从堆叠的溅射痕迹中划分像素网格来生成像素值，每个图像块一旦有像素的不透明度达到饱和，就停止对应线程。

### 2.1.2.5 损失计算

3DGS 算法对渲染图像和真实图像之间误差的计算如下：

$$L = (1 - \lambda)L_1 + \lambda L_{SSIM}$$

$L_1$  表示图像中M个像素的绝对值灰度误差，

$L_{SSIM}$  表示图像之间的结构相似度，包含亮度、对比度以及结构3个指标上的差异。

$$L_1 = \frac{1}{M} \sum_{i=1}^M |\mathbf{I}(i) - \hat{\mathbf{I}}(i)|$$

$$L_{SSIM} = \frac{(2\mu_I \mu_{\hat{I}} + C_1)(\sigma_{I\hat{I}} + C_2)}{(\mu_I^2 + \mu_{\hat{I}}^2 + C_1)(\sigma_I^2 + \sigma_{\hat{I}}^2 + C_2)}$$

### 2.1.2.6 自适应密度控制

计算完图像损失之后，梯度会沿着数据流反向传播，对3D 高斯基元的参数进行优化，而重建不充分的区域往往有着更大的梯度，是需要优化的重点区域，这里将重建不充分的情况划分为2种：

第1 种情况：**欠重建 (under reconstruction)**，指的是现有3D 高斯基元不足以覆盖需要重建的区域，

要对已有3D 高斯基元进行克隆操作。

第2 种情况：**过重建 (over reconstruction)**，指的是现有的3D 高斯基元虽然能够覆盖所建区域，

但是不够细化，要对已有3D 高斯基元进行分割操作。

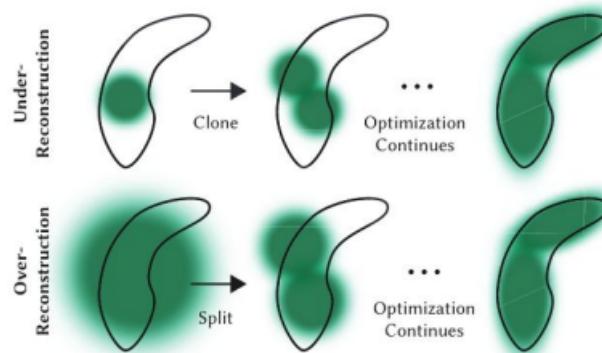


图5：欠重建与过重建的示意图

### 2.1.3 Advantages and Disadvantages

Advantages:

- **高效的重建与渲染效率**：通过显式的点云表达方式以及高度并行化的可微分光栅化管线，3DGS 技术在优化速度、渲染速度方面表现出色，能够实现快速且高质量的三维重建，可达到毫秒级渲染，支持实时交互应用和下游应用，如虚拟现实、增强现实、游戏等场景。
- **高细节保真度与动态适应性**：高斯点的各向异性协方差矩阵可精确模拟复杂几何形状，如曲面、透明物体等，以及各种光照效果，如反射、折射等，能够很好地捕捉场景中的细节和动态变化，适用于处理复杂的、不规则的物体，如云、烟雾、流体等，还可应对动态场景的挑战，通过自监督学习分离静态与动态元素，提升场景重建的时空一致性。
- **数据与计算效率高**：仅需少量多视角图像即可生成高保真模型，相较于一些传统方法，数据获取成本和处理成本较低，并且其存储空间比神经辐射场 NeRF 等方法减少 50% 以上，还支持 GPU 实时解压。
- **模型轻量化与实时性**：该技术可以生成影视级真实感的三维模型，同时在轻量化方面展现出极强的潜力，使得模型更易于在各种设备上实时展示和交互，特别适合对实时性要求较高的应用，如移动设备上的 AR/VR 应用等

Disadvantages:

- **点云数据固有缺陷的影响**：作为输入源之一的 3D 扫描设备所获取的数据往往具有稀疏性、无序性和不规则性等特点，这些问题会增加后续分析与处理的难度，可能会影响 3DGS 的重建效果和性能发挥。
- **参数优化难度大**：优化每个高斯点的参数，如协方差、强度等，可能是一项具有挑战性的任务，尤其对于复杂场景中有许多物体的情况下，需要大量的计算资源和时间来确保参数的准确性和合理性。

- **与现有渲染管线的兼容性问题**：3DGS 与现有的传统渲染管线不太兼容，可能需要对现有的图形渲染系统和流程进行较大的改动和适配，这给其在一些已有的应用系统中的集成和推广带来了一定的困难。
- **在精确几何建模方面的局限性**：3DGS 的重建效果在表现精确几何形状的场景中可能不如传统的基于网格的表示方法，因此不太适合对精确几何形状要求较高的场景，如建筑、道路等硬表面物体的建模

