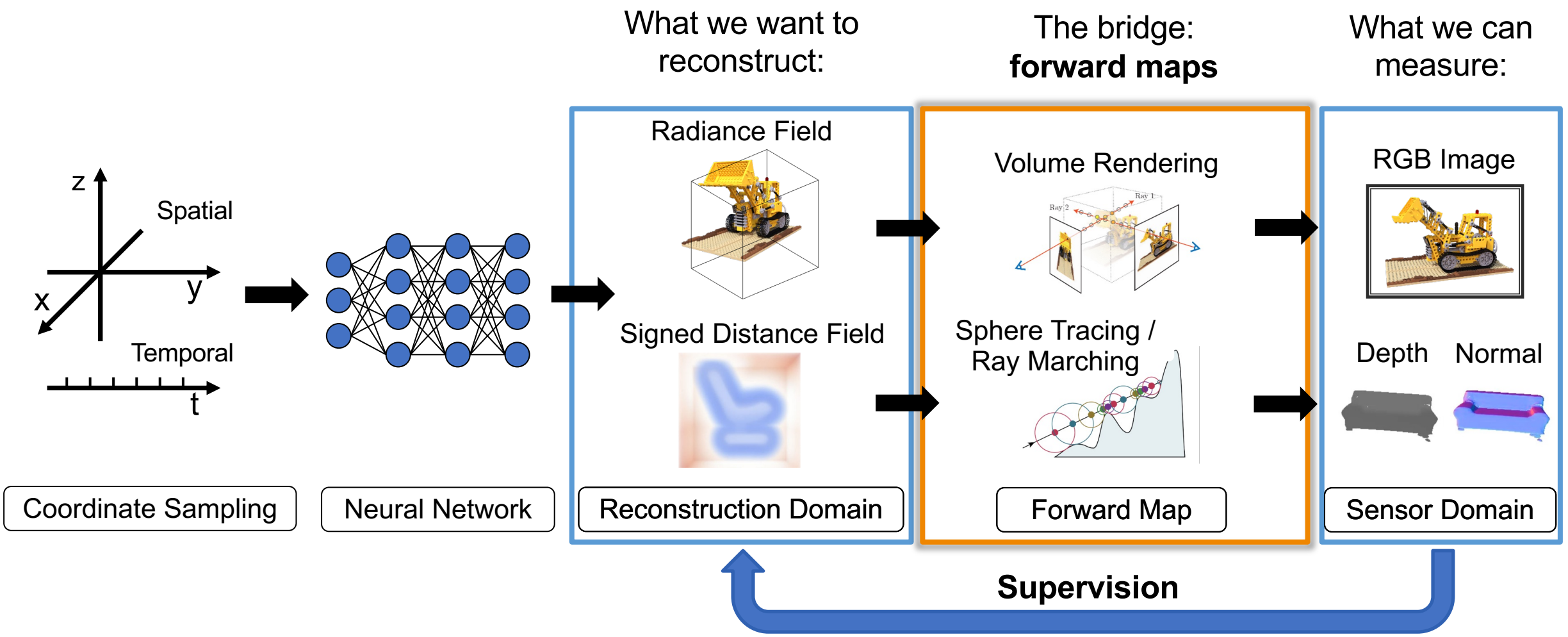
Lecture: Neural Fields Part 2

Neural Field General Framework



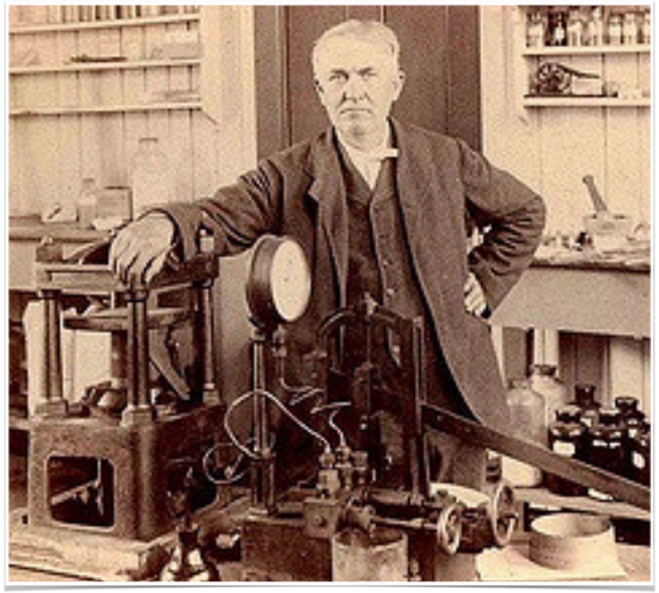
Outline

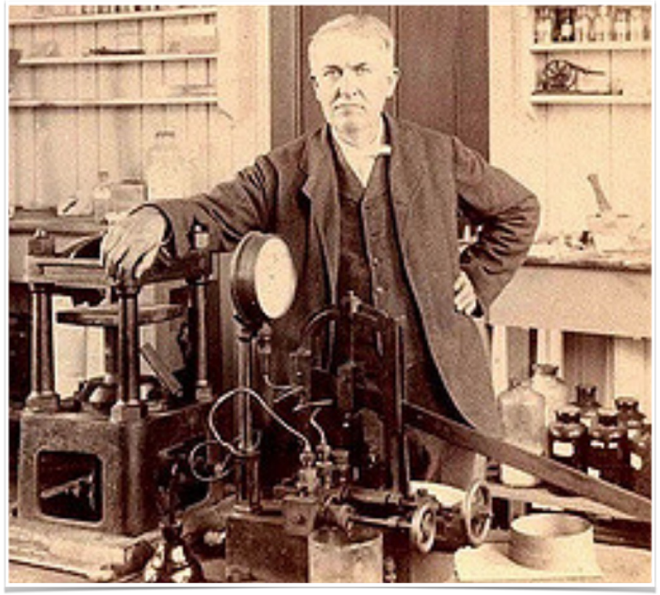
- Motivation for NeRF: View synthesis
- Introduction Volume Rendering
- Neural Radiance Fields (NeRFs)

Outline

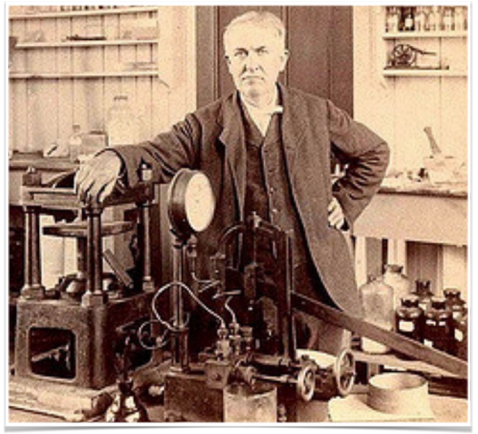
- Motivation for NeRF: View synthesis
- Introduction Volume Rendering
- Neural Radiance Fields (NeRFs)

Side Topic: Stereo Photography

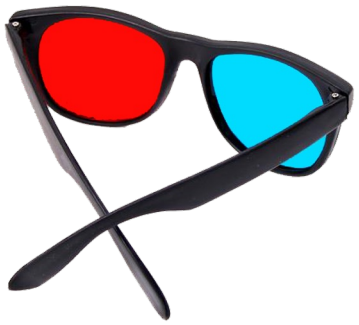




Stereo Photography

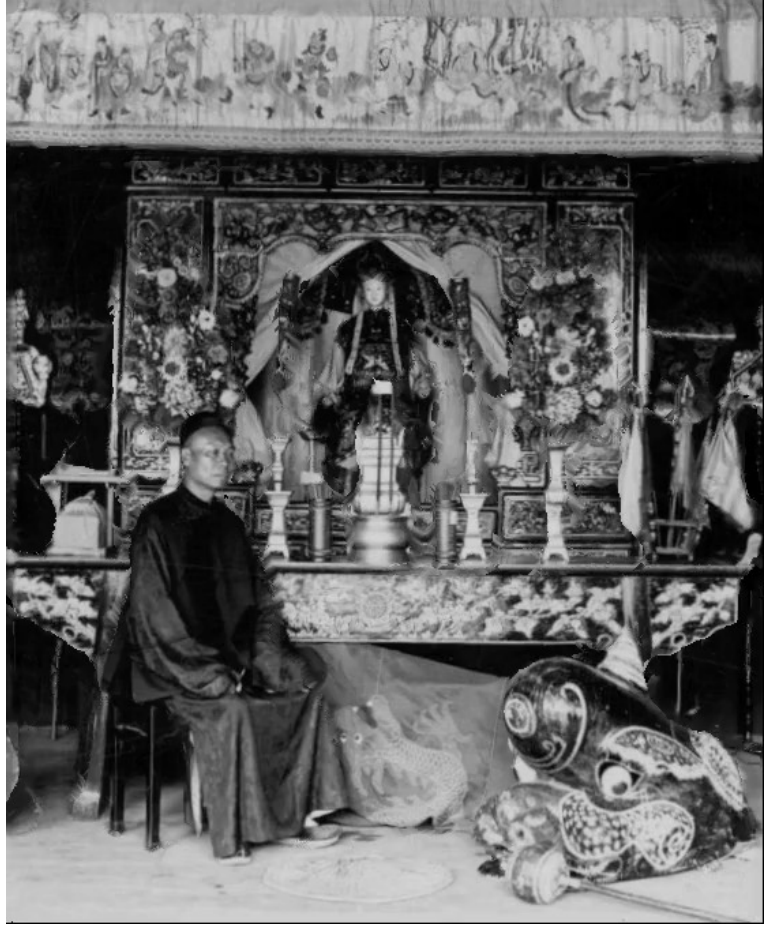
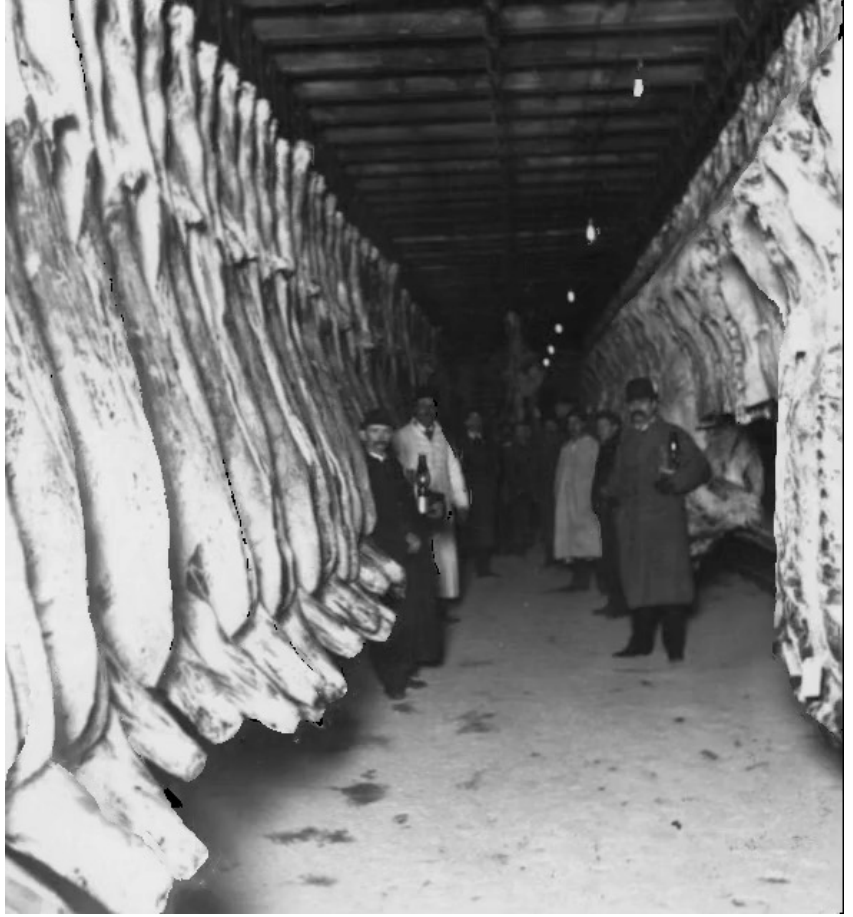


Viewing Devices



Stereo Photography

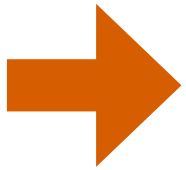
Queen Victoria at World Fair, 1851

“Keystone Depth”, Xuan Luo et. al.

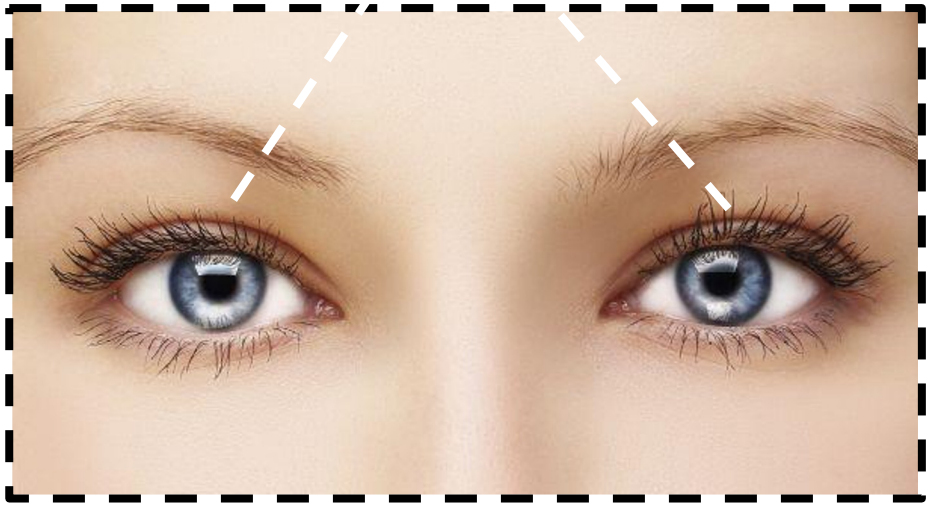
Stereo cameras were invented in the 1850s and hundreds of thousands of antique stereographs are available today. We introduce *KeystoneDepth*, a collection of 37,239 antique stereographs of historical scenes captured between 1864 and 1966 with clean rectified stereo image pairs, disparity maps, and meta data.

Stereo Photography



Issue: Narrow Baseline

~6.5 cm



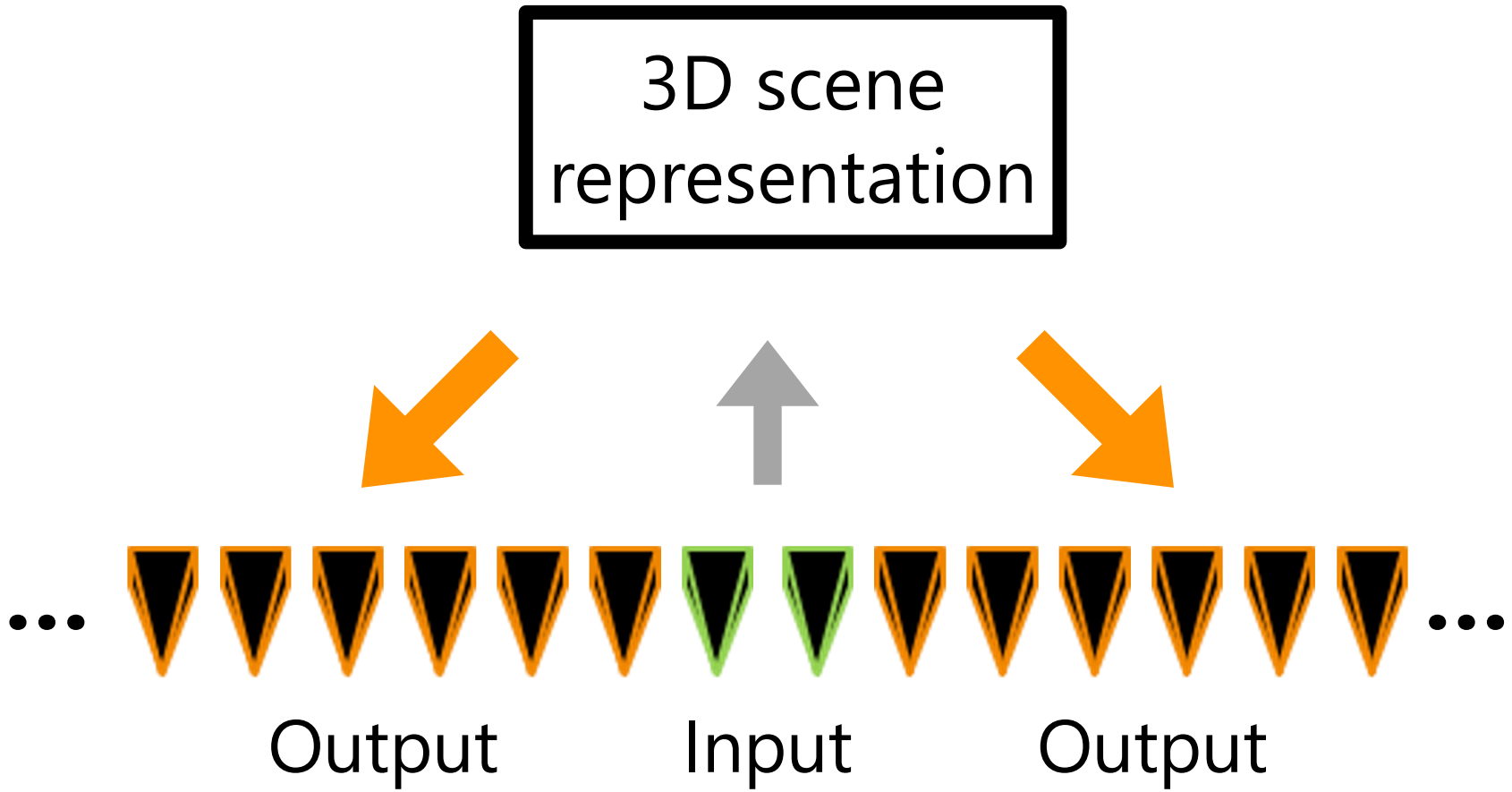
~1.5 cm


Left
MAZZER


Right
MAZZER


MAZZER

Problem Statement



Challenges

Extrapolation

Large disocclusion

Output ... Input ... Output

Non-Lambertian Effects

Reflections, transparencies, etc.

Soft 3D

(Penner & Zhang 2017)

Culmination of non-deep stereo matching techniques

Multiplane image methods

Stereo Magnification (Zhou et al. 2018)

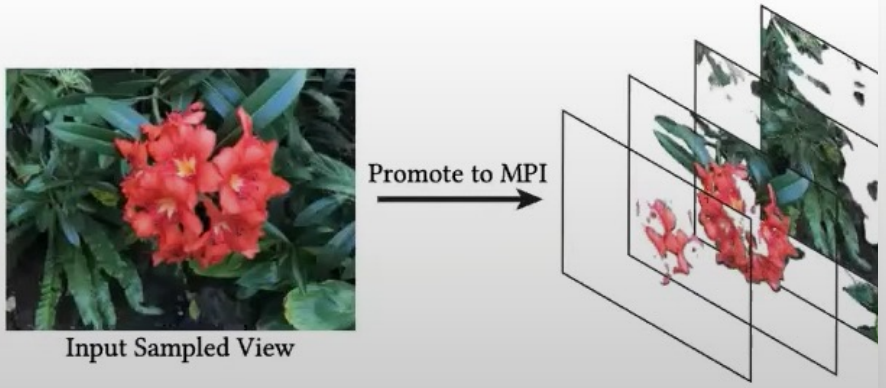
Pushing the Boundaries… (Srinivasan et al.

