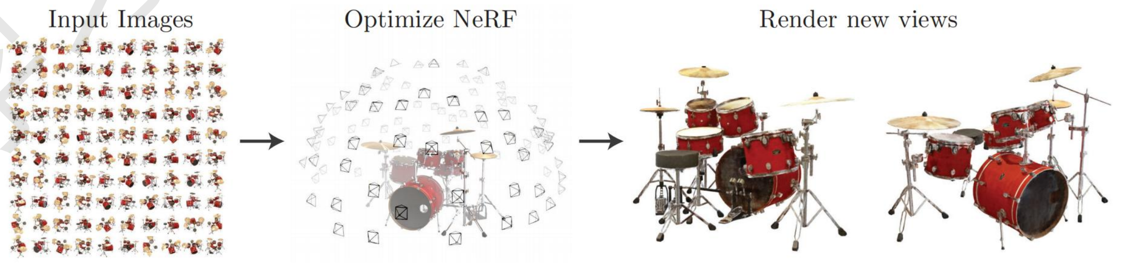
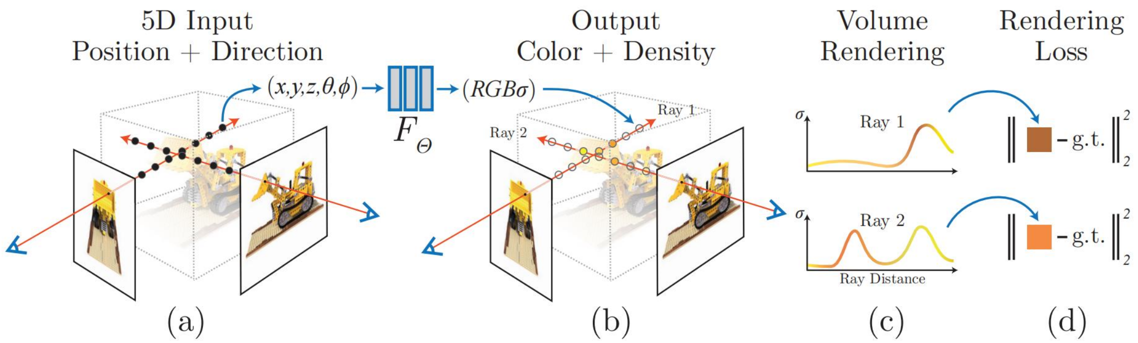
NeRF

