NERF-01:开山之作

Mildenhall B, Srinivasan P P, Tancik M, et al. Nerf: Representing scenes as neural radiance fields for view synthesis[J]. Communications of the ACM, 2021, 65(1): 99-106.

一、回答如下问题【带着问题去思考】：

1. 明确核心：解决什么问题（motivation）？用的什么方法（methods）？实验效果如何（performence）？

- 如何用NeRF来表示3D场景?
- 如何基于NeRF渲染出2D图像?
- 如何训练NeRF?

要理解NeRF是如何从一系列2D图像中学习到3D场景，又是如何渲染出2D图像。

2. 输入具体有什么？每个模块的具**体输入输出有什么**，每个**模块是如何工作的**？分为以下几个部分回答：

- 光线生成模块
- 光线点采样模块
- 粗网络（coarse network）
- 重要性采样模块
- 精网络（fine network）
- 渲染 - loss evaluation 模块

二、阅读论文过程中笔记：【method】

1.核心：

1）解决了什么问题：简要概括为用一个MLP神经网络去隐式地学习一个静态3D场景。为了训练网络，针对一个静态场景，需要提供大量相机参数已知的图片。基于这些图片训练好的神经网络，即可以从任意角度渲染出图片结果。

2）主要贡献：

i. 提出一种将复杂几何连续场景表示为5D神经辐射场的方法，将引入MLP网络模型来优化模型。

ii. 提出一种可微分的体渲染方法，并采用分层采样策略，来不断优化场景表示。

iii. 使用位置编码将每个输入的5D坐标映射到更高的维度空间，使模型能够成功地优化神经辐射场来表示高频场景（细节）内容。

3）实验效果如何：

Table 1. Our method quantitatively outperforms prior work on datasets of both synthetic and real images.

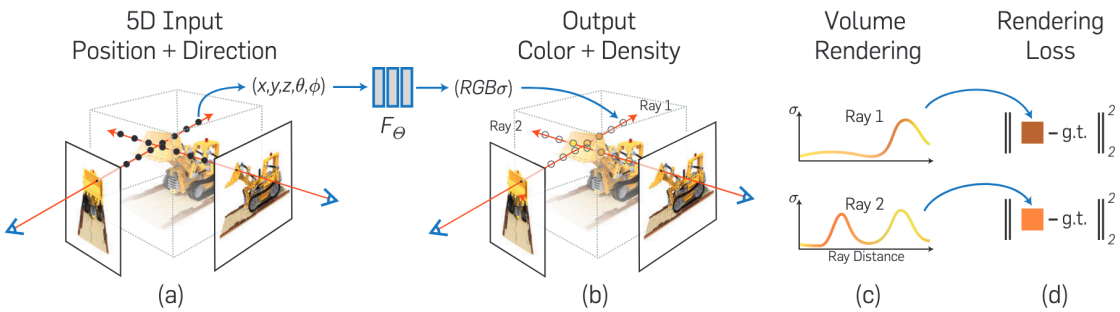
	Diffuse Synthetic 360°[20]			Realistic Synthetic 360°			Real ForwardFacing[12]		
Method	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**PSNR↑**	**SSIM↑**	**LPIPS↓**
SRN[21]	33.20	0.963	0.073	22.26	0.846	0.170	22.84	0.668	0.378
NV[8]	29.62	0.929	0.099	26.05	0.893	0.160	–	–	–
LLFF[12]	34.38	0.985	0.048	24.88	0.911	0.114	24.13	0.798	**0.212**
Ours	**40.15**	**0.991**	**0.023**	**31.01**	**0.947**	**0.081**	**26.50**	**0.811**	0.250

