THE UNIVERSITY OF TEXAS AT DALLAS

# Neural Radiance Fields (NeRFs)


CS 6384 Computer Vision

Professor Yapeng Tian

Department of Computer Science


Many slides borrowed from Noah Snavely and Angjoo Kanazawa

# Inverse Rendering

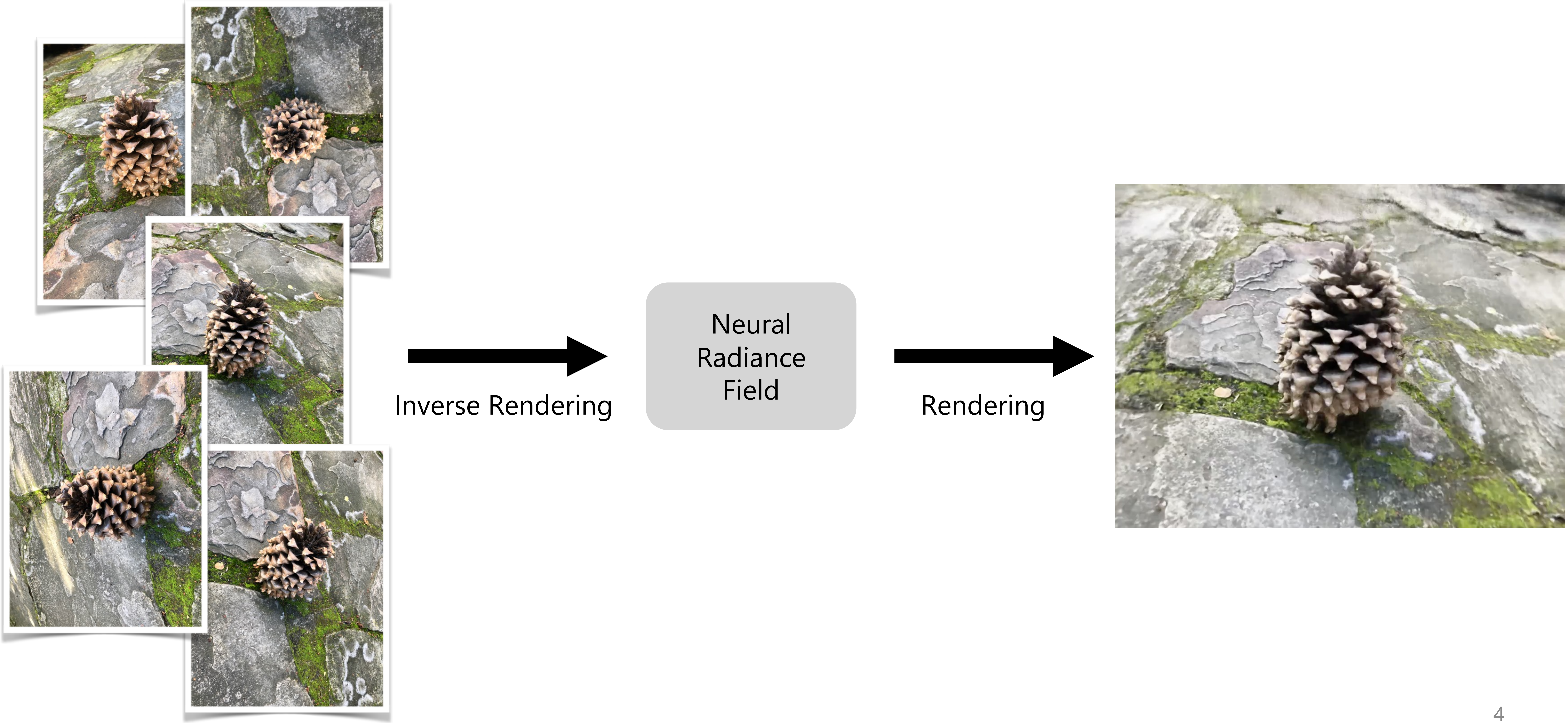
3D
Scene
Representation

Rendering

# Inverse Rendering

Inverse Rendering → 3D Scene Representation → Rendering

# Neural Radiance Fields (NeRF) as an approach to inverse rendering



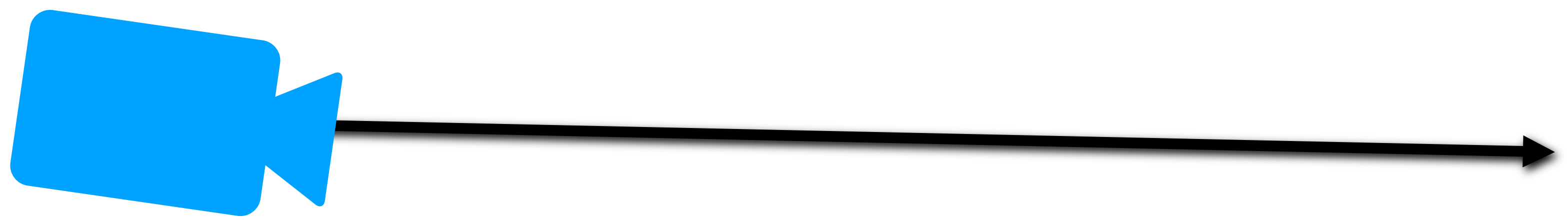
NeRF == Differentiable Rendering with a **Neural Volumetric Representation**

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

# Neural Volumetric Rendering

# Neural Volumetric Rendering

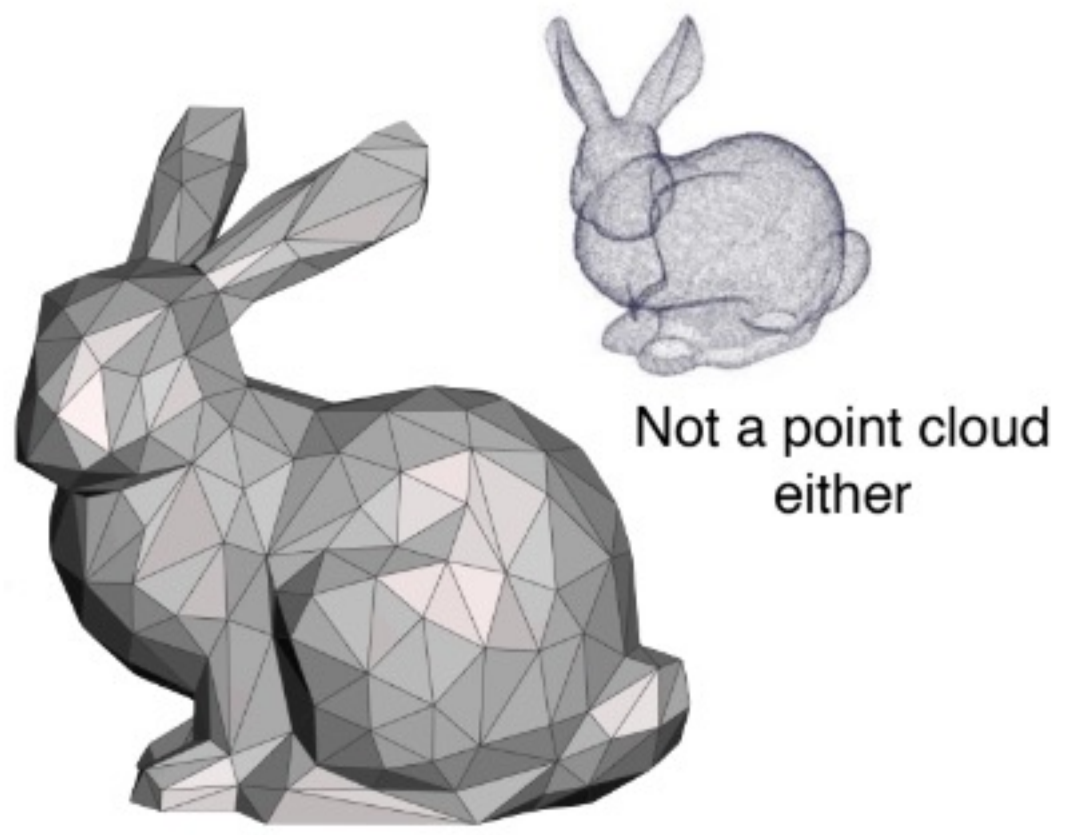
querying the radiance value along rays through 3D space



What color?

# Neural **Volumetric** Rendering

continuous, differentiable
rendering model without
concrete ray/surface intersections

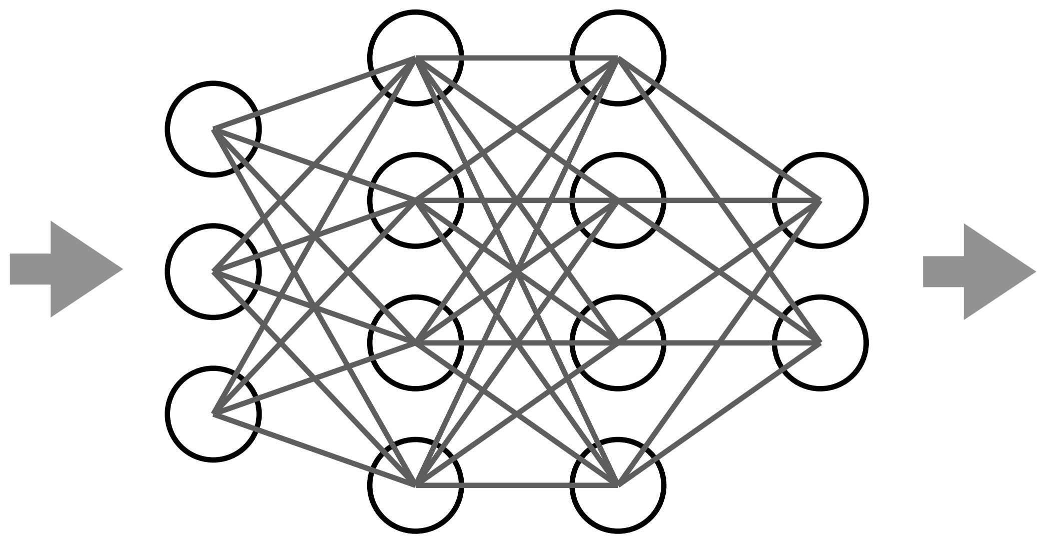



It’s continuous voxels made of shiny transparent cubes

# 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

Optimize a NeRF model

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

3D reconstruction viewable from any angle

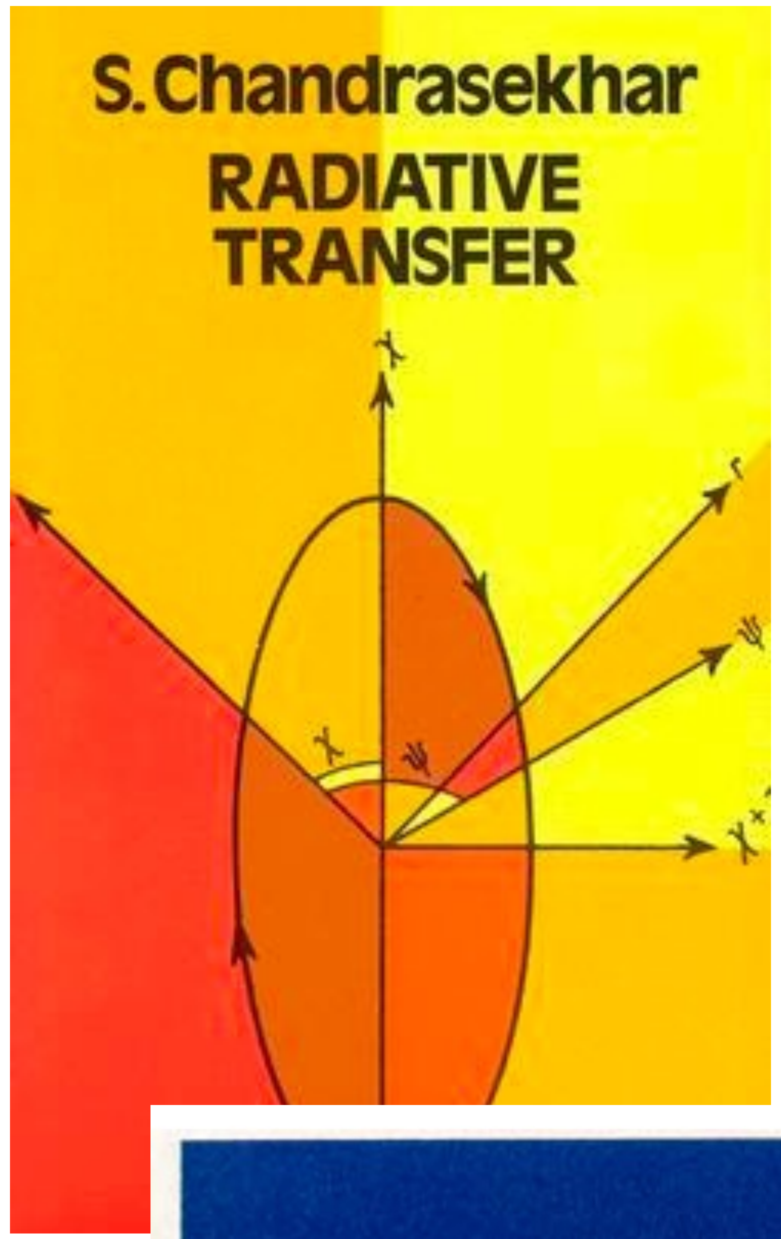
# NeRF Overview

- Volumetric rendering
- Neural networks as representations for spatial data
- Neural Radiance Fields (NeRF)

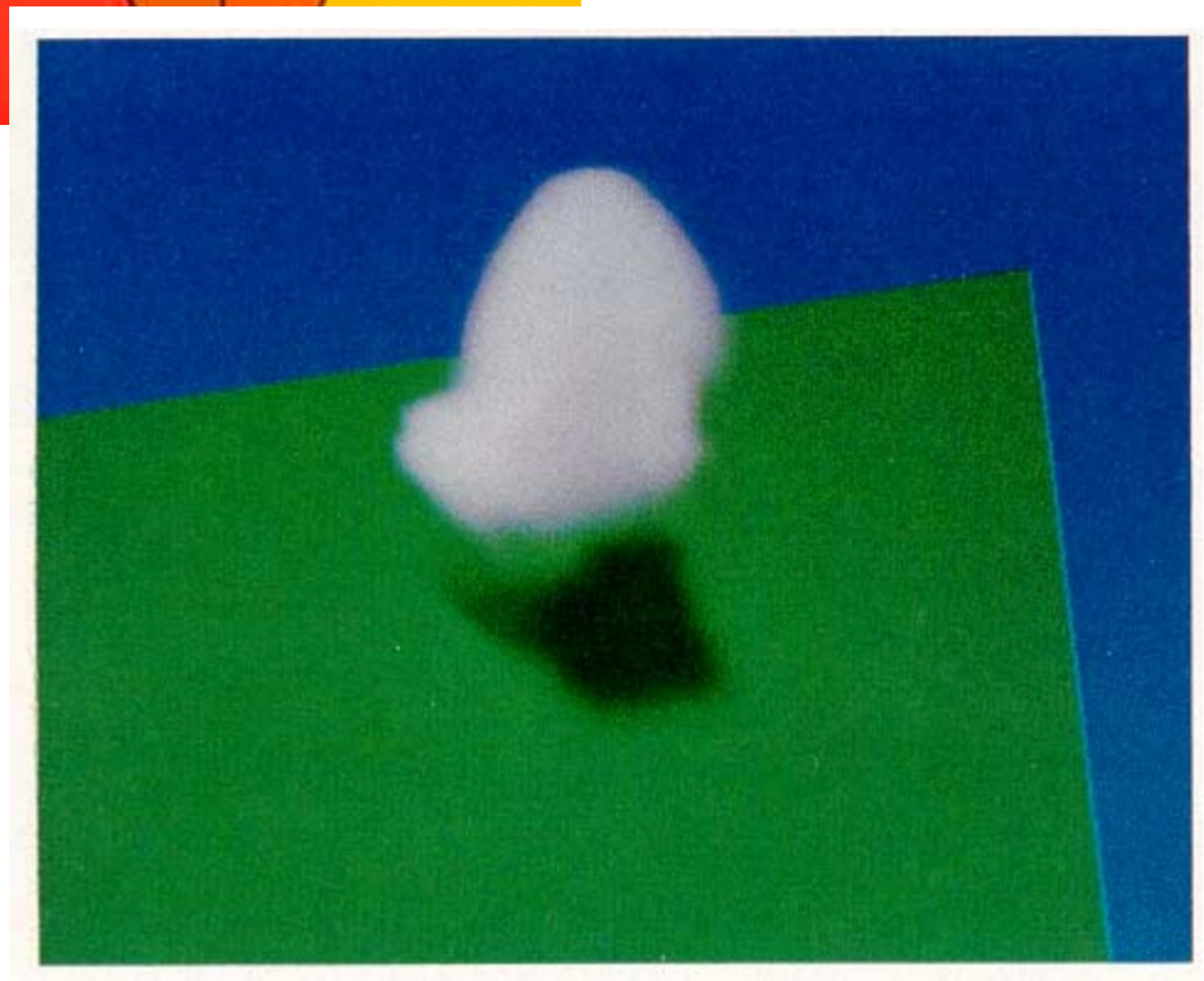
# NeRF Overview

- Volumetric rendering
- Neural networks as representations for spatial data
- Neural Radiance Fields (NeRF)