RePresenting Scenes as Neural Radiance Fields for View Synthesis

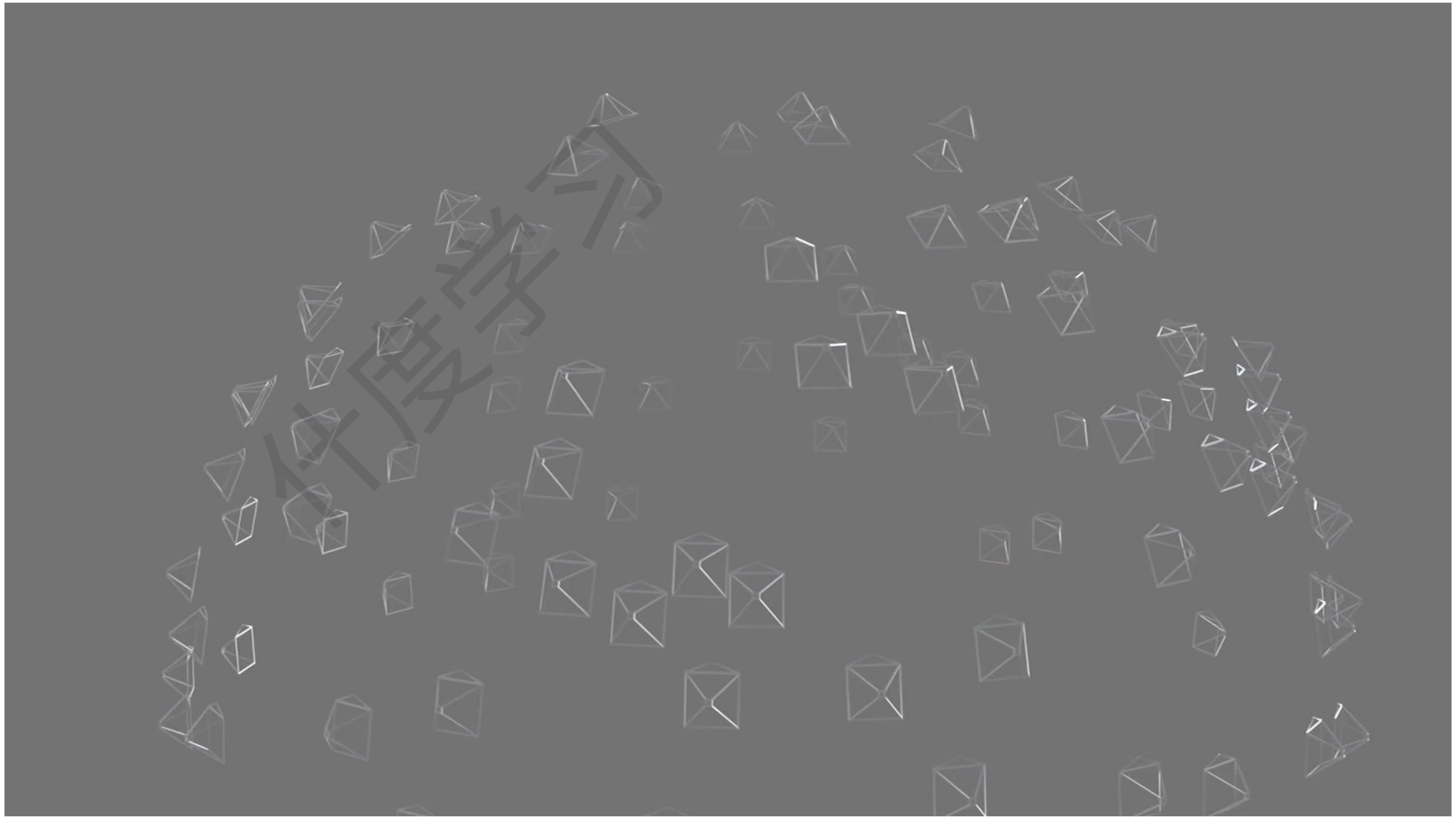
NeRF

- NeRF 是 2020年 ECCV 的 best paper
- NeRF 解决新视图合成问题
- NeRF 是可微渲染的一种

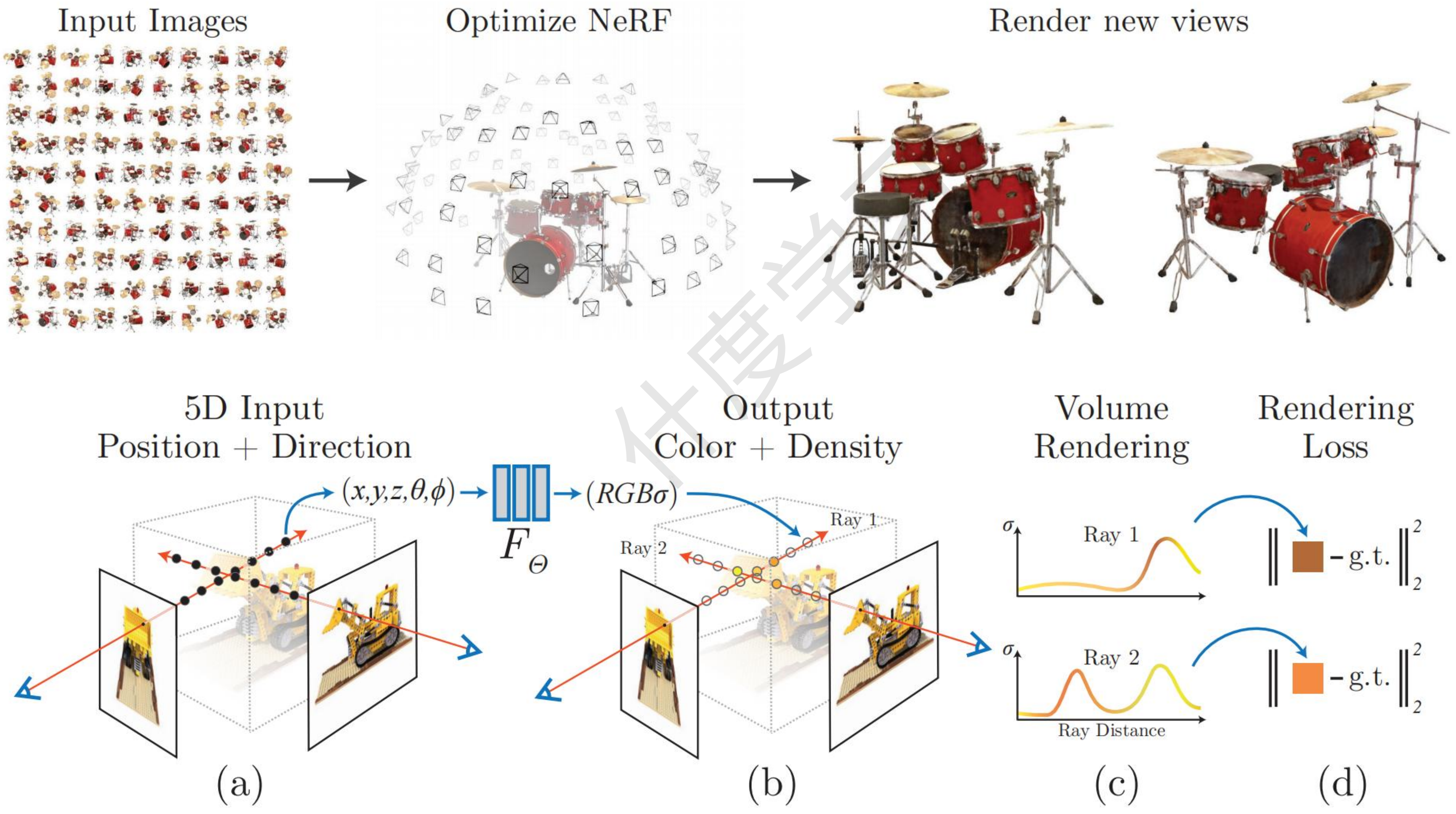




NeRF

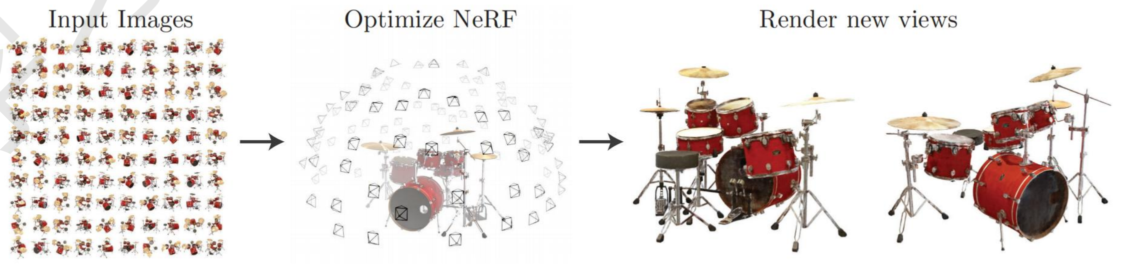


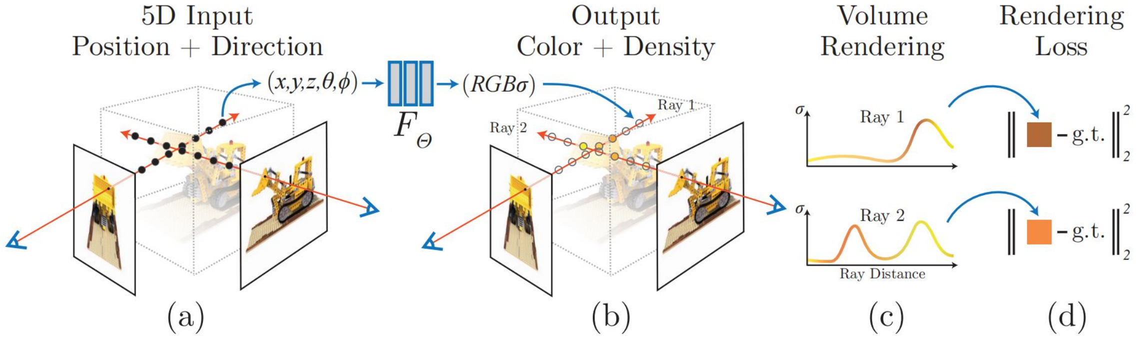
NeRF


Input Images
