NeRF

论文：《NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis》
地址：https://arxiv.org/abs/2003.08934
年份：ECCV 2020 (oral)

Introduction

任务：新视角合成
技术贡献：
(1) 使用 5D 的神经辐射场表示连续场景中的几何与材质 (全新的方法)，并参数化为 MLP。相较于显示表征，NeRF 属于隐式表征，能够减少存储成本。
(2) 使用基于传统体渲染的可微渲染方法，以及一种层级采样策略，能够有效优化隐式表征。
(3) 针对 5D 输入提出一种位置编码，使得模型能够表示场景中的高频信息。

Method

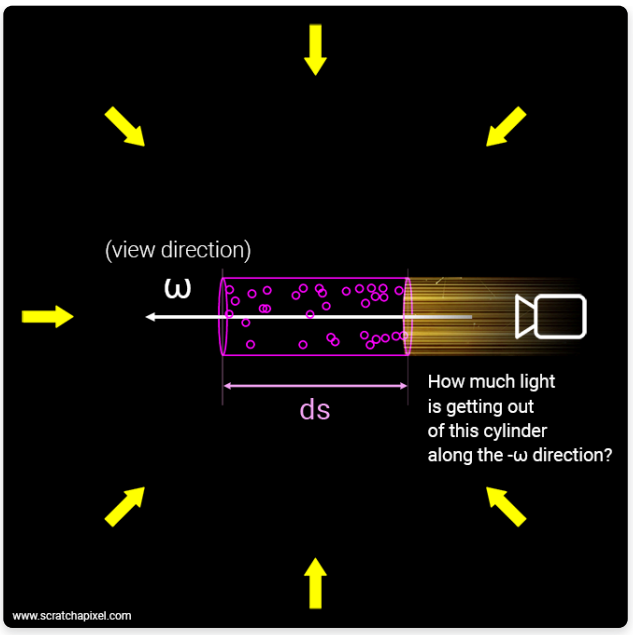
NeRF 用于一个场景下的新视角合成，输入是一个静态场景的一系列 RGB 图像，输出是不同视角下的图像。

具体来说，文章将场景建模为一个输入为 5D 函数，分别是 3D 的位置向量 $\mathbf{x}=(x,y,z)$ 和 2D 的观察视角 (θ,ϕ)，函数的输出是这个观察视角下的颜色 $\mathbf{c}=(r,g,b)$ 和体密度 σ。为了方便代码实现，2D 的观察视角使用 3D 的向量 $\mathbf{d}$ 表示。使用 MLP 来拟合这个 5D 函数表示的场景：$F_\Theta:(\mathbf{x},\mathbf{d})\rightarrow(\mathbf{c},\sigma)$。
接下来从两个方面展开：1) 体渲染的流程，如何借助上述 MLP 来完成图像渲染；2) 如何有效地优化 MLP，使其能准确表达场景信息。