Local Light Field Fusion (Mildenhall et al. 2

DeepView (Flynn et al. 2019)

Single-View… (Tucker & Snavely 2020)

Typical deep learning pipelines - ima go into a 3D CNN, big RGBA 3D vo comes out

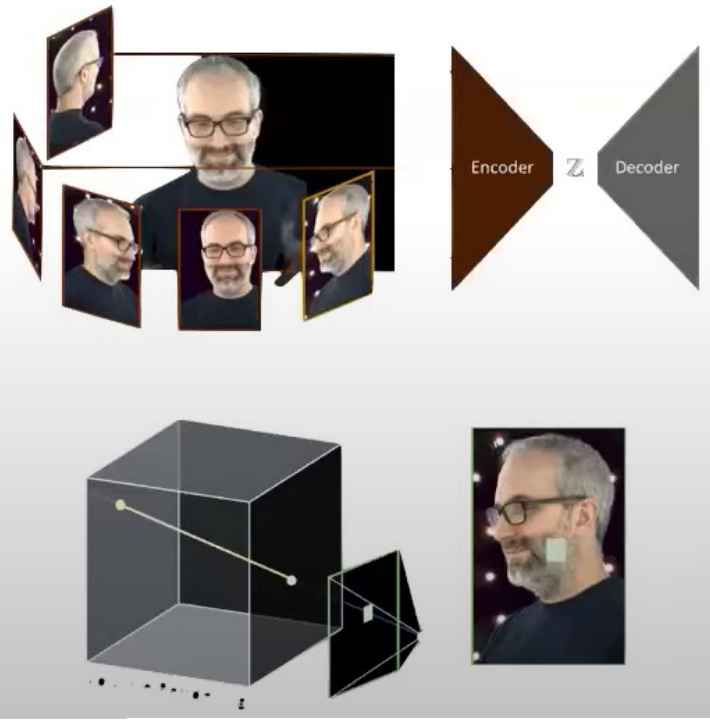


Input Sampled View

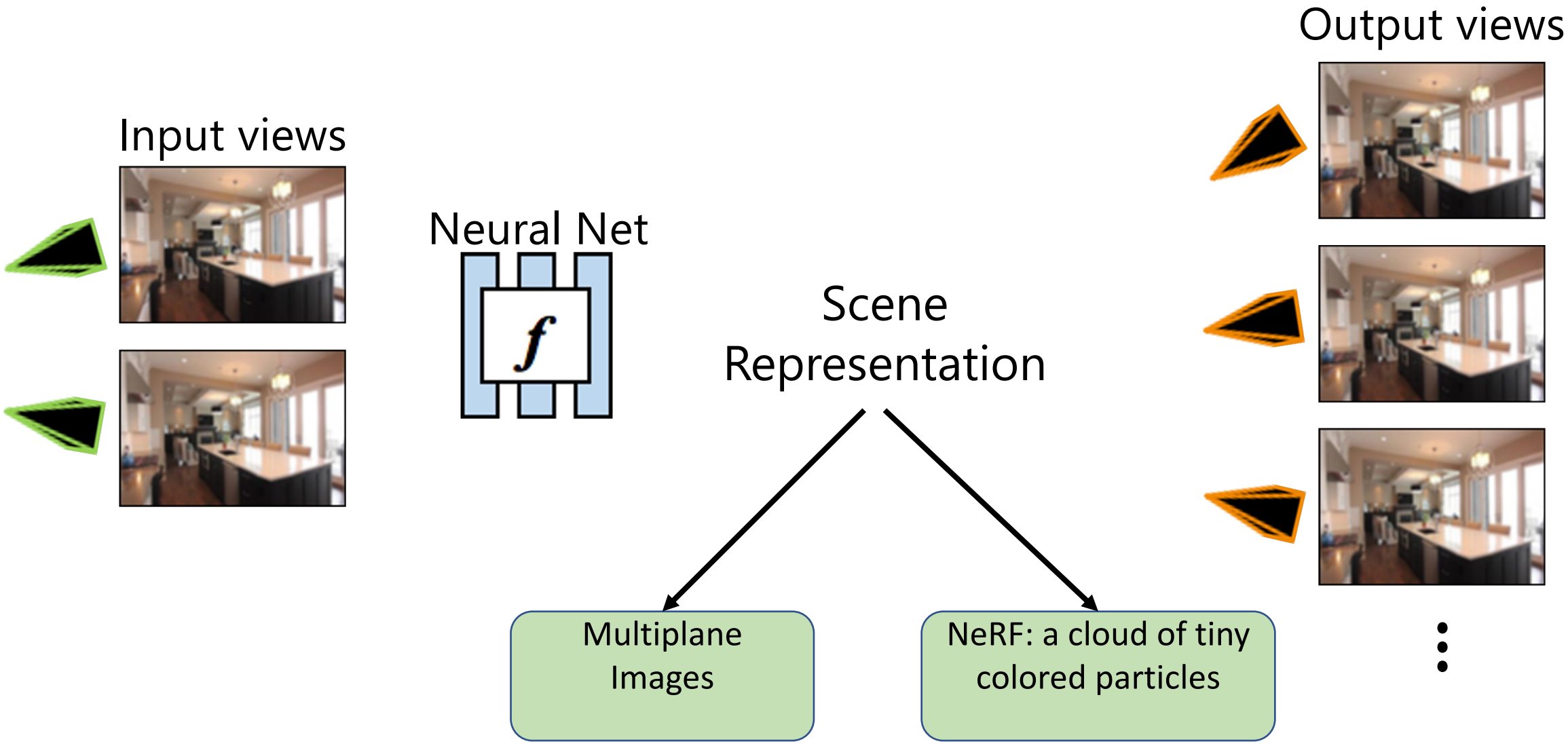
Neural Volumes

(Lombardi et al. 2019)

Direct gradient descent to optimize an RGBA volume, regularized by a 3D CNN



Neural prediction of scene representations

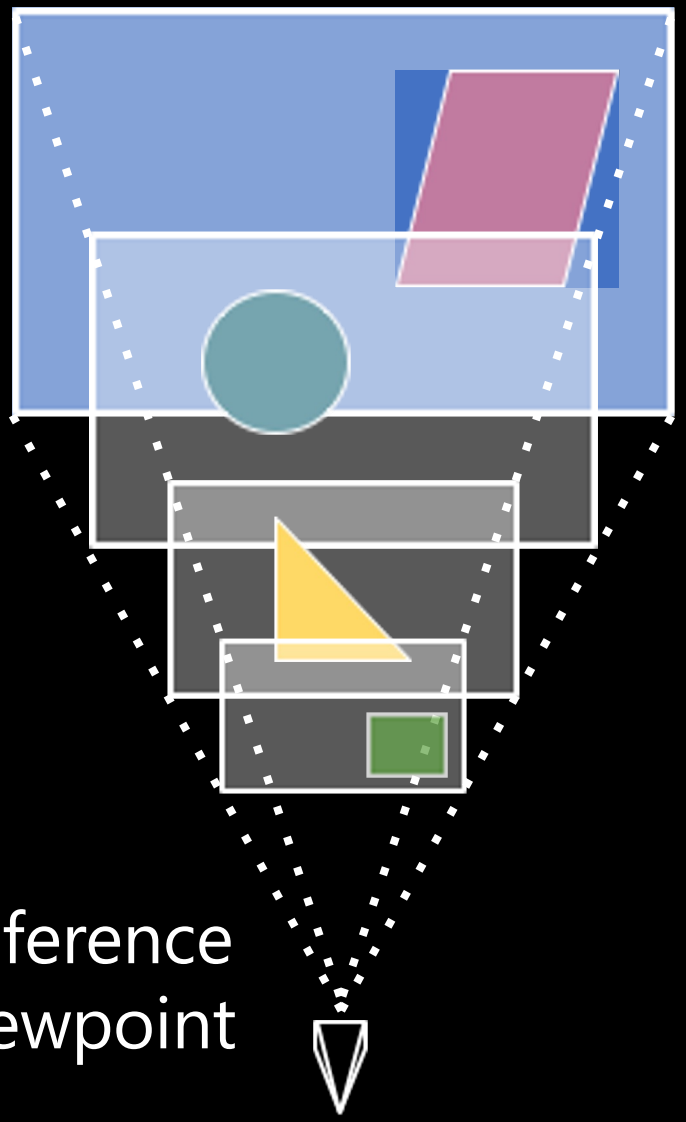


Stereo Magnification: Learning View Synthesis using Multiplane Images

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, Noah Snavely

SIGGRAPH 2018

Multiplane Images (MPIs)



Each plane is at a fixed depth and encoded by an RGBA image

Reference Viewpoint

Multiplane Camera (1937)

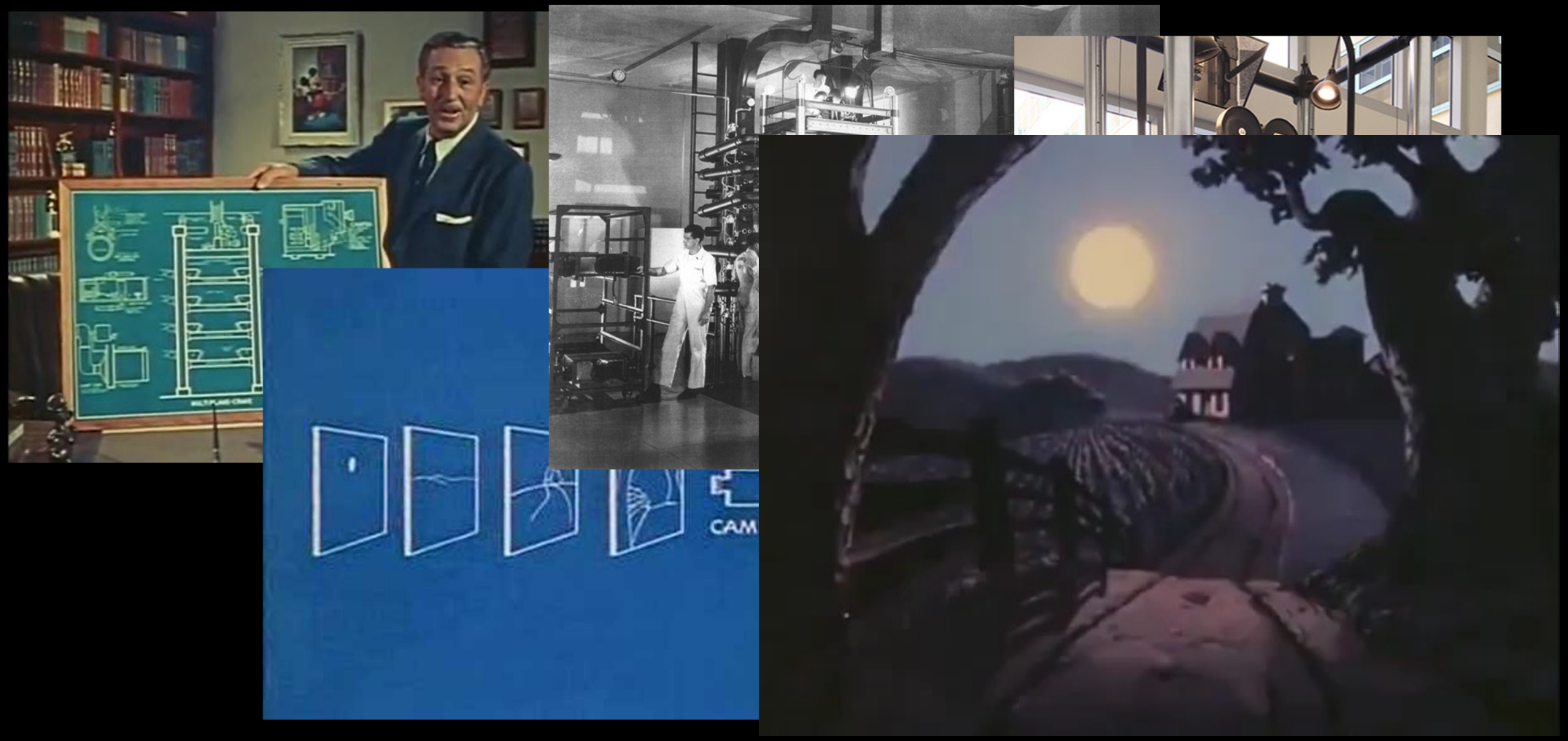
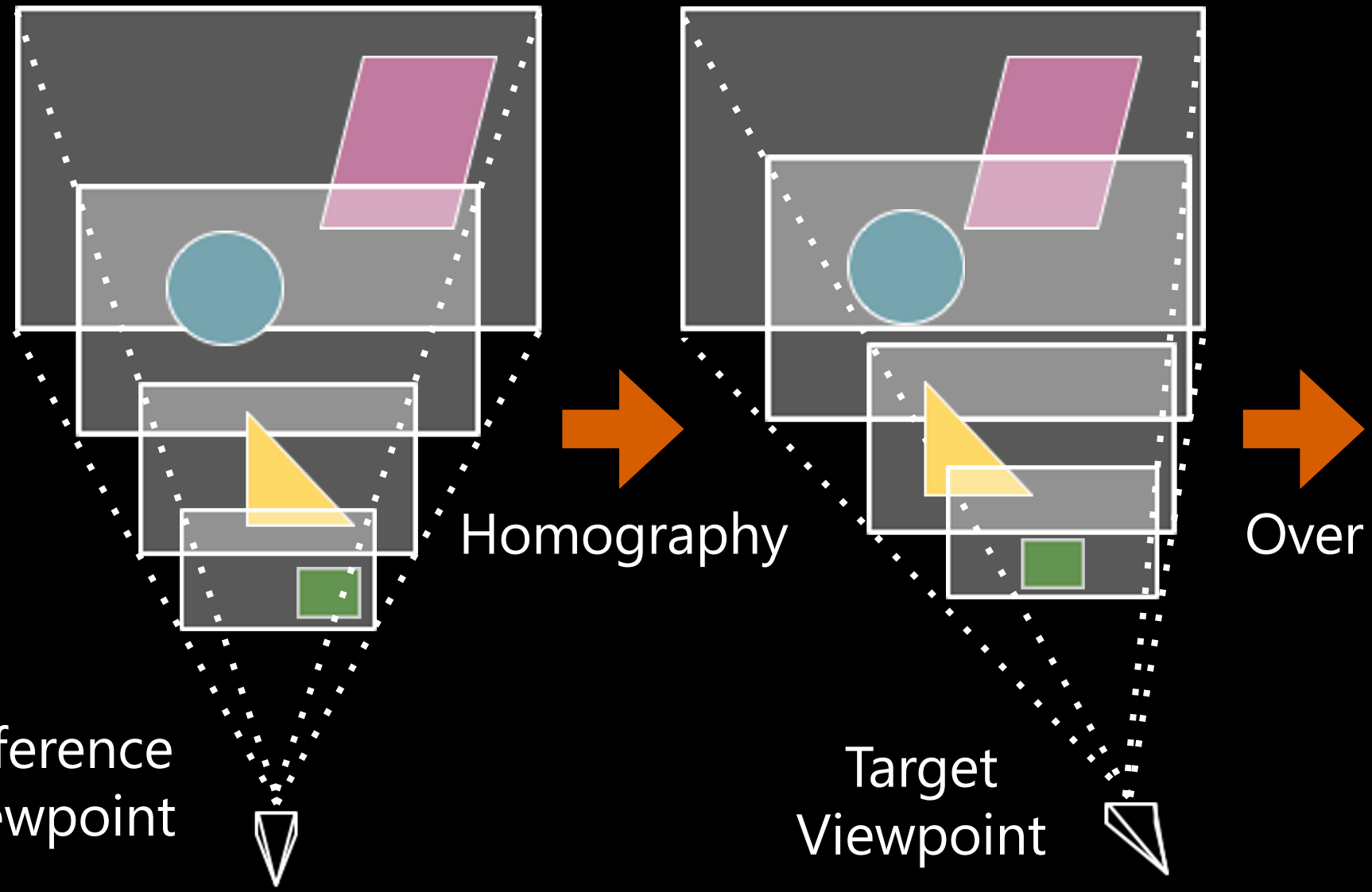


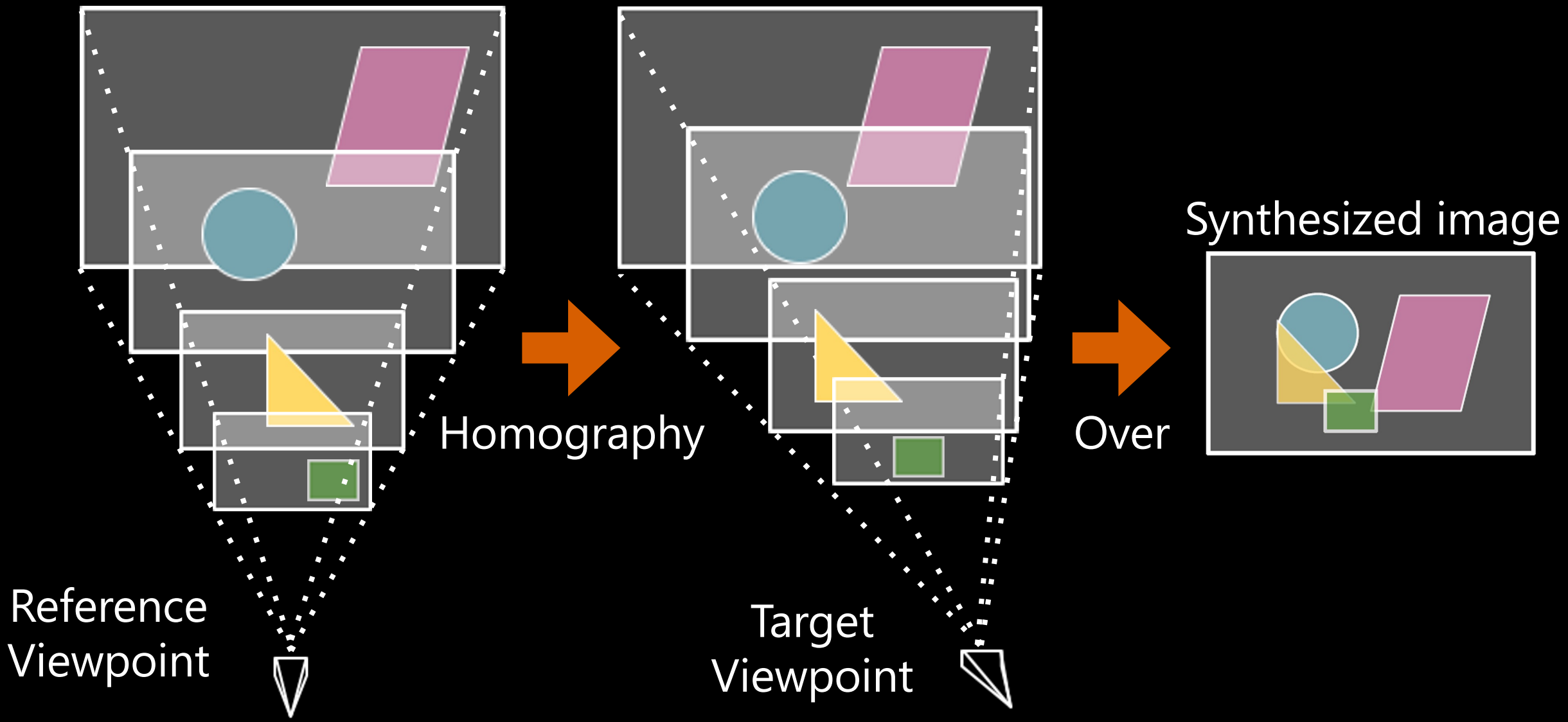
Image credits: Disney

https://www.youtube.com/watch?v=kN-eCBAOw60 (from 1957)

View Synthesis using Multiplane Images

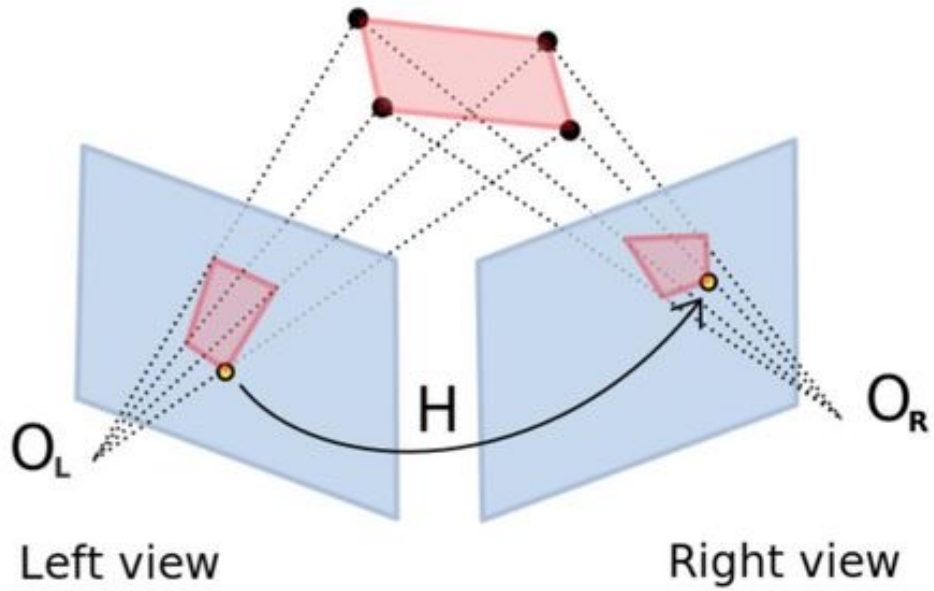


View Synthesis using Multiplane Images

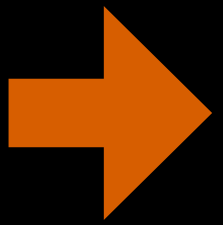


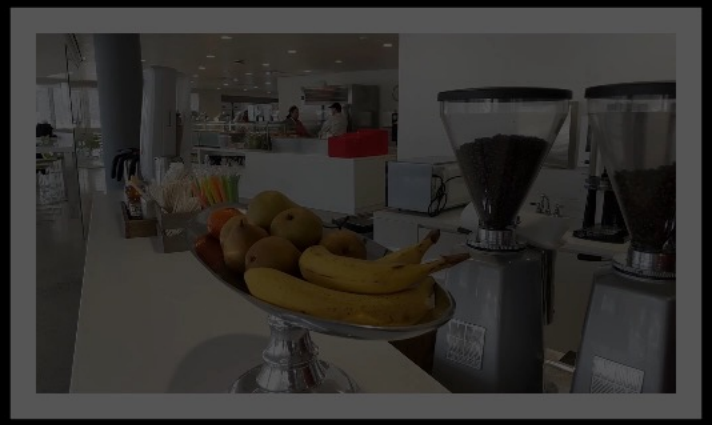
Homography

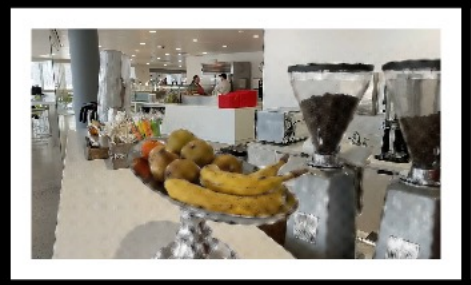
- Transformation relating two images undergoing a rotation about the camera center



- Estimating a 2D homography from a pair of images
  - Fundamental task in computer vision
  - Essential part of monocular SLAM systems
    - Rotation only movements
    - Detecting a planar space
    - Scenes in which objects are very far from the viewer



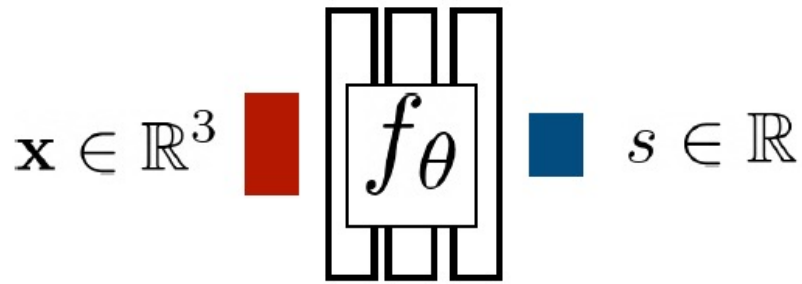




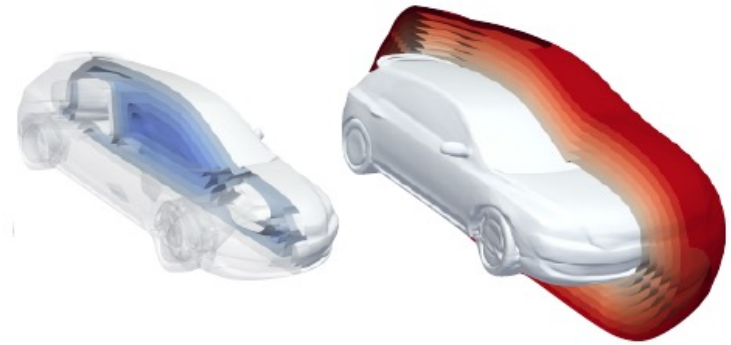
Outline

- Motivation for NeRF: View synthesis
- Introduction Volume Rendering
- Neural Radiance Fields (NeRFs)

Instance-specific SDFs



$$f_\theta : \mathbb{R}^3 \rightarrow \mathbb{R}$$

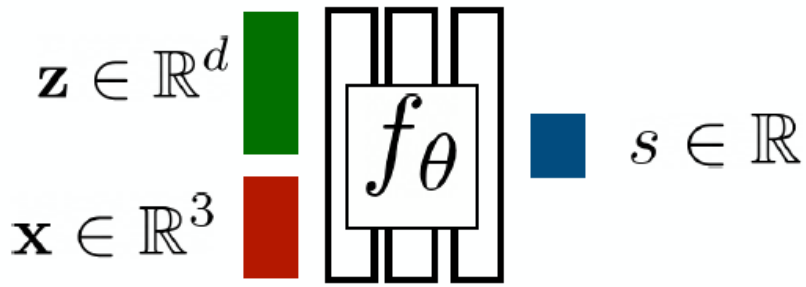


Signed Distance Field (for a single instance):

(position) → (distance)

if 6 layer network with 1000-dim feature space, about 6M parameters per instance!