Optimize NeRF
Render new views
5D Input
Position + Direction
(x,y,z,θ,ϕ)
F_Θ
(RGBσ)
Output
Color + Density
Ray 1
Ray 2
Volume
Rendering
Rendering
Loss
σ
Ray 1
Ray 2
Ray Distance
‖ ■ − g.t. ‖²₂
‖ ■ − g.t. ‖²₂
(a)
(b)
(c)
(d)

Core

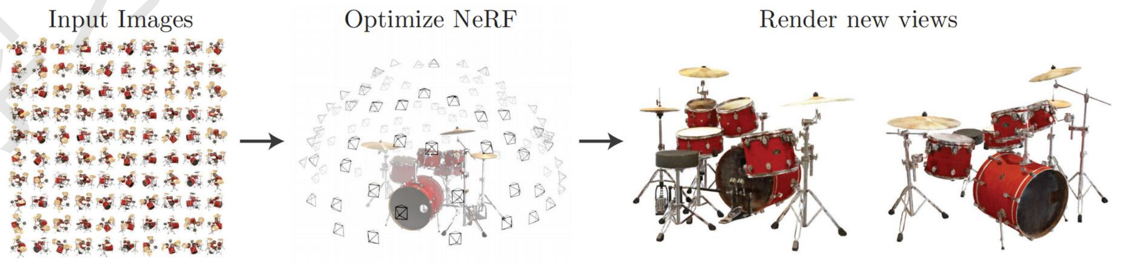
核心内容:
1. 体渲染
2. MLP
3. Positional Encoding
4. Hierarchical sampling

