NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

ECCV 2020, Oral, Best Paper Honorable Mention

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

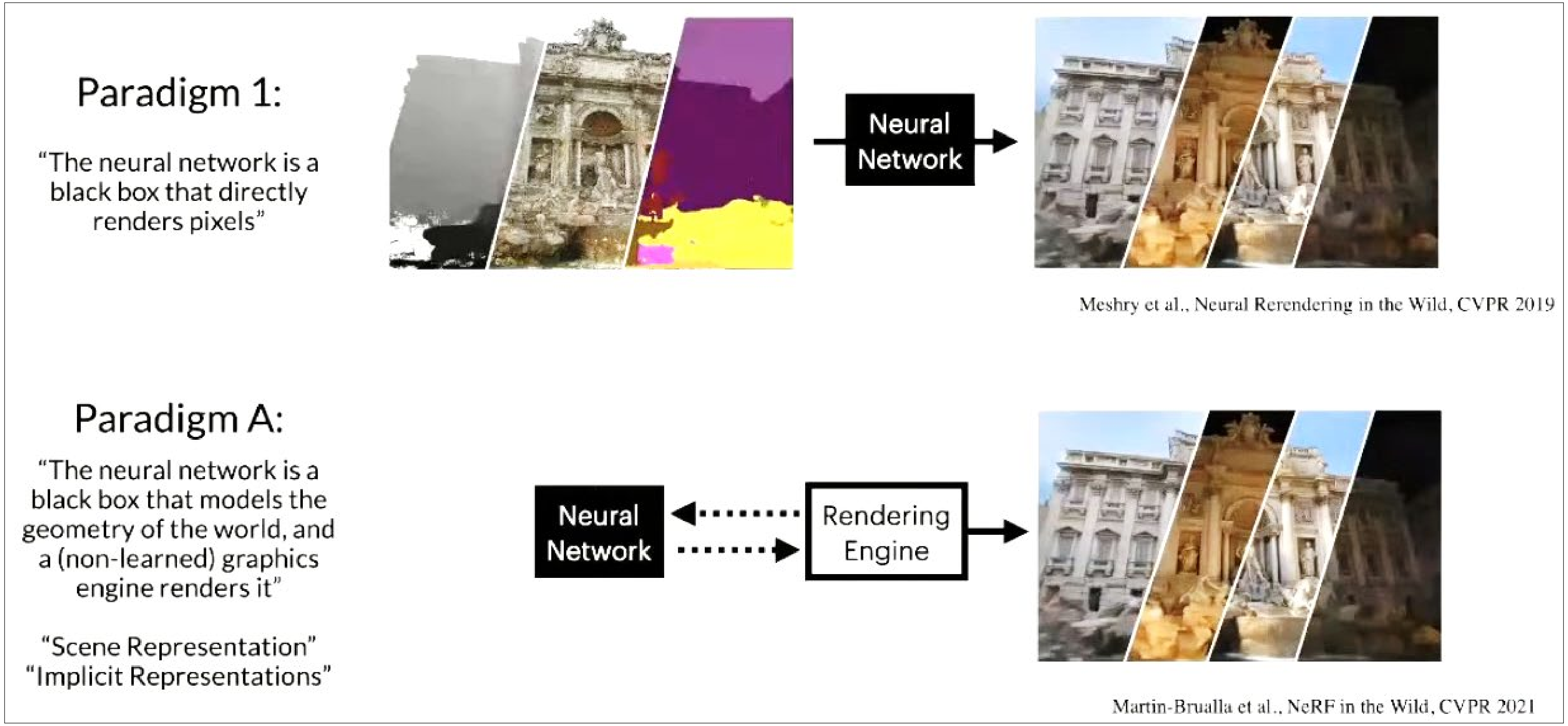
Donggun KIM
dgkim@vclab.kaist.ac.kr

*Some figures are excerpted from the original paper

Neural Rendering?

Explicit, narrow paradigm of “neural rendering”

NeRF



Jon Barron, EGSR 2021 Keynote

Recently, both are called “neural rendering”

Introducing NeRF

Method

Neural network based differentiable volume **Rendering**

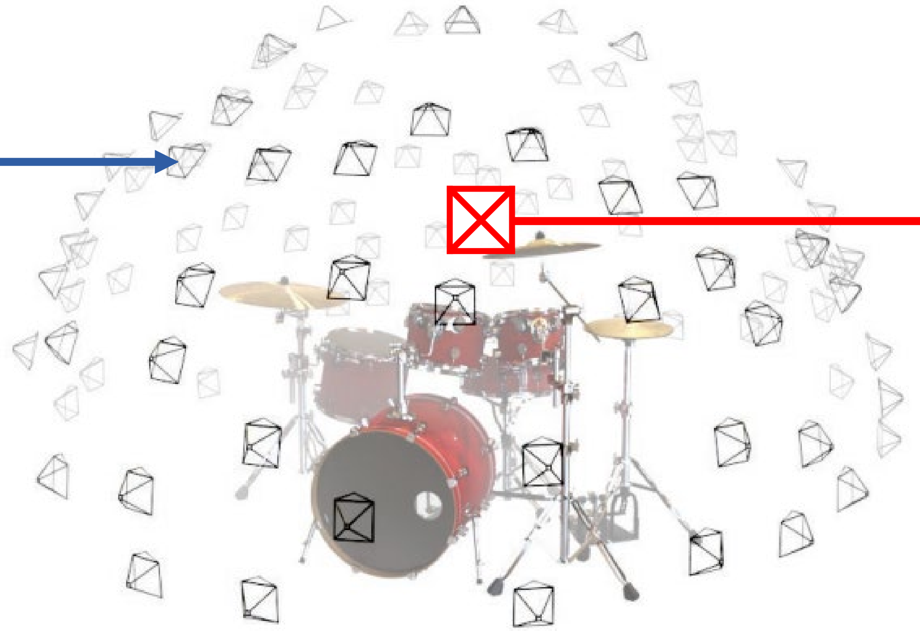
What to solve

View synthesis

Problem Definition: View Synthesis

Rendering at the novel view point with given images

Input images



Render
novel view image
(=Not observed)

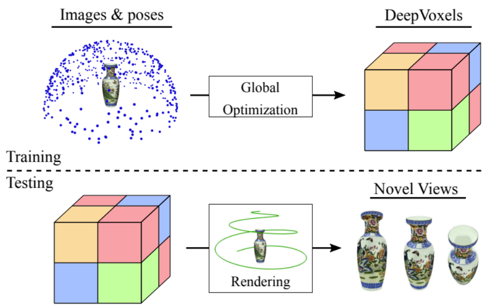
It's straightforward if we have scene geometry and light

But it's challenging in the real world!

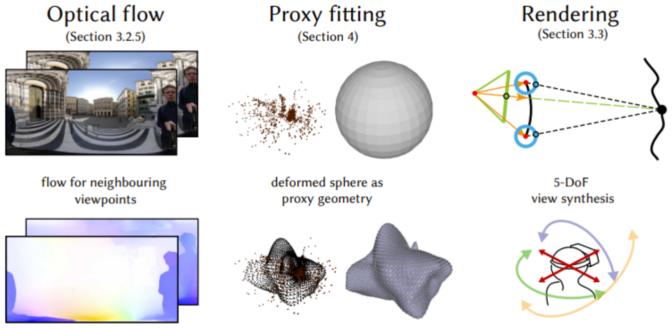
Instead, we can easily capture images

Solving View Synthesis

Reconstruct geometry (mesh, voxel) with texture

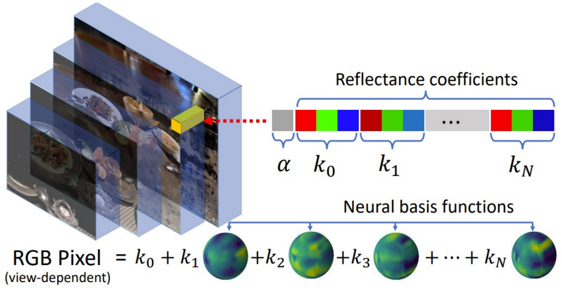


DeepVoxels
[Sitzmann et al., CVPR 2019]



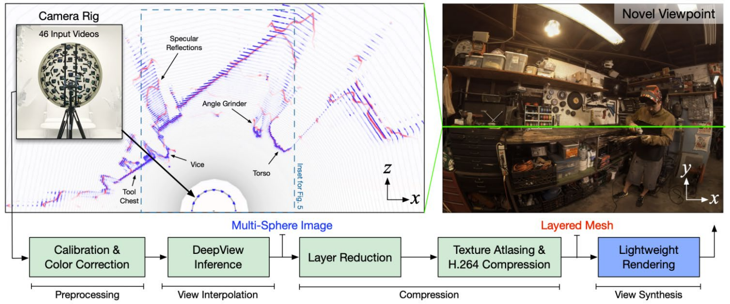
Omniphotos
[Bertel et al., SIGGRAPH Asia 2020]

High-dimensional images (MPI, MSI, Light field)



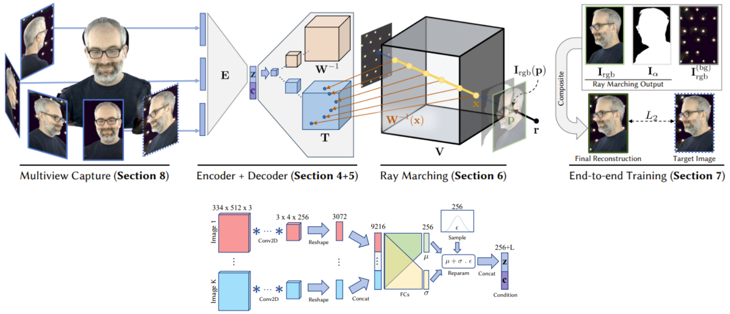
NeX
[Wizadwongsa et al., CVPR 2021]

→Jaemin Cho

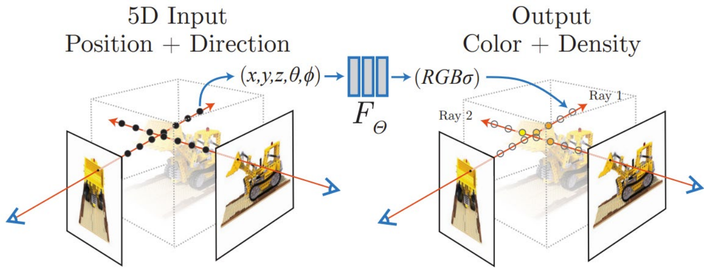


Light Field Video
[Broxton et al., SIGGRAPH Asia 2019]

Reconstruct implicit representation



Neural Volume
[Lombardi et al., SIGGRAPH 2019]