Latent Conditioning based SDFs



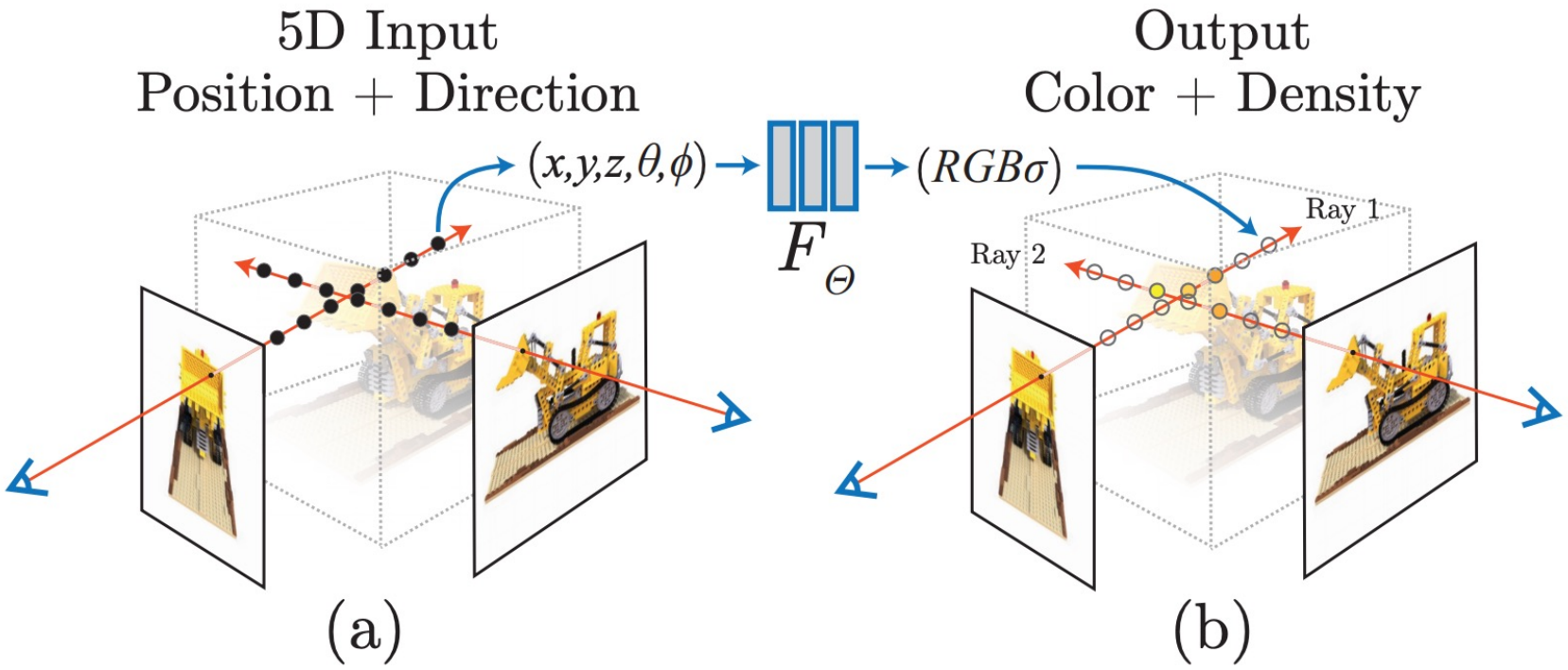
$$f_\theta : \mathbb{R}^3 \times \mathbb{R}^d \rightarrow \mathbb{R}$$

Generalizable Signed Distance Field:
(latent code, position) → (distance)

Each object is represented by a corresponding latent code (only *d* parameters per instance)

The same neural net parameters across **all** objects

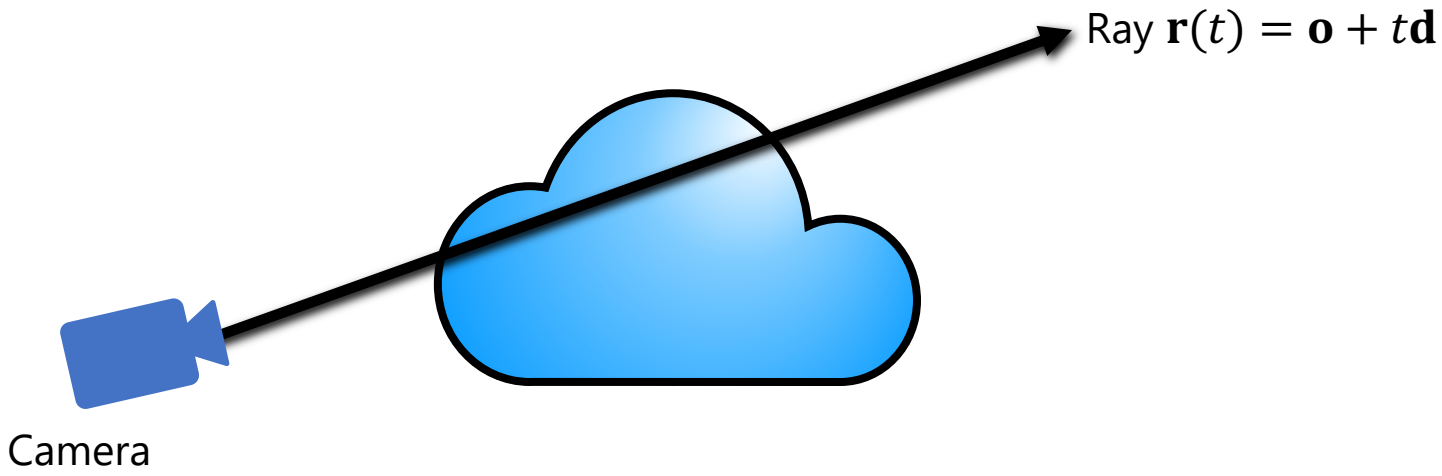
Volume Rendering in NeRF



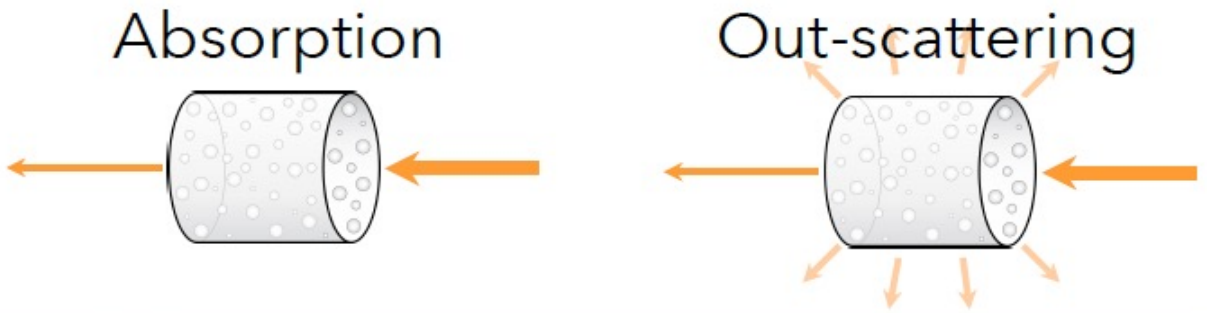
In NeRF, we model a 3D scene with a ‘cloud of tiny colored particles’.

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?



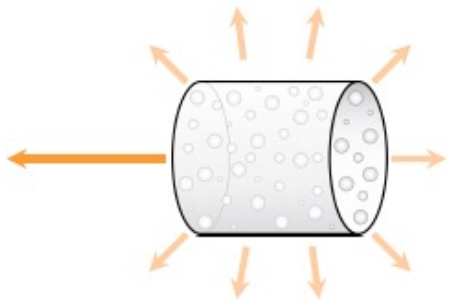
Radiative Transfer Equation



$$dL(\mathbf{x},\vec{\omega}) = \underbrace{-\sigma_a(\mathbf{x})L(\mathbf{x},\vec{\omega})dz \quad -\sigma_s(\mathbf{x})L(\mathbf{x},\vec{\omega})dz}_{\text{Losses}} \underbrace{+\sigma_a(\mathbf{x})L_e(\mathbf{x},\vec{\omega})dz \; +\sigma_s(\mathbf{x})L_s(\mathbf{x},\vec{\omega})dz}_{\text{Gains}}$$

Emission-Absorption Model (Ignoring Scattering)

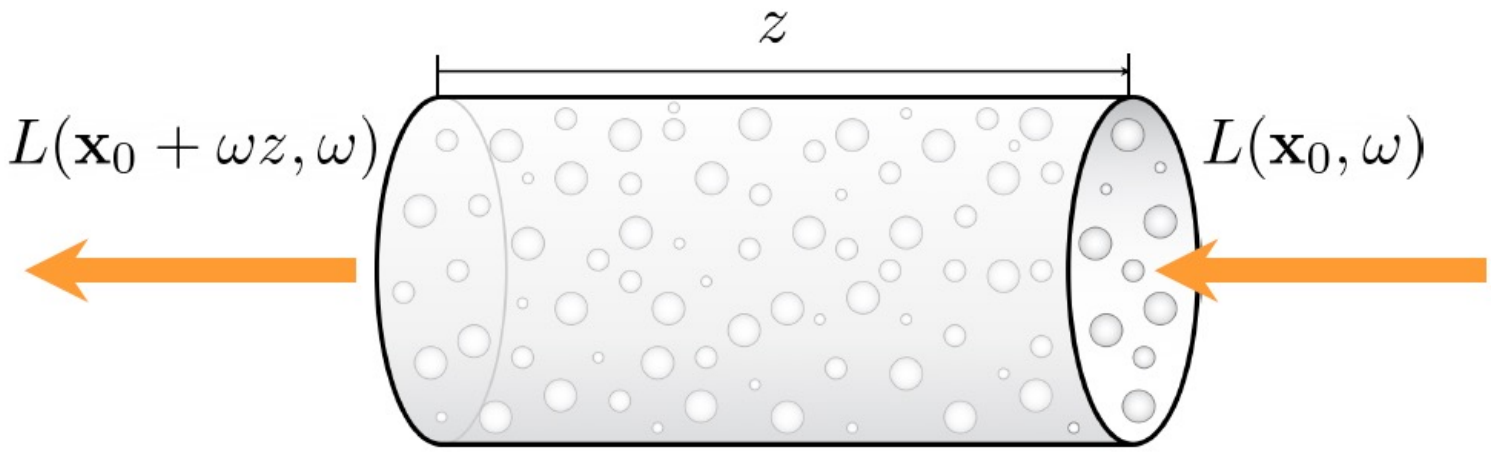
$$dL(\mathbf{x},\vec{\omega}) = -\sigma_a(\mathbf{x})L(\mathbf{x},\vec{\omega})dz \quad -\sigma_s(\mathbf{x})L(\mathbf{x},\vec{\omega})dz$$
$$+\sigma_a(\mathbf{x})L_e(\mathbf{x},\vec{\omega})dz \quad +\sigma_s(\mathbf{x})L_s(\mathbf{x},\vec{\omega})dz$$



Will drop the subscript moving forward

$$\sigma_a \equiv \sigma$$

Absorption-only Volume Rendering



Q: What if we have a homogenous medium? (uniform coefficient)

$$dL(\mathbf{x}, \omega) = -\sigma L(\mathbf{x}, \omega) dz$$

Can you prove this?

$$L(\mathbf{x}_0 + \omega z, \omega) = e^{-\sigma z} L(\mathbf{x}_0, \omega)$$

What if we have a non-homogenous medium?
In non-homogenous medium, coefficient of absorption σ varies with location

$$T(\mathbf{x}, \mathbf{y}) = e^{-\int_{t=0}^{z} \sigma(\mathbf{x}+\omega \mathbf{t}) dt} L(\mathbf{x}_0, \omega)$$

Transmittance

Transmittance

$$T(\mathbf{x},\mathbf{y}) = e^{-\int_{t=0}^{z}\sigma(\mathbf{x}+\omega\mathbf{t})dt} L(\mathbf{x}_0,\omega)$$

Transmittance

$T(\mathbf{x},\mathbf{y})$

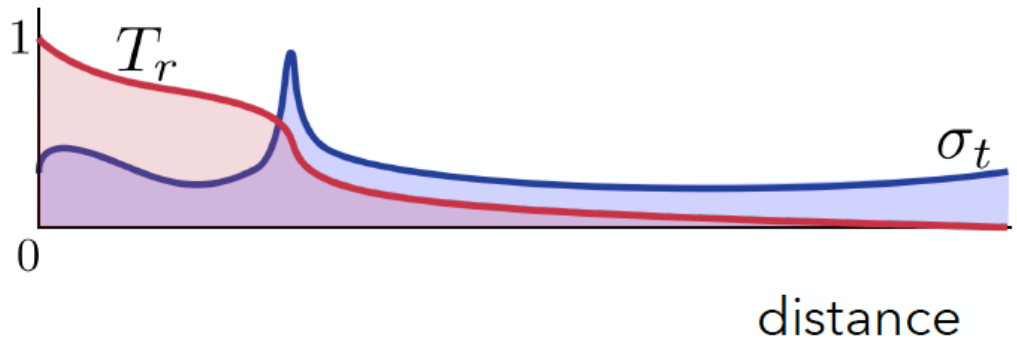
What fraction of radiance at **x** in direction of **y**, reaches **y** ?
(along a straight line under absorption-only model)

Homogenous Medium:

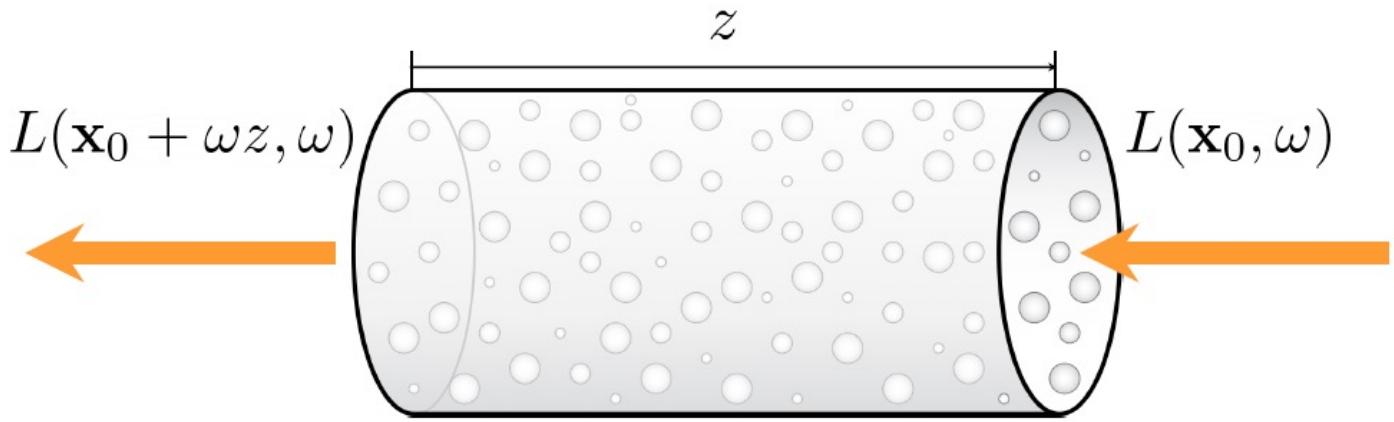
$$e^{-\sigma\|\mathbf{x}-\mathbf{y}\|}$$

Non-Homogenous Medium:

$$e^{-\int_{t=0}^{\|\mathbf{x}-\mathbf{y}\|}\sigma(\mathbf{x}+\omega\mathbf{t})}$$

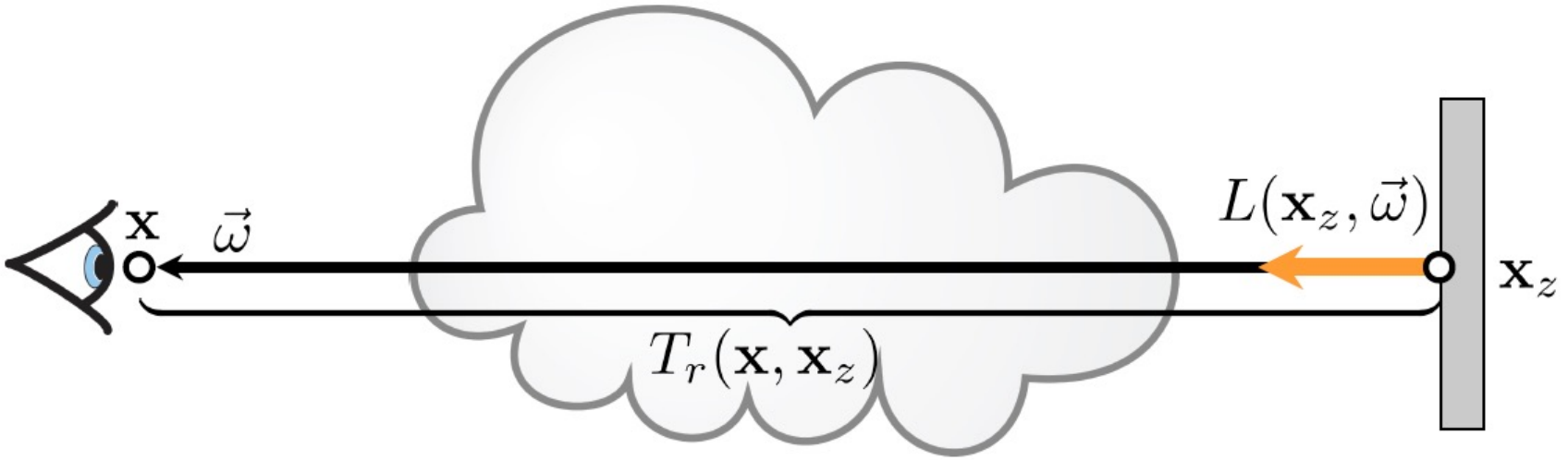


Absorption-only Volume Rendering



$$L(\mathbf{x}_0 + \omega z, \omega) = T(\mathbf{x}_0, \mathbf{x}_0 + \omega z) L(\mathbf{x}_0, \omega)$$

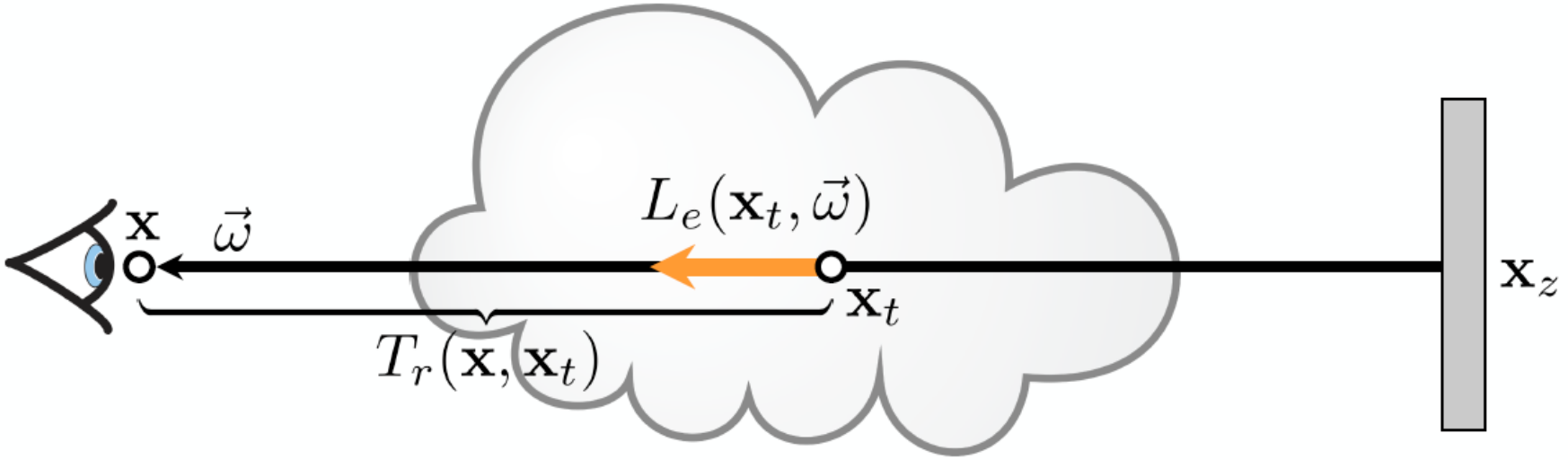
Absorption-only Volume Rendering



$$L(\mathbf{x}, \omega) = T(\mathbf{x}, \mathbf{x}_z) L(\mathbf{x}_z, \omega)$$

Radiance from 'outside' the medium

Emission-Absorption Volume Rendering



$$L(\mathbf{x}, \omega) = T(\mathbf{x}, \mathbf{x}_z) L(\mathbf{x}_z, \omega)$$

$$+ \int_0^z T(\mathbf{x}, \mathbf{x}_t) \sigma(\mathbf{x}_t) L_e(\mathbf{x}_t, \omega) dt$$