#### 2.1.4 Presonal Overview and some ideas

从目前来看，3DGS已经完全取代了NeRF的结构，能实现实时渲染，且易于应用到下游任务中。

那么把前面NeRF所做的所有任务，迁移到3DGS的框架中，就是一片学术蓝海。

下面介绍的两篇，首先是首次实现开放词汇三维语义分割的Lerf，下面就是使用3DGS框架实现的LangSplat

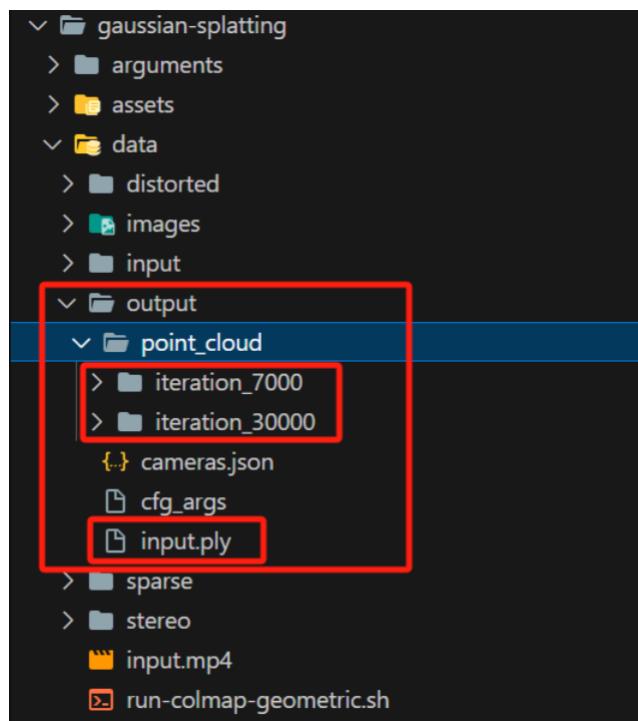
我感觉也印证了我前面说的话。

读完这篇文章，觉得可以做的idea的方向：**把前面NeRF所做的所有任务，迁移到3DGS的框架中，实现更快速的渲染，和更容易应用到下游任务的框架**

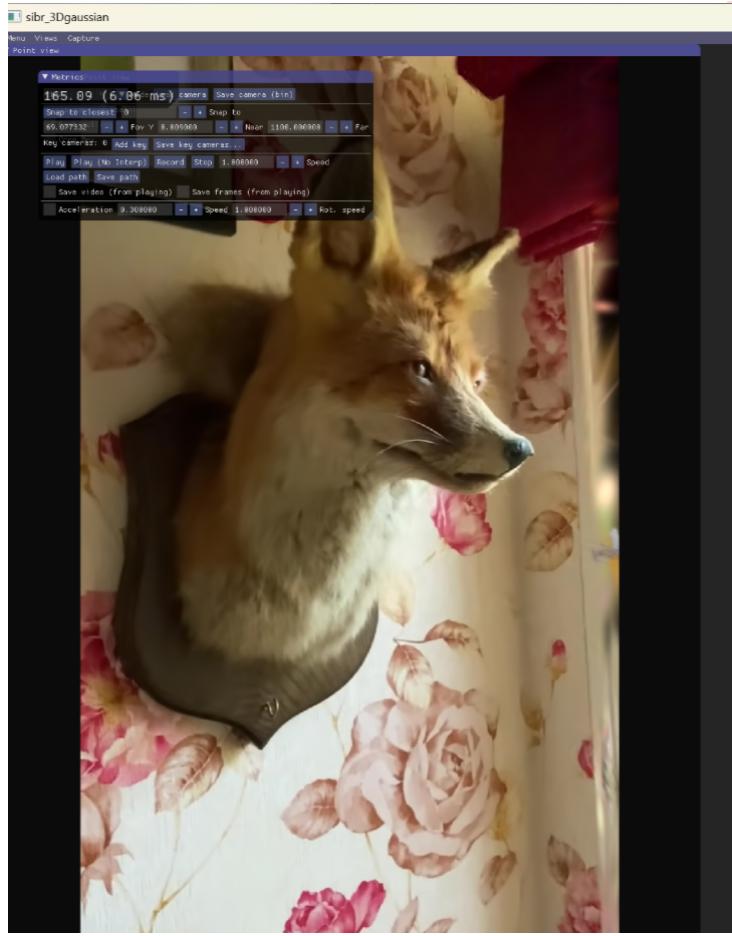
#### 2.1.5 My experiment

我前面也自己复现过3DGS，放几张图简单展示一下效果：

训练7000与30000迭代次的顶点和表面的模型文件(input.ply)



可视化界面下的场景：



## 2.2 Lerf: Language embedded radiance fields

### 2.2.1 Motivation

传统 NeRF (Neural Radiance Fields)，前面已经介绍过，在三维场景重建和新视角合成上表现出色，但缺乏对**开放词汇语义查询**的支持。例如，用户无法直接询问“找到桌上的马克杯”或“定位沙发附近的插座”。Lerf 的提出旨在解决以下核心问题：

1. **语义-几何对齐**：将开放式的自然语言描述与 3D 场景的几何/纹理信息关联。
2. **零样本查询支持**：无需预定义类别标签，适应任意语言描述（如“布满锈迹的水管”）。
3. **跨模态融合**：弥合视觉渲染模型 (NeRF) 与语言模型 (CLIP) 的鸿沟，实现可交互的语义场景理解。

之前也有很多工作，比如Semantic-NeRF等，但都没有实现真正意义上的开放词汇查询。

Lerf，我感觉是三维语义分割的开山之作，最重要的是**引入了CLIP这个预训练模型**，这样为实现真正的开放语义查询提供了新思路。

### 2.2.2 Methods

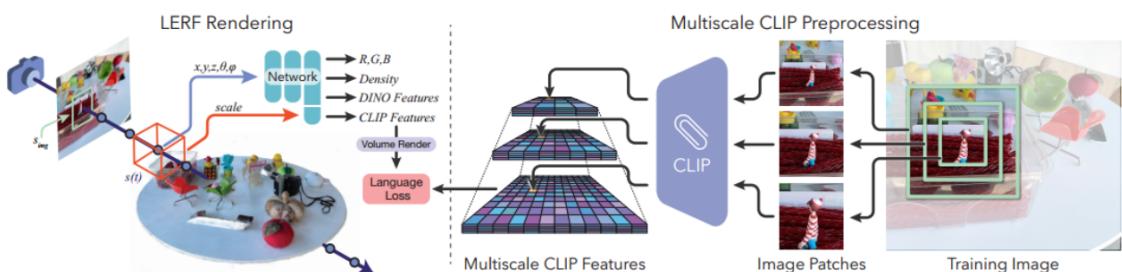


图7：LERF方法架构图

### 2.2.2.1 体渲染

如左图所示：LERF 通过沿训练光线体积渲染 CLIP 嵌入，来学习 NeRF 内部的**密集多尺度语言场**

相对NeRF的**新的输入**：规模 $s(t)$

**新的输出**：DINO正则化特征，CLIP特征

**DION正则化特征**：DINO 已被证明即使在没有标签的情况下进行训练也能表现出新兴的对象分解属性，并且可以很好地提炼到 3D 场，可以帮助进行语义分割。

### 2.2.2.2 多尺度监督

如右图所示：在训练视图中监督这些嵌入，以提供多视图一致性并平滑底层语言场。预先计算多个图像裁剪尺度上的**图像金字塔**，并存储每个裁剪的 CLIP 嵌入。在训练期间，我们会在输入视图中均匀地随机采样射线原点。

### 2.2.2.3 Lerp的查询

查询 LERF 分为两个部分：1) 获取渲染嵌入的相关性分数和 2) 根据提示自动选择尺度  $s$ 。

渲染相关性分数为： $\min_i \frac{\exp(\phi_{\text{lang}} \cdot \phi_{\text{quer}})}{\exp(\phi_{\text{lang}} \cdot \phi_{\text{canon}}^i) + \exp(\phi_{\text{lang}} \cdot \phi_{\text{quer}})}$ ，这个分数表示渲染嵌入与规范嵌入相比与查询嵌入的接近程度。

对于每个查询，我们计算一个尺度  $s$  来评估。为此，我们以生成 0 到 2 米尺度范围内的相关性图，并选择产生最高相关性分数的尺度。此尺度用于输出相关性图中的所有像素。这是一种启发式方法。

## 2.2.3 Advantages and Disadvantages

### Advantages

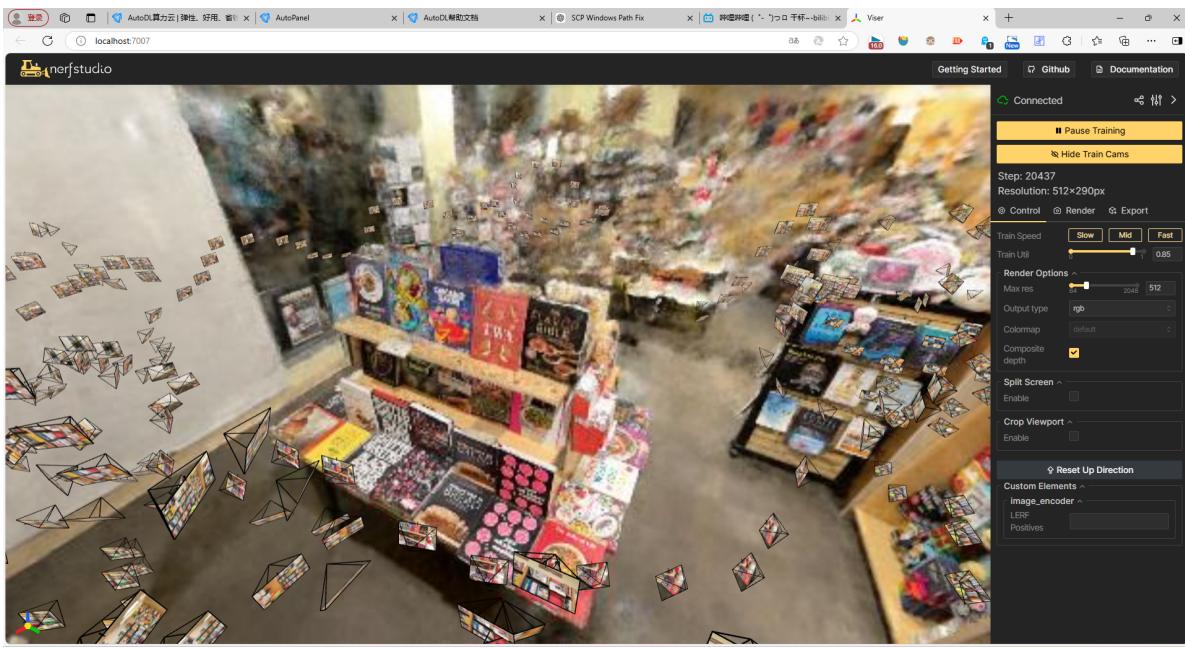
- **首次实现开放词汇语义查询**：支持任意自然语言描述（如“斑驳的蓝色信箱”），无需预定义类别标签，突破传统语义分割的封闭词汇限制。
- **零样本泛化能力**：直接继承 CLIP 模型的跨模态理解能力，对未见过的物体描述（如新型家电术语）具备鲁棒性。
- **细粒度空间定位**：多尺度特征池化技术可精确定位物体部件（如“笔记本电脑的触摸板”“椅子的滚轮”）。

