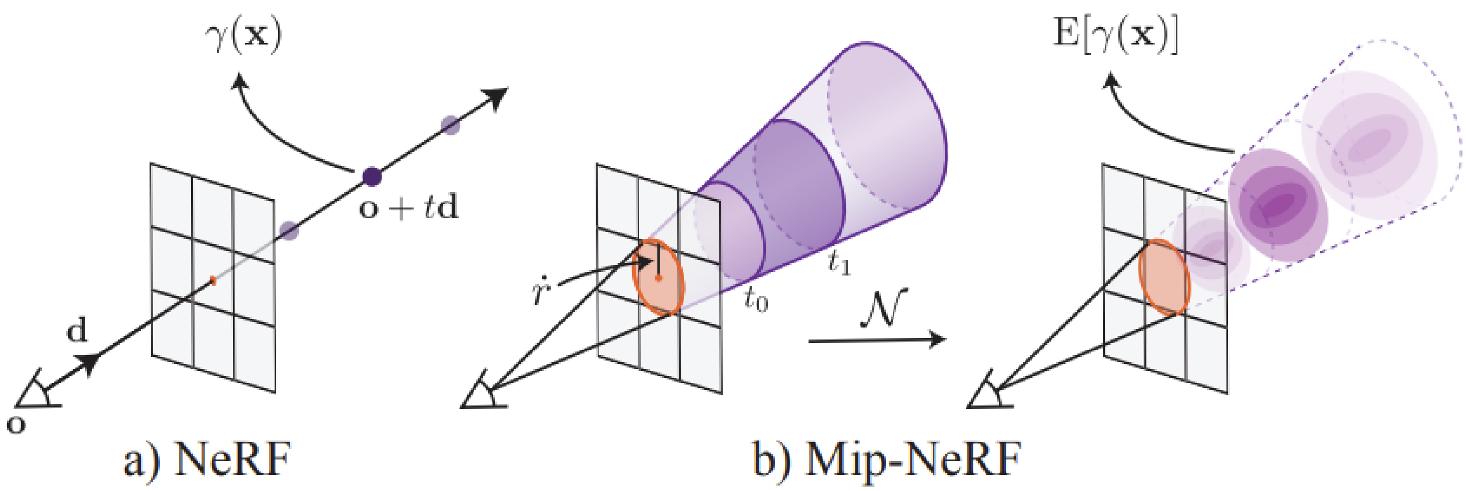
Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields

Paper: https://arxiv.org/pdf/2111.12077v3.pdf
Year: 2022

Mip Nerf



贡献

1. 解决了Mip-Nerf不能很好用于360度无边界场景渲染的问题;
2. 解决了采样效率低的问题;
3. 解决了密度构建中的结构不准确问题。

360度无边界场景的渲染

目前渲染场景主要分为三种：有边界小场景，前向无边界场景，360度无边界场景。

https://www.matthewtancik.com/nerf
https://jonbarron.info/mipnerf360/

Nerf对无边界场景的建模通常是要比有边界建模难度高的。

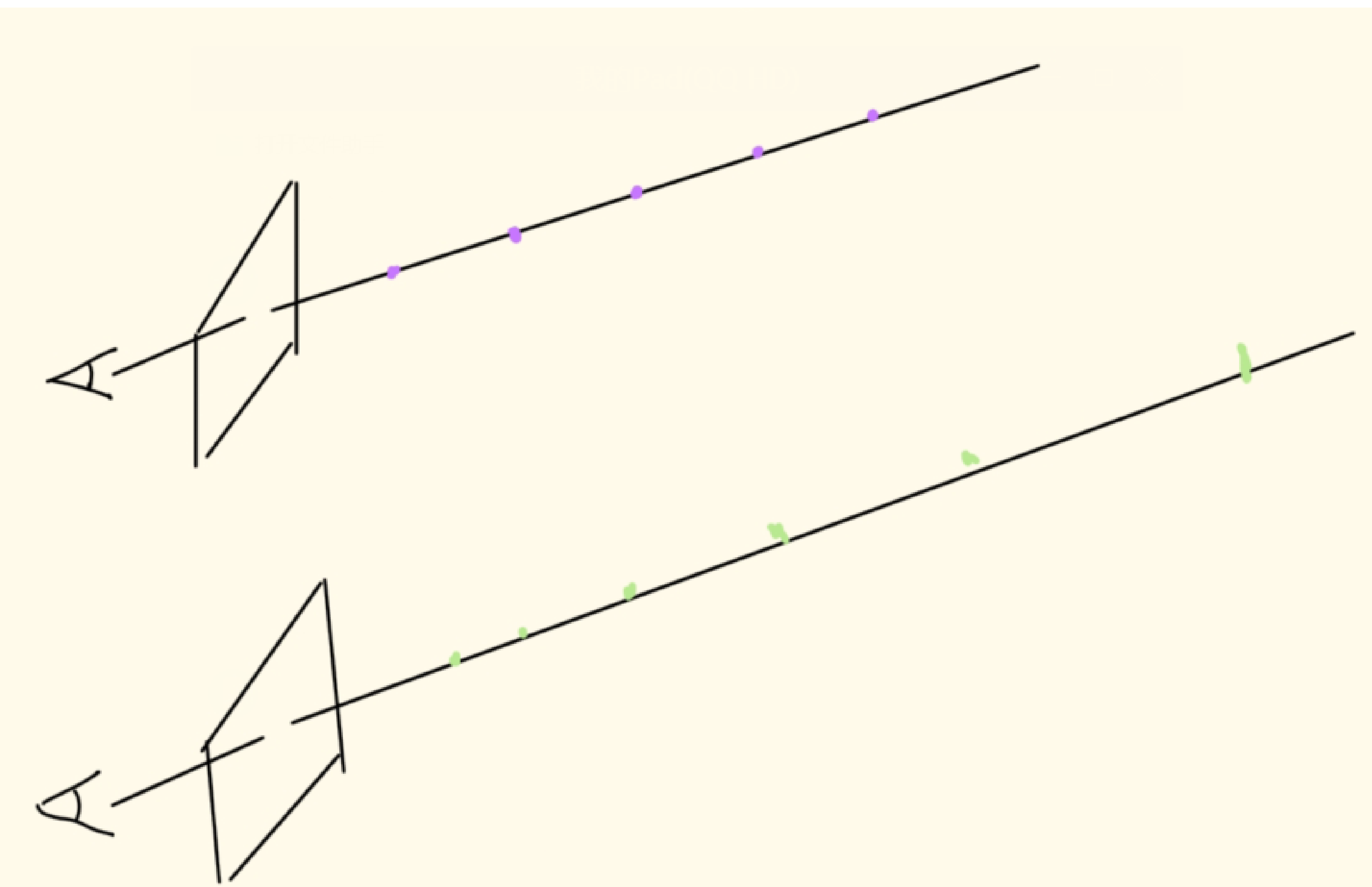
(a) bounding volume for the truck only (b) bounding volume for the entire scene