Accumulated Emitted Radiance from inside

Emission-Absorption Volume Rendering
Special Case: Homogenous non-emitting medium

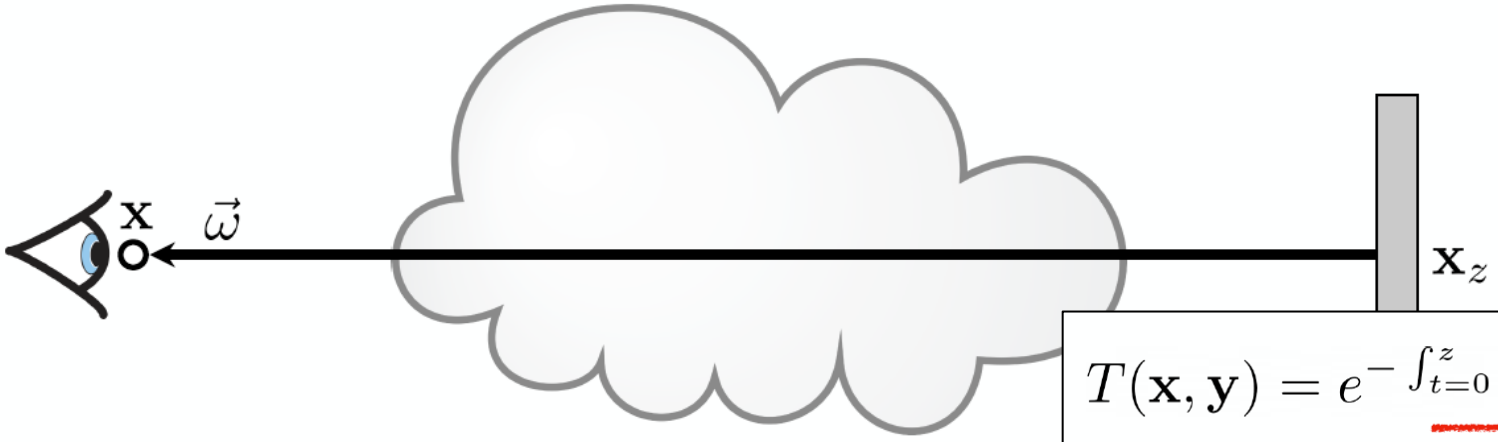
$$T(\mathbf{x}, \mathbf{y}) = e^{-\int_{t=0}^{z} \sigma(\mathbf{x}+\omega \mathbf{t}) dt} L(\mathbf{x}_0, \omega)$$

Transmittance

$$L(\mathbf{x}, \omega) = T(\mathbf{x}, \mathbf{x}_z) L(\mathbf{x}_z, \omega)$$

$$T(\mathbf{x}, \mathbf{x}_z) = e^{-\sigma \Delta}$$

Emission-Absorption Volume Rendering
Special Case: Homogenous emitting medium



$$T(\mathbf{x},\mathbf{y}) = e^{-\int_{t=0}^{z}\sigma(\mathbf{x}+\omega\mathbf{t})dt}L(\mathbf{x}_0,\omega)$$

Transmittance

$$L(\mathbf{x},\omega) = e^{-\sigma\Delta}L(\mathbf{x}_z,\omega) + \int_0^z T(\mathbf{x},\mathbf{x}_t)\sigma(\mathbf{x}_t)L_e(\mathbf{x}_t,\omega)dt$$

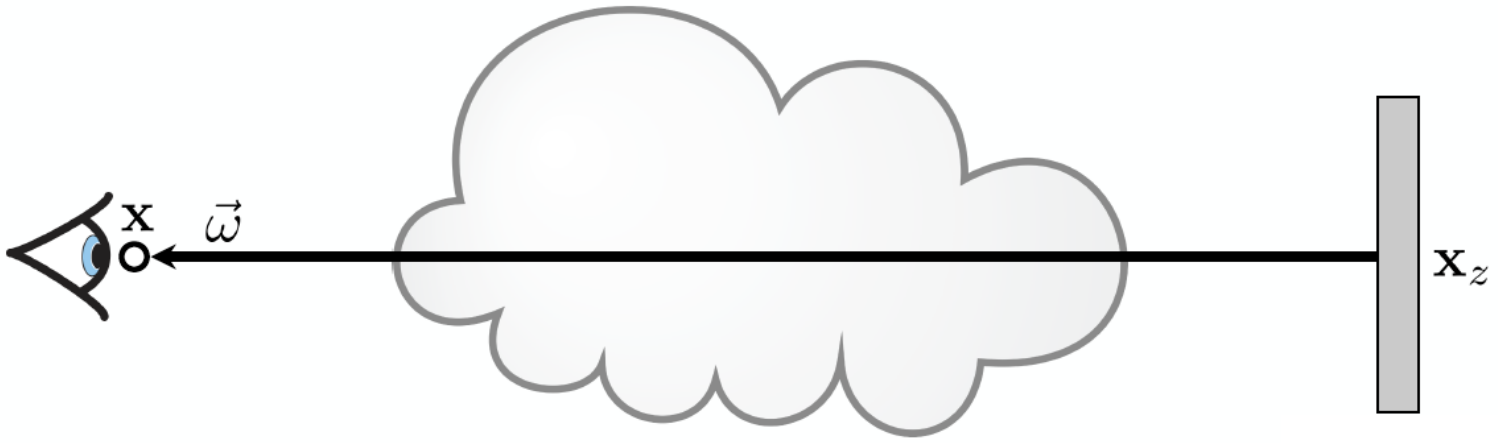
$$L_e(\mathbf{x},\omega) \equiv C$$

$$L(\mathbf{x},\omega) = e^{-\sigma\Delta}L(\mathbf{x}_z,\omega) + \int_{\delta=0}^{\Delta} e^{-\sigma\delta}\sigma C\ d\delta$$

$$\int \lambda e^{-\lambda x}dx = -e^{\lambda x} + c$$

$$L(\mathbf{x},\omega) = e^{-\sigma\Delta}L(\mathbf{x}_z,\omega) + (1-e^{-\sigma\Delta})C$$

Emission-Absorption Volume Rendering
Special Case: Homogenous emitting (only) medium



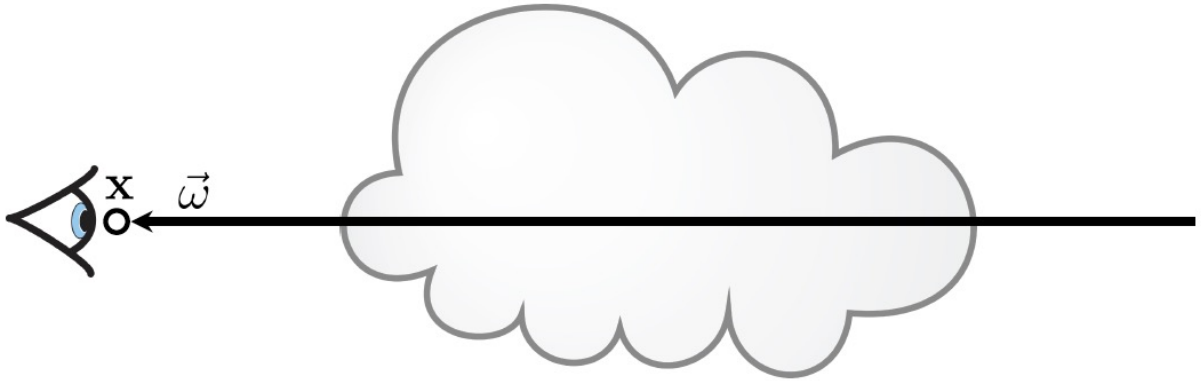
$$L(\mathbf{x},\omega) = \int_0^z T(\mathbf{x},\mathbf{x}_t)\sigma(\mathbf{x}_t)L_e(\mathbf{x}_t,\omega)dt$$

Let's ignore light from outside medium (for now)

$$L(\mathbf{x},\omega) = e^{-\sigma\Delta}L(\mathbf{x}_z,\omega)$$

$$+\int_0^z T(\mathbf{x},\mathbf{x}_t)\sigma(\mathbf{x}_t)L_e(\mathbf{x}_t,\omega)dt$$

$$L(\mathbf{x},\omega) = e^{-\sigma\Delta}L(\mathbf{x}_z,\omega)$$
$$+(1-e^{-\sigma\Delta})C$$



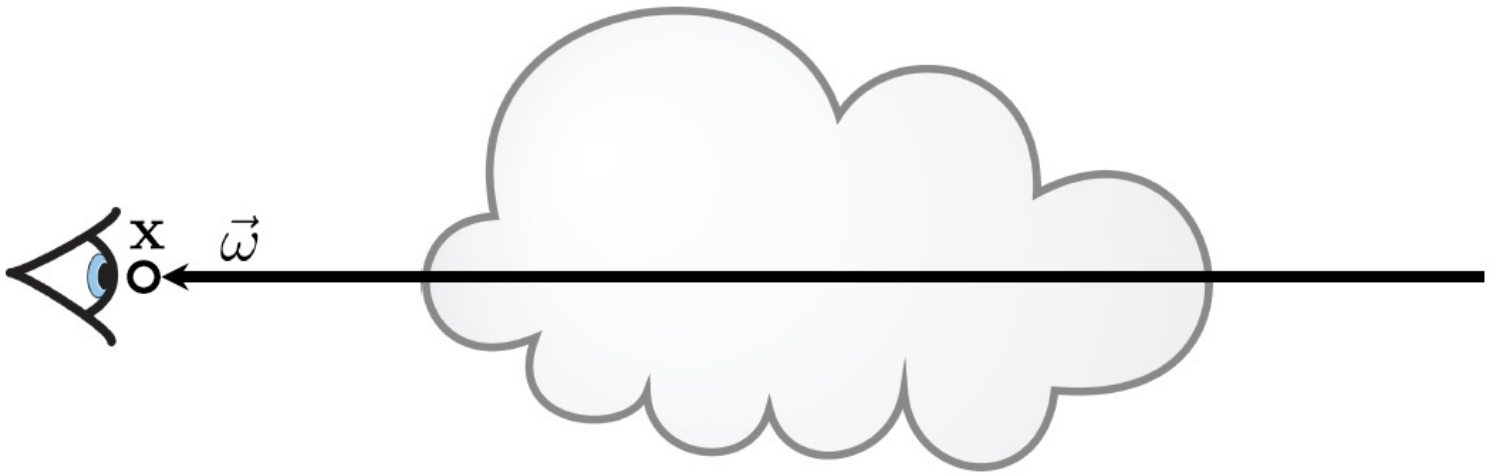
$$L(\mathbf{x},\omega) = (1-e^{-\sigma\Delta})C$$

Emission from a homogenous segment of length Δ

Emission-Absorption Volume Rendering
Special Case: Homogenous emitting (only) medium

Volume Rendering Model to be used in NeRF

Assumption: Ignore light from outside medium.

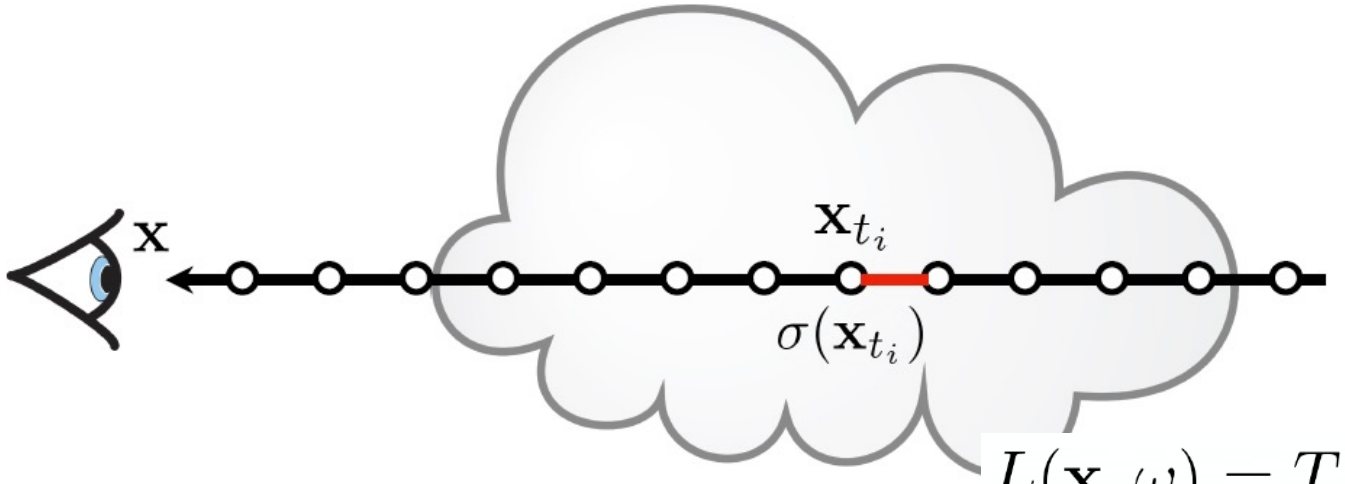


$$L(\mathbf{x}, \omega) = (1 - e^{-\sigma\Delta})C$$

Emission from a homogenous segment of length Δ

For NeRF this means: the object you are trying to render only emits light, There is no lighting being emitted by the background.

Computational Volume Rendering: Ray Marching



$$L(\mathbf{x},\omega) = T(\mathbf{x},\mathbf{x}_z)L(\mathbf{x}_z,\omega)$$

$$L(\mathbf{x},\omega) = \sum_{i=1}^{N} \text{(contribution from i}^{\text{th}}\text{ segment)} \quad \Rightarrow \quad L(\mathbf{x},\omega) = \sum_{i=1}^{N} T(\mathbf{x},\mathbf{x}_{t_i}) \cdot \text{x (emission from i}^{\text{th}}\text{ segment)}$$

Approximate with a discrete sum

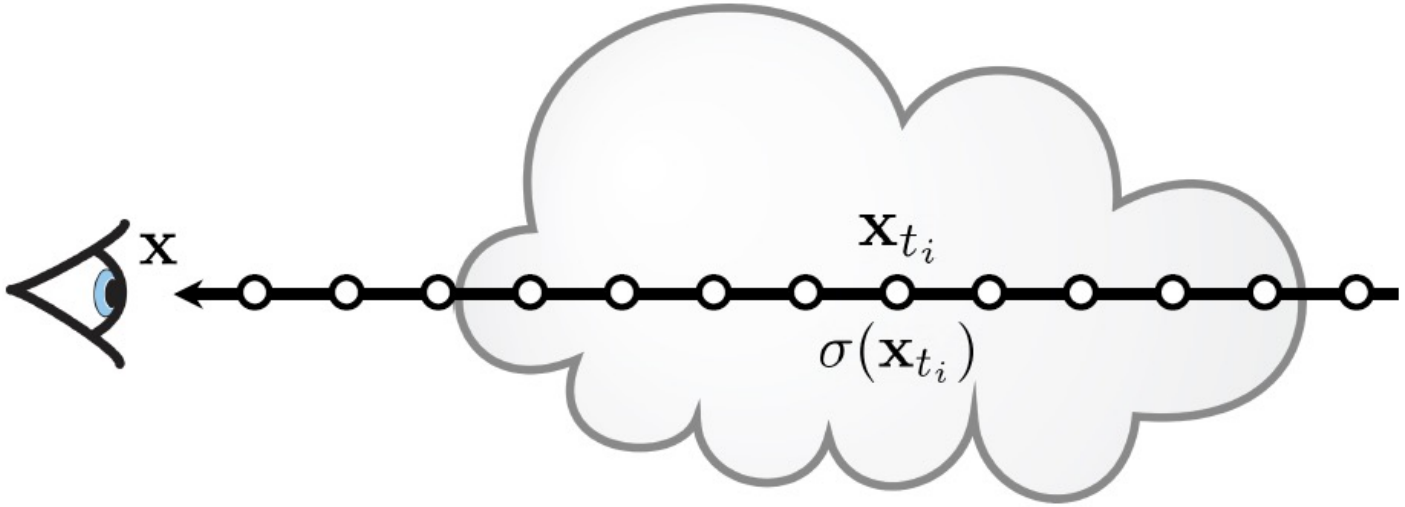
$\mathbf{x}_{t_i}$: i$^{\text{th}}$ sample along ray at depth t_i

Δt : distance between successive samples

$$L(\mathbf{x},\omega) = (1 - e^{-\sigma\Delta})C$$

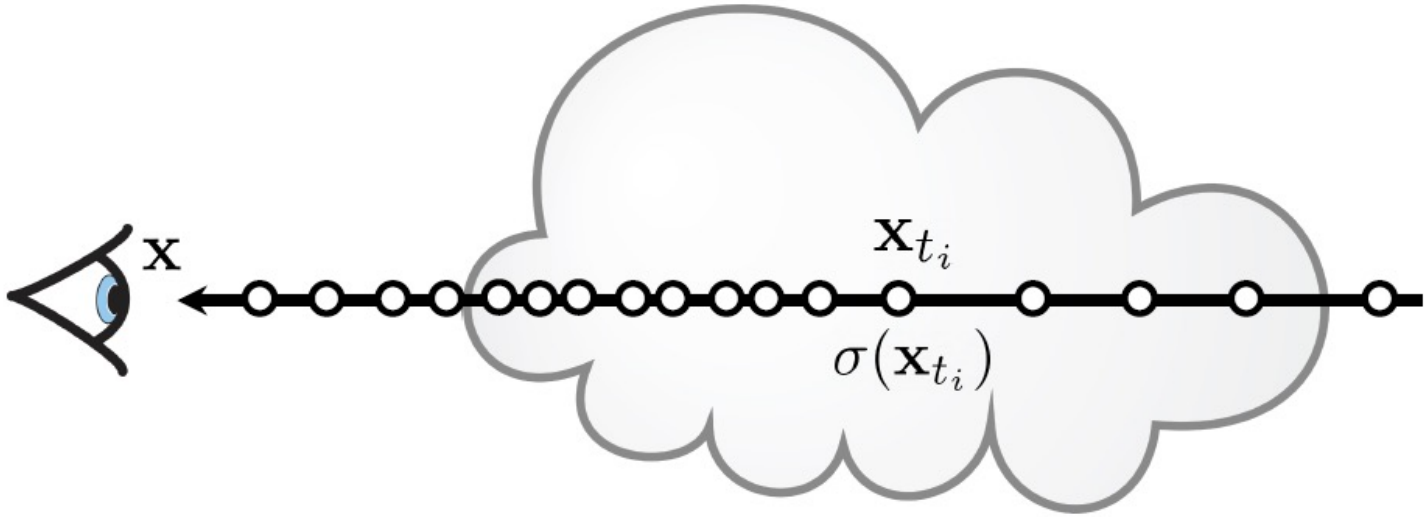
$$L(\mathbf{x},\omega) = \sum_{i=1}^{N} T(\mathbf{x},\mathbf{x}_{t_i}) \cdot (1 - e^{\sigma_{t_i}\Delta t}) L_e(\mathbf{x}_{t_i},\omega)$$

Computational Volume Rendering: Ray Marching



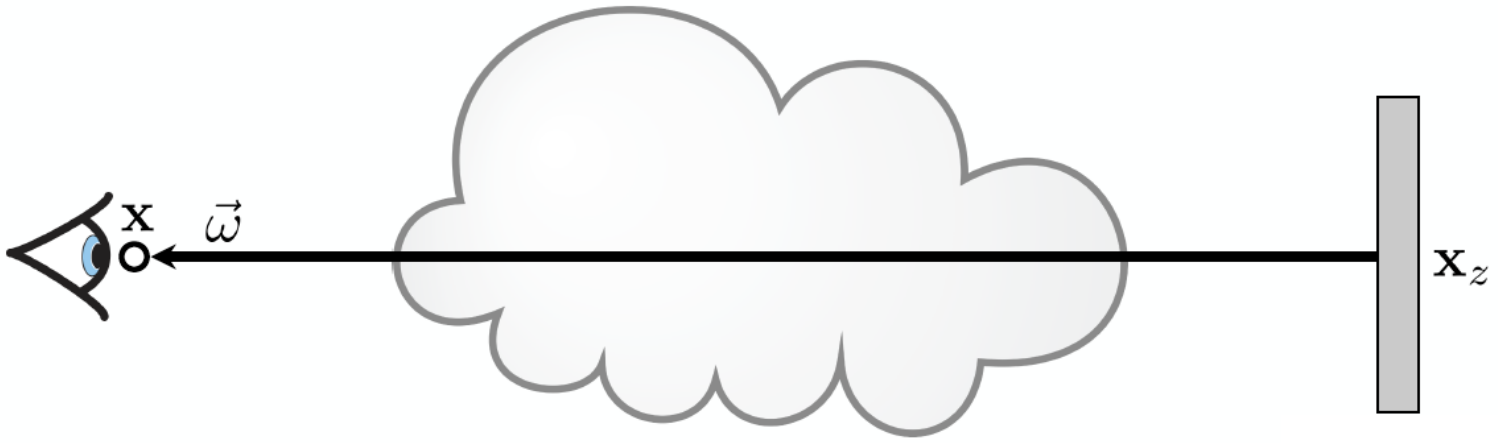
1. Draw uniform samples along a ray (N segments, or N+1 points)
2. Compute transmittance between camera and each sample
3. Aggregate contributions across segments to get overall radiance (color)

Computational Volume Rendering: Ray Marching



1. Draw **non-uniform** samples along a ray
2. Compute transmittance between camera and each sample
3. Aggregate contributions across segments to get overall radiance (color)

Emission-Absorption Volume Rendering
Special Case: Homogenous emitting (only) medium