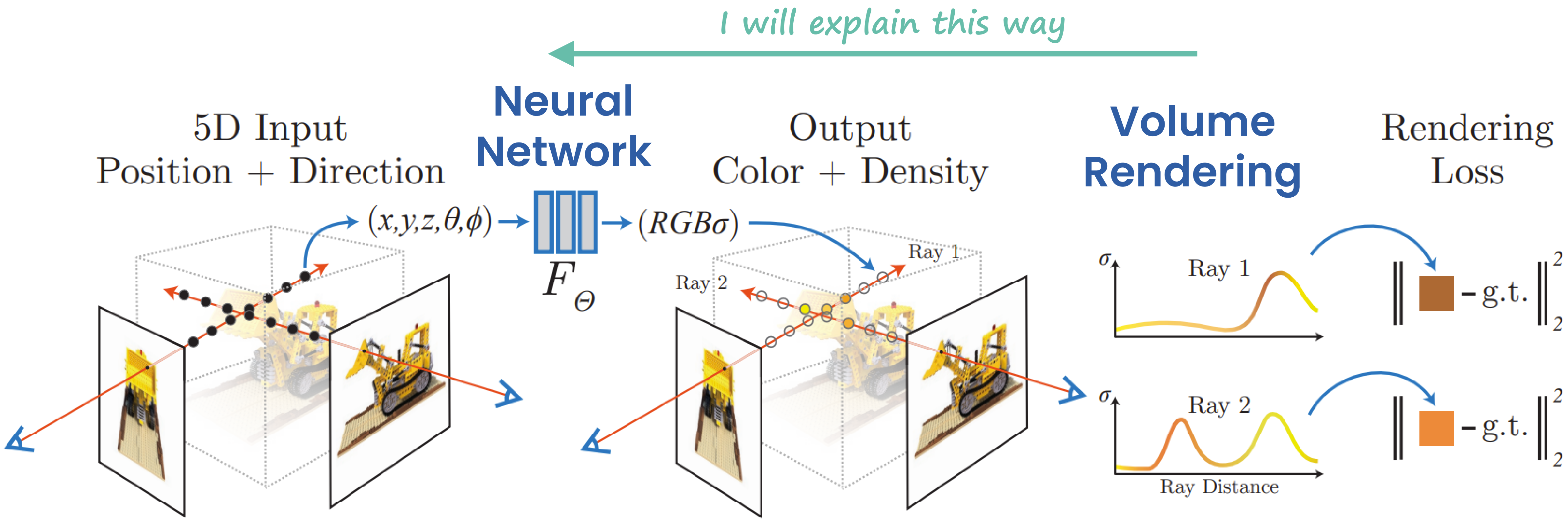
NeRF

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, which we use to optimize these representations from standard RGB images. This includes a hierarchical sampling strategy to allocate the MLP's capacity towards space with visible scene content
- A positional encoding to map each input 5D coordinate into a higher dimensional space, which enables us to successfully optimize neural radiance fields to represent high-frequency scene content

NeRF Overview

Recall: **Neural network** based differentiable **volume rendering**

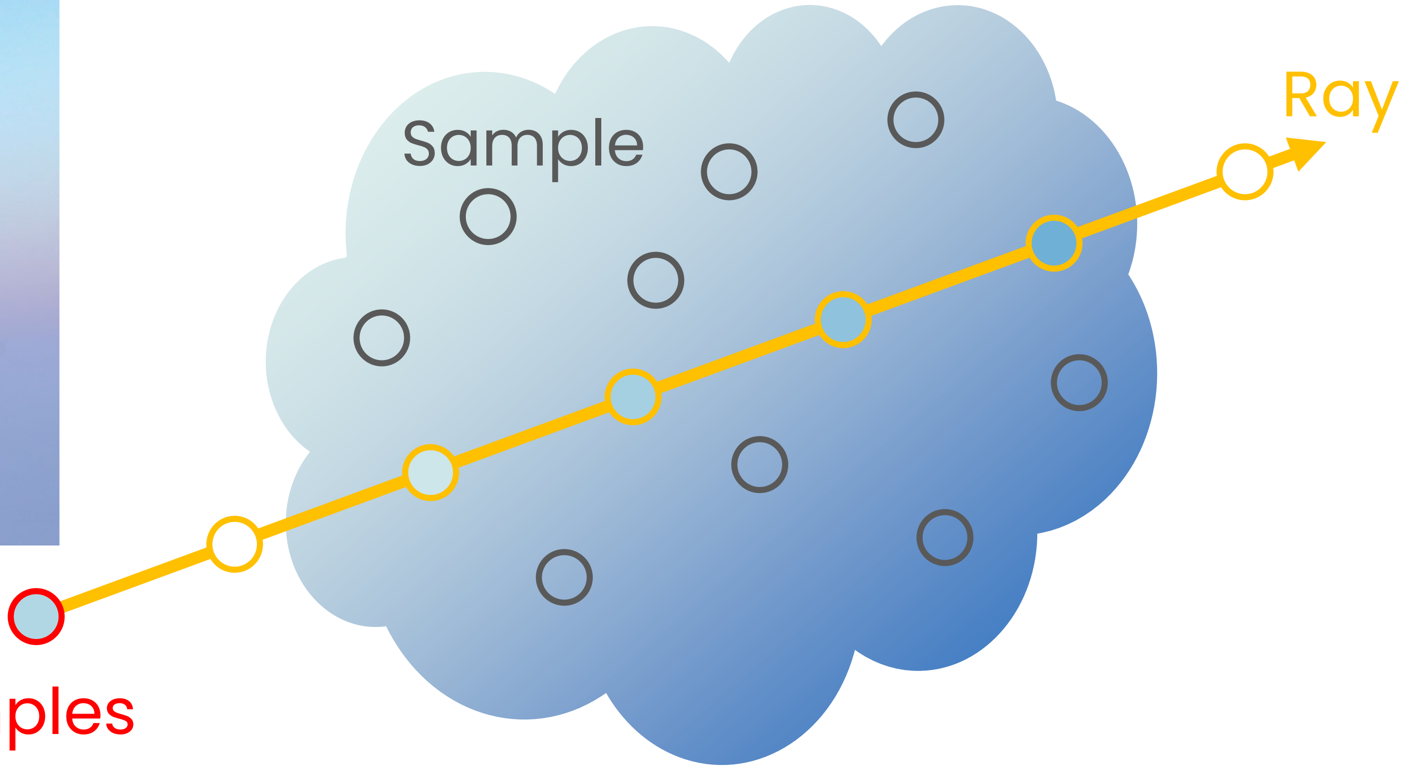


Concept of Volume Rendering

Discretized

Cloud, smoke, fog, …

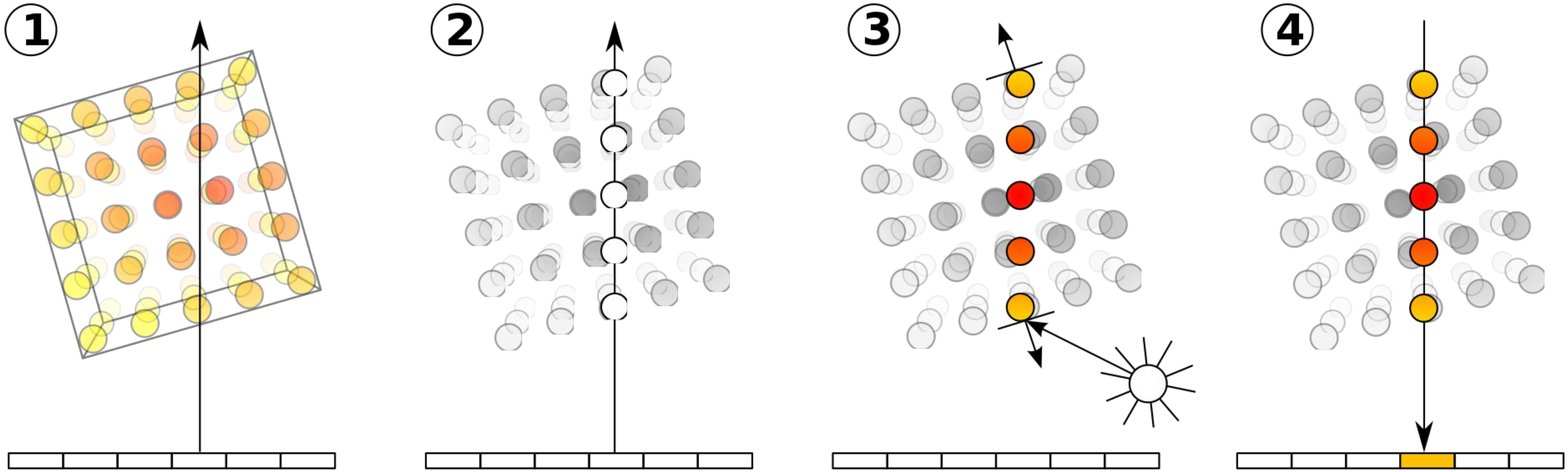
"Volume"



What we see
= Mixed samples

Originally proposed in ~1980s

Volume Rendering



Ray casting

Sampling

Shading (Color & density)

Compositing

https://en.wikipedia.org/wiki/Volume_ray_casting

Volume Rendering is Differentiable

[Max, Optical Models for Direct Volume Rendering, IEEE TVCG 1995]
[Max et al., Local and Global Illumination in the Volume Rendering Integral, 2010]

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i \alpha_i \mathbf{c}_i$$

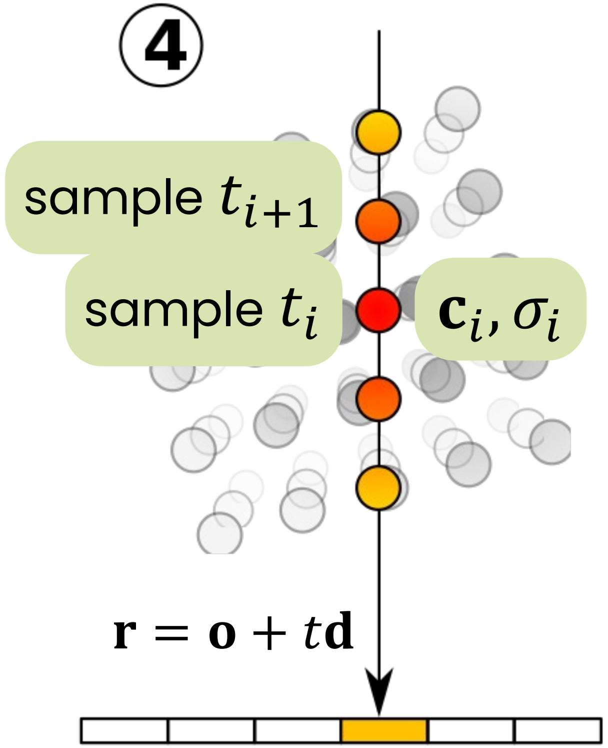
δ_i: distance of light segment, $t_{i+1} - t_i$

