Lecture 22:
Neural Radiance Fields (NeRFs)

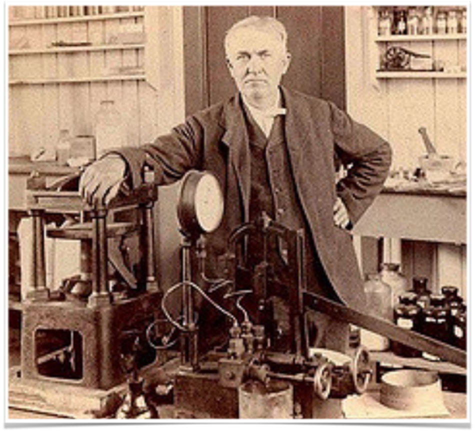
COMP 590/776: Computer Vision

Instructor: Soumyadip (Roni) Sengupta

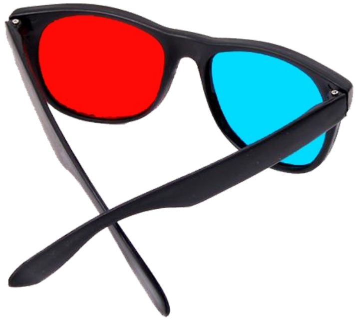
TA: Mykhailo (Misha) Shvets

Course Website:
Scan Me!

Stereo Photography



Viewing Devices




Left
MAZZER


Right
MAZZER


MAZZER

NeRF (Neural Radiance Field) has revolutionized Computer Vision & Graphics in past 3 years!

Let’s look at some of the stunning results it produced!

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

ECCV 2020

Ben Mildenhall*

UC Berkeley

Pratul Srinivasan*

UC Berkeley

Matt Tancik*

UC Berkeley

Jon Barron

Google Research

Ravi Ramamoorthi

UC San Diego

Ren Ng

UC Berkeley

Given a set of sparse views of an object with known camera poses

Optimize a NeRF model

3D reconstruction viewable from any angle

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis,
Ben Mildenhall*, Pratul Srinivasan*, Matthew Tancik*, Jonathan Barron, Ravi Ramamoorthi, Ren Ng, ECCV 2020.

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis,
Ben Mildenhall*, Pratul Srinivasan*, Matthew Tancik*, Jonathan Barron, Ravi Ramamoorthi, Ren Ng, ECCV 2020.

Block-NeRF: Scalable Large Scene Neural View Synthesis, CVPR 2022.

(a) Capture Process (b) Input (c) Nerfie (d) Nerfie Depth

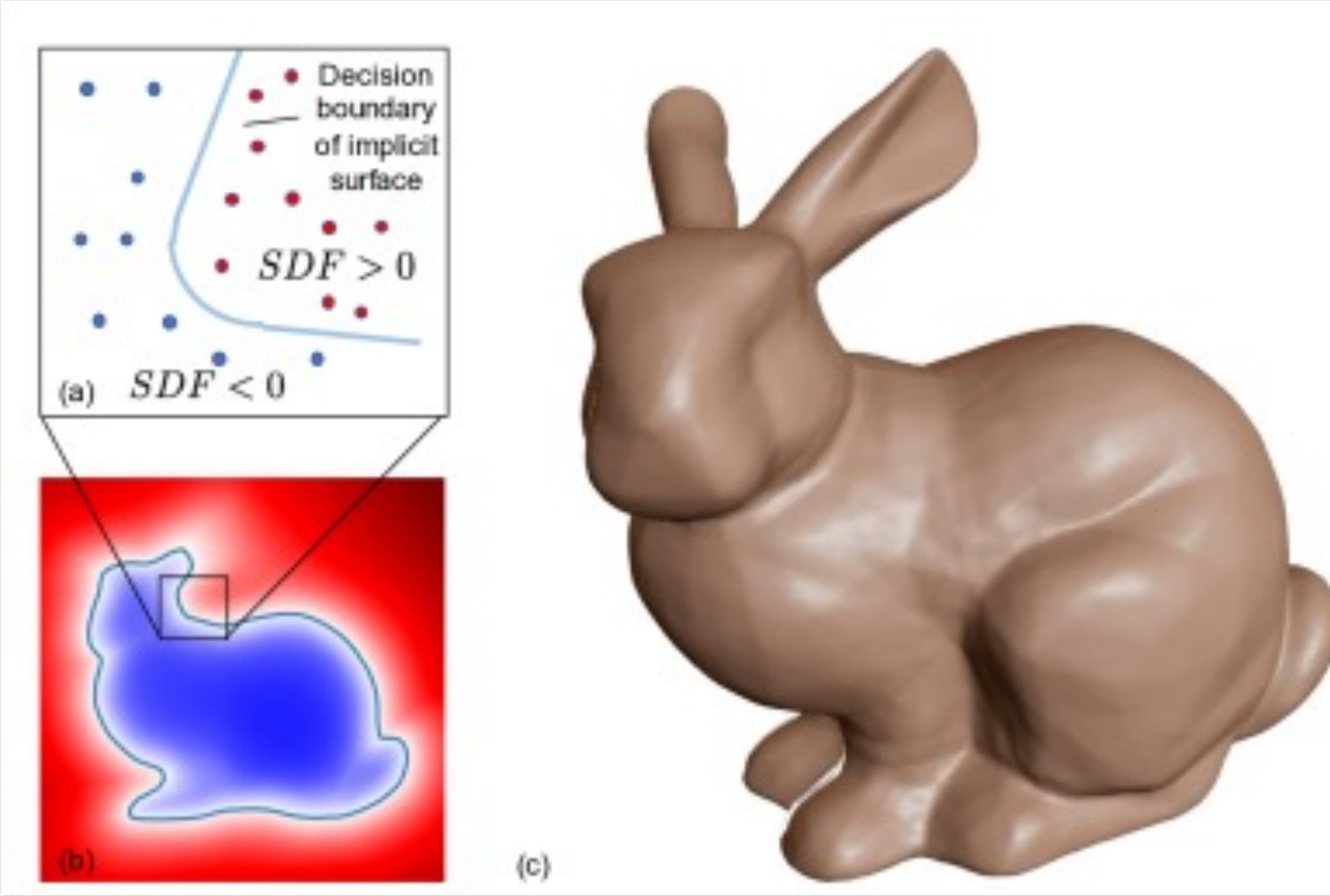
NeRFies: Deformable Neural Radiance Fields, Keunhong Park et al., ICCV 2021.

Neural 3D Video Synthesis
from Multi-view Video,
Li et al., CVPR 2022

Surface Representation:
Signed Distance Function (SDF)
- implicit representation via level set

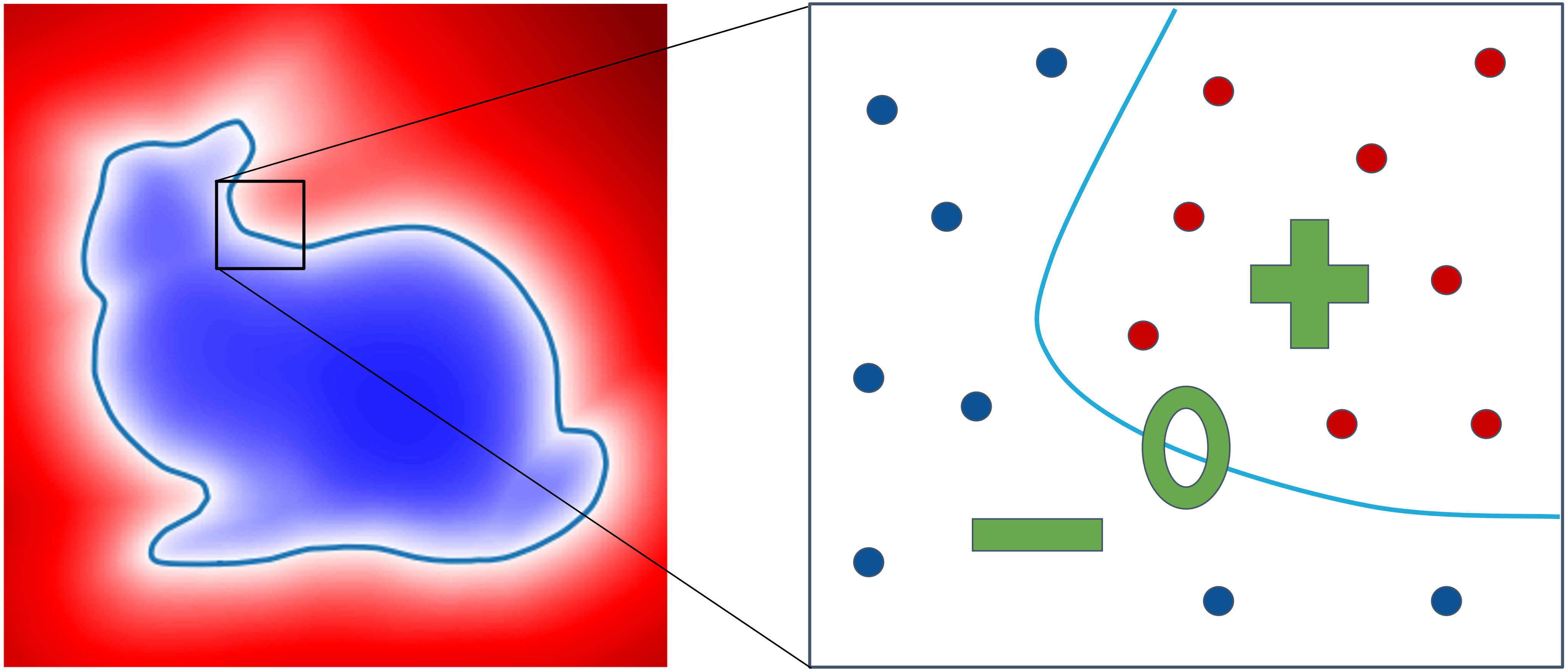
SDF(X) = 0, when X is on the surface.
SDF(X) > 0, when X is outside the surface
SDF(X) < 0, when X is inside the surface

Note: SDF is an implicit representation!
Suitable for neural networks but hard to import inside existing graphics software.

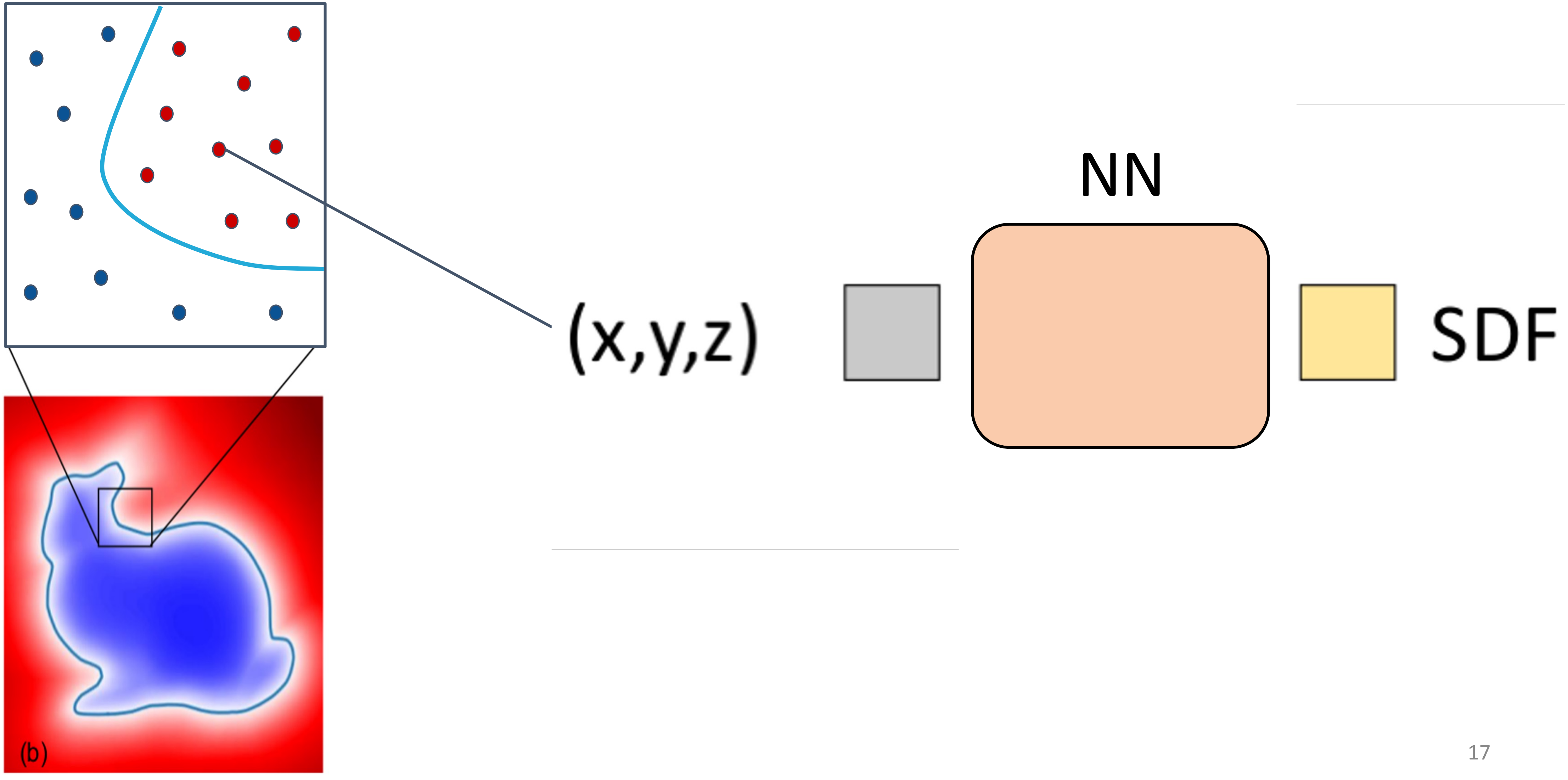


Deep SDF: Use a neural network (co-ordinate based MLP) to represent the SDF function.

Signed Distance Function

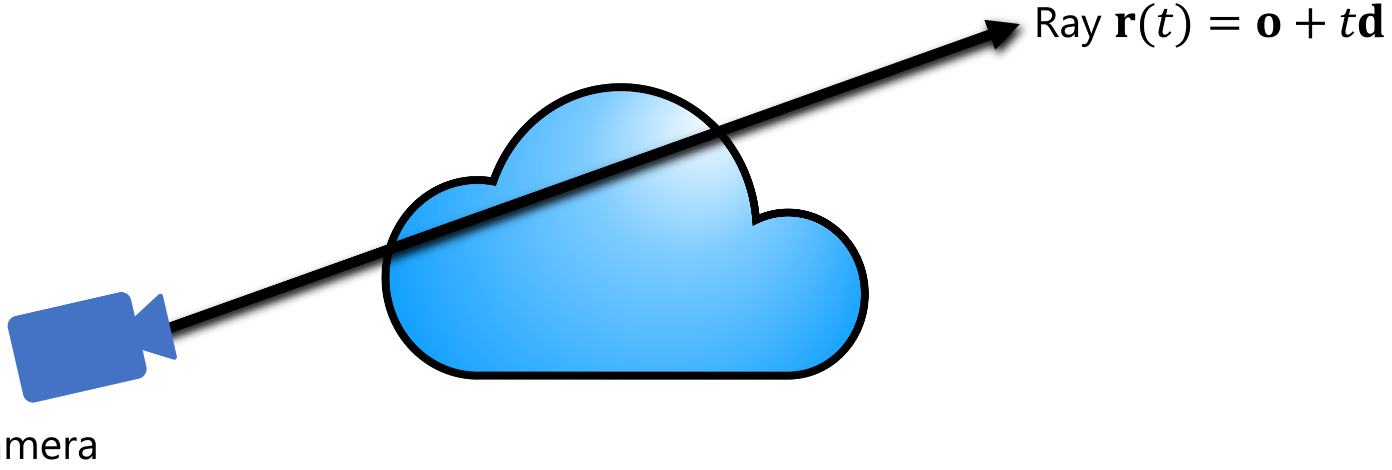


Regression of Continuous SDF

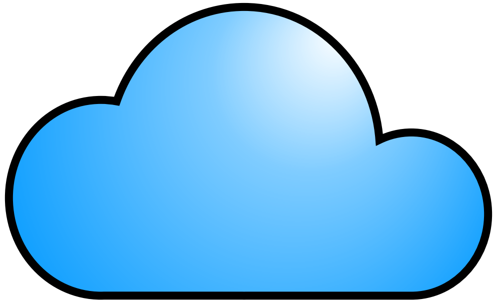


What is Volume Rendering?

