NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

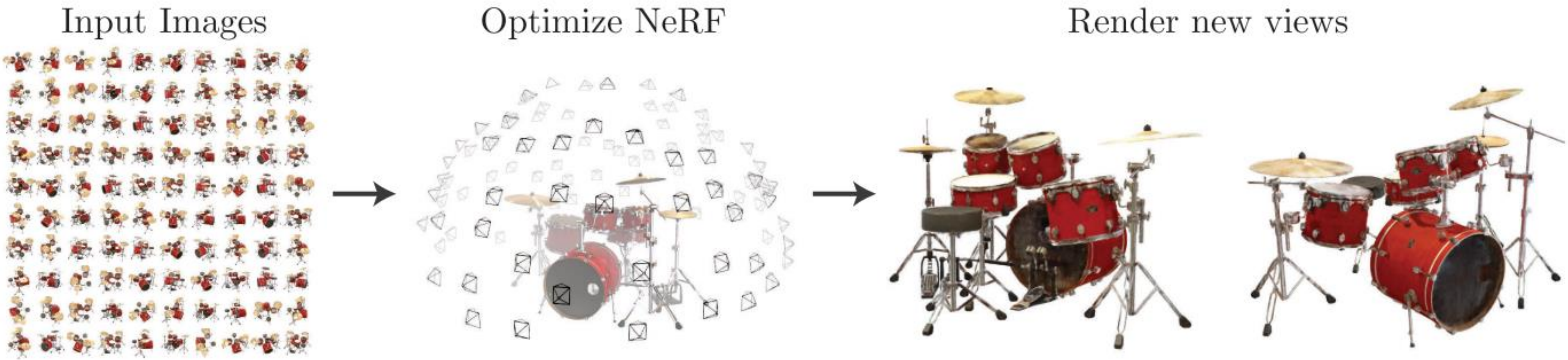
Introduction – Task definition

Task: Real-world view synthesis

Inputs: sparsely sampled images

Outputs: synthesis views of the scene

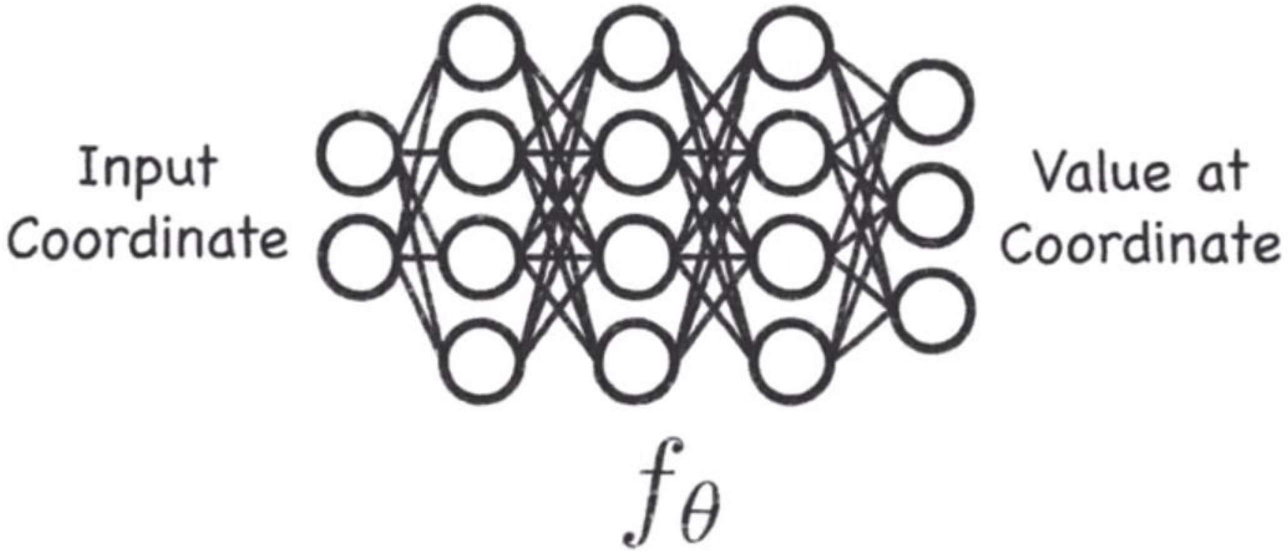
Introduction – Task definition



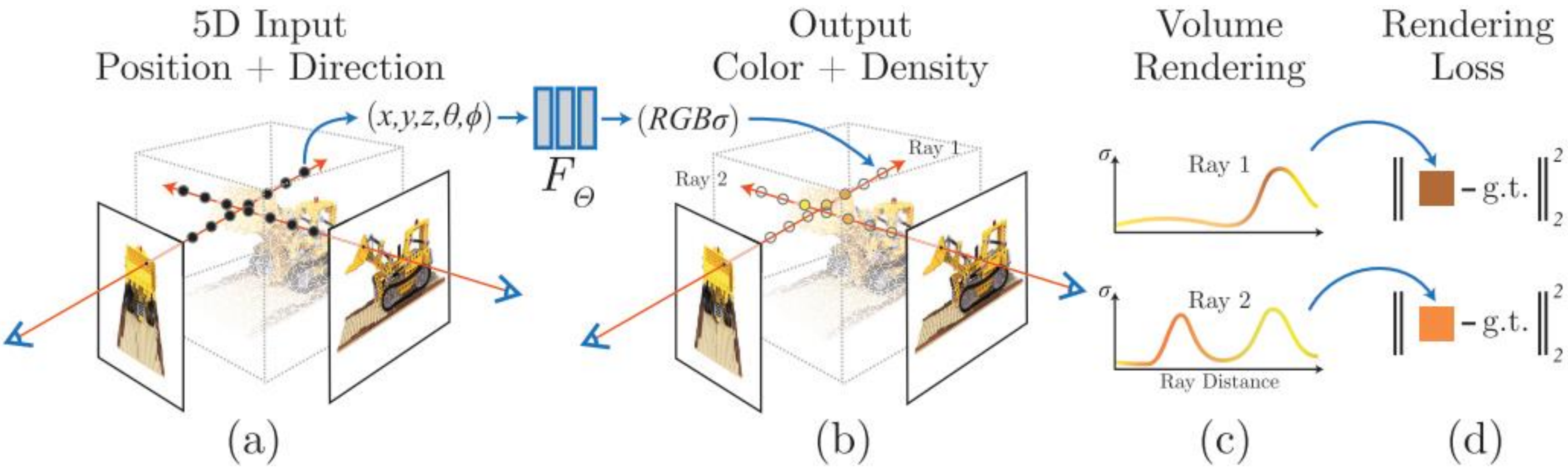
Contribution

- An approach for representing continuous scenes with complex geometry and materials as **5D neural radiance fields, parameterized as basic MLP networks**.
- A differentiable rendering procedure based on classical volume rendering techniques. This includes a **hierarchical sampling strategy** to allocate the MLP's capacity towards space with visible scene content
- A **positional encoding** to map each input 5D coordinate into a higher dimensional space, which enables us to successfully optimize neural radiance fields to represent high-frequency scene content.

