完全显式建模

1.Plenoctree（略读了解方法即可）： Yu A, Li R, Tancik M, et al. Plenoctrees for real-time rendering of neural radiance fields[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021: 5752-5761.

2.**Plenoxel（精读）: Fridovich-Keil S, Yu A, Tancik M, et al. Plenoxels: Radiance fields without neural networks[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 5501-5510.**

理解球面谐波函数如何建模场景。

第一篇：Plenoctrees for real-time rendering of neural radiance fields

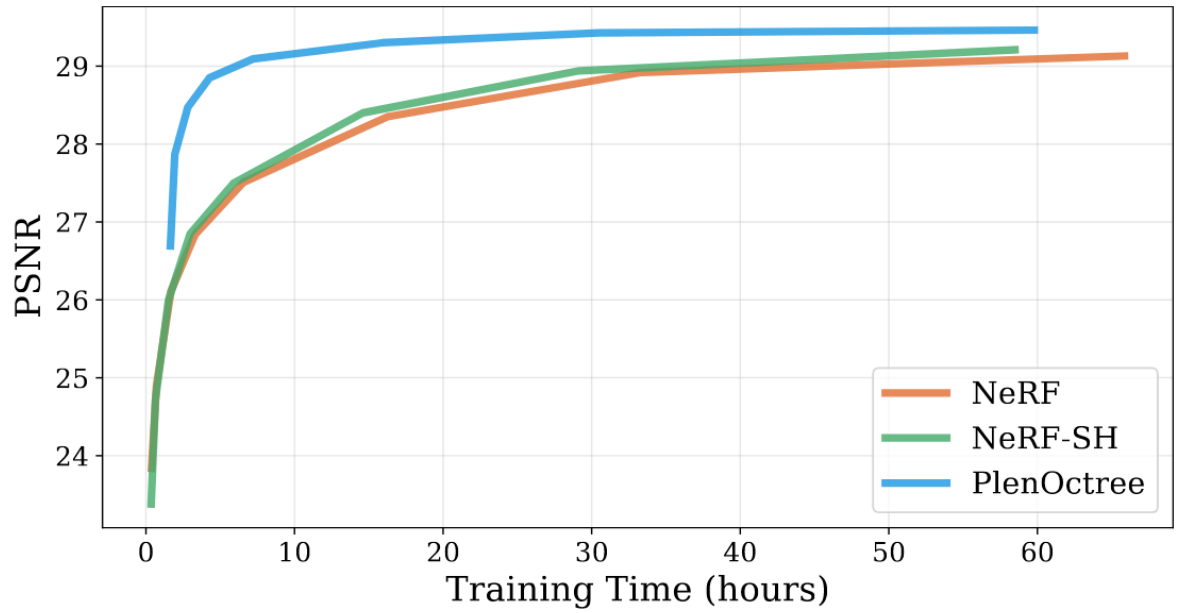
1.总结：

1）创新点?

1.引入球谐函数来查询渲染RGB值。

2.提出了PlenOctree，一种稀疏八叉树的数据结构，用于存储密度和SH系数。

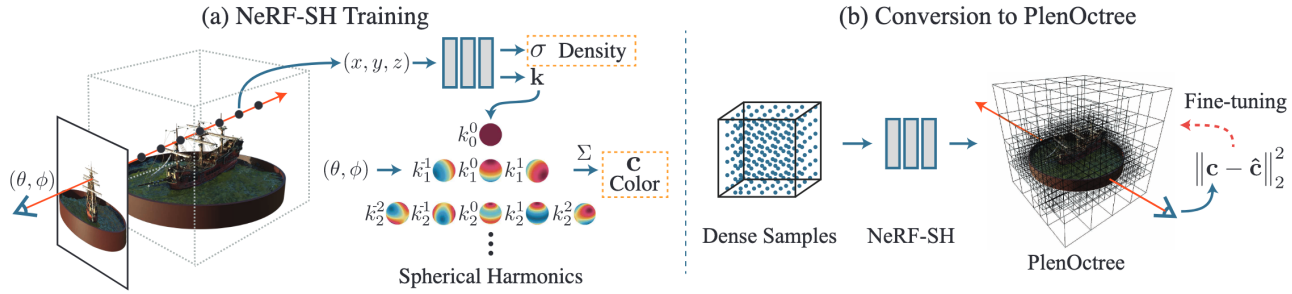
2）效果与局限性?



如图，在渲染出同质量的情况下，PlenOctree用时更小，整体训练质量也更好一些。

缺点就是非常吃内存，没有被压缩的PlenOctree占内存非常之大。

2.核心method：



提出了一种通过训练改进的 NeRF 模型 (NeRF-SH) 并将其转换为 PlenOctree 来快速渲染 NeRF 的方法，这是一种捕获视图相关效果的八叉树。

a) NeRF-SH 模型使用相同的优化过程和 NeRF中提出的体绘制方法。然而网络不是直接预测 RGB 颜色 c，而是预测球谐系数 k。颜色 c 是通过对相应射线方向 (θ, φ) 评估的加权球谐基求和来计算的。球谐函数使表示能够对视相关的外观进行建模。橙色框中的值用于体渲染。

SH：球谐函数。本文中使用球谐函数进行建模位置x处的光照情况。通俗的讲，球谐函数可以表示为多个基底函数相加的形式，取出所有基底函数的系数即为【球谐函数的系数】。这里就是和nerf不同的之处，由神经网络合成出的不是RGB颜色，而是这些球谐函数的系数。拟合出这些系数，就得到了x处的球谐函数，在渲染阶段就可以通过查询某个角度的球谐函数的结果，来得到该位置在某个方向上的RGB。

b) 为了构建 PlenOctree，我们在目标对象周围的体积中密集采样 NeRF-SH 模型，并将密度和 SH 系数制成表格。我们可以直接使用训练图像进一步优化 PlenOctree 以提高其质量。

第二篇：Plenoxels: Radiance fields without neural networks

1.总结：

1）.解决了什么问题?