- Assume a cloud of tiny colored particles in 3D. Each particle has a RGB color and a density.
- Take a pixel on image plane, and shoot a ray from the camera center, through the pixel and into the 'cloud of tiny colored particles'
- What should be the color for that pixel?



Volumetric formulation for NeRF

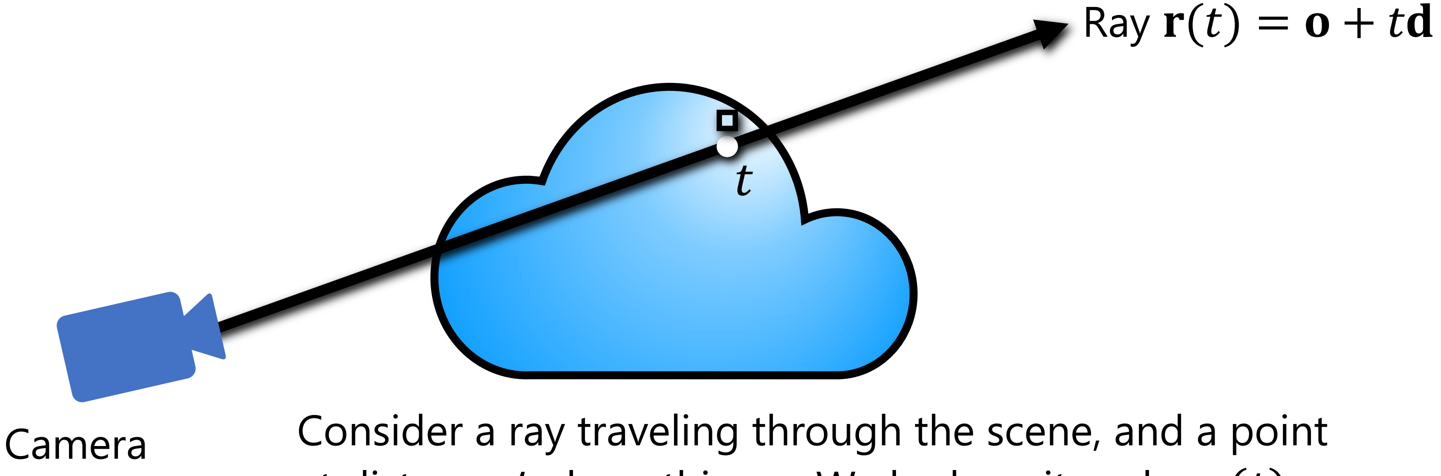


Scene is a cloud of colored fog

Max and Chen 2010, Local and Global Illumination in the Volume Rendering Integral

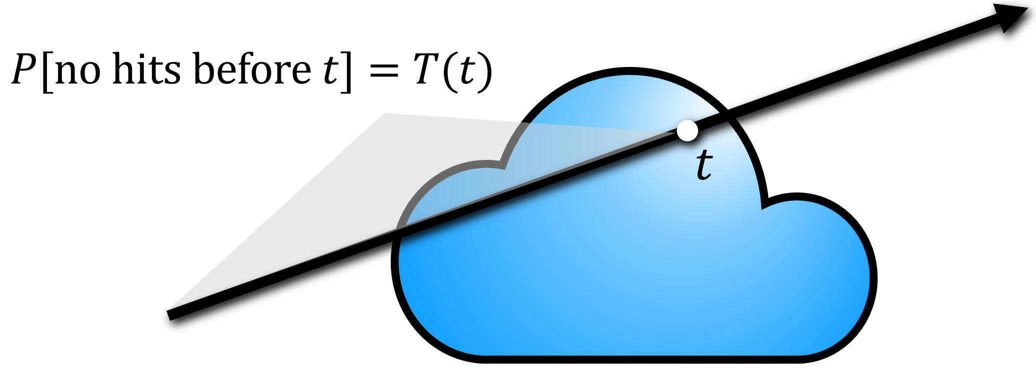
19

Volumetric formulation for NeRF



Consider a ray traveling through the scene, and a point at distance t along this ray. We look up its color $\mathbf{c}(t)$, and its opacity (alpha value) $\alpha(t)$

Volumetric formulation for NeRF



But t may also be blocked by earlier points along the ray. $T(t)$: probability that the ray didn't hit any particles earlier.
$T(t)$ is called "transmittance"

Volume rendering estimation: integrating color along a ray

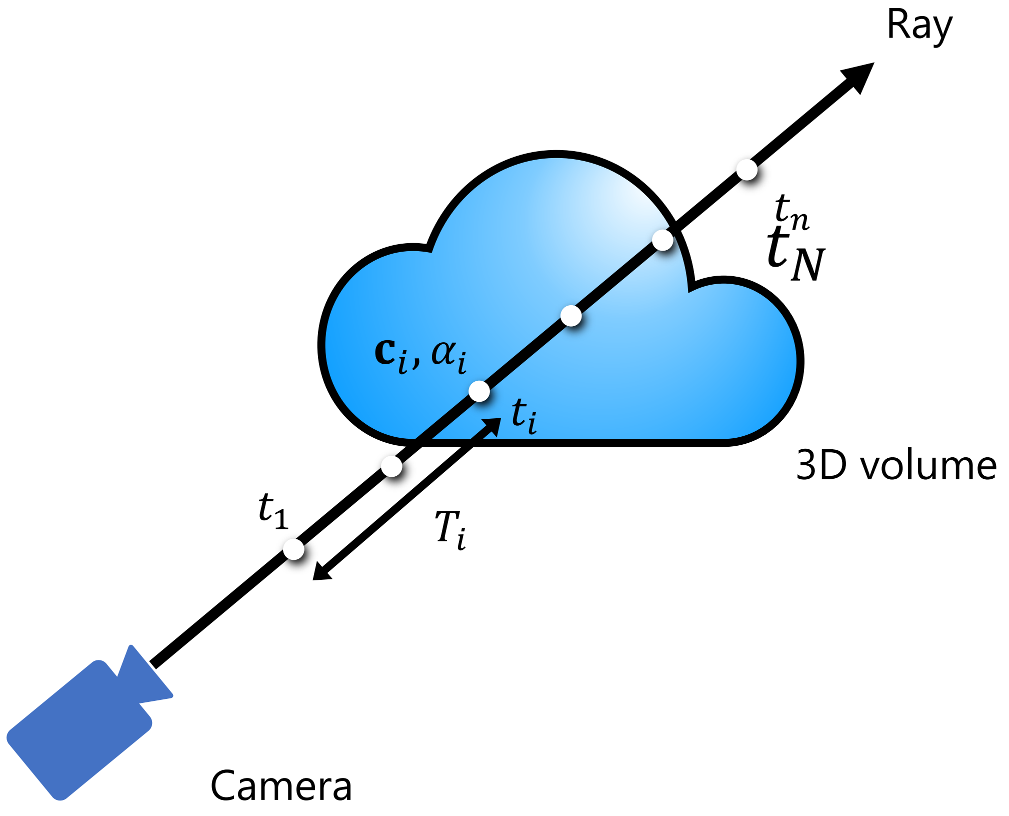
Rendering model for ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

$$\mathbf{c} \approx \sum_{i=1}^{n} T_i \alpha_i \mathbf{c}_i$$

final rendered color along ray

weights

colors



How much light is blocked earlier along ray:

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

Computing the color for a set of rays through the pixels of an image yields a rendered image

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i)$$

Slight modification: α is not directly stored in the volume, but instead is derived from a stored volume density sigma (σ) that is multiplied by the distance between samples delta (δ):

Volume rendering estimation: integrating color along a ray

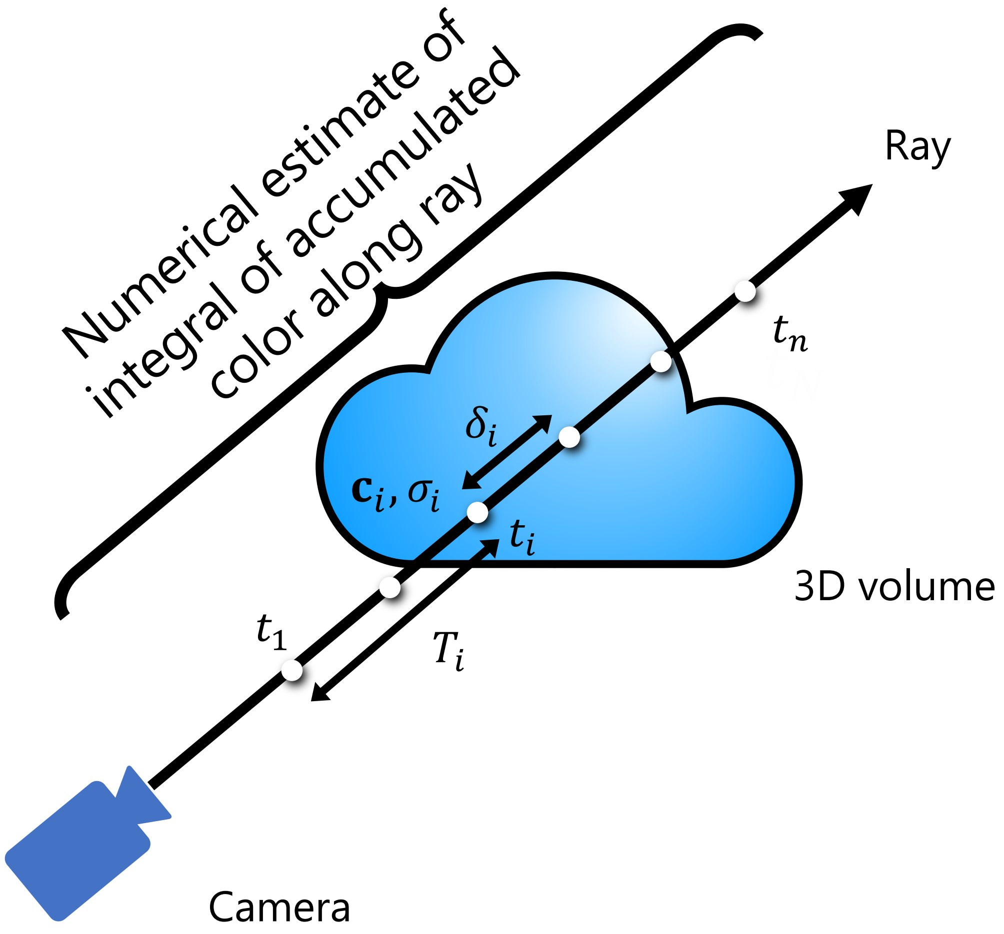
Rendering model for ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

$$\mathbf{c} \approx \sum_{i=1}^{n} T_i \alpha_i \mathbf{c}_i$$

final rendered color along ray

weights

colors



How much light is blocked earlier along ray:

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

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i)$$

Volume rendering estimation: integrating color along a ray

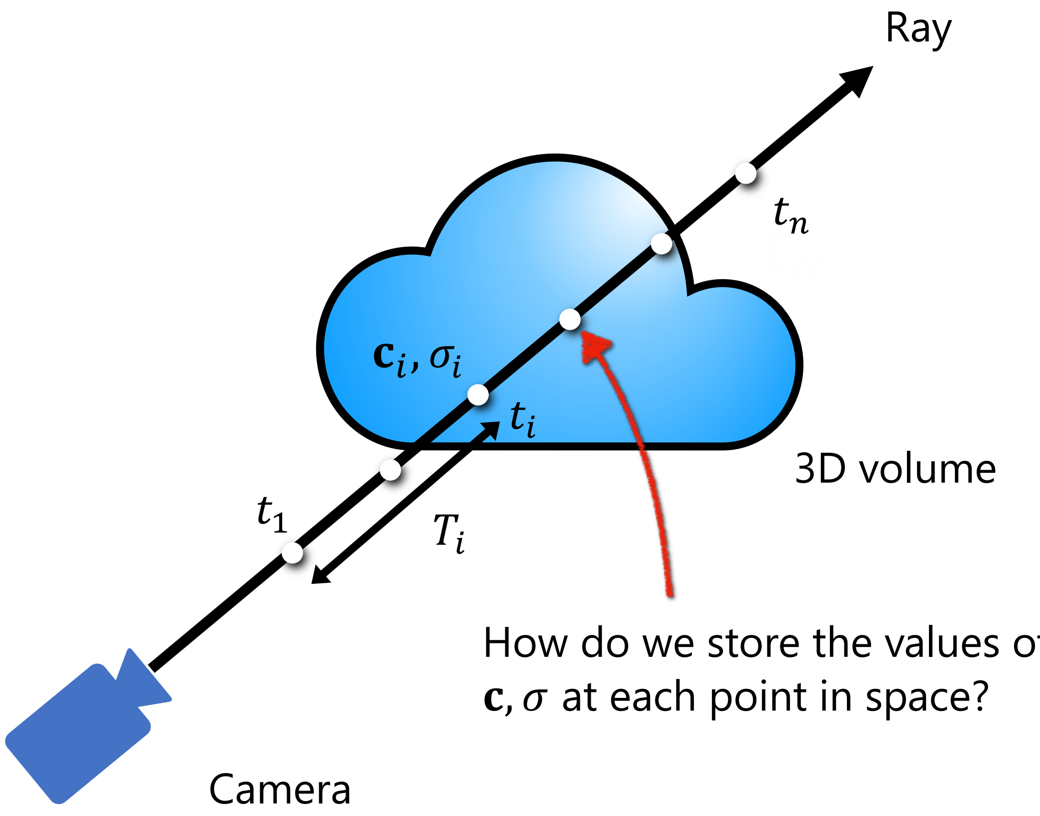
Rendering model for ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

$$\mathbf{c} \approx \sum_{i=1}^{n} T_i \alpha_i \mathbf{c}_i$$

final rendered color along ray

weights

colors

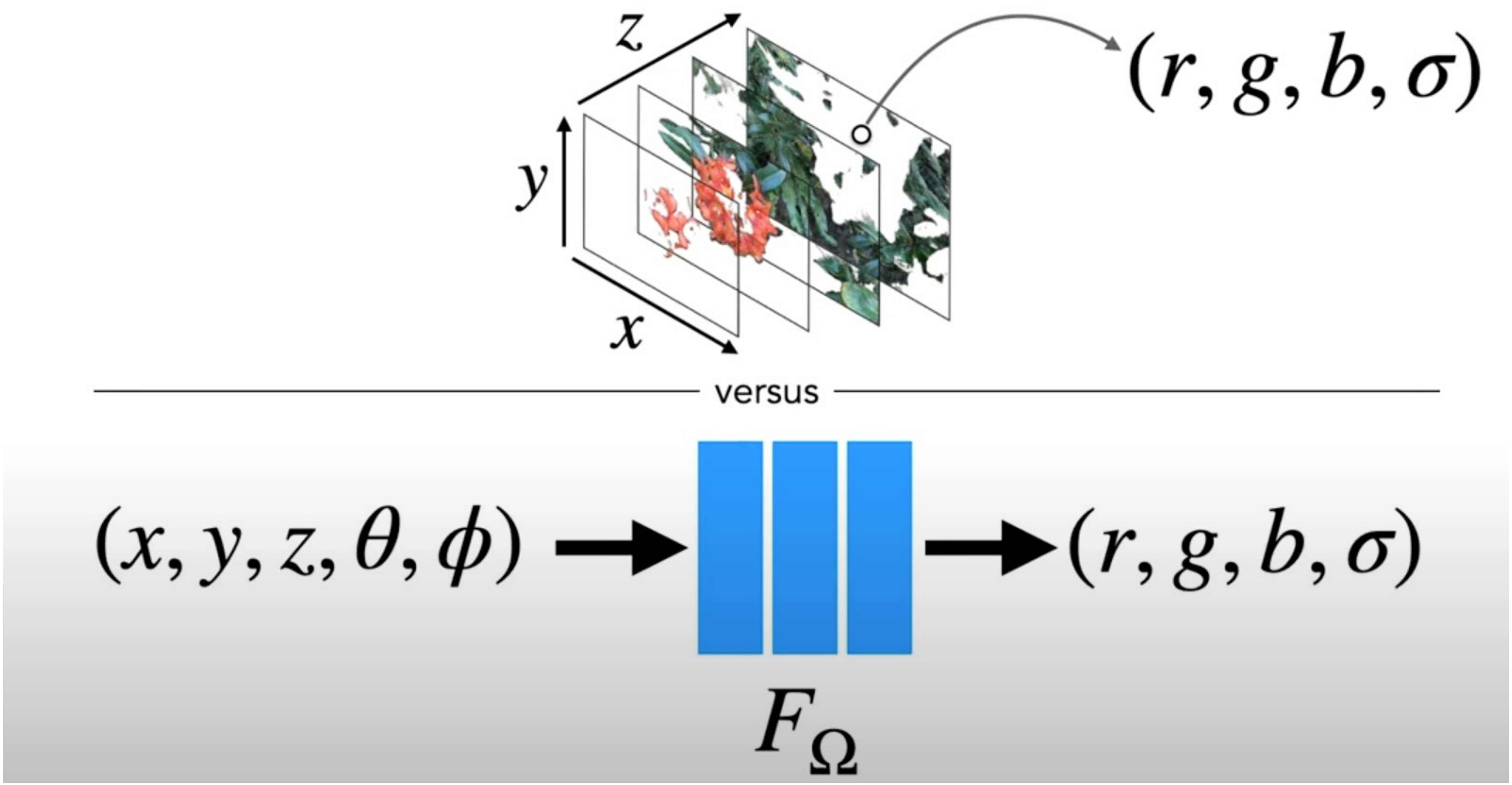


How do we store the values of $\mathbf{c}, \sigma$ at each point in space?