补充知识点：

PSNR（峰值信噪比）、SSIM（结构相似性指数）和LPIPS（感知相似性指标）都是用于评估图像或视频质量的指标：

1. PSNR（Peak Signal-to-Noise Ratio，峰值信噪比）：
   - 含义：PSNR是一种衡量图像质量的指标，它用于比较原始图像和经过处理（例如压缩或编码）后的图像之间的差异。**PSNR的值越高，表示图像质量越好。**
   - 衡量指标：PSNR的计算基于原始图像和处理后图像之间的均方误差（MSE），具体计算公式为：PSNR = 10 * log10((Max^2) / MSE)，其中Max表示像素值的最大可能值（通常是255），MSE是均方误差。
   - 高PSNR值表示较低的图像失真，但它不一定能准确地反映人眼对图像质量的主观感受，因为它忽略了人眼的感知差异。
2. SSIM（Structural Similarity Index，结构相似性指数）：
   - 含义：SSIM是一种用于测量图像的结构信息丢失程度的指标。它考虑了亮度、对比度和结构之间的差异，以更好地模拟人眼的感知。
   - 衡量指标：SSIM的计算涉及到原始图像和处理后图像的亮度、对比度和结构相似性的比较，以综合评估它们的相似性。SSIM的值在-1到1之间，**越接近1表示图像质量越高**。
   - SSIM通常比PSNR更能反映人眼的主观感受，因为它模拟了人眼的感知机制。
3. LPIPS（Learned Perceptual Image Patch Similarity，感知相似性指标）：
   - 含义：LPIPS是一种基于深度学习的感知相似性指标，用于衡量图像之间的感知差异。它使用卷积神经网络（CNN）学习了图像特征，并考虑了人眼的感知差异。
   - 衡量指标：LPIPS的计算基于处理前后图像的特征表示之间的距离，以评估它们的感知相似性。**LPIPS值越低表示图像越相似**，感知差异越小。
   - LPIPS更适合评估经过复杂处理的图像，如生成对抗网络（GAN）生成的图像，因为它考虑了感知差异而不仅仅是像素差异。

2.具体方法



模型的主要思路：

1. 让相机光线穿过场景，采样得到3D点（图2-a中黑点）
2. 将采样点和对应 view 的方向作为神经网络的输入得到输出的颜色和体密度（对应图2-b）
3. 使用体渲染技术将输出的颜色和体密度进行累积，映射到对应视图的图像中，然后与gt求误差（c 和 d）

具体实现细节：

1.neural radiance field scene representation

1）Input： 5D vector-valued function : 3D location $X = (x, y, z)$ and 2D viewing direction (θ, ϕ) .

2）Output: an emitted color $C = (r, g, b)$ and volume density σ [体积密度，或者说透明度].

3）使用3D笛卡尔单位向量 d 表示射线方向，so 使用MLP网络$F_\Theta : (X, d) \rightarrow (c, \sigma)$. 其中：

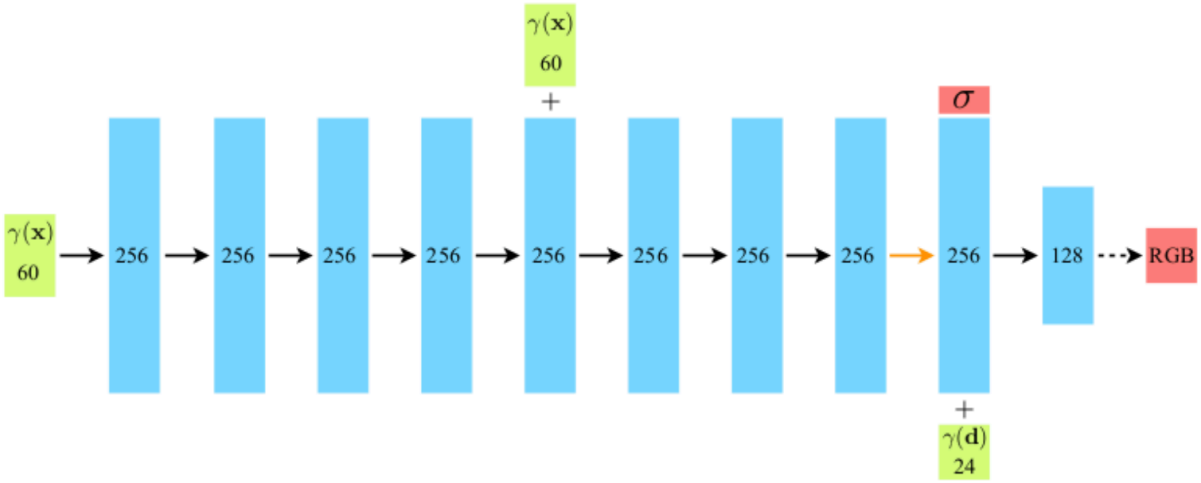
1.通过优化权重Θ 来从 Input5D坐标 映射到 C和σ.

