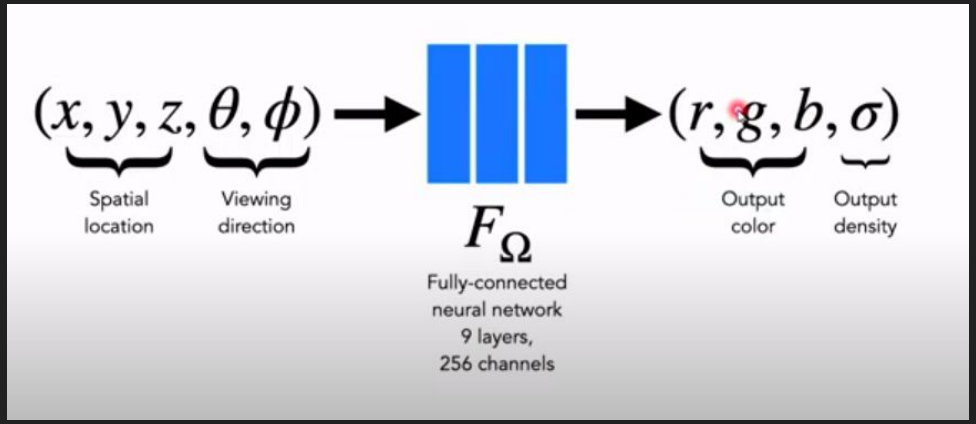
Neural Radiance Fields: NeRF

3DVSS

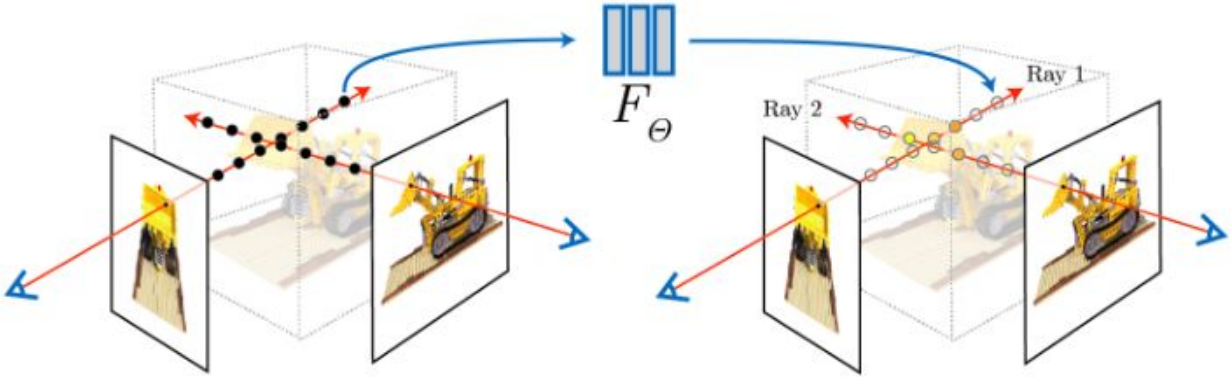
https://www.matthewtancik.com/nerf

Overview

We use a MLP to represent a scene. We input the coordinates of a point and the camera position (x,y,z,theta,phi) and output the color and volume density of that point. Once trained we can then give the network any point and view the scene from that point. This is also referred to as Novel View Synthesis.

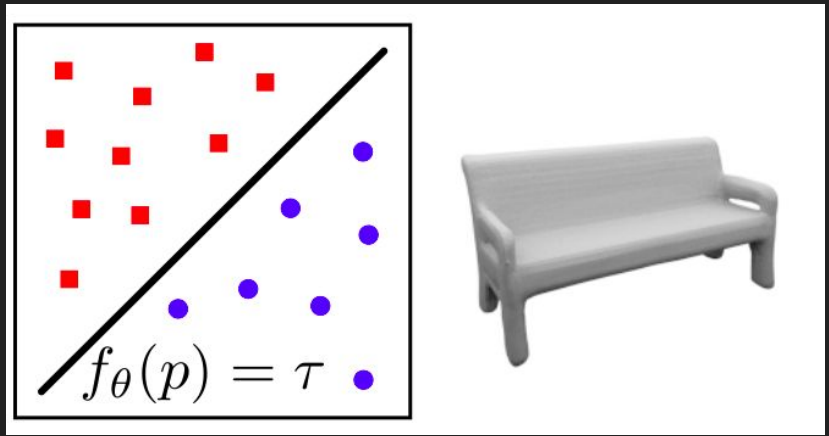


A NeRF stores a volumetric scene representation as the weights of an MLP, trained on many images with known pose.



