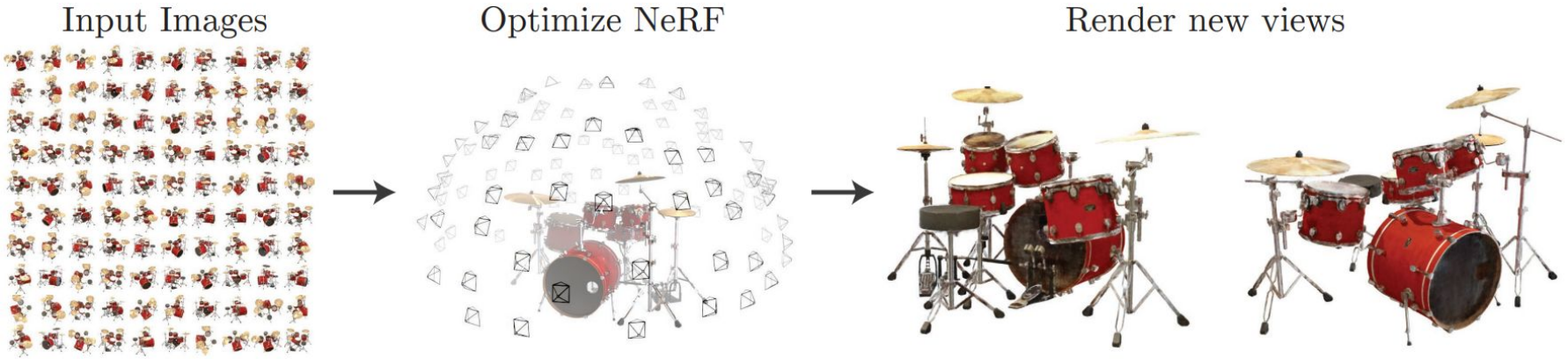
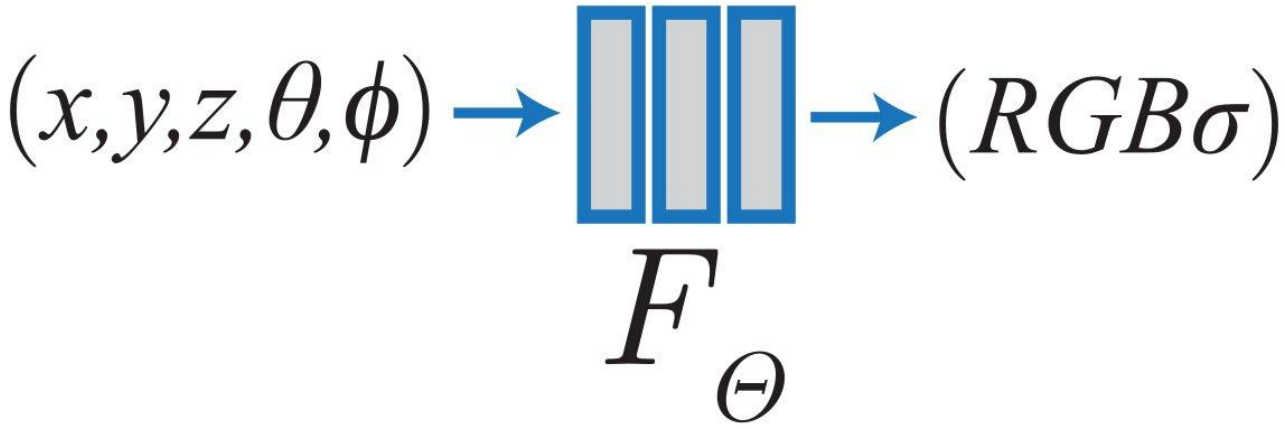
NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Ложкин Павел