How much light is blocked earlier along ray:

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

$$\alpha_i = 1 - \exp(-\sigma_i \delta_i)$$



z
y
x
(r, g, b, σ)
versus
(x, y, z, θ, ϕ)
(r, g, b, σ)
F_{Ω}

Toy problem: storing 2D image data

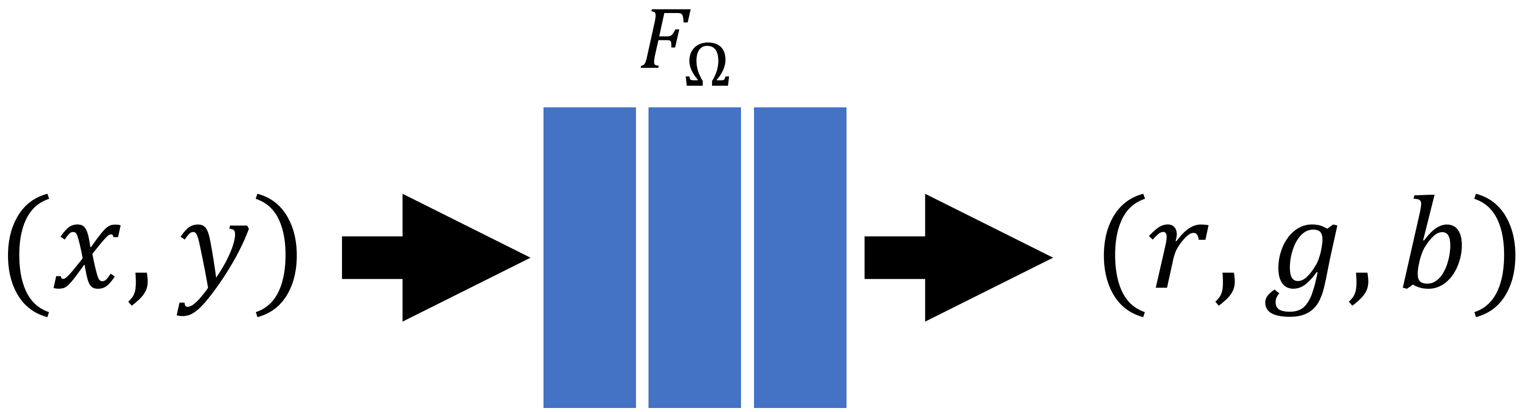
(x, y)

(r, g, b)

Usually we store an image as a
2D grid of RGB color values

26

Toy problem: storing 2D image data



What if we train a simple fully-connected network (MLP) to do this instead?

27

Naive approach fails!

Ground truth image

Neural network output fit
with gradient descent

Problem:

- “Standard” coordinate-based MLPs cannot represent high frequency functions.

Solution:

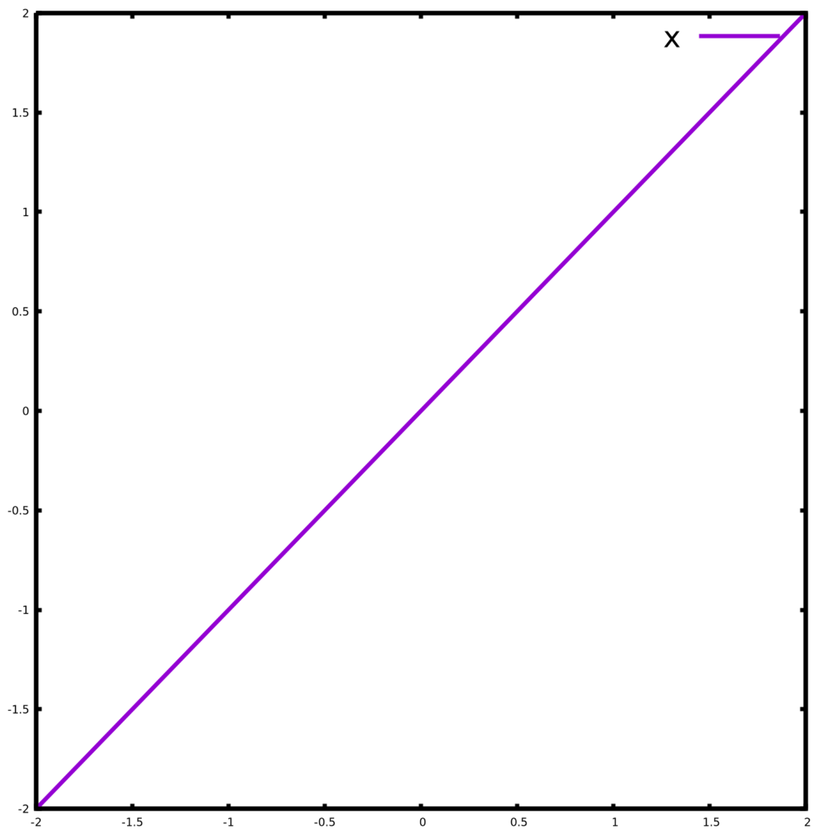
- Pass input coordinates through a high frequency mapping first.

29

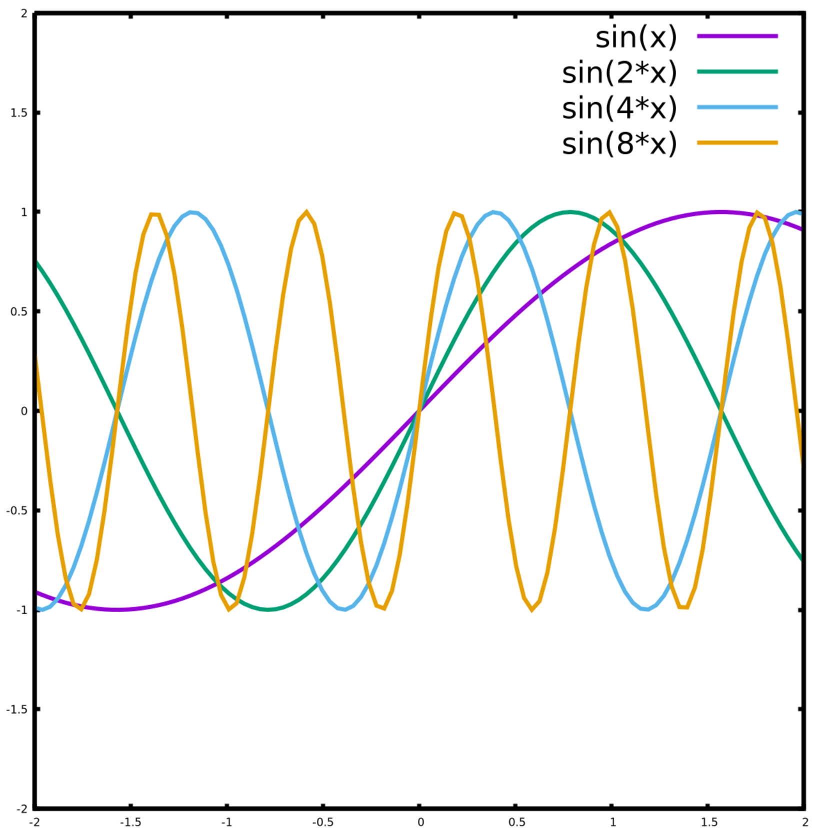
Example mapping: “positional encoding”

$$\left(\begin{array}{c} \sin(\mathbf{v}), \cos(\mathbf{v}) \\ \sin(2\mathbf{v}), \cos(2\mathbf{v}) \\ \sin(4\mathbf{v}), \cos(4\mathbf{v}) \\ \dots \\ \sin(2^{L-1}\mathbf{v}), \cos(2^{L-1}\mathbf{v}) \end{array}\right)$$

Positional encoding



Raw encoding of a number ***x***

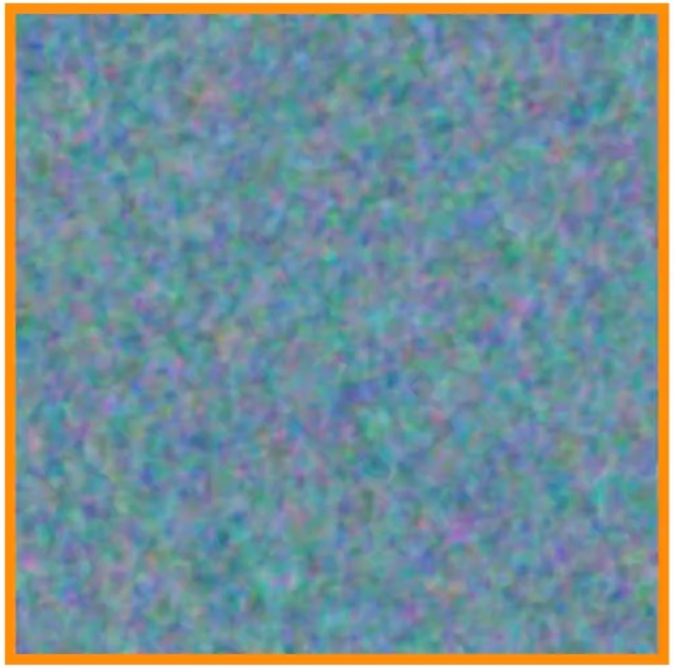


“Positional encoding” of a number ***x***

Problem solved!

Ground truth image

Neural network output without high frequency mapping



Neural network output with high frequency mapping

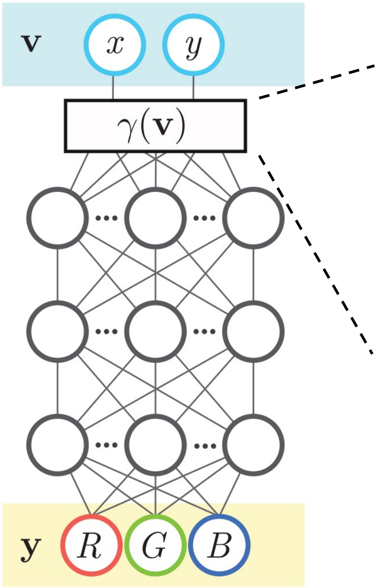
32

Network Architecture: Overcoming Spectral Bias

[Baatz et al. 2021]

The signals we want are high frequency!

Network Architecture: Input Encodings



Random Fourier Encodings

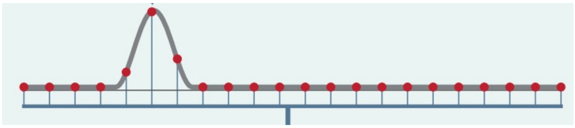
[Tancik et al. 2020]

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

Non-axis aligned sine embeddings

One-blob Encodings

[Müller et al. 2020]



Super Gaussian Encodings

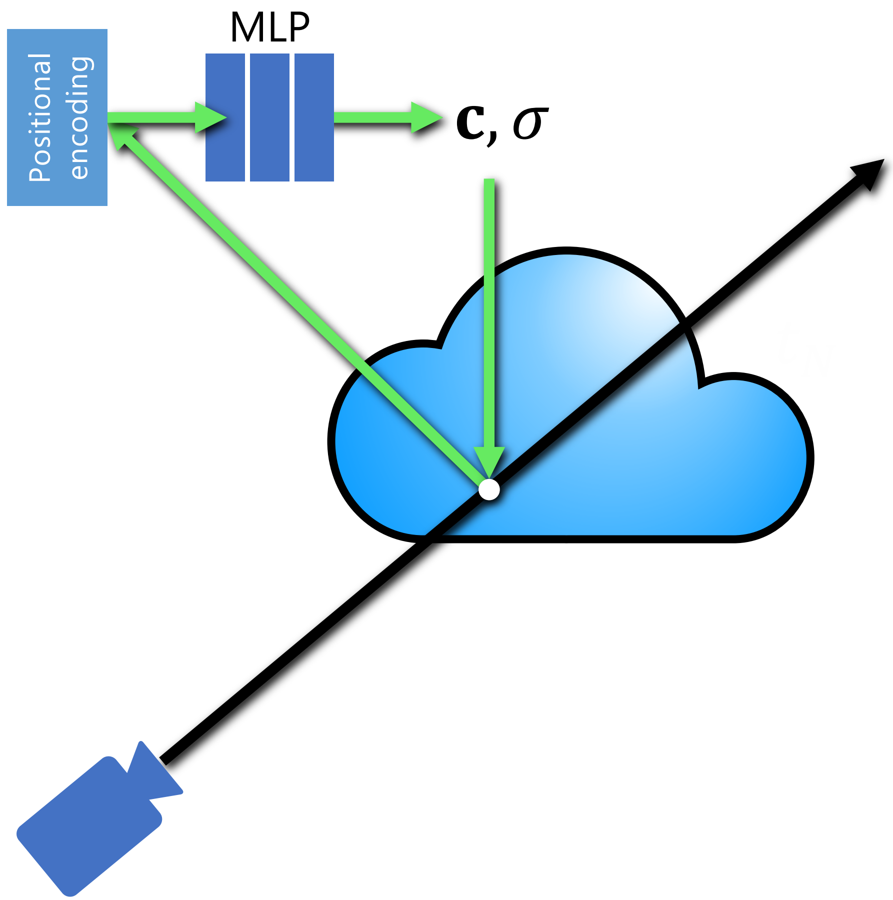
[Ramasinghe et al. 2021]

$$\Phi(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \ldots, \phi_D(\boldsymbol{x})]^T,$$

$$\left[e^{-\frac{(\boldsymbol{x}\cdot\alpha - t_i)^2}{2\sigma_{\boldsymbol{x}}^2}}\right]^b$$