2.通过限制神经网络，只让 location $X = (x, y, z)$ 控制 volume density σ 的预测，让 location X 和 viewing direction (θ, ϕ)一起预测color C. 【为了保持多视图的一致性】

3.MLP网络具体步骤如下：

（1）使用8个全连接层来处理输入的3D location $X = (x, y, z)$ 【使用ReLU激活和每层256个通道】，然后输出σ和256维特征向量；

（2）将该特征向量与相机2D viewing direction (θ, ϕ) 拼接，然后使用全连接层【使用ReLU激活和128通道】进行处理，得到输出的RGB颜色值。



疑问：y(x) 两次输入的60是什么含义？还有输入的24有是什么含义？

答案：在5.1部分【Positional encoding】，作者提出了一种优化思想，即在将坐标输入网络之前进行一个高频函数映射，这样可以更好的拟合包含高频变化的数据。

γ 函数是用来映射空间R 到高维的 R^{2L}.

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

所以，输入的3D location $X = (x, y, z)$ 是三维，作者在实验中对于X的L值取10，所以新维度 = $3 * 2 * L = 3 * 2 * 10$ = 60。同理，d是3D笛卡尔单位向量，作者在实验中对于d的L值取4，所以新维度就是24。

2. volume rendering with radiance fields

1.classical volume rendering：

The expected color $C(r)$ of camera ray $r(t)=o+td$ with near and far bounds t_n and t_f is

$$C(\boldsymbol{r})=\int_{t_n}^{t_f} T(t)\sigma(\boldsymbol{r}(t))\boldsymbol{c}(\boldsymbol{r}(t),\boldsymbol{d})dt, \\ where\ T(t)=e^{-\int_{t_n}^{t}\sigma(\boldsymbol{r}(s))ds}. \tag{3}$$

$\sigma(r(t))$ 表示体密度，也是指density，$\mathbf{c}(\cdot)$ 表示颜色。

$T(t)$ 表示的含义是accumulated transmittance along the ray from t_n to t_f ,也就是说是碰撞检测函数。当碰到第一个表面时，σ 比较大，此时 $T(t)$ 迅速衰减为一个较小的数，所以当碰到第二个表面时，$T(t)$较小，则$C(r)$不会对后面的表面进行累积，也就是图2-c中将第二个波峰的能量剔除。

2.实际上，计算机无法对连续的3D点进行处理，故将连续积分问题转换为可微的离散累积问题。将该问题进行离散化。

粗采样【coarse network】：

将$[t_n,t_f]$ 均分成为N份，然后从每份里面随机均匀的抽取样本。

$$t_i\sim u[t_n+\frac{i-1}{N}(t_f-t_n),t_n+\frac{i}{N}(t_f-t_n)] \tag{4}$$

所以上（2）式就可以离散化为：

$$\hat{C}(r)=\sum_{i=1}^{N}T_i(1-e^{-\sigma_i\delta_i})\mathbf{c}_i \\ T_i=e^{-\sum_{j=1}^{i-1}\sigma_j\delta_j} \tag{5}$$

其中$\delta_i=t_{i+1}-t_i$ ，是相邻样本之间的距离。

细采样【fine network】：

NeRF的渲染过程计算量很大，每条射线都要采样很多点。但实际上，一条射线上的大部分区域都是空区域，或者是被遮挡的区域，对最终的颜色没有啥贡献。因此，作者采用了一种“coarse to fine" 的形式，同时优化coarse网络和fine 网络。

