

Computer Vision

Inverse Graphics & Neural Radiance Fields

(NeRFs)



NeRF Slides adapted from material courtesy of Pratul Srinivasan

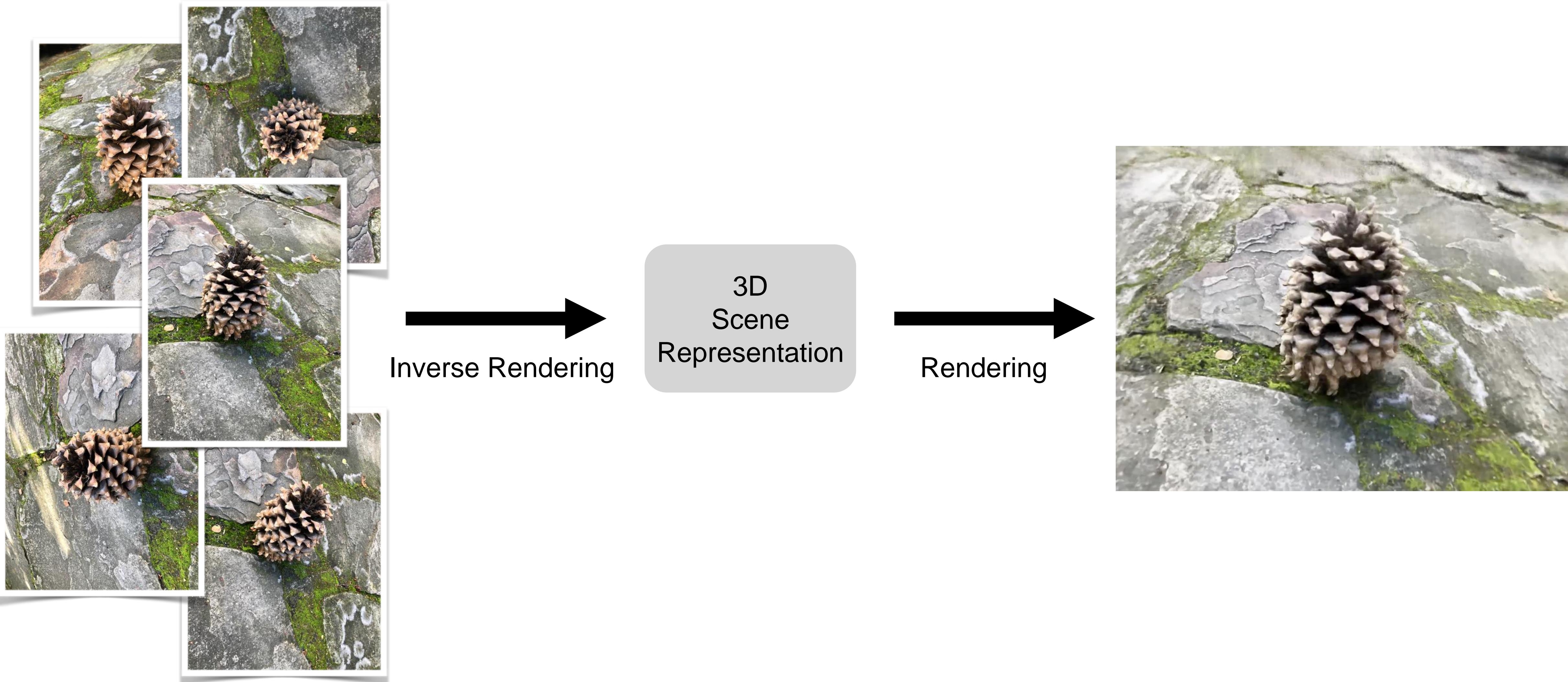
Rendering in computer graphics

3D
Scene
Representation

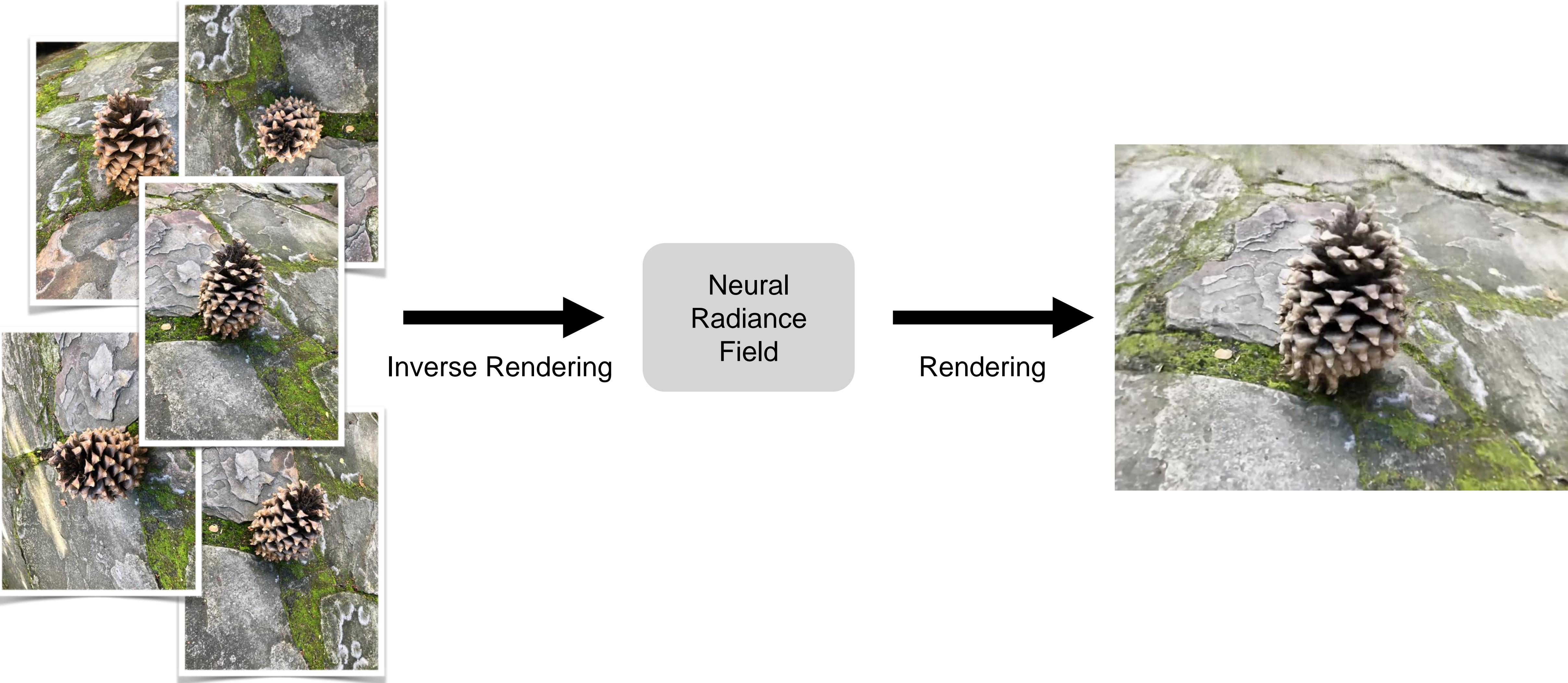
→
Rendering



Computer vision as inverse rendering



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



Adapted from material from Pratul Srinivas

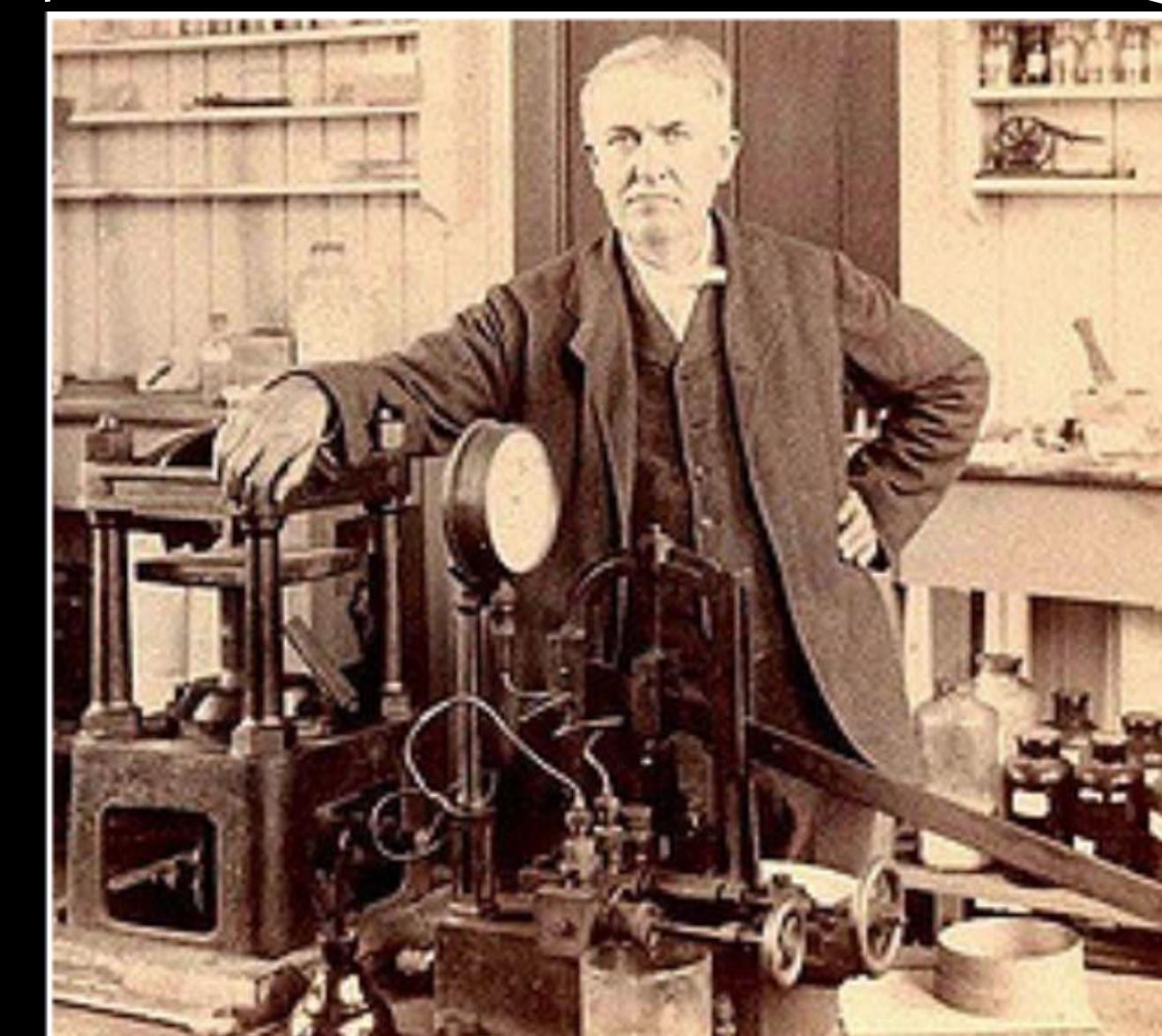
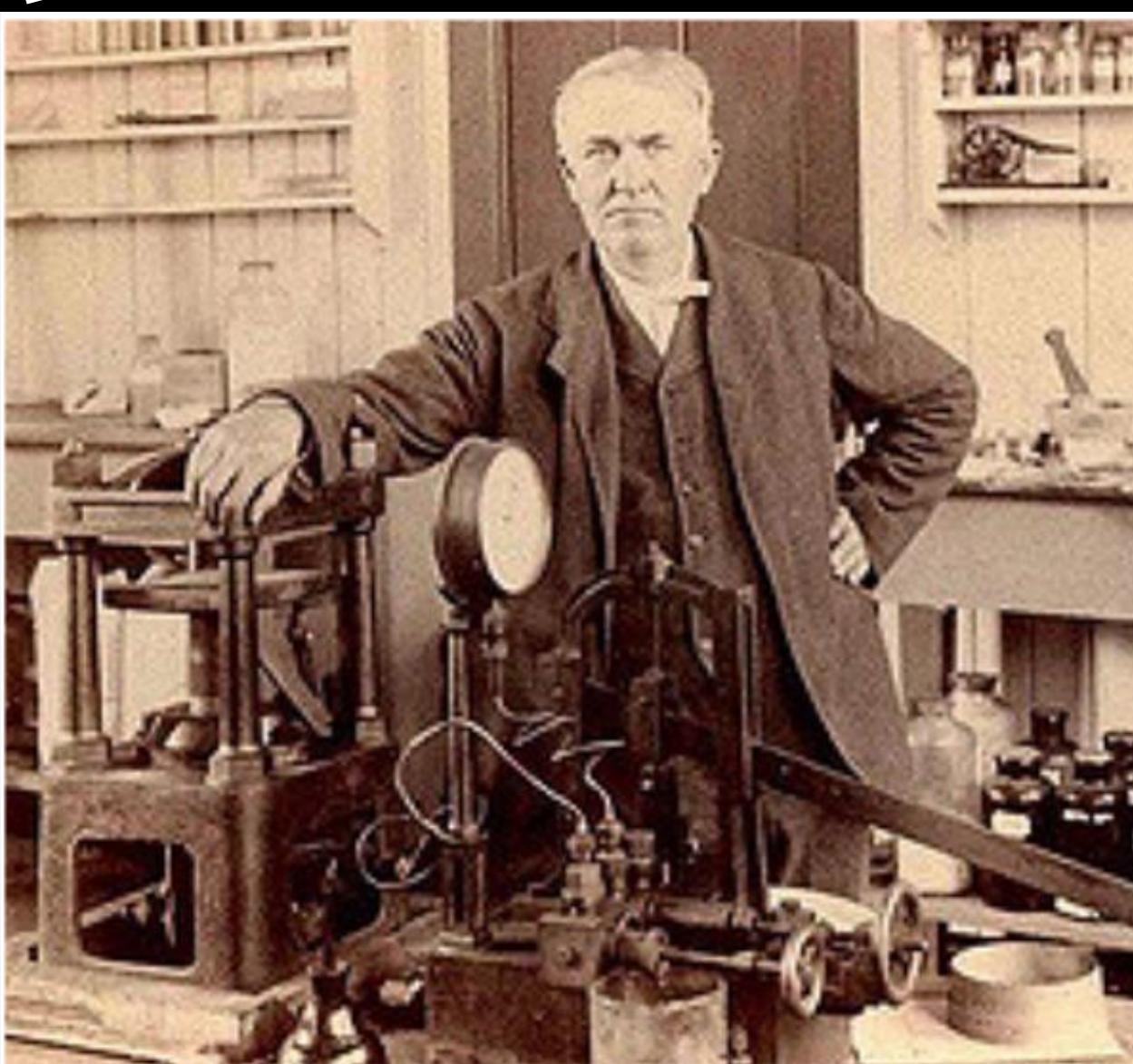
Deep learning for 3D reconstruction

- Previously: we reconstruct geometry by running stereo or multi-view stereo on a set of images
 - “Classical” approach
- How can we leverage powerful tools of deep learning?
 - Deep neural networks
 - GPU-accelerated stochastic gradient descent

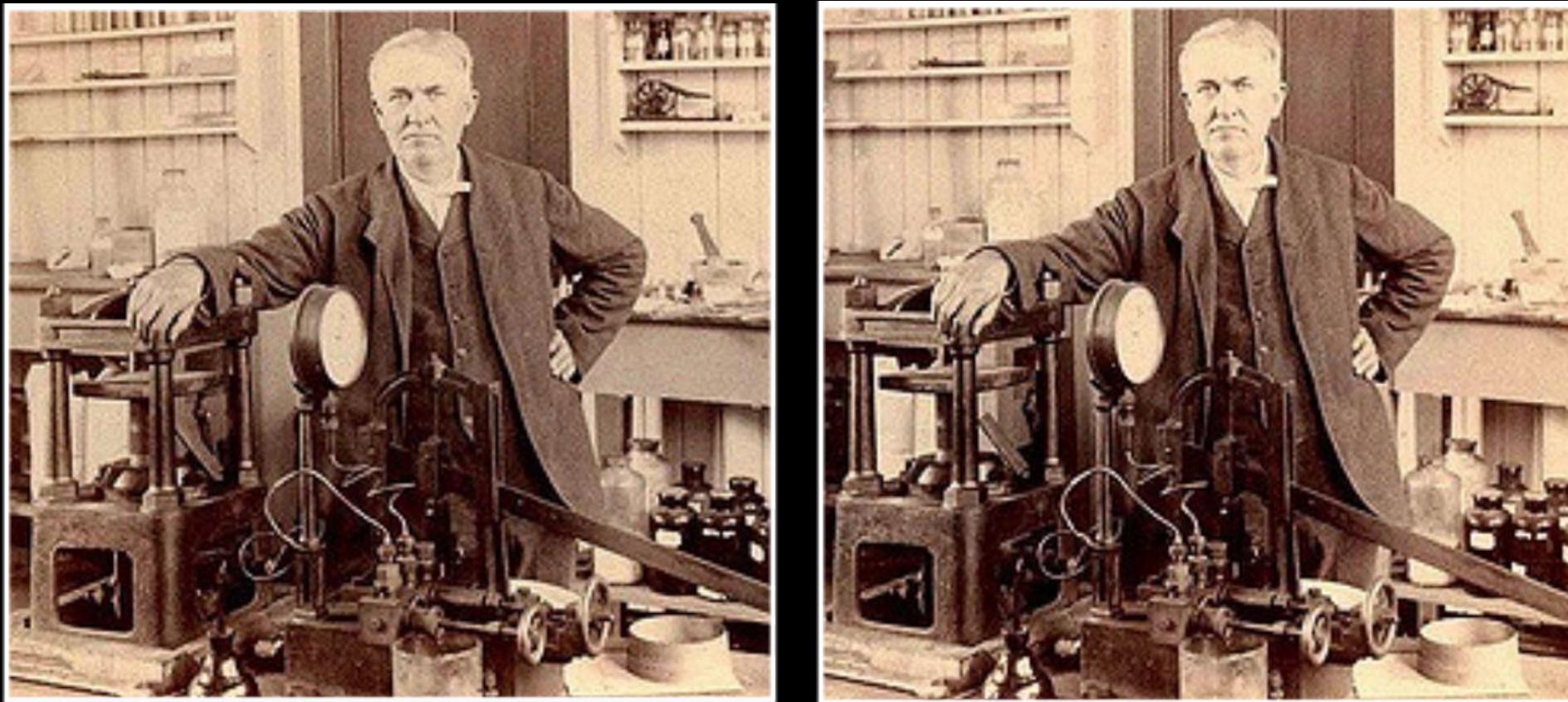
NeRF and related methods – Key ideas

- We need to create a loss function and a scene representation that we can optimize using gradient descent to reconstruct the scene
- *Differentiable rendering*

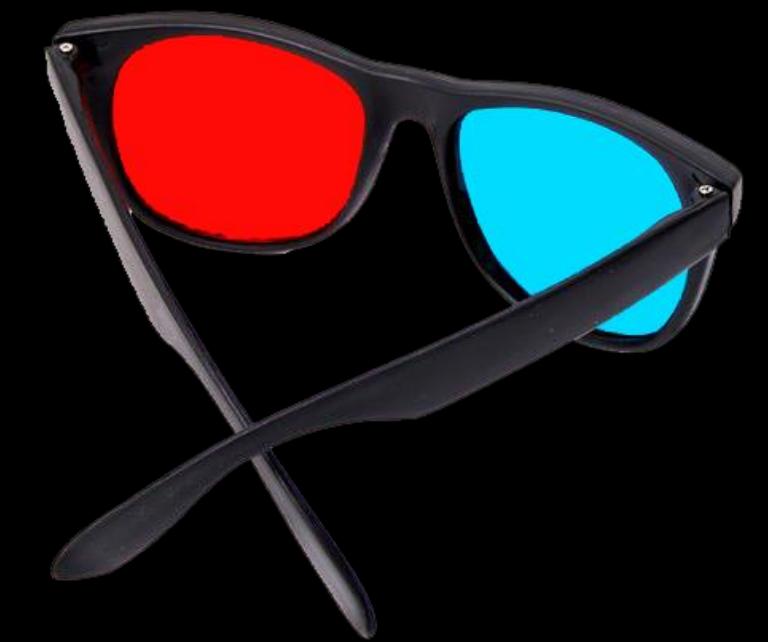
Side Topic: Stereo Photography



Stereo Photography



Viewing Devices

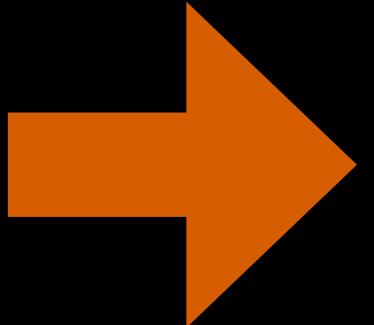


Stereo Photography



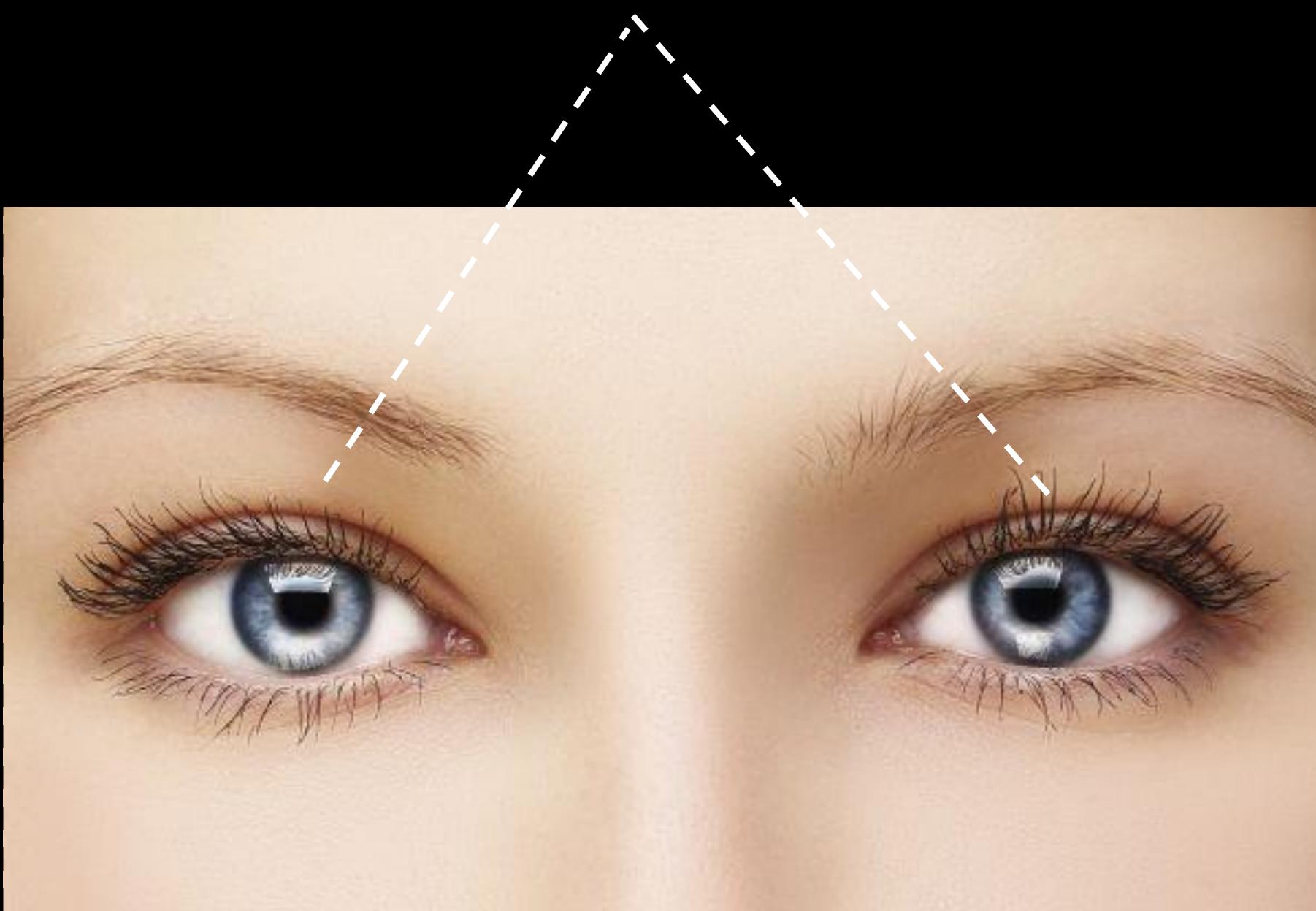
Queen Victoria at World Fair, 1851

Stereo Photography



Issue: Narrow Baseline

~6.5 cm



~1.5 cm



Left



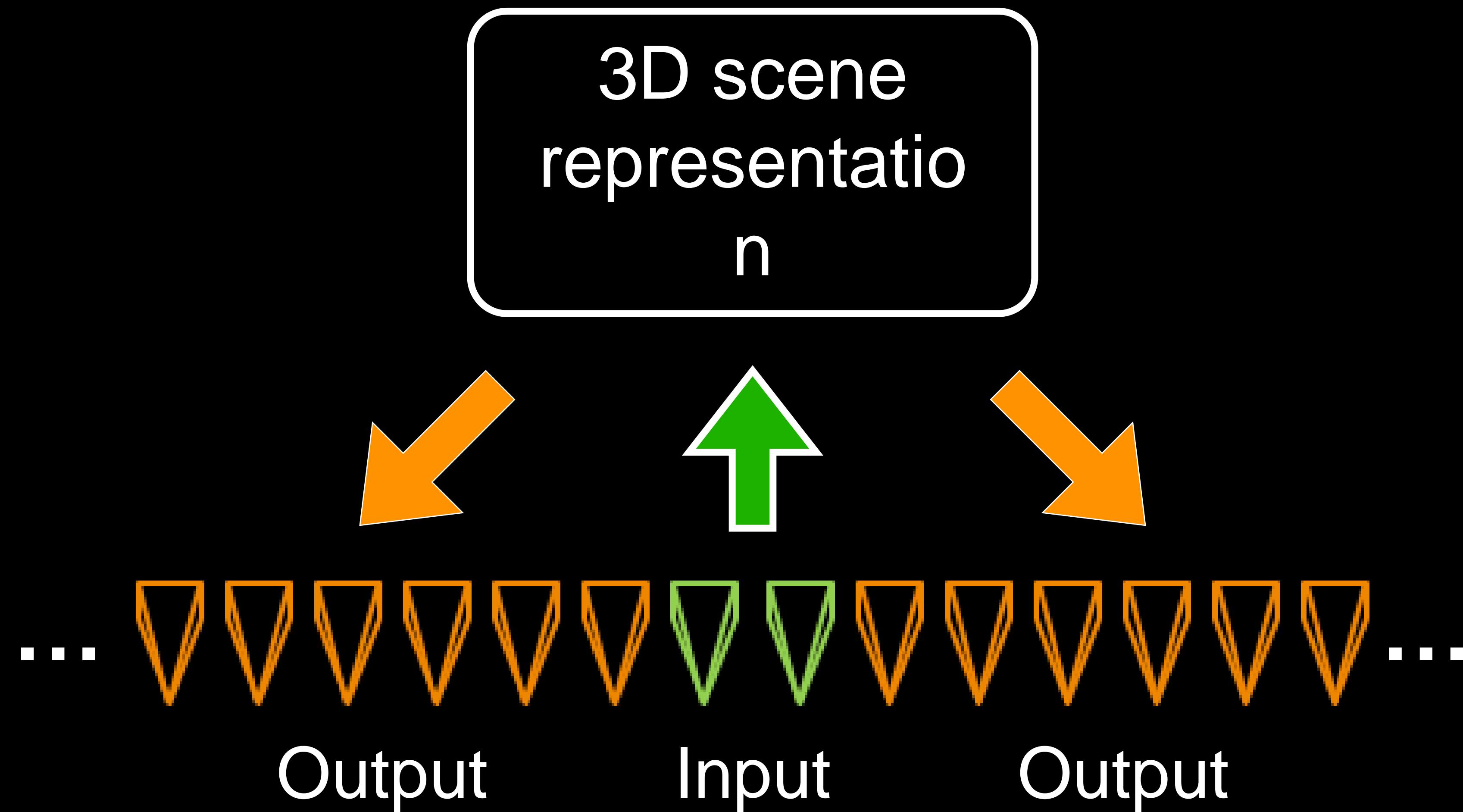
Right



Output

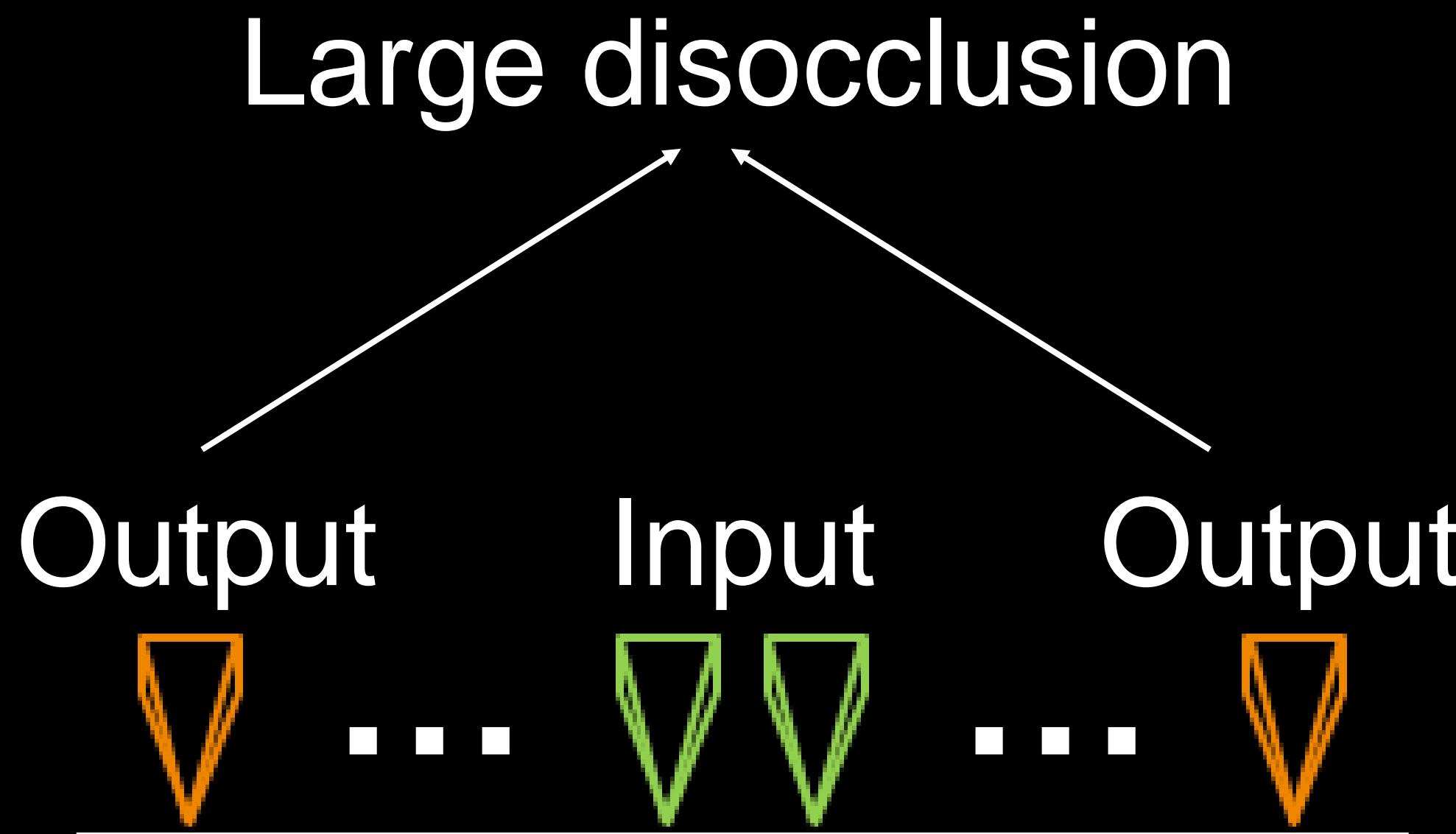


Problem Statement



Challenges

Extrapolation

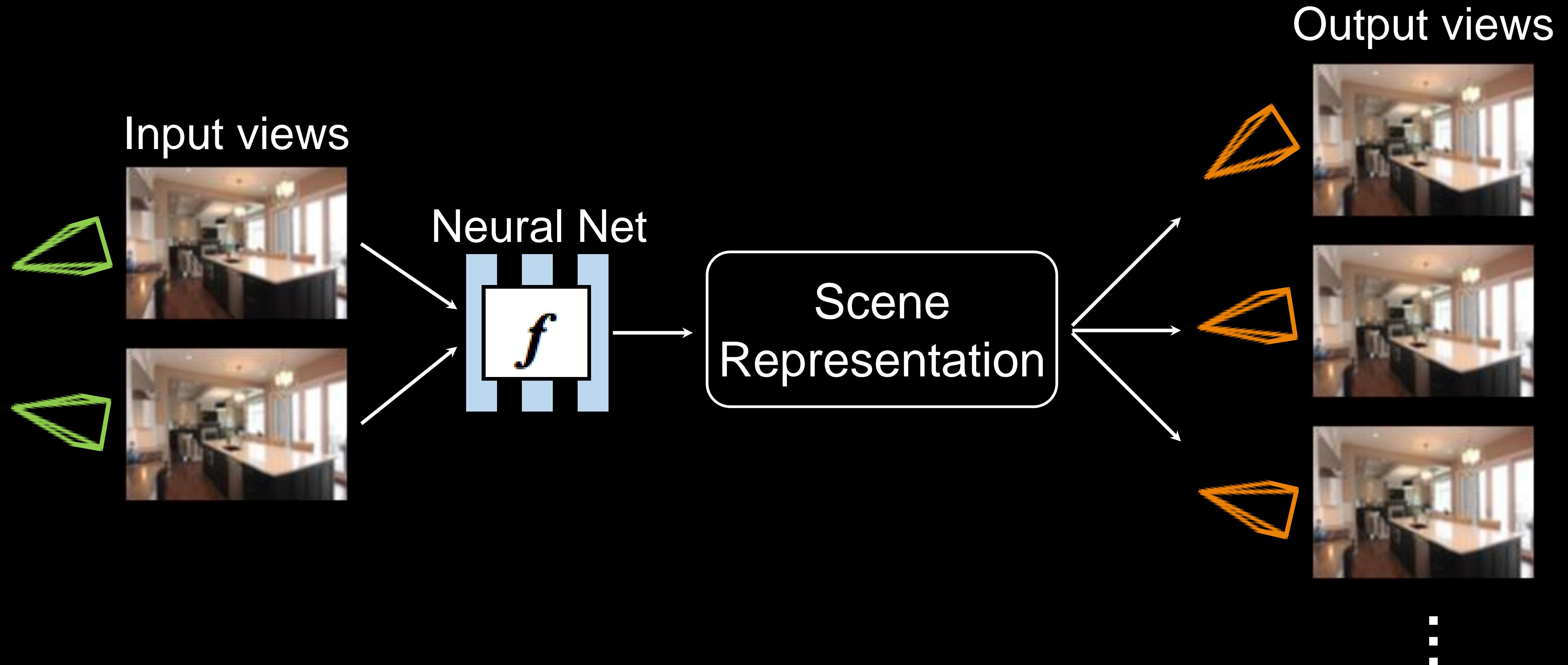


Non-Lambertian Effects

Reflections, transparencies,
etc



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 Camera (1937)

