NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Ben Mildenhall   Pratul P. Srinivasan   Matthew Tancik
Jonathan T. Barron   Ravi Ramamoorthi   Ren Ng

UC Berkeley
Google Research
UC San Diego

- Static scene represented as a continuous 5D function:

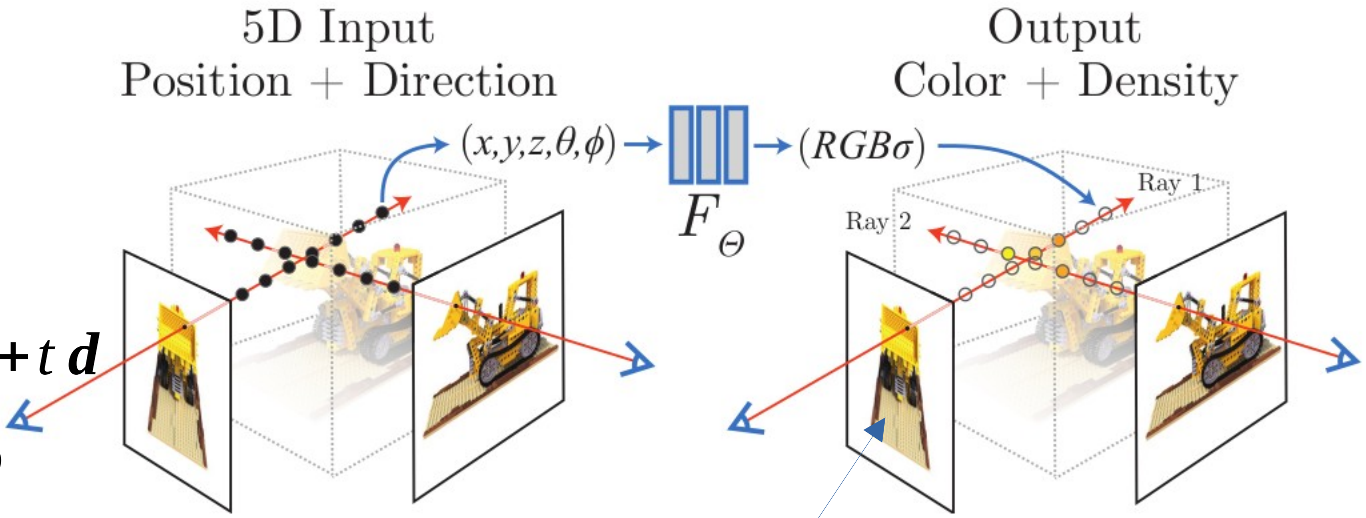
$$F_\Theta : \left((x, y, z) \stackrel{\text{def}}{=} \boldsymbol{x}, (\theta, \phi) \stackrel{\text{def}}{=} \boldsymbol{d} \right) \rightarrow \left(\boldsymbol{c}(\boldsymbol{x}, \theta, \phi), \boldsymbol{\sigma}(\boldsymbol{x}) \right)$$

  i.e., a *neural radiance field* (NeRF),
- From which an image from any viewpoint can be generated, by
  - marching camera rays $(\boldsymbol{r} = \boldsymbol{o} + t\,\boldsymbol{d})$ through the image plane to the scene to sample a set of 3D points,
  - Extracting $(\boldsymbol{c}, \boldsymbol{\sigma})$, then using classical volume rendering techniques to accumulate into a 2D image:

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

Integrate color × density × transmittance from near field to far field;
transmittance goes to 0 after some region (i.e., a blocking object) where density ↑ , meaning integrated color × density × transmittance → after that region of blockage.

Introduction



$\boldsymbol{r} = \boldsymbol{o} + t\,\boldsymbol{d}$

$\boldsymbol{o}$

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

- Ray originating from camera generated for each pixel for full 2D render

[1] Figure 2 from *Ben Mildenhall, et al. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis"*

Related Work: Neural 3D Shape Repr.

- Implicit repr. Of continuous 3D shapes via:
  - Signed Distance Functions (SPF) [15, 32]:
    output signed (+: outside, -: inside) distance given a 3D point, or
  - Occupancy Fields [11, 27].

- Despite the potential to represent complicated & highres geometry, so far been limited to simple shapes with low geometric complexity,
  resulting in oversmoothed renderings

Related Work: View Synth & Image-Based Rendering

- Given a dense sampling of views, novel views can be reconstructed by simple light filed *sampling interpolation techniques* [21, 5, 7]

- With sparser view samplings, novels views are synthesized by predicting **traditional geometry** and **appearance representations** from observed images:
  - Gradient-based mesh optimization based on image representation often difficult due to local minima or poor conditioning of loss
  - Requires template mesh as initialization before optimization (typically unavailable for real-world scenes)

Related Work: View Synth & Image-Based Rendering

- Volumetric Representations: Able to realistically represent complex shapes and materials & well-suited for gradient-based optimizations & produces less visual artifacts than mesh-based methods.
  - Color **voxel** grids [19, 40, 45]
  - Sampled volumetric representation from a set of input images (point cloud?) → alpha compositing (combining multiple images?) or learned compositing along rays

Highres imagery quadruples time and space complexity because of discrete sampling of 3D space.