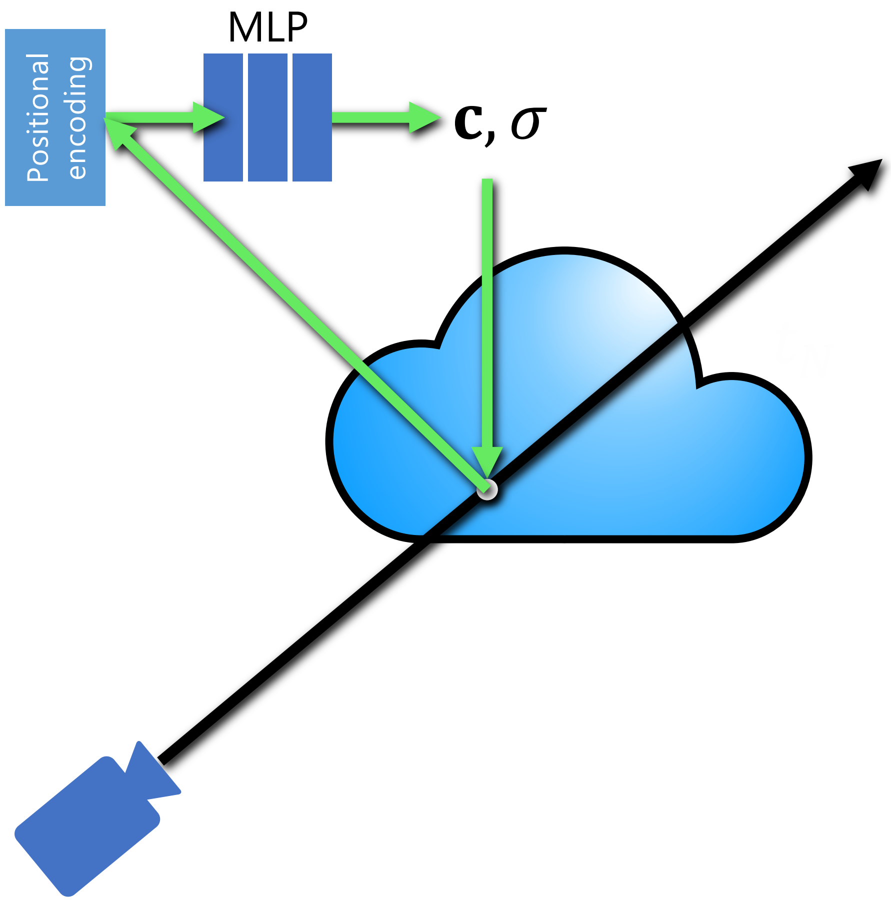
Gaussian embeddings

NeRF = volume rendering + coordinate-based network

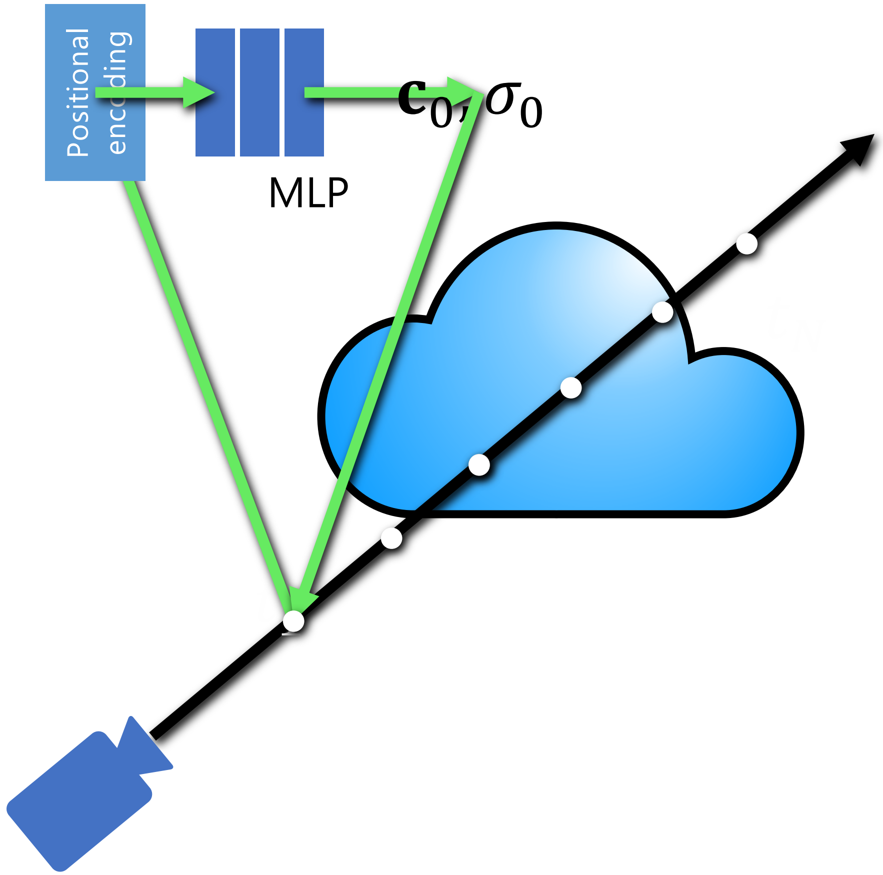


$\mathbf{c}, \sigma$

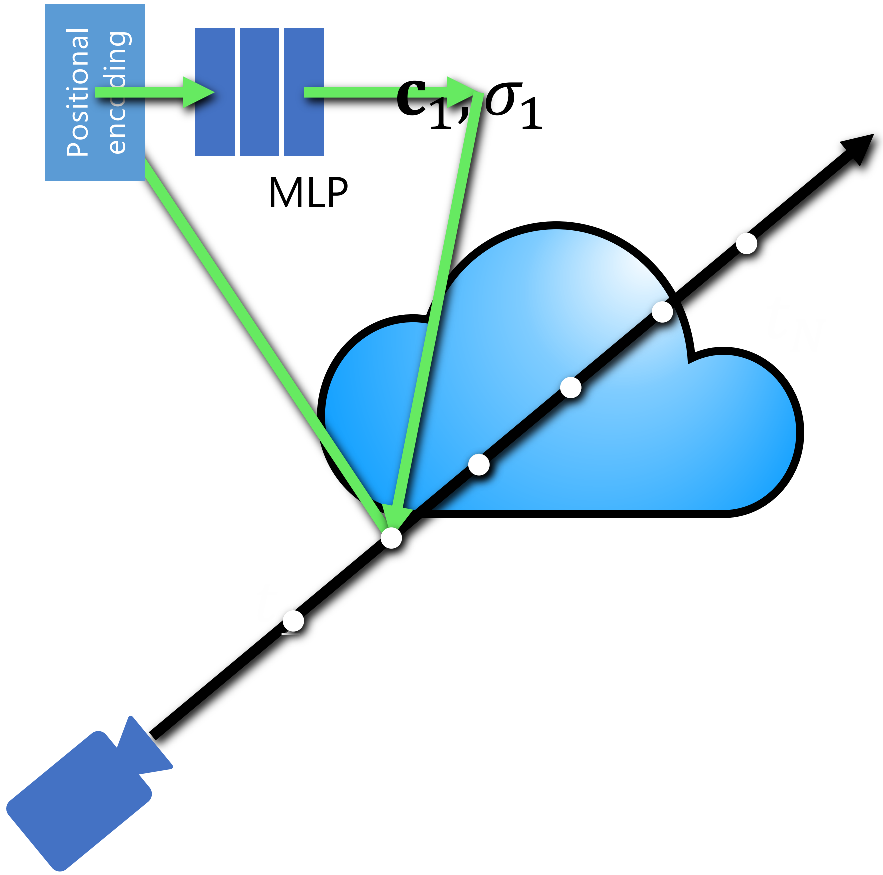
$\mathbf{c}, \sigma$



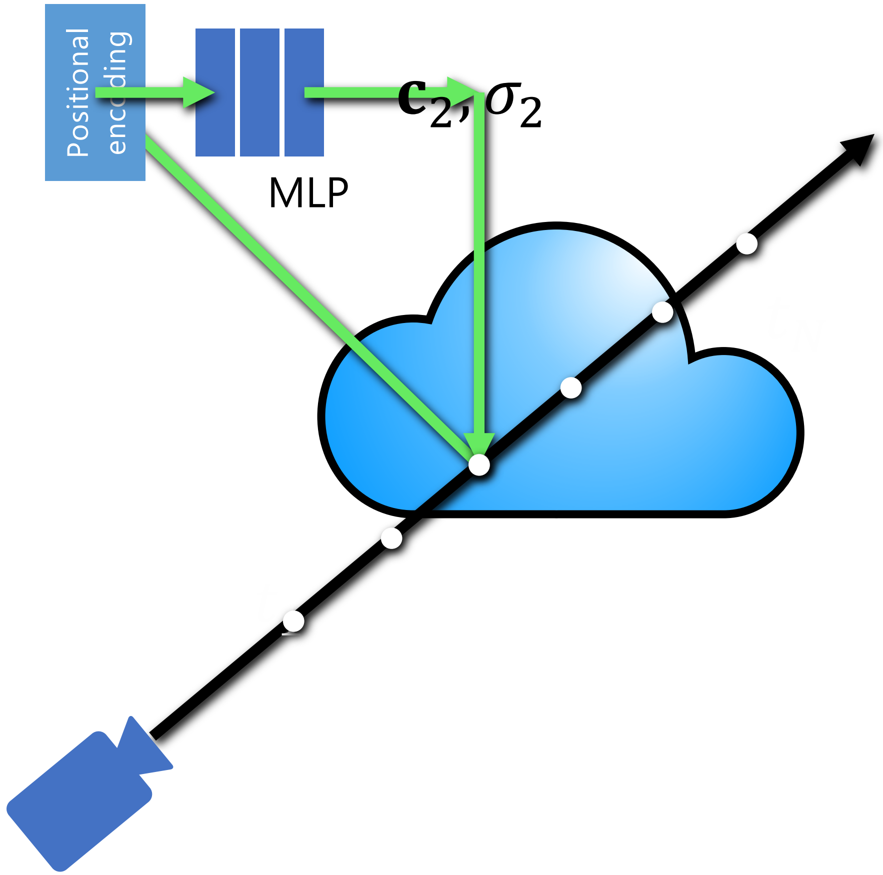
$\mathbf{c}, \sigma$



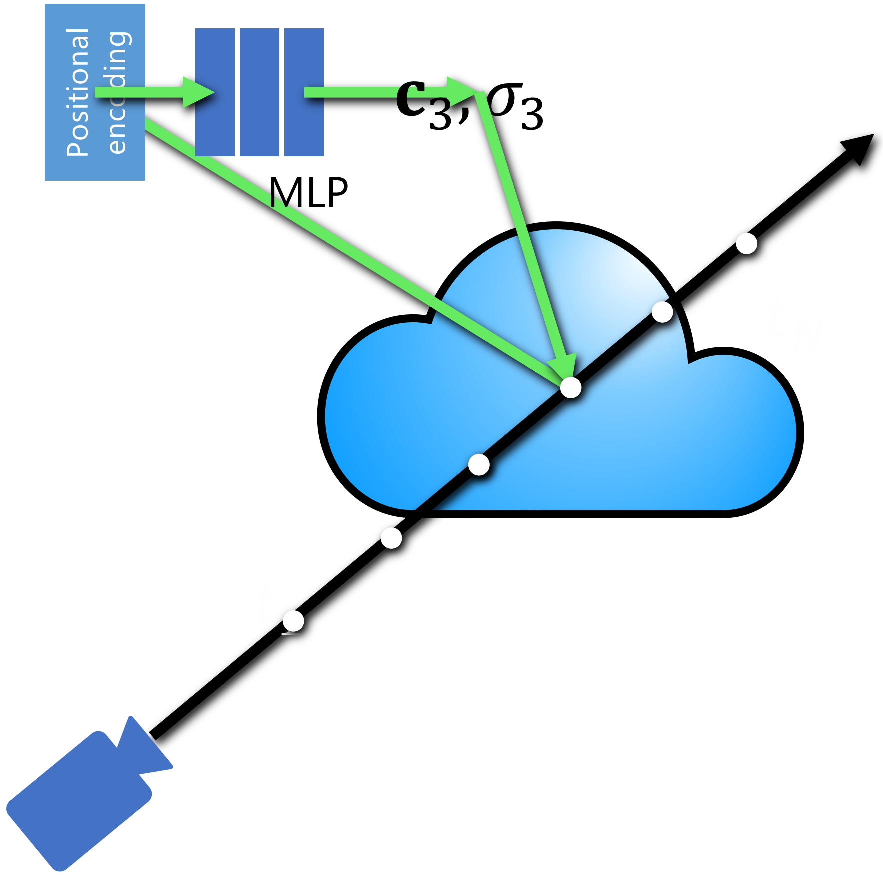
$\mathbf{c}, \sigma$



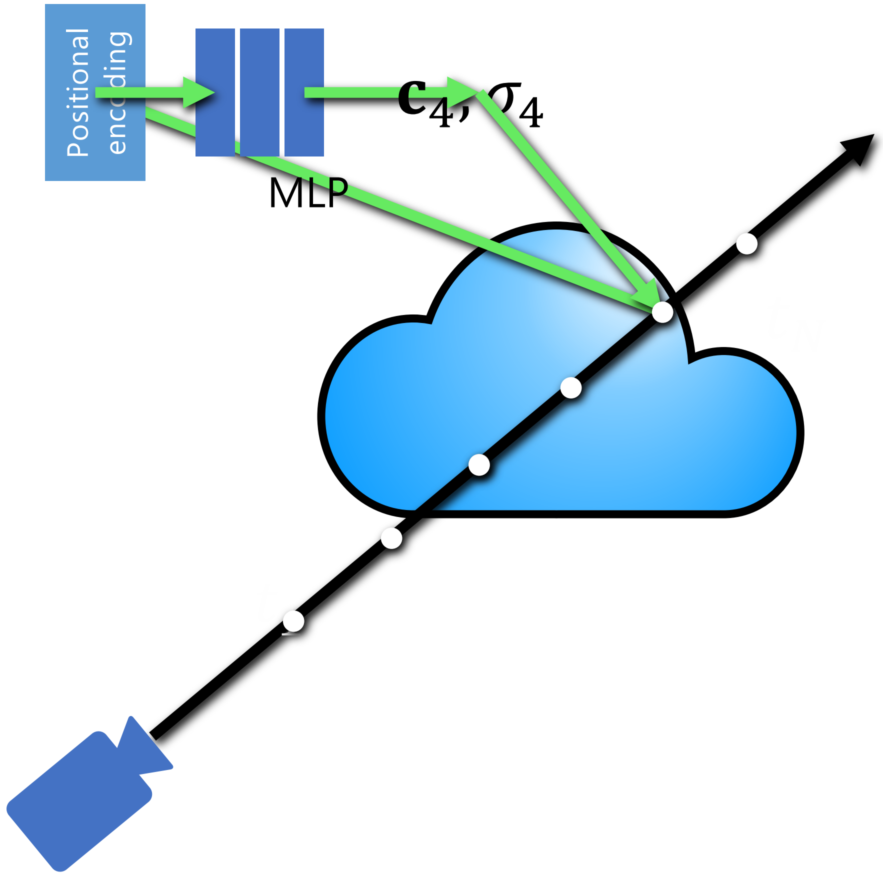
$\mathbf{c}, \sigma$



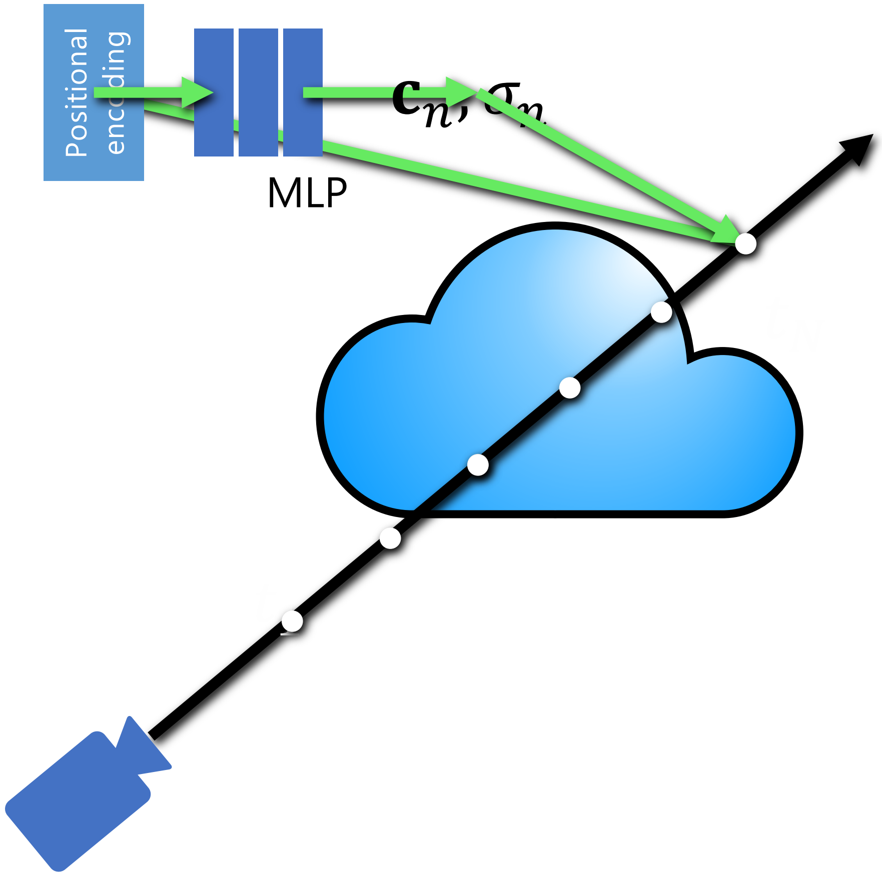
$\mathbf{c}, \sigma$

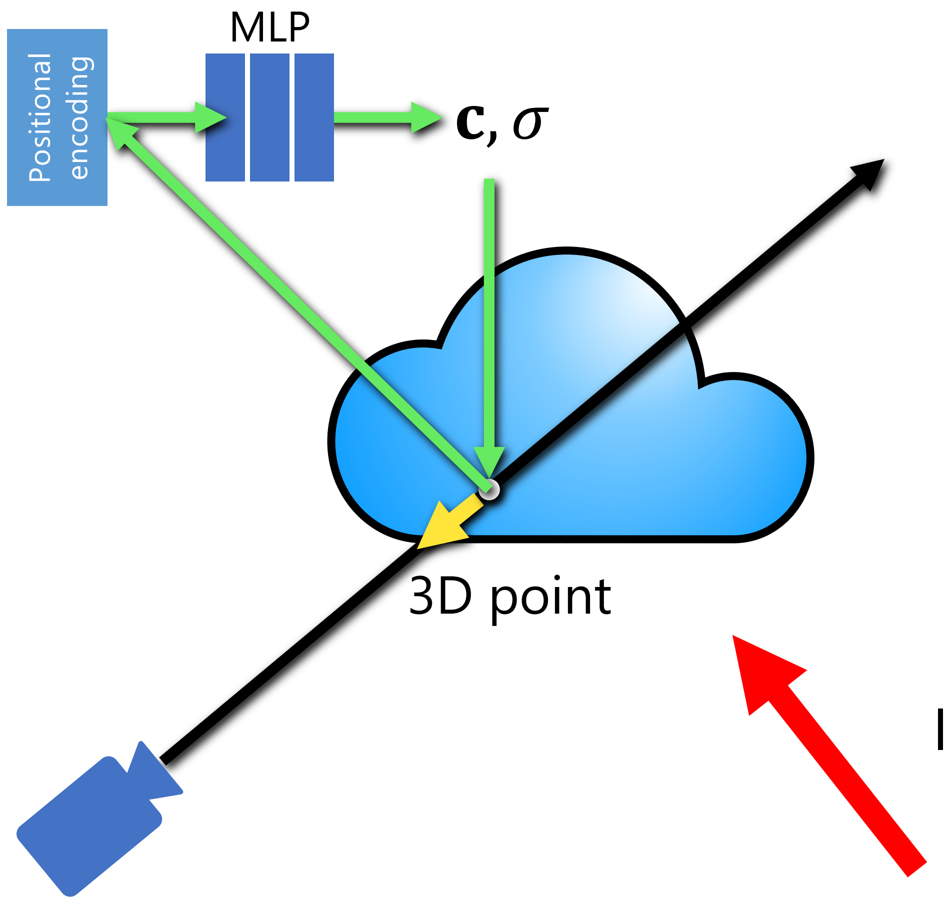


$\mathbf{c}, \sigma$