首先对于coarse网络，可以先采样较为稀疏的N_c个点 ，并将（4）式中的离散求和函数重新表示为：

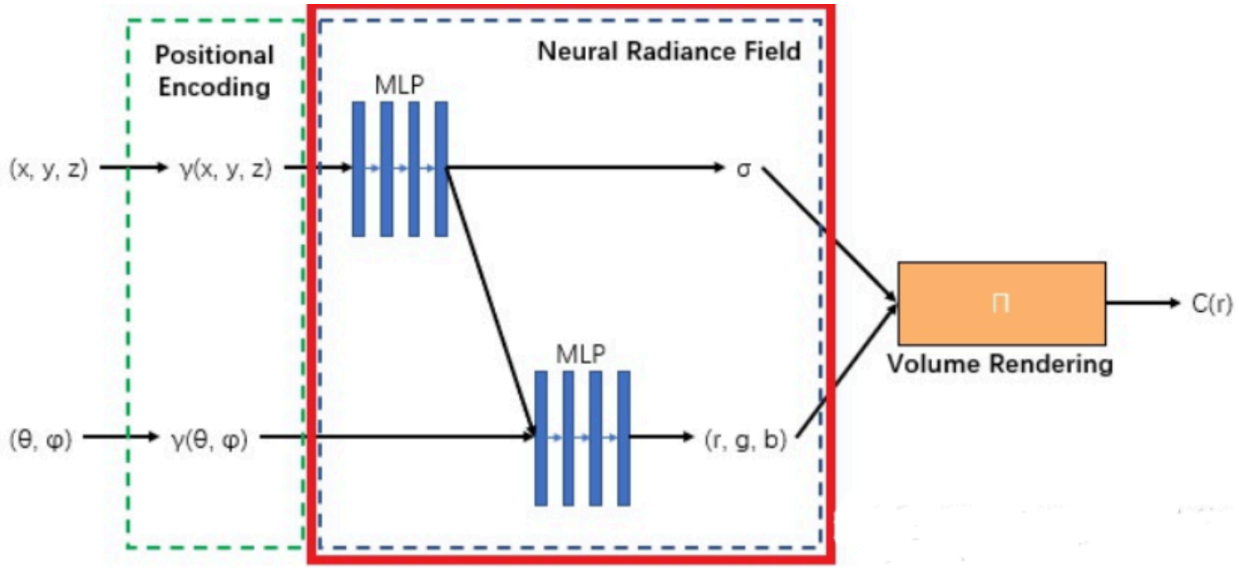
$$\hat{C}_c(r)=\sum_{i=1}^{N_c}w_i\mathbf{c}_i\ , \\ w_i=T_i(1-e^{-\sigma_i\delta_i})\quad ,T_i=e^{-\sum_{j=1}^{i-1}\sigma_j\delta_j} \tag{6}$$

接下来，对w_i进行归一化：

$$\hat{w}_i=\frac{w_i}{\sum_{j=1}^{N_c}w_j} \tag{7}$$

作者选用了标准化的归一化方法，这样的目的是确保数据的所有值的总和等于特定的值1，这样就可以把 $\hat{w}_i$ 看做是沿着射线的概率密度函数。so，通过这个概率密度函数，就可以粗略的得到射线上物体的分布情况。

接下来，再基于得到的概率密度函数来采样N_f个点，并用这 N_f个点和前面的 N_c个点一同计算fine 网络的渲染结果 $\hat{C}_f(r)$.