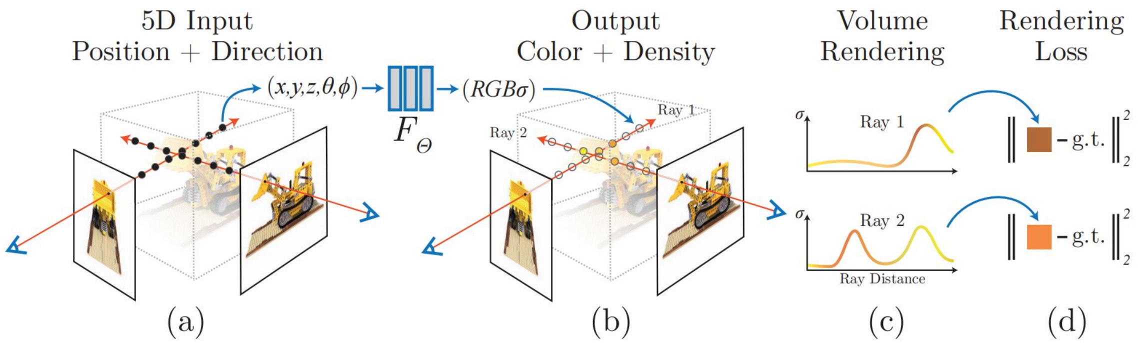




NeRF

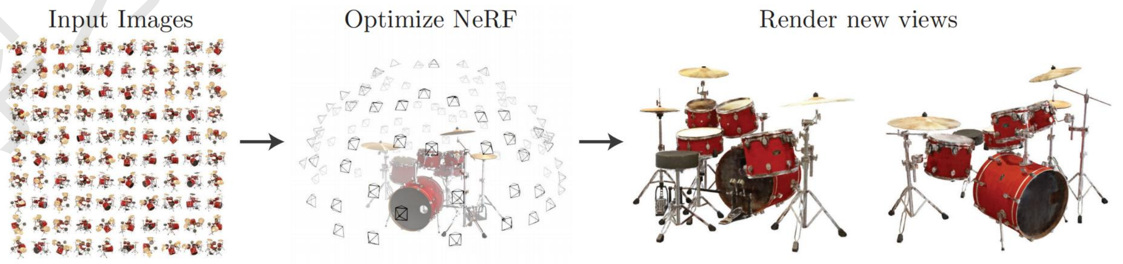
- 体渲染
- Positional Encoding
- Hierarchical sampling
- 实现
- 数据集
- 评价指标

