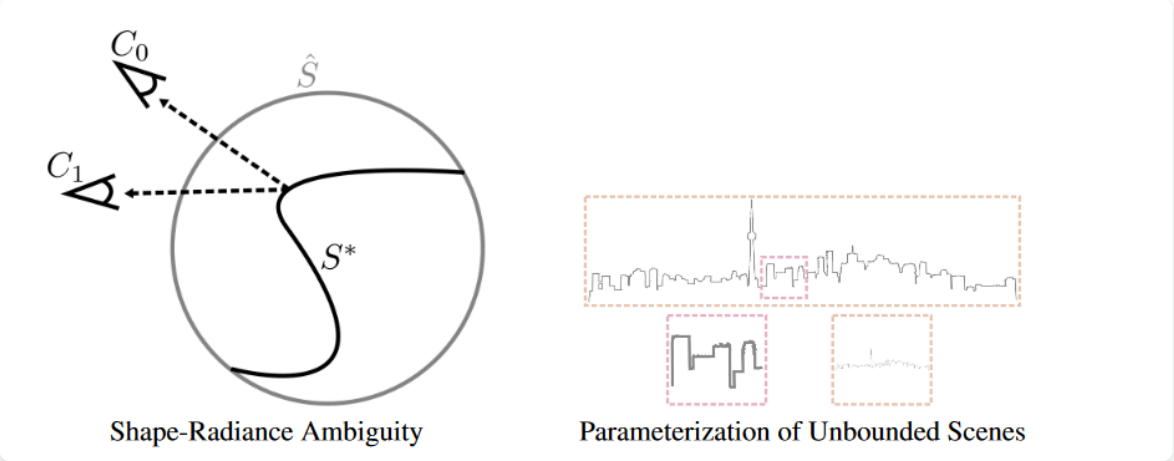
NeRF++

论文：《NeRF++: Analyzing and Improving Neural Radiance Fields》
地址：https://arxiv.org/abs/2010.07492
年份：arXiv preprint 2020

Introduction

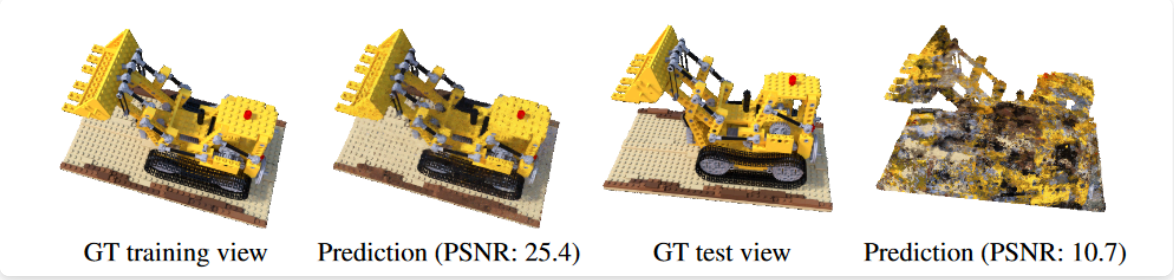
任务：新视角合成
技术贡献：
(1) 提出了原始 NeRF 中可能存在的 failure mode，在理论上分析了 NeRF 实际上能够避免重建失败的原因，并通过实验验证了提出的猜想。
(2) 针对原始 NeRF 对 360° 无边界场景重建效果差的问题，提出逆球面重参数化 (inverted sphere parameterization)，提升 NeRF 在无边界场景下的表现。