# Traditional volumetric rendering



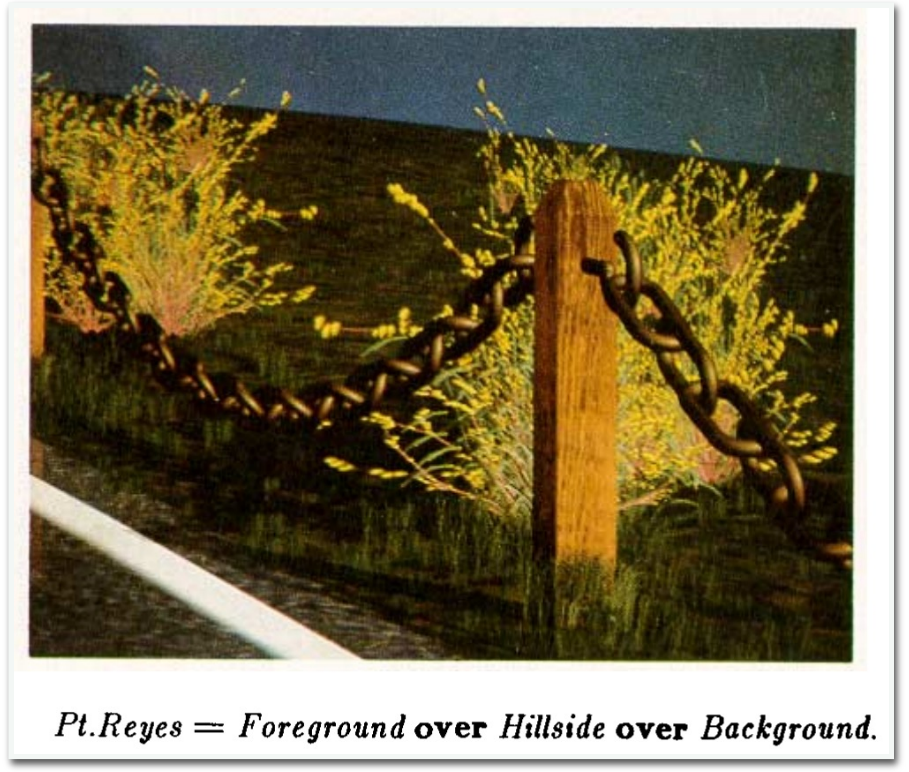

- Theory of volume rendering co-opted from physics in the 1980s: absorption, emission, out-scattering/in-scattering


Ray tracing simulated cumulus cloud [Kajiya]

Chandrasekhar 1950, Radiative Transfer
Kajiya 1984, Ray Tracing Volume Densities

# Traditional volumetric rendering

Medical data visualisation [Levoy]




Alpha compositing [Porter and Duff]

- Theory of volume rendering co-opted from physics in the 1980s: absorption, emission, out-scattering/in-scattering
- Adapted for visualising medical data and linked with alpha compositing

Chandrasekhar 1950, Radiative Transfer
Kajiya 1984, Ray Tracing Volume Densities
Levoy 1988, Display of Surfaces from Volume Data
Max 1995, Optical Models for Direct Volume Rendering
Porter and Duff 1984, Compositing Digital Images

# Traditional volumetric rendering


Physically-based Monte Carlo rendering [Novak et al]

- Theory of volume rendering co-opted from physics in the 1980s: absorption, emission, out-scattering/in-scattering
- Adapted for visualising medical data and linked with alpha compositing
- Modern path tracers use sophisticated Monte Carlo methods to render volumetric effects

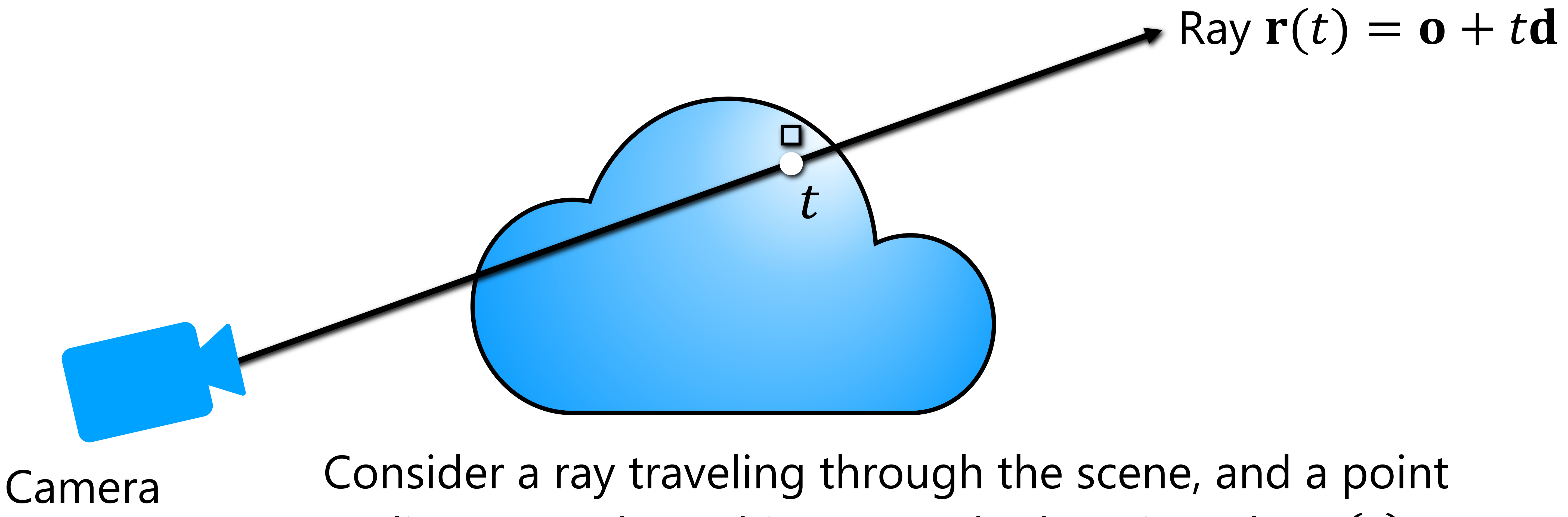
Chandrasekhar 1950, Radiative Transfer
Kajiya 1984, Ray Tracing Volume Densities
Levoy 1988, Display of Surfaces from Volume Data
Max 1995, Optical Models for Direct Volume Rendering
Porter and Duff 1984, Compositing Digital Images
Novak et al 2018, Monte Carlo methods for physically based volume rendering

# Volumetric formulation for NeRF

Scene is a cloud of colored fog

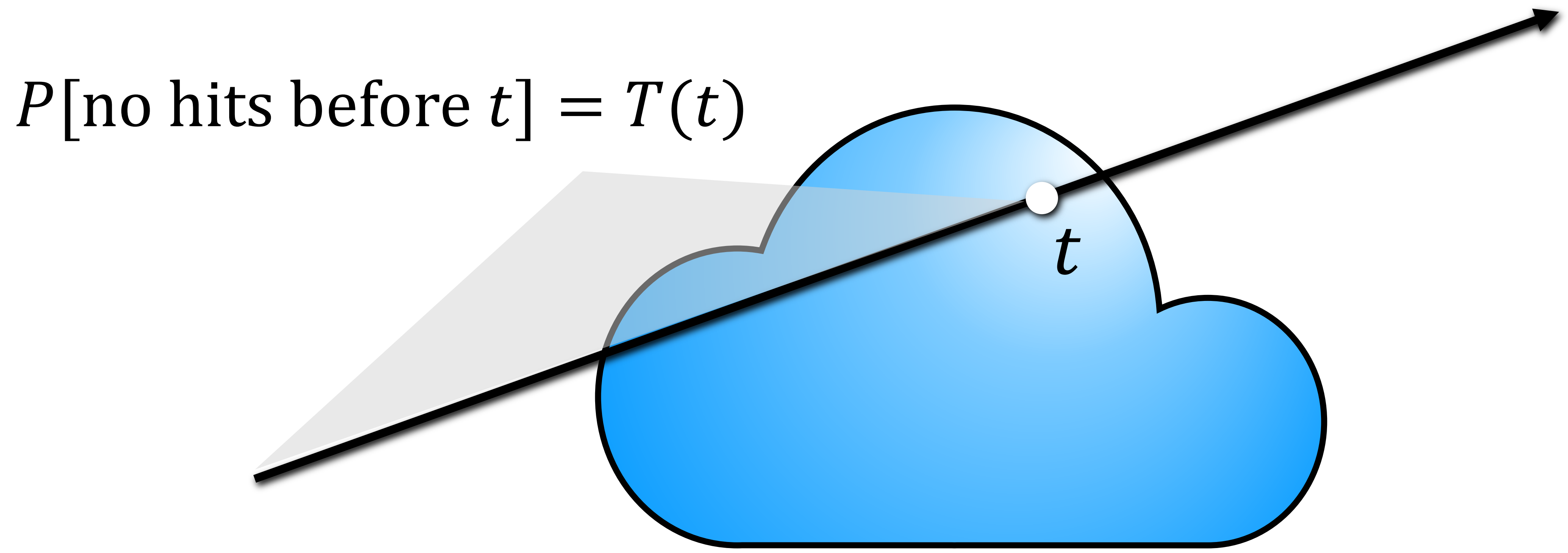
Max and Chen 2010, Local and Global Illumination in the Volume Rendering Integral

# 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




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 $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

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

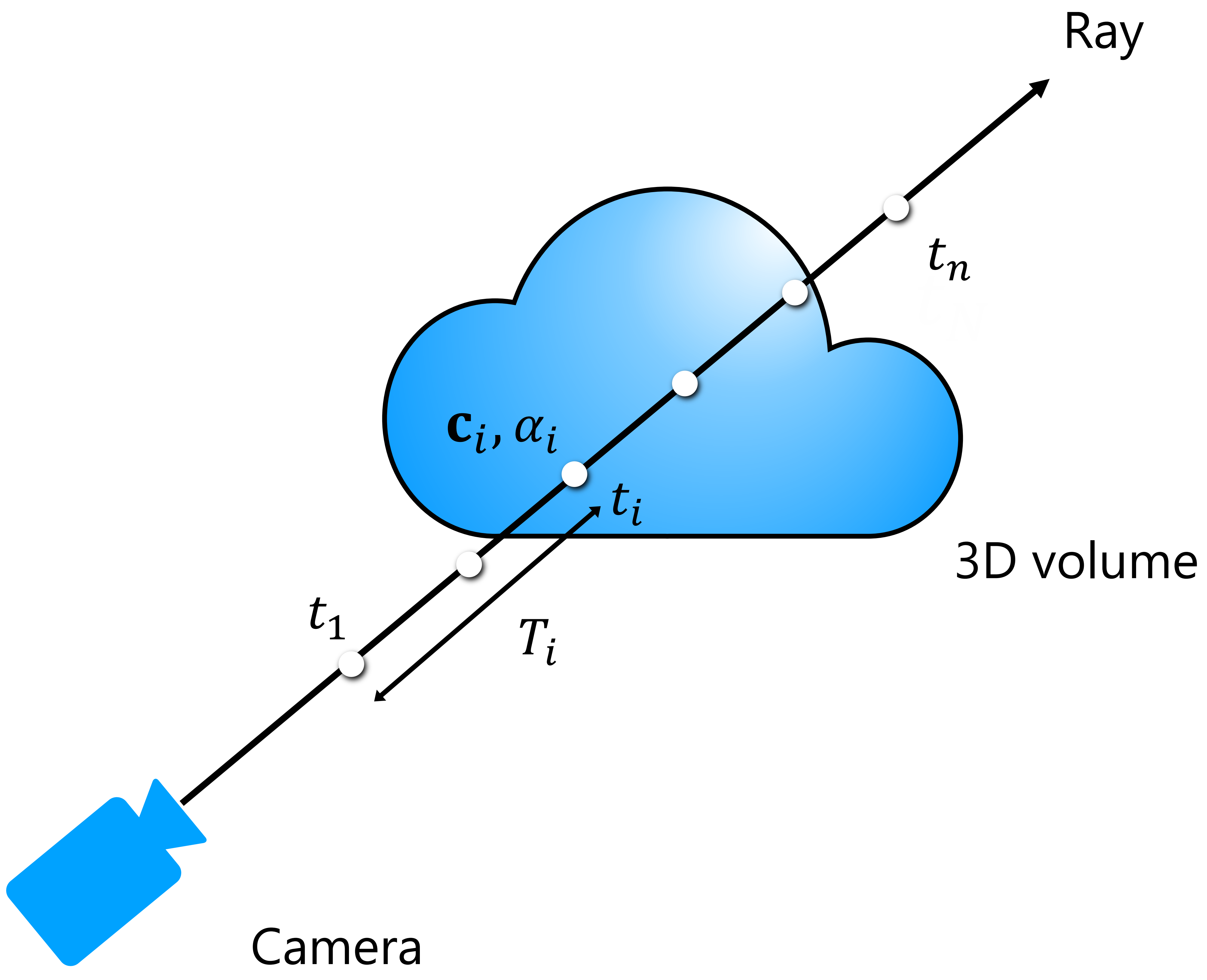
final rendered color along ray

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

# Volume rendering estimation: integrating color along a ray

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

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

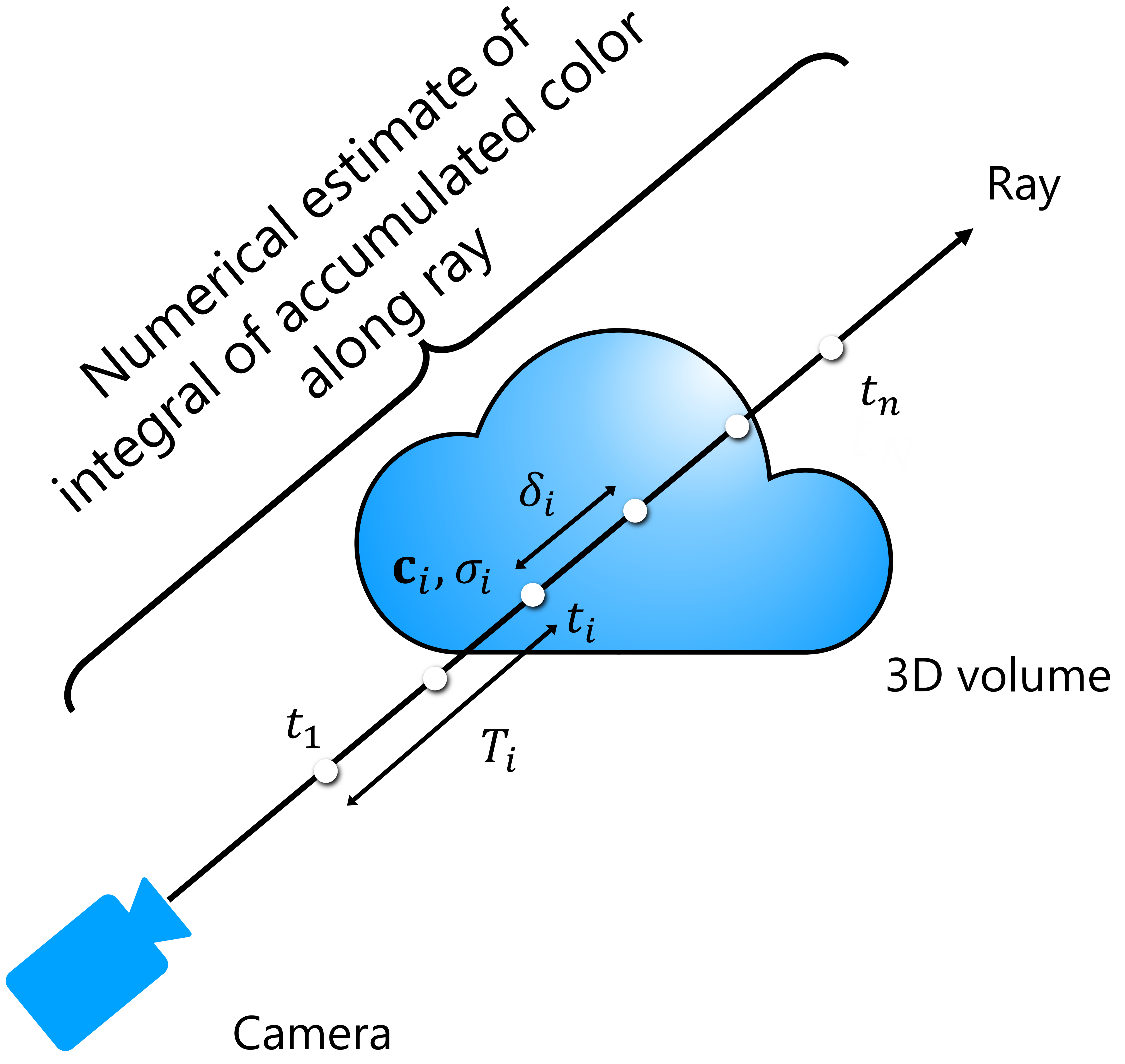
final rendered color along ray

weights

colors

How much light is blocked earlier along ray:

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




Slight modification: $\alpha$ 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):

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

# Volume rendering estimation: integrating color along a ray

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

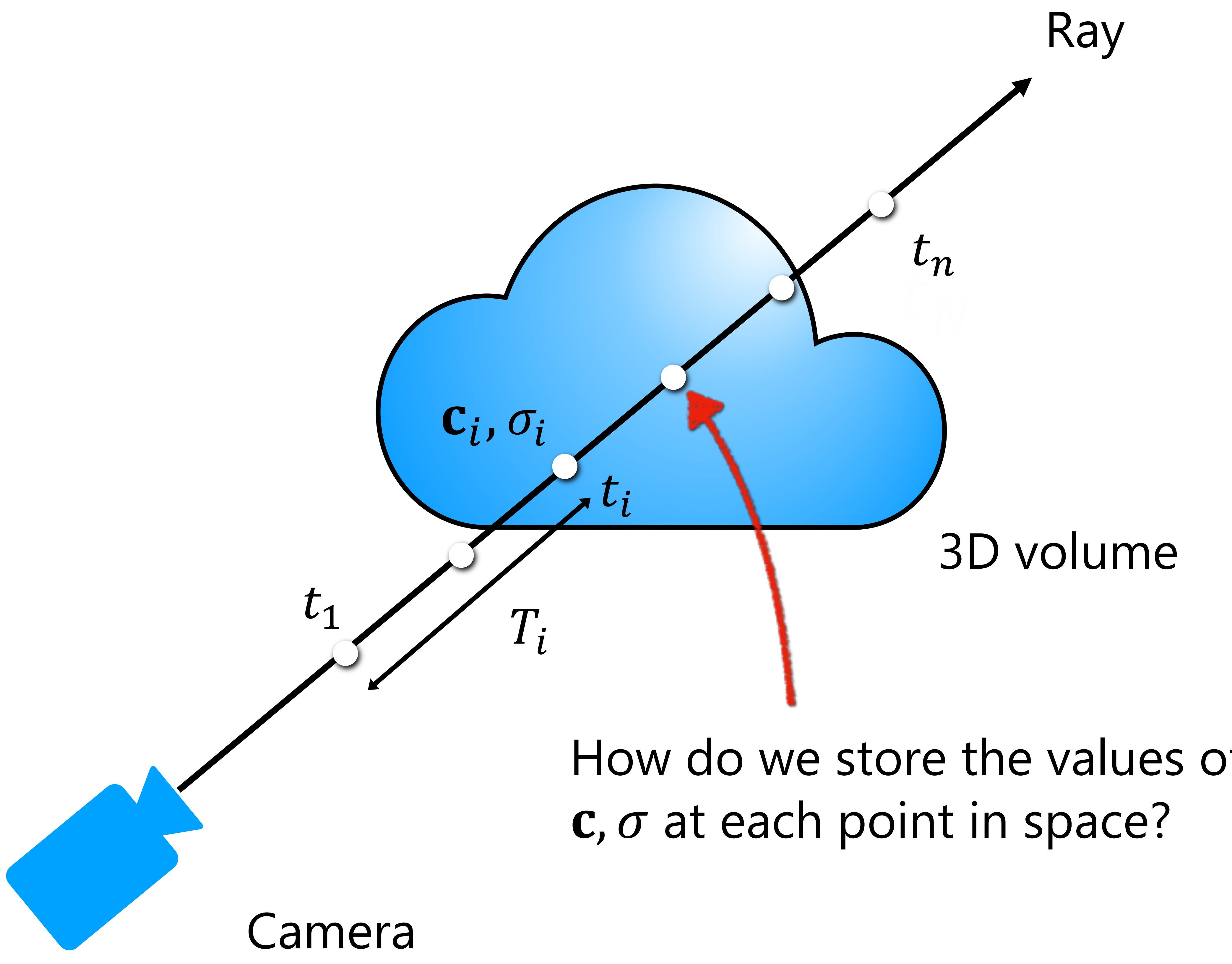
$$\underbrace{\mathbf{c}}_{\text{final rendered color along ray}} \approx \sum_{i=1}^{n} \underbrace{T_i \alpha_i}_{\text{weights}} \underbrace{\mathbf{c}_i}_{\text{colors}}$$

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





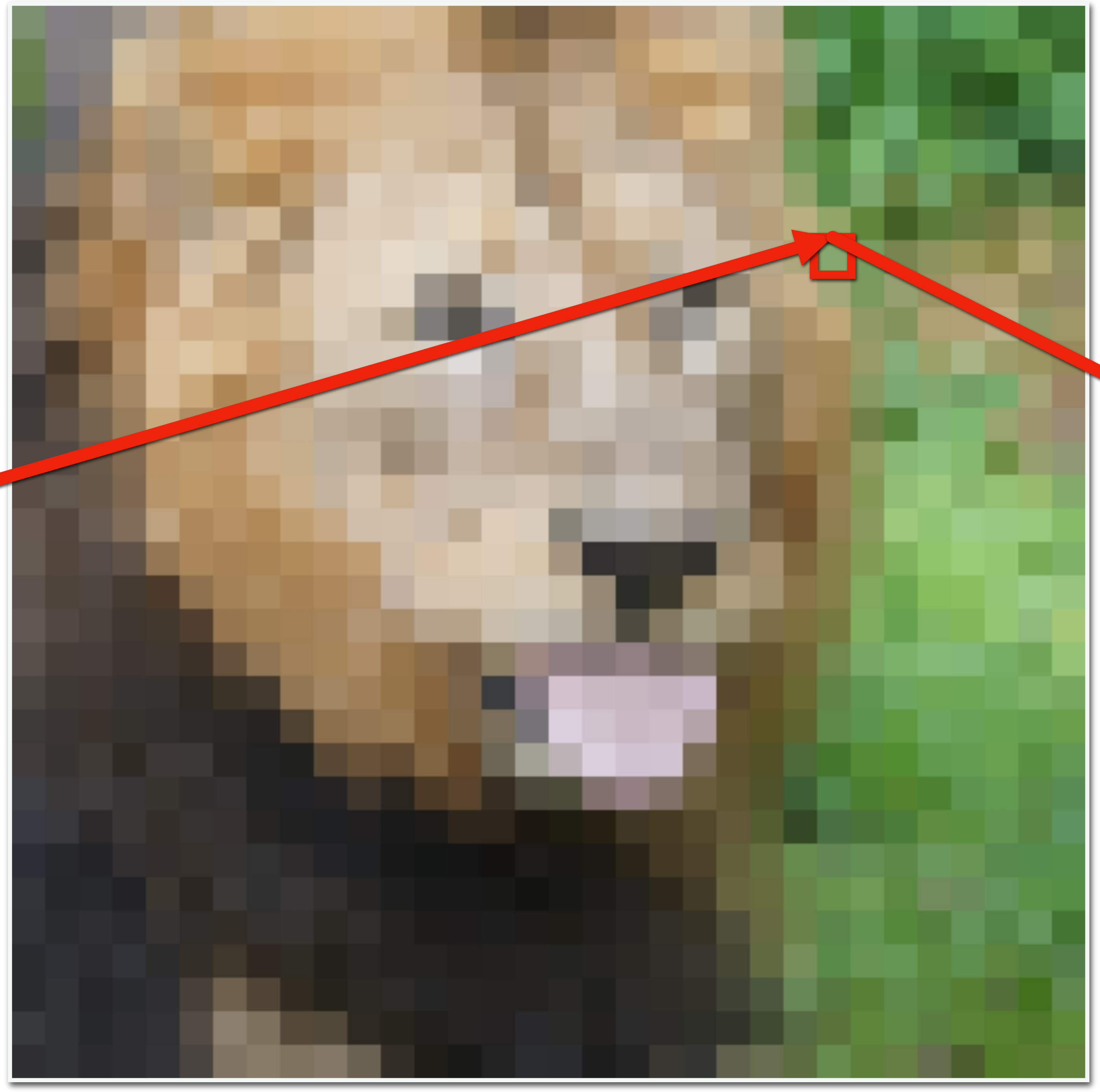

How do we store the values of $\mathbf{c}, \sigma$ at each point in space?

# NeRF Overview

- Volumetric rendering
- Neural networks as representations for spatial data
- Neural Radiance Fields (NeRF)

# Toy problem: storing 2D image data

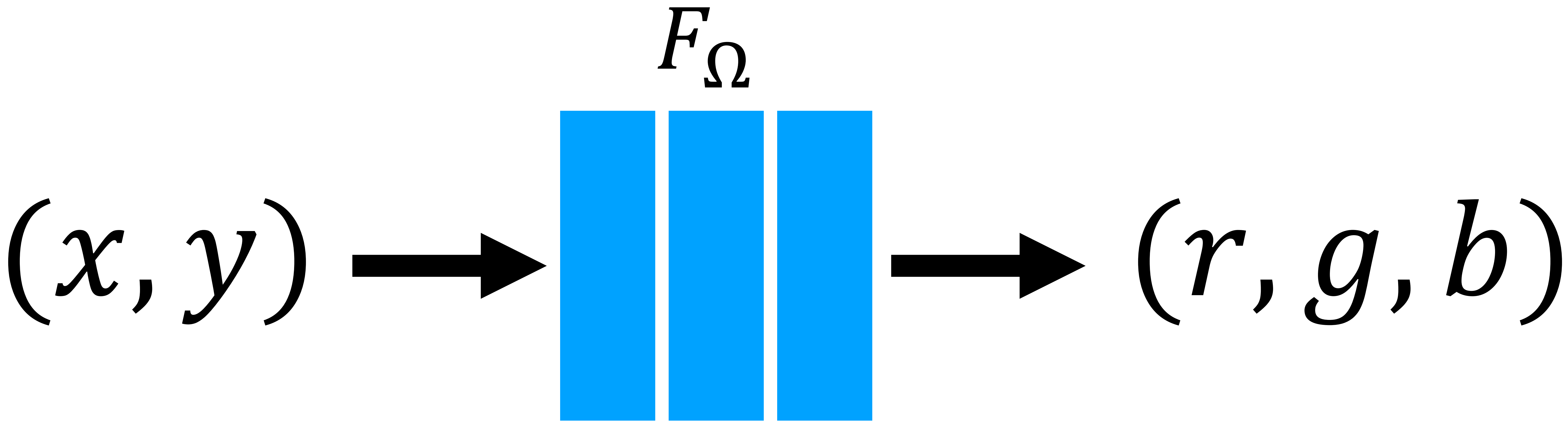
$(x, y)$



$(r, g, b)$

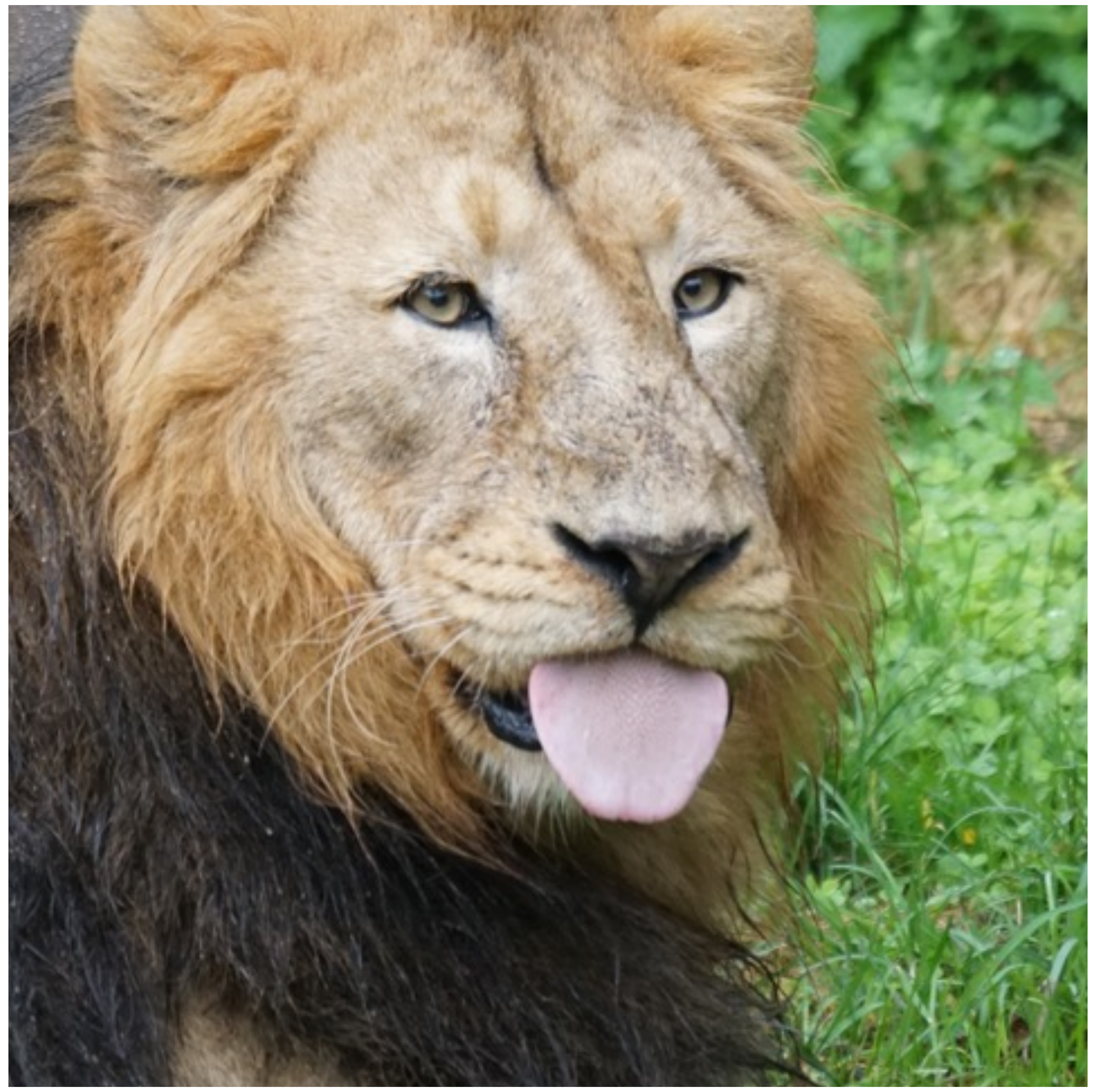
Usually we store an image as a
2D grid of RGB color values

# 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:

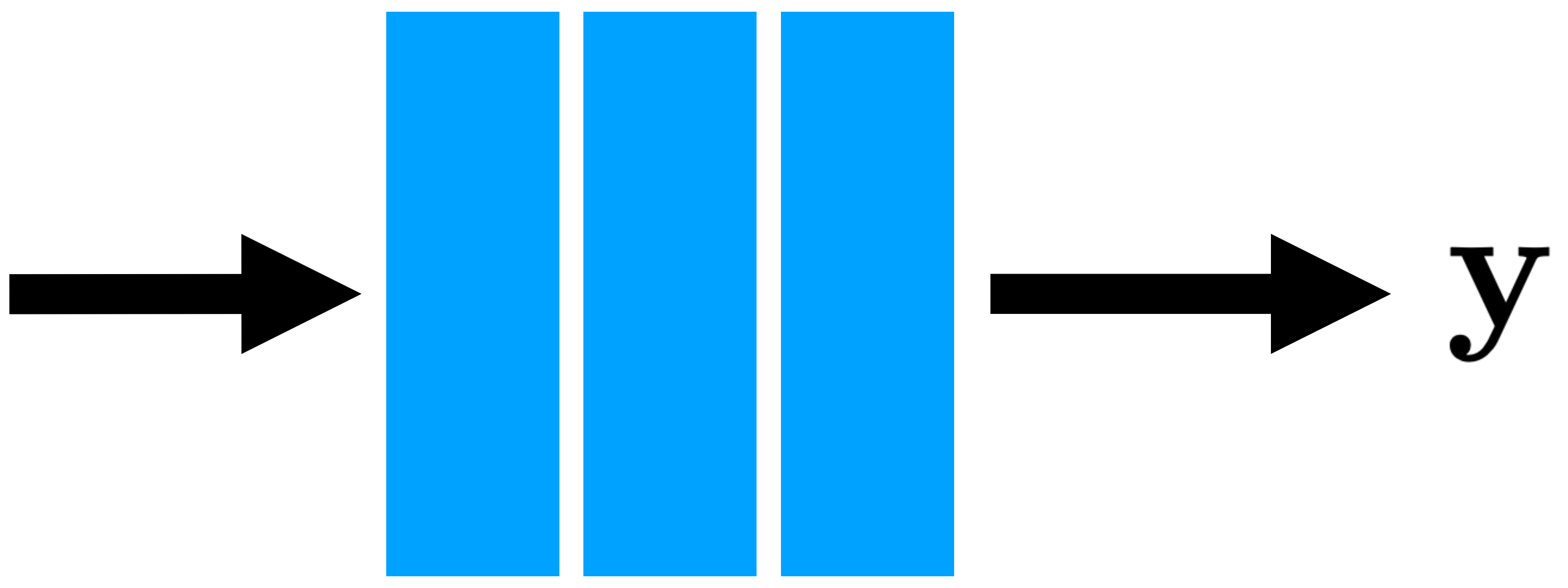
## “Standard” coordinate-based MLPs cannot represent high frequency functions