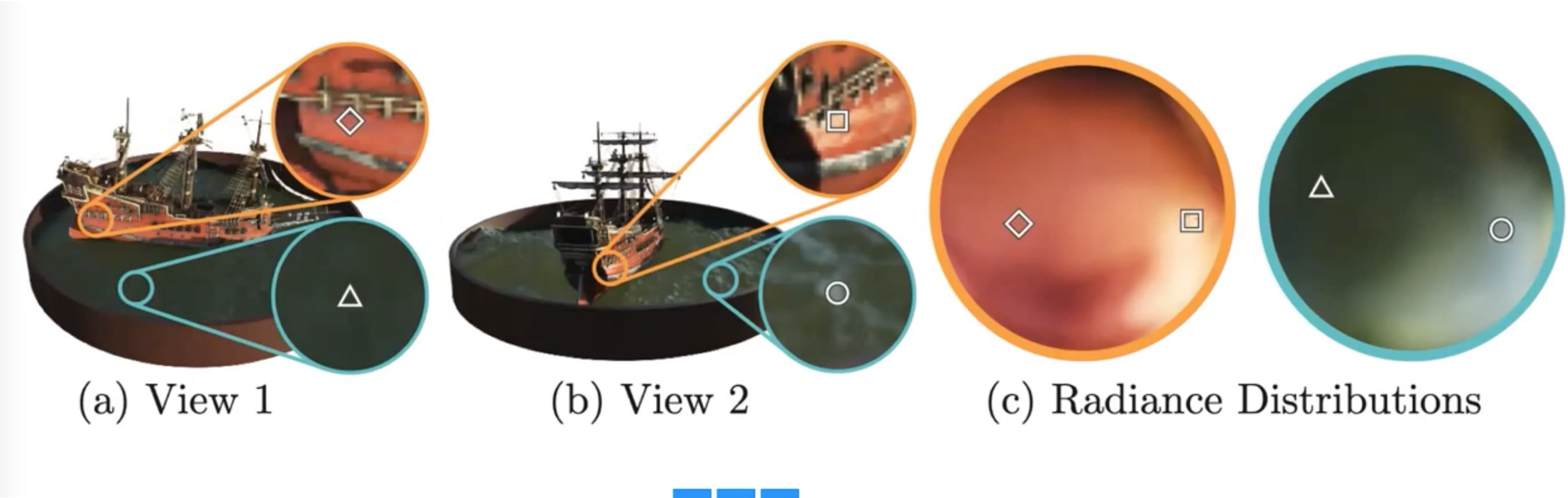
$\mathbf{c}, \sigma$

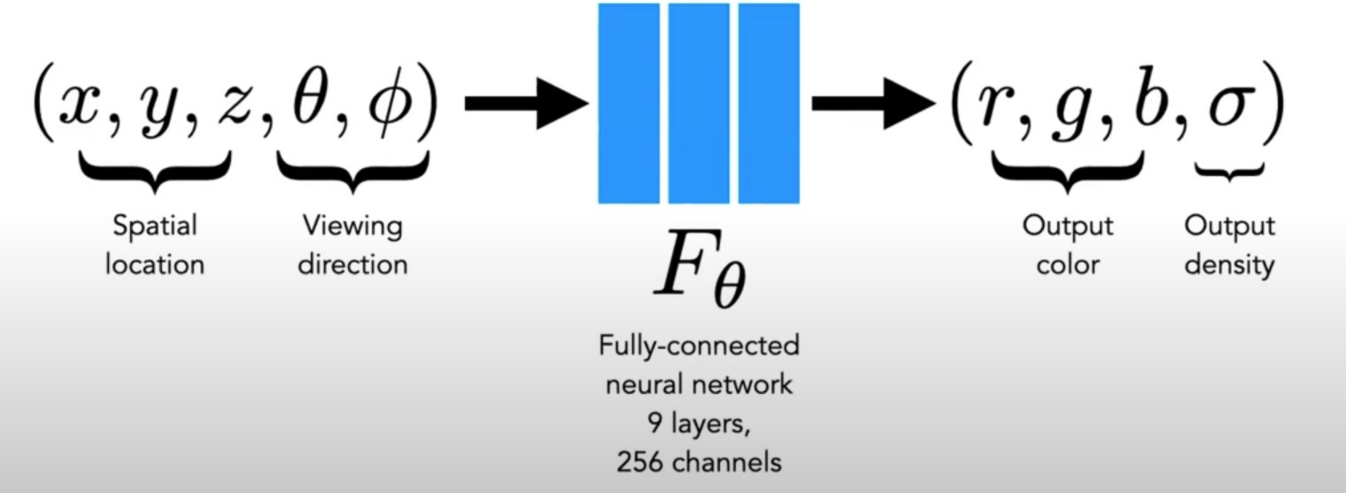




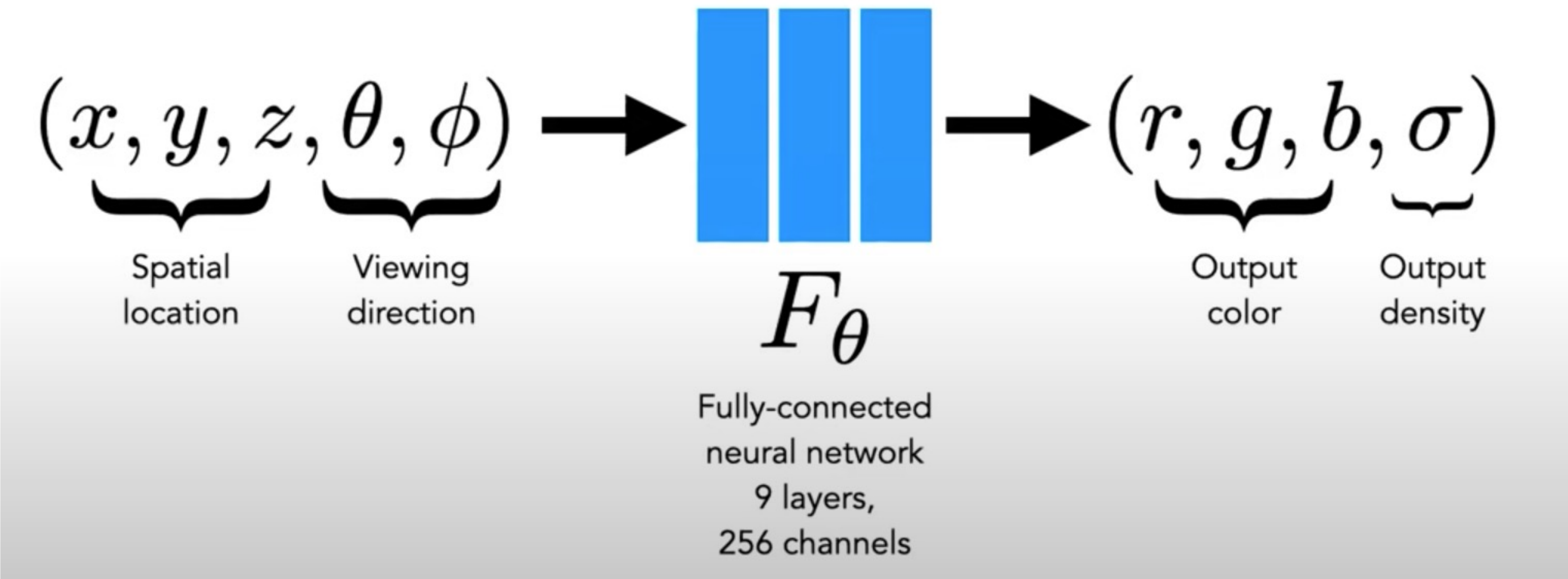
Include the ray direction in the input to the MLP → allows for capturing and rendering view-dependent effects (e.g., shiny surfaces)

Modeling view dependent effects



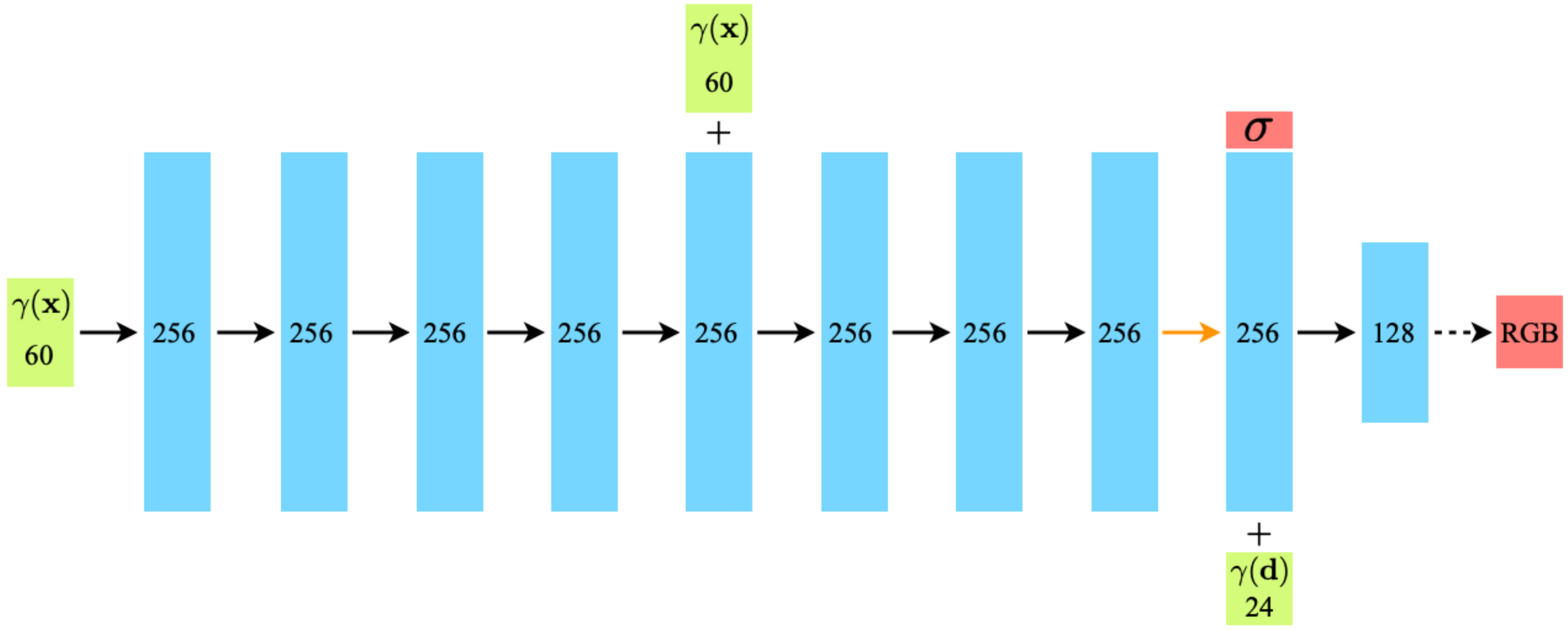


What do we learn in NeRF?



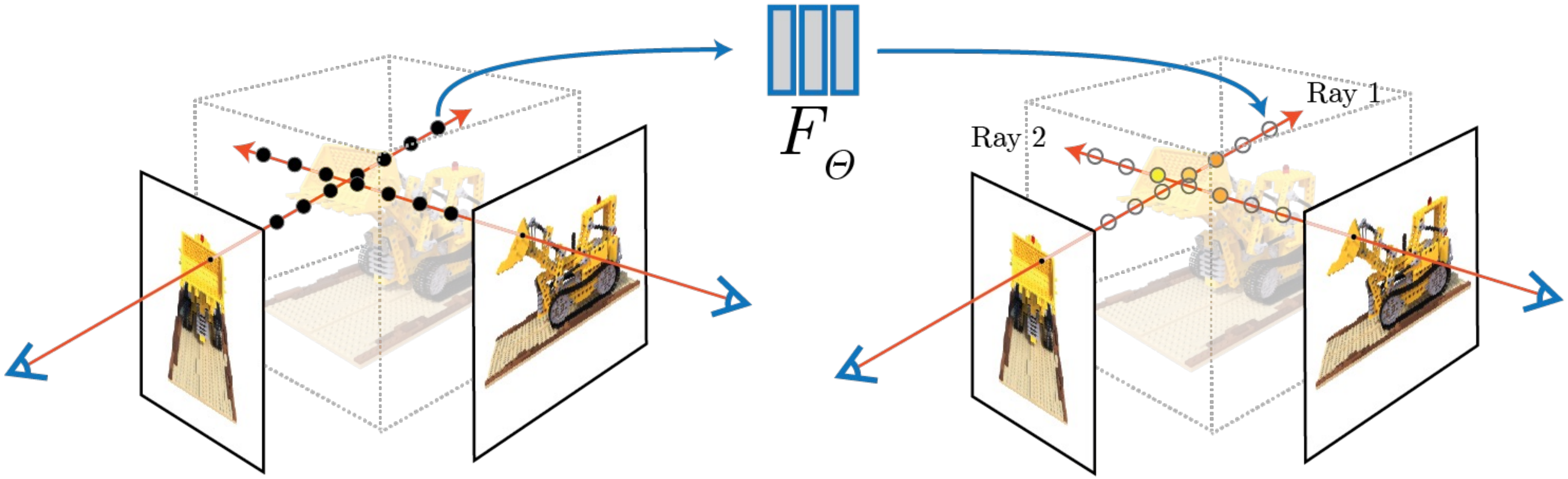
DeepSDF Extensions: NeRF

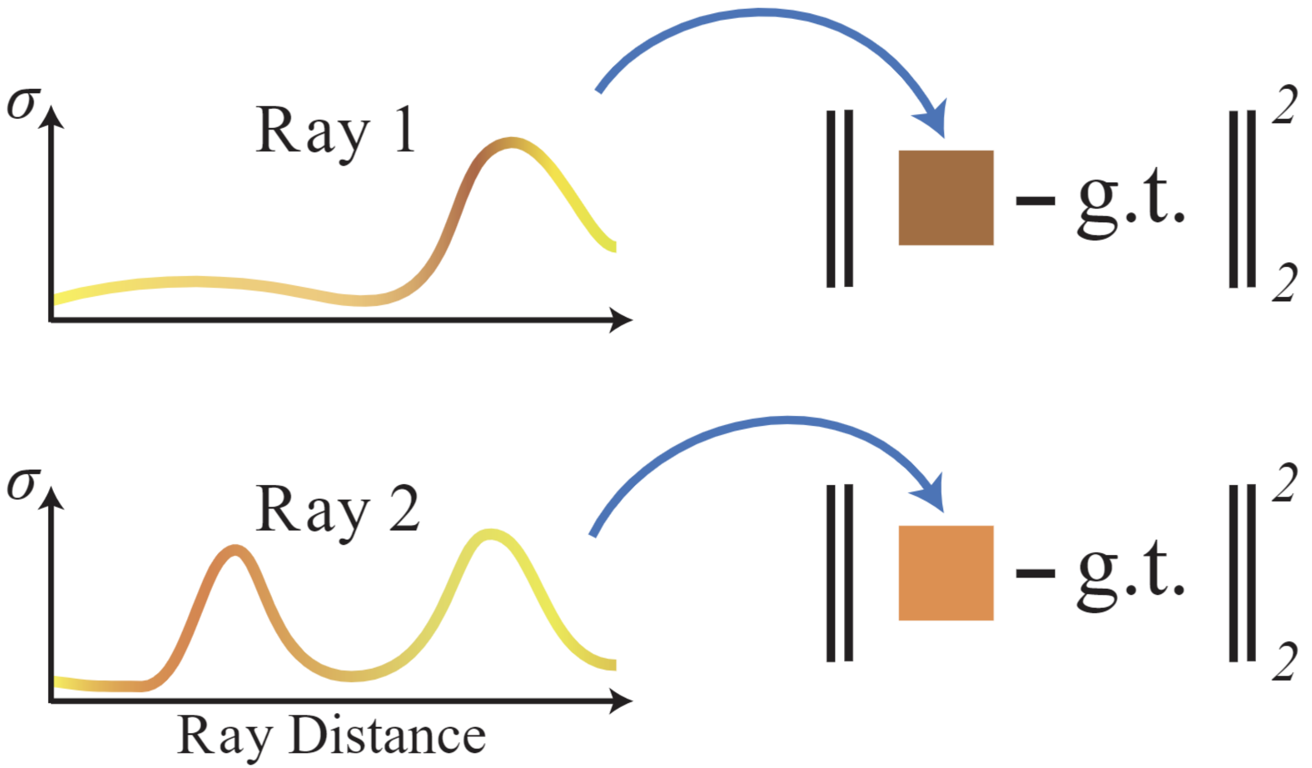
- Coordinate-based modeling of RGB and Densities Instead of SDFs

