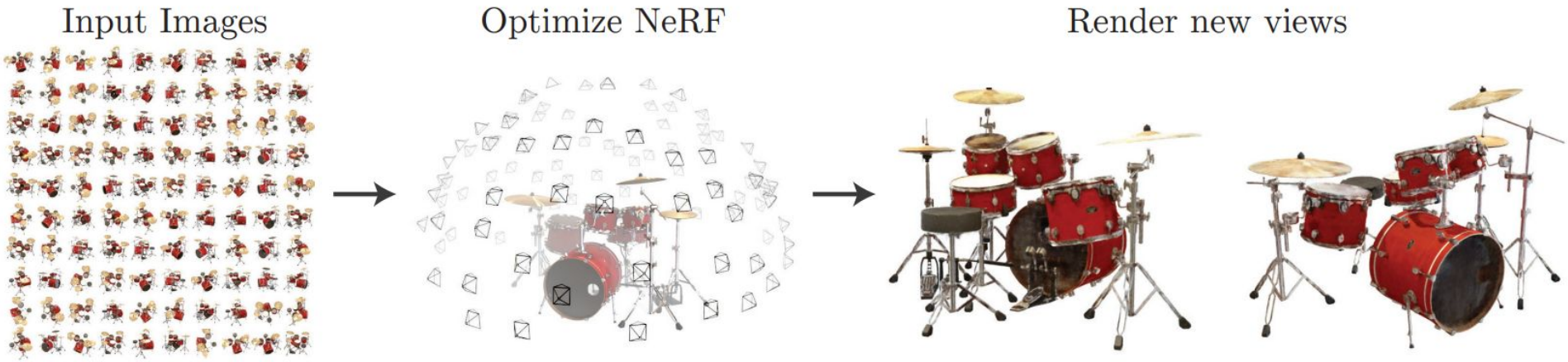
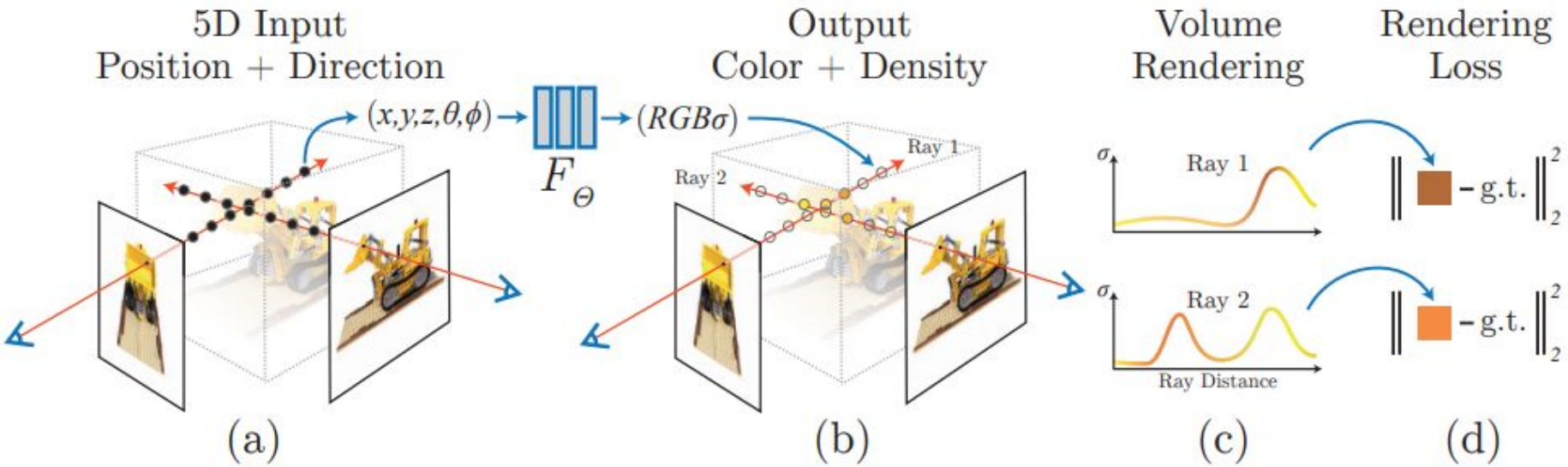
NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Желтовская Юлия, БПМИ202