# Solution:

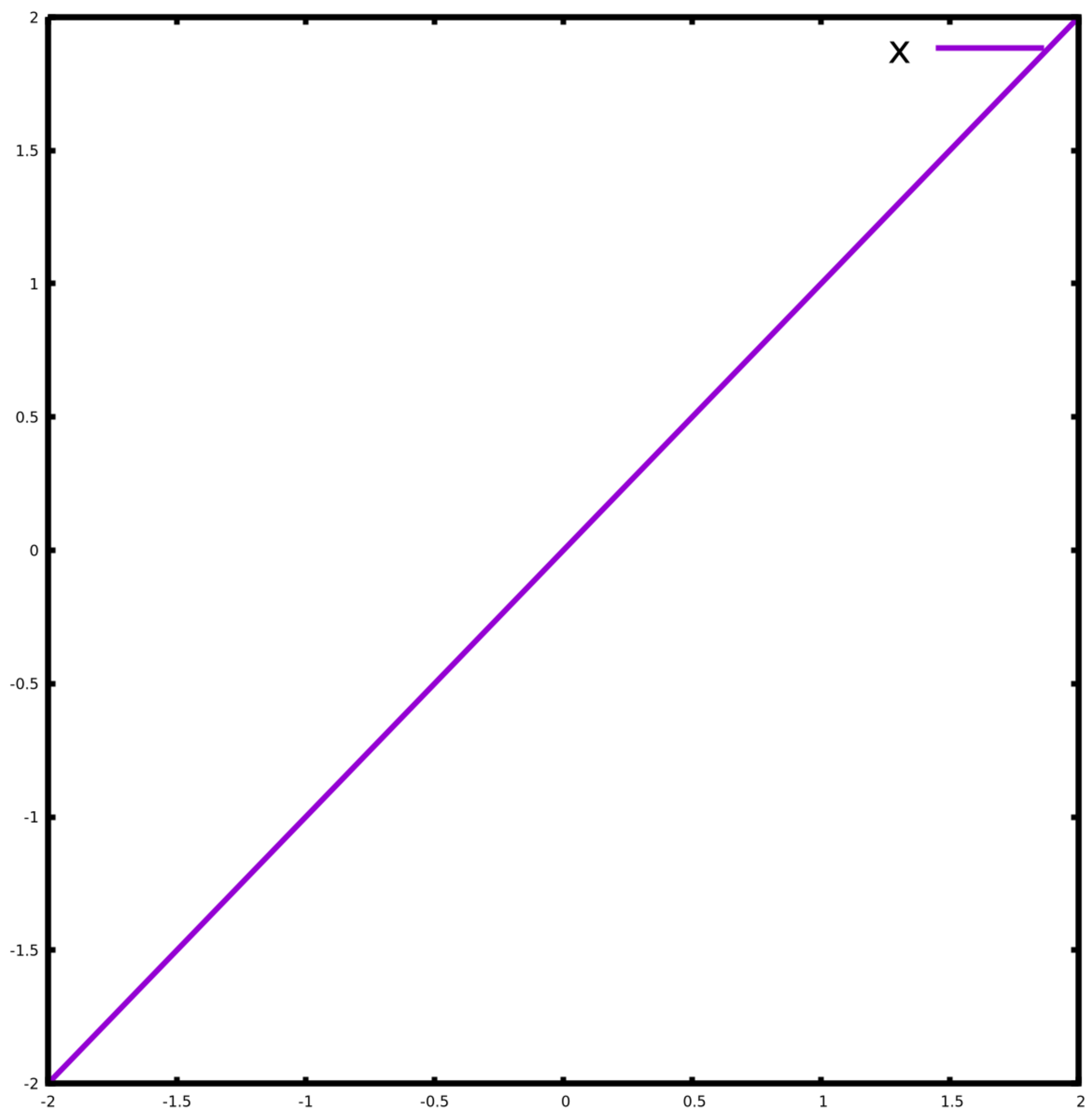
Pass input coordinates through a high frequency mapping first

# Example mapping: “positional encoding”

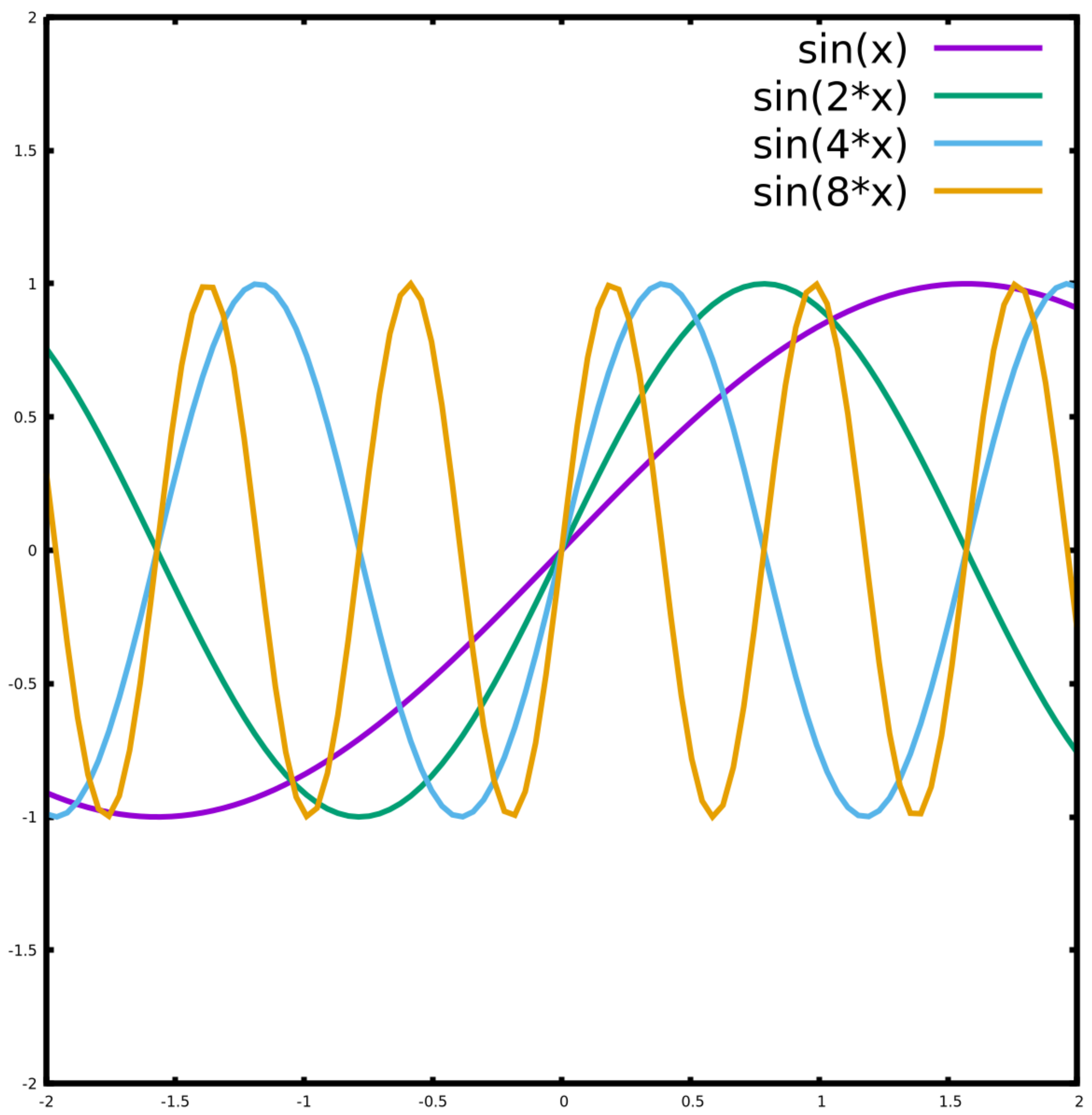

$$\begin{pmatrix} \sin(\mathbf{v}), \cos(\mathbf{v}) \\ \sin(2\mathbf{v}), \cos(2\mathbf{v}) \\ \sin(4\mathbf{v}), \cos(4\mathbf{v}) \\ \dots \\ \sin(2^{L-1}\mathbf{v}), \cos(2^{L-1}\mathbf{v}) \end{pmatrix}$$