$\mathbf{c}_i$: color of sample t_i

σ_i: density of sample t_i

$\alpha_i = 1 - \exp(-\sigma_i \delta_i)$: compositing value

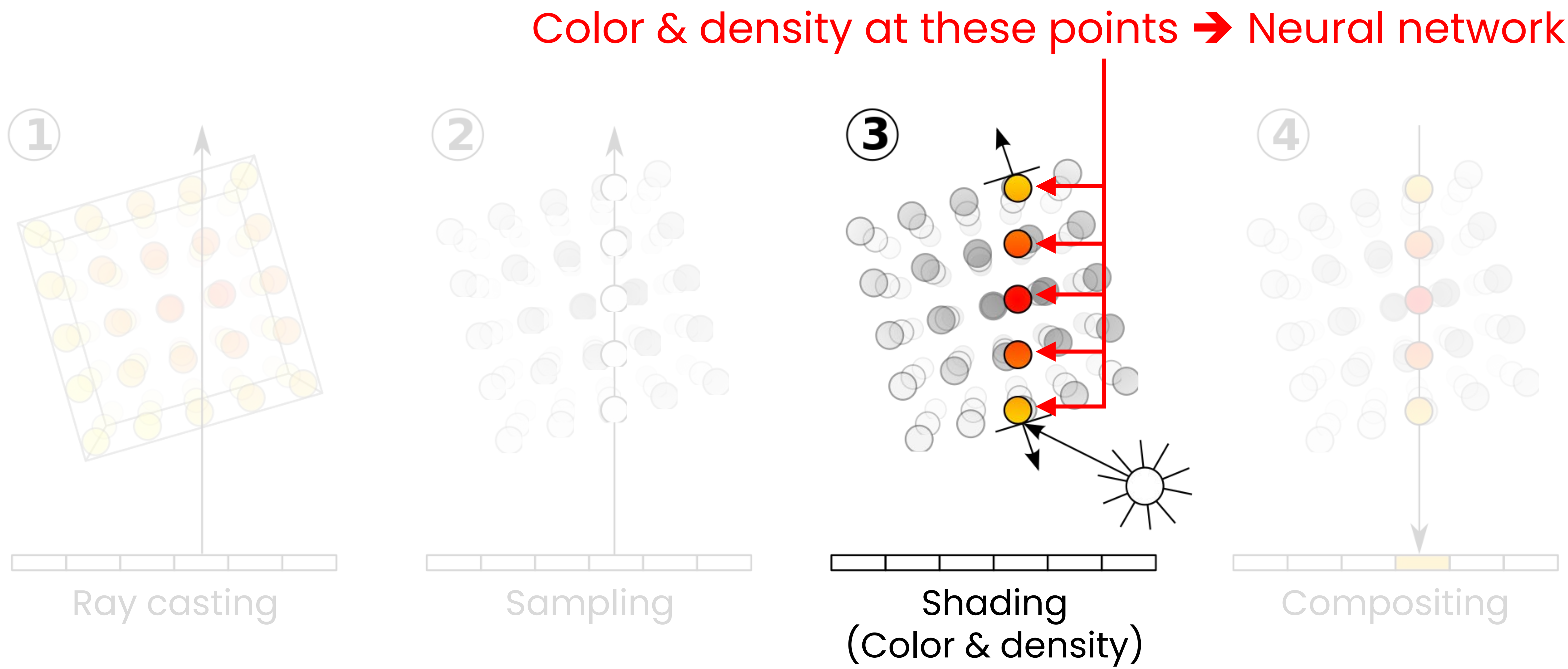
$T_i = \exp\left(-\sum_{j}^{i-1} \sigma_j \delta_j\right)$: accumulated transmittance



Nothing but exponential, add, multiply

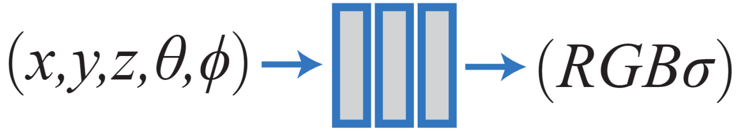
Digression: Path tracing also can be differentiable, but requires complex math [Zhang et al., SIGGRAPH 2020]

Now What We Need?

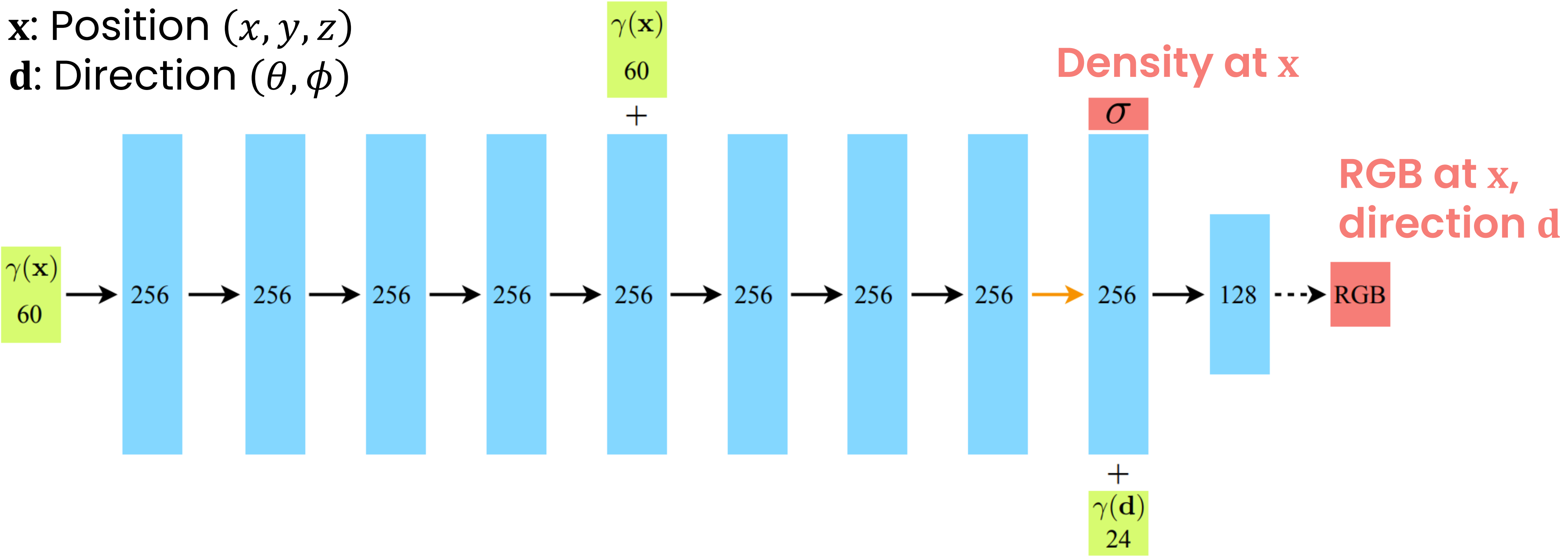


Neural Network

Simple MLP (Multi-Layer Perceptron) is enough



$\mathbf{x}$: Position (x, y, z)
$\mathbf{d}$: Direction (θ, ϕ)



In fact, it's not enough... We need more: $\gamma(\cdot)$

Positional Encoding

Also called **Fourier features**

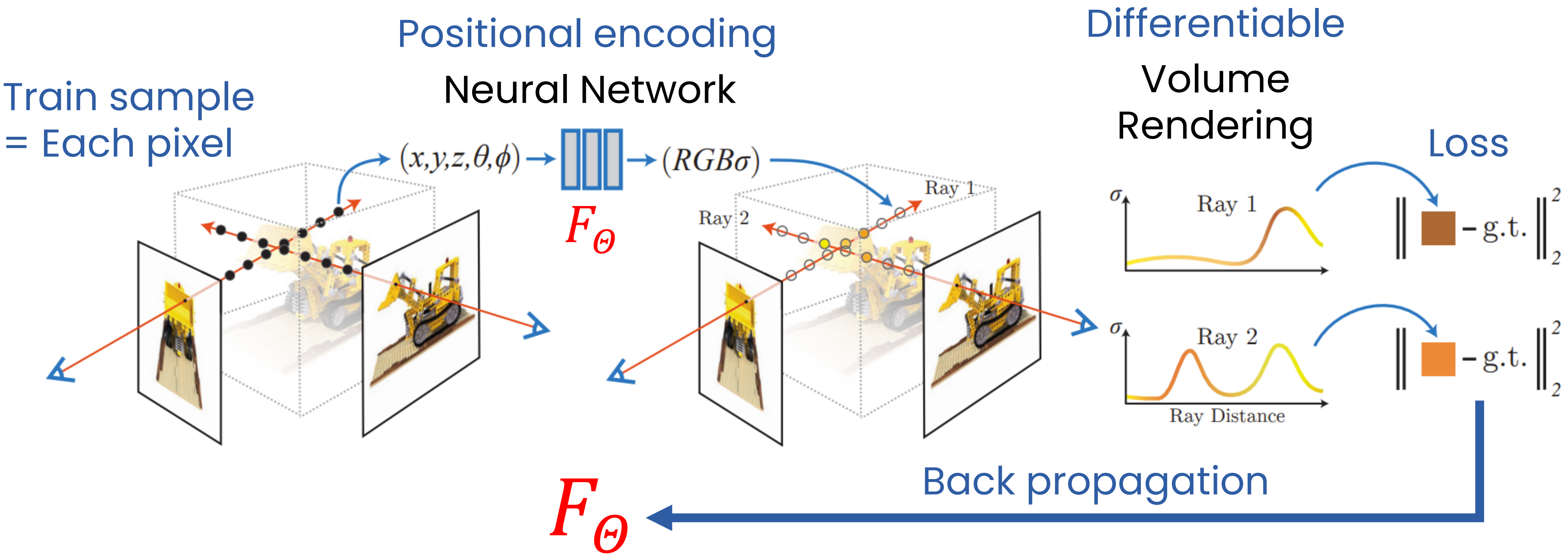
$$\gamma(p) = \left(\sin(2^0\pi p), \cos(2^0\pi p), \cdots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p)\right)$$

(a) Coordinate-based MLP (b) Image regression $(x,y) \rightarrow$ RGB (c) 3D shape regression $(x,y,z) \rightarrow$ occupancy (d) MRI reconstruction $(x,y,z) \rightarrow$ density (e) Inverse rendering $(x,y,z) \rightarrow$ RGB, density

[Tancik et al., NeurIPS 2020]

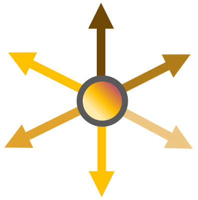
- This enables NeRF to reconstruct both high frequency and low frequency details
- Later, more details are analyzed at [Tancik et al., NeurIPS 2020]

Training Pipeline



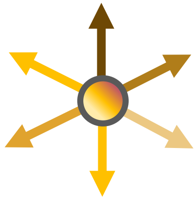
Train neural network that implicitly encode scene representation

"Neural radiance fields"