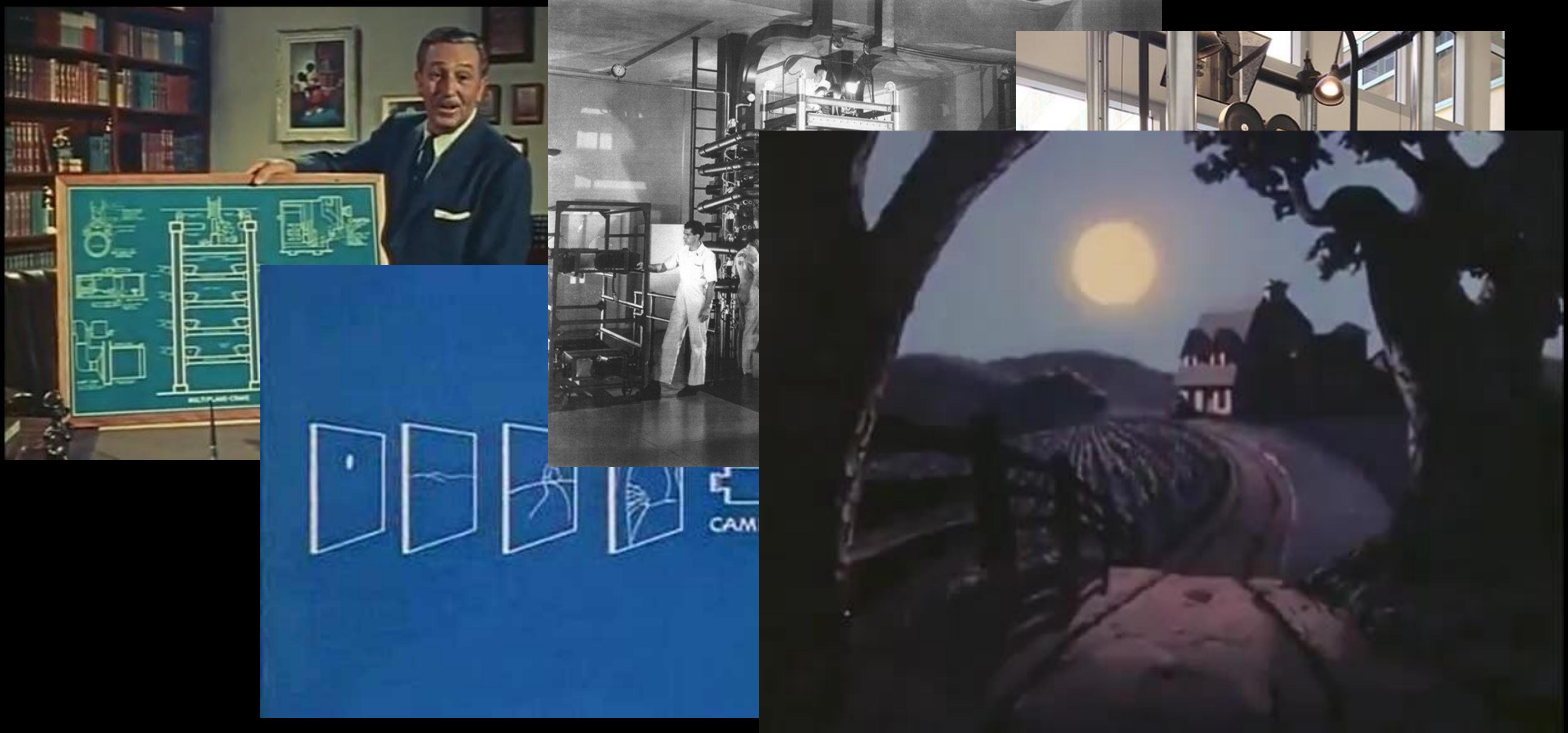
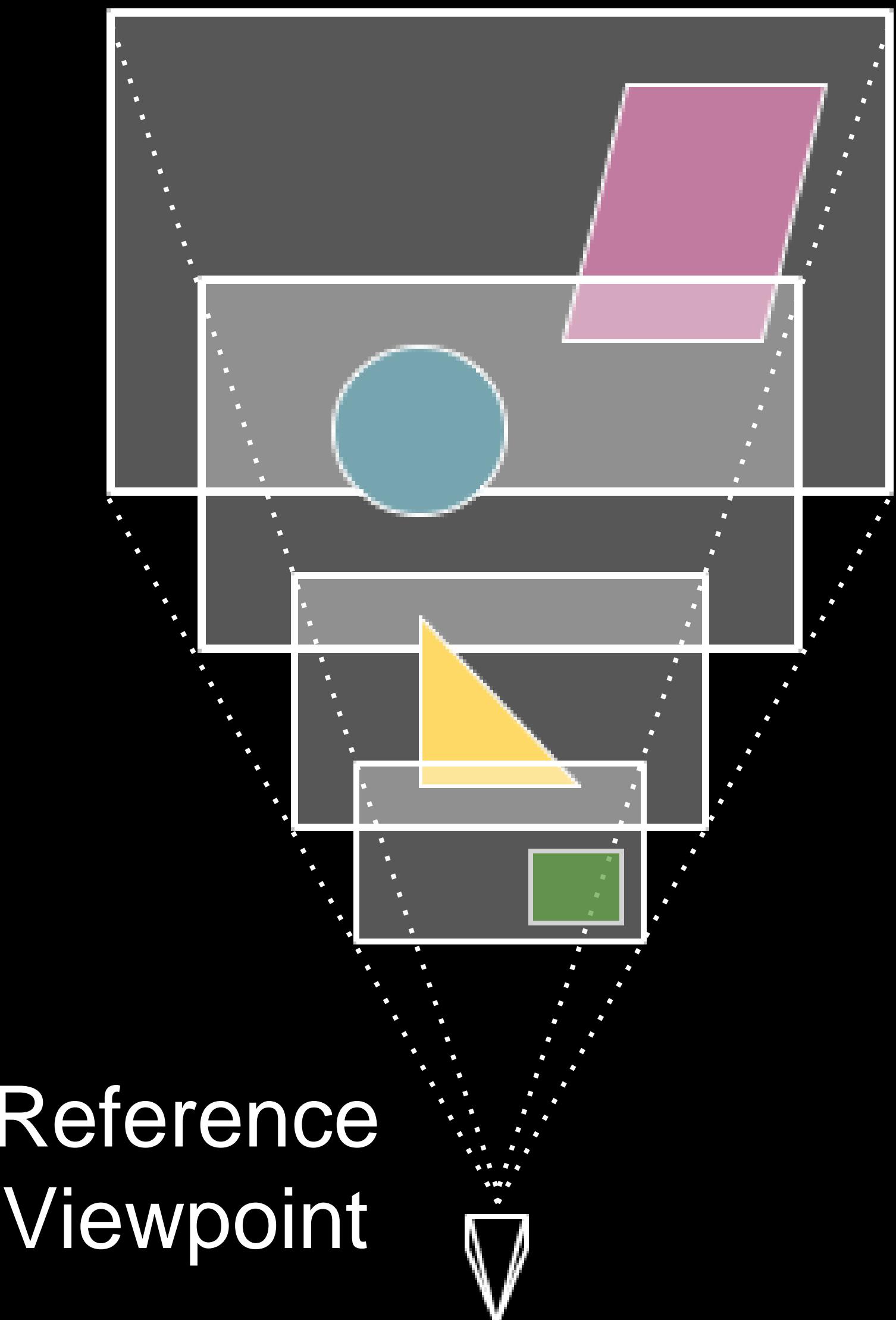


Image credits: Disney

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

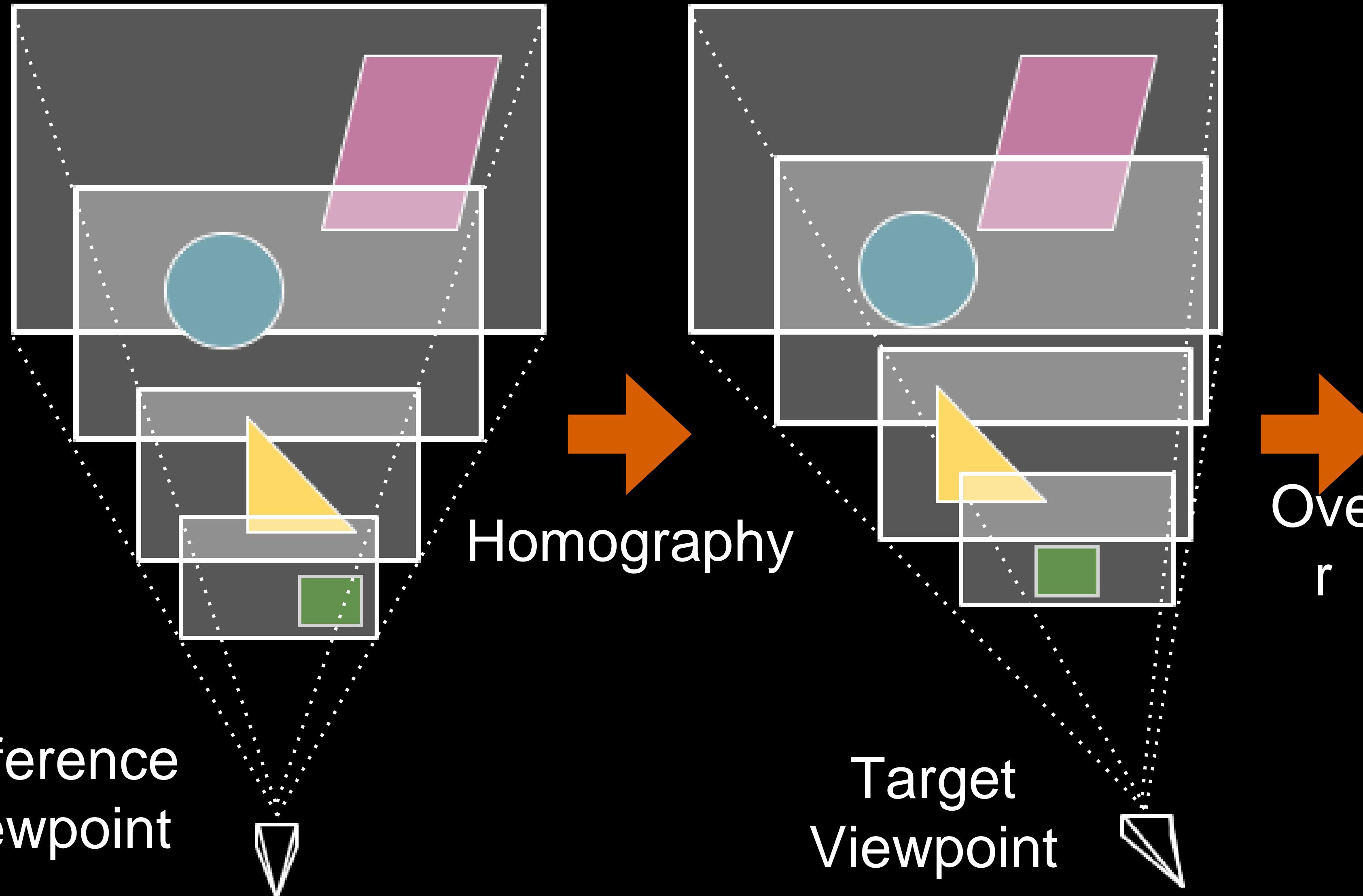
Multiplane Images (MPIs)



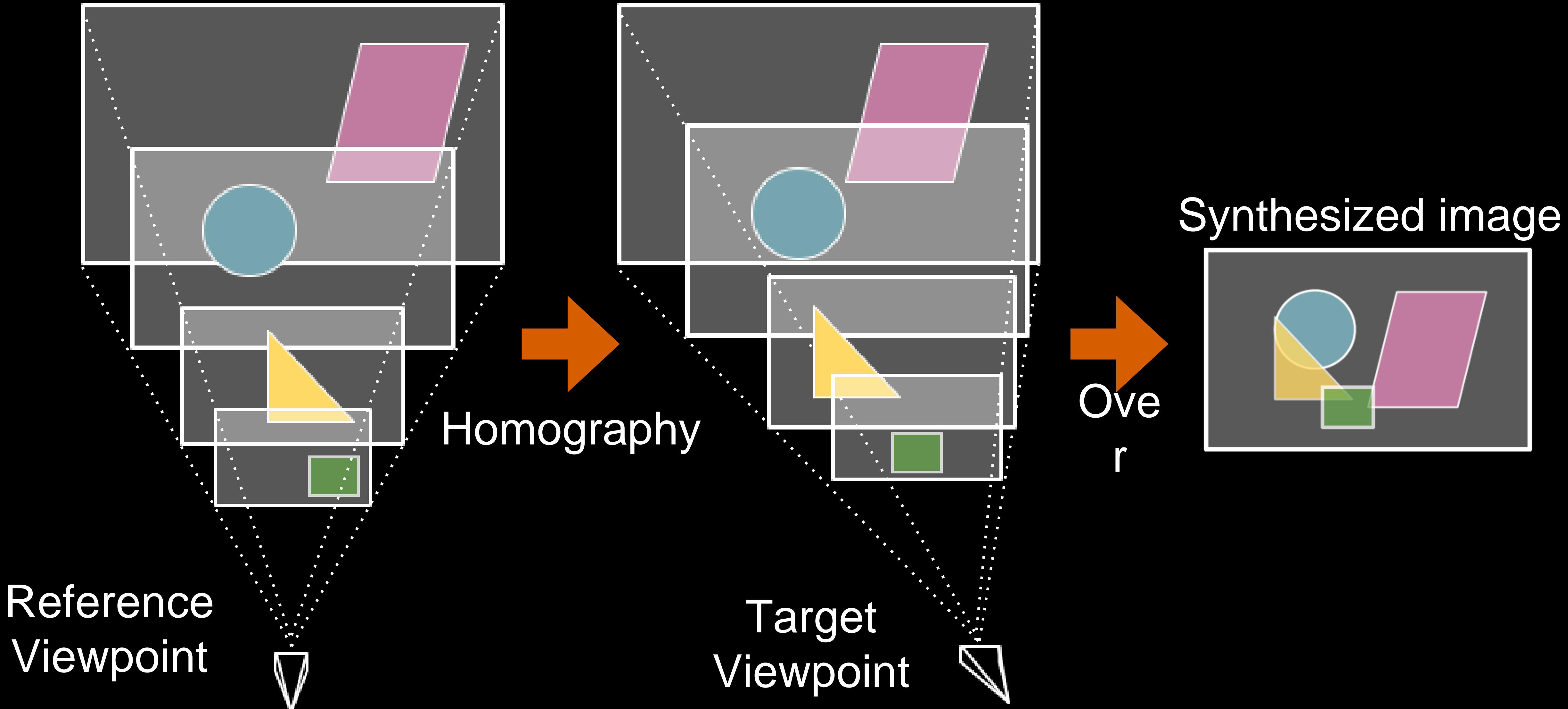
Each plane is at a fixed
depth and encoded by
an RGBA image

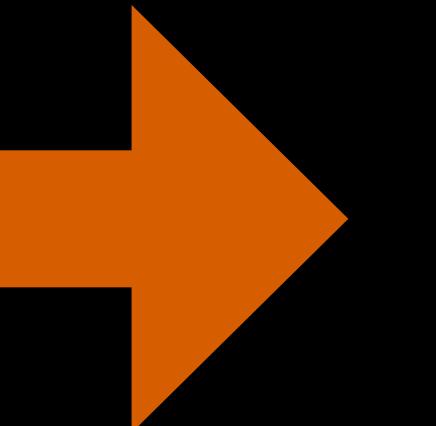
Reference
Viewpoint

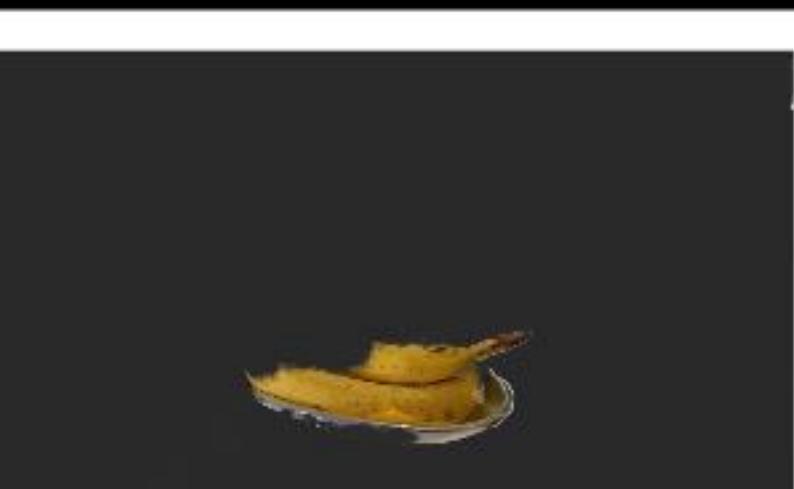
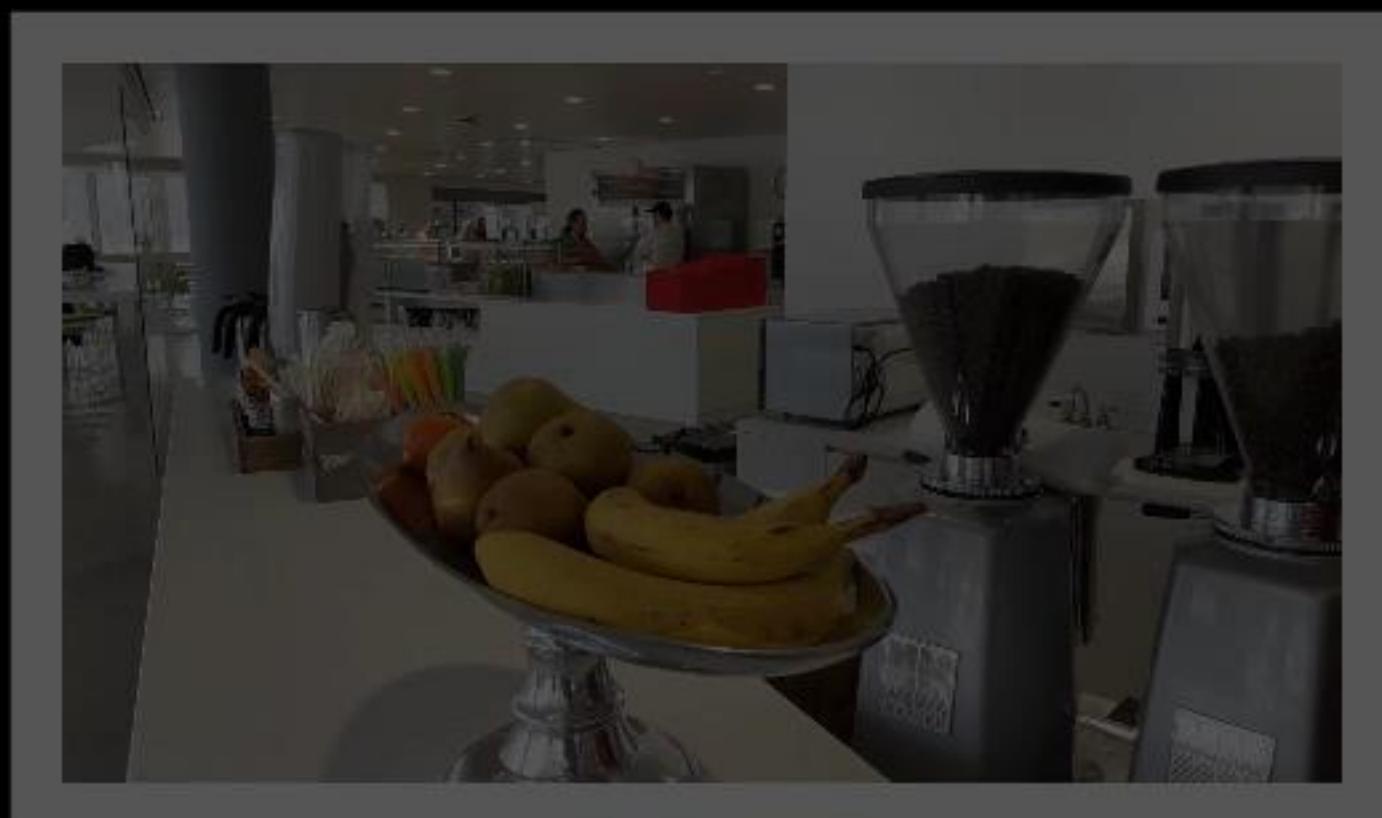
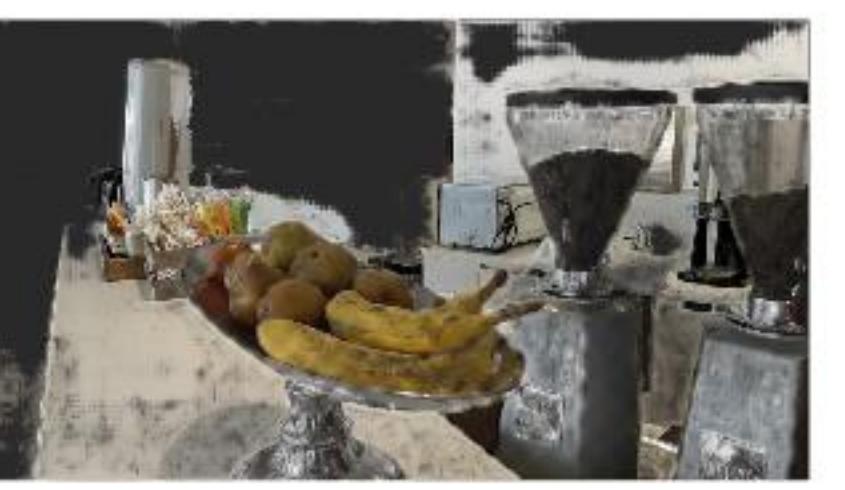
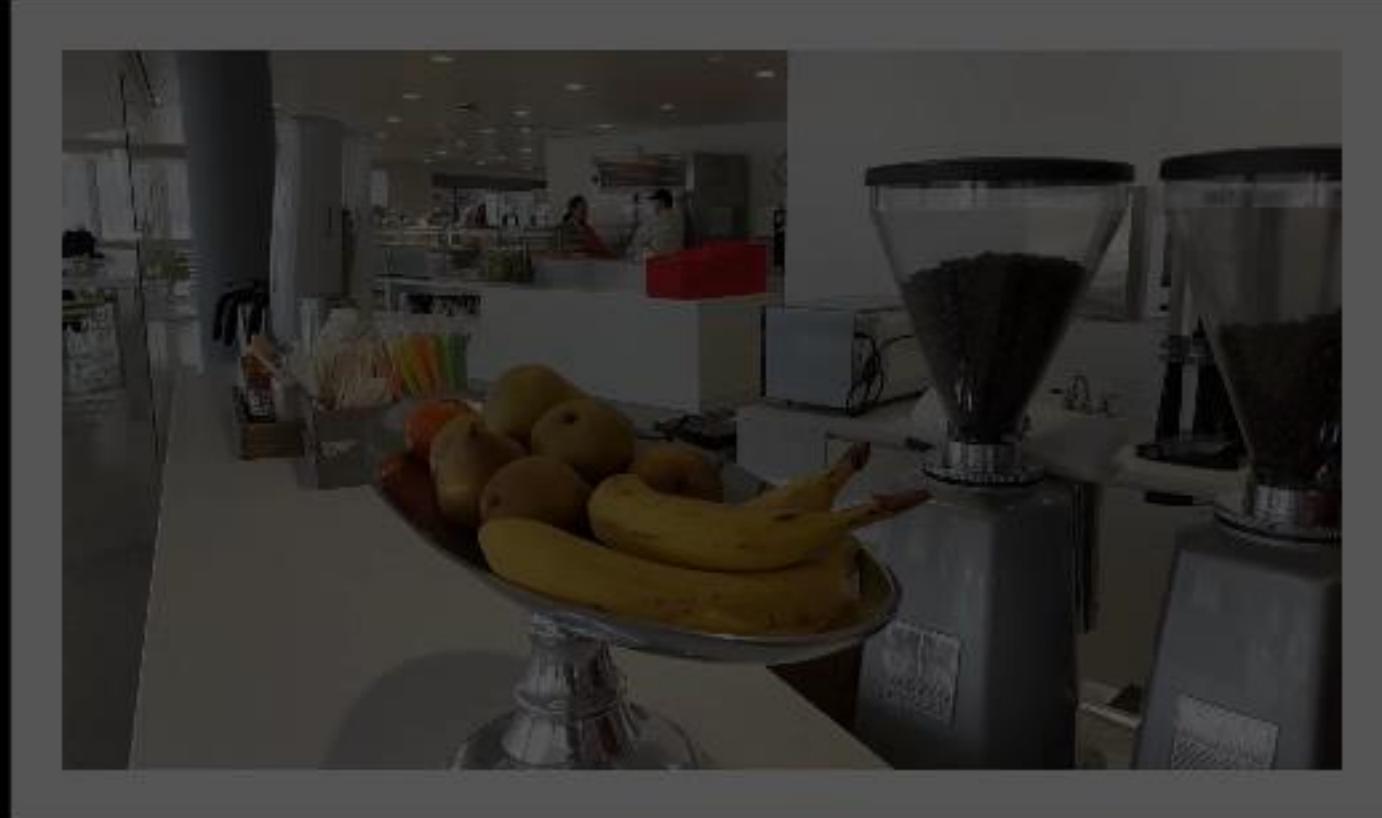
View Synthesis using Multiplane Images