前向无边界场景里，可以通过将世界坐标转为NDC坐标来一定程度解决。
image.png
但是在360度场景里面就比较麻烦了。

要想对360度无边界场景进行建模，有两方面是需要关注的：

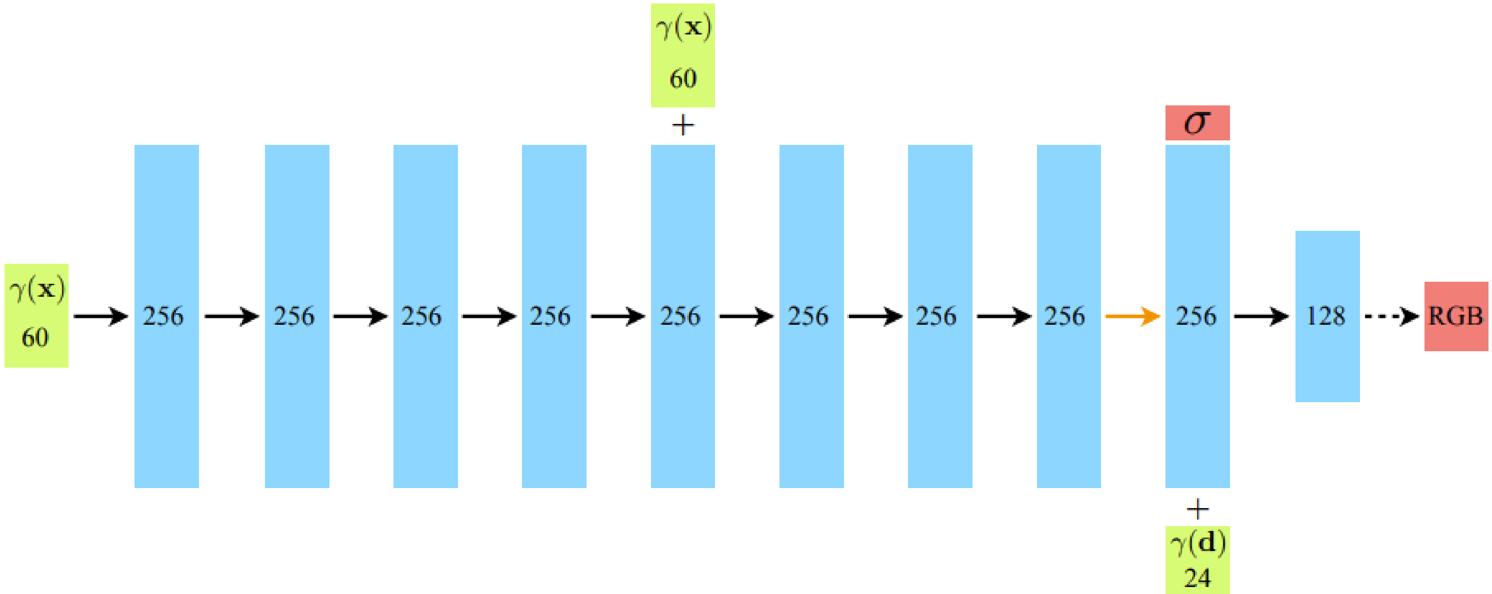
1. 采样策略；
2. 输入策略；

非均匀采样



在场景下，有粗采样与细采样，粗采样一般使用等距采样，但是根据前面的分析，这种等距采样，在大场景下是有问题的，比较合理的一种采样方式是，近处的采样点多一些，远处的采样点少一些，因此近处的细节比较多。