Цель NeRF: Визуализировать 3D-сцену



Процесс обучения



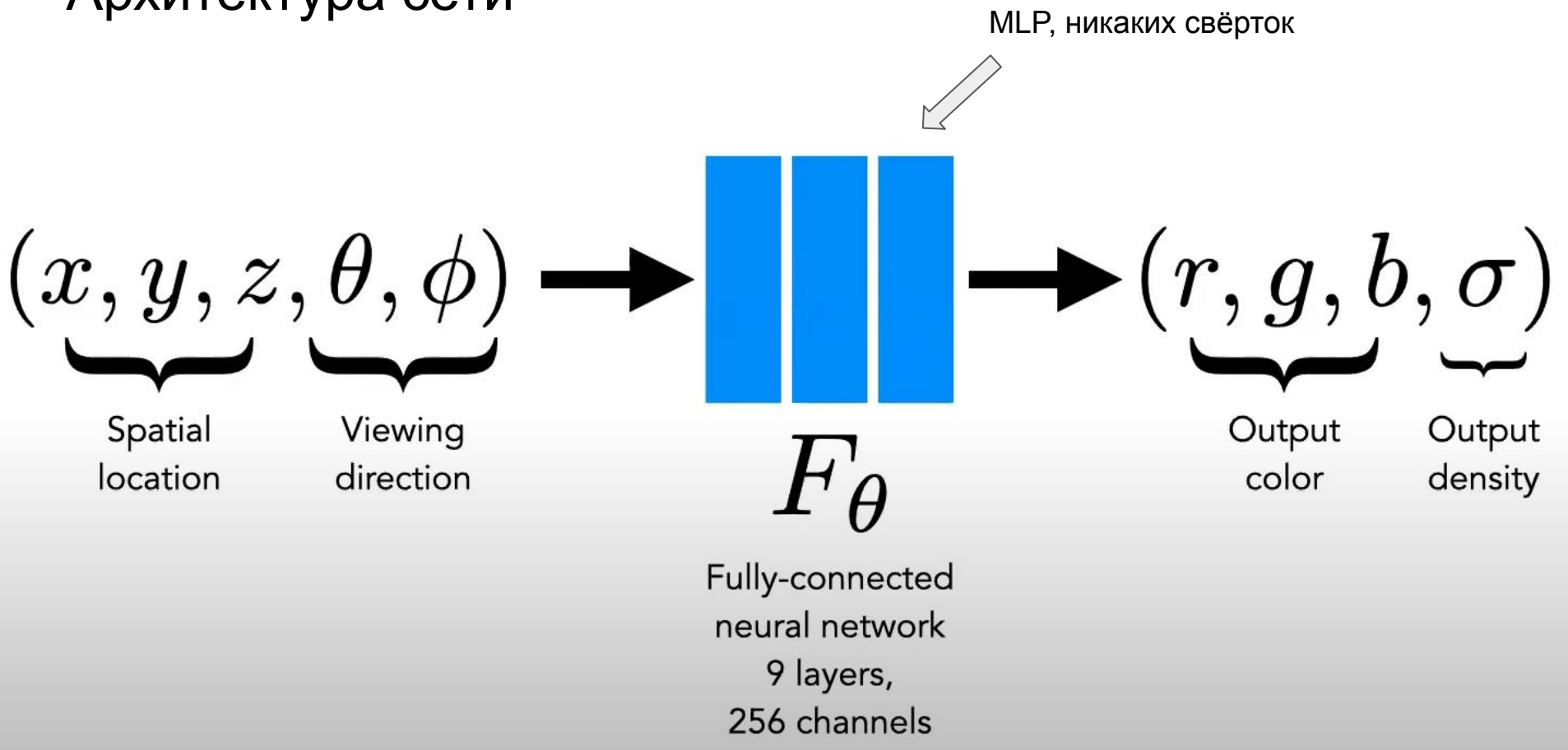
$\mathbf{x} = (x, y, z)$ — точка в 3D-пространстве (spatial location)

(θ, ϕ) — направление взгляда (viewing direction)

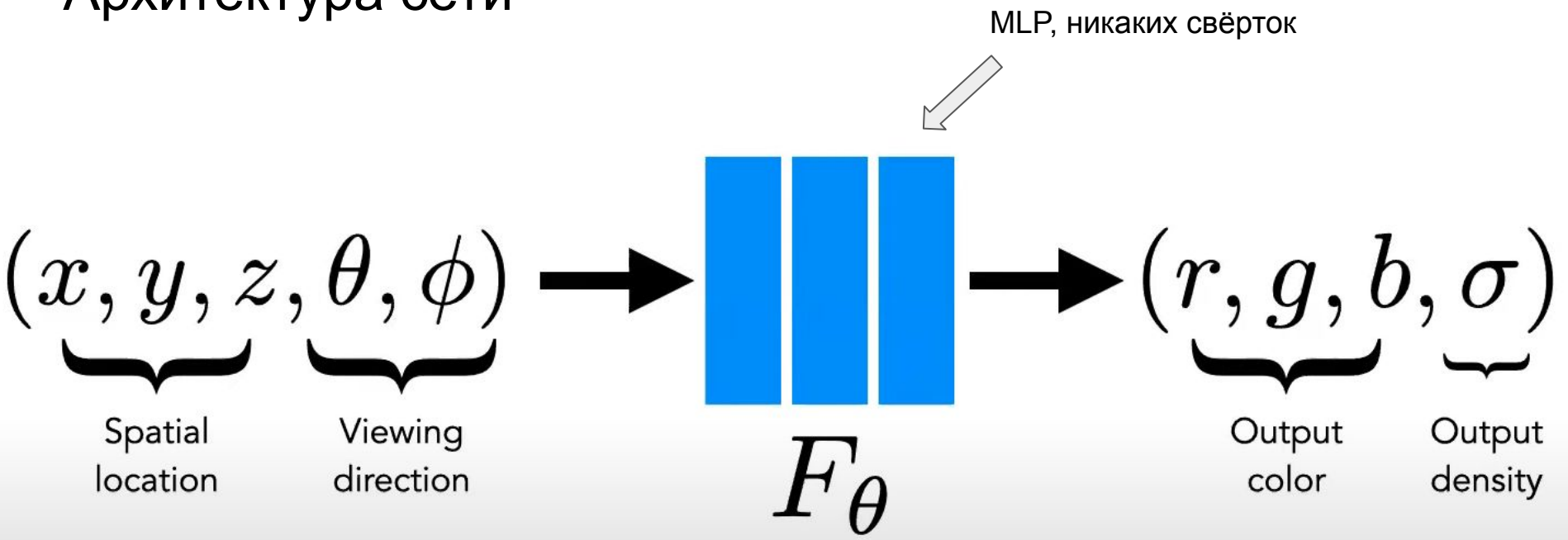
(r, g, b) — цвет в точке $\mathbf{x}$

σ — плотность в точке (volume density)