Radiance Fields



NERF



1. Ray composition
   r(t) = o + td

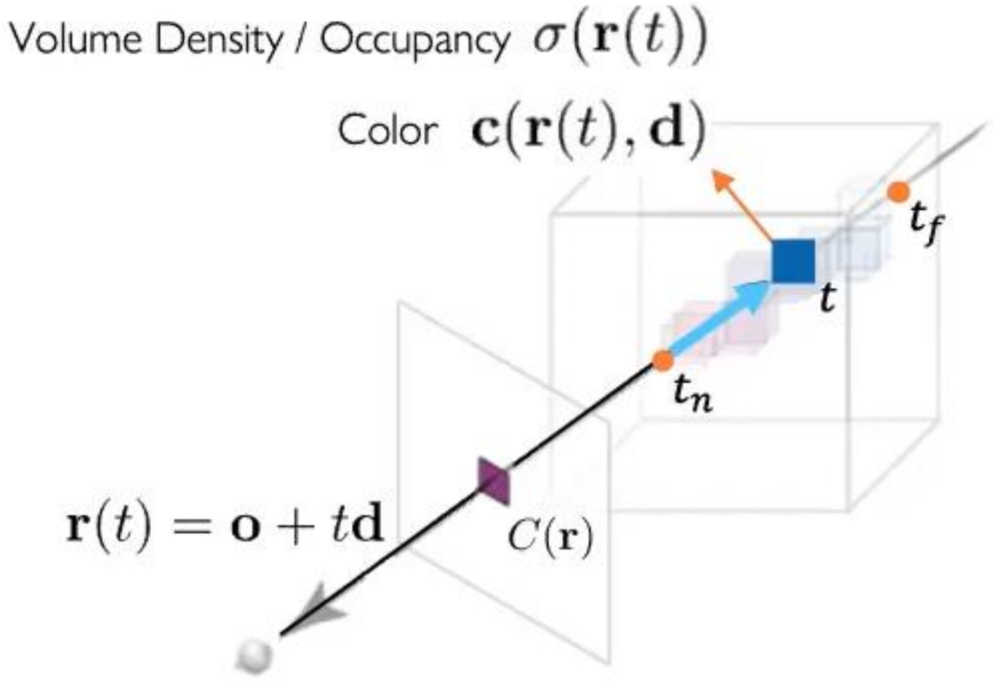
2. Sample 5D points from the ray
   (Coarse, fine Model)

3. Query into MLP

4. Volume Rendering

Volume Rendering

- The ray is **weighted sum of each 3D point** using color, density prosperities



Accumulated transmittance

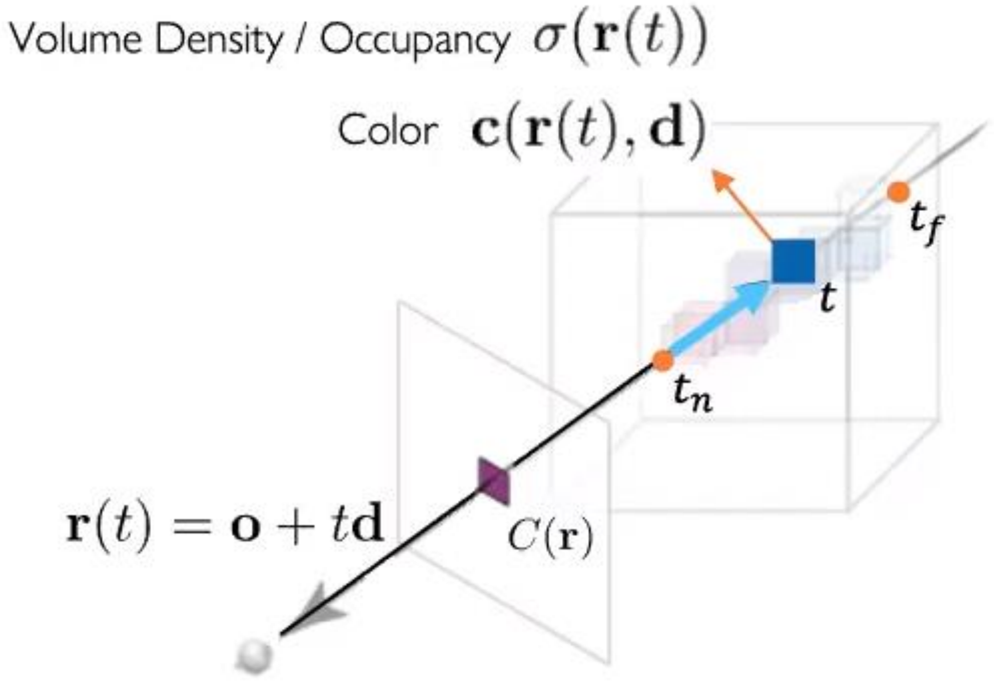
$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

