NeRF —— Neural Radiance Fields

研究进展，文献阅读，试验分析，方向感悟

NeRF (Neural Radiance Fields，神经辐射场) **，是一种用于三维场景重建的深度学习方法。它的主要思想是：通过输入（唯一）一组具有已知相机姿势的图像**，捕捉场景中每个点的辐射量，训练神经网络，将场景重现为用于视图合成的神经辐射场，从而实现对三维场景的高质量建模。

注意：NeRF 不是直接恢复整个 3D 场景几何形状，而是生成一种称为 **“辐射场”** 的体积表示，它能够为相关 3D 空间中的每个点创建**颜色和密度**。虽然它可以捕捉场景的复杂光学属性，但这些信息通常是直接嵌入到渲染的2D图像中，而不能直接作为独立的贴图保存。

这属于反渲染Inverse Rendering技术。另外，三维重建也是广义上的反渲染，但核心是重建几何，不注重appearance（材质与光线）。

1 整体思路

- 首先，将**静态**场景表示为一个**连续的 5D 函数**，该函数输出空间中每个点 (x, y, z) 在每个方向 $(θ, φ)$ 发射的辐射。然后定义密度：每个点的密度控制在该点通过的光线积累了多少辐射，其作用类似于微分不透明度。

- 接着，NeRF通过优化一个没有任何卷积层的**深度全连接神经网络**（或称 多层感知器MLP），通过从单个5D坐标 $(x, y, z, θ, φ)$ 回归到点的与视图相关的RGB颜色和体积密度 $(R, G, B, σ)$。

优化过程中，使用**体积渲染**的技术，沿着光线积累此场景表示的样本，以便从任何视点渲染场景。为了从特定的角度来渲染此神经辐射场 （NeRF），作者：

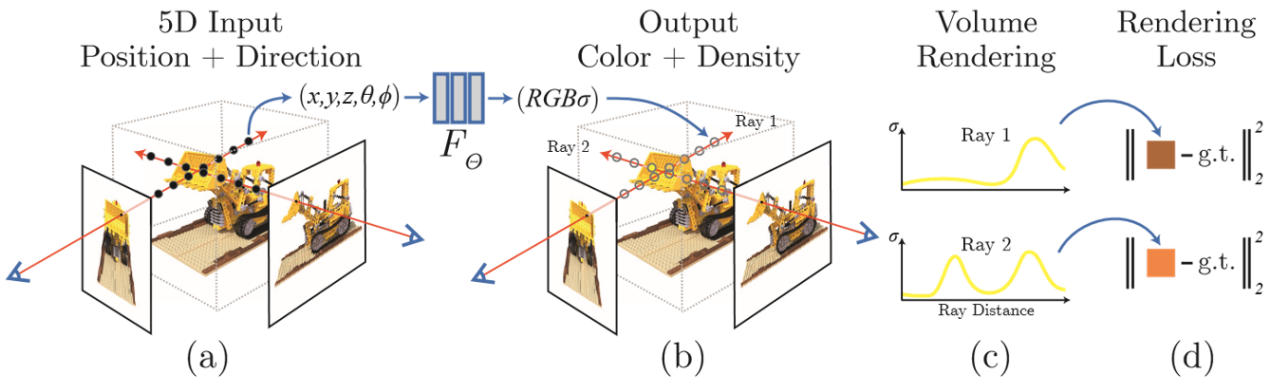
1. 在场景中移动相机光线以生成一组采样的 3D 点
2. 使用这些点及其相应的 2D 观看方向作为神经网络的输入，以生成一组颜色和密度的输出
3. 使用经典的体积渲染技术将这些颜色和密度累积到 2D 图像中。

Volume Rendering：在体积渲染中，每个体素（体积像素）都有一个关联的属性，比如密度、颜色、透明度等。渲染算法通过考虑这些属性以及光线在体积中的传播，将3D数据（通常是3D标量场）渲染为2D图像。

由于这个过程是自然可微的，作者使用简单的**梯度下降**来优化这个模型，具体优化目标为**最小化每个观察到的图像与渲染的相应视图之间的误差。**

2 神经辐射场场景表示

将一个连续场景表示为 5D 向量值函数，其输入是 3D 位置和 2D 观看方向 $x = (x, y, z) , d = (θ, φ)$，其输出是发射颜色和体积密度 $c = (R, G, B), σ$。



1. 将方向表示为 3D 笛卡尔单位向量 d
2. 使用 MLP 网络 F_{Θ} ：$(x，d) → (c，σ)$ 近似这种连续的 5D 场景表示，并优化其权重 $Θ$，以从每个输入 5D 坐标映射到其相应的体积密度和方向发射颜色。
3. 为使多视图一致，限制网络仅将体积密度 $σ$ 预测为位置 x 的函数，同时允许将 RGB 颜色 c 预测为位置和观看方向的函数。
4. 神经网络架构：MLP F_{Θ} 首先处理具有 8 个全连接层的输入 3D 坐标 x（使用 ReLU 激活和每层 256 个通道），然后输出 σ 和 256 维特征向量。然后，该特征向量与相机光线的观看方向连接，并传递到一个额外的全连接层（使用 ReLU 激活和 128 个通道），该层输出与视图相关的 RGB 颜色。

