
NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Ben Mildenhall
UC Berkeley /
Google Research

Pratul P. Srinivasan
UC Berkeley /
Google Research

Matthew Tancik
UC Berkeley

Jonathan T. Barron
Google Research

Ravi Ramamoorthi
UC San Diego

Ren Ng
UC Berkeley

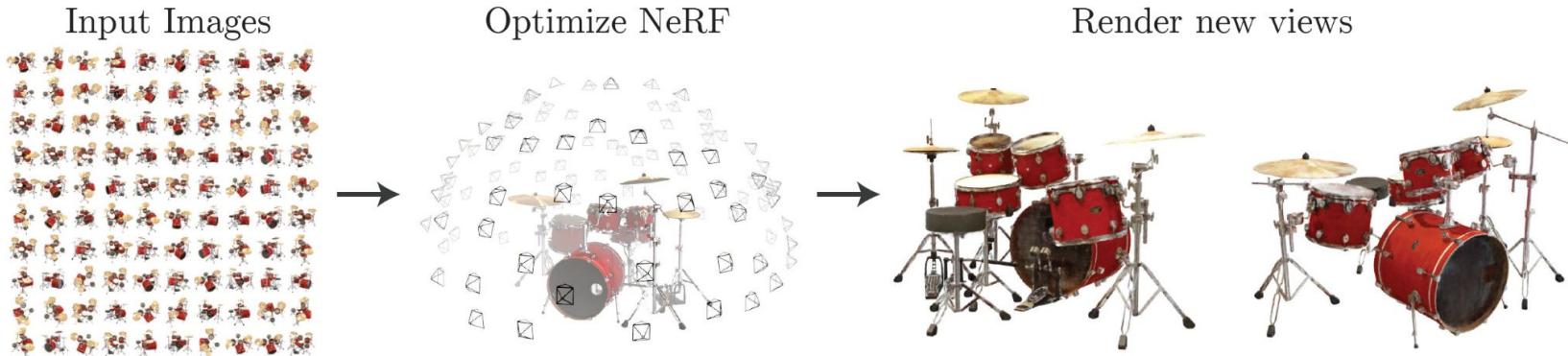
By : Sheung Hang Sean Kan & Vatsal Thakkar

The University of Georgia



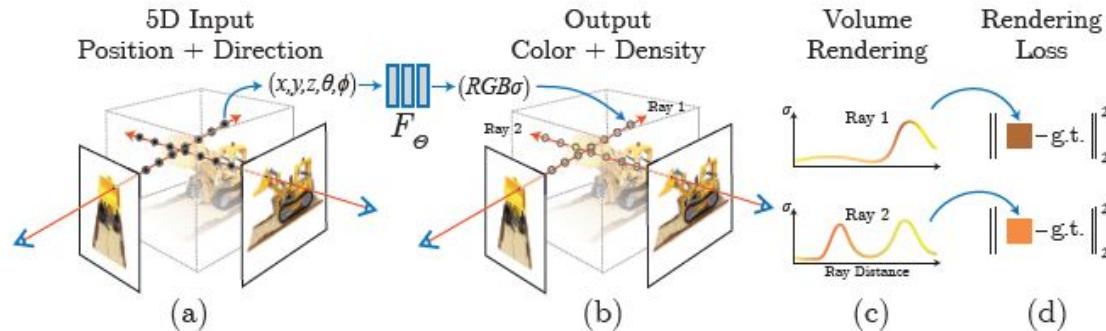
Introduction

The paper presents a state of the art results for synthesizing novel views of complex scenes by optimizing an underlying continuous volumetrics scene function using sparse set of input views





The input of the model would be static scene as a continuous 5D function that outputs the radiance emitted in each direction (θ, ϕ) at each point (x, y, z) in space, and a density at each point.



This method optimized a deep fully connected neural network without any convolutional layers to represent this function by regressing from a single 5D coordinate (x, y, z, θ, ϕ) to a single volume density and view-dependent RGB colour.



To achieve the above setting:

1. March camera rays through the scene to generate a sampled set of 3D points
2. Use those points and their corresponding 2D viewing direction as input to the neural network to produce an output set of colours and density
3. Use classical volume rendering technique to accumulate those colours and densities into a 2D image

Involve gradient descent to optimize the model by minimizing the error between each observed image and the corresponding views rendered. (i.e. try to overfit the model)

Such method will encourage the network to predict a coherent model of the scene.

Motivation

To effectively optimize neural radiance fields to render photorealistic novel views of scenes with complicated geometry and appearance, and demonstrate results that outperform prior work on neural rendering and view synthesis.

Motivation

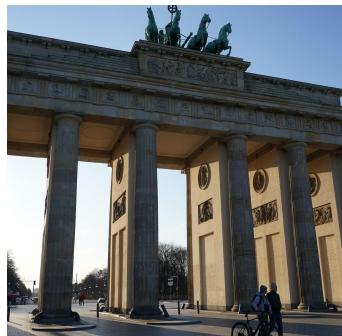


Sparsely Sampled images of scene



New Views of Same Scene NeRF (Output)

Some More Motivation



NeRF

Advantage of the method

- Can represent complex real-world geometry and appearance and are well suited for gradient-based optimization using projected image.
- Overcomes the prohibitive storage costs of discretized voxel grids when modeling complex scene at high-resolution.

Related Work

Recent direction in computer vision: Encoding object and scene in weights of an MLP -> map from 3D spatial location to representation of the object

Disadvantage: unable to capture complex geometry + low resolution

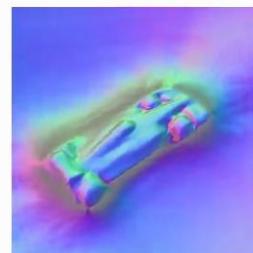
Neural 3D representation: Mapping xyz coordinates to signed distance function. Required ground truth 3D geometry obtained by ShapeNet.

View synthesis and image-based rendering: take a set of input RGB images and rendering into complex shapes and materials.