输入

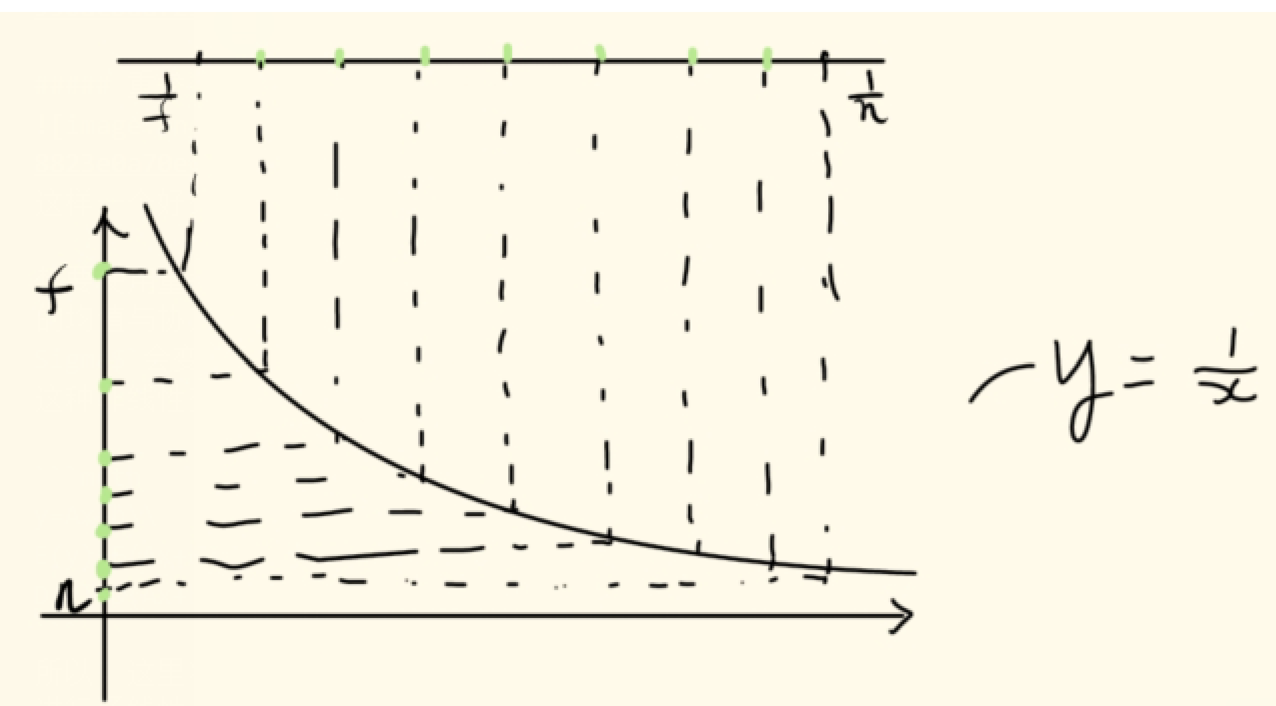


本文方法

差异采样技术

1. 先在0-1之间均匀采样一些点;
2. 设采样到的点是 s , $s \in [0,1]$, 则新的采样点是：

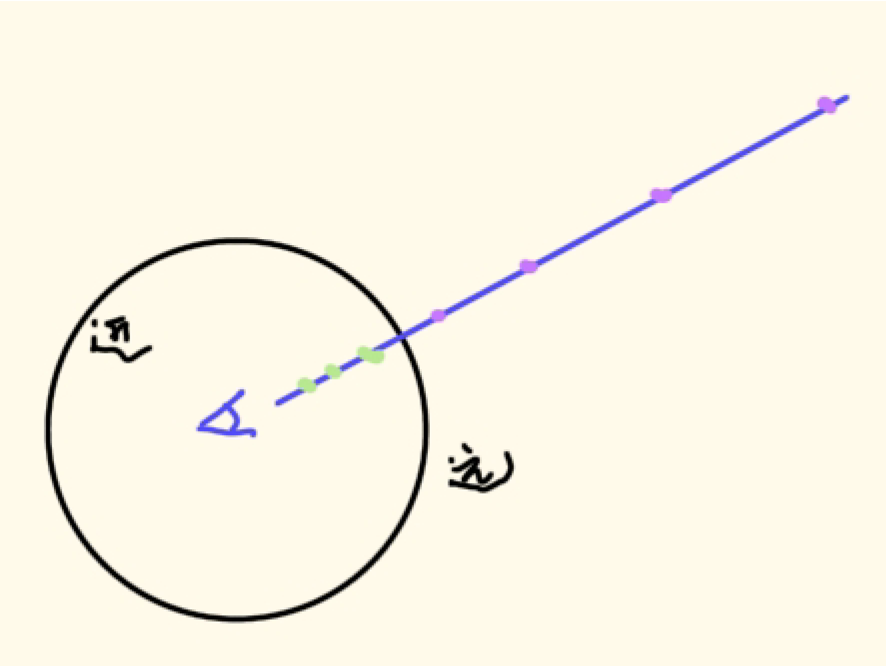
$$s' = \frac{1}{s\frac{1}{f} + (1-s)\frac{1}{n}}$$