Model Formulation: Differentiable Rendering

- The model: $F_{\Theta}:(\boldsymbol{x},\boldsymbol{d})\rightarrow(\boldsymbol{c},\boldsymbol{\sigma})$
- We want to generate rays for each pixel on the image plane to compute its color using:

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

which can be numerically estimated using quadrature:

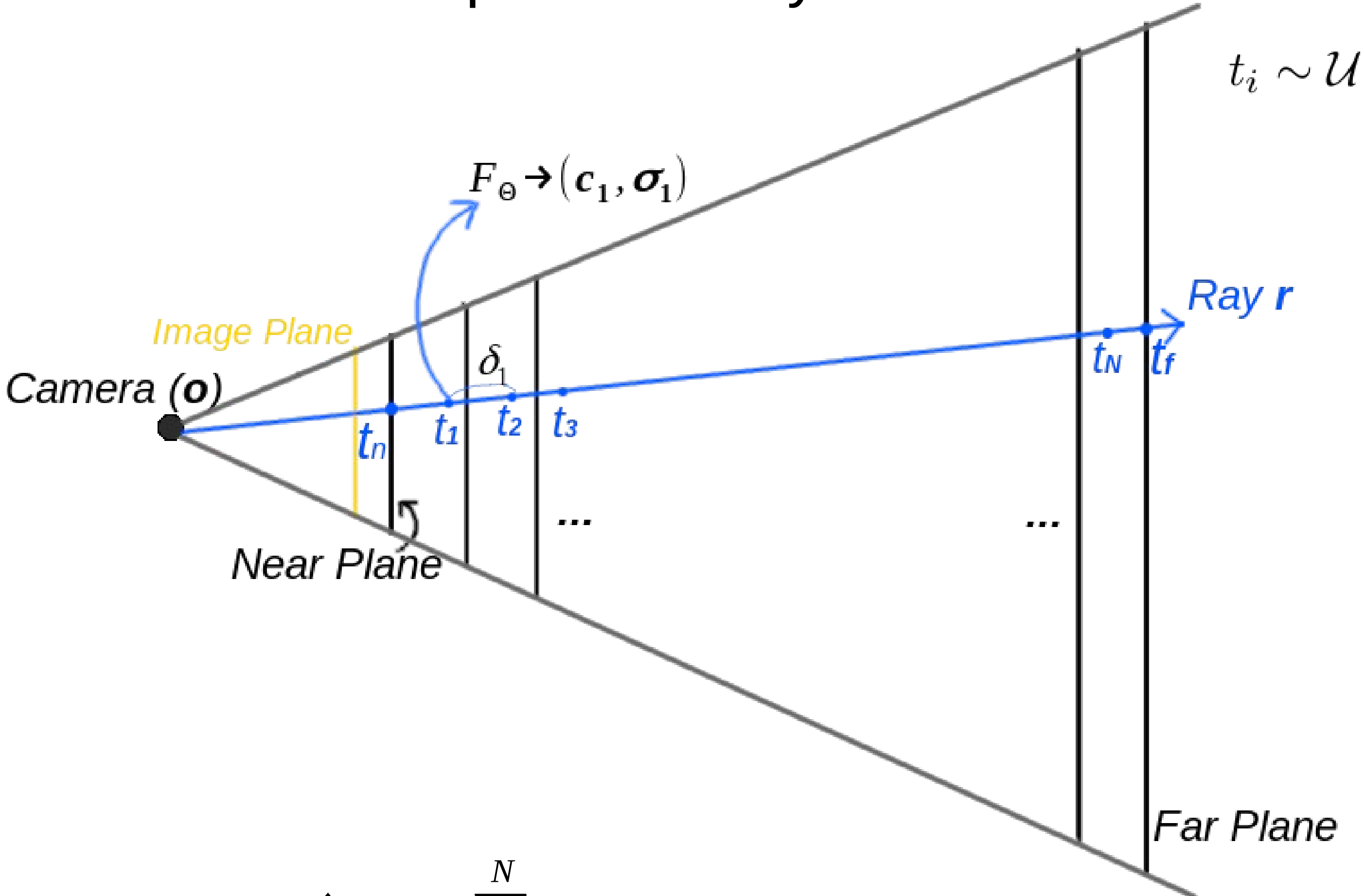
$$\hat{C}(\boldsymbol{r})=\sum_{i=1}^{N} T_i(1-\exp(-\sigma_i\delta_i))\boldsymbol{c_i}\, where\, T_i=\exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right),\ \delta_i=t_{i+1}-t_i$$