体渲染

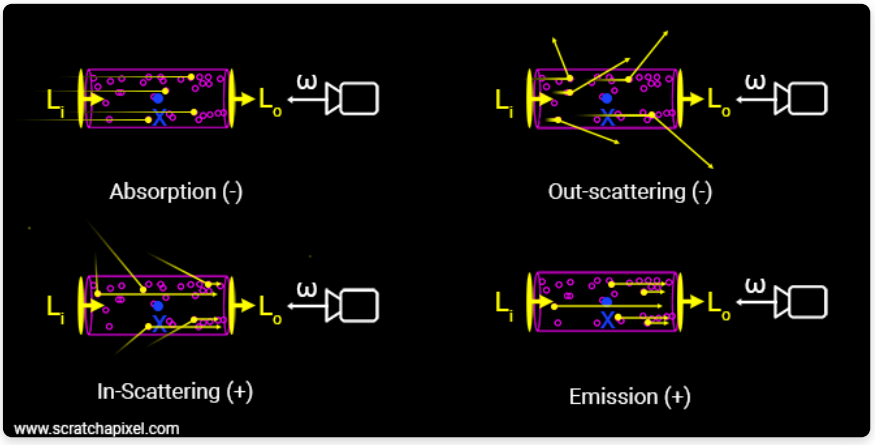
概述

体渲染把气体等物质抽象成一团飘忽不定的粒子群。光线在穿过这类物体时，其实就是光子在跟粒子发生碰撞的过程。
下图是体渲染建模的示意图。光沿直线方向穿过一堆粒子 (粉色部分)，如果能计算出每根光线从最开始发射，到最终打到成像平面上的辐射强度，我们就可以渲染出投影图像。而体渲染要做的，就是对这个过程进行建模。为了简化计算，我们就假设光子只跟它附近的粒子发生作用，这个范围就是图中圆柱体大小的区间。



记 L_i 为入射光强度，L_o 为出射光强度，ω 为观察方向。
体渲染把光子与粒子发生作用的过程，进一步细化为四种类型：

- 吸收 (Absorption)：部分光线被粒子吸收，辐射强度减弱；
- 外散射 (Out-scattering)：部分光线与粒子碰撞发生散射，方向发生变化，会减弱辐射强度；
- 内散射 (In-scattering)：其他方向的光子在撞到粒子后，可能和当前方向上的光子重合，从而增强辐射强度；
- 放射 (Emission)：粒子本身可能发光，比如气体加热到一定程度就会离子化，这会增强辐射强度；



吸收、散射和衰减系数

吸收系数 σ_a 表示光在介质中传播时每单位距离被吸收的概率密度。散射系数 σ_s 表示光在介质中传播时每单位距离发生散射的概率密度。衰减系数即两个系数之和：$\sigma_t = \sigma_a + \sigma_s$ 。

朗伯比尔定律的推导

对于一束从 x 出发，沿 $-\omega$ 方向传播的光线，其辐射强度的变化率为：

$$dL = -\sigma_t L(x, \omega)$$

如果介质是均匀的 (即 σ_t 为常数)，通过解常微分方程可以得到：

$$L(s) = e^{-\sigma_t s}$$

透射率可定义为：

$$T = \frac{L_o}{L_i} = e^{-\sigma_t s} = e^{\tau}$$

其中 τ 称为光学深度 (optical depth)。
对于非均匀的介质，有

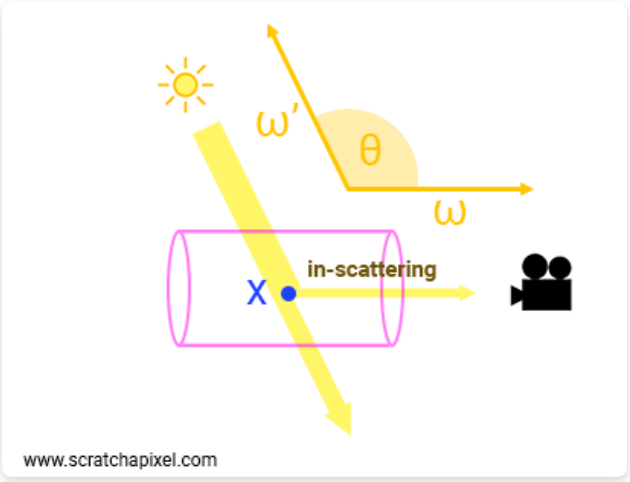
$$\tau = -\int_{s=0}^{d} \sigma_t(x_s) ds$$

那么朗伯比尔定律的一般形式可以写成：

$$T(d) = \exp(-\int_{s=0}^{d} \sigma_t(x_s) ds)$$

内散射

如下图所示，当有其他光源照射到当前光线时，来自该光源的部分光子会在穿过这一区域时发生散射，方向变为 $-\omega$，汇入当前光线中。通过相位函数 (phase function) 确定汇入光束的能量。



相位函数有两个性质：

- 对称性：$f_p(x, \omega', \omega) = f_p(x, \omega, \omega')$；
- 归一化为 1：$\int_{S^2} f_p(x, \omega, \omega') d\theta = 1$