这就是论文中给出的这个公式：

$$s \triangleq \frac{g(t) - g(t_n)}{g(t_f) - g(t_n)}, \quad t \triangleq g^{-1}(s \cdot g(t_f) + (1-s) \cdot g(t_n)), \quad (11)$$

重参数化技巧



$$\text{contract}(\mathbf{x}) = \begin{cases} \mathbf{x} & \|\mathbf{x}\| \le 1 \\ \left(2 - \frac{1}{\|\mathbf{x}\|}\right)\left(\frac{\mathbf{x}}{\|\mathbf{x}\|}\right) & \|\mathbf{x}\| > 1 \end{cases} \qquad (10)$$

这样一个好处是，重参数化以后，坐标值的变化不会很大。

想要使用重参数化技术，还有一个问题要解决，就是重参数化以后怎么求一个截椎体里面的均值与协方差。也就是说原数据映射到f(x)上之后，原来的均值 u 与协方差矩阵 Σ 会变为多少?
这种非线性变化下，进行求解，是比较困难的，但是线性下求解相对容易:

$$\begin{gathered} y = Ax + b \\ E(y) = E(Ax+b) = E(Ax) + E(b) = AE(x) + b = Au + b \\ D(y) = D(Ax+b) = AD(x)A^T \end{gathered}$$

所以，这里将 $f(x)(contract(x))$ 在 u 点进行一阶泰勒展开，将一个非线性的函数进行了线性化处理。

$$f(x) = f(u) + J_f(u)(x-u)$$

线性化以后:

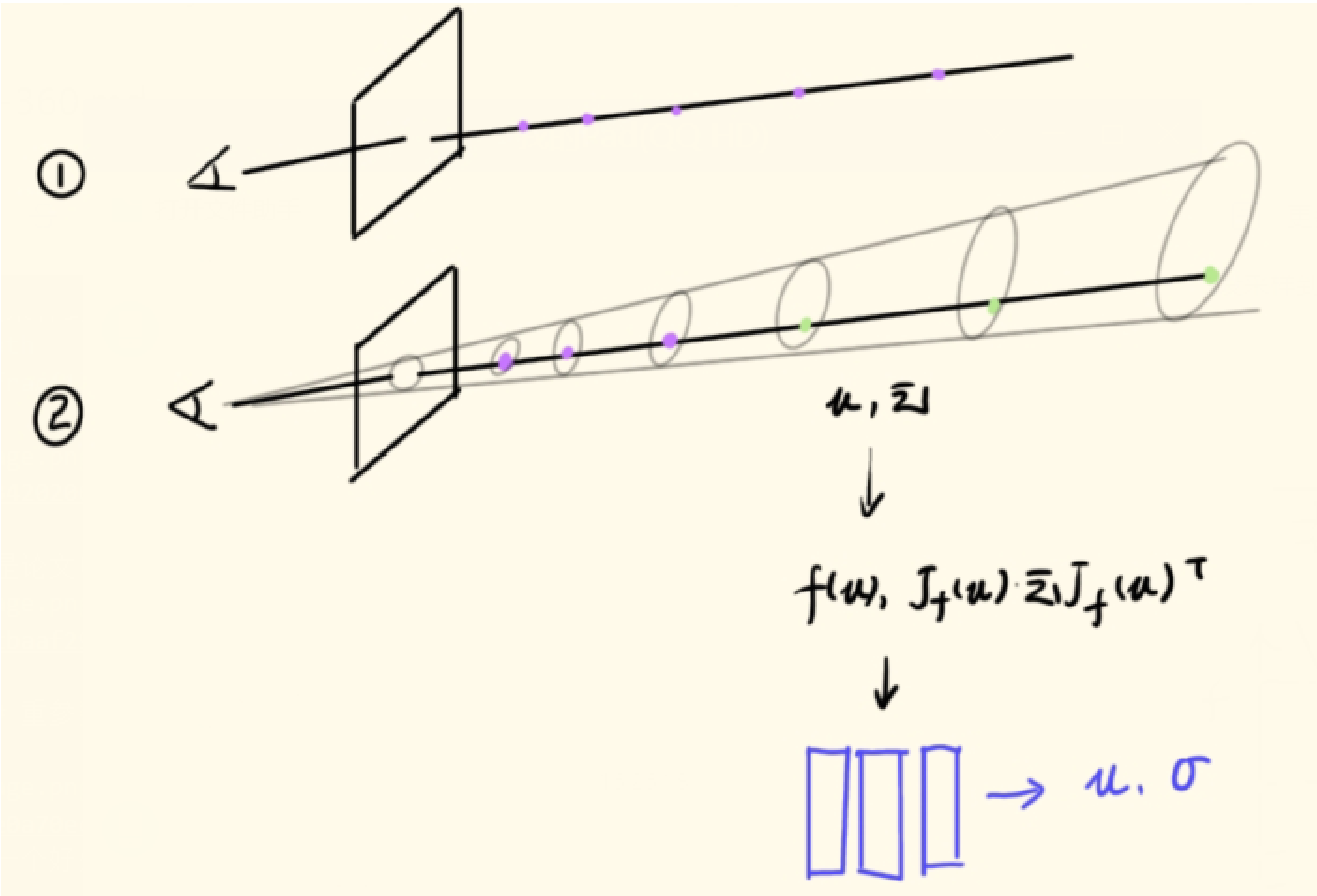
$$\begin{gathered} E(f(x)) = E(f(u) + J_f(u)(x-u)) \\ = f(u) + E(J_f(u)(x-u)) \\ = f(u) + J_f(u)E(x-u) \\ = f(u) \end{gathered}$$