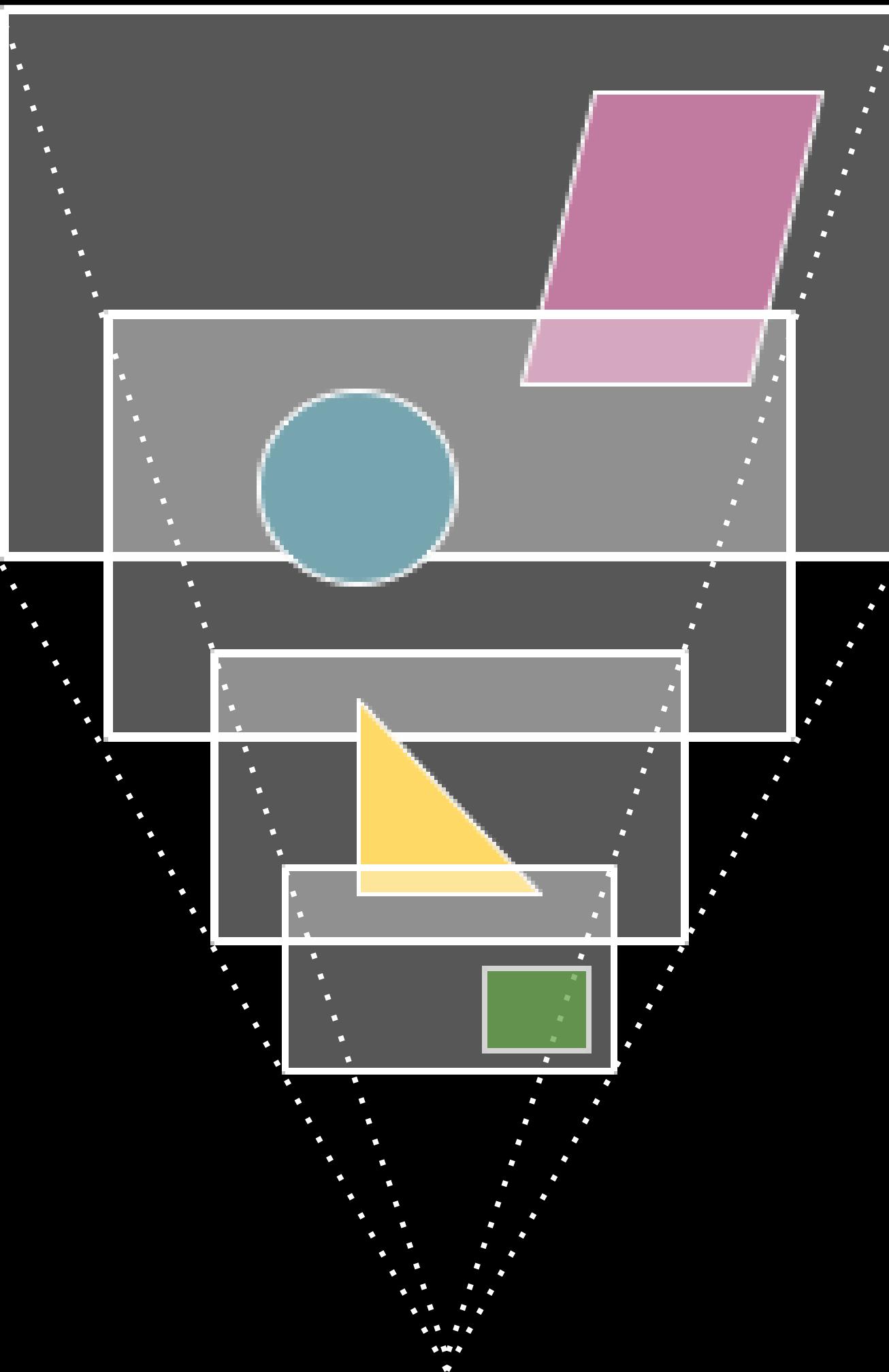
View Synthesis using Multiplane Images





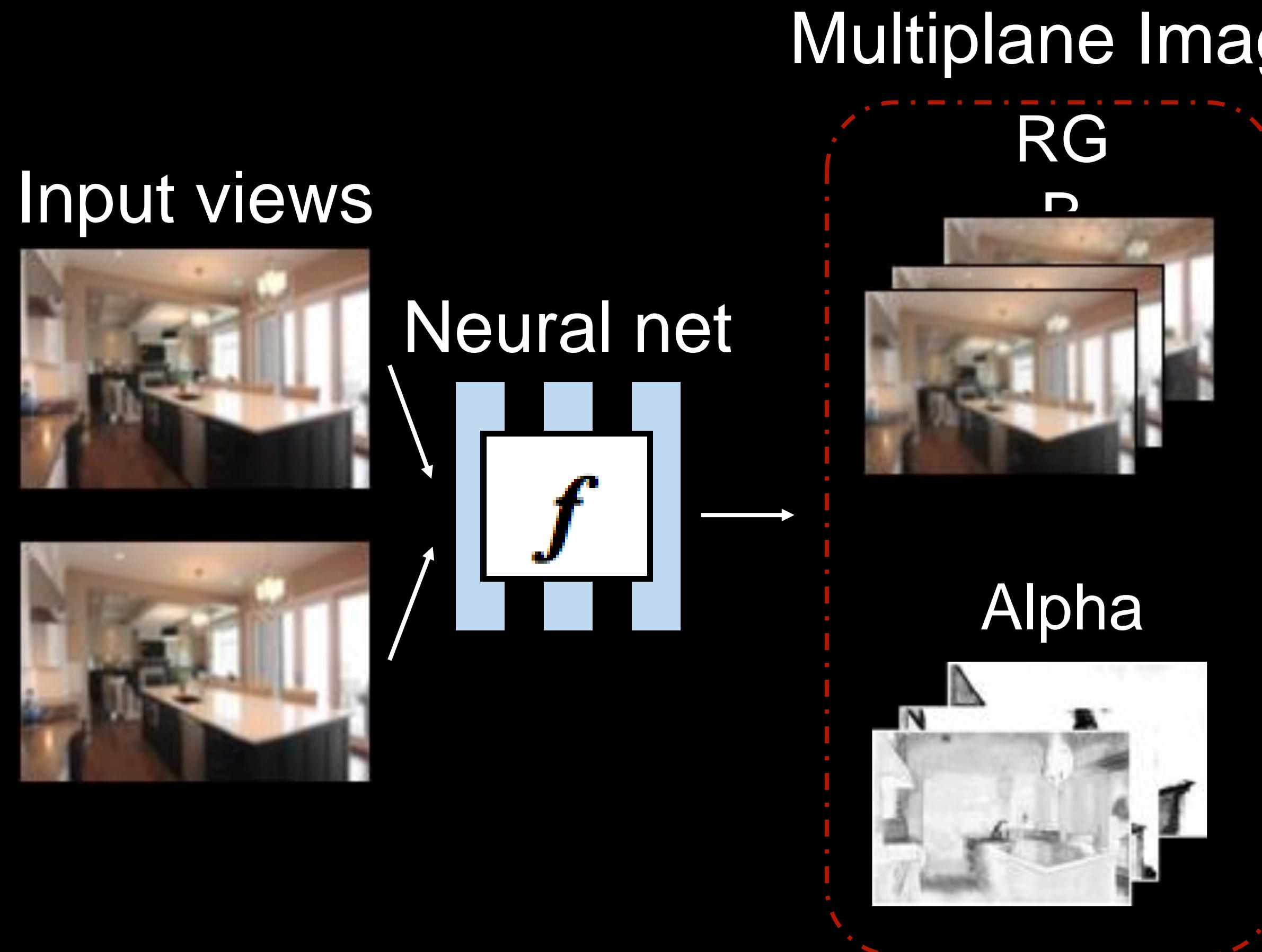


Properties of Multiplane Images

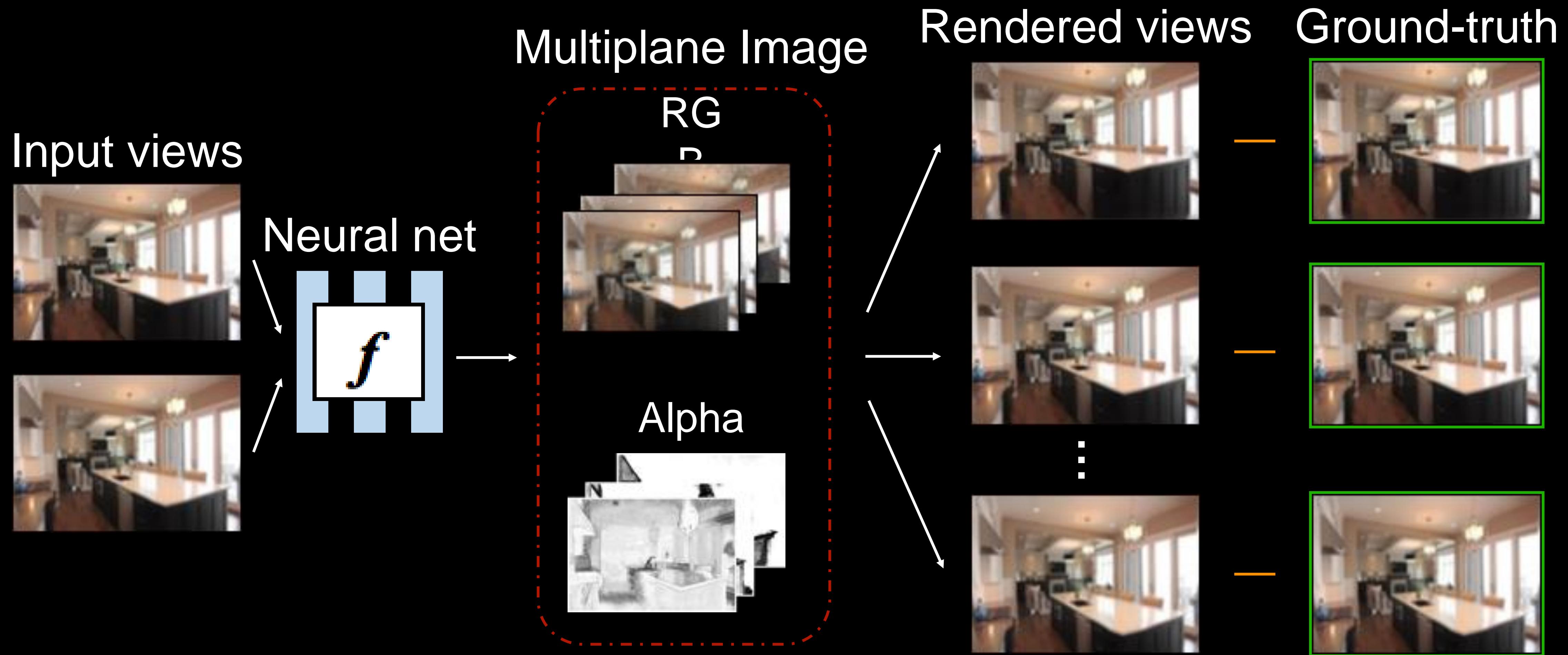


- Models disocclusion
- Models soft edges and non-Lambertian effects
- Efficient for view synthesis
- Differentiable rendering

Learning Multiplane Images



Learning Multiplane Images



Mapping image-shaped inputs to image-shaped outputs with the UNet architecture

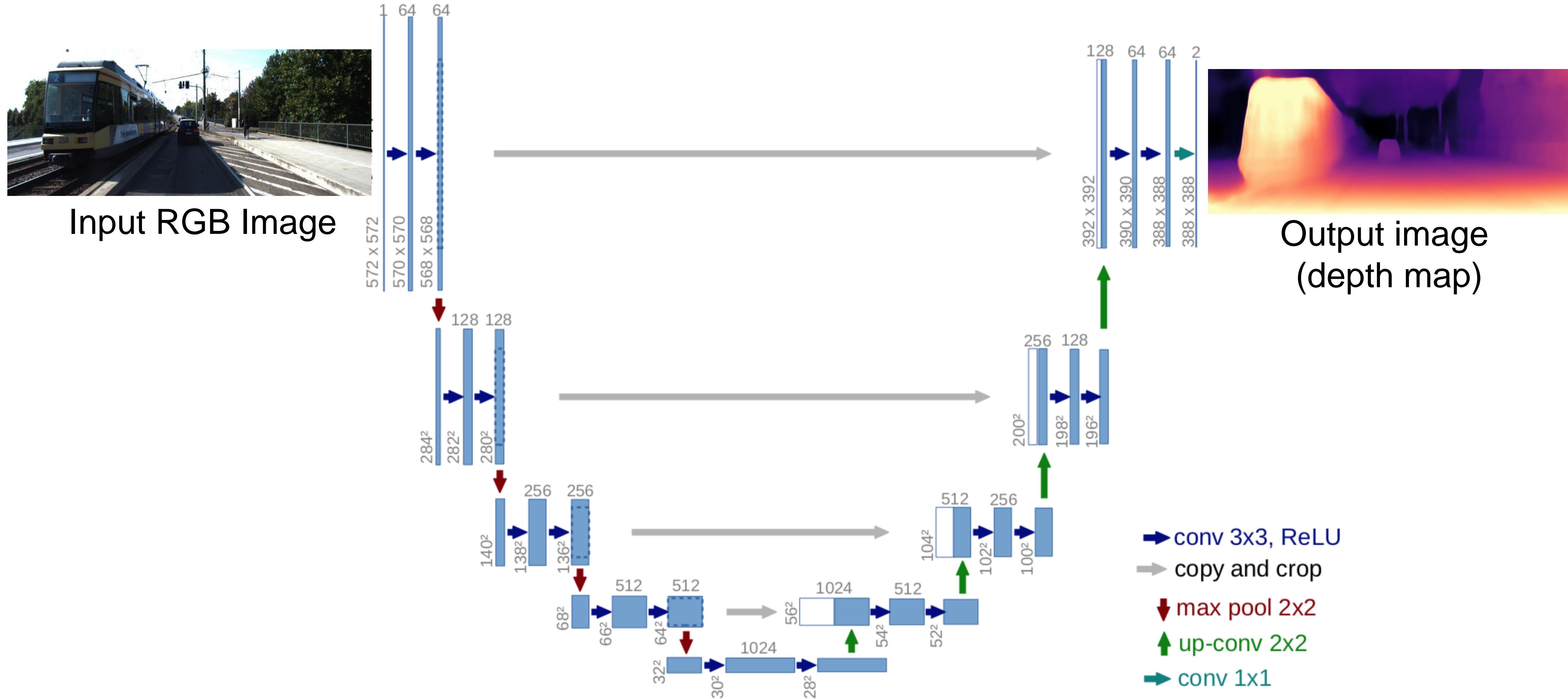
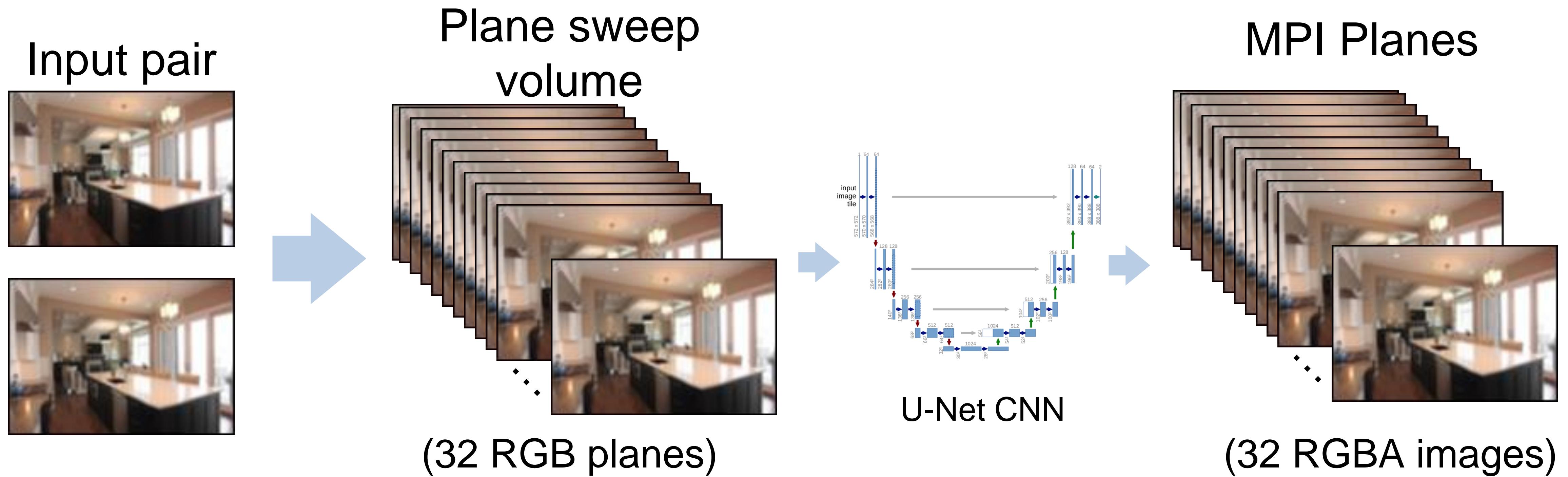


Image Pair → Multiplane Image

Suppose we want to map a pair of images to a 32-plane MPI



Training Data

Input views

(



,

(



,

(



,

:

Target view



)

)

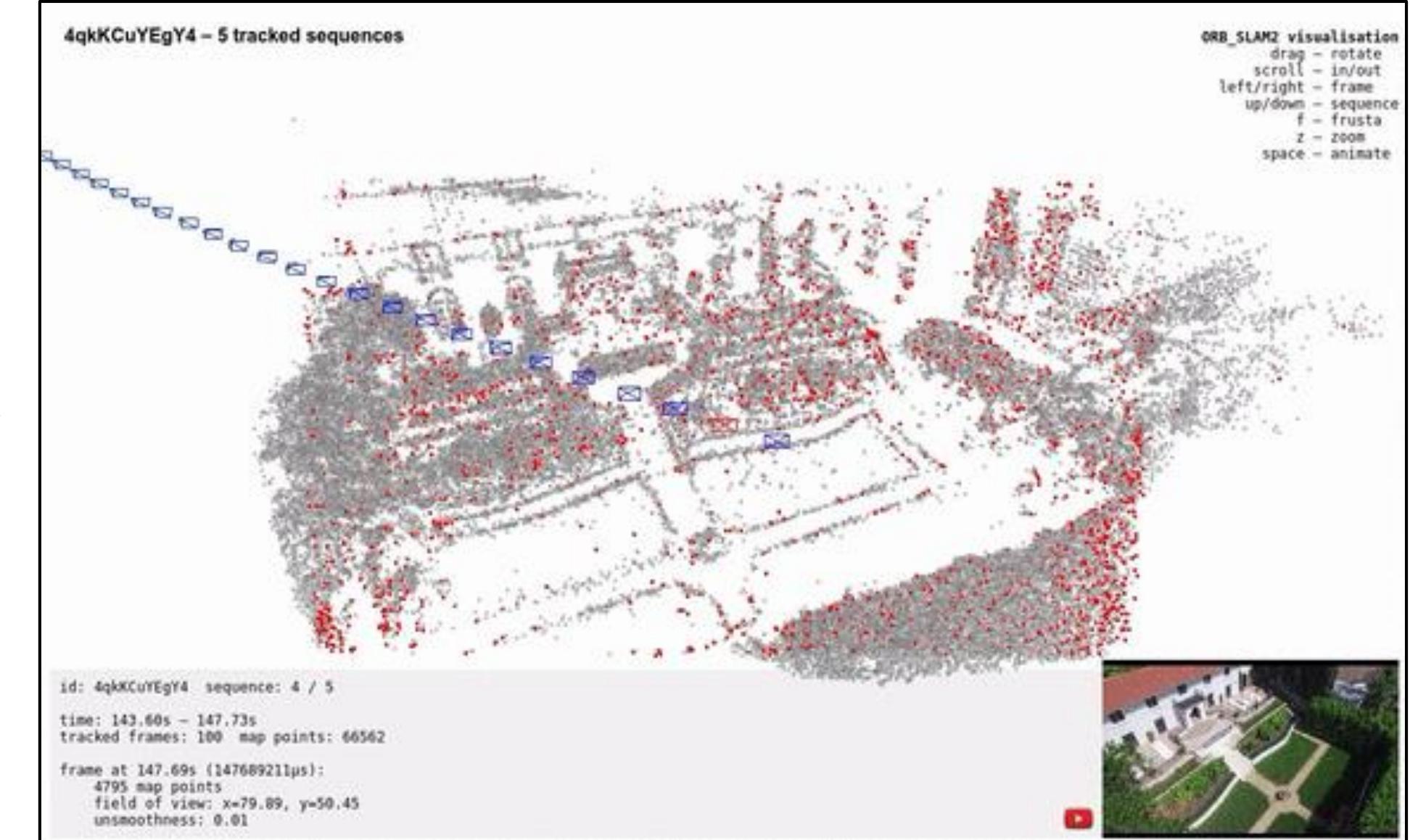
)

Need massive set of
triplets with known
camera poses

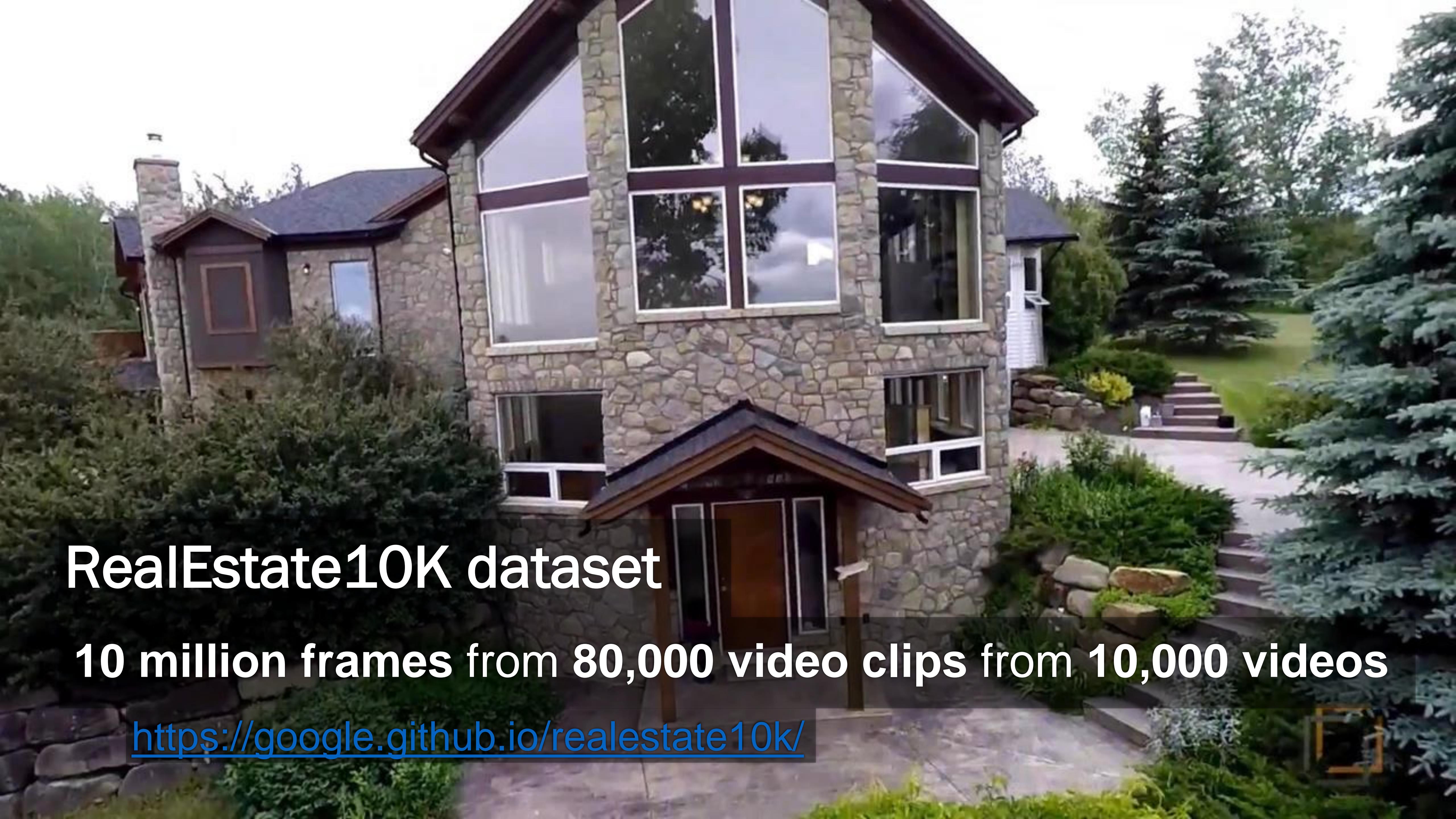
RealEstate10K



SLAM



Running SLAM / SfM on YouTube videos at scale

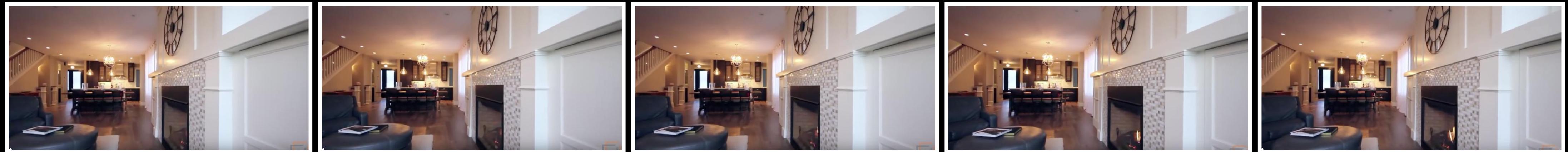


RealEstate10K dataset

10 million frames from 80,000 video clips from 10,000 videos

<https://google.github.io/realestate10k/>

Sampling Training Examples



Input

Input

Target
(Extrapolated)

Sampling Training Examples



Input

Target
(Interpolated)

Input

Results

Left



Right



Output



Image 1



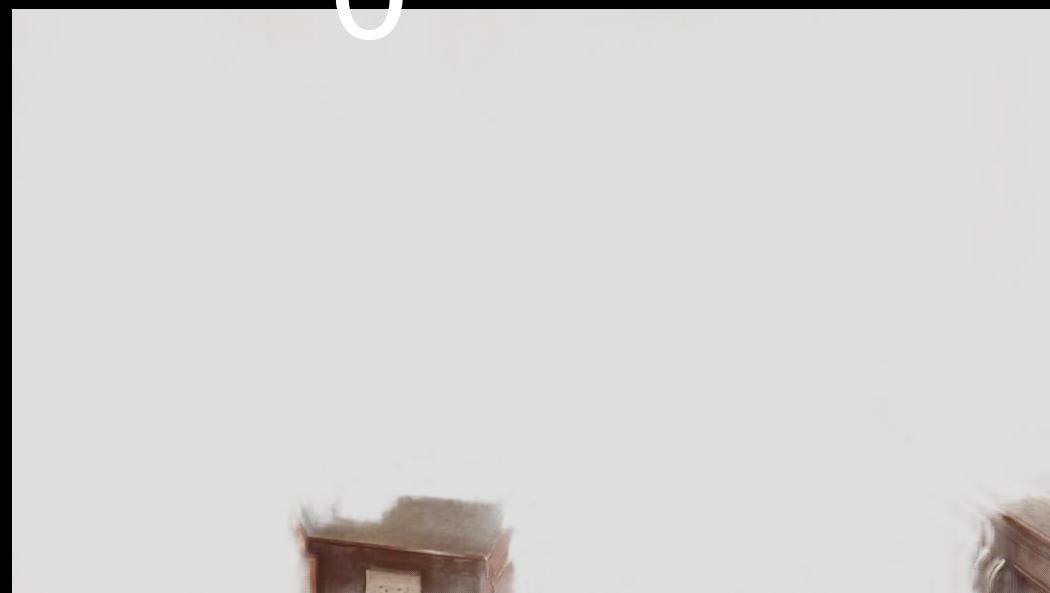
Image 2



Multi-plane Image (MPI)

