

# Paper Summary NeRF

## Abstract

Represent a scene using a fully-connected deep network, whose input is a single continuous **5D** coordinate (**spatial location**  $(x, y, z)$  **and viewing direction**  $(\theta, \phi)$ ). **Output** is the volume density and view-dependent emitted radiance at that spatial location.

## 1. Introduction

Represent function by regressing from a single 5D coordinate  $(x, y, z, \theta, \phi)$  to a single volume density and view-dependent RGB color. To render this **NeRF** from a particular point:

1. March camera rays through the scene to generate a sampled set of 3D points.
2. Use those **points** and their **corresponding 2D viewing directions** as input to the NN to produce **an output set of colors and densities**.
3. Use classical volume rendering techniques to accumulate those colors and densities into 2D images.

Solve issue: 1. Transferring **input 5D coordinates** with a positional encoding that enables the MLP to represent higher frequency functions.

2. Propose a hierarchical sampling procedure to reduce the number of queries required to adequately sample this high-freq scene representation.

## 2. Related Work

Recent direction: Maps from a 3D spatial location to an implicit representation of the shape.

\* Paper circumvent problem by encoding a *continuous* volume within the parameters of a deep fully-connected NN, which not only produces significantly higher quality renderings than prior volumetric approaches, but also requires just a fraction of the storage cost of those *sampled* volumetric representations.

## 3. Neural Radiance Field Scene Representation

Input: a 3D location  $x = (x, y, z)$  and 2D viewing direction  $(\theta, \phi)$ .

Output: an emitted color  $c = (r, g, b)$  and volume density  $\sigma$ .

In practice, expressing direction as 3D Cartesian unit vector  $d$ . Approximate this continuous 5D scene representation with an MLP network  $F_{\Theta} : (x, d) \rightarrow (c, \sigma)$  and optimize its weights  $\Theta$ .

## 4. Volume Rendering with Radiance Field

$\sigma(x)$  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  $t_n$  and  $t_f$  is:

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

where  $T(t) = \exp(-\int_{t_n}^t \sigma(r(s)) ds)$ .

## 5. Optimizing a Neural Radiance Field

Introducing two improvements to enable representing high-resolution complex scenes. The first is a **positional encoding** of the input coordinates that assist the MLP in representing high-frequency, and the second is a hierarchical sampling procedure that allows us to efficiently sample this high-frequency representation.

### 5.1 Positional encoding

1. Deep network are biased toward learning lower frequency functions

2. 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.

\* Reformulating  $F_\theta$  as a composition of two function  $F_\theta = F'_\Theta \circ \gamma$ , one learned and one not.

$\gamma$  is a mapping from  $R$  into a higher dimensional space  $R^{2L}$  and  $F'_\Theta$  is still a regular MLP. Formally, the encoding function used is:

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

$\gamma(\cdot)$  is applied separately to each of the three coordinate values in  $\mathbf{x}$  and to the three components of the Cartesian viewing direction unit vector  $\mathbf{d}$

### 5.2 Hierarchical Volume Sampling

## Part 2

The task of NeRF is "Novel View Synthesis".

**Def** of this task is "creating new views of a specific subject starting from a number of pictures taken from given point of views".

What is "Radiance Field"?

Radiance field is a functional representation that jointly model **geometry** and **appearance**, and is able to model view-dependent effects.

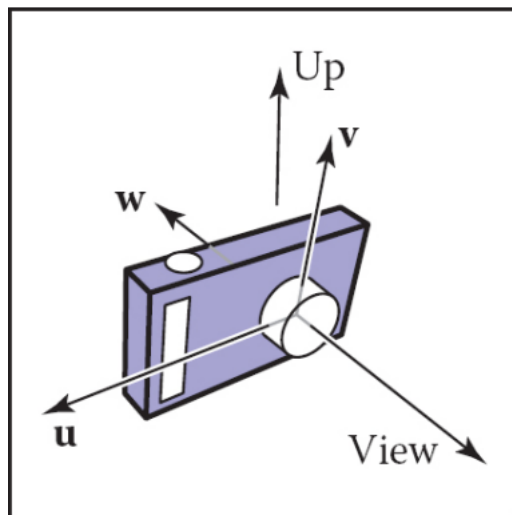
Geometry: 3d points location  $\mathbf{x} = (x, y, z)$  and view direction  $\mathbf{d}$  as a part of input.