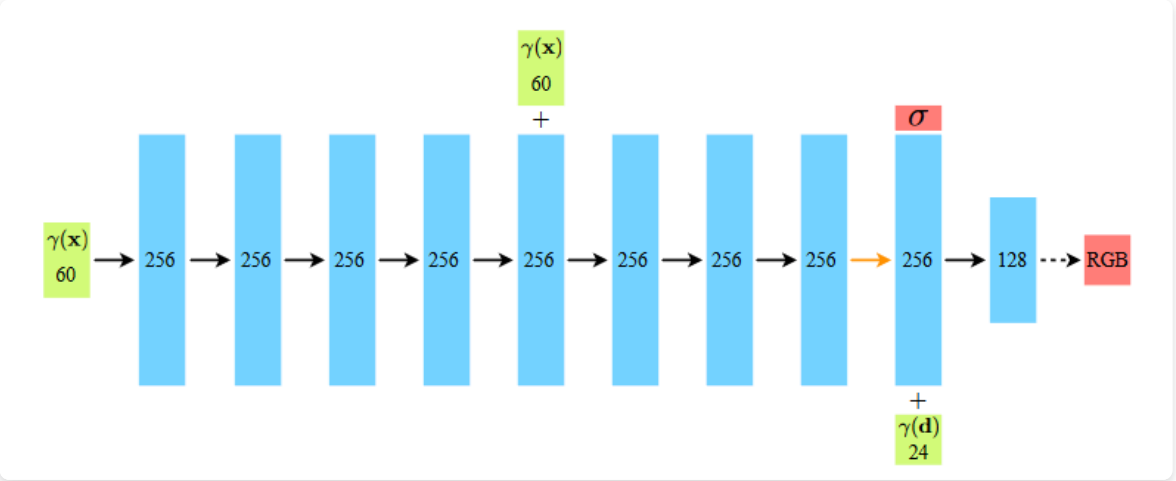
Shape-Radiance Ambiguity　　Parameterization of Unbounded Scenes

Shape-Radiance Ambiguity

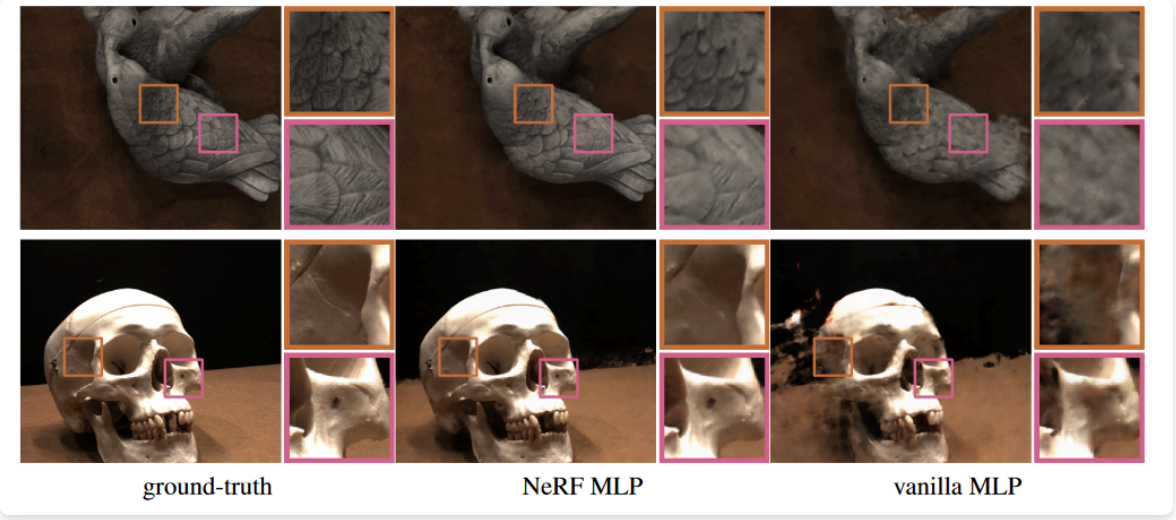
文章认为 NeRF 存在 shape-radiance ambiguity 的情况，NeRF 只使用渲染图像与真实图像之间的像素颜色差异作为损失函数，而没有额外的正则项来约束影响渲染结果的 $\sigma(\mathbf{r}(t))$ 和 $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$。因此就会存在这样一种情况：NeRF 预测错误 $\sigma(\mathbf{r}(t))$ 的同时，也预测错误的 $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$，两个错误的值反而能得到正确的像素颜色。也就是说模型能够正确拟合给定的训练图像，但无法渲染出正确的新视角图像。如上图左侧所示，模型可能学习到错误的物体表面 $\hat{S}$，但由于颜色是 view-dependent 的，所以仍然可以渲染出正确的颜色。文章做了一个实验证明了这种情况的存在，设置一个单位球，其表面的体密度固定为 1，其余位置全为 0，只对颜色进行训练，结果如下图所示。


GT training view　Prediction (PSNR: 25.4)　GT test view　Prediction (PSNR: 10.7)