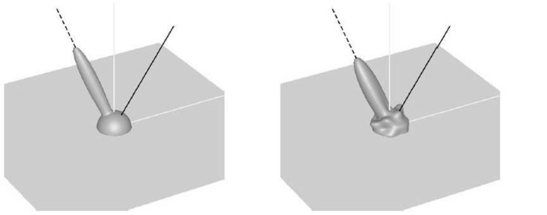


Returns out going radiance @ any 3D point, direction

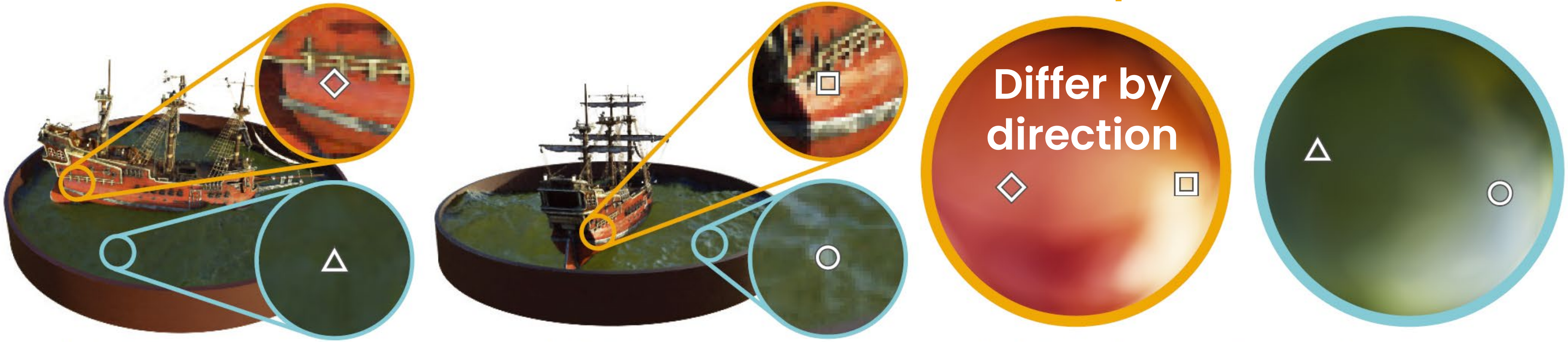
Neural Radiance Fields



Returns out going radiance @ any 3D point, direction



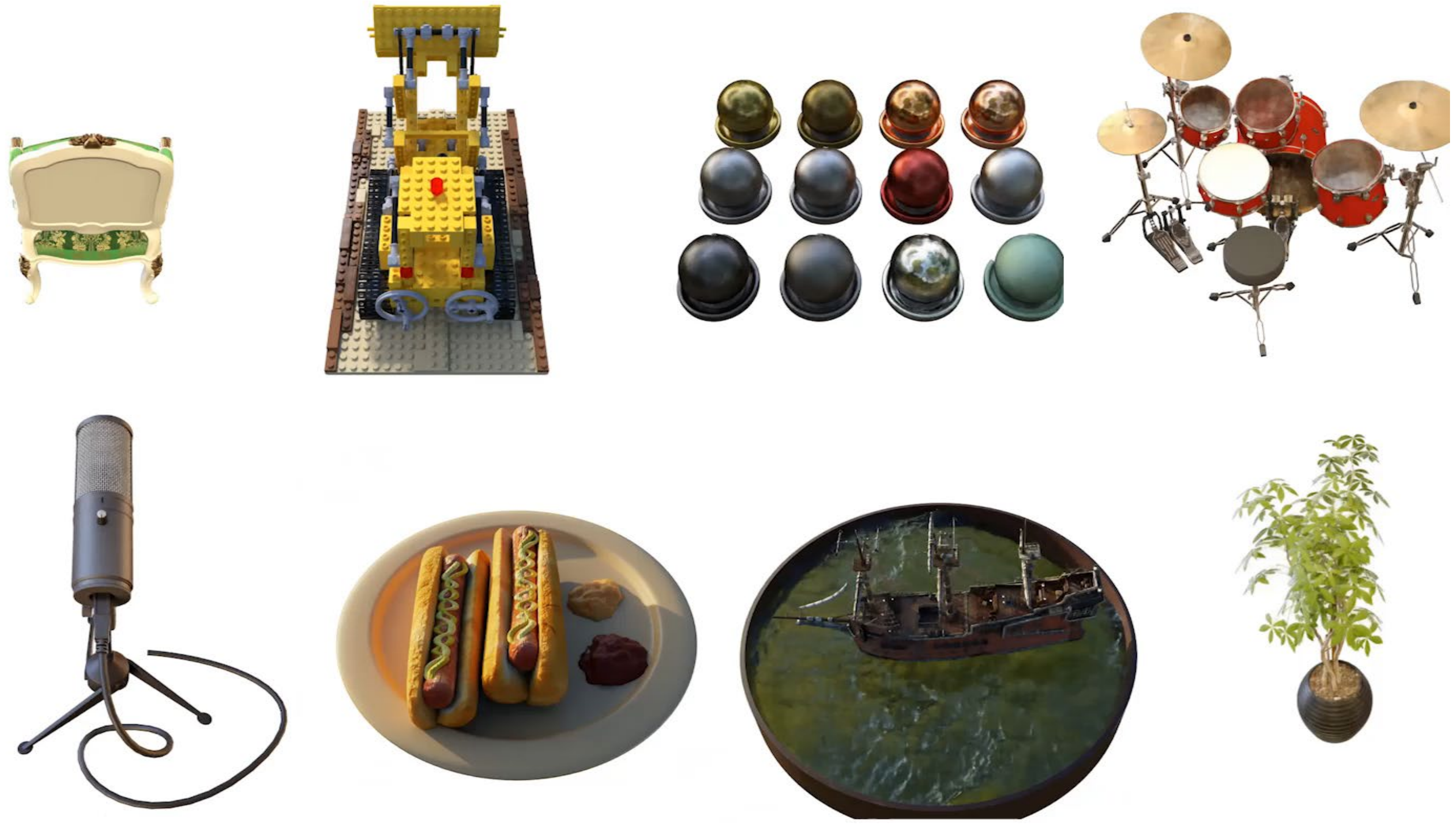
Fixed 3D point



(a) View 1 (b) View 2 (c) Radiance Distributions

Results

Video frames are made by view synthesis

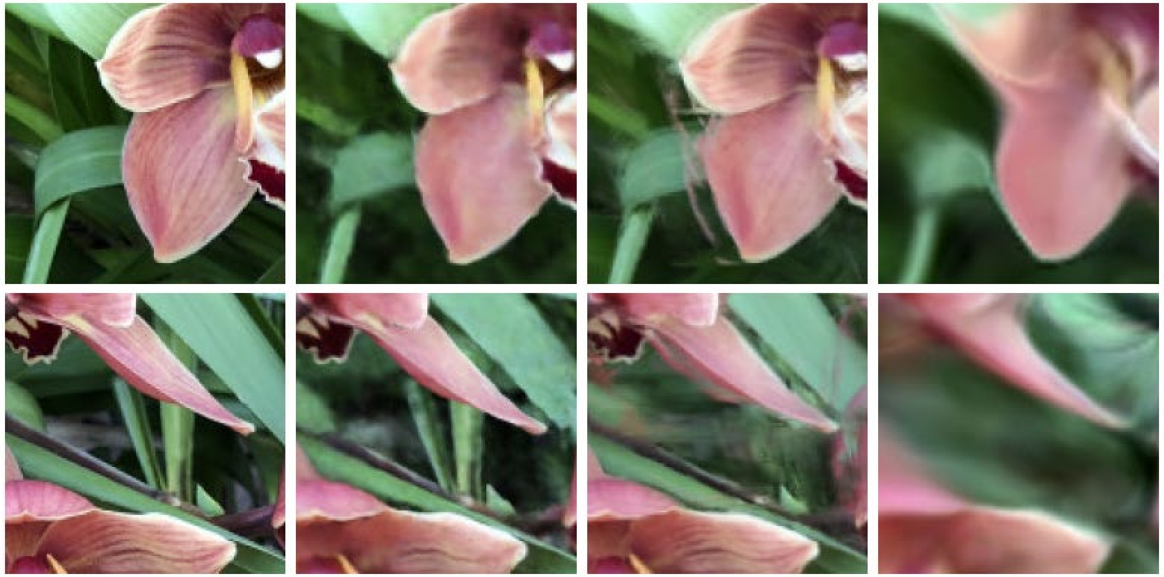


https://www.matthewtancik.com/nerf

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

Orchid



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

NeRF Problems & Improvements

Slow speed

➔ KiloNeRF, **Plenoxels**, FastNeRF, …

→*Kiseok Choi*

Scale dependency (aliasing effect)

➔ Mip-NeRF, BACON, …

→*Dongyoung Choi, Kiseok Choi*

Requires accurate camera calibration

➔ NeRF in the Wild, BARF, NeRF--, …

Cannot handle dynamic scenes / moving objects.

➔ Nerfies, HyperNeRF, NeRFlow, D-NeRF, …

→*Jaehoon Yoo*

Plenoxels: Radiance Fields without Neural Networks

CVPR 2022, Oral

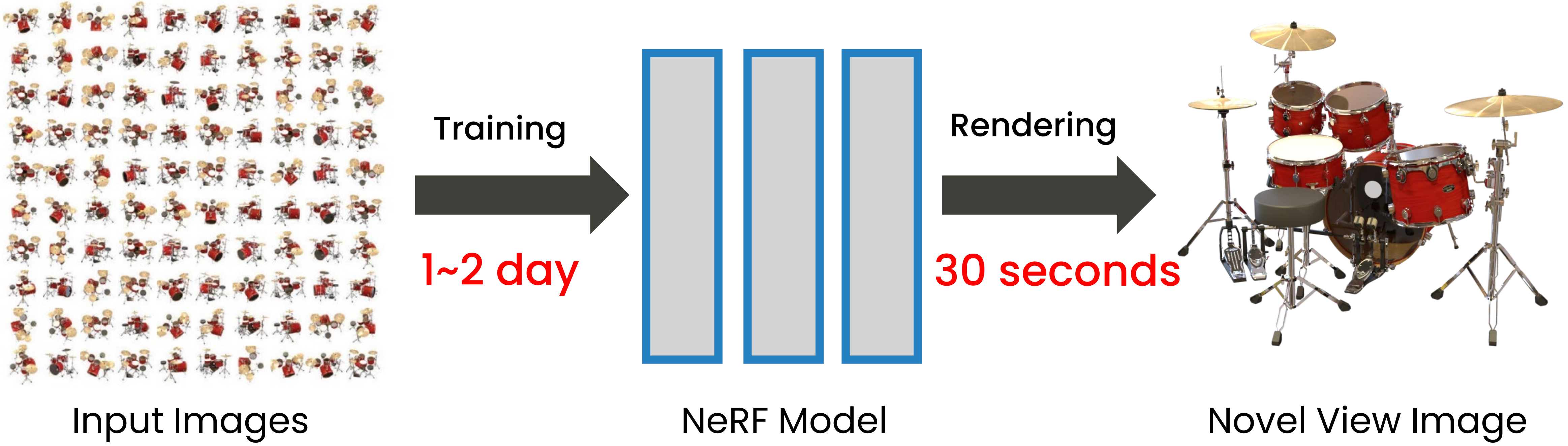
Alex Yu, Sara Fridovich-Keil, Matthew Tancik,
Qinhong Chen, Benjamin Recht, Angjoo Kanazawa