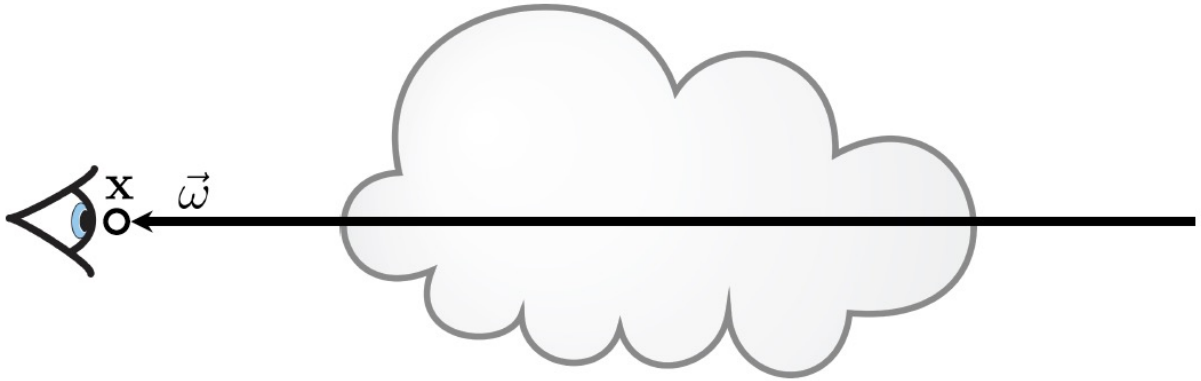
$$L(\mathbf{x},\omega) = \int_0^z T(\mathbf{x},\mathbf{x}_t)\sigma(\mathbf{x}_t)L_e(\mathbf{x}_t,\omega)dt$$

Let's ignore light from outside medium (for now)

$$L(\mathbf{x},\omega) = e^{-\sigma\Delta}L(\mathbf{x}_z,\omega)$$

$$+\int_0^z T(\mathbf{x},\mathbf{x}_t)\sigma(\mathbf{x}_t)L_e(\mathbf{x}_t,\omega)dt$$

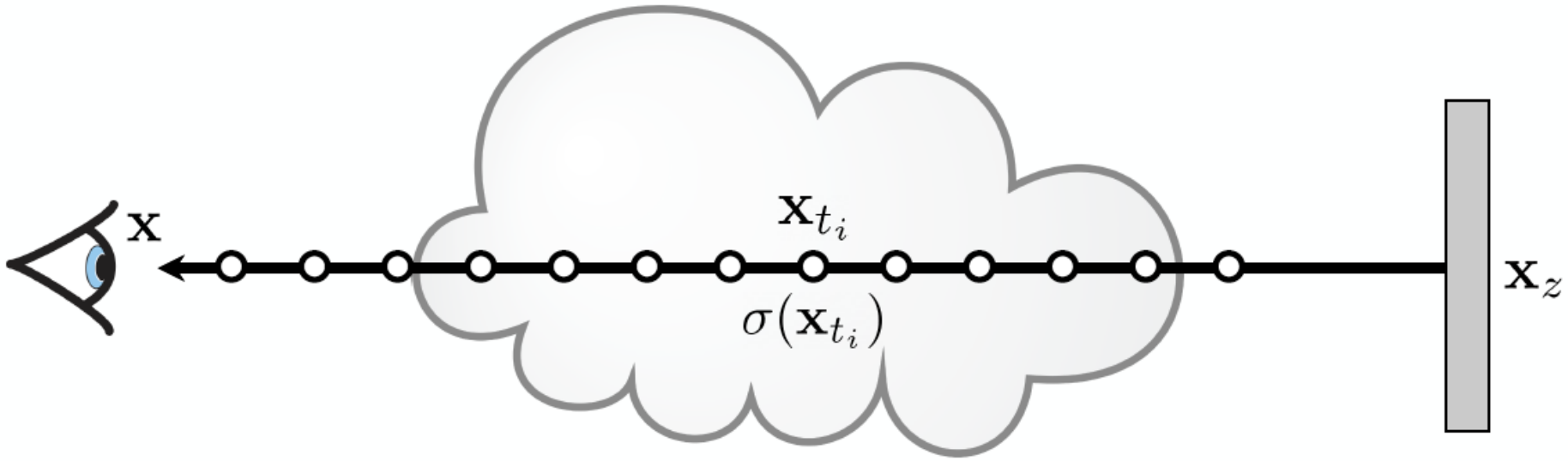
$$L(\mathbf{x},\omega) = e^{-\sigma\Delta}L(\mathbf{x}_z,\omega)$$
$$+(1-e^{-\sigma\Delta})C$$



$$L(\mathbf{x},\omega) = (1-e^{-\sigma\Delta})C$$

Emission from a homogenous segment of length Δ

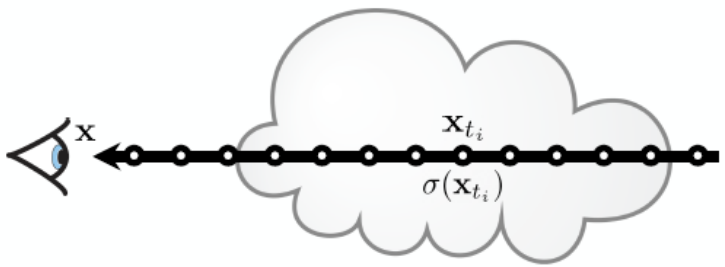
Enabling Background Radiance



$$L(\mathbf{x},\omega) = \sum_{i=1}^{N} T(\mathbf{x},\mathbf{x}_{t_i})(1-e^{-\sigma_{t_i}\Delta t})L_e(\mathbf{x}_{t_i},\omega)$$

$$T(\mathbf{x},\mathbf{x}_{t_i}) = T(\mathbf{x},\mathbf{x}_{t_{i-1}})e^{-\sigma_{t_{i-1}}\Delta t}$$

Computational Volume Rendering: A summary



$$L(\mathbf{x},\omega) = \sum_{i=1}^{N} T(\mathbf{x},\mathbf{x}_{t_i})(1 - e^{\sigma_{t_i}\Delta t})L_e(\mathbf{x}_{t_i},\omega)$$

$$\sigma_{t_i} \equiv \sigma(\mathbf{x}_{t_i})$$

$$L_e(\mathbf{x}_{t_i},\omega)$$

$$\longrightarrow \quad L(\mathbf{x},\omega)$$

If we can compute:

a) (per-point) density

b) (per-point, direction) emitted light,

we can render **any** ray through the medium

Equivalently, we can render an image from any camera viewpoint (using H*W rays)

Note: **Differentiable** process w.r.t. the **density**, **emitted light**

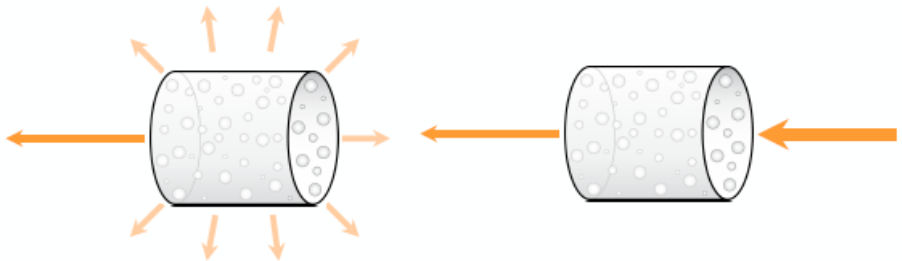
and also camera parameters if density, emission are differentiable functions of position, direction

Volume Rendering in NeRF

$$L(\mathbf{x}, \omega) = \sum_{i=1}^{N} T(\mathbf{x}, \mathbf{x}_{t_i}) \cdot (1 - e^{\sigma_{t_i} \Delta t}) L_e(\mathbf{x}_{t_i}, \omega)$$

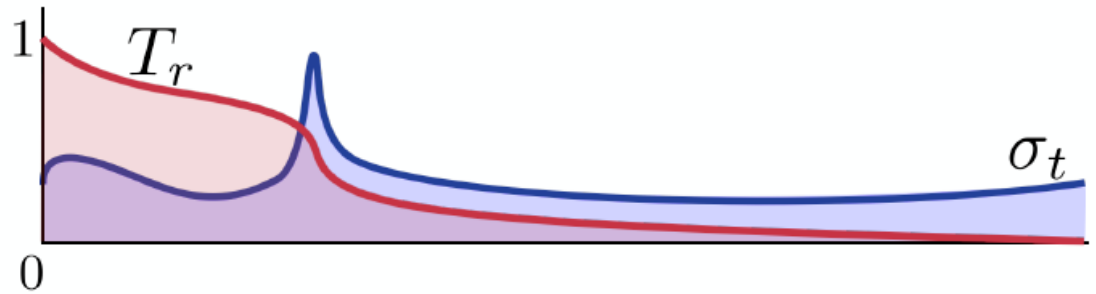
$L_e(.)$ = RGB color of cloud of tiny particles.
σ = density of tiny colored particles

Summary

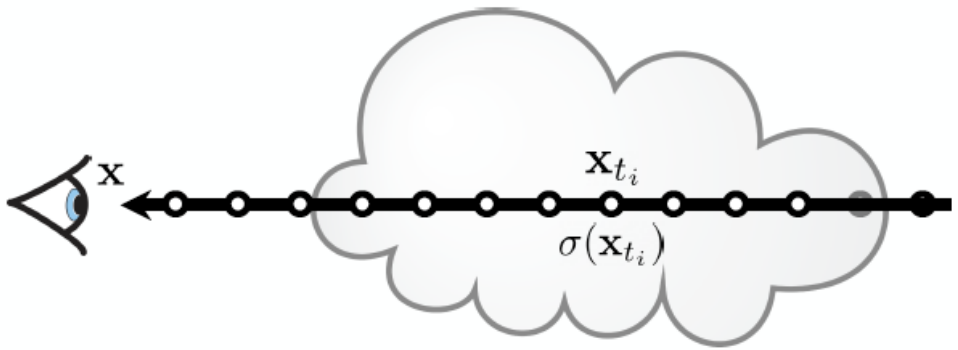


$$dL(\mathbf{x},\vec{\omega}) = -\sigma_a(\mathbf{x})L(\mathbf{x},\vec{\omega})dz + \sigma_a(\mathbf{x})L_e(\mathbf{x},\vec{\omega})dz$$

Emission-Absorption Model



Transmittance



$$L(\mathbf{x},\omega) = \sum_{i=1}^{N} T(\mathbf{x},\mathbf{x}_{t_i})\cdot(1-e^{\sigma_{t_i}\Delta t})L_e(\mathbf{x}_{t_i},\omega)$$

$$T(\mathbf{x},\mathbf{x}_{t_i}) = T(\mathbf{x},\mathbf{x}_{t_{i-1}})e^{-(\sigma_{t_i}\Delta t)}$$

A computational algorithm

Outline

- Motivation for NeRF: View synthesis
- Introduction Volume Rendering
- Neural Radiance Fields (NeRFs)

NeRF == Differentiable Rendering with a **Neural Volumetric Representation**

Barron et al 2021, Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields

Neural Volumetric Rendering

Neural Volumetric Rendering

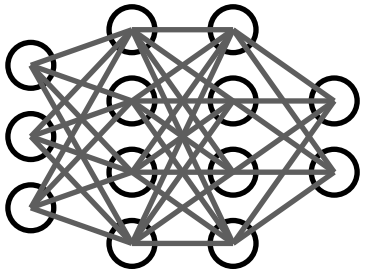
querying the radiance value
along rays through 3D space

Neural Volumetric Rendering

continuous, differentiable
rendering model without
concrete ray/surface intersections

Neural Volumetric Rendering

using a neural network as a scene representation, rather than a voxel grid of data

(x, y, z) →  → Scene properties

Multi-layer Perceptron (Neural Network)

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

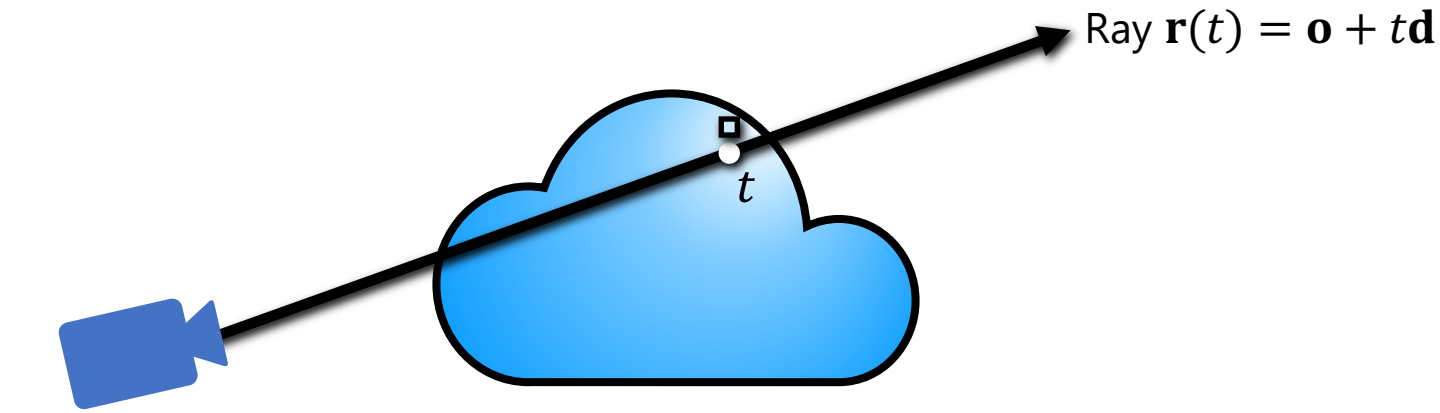
3D reconstruction viewable from any angle

Volumetric formulation for NeRF

Scene is a cloud of colored fog

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

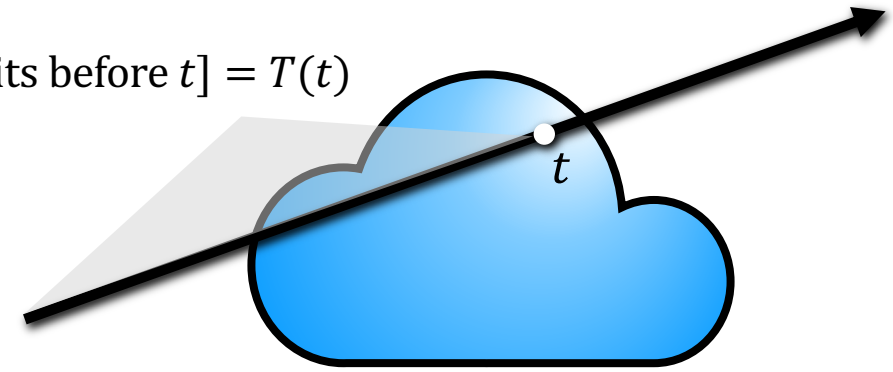
Volumetric formulation for NeRF



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

$P[\text{no hits before } t] = T(t)$



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

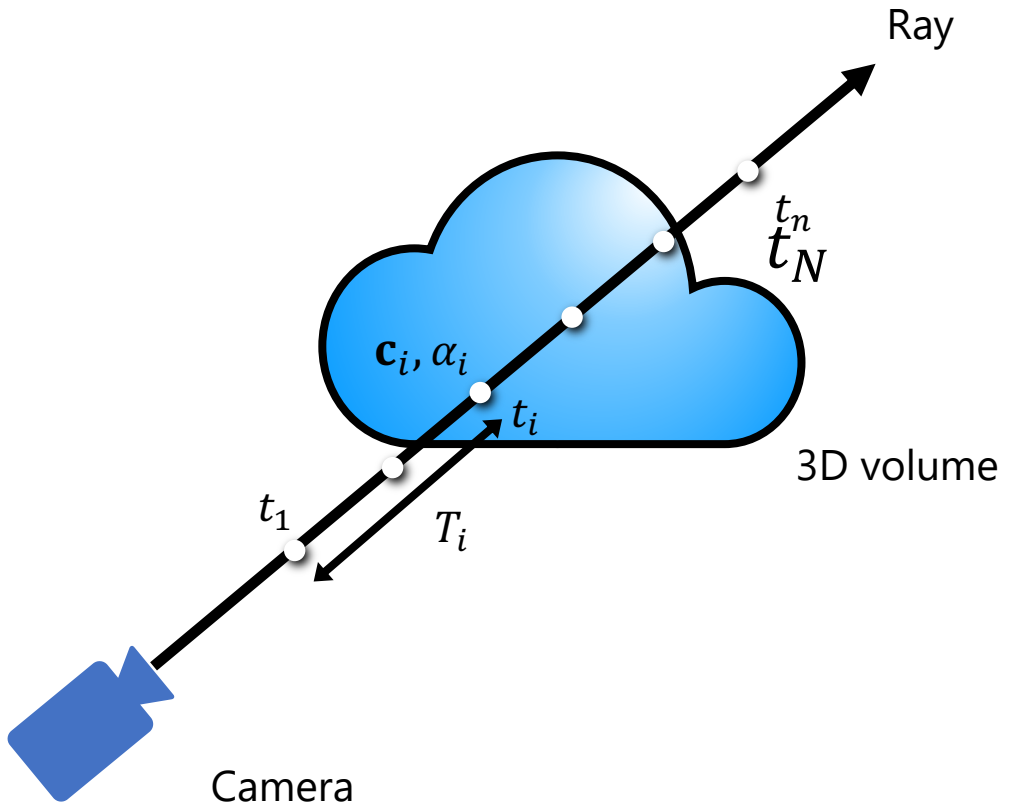
Rendering model for ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

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

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

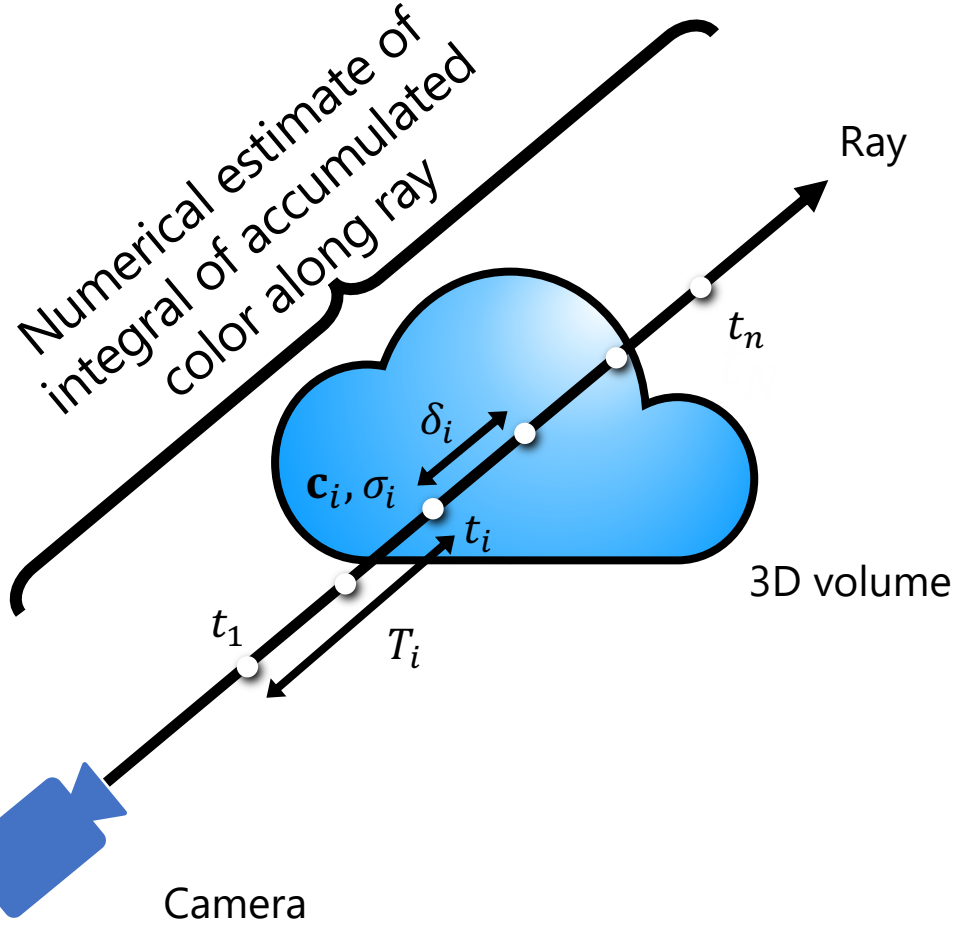
Rendering model for ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

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

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

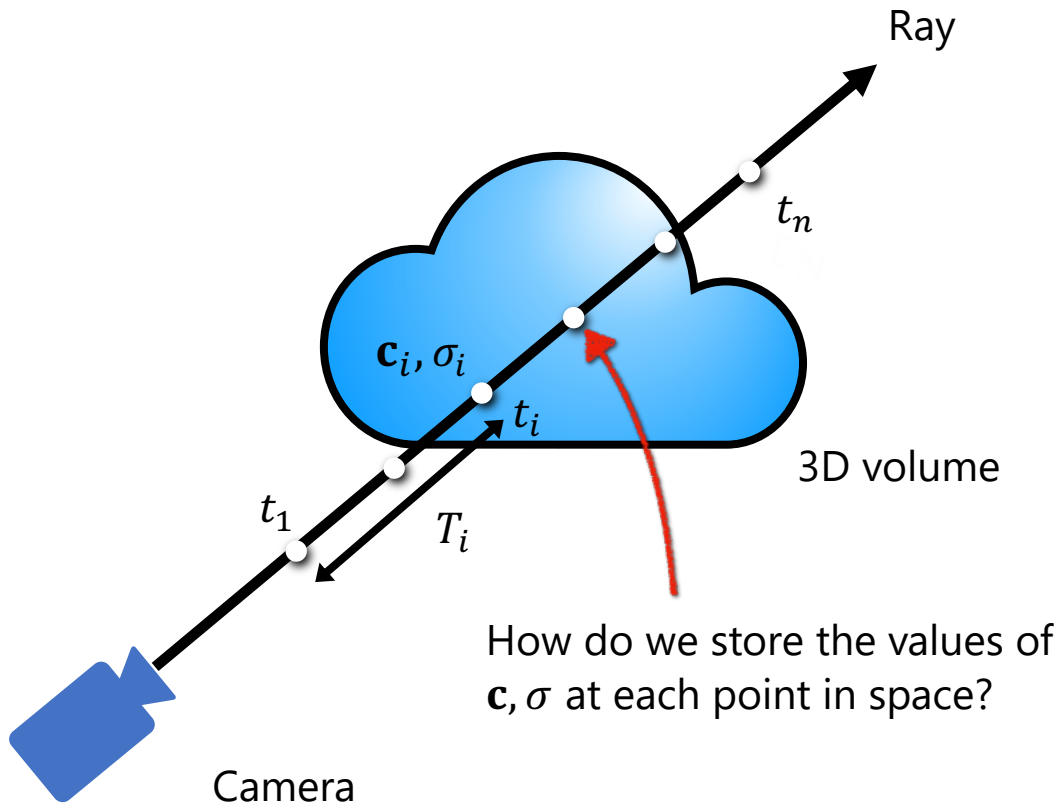
Rendering model for ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