$$\begin{gathered} D(f(x)) = D(f(u) + J_f(u)(x-u)) \\ = D(J_f(u)(x-u)) \\ = D(J_f(u)x) \\ = J_f(u)D(x)J_f(u)^T \end{gathered}$$

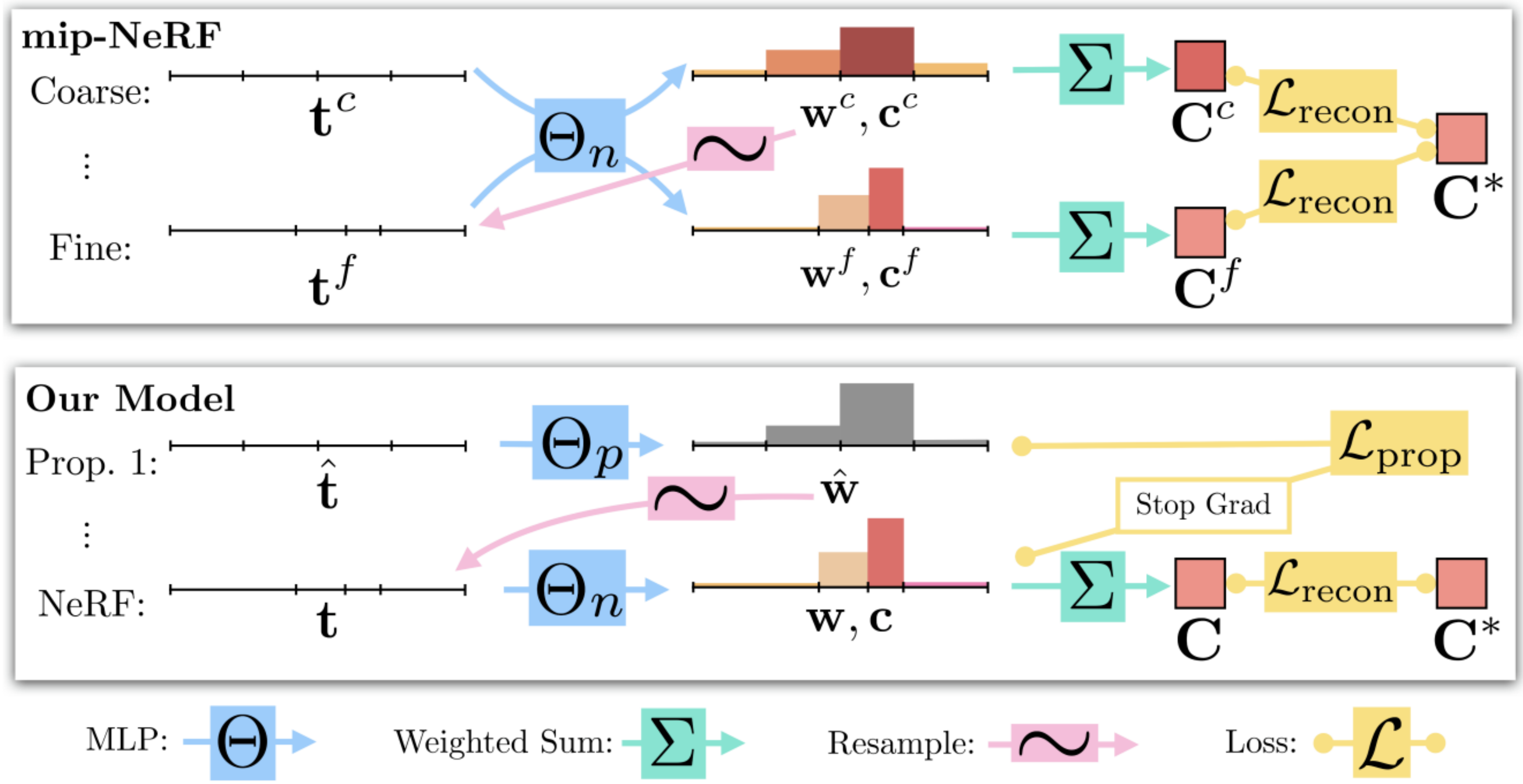
$$f(\mathbf{x}) \approx f(\boldsymbol{\mu}) + \mathbf{J}_f(\boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu}) \tag{8}$$