# Positional encoding

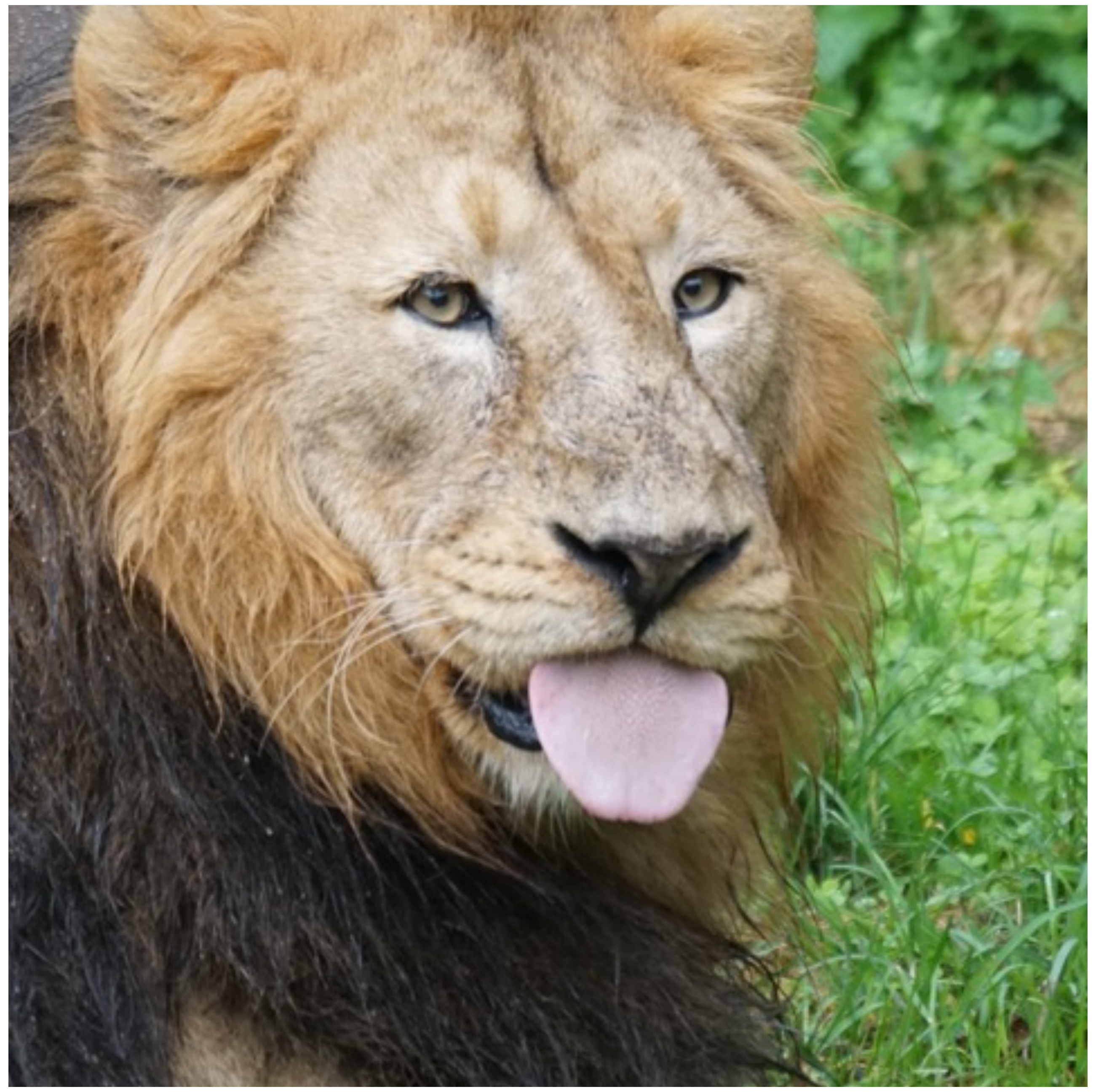



Raw encoding of a number ***x***



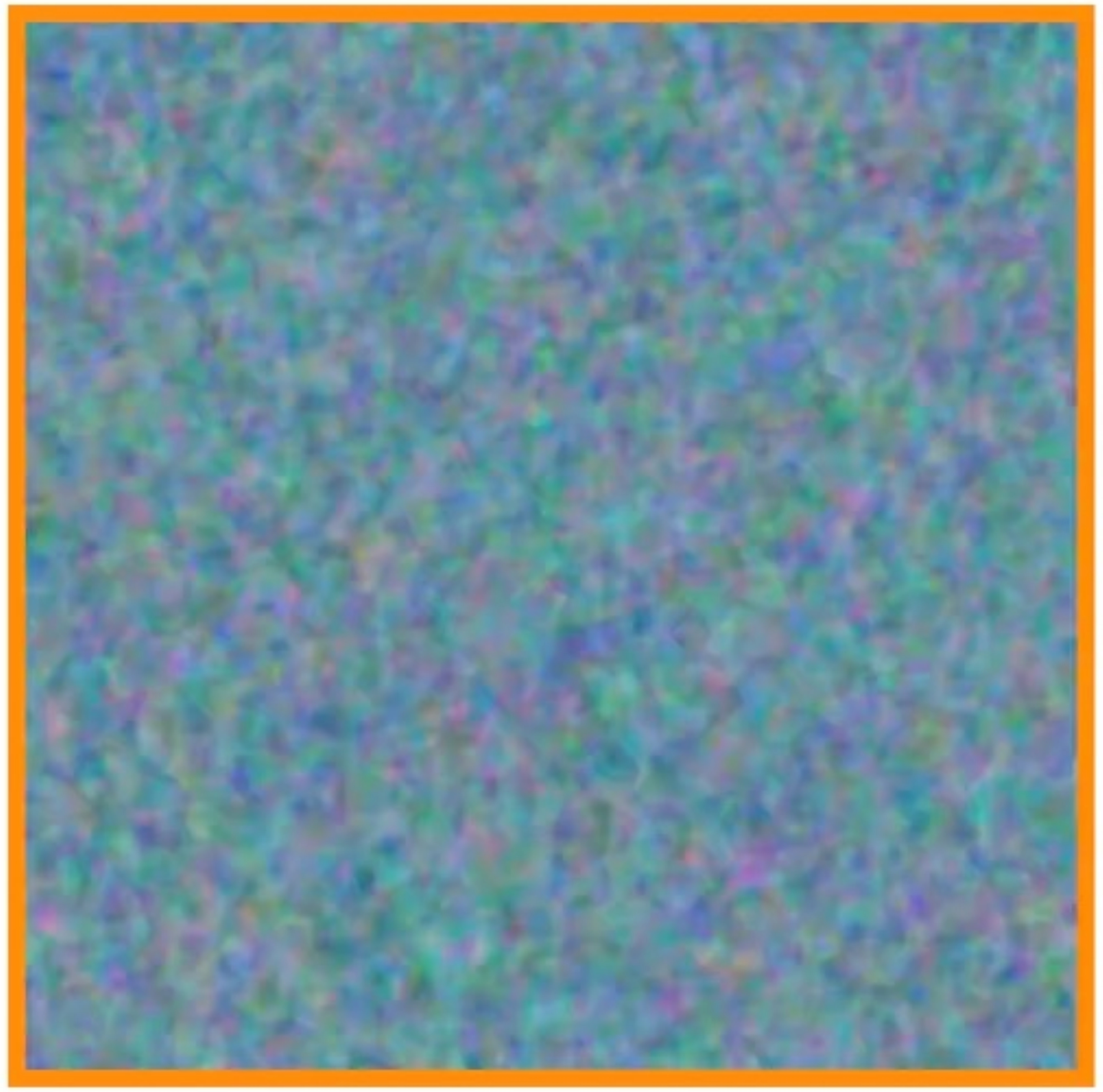

"Positional encoding" of a number ***x***

# Problem solved!



Ground truth image

Neural network output without high frequency mapping



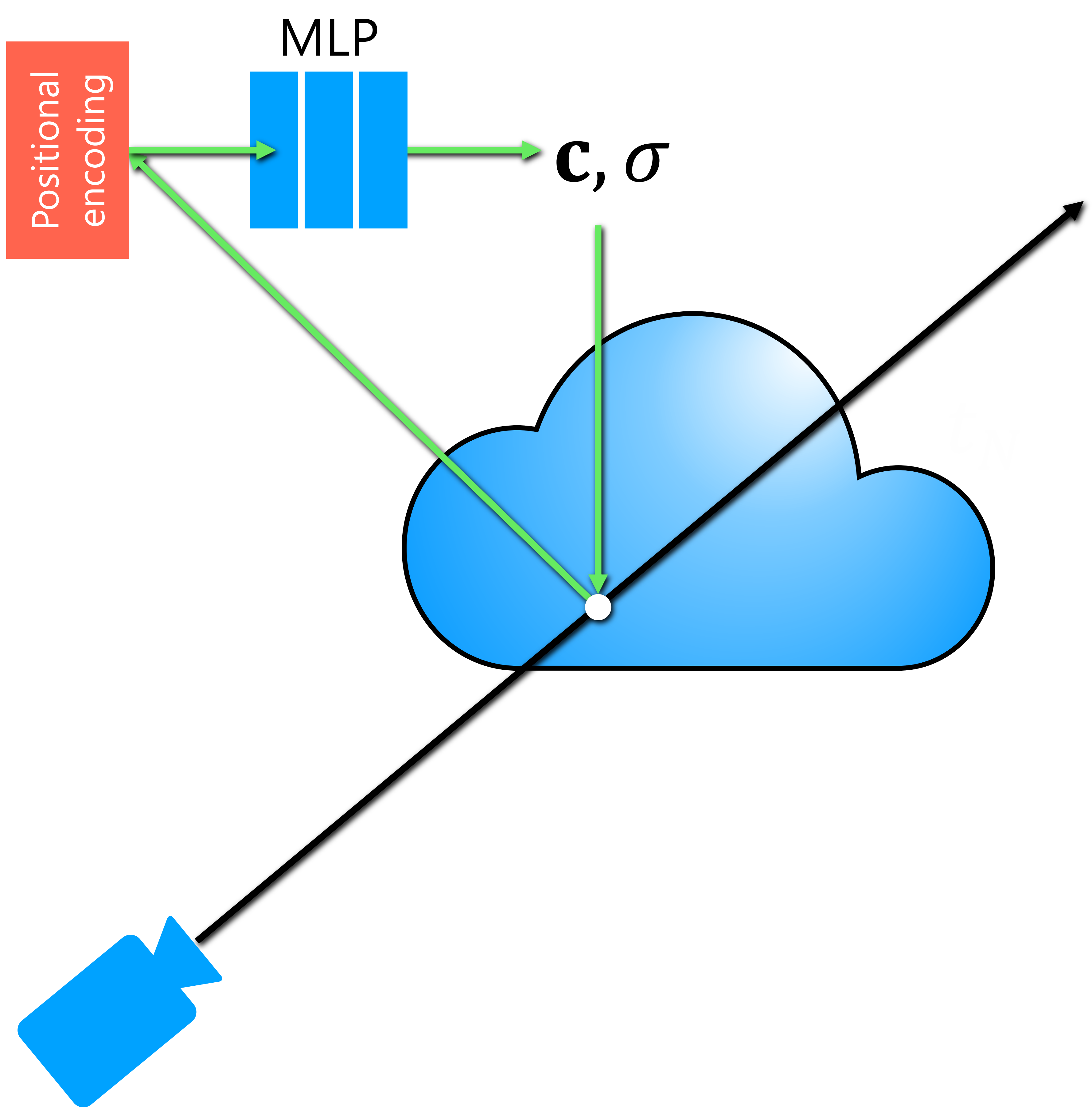
Neural network output with high frequency mapping

# NeRF Overview

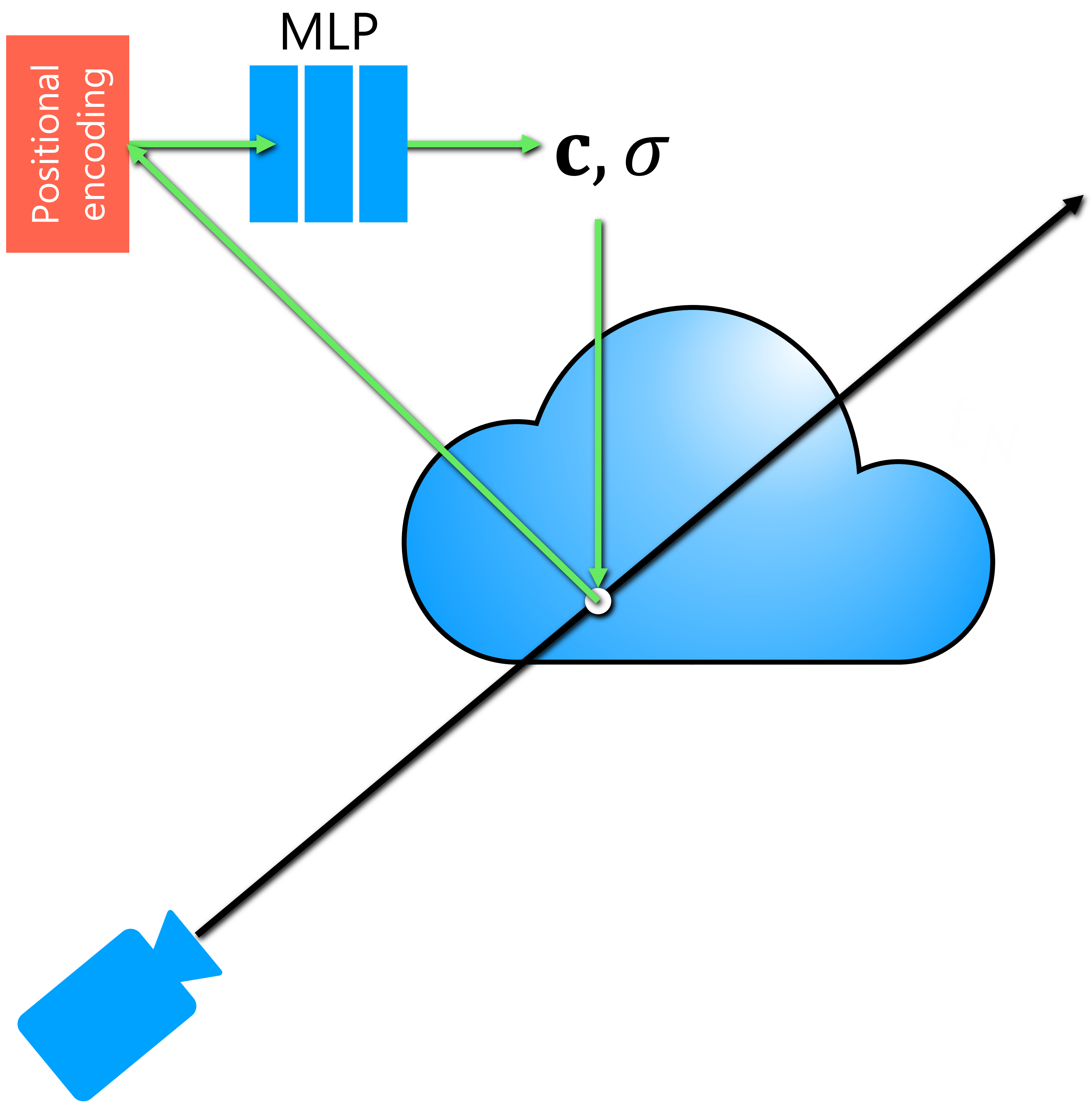
- Volumetric rendering
- Neural networks as representations for spatial data
- Neural Radiance Fields (NeRF)

NeRF = volume rendering + coordinate-based network

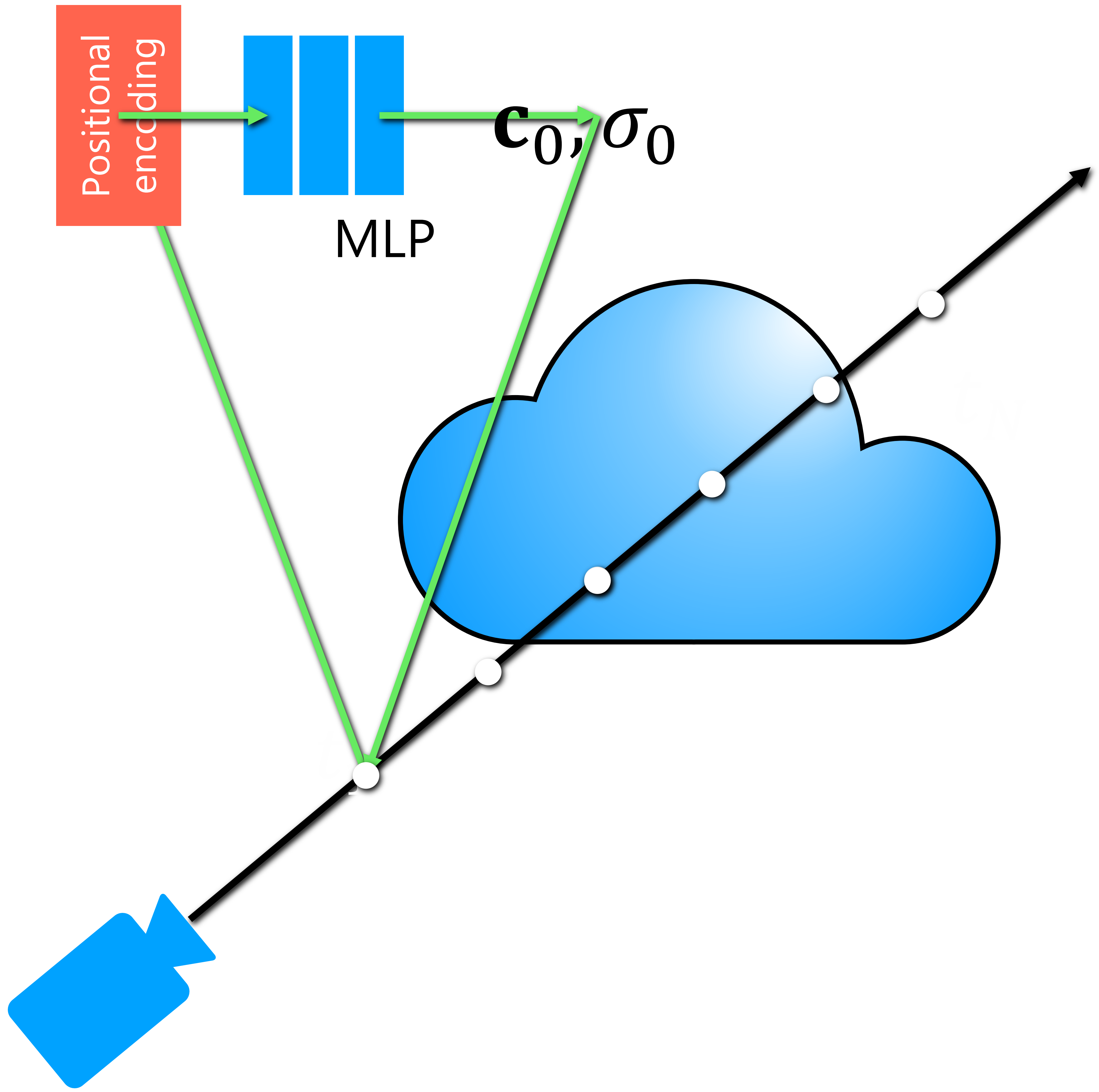
# How do we store the values of $\mathbf{c}, \sigma$ at each point in space



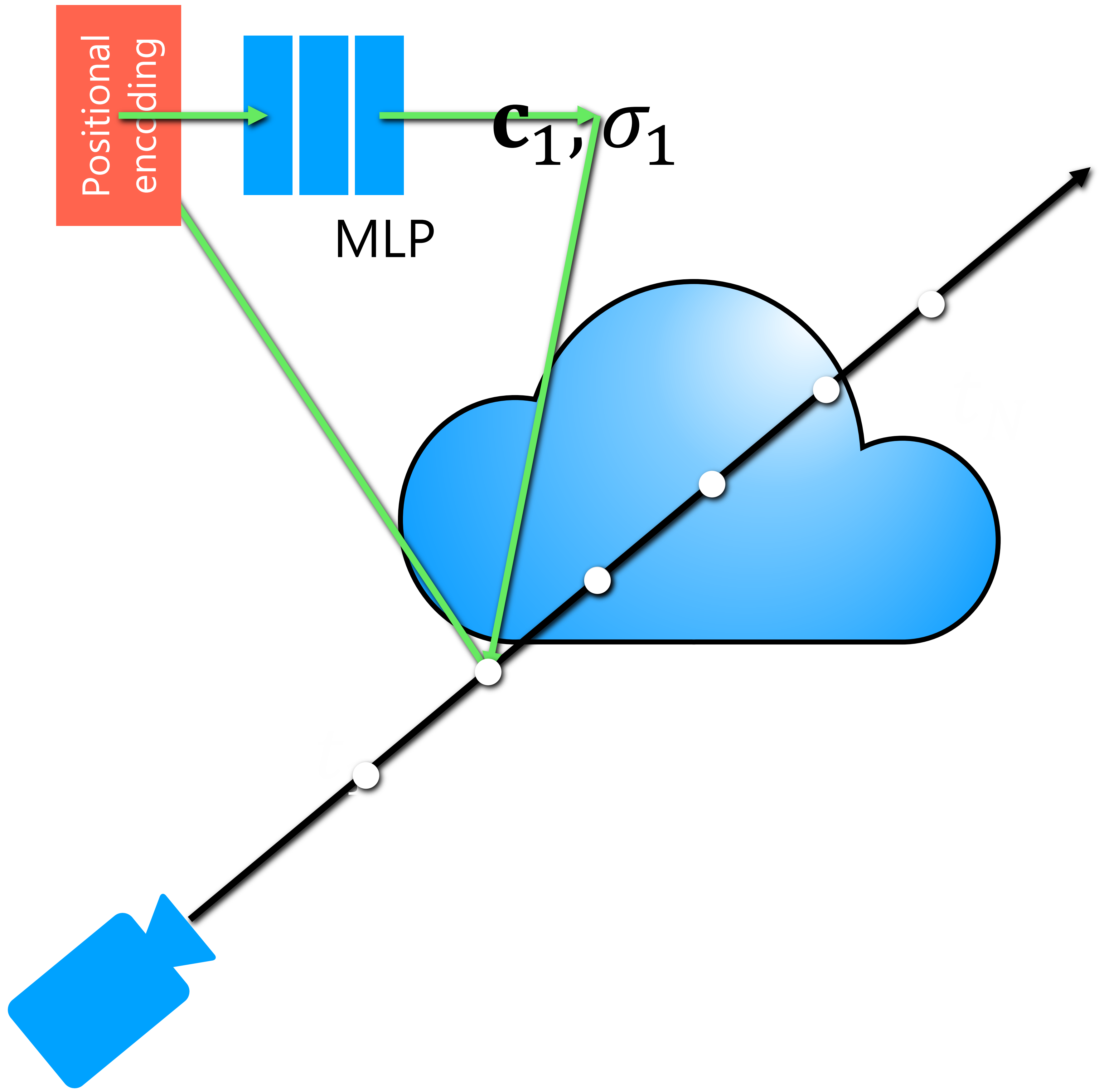
# How do we store the values of $\mathbf{c}, \sigma$ at each point in space



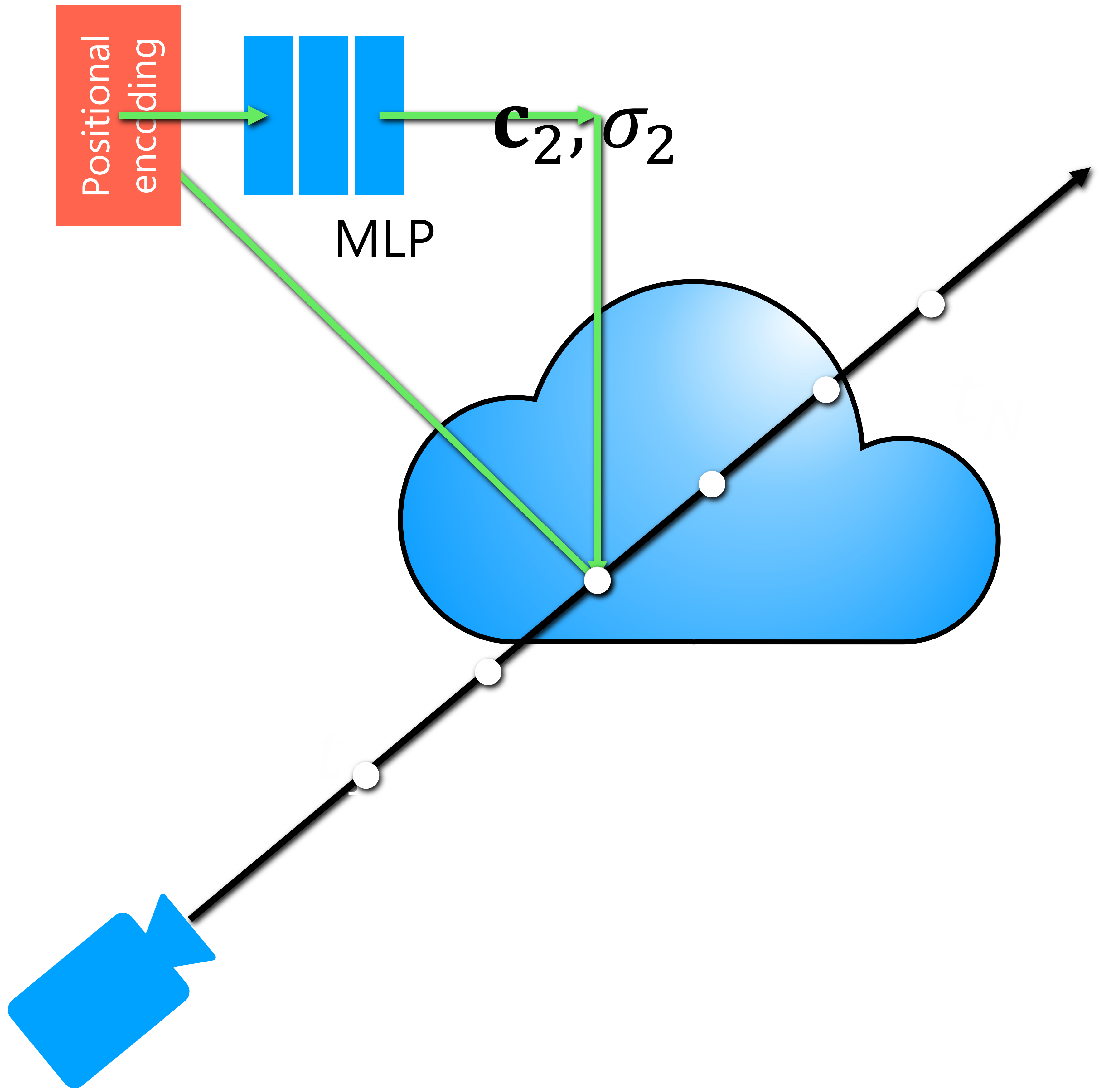
# How do we store the values of $\mathbf{c}, \sigma$ at each point in space