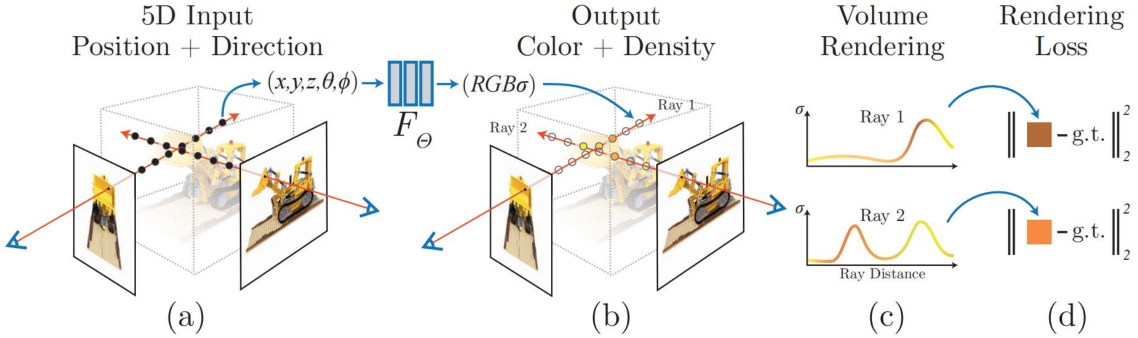




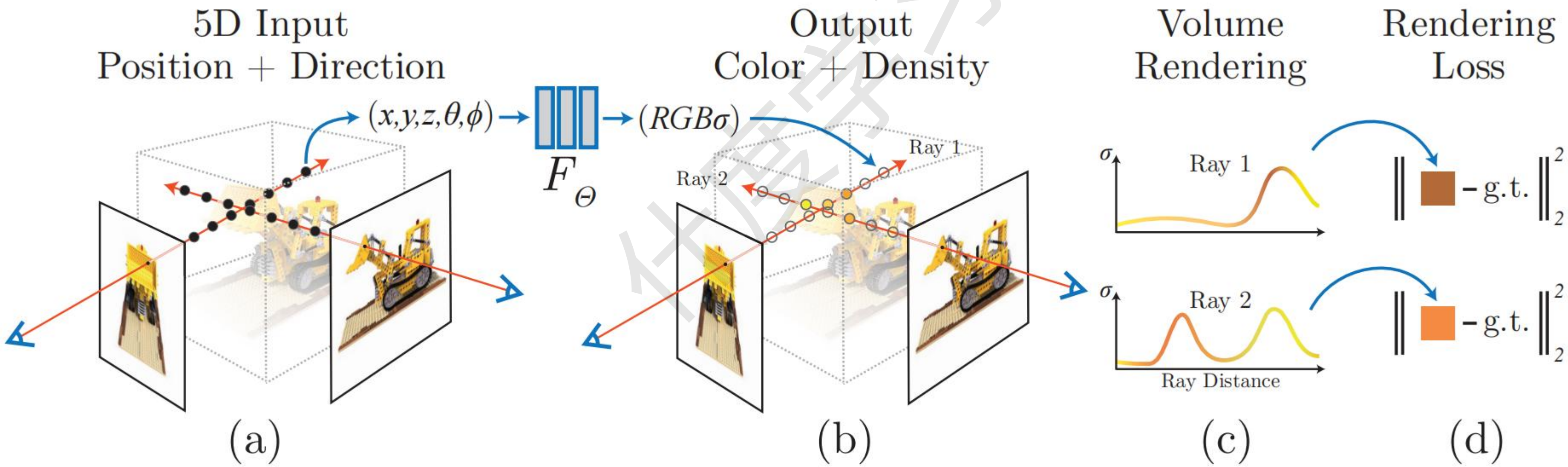
NeRF

- Neural Radiance Field 是2D -> 3D 的过程
- Volume Rendering 是3D -> 2D 的过程



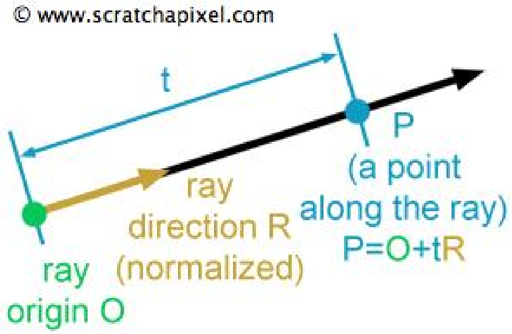


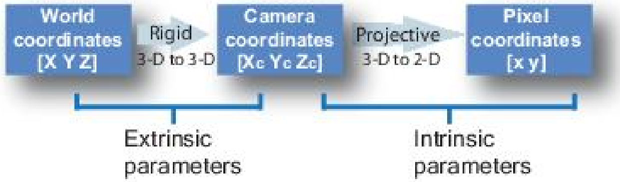
辐射场-体渲染

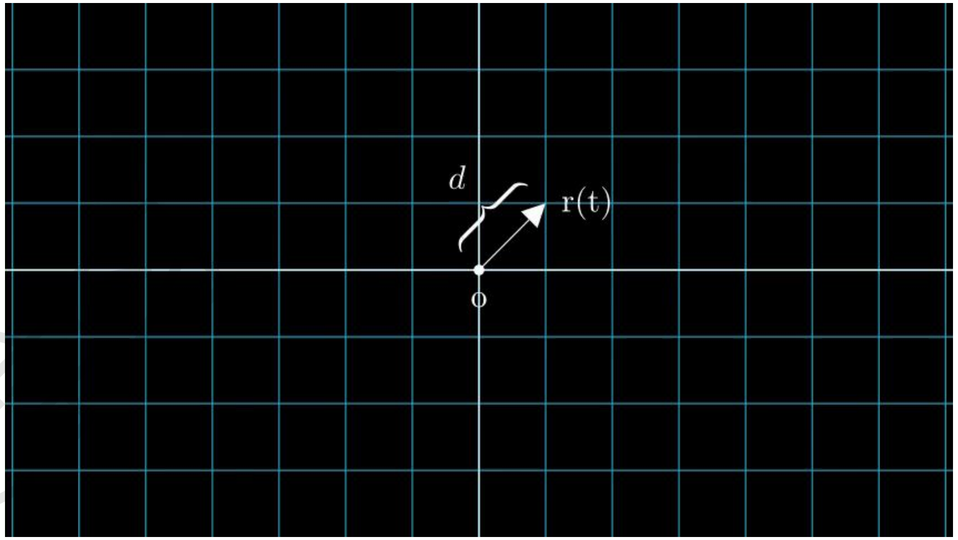

5D Input
Position + Direction
(x,y,z,θ,ϕ)
F_Θ
$(RGB\sigma)$
Output
Color + Density
Ray 1
Ray 2
Volume
Rendering
Rendering
Loss
σ
Ray 1
Ray 2
Ray Distance
$\left\| \blacksquare - \mathrm{g.t.} \right\|_2^2$
(a)
(b)
(c)
(d)

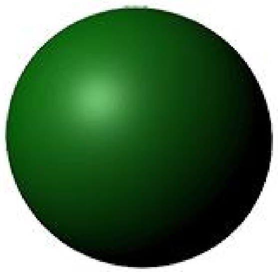
辐射场-体渲染

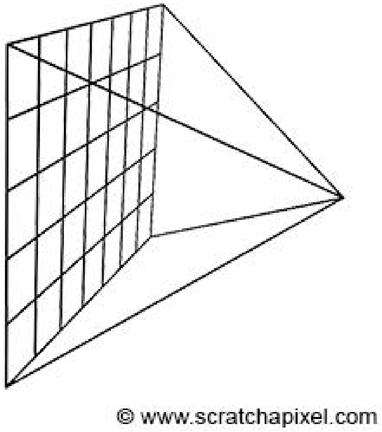
光线: $r(t) = o + td$











辐射场-体渲染

光线的颜色值公式

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

辐射场-体渲染

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

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n),\ t_n + \frac{i}{N}(t_f - t_n) \right]$$