### Disadvantages

- **计算成本高昂和训练时间长**：NeRF本身存在的劣势+多尺度金字塔监督计算CLIP嵌入
- **查询边界不清晰**：难以处理不精确且模糊的 3D 语言场，无法辨别对象之间的清晰边界，大多数具有不同尺度的块通常无法准确地包含对象，要么经常包括背景中的其他对象，要么省略目标对象的某些部分
- **跨视角一致性挑战**：因为一个对象的不同视图可以提供接近但不同的 CLIP 特征

### 2.2.4 My Experiment

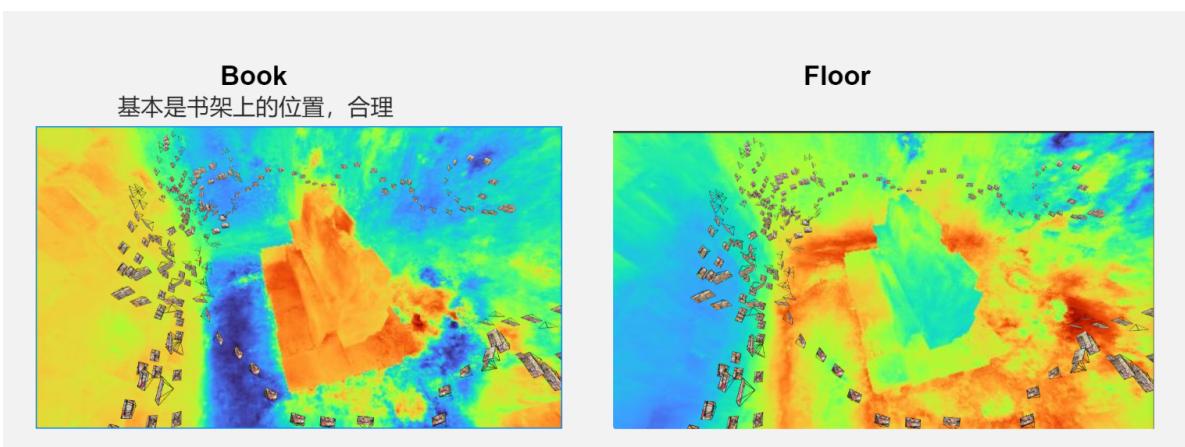
这个文章也进行过复现，借助了nerfstudio集成平台，使用了lerf-lite(lerf的轻量化版本)，效果大概如图所示：



我们基本重现了三维的书店场景，可能因为训练次数过少等原因，部分场景还存在模糊

网页段支持转动视角，观看多角度的三维场景

下面根据prompt进行lerf效果的测试，主要观察相关性图relevancy\_0



## 2.3 Langsplat: 3d language gaussian splatting

### 2.3.1 Motivation

前面提到的Lerf,开辟了三维语义分割的赛道，但存在**计算成本高昂和训练时间长和查询边界不清晰**两大严重缺点。

3DGS的出现，已经从各方面取代了NeRF，那自然而然就想在3DGS上能不能实现三维语义查询，实现更低成本和更高效的查询呢？

加上在**二维图像分割领域**，也有工作首先使用了CLIP与SAM两个预训练模型，来实现二维图像的开放语义分割。

我想这些都是本文工作的自然而然的motivation，本文也成功开辟了3DGS实现三维语义分割的新赛道！

---

Lerf 等语言嵌入辐射场方法存在**训练慢、推理延迟高**的瓶颈，难以应用于实时交互场景。LangSplat 的提出旨在解决：

- 实时语言查询需求**: 机器人导航、AR 等场景需毫秒级响应自然语言指令 (如“避开左侧障碍物”)
- 显式语义控制**: 避免 NeRF 隐式表示导致的语义模糊性，实现高精度开放词汇定位。
- 高效多模态融合**: 将 CLIP 特征直接编码至显式 3D 表示中，减少计算冗余。

**核心创新**: 用 3D 高斯泼溅 (3DGS) 替代 NeRF 作为几何基座，实现语言特征的实时渲染。

### 2.3.2 Methods

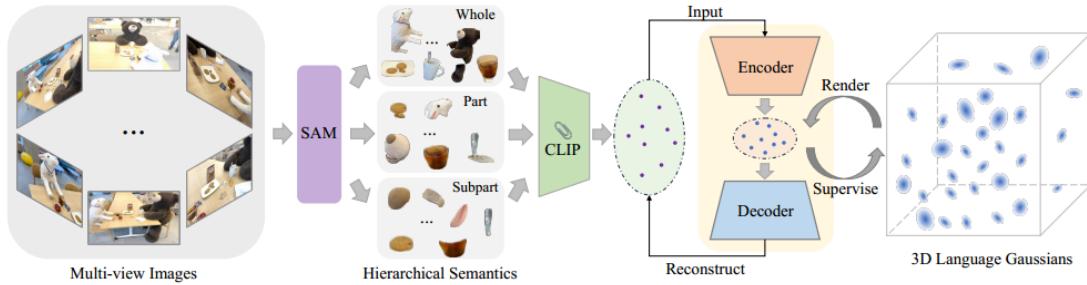


图8：LangSplat架构图

首先，LangSplat 利用 SAM 学习分层语义来解决点模糊性问题。然后将片段掩码发送到 CLIP 图像编码器以提取相应的 CLIP 嵌入。

其次，LangSpalt设计了一套场景特化的编码器与译码器，使 3D 语言高斯在场景特定的潜在空间上学习语言特征以降低内存成本。首先使用这些获得的 CLIP 嵌入学习自动编码器。在查询期间，渲染的语言嵌入被发送到解码器以恢复 CLIP 空间上的特征。

#### 2.3.2.1 3D语言场的挑战

**存在两个问题：**

1. CLIP 嵌入是图像对齐的，而不是像素对齐的。
2. 对像素对齐的语言特征进行建模面临点歧义的问题，因为对象上的单个点会对多个语义级别的区域做出贡献。

为了解决这些问题，大多数现有方法从裁剪的图像块中提取一层层的 CLIP 特征。然而，这种多尺度方法**有两个局限性**。

1. 图像块特征不精确，因为它们通常包含额外的上下文物体信息，导致语言场过于平滑，物体边界不清晰。为了缓解图像块问题，大多数方法利用额外的像素对齐的 DINO 特征来监督网络。然而，学习到的 3D 语言特征仍然不精确。
2. 它需要在推理过程中同时在多个尺度上进行渲染才能找到最佳尺度。由于尺度数可能高达 30，这会显著降低推理速度。