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

final rendered color along ray

weights

colors



How much light is blocked earlier along ray:

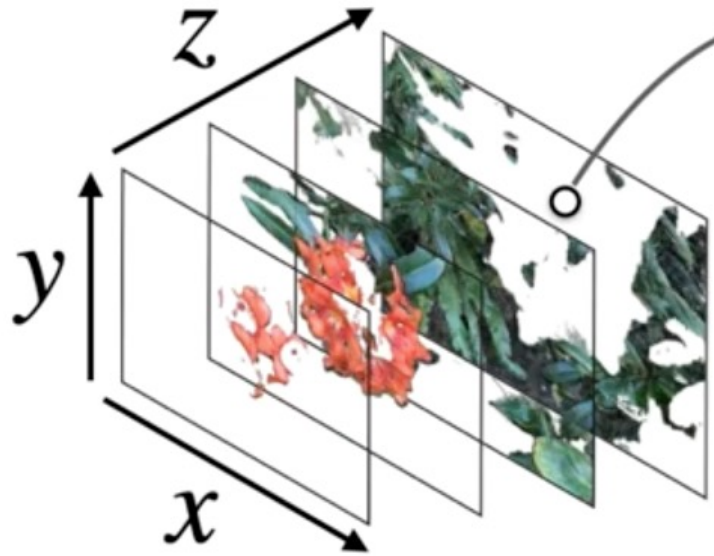
$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

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

(Recap) Volume Rendering in NeRF

$$L(\mathbf{x}, \omega) = \sum_{i=1}^{N} T(\mathbf{x}, \mathbf{x}_{t_i}) \cdot (1 - e^{\sigma_{t_i} \Delta t}) L_e(\mathbf{x}_{t_i}, \omega)$$

$L_e(.)$ = RGB color of cloud of tiny particles.
σ = density of tiny colored particles



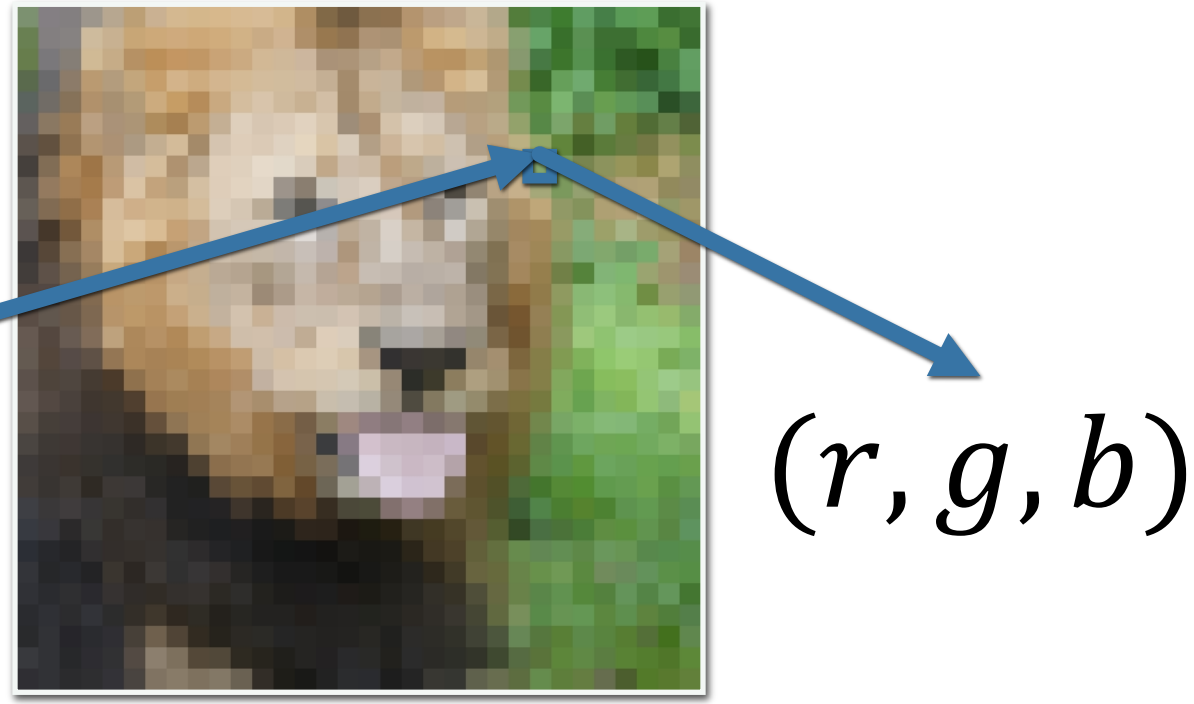
(r, g, b, σ)

versus

(x, y, z, θ, ϕ) →  → (r, g, b, σ)

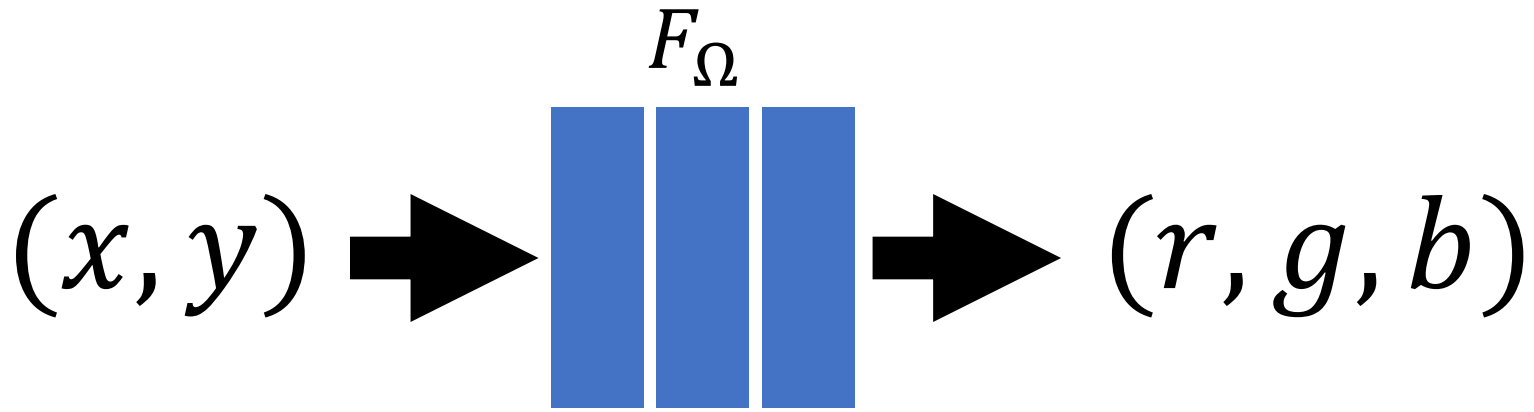
F_Ω

Toy problem: storing 2D image data



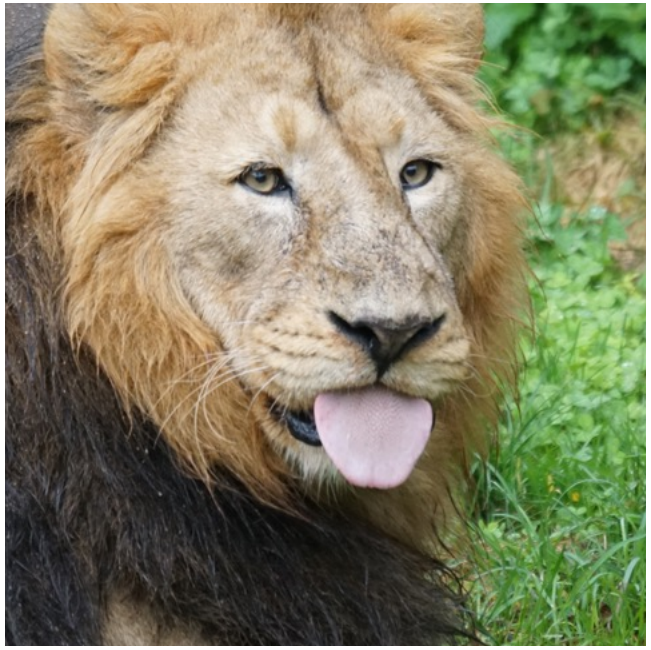
Usually we store an image as a
2D grid of RGB color values

Toy problem: storing 2D image data



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

Naive approach fails!



Ground truth image

Neural network output fit
with gradient descent

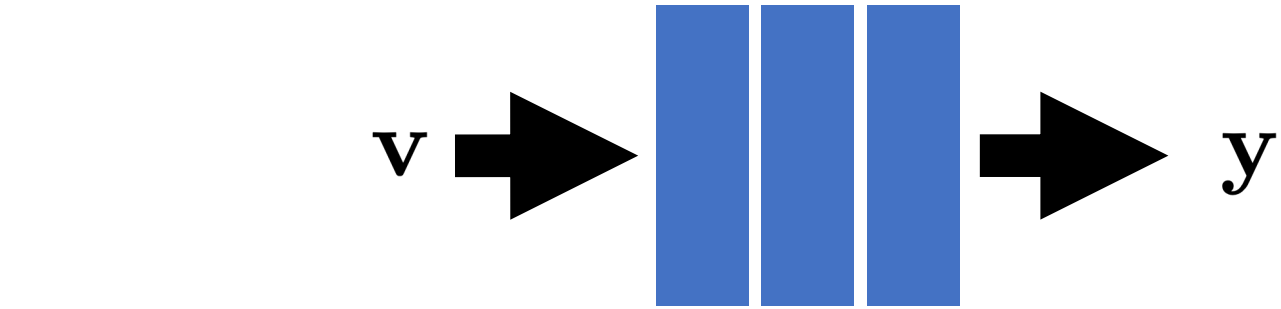
Problem:

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

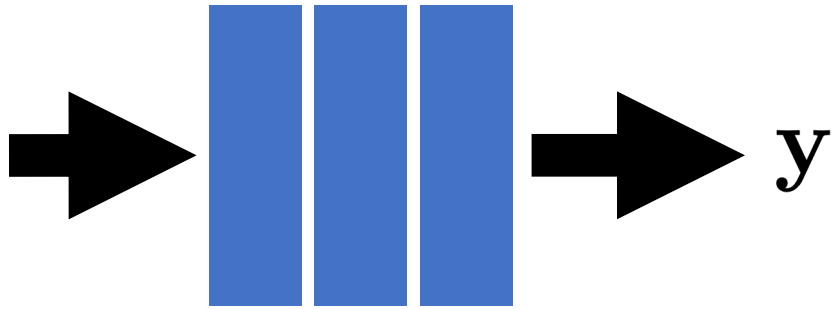
Solution:

- Pass input coordinates through a high frequency mapping first.

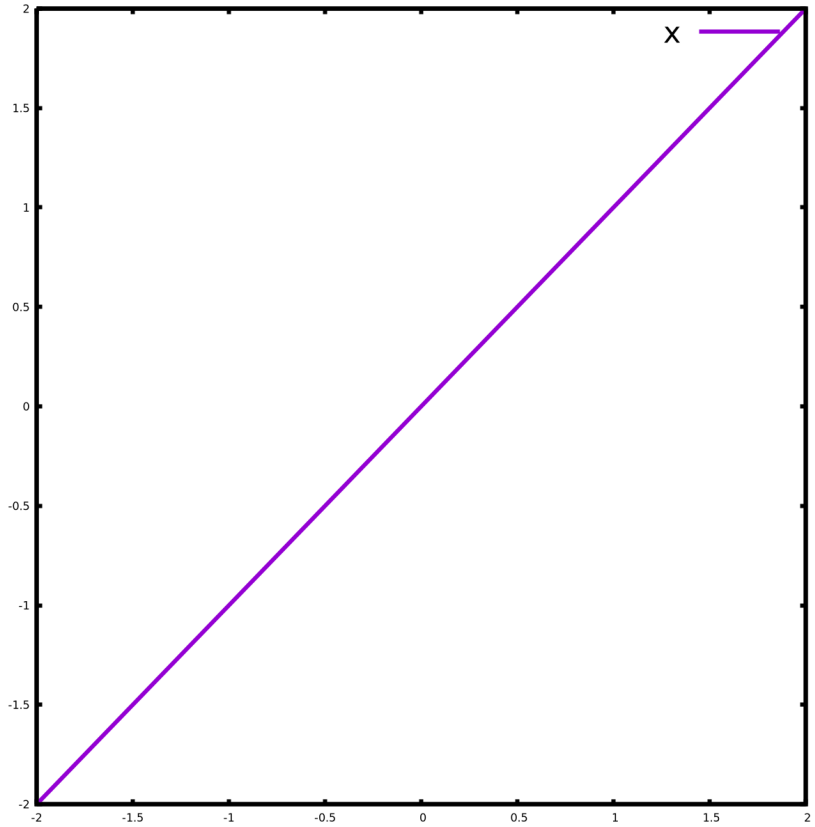
Example mapping: “positional encoding”



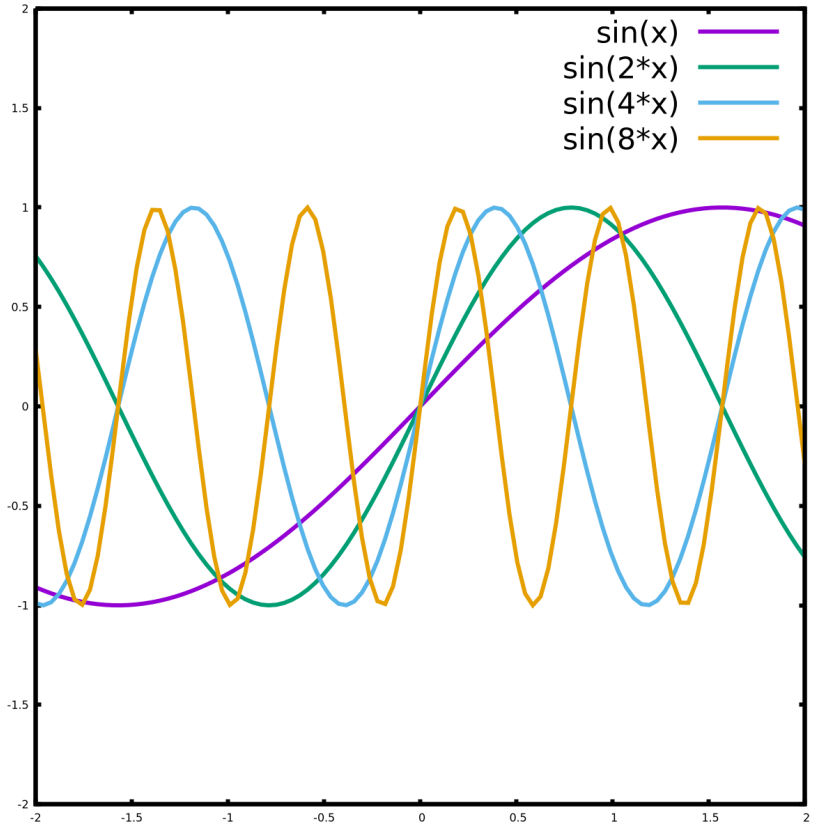
$$\begin{pmatrix} \sin(\mathbf{v}), \cos(\mathbf{v}) \\ \sin(2\mathbf{v}), \cos(2\mathbf{v}) \\ \sin(4\mathbf{v}), \cos(4\mathbf{v}) \\ \dots \\ \sin(2^{L-1}\mathbf{v}), \cos(2^{L-1}\mathbf{v}) \end{pmatrix}$$



Positional encoding

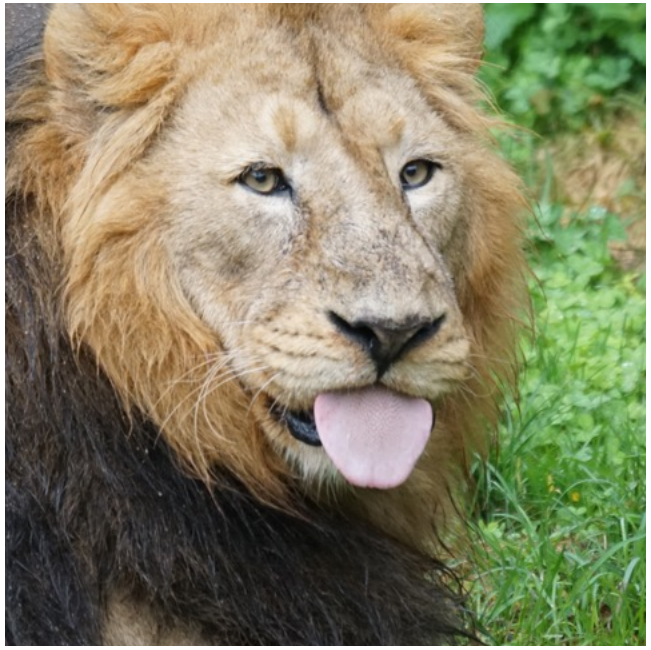


Raw encoding of a number ***x***



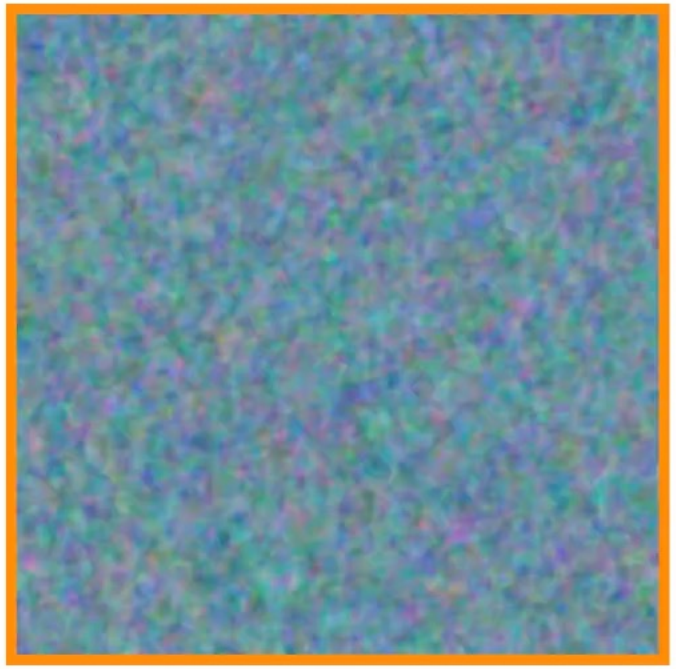
"Positional encoding" of a number ***x***

Problem solved!



Ground truth image

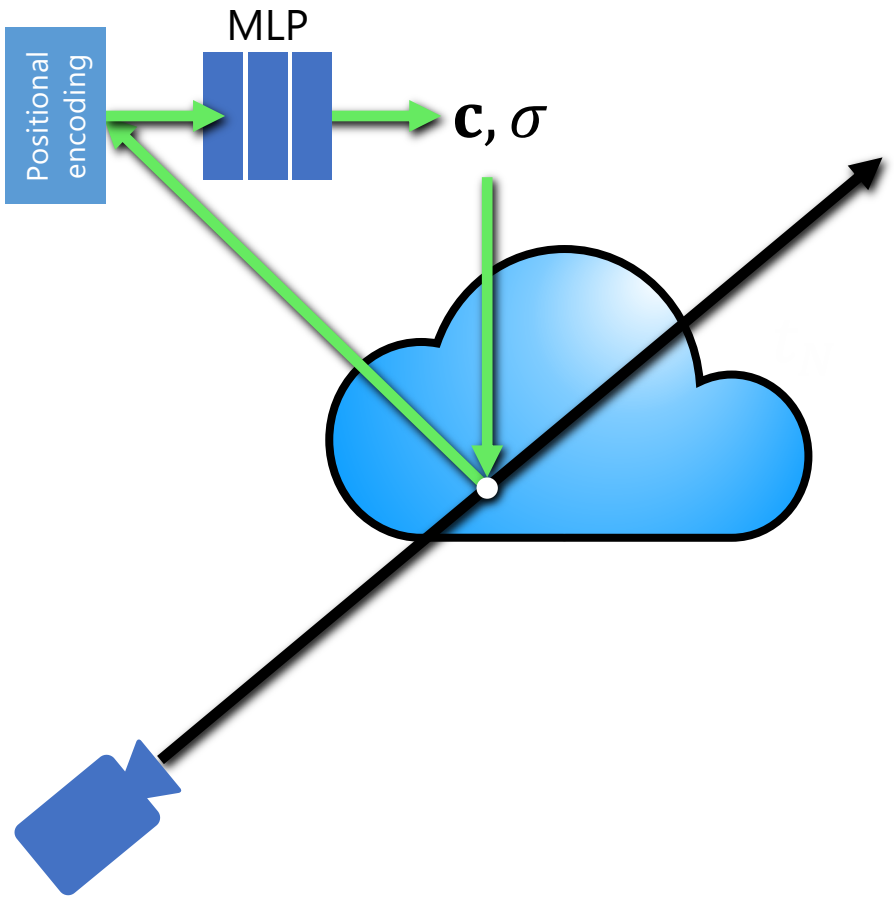
Neural network output without high frequency mapping