Постановка задачи



Основная идея



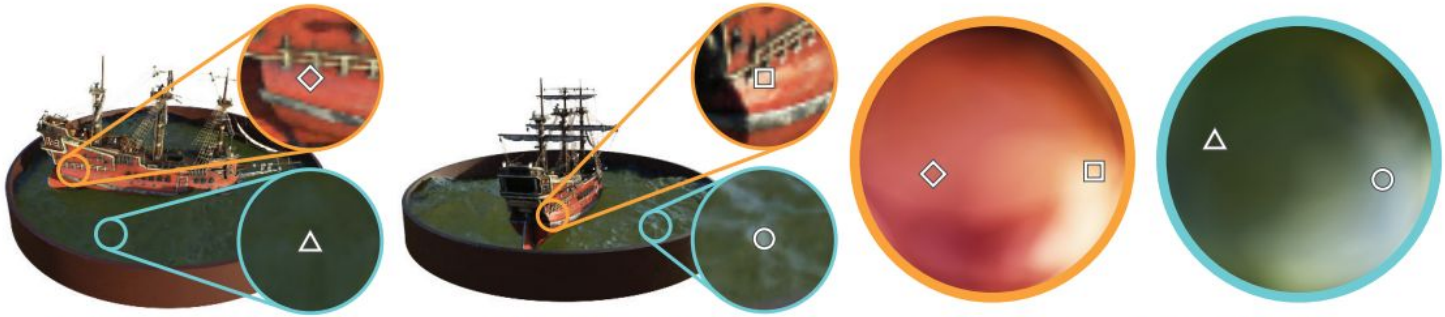
$\mathbf{x} = (x, y, z)$ - положение пикселя

$\sigma(\mathbf{x})$ - плотность пикселя

$\mathbf{d} = (\theta, \phi)$ - направление луча

$RGB(\mathbf{x}, \mathbf{d})$ - цвет пикселя

Основная идея



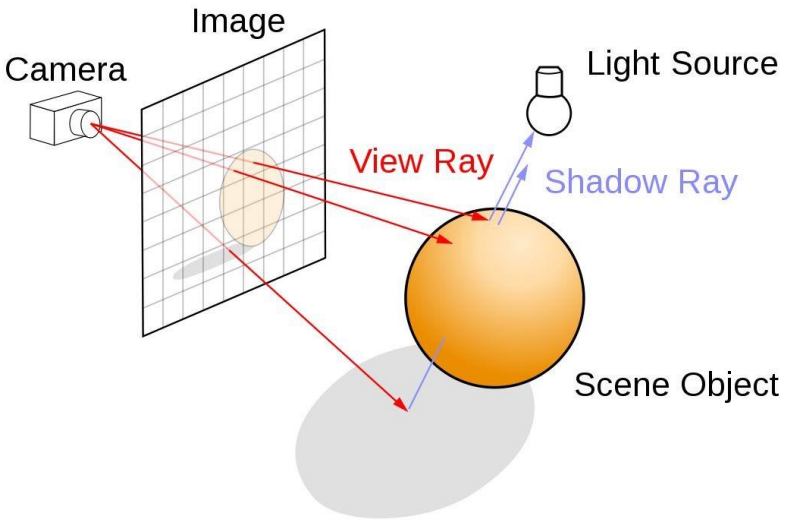
(a) View 1 (b) View 2 (c) Radiance Distributions