Scene Representation Networks (SRNs)

- Continuous scene as an opaque(non-transparent) surface,
- Defined by a MLP that maps each (x,y,z) coordinate to a feature vector
- Uses recurrent neural network to predict the scenes of the object other than representation scene.
- **Limitation:** discrete voxel grids do not scale well and lose fine detail at high resolutions
- **Limitation:** requires a bounded volume and knowledge of the background

<https://www.vincentsitzmann.com/srns/>



Local Light Field Fusion (LLFF)

- Designed for producing photorealistic novel views for well-sampled forward facing scene. (i.e. high-resolution with high amount of sample)
- Uses 3D convolutional network to predict RGB grid for each view, then render them into novel view
- Limited to simple shapes with low geometric complexity

<https://bmild.github.io/llff/>



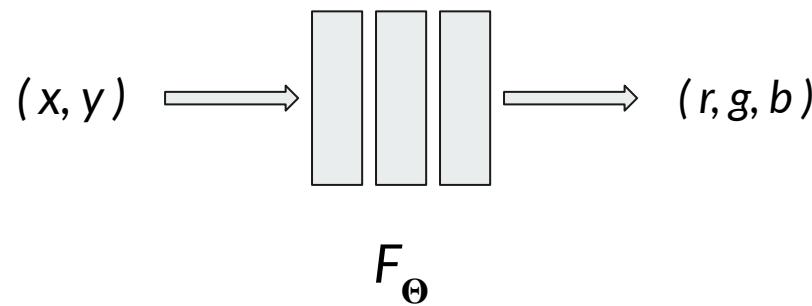
Neural Volumes

- Synthesizes novel views of object that lie entirely within a bounded volume in front of a distinct background.
- Optimises a deep 3D convolutional network to predict discretized RGB voxel grid with fixed samples in the bounded volume.
- Fast to train (<10 minutes) at the cost of large storage requirements (~GB for each scene)

Some footage of the results

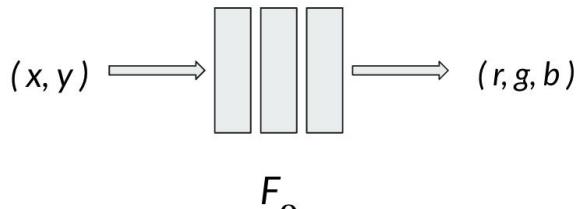


Memorizing 2D Image

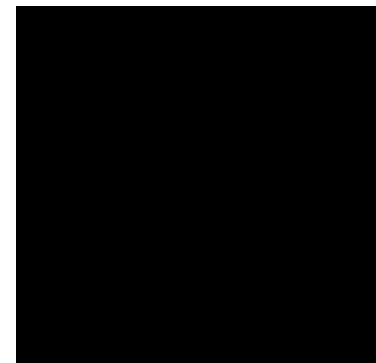
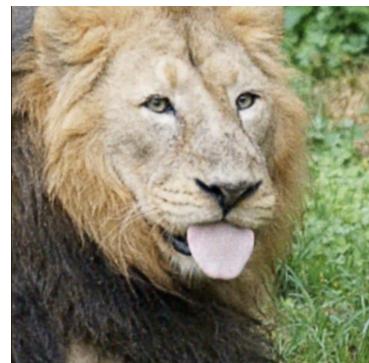


CPPN : Compositional pattern-producing networks

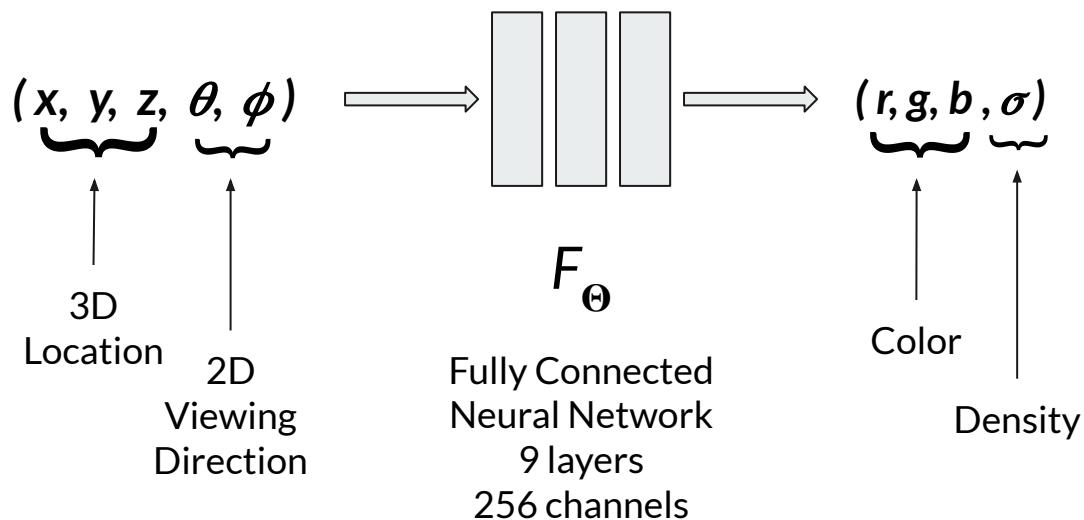
Memorizing 2D Image



Fully Connected
Neural Network



NeRF (Neural Radiance Fields)



Problem Setup

- What we have?

Given a dataset containing RGB images of a static scene, their corresponding camera poses, and intrinsic parameters.