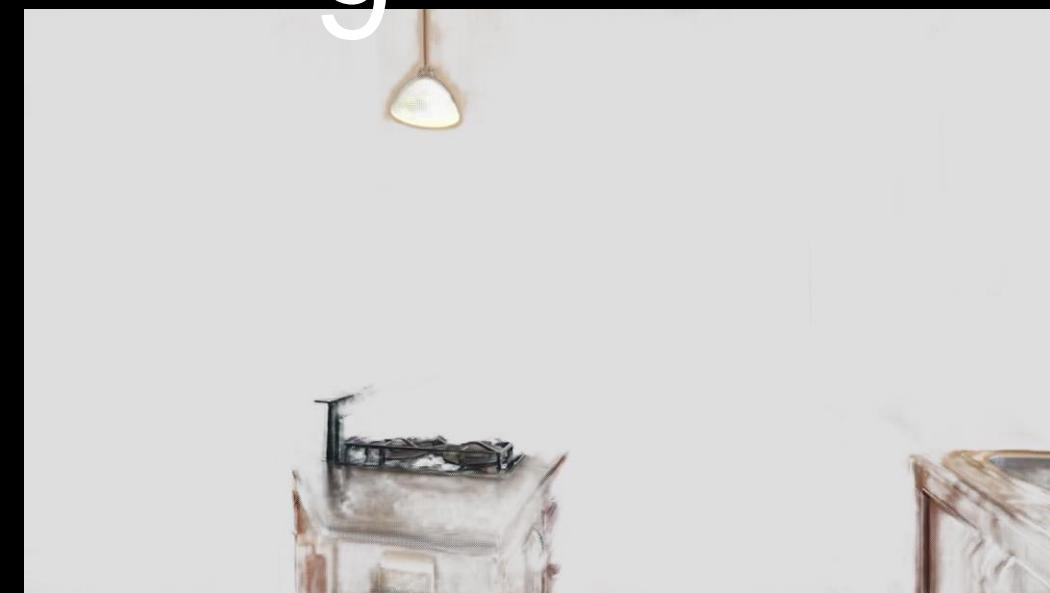
Plane

0

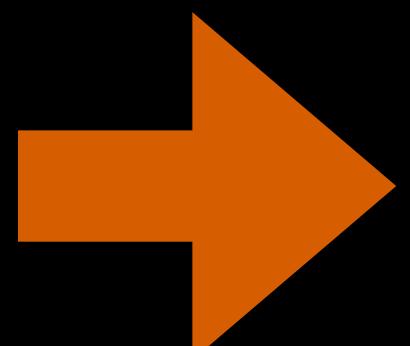


Plane

9



Reference input view



Plane 13



Plane 16

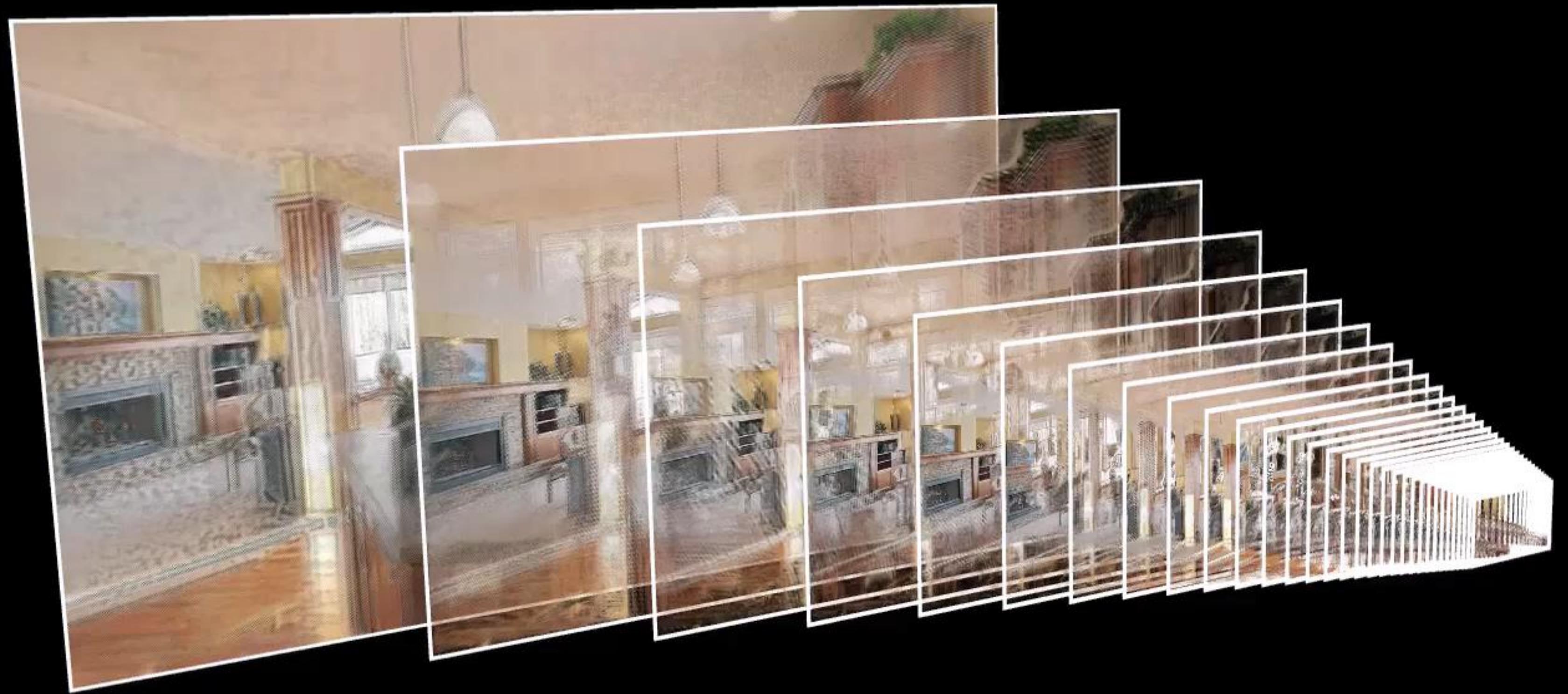


Plane 24

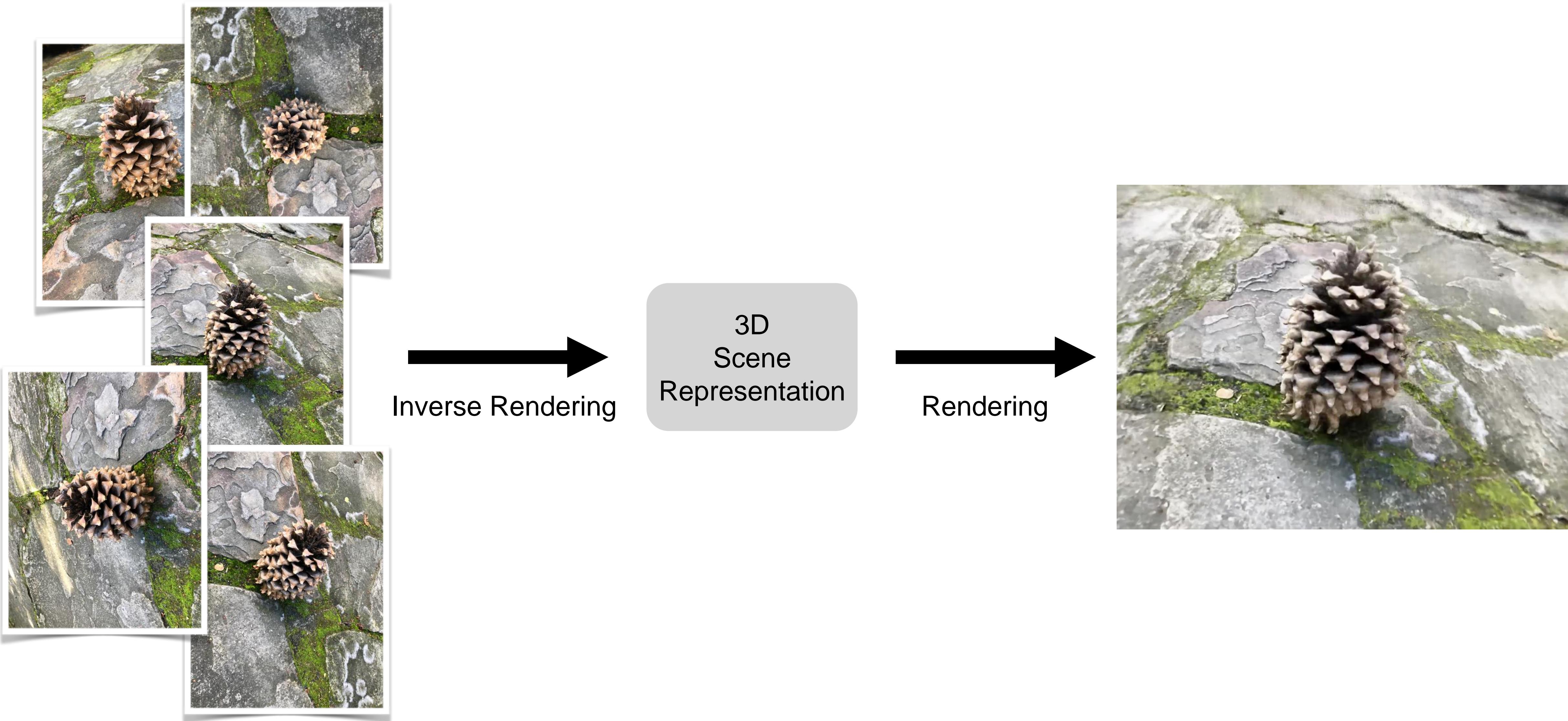


Plane 26



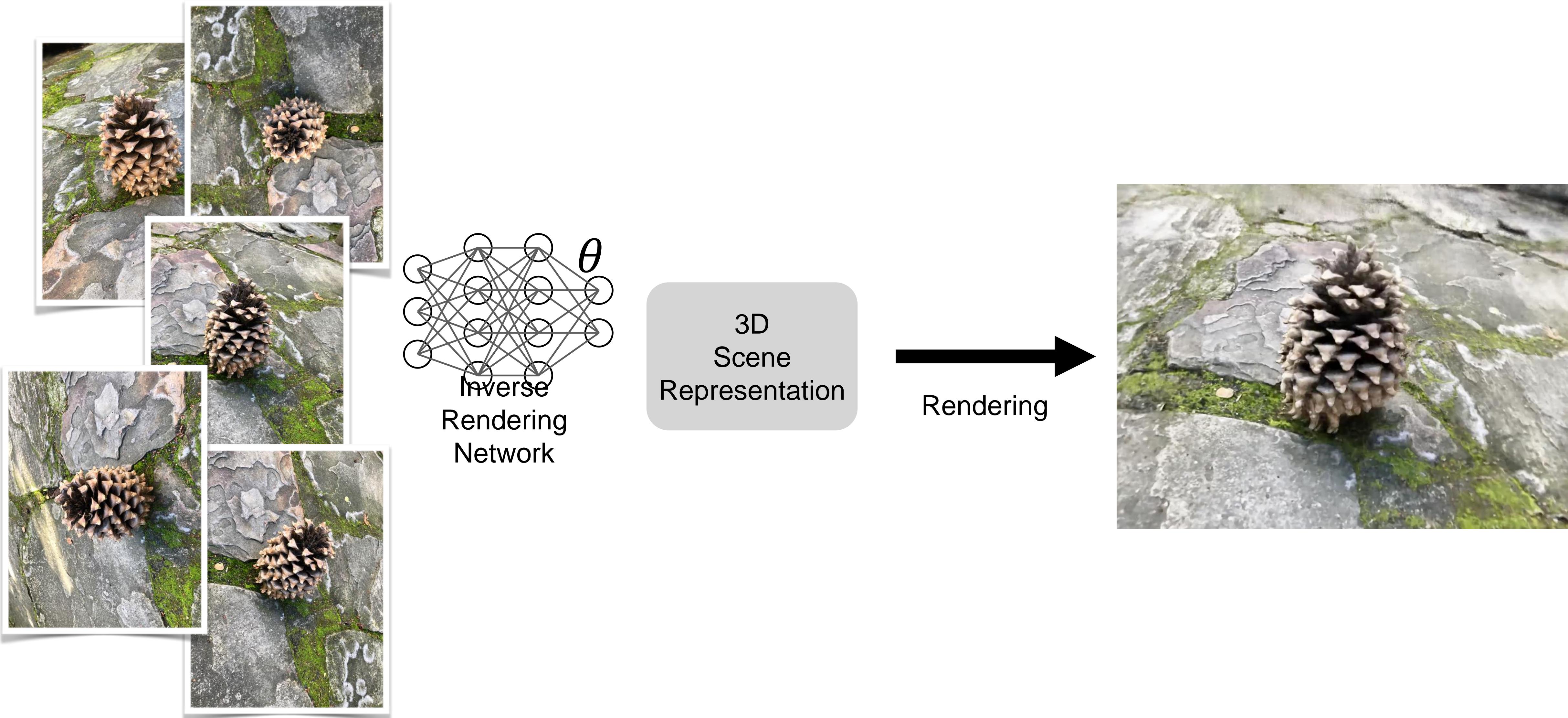


Computer vision as inverse rendering

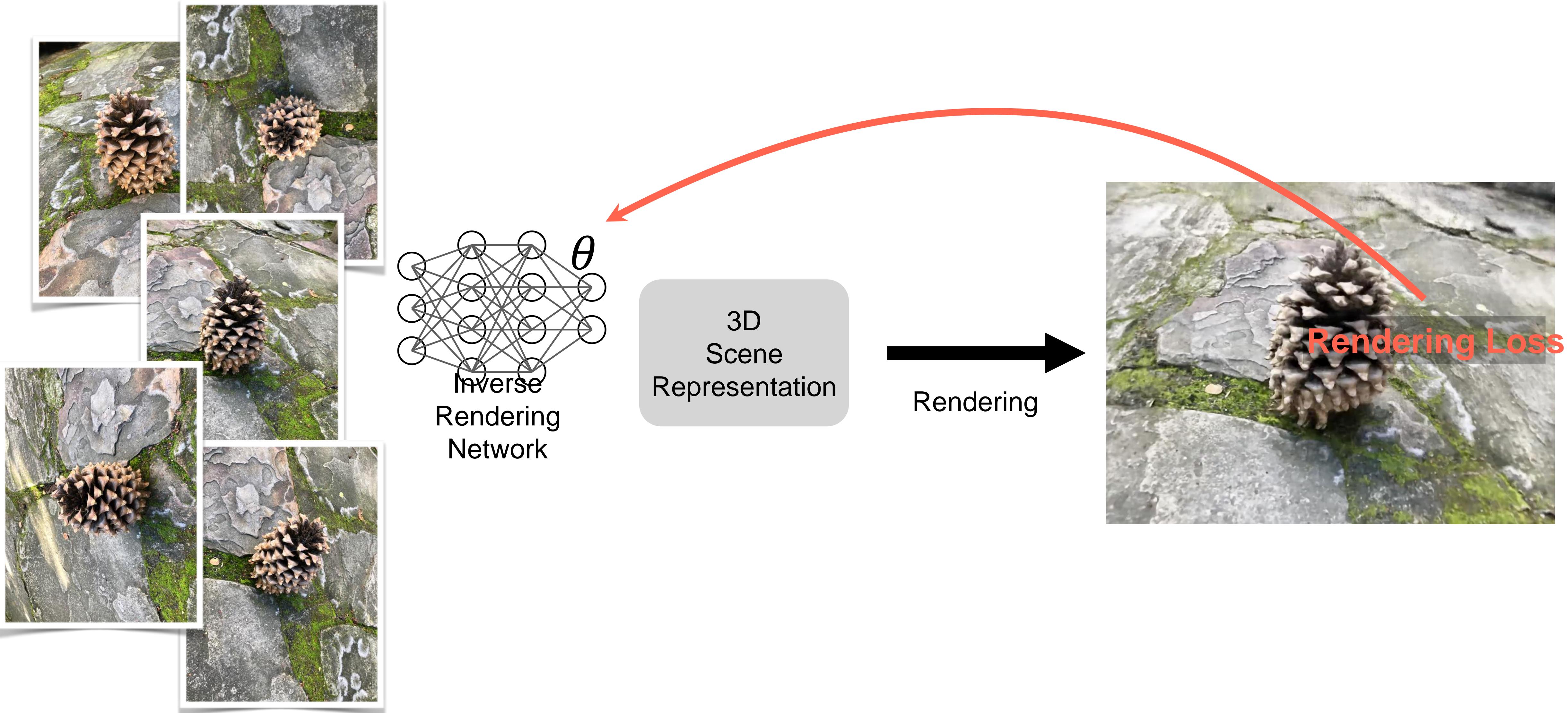


Adapted from material from Pratul Srinivas

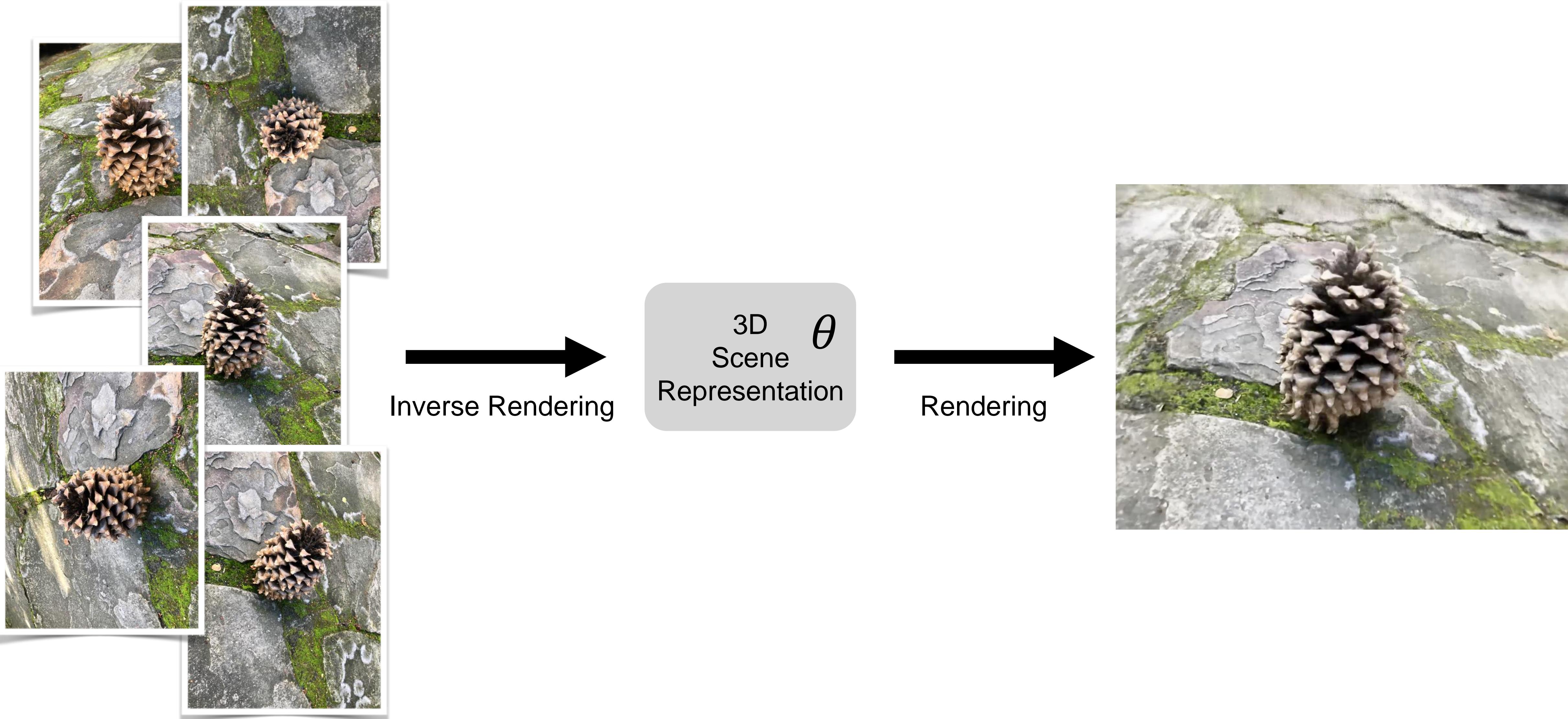
Paradigm 1: “Feedforward” inverse rendering



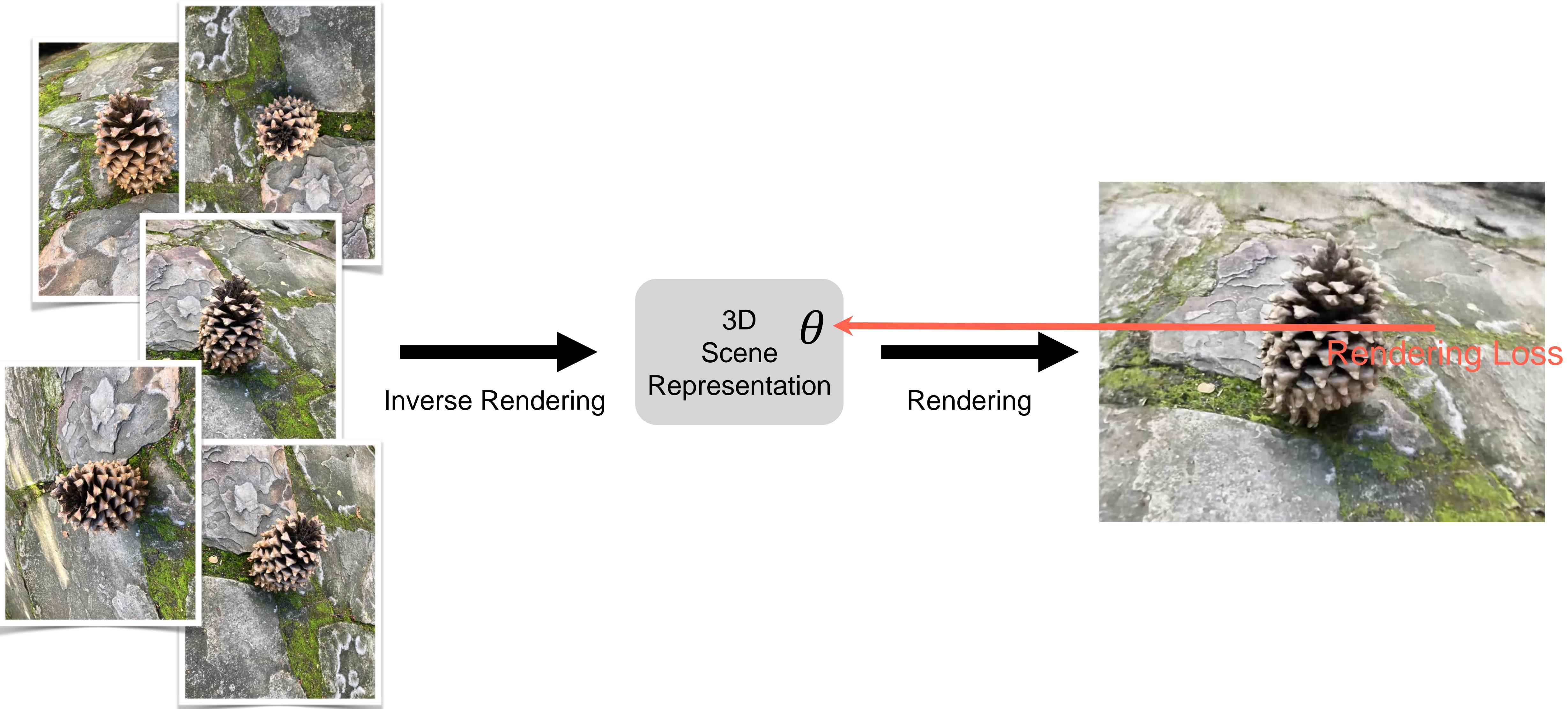
Paradigm 1: “Feedforward” inverse rendering



Paradigm 2: “Render-and-compare”

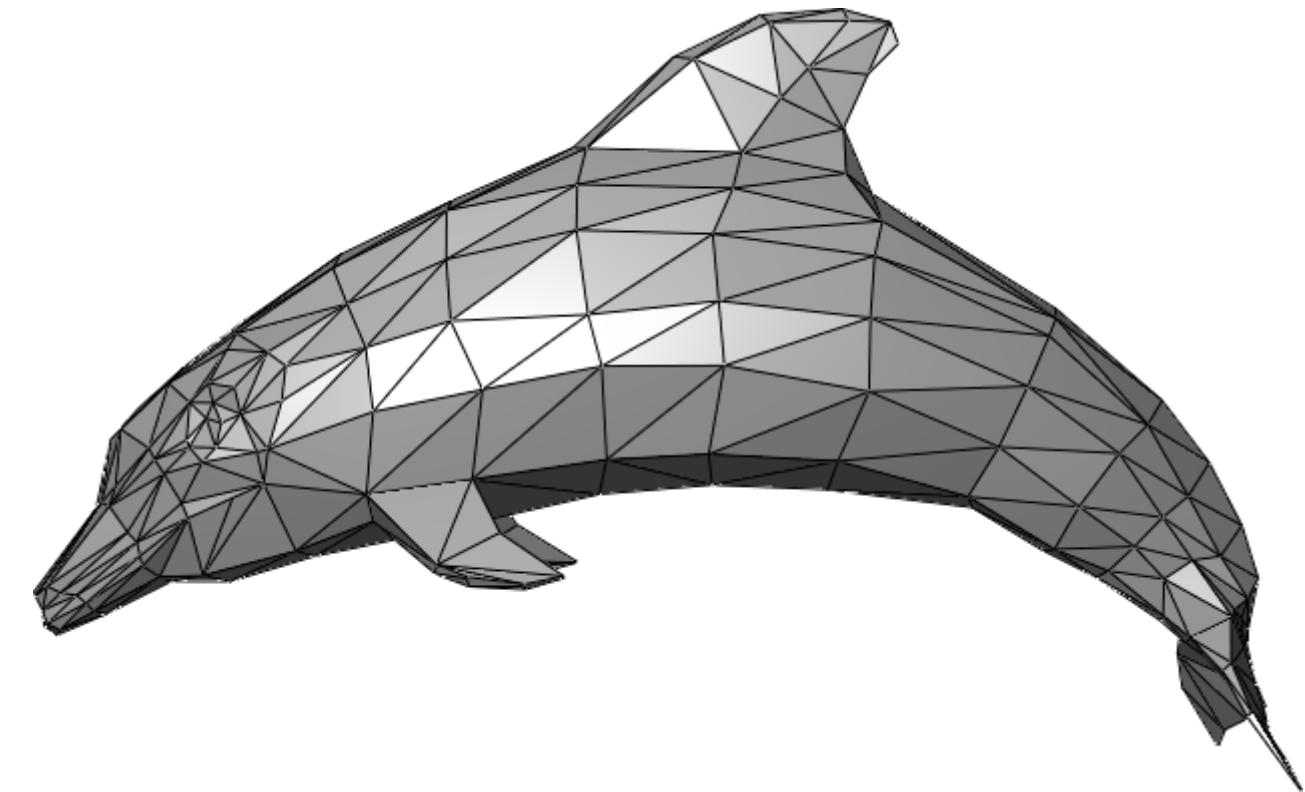


Paradigm 2: “Render-and-compare”



What representation to use?

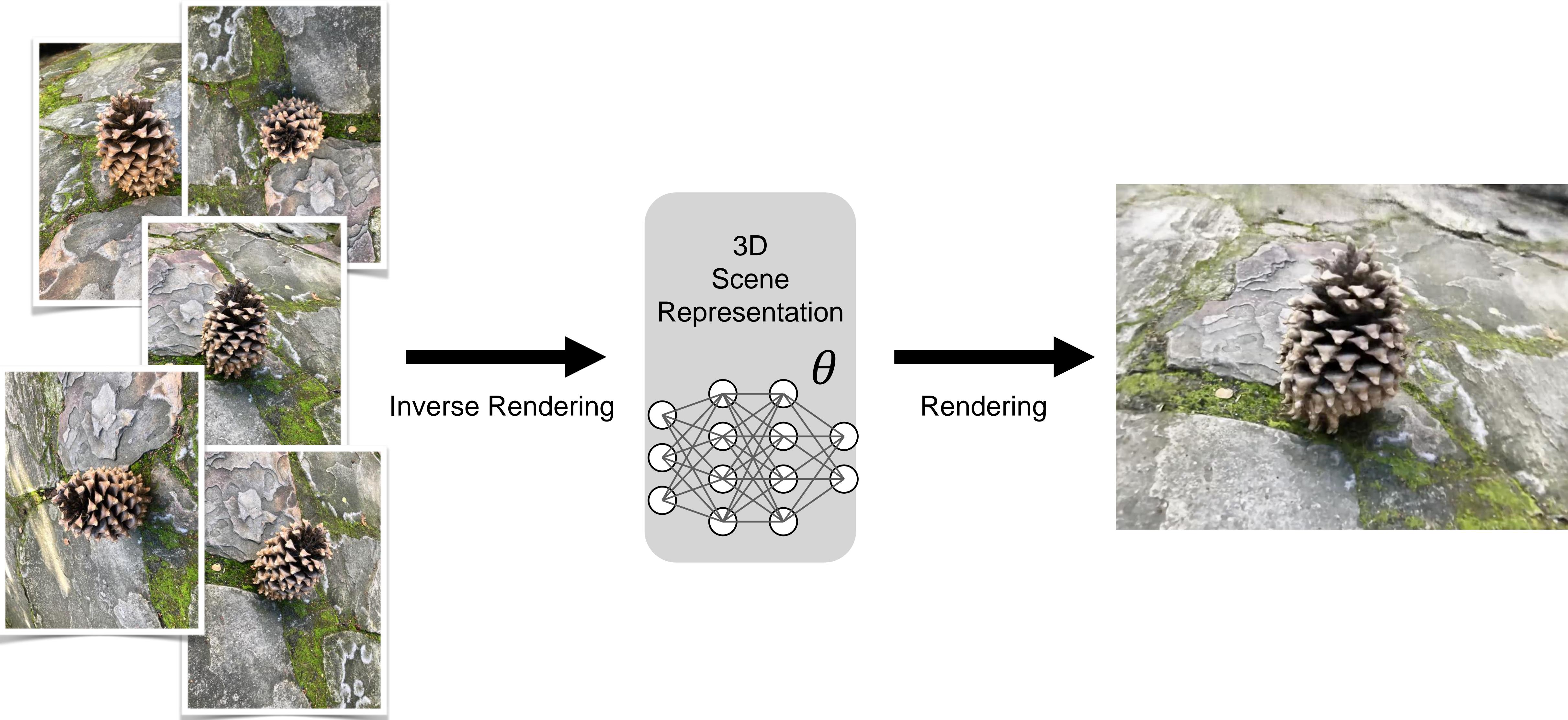
- Could use triangle meshes, but hard to differentiate during rendering
- Multiplane images (MPIs) are easy to differentiate, but only allow for rendering a small range of views





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

Paradigm 2: “Render-and-compare”

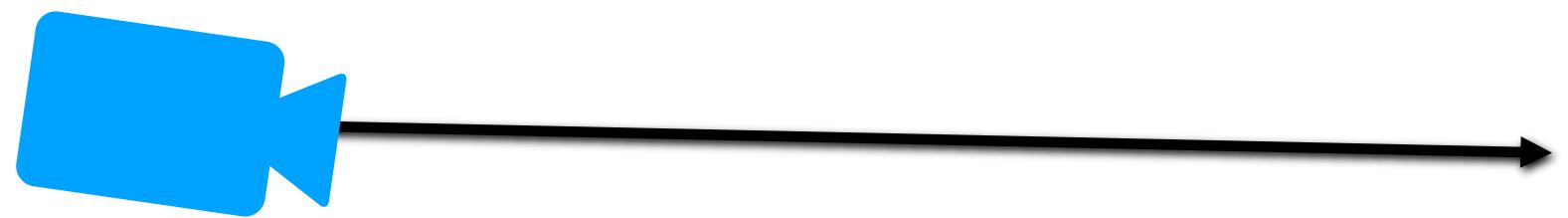




Neural Volumetric Rendering

Neural Volumetric **Rendering**

querying the radiance value
along rays through 3D space



What color?