Loss function:

直接定义在渲染结果上的L2损失 (同时优化coarse 和 fine）：

$$L = \sum_{r \in R} ||\hat{C}(r) - C(r)||_2^2 \tag{8}$$

补充：

1.体渲染 与 Nerf [nerf 对于传统体渲染到底取代了哪一部分？]

传统体渲染过程：

光沿直线方向穿过一堆粒子，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，就可以渲染出投影图像。体渲染把每一条光路建模成由一个个粒子构成。【光子在光路传播中，可能因为粒子的遮挡损失能量，导致入射光减弱。光路中的粒子也可能本身会发光，抑或是周围光路的光子被弹射到当前光路，导致光线增强。】

在计算机中，由于计算资源限制，不可能建模出所有粒子的状态，因此通常会在每条光路上采样一些点，由这些采样点上的粒子来代表整条光线，最终每条光线渲染出来的颜色值都可以用下面的公式表示

The expected color $C(r)$ of camera ray $r(t) = o + td$ ：

$$\hat{C}(r) = \sum_{i=1}^{N} T_i (1 - e^{-\sigma_i \delta_i}) \mathbf{c}_i$$
$$T_i = e^{-\sum_{j=1}^{i-1} \sigma_j \delta_j} \tag{9}$$

其中$\delta_i = t_{i+1} - t_i$，是相邻样本之间的距离。

也就是说，在传统体渲染的过程中，我们必须知道整个场景中每条光线上的每个采样点的粒子状态，才能渲染出整个画面。计算光线上的粒子状态本身是一件很复杂的事情，所以Nerf让神经网络去学习来计算这些数值。

2.Nerf做了：

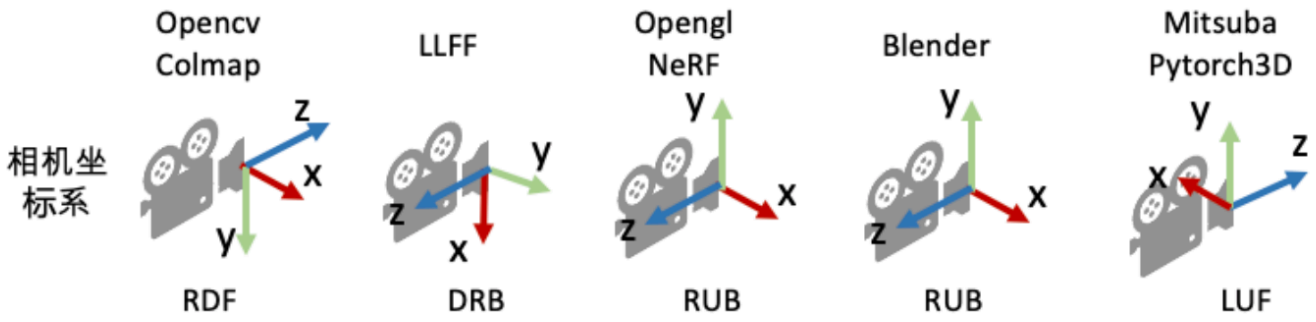
给定相机位置和朝向后，我们可以确定出当前的成像平面。然后，将相机的位置坐标和平面上的某个像素相连，就确定了一条光线 (也即确定了光线的方向)。接着用网络预测出光线上每个采样点的粒子信息，就可以确定像素颜色。这个过程重复下去，直到每个像素都渲染完为止。

这些排列整齐的光线，构成了类似磁场一样的东西，而光线本身就是一种辐射，因此叫辐射场。而每条光线上的粒子信息又都是由神经网络预测的，因此作者给整个过程命名为**神经辐射场**。

3.关于生成光线、相机参数

我们要做的是：生成每个方向下的像素点到光心的单位方向(z轴为单位1)，通过这个单位方向，可以通过调整z轴的坐标来生成空间中每一个点坐标，借此模拟出一条光线。这个射线是怎么构造的。**给定一张图像的一个像素点，我们的目标是构造以相机中心为起始点，经过相机中心和像素点的射线。**该像素点就是位于成像平面的像素点。

1.为了唯一地描述每一个空间点的坐标以及相机的位置和朝向，我们需要先定义一个世界坐标系。



常见的相机坐标系定义习惯（右手坐标系）。注意：在OpenCV/COLMAP的相机坐标系里相机朝向+z轴，在LLFF/NeRF的相机坐标系中里相机朝向-z轴。有时我们会按坐标系的xyz朝向描述坐标系，如OpenCV/COLMAP里使用的RDF表述X轴指向right，Y轴指向Down，Z轴指向Foward。