Occupancy Grids

Before we move on to NeRF, lets understand the direction. 3D scenes can be represented by Point Clouds, Meshes, Voxels. However a new kind of representation is called Implicit Representation where the Neural Network is used to predict if a point is inside the object or outside. This is called occupancy grid. This saves memory and gives smoother surfaces.



Why NeRF?

1. A large scene can be represented in just 5-10 mb. So can be used for compressing the scene in a MLP.
2. Implicitly learns depth in completely self supervised way.
3. Learn scene as continuous representation.
4. Gives a lot of control over the scene parameters like illumination, scene editing etc.

Volume Rendering

The 5D neural radiance field represents a scene as the volume density and directional emitted radiance at any point in space. The volume density σ(x) can be interpreted as the differential probability of a ray terminating at an infinitesimal particle at location x. The expected color C(r) of camera ray r(t) = o + td with near and far bounds tn and tf is

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

Rendering a view from our continuous neural radiance field requires estimating this integral C(r) for a camera ray traced through each pixel of the desired virtual camera. Since this is an integral inside an integral we estimate these integrals using Monte Carlo methods where we sample a lot of points in the function and take an average of them. Delta is the diff b/w 2 points sampled on the ray.

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

Rendering model for ray r(t) = o + td:

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

weights

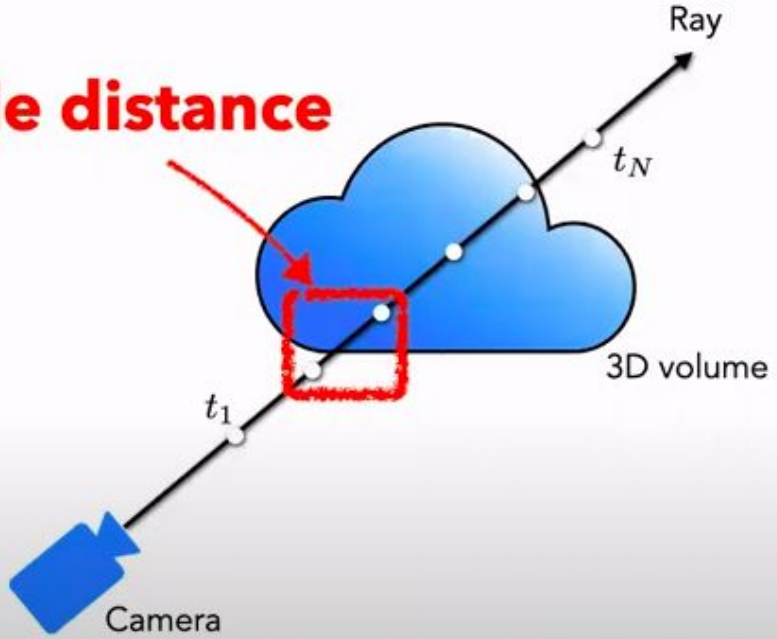
colors

How much light is blocked earlier along ray:

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

How much light is contributed by ray segment *i*:

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

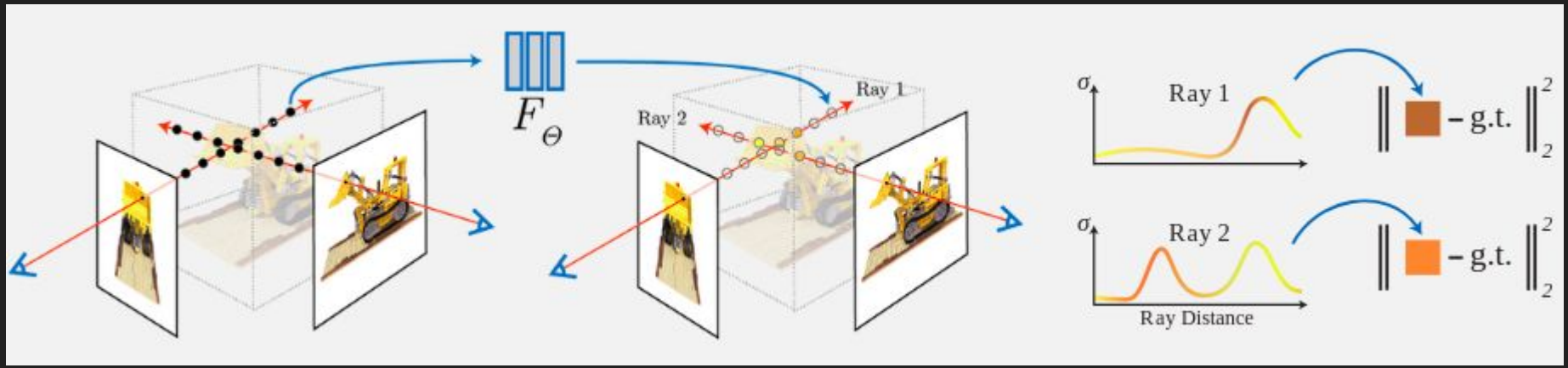


How to run NeRF

Input: 40-50 Images of the scene along with their camera poses. This can be obtained using Structure from Motion softwares like COLMAP.

Steps:

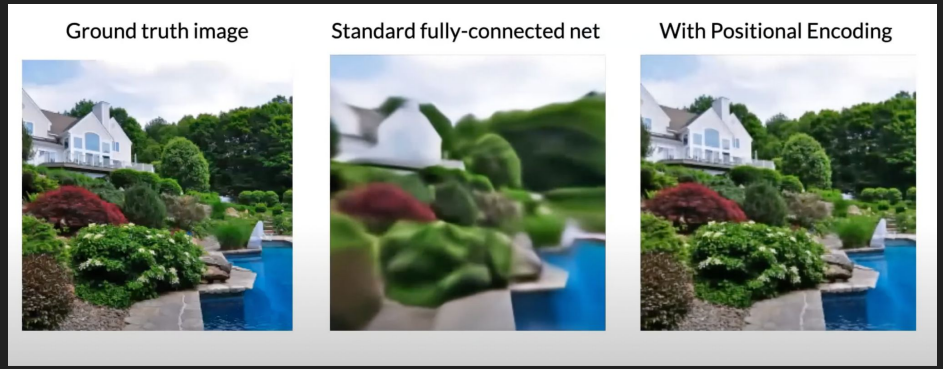
1. March Camera rays through the scene
2. Sample points on the camera rays. Take each point as input to the network and predict the color and volume density at that point.
3. Use alpha compositing to combine the colors at each point on the ray to give the color of the pixel.
4. Use traditional volume rendering to render the image
5. Take L1/MSE loss with GT image.