3 使用辐射场进行体积渲染

5D 神经辐射场将场景表示为空间中**任何点**的**体积密度**和**定向发射辐射**。作者使用*经典体积渲染的原理* 来渲染穿过场景的任何光线的颜色

> Kajiya, J.T., Herzen, B.P.V.: Ray tracing volume densities. Computer Graphics (SIGGRAPH) (1984)

预期颜色 $C(r)$ 的积分形式：

$$ C(\mathbf{r})=\int_{t_n}^{t_f}T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t),\mathbf{d})dt,\text{where}:T(t)=\exp\left(-\int_{t_n}^t\sigma(\mathbf{r}(s))ds\right) $$

- $C(r)$：表示在点 r 处的颜色。
- t_n和 t_f：表示路径的起始点和终止点。
- $σ(x)$: 体积密度，可以解释为光线终止于位置 x 处的无穷小粒子的微分概率。 -- `alpha = raw2alpha(raw[..., 3] + noise, dists)`
- $T(t)$：透射率，表示从路径起始点 t_n 到当前点 t 的透射率，通过对路径上的吸收函数 $σ(r(s))$ 进行积分来计算。即，射线从tn传播到t而不击中任何其他粒子的概率。 -- `weights`
- $\sigma(r(t))$：表示在路径上的点 $\mathbf{r}(t)r(t)$ 处的体积密度（opacity）。这由网络预测并通过函数 `raw2alpha` 转换得到，表示为 `alpha`。
- ${c}(r(t),d)$：表示在路径上的点 $r(t)$ 处的颜色，其中 d 是观察方向。
- $r(t) = o + td$: 相机光线路径，其中 o 是光线原点。 -- `pts = rays_o[..., None, :] + rays_d[..., None, :] * z_vals[..., :, None]`
- d: 光线方向。
- t: 一个沿着光线方向的距离参数。

从连续神经辐射场渲染视图需要估算这个积分 $C(r)$，以获得通过所需虚拟摄像机的每个像素追踪的相机光线。

尽管NeRF使用一组离散的样本来估计积分，但因为分层采样使MLP 在优化过程中在连续位置上被计算评价，所以仍然能够表示一个连续的场景。使用这些样本和 *早期研究* 在体积渲染评论中讨论的正交规则来估计 $C(r)$：

$$ \hat{C}(\mathbf{r})=\sum_{i=1}^NT_i(1-\exp(-\sigma_i\delta_i))\mathbf{c}*i*,:*text{where}:T_i=\exp\left(-\sum*{j=1}^{i-1}\sigma_j\delta_j\right) $$

- $\hat{C}(\mathbf{r})$ ：这是沿着光线 (\mathbf{r}) 累积的估计颜色，通过对沿光线的离散点求和来近似。
- σ_i：表示第 i 个采样点的密度。 -- $\sigma(r(t))$
- δ_i：表示第 i 个采样点和前一个采样点之间的距离。 -- 积分步长
- c_i：表示第 i 个采样点的颜色。 -- ${c}(r(t),d)$
- T_i：是从光线起点到第 i 个采样点之间的累积透明度，计算方法是对之前所有采样点的 $\sigma_j\delta_j$ 求和再取负指数。

代码见 `render_rays()` 函数，输出渲染结果 `ret = {'rgb_map': rgb_map, 'disp_map': disp_map, 'acc_map': acc_map}` RGB 映射、视差图、累积不透明度

4 改良神经辐射场

上述组件已经将场景和建模篇为神经辐射场，但不能达到最高质量，因而进行改良 —— 输入坐标的**位置编码**，有助于MLP表示高频函数; **分层采样**程序，使齐能够有效地对这种高频表示进行采样.