#### 2.3.2.2 使用SAM学习分层语义

SAM作为图像分割的基础模型，SAM 可以准确地将一个像素与其周围的属于同一对象的像素分组，从而将图像分割成许多具有清晰边界的对象掩码。此外，SAM 通过为点提示生成**三个不同的掩码**（即整体、部分和子部分）来解决点模糊性问题，这三个掩码代表了三个层次的语义。**我们借鉴SAM解决点歧义的思路来解决语义层次问题。**

在本文中，我们提出利用 SAM 来获取精确的对象掩码，然后使用这些掩码来获取像素对齐的特征。我们还明确地对 SAM 定义的语义层次进行建模，以解决点模糊性问题。借助 SAM，我们可以捕捉 3D 场景中对象的语义层次，为每个输入图像提供准确的多尺度分割图。

具体来说，我们将一个  $32 \times 32$  点提示的规则网格输入 SAM，以获得三个不同语义级别下的掩码：Ms0、Mp0、Mw0，其中 Ms0、Mp0 和 Mw0 分别代表子部分、部分和整体级别的掩码。然后，我们根据预测的 IoU 分数、稳定性分数和掩码之间的重叠率，为三个掩码集中的每一个删除冗余掩码。每个过滤后的掩码集根据其各自的语义级别独立执行全面的全图像分割，得到 **三个分割图：Ms、Mp、Mw**。这些分割图精确地描绘了对象在其层次结构级别的边界，有效地将场景划分为具有语义意义的区域。利用获得的分割图，我们继续为每个分割区域提取 CLIP 特征。

由于我们有“整体”、“部分”和“子部分”级别的不同分割图，因此我们可以在**这些预定义的尺度上直接查询**。这样就无需在多个绝对尺度上进行密集搜索，从而使查询过程更加高效。

### 2.3.2.3 3D语言高斯

在本文中，我们提出了 3D 语言高斯 Splatting，它使用**三个语言嵌入 {fs, fp, fw}** 来增强每个 3D 高斯。这些嵌入源自 CLIP 特征，可捕获 SAM 提供的分层语义。增强的高斯被称为 **3D 语言高斯**。我们还采用**基于图块的光栅化器**来保持渲染效率：

$$F^l(v) = \sum_{i \in \mathcal{N}} f_i^l \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), l \in \{w, p, s\},$$

作为一种显式建模方法，我们的 LangSplat 可以创建数百万个 3D 点来建模复杂的 3D 场景。为了降低内存成本并提高效率，我们引入了**场景语言自动编码器**。该自动编码器将场景中的 CLIP 嵌入映射到低维潜在空间，从而降低内存需求。该自动编码器将场景中的 CLIP 嵌入映射到低维潜在空间，从而降低内存需求。

在训练自动编码器后，我们将所有 CLIP 嵌入转换为**场景特定的潜在特征**。我们让 3D 语言高斯在场景特定的潜在空间而不是 CLIP 潜在空间中学习语言嵌入。

### 2.3.2.4 开放语义查询任务

选择产生最高相关性得分的语义级别。

对于 3D 物体定位任务，直接选择相关性得分最高的点。

对于 3D 语义分割任务，我们过滤掉相关性得分低于所选阈值的点，并使用剩余区域预测对象掩码。

## 2.3.3 Advantages and Disadvantages

Advantages：

- **高效的渲染与查询**

通过将语言特征编码到一组三维高斯体（Gaussians）上，LangSplat 在  $1440 \times 1080$  分辨率下实现了  $\approx 199 \times$  的速度提升，相比先前的 NeRF + CLIP 方法（如 LERF）显著加速了语言查询和渲染过程。

- **精确的三维语义场**

LangSplat 在构建 3D 语言场时，能准确捕捉物体边界，不再像基于 NeRF 的方法那样产生模糊、不明确的语义响应，大幅提升了开放词汇的 3D 目标定位和语义分割精度。

- **内存与计算需求降低**

论文先训练场景特定的语言自编码器，将 CLIP 嵌入映射到低维 latent space，再学习语言特征，这种两阶段策略有效减少了直接在高维 CLIP 空间建模的巨大内存开销。

- **分层语义学习**

利用 SAM (Segment Anything Model) 引入多层次语义，使模型无需对所有尺度都进行大规模查询，也不依赖 DINO 特征正则化，从而简化了训练并提升了不同尺度对象的识别准确性。(十分自然的思路)

Disadvantages:

- **查询速度不够快**

每个新场景都要先训练语言自编码器和高斯体分布，训练过程仍需耗费若干分钟至上小时，不适用于对“零训练”或实时更新场景的应用。

- **对 CLIP 及 SAM 质量依赖较大**

作为底层特征源，CLIP 在极端或专业领域文本理解上可能存在盲区；SAM 分割错误也会传导到 3D 语义场，影响定位和分割精度。

- **可扩展性与通用性验证不足**

论文主要在 LERF 数据集和 3D-OVS 上测试，尚缺乏对大规模、百变场景（如室外城市、自然景观）的广泛评估，对真正“开放世界”部署还有进一步验证的需求。

#### 2.3.4 My Experiment

这篇文章我也进行了复现，最终结果结构大概如图所示：

即渲染的图像与语义信息，ground truth的图像与语义信息。

📁 gt	2025-02-26 17:18	文件夹
📁 gt_npy	2025-02-26 17:18	文件夹
📁 renders	2025-02-26 17:18	文件夹
📁 renders_npy	2025-02-26 17:18	文件夹