Получение картинки: Volume Rendering

Луч $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$

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

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$



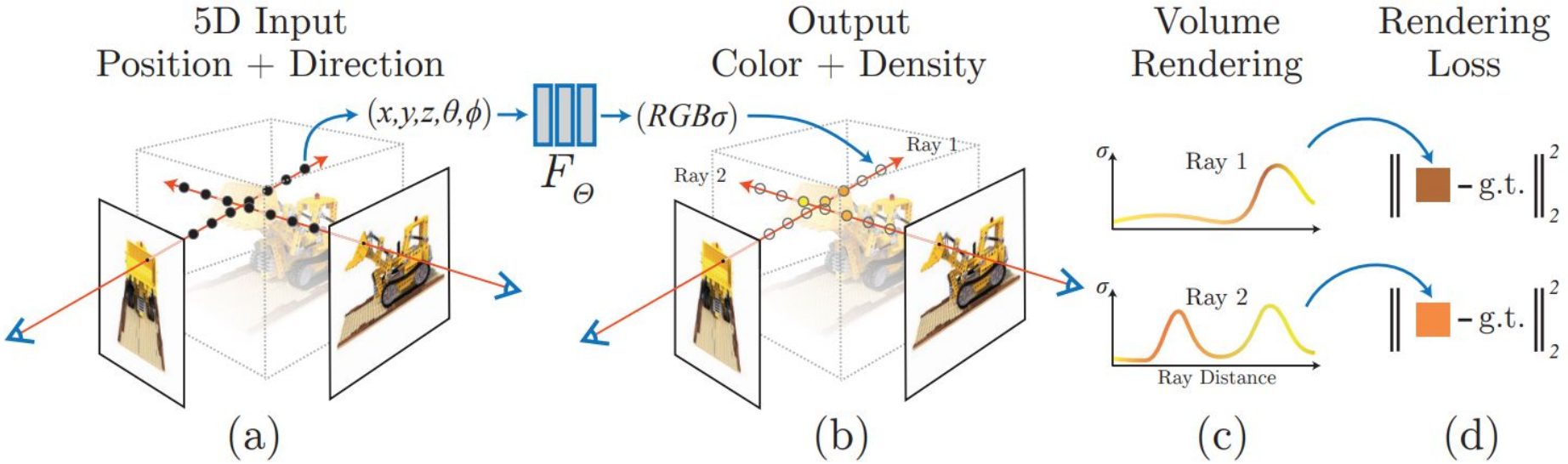
Получение картинки: Volume Rendering

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

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i (1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i \qquad \delta_i = t_{i+1} - t_i$$

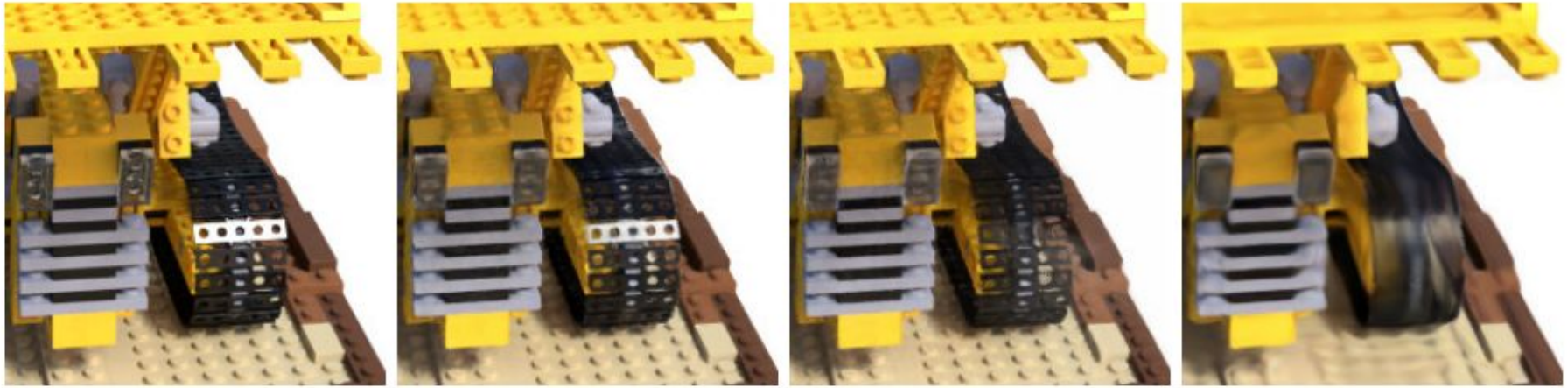
$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

Общий пайплайн



Оптимизации: Positional Encoding

- Модель плохо передает высокочастотные изменения цветов



Оптимизации: Positional Encoding

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

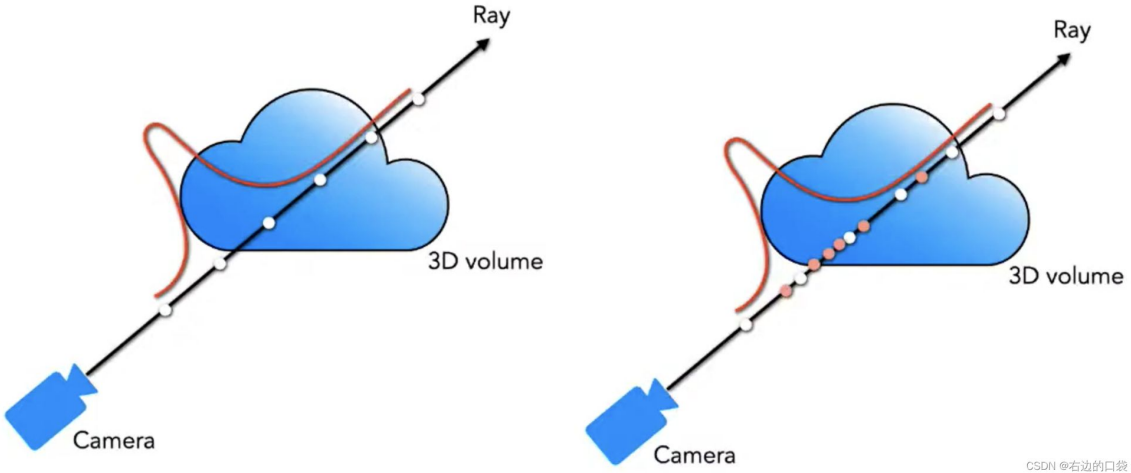
Нормализуем координаты (x, y, z) до отрезка [-1, 1]