Positional encoding

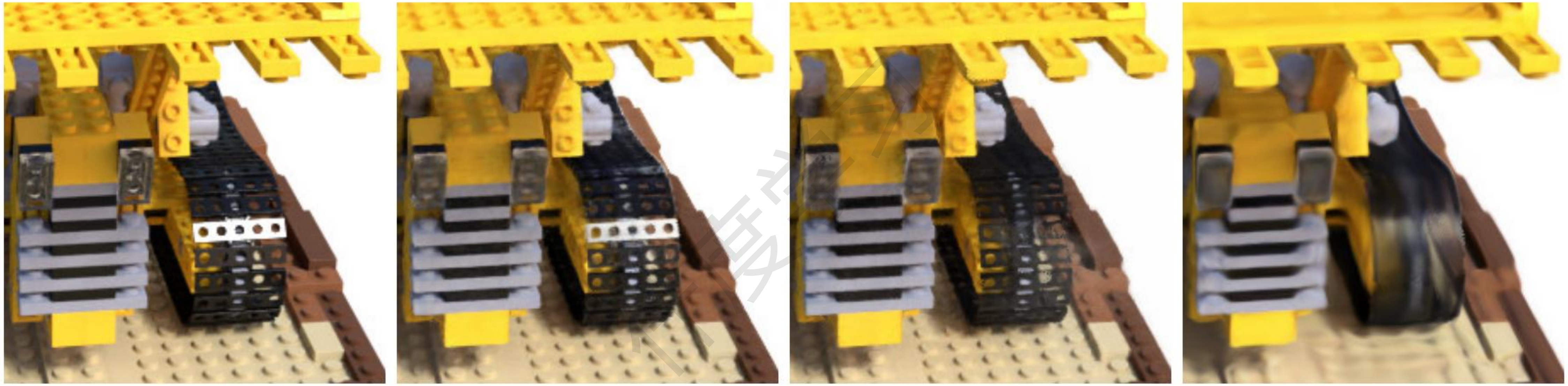
$$F_\Theta = F'_\Theta \circ \gamma$$

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

在实验中，空间坐标的三项 L=10，方向的两项 L=4

	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**

Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains



Ground Truth | Complete Model | No View Dependence | No Positional Encoding

对比Transformer PE

NeRF

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

Transformer

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

Hierarchical sampling

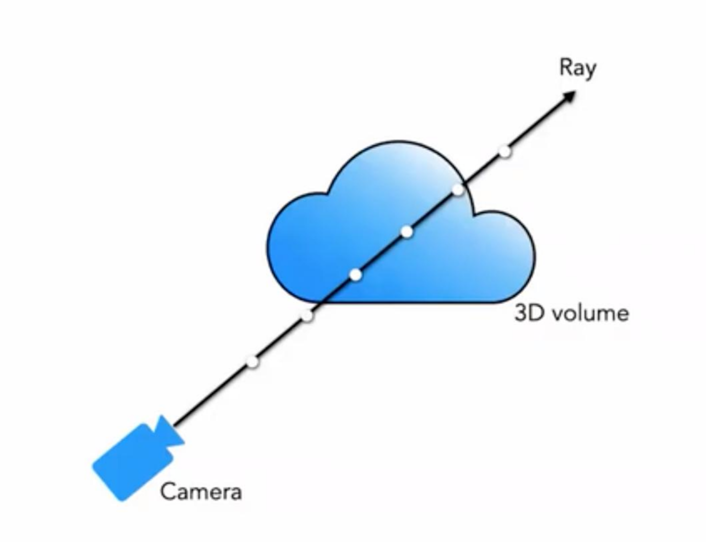
$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i \,, \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

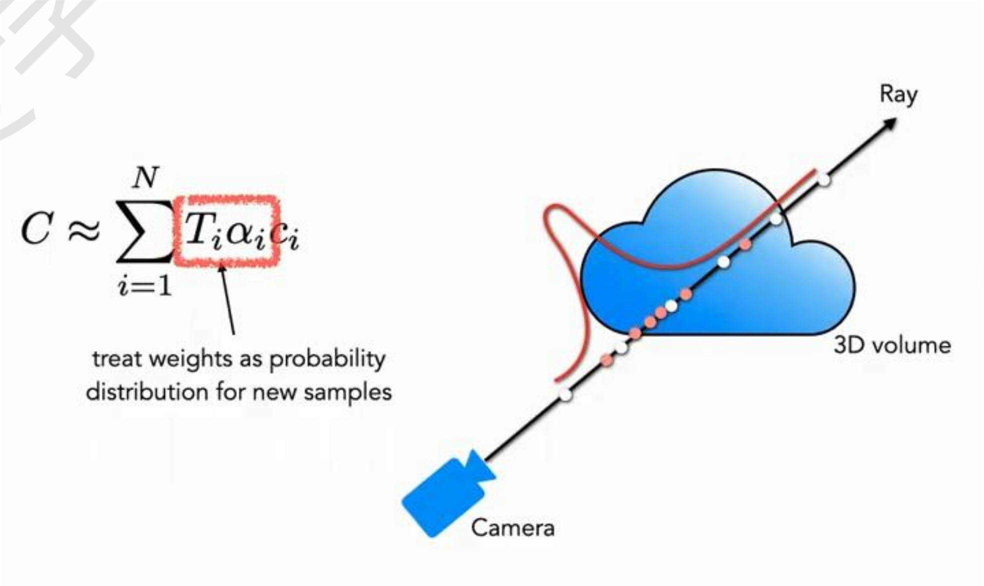
$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i c_i \,, \quad w_i = T_i(1 - \exp(-\sigma_i \delta_i))$$