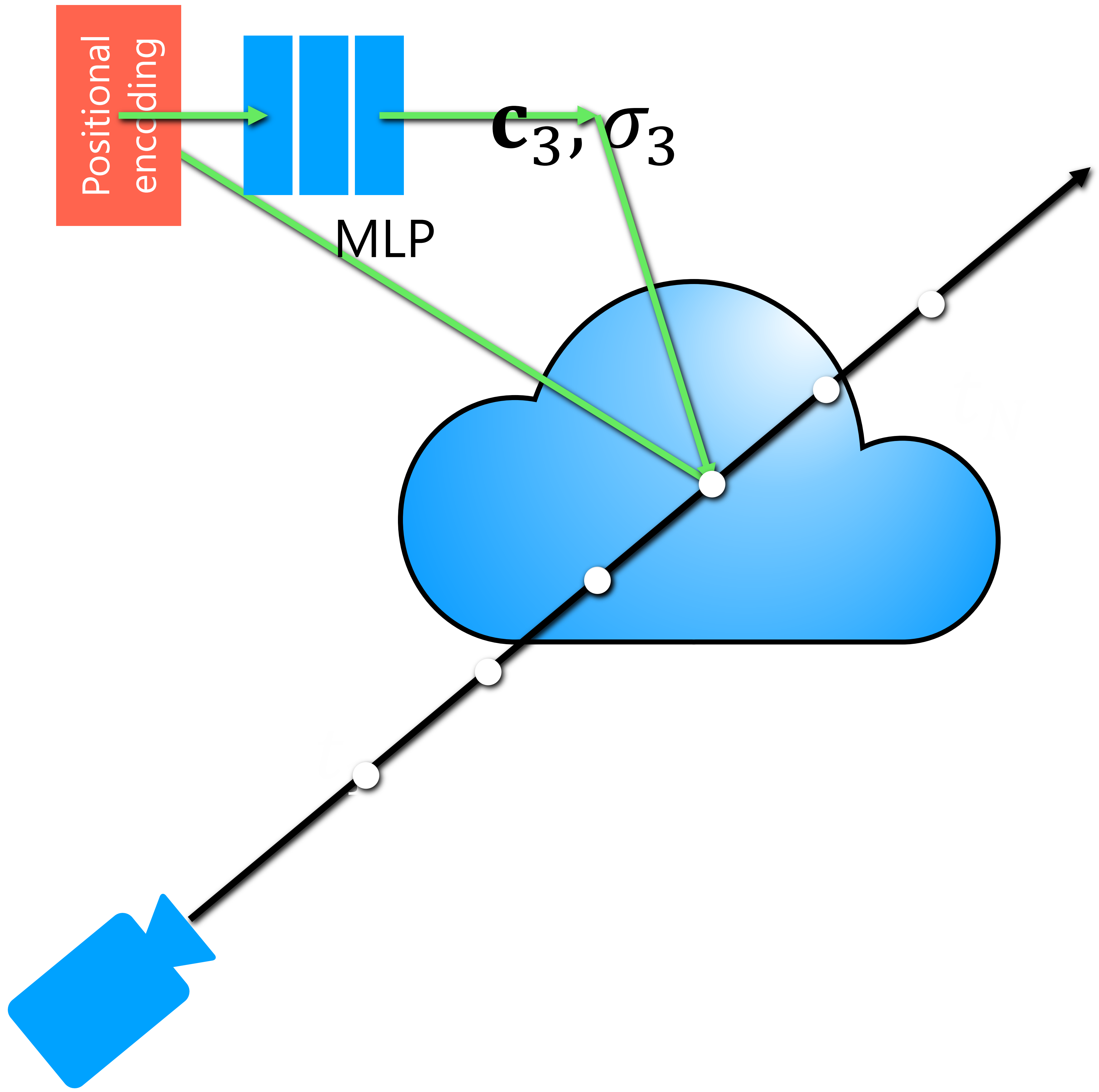
# How do we store the values of $\mathbf{c}, \sigma$ at each point in space



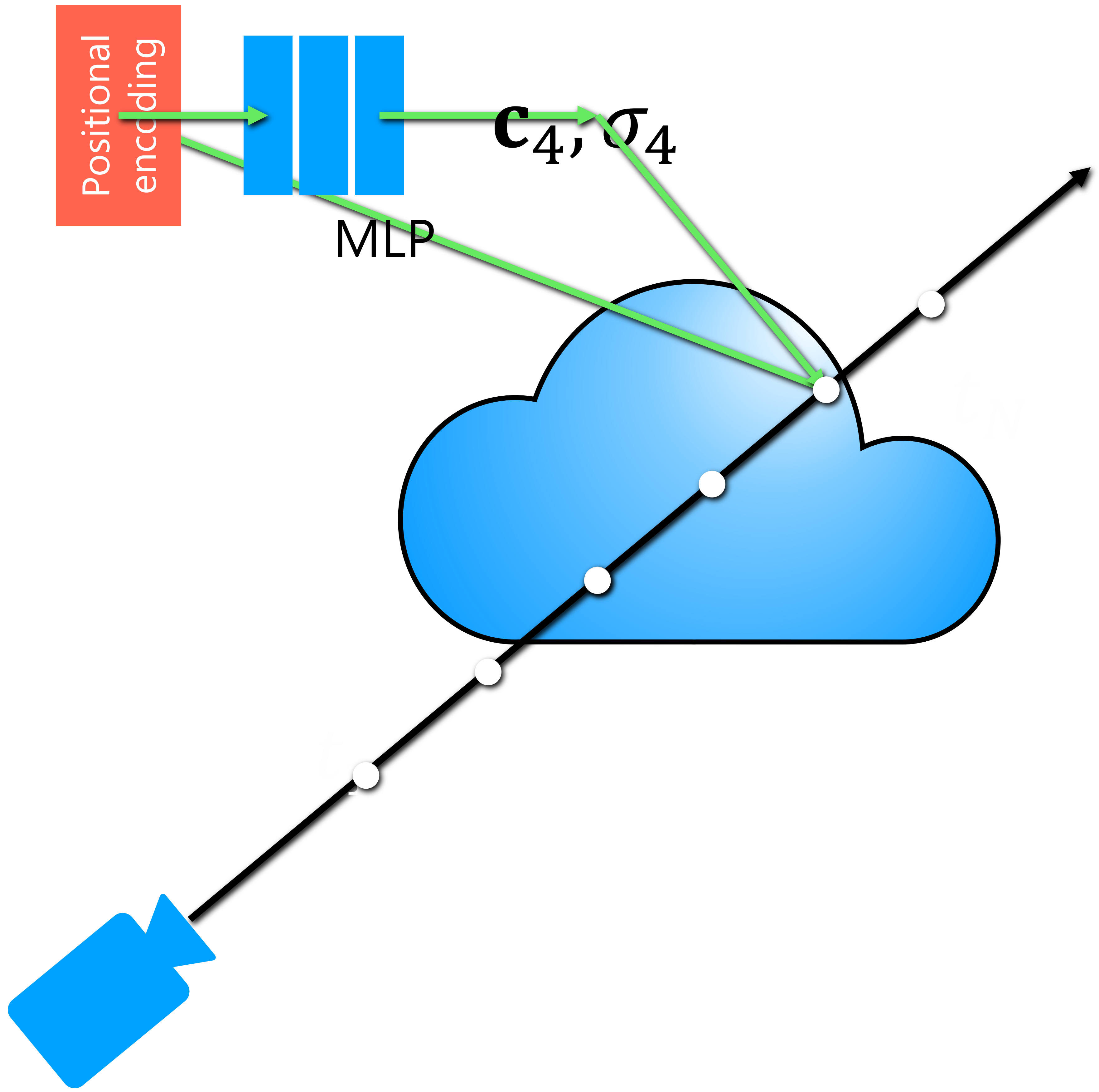
# How do we store the values of $\mathbf{c}, \sigma$ at each point in space



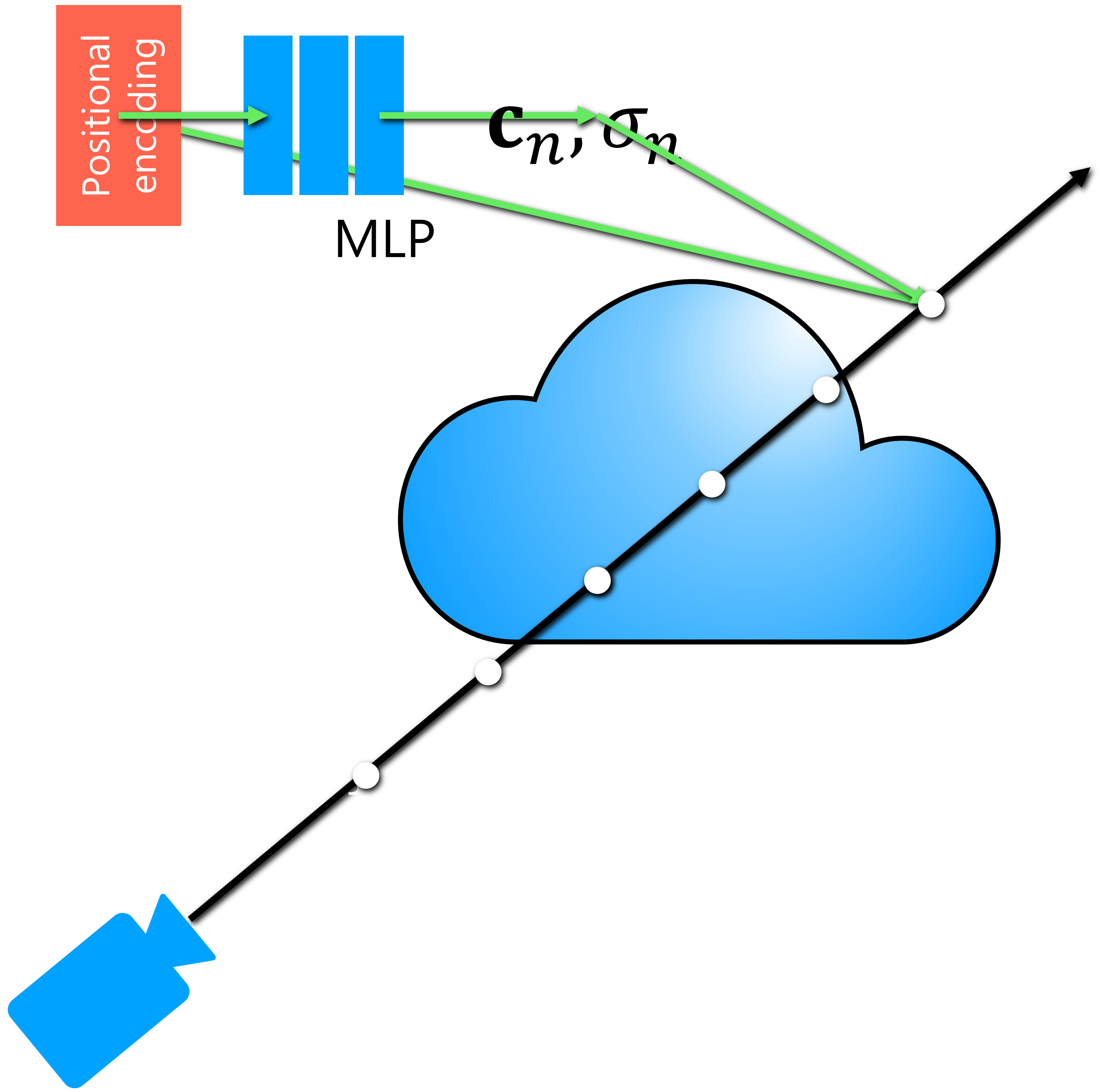
# How do we store the values of $\mathbf{c}, \sigma$ at each point in space



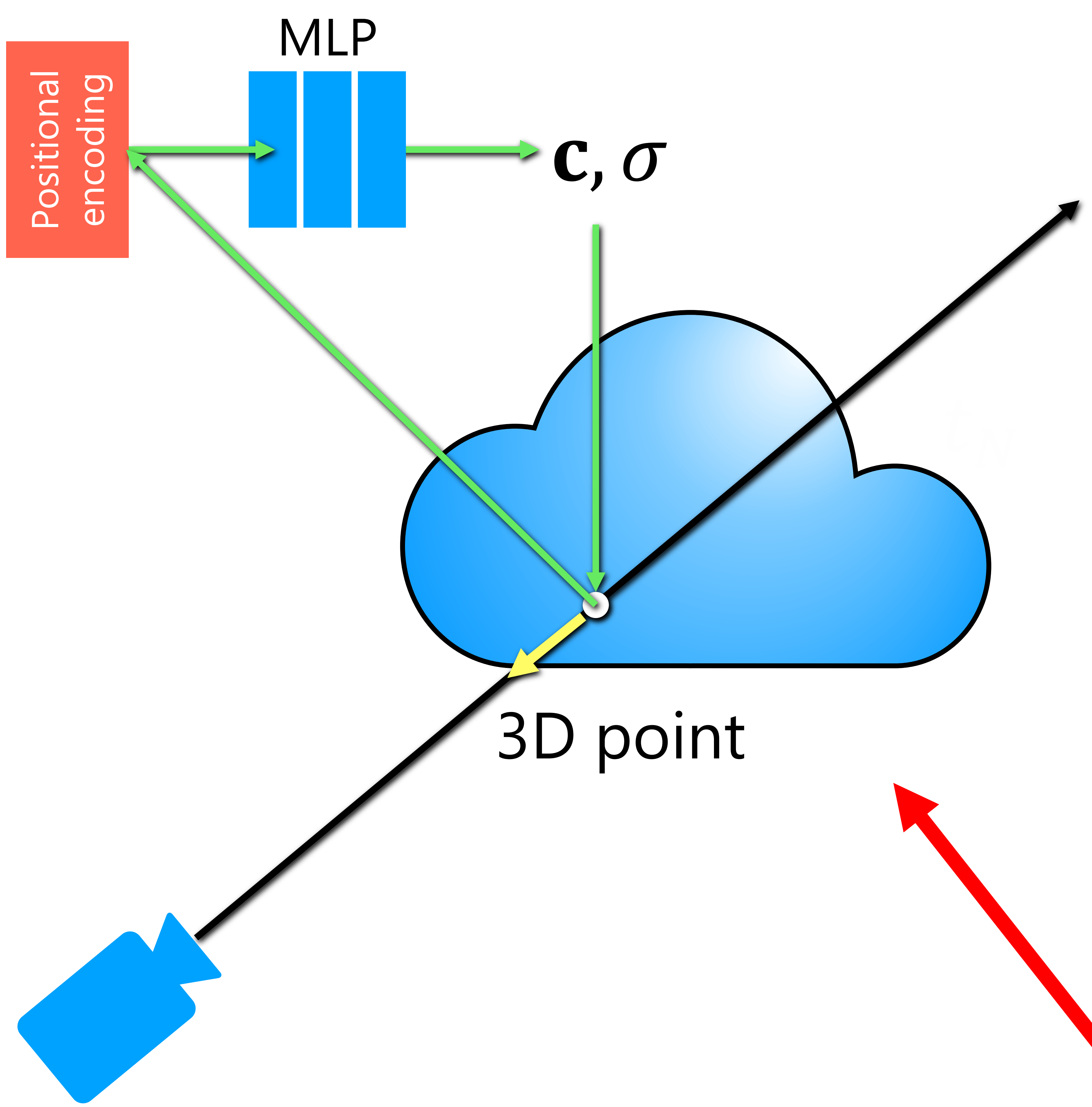
# How do we store the values of $\mathbf{c}, \sigma$ at each point in space