$$(x, y, z) \rightarrow (\gamma_{10}(x), \gamma_{10}(z), \gamma_{10}(y))$$

$$(d_1, d_2, d_3) \rightarrow (\gamma_4(d_1), \gamma_4(d_2), \gamma_4(d_3))$$

Оптимизации: Hierarchical Volume Sampling

- Мы семплируем слишком много точек, которые расположены просто в воздухе или внутри объекта и не вносят вклад в итоговый цвет



Оптимизации: Hierarchical Volume Sampling

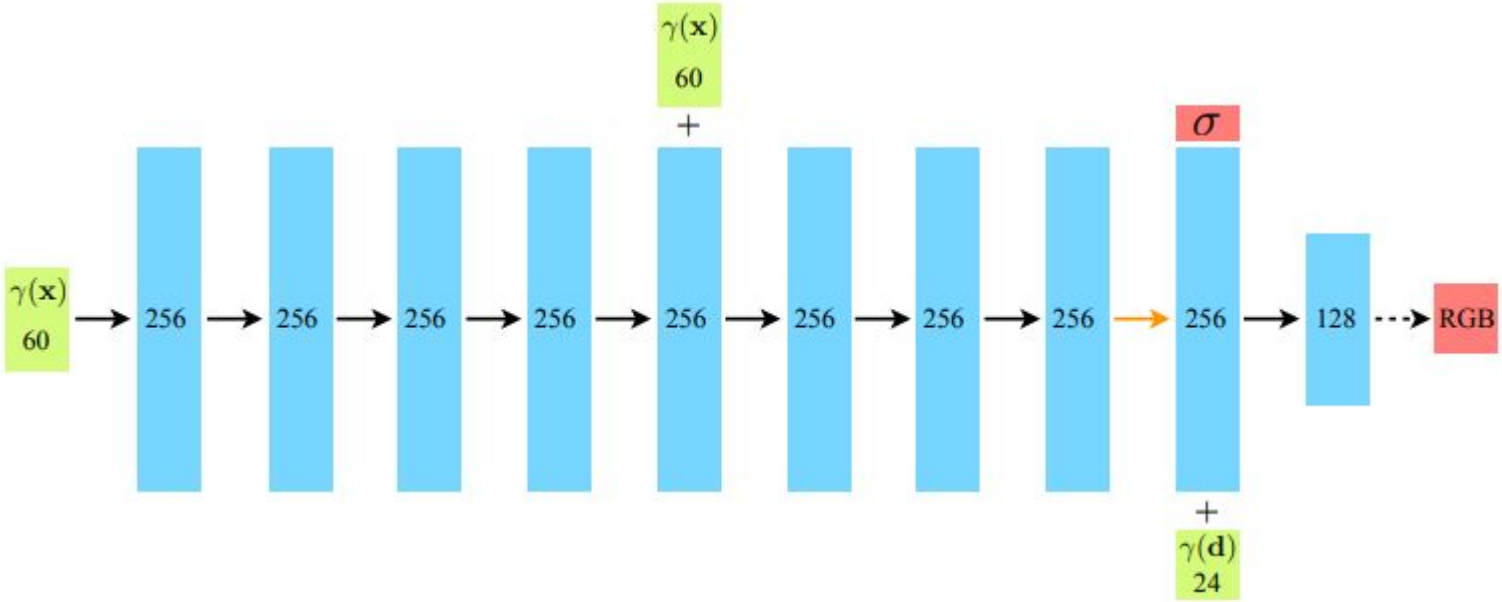
$$\hat{C}_c(\mathbf{r}) = \sum_{i=1}^{N_c} w_i c_i \qquad w_i = T_i(1 - \exp(-\sigma_i \delta_i))$$