对于均匀介质，有

$$f_p(x, \omega, \omega') = \frac{1}{4\pi}$$

辐射传输方程和体渲染方程

考虑散射与衰减，有：

$$\frac{dL(x,\omega)}{dx} = -\sigma_t L(x,\omega) + \sigma_s \int_{S^2} f_p(x,\omega,\omega') L(x,\omega') d\omega'$$

上式第二项可写为：

$$L_s(x,\omega) = \int_{S^2} f_p(x,\omega,\omega')L(x,\omega')d\omega'$$

这相当于解一个一阶非齐次线性常微分方程：

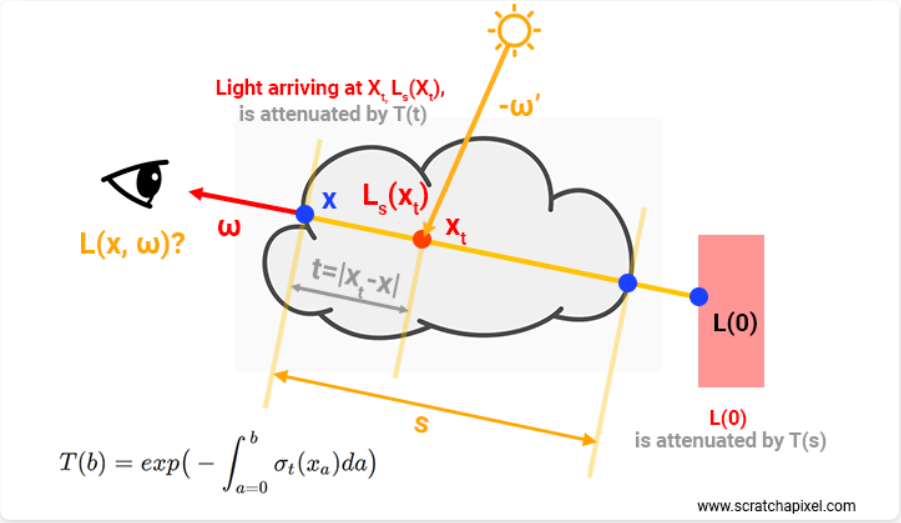
$$y' + p(x)y = q(x)$$

其解为

$$y(x) = \int_{t=0}^{x} q(t)e^{-\int_t^x p}dt + C_1 e^{-\int^x p}$$

令 $p(x) = \sigma_t(x), q(x) = \sigma_s(x)$，有

$$L(x,\omega) = \int_{t=0}^{x} \exp(-\int_{q=0}^{t} \sigma_t(x_q)dq)[\sigma_s(x_t)L_s(x_t)]dt + L(0)\exp(-\int_{t=0}^{s} \sigma_t(x_t)dt)$$



可以认为第一项是其他光源汇入当前光线中传播衰减后的强度，第二项是当前光束衰减后的辐射强度。
如果考虑放射，则有

$$L(x,\omega) = \int_{t=0}^{x} \exp(-\int_{q=0}^{t} \sigma_t(x_q)dq)[\sigma_s(x_t)L_s(x_t) + \sigma_a(x_t)L_e(x_t)]dt + L(0)\exp(-\int_{t=0}^{s} \sigma_t(x_t)dt)$$

令 $T(s) = \exp(-\int_{t=0}^{s} \sigma_t(x_t)dt)$，有

$$L(x,\omega) = \int_{t=0}^{x} T(t)[\sigma_s(x_t)L_s(x_t) + \sigma_a(x_t)L_e(x_t)]dt + T(s)L(0)$$

考虑 σ_s 与 σ_a 相同的情况，并令 $C = L_s + L_e$ 则有

$$L(s) = \int_{t=0}^{s} T(t)\sigma(t)C(t)dt + T(s)L(0)$$

这与原始 NeRF 的体渲染公式非常接近，只多了背景光一项。

应用

借助体渲染算法和 MLP，可以渲染出特定视角下的一张图片。考虑一张图片中的一个像素点，想要得到这个像素点对应的颜色，首先可以得到一条从相机发射出去的一条光线 $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$，其中 $\mathbf{o}$ 为相机所在位

