NeRF

Representing Scenes as Neural Radiance Fields for View Synthesis

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

Presenter:Chengzhe Li

What is NeRF? Why is it good?

ECCV 2020 Oral - Best Paper Honorable Mention

Impactful: A lot of follow up researches like FastNeRF, NeX, PixelNeRF etc ..

Novel, interesting paper, generating scenes or videos from couples of 2D pictures.

A novel way to addressing a long-standing problem of view synthesis

Prominently Outperform previous approaches

Some Examples

How does it work?

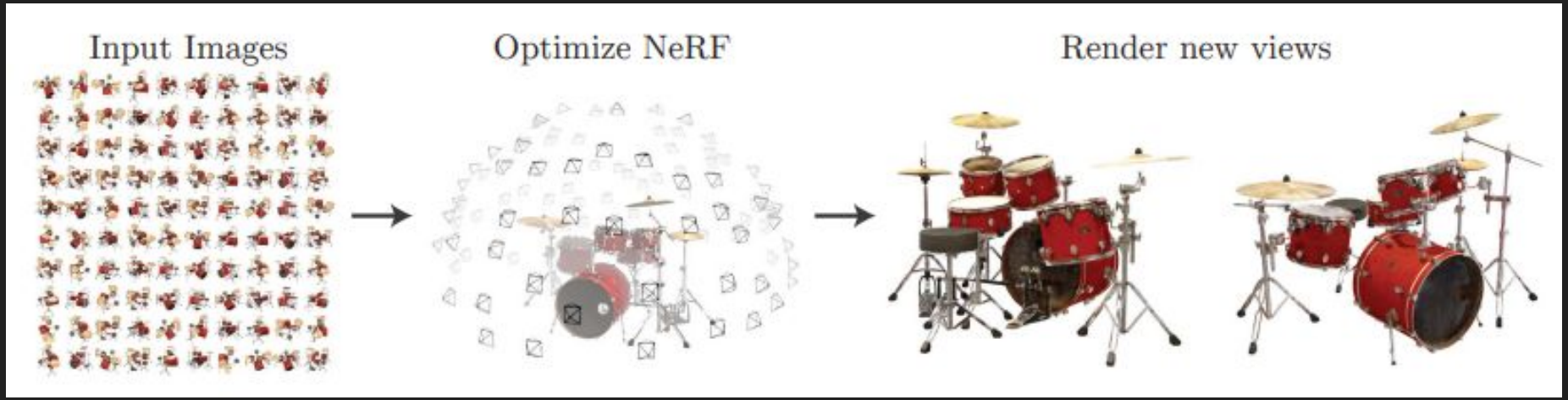
Optimizing an underlying continuous volumetric scene function using a sparse set of input views

Representing a scene using a 5D fully-connected deep network (non-conventional).

5D: spatial location (x,y,z) and viewing direction (θ, φ), output: volumetric density and emitted radiance

