CS5670: Computer Vision

Neural Rendering & Neural Radiance Fields (NeRFs)

Many slides based on material from Pratul Srinivasan

Announcements

- Project 5 released today, due Wednesday, May 4, 2022 (8:00 pm)
  - To be done in groups of 2

- Sample final exam online – see Ed Stem

- Final exam in-class on May 10

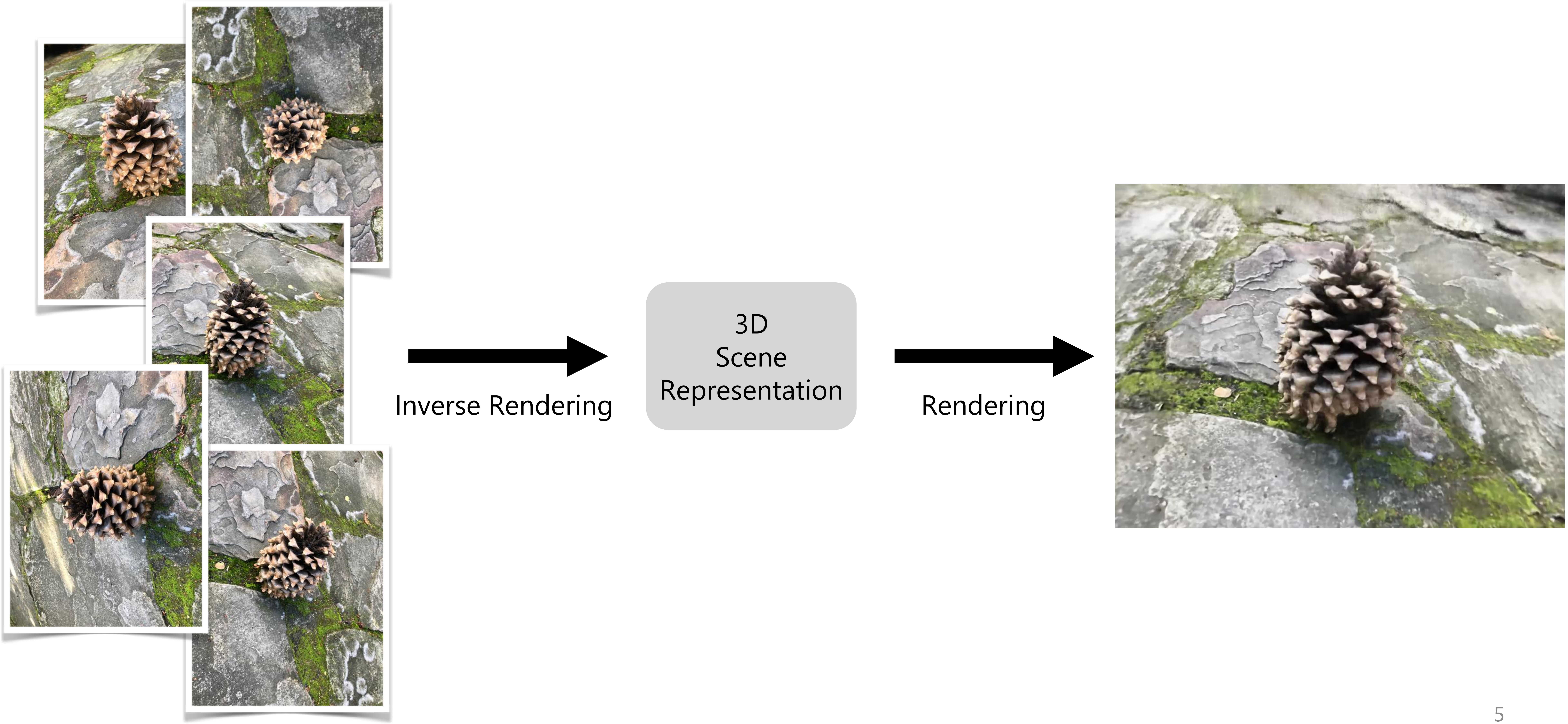
Project 5 Demo

Computer vision as inverse rendering

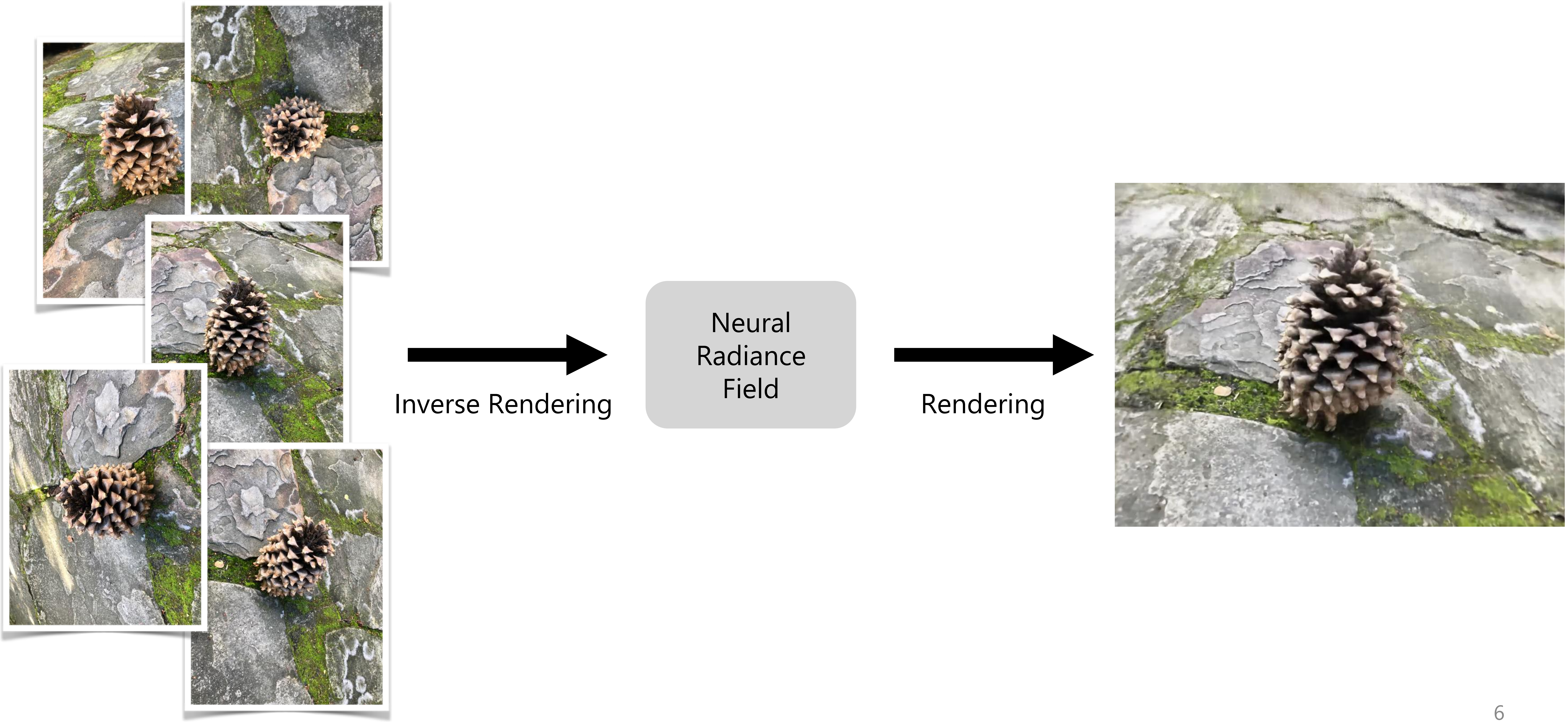
3D
Scene
Representation

Rendering

Computer vision as inverse rendering



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



Deep learning for 3D reconstruction

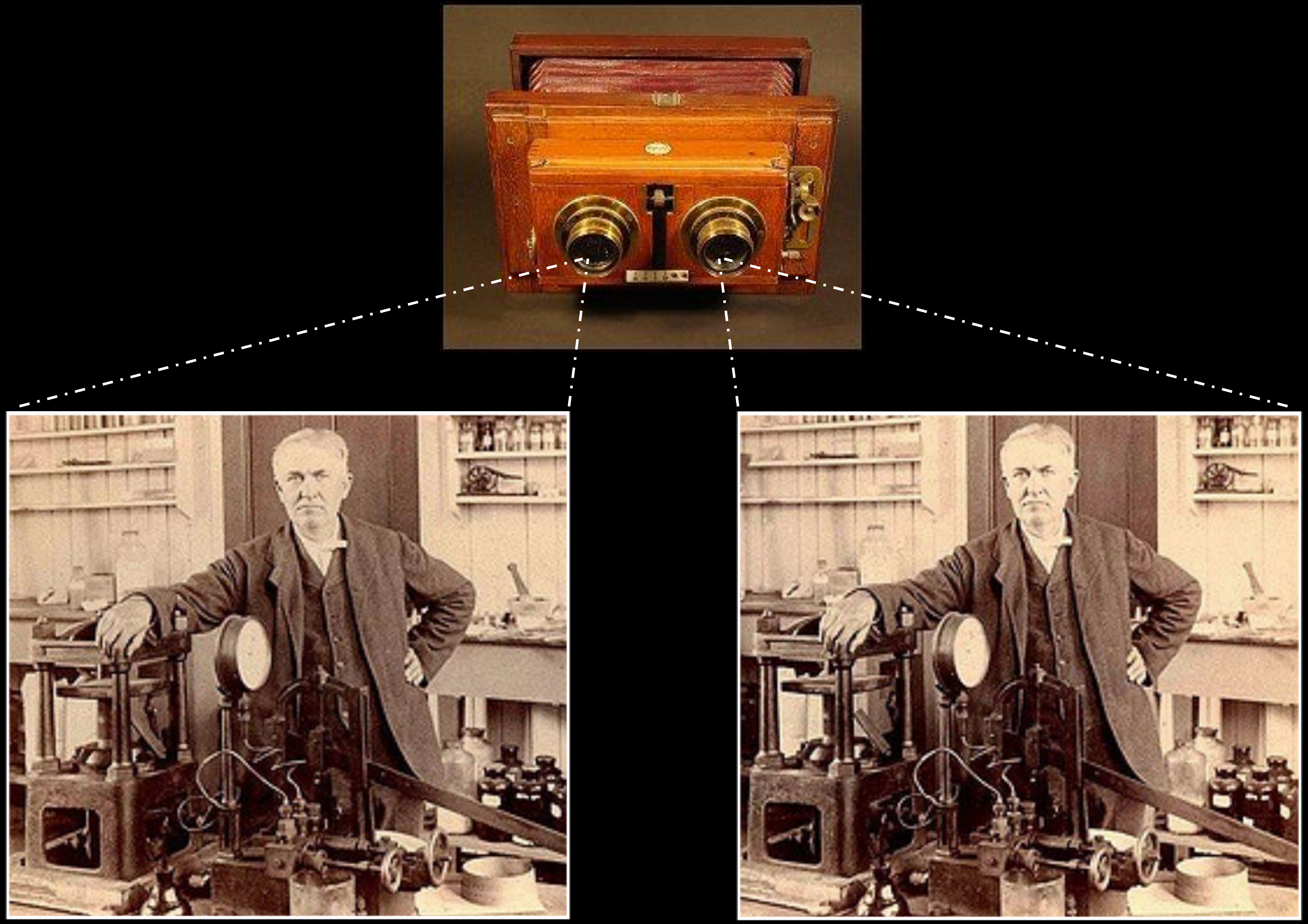
- Previously: we reconstruct geometry by running stereo or multi-view stereo on a set of images
  - "Classical" approach

- How can we leverage powerful tools of deep learning?
  - Deep neural networks
  - GPU-accelerated stochastic gradient descent

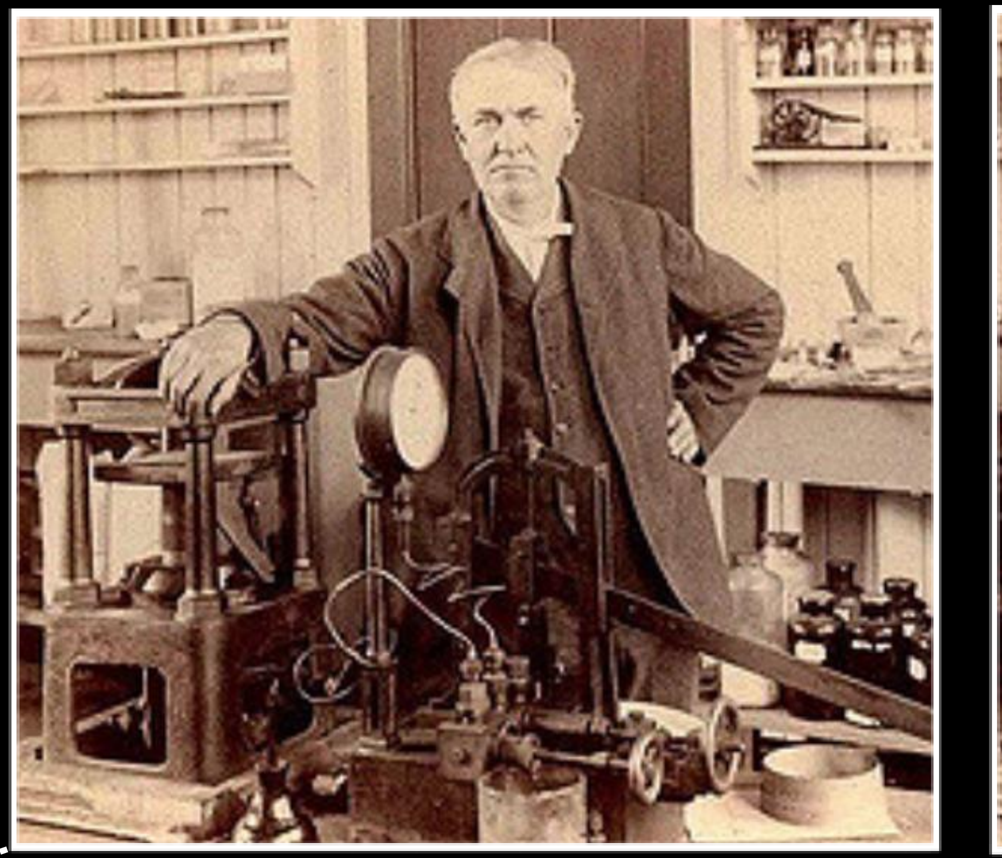
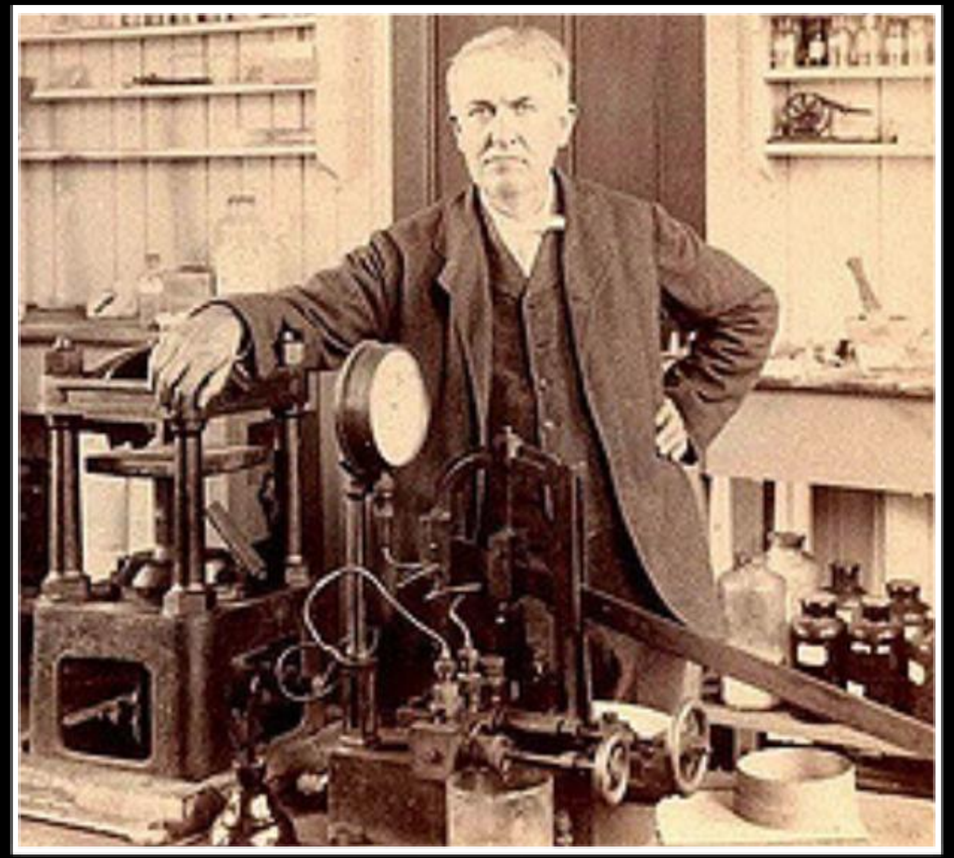
NeRF and related methods – Key ideas