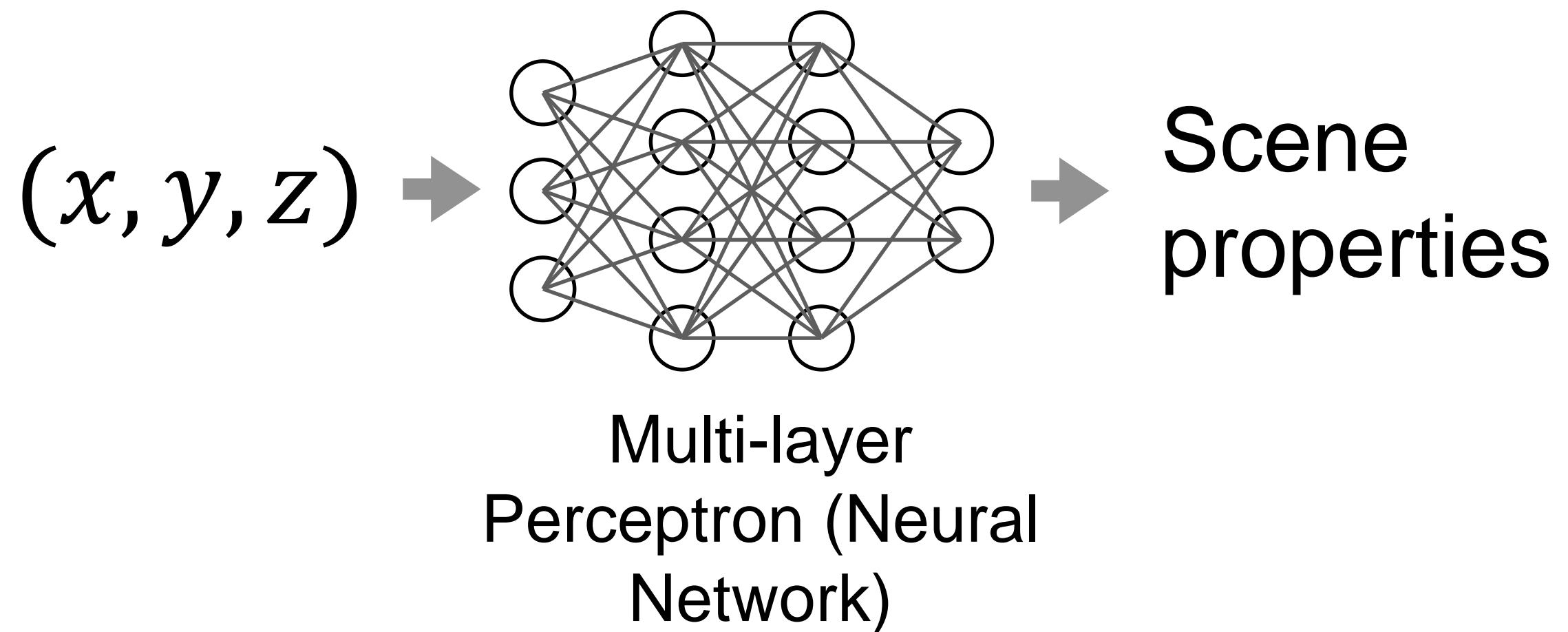
Neural Volumetric Rendering

continuous, differentiable
rendering model without concrete
ray/surface intersections



Neural Volumetric Rendering

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



NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

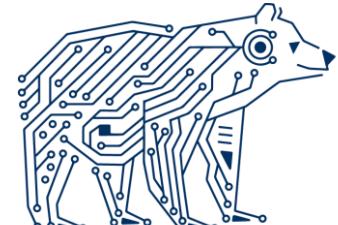
ECCV 2020



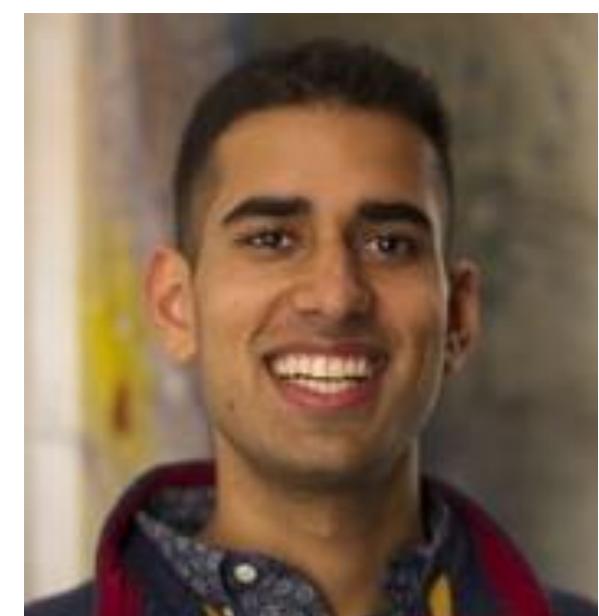
Ben Mildenhall*



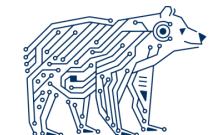
UC Berkeley



Pratul Srinivasan*



UC Berkeley

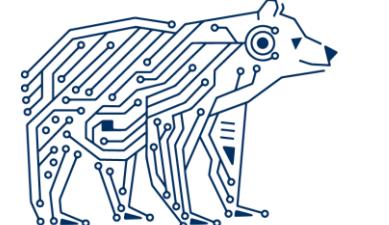


Google

Matt Tancik*



UC Berkeley



Jon Barron



Google Research

Google

Ravi Ramamoorthi



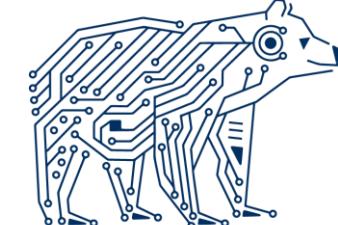
UC San Diego

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

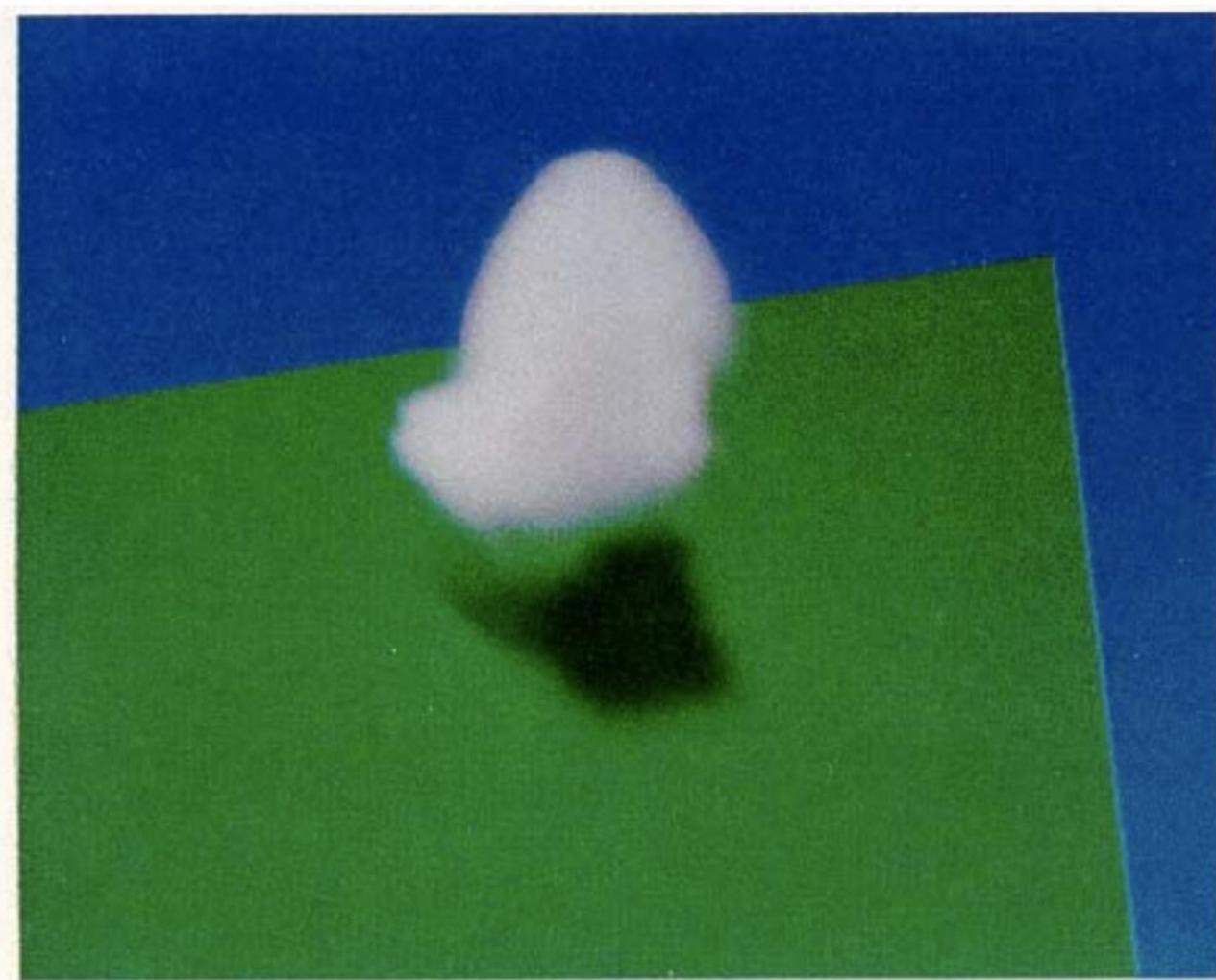
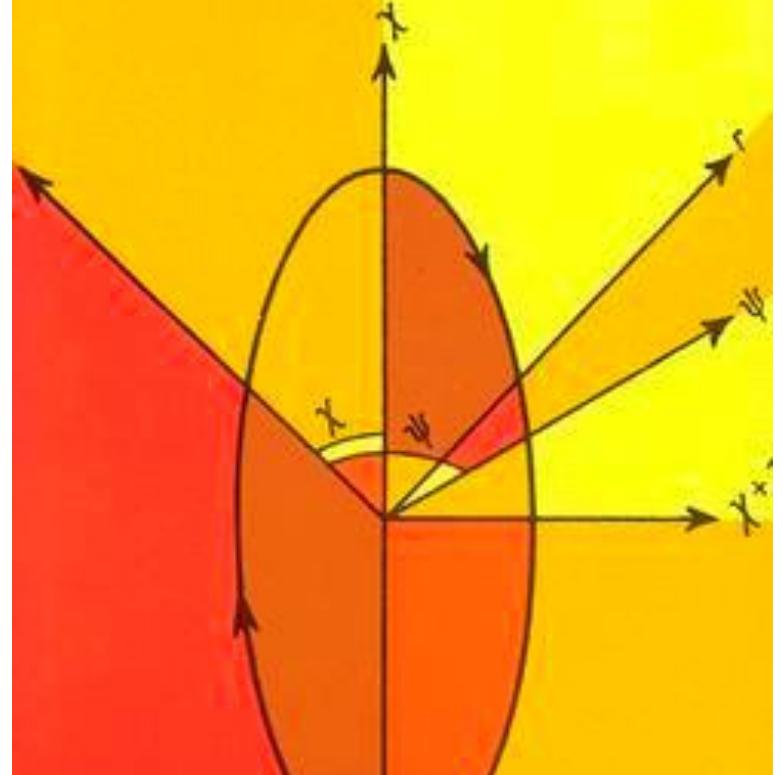
- ▶ Volumetric rendering
- ▶ Neural networks as representations for spatial data
- ▶ Neural Radiance Fields (NeRF)

NeRF Overview

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

S.Chandrasekhar

RADIATIVE
TRANSFER



Ray tracing simulated cumulus cloud [Kajiya]

Traditional volumetric rendering

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

Chandrasekhar 1950, Radiative Transfer

Kajiya 1984, Ray Tracing Volume Densities

Full volumetric rendering formulation

Absorption



<http://commons.wikimedia.org>

Scattering



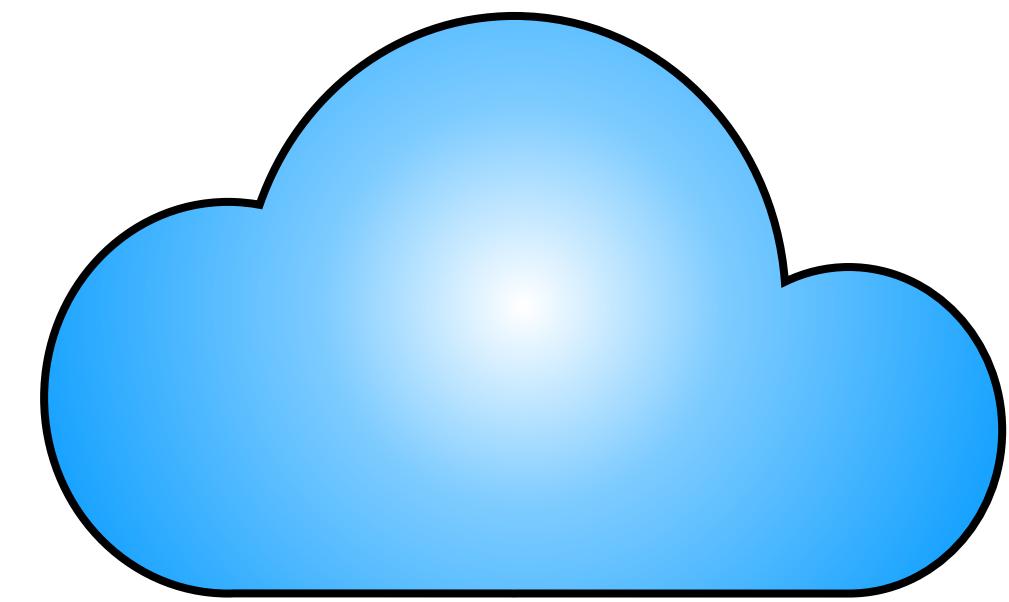
<http://coclouds.com>

Emission



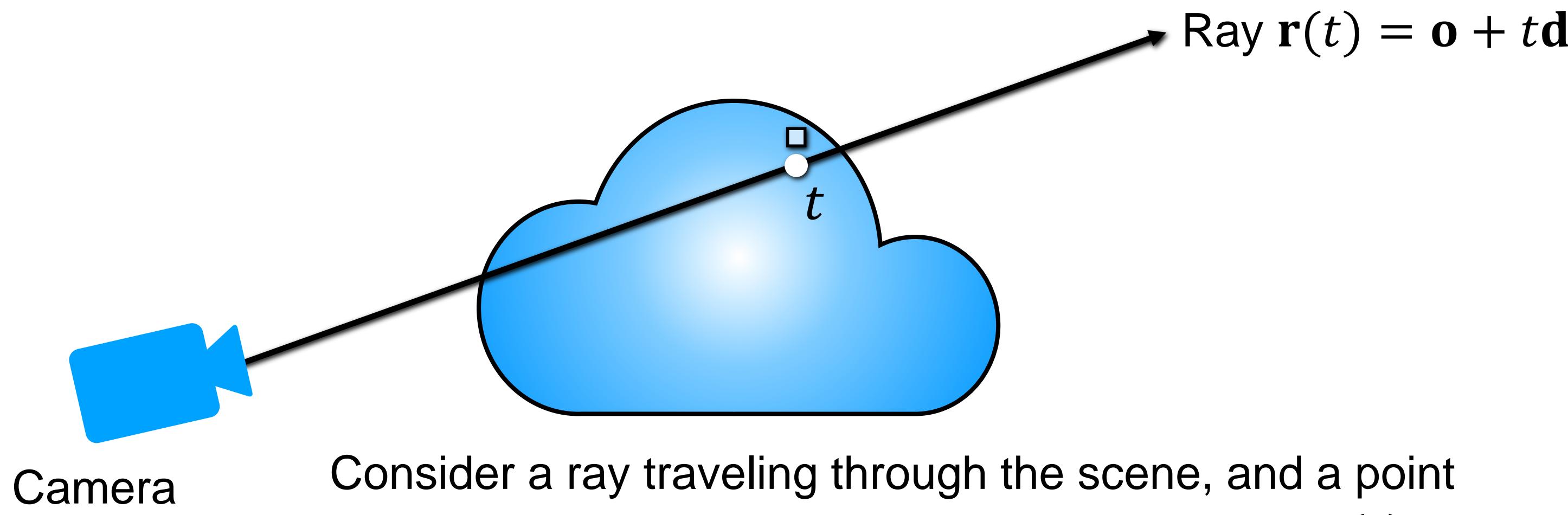
<http://wikipedia.org>

Volumetric formulation for NeRF

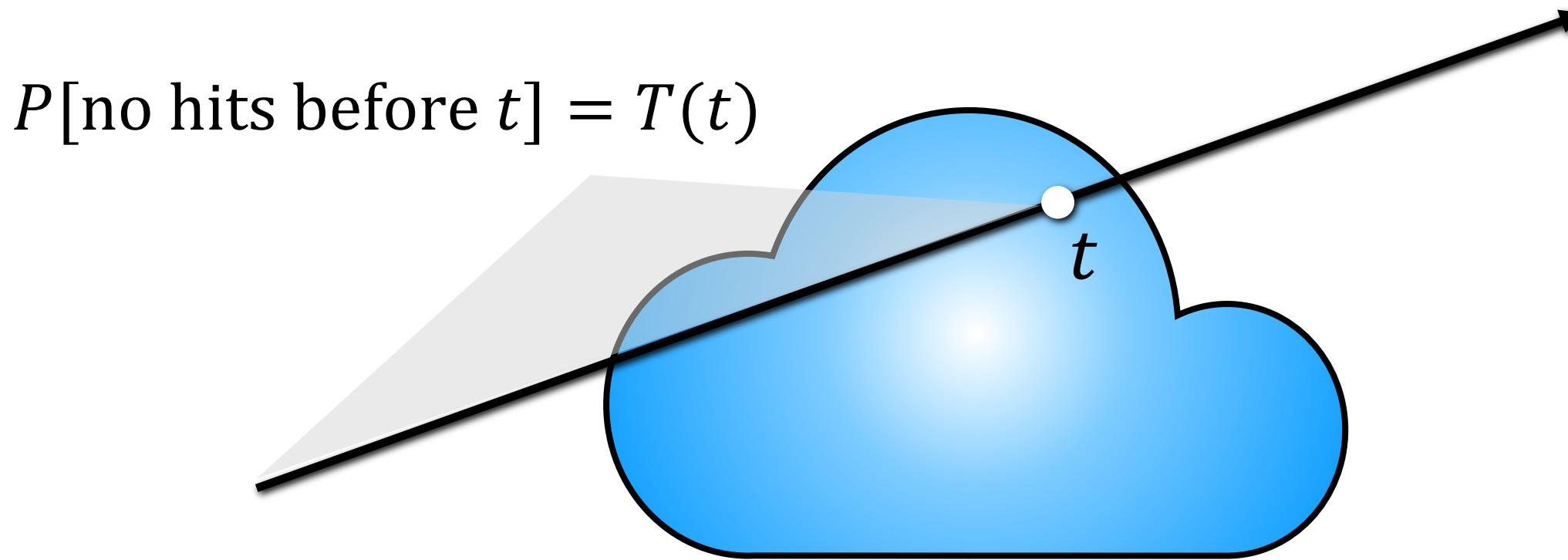


Scene is a cloud of colored fog

Volumetric formulation for NeRF



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

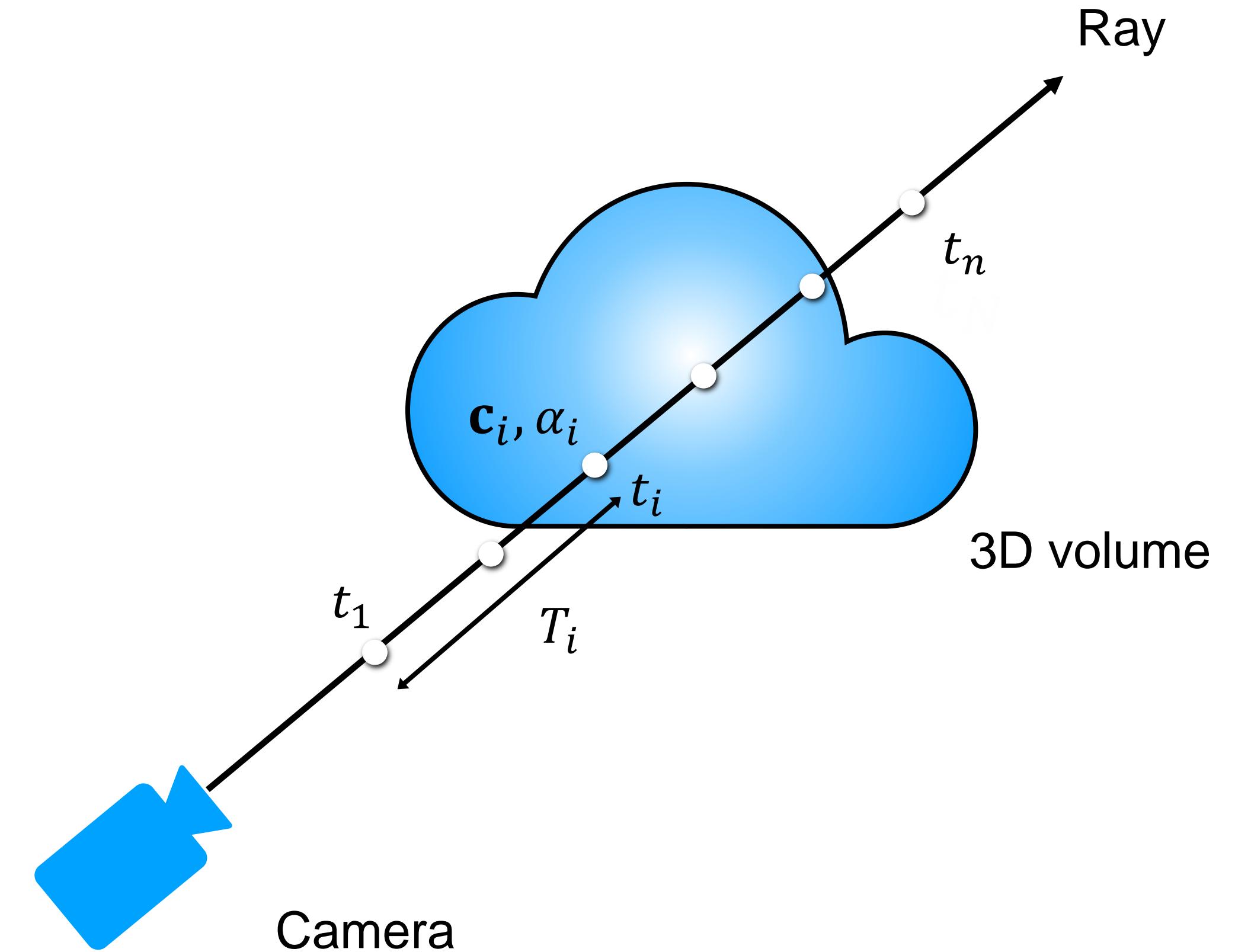
$$\mathbf{c} \approx \sum_{i=1}^n T_i \alpha_i \mathbf{c}_i$$

final rendered color along ray colors
weights

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

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

final rendered color along ray

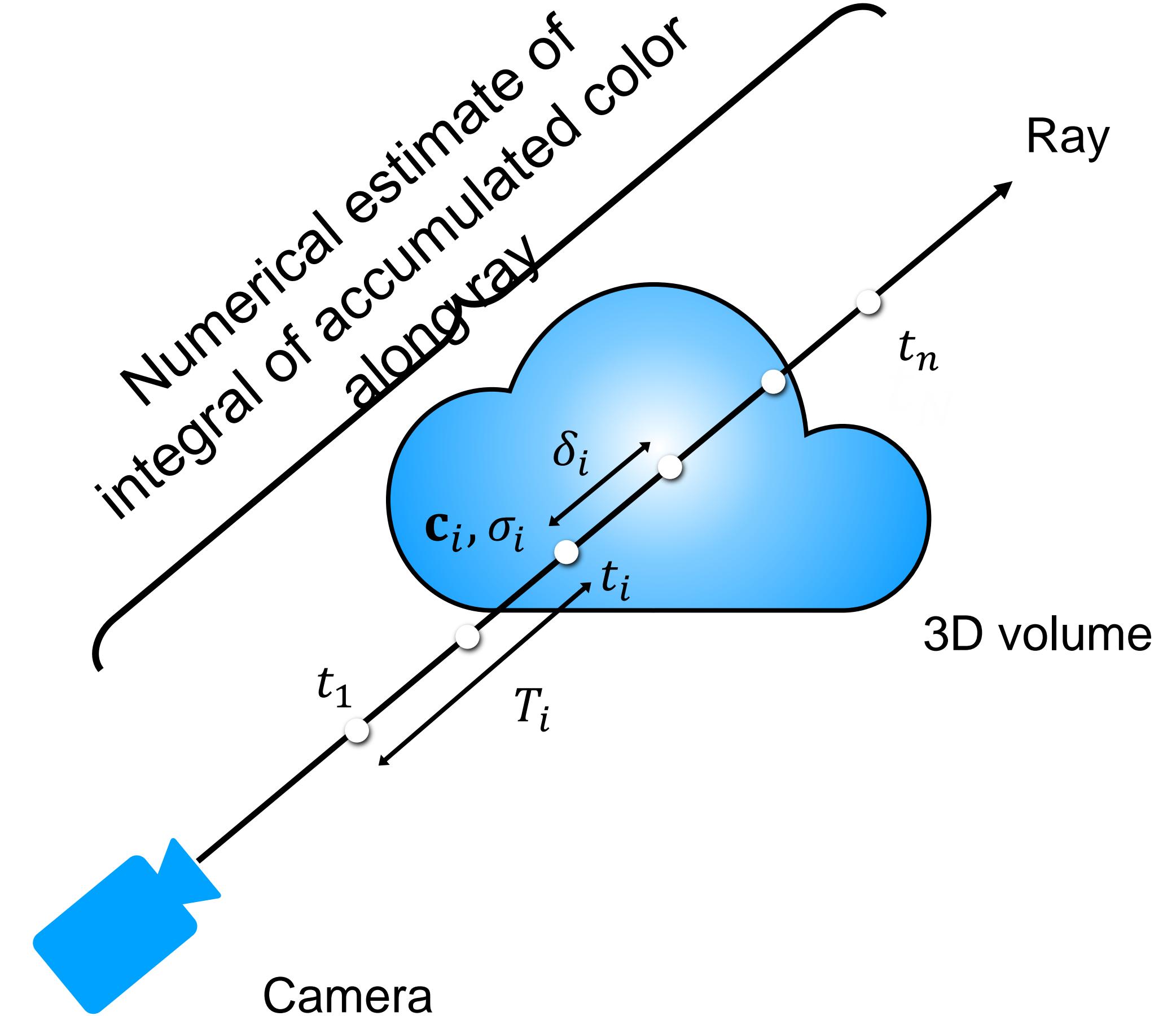
weights

colors

How much light is blocked earlier along ray:

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

Slight modification: 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 $\underline{\delta}$:
$$\alpha_i = 1 - \exp(-\sigma_i \delta_i)$$



Volume rendering estimation: integrating color along a ray

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

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

final rendered color along ray

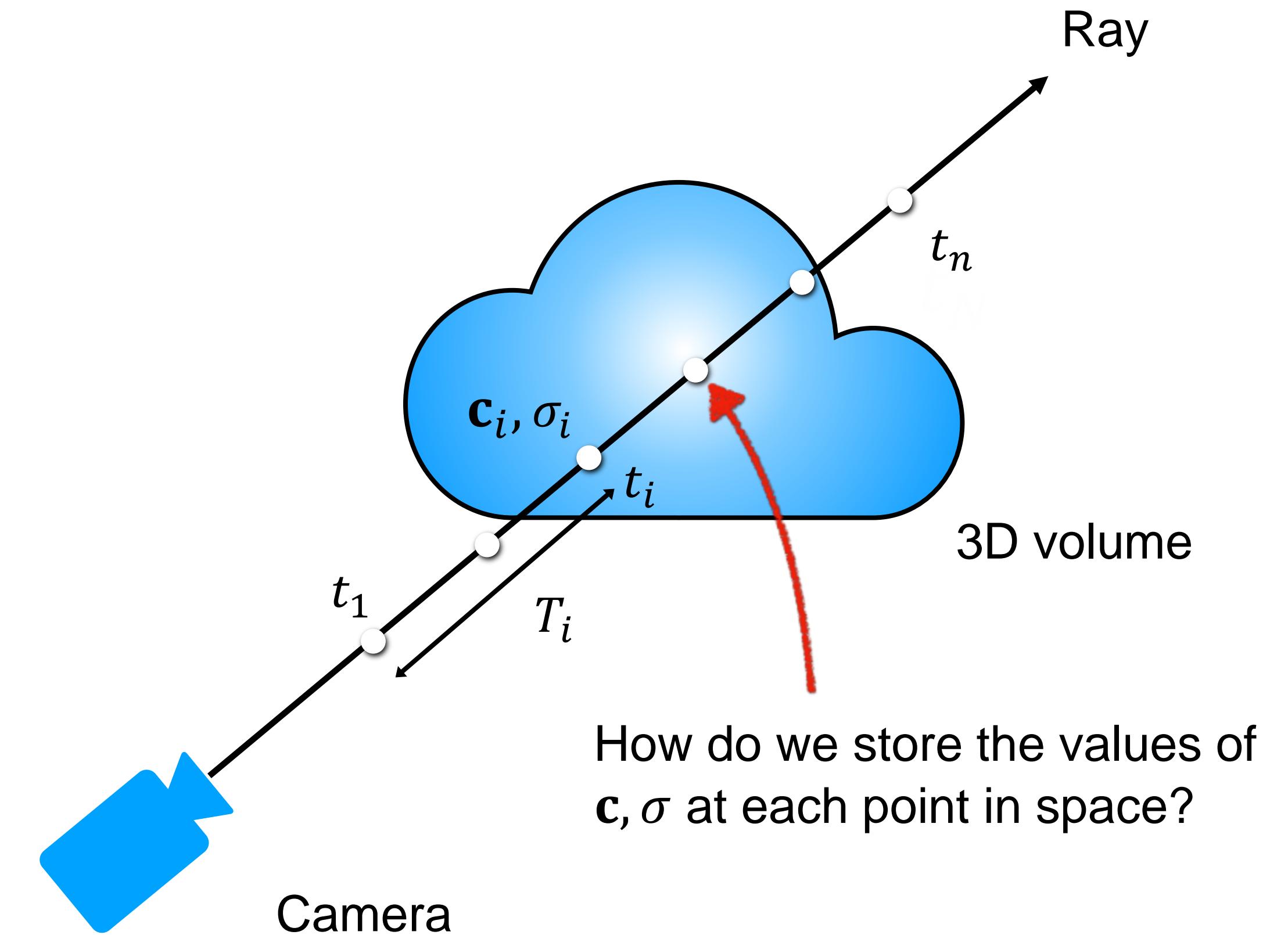
weights

colors

How much light is blocked earlier along ray:

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

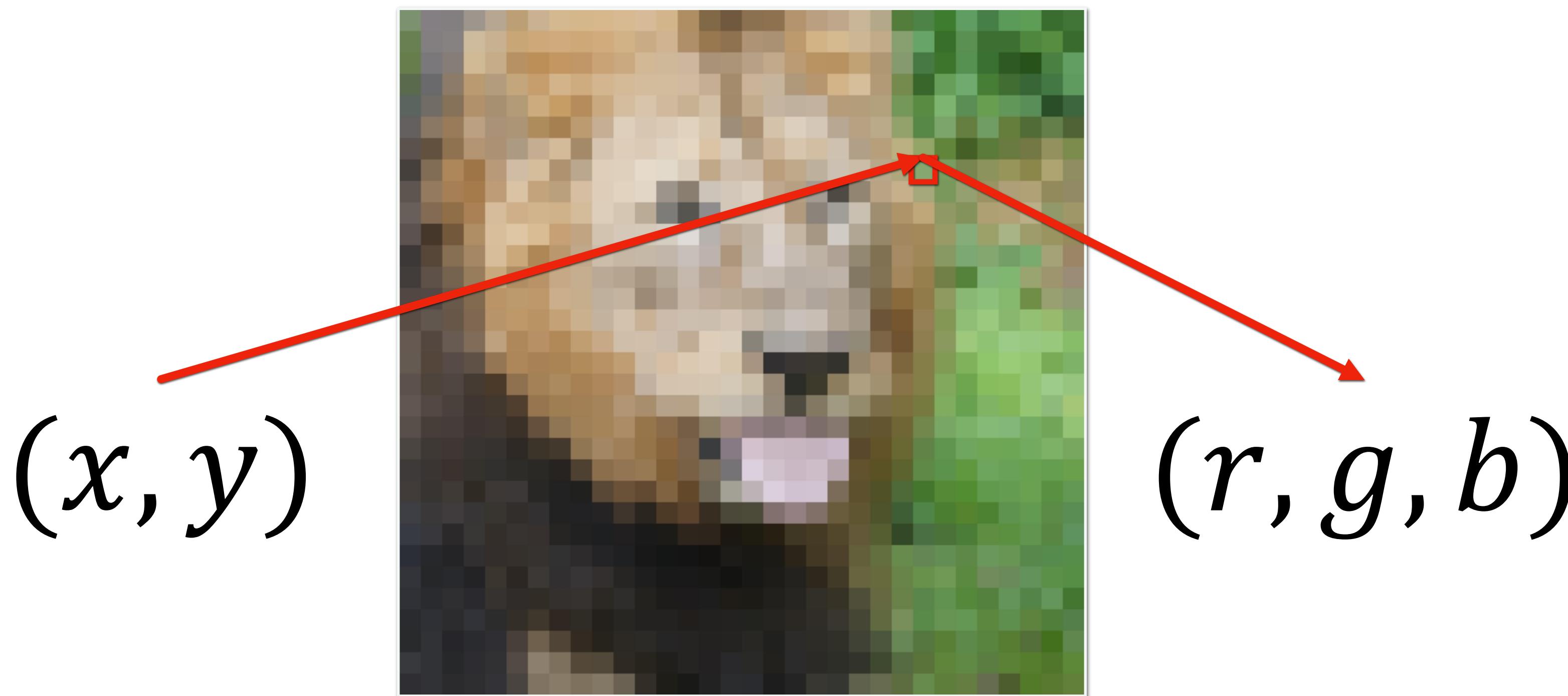
Computing the color for a set of rays through the pixels of an image yields a rendered image