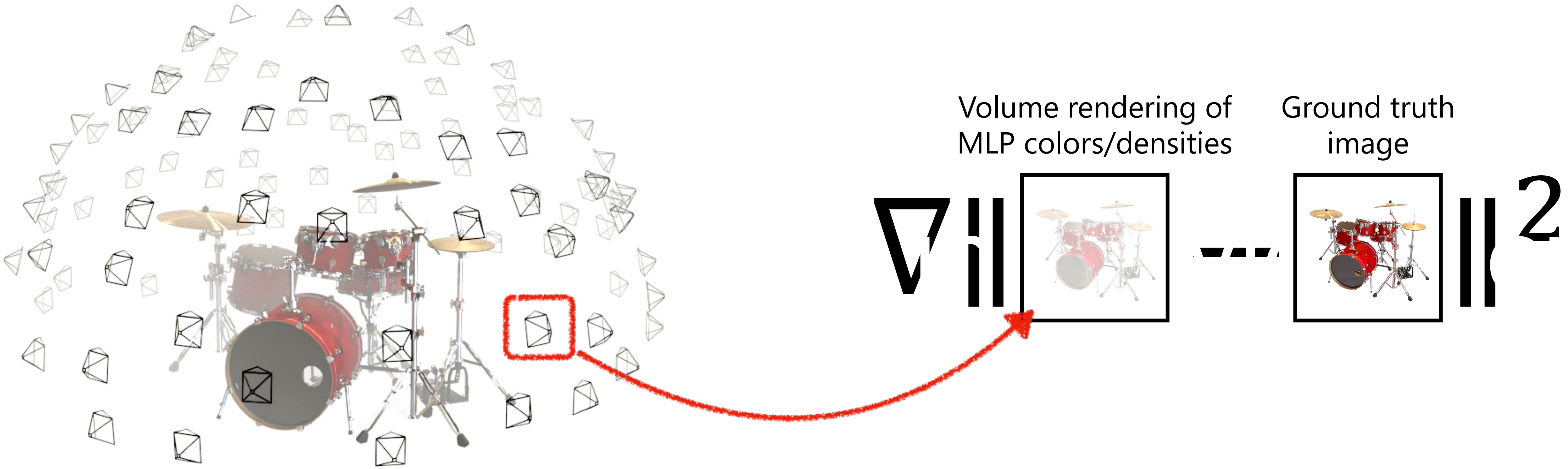


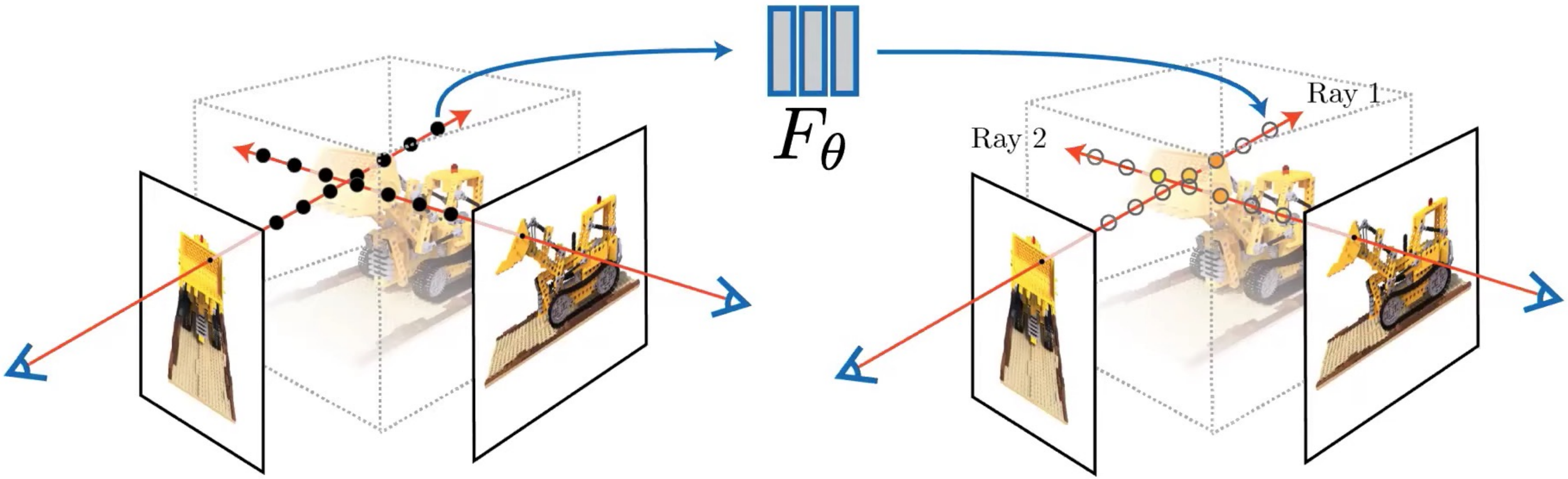
Mildenhall et al. 2020

Training NeRFs






Volume rendering of
MLP colors/densities
Ground truth
image

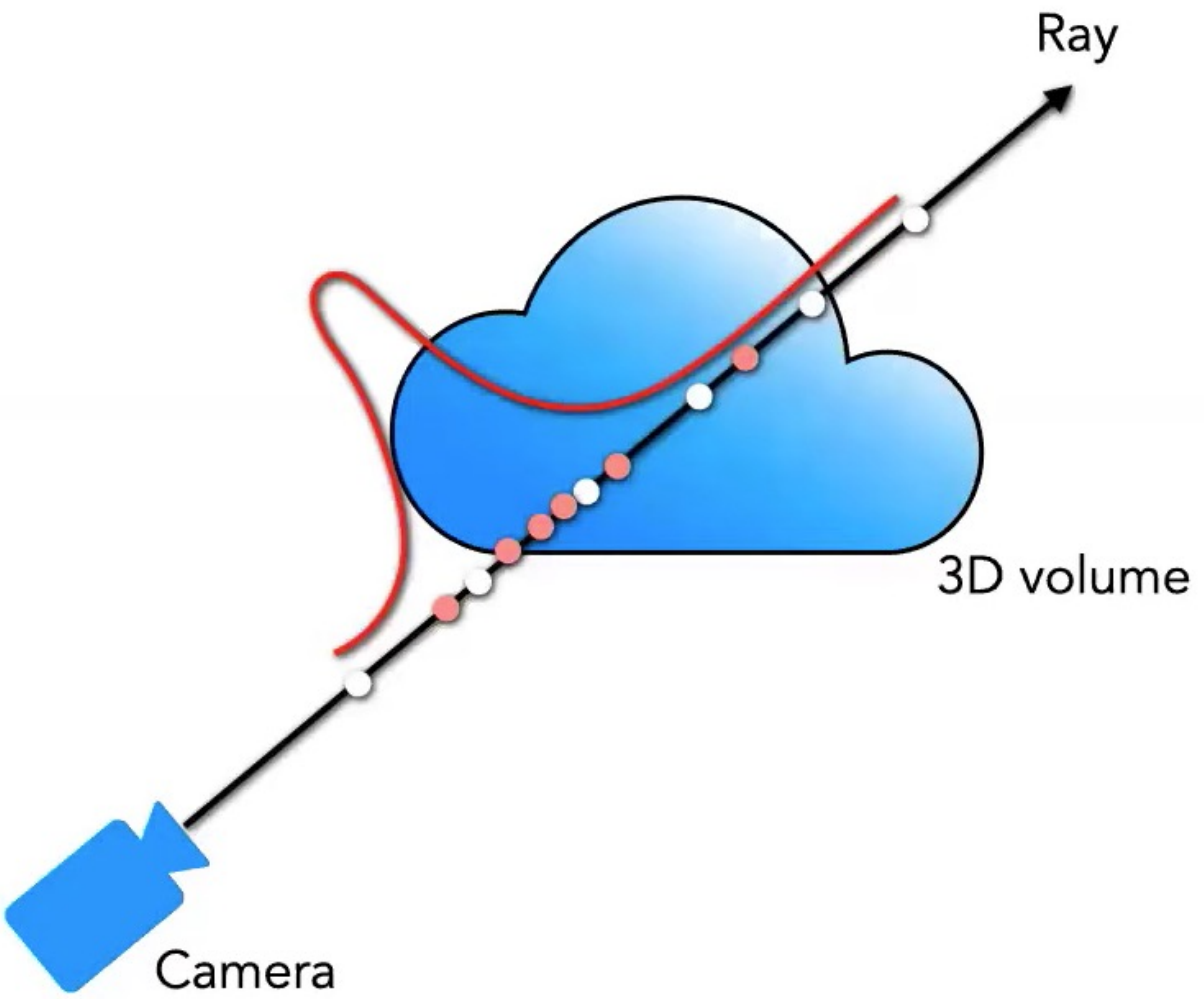


$$\min_\theta \sum_i || \mathrm{render}_i(F_\theta) - I_i ||^2$$

Importance Sampling

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

treat weights as probability distribution for new samples



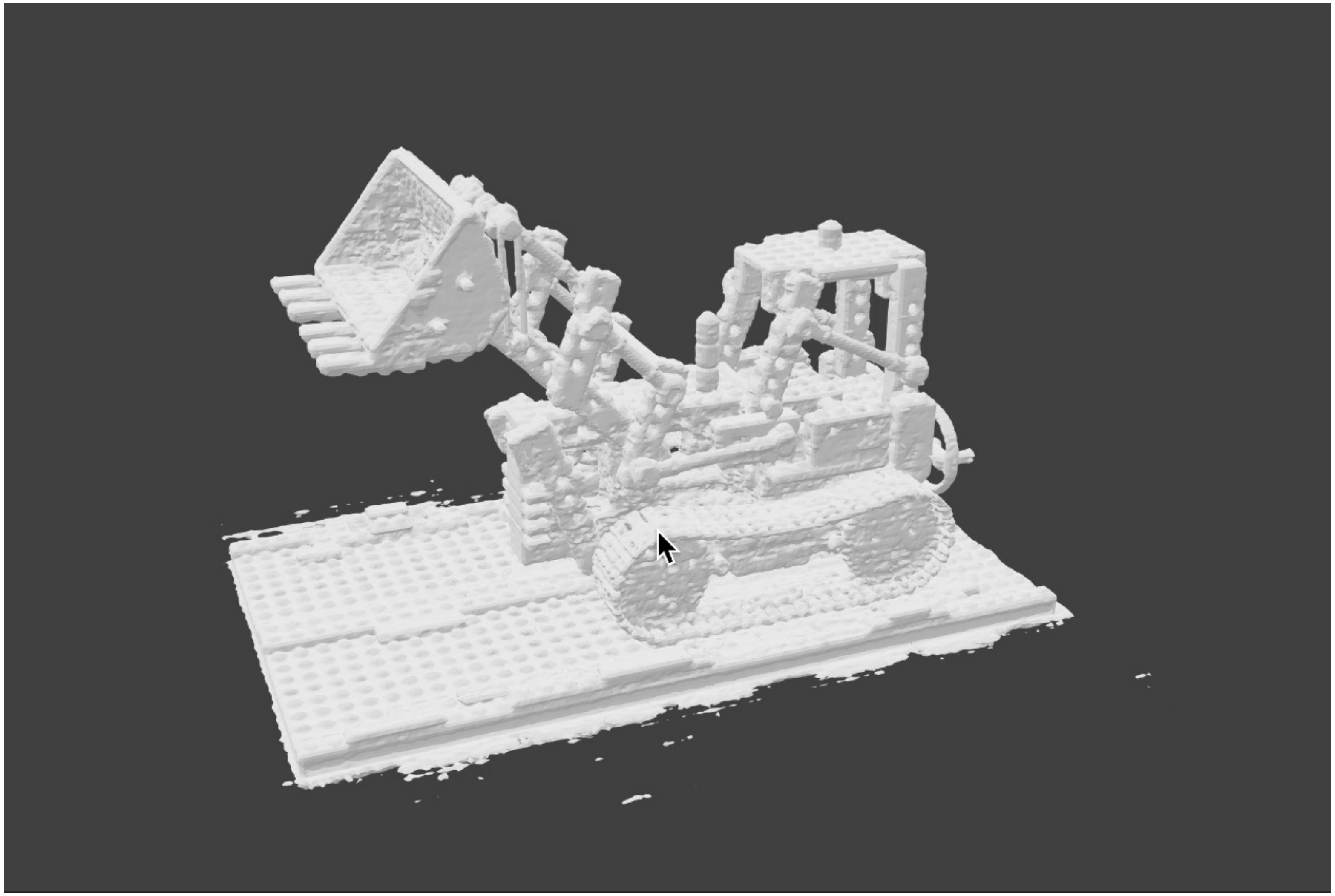

NOR-CAL
MOVING SERVICES

NOR-CAL
MOVING SERVICES

NeRF encodes convincing view-dependent effects using directional dependence

Building 3D models from NeRFs

Apply marching cubes algorithm on NeRF predicted volume density (σ)



Summary

- Represent the scene as volumetric colored "fog"
- Store the fog color and density at each point as an MLP mapping 3D position (x, y, z) to color c and density σ
- Render image by shooting a ray through the fog for each pixel
- Optimize MLP parameters by rendering to a set of known viewpoints and comparing to ground truth images