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

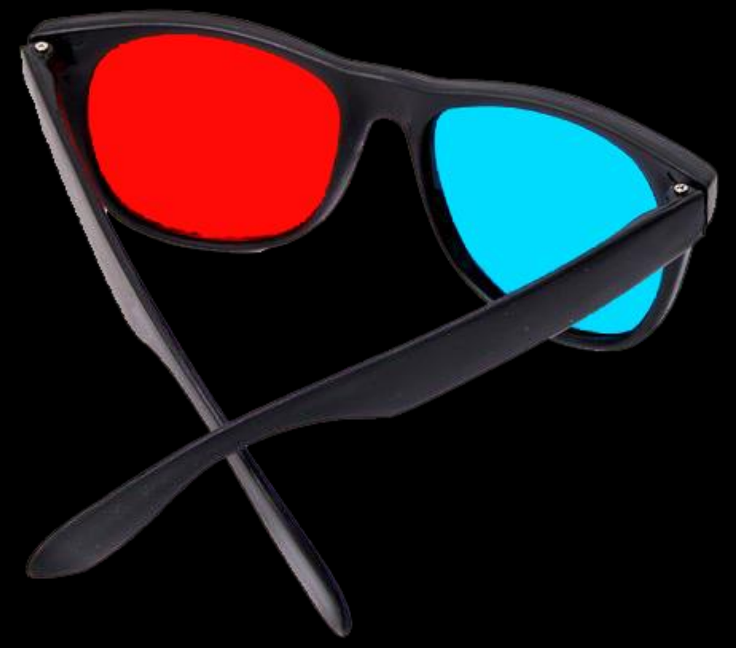
Side Topic: Stereo Photography



Stereo Photography

Viewing Devices

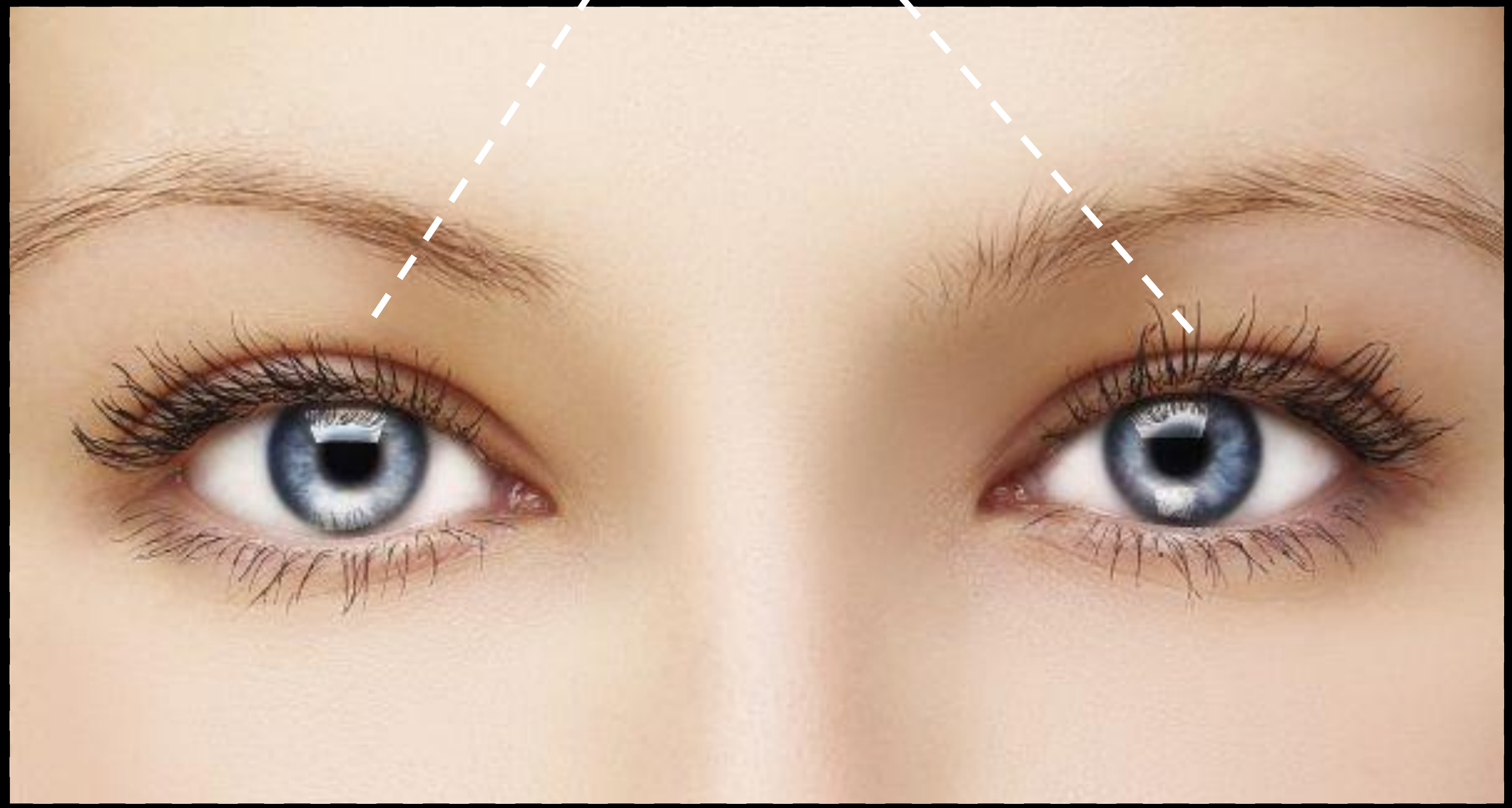

Stereo Photography

Queen Victoria at World Fair, 1851

Stereo Photography

Issue: Narrow Baseline

~6.5 cm



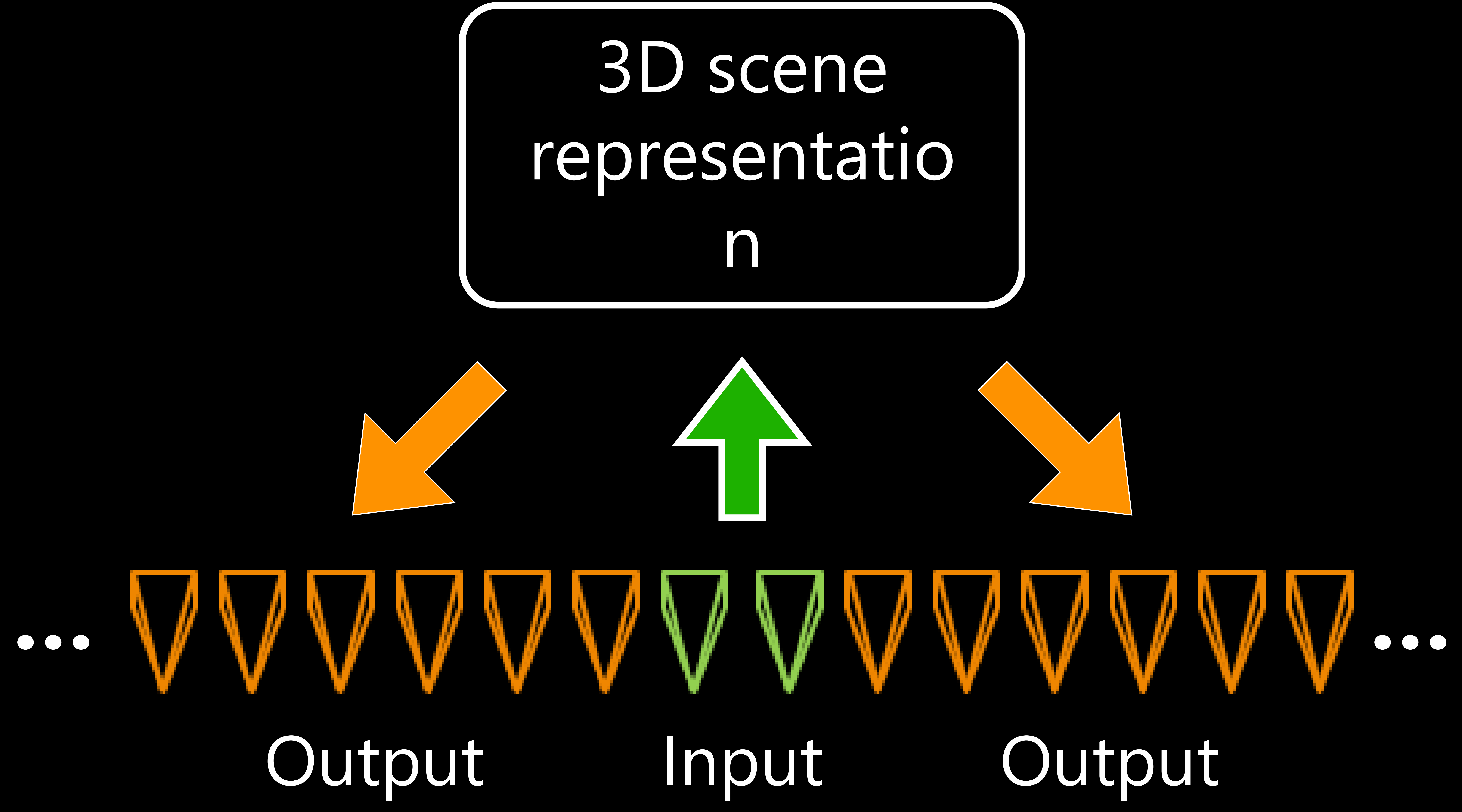
~1.5 cm


Left
MAZZER


Right
MAZZER


MAZZER

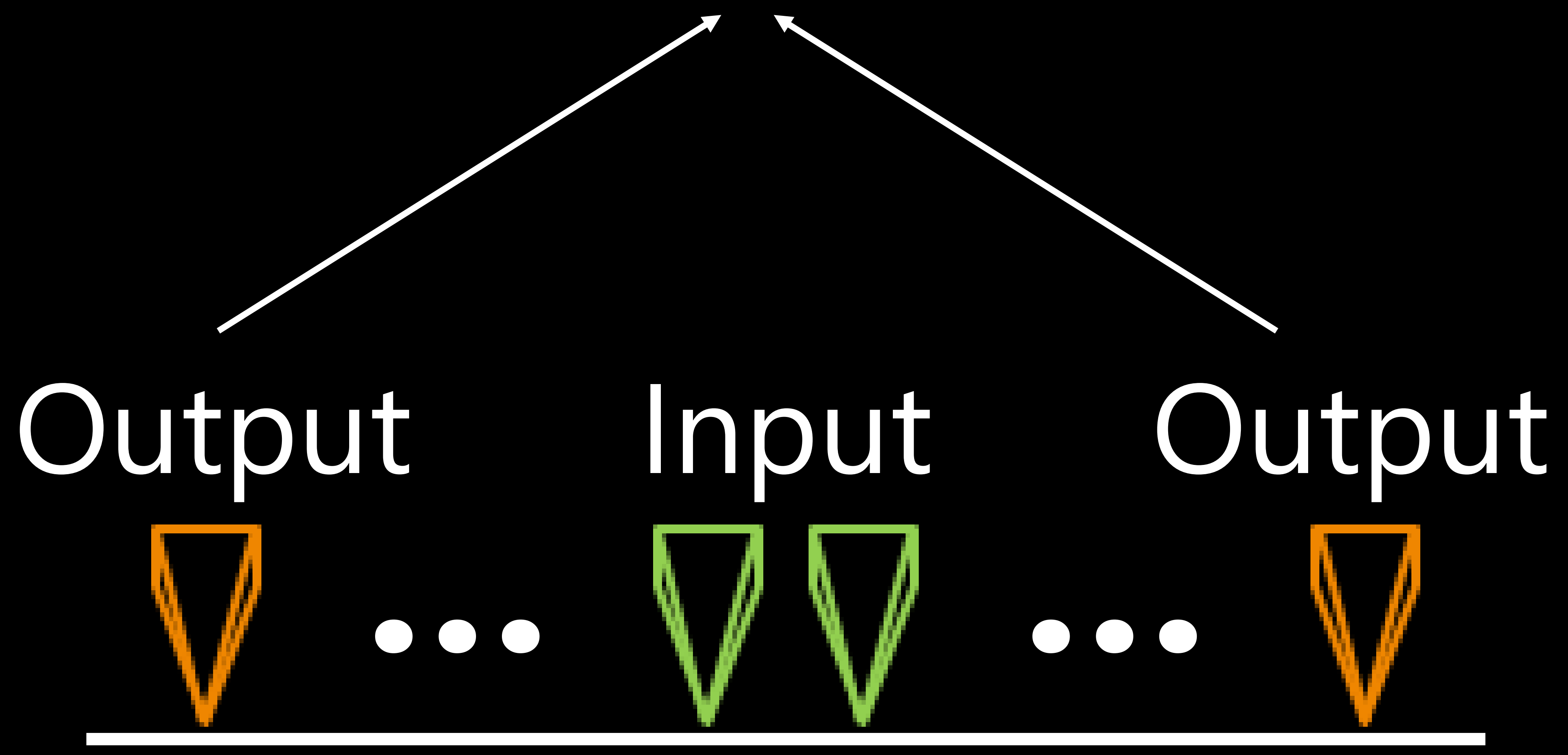
Problem Statement



Challenges

Extrapolation

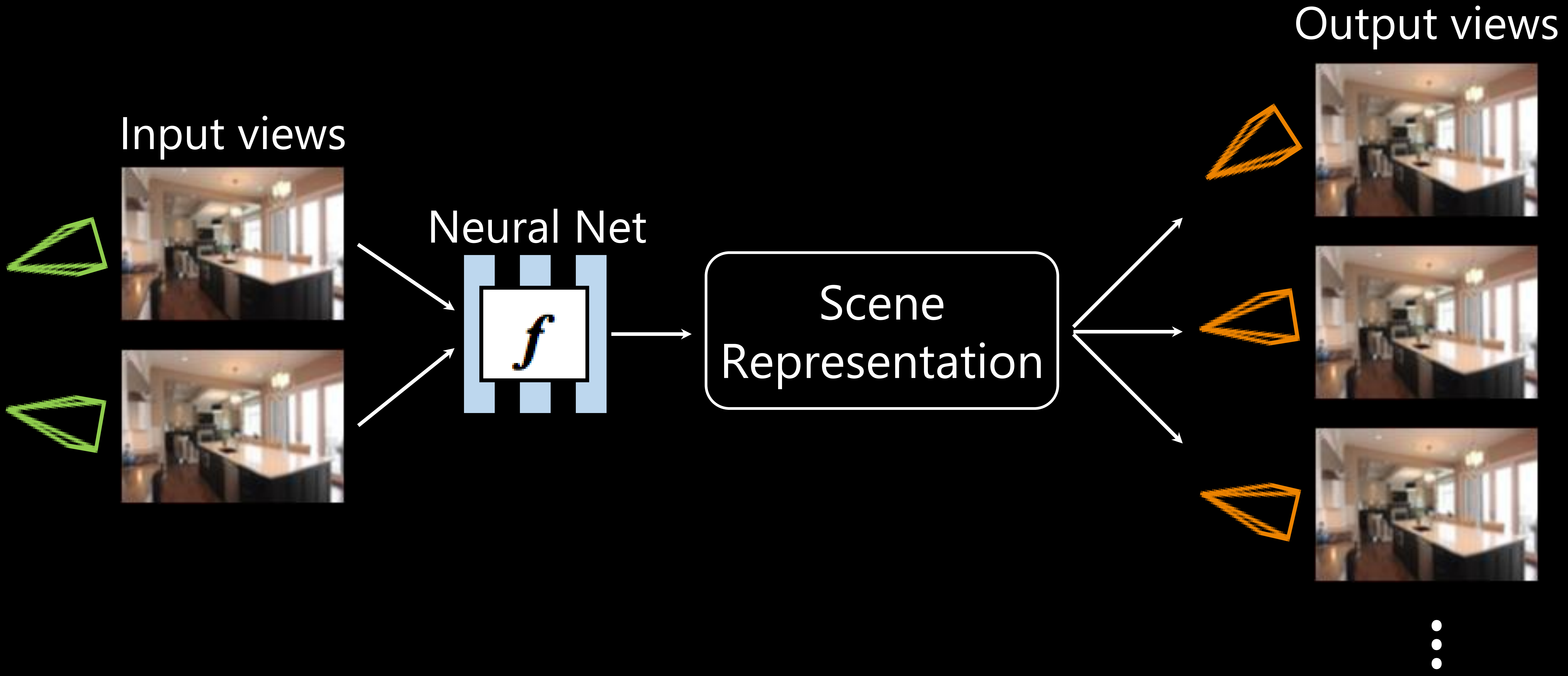
Large disocclusion



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)

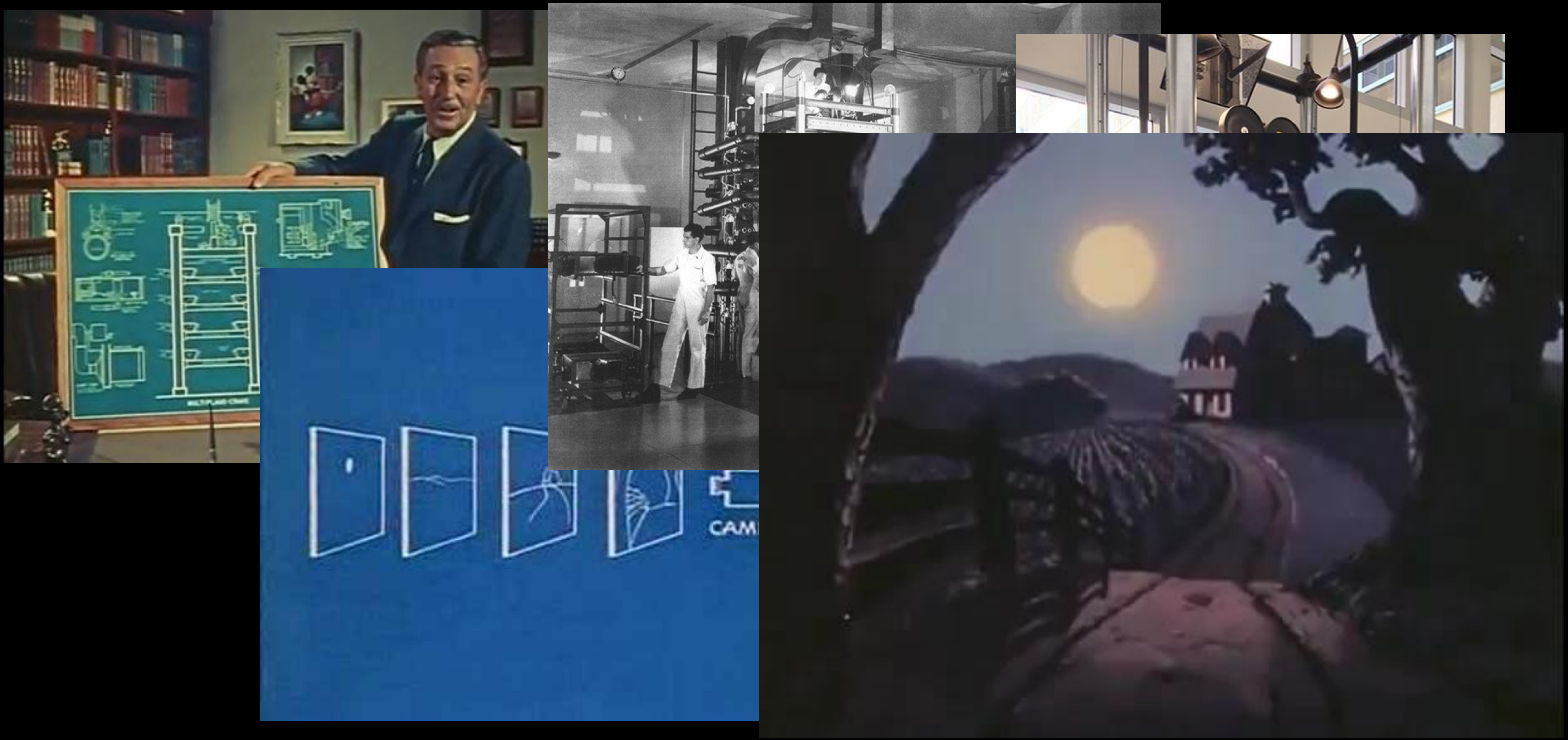
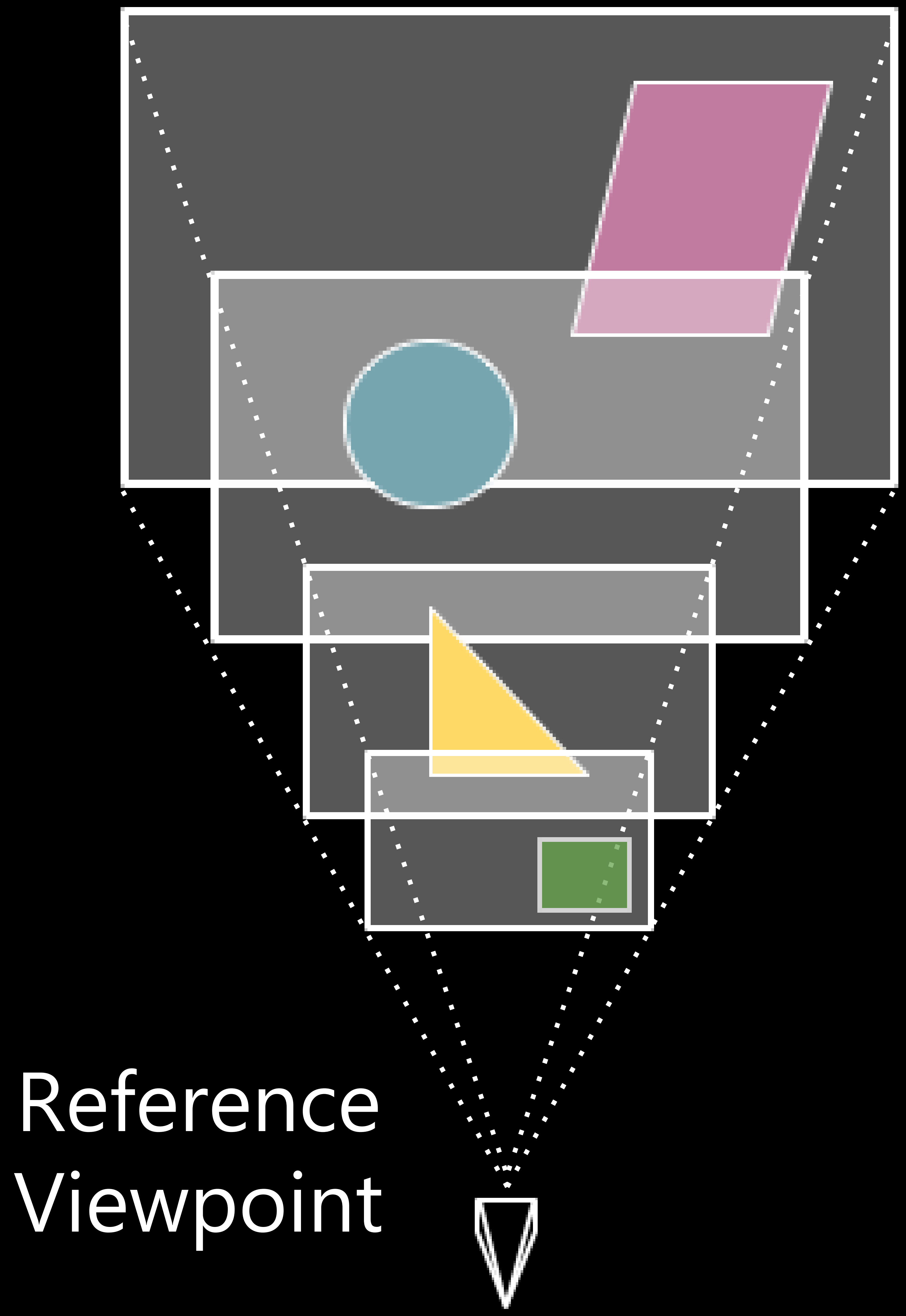