→ Trivially Differentiable

Model Formulation: Differentiable Rendering

- Stratified Sampling: Uniformly partition $[t_n, t_f]$ into N evenly spaced bins and draw one sample uniformly at random from each bin, i.e.,:

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



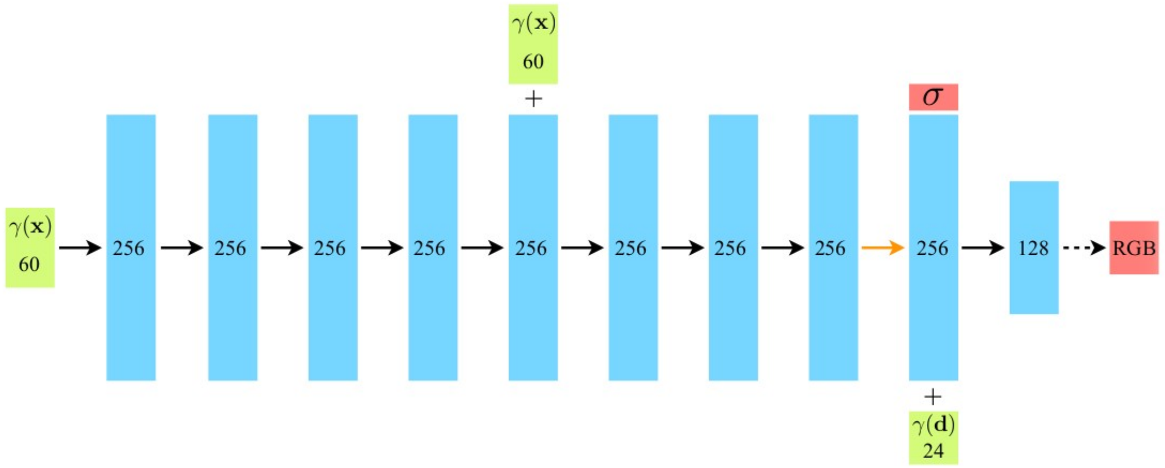
$$\hat{C}(\boldsymbol{r}) = \sum_{i=1}^{N} T_i (1 - \exp(-\sigma_i \delta_i)) \boldsymbol{c_i} \; where \; T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right), \delta_i = t_{i+1} - t_i$$

Model Summary & Training

- Loss: Total squared error btw. The rendered and true pixel colors:

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

- Batch of 4096 rays, each sampled at $N_c = 64$, $N_f = 128$,
- Adam optimizer, learning rate: $5\times10^{-4} \rightarrow 5\times10^{-5}$ exponential decay
- Optimization for a single scene takes around 100-300k iterations to converge on a single NVIDIA V100 GPU (1-2 days).



Model Optimization: Positional Encoding

- Despite the fact that NN are universal function approximators [14], directly inputting $(\boldsymbol{x}, \boldsymbol{d})$ results in poor estimation of high-frequency variation in color and geometry.
- Rahaman et al. [35] shows deep networks are biased towards learning lower frequency functions, and **mapping inputs to a higher dimensional space using high frequency functions** before passing to network results in better fitting of data that contains high frequency variation.
- So we map $(\boldsymbol{x}, \boldsymbol{d})$ to $(\gamma_{L=10}(\boldsymbol{x}), \gamma_{L=4}(\boldsymbol{d}))$ (x is normalized to lie in [-1, 1], and ɣ is applied separately to each coordinate value) where

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

$$e^{j 2^{L-1} \pi p}\Big|_{L=4,\,10} \xrightarrow{\mathscr{F}} 2\pi\delta(\omega - 687), 2\pi\delta(\omega - 3 \times 10^9)$$

Model Optimization: Hierarchical Volume Sampling

- Densely sampling N points in a ray is inefficient if part of the path is occluded.

- So we optimize two networks: *coarse*—$\hat{C}_c(\boldsymbol{r})$— and *fine*—$\hat{C}_f(\boldsymbol{r})$—, where the fine network makes sampling decisions based on the output of the coarse network.

Results

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

Results

Method	Diffuse Synthetic 360° [41] PSNR↑	SSIM↑	LPIPS↓	Realistic Synthetic 360° PSNR↑	SSIM↑	LPIPS↓	Real Forward-Facing [28] 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

- Diffuse Synthetic: Lambertian materials only
- Metrics:
  - PSNR (Peak Signal-to-Noise Ratio): $10\log MAX^2/MSE$
  - SSIM (Structural Similarity Index Measure): evaluation based on brightness, contrast, and structure; $SSIM\in[0,1]$
  - LPIPS [50]: perceptual loss; DL-based

Discussions

- A full training of images annotated with camera parameters is required to produce images from novel angles.
- Poor performance when camera strays from center of object?
- Poor mesh restoration?