进行语义分割后的图片大概如图所示：



## 2.4 FastIgs: Speeding up language embedded gaussians with feature grid mapping

### 2.4.1 Motivation

这篇同样是进一步follow Langsplat的工作，我挑选这篇文章的意图是：

我觉得它代表了三维语义分割领域在结合3DGS后的一个发展方向，即进一步追求查询的准确性和效率。

以目前CLIP+SAM的模板来说：

存在三个改进方向：

1.如何使用更好的方法，得到与像素对齐的CLIP embedding

2.如何实现更好的查询方式，实现更快更准确的语义查询

3.如何巧妙地在3DGS中融入语义信息，降低渲染的时间与成本

这篇文章的出发点应该是实现更好的1, 3

### 2.4.2 Methods

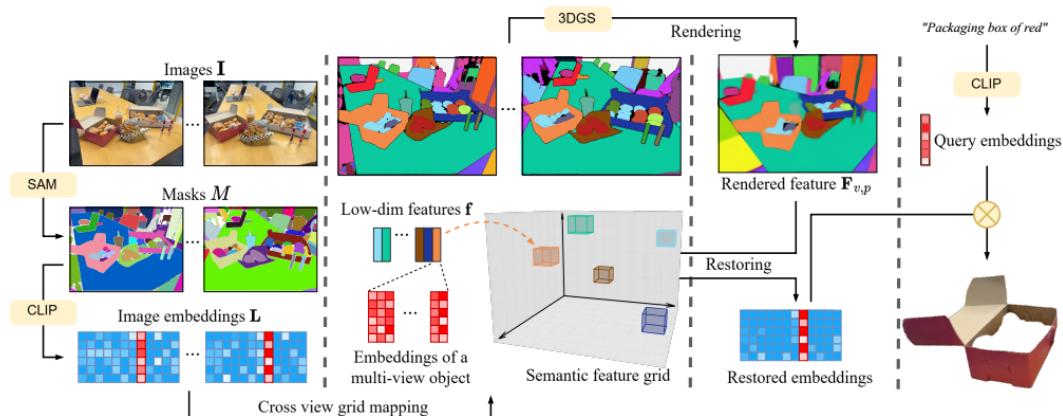


图9:fastlgs方法结构图

左图:初始化。中间的图:特征网格构建和嵌入恢复。右图:使用开放词汇提示进行查询。

#### 2.4.2.1 初始化

对于每一个输入的图像，使用SAM获得每一个对象的分割掩码，再通过分割掩码或者对应的CLIP embedding

#### 2.4.2.2 语义网格网络

为了解决这些问题，我们提出语义特征网格，它存储场景中每个对象的多视图语言特征，并映射到一个低维特征。

低维特征将根据相应的掩码按像素方式分配，并在3DGS中有效地训练，而网格可用于在推断期间基于渲染的低维特征恢复准确的剪辑嵌入。在实践中，这种策略能够实现更少的消耗和更快的速度。

#### 2.4.2.3 映射策略

为了实现跨视图的对象的一致特征，我们顺序匹配相邻图像集中去噪掩模并分配特征。考虑到上述多视图中剪辑特征的不稳定性，匹配过程包括关键点对应和特征相似性比较。

**关键点对应：**我们通过使用SIFT和k-最近邻(KNN)算法获得的关键点来区分掩模对应的优先级，以实现精确和鲁棒的匹配。

**特征相似性：**由于一些具有平滑像素值的分割具有很少的关键点，所以根据由剪辑嵌入L和颜色分布C的混合特征计算的相似性来采用匹配过程。设置阈值 $\theta$ ，使得相似性sim高于 $\theta$ 的掩模将具有相同的特征f，而其他掩模将被分配给新的低维特征。

对于两个分割掩模 $M_i, M_j$ ，相似性 $sim_{i,j}$ 计算如下：

$$sim_{i,j} = \alpha sim_{i,j}^{color} + (1 - \alpha) sim_{i,j}^{CLIP},$$

其中 $\alpha$ 是颜色分布特征C的相似性的权重。 $sim_{i,j}^{CLIP}$ 是基于C的Bhattacharyya距离计算的

#### 2.4.2.4 查询和高斯训练

基本和前面的一样，这里不多赘述

### 2.4.3 Advantages and Disadvantages

Advantages:

#### 1. 进一步的速度提升

FastLGS 在 1440×1080 分辨率下支持实时 (<1 s) 开放词汇查询，相比 LERF 快  $\approx 98\times$ ，相比 LangSplat 快  $\approx 4\times$ ，相比 LEGaussians 快  $\approx 2.5\times$ 。

#### 2. 进一步更低的计算与内存开销

通过将多视图 CLIP 特征先存入稠密的特征网格，再映射到低维特征场，FastLGS 摒弃了传统 MLP 的高昂计算，显著降低了训练与推理的资源消耗。

#### 3. 进一步高精度语义定位与分割

实验表明，FastLGS 在开放词汇目标掩码生成和语义相关性上优于 LangSplat，在多种查询（单对象或多对象）下均能产生更准确的 3D 语义响应。

#### 4. 易于下游任务集成

FastLGS 的语义特征网格可无缝应用于 3D 分割、3D 对象填充 (inpainting) 等多种三维操作系统，展现了良好的适应性和通用性。

#### Disadvantages

##### 1. 对象中心范式

当前方法以“整个对象”为单位进行查询，对于细分部件（如“自行车车把”或“椅子椅脚”）的精细定位仍表现不足，有待通过多粒度特征网格改进。

##### 2. 对预训练模型的依赖

FastLGS 依赖 CLIP 提供的语言-视觉嵌入，以及 SAM 生成的分割掩码；当底层模型在专业领域或弱光等极端条件下表现不佳时，会直接影响语义场质量。

##### 3. 评估场景有限

论文主要在室内和合成数据集（如 LERF、3D-OVS）上测试，尚缺乏对城市街景、大型户外自然场景等开放世界应用的广泛评估，通用性待进一步验证。

## 2.5 Vggt: Visual geometry grounded transformer

### 2.5.1 Motivation

VGGT为了解决如下问题：

#### 摆脱后处理优化的束缚

传统三维重建依赖 Bundle Adjustment (BA) 等迭代几何优化，计算复杂且耗时。VGGT 探索能否在 **单次前向推理** 中完成相机参数估计、稠密重建及点跟踪，而无需后续优化，从而实现实时级别的三维推断。

#### 统一多任务的神经网络骨干

过去多数深度网络只专注单一三维任务（如单目深度估计或新视图合成）。VGGT 的动机在于：能否训练一个标准的大型 Transformer，**同时输出相机内外参数、深度图、点云及点跟踪**，并让这些互相关联的任务相互促进，提升整体精度

#### 向大模型范式靠拢

类比 GPT、CLIP、Stable Diffusion 等大规模骨干模型的多功能性，VGGT 也采用 **几乎零三维先验**（仅交替的帧内与全局注意力），并依赖大量带注释的三维数据进行训练，期望获得与大模型类似的泛化与迁移能力



Figure 1. VGGT is a large feed-forward transformer with minimal 3D-inductive biases trained on a trove of 3D-annotated data. It accepts up to hundreds of images and predicts cameras, point maps, depth maps, and point tracks for all images at once in less than a second, which often outperforms optimization-based alternatives without further processing.

图11：VGGT效果展示图

我们引入了VGGT，这是一种前馈神经网络，可从一个、几个甚至数百个场景输入视图中执行 3D 重建。VGGT 可预测全套 3D 属性，包括相机参数、深度图、点云图和 3D 点轨迹，且只需一次前向传播，耗时仅数秒。值得注意的是，即使不进行进一步处理，它的表现也常常优于基于优化的替代方法。

### 2.5.2 Methods

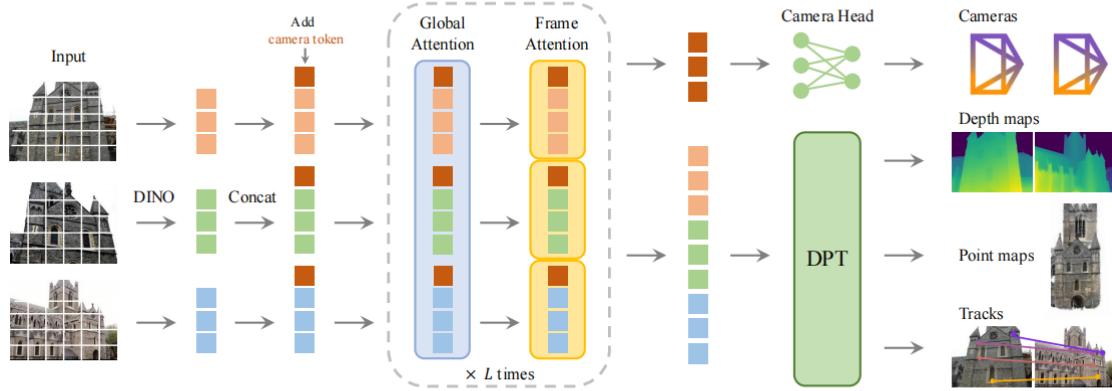


Figure 2. **Architecture Overview.** Our model first patchifies the input images into tokens by DINO, and appends camera tokens for camera prediction. It then alternates between frame-wise and global self attention layers. A camera head makes the final prediction for camera extrinsics and intrinsics, and a DPT [87] head for any dense output.