其中：

1）**相机的位置和朝向：由外参决定【外参 World-To-camera，w2c】。**

外参：4*4的矩阵M，作用是将世界坐标系的点$P_{world}=[x,y,z,1]$变换到相机坐标系P_{camera}下：

$$P_{camera}=MP_{world} \tag{10}$$

相机外参的逆矩阵被称为**camera-to-world (c2w)矩阵**，左上角3x3是旋转矩阵R，右上角的3x1向量是平移向量T：

$$c2w=\begin{bmatrix}R & T\\ 0 & 1\end{bmatrix}=\left[\begin{array}{ccc|c}r_{11} & r_{12} & r_{13} & t_1\\ r_{21} & r_{22} & r_{23} & t_2\\ r_{31} & r_{32} & r_{33} & t_3\\ \hline 0 & 0 & 0 & 1\end{array}\right]=\begin{bmatrix}X & Y & Z & O\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{11}$$

旋转矩阵**R**的第一列到第三列分别表示了相机坐标系的X, Y, Z轴在世界坐标系下对应的方向；

平移向量**T**表示的是相机原点O在世界坐标系的对应位置。

外参由数据集中['transform_matrix']得到。

2）**投影属性：由内参决定。**

内参：3*3的矩阵K，作用是将相机坐标系下的3D坐标映射到2D的图像平面：

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{12}$$

f_x和f_y是相机的水平和垂直**焦距**（对于理想的针孔相机，$f_x = f_y$）。焦距的物理含义是相机中心到成像平面的距离，长度以像素为单位。

c_x和c_y是图像原点相对于相机光心的水平和垂直偏移量。可以用图像宽和高的1/2近似。

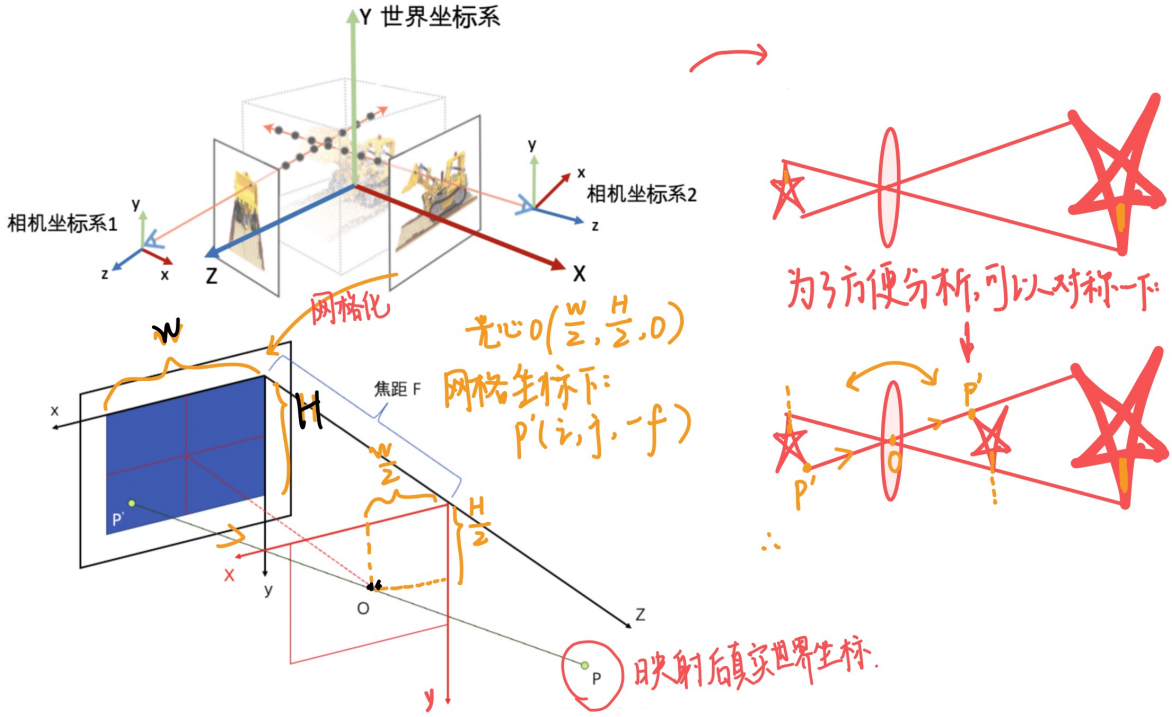
内参由焦距、图像的宽、高得到。

2.Nerf：NeRF所做的是在相机坐标系下构建射线，然后再通过camera-to-world (c2w)矩阵将射线变换到世界坐标系。

step1： 写出相机中心、像素点在相机坐标系下的3D坐标

step2： 使用c2w矩阵变换到世界坐标系上去

过程如下：



4. **关于体渲染的离散公式推导与代码实现**

将$[t_n, t_f]$ 均分成为N份，然后从每份里面随机均匀的抽取样本。

$$t_i \sim u[t_n + \frac{i-1}{N}(t_f - t_n), t_n + \frac{i}{N}(t_f - t_n)] \tag{13}$$

所以上（2）式就可以离散化为：