置，$\mathbf{d}$ 表示相机与位于成像平面的像素点所连成的直线方向，通过取不同的 t，我们就可以得到这条光线上所有点的位置，对于 t 我们限制了取值范围，最小为 t_n，最大为 t_f，即我们只考虑近平面和远平面之间的点对颜色的影响。
给定空间位置 $\mathbf{r}(t)$ 和观察方向 $\mathbf{d}$，可以借助 MLP 得到该点的颜色 $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$，再套用体渲染公式，就可以得到这个像素点的颜色：

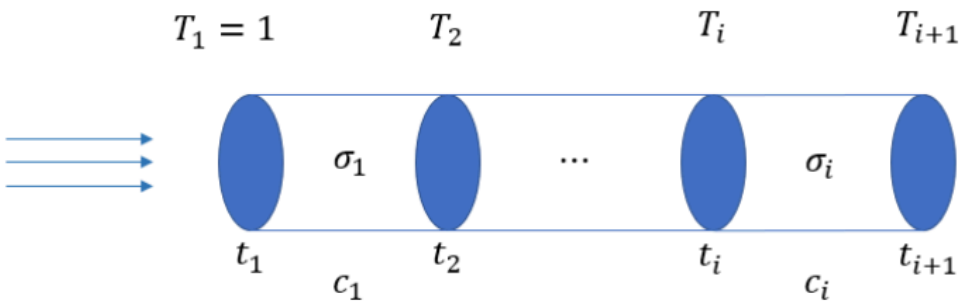
$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt$$

其中累计透射率 $T(t) = \exp(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds)$

离散化

我们还需要将以上公式进行离散化，以便计算机计算。具体推导可见下图：

实际上这里的离散形式并不是将积分直接离散化，而是通过把路径上的density和color设定为分段常数，如下图所示



于是先计算T，

$$\begin{aligned} T_i = T(t_i) &= \exp(-\int_0^{t_i} \sigma(t)dt) \\ &= \exp(-\sum_{j=1}^{i-1}\int_{t_j}^{t_{j+1}} \sigma(t)dt) \\ &= \exp(-\sum_{j=1}^{i-1} \sigma_j\delta_j), \end{aligned}$$

其中$\delta_i = t_{i+1} - t_i$。然后计算C，

$$\begin{aligned} C(\mathbf{r}) = \int_0^{t_{N+1}} T(t)\sigma(t)\mathbf{c}(t)dt &= \sum_{i=1}^{N}\int_{t_i}^{t_{i+1}} T(t)\sigma(t)\mathbf{c}(t)dt \\ &= \sum_{i=1}^{N}\int_{t_i}^{t_{i+1}} \exp(-\int_0^t \sigma(s)ds)\sigma_i\mathbf{c}_i dt \\ &= \sum_{i=1}^{N}\int_{t_i}^{t_{i+1}} T_i\exp(-\int_{t_i}^t \sigma(s)ds)\sigma_i\mathbf{c}_i dt \\ &= \sum_{i=1}^{N} T_i\sigma_i\mathbf{c}_i\int_{t_i}^{t_{i+1}} \exp(-\sigma_i(t-t_i))dt \\ &= \sum_{i=1}^{N} T_i\sigma_i\mathbf{c}_i \cdot -\frac{1}{\sigma_i}\exp(-\sigma_i(t-t_i))|_{t_i}^{t_{i+1}} \\ &= \sum_{i=1}^{N} T_i\mathbf{c}_i \cdot (1-\exp(-\sigma_i\delta_i)). \end{aligned}$$