那么原始 NeRF 为什么能够避免这种情况，文章分析了以下 2 个原因：
(1) 错误的表面导致 $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$ 必须是一个高频的函数，因为同一个位置在不同视角下的颜色变化会相当剧烈 (想象第一张图左边的情况)。如果 $\sigma(\mathbf{r}(t))$ 是正确的，那么 $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$ 会比较光滑，因为大部分区域是漫反射，只有少数区域存在镜面反射。而且 MLP 相对来说是不好表征高频函数。
(2) 原始 NeRF 设计的 MLP 结构也隐含了 BRDF 是光滑的这个先验信息。参照原始 NeRF 的网络结构，观测视角 $\mathbf{d}$ 的信息是在 MLP 的末端注入的，只有少量的 MLP 参数用于建模 view-dependent 的效果。同时 $\mathbf{d}$ 的位置编码 γ 阶数为 4，而 $\mathbf{x}$ 的位置编码阶数为 10，$\gamma(\mathbf{d})$ 只包含了低频信息。因此 $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$ 的 view-dependent 的效果是比较受限的。



文章也通过实验验证了以上猜想，修改原始 NeRF 的网络，将 $\gamma(\mathbf{d})$ 从一开始就输入到网络中，且将位置编码阶数升为 10，可以看出修改后的模型无法正确合成新视角图像。



Inverted Sphere Parameterization

原始 NeRF 适合处理有界场景以及 forward facing 的无界场景，在 360° 无界场景下，效果是不好的。

(a) bounding volume for the truck only　　(b) bounding volume for the entire scene

对于 360° 无界场景，如果只采样物体范围内的点，就会导致背景点采样不够而模糊 (左图)；如果对整个场景采样，则会导致近处的点采样不够而模糊 (右图)。

文章采用前景和背景分开处理的方法来解决这个问题。针对 360° 无界场景，首先将所有相机归一化到一个单位球内，这样物体会位于这个单位球内，背景部分在单位球外。对于单位球内部，使用正常的 NeRF 渲染方式；对于外部，则使用逆球面重参数化的方法进行处理。

对于一个 3D 点 (x, y, z)，其与原点距离为 $r = \sqrt{x^2 + y^2 + z^2} > 1$，将其重参数化为四元组 $(x', y', z', 1/r), x'^2 + y'^2 + z'^2 = 1$，这样的话 (x', y', z') 是球面上的点，其方向与原先点相同。$1/r \in [0, 1]$ 就可以表示距离为 1 到无穷的所有点。

这样做有两个好处：

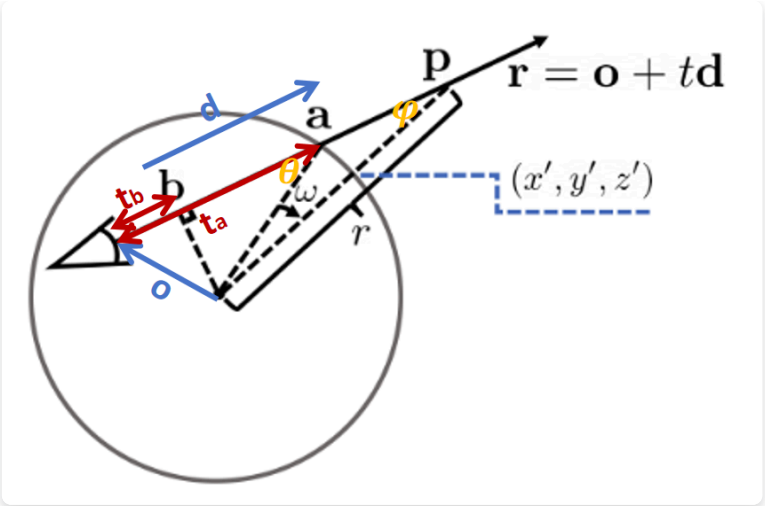
(1) 保证了数值稳定性，因为将 $[1, \infty]$ 变换到了 $[0, 1]$ 的范围；