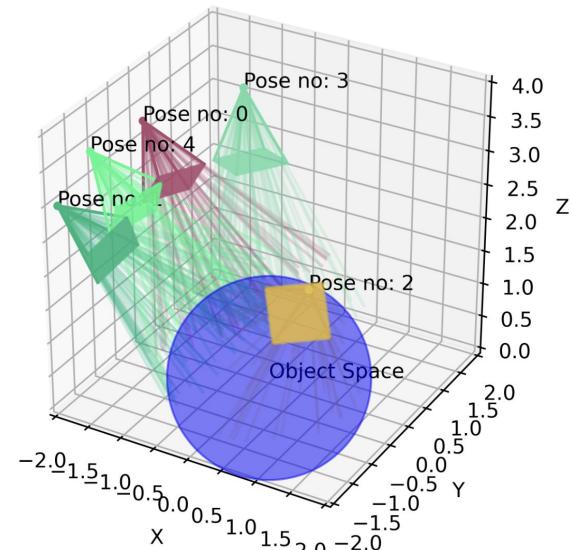
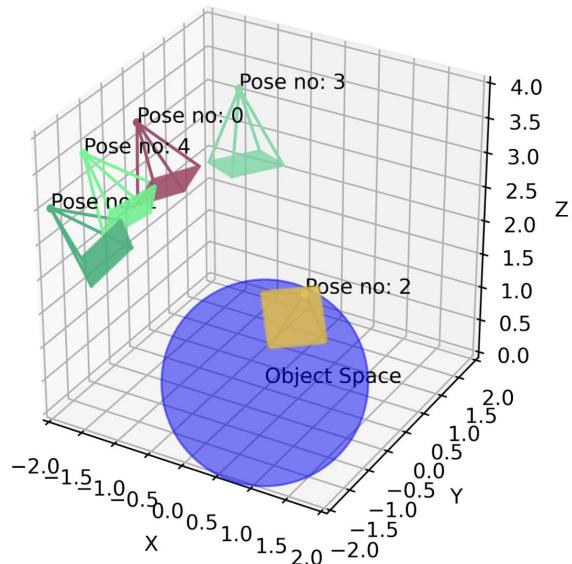
- Set of Camera Poses,

$$\{(x_c, y_c, z_c), (\theta_c, \phi_c)\}_n = \{X_c, d_c\}_n$$

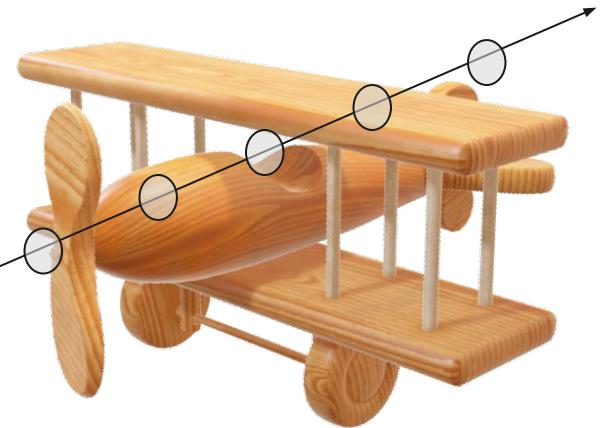
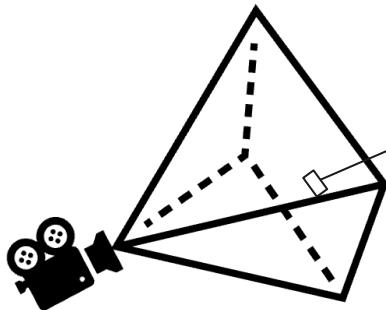


Problem Setup

- Imagine a plane in place of camera, this plane is where all the 3D information from the rays will be aggregated to render a 2D image (3D to 2D projection).
- Size of this plane is same size of input image



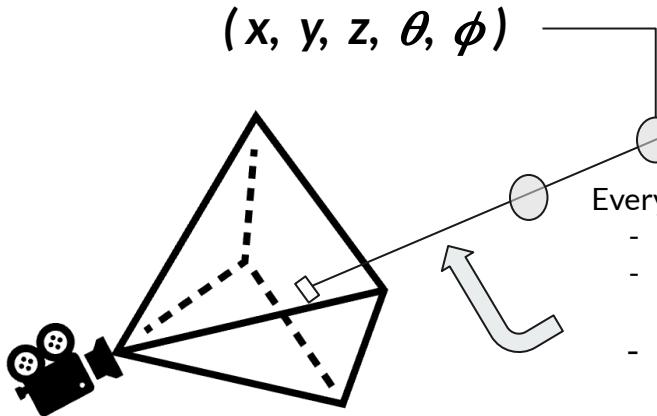
Volume Rendering with Radiance Field



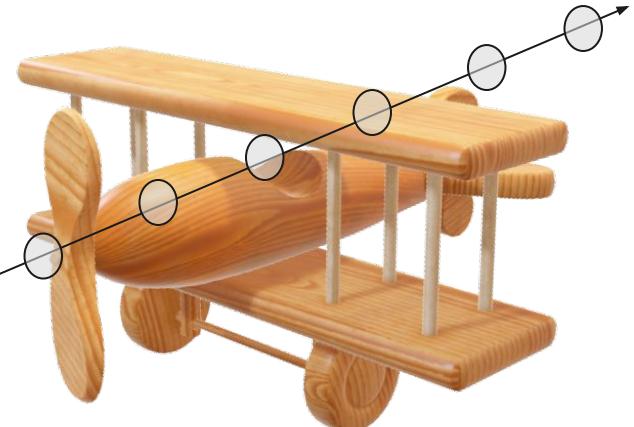
Ground Truth
Image taken by
this Camera



Volume Rendering with Radiance Field

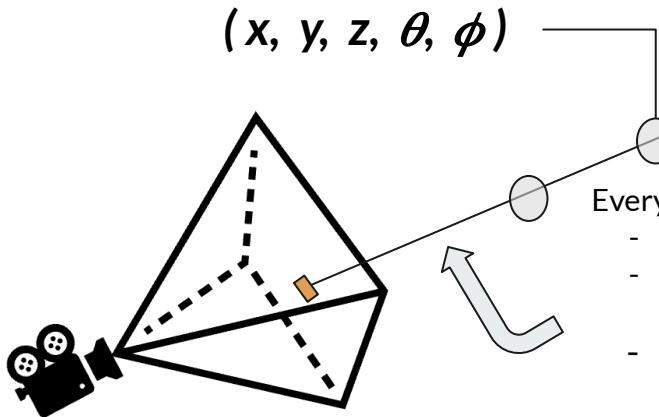


- Every ray is described by two vectors
- Vector that specifies the origin of ray (\mathbf{o}).
 - A normalized vector that specifies the direction of the ray (\mathbf{d}).
 - we make the t parameter larger (thus extending our rays) until a ray reaches some interesting location in the object space.