Rendering the scenes by neural rendering for volumetric

Pipeline



Contributions

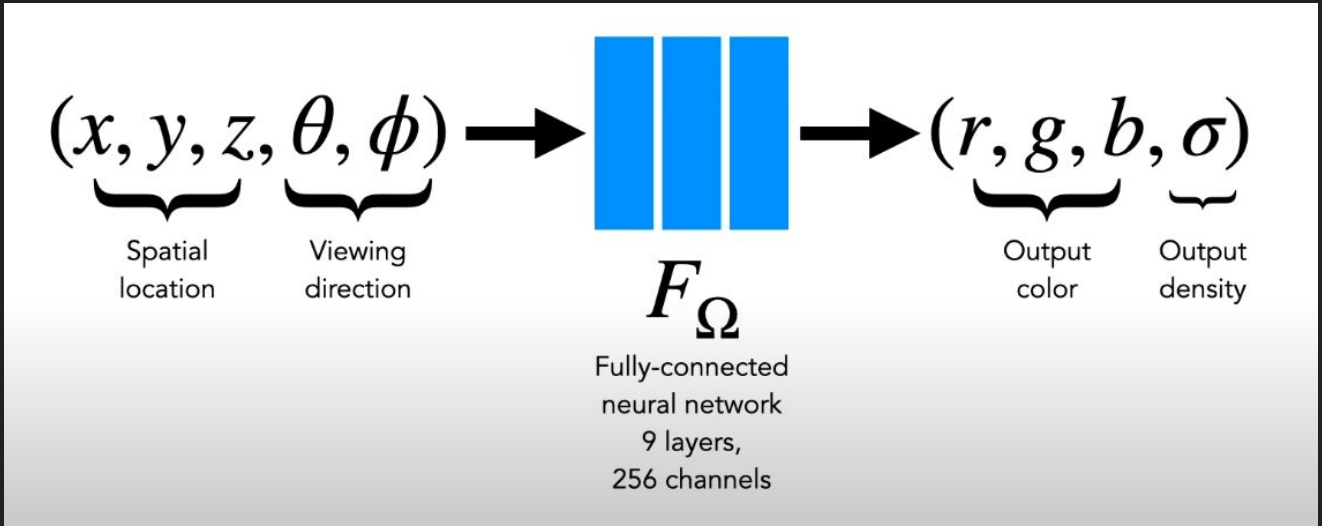
An approach for representing continuous scenes with complex geometry and materials as 5D neural radiance fields, parameterized as basic MLP networks.

A differentiable rendering procedure based on classical volume rendering techniques

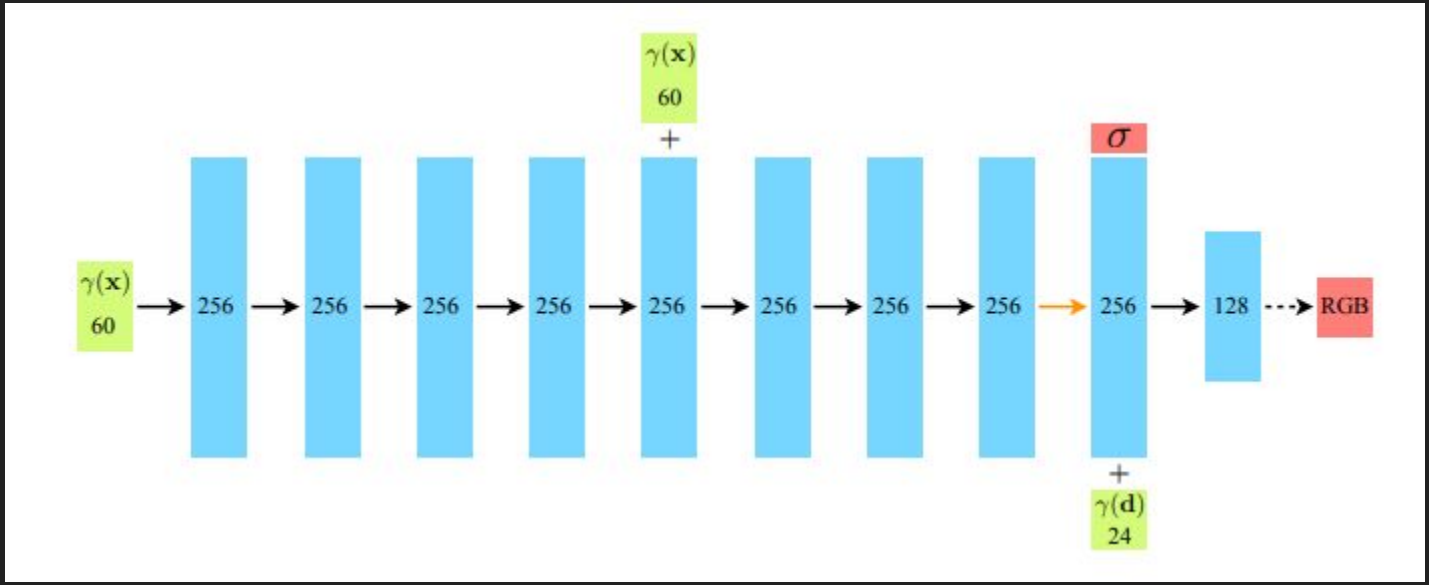
A positional encoding to map each input 5D coordinate into a higher dimensional space