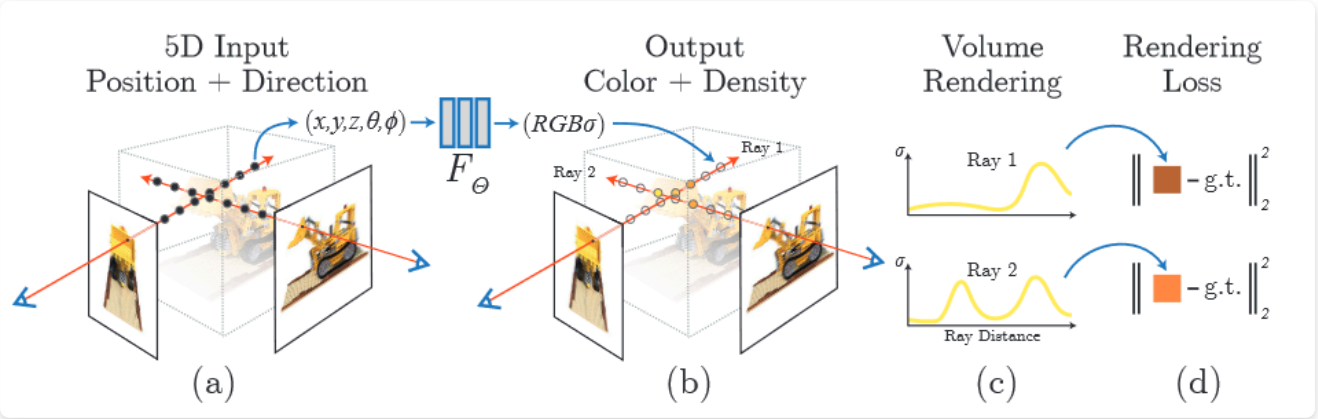
就跟原文中一样啦。

如果我们设 $\alpha_i = 1 - \exp(-\sigma_i\delta_i)$，那么 $T_i = \prod_{j=1}^{i-1}(1-\alpha_j)$，变为了传统的 alpha 混合：

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} \mathbf{c}_i\alpha_i\prod_{j=1}^{i-1}(1-\alpha_j)$$

优化网络

借助体渲染算法，我们可以渲染得到特定视角下的图片，通过设计损失函数，我们就可以通过梯度下降的方式优化 MLP，整体流程如下图所示。

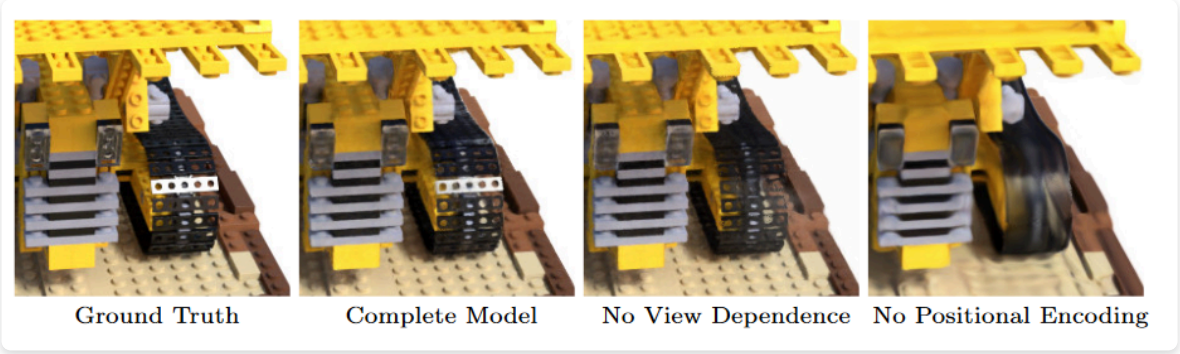


仅仅通过以上内容优化得到的 MLP 效果并不是很好，文章还提出了两个方法来帮助提升效果：位置编码和层级采样。

位置编码

关于使用位置编码的理由，是基于过去大家对MLP的观察。人们发现MLP的参数在学习的时候，会倾向于学习低频信息，所以直接对网络使用5D输入，效果会很糊。使用位置编码后，原本在低维接近的点，在高维会离得很远。换句话说，MLP在高维学到的低频信息，往往对低维来说不是低频的。
具体做法就是采用一个从 $\mathbb{R}$ 映射到 $\mathbb{R}^{2L}$ 的函数 γ：