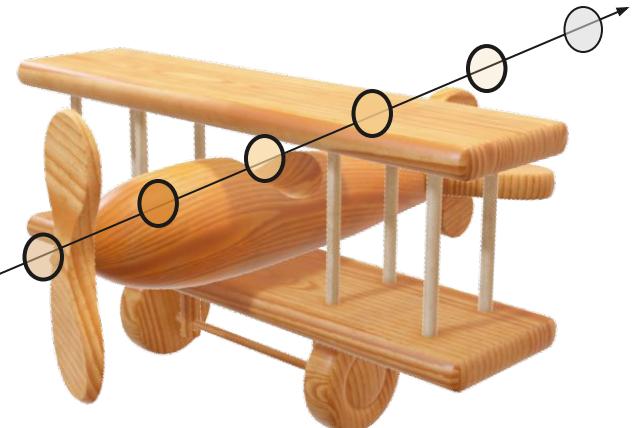
$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$$

Volume Rendering with Radiance Field

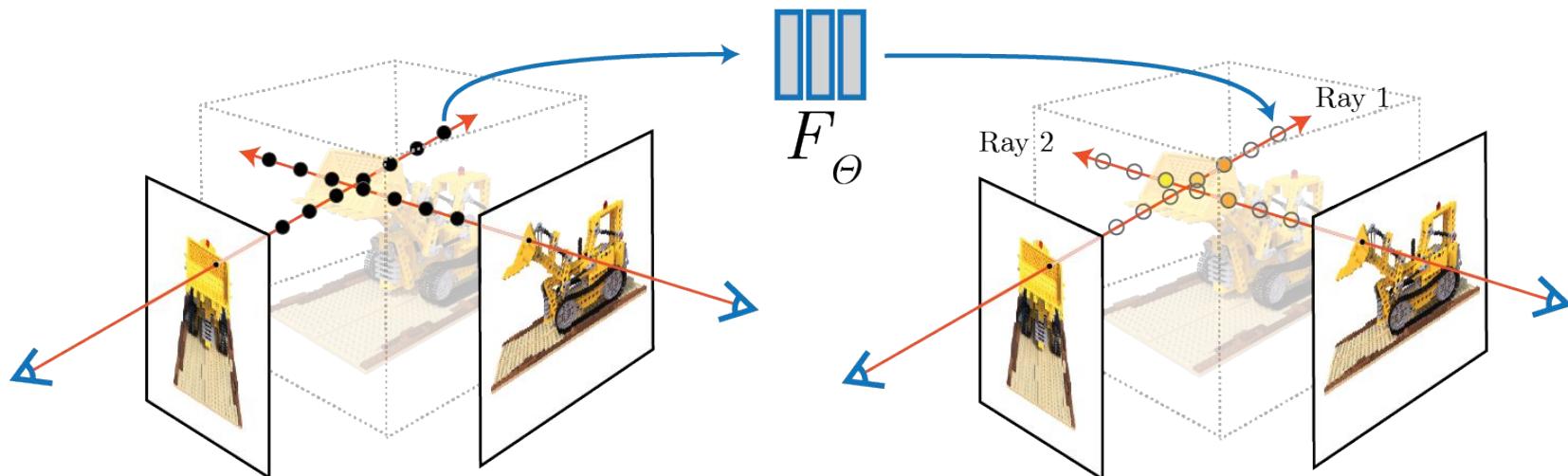


- Every ray is described by two vectors
- Vector that specifies the origin of ray (\mathbf{o}).
 - A normalized vector that specifies the direction of the ray (\mathbf{d}).
 - We make the t parameter larger (thus extending our rays) until a ray reaches some interesting location in the object space.

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

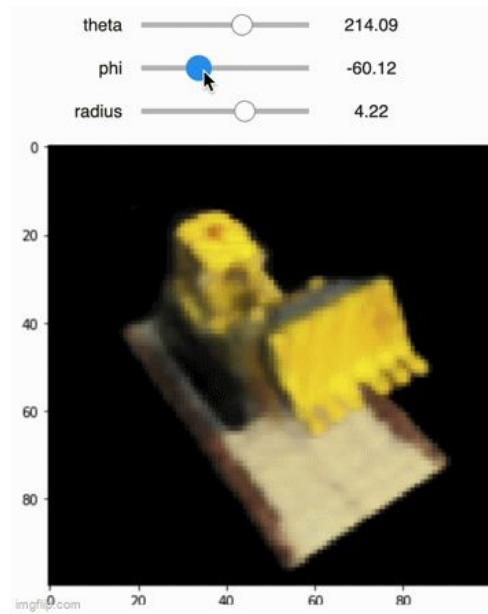
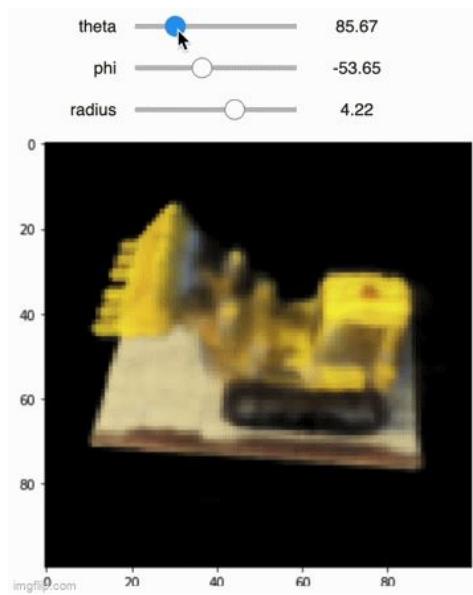