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



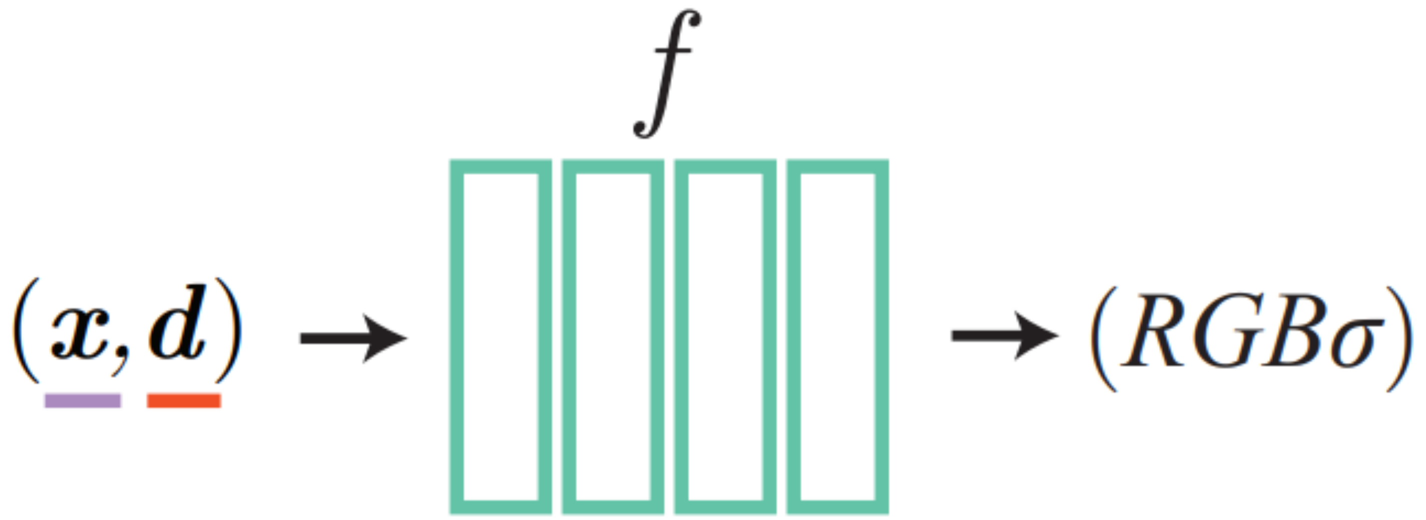
# Extension: view-dependent field

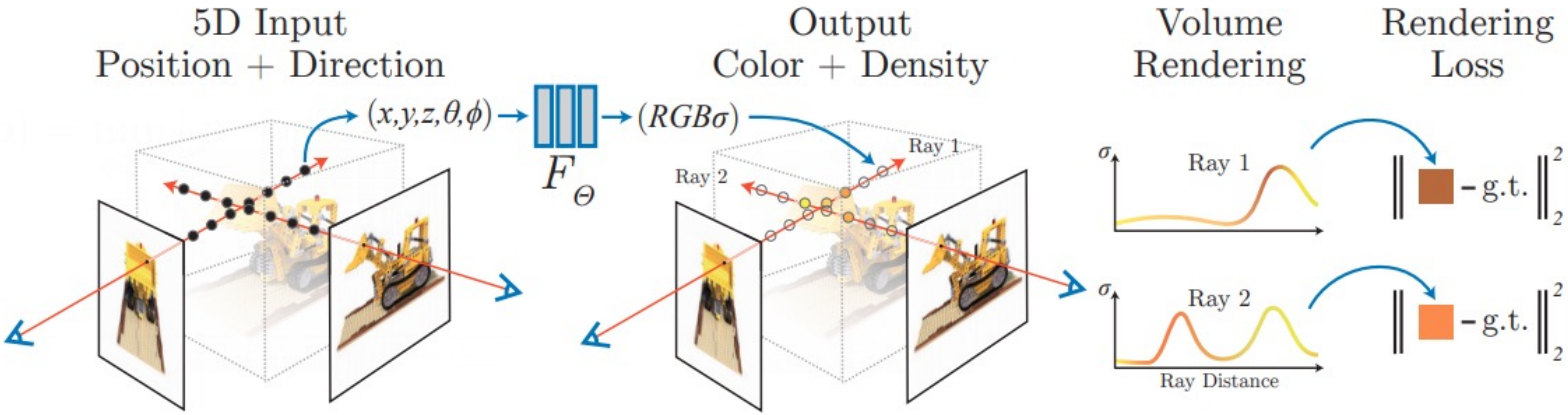



Include the ray direction in the input to the MLP → allows for capturing and rendering view-dependent effects (e.g., shiny surfaces)

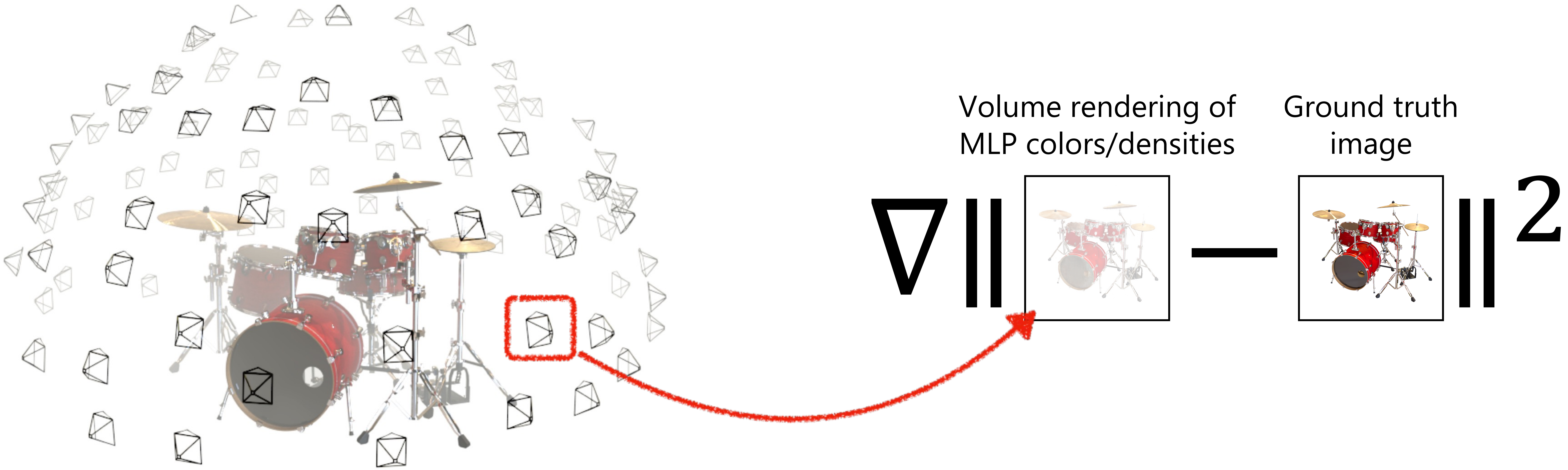
# Putting it all together

- Continuous neural networks as a view-dependent volumetric scene representation (position $\boldsymbol{x}$ + view direction $\boldsymbol{d}$)
- Using volumetric rendering to synthesize new views



# Train network using gradient descent to reproduce all input views of scene



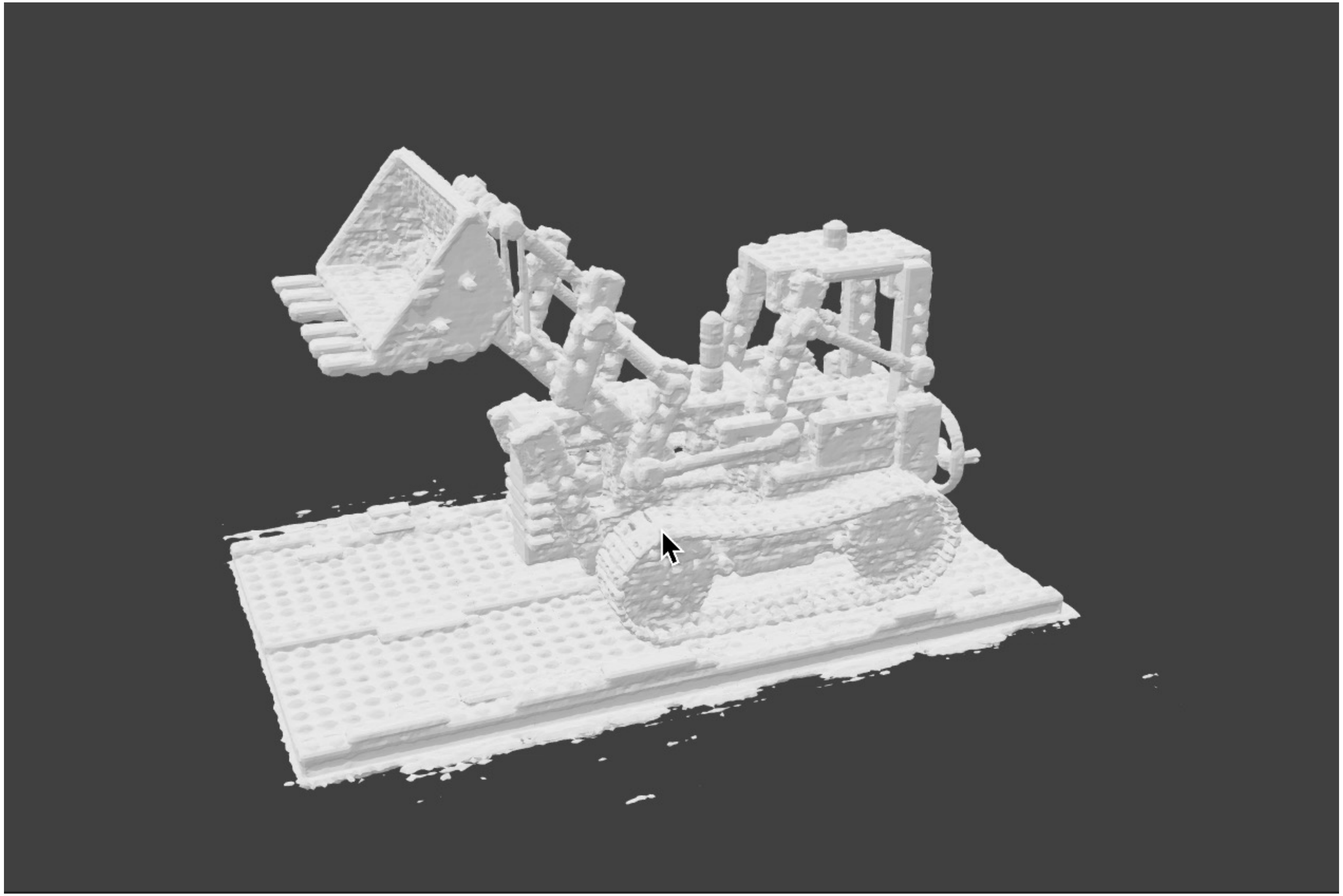
# Results

# NeRF encodes convincing view-dependent effects using directional dependence

# NeRF encodes detailed scene geometry with occlusion effects

# NeRF encodes detailed scene geometry with occlusion effects

# NeRF encodes detailed scene geometry



# 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

# It has been three years

- Original NeRF paper: 2750 citations in 3 years

# Handling Appearance Changes



Nerf-W [Martin-Brualla et al. CVPR 2021]

# Real-time Rendering

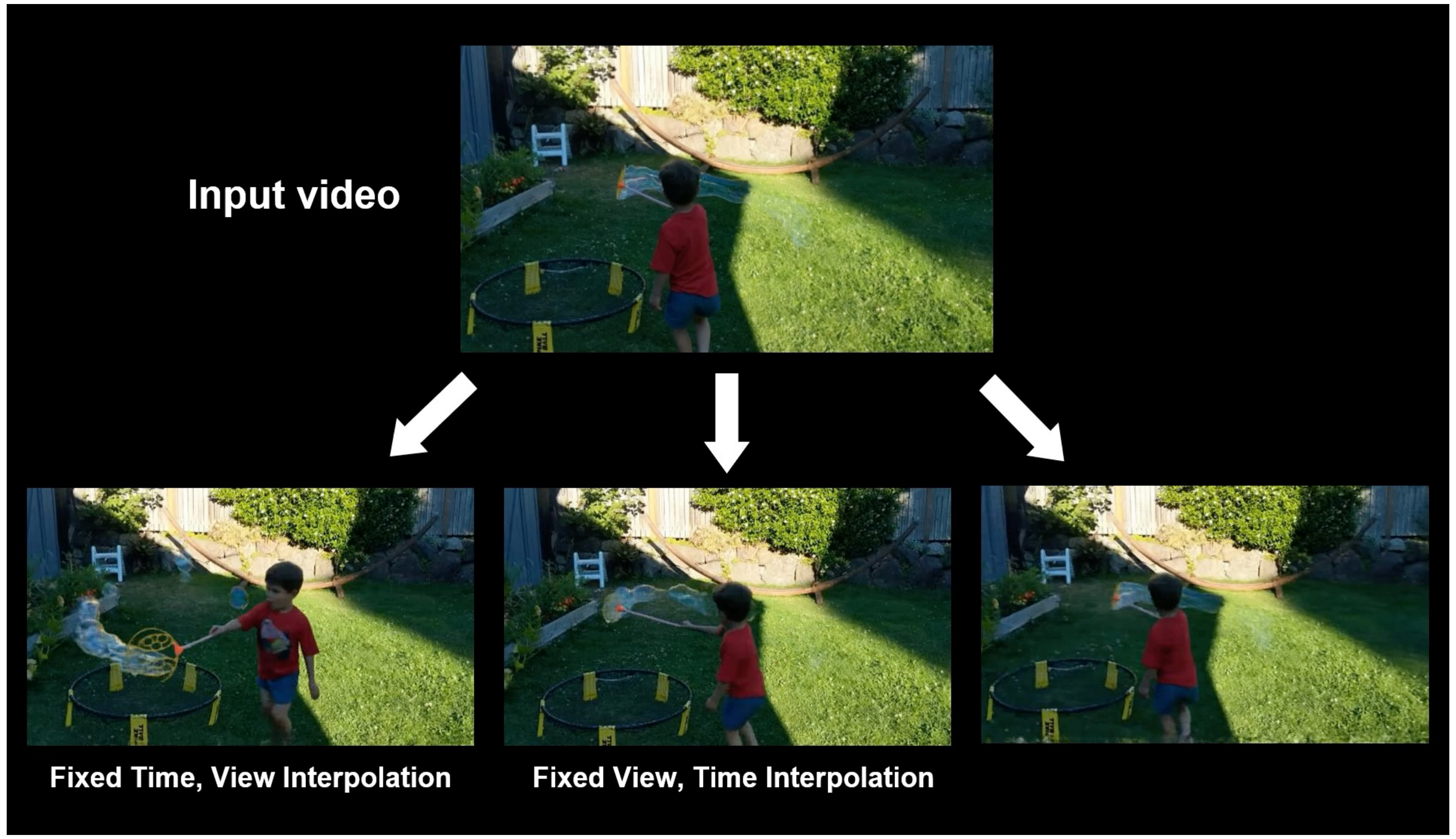



Video from PlenOctrees [Yu et al. CVPR 2021]

# Dynamic NeRFs

[Xian et al., CVPR 2021]

Nerfies [Park et al., ICCV 2021]    HyperNeRF [Park et al., SigAsia 2021]