$$f(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \left(f(\boldsymbol{\mu}), \mathbf{J}_f(\boldsymbol{\mu})\boldsymbol{\Sigma}\mathbf{J}_f(\boldsymbol{\mu})^{\mathrm{T}}\right)$$

总体过程：



采样效率低



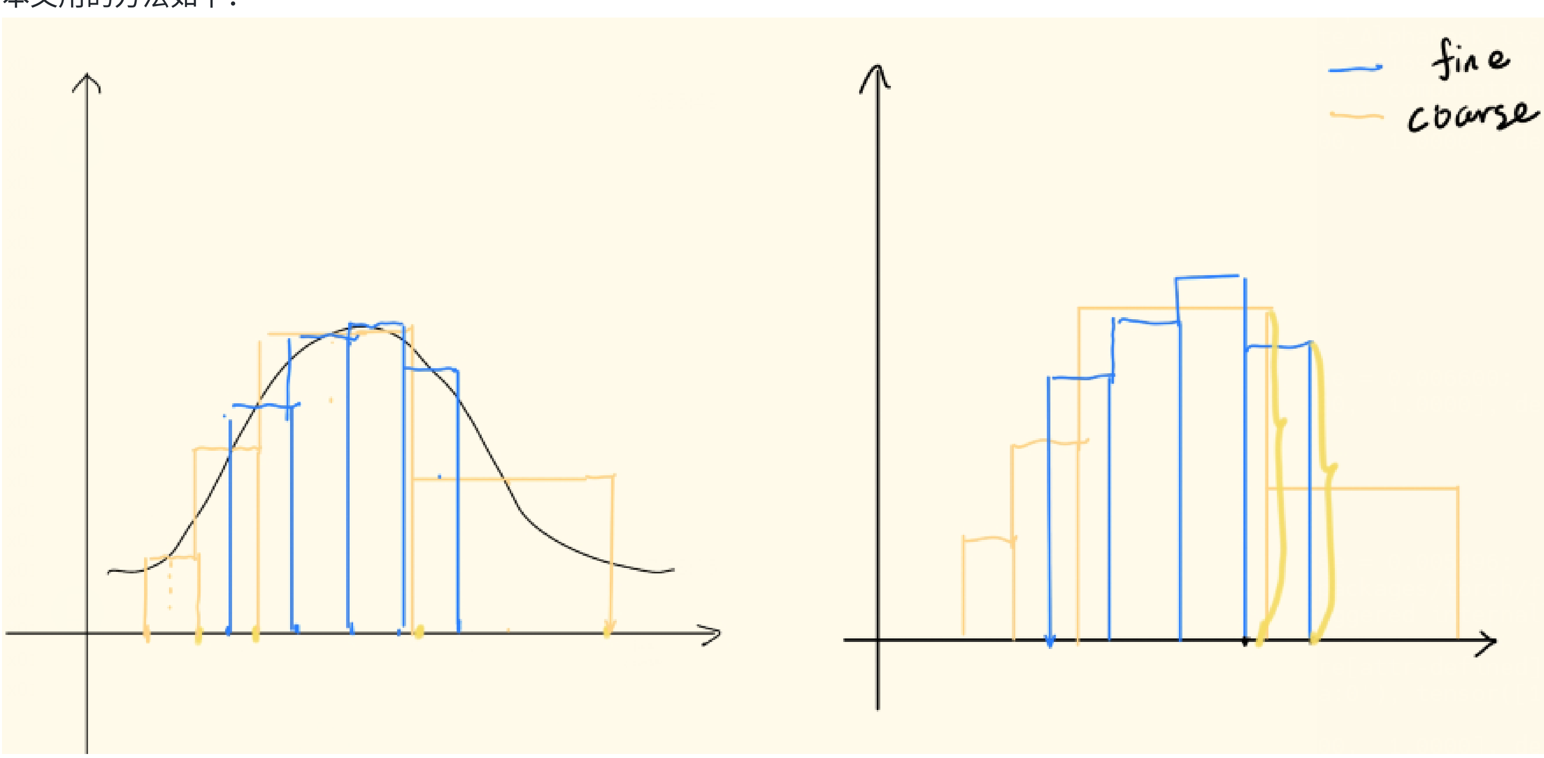
proposal MLP: 新加的，输入xyz坐标，得到weight的值。
NeRF MLP：跟原来一样，输入xyz坐标与view dir, 得到密度,weight与颜色值。