Appearance: density  $\sigma$  and color  $(r, g, b)$ .

$$F_{\Theta} : (\mathbf{x} = (x, y, z), \mathbf{d} = (u, v)) \rightarrow (\mathbf{c} = (r, g, b), \sigma([(x, y, z)]))$$

\*  $\sigma : \sigma(\mathbf{x})$  a function of only the location  $\rightarrow$  **multi-view consistent** because density  $\sigma$  is irrelevant with view direction, but just location  $(x, y, z)$ .

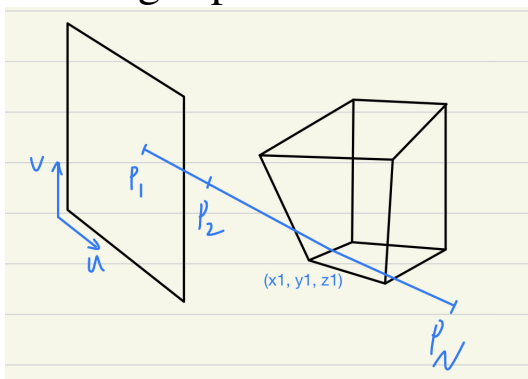
\*  $\mathbf{c}$  is relevant with both **view direction** and **position** because of relation between **specular characteristic** and view direction.



How to represent  $\mathbf{d}$ ?

$\mathbf{d} = (u, v)$

## Training Pipeline



1. Plug view direction  $\mathbf{d} = (\mathbf{u}, \mathbf{v})$  and position of points  $p_1 = (x_1, y_1, z_1), \dots, p_N = (x_N, y_N, z_N)$  into neural radiance field  $F_{\Theta}(\mathbf{d}, p_i)$ . Note: each point each time, and for these points in the fig above, they share the same view direction. After this, we will get  $N$  numbers of  $(\mathbf{c}, \sigma)$ .

Procedure **1** and **2** in the paper.

2. Using classical volume rendering techniques ([16] Ray Tracing Volume Densities) to accumulate those colors and densities into 2D image. Using formulae:

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

We can directly substitute  $\sigma$  and  $\mathbf{c}$  into formulae, and also all elements in  $T(t) = \exp(-\int_{t_n}^t \sigma(r(s)) ds)$  are known. We can easily get the value of expected color  $C(r)$ .

$t_n, \dots, t_f$  are the same with points  $p_1, \dots, p_N$ . In practice, they use weighted sum to numerically approximate, where we can treat  $T(t)$  as weight. **Uniform discretization for coarse network**

3. real color (or ground truth) – expected color  $\rightarrow L^2$  norm. That's how to train.

## Optimization Techniques

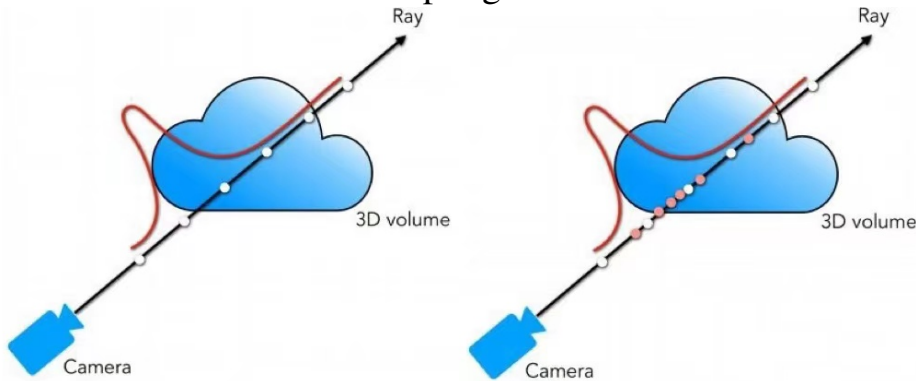
### 1. Positional encoding

Deep networks are biased towards learning low-frequency functions. Reformulating  $F_{\Theta} = F'_{\Theta} \circ \gamma$ . We can treat  $\gamma$  as a kind of **transformation** so that we can **increase the frequency** of our original input data to **decrease the blurring of performance**, and  $F'_{\Theta}$  will train these transformed data. How does these work? Recall the formulation of  $\gamma(p)$ :

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

as the value of order increasing, the frequency of each transformed data will increase.  $\gamma$  is actually a mapping to map original data (**low-frequency**) to be high-frequency and increase performance.

### 2. Hierarchical Volume Sampling



Left: coarse, uniform sampling. Predict the density distribution according to  $\sigma$ , and then add more sample points at where density is high.

Right: fine sampling, Compute the final rendered color of the ray  $\hat{C}_f(r)$  using all  $N_c + N_f$  samples.

Loss function:

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