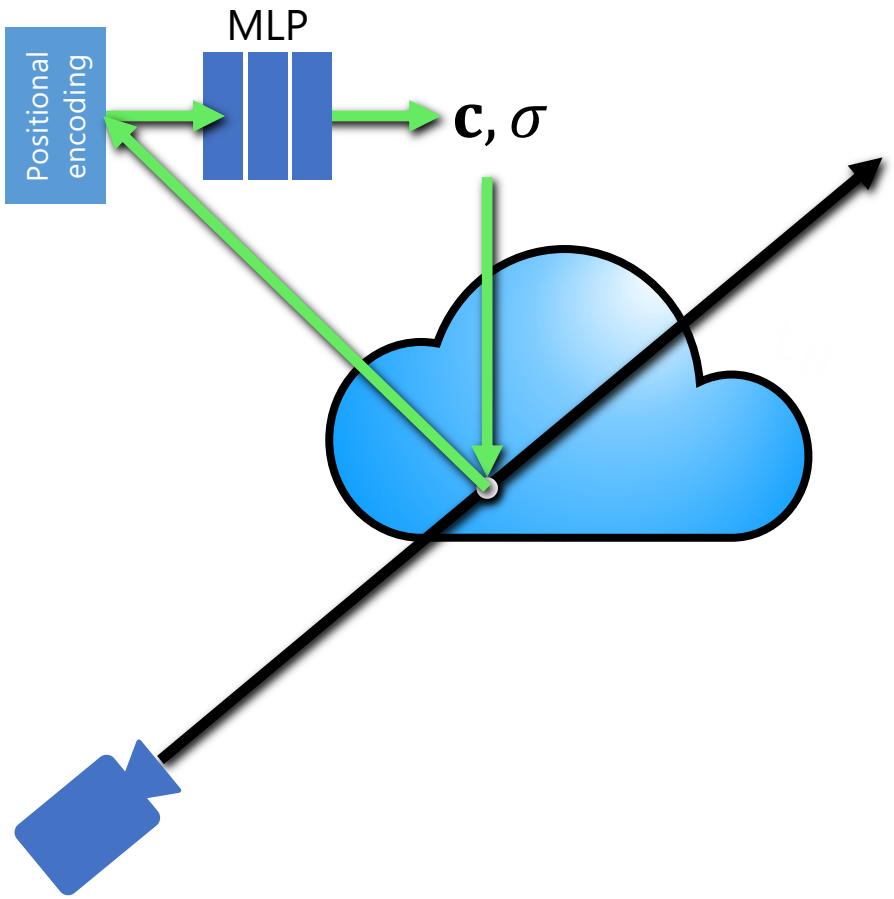
Neural network output with high frequency mapping