Архитектура сети

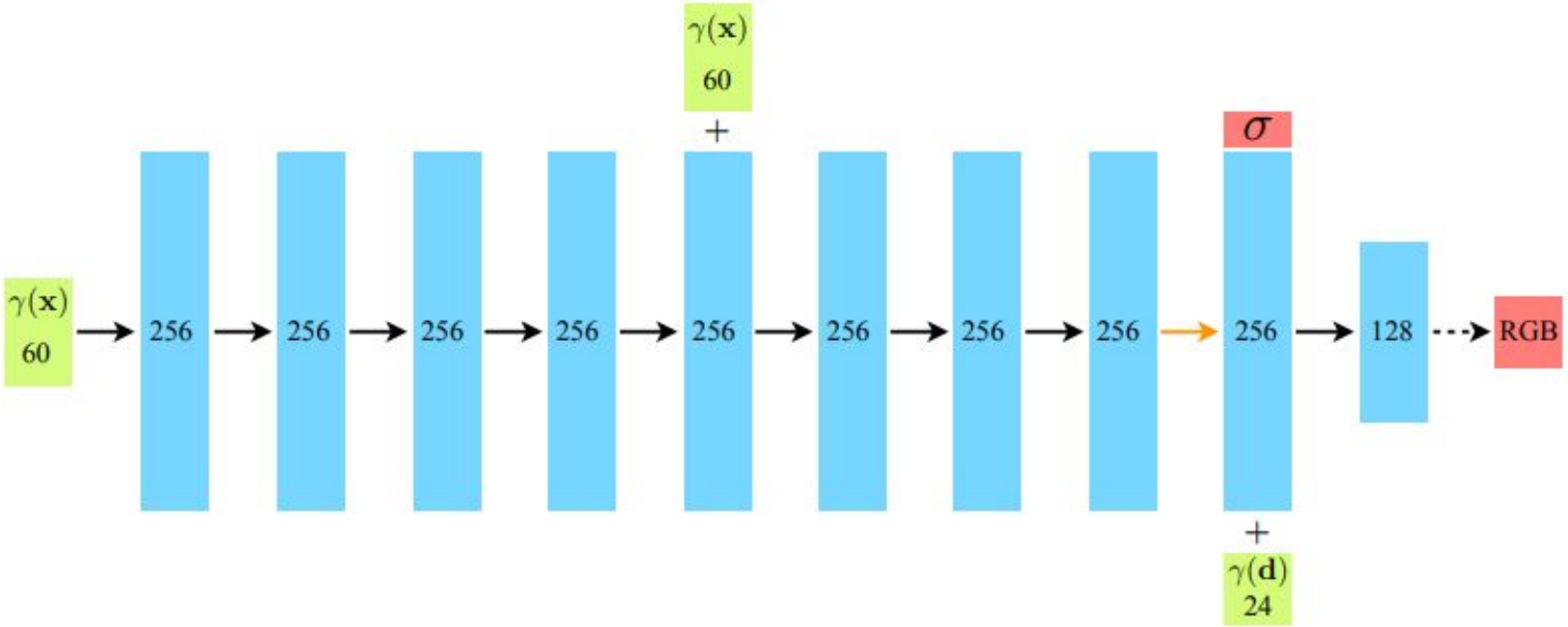


Архитектура сети



Направление взгляда влияет только на цвет. Плотность строится для точки пространства вне зависимости от угла осмотра!