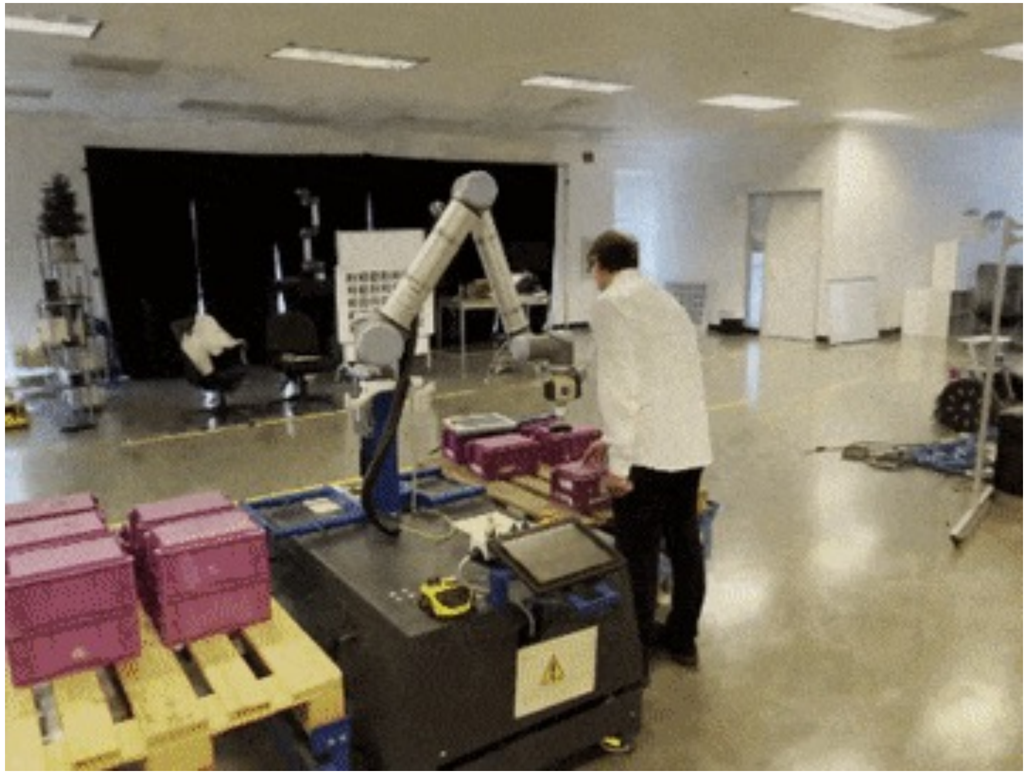
Key limitations of the original NeRF

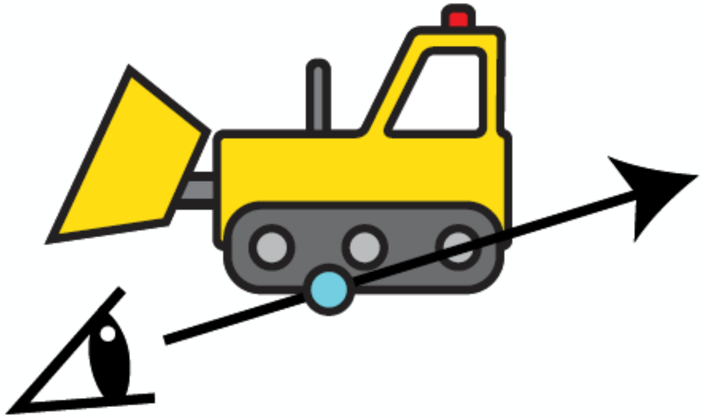
- Very slow in training and inference
- Requires Ground-Truth poses
- Do not generalize to new scenes

Key limitations of the original NeRF

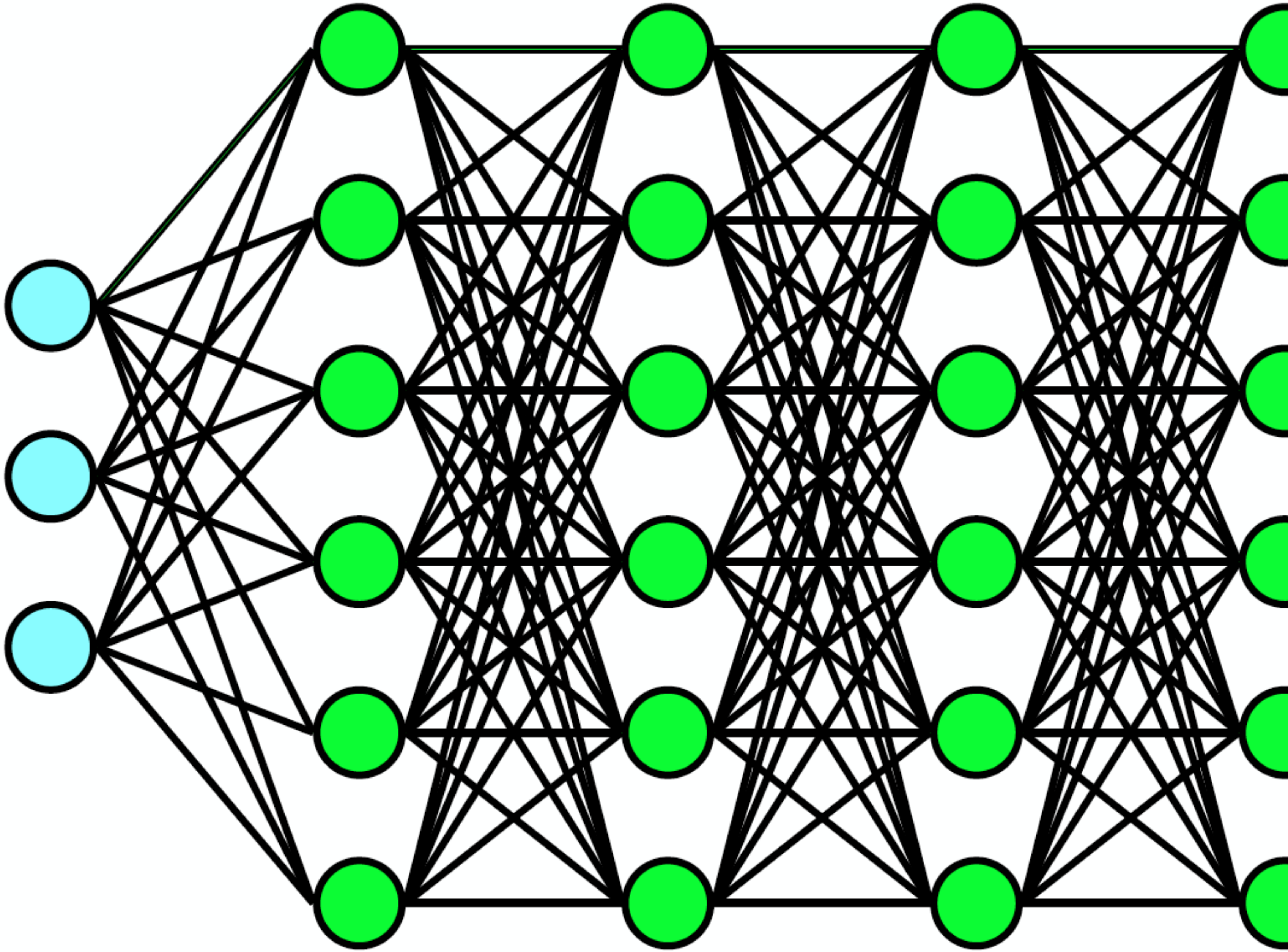
- Very slow in training and inference
- Requires Ground-Truth poses
- Do not generalize to new scenes

Instant NGP: Superfast training and inference with NeRF using multi-resolution hash-table



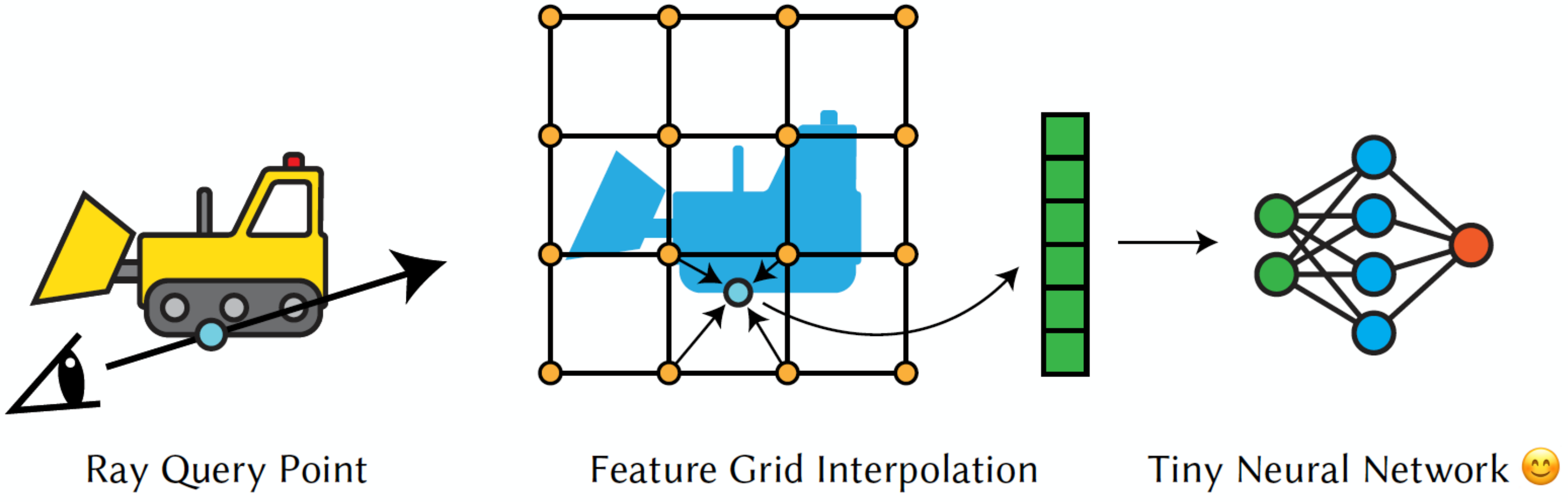


Ray Query Point



Huge Neural Network ☹️

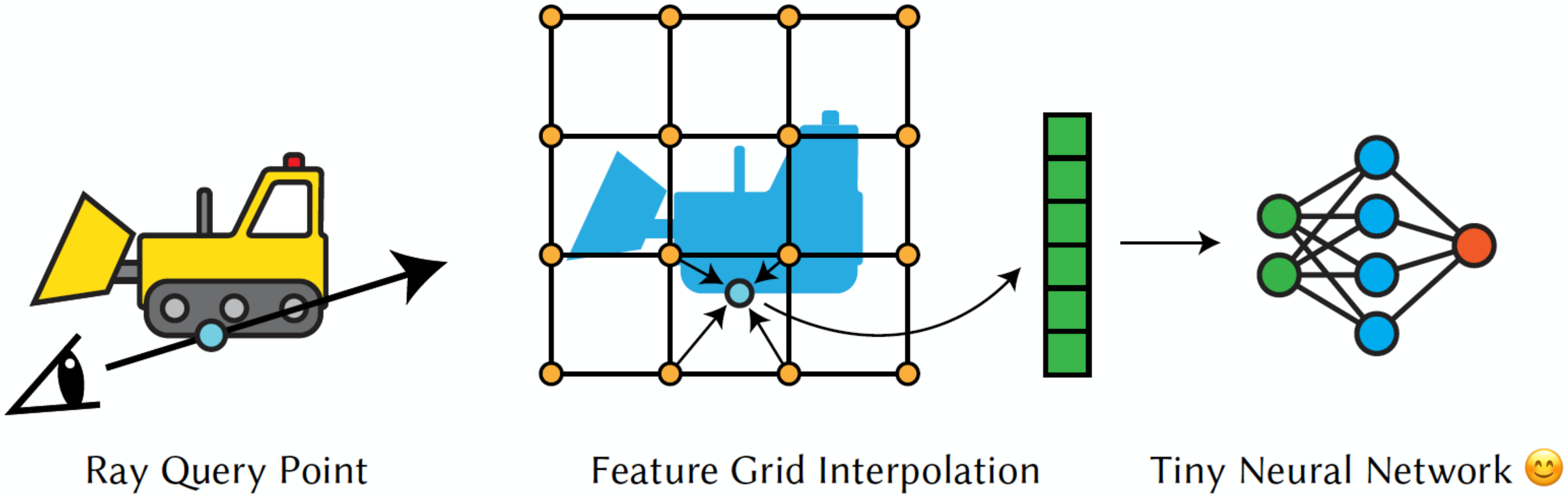
Hybrid representation



Features:

- are also parameters that can be updated while training the NeRF. (slight increase in memory, significantly faster training & inference)
- are individual NeRFs trained on a small section of a scene (for large city-size scene)
- are priors obtained from ConvNets, e.g. VGG-features (used for generalization)

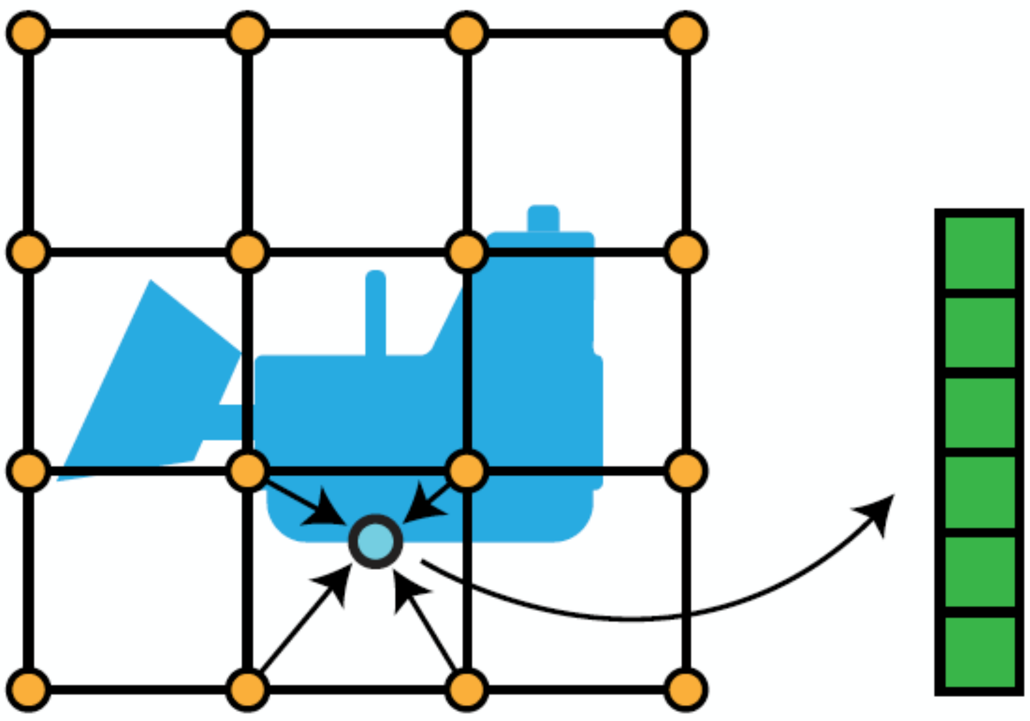
Hybrid representation: It's all about Data Structures!



Why hybrid representation?

- Reduce the size of neural network -> fast inference & rendering.
- Helps in rendering large scale scenes.
- Helps in generalization.