Volume Rendering with Radiance Field



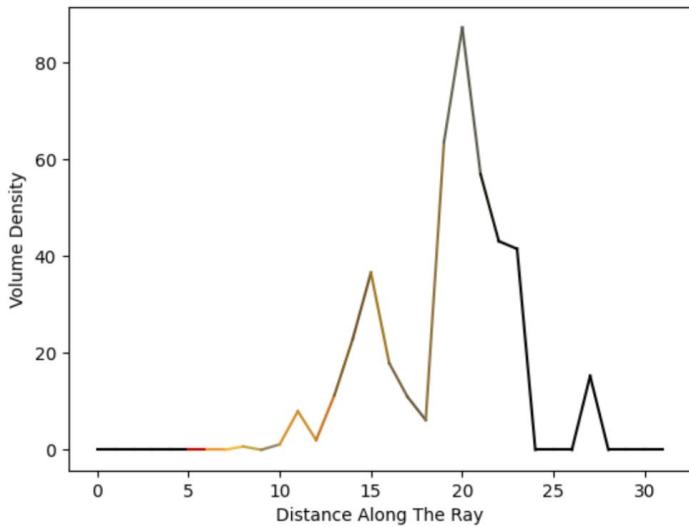
$$\min_{\theta} \sum_i \| \text{render}_i(F_\theta) - I_i \|^2$$

NeRF





Example of Density profile of a Ray



Volume Rendering with Radiance Field

- Rendering a view from this continuous neural radiance field requires estimating this integral $C(\mathbf{r})$

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

↑ ↑ ↑

How much light has been blocked up to point t ?

Color at $\mathbf{r}(t)$ from viewing angle \mathbf{d}

Density at that point

Volume Rendering with Radiance Field

- Numerically estimate this continuous integral using **quadrature**. Which makes it differentiable.
- To do that, partition $[t_n; t_f]$ into N evenly-spaced bins and then draw one sample uniformly at random from within each bin.

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

Volume Rendering with Radiance Field

- Estimate $C(r)$ with the quadrature rule

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

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

weights colors

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

↑

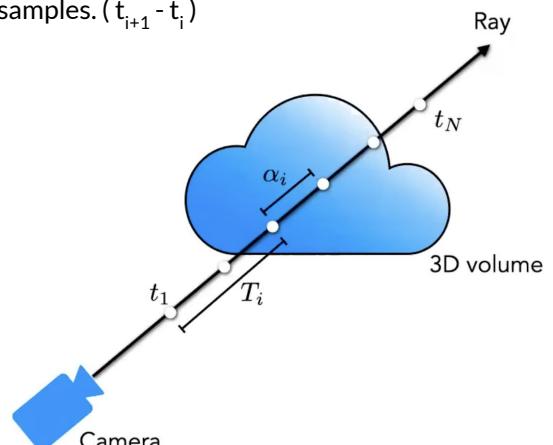
$$\alpha_i = 1 - e^{-\sigma_i \delta t_i}$$

↑

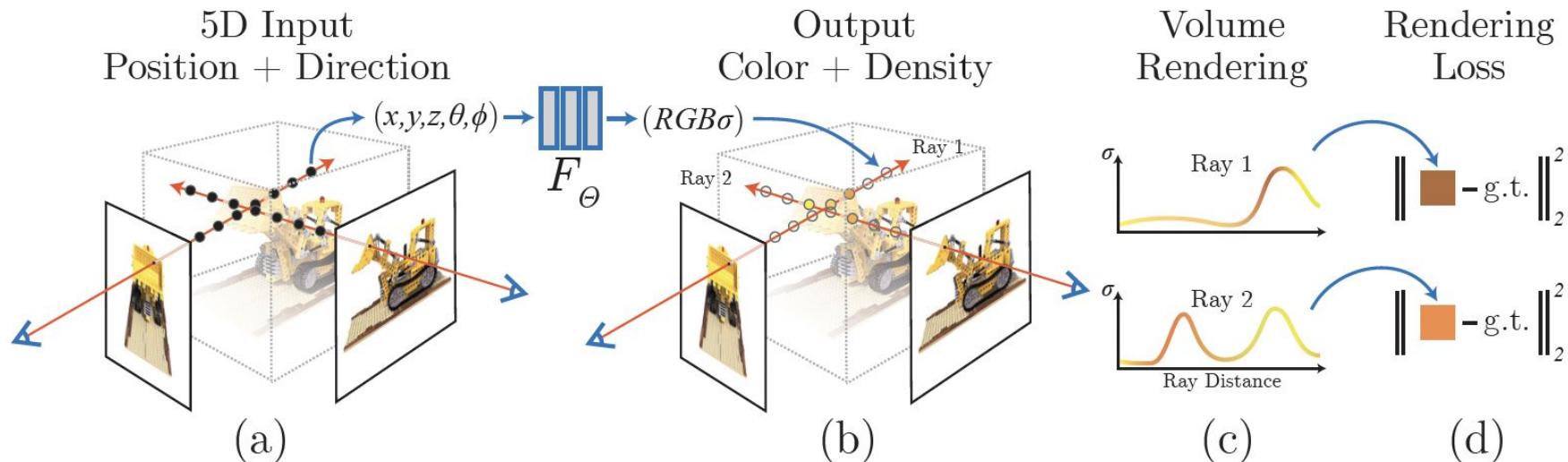
How much light has been blocked up to point t ?

How much light is contributed by ray segment i ?

Distance between adjacent samples. ($t_{i+1} - t_i$)



Overview of NeRF scene representation

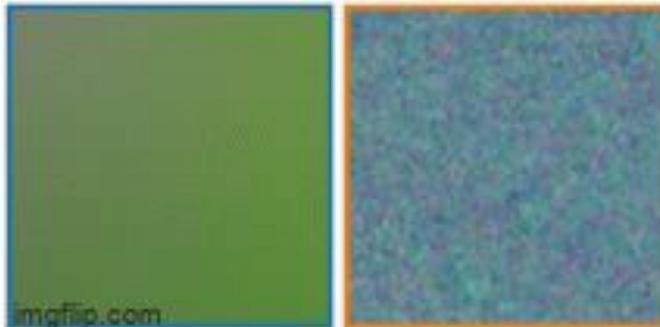


Optimizing a Neural Radiance Field

- Positional Encoding
- Hierarchical volume sampling

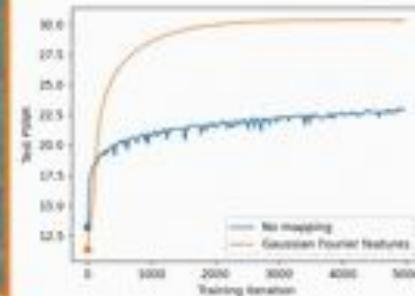
Positional Encoding

- Why Positional Encoding ?