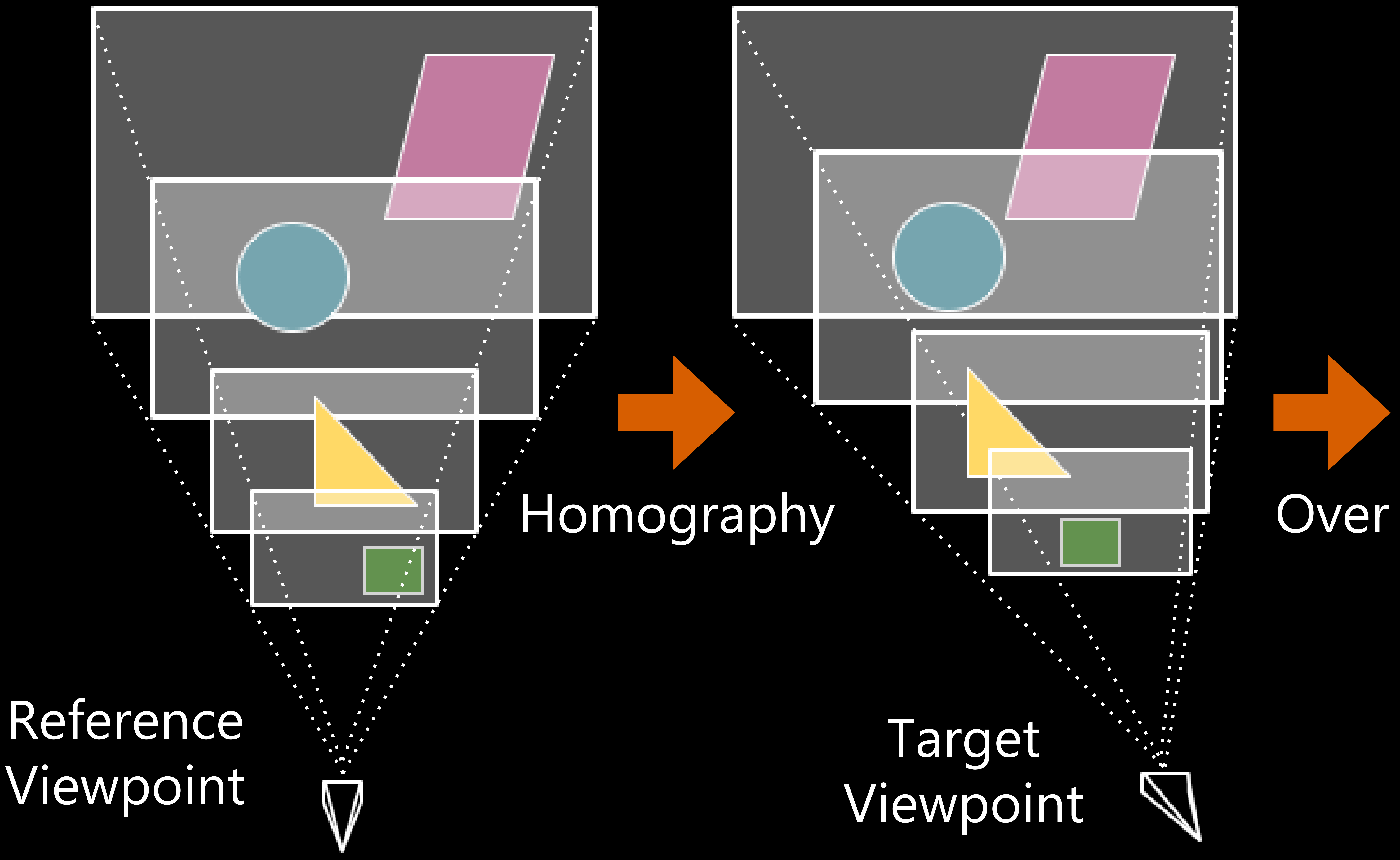
Image credits: Disney https://www.youtube.com/watch?v=kN-eCBAOw60 (from 1957)

Multiplane Images (MPIs)