Визуализация архитектуры

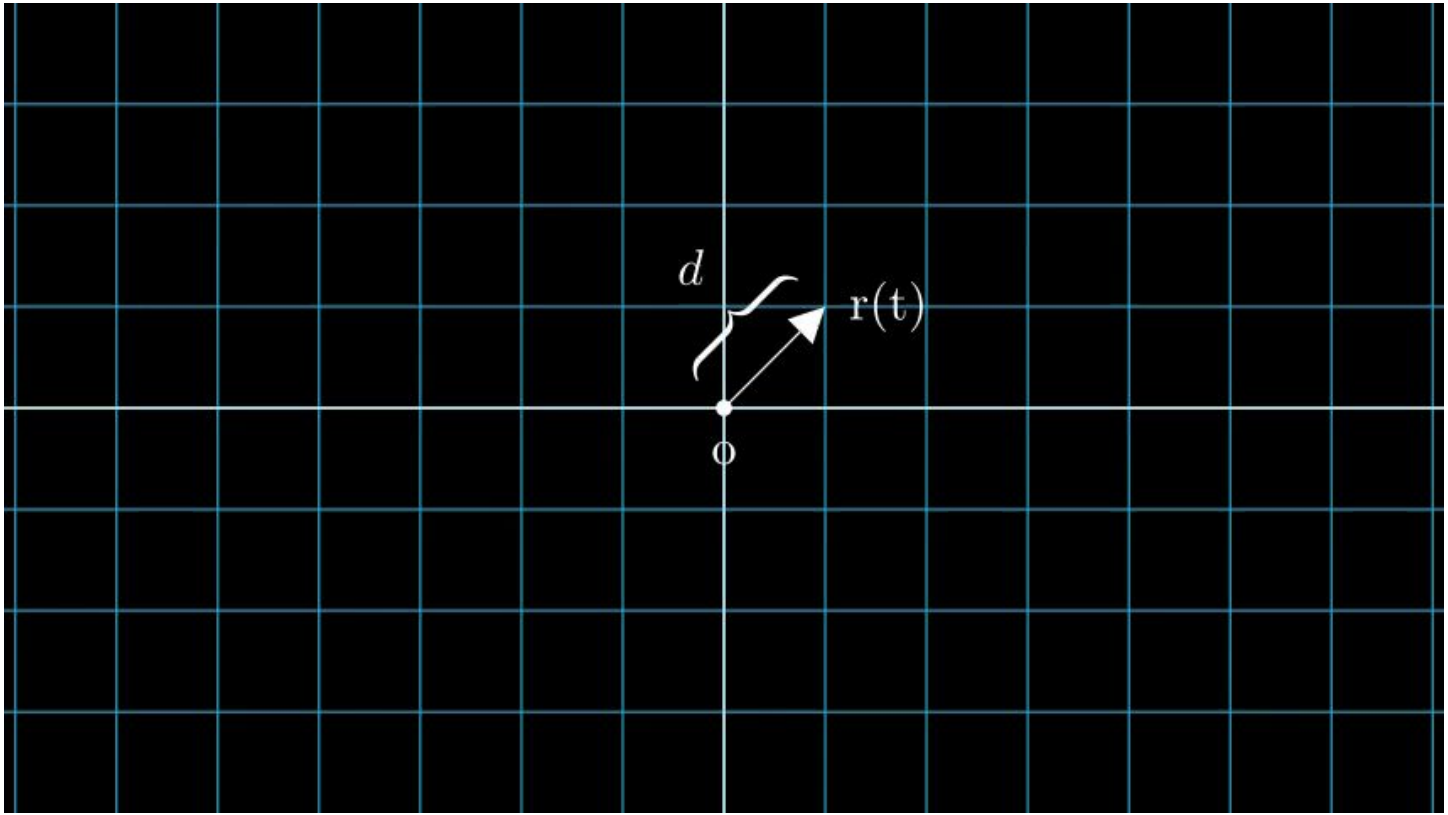


Viewing direction

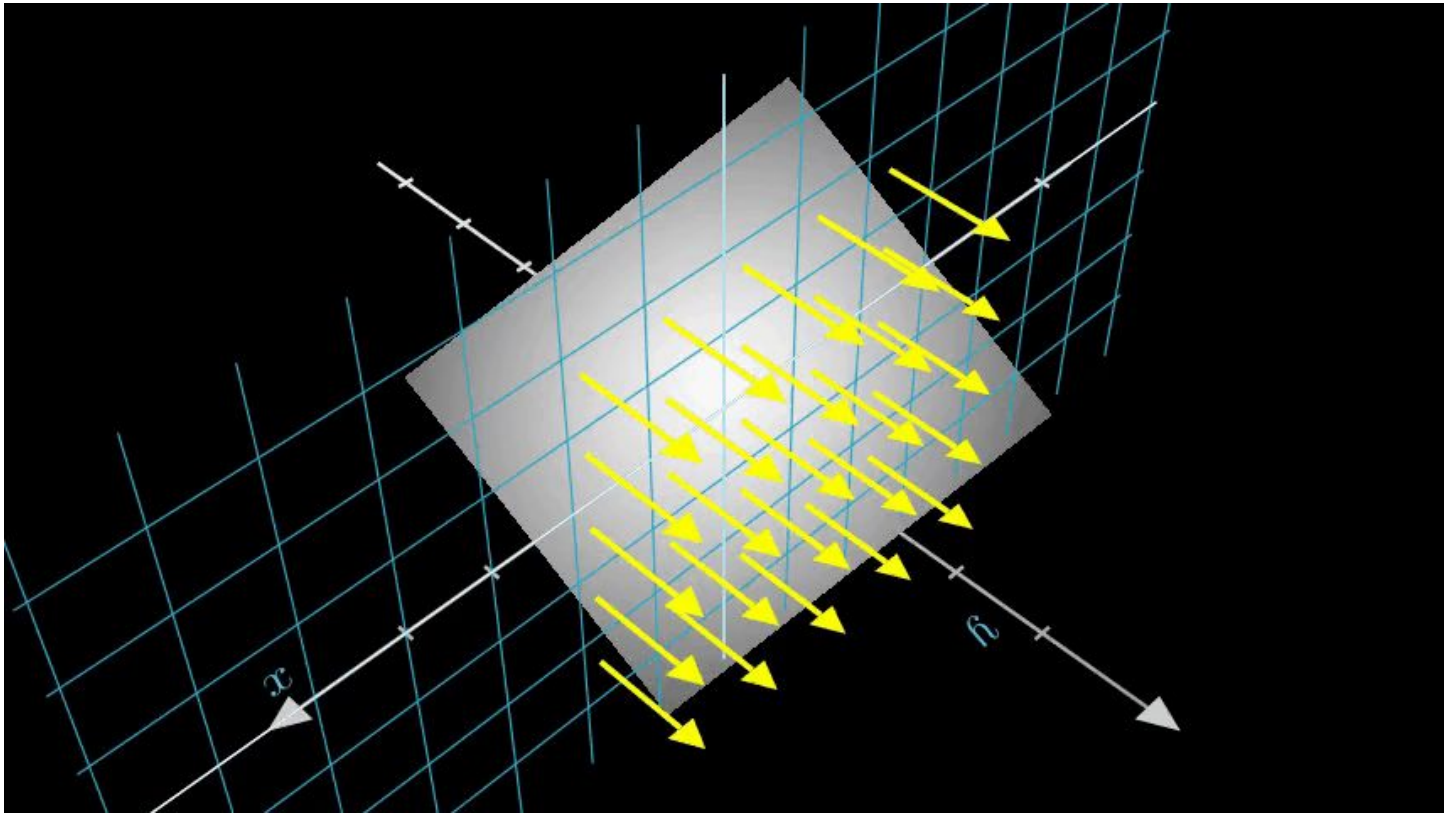
На практике вместо двух углов (θ, ϕ) используется вектор $\mathbf{d}$ в декартовой системе координат с $r = 1$

$$\overline{x} = r \sin\theta \cos\phi$$
$$\overline{y} = r \sin\theta \sin\phi$$
$$\overline{z} = r \cos\theta$$

$$\mathbf{d} = (\overline{x}, \overline{y}, \overline{z})$$



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



Shooting rays from all the pixels of an image in 3D

Volume Rendering

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

Предсказанный цвет, величина для минимизации функции потерь

Плотность в точке

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

Цвет в точке r(t) с позиции d

Вероятность того, что луч пройдет из tn в t, не задев другие частицы

Приближение интеграла

Quadrature rule in volume rendering review by Max, N[1]:

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

$\delta_i = t_{i+1} - t_i$ – расстояние между соседними семплами на луче

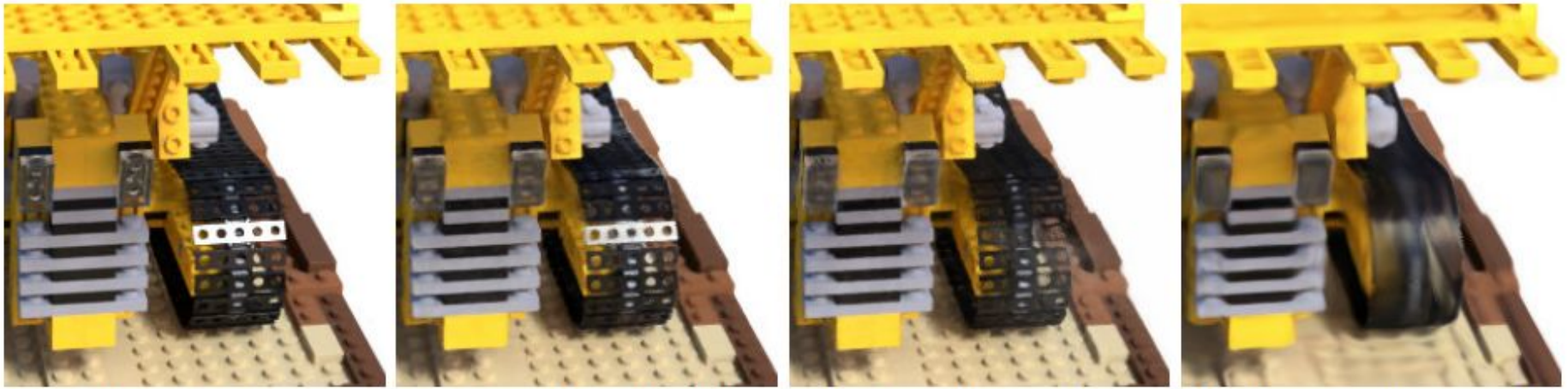
$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n),\; t_n + \frac{i}{N}(t_f - t_n) \right]$$