Donggun KIM
dgkim@vclab.kaist.ac.kr

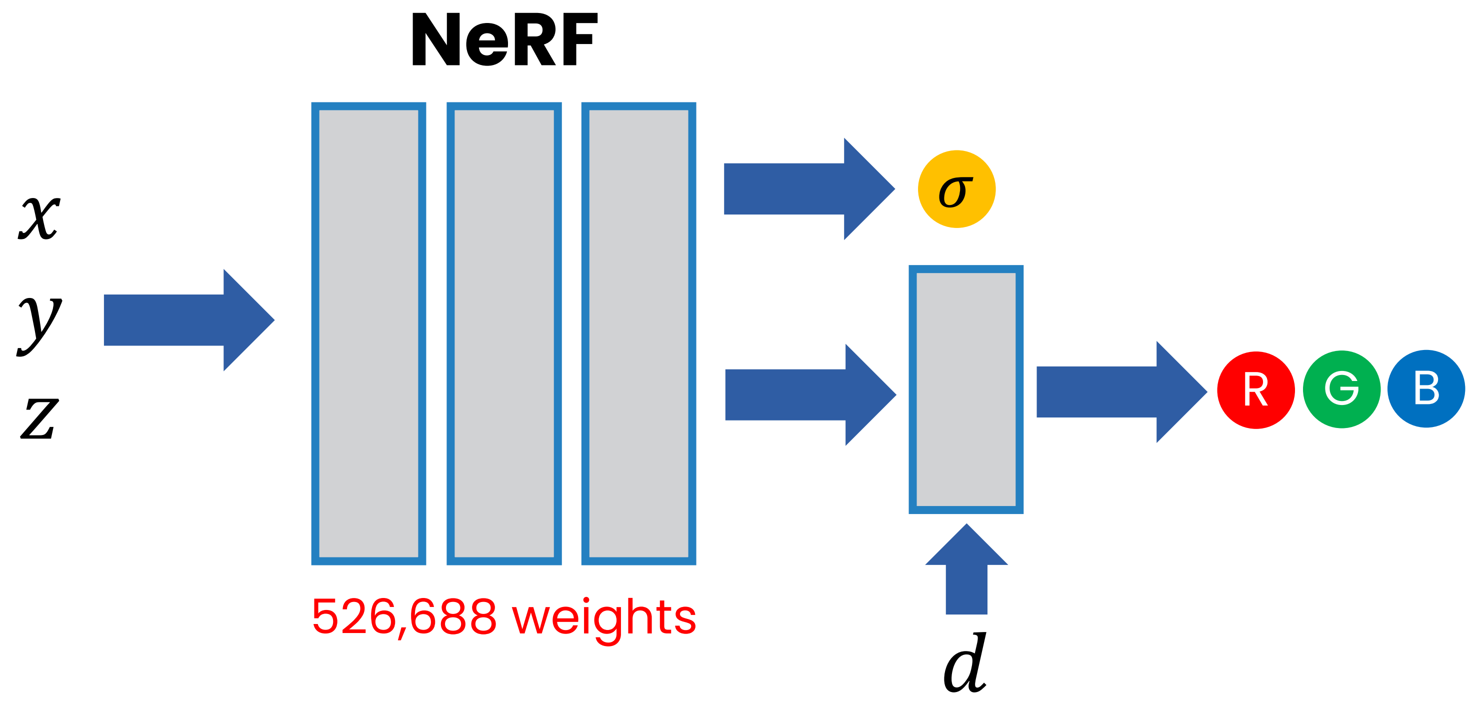
*Some figures are excerpted from the original paper

Problems of NeRF

Slow speed



Why This Happens?



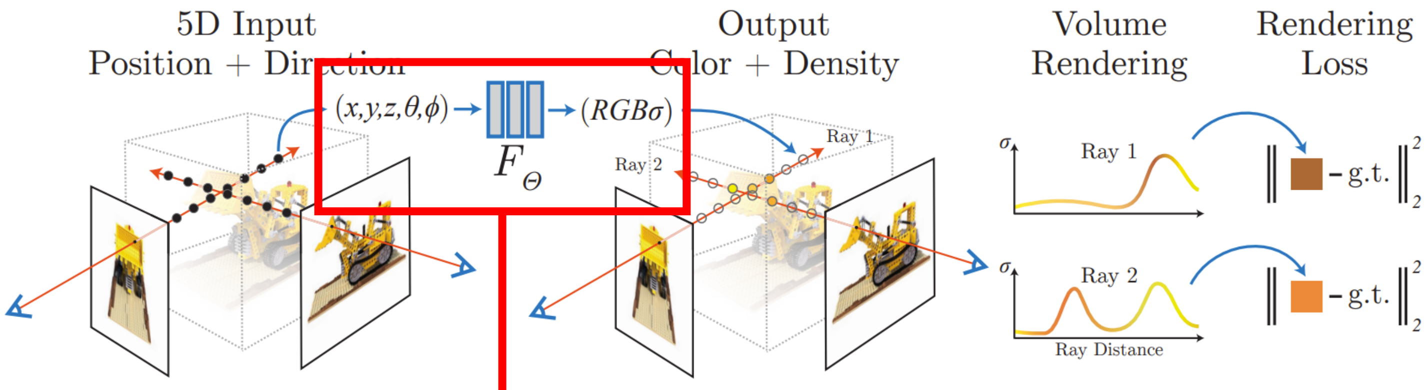
for each image
 for each pixel (800, 800)
  for each sample (256)
   Eval NeRF network

You have to sample densely in $\mathbb{R}^5$

~163 million neural network evaluation / 1 image

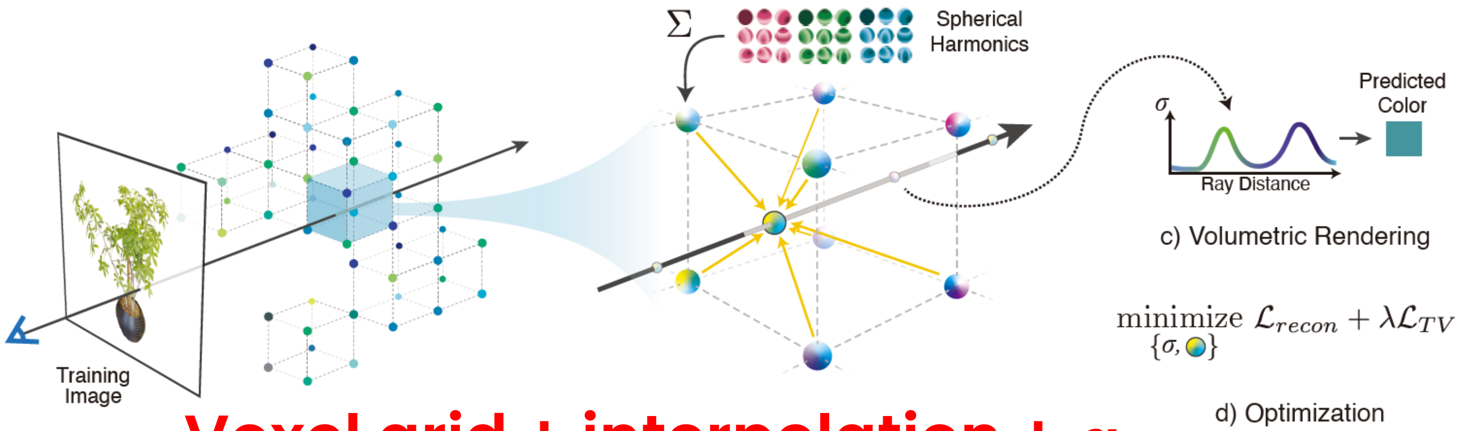
Introducing Plenoxels

NeRF



Without neural network

Plenoxels

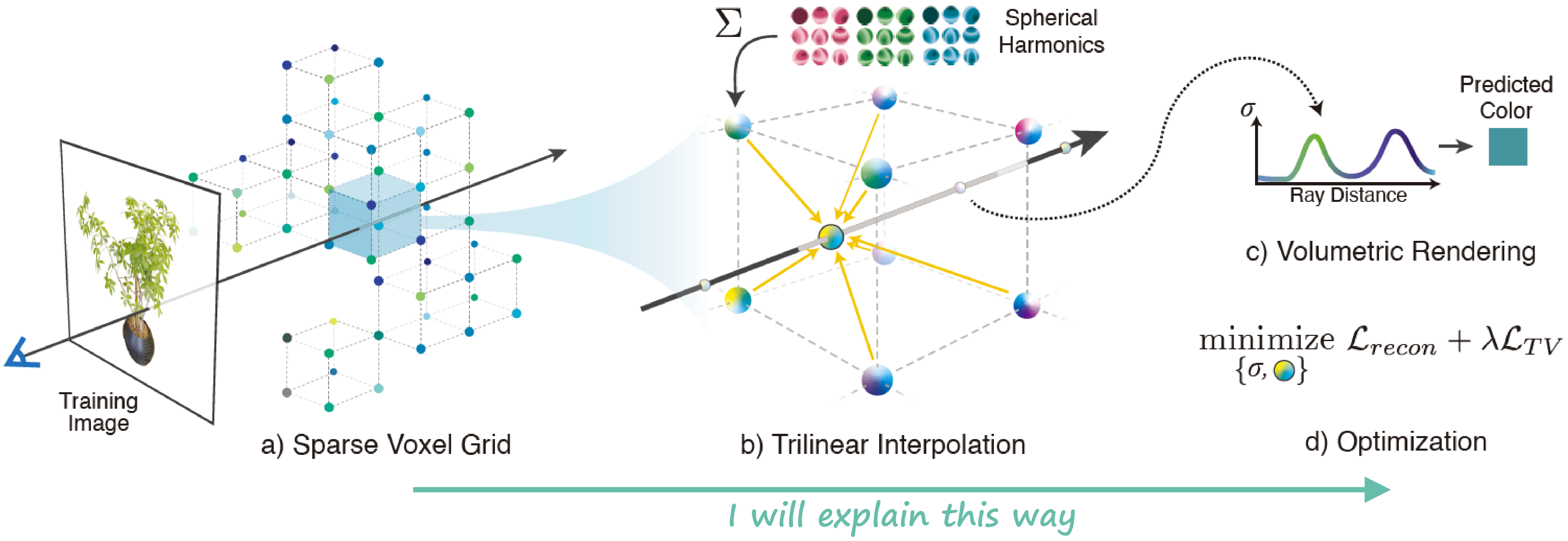


Voxel grid + interpolation + α

Contributions

- Train a radiance field from scratch, without neural networks, while maintaining NeRF quality and reducing optimization time by two orders of magnitude
- An explicit volumetric representation, based on a view-dependent sparse voxel grid without any neural networks
- Plenoptic volume elements named Plenoxel, which consists of a sparse voxel grid in which each voxel stores opacity and spherical harmonic coefficients

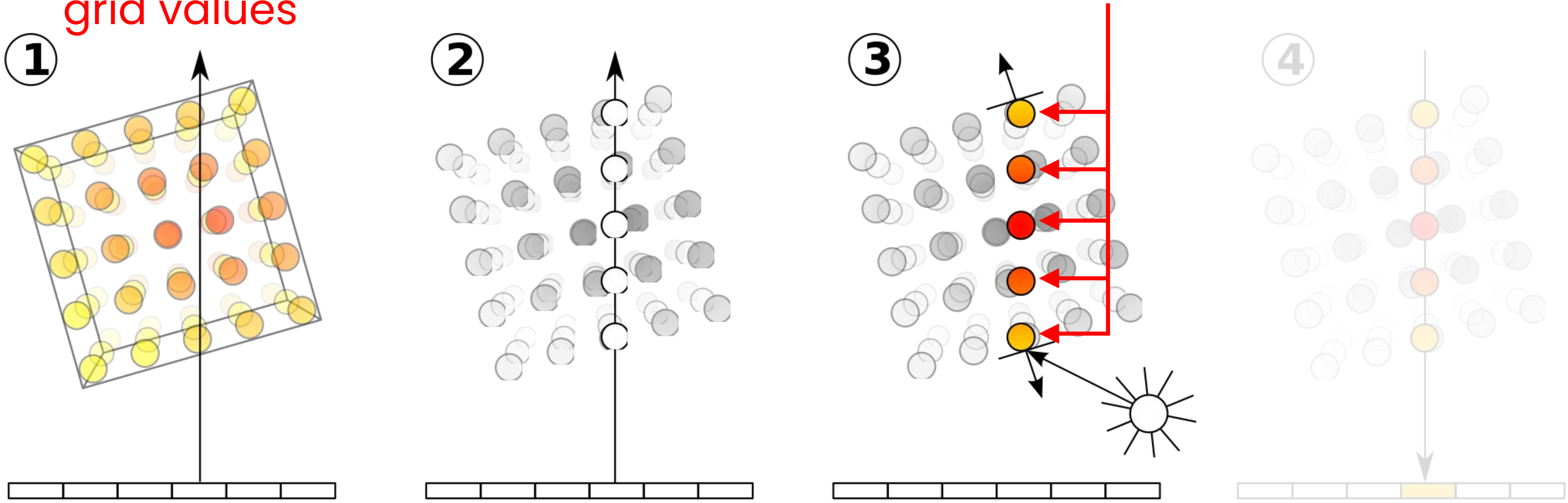
Plenoxels Overview



Without Neural Network?