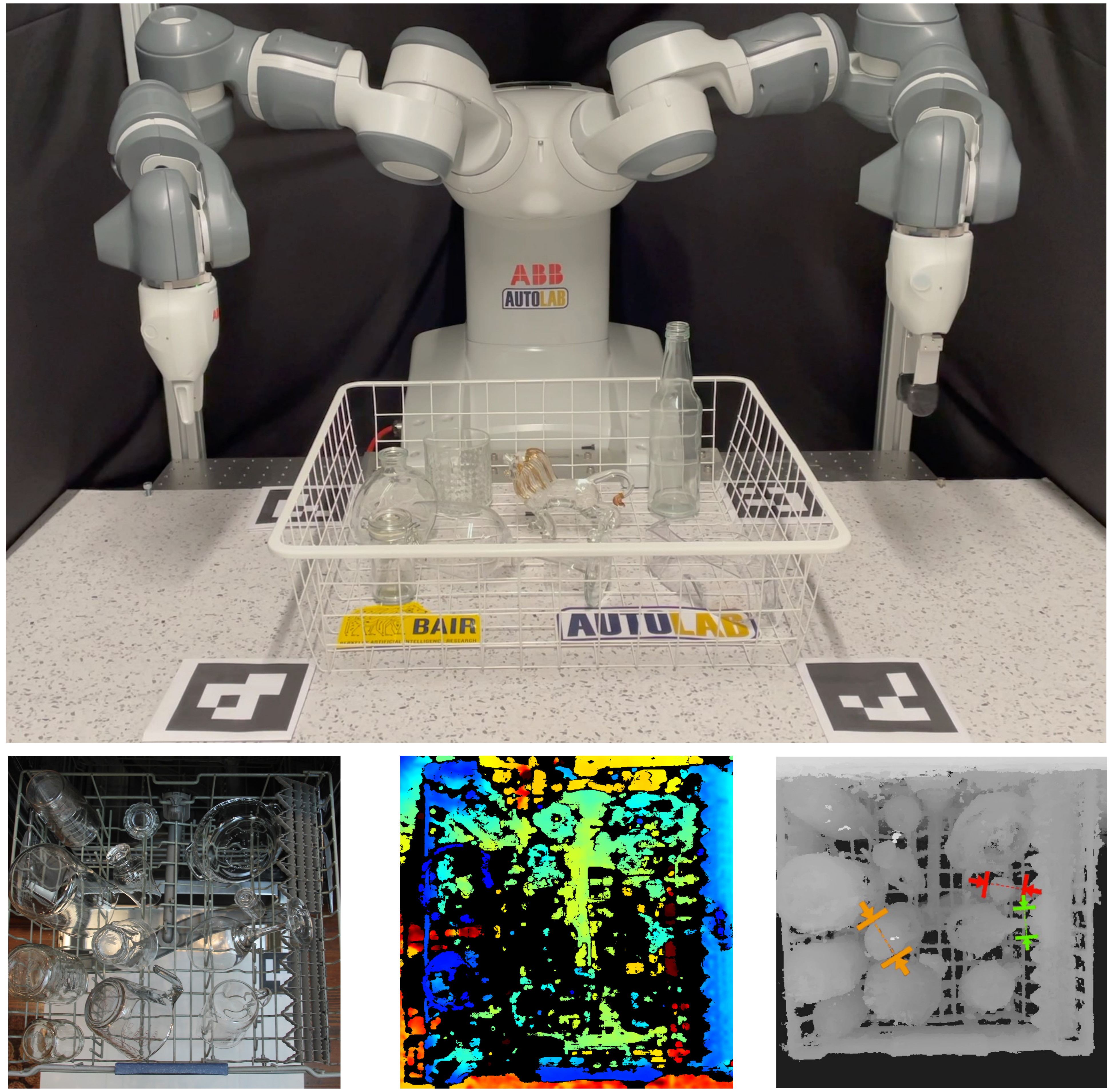

NSFF [Li et al., CVPR 2021]

# City-Scale NeRFs

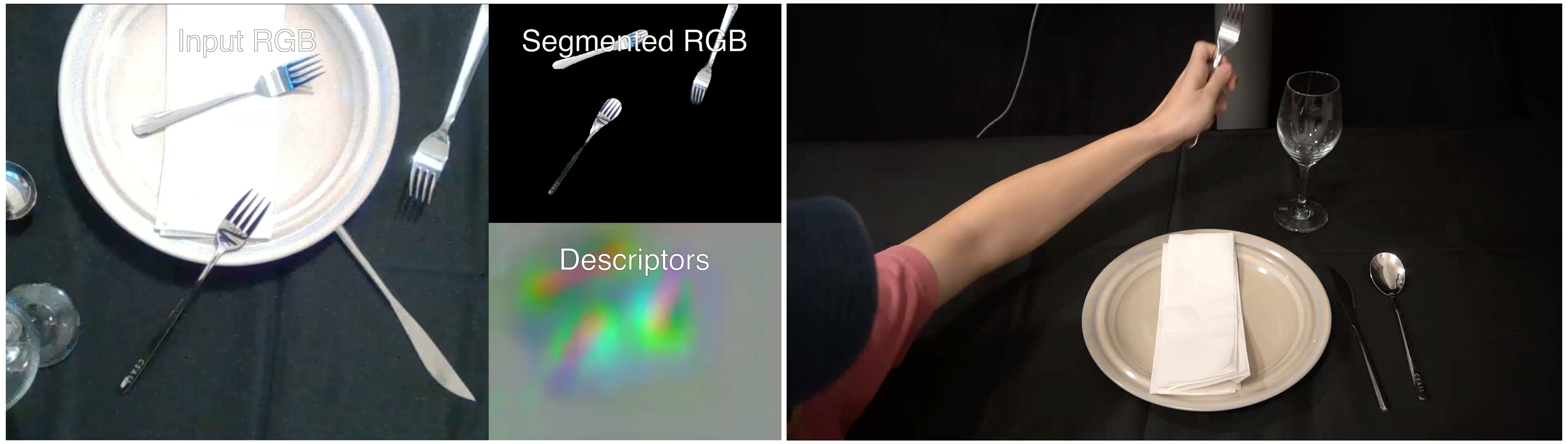
BlockNeRF [Tancik et al. CVPR 2022]

RawNeRF [Mildenhall et al. CVPR 2022]

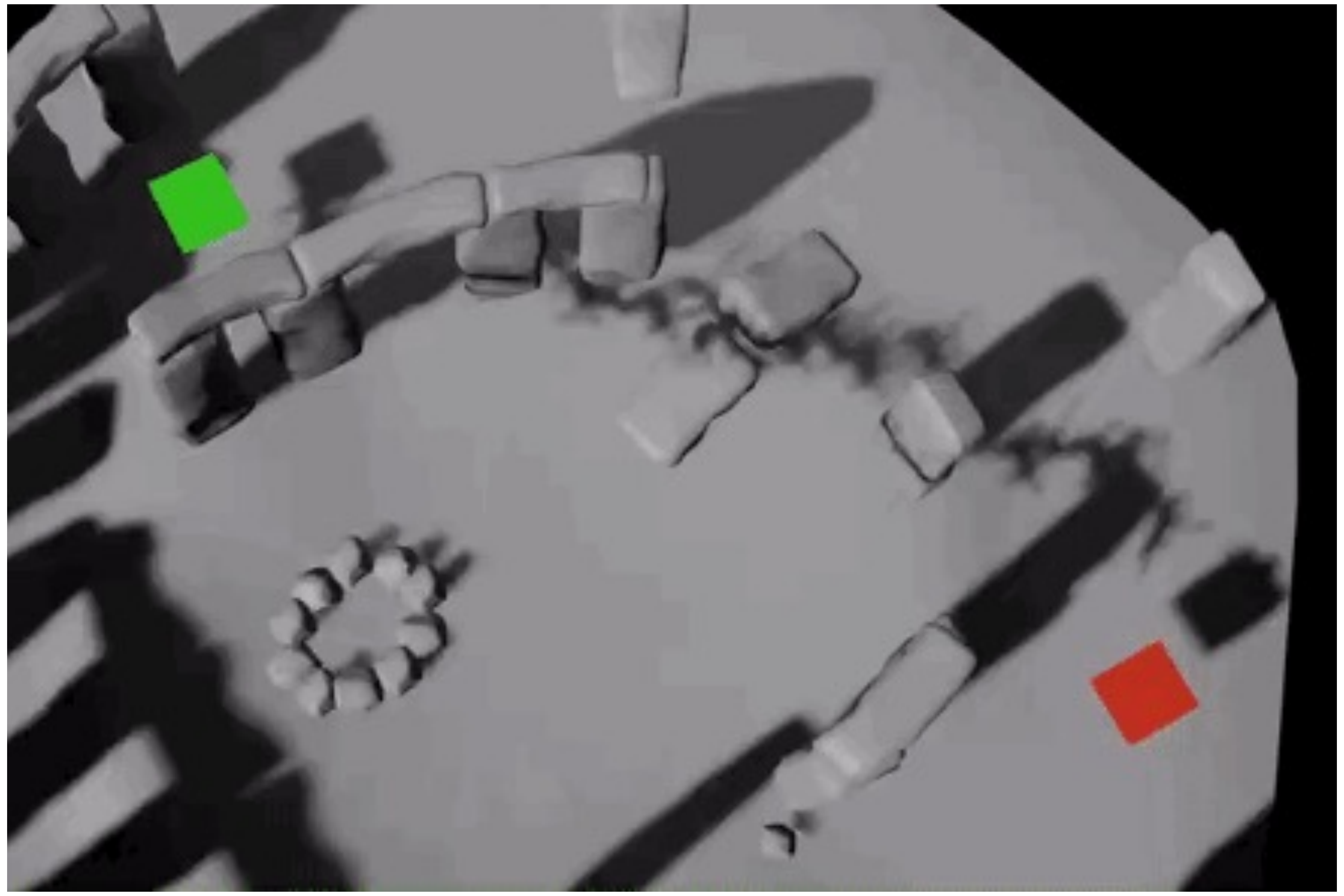
# Robotics



Dex-NeRF: Using a Neural Radiance field to Grasp Transparent Objects, [Ichnowski and Avigal et al. CoRL 2021]




NeRF-Supervision: Learning Dense Object Descriptors from Neural Radiance Fields, [Yen-Chen et al. ICRA 2022]

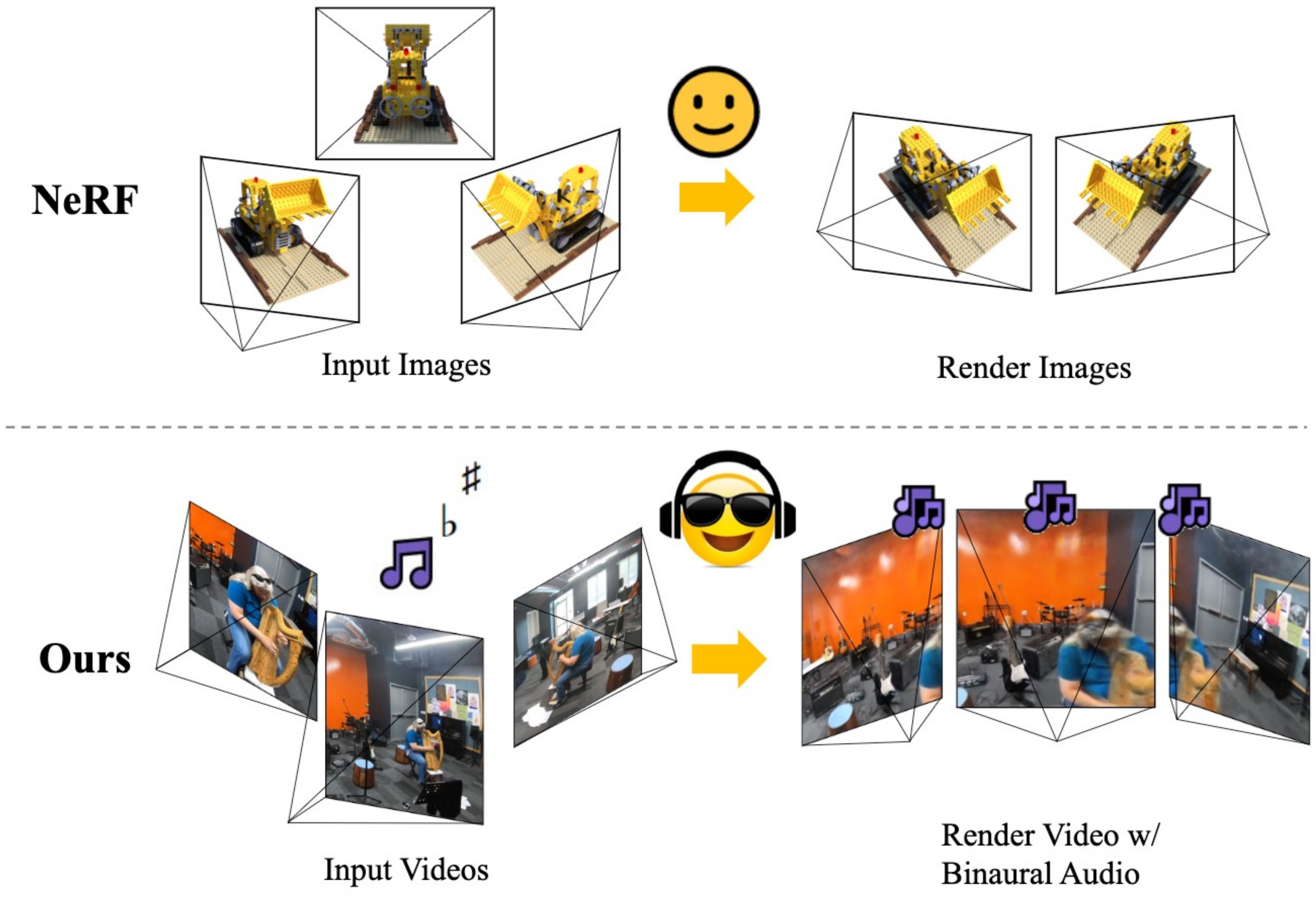


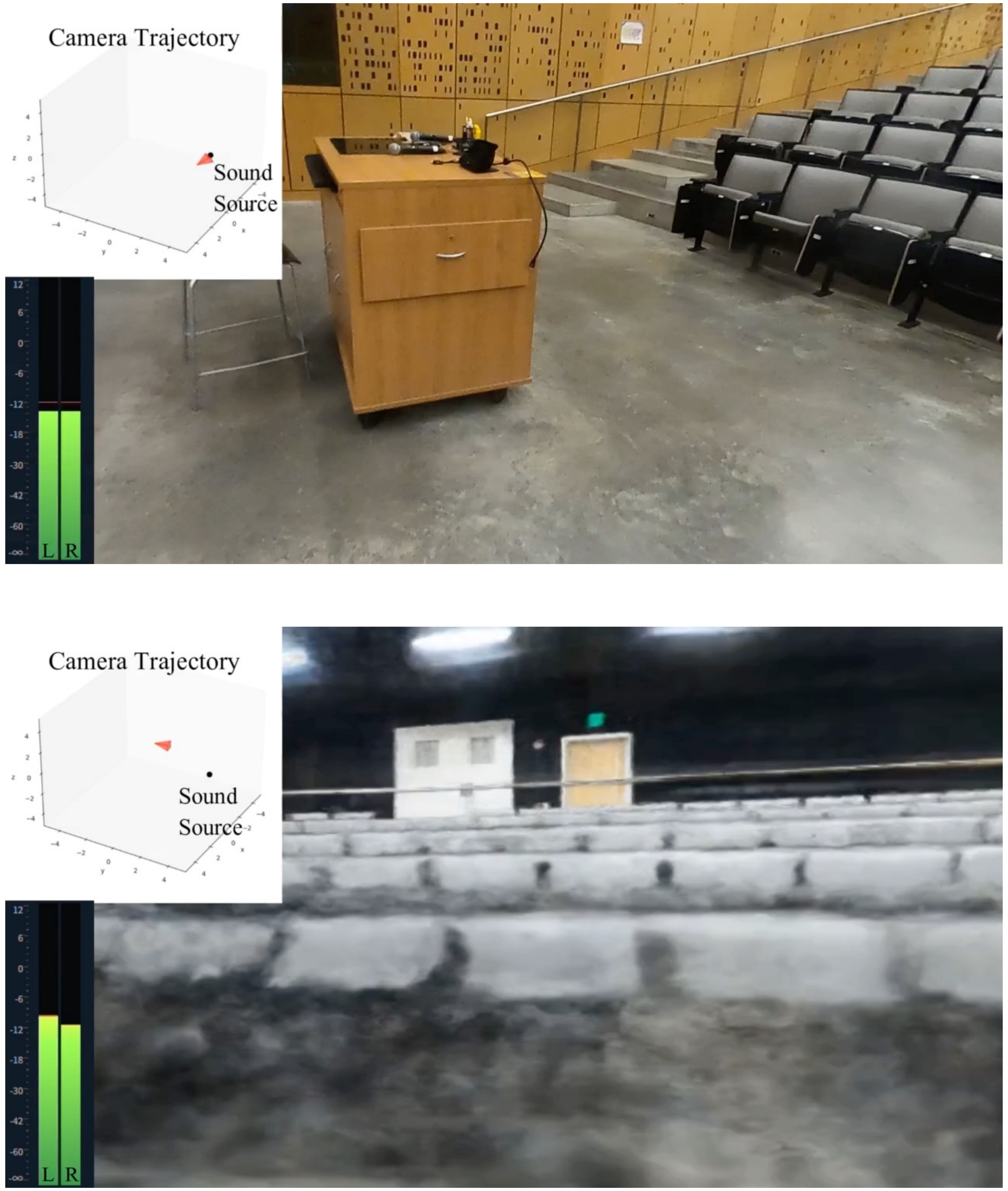
Vision-Only Robot Navigation in a Neural Radiance World [Adamkiewicz and Chen et al. ICRA 2022]

Generating 3D scenes with diffusion models

DreamFusion [Poole et al. ICLR 2023]

# Beyond Visual




Given the position (x,y,z) and viewing direction (θ,ϕ) of a listener, our method can render an image the listener would see and the corresponding binaural audio the listener would hear.

AV-NeRF [Liang et al. ArXiv 2023]

# Reading List & Implementation

- https://github.com/awesome-NeRF/awesome-NeRF
- https://sites.google.com/berkeley.edu/nerf-tutorial/home
- https://docs.nerf.studio/en/latest/