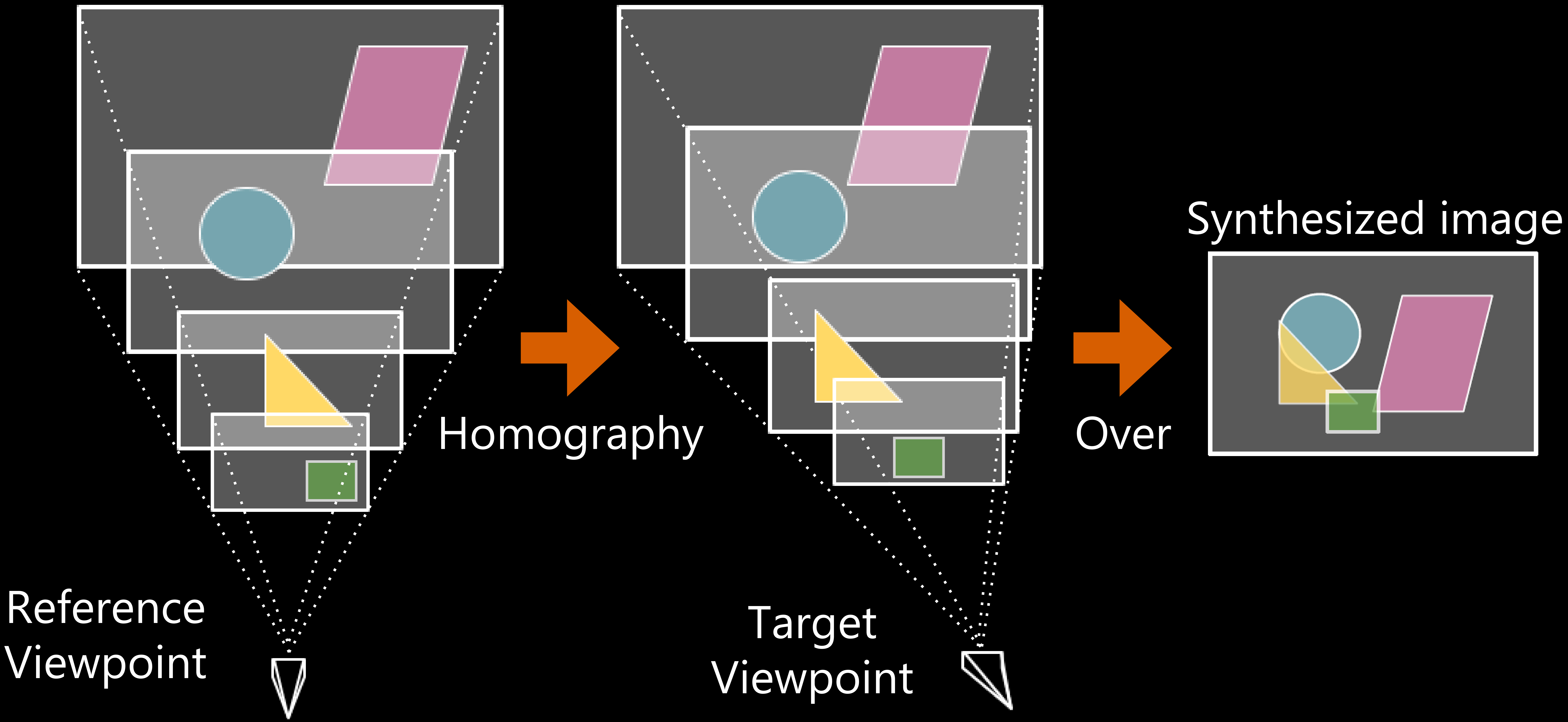


Each plane is at a fixed depth and encoded by an RGBA image

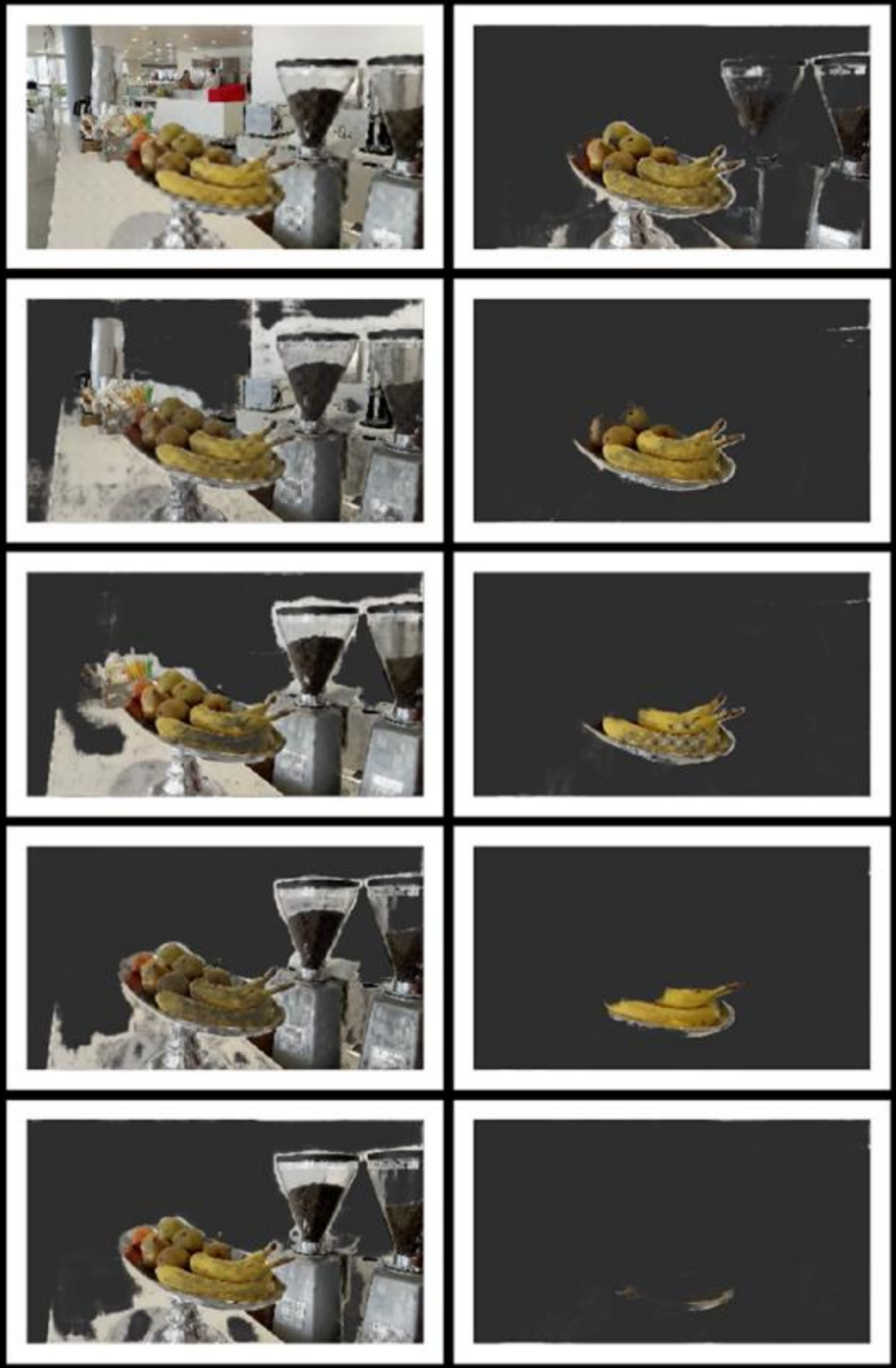
View Synthesis using Multiplane Images



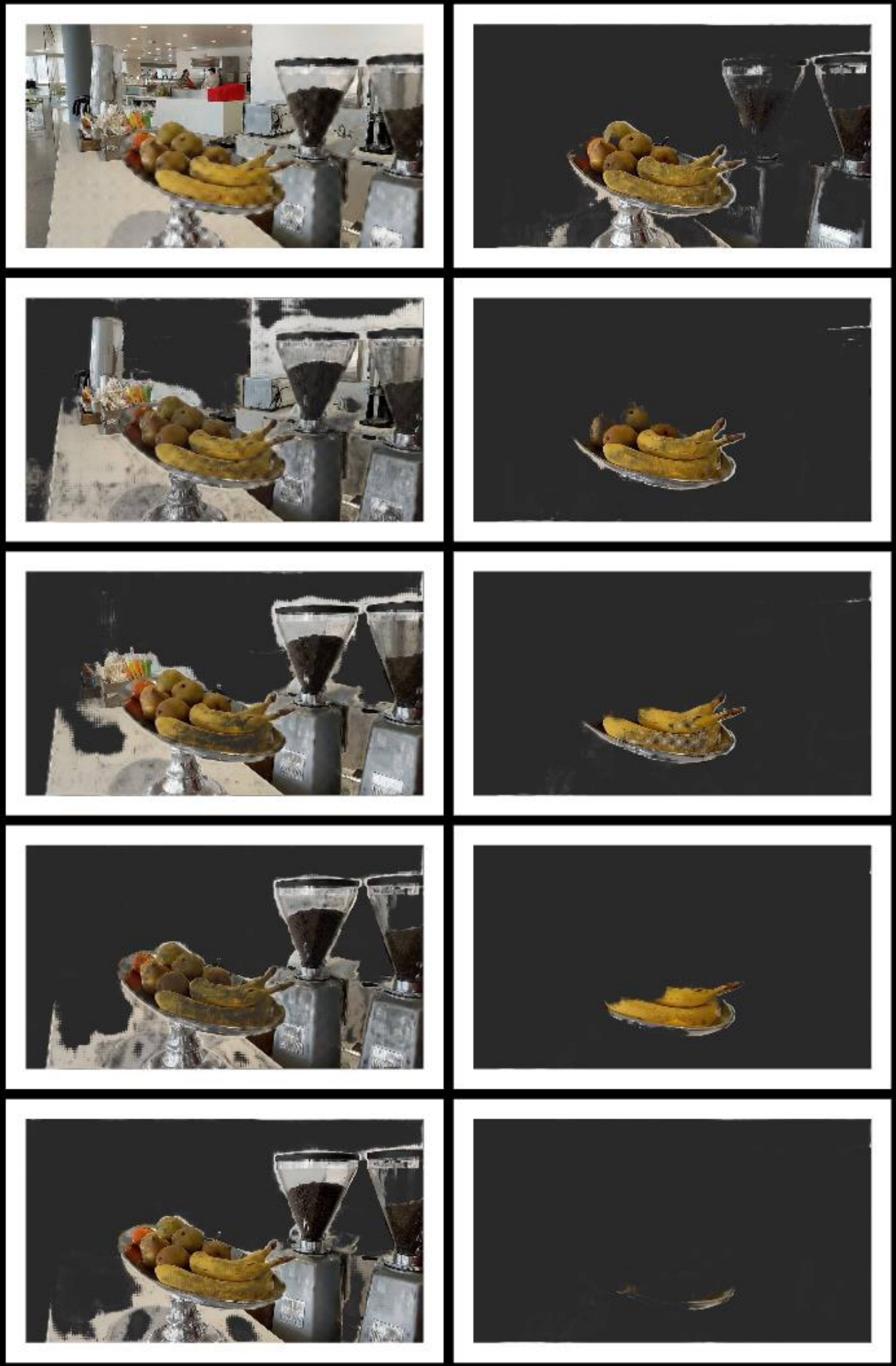
View Synthesis using Multiplane Images



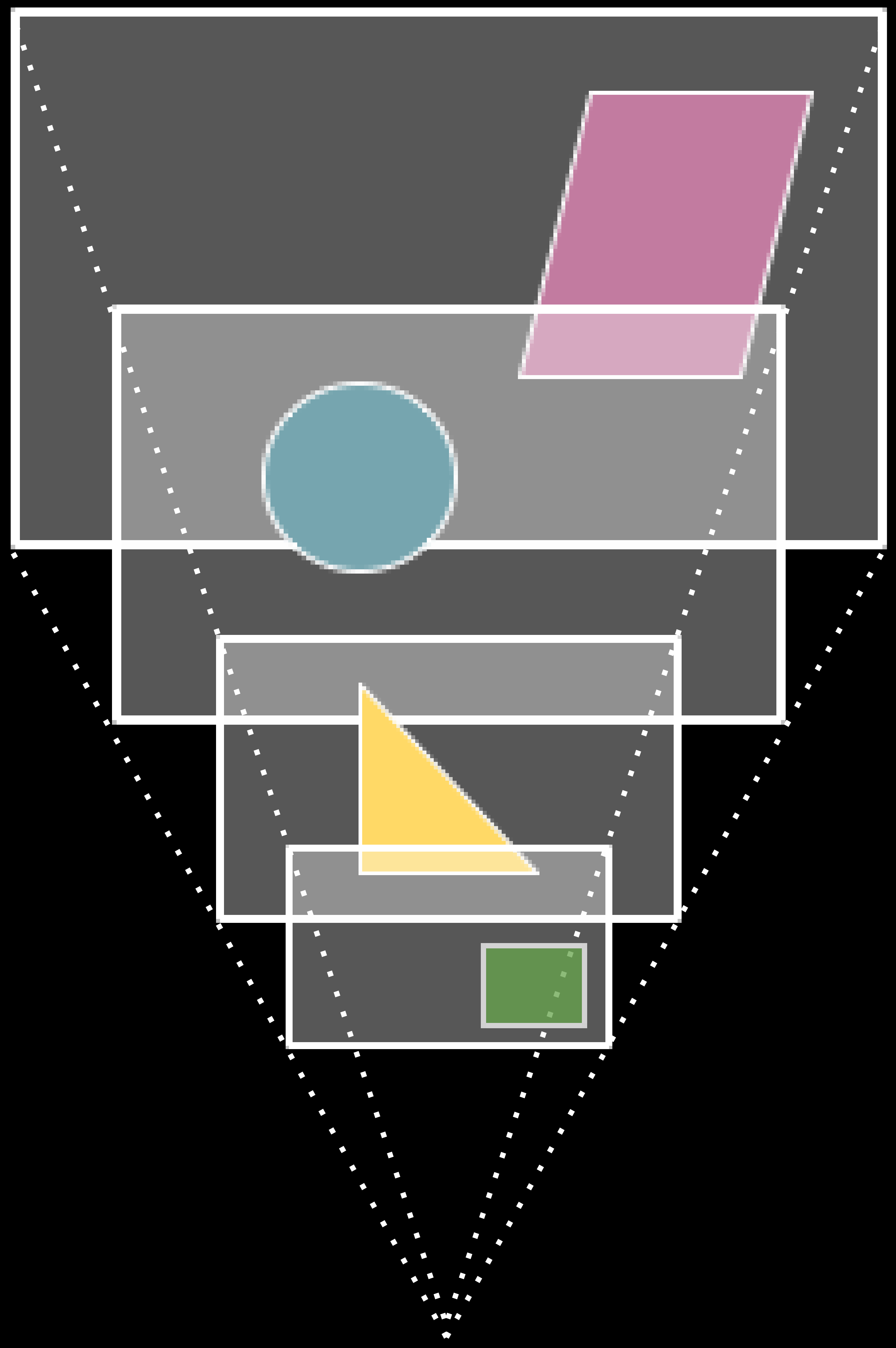

MAZZER





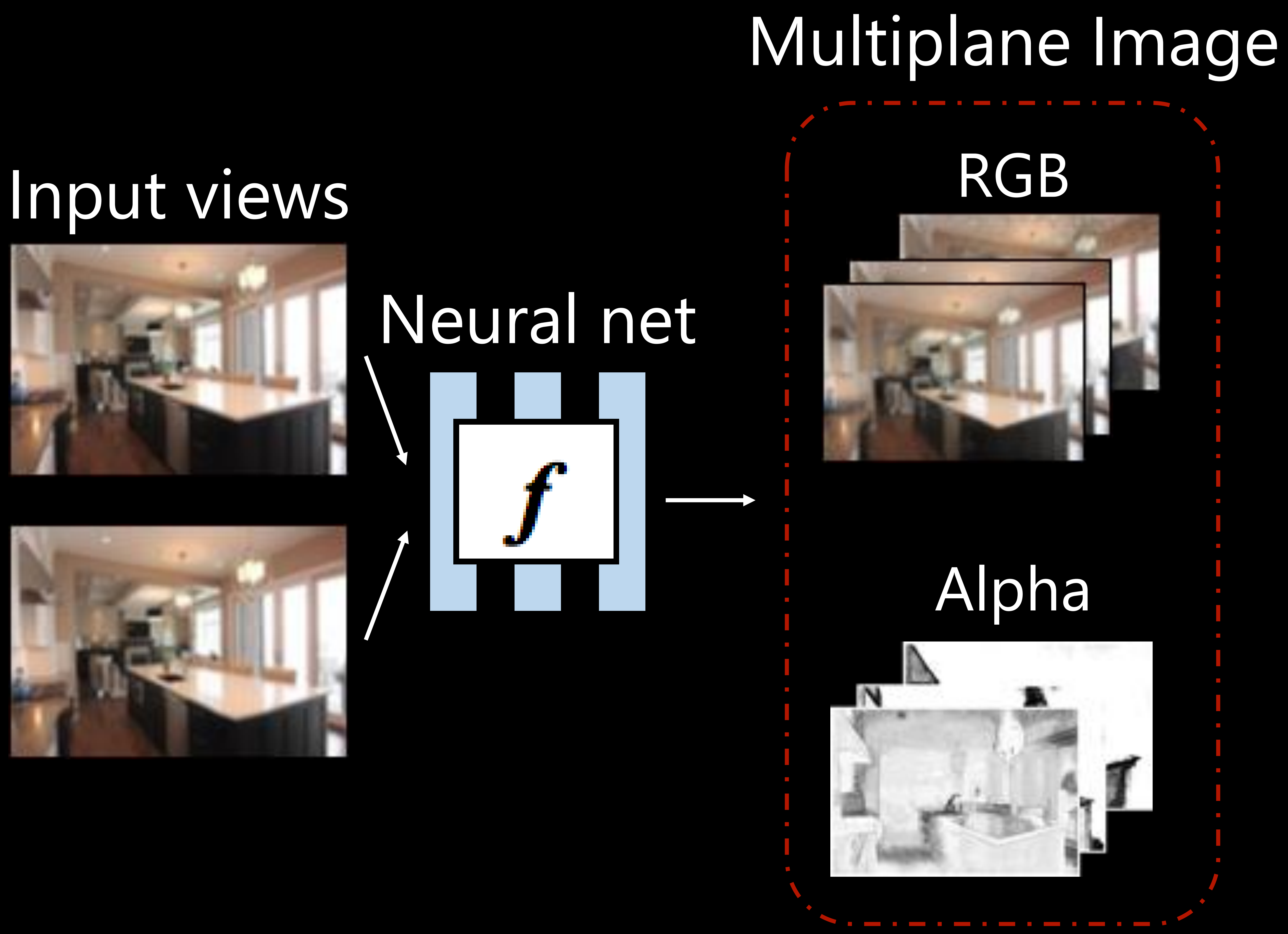


Properties of Multiplane Images

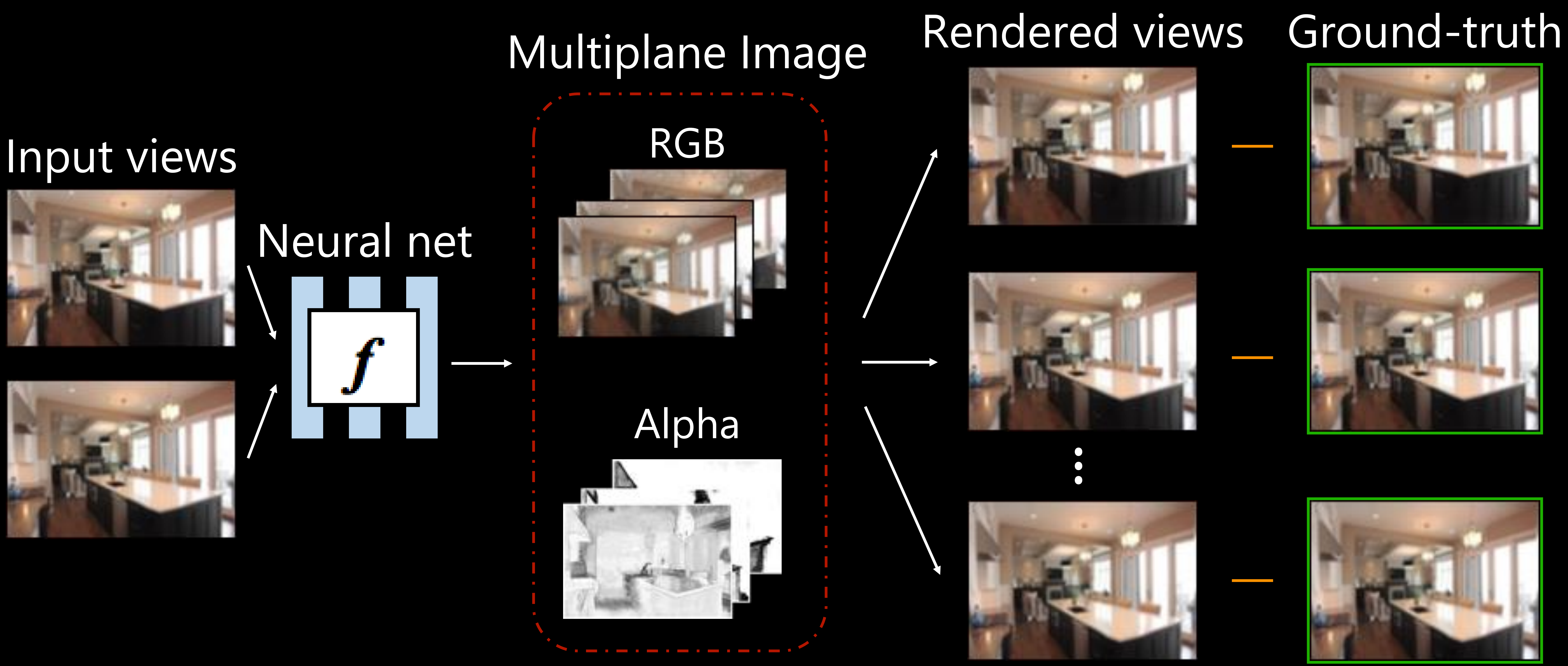


- Models disocclusion
- Models soft edges and non-Lambertian effects
- Efficient for view synthesis
- Differentiable rendering

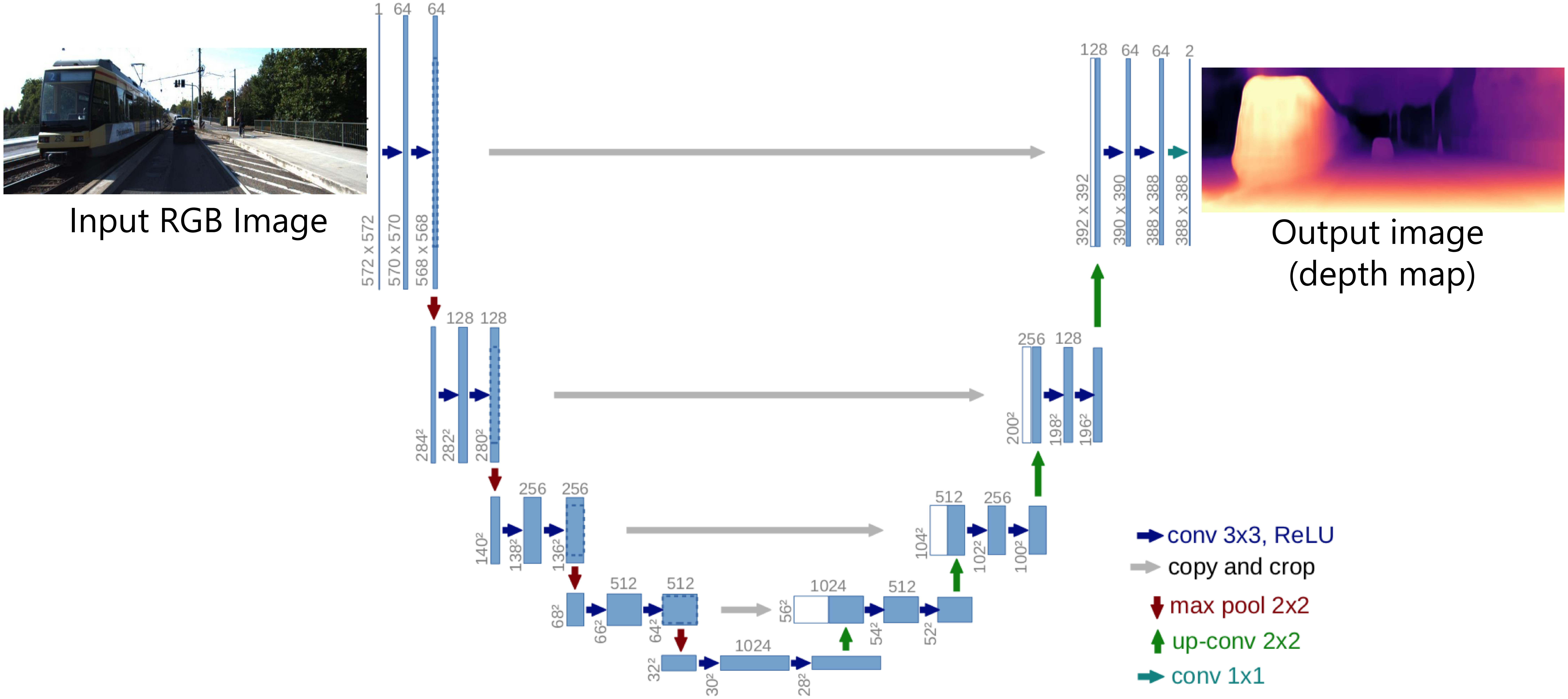
Learning Multiplane Images



Learning Multiplane Images

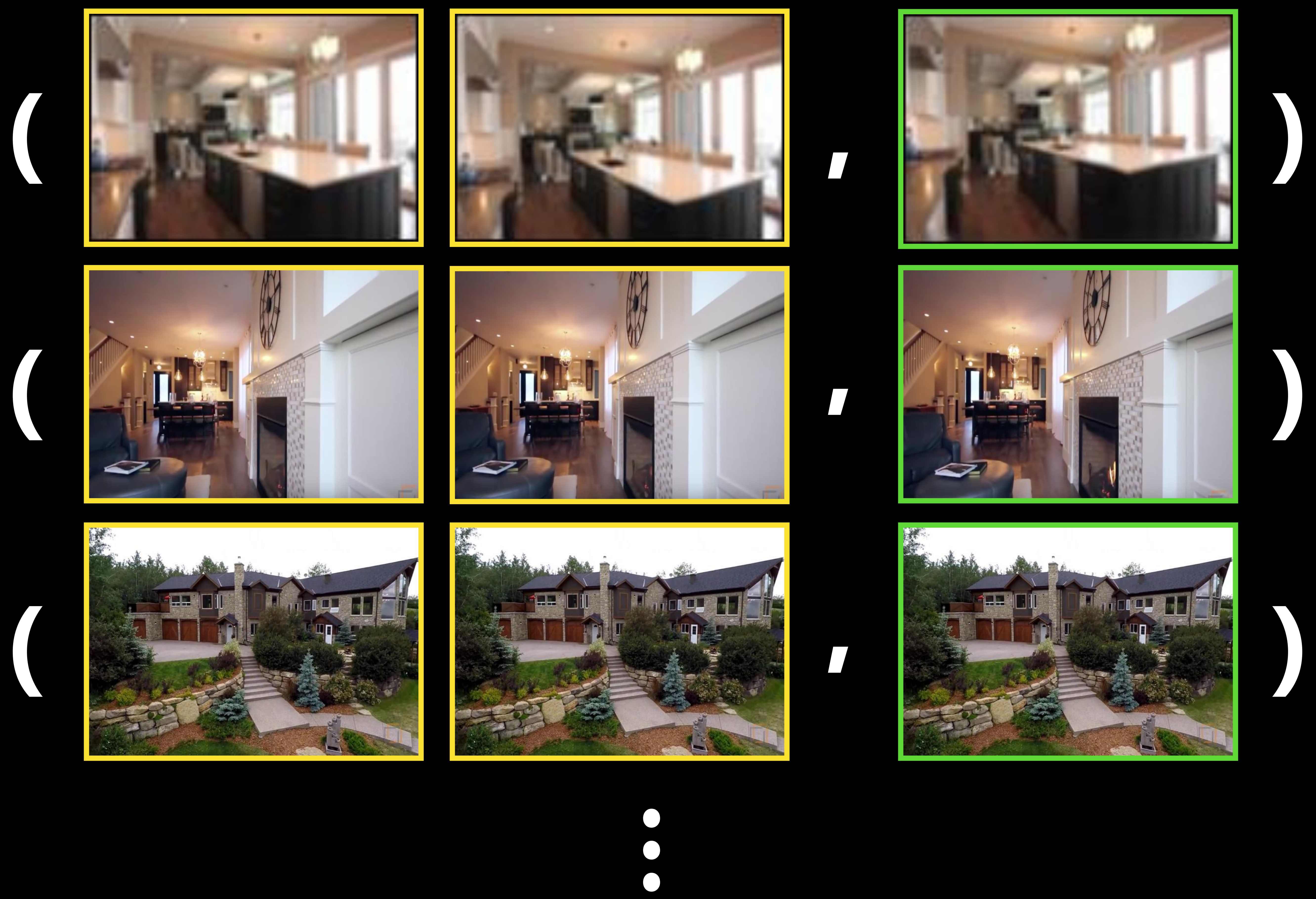


Common architecture for mapping images to images: UNet architecture



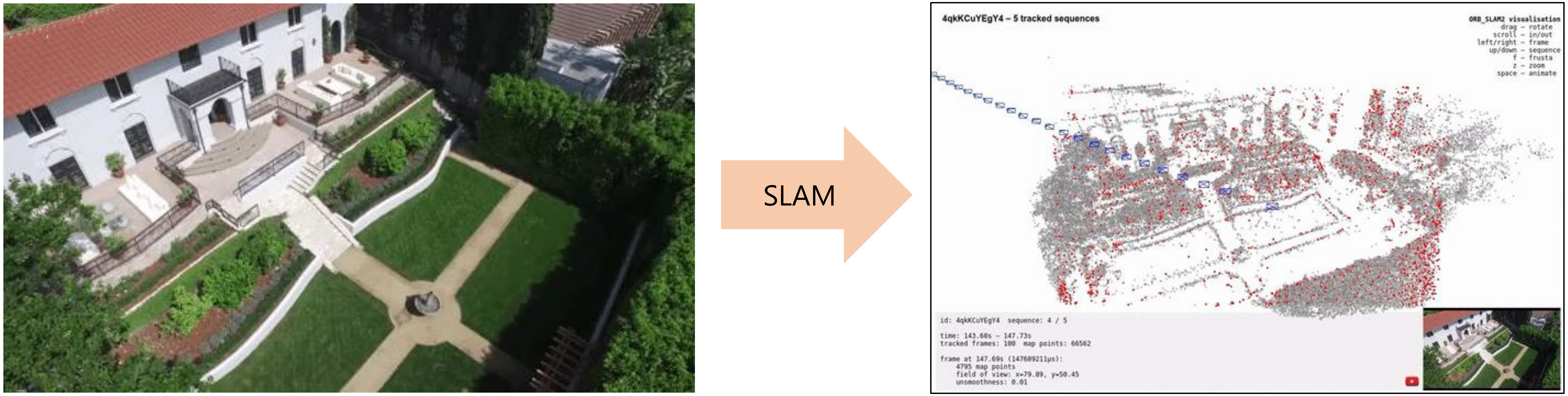
Training Data

Input views | Target view



Need massive set of triplets with known camera poses

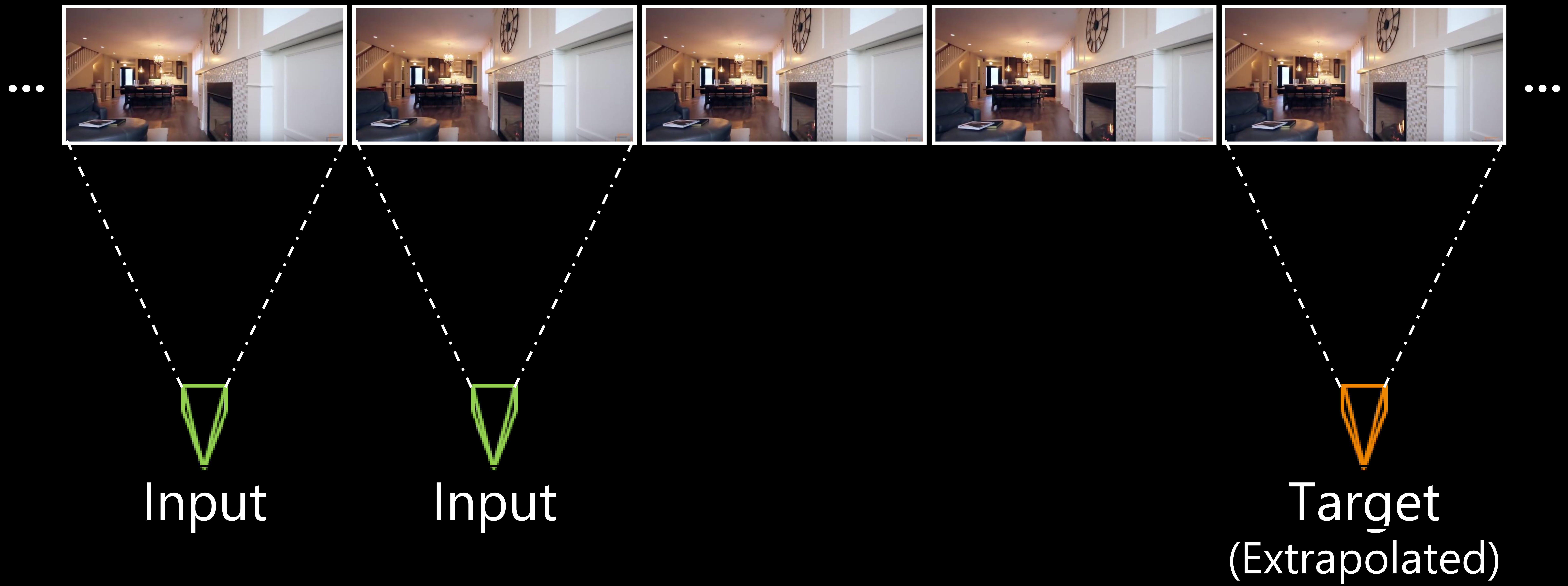
RealEstate10K



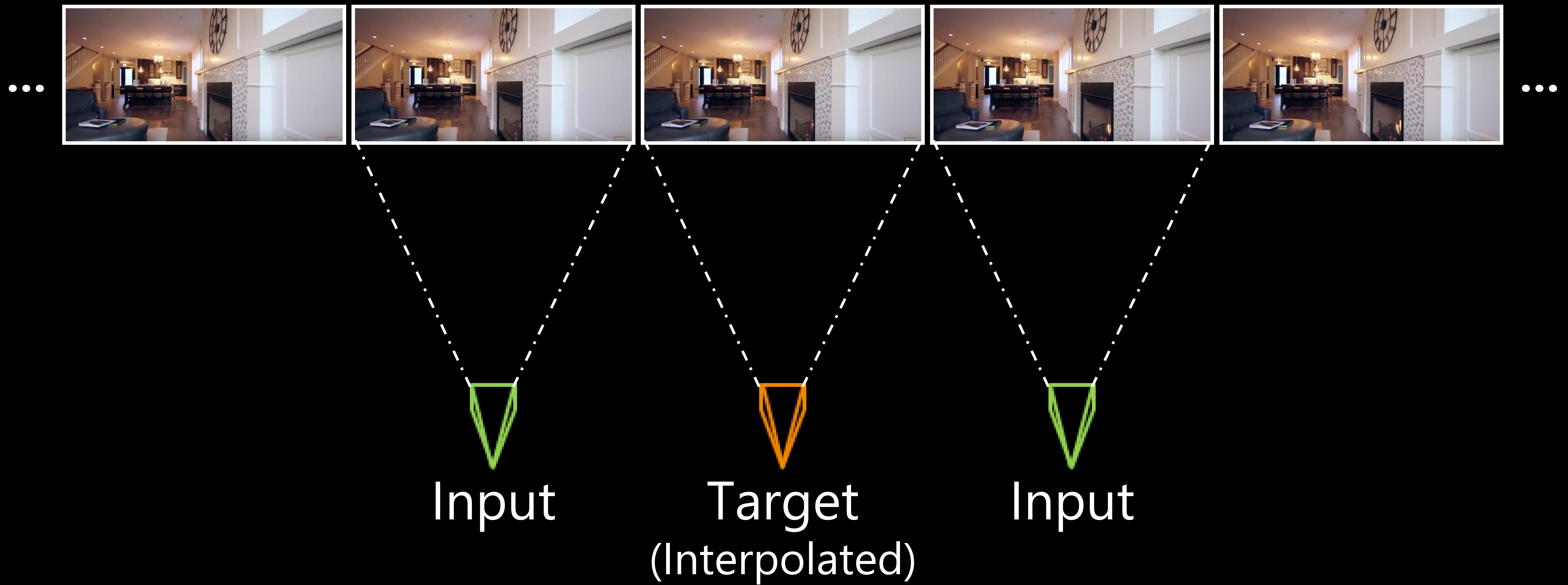
10 million frames from **80,000 video clips** from **10,000 videos**
https://google.github.io/realestate10k/


RealEstate10K dataset
10 million frames from 80,000 video clips from 10,000 videos
https://google.github.io/realestate10k/

Sampling Training Examples



Sampling Training Examples



Results

Left

Right

Output


Image 1


Image 2

Output

Multi-plane Image (MPI)