Color at a 3D point lying on the ray

Volume density or Occupancy at a 3D point lying on the ray

Volume Rendering

- The ray is weighted **sum of each 3D point** using color, density prosperities

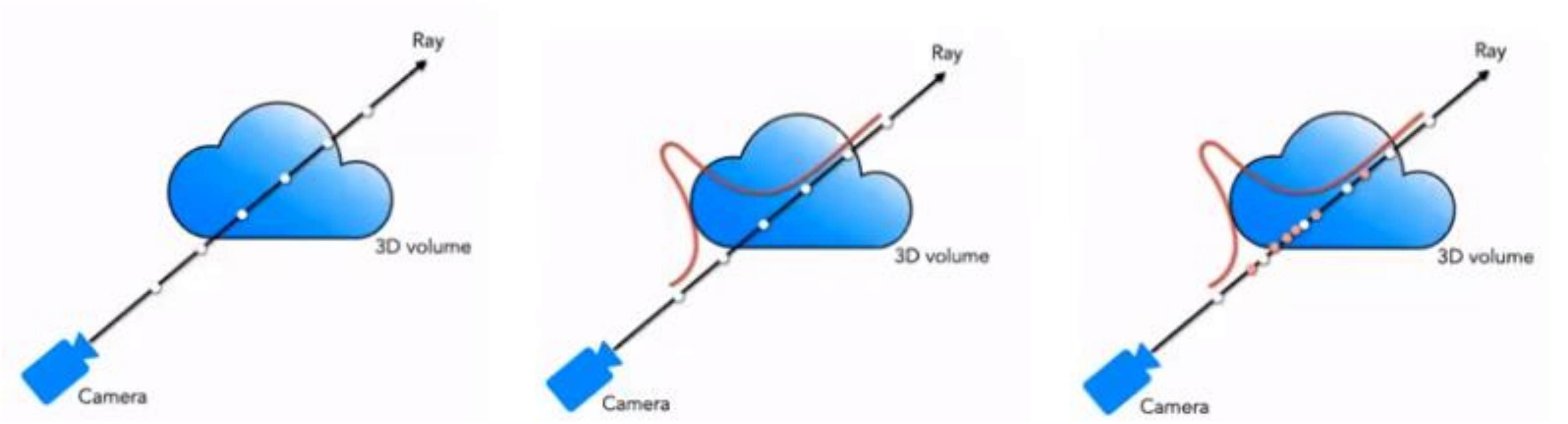


$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{c}_i \,, \text{ where } T_i = \exp\left(- \sum_{j=1}^{i-1} \sigma_j \delta_j \right) ,$$

Hierarchical volume sampling

- Problem
  - Rendering strategy of densely evaluating the neural radiance field network at N query points along each camera ray is **inefficient:** free space and occluded regions that do not contribute to the rendered image are still sampled repeatedly.

Hierarchical volume sampling



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

Sample points along the ray in 2 steps. (Coarse to Fine manner)

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

1) Uniformly divide a ray into N_c bins, and sample one point per each bin, total N_c points. (Stratified sampling)

👉 Then feed forward those points to MLP and get contribution weights. (Coarse Network)

2) Sample N_f points according to the probability distribution. (*Inverse transform sampling*)

👉 Finally feed forward N_c+N_f sampled points to MLP (Fine Network) and get colors and volume density of that points.