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

По $\hat{w}$ можно построить новую кусочно-линейную функцию плотности и насемплировать из нее еще N_f точек

По получившимся $N_c + N_f$ точкам строим предсказание $\hat{C}_f(\mathbf{r})$ модели

Итоговая архитектура



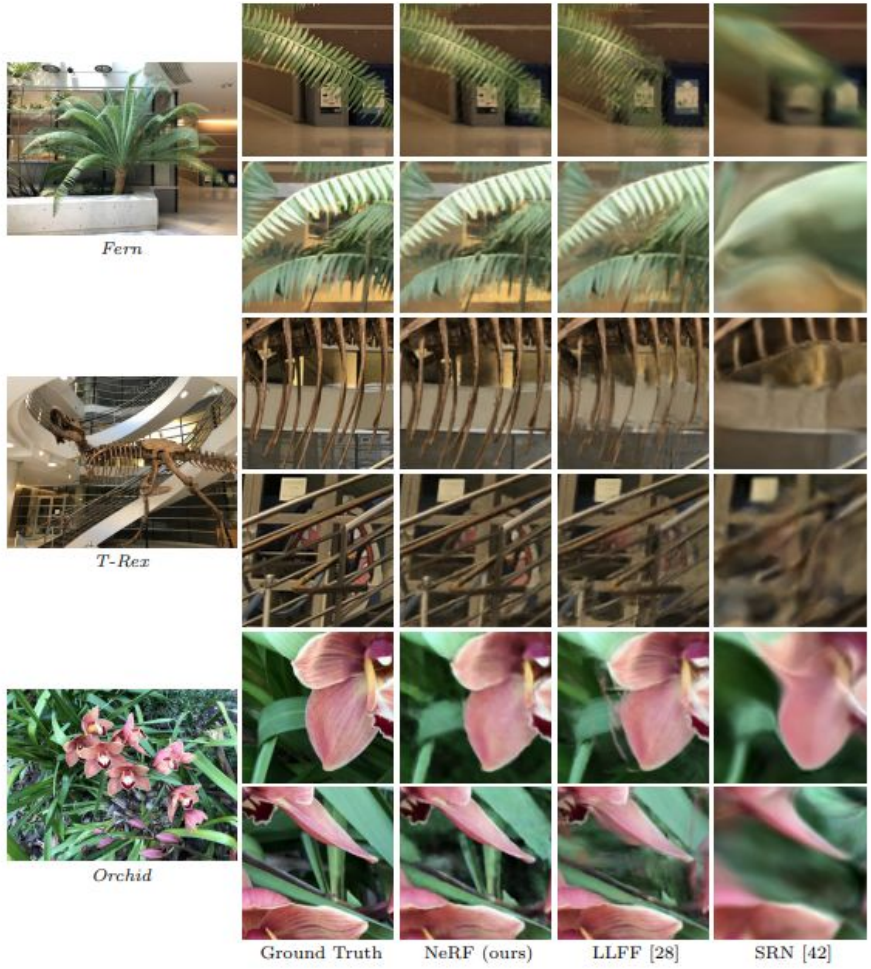
$$\mathcal{L} = \sum_{\mathbf{r}\in\mathcal{R}} \left[\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

Сравнение с другими моделями

Method	Diffuse Synthetic 360° [41]			Realistic Synthetic 360°			Real Forward-Facing [28]		
	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

Neural Volumes (NV) - свертки и volume render
Scene Representation Networks (SRN) - рекурентный volume render
Local Light Field Fusion (LLFF) - 3d свертки, усреднение векторов признаков близлежащих камер


Ship
Lego
Microphone
Materials
Ground Truth
NeRF (ours)
LLFF [28]
SRN [42]
NV [24]


Fern
T-Rex
Orchid
Ground Truth
NeRF (ours)
LLFF [28]
SRN [42]

Проблемы NeRF

- 1-2 дня обучения на одну сцену (InstantNeRF)
- Необходимо одинаковое освещение на тренировочных данных (NeRF-w)
- Неизменяемый источник света (NeRV)
- Долгий инференс (InstantNeRF)
- Отдельная модель на каждую сцену (PixelNeRF)
- Сильное ухудшение качества из-за неточных позиций камер (NeRF--, BaRF)

Новые модели

zip-NeRF

InstantNeRF