Scene representation

5D continuous neural network as a volumetric scene representation (x,y,z,θ, φ)



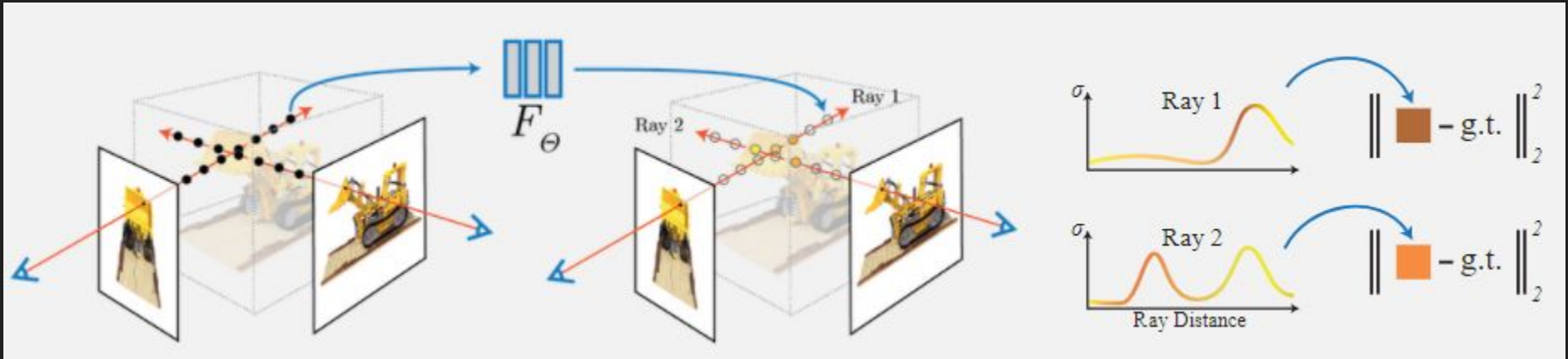
Network Details



Add the color of rays together to get the color of each pixel

uses quadrature estimate

Allowing colors to be different when view changes for view-depend effects



Rendering

Use volume rendering model to synthesize new views

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right). \quad (1)$$

Differentiable

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

Optimizing

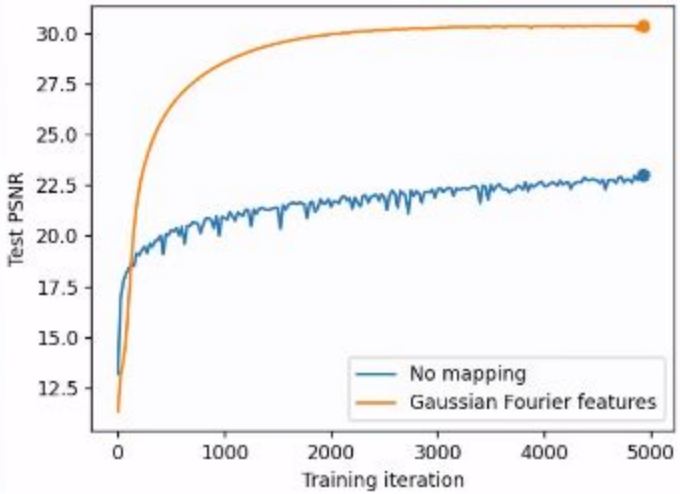
-Positional Encoding

Helps with high frequency details(Gaussian Fourier features, another paper)

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

-Hierarchical volume sampling

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


30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
Test PSNR
0
1000
2000
3000
4000
5000
Training iteration
No mapping
Gaussian Fourier features