Reference input view

Plane 0

Plane 9

Plane 13

Plane 16

Plane 24

Plane 26

Extrapolating Cellphone Footage

1.4 cm

6.3 cm


Right input image

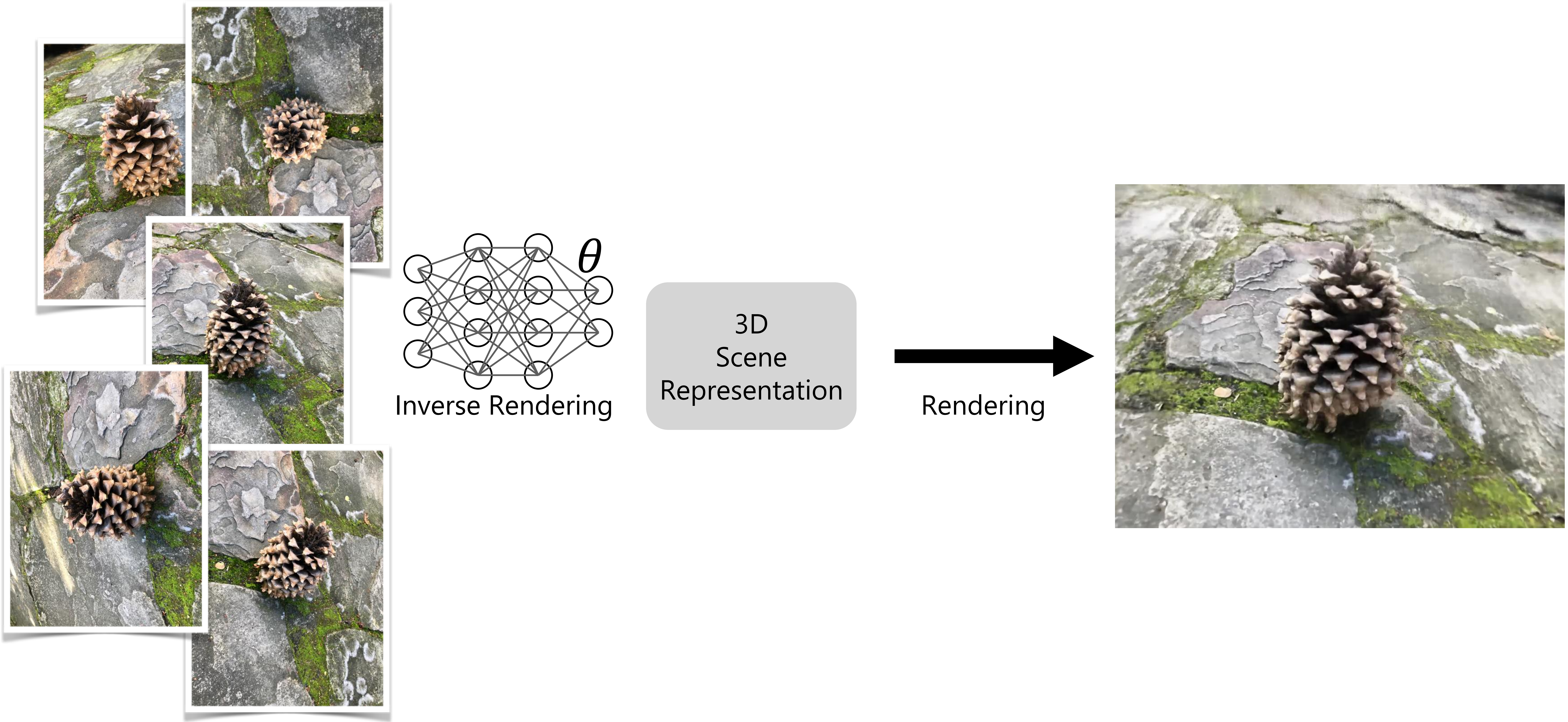
Computer vision as inverse rendering

Inverse Rendering

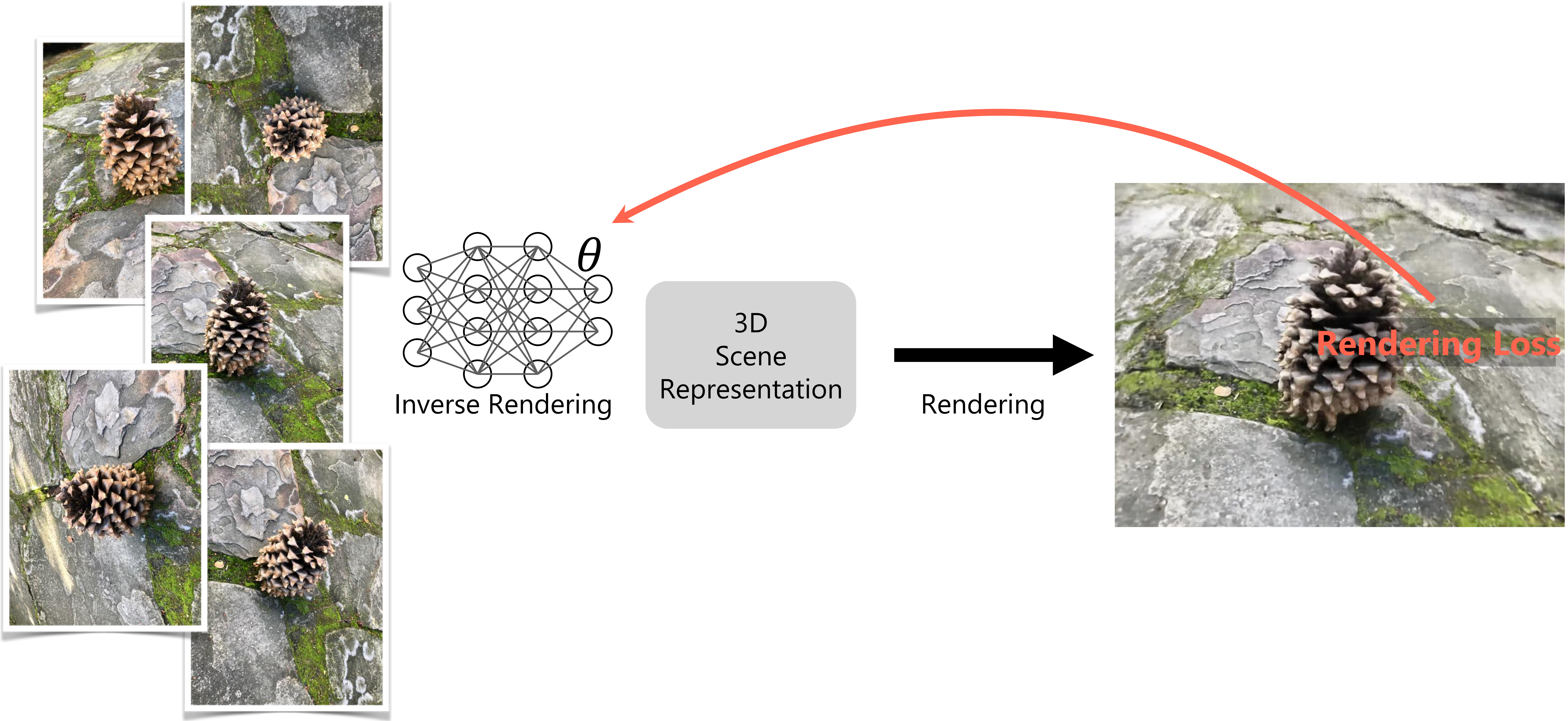
3D Scene Representation

Rendering

Paradigm 1: “Feedforward” inverse rendering



Paradigm 1: “Feedforward” inverse rendering



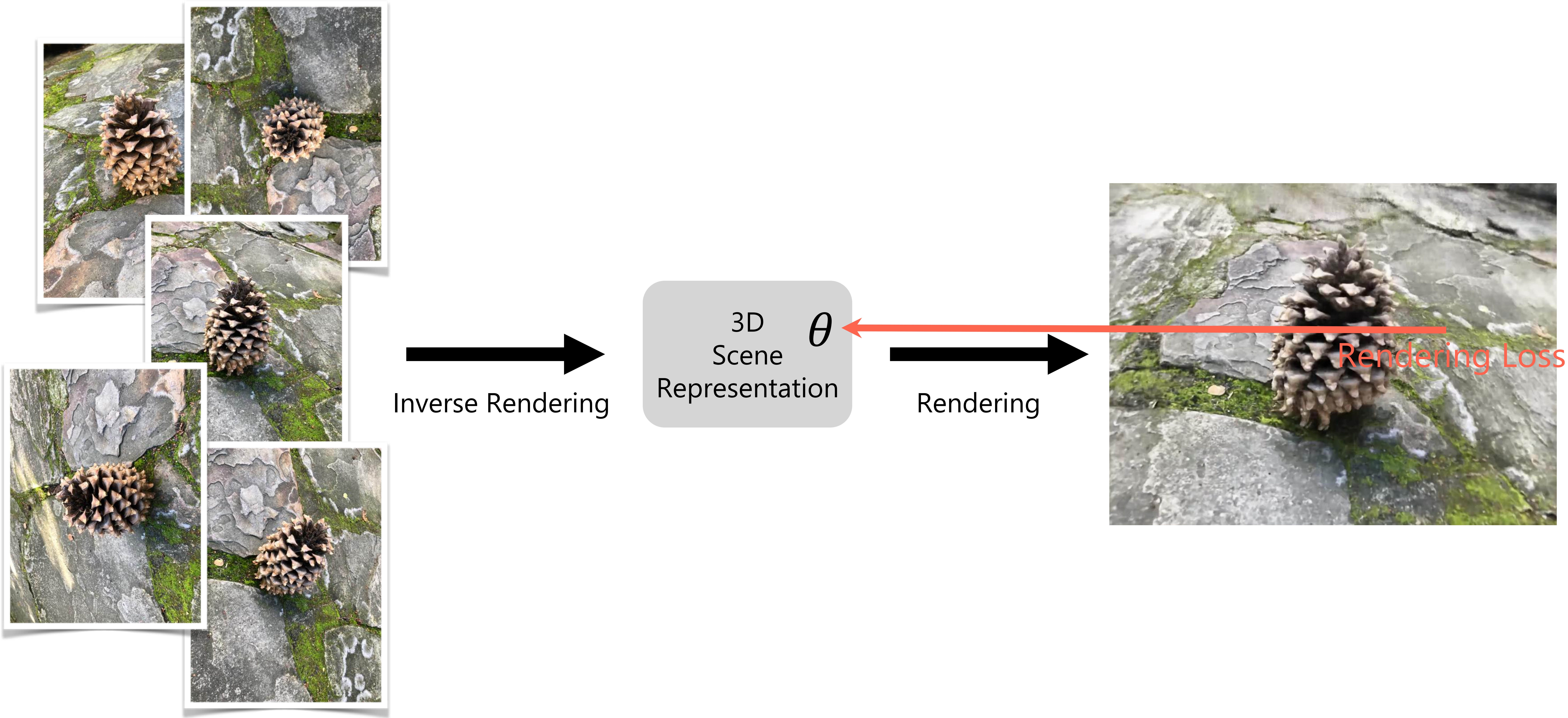
Paradigm 2: “Render-and-compare”

Inverse Rendering

3D Scene Representation θ

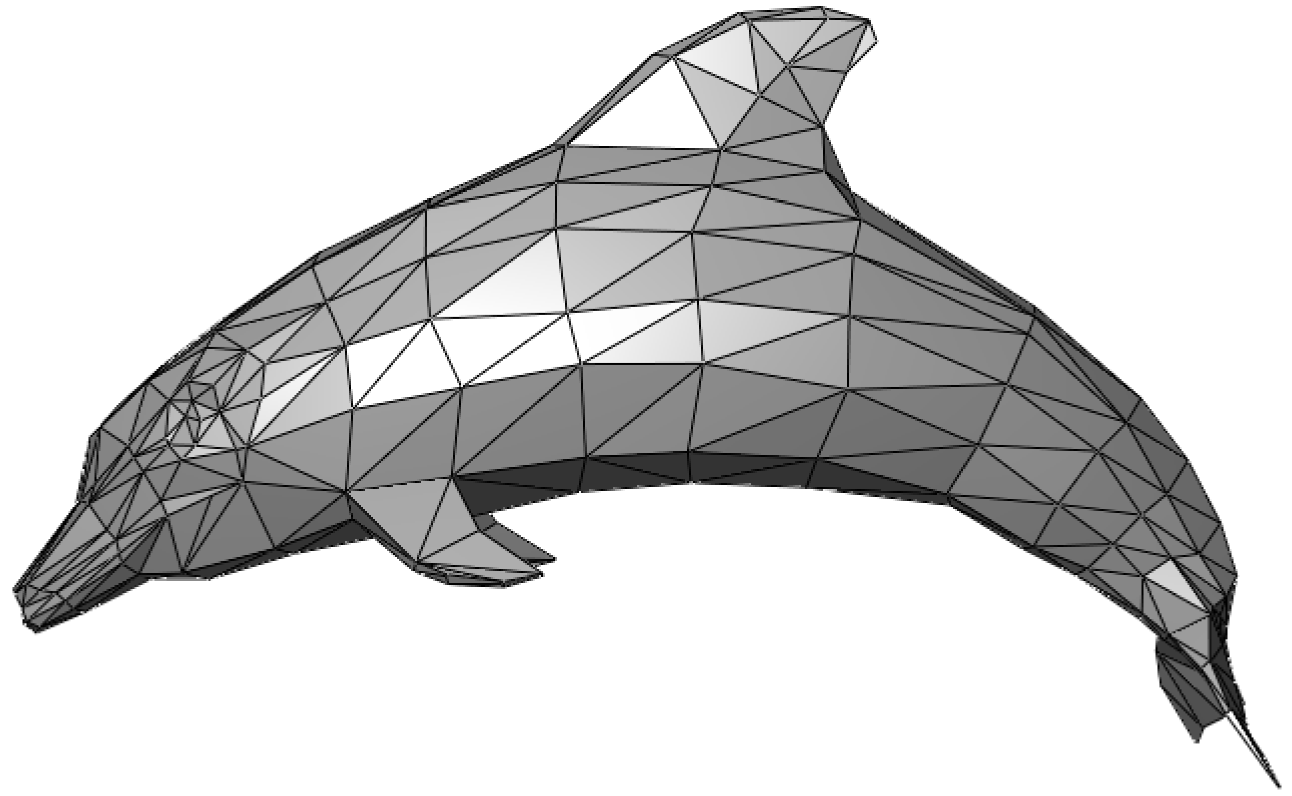
Rendering

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**

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

Neural Volumetric Rendering

Neural Volumetric Rendering

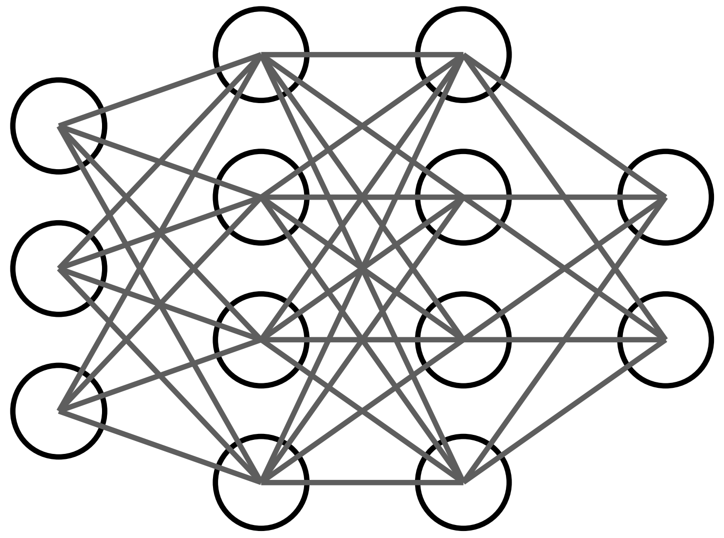
querying the radiance value
along rays through 3D space

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

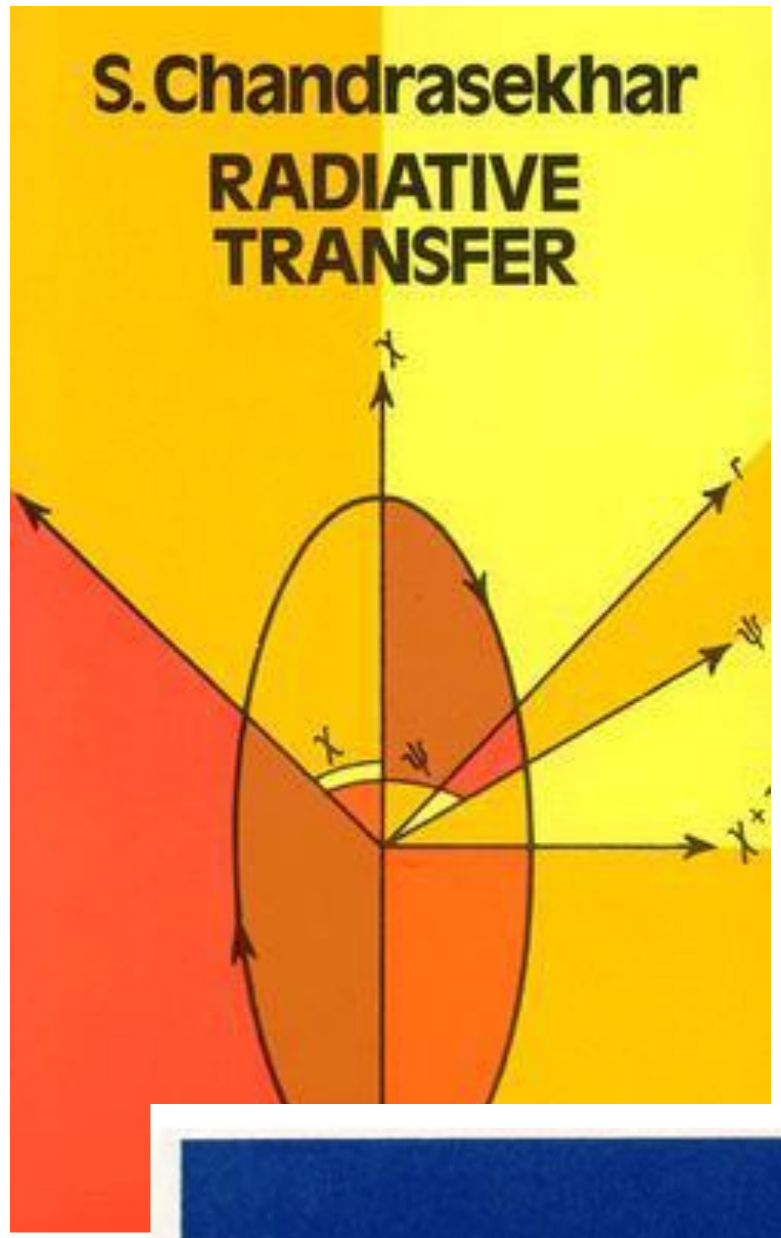
NeRF Overview

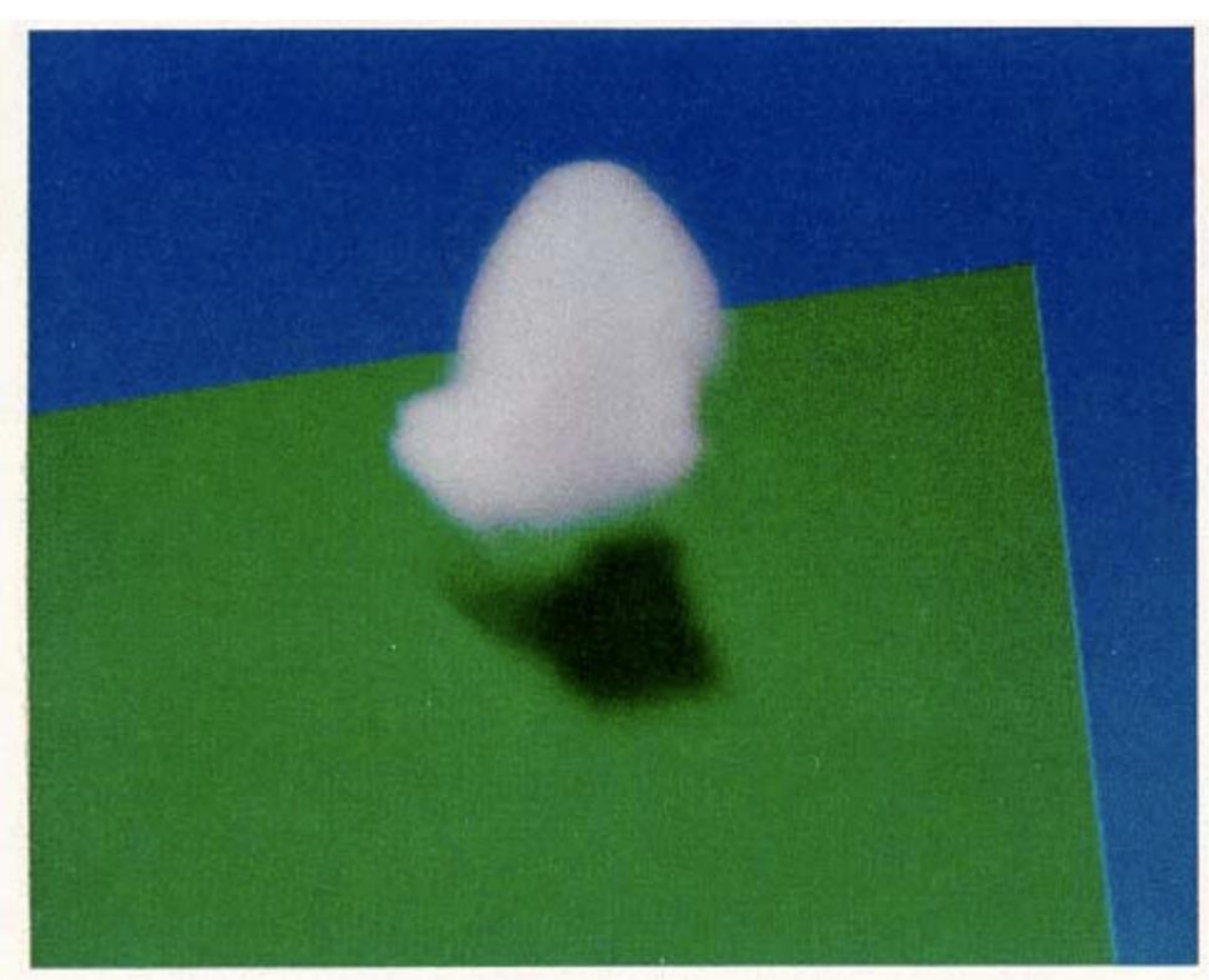
- Volumetric rendering
- Neural networks as representations for spatial data
- Neural Radiance Fields (NeRF)