Without Using
Fourier Feature

Using Fourier
Feature



Positional Encoding



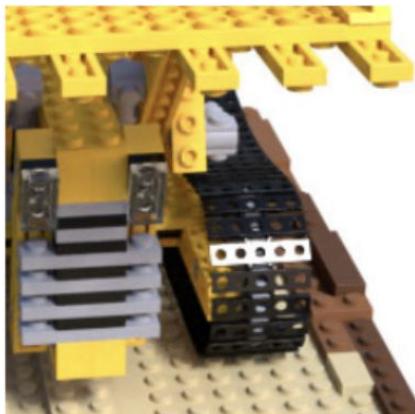
Without Positional Encoding
(Naive NeRF)



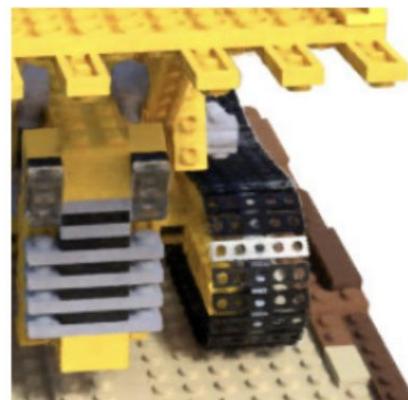
With Positional Encoding
(NeRF)

Positional Encoding

The authors claim 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



Ground Truth



Complete Model



No Positional Encoding

Positional Encoding

- The authors claim 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
- This function $\gamma(\cdot)$ is applied separately to each of the three coordinate values in \mathbf{X} and \mathbf{d} .

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

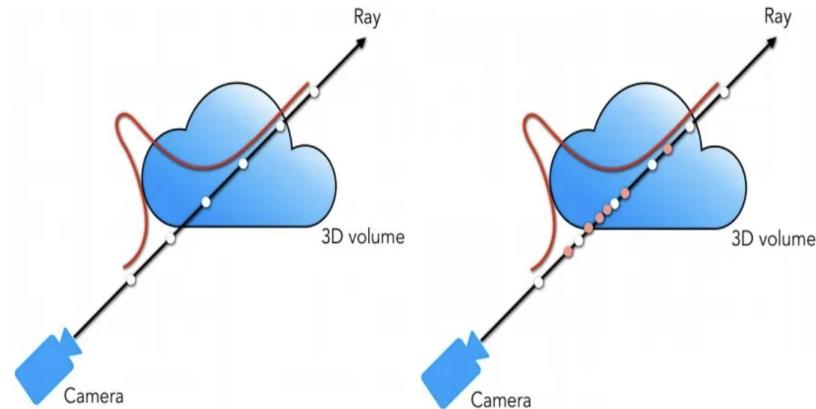
- In experiment, $L = 10$ for (\mathbf{X}) and $L = 4$ for (\mathbf{d}).
- Why is this? (Explored by the authors)
 - [Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains](#)

Hierarchical volume sampling

- If we sample a lot of points that do not belong to the object space we won't get any useful information.
- Still, if we only sample some high-volume density regions (points around the mode of the volume density distribution), we may miss out on some other interesting areas.
- What Paper does to increase rendering efficiency?

Hierarchical volume sampling

- Instead of just using a single network to represent the scene, only optimize two networks: one “**Coarse**” and one “**Fine**”.
- First sample a set of N_c locations using stratified sampling, and evaluate the “coarse” network
- Given the output of this “**Coarse**” network, we then produce a more informed sampling of points N_f along each ray where samples are biased towards the relevant parts of the volume.



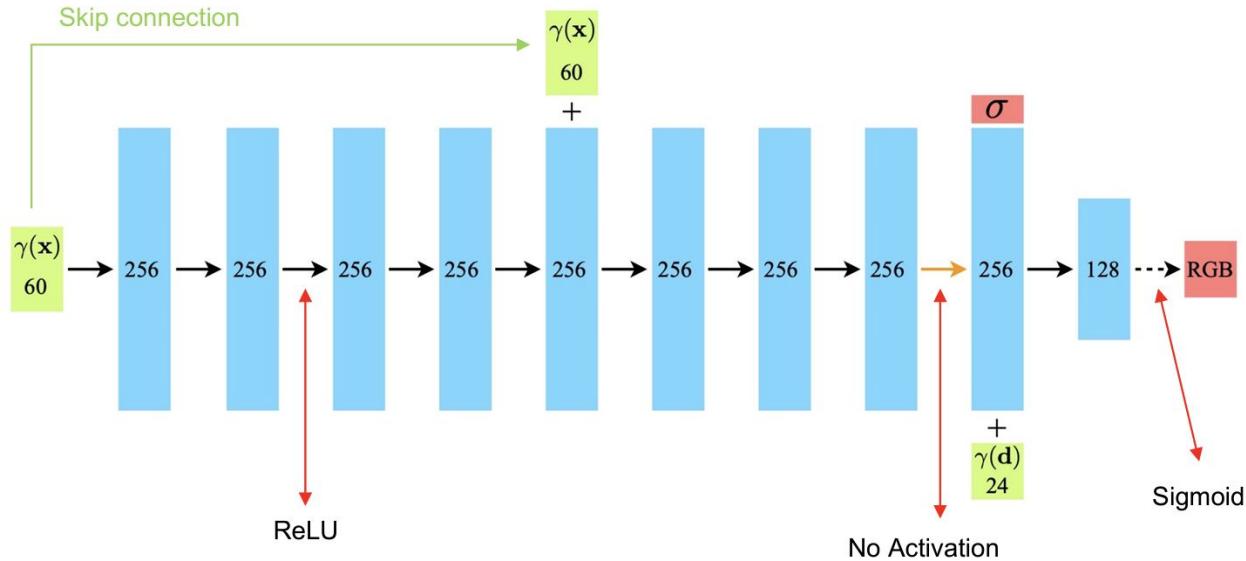
Loss Function

- At each optimization iteration,
 - Randomly sample a batch of camera rays from the set of all pixels in the dataset (**Batch size: 4096 rays**)
 - Follow the hierarchical volume sampling ($N_c = 64$ & $N_f = 128$)
 - Use Volume rendering to render the color of each ray from both set of samples.
 - Calculate loss between rendered and true pixel colors,