NeRF Overview

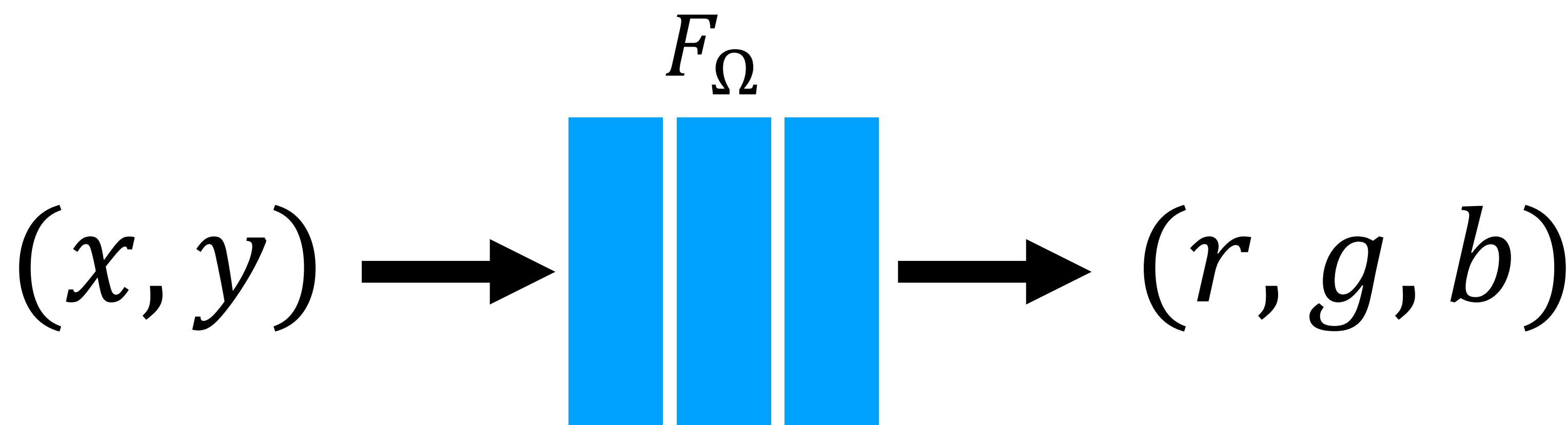
- ▶ Volumetric rendering
- ▶ Neural networks as representations for spatial data
- ▶ Neural Radiance Fields (NeRF)

Toy problem: storing 2D image data



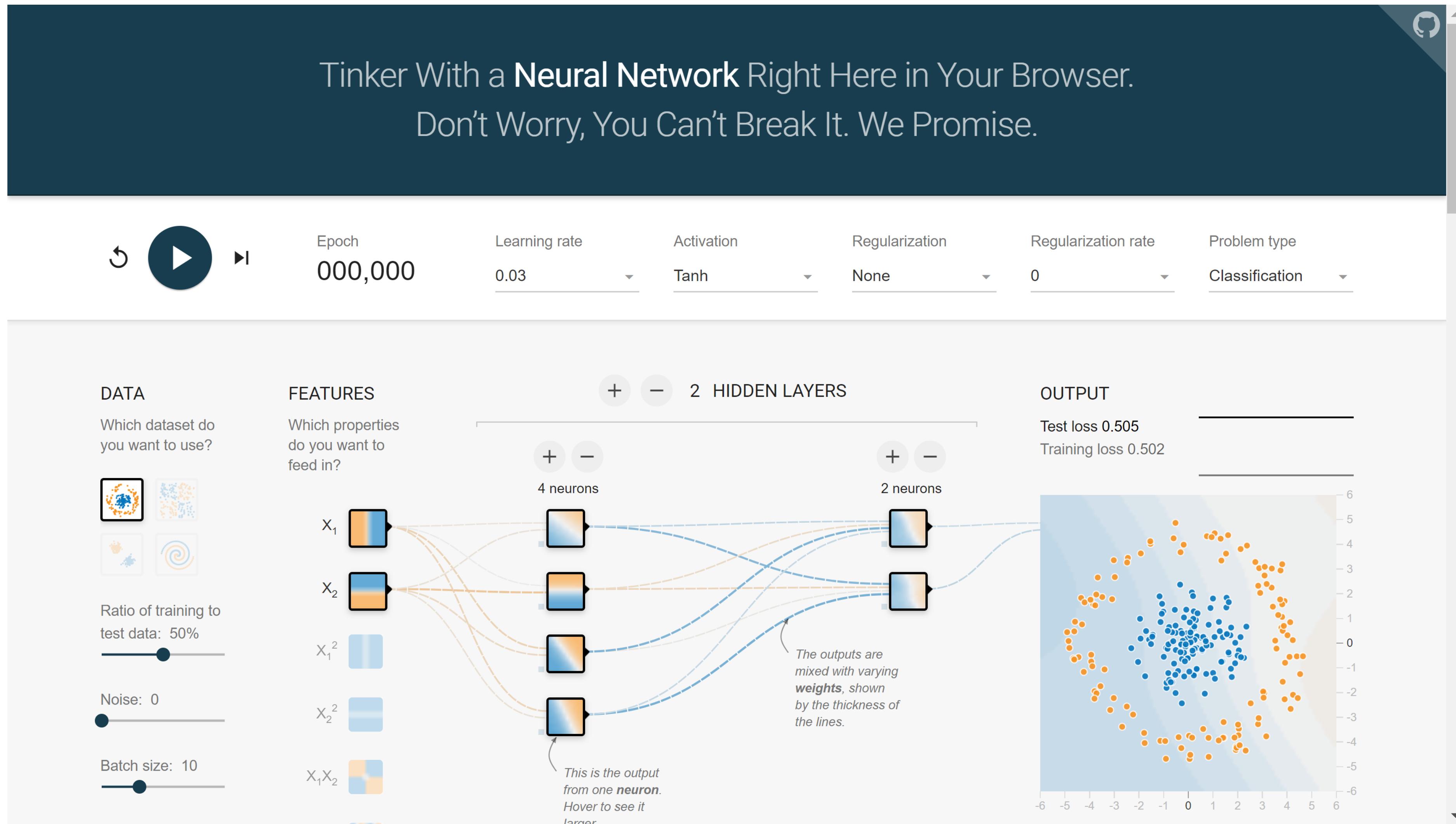
Usually we store an image as a
2D grid of RGB color values

Toy problem: storing 2D image data



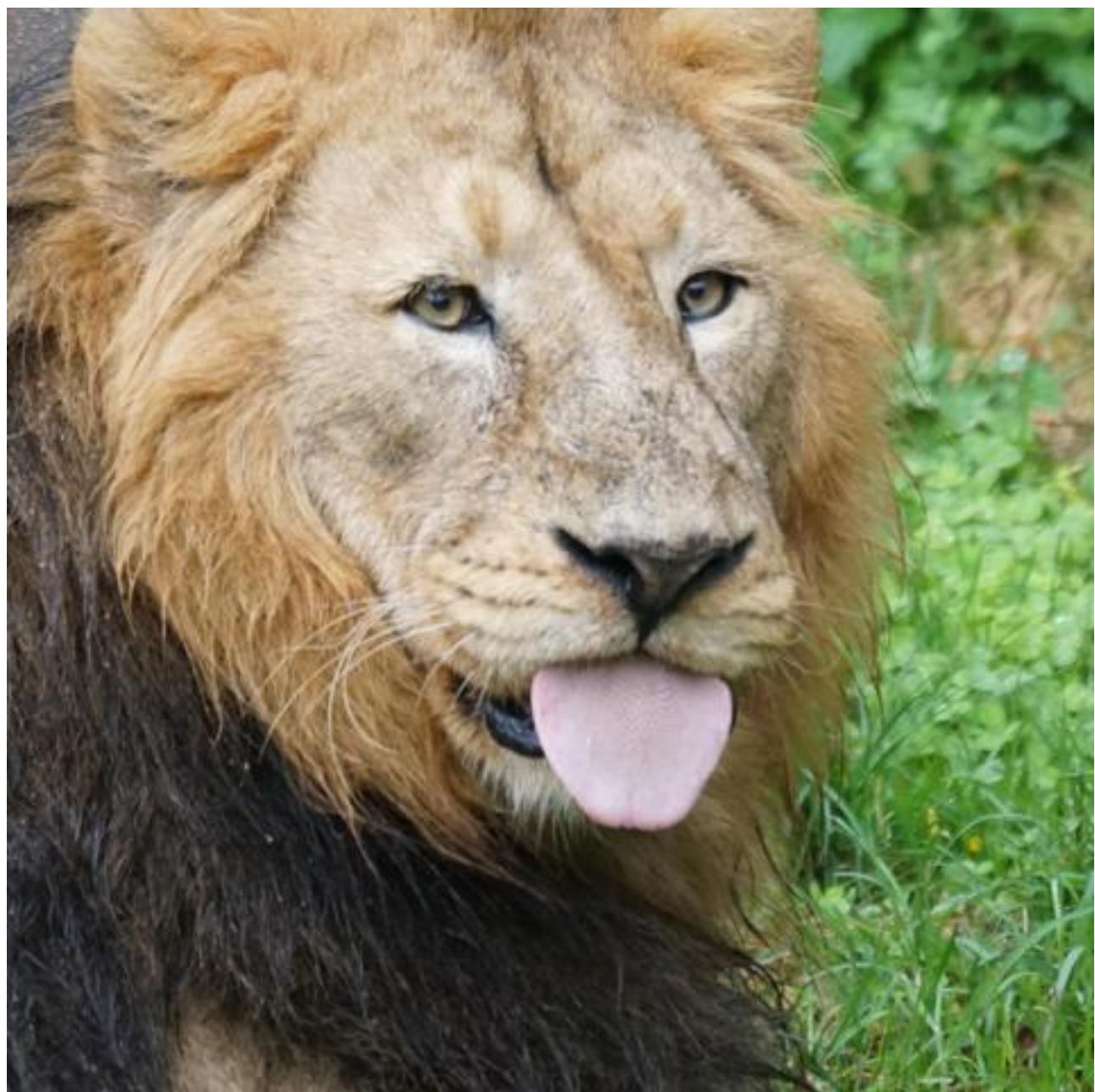
What if we train a simple fully-connected network (MLP) to do this instead?

Recall the TensorFlow playground



Same concept as before, except we are computing an image, instead of a classifier!

Naive approach fails!



Ground truth image



Neural network output fit
with gradient descent

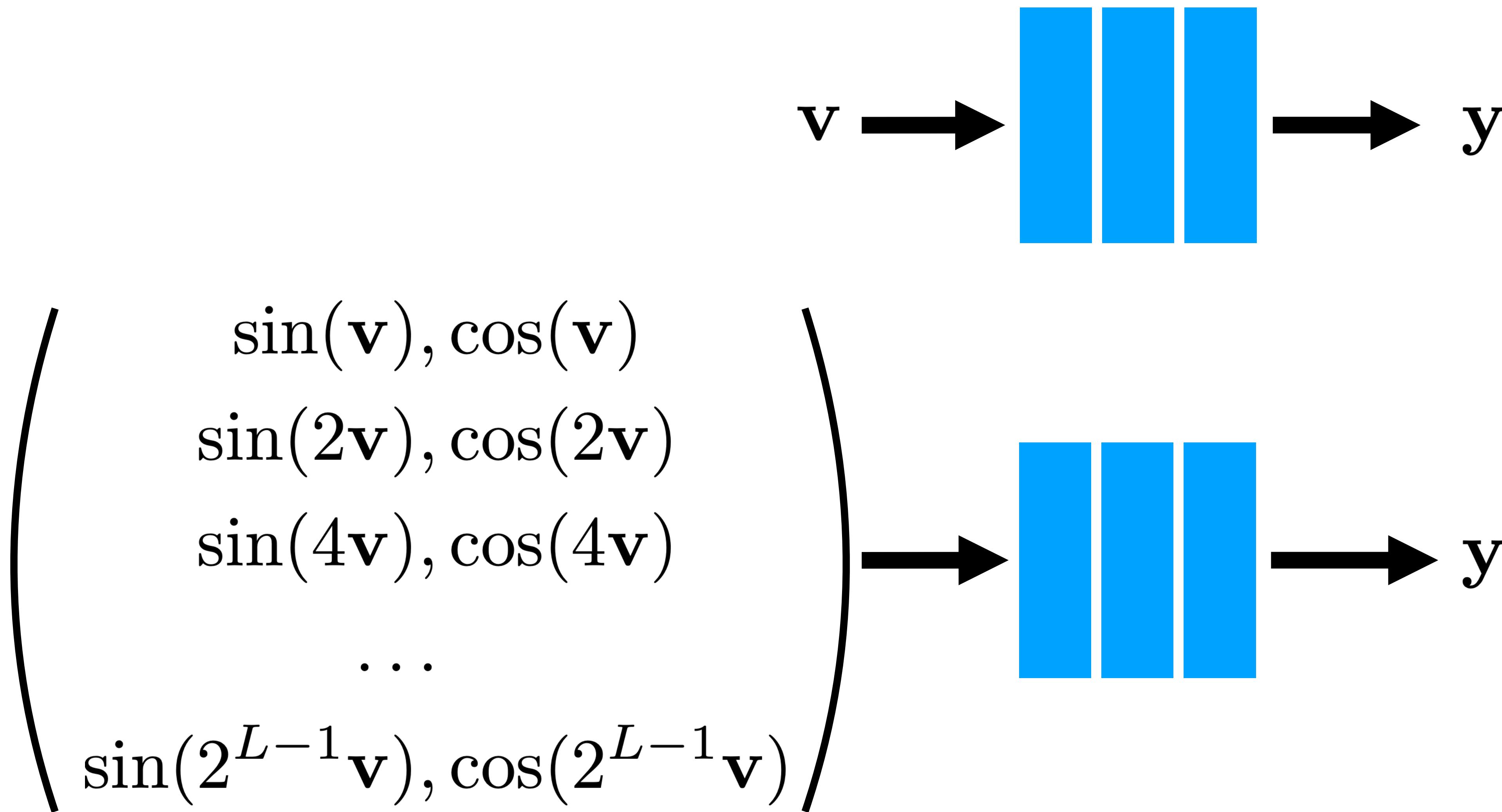
Problem:

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

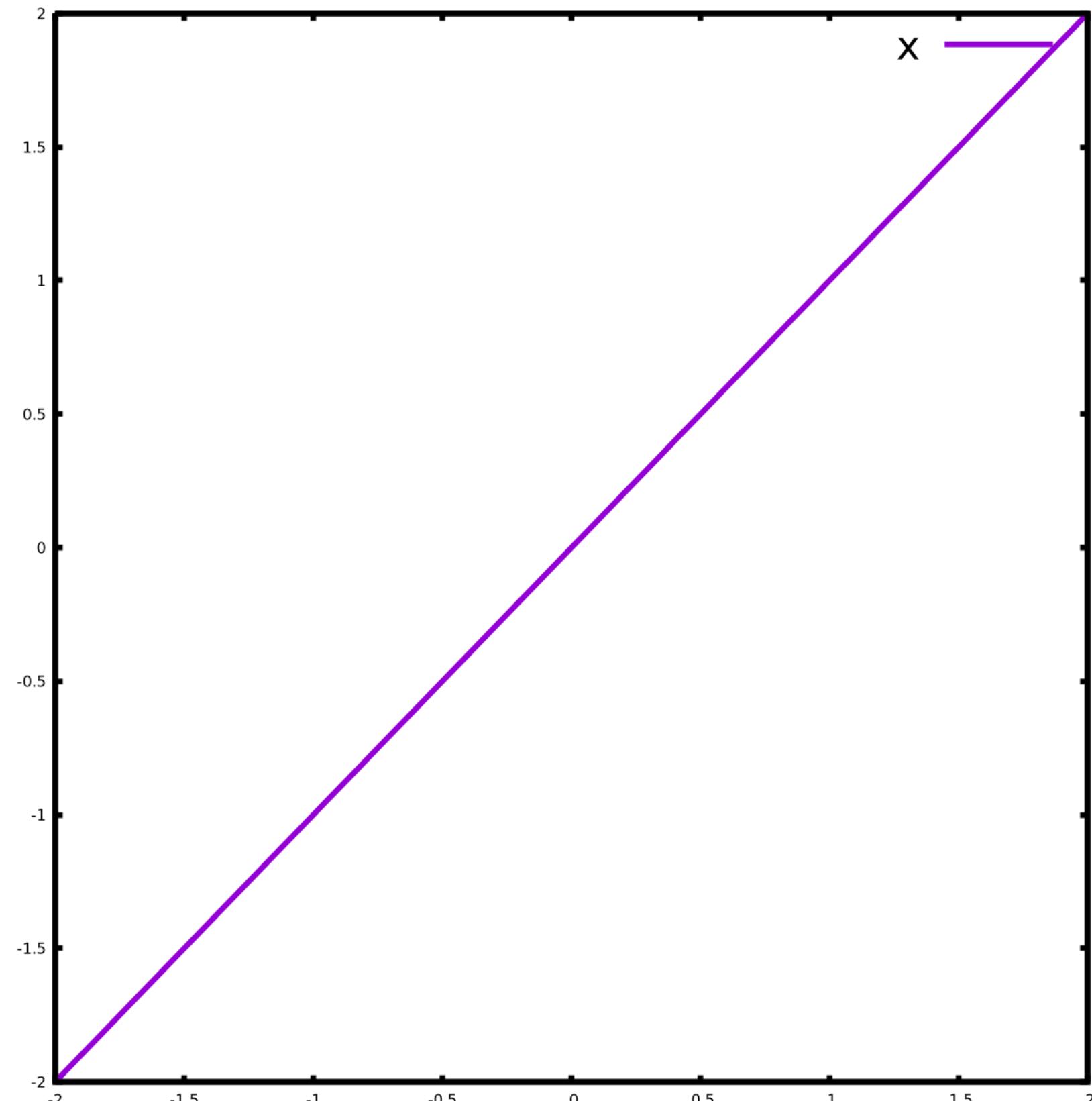
Solution:

Pass input coordinates through a
high frequency mapping first

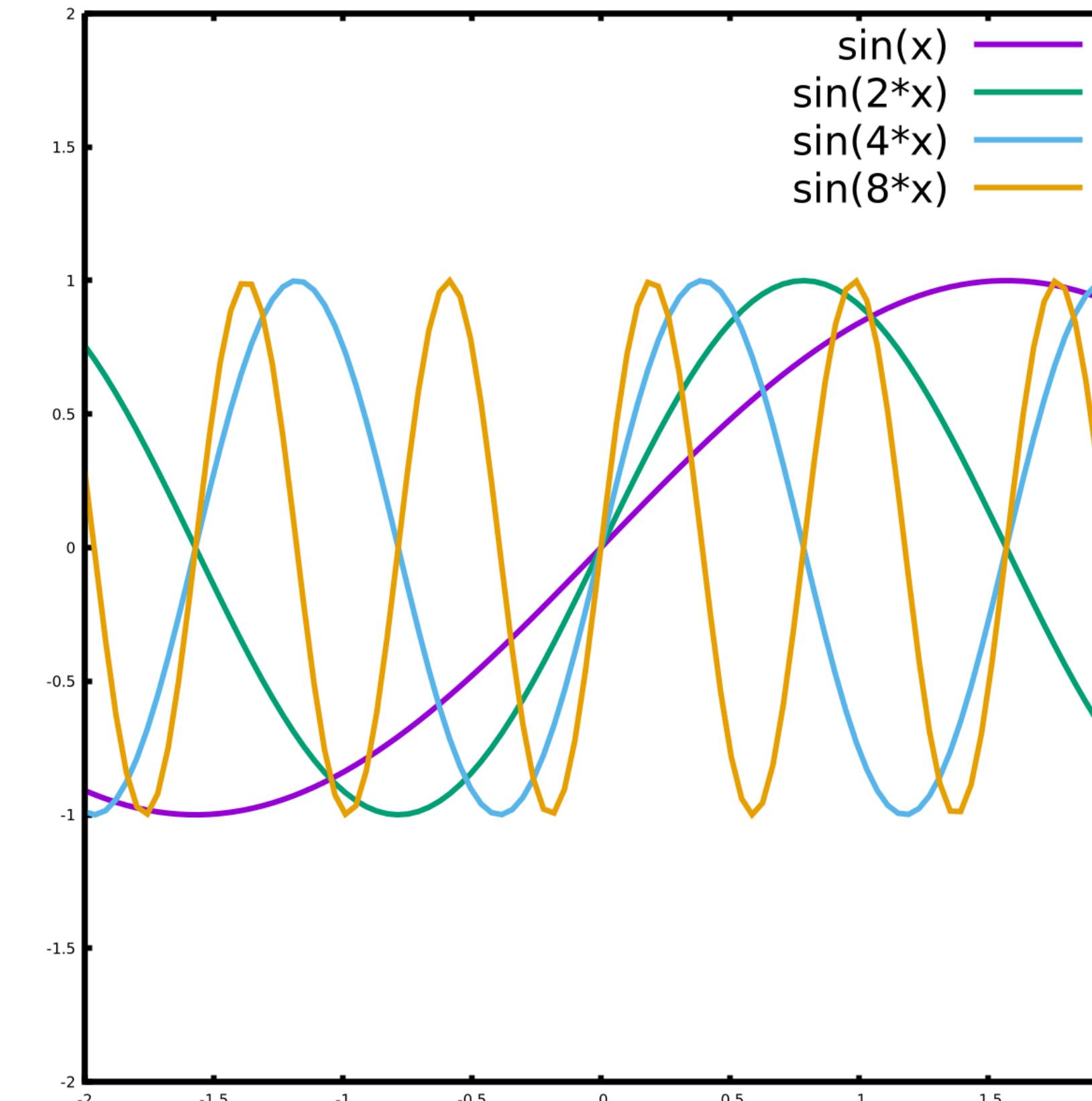
Example mapping: “positional encoding”



Positional encoding

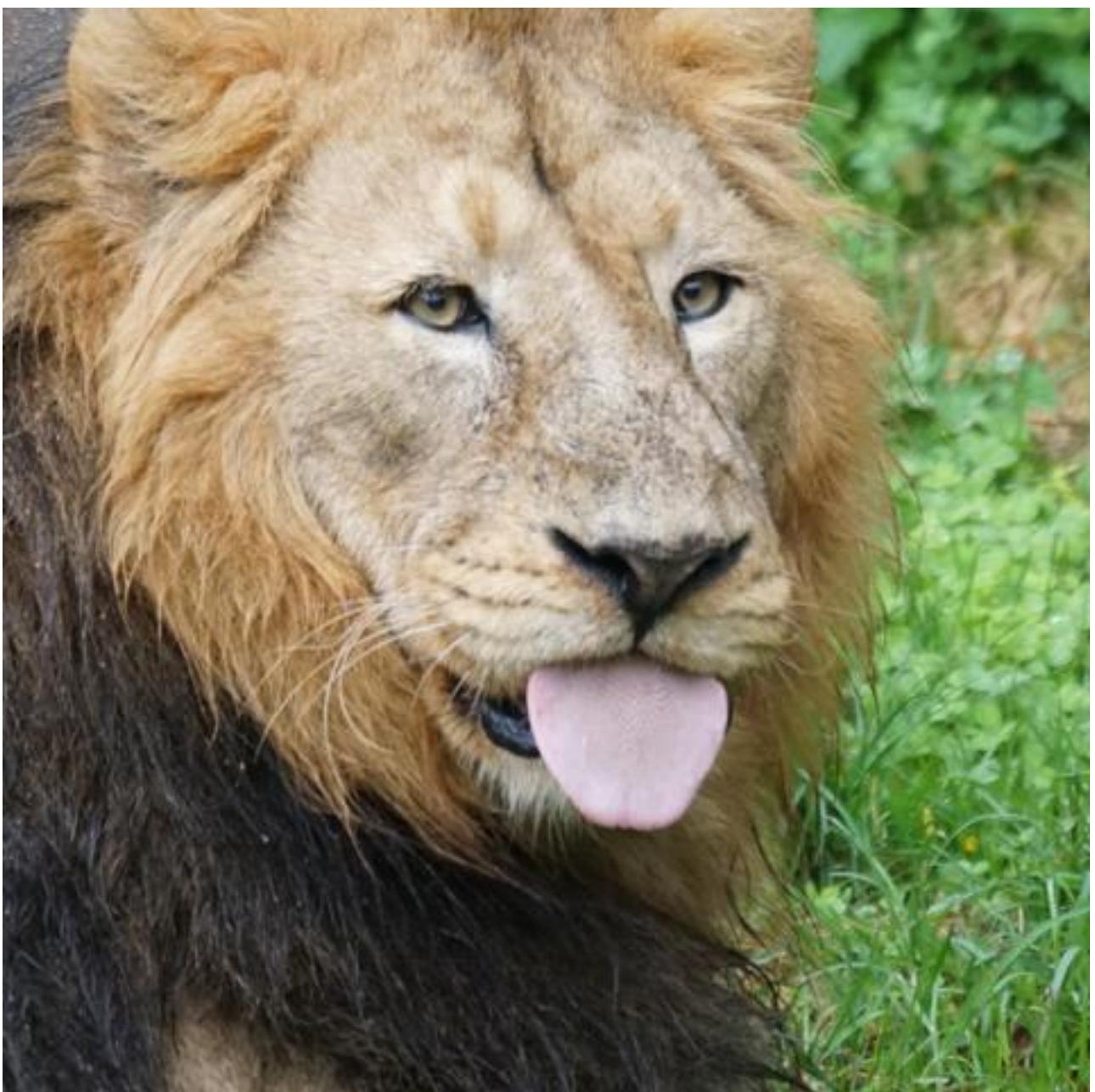


Raw encoding of a number x



“Positional encoding” of a number x

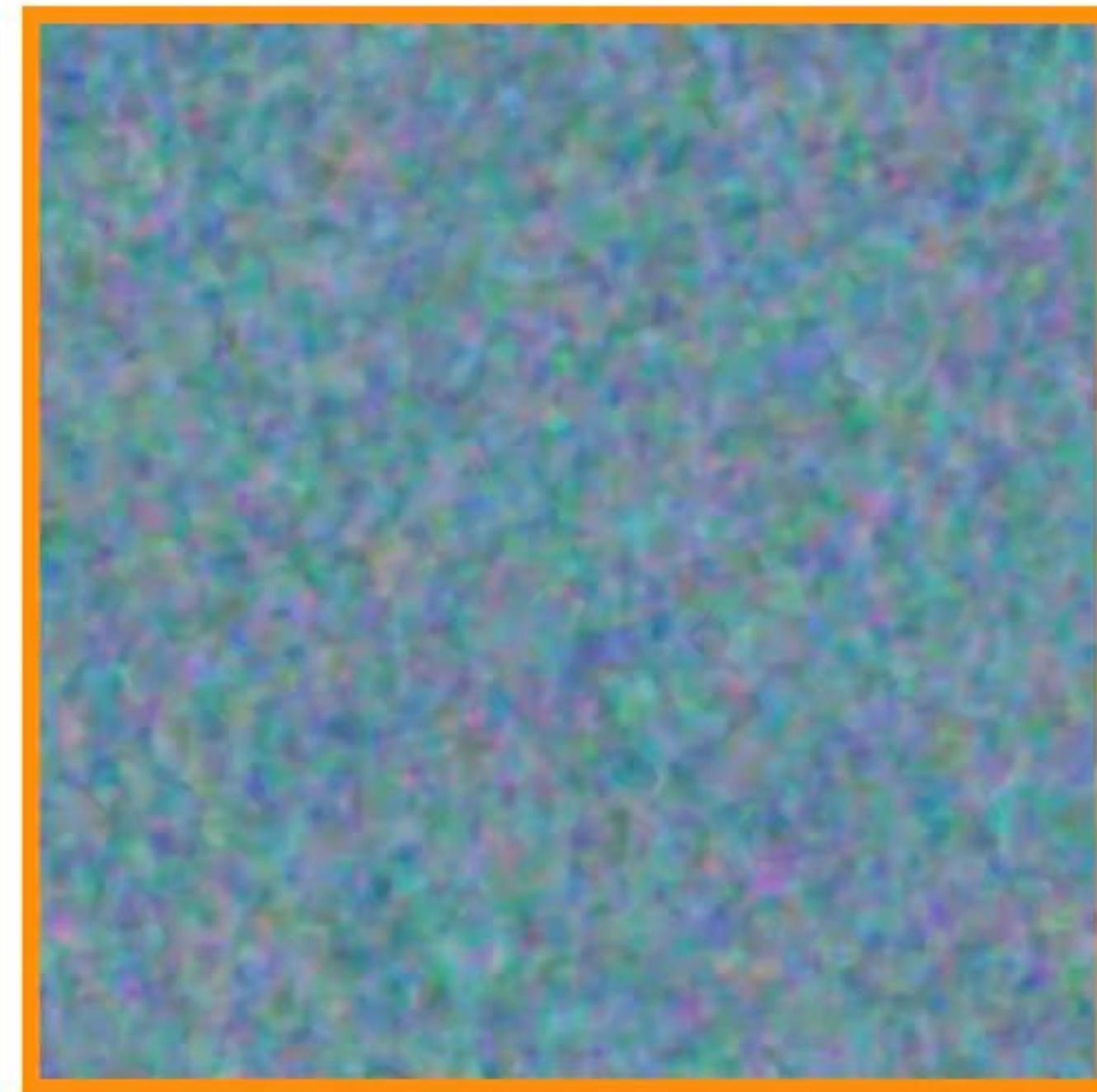
Problem solved!