NeRF Overview

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

Traditional volumetric rendering





Ray tracing simulated cumulus cloud [Kajiya]

- 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

Traditional volumetric rendering

Medical data visualisation [Levoy]

Pt.Reyes = *Foreground* **over** *Hillside* **over** *Background.*

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

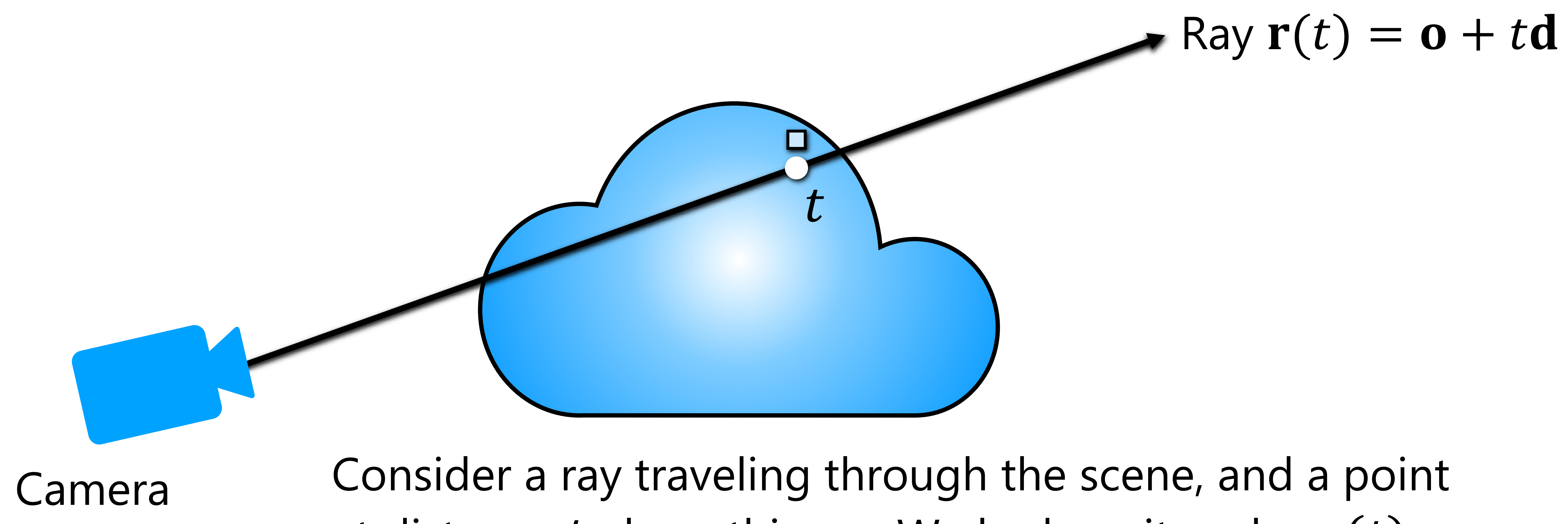
Chandrasekhar 1950, Radiative Transfer
Kajiya 1984, Ray Tracing Volume Densities
Levoy 1988, Display of Surfaces from Volume Data
Max 1995, Optical Models for Direct Volume Rendering
Porter and Duff 1984, Compositing Digital Images
Novak et al 2018, Monte Carlo methods for physically based volume rendering

Volumetric formulation for NeRF

Scene is a cloud of colored fog

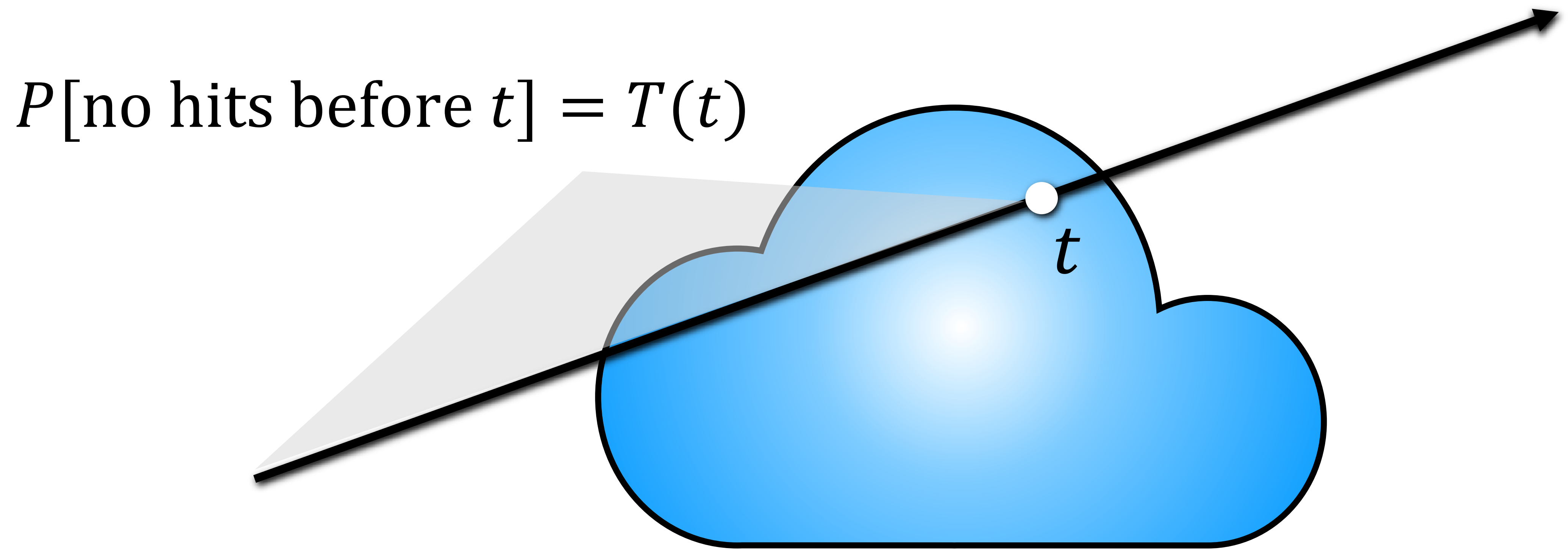
Max and Chen 2010, Local and Global Illumination in the Volume Rendering Integral

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

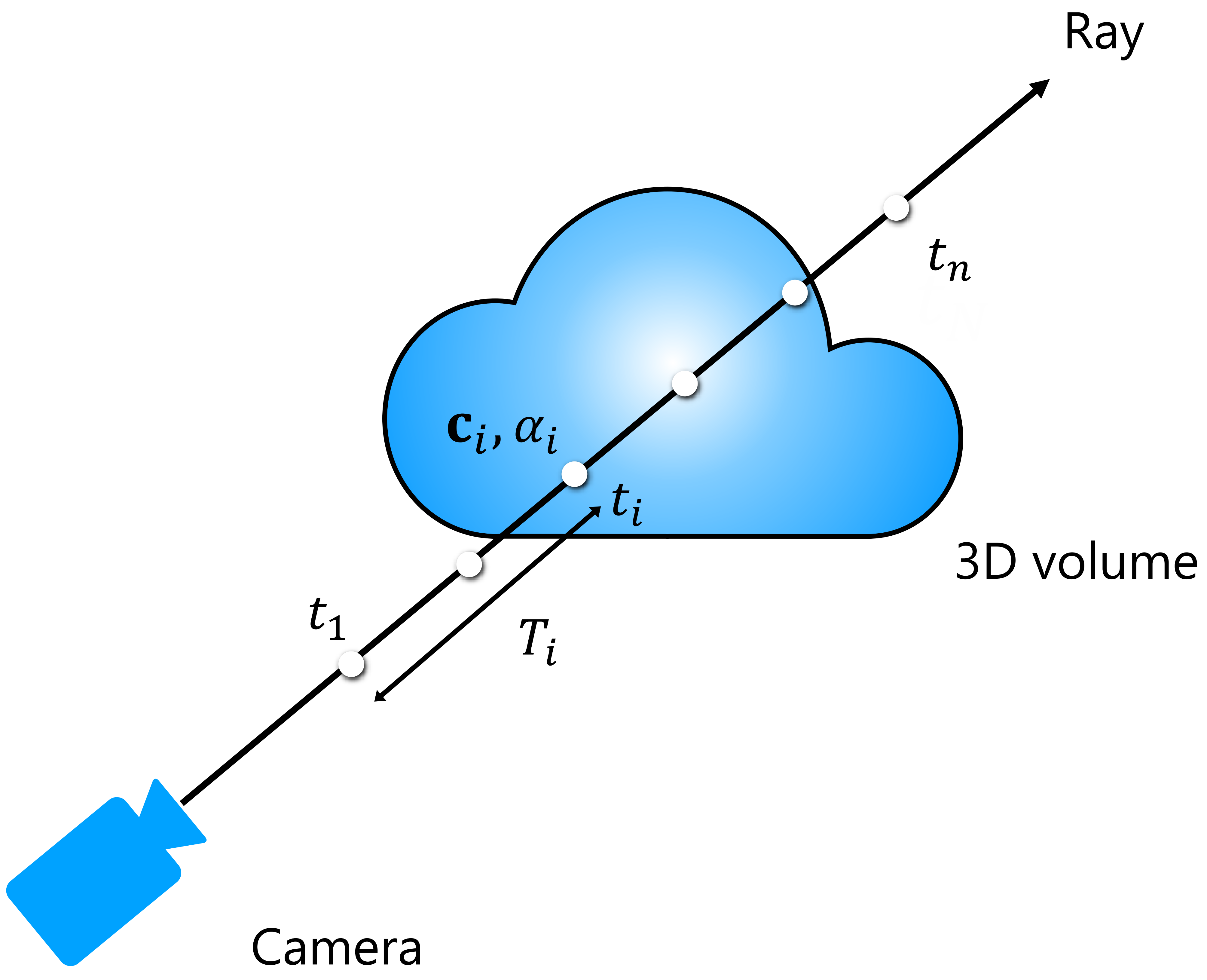
weights

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

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

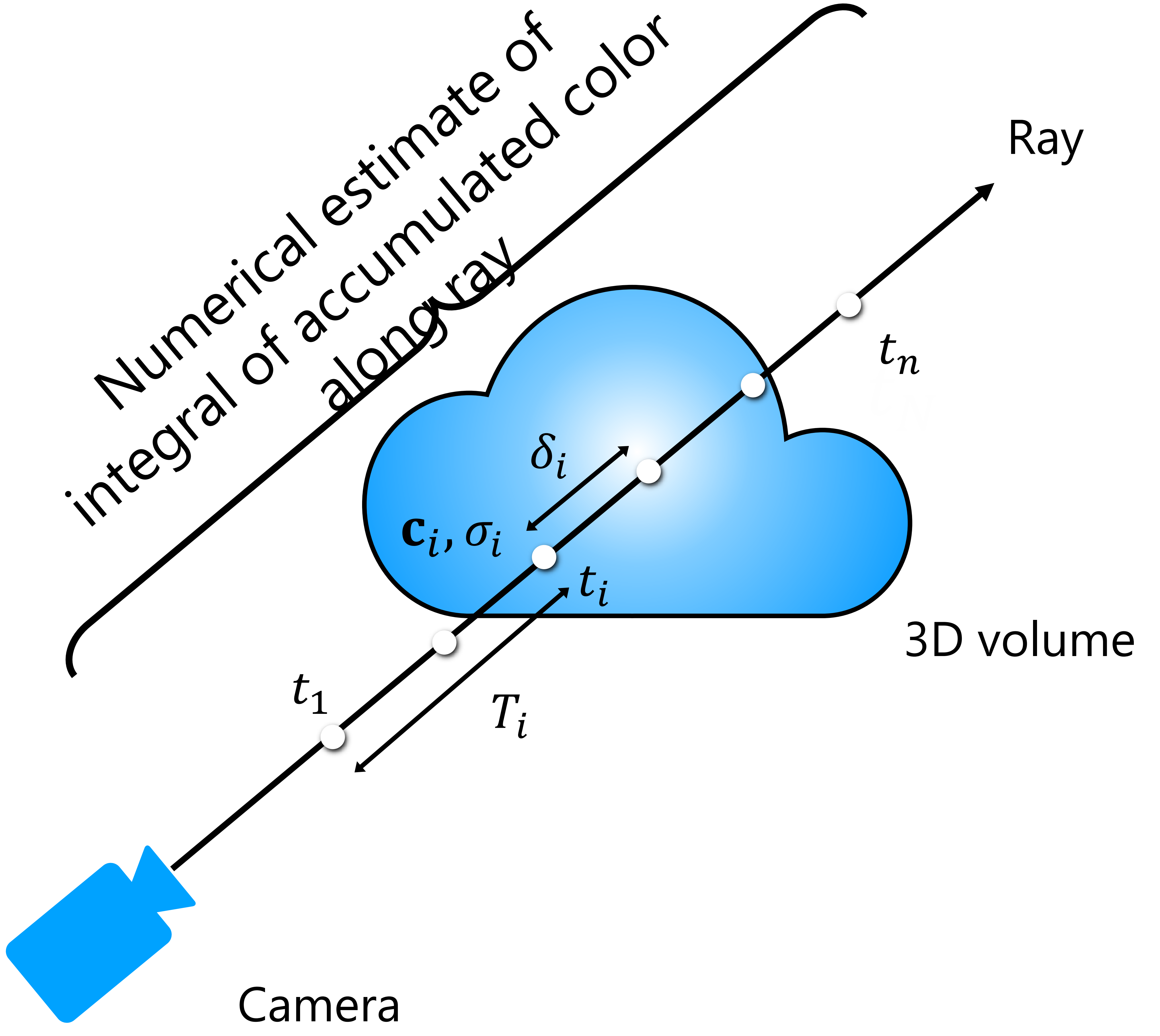
final rendered color along ray

weights

colors

How much light is blocked earlier along ray:

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



Slight modification: 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):

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

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

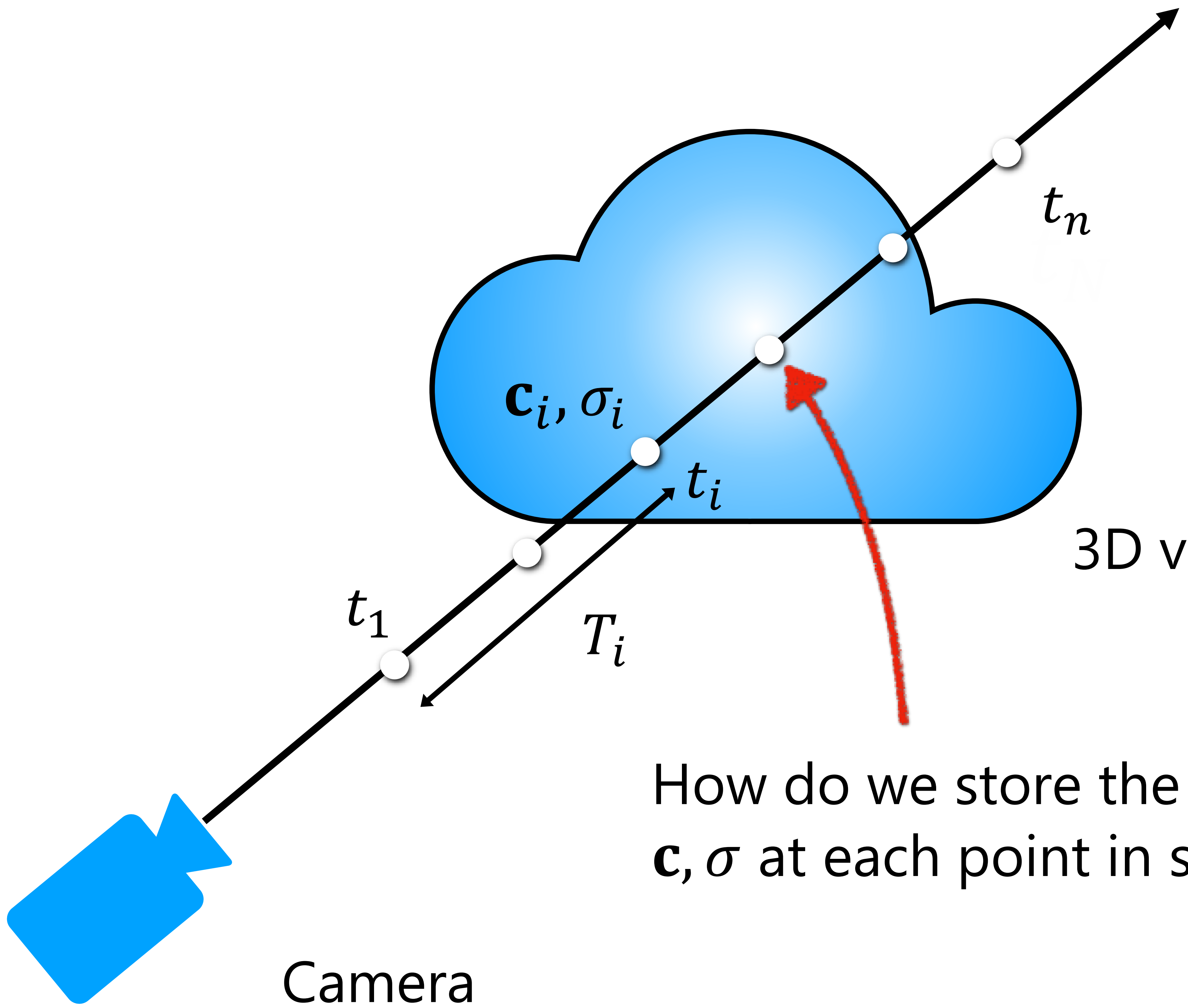
weights

colors

How much light is blocked earlier along ray:

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

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

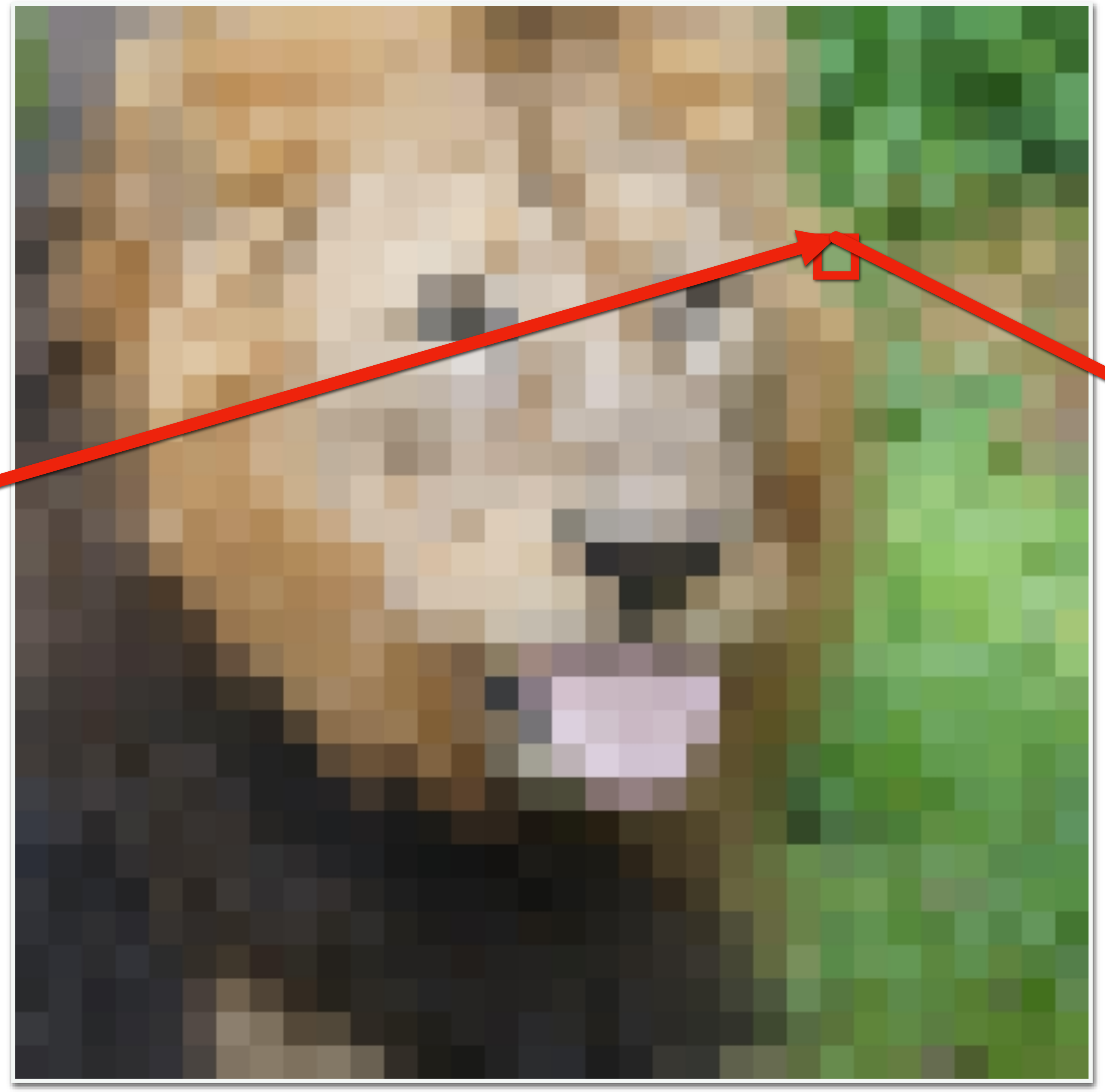


NeRF Overview

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

Toy problem: storing 2D image data

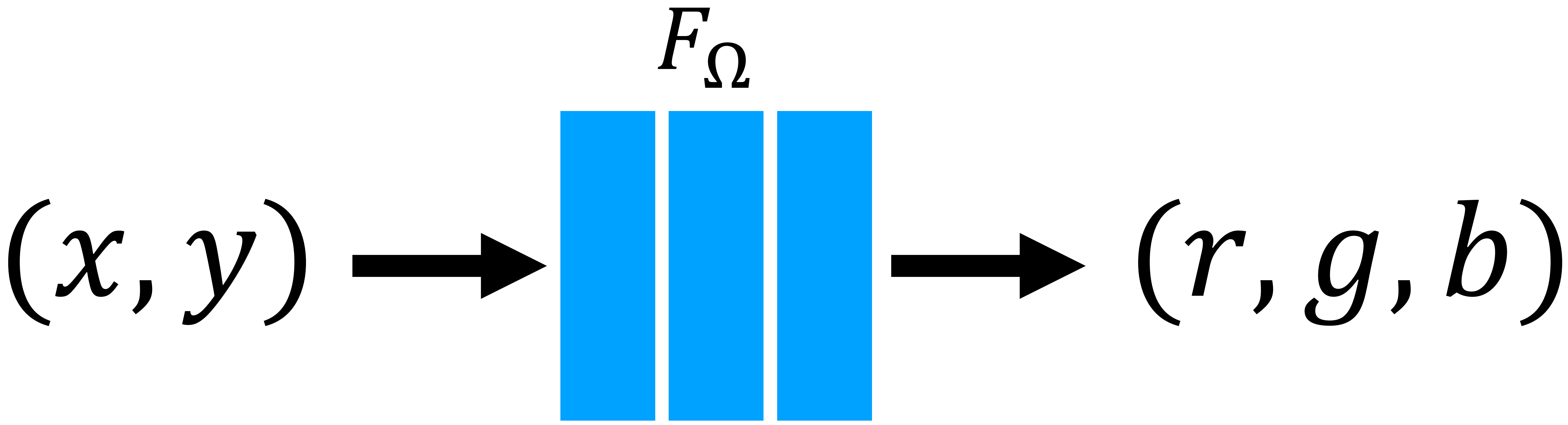
(x, y)



(r, g, b)

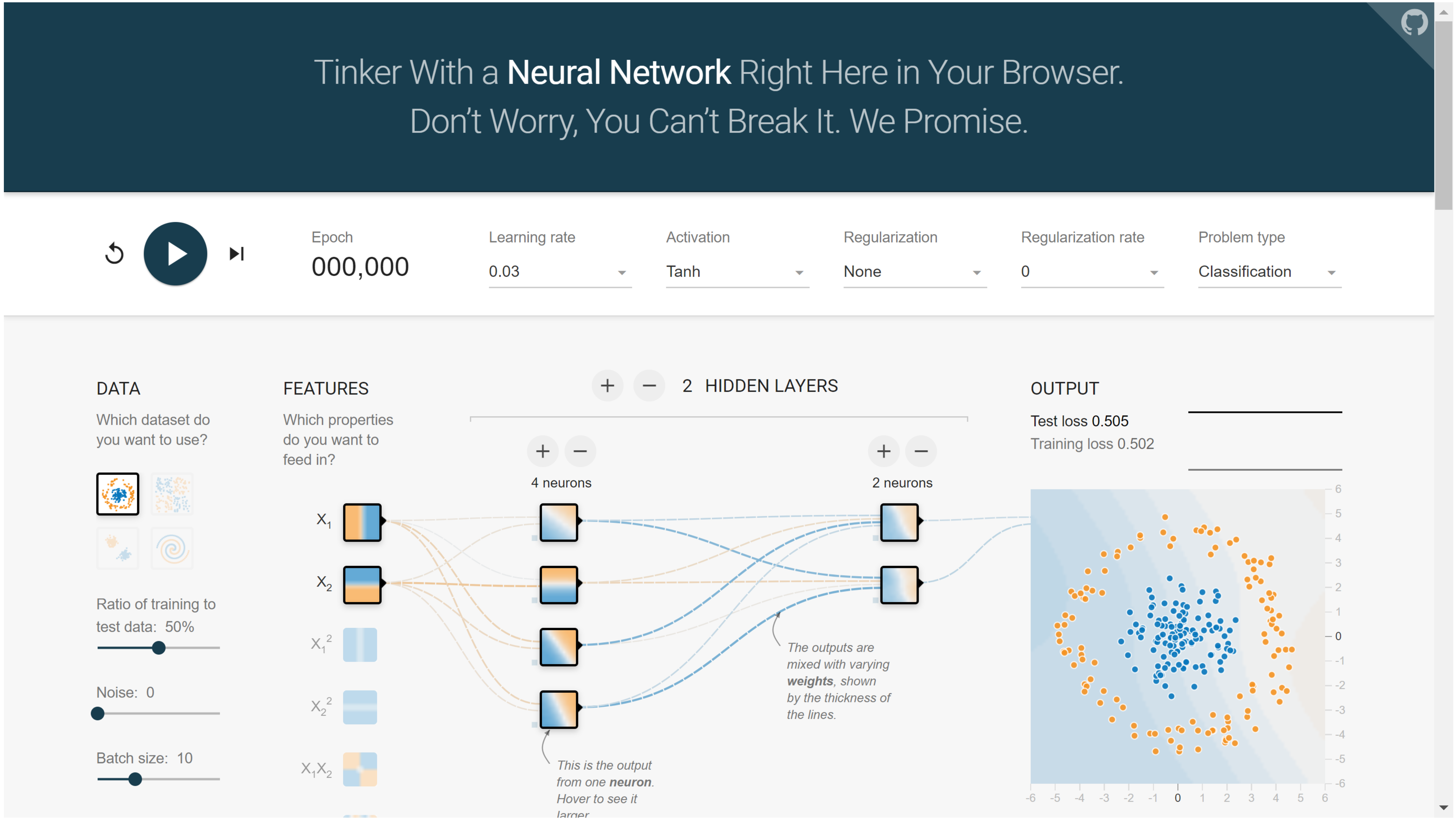
Usually we store an image as a
2D grid of RGB color values

Toy problem: storing 2D image data



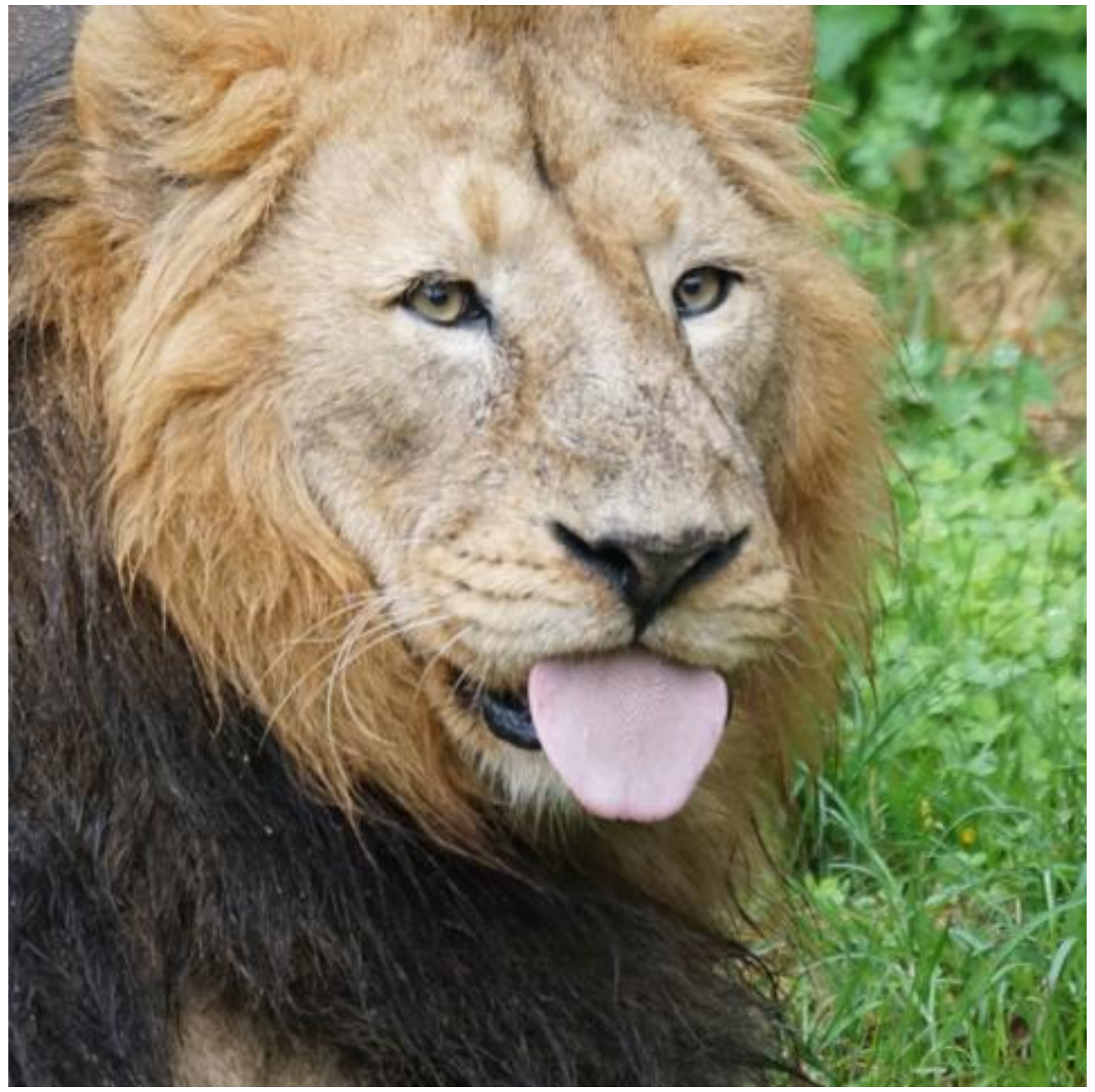
What if we train a simple fully-connected network (MLP) to do this instead?

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:

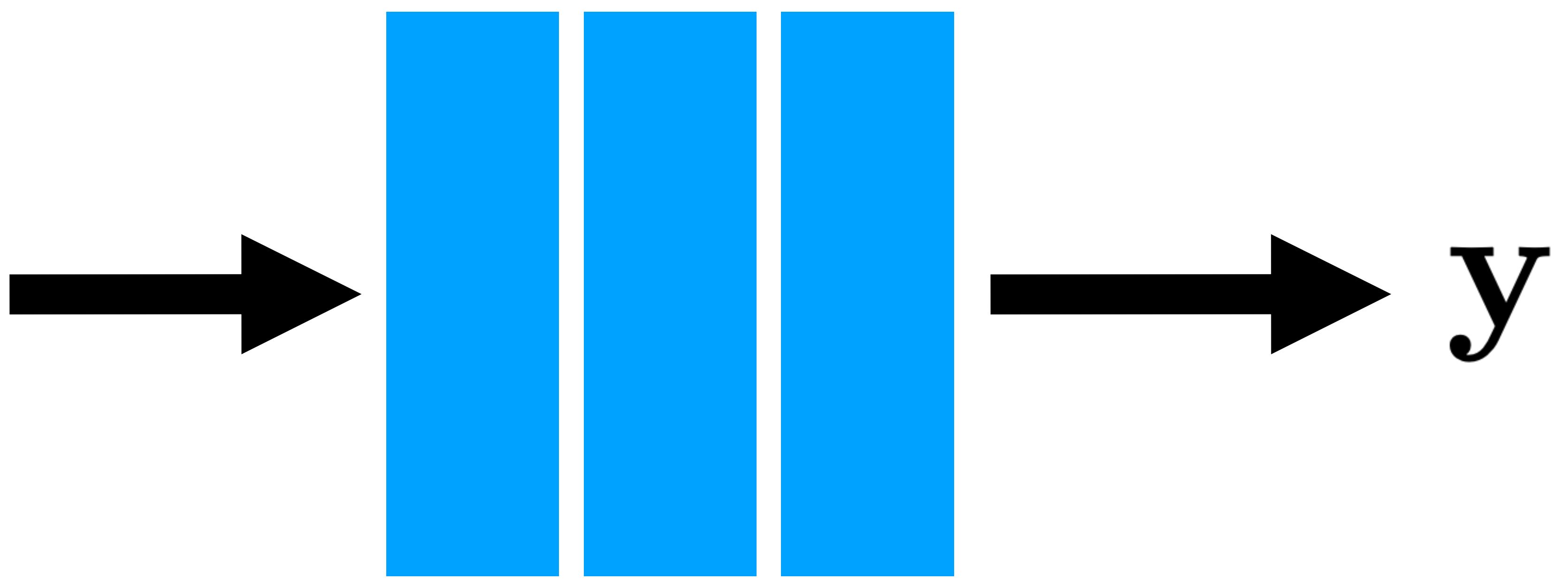
"Standard" coordinate-based MLPs cannot represent high frequency functions

Solution:

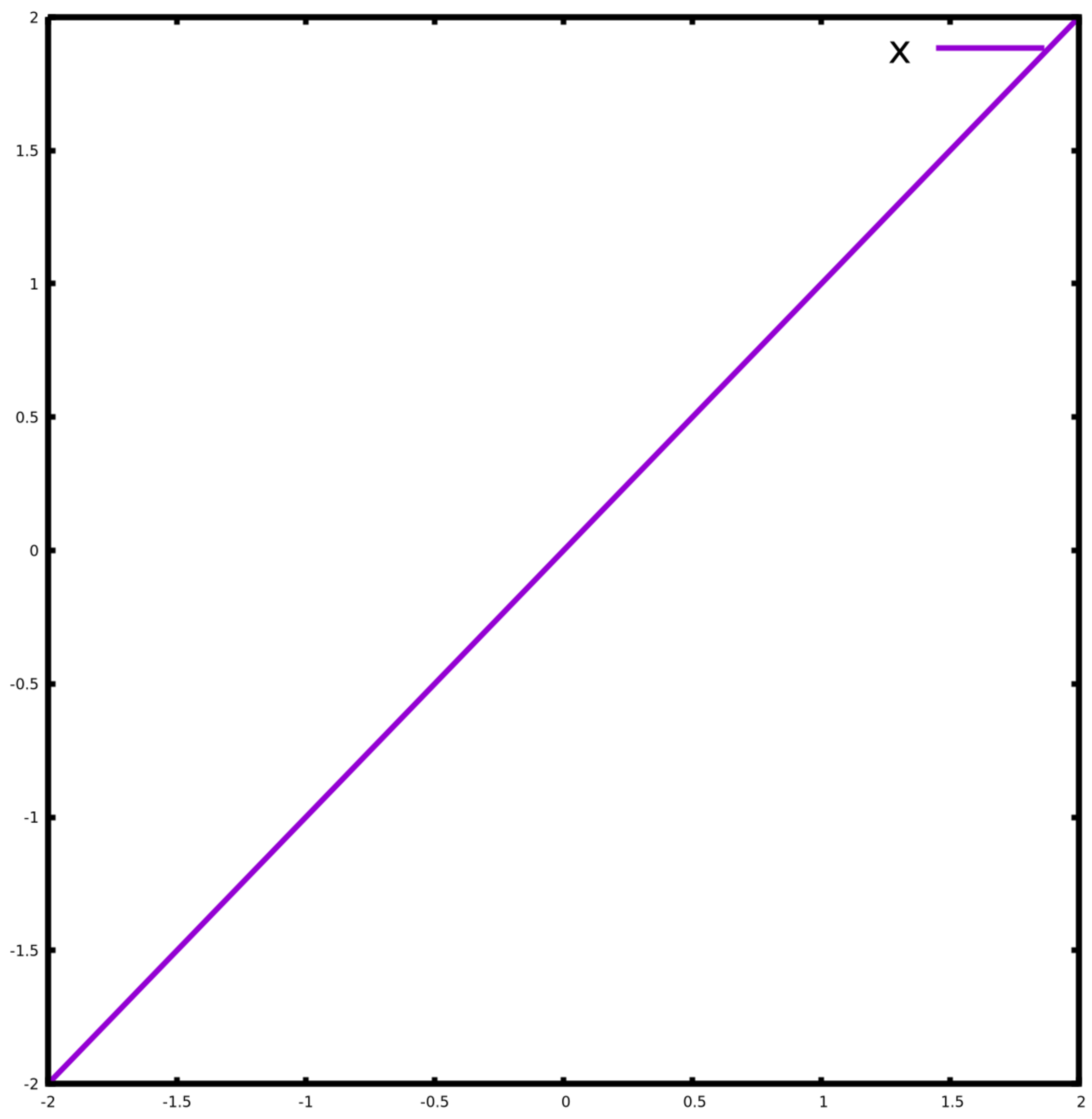
Pass input coordinates through a high frequency mapping first

Example mapping: "positional encoding"

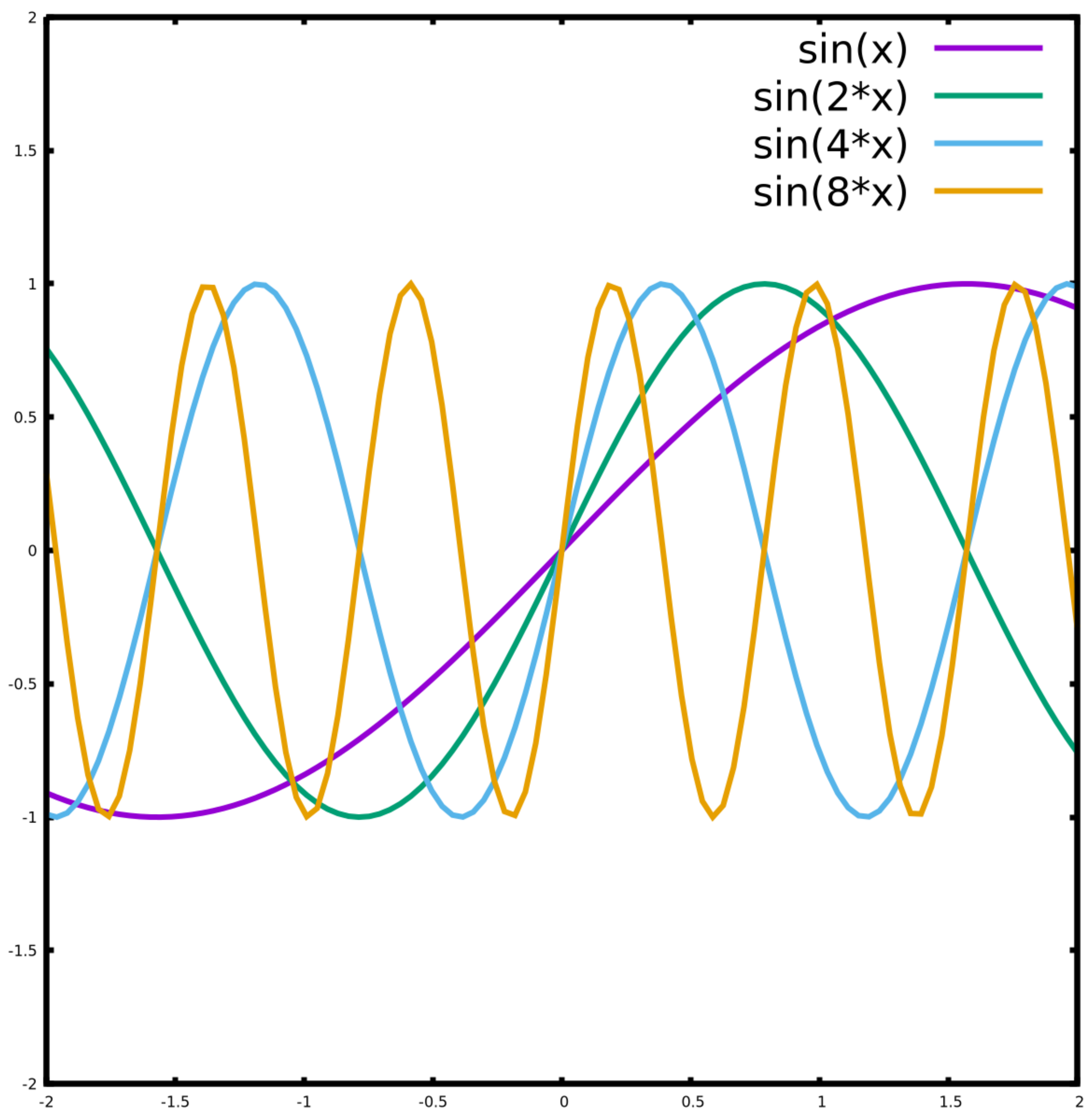
$$\begin{pmatrix} \sin(\mathbf{v}), \cos(\mathbf{v}) \\ \sin(2\mathbf{v}), \cos(2\mathbf{v}) \\ \sin(4\mathbf{v}), \cos(4\mathbf{v}) \\ \dots \\ \sin(2^{L-1}\mathbf{v}), \cos(2^{L-1}\mathbf{v}) \end{pmatrix}$$