1. Max, N.: Local and Global Illumination in the Volume Rendering Integral

Sampling the points from a ray

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

Однако базового подхода недостаточно



Positional encoding

Проблема: сеть, работающая только с входными xyzθφ координатами слабо визуализирует высокочастотные изменения цвета и геометрии.

Это согласуется с недавней работой Rahaman et al. [1], где показывается, что глубокие сети предвзяты в сторону низкочастотных функций.

1. Rahaman et al. On the spectral bias of neural networks. In: ICML (2018)

Positional encoding

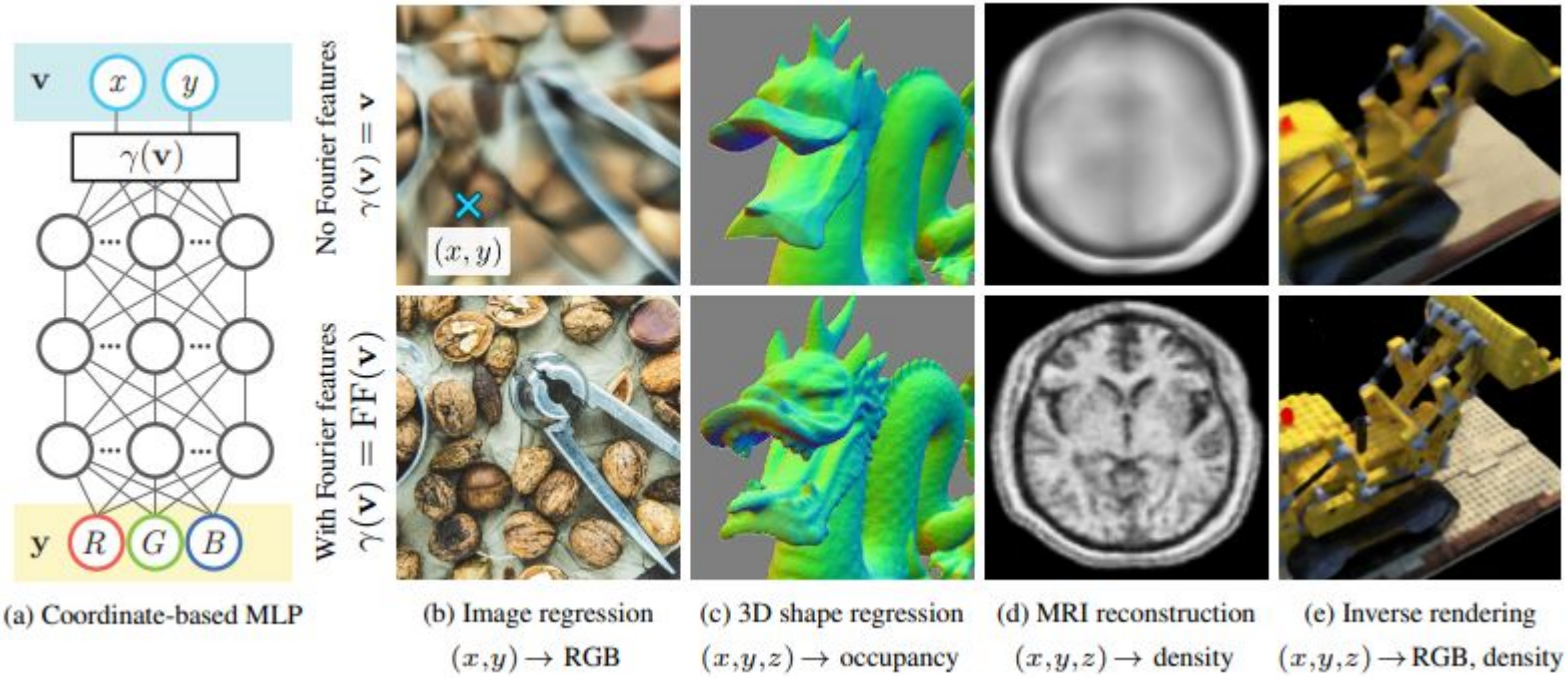
Используем дополнительное преобразование :

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

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

$$L = 10 \text{ for } \gamma(\mathbf{x}) \text{ and } L = 4 \text{ for } \gamma(\mathbf{d})$$

$\gamma(\cdot)$ применяется отдельно ко всем координатам $\mathbf{x}$, нормализованным до [-1, 1].
И отдельно к трём координатам вектора $\mathbf{d}$, лежащим в [-1, 1] по построению.