Positional encoding

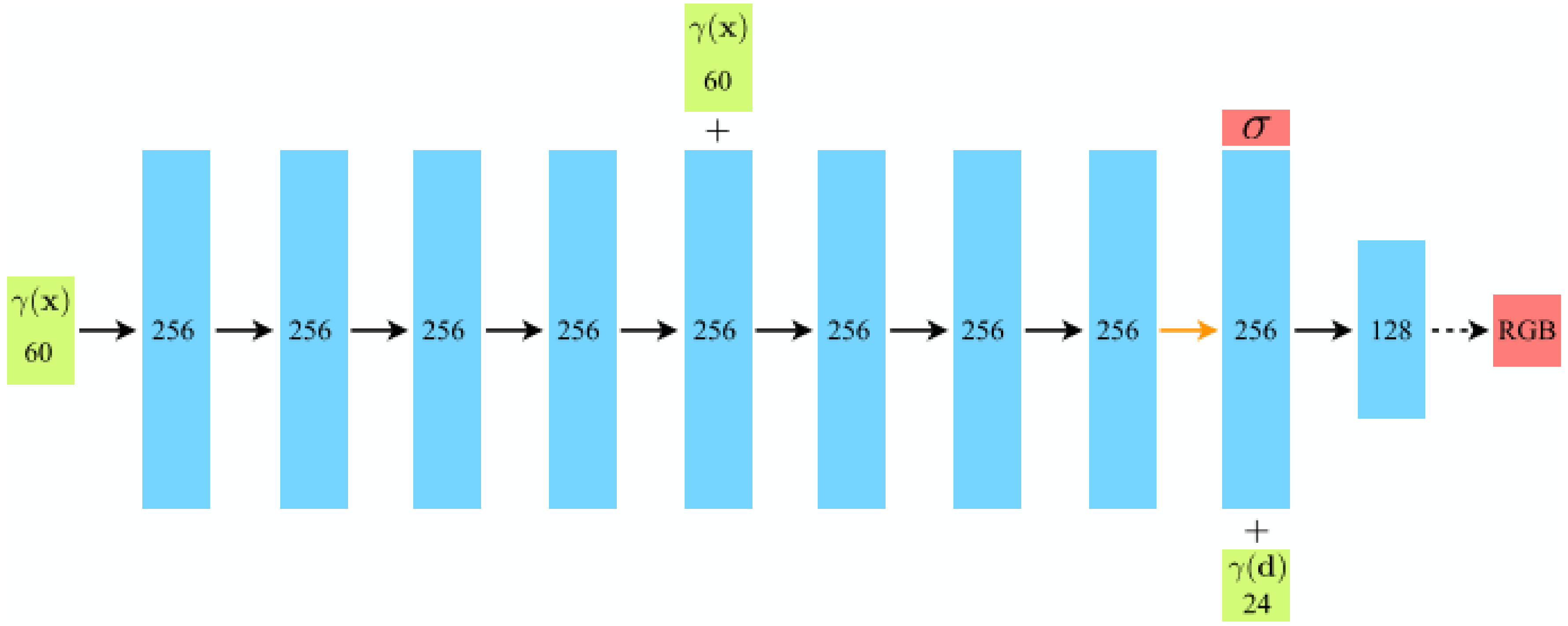
$$\gamma(p) = \left(\sin\left(2^0 \pi p\right), \cos\left(2^0 \pi p\right), \cdots, \sin\left(2^{L-1} \pi p\right), \cos\left(2^{L-1} \pi p\right) \right).$$

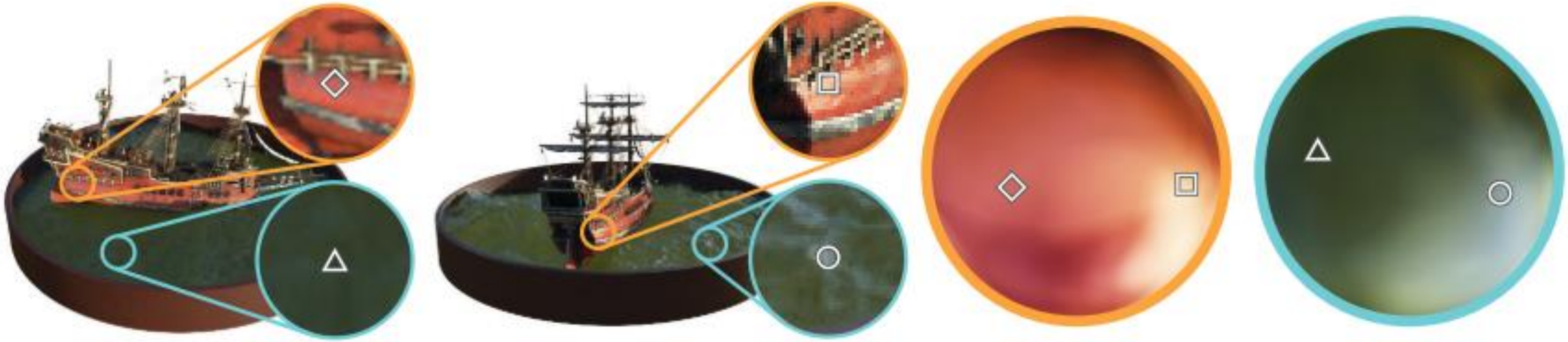
- γ: $R^L \rightarrow R^{2L}$
- Problem
  - Having the network directly operate on xyzθφ input coordinates results in renderings that perform poorly at representing high-frequency variation in color and geometry.
- They additionally show that mapping the inputs to a higher dimensional space using high frequency functions before passing them to the network enables better fitting of data that contains high frequency variation.