1.本文提出**神经辐射场的关键元素不是神经网络，而是可微的体积渲染器**这一观点。

2.引入了**Plenoxels** (plenoptic voxels)：使用3D稀疏体素网格存储球谐函数来表示场景，这种表示可以从梯度方法和正则化优化来校准图像，从而不需要任何神经网络。

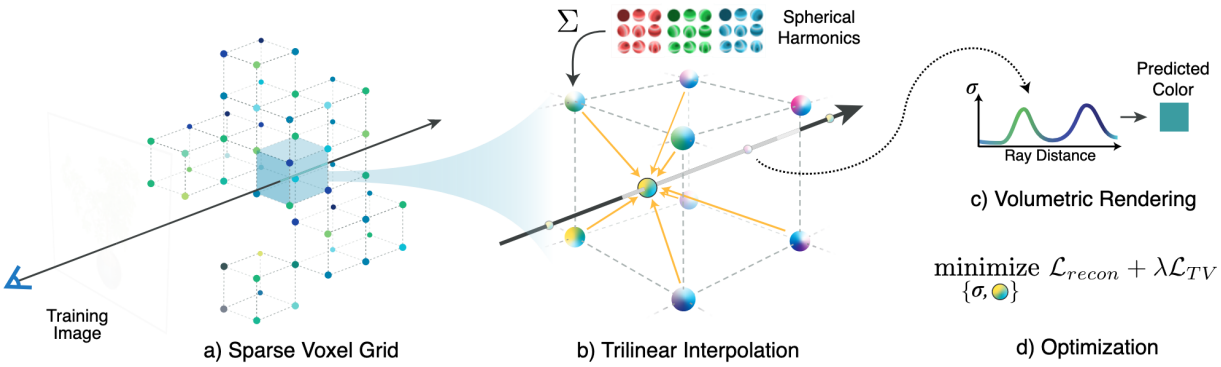
2）.效果与局限性?

1.效果：和SOTA方法接近，且快了一两个数量级。

	PSNR ↑	SSIM ↑	LPIPS ↓	Train Time
Ours	31.71	**0.958**	**0.049**	**11 mins**
NV [22]	26.05	0.893	0.160	>1 day
JAXNeRF [8, 28]	**31.85**	0.954	0.072	1.45 days
Ours	26.29	**0.839**	**0.210**	**24 mins**
LLFF [27]	24.13	0.798	0.212	—*
JAXNeRF [8, 28]	**26.71**	0.820	0.235	1.62 days
Ours	20.40	**0.696**	**0.420**	**27 mins**
NeRF++ [60]	**20.49**	0.648	0.478	~4 days

2.局限性：有伪影，且每个场景的超参数都得调整。

2.核心：具体实现过程



给定一组物体或场景的图像，重建一个(a)稀疏体素(“Plenoxel”)网格，每个体素具有密度和球形谐波系数。为了渲染光线，(b)通过相邻体素系数的三线性插值计算每个样本点的颜色和不透明度。使用(c)可微体渲染来整合这些样本的颜色和不透明度。然后，可以使用相对于训练图像的标准MSE重建损失，以及总变异正则化器来优化体素系数。

2.1 体素网格与球面谐波

本文提到存储方式与plenOctrees相似，只不过数据结构有所差异，plenOctrees采用八叉树，本文而是采用**体素网格**。

体素网格使用了密集的3D索引数组，指针指向一个单独的数据数组，数据数组只包含所占用的体素的值，每个被占用的体素存储每个颜色通道的标量透明度σ和球形谐波SH系数向量。

球面谐波形成了定义在球面上的函数的正交基，低次谐波编码平滑(更朗伯)的颜色变化，高次谐波编码更高频率(更镜面)的效果，样本c_i的颜色只是每个颜色通道的这些谐波基函数的和，由相应的优化系数加权，并在适当的观看方向评估。

作者在本文使用2次谐波，每个颜色通道需要9个系数，所以每个体素网格需要27个系数。

2.2 插值：三线性插值

三线性插值：用于在一个3D的立方体中，通过给定顶点的数值然后计算立方体中其他点的数值的线性插值方法。

插值通过表示颜色和不透明度的子体素变化来提高有效分辨率，插值产生连续函数逼近，这是成功优化的关键，其中三线性插值在学习率变化方面更稳定。作者对比了邻近插值和三线性插值方法下的PSNR，确实三线性插值表现得更好一些。

2.3 coarse to fine

用粗到精的策略来实现高分辨率，从分辨率较低的密集网格开始，优化、删除不必要的体素，通过在每个维度中将每个体素细分为两半来细化剩余的体素，并继续优化。

此处是跟上2.2提到的三线性插值结合起来一起成功优化的关键。举例：从分辨率256^3 上采样到 512^3时，先将每个体素网格细分之后，使用三线性插值来初始化新增的网格值，接着判断是否需要会被修剪（设置最大权重为阈值）。

2.4 优化

通过渲染像素颜色的均方误差(MSE)与总变化(TV)正则化 优化体素不透明度和球面调和系数