4.1 位置编码

位置编码 的名称来源于 Transformer 架构，与文中方法类似

研究表明，深度网络偏向于学习**低频函数**：让网络 F_{Θ} 直接对 $(x, y, z, θ, φ)$ 输入坐标进行操作会导致渲染在表示颜色和几何的高频变化方面表现不佳。

在将输入传递到网络之前，使用高频函数将输入映射到更高维度的空间，可以更好地拟合包含高频变化的数据。

> 频谱偏差（`spectral bias`）：频谱偏差是指神经网络倾向于更容易地学习平滑、低频的函数，而在学习高频变化（例如细节丰富的纹理或尖锐的边缘）时存在困难。通过将输入数据映射到更高维的空间，`NeRF` 能够将这些高频变化编码为网络更容易学习的形式。
> 常用正弦函数（`sin`）或余弦函数（`cos`）作为高频函数。这些函数可以产生周期性的模式，这对于捕获和表示数据中的重复或周期性模式非常有用。例如，在图像或三维形状中，可能存在许多重复或周期性的模式，如纹理、颜色变化等。通过使用高频函数，我们可以更好地捕获和表示这些模式，从而提高网络的性能和准确性。

具体公式如下：

$$F_{\Theta}=F_{\Theta}^{\prime}\circ\gamma $$

γ 是 R 到更高维 R^{2L} 的映射；$F_{\Theta}^{\prime}$ 并且仍然只是一个常规的 MLP。编码函数如下：

$$\gamma(p)=\left(\sin\left(2^0\pi p\right),\cos\left(2^0\pi p\right),:\cdots,\sin\left(2^{L-1}\pi p\right),:\cos\left(2^{L-1}\pi p\right)\right)$$

4.2 分层体积采样

无差别密集采样是低效的，因自由空间和被遮挡区域无需重复采样 —— 可以**通过按比例分配样本对最终渲染的预期效果来提高渲染效率。**

分层，即使用两个网络来表示场景，同时优化，一个粗略，一个精细。大体步骤如下：

1. 首先使用分层采样对一组 Nc 位置进行采样，并评估这些位置的“粗略”网络，如方程 2 和 3 中所述。给定这个“粗略”网络的输出。
2. 对每条射线沿线的点进行更密集的采样，其中样本偏向体积的相关部分，即“精细”部分。

以下是详细步骤，这个过程将更多的样本分配给期望包含可见内容的区域：

1. 首先将方程3中来自“粗糙”网络的 alpha 混合颜色 $\hat{C}_c(r)$ 重写为沿着光线采样的所有颜色c_i 的加权和：

$$\hat{C}*c(\mathbf{r})*=\sum\limits{i=1}^{N_c}w_ic_i:,\quad w_i=T_i(1-\exp(-\sigma_i\delta_i))$$

2. 将这些权重归一化为 $\hat{w_i}:=:w_i\Big/\sum_{j=1}^{N_c}w_j$ ，这是会在光线上产生分段常数的概率密度函数（PDF）

3. 使用逆变换采样从这个分布中抽取第二组 N_f 个位置

4. 得到细致网络：在所有 N_c+N_f 个样本的基础上，使用 之前的估计方程 计算光线的最终渲染颜色 $\hat{C}_f(r)$

4.3 NeRF流程中的其他细节

为每个场景优化一个神经网络，只需要：一个包含场景的RGB图像、相应的相机姿态、内参数、场景边界数据集。

合成数据可直接用上述参数；真实数据要用COLMAP运动结构包来估计参数。

在每次优化迭代中，作者从数据集中所有像素的集合中随机采样一批相机光线，然后按照第 4.2 节中描述的分层采样，从粗网络查询 N_c 样本，从精细网络查询 $N_c + N_f$ 样本。然后，作者使用第 3 节中描述的体积渲染过程从两组样本中渲染每条光线的颜色。

$Loss$ 为（粗渲染-精渲染）与（精渲染-真实RGB）的**总平方误差：**

$$\mathcal{L}=\sum_{\mathbf{r}\in\mathcal{R}}\left[\left|\hat{C}_c(\mathbf{r})-C(\mathbf{r})\right|_2^2+\left|\hat{C}_f(\mathbf{r})-C(\mathbf{r})\right|_2^2\right]$$

where R is 光线集合 in each batch, and $C(r)$, $\hat{C}_c(r)$, and $\hat{C}_f(r)$ are the ground truth, coarse volume predicted, and fine volume predicted RGB colors for ray r respectively.

5 总结

优点

- 高质量：可以创建复杂场景的高质量 3D 重建，包括精细的表面细节和反射。（超越此前研究）
- 连续：NeRF 提供的模型是连续的，可以在任何方向合成视图，从而实现对象操作和渲染等应用程序。
- 无监督：可以在没有**明确监督**的情况下学习重建场景。
  NeRF 通过使用来自多个视角的照片作为输入，自行推断出三维场景的结构和外观，无需额外的场景结构或深度信息。这种方法利用了从不同视角拍摄的照片之间的几何和光照一致性，让模型自主学习这些几何关系，而不是依赖于传统的带标签的训练数据。
- 适用性广：包括室外场景、室内场景，甚至微观结构。

缺点

- 训练、渲染慢
- 仅能表示静态场景
- it bakes lighting，失去了动态光的灵活性，无法改变光的方向
- A trained NeRF 无法泛化
- NeRF训练的模型无法编辑纹理贴图
- 相机位置苛刻，需要的相片数量多，实际应用受到限制
- ……