Rendering Loss

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

architecture





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

Experiment setting

- batch size of 4096 rays, each sampled at Nc = 64 coordinates in the coarse volume and Nf = 128 additional coordinates in the fine volume.

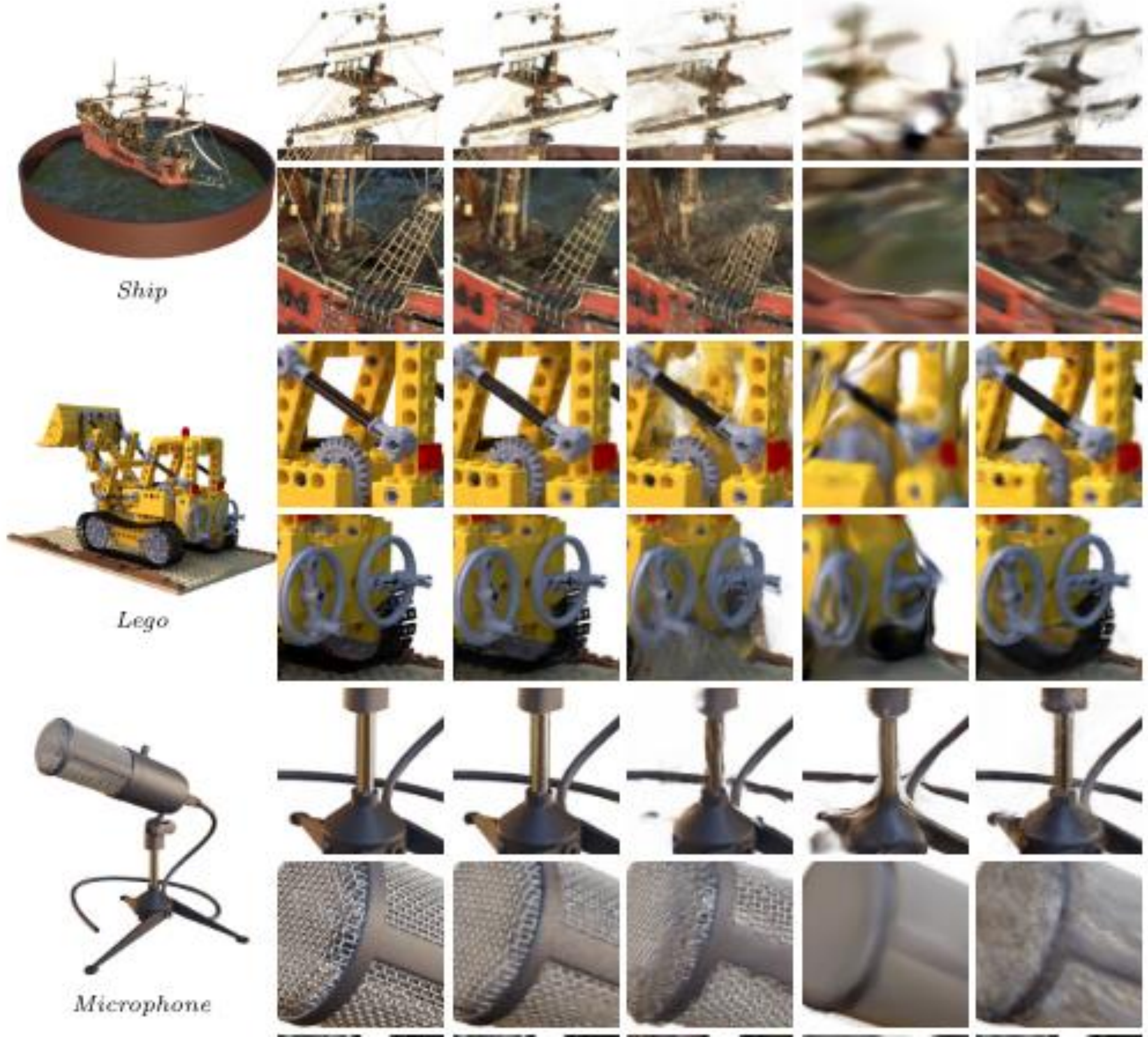
- Adam optimizer

- The optimization for a single scene typically take around 100–300k iterations to converge on a single NVIDIA V100 GPU (about 1–2 days).