F_Θ
Ray 1
Ray 2
σ
Ray 1
– g.t.
σ
Ray 2
– g.t.
Ray Distance

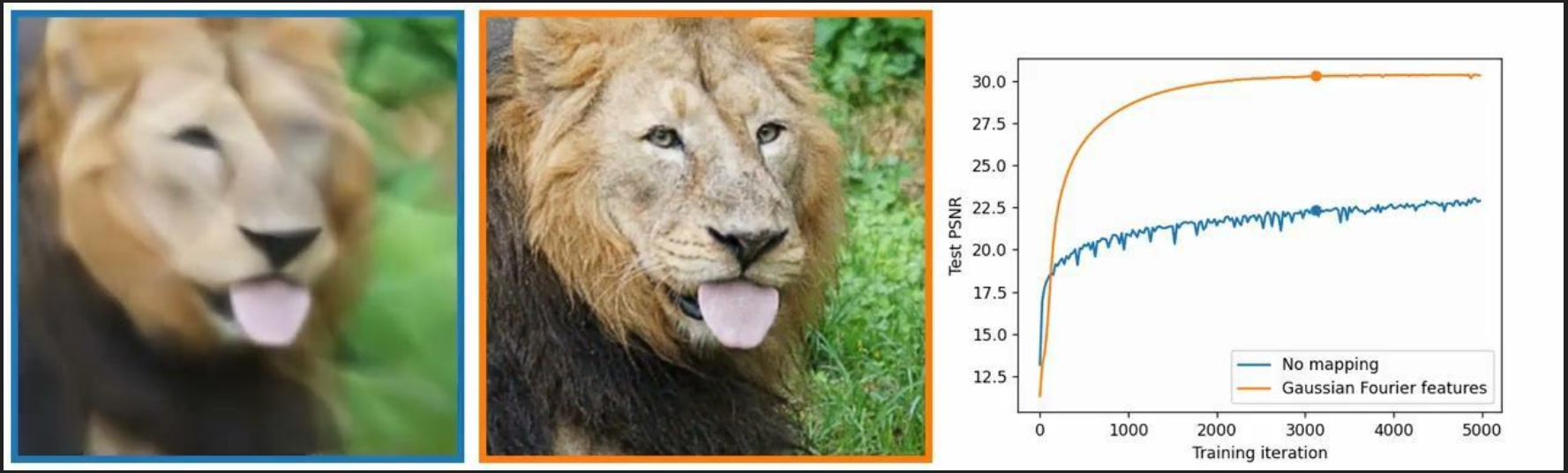
Positional Encoding

Consider a MLP which takes in x,y of the image and outputs color. This simple MLP we observe isnt able to represent the high frequency data. Thus we need to map this into some other dimension.For an input point v (for the example above, (x, y) pixel coordinates) and a random Gaussian matrix B, where each entry is drawn independently from a normal distribution N(0, σ2)

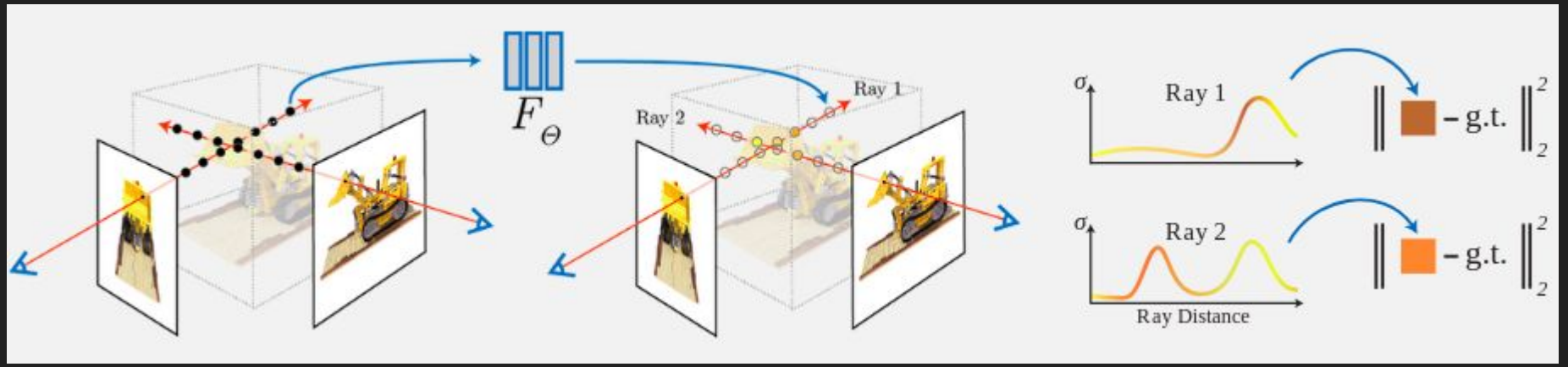
$$\gamma(\mathbf{v}) = [\cos(2\pi \mathbf{B}\mathbf{v}), \sin(2\pi \mathbf{B}\mathbf{v})]^{\mathrm{T}}$$



Positional Encoding



Volumetric Rendering Problems



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

Issues

1. It is extremely slow to render. Inference time per image takes 30-40 seconds
2. It bakes in color so you cannot change the lighting and material
3. It only works for a small bounded scene and not for outdoors
4. Object centric and doesn’t generalize
5. Only for rigid scenes.

Demo

1. Training in Blender Data
2. Inference of pretrained model
3. COLMAP overview.
4. Instant NGP
5. Positional Encoding Vis

Variants of NeRF

1. NeRF in the wild
2. BARF
3. HyperNeRF
4. RegNeRF
5. City NeRF
6. NeRF in the Dark
7. Block NeRF
8. Instant NGP
9. Animatable NeRF
10. Dynamic view synthesis from dynamic monocular video

Resources

- Project Page
- Mathew Tannick's Talk
- Neural Rendering course Siggraph Asia 2021

- https://tinyurl.com/y4vubxba
- https://tinyurl.com/5dtsy7d2

https://discord.gg/qj7uDJuu

https://twitter.com/smallfly/status/1513274703150034947?t=4FtLo_-VYurGv2nAtYa6ow&s=09