Uniform Grids



[PIFu (Saito et al.), Neural Volumes (Lombardi et al.), etc]

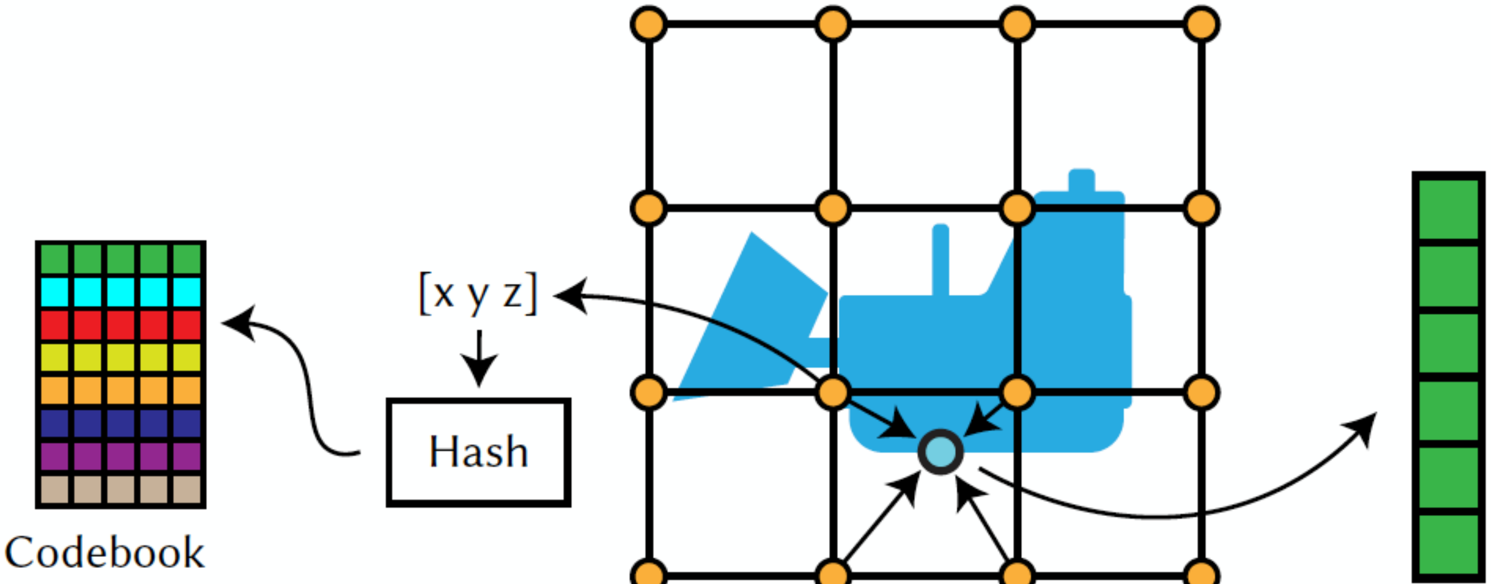
Pros:

- Easy to implement
- Algorithmically fast access [O(1)]
- Established operations like convolutions
- Simple topology

Cons:

- Expensive in memory and bandwidth
- Limited by Nyquist

Hash Grids



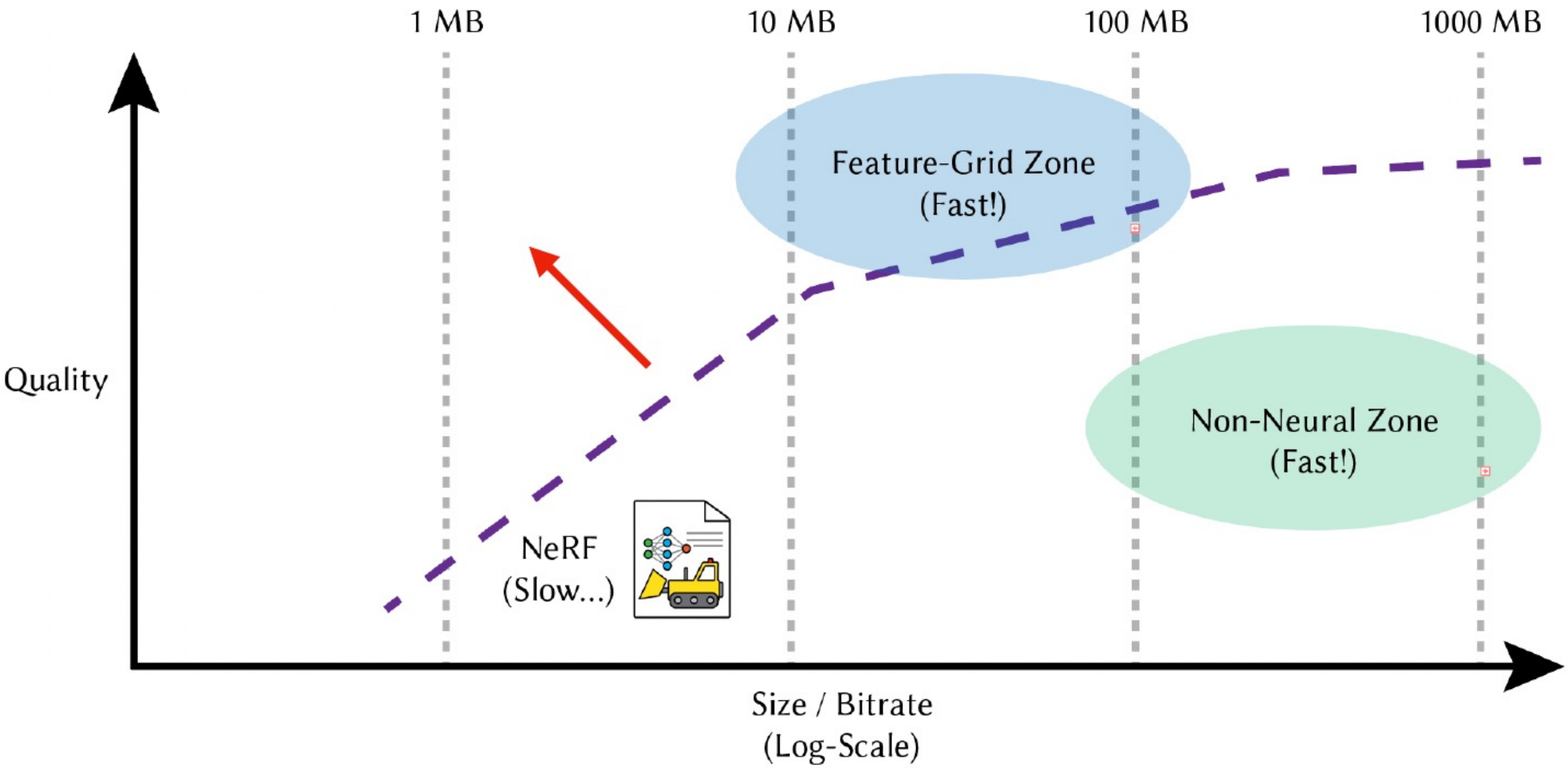
[Instant-NGP (Muller et al.)]

Pros:

- Densely supported
- Disaggregate resolution from memory cost
- No complex data structures
- Performant memory access if codebook is small enough

Cons:

- Multiresolution and large codebooks needed for collision resolution
- Features not spatially local



1 MB
10 MB
100 MB
1000 MB
Feature-Grid Zone
(Fast!)
Quality
Non-Neural Zone
(Fast!)
NeRF
(Slow...)
Size / Bitrate
(Log-Scale)


BakedSDF

Key limitations of the original NeRF

- Very slow in training and inference
- Requires Ground-Truth poses
- Do not generalize to new scenes

BARF 🤮: Bundle-Adjusting Neural Radiance Fields

Chen-Hsuan Lin 😷 Wei-Chiu Ma 🤢 Antonio Torralba 🤢 Simon Lucey 😷🤒

😷 Carnegie Mellon University 🤢 Massachusetts Institute of Technology 🤒 The University of Adelaide

IEEE International Conference on Computer Vision (**ICCV**), 2021

oral presentation

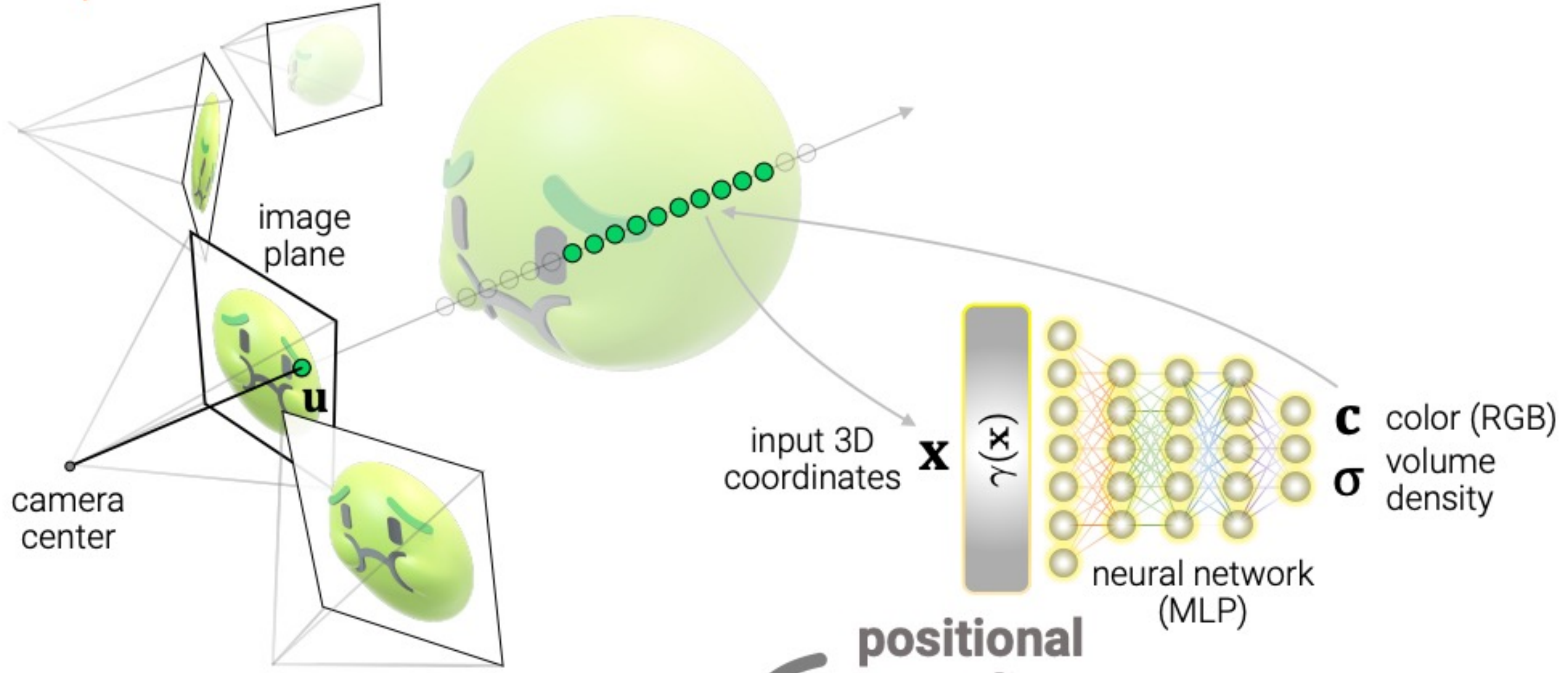
NeRF in a nutshell

We want to optimize the poses as well!

frames

$$\min_{\mathbf{p}_1,\ldots,\mathbf{p}_M,\boldsymbol{\Theta}} \sum_{i=1}^{M} \sum_{\mathbf{u}} \left\| \hat{\mathcal{I}}(\mathbf{u};\mathbf{p}_i,\boldsymbol{\Theta}) - \mathcal{I}_i(\mathbf{u}) \right\|_2^2$$

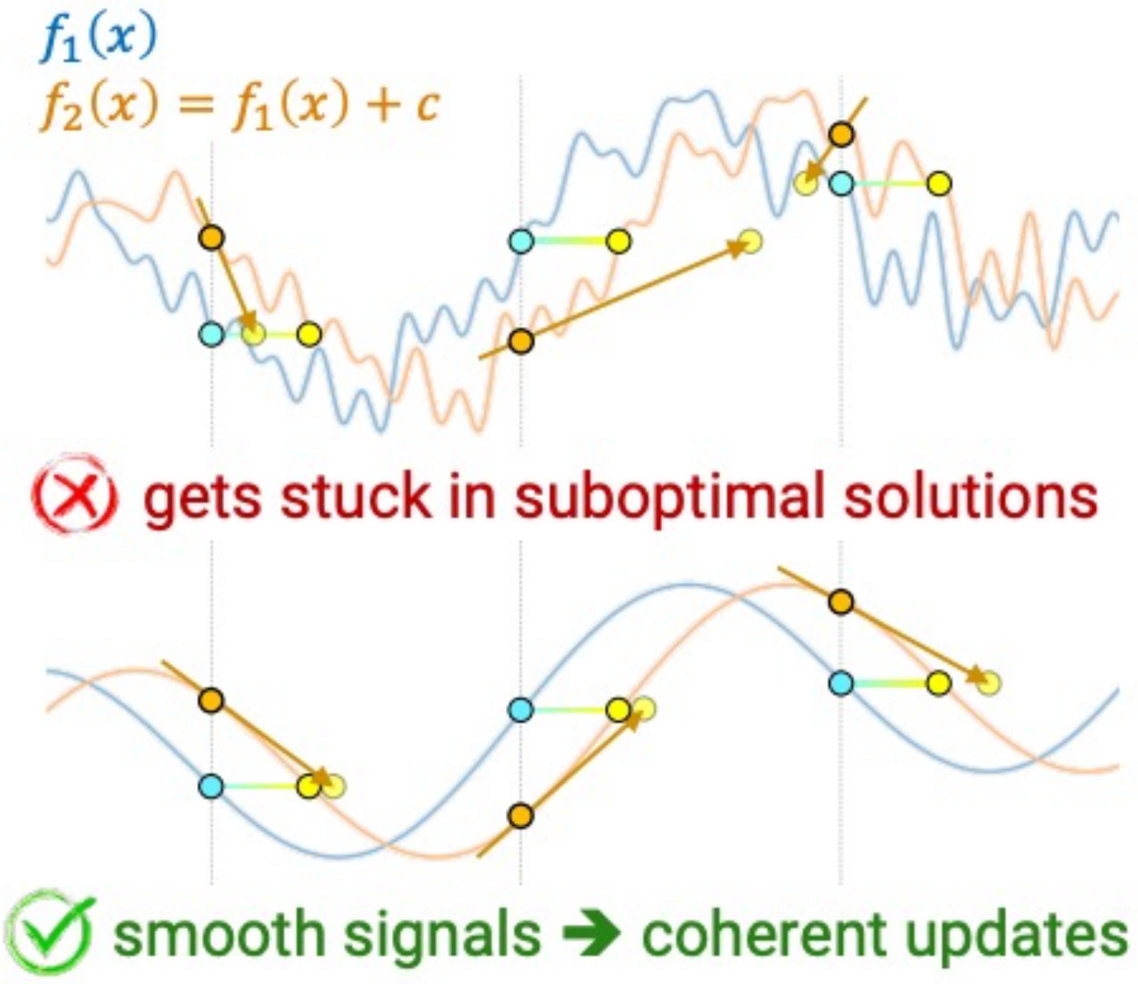
RGB (rendered) · camera pose · network params. · RGB



positional encoding

✓ encourages representation with high frequency

✗ detrimental to gradient-based **registration**!!



V 2020

SOLUTION 🤩 **:**

make it **coarse-to-fine**!

Resolve large pose misalignment & coarse scene representation

Gradually activate higher-frequency components in positional encoding

Refine granular pose misalignment & high-fidelity scene representation

Key limitations of the original NeRF

- Very slow in training and inference
- Requires Ground-Truth poses
- Do not generalize to new scenes

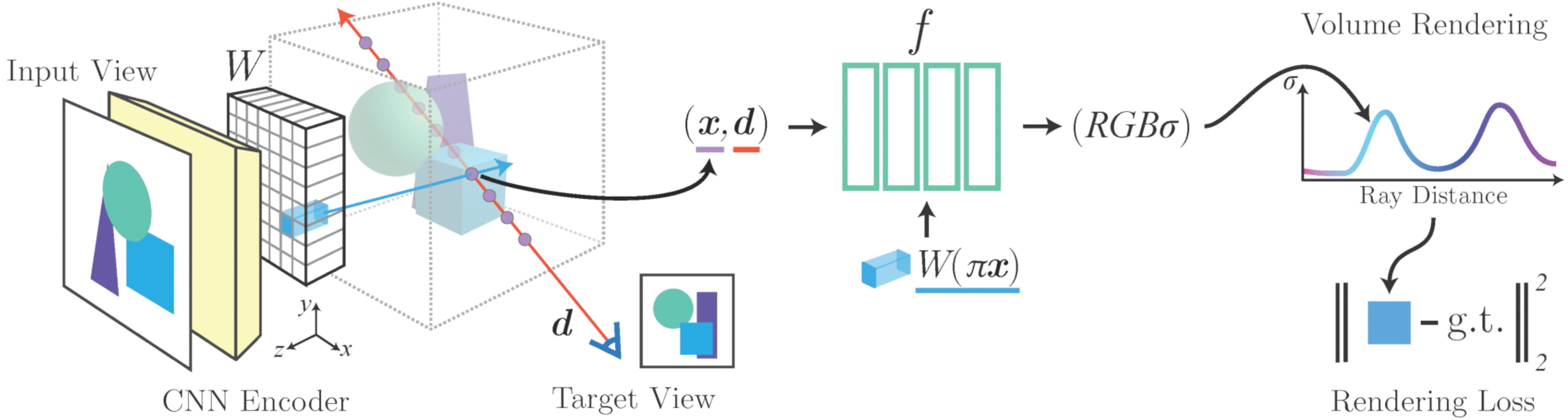
pixelNeRF

Neural Radiance Fields from One or Few Images

CVPR 2021

Alex Yu Vickie Ye Matthew Tancik Angjoo Kanazawa

UC Berkeley




Input Images
PixelNeRF
NeRF

Slide Credits

- ”Introduction to Computer Vision”, Noah Snavely, Cornell Tech, Spring 2022
- “Understanding and Extending Neural Radiance Field”, Jon Barron MIT & Tu Munich Lecture.
- “Neural Fields in Computer Vision”, CVRP 2022 Tutorial.
- Shubham Tulsiani, “Learning for 3D Vision”, Spring 2022, CMU
- Leo Guibas, JJ Park, “Neural Models for 3D geometry”, Spring 2022, Stanford.