图12：VGGT架构图

- 输入处理：输入(一张或多张图像) → DINO处理 → patches → 图像tokens + 相机信息token
- 特征提取：Alternating-Attention 交替注意力（全局 + 帧内）
- 输出预测：Camera Head相机头【预测相机内外参】 + DPT (Dense Prediction Transformer) 深度头【生成深度图和对应的置信度分数】 + Point Head点云头【生成3D世界点图和对应的置信度分数】 + Track Head跟踪头【跟踪多个视图中的点】

#### 2.5.2.1 问题定义

- **输入：** 输入是一系列RGB图像  $(I_i)_{i=1}^N$ ，其中  $N$  表示图像数量，每张图像  $I_i \in \mathbb{R}^{3 \times H \times W}$ ，表示图像的高度为  $H$ 、宽度为  $W$  的三通道RGB图像。这些图像观察的是同一个3D场景。
- **输出：** VGGT模型的目标是将输入图像序列映射为每帧图像对应的3D标注信息，包括：
  - 相机参数  $g_i \in \mathbb{R}^9$ ：包括相机的内参和外参，具体表示为旋转四元数  $q \in \mathbb{R}^4$ 、平移向量  $t \in \mathbb{R}^3$  和视场角  $f \in \mathbb{R}^2$ 。
  - 深度图  $D_i \in \mathbb{R}^{H \times W}$ ：与输入图像分辨率相同，表示每个像素对应的深度值。
  - 点云图  $P_i \in \mathbb{R}^{3 \times H \times W}$ ：表示每个像素对应的3D场景点，这些点以第一帧相机的坐标系为参考。
  - 跟踪特征图  $T_i \in \mathbb{R}^{C \times H \times W}$ ：用于点跟踪的特征图， $C$  表示特征的维度。

### 2.5.2.2 主干网络（用于处理输入图像并提取特征的Transformer网络）

- 交替注意力机制 (Alternating-Attention, AA)
  - 帧内自注意力 (frame-wise self-attention) + 全局自注意力 (global self-attention)
  - 仅使用自注意力机制，而没有采用交叉注意力 (cross-attention)
  - 默认使用24层交替注意力模块

### 2.5.2.3 Prediction heads 预测头

#### 相机参数预测 (Camera Predictions)

- 预测方法：相机参数  $(\hat{g}_i)_{i=1}^N$  是从输出的相机token  $(\hat{t}_{g_i})_{i=1}^N$  中预测出来的，具体过程是使用额外的自注意力层，随后通过一个线性层来预测相机的内参和外参。这个过程形成了一个“相机头”(camera head)，用于预测相机参数。
- 坐标系：所有相机参数、点云图和深度图都是在第一帧相机  $g_1$  的坐标系中预测的，这个坐标系也被当作世界参考坐标系。因此，第一帧的相机外参被设置为单位变换，即旋转四元数  $q_1 = [0, 0, 0, 1]$  和平移向量  $t_1 = [0, 0, 0]$ 。
- 特殊token的作用：第一帧的特殊相机token和寄存器token  $(\hat{t}_g, \hat{t}_R)$  使得Transformer能够区分第一帧与其他帧，并在第一帧的坐标系中表示3D预测结果。

#### 密集预测 (Dense Predictions)

- 深度图、点云图和跟踪特征的预测：使用输出的图像token  $\hat{t}_{I_i}$  来预测密集的输出，包括深度图  $D_i$ 、点云图  $P_i$  和跟踪特征图  $T_i$ 。具体来说，首先将  $\hat{t}_{I_i}$  转换为密集特征图  $F_i \in \mathbb{R}^{C'' \times H \times W}$ ，然后通过一个  $3 \times 3$  的卷积层分别映射到对应的深度图和点云图。同时，DPT头还会输出密集特征图  $T_i \in \mathbb{R}^{C \times H \times W}$ ，这些特征图将作为跟踪头的输入。
- 不确定性预测：模型还会预测深度图和点云图的不确定性图  $\Sigma_{D_i}$  和  $\Sigma_{P_i}$ ，这些不确定性图在损失函数中使用，并且在训练完成后，它们与模型对预测结果的置信度成正比。

#### 点跟踪 (Tracking)

- 跟踪模块的实现：跟踪模块  $T$  使用CoTracker2架构，它以密集跟踪特征图  $T_i$  作为输入。具体来说，给定一个查询点  $y_j$  在查询图像  $I_q$  中（在训练阶段，始终设置  $q = 1$ ，但其他图像也可以作为查询图像），跟踪头  $T$  预测出所有图像  $I_i$  中与  $y_j$  对应的2D点集合  $(\hat{y}_{j,i})_{i=1}^N$ 。
- 跟踪过程：首先，通过双线性插值在查询图像的特征图  $T_q$  上采样查询点  $y_j$  的特征。然后，将这个特征与所有其他特征图  $T_i$  ( $i \neq q$ ) 进行相关性计算，得到一组相关性图。这些相关性图经过自注意力层处理后，最终预测出对应的2D点  $\hat{y}_i$ ，这些点与  $y_j$  对应于同一个3D点。
- 跟踪模块的特点：与VGG-SfM类似，跟踪器不假设输入帧之间存在时间顺序，因此可以应用于任意一组输入图像，而不仅仅是视频。

#### 2.5.2.4 训练损失函数

注意，VGGT跟DUST3R不一样，各个输出并不是独立的（DUST3R的部分结果是额外推理得到的，比如pose可以通过pnp来计算出来）。因此，VGGT训练的时候是一个针对四个task的统一的loss，如下图所示

1. 相机损失 ( $\mathcal{L}_{\text{camera}}$ ):

$$\mathcal{L}_{\text{camera}} = \sum_{i=1}^N \|\hat{g}_i - g_i\|_\epsilon$$

该损失函数通过Huber损失  $\|\cdot\|_\epsilon$  比较预测的相机参数  $\hat{g}_i$  与真实值  $g_i$ 。

2. 深度损失 ( $\mathcal{L}_{\text{depth}}$ ):

$$\mathcal{L}_{\text{depth}} = \sum_{i=1}^N \left( \|\hat{\Sigma}_{D_i} \odot (\hat{D}_i - D_i)\| + \|\hat{\Sigma}_{D_i} \odot (\nabla \hat{D}_i - \nabla D_i)\| - \alpha \log \hat{\Sigma}_{D_i} \right)$$

该损失函数结合了预测的不确定性图  $\hat{\Sigma}_{D_i}$ ，通过加权预测深度图  $\hat{D}_i$  与真实深度图  $D_i$  的差异，以及它们的梯度差异。此外，还包含一个负对数似然项  $-\alpha \log \hat{\Sigma}_{D_i}$  以鼓励模型学习不确定性。

3. 点云图损失 ( $\mathcal{L}_{\text{pmap}}$ ):

$$\mathcal{L}_{\text{pmap}} = \sum_{i=1}^N \left( \|\hat{\Sigma}_{P_i} \odot (\hat{P}_i - P_i)\| + \|\hat{\Sigma}_{P_i} \odot (\nabla \hat{P}_i - \nabla P_i)\| - \alpha \log \hat{\Sigma}_{P_i} \right)$$