$$\hat{C}(r) = \sum_{i=1}^{N} T_i (1 - e^{-\sigma_i \delta_i}) \mathbf{c}_i$$
$$T_i = e^{-\sum_{j=1}^{i-1} \sigma_j \delta_j} \tag{14}$$

尽管我们使用一组离散的样本来估算积分，但分层采样使我们能够表示连续的场景表示，因为它导致在优化过程中在连续位置对 MLP 进行评估。我们使用这些样本通过quadrature rule估计 $\mathrm{C}(\mathrm{r})$，也就是文中的等式3：

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

其中$\delta_{\mathrm{i}} = \mathrm{t}_{\mathrm{i}+1} - \mathrm{t}_{\mathrm{i}}$ 是相邻样本之间的距离。这个从$(\mathrm{c}_{\mathrm{i}}, \sigma_{\mathrm{i}})$ 值的集合中计算 $\hat{\mathrm{C}}(\mathrm{r})$ 的函数是微分的，并简化为alpha 值 $\alpha_{\mathrm{i}} = 1 - \exp(-\sigma_{\mathrm{i}}\delta_{\mathrm{i}})$ 的传统alpha合成。

> 什么是alpha合成?
> Alpha合成（alpha compositing）是一种将图像与背景结合的过程，结合后可以产生部分透明或全透明的视觉效果。图像里记录着每个像素的颜色信息，额外的信息以 0 和 1 之间的值表示，记录在Alpha通道里。0 表示该像素是透明的，即图中的几何体没有覆盖到本像素；而 1 则表示像素不透明，几何体完全覆盖了此像素。
> Alpha混合（alpha blending）是将半透明的前景色与背景色结合的过程，可以得到混合后的新颜色。前景色的透明度不限，从完全透明到完全不透明都可以。如果前景色完全透明，混合后的颜色就是背景色；如果前景色完全不透明，混合后的颜色就是前景色；如果在这两种极端情况之间，混合后的颜色可以通过前景色和背景色的加权平均计算。
>
> $$\begin{cases} \mathrm{out}_A = \mathrm{src}_A + \mathrm{dst}_A(1 - \mathrm{src}_A) \\ \mathrm{out}_{RGB} = \left(\mathrm{src}_{RGB}\mathrm{src}_A + \mathrm{dst}_{RGB}\mathrm{dst}_A\left(1 - \mathrm{src}_A\right)\right) \div \mathrm{out}_A \\ \mathrm{out}_A = 0 \Rightarrow \mathrm{out}_{RGB} = 0 \end{cases}$$