NeRF: Color & density at these points → Neural network

Interpolate these grid values ← Other ways?



Ray casting

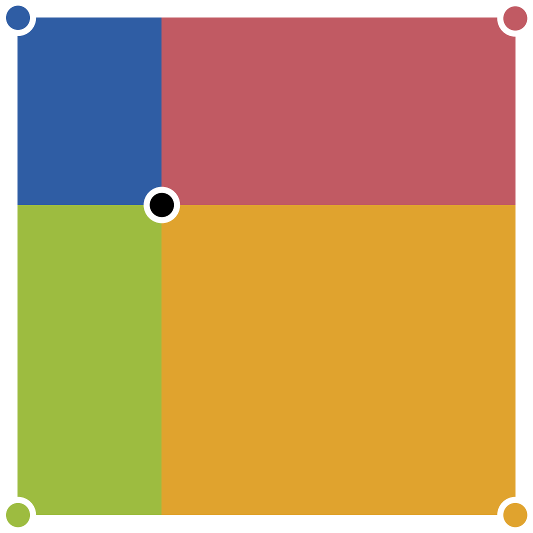
Sampling

Shading
(Color & density)

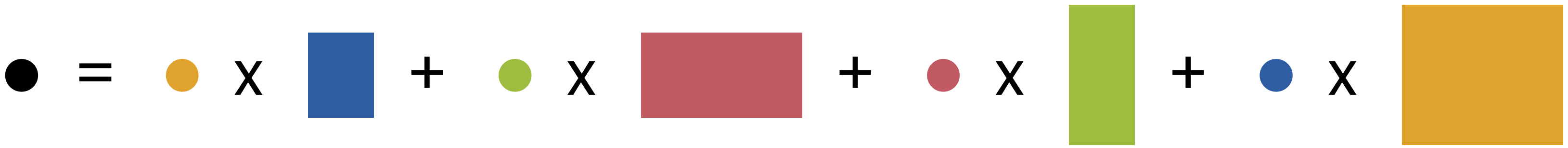
Compositing

Recall: Bilinear Interpolation

How can we get intermediate color with given image grid?



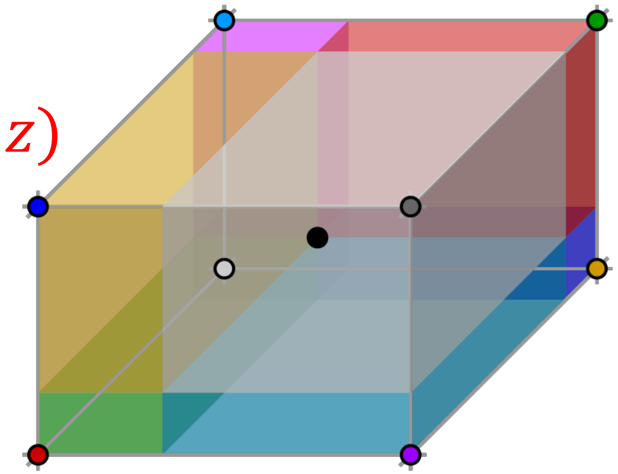
Interpolation (bilinear): super simple, super fast

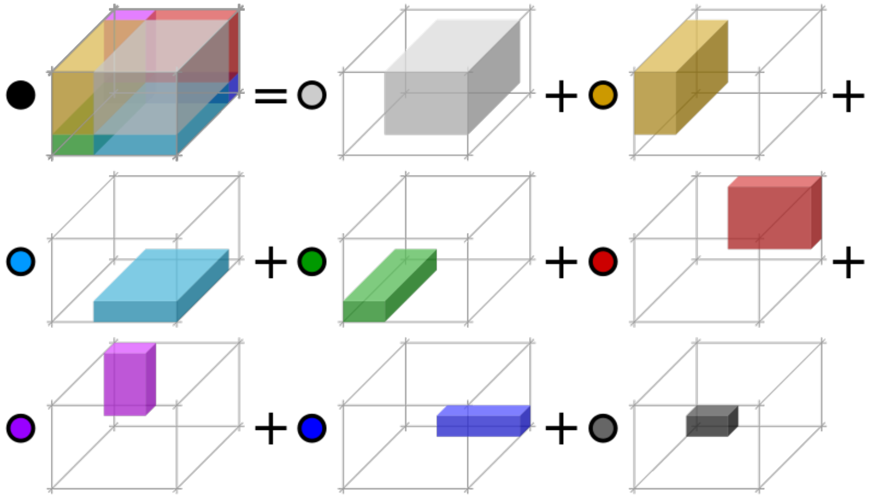


Voxel Grid Interpolation

So, can we do this in 3D? ➔ Yes, trilinear interpolation

Color = $f(x, y, z)$





Is it enough?

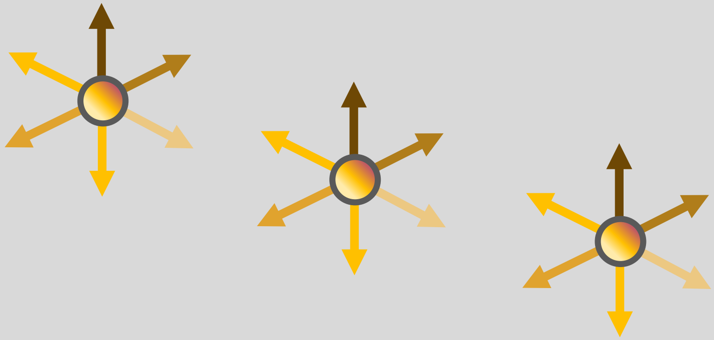
No! radiance fields are not just RGB color

If we do like this, we loose directional dependency

https://en.wikipedia.org/wiki/Trilinear_interpolation

Representing Radiance Field

Recall:
Neural radiance fields



5D function: (x, y, z, θ, ϕ)

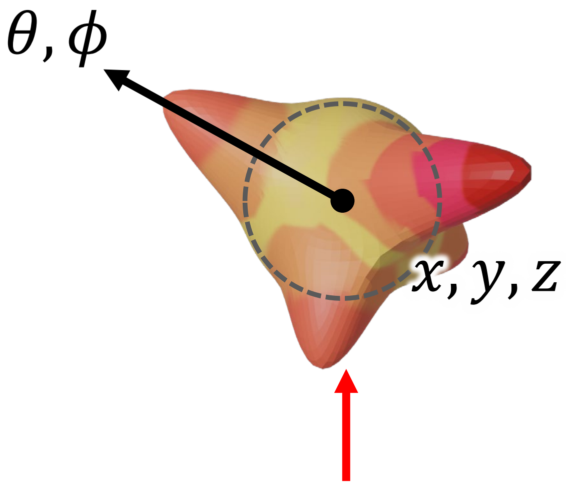
Returns out going radiance @ any 3D point, direction

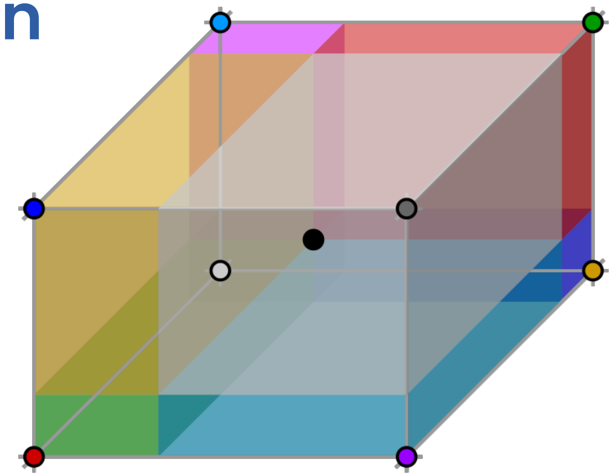
So, what we need is:

For given (x, y, z) and direction (θ, ϕ),
Returns radiance (RGB)

Plenoptic function

$L = f(x, y, z, \theta, \phi)$





How can we represent these kind of function in $\mathbb{R}^3$?

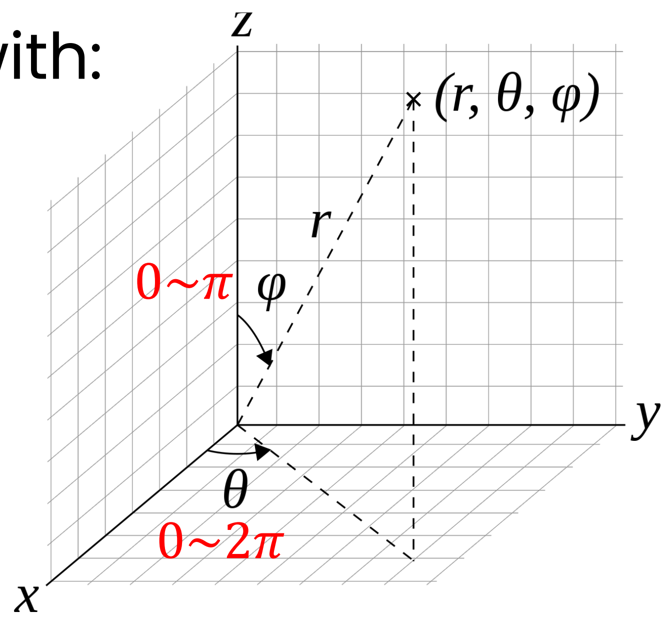
Representing Function in $\mathbb{R}^3$

We can represent any function on bounded interval (1D) with:

➔ $\sin(x), \cos(x)$ *Fourier series:* $a_n \cos(nx) + b_n \sin(nx)$

We can represent any function on unit sphere (3D) with:

➔ Spherical harmonics



Orthonormal basis function of solution from solving Laplace's equation on the sphere

$$Y_{\ell m} = \begin{cases} (-1)^m \sqrt{2} \sqrt{\dfrac{2\ell+1}{4\pi} \dfrac{(\ell-|m|)!}{(\ell+|m|)!}} P_\ell^{|m|}(\cos\theta)\ \sin(|m|\varphi) & \text{if } m < 0 \\ \sqrt{\dfrac{2\ell+1}{4\pi}} P_\ell^m(\cos\theta) & \text{if } m = 0 \\ (-1)^m \sqrt{2} \sqrt{\dfrac{2\ell+1}{4\pi} \dfrac{(\ell-m)!}{(\ell+m)!}} P_\ell^m(\cos\theta)\ \cos(m\varphi) & \text{if } m > 0 \end{cases}$$

What ???????

https://en.wikipedia.org/wiki/Spherical_coordinate_system

Spherical Harmonics

Just for understanding: sin, cos like basis function in 3D

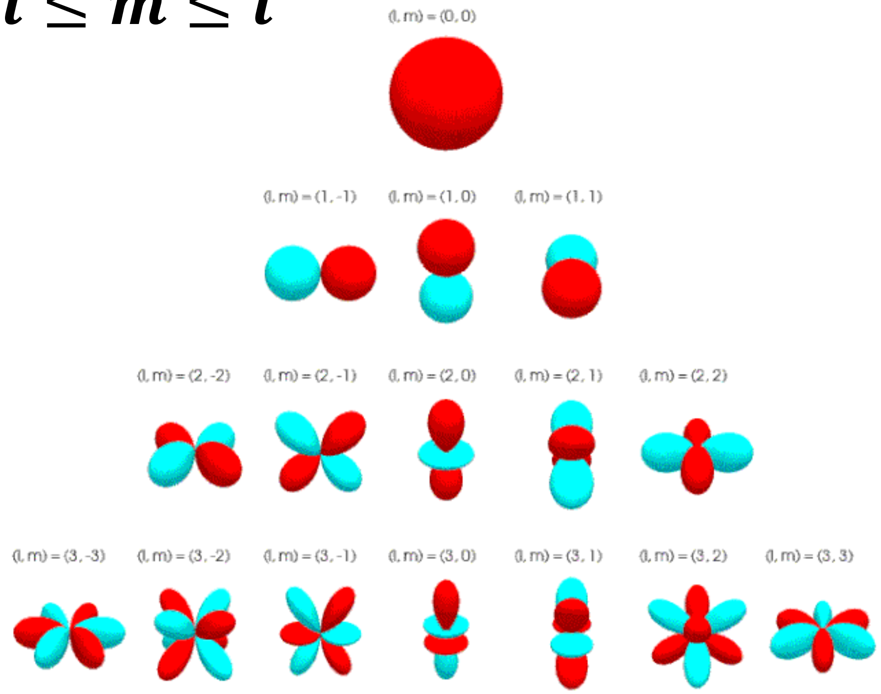
$l \in \mathbb{Z}, -l \leq m \leq l$

$l = 0$

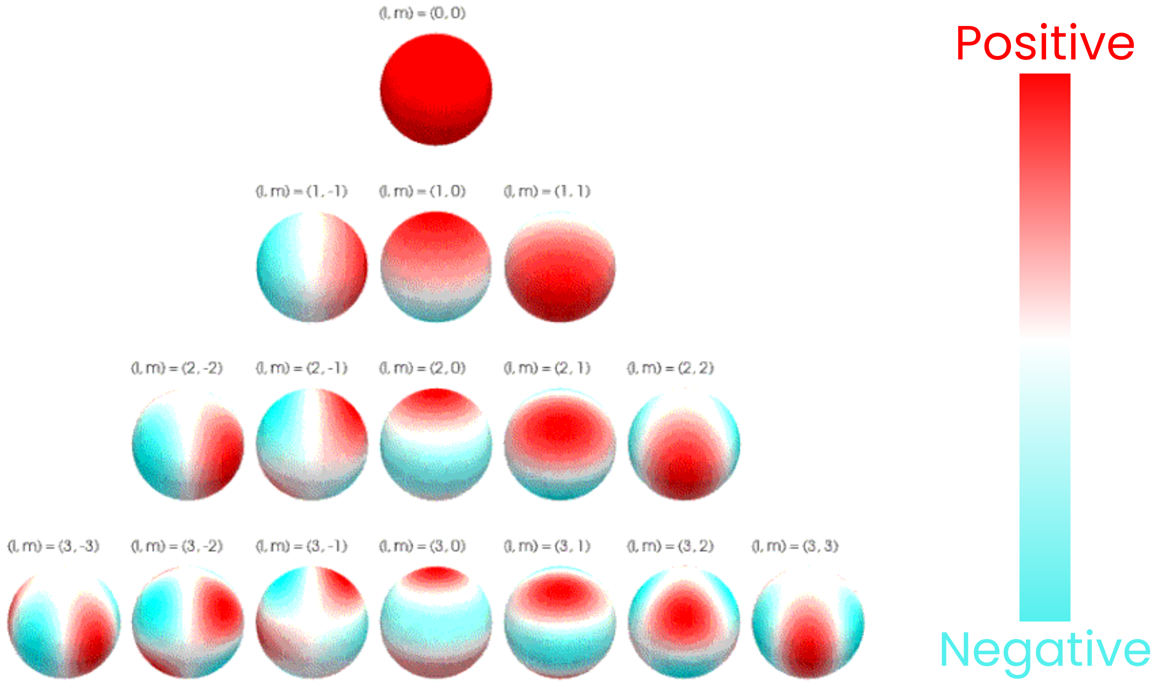
$l = 1$

$l = 2$

$l = 3$



Visualized by radius



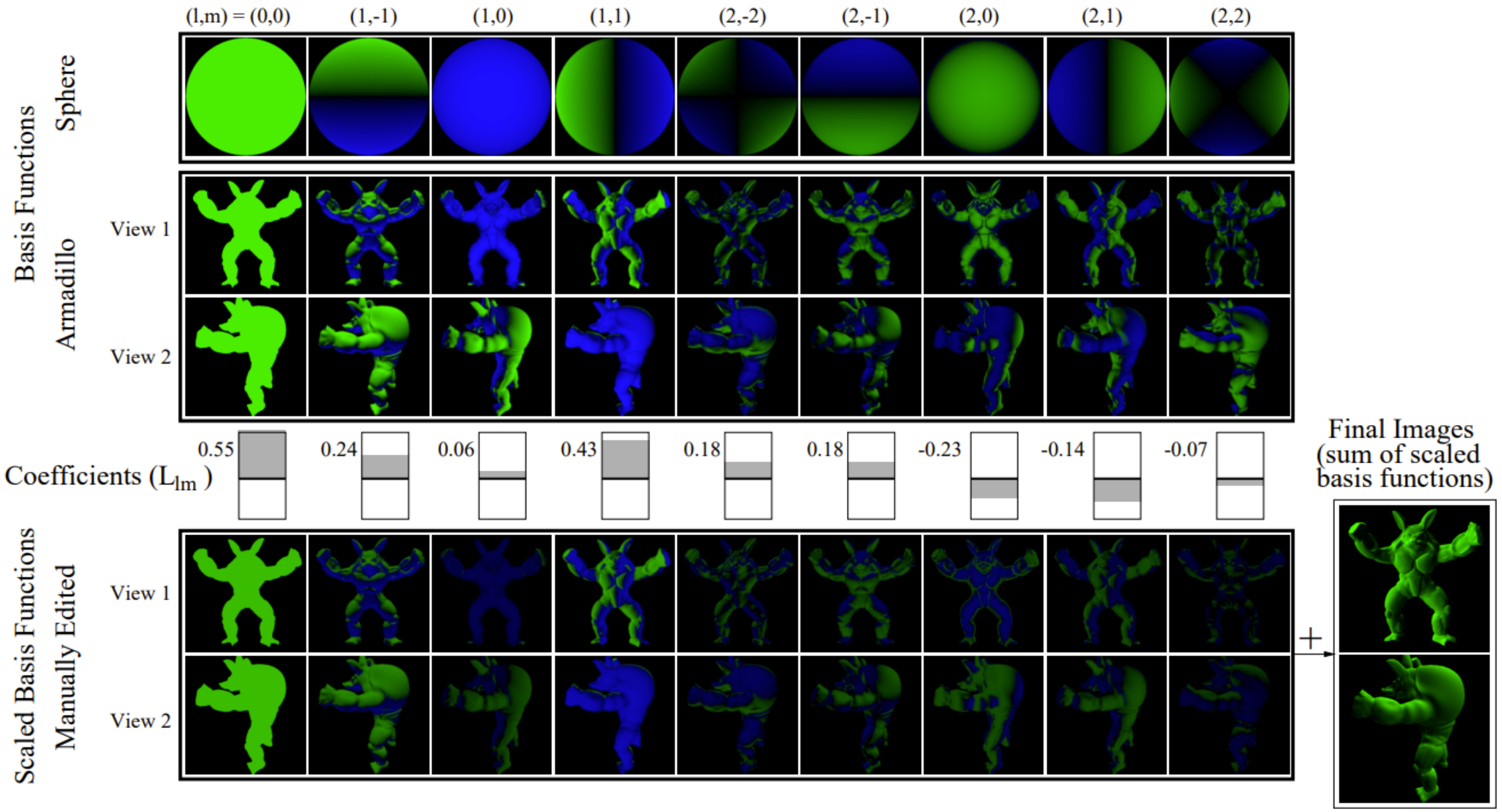
Visualized by color

→ *Seen before? (recall Chemistry 101)*

https://en.wikipedia.org/wiki/Table_of_spherical_harmonics#Visualization_of_Real_Spherical_Harmonics

Spherical Harmonics + Computer Graphics

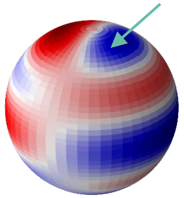
Many function on sphere (hemisphere) can be represented!