该损失函数的形式与深度损失类似，但使用的是点云图的不确定性  $\hat{\Sigma}_{P_i}$ 。

4. 跟踪损失 ( $\mathcal{L}_{\text{track}}$ ):

$$\mathcal{L}_{\text{track}} = \sum_{j=1}^M \sum_{i=1}^N \|y_{j,i} - \hat{y}_{j,i}\|$$

该损失函数计算所有查询点  $y_j$  在查询图像  $I_q$  中的真实对应点  $y_{j,i}$  与预测对应点  $\hat{y}_{j,i}$  之间的距离。此外，还应用了一个可见性损失（二元交叉熵）来估计一个点在给定帧中是否可见。

#### 真值坐标的归一化（Ground Truth Coordinate Normalization）

为了消除尺度和全局参考坐标系的歧义，我们对数据进行归一化处理。具体来说，我们将所有量表示在第一帧相机  $g_1$  的坐标系中，并计算点云图  $P$  中所有3D点到原点的平均欧几里得距离，使用这个尺度来归一化相机平移  $t$ 、点云图  $P$  和深度图  $D$ 。重要的是，我们不对Transformer输出的预测结果进行这种归一化，而是让模型从训练数据中学习我们选择的归一化方式。

### 2.5.3 Experiment Result

论文的实验可以说是非常solid的，也展示非常强大的泛化能力

1. 首先是pose estimation，如下图所示。采用的评判指标是AUC（综合了Relative Rotation Accuracy (RRA)和Relative Translation Accuracy RTA，虽然是SLAM中比较少见的，但估计是为了跟PoseDiffusion对比）。可以看到，时间上用时是最短的，而精度也是最高的。序列都是在Co3Dv2下训练，没有在Re10K下训练过的。对于像DUST3R和MASt3R这种还是有global BA的也远不如VGTT。

此外，加上BA后，VGTT的性能会更好，而VGTT本身就可以获得精度比较搞的point/depth map，这可以给BA提供较好的初始化，也不需要三角化或者iterative refinement等一系列的操作

Methods	Re10K ( <i>unseen</i> )	CO3Dv2	Time
	AUC@30 ↑	AUC@30 ↑	
Colmap+SPSG [92]	45.2	25.3	~ 15s
PixSfM [66]	49.4	30.1	> 20s
PoseDiff [124]	48.0	66.5	~ 7s
DUST3R [129]	67.7	76.7	~ 7s
MASt3R [62]	76.4	81.8	~ 9s
VGGSfM v2 [125]	78.9	83.4	~ 10s
MV-DUST3R [111] <sup>‡</sup>	71.3	69.5	~ 0.6s
CUT3R [127] <sup>‡</sup>	75.3	82.8	~ 0.6s
FLARE [156] <sup>‡</sup>	78.8	83.3	~ 0.5s
Fast3R [141] <sup>‡</sup>	72.7	82.5	~ 0.2s
Ours (Feed-Forward)	<u>85.3</u>	<u>88.2</u>	~ 0.2s
Ours (with BA)	<b>93.5</b>	<b>91.8</b>	~ 1.8s

Table 1. **Camera Pose Estimation on RealEstate10K [161] and CO3Dv2 [88]** with 10 random frames. All metrics the higher the better. None of the methods were trained on the Re10K dataset. Runtime were measured using one H100 GPU. Methods marked with <sup>‡</sup> represent concurrent work.

2. 深度估计的实验对比，如下所示。只有DUST3R和VGTT不需要GT pose，而MASt3R要估算深度是需要基于camera pose来做三角化的。而VGTT也是取得最佳的performance的

Known GT camera	Method	Acc.↓	Comp.↓	Overall↓
✓	Gipuma [40]	<b>0.283</b>	0.873	0.578
✓	MVSNet [144]	0.396	0.527	0.462
✓	CIDER [139]	0.417	0.437	0.427
✓	PatchmatchNet [121]	0.427	0.377	0.417
✓	MASt3R [62]	0.403	0.344	0.374
✓	GeoMVSNet [157]	0.331	<b>0.259</b>	<b>0.295</b>
✗	DUST3R [129]	2.677	0.805	1.741
✗	Ours	<b>0.389</b>	<b>0.374</b>	<b>0.382</b>

Table 2. **Dense MVS Estimation on the DTU [51] Dataset.** Methods operating with known ground-truth camera are in the top part of the table, while the bottom part contains the methods that do not know the ground-truth camera.

3. point map估算的效果。如下所示，DUST3R输入多张image的时候是成对的处理的，因此当32张图输入，时间爆炸式增长到200s但是VGTT仍然是0.6s。相比起单张或者两张图片的输入，时间增长并没有DUST3R那么大。

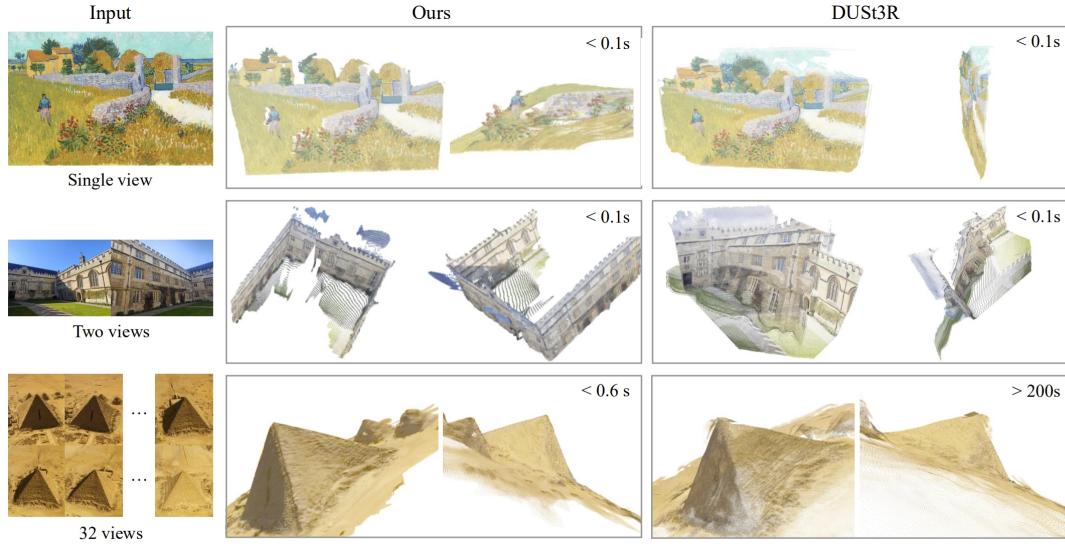


Figure 3. **Qualitative comparison of our predicted 3D points to DUS3R on in-the-wild images.** As shown in the top row, our method successfully predicts the geometric structure of an oil painting, while DUS3R predicts a slightly distorted plane. In the second row, our method correctly recovers a 3D scene from two images with no overlap, while DUS3R fails. The third row provides a challenging example with repeated textures, while our prediction is still high-quality. We do not include examples with more than 32 frames, as DUS3R runs out of memory beyond this limit.



Figure 4. **Additional Visualizations of Point Map Estimation.** Camera frustums illustrate the estimated camera poses. Explore our interactive demo for better visualization quality.

## 2.5.4 My Test

这里我们直接使用了镜像进行测试：

用时很短，效果也是非常好

## VGGT: Visual Geometry Grounded Transformer

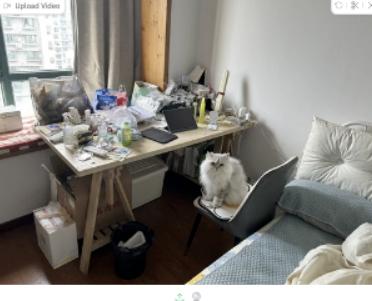
[GitHub Repository](#) | [Project Page](#)

Upload a video or a set of images to create a 3D reconstruction of a scene or object. VGGT takes these images and generates all key 3D attributes, including extrinsic and intrinsic camera parameters, point maps, depth maps, and 3D point tracks.