Ground truth image

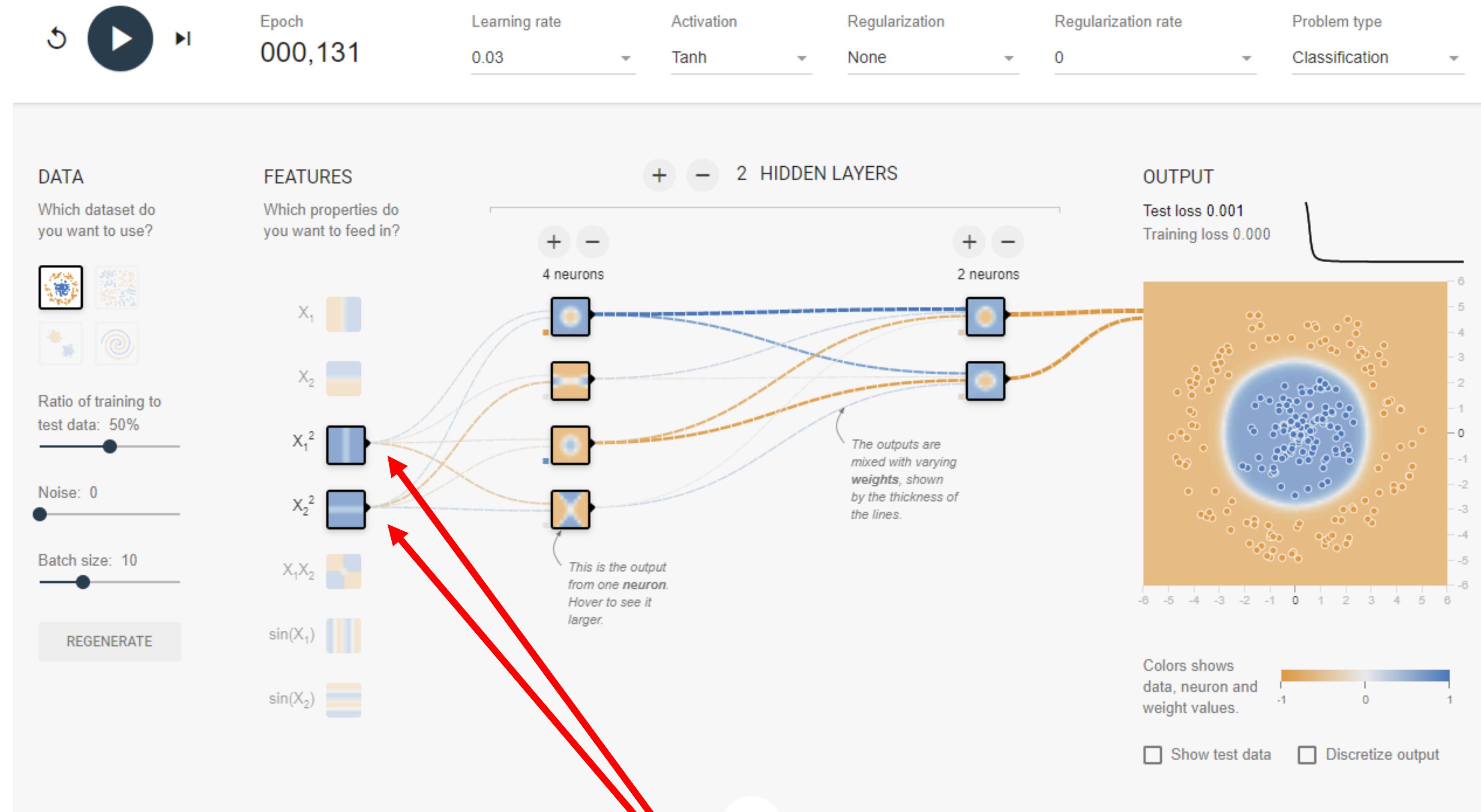


Neural network output without
high frequency mapping



Neural network output with high
frequency mapping

Sometimes a better input encoding is all you need



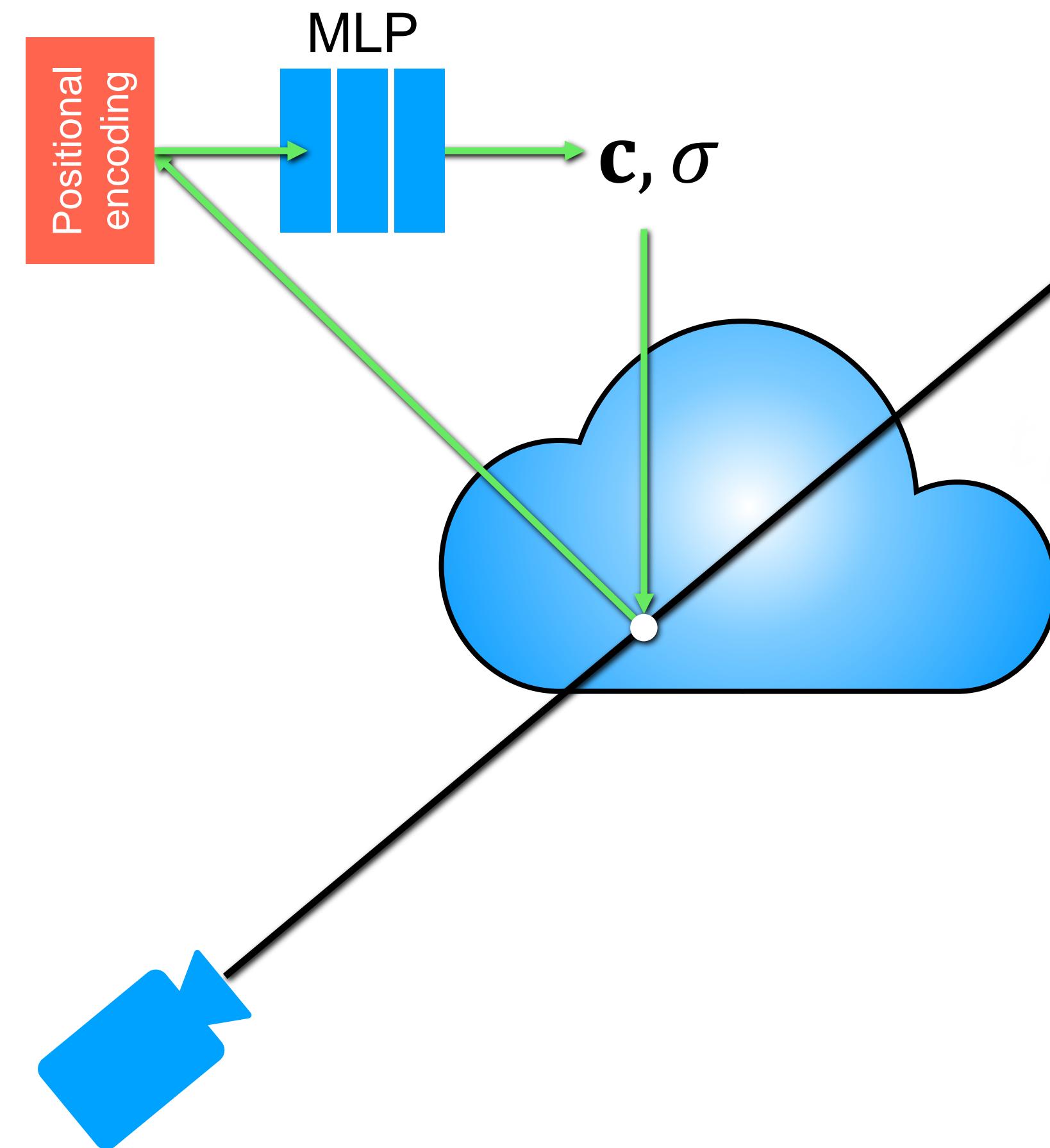
Recall “squared” encoding in TensorFlow Playground

NeRF Overview

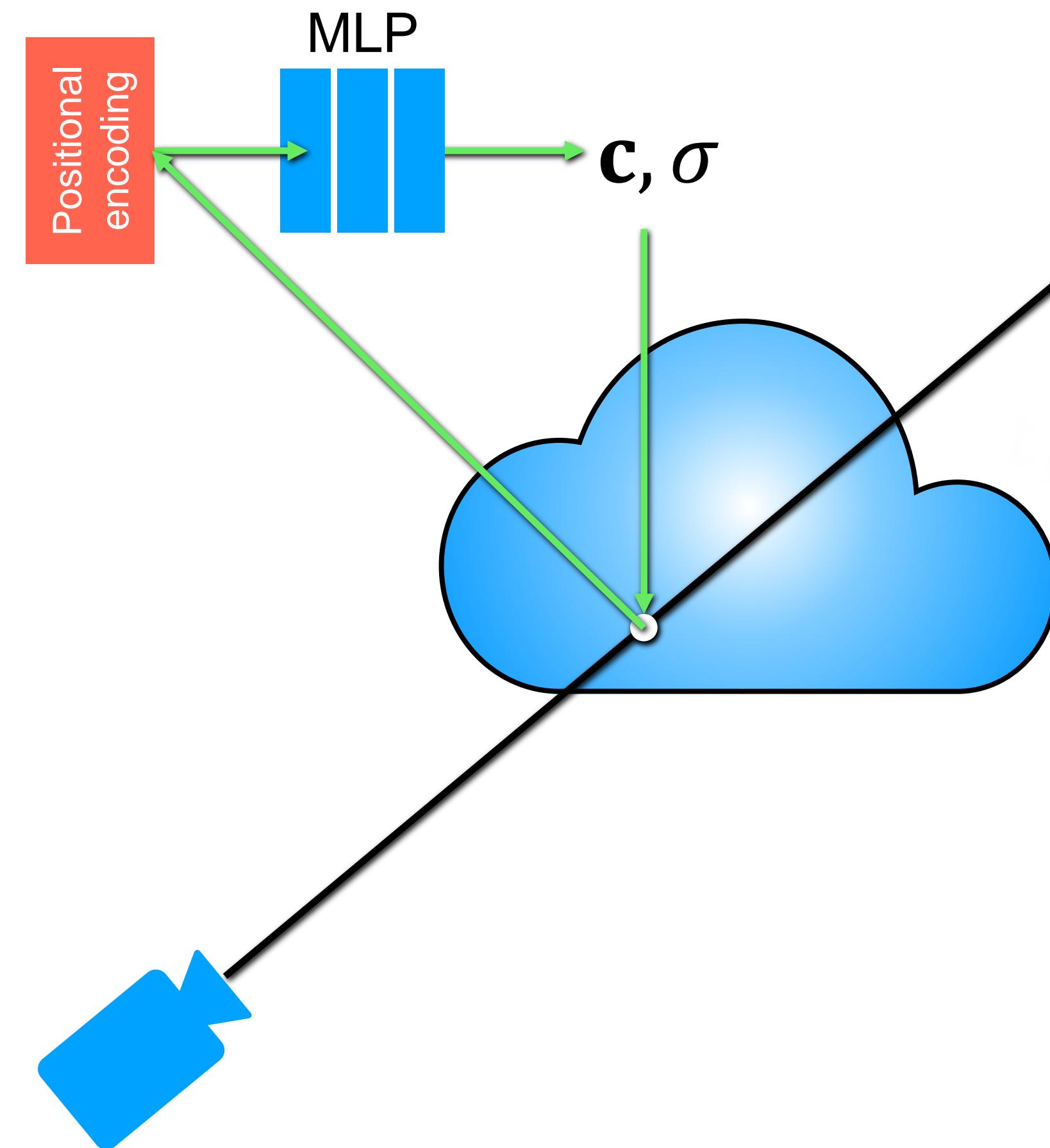
- ▶ Volumetric rendering
- ▶ Neural networks as representations for spatial data
- ▶ Neural Radiance Fields (NeRF)

**NeRF = volume rendering +
coordinate-based network**

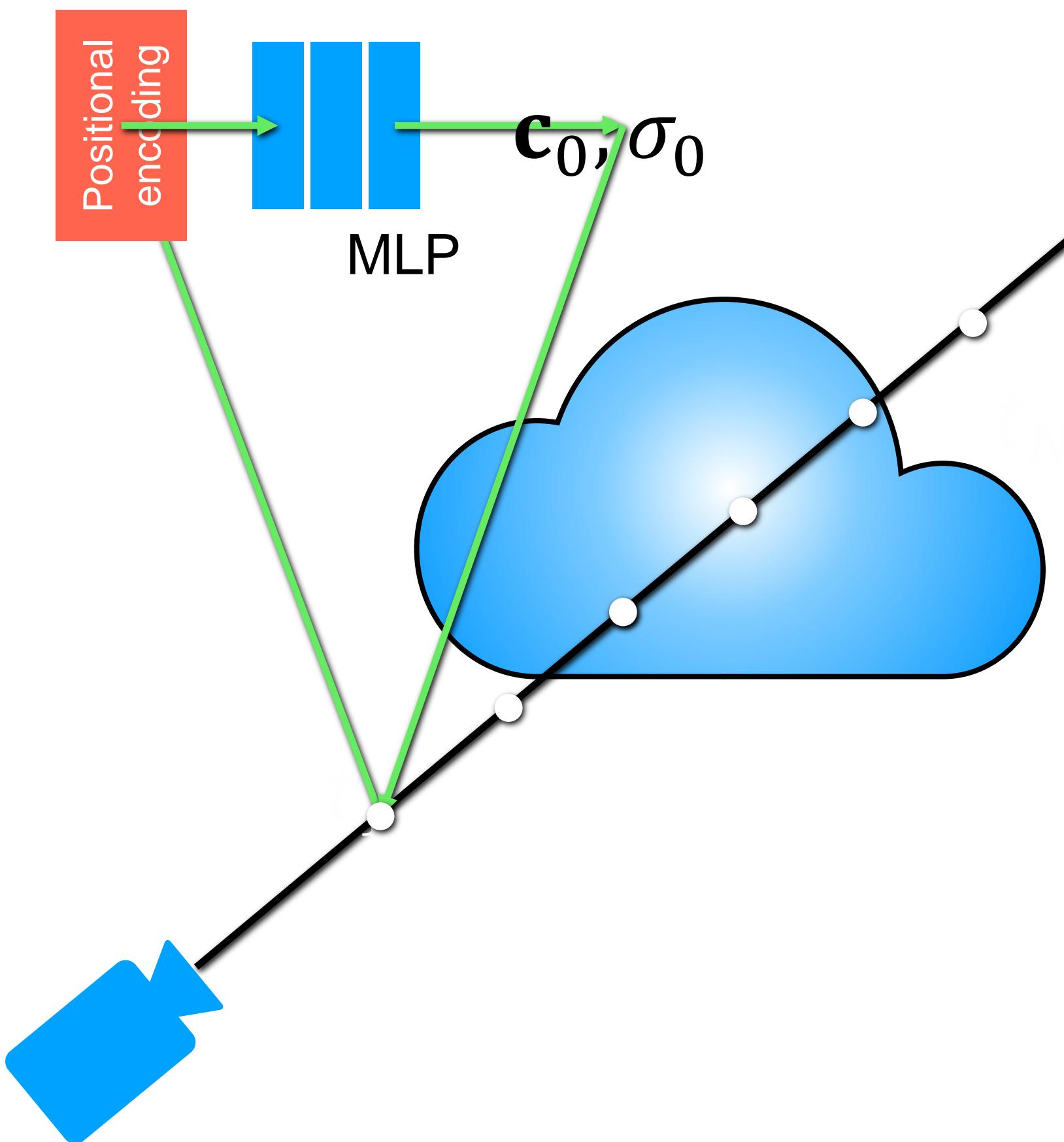
How do we store the values of c, σ at each point in space?



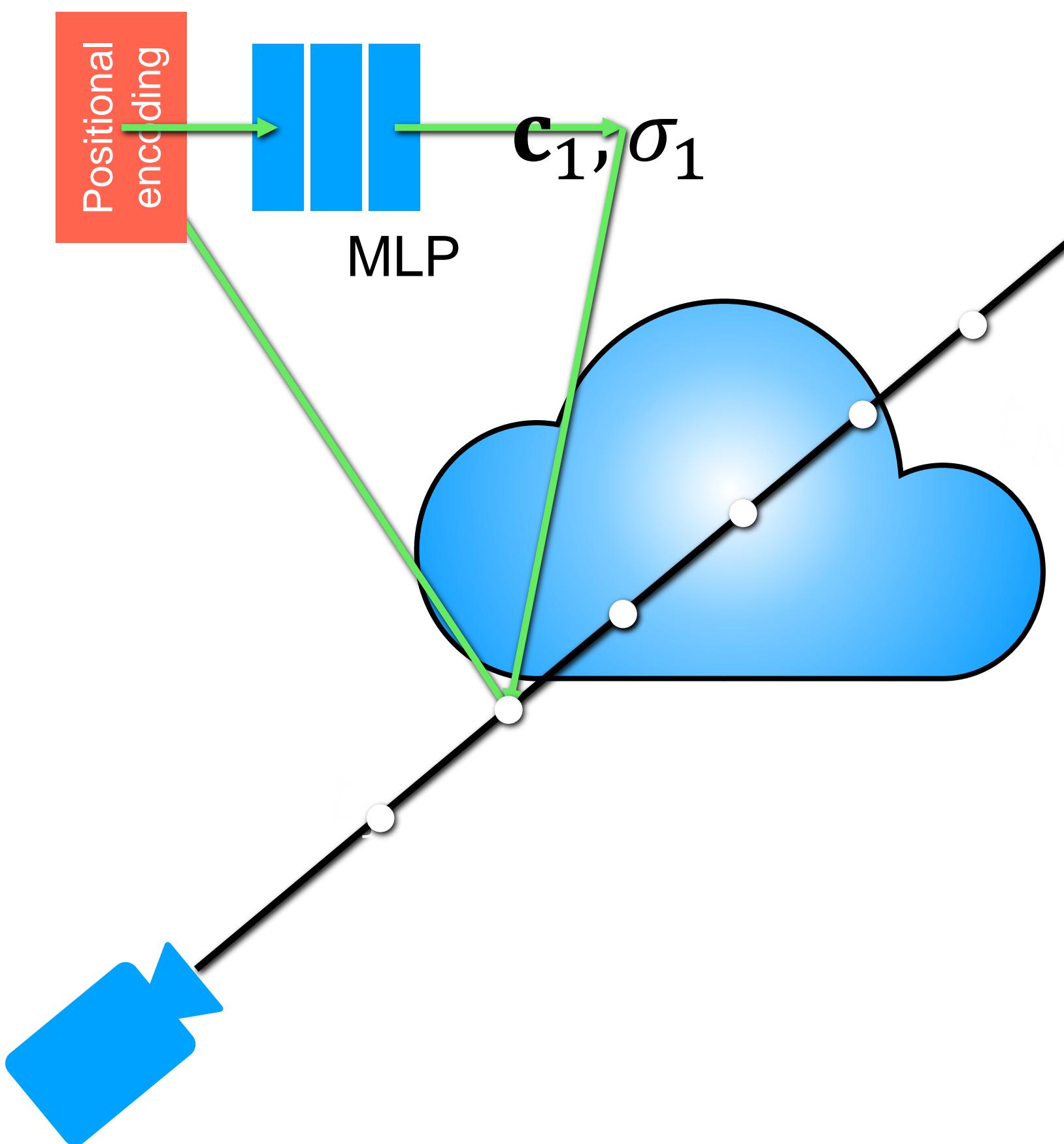
How do we store the values of c, σ at each point in space?



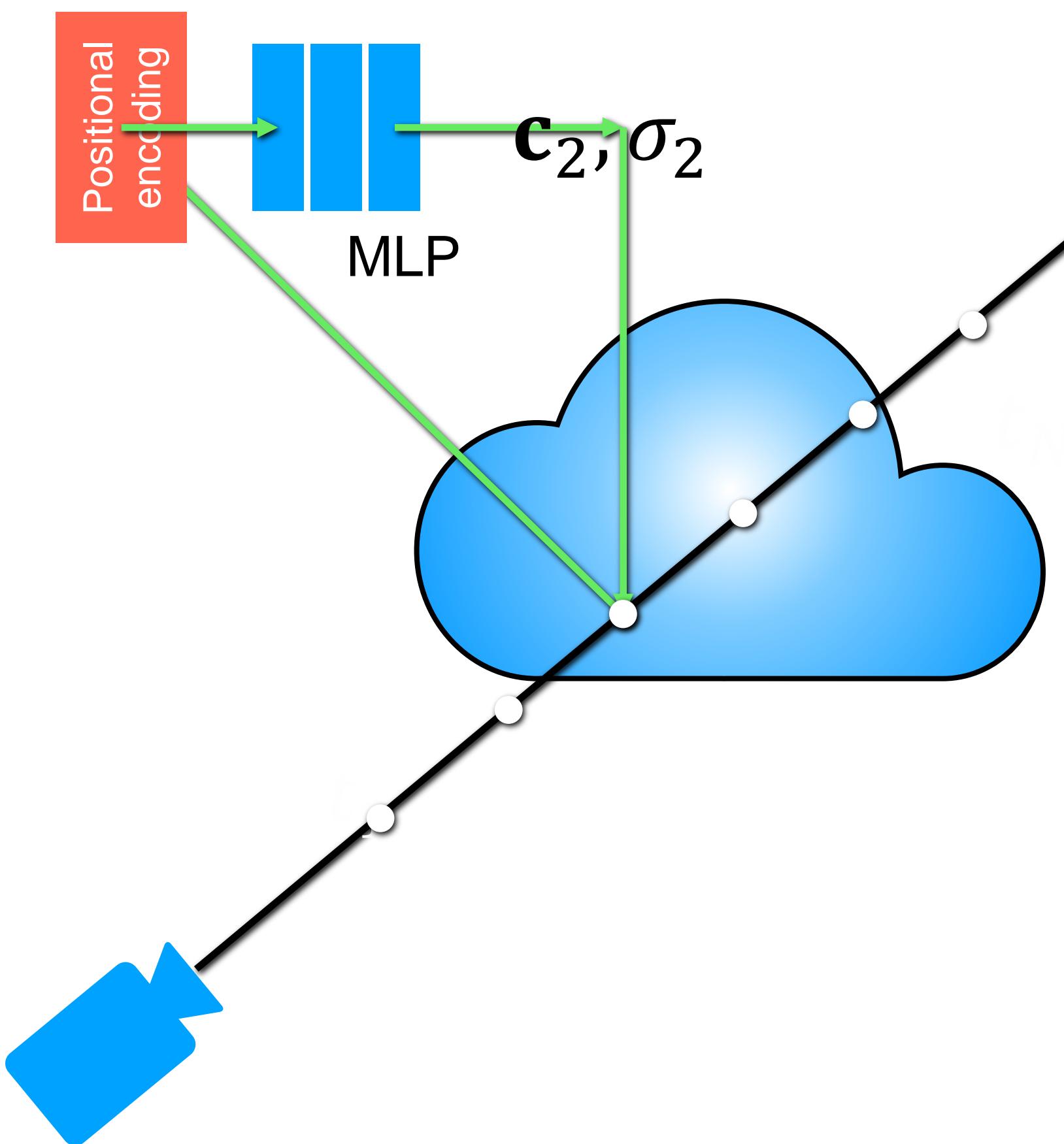
How do we store the values of c, σ at each point in space?



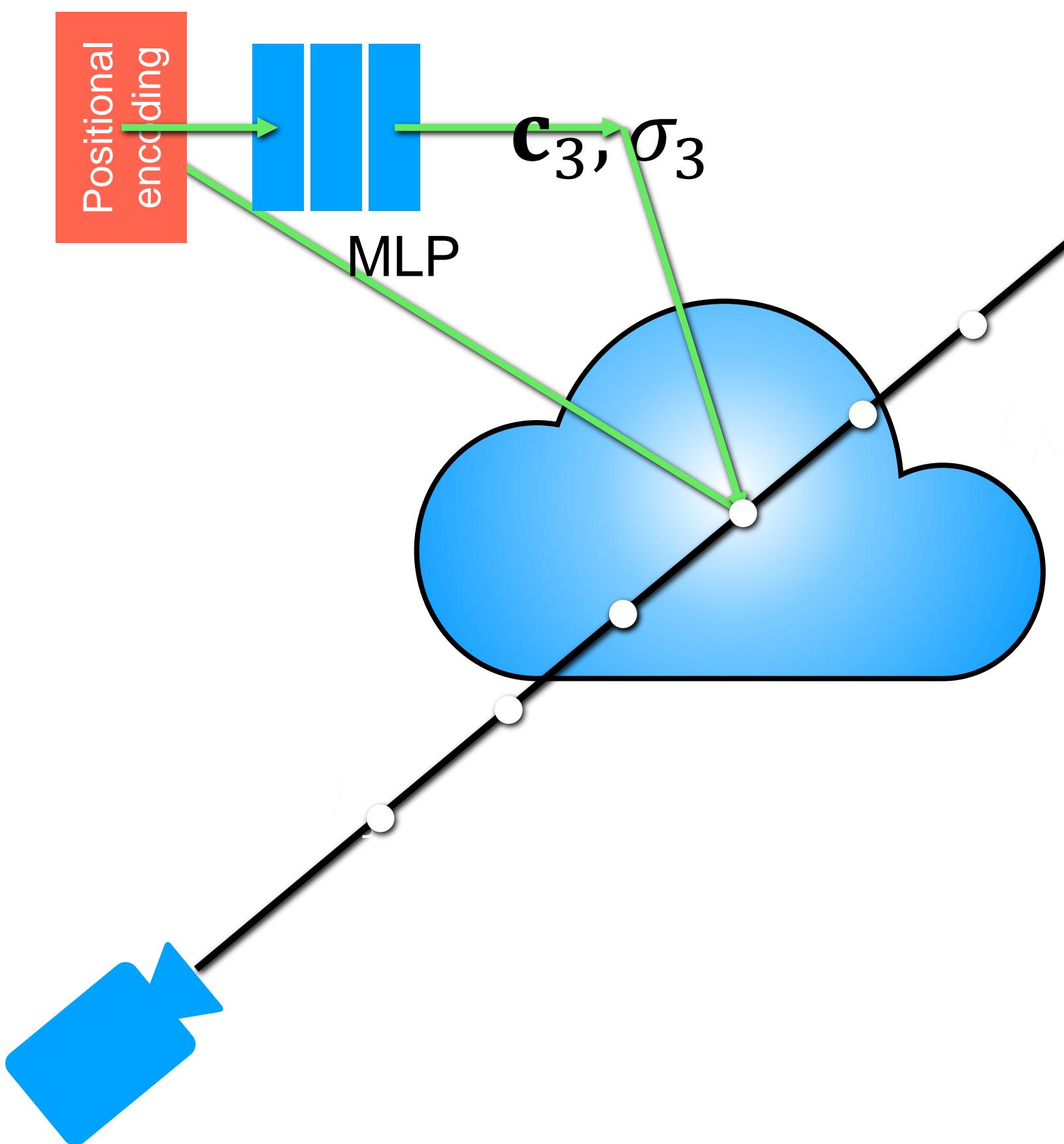
How do we store the values of c, σ at each point in space?



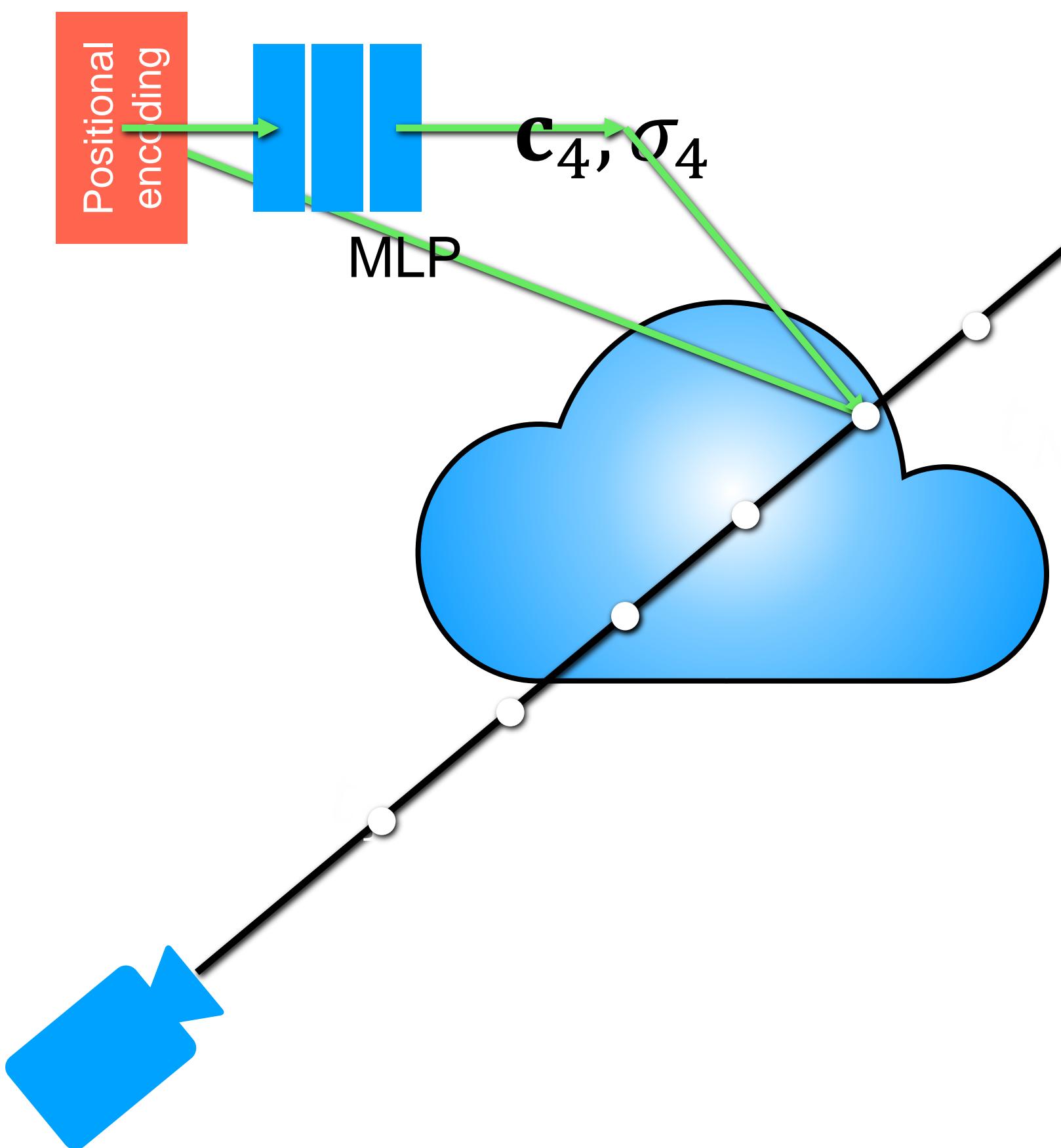
How do we store the values of c, σ at each point in space?