Contrast

NeRF (Naive)

NeRF (with positional encoding)

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

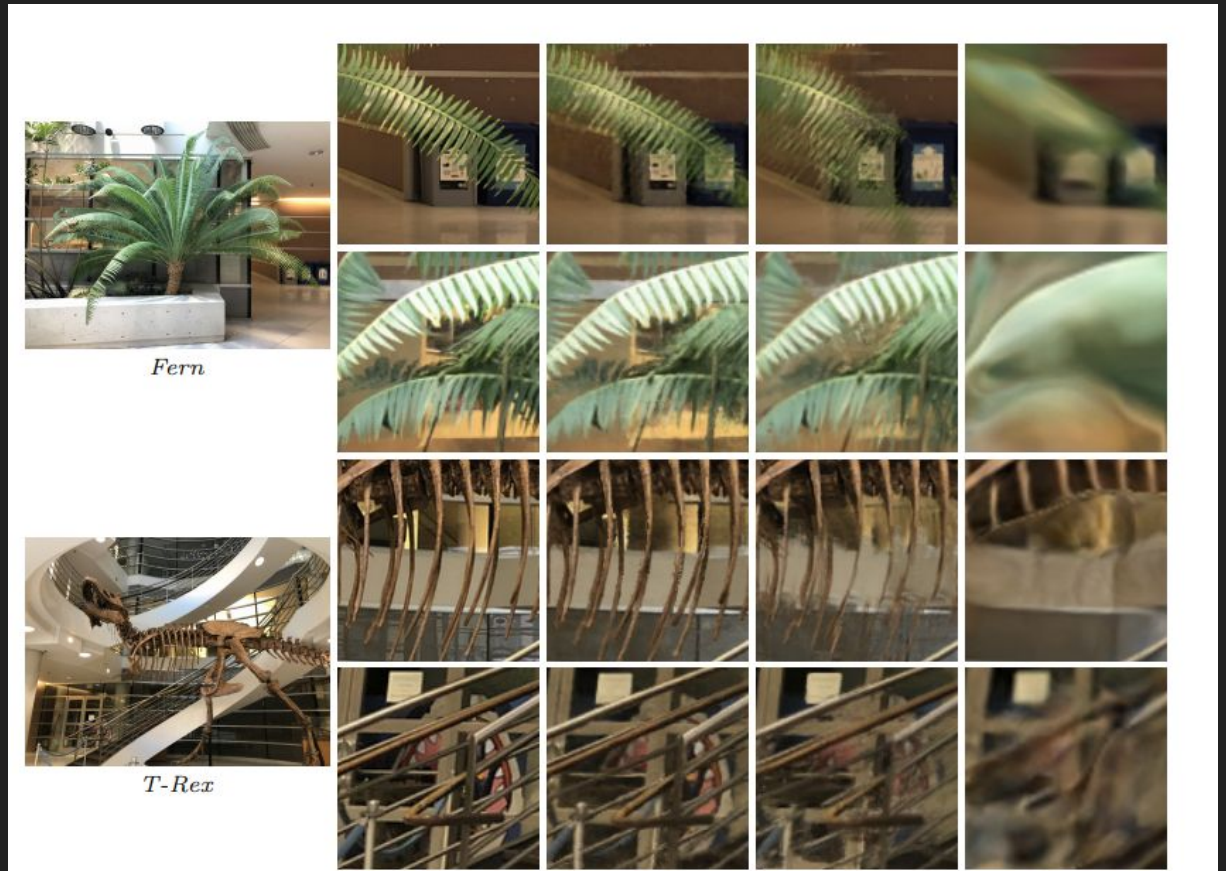
Examples: Synthesis

Microphone

Materials

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

Example: Real world



Accurate depth map

Conclusion

Outperform LLFM for almost all metrics while using only input images as training set

Save a ton of spaces compares to voxel grid (3000x, 15GB vs. 5MB)

Time-space tradeoff

My thoughts