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

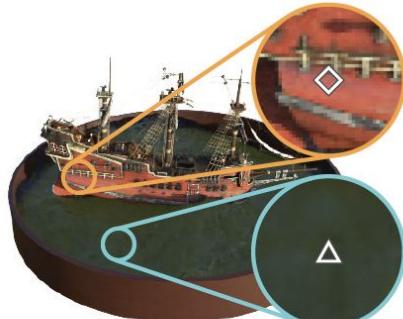
$C(r)$, $\hat{C}_c(r)$, and $\hat{C}_f(r)$ are the ground truth, coarse volume predicted, and Fine volume predicted RGB colors for ray r respectively

Network Architecture

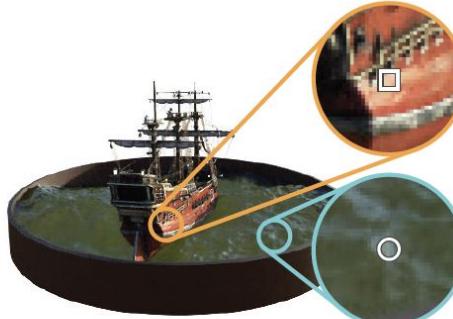


Additional Things

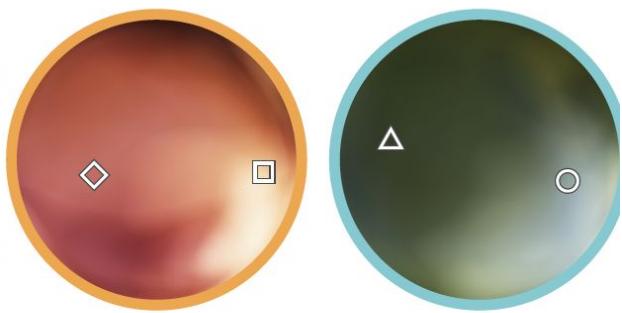
- To be multiview consistent paper restricting the network to predict the volume density σ as a function of only the location X , while allowing the RGB color c to be predicted as a function of both location and viewing direction.



(a) View 1

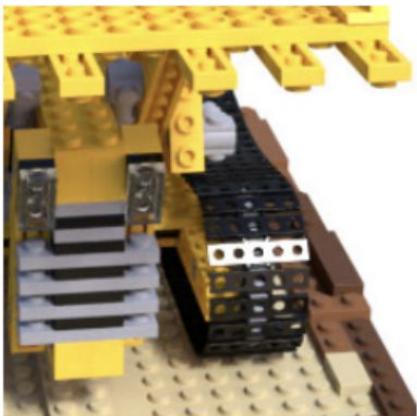


(b) View 2



(c) Radiance Distributions

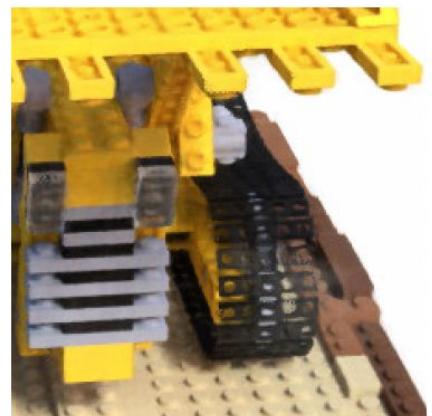
Additional Things



Ground Truth



Complete Model



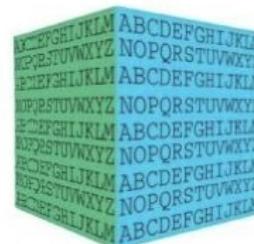
No View Dependence

Dataset & Results

- Synthetic rendering of objects
 - Deep Voxel Dataset (Diffuse Synthetic 360°)
 - Authors own Dataset (Realistic Synthetic 360°)
- Real Image of Complex Scenes.

Deep Voxel Dataset (Diffuse Synthetic 360°)

- 4 Objects with simple Geometry.
- Each rendered at 512 x 512 Pixels from viewpoint sampled on the upper hemisphere. (479 as input and 1000 for testing)



Realistic Synthetic 360°

- 8 Objects with simple Geometry.
- Each rendered at 800 x 800 Pixels out of 6 from viewpoint sampled on the upper hemisphere and 2 from viewpoint sampled on the full hemisphere (100 as input and 200 for testing)



Real Image of Complex Scenes

- Complex real-world scenes captured with roughly forward-facing images
- Consists of 8 scenes captured with 20 to 62 images, and hold out 1/8 of these for the test set.
- All images are 1008 x 756 pixels.