(2) 具有距离越远采样数越少的性质，因为如果对 $1/r$ 进行均匀采样的话，采样点会集中于 $r = 1$ 这一端。

采用以上方法后，体渲染公式就可以拆分为前景和背景两部分，如下图所示。对于 (i)(ii)，使用正常的 NeRF 渲染流程即可，对于 (iii)，实际上不再是对 t 进行积分，且 σ 和 $\mathbf{c}$ 都是由另一个网络预测，具体计算会在后续提到。

$$\mathbf{C}(\mathbf{r}) = \underbrace{\int_{t=0}^{t'} \sigma(\mathbf{o}+t\mathbf{d}) \cdot \mathbf{c}(\mathbf{o}+t\mathbf{d}, \mathbf{d}) \cdot e^{-\int_{s=0}^{t} \sigma(\mathbf{o}+s\mathbf{d}) ds} dt}_{\text{(i)}} + \underbrace{e^{-\int_{s=0}^{t'} \sigma(\mathbf{o}+s\mathbf{d}) ds}}_{\text{(ii)}} \cdot \underbrace{\int_{t=t'}^{\infty} \sigma(\mathbf{o}+t\mathbf{d}) \cdot \mathbf{c}(\mathbf{o}+t\mathbf{d}, \mathbf{d}) \cdot e^{-\int_{s=t'}^{t} \sigma(\mathbf{o}+s\mathbf{d}) ds} dt}_{\text{(iii)}}. \quad (4)$$

具体来说，对于光线 $\mathbf{r} = \mathbf{o} + t\mathbf{d}$，对于前景部分，只需要根据 near 和 far 平面采样一系列 t，带入光线得到 3d 坐标，进行正常体渲染。对于背景部分，这一部分位于单位球外，我们进行了逆球面重参数化，所以不再是采样 t 得到坐标的计算方式，而是采样 $1/r$ 计算对应四元组中的 x', y', z'，然后将四元组 $(x', y', z', 1/r)$ 以及 $\mathbf{d}$ 作为网络的输入，去预测 σ 和 $\mathbf{c}$，且此时相当于是对 $1/r$ 做积分。因此，现在需要解决的问题就是如何通过 r 得到对应的 (x', y', z')。



如上图所示，首先定义：光线与单位球相交点为 $\mathbf{a}$，点 $\mathbf{b}$ 为光线在球中形成的弦的中点，我们要求 $\mathbf{p}$ 点对应的 (x', y', z')。思路是求出 $\mathbf{a}$ 的坐标，将其绕着轴 $\mathbf{b} \times \mathbf{d}$ 旋转 ω 角度，得到 (x', y', z')。

首先，根据 $\mathbf{d}^T(\mathbf{o} + t_b\mathbf{d}) = 0$ 解出 $t_b = -\mathbf{d}^T\mathbf{o}/\mathbf{d}^T\mathbf{d}$，然后关注 $\mathbf{a}, \mathbf{b}, \mathbf{a} - \mathbf{b}$ 形成的直角三角形，其模长分别为 $1, ||\mathbf{b}||, ||(t_a - t_b)\mathbf{d}||$，因此有 $t_a - t_b = \sqrt{1 - ||\mathbf{b}||^2}/||\mathbf{d}||$。然后根据 $\sin\phi = ||\mathbf{b}||/r, \sin\theta = ||\mathbf{b}||$ 就可以计算得到 ω，最后根据旋转公式就可以得到 (x', y', z')，然后将完整的四元组 $(x', y', z', 1/r)$ 作为网络的输入即可。(以上也是代码中 depth2pts_outside 函数的操作过程)

Experiments

文章的实验部分非常简单，在 T&T 数据集和 LF 数据集上进行训练，与原始 NeRF 进行比较，评价指标为 LPIPS，SSIM 和 PSNR。为了进行公平的比较，针对 NeRF，文章将场景进行了归一化，使所有相机位于半径为 $1/8$ 的球内，使得单位球能覆盖大部分的背景区域。由于 NeRF++ 分别对前景和背景进行采样，所以将 NeRF 的采样点数量提升一倍，使得两者的计算消耗大致相同。

	Truck			Train			M60			Playground		
	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR
NeRF	0.513	0.747	20.85	0.651	0.635	16.64	0.602	0.702	16.86	0.529	0.765	21.55
NeRF++	**0.298**	**0.823**	**22.77**	**0.523**	**0.672**	**17.17**	**0.435**	**0.738**	**17.88**	**0.391**	**0.799**	**22.37**
	Africa			Basket			Torch			Ship		
	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR	↓LPIPS	↑SSIM	↑PSNR
NeRF	0.217	0.894	26.16	0.377	0.805	20.83	0.347	0.811	22.81	0.372	0.801	23.24
NeRF++	**0.163**	**0.923**	**27.41**	**0.254**	**0.884**	**21.84**	**0.226**	**0.867**	**24.68**	**0.241**	**0.867**	**25.35**