其中，图中的 L_{prop} 是用来约束, coarse阶段出来的weight和fine阶段出来的weight分布要差不多。

L_{prop} 的计算

要使proposal MLP和NeRF MLP输出的weight分布是一致的，一个直观的想法是，可以计算两个直方图的损失，但是由于coarse和fine阶段采样的点不一样，所以不太方便直接使用MSE或者L1损失。
以下这个图为例，表示的都是同一weight分布，但是由于两个采样点的位置不同，其直方图看起来非常不一样。

本文用的方法如下：

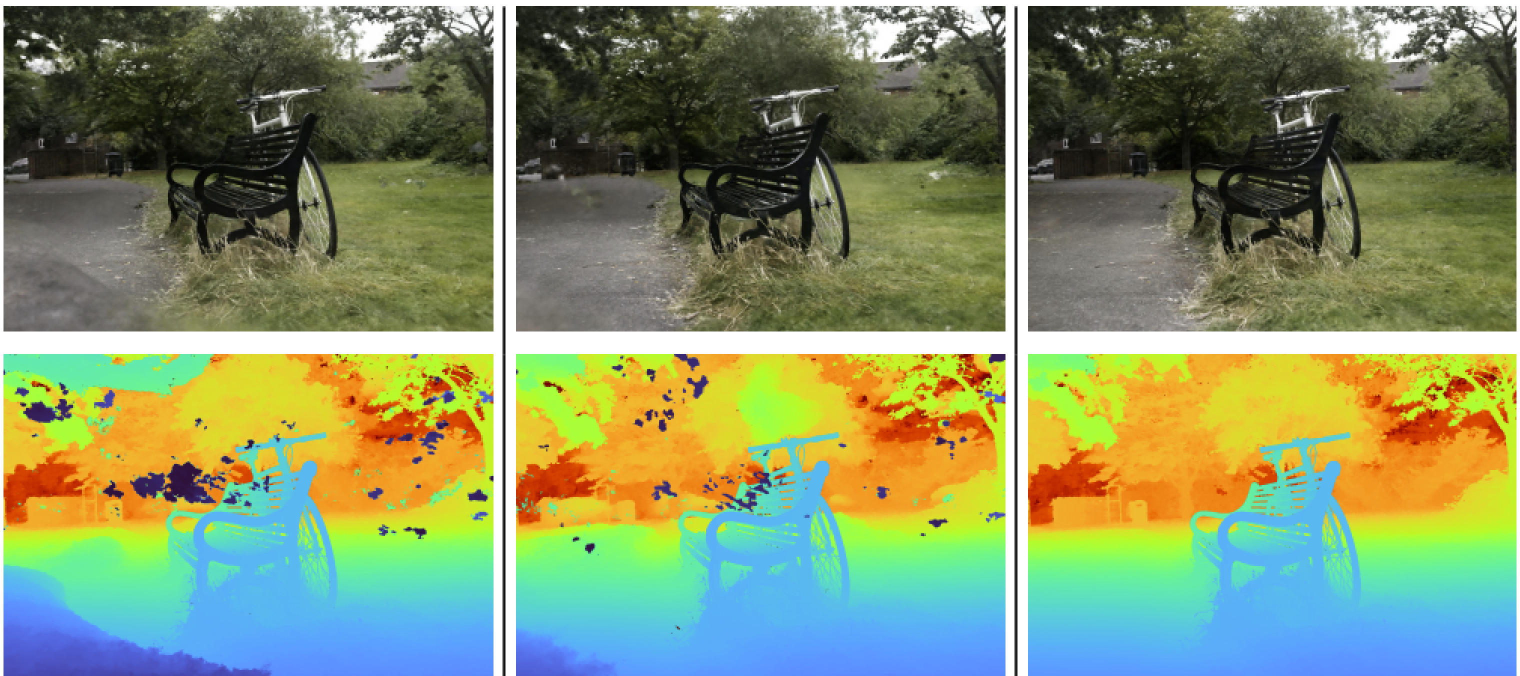


书上的公式如下：

$$\operatorname{bound}(\hat{\mathbf{t}}, \hat{\mathbf{w}}, T) = \sum_{j:\, T \cap \hat{T}_j \neq \varnothing} \hat{w}_j \,. \tag{12}$$

$$\mathcal{L}_{\text{prop}}(\mathbf{t}, \mathbf{w}, \hat{\mathbf{t}}, \hat{\mathbf{w}}) = \sum_i \frac{1}{w_i} \max(0, w_i - \operatorname{bound}(\hat{\mathbf{t}}, \hat{\mathbf{w}}, T_i))^2 \,, \tag{13}$$

密度结构不准确



(a) no $\mathcal{L}_{\text{dist}}$ (b) no $\mathcal{L}_{\text{dist}}$, w/noise [33] (c) with $\mathcal{L}_{\text{dist}}$