(a) Coordinate-based MLP

(b) Image regression
$(x,y) \rightarrow$ RGB

(c) 3D shape regression
$(x,y,z) \rightarrow$ occupancy

(d) MRI reconstruction
$(x,y,z) \rightarrow$ density

(e) Inverse rendering
$(x,y,z) \rightarrow$ RGB, density

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

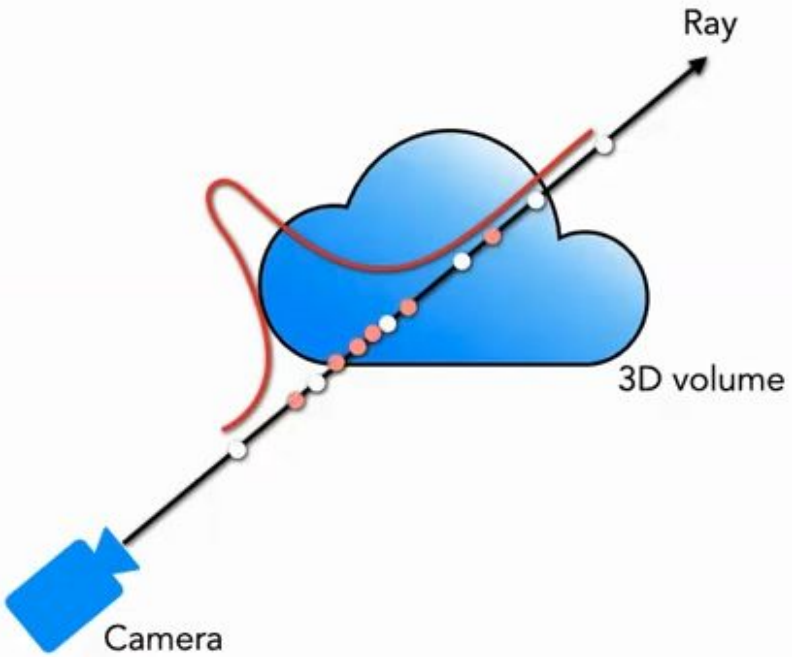
Hierarchical volume sampling

Проблема: оценивать цвет по N точкам, взятым на луче, может быть неэффективно, так как попадают свободные пространства, закрытые области – не влияющие на визуализацию.

Hierarchical volume sampling

$$C \approx \sum_{i=1}^{N} T_i \alpha_i c_i$$

treat weights as probability distribution for new samples



Hierarchical volume sampling

Будут строиться два типа предсказаний: coarse и fine.

Coarse считается как и раньше для N_c позиций на луче.

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

Нормализуя веса $\hat{w}_i = w_i / \sum_{j=1}^{N_c} w_j$, получаем кусочно-заданную функцию плотности.
Затем семплируется ещё N_f точек из этого распределения.
Теперь оцениваем цвет для $N_c + N_f$ позиций, и предсказываем цвет $\hat{C}_f(\mathbf{r})$

Loss function

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

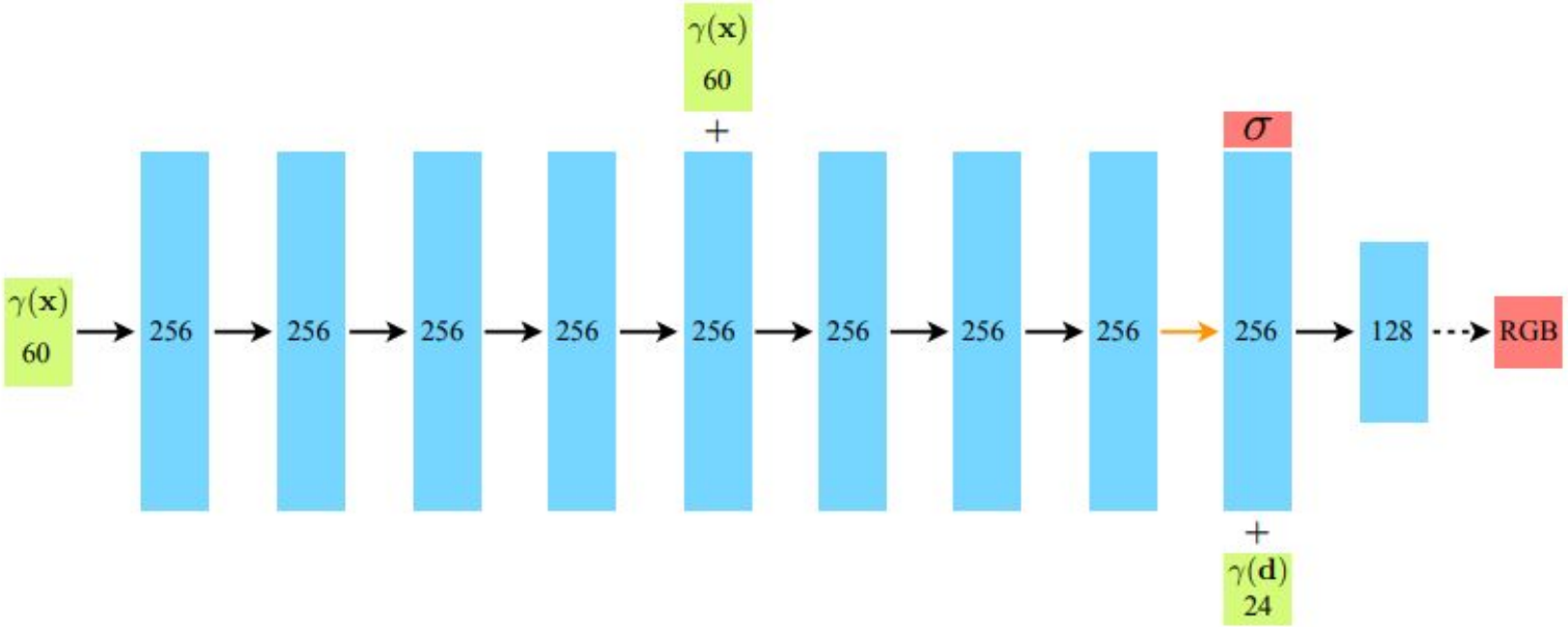
$\mathcal{R}$ — набор лучей в батче

$\hat{C}_c(\mathbf{r})$ — цвет, предсказанный coarse

$\hat{C}_f(\mathbf{r})$ — цвет, предсказанный fine

$C(\mathbf{r})$ — настоящий цвет

Визуализация архитектуры



Гиперпараметры

Размер батча: 4096 лучей, для каждого $N_f = 128$ $N_c = 64$ семплов для coarse volume и fine volume.

В качестве оптимизатора используется Adam (с дефолтными гиперпараметрами), learning rate от 5 × 10−4, экпоненциально меняющийся до 5 × 10−5.