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

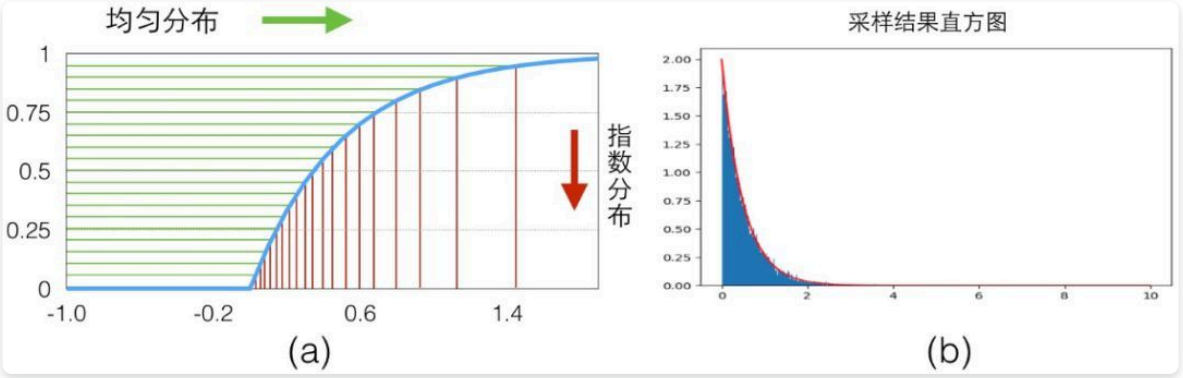


层级采样

对于渲染的图片来说，物体前面的空白区域以及被物体遮盖的区域对颜色的贡献是非常小的 (前者 σ 小，后者透射率 T 小)，因此我们希望从光线上采样计算积分结果时，采样点更集中于物体表面，其他区域的采样点可以少一些。使用层级采样策略来实现。
具体来说是采用了 "coarse to fine" 的策略，同时优化两个网络：一个 coarse 网络，一个 fine 网络。首先按照之前所说的，将光线分为一段一段，每段随机采样一个点，一共采样 N_c 个点，输入到 coarse 网络中，计算颜色。计算过程可以写为：

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i c_i, \; w_i = T_i(1 - \exp(-\sigma_i \delta_i)).$$

可以认为将颜色之外的项看作权重，权重越大对最终结果的影响越大。对权重进行归一化后 $\hat{w}_i = w_i / \sum_{j=1}^{N_c} w_j$，这相当于是一个分段的概率密度函数，每一段上为常数。有了概率密度函数我们就能知道对应的概率分布函数。可以使用逆变换采样 (inverse transform sampling) 来得到符合该概率分布的随机样本，这样就能在权重大的位置多采样一些点，提升渲染效率与质量。
先简单描述一下逆变换采样。对于一个随机变量 X，如果已知其 CDF 函数 $F_X(X)$，则用该函数将随机变量映射为 $Y = F_X(X)$，即可得到均匀分布 $Y \sim U(0,1)$。逆变换采样相当于前面过程的反向：对于服从均匀分布的 Y 使用函数 F_X^{-1} 将其映射为 $F_X^{-1}(Y)$，其与 X 有相同的分布。这样就可以通过对均匀分布采样，得到从分布 F_X 中生成的随机样本。