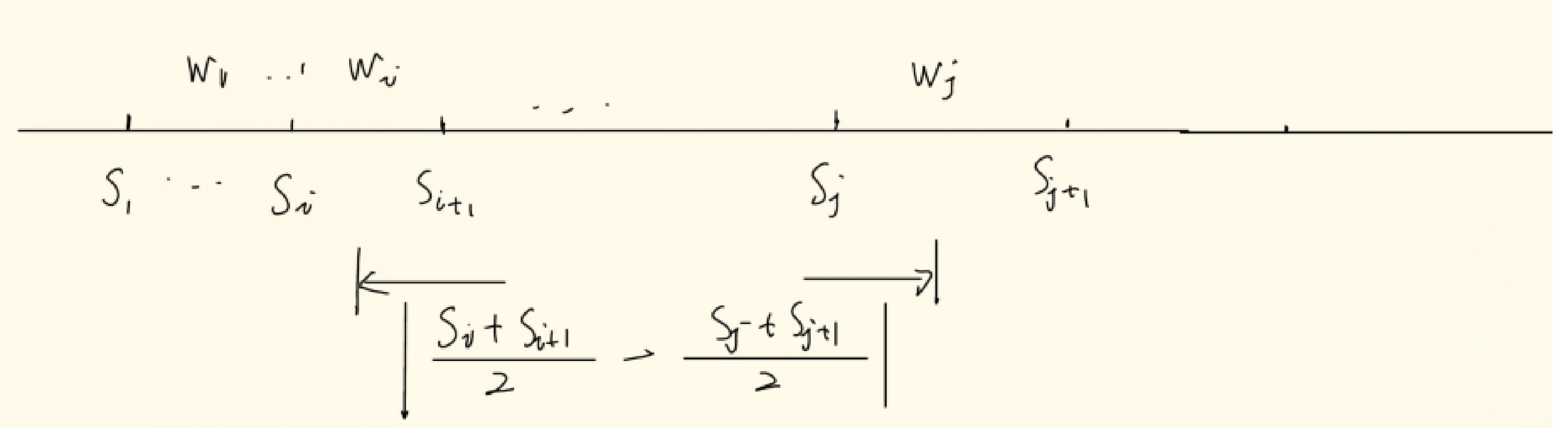
大场景对密度的预测不是太准确，会有一些半透明的物体漂浮在空中，一个准确的密度的预测其weight分布应该是这样的：

mip nerf 360里面使用了一个损失函数对其进行了约束：

$$\mathcal{L}_{\text{dist}}(\mathbf{s}, \mathbf{w}) = \iint_{-\infty}^{\infty} \mathbf{w}_{\mathbf{s}}(u) \mathbf{w}_{\mathbf{s}}(v) |u - v| \, d_u \, d_v \,, \tag{14}$$

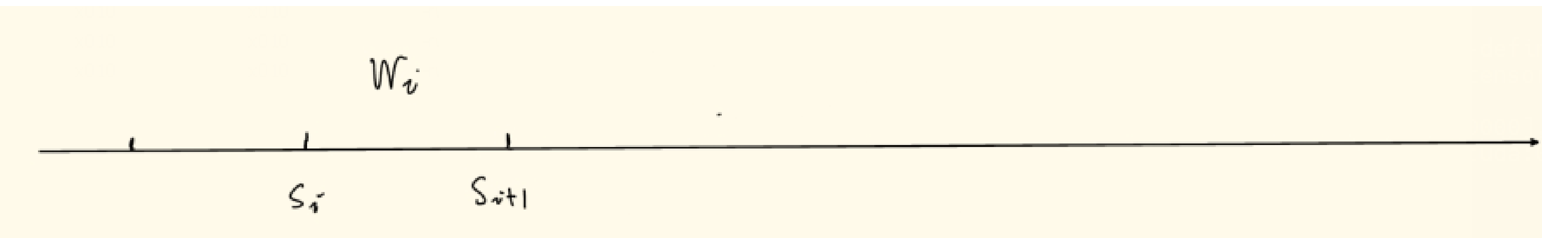
离散形式：

当两个点在不同的采样段上：



$$L_{dist}(S,W)=\sum_{i,j} w_i w_j \left|\frac{s_i+s_{i+1}}{2}-\frac{s_j+s_{j+1}}{2}\right|$$

当两个点在同一采样段上：



$$\begin{aligned}L_{dist}(s_i,w_i)&=\int_{s_i}^{s_{i+1}}\int_{s_i}^{s_{i+1}} w_i w_i |u-v| du dv\\&=w_i^2\int_{s_i}^{s_{i+1}}\int_{s_i}^{s_{i+1}} |u-v| du dv\\&=w_i^2\frac{1}{3}(s_{i+1}-s_i)\end{aligned}$$

将每一段的都加起来：

$$L_{dist}(S,W)=\frac{1}{3}\sum_i w_i^2(s_{i+1}-s_i)$$

书里给的公式：

$$\begin{aligned}\mathcal{L}_{\mathrm{dist}}(\mathbf{s},\mathbf{w})=&\sum_{i,j} w_i w_j \left|\frac{s_i+s_{i+1}}{2}-\frac{s_j+s_{j+1}}{2}\right|\\&+\frac{1}{3}\sum_i w_i^2(s_{i+1}-s_i)\end{aligned}\tag{15}$$