Для одной сцены требуется 100–300k итераций до сходимости на NVIDIA V100 GPU (около 1–2 дней).

Эксперименты

	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**

Сравнение с другими методами

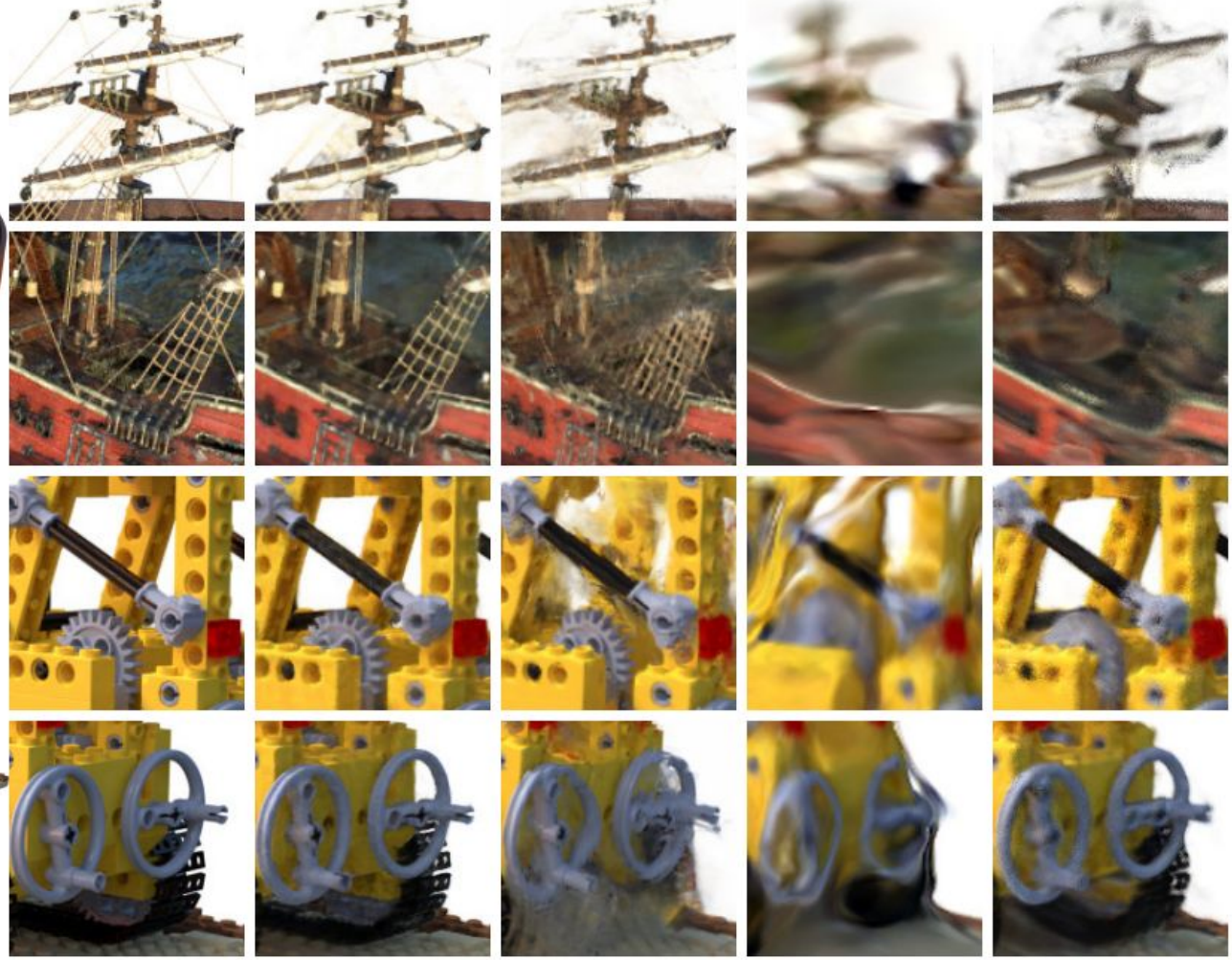
Method	Diffuse Synthetic 360° [41]			Realistic Synthetic 360°			Real Forward-Facing [28]		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SRN [42]	33.20	0.963	0.073	22.26	0.846	0.170	22.84	0.668	0.378
NV [24]	29.62	0.929	0.099	26.05	0.893	0.160	-	-	-
LLFF [28]	34.38	0.985	0.048	24.88	0.911	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

Методы: Scene Representation Networks (SRN), Neural Volumes (NV), Local Light Field Fusion (LLFF)

Метрики: PSNR (Peak Signal to Noise Ratio), SSIM (structural similarity index), LPIPS (Learned Perceptual Image Patch Similarity)

Ship

Lego



Ground Truth NeRF (ours) LLFF [28] SRN [42] NV [24]