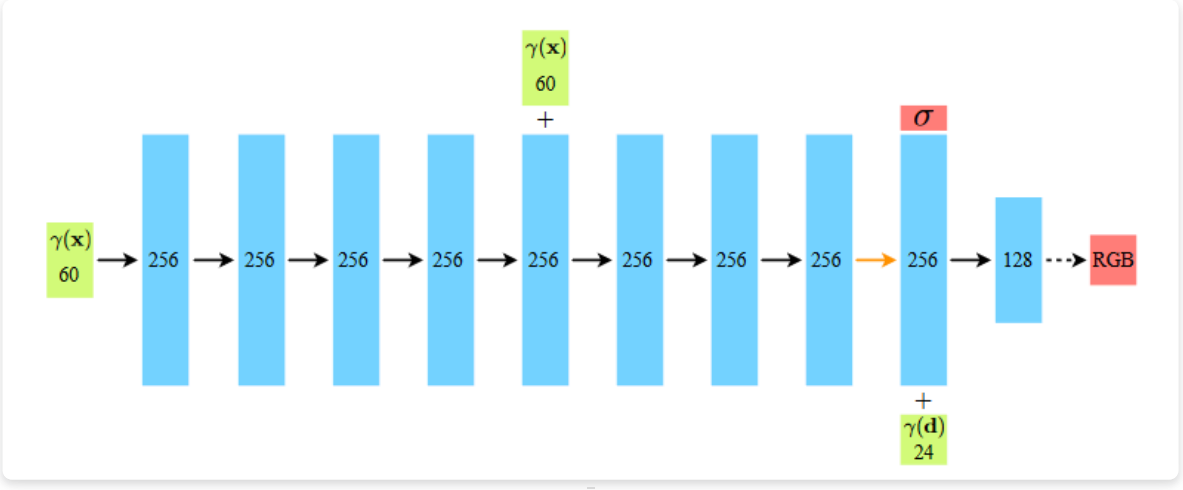


通过上述权重的 CDF，我们得到了 N_f 个满足对应分布的点，我们用 $N_c + N_f$ 这些点输入到 fine 网络中，计算得到颜色 $\hat{C}_f(\mathbf{r})$。这样就完成了在物体表面附近多采样点的目标。

Experiments

实现细节

网络结构



网络结构如上图所示，由于 σ 只与位置有关，RGB 与位置和观测方向有关，因此先通过 8 个全连接层只根据位置预测 σ，最后再通过一层引入方向信息，预测 RGB 值。对于位置编码，$\gamma(\mathbf{x})$ 的 L 为 10，$\gamma(\mathbf{d})$ 的 L 为 4。同时沿用了 DeepSDF 的设置，加入了一个 skip connection，将输入与第五层的向量做拼接作为后续层的输入。

消融实验

	Input	#Im.	L	(N_c , N_f)	PSNR↑	SSIM↑	LPIPS↓
1) No PE, VD, H	xyz	100	-	(256, -)	26.67	0.906	0.136
2) No Pos. Encoding	$xyz\theta\phi$	100	-	(64, 128)	28.77	0.924	0.108
3) No View Dependence	xyz	100	10	(64, 128)	27.66	0.925	0.117
4) No Hierarchical	$xyz\theta\phi$	100	10	(256, -)	30.06	0.938	0.109
5) Far Fewer Images	$xyz\theta\phi$	25	10	(64, 128)	27.78	0.925	0.107
6) Fewer Images	$xyz\theta\phi$	50	10	(64, 128)	29.79	0.940	0.096
7) Fewer Frequencies	$xyz\theta\phi$	100	5	(64, 128)	30.59	0.944	0.088
8) More Frequencies	$xyz\theta\phi$	100	15	(64, 128)	30.81	0.946	0.096
9) Complete Model	$xyz\theta\phi$	100	10	(64, 128)	**31.01**	**0.947**	**0.081**

可以看到位置编码和 view dependence 对指标的提升是较大的。增加位置编码的 L 在一定范围内有效，超过最大值之后效益就有限了。

Code

_load_data 输入：数据路径 basedir，缩放比例 factor；输出：poses，场景范围 bds，imgs
根据 basedir 从 poses_bounds.npy 读数据，大小为 $N \times 17$，前 15 个参数为 c2w 矩阵、图像宽高和相机焦距，后两个参数为场景范围。同时从图像文件夹读取 imgs，根据 factor 调整相机内参。
render_path_spiral 输入：c2w，up，半径 rads，focal，zrate，rots，N；输出：用于渲染的相机位姿
生成一条螺旋线上的相机位姿，用于渲染
load_llff_data：输入：