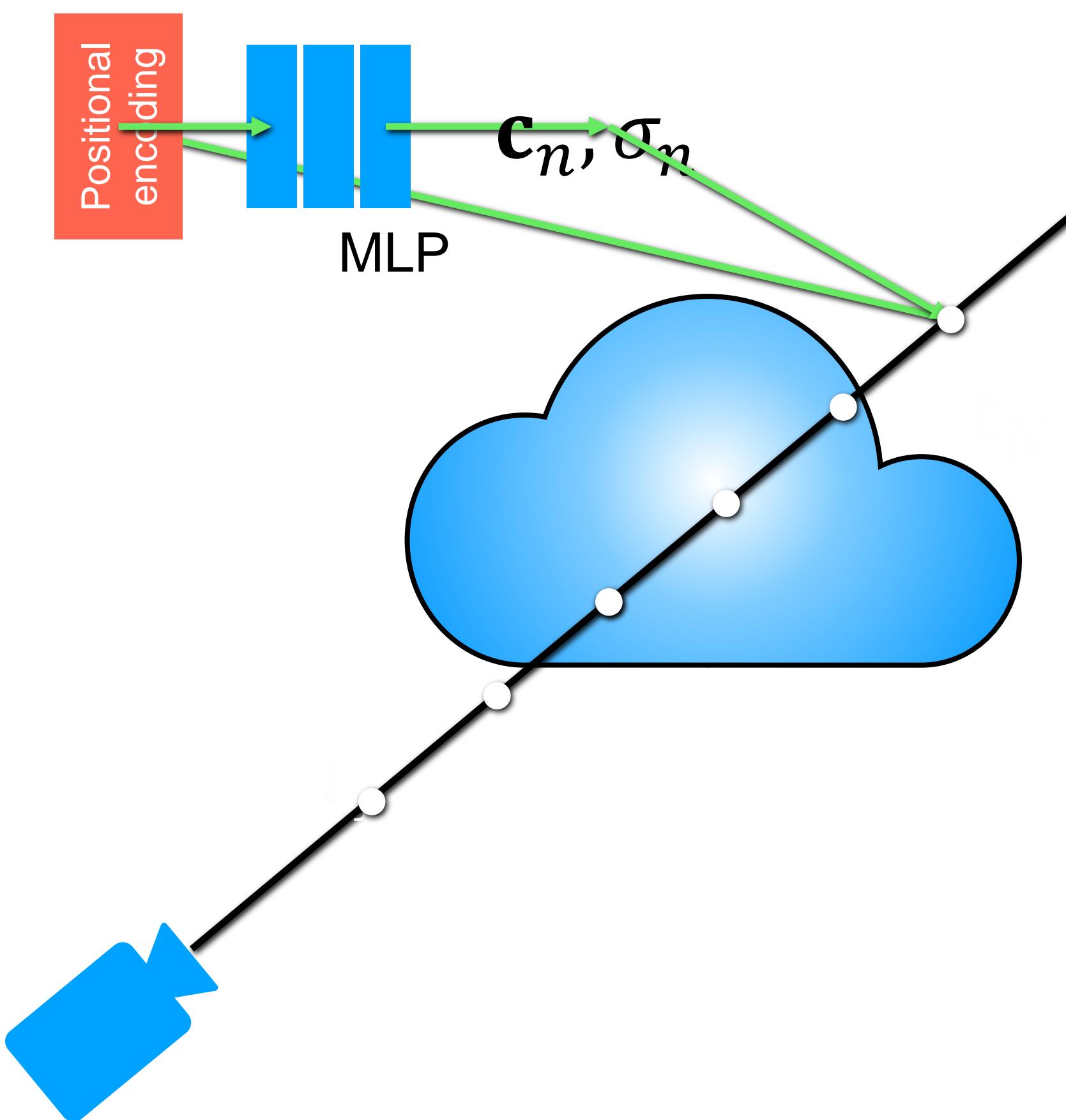
How do we store the values of c, σ at each point in space?



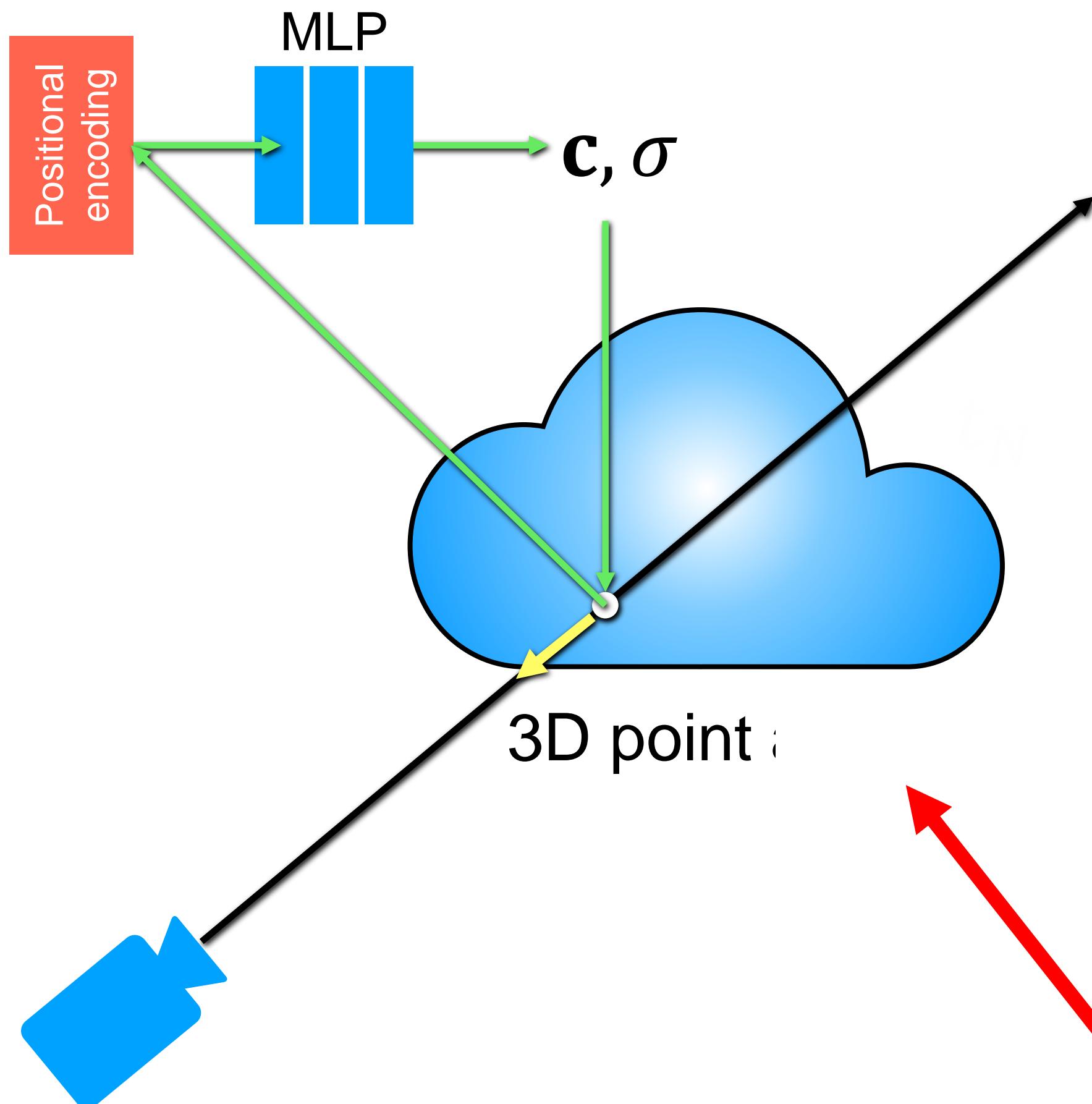
How do we store the values of c, σ at each point in space?



How do we store the values of c, σ at each point in space?

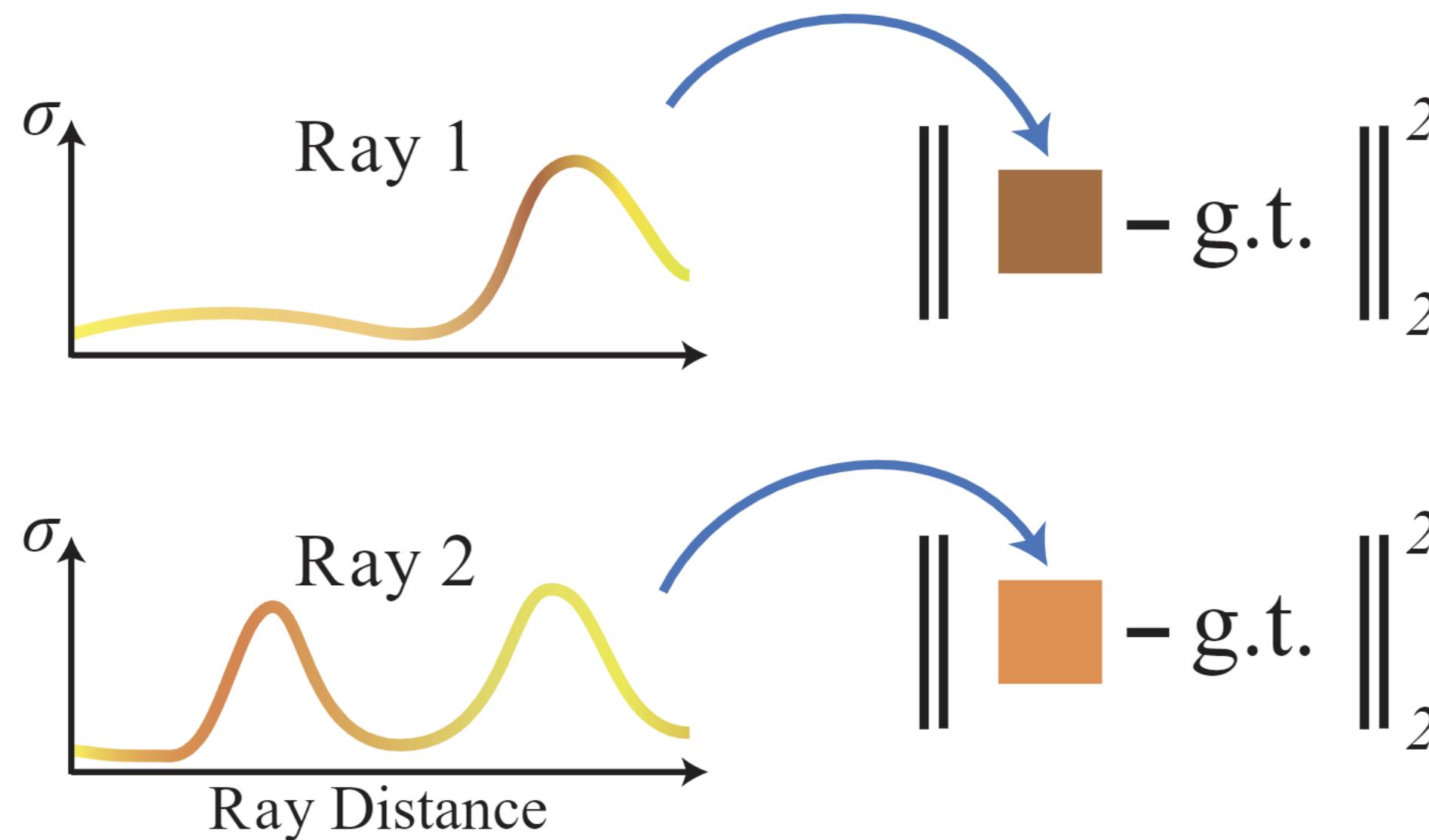
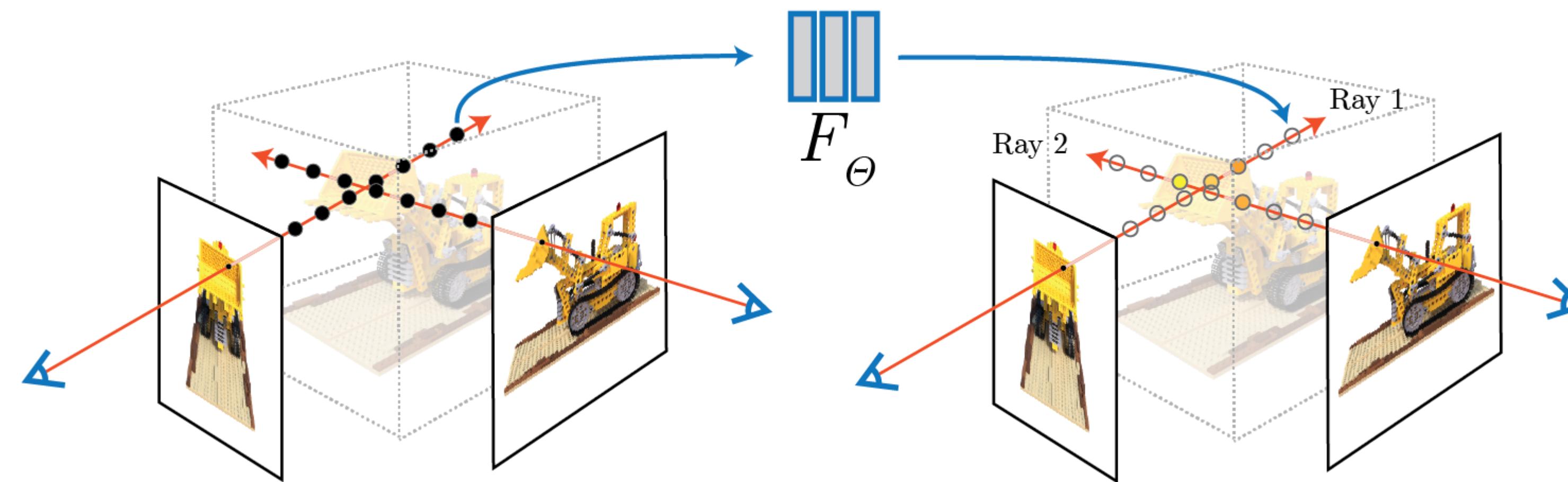


Extension: view-dependent field



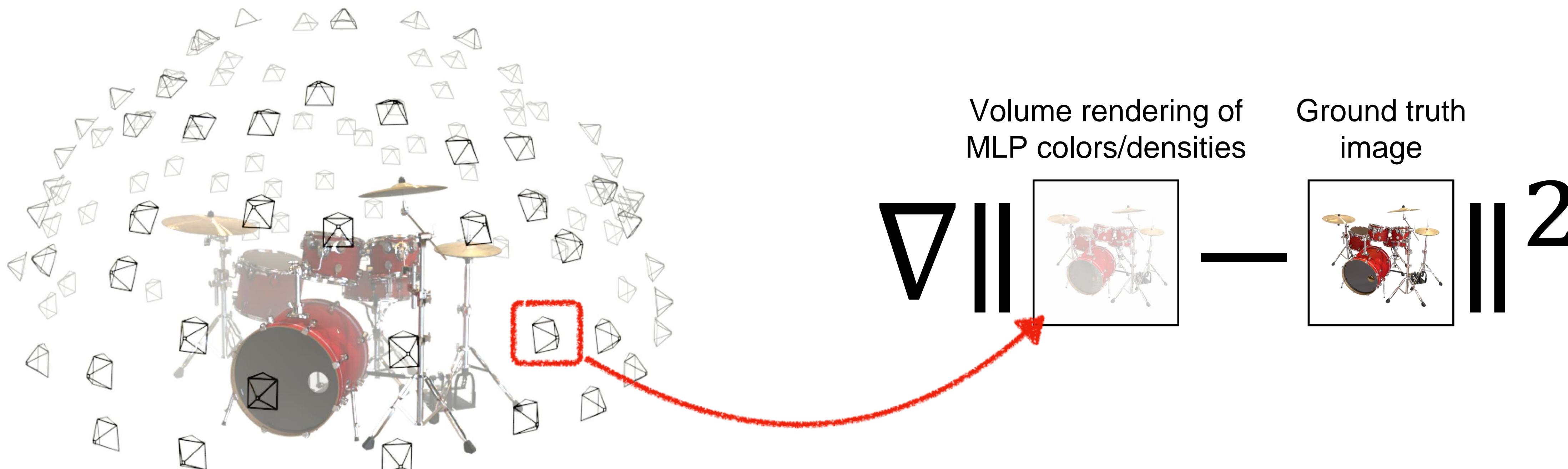
Include the ray direction in the input to the MLP → allows for capturing and rendering view-dependent effects (e.g., shiny surfaces)

Putting it all together



Adapted from material from Pratul Srinivas

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



Results



NeRF encodes convincing view-dependent effects using directional dependence



Adapted from material from Pratul Srinivas

NeRF encodes convincing view-dependent effects using directional dependence



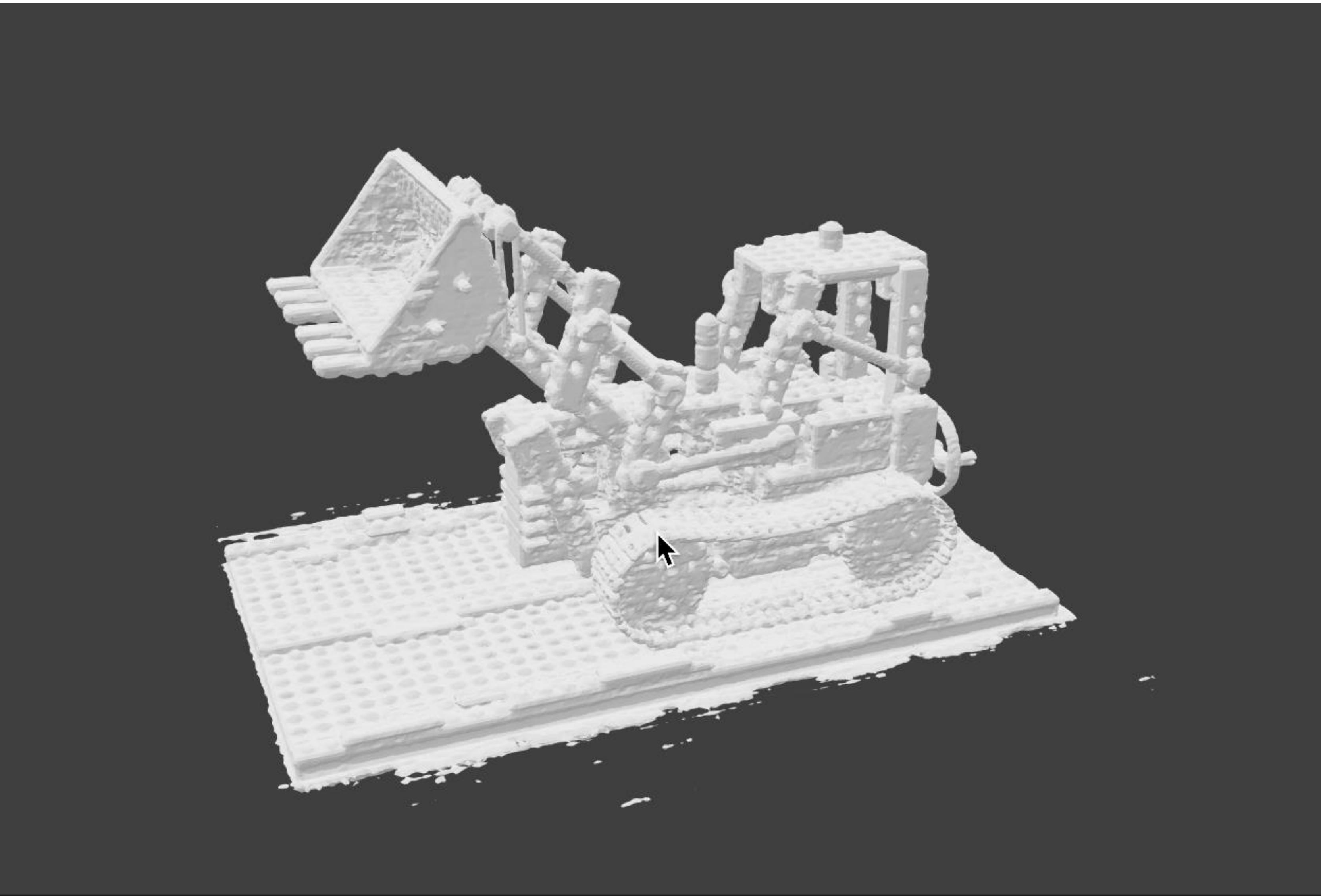
Adapted from material from Pratul Srinivas

NeRF encodes detailed scene geometry with occlusion effects



Adapted from material from Pratul Srinivas

NeRF encodes detailed scene geometry

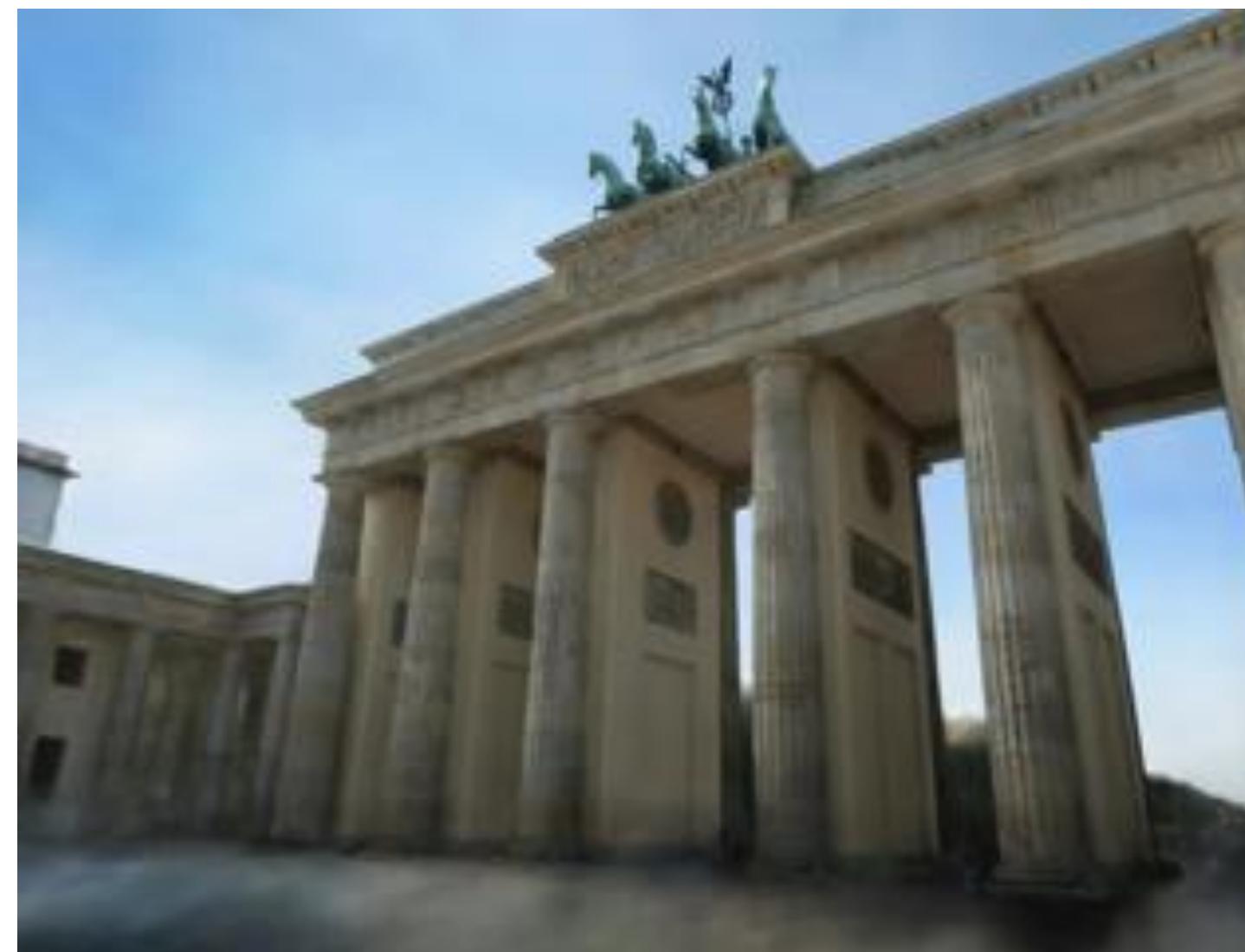


Adapted from material from Pratul Srinivas

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

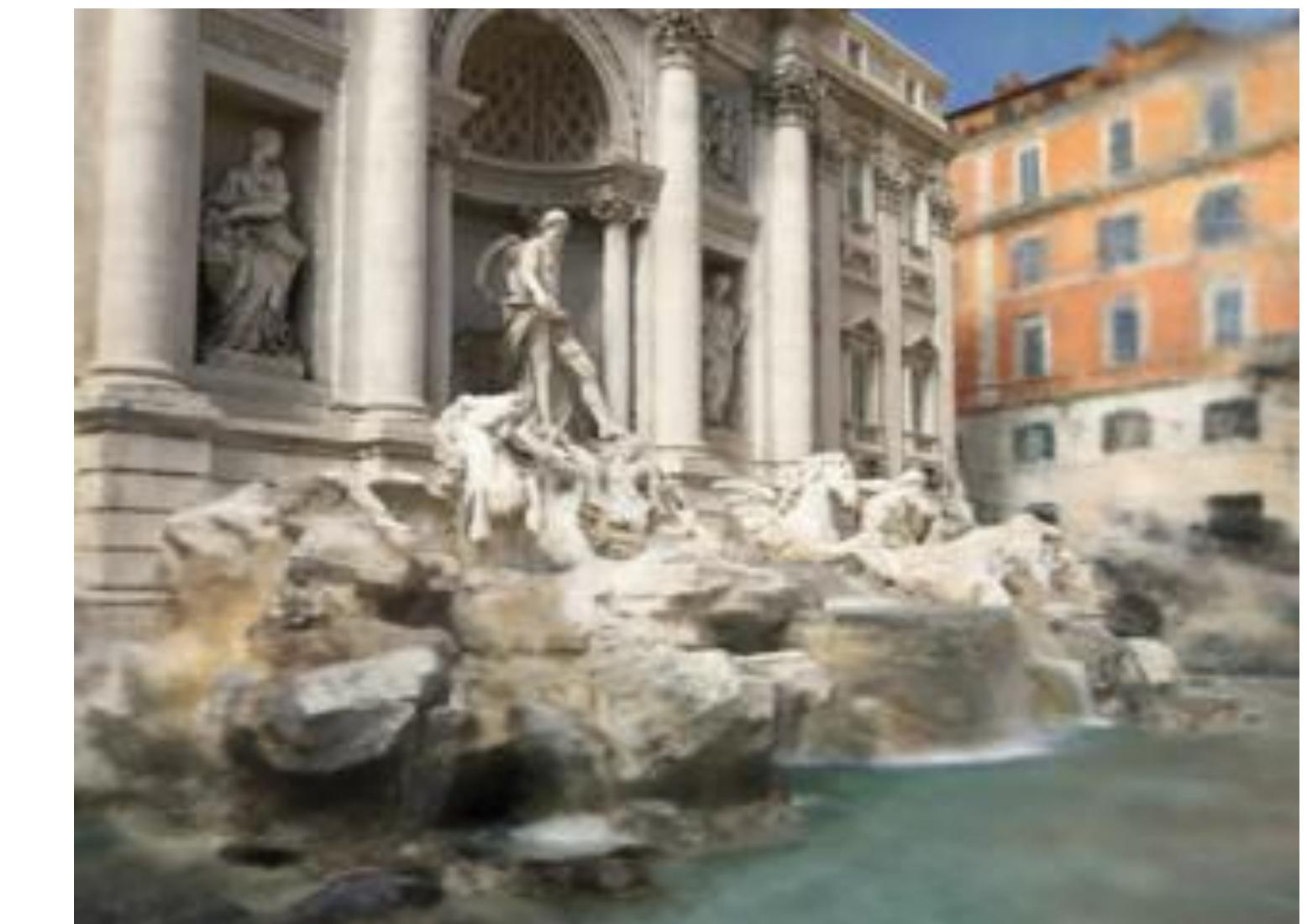
Extension: NeRF in the Wild (NeRF-W)



Brandenburg Gate



Sacre Coeur

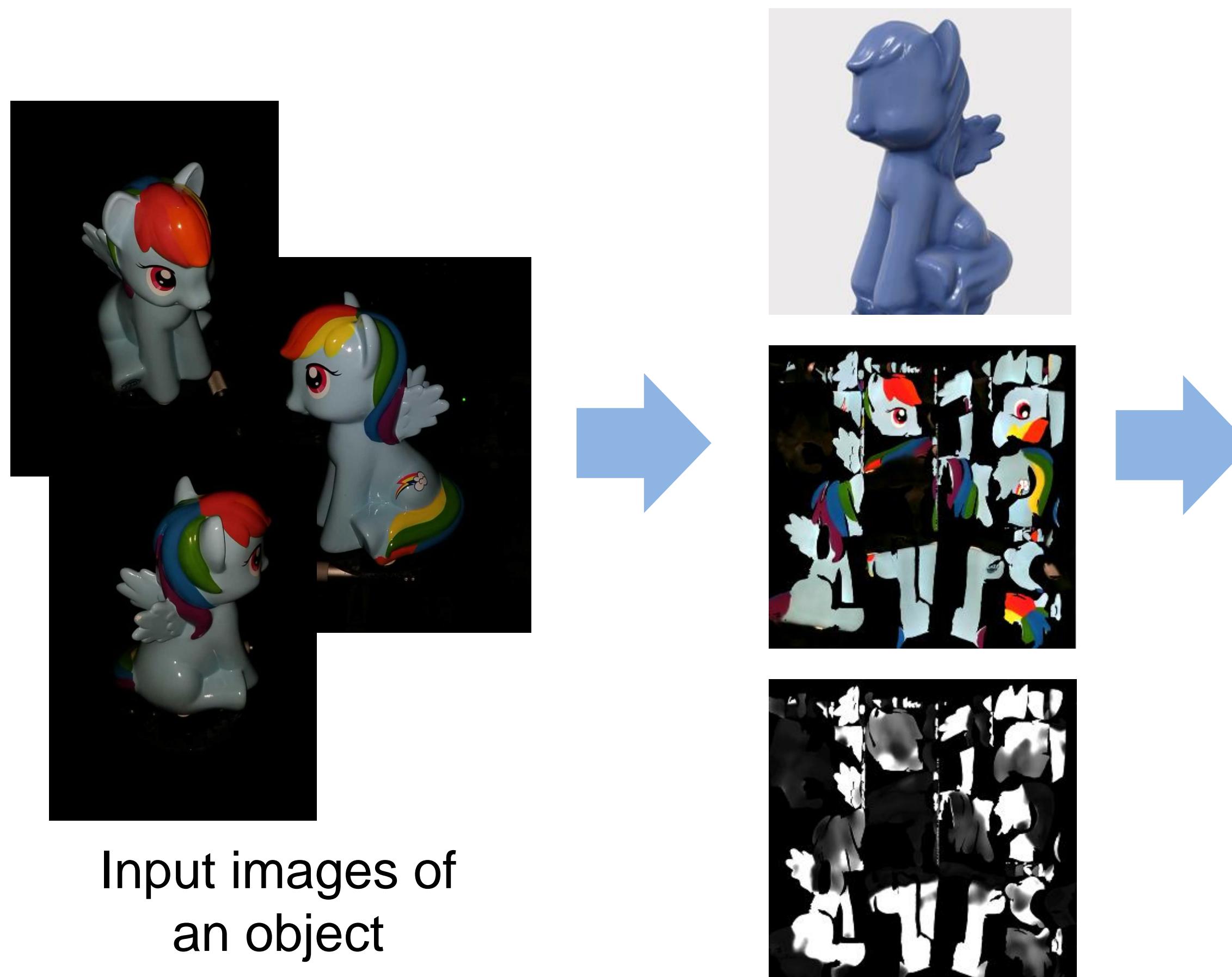


Trevi Fountain

Martin-Brualla*, Radwan*, Sajjadi*, Barron, Dosovitskiy, Duckworth.
NeRF in the Wild. CVPR 2021.

<https://www.youtube.com/watch?v=mRAKVQj5LRA>

Inverse graphics beyond shape and color



Reconstructed models inserted into scene with new lighting

Questions?