NeRF = volume rendering + coordinate-based network

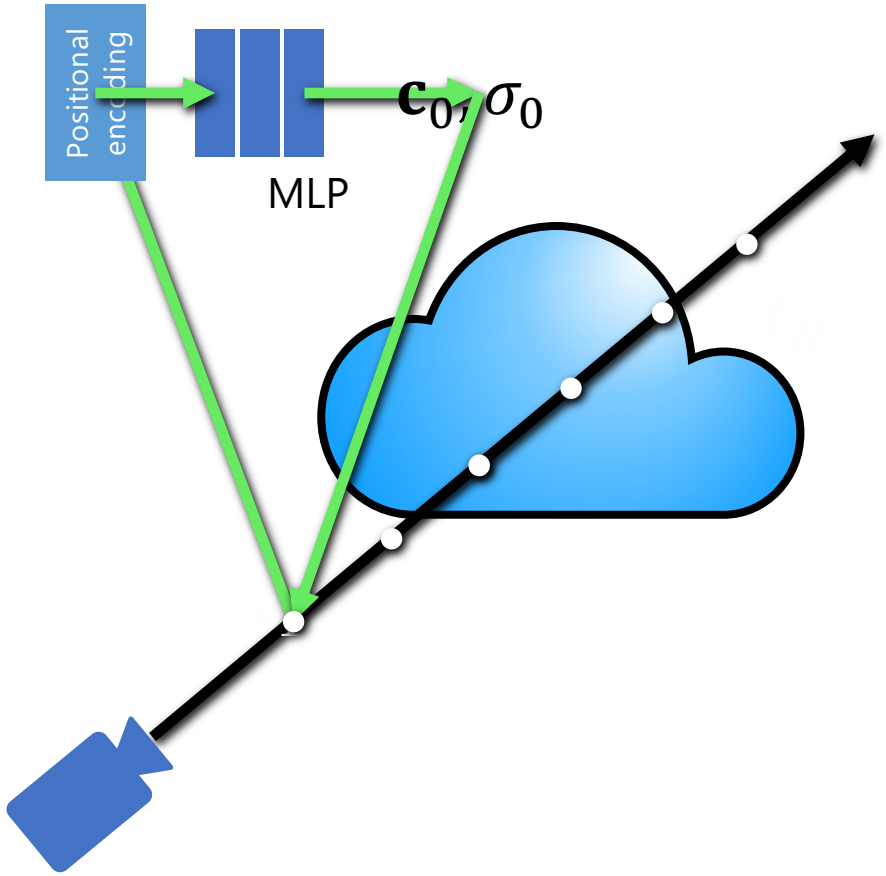
$\mathbf{c}, \sigma$



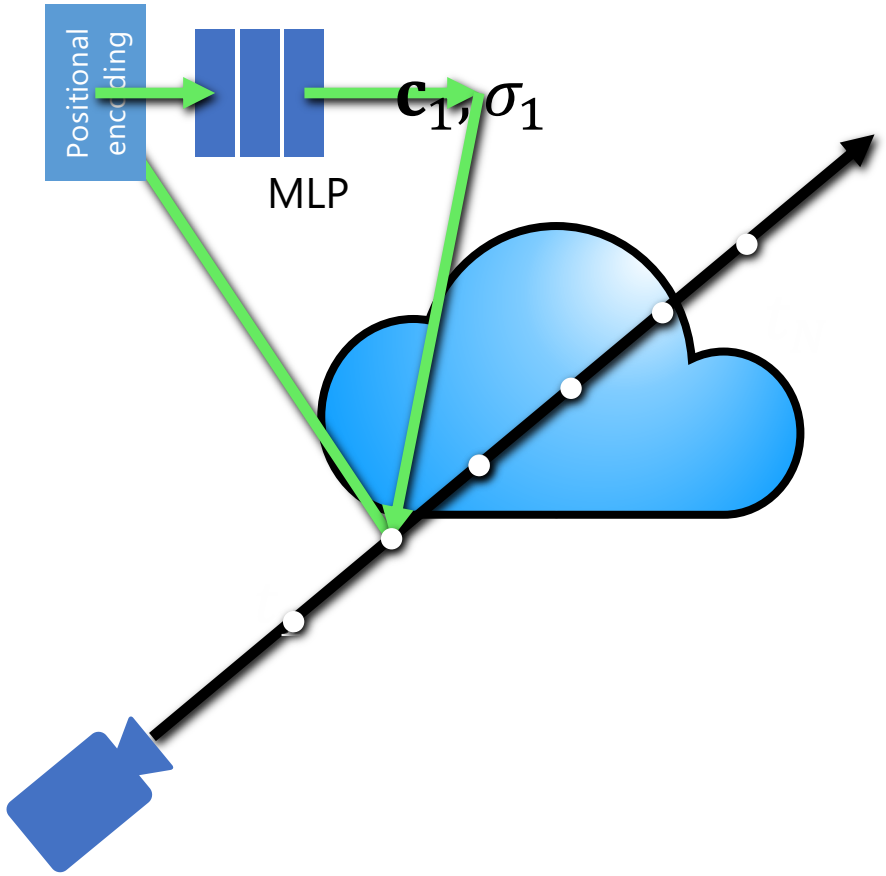
$\mathbf{c}, \sigma$



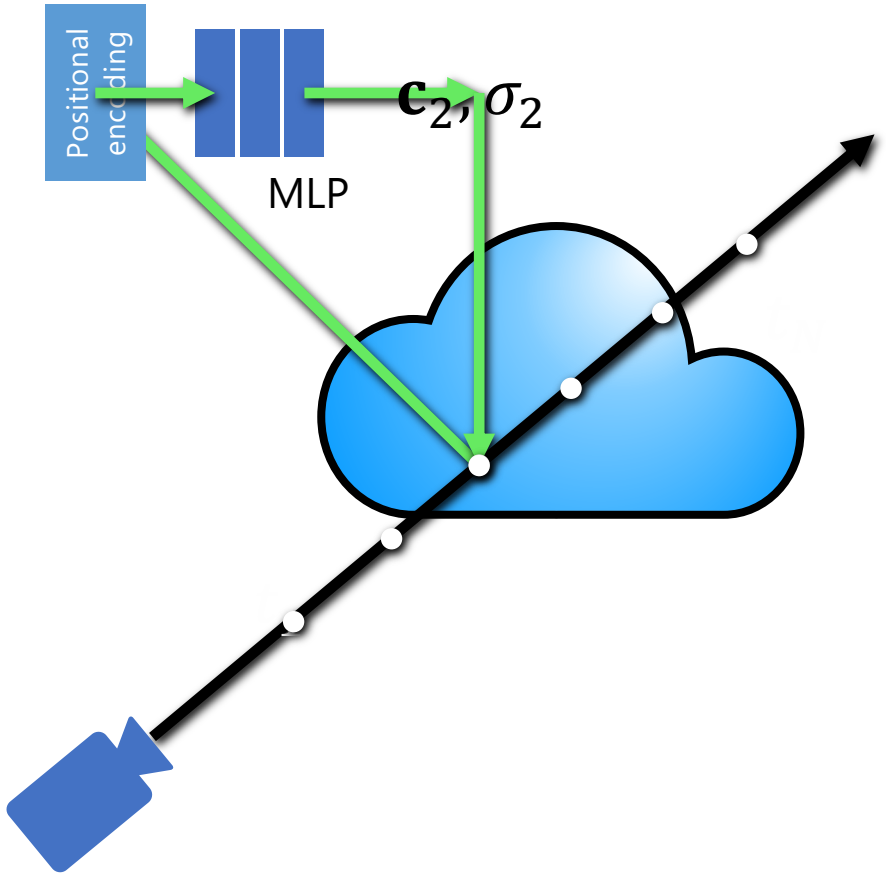
$\mathbf{c}, \sigma$



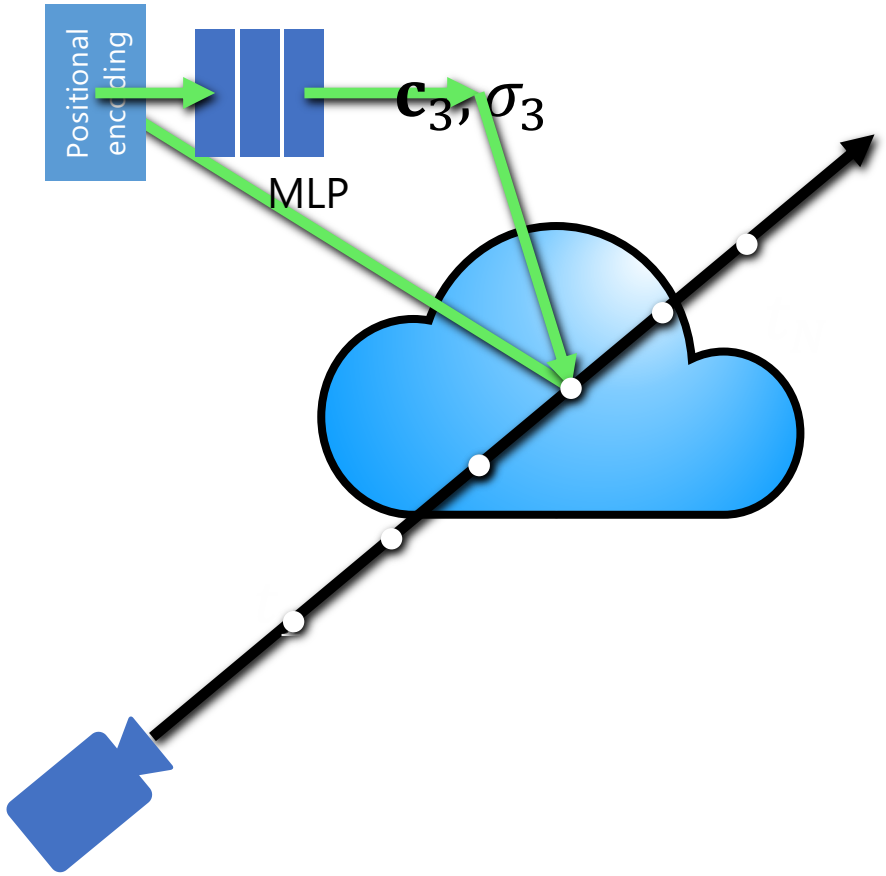
$\mathbf{c}, \sigma$



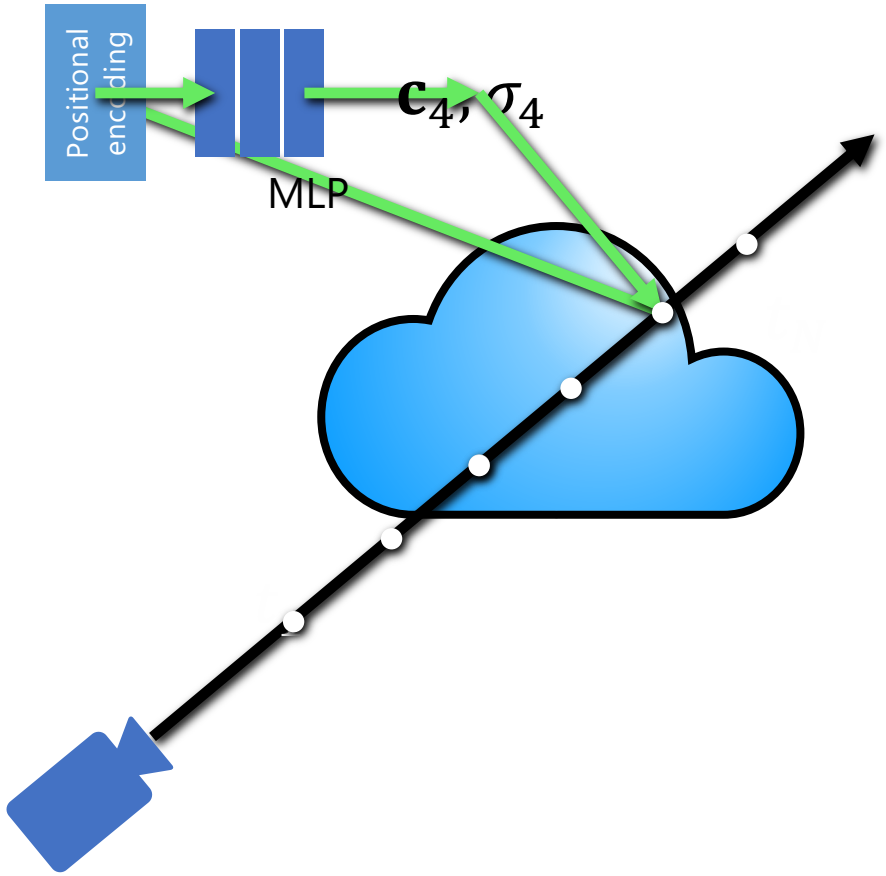
$\mathbf{c}, \sigma$



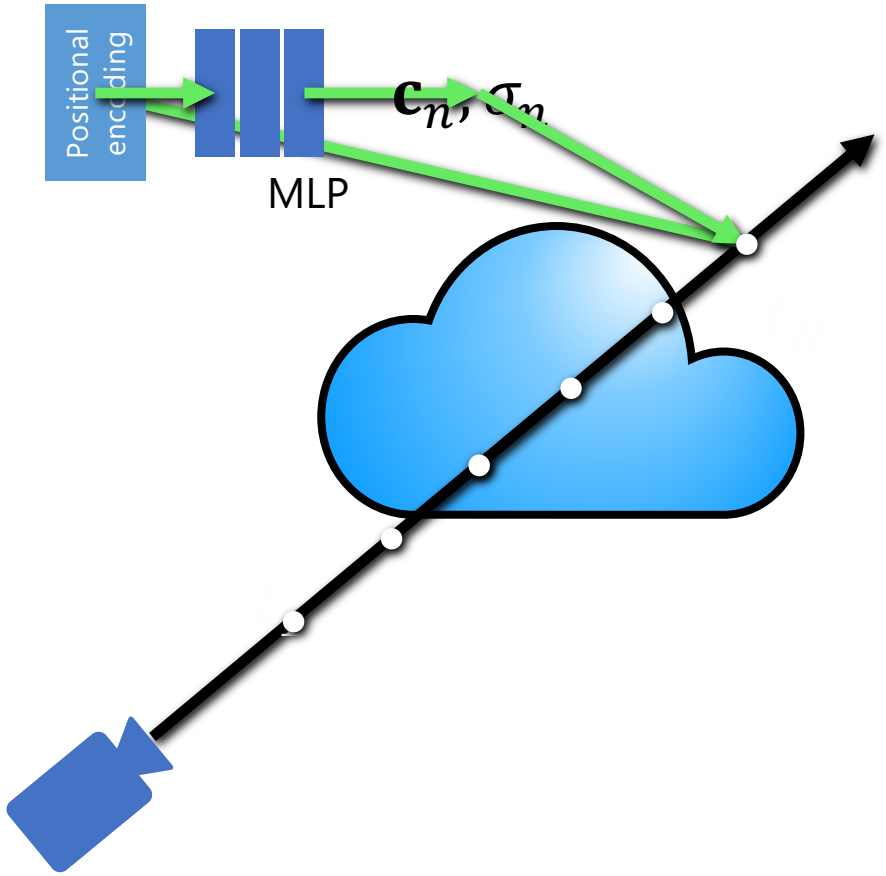
$\mathbf{c}, \sigma$

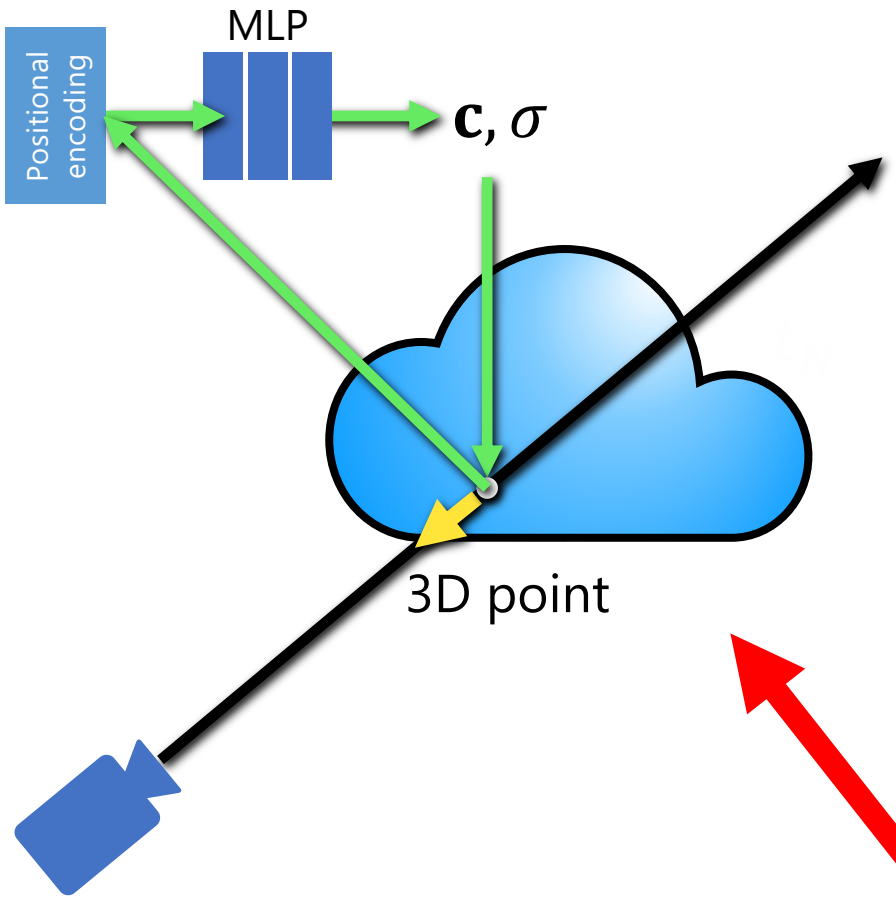


$\mathbf{c}, \sigma$



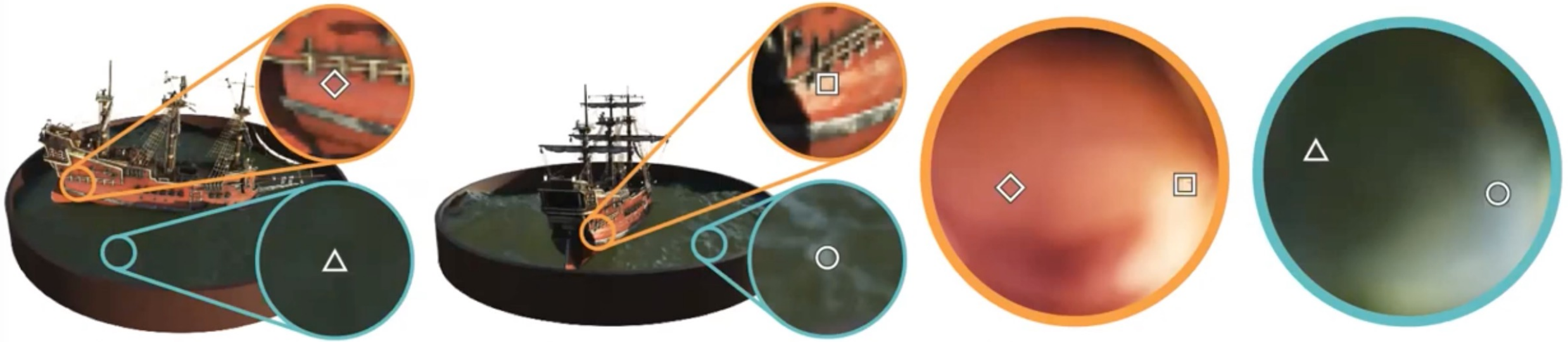
$\mathbf{c}, \sigma$



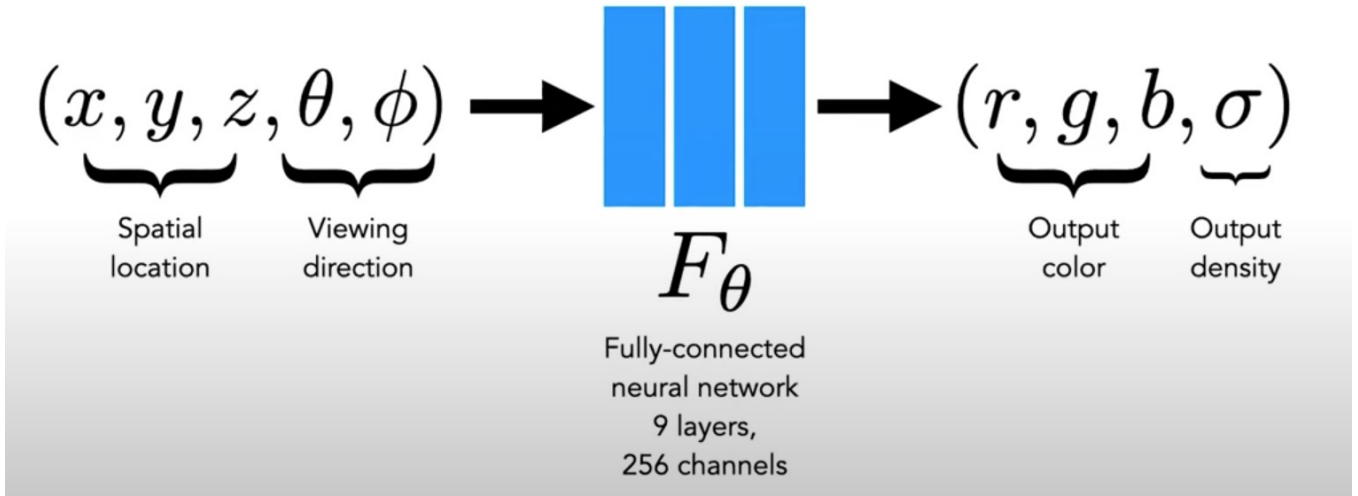


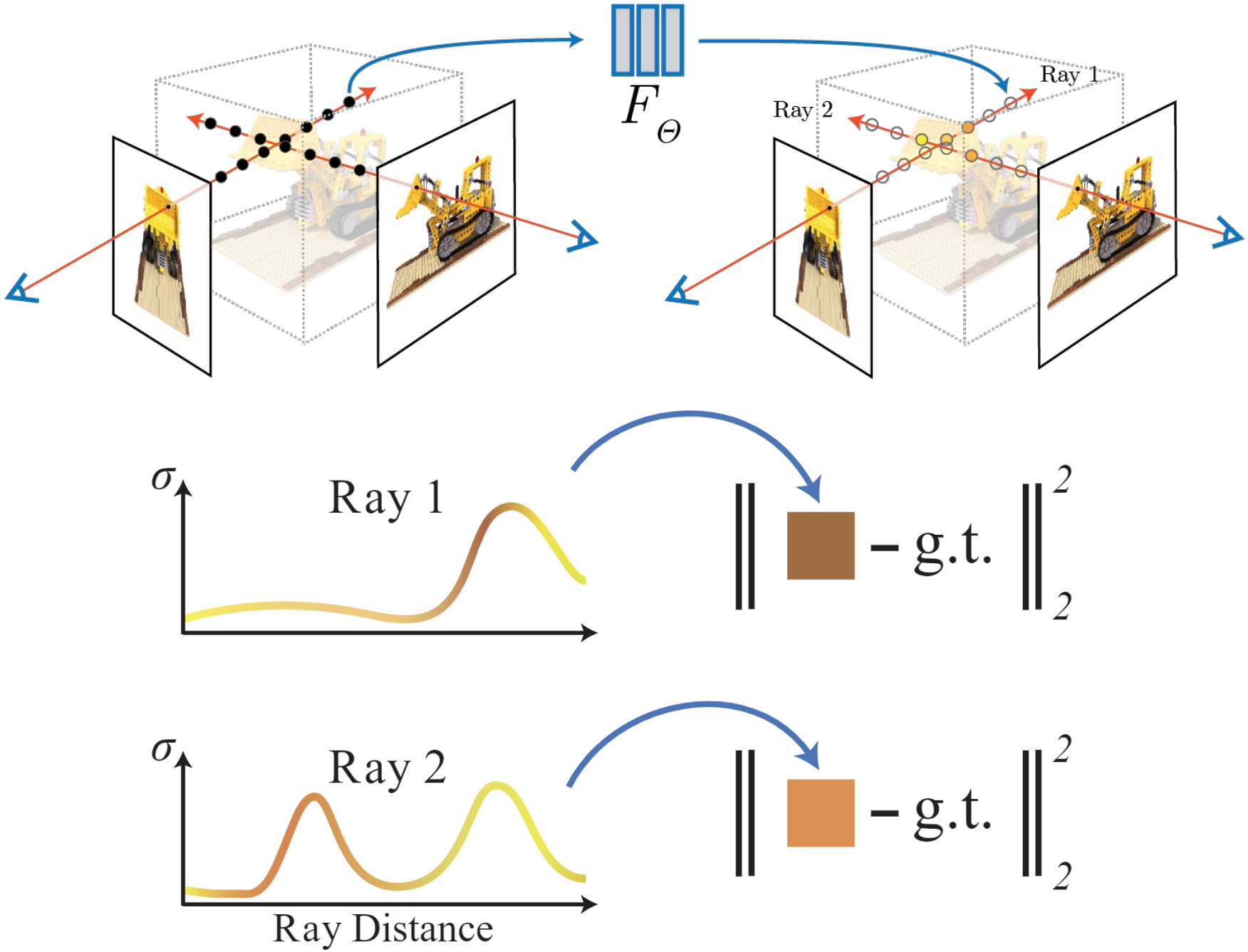
Include the ray direction in the input to the MLP → allows for capturing and rendering view-dependent effects (e.g., shiny surfaces)

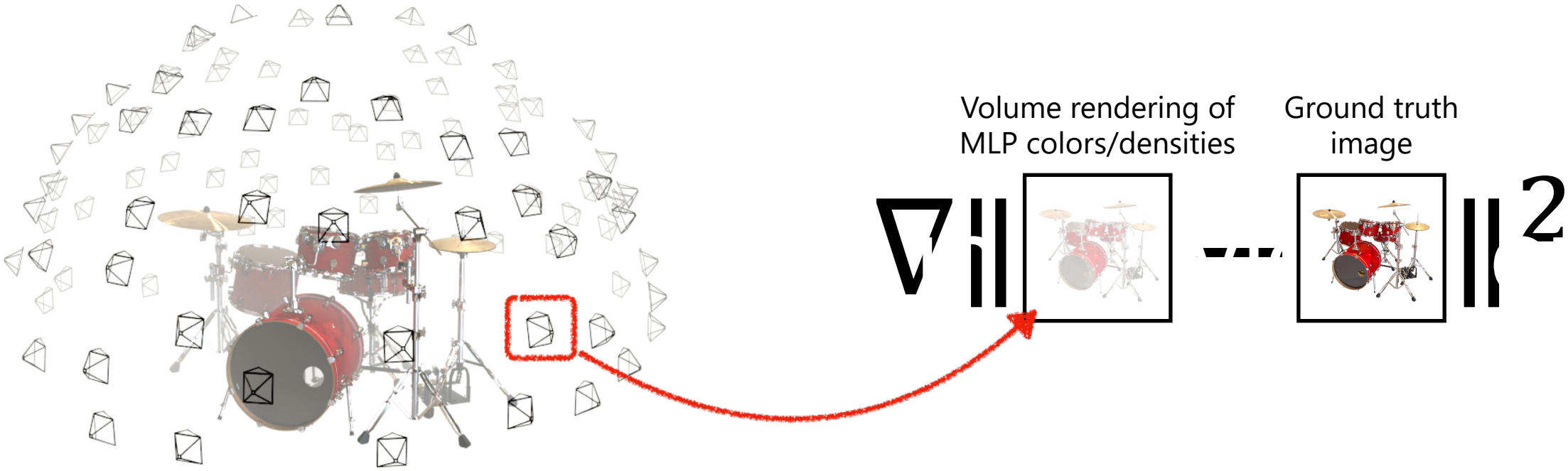
Modeling view dependent effects

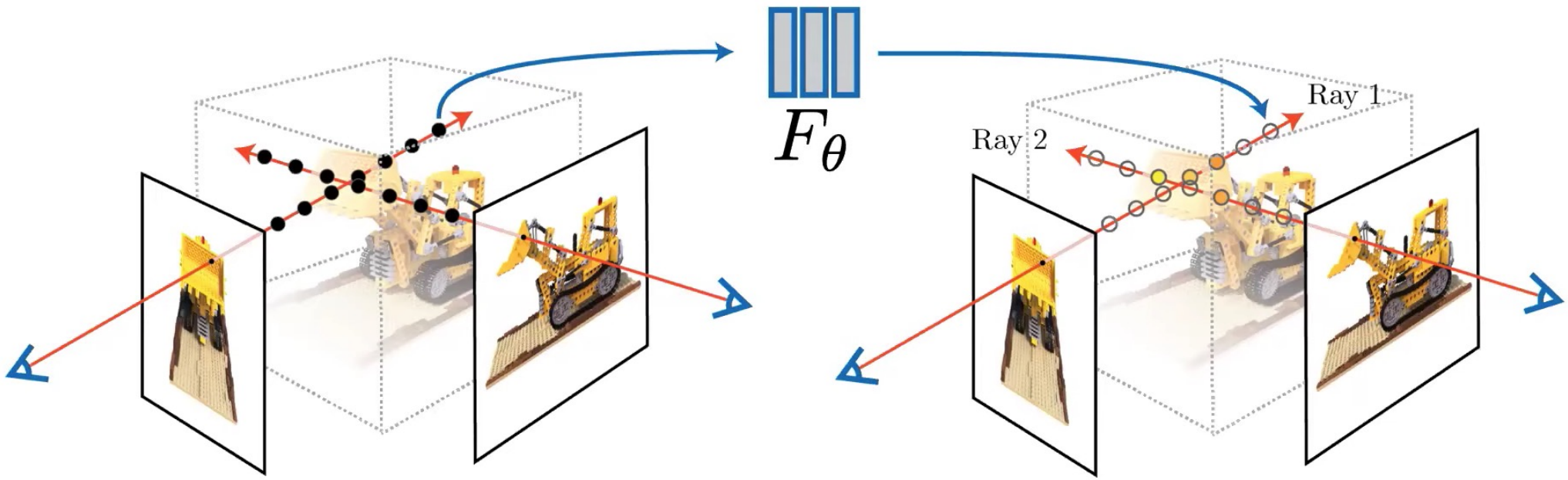


(a) View 1 (b) View 2 (c) Radiance Distributions




F_Θ
Ray 1
Ray 2
σ
Ray 1
σ
Ray 2
Ray Distance
$\| \quad - \text{g.t.} \|_2^2$
$\| \quad - \text{g.t.} \|_2^2$


Volume rendering of
MLP colors/densities
Ground truth
image
2

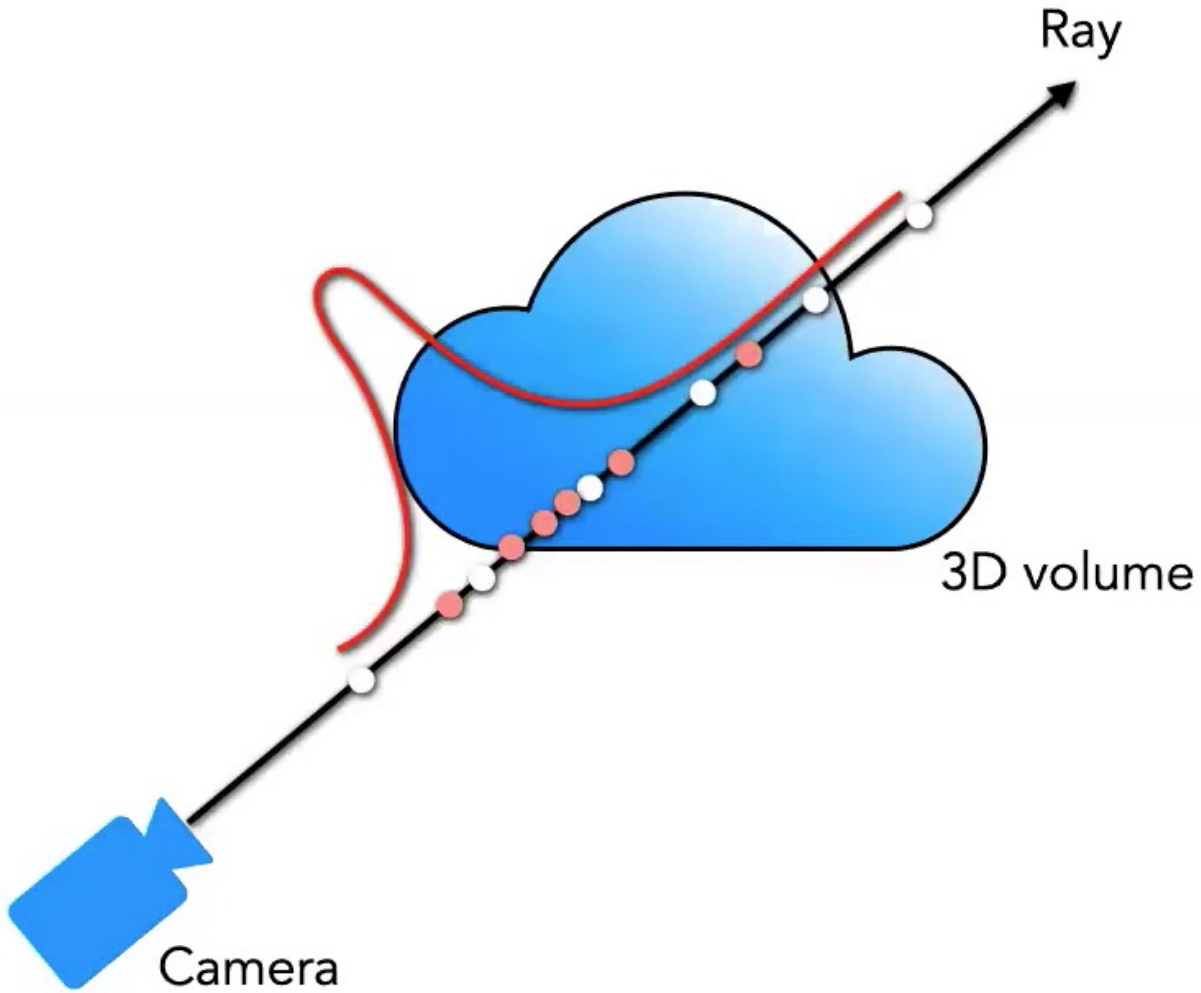


$$\min_\theta \sum_i || \text{render}_i(F_\theta) - I_i ||^2$$

Importance Sampling

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

treat weights as probability distribution for new samples

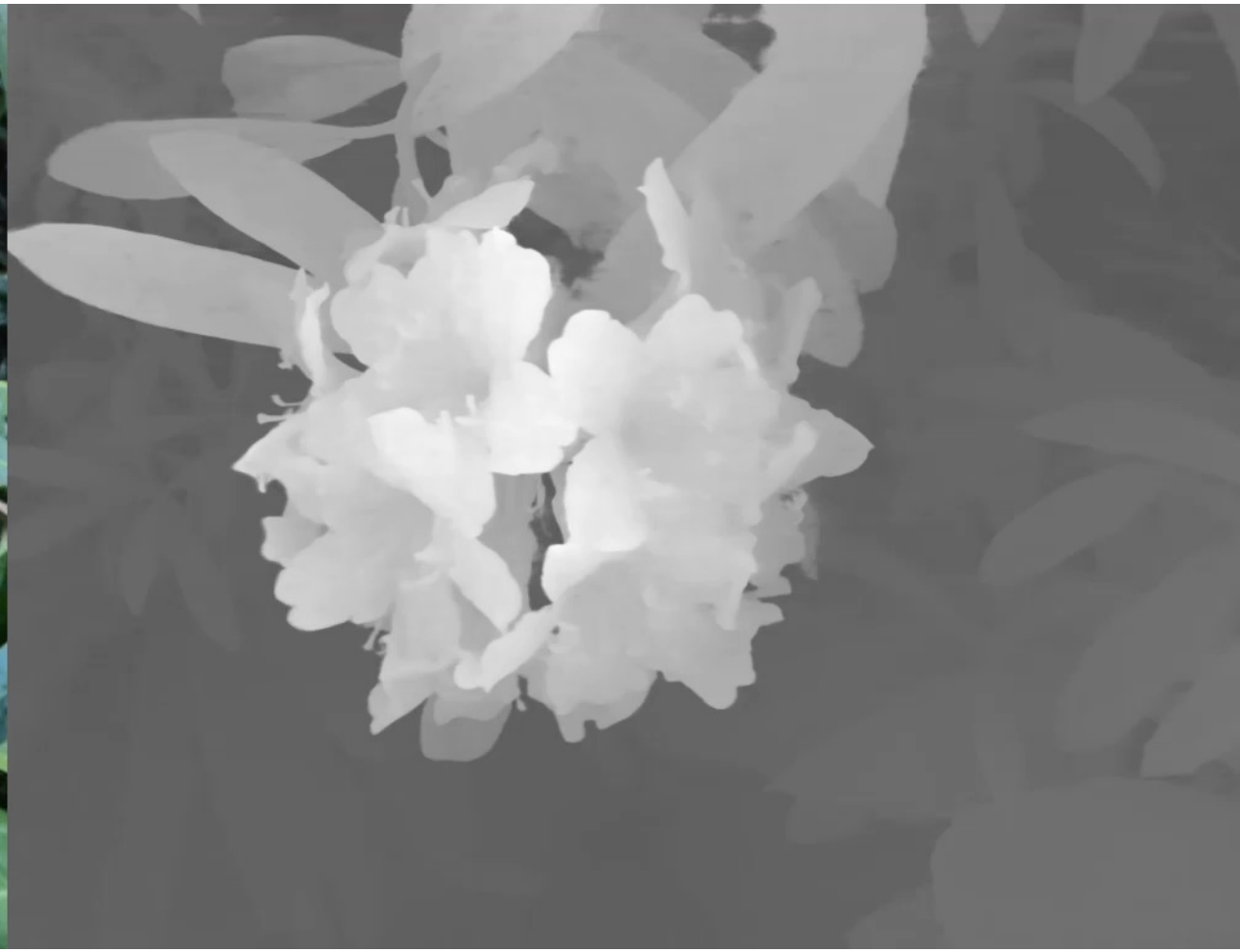


Results

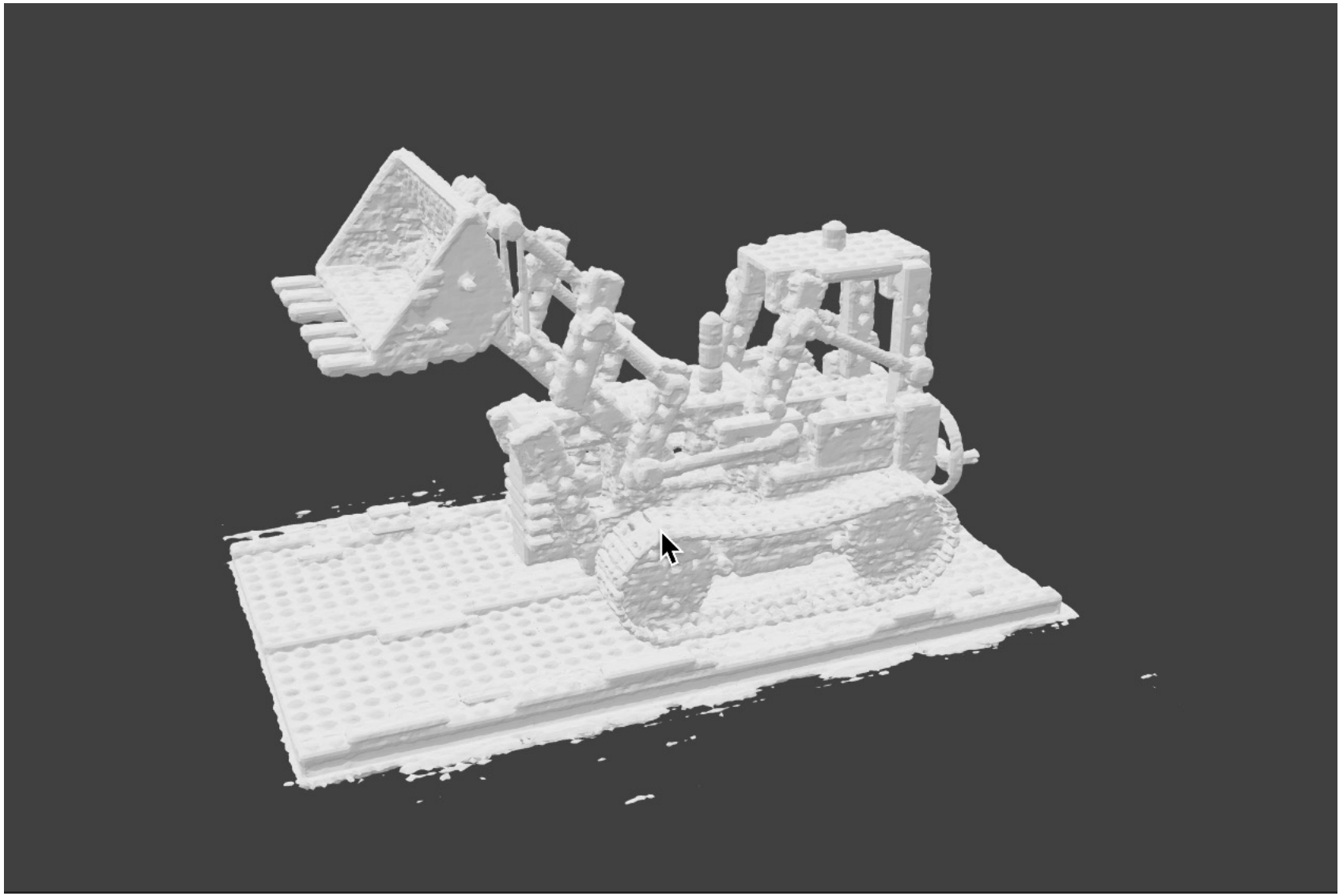

NOR-CAL
MOVING SERVICES


NOR-CAL
MOVING SERVICES

NeRF encodes convincing view-dependent effects using directional dependence



NeRF encodes detailed scene geometry with occlusion effects



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

Next Class:

- Details of NeRF neural network architecture
- Hybrid Representation in NeRF
- Generalization with NeRF
- Conditional Generation / Editing with 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.