Results



Results



Extra Result : 360° Scene Capture with Real Data

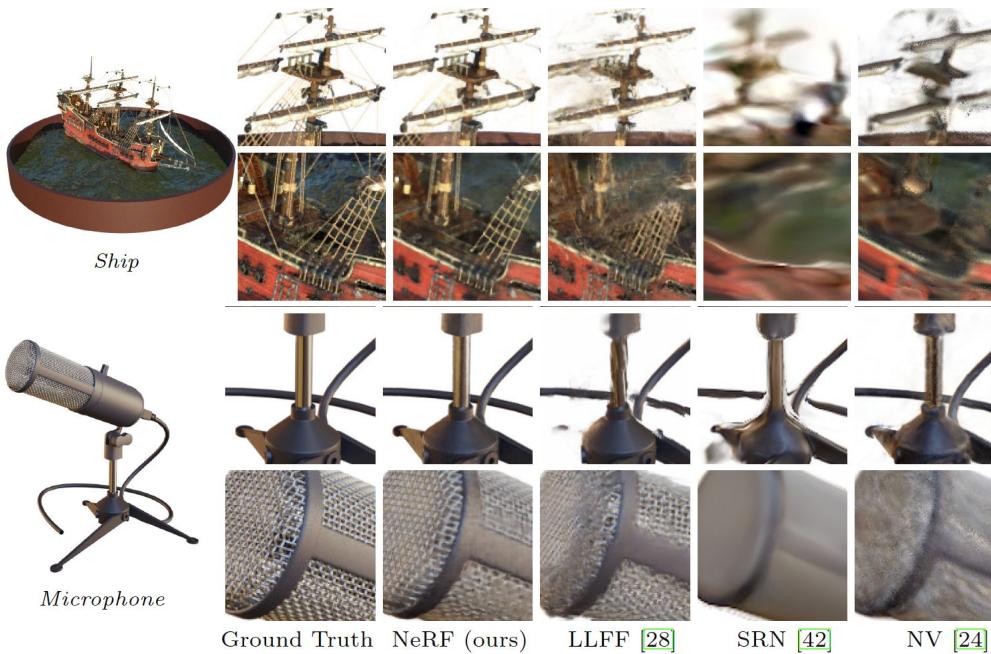


Comparison of Results

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

- **PSNR** : Peak Signal-to-Noise Ratio (**higher is better**)
 - used to quantify reconstruction quality for images and video subject to lossy compression
- **SSIM** : Structural Similarity Index Measure (**higher is better**)
 - used for measuring the similarity between two images
- **LPIPS** : Learned Perceptual Image Patch Similarity (**lower is better**)
 - computes the similarity between the activations of two image patches for some pre-defined network

Qualitative Results Comparison : Realistic Synthetic 360°



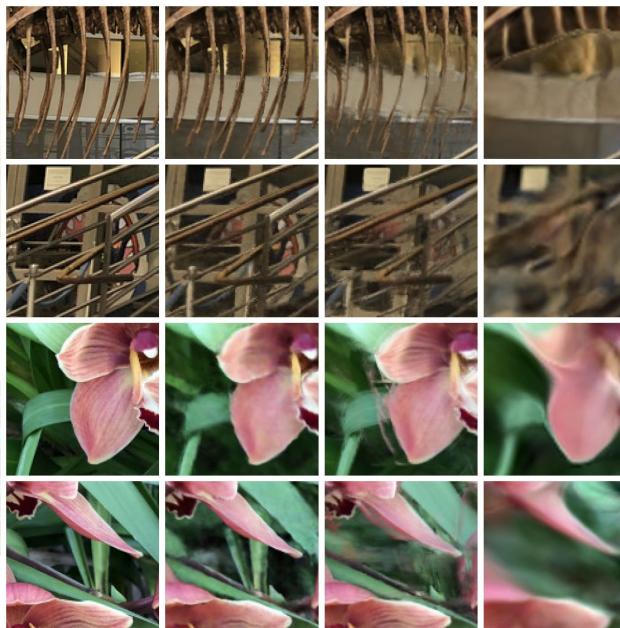
Qualitative Results Comparison : Real World Scenes



T-Rex



Orchid



Ablation Studies

	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

On Realistic Synthetic 360° Dataset

Good Things About NeRF

- NeRF preserves fine details much better than other algorithms
- NeRF is able to render partially occluded regions
- The trained MLP has relatively low storage requirements: about 5MB
- The authors show very impressive qualitative results and show state-of-the-art performance with quantitative metrics and different scene types.

Limitations

- NeRF only works with static scenes.
- It requires a significant number of images of the same object.
- A trained NeRF model does not generalize to more than one scene.
- Computationally expensive: 1-2 days to train each individual scene on a modern GPU.
- Inference is slow : each pixel in a synthesized image requires volume rendering
- Camera pose of each image is required.
- I don't know but : think how well would it work for Transparent or Semi-Transparent Objects (Like Prism).

Future Direction

- How can we improve Improve Speed?
- Can we make make more generalize modal?
- NeRF works well on images of static subjects captured under controlled settings, it is incapable of modeling many ubiquitous, real-world phenomena in uncontrolled images, such as variable illumination or transient occluders. (How can we do that?)
 - [NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections](#)

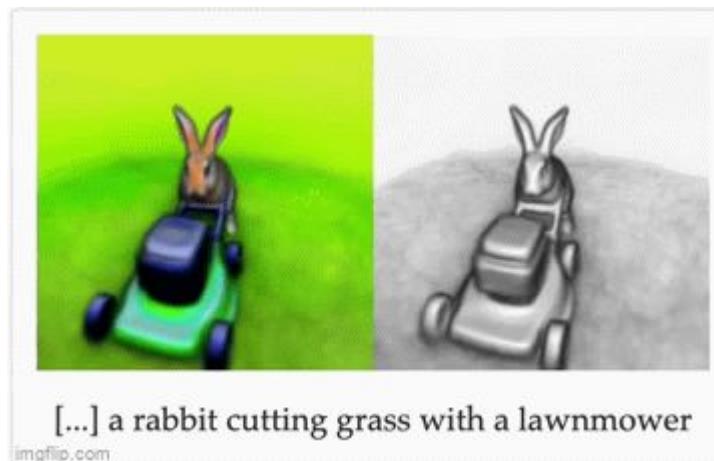
Future Direction

- Can we change the only lighting condition in the generated output?
 - [NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis](#)



Future Direction

- One last thing, Using nerf with generative model like Diffusion.
 - [DreamFusion: Text-to-3D using 2D Diffusion](#)



Summary

- Novel view synthesis generates images of scenes at previously unseen viewpoints.
- Prior works are limited to simple shapes and do not scale well to high-resolution images.
- NeRF encodes a static scene within the parameters of a feedforward neural network.
- The authors show very impressive qualitative results and show state-of-the-art performance with quantitative metrics and different scene types.