### Getting Started:

1. Upload Your Data: Use the "Upload Video" or "Upload Images" buttons on the left to provide your input. Videos will be automatically split into individual frames (one frame per second).
2. Preview: Your uploaded images will appear in the gallery on the left.
3. Reconstruct: Click the "Reconstruct" button to start the 3D reconstruction process.
4. Visualize: The 3D reconstruction will appear in the viewer on the right. You can rotate, pan, and zoom to explore the model, and download the GLB file. Note the visualization of 3D points may be slow for a large number of input images.
5. Adjust Visualization (Optional): After reconstruction, you can fine-tune the visualization using the options below (click to expand):  
Please note: Our model itself usually only needs less than 1 second to reconstruct a scene. However, visualizing 3D points may take tens of seconds due to third-party rendering, which are independent of VGGT's processing time. Please be patient or, for faster visualization, use a local machine to run our demo from our [GitHub repository](#).

Upload Video



Upload Images

将文件拖放到此处  
或  
点击上传



Click any row to load an example.

Upload Video	num_images	Confidence Threshold (%)	Filter Black Background	Filter White Background	Show Camera	Filter Sky	Select a Prediction Mode	is_example
	22	20	false	false	true	false	Depthmap and Camera Branch	True
	30	35	false	false	true	false	Depthmap and Camera Branch	True
	1	15	false	false	true	false	Depthmap and Camera Branch	True
	1	20	false	false	true	true	Depthmap and Camera Branch	True
	8	5	false	false	true	false	Depthmap and Camera Branch	True
	25	50	false	false	true	false	Depthmap and Camera Branch	True
	20	45	false	false	true	false	Depthmap and Camera Branch	True

3D Reconstruction (Point Cloud and Camera Poses)

Loading and Reconstructing...

3D Model

Reconstruct

Clear

Select a Prediction Mode

Depthmap and Camera Branch

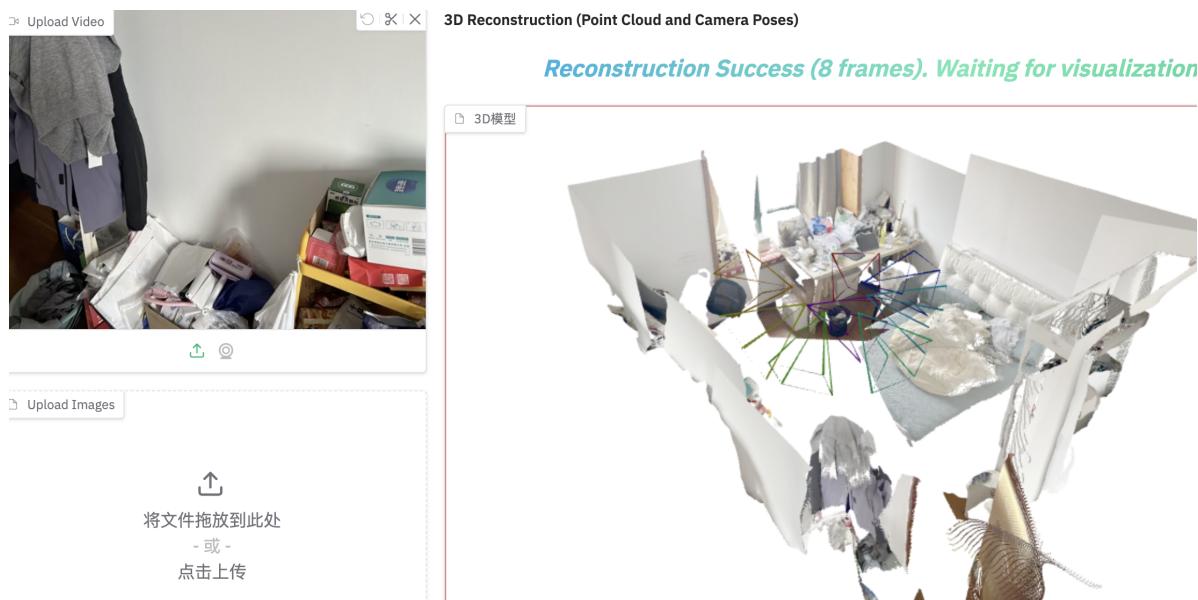
Pointmap Branch

Confidence Threshold (%)Show Points from Frame



Show CameraFilter SkyFilter Black BackgroundFilter White Background

processing | 22.77/7.5s



## 2.5.5 Advantages and Disadvantages

Advantages:

1. **速度**: 单次前馈推理即可完成所有预测，VGGT在单个GPU上处理100张图像大约只需2秒，同时实现了比耗时多50-100倍的方法更高的精度。
2. **精度**: 在多项任务中超越传统优化方法
3. **通用性**: 无需特定3D先验，适用于多种下游任务

Disadvantages:

1. 当前模型仅适用于常规视角图像，无法处理鱼眼或全景图像。并且当输入图像存在极端旋转时，重建性能会显著下降。然模型能够处理轻微的非刚性运动，但在显著非刚性变形（如大幅度的物体形变）场景中表现不佳。
2. 训练此类模型仍然是计算密集型任务，需要算力
3. 参数量大，作为一款12亿参数的模型，未经优化过大，不适合移动部署。需要蒸馏或量化版本，才能推广使用。

## 2.5.6 Personal Overview and some ideas

我觉得目前，3DGS已经完全取代了NeRF，很明显的感受是，如果你现在用NeRF为基本方法去做工作，你的审稿人会和你说能不能用3DGS做，我觉得这是一个时代的跨越。

正如论文作者所述，我们正在见证视觉几何从「手工设计」到「数据驱动」的范式迁移，而这可能仅仅是个开端。「简单架构 + 数据驱动」的模式是否能如2D视觉和NLP领域般彻底重塑3D任务？

视觉重建作为所有3D任务的核心，VGGT的成功标志着3D视觉领域或许即将迎来一个全新的，基础模型的时代。

## 3. Conclusion

我自己仔细阅读完这些论文后，更多可能有自己想做的一些未来的idea：

1. 文2, 3, 4的三维语义分割任务，目前最优解还是使用SAM+CLIP这两个预训练模型，能否用多模态大模型代替这个模板，实现更快更准确的查询？

或者说目前的生态都是英文的，能不能用千问等国产大模型，实现一个中文的三维语义分割任务？

2.文5作为CVPR 2025 Best Paper，能否再次超越3DGS，实现为这些进一步的任务赋能

如果能做出来一个VGST+SAM+CLIP实现开放语义查询，那应该也是很受欢迎引用量很高的，solid的工作

我对三维视觉这类的工作非常感兴趣，主要是：

我自己对三维视觉的感觉是，背后的图形学原理是有点难以理解，但是做出来的东西可以进行交互，而且下游应用到机器人与无人驾驶等领域非常明确，感觉做的工作很有意义，看着一组二维图像作为输入能得到一个可以进行交互的三维场景，感觉很神奇很奇妙。

另外我觉得，这个领域未来也与具身智能等风向标息息相关，我也十分相信，具身智能可以带动大模型作为风口的同时，也成为下一个科研与产业的风口。