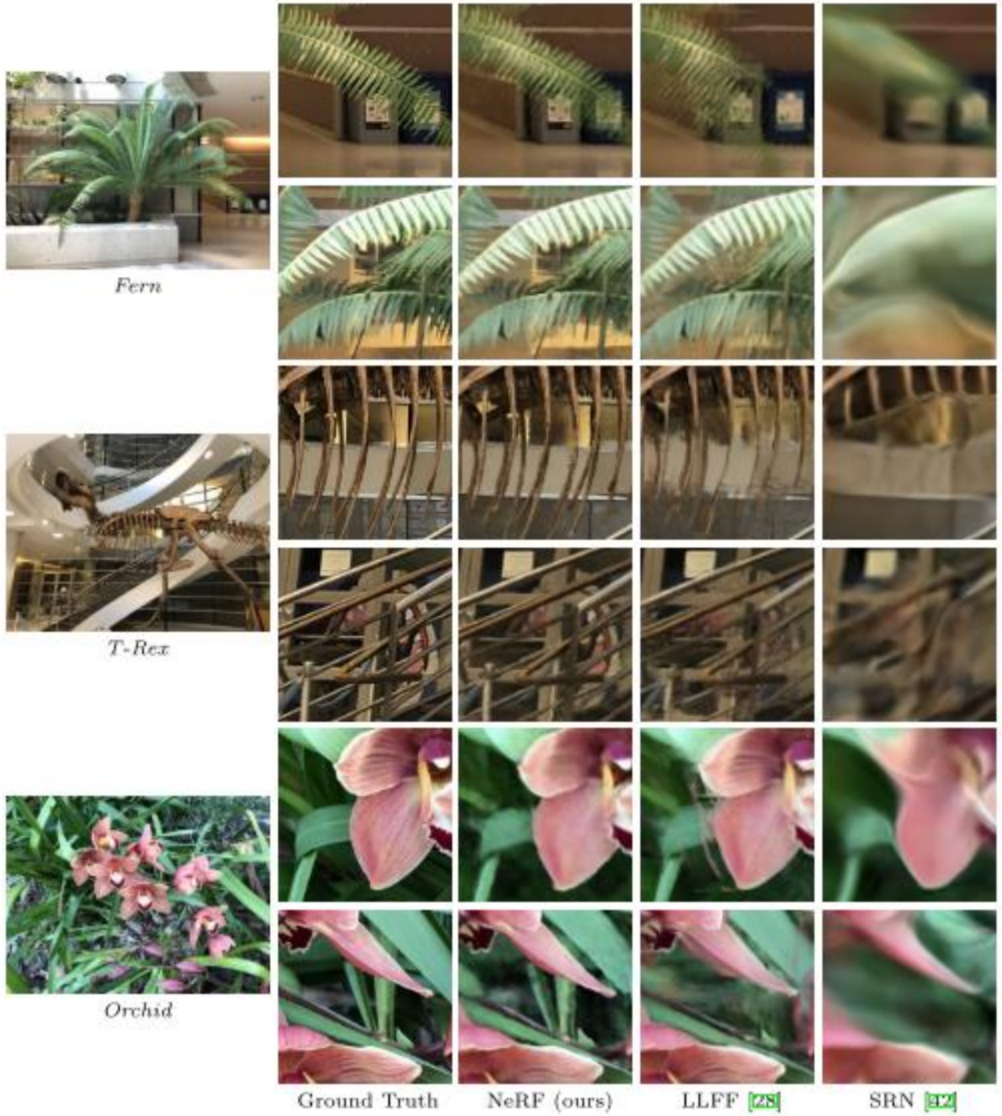
Results

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

Results



Ship
Lego
Microphone



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

Tradeoffs

- The biggest practical tradeoffs between these methods are time versus space.

- LLFF can process a small input dataset in under 10 minutes. However, LLFF produces a large 3D voxel grid for every input image, resulting in enormous storage requirements (over 15GB for one “Realistic Synthetic” scene).

- NERF requires only 5 MB for the network weights