Hierarchical sampling

$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i c_i \,, \quad w_i = T_i(1 - \exp(-\sigma_i \delta_i)) \quad \hat{w}_i = w_i / \sum_{j=1}^{N_c} w_j$$

权重可以看成沿着射线的分段常数概率密度函数 (Piecewise-constant PDF)





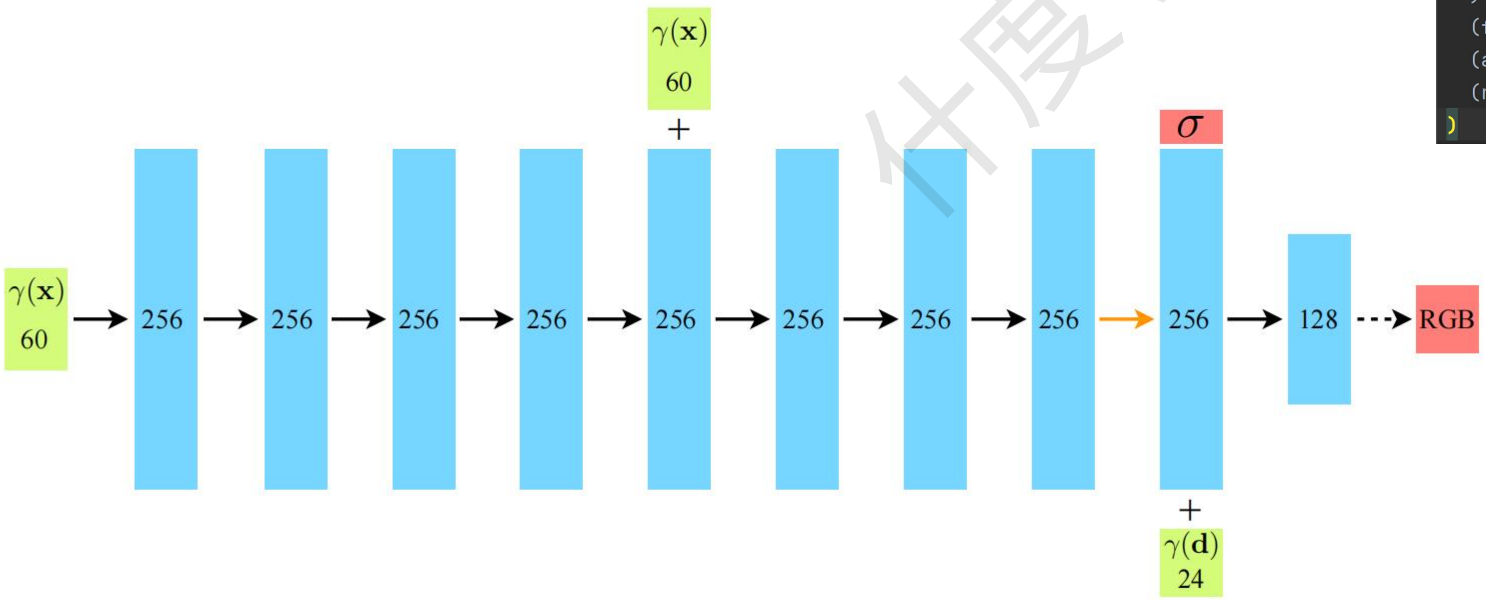
Hierarchical sampling

- 使用两层网络，第一次的计算为粗网络模型，第二次的计算为精细网络模型
- 粗网络模型的采样点位为64个，精细网络模型的采样点位数为64+128
- 一条光线的总点位数量为64+64+128=256

Implementation

Loss:

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




```
NeRF(
  (pts_linears): ModuleList(
    (0): Linear(in_features=63, out_features=256, bias=True)
    (1): Linear(in_features=256, out_features=256, bias=True)
    (2): Linear(in_features=256, out_features=256, bias=True)
    (3): Linear(in_features=256, out_features=256, bias=True)
    (4): Linear(in_features=256, out_features=256, bias=True)
    (5): Linear(in_features=319, out_features=256, bias=True)
    (6): Linear(in_features=256, out_features=256, bias=True)
    (7): Linear(in_features=256, out_features=256, bias=True)
  )
  (views_linears): ModuleList(
    (0): Linear(in_features=283, out_features=128, bias=True)
  )
  (feature_linear): Linear(in_features=256, out_features=256, bias=True)
  (alpha_linear): Linear(in_features=256, out_features=1, bias=True)
  (rgb_linear): Linear(in_features=128, out_features=3, bias=True)
)
```