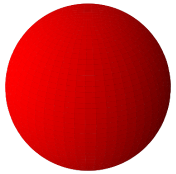
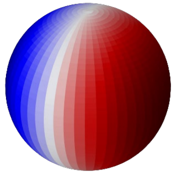
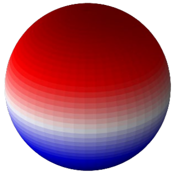
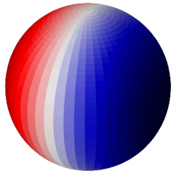
An Efficient Representation for Irradiance Environment Maps
[Ramamoorthi and Hanrahan, SIGGRAPH 2001]

Plenoxels

Note that blue color here is for visualization
There is no negative radiance



$$L(x,y,z) = a_{0,0}\,Y_{0,0} + a_{1,-1}\,Y_{1,-1} + a_{1,0}\,Y_{1,0} + a_{1,1}\,Y_{1,1} + \cdots$$

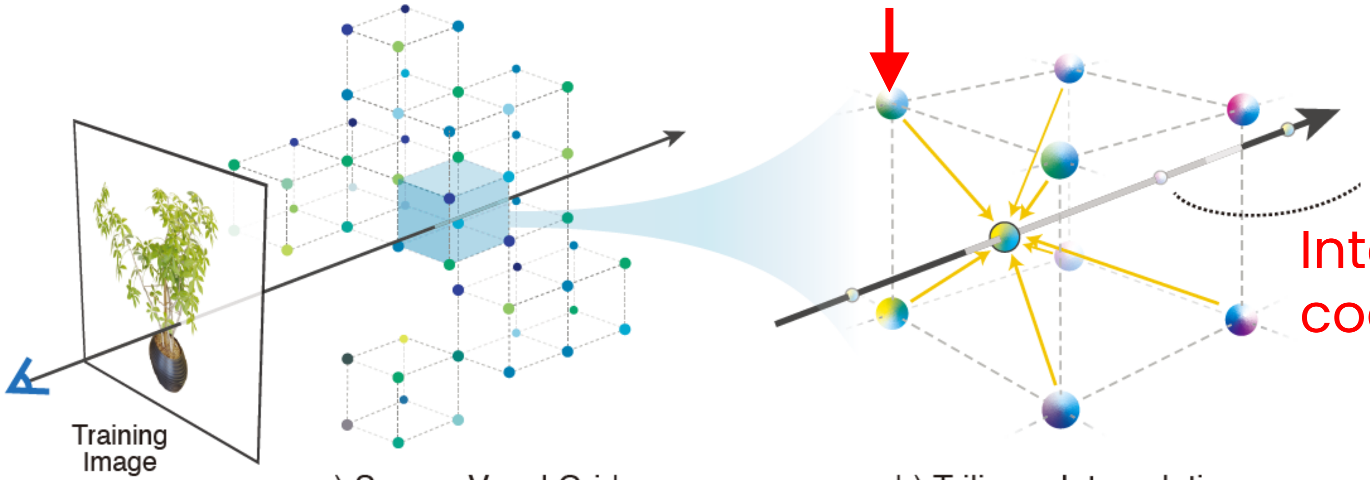
   

Radiance field

➔ “Plenoxel” (Plenoptic function + Voxel)

Coefficients are stored: $a_{0,0}\ \ a_{1,-1}\ \ a_{1,0}\ \ a_{1,1} \cdots$

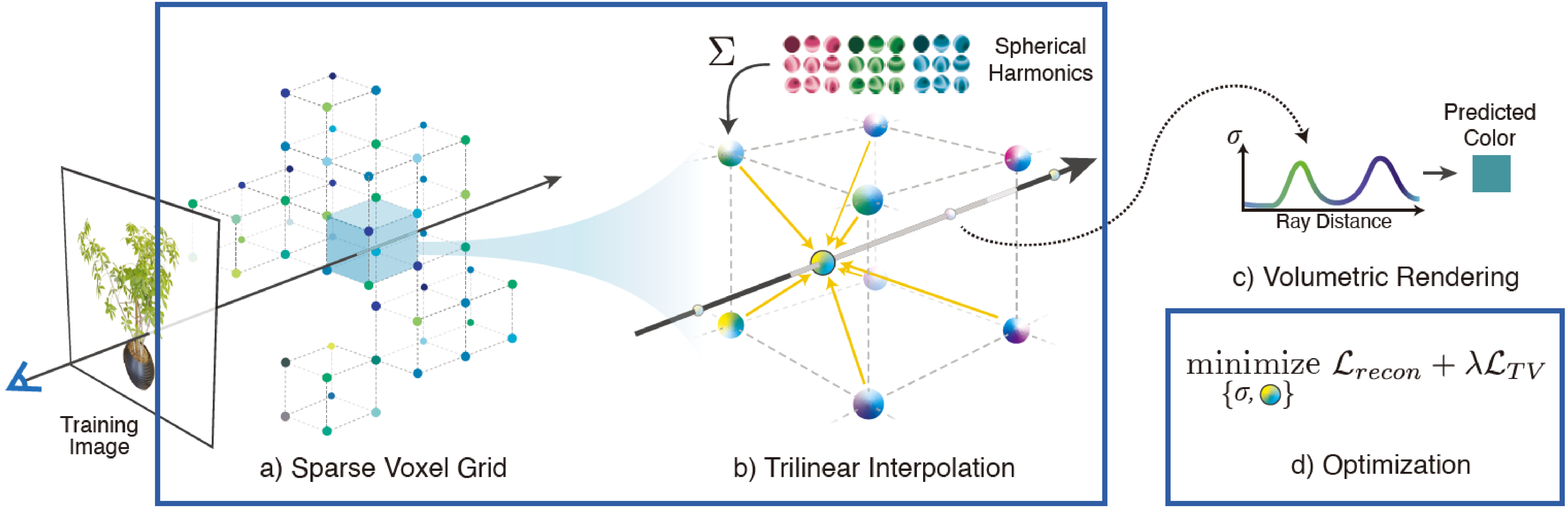
9 coefficient / 1 color channel



a) Sparse Voxel Grid b) Trilinear Interpolation

Interpolating coefficients are enough!

How About Loss Functions?



Simplification of NeRF

Same as NeRF?

➔ No, we need more regularization

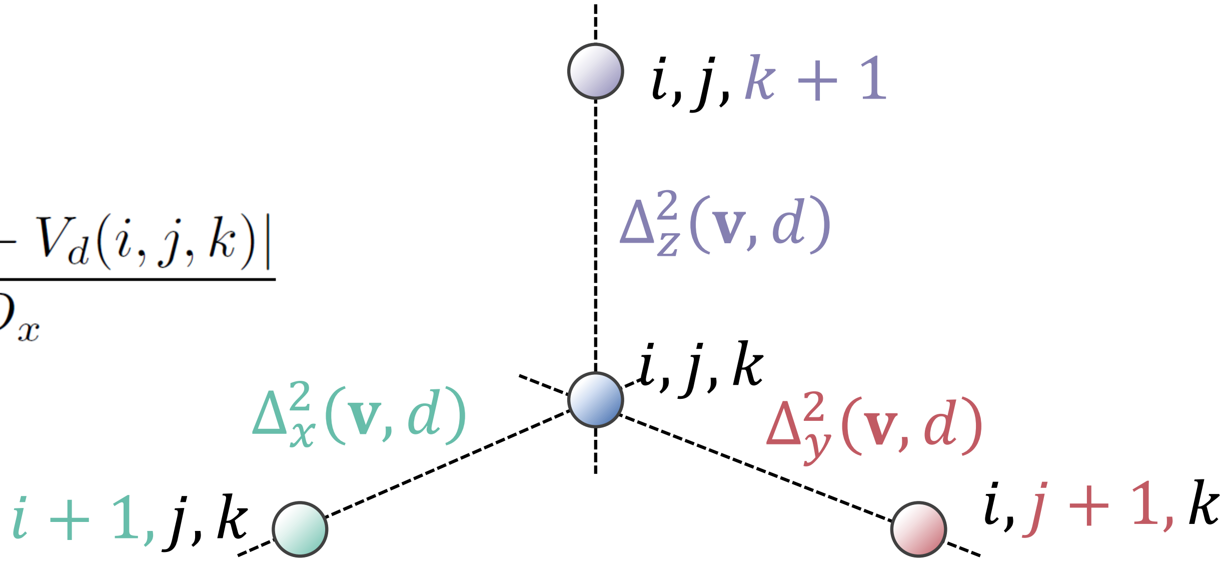
Total Variation Loss

$$\underset{\{\sigma, \bullet\}}{\text{minimize}} \ \mathcal{L}_{recon} + \lambda \mathcal{L}_{TV}$$

$$\mathcal{L}_{TV} = \frac{1}{|\mathcal{V}|} \sum_{\substack{\mathbf{v} \in \mathcal{V} \\ d \in [D]}} \sqrt{\Delta_x^2(\mathbf{v}, d) + \Delta_y^2(\mathbf{v}, d) + \Delta_z^2(\mathbf{v}, d)}$$

$$\Delta_x((i, j, k), d) = \frac{|V_d(i+1, j, k) - V_d(i, j, k)|}{256 / D_x}$$

D_x: voxel grid resolution



Other $+\alpha$

Sparsity prior (real scenes)

→ Encourage voxels to be empty

$$\mathcal{L}_s = \lambda_s \sum_{i,k} \log\left(1 + 2\sigma(\mathbf{r}_i(t_k))^2\right)$$

Opacity

Beta-distribution regularizer (real 360 scenes)

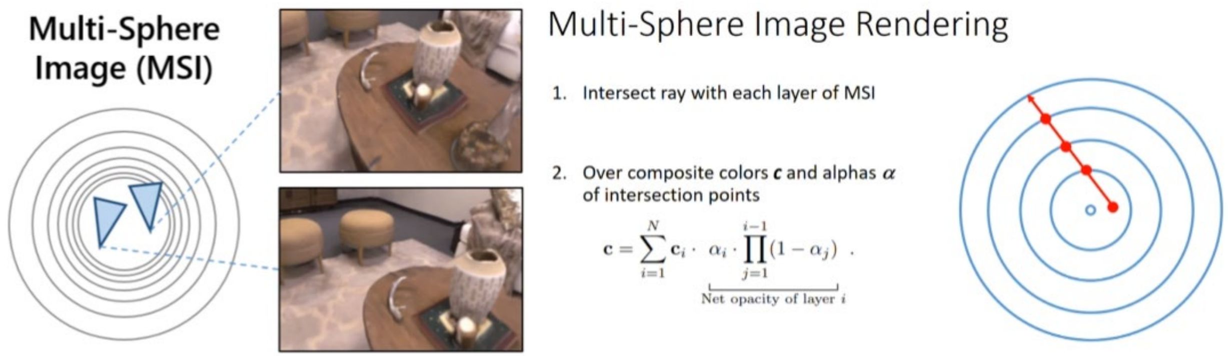
→ Foreground should be either fully opaque or empty

$$\mathcal{L}_\beta = \lambda_\beta \sum_{\mathbf{r}} \left(\log(T_{FG}(\mathbf{r})) + \log(1 - T_{FG}(\mathbf{r}))\right)$$

Accumulated transmittance

Multi-sphere image (real 360 scenes)

→ Voxels are warped to sphere



Attal et al., ECCV 2020, MatryODShka: Real-time 6DoF Video View Synthesis using Multi-Sphere Images

Results

https://alexyu.net/plenoxels/

Conclusion

- Less train time
- Straightforward (Trilinear interpolation of voxels)
- Volume rendering is key part of NeRF

Limitations

- Suffers from artifacts
- Hard to find optimal weight of loss terms
- Scalability (Mip-NeRF)

Ground Truth JAXNeRF [7, 26] Plenoxels

$$\underset{\{\sigma, \bullet\}}{\text{minimize}} \; \mathcal{L}_{recon} + \lambda \mathcal{L}_{TV}$$

Closing Remarks

Quiz

1. Neural radiance field is the function that takes ( ) dimensional input and returns color (RGB) and density.

2. Any function on the unit sphere can be represented as linear combination of ( ).

Take Home Messages

NeRF

1. How ➔ Neural network + volume rendering
2. Radiance ➔ Simple MLP
3. Positional encoding ➔ High frequency detail

Plenoxels

1. Improve speed
2. Plenoxels = Plenoptic function + voxel
3. Spherical harmonics = sin/cos function on unit the sphere
4. Radiance ➔ Trilinear interpolation of spherical harmonics coefficient
5. Additional loss terms for regularization