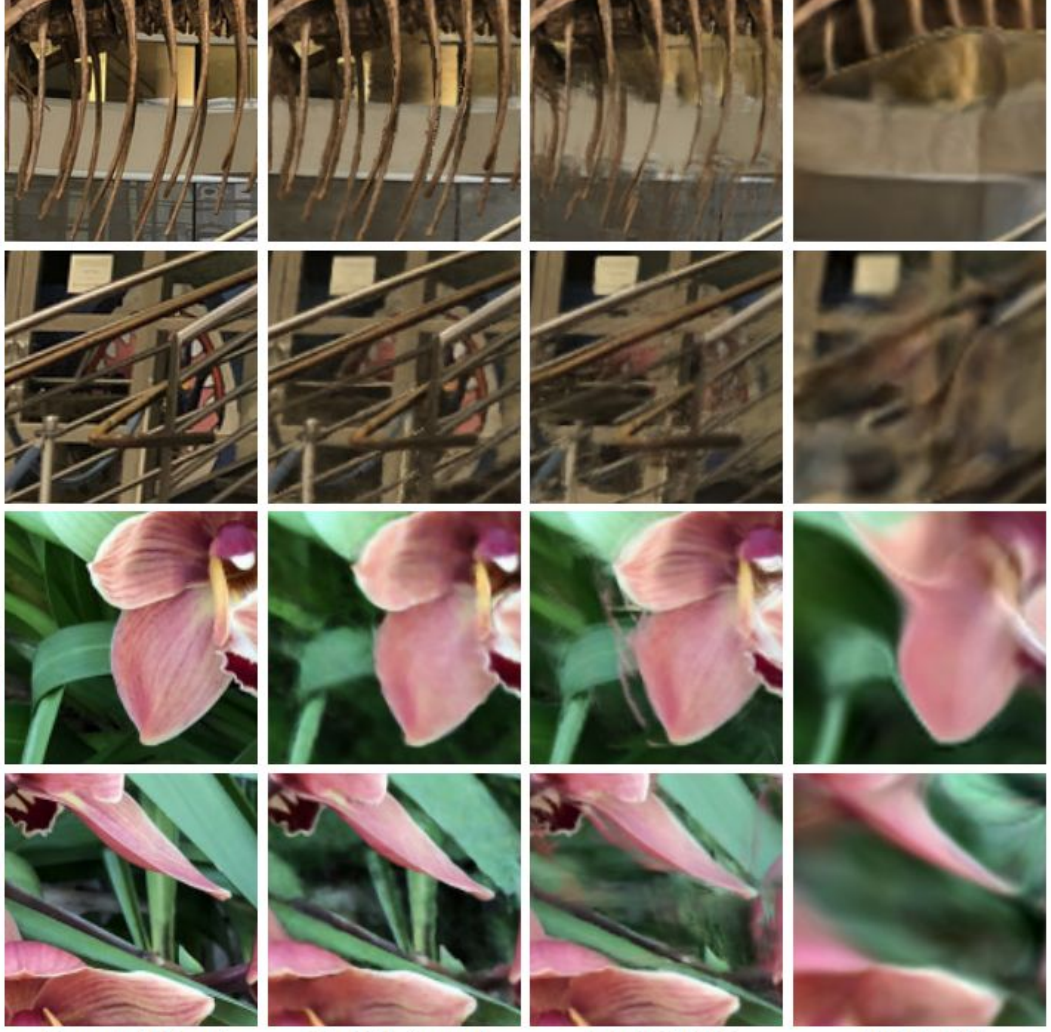
Microphone

Materials

Ground Truth NeRF (ours) LLFF [28] SRN [42] NV [24]

T-Rex

Orchid



Ground Truth NeRF (ours) LLFF [28] SRN [42]

Более новые работы

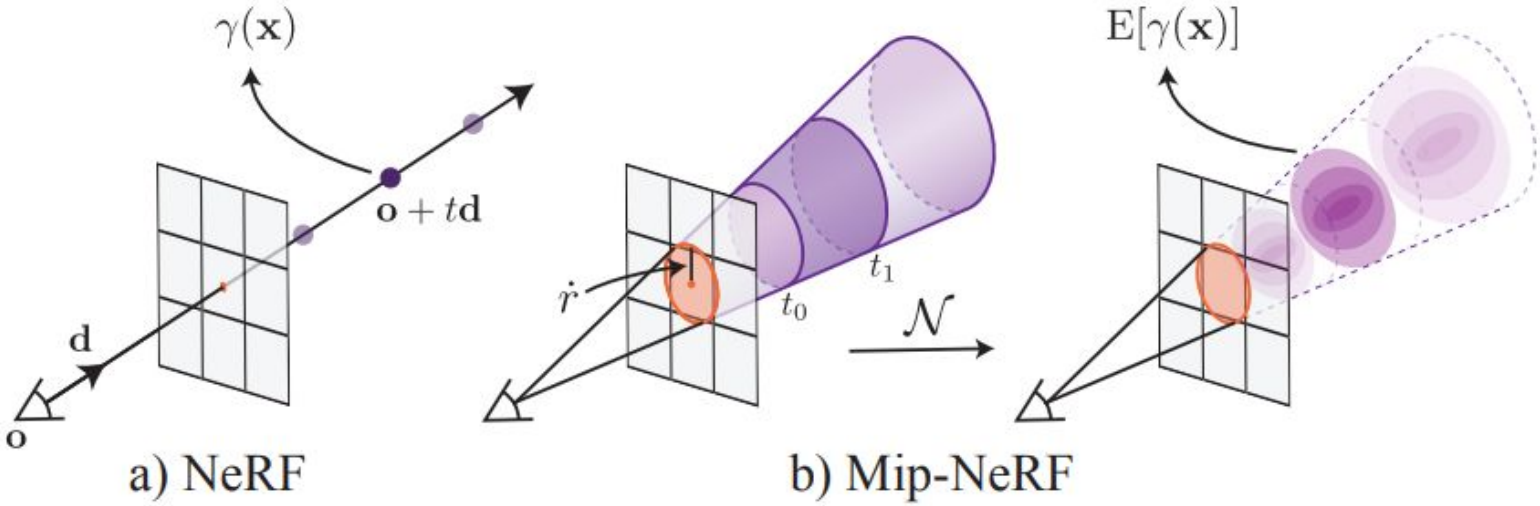
1. Zhang et al. NeRF++: Analyzing and Improving Neural Radiance Fields. 2020 – https://arxiv.org/abs/2010.07492

Рассматривает проблемы оригинального NeRF. Улучшает его на визуализацию неограниченных 3D сцен.

Более новые работы

2. Barron et al. Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields. 2021 – https://arxiv.org/abs/2103.13415

Меняют идею запускать лучи, используют усечённые конусы. Улучшают качество и делают NeRF в разы быстрее.



Более новые работы

3. InstantNeRF от NVIDIA. 2022 – https://github.com/NVlabs/instant-ngp

Статья: MÜLLER et al. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. 2022 – https://nvlabs.github.io/instant-ngp/assets/mueller2022instant.pdf

Источники

1. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis – https://arxiv.org/abs/2003.08934
2. Больше примеров картинок и код – https://www.matthewtancik.com/nerf
3. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains – https://arxiv.org/abs/2006.10739
4. 3D volumetric rendering with NeRF – https://keras.io/examples/vision/nerf/