数据集

- Synthetic renderings of objects
  - 合成的物体
  - 背景是透明的
  - 一张图像几百kb左右，像素800x800
- Real images of complex scenes
  - 生活中的真实图像
  - 复杂的目标以及背景
  - 一张图像几兆左右，像素4kx3k左右

数据集

合成图像数据集

- test
- train
- val
- .DS_Store
- transforms_test.json
- transforms_train.json
- transforms_val.json

真实图像数据集

- images
- images_4
- images_8
- sparse
- database.db
- poses_bounds.npy
- simplices.npy
- trimesh.png

评价指标

1. PSNR
2. SSIM
3. LPIPS

	Diffuse Synthetic 360° [41]			Realistic Synthetic 360°			Real Forward-Facing [28]		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SRN [42]	33.20	0.963	0.073	22.26	0.846	0.170	22.84	0.668	0.378
NV [24]	29.62	0.929	0.099	26.05	0.893	0.160	-	-	-
LLFF [28]	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

评价指标-PSNR

PSNR：Peak Signal to Noise Ratio 峰值信噪比

$$\mathrm{PSNR} = 10 \times lg\left(\frac{\mathrm{MaxValue}^2}{\mathrm{MSE}}\right)$$

值越大越好
MaxValue 为像素值的最大取值，为255

评价指标-SSIM

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = [l(\mathbf{x},\mathbf{y})]^{\alpha} \cdot [c(\mathbf{x},\mathbf{y})]^{\beta} \cdot [s(\mathbf{x},\mathbf{y})]^{\gamma}$$

值越大越好

$$l(\mathbf{x},\mathbf{y}) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \longrightarrow \text{亮度}$$

$$c(\mathbf{x},\mathbf{y}) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \longrightarrow \text{对比度}$$

$$s(\mathbf{x},\mathbf{y}) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \longrightarrow \text{结构}$$

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

评价指标-LPIPS

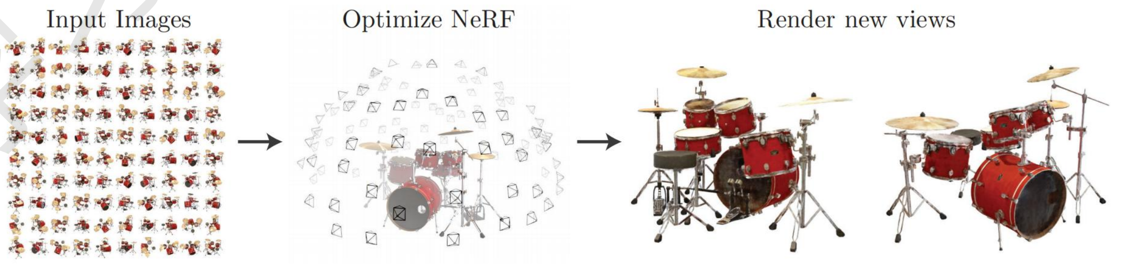
学习感知图像相似度

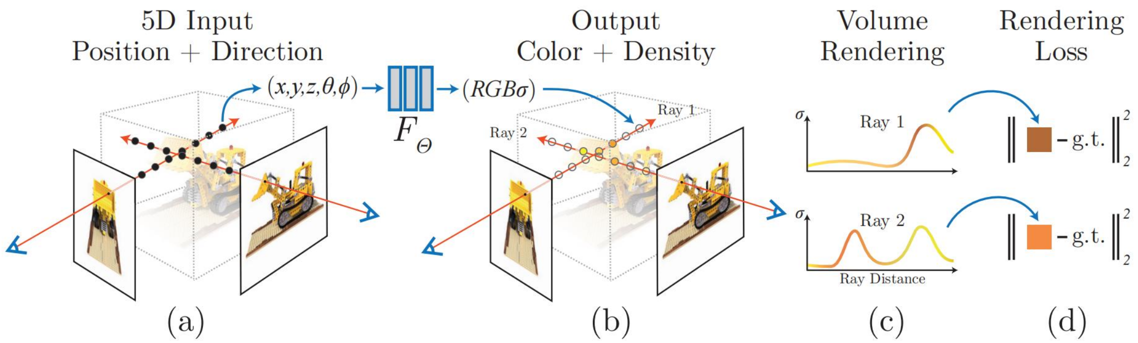
数值越小表示图像的相似性越高

$$d(x, x_0) = \sum_{l} \frac{1}{H_l W_l} \sum_{h,w} ||w_l \odot (\hat{y}^l_{hw} - \hat{y}^l_{0hw})||_2^2$$

NeRF 劣势

1. 它很慢，训练和推理都很慢
2. 它只能表示静态的场景
3. 对光照处理的不好
4. 训练的模型都仅能代表一个场景，没有泛化能力





参考

- https://github.com/yenchenlin/awesome-NeRF
- https://arxiv.org/abs/2101.05204 (survey)
- https://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-generating-camera-rays/definition-ray
- https://blog.csdn.net/zhuoqingjoking97298/article/details/122161124
- https://keras.io/examples/vision/nerf/
- https://blog.csdn.net/YuhsiHu/article/details/124318473 (NeRF的数学公式推导)