Positional encoding

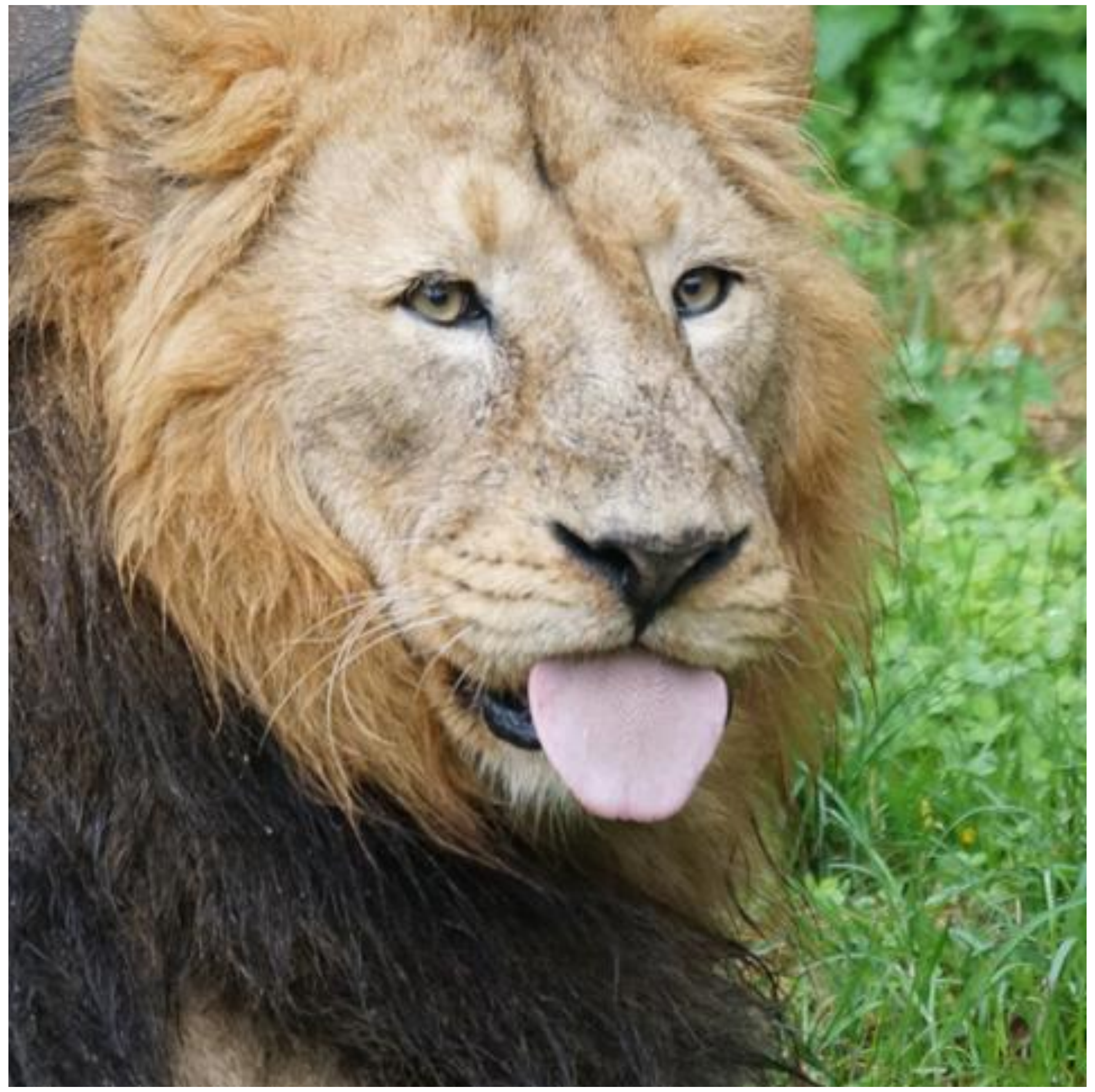


Raw encoding of a number ***x***



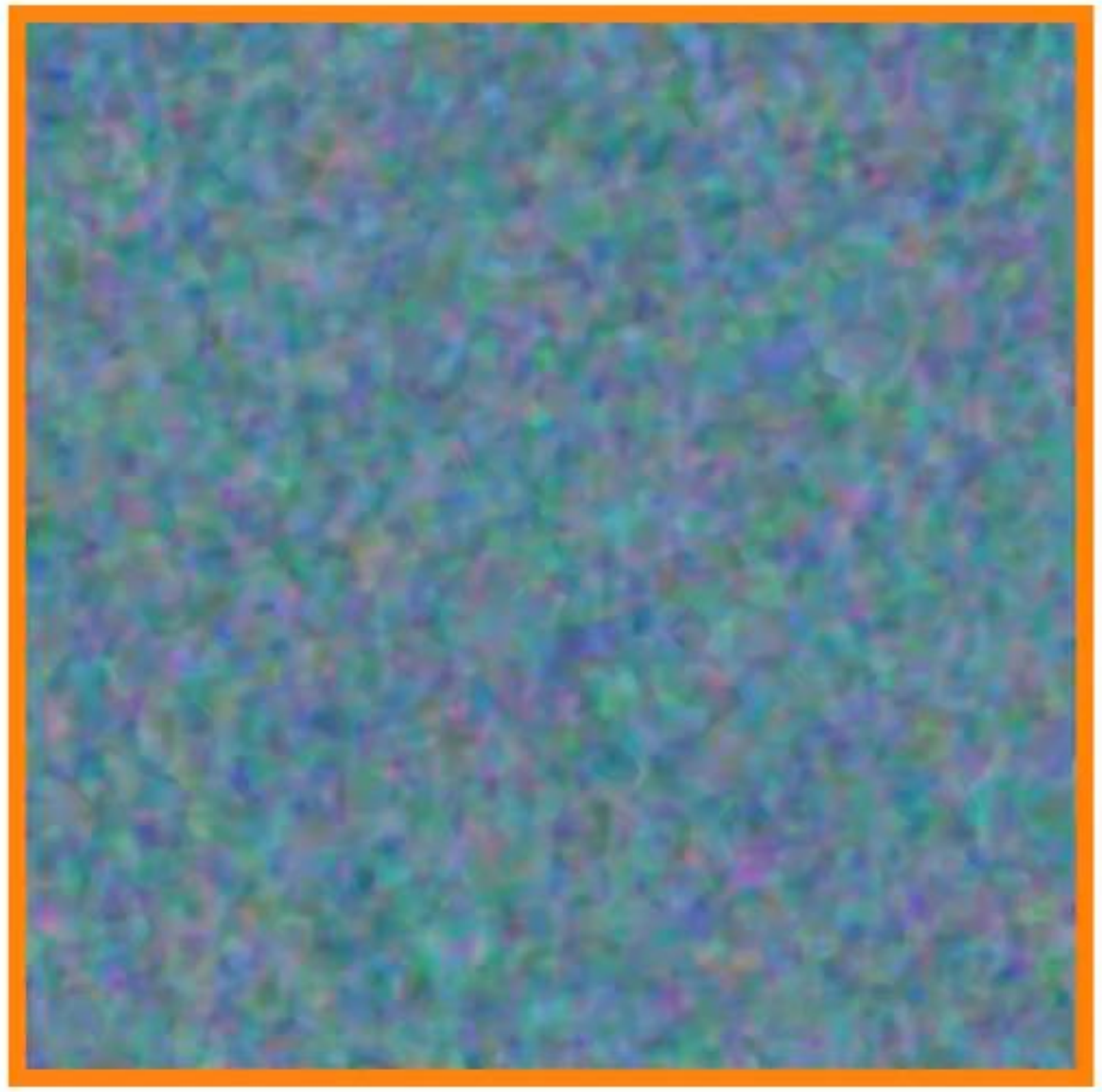
"Positional encoding" of a number ***x***

Problem solved!



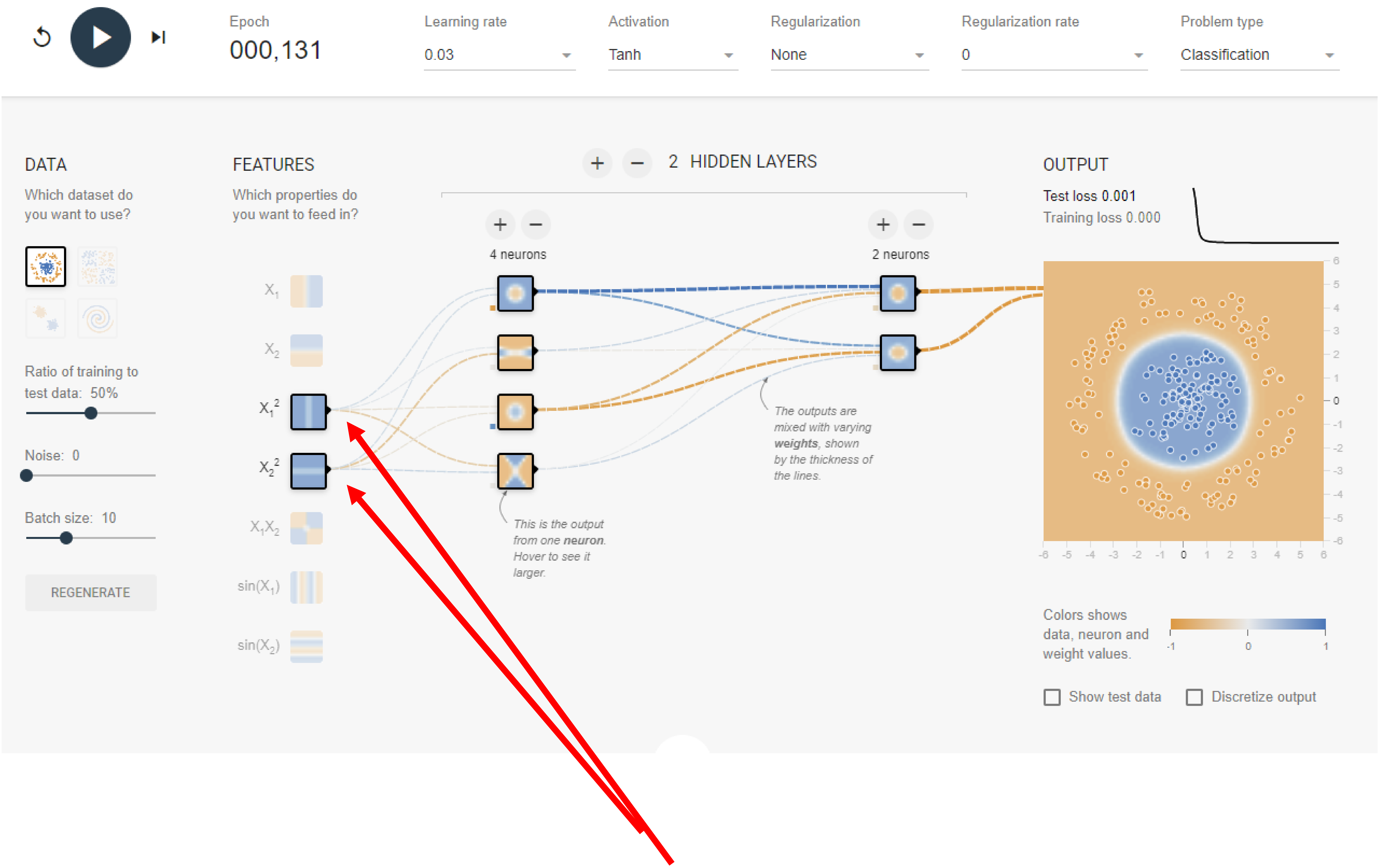
Ground truth image

Neural network output without high frequency mapping



Neural network output with high frequency mapping

Sometimes a better input encoding is all you need



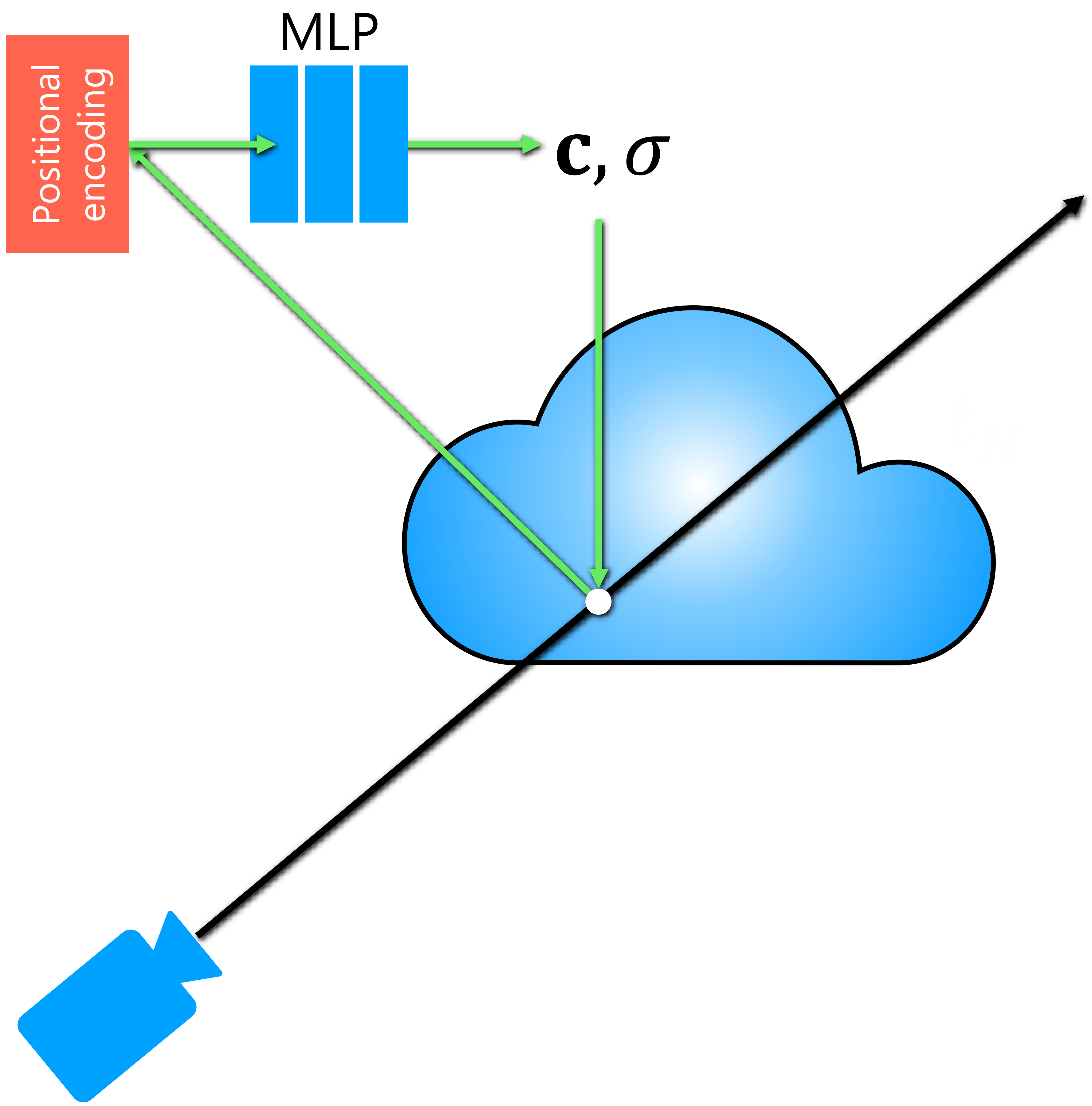
Recall “squared” encoding in TensorFlow Playground

NeRF Overview

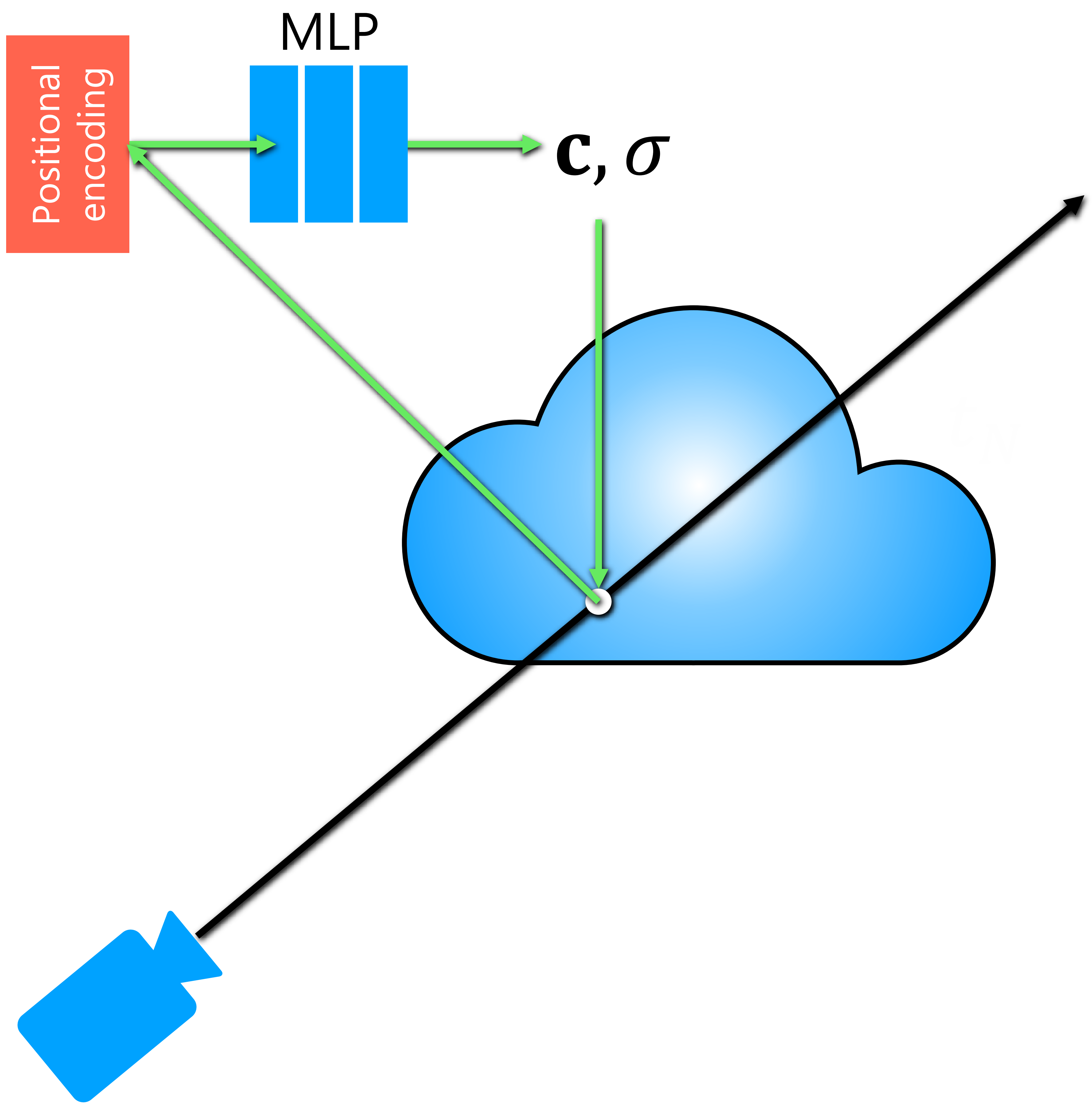
- Volumetric rendering
- Neural networks as representations for spatial data
- Neural Radiance Fields (NeRF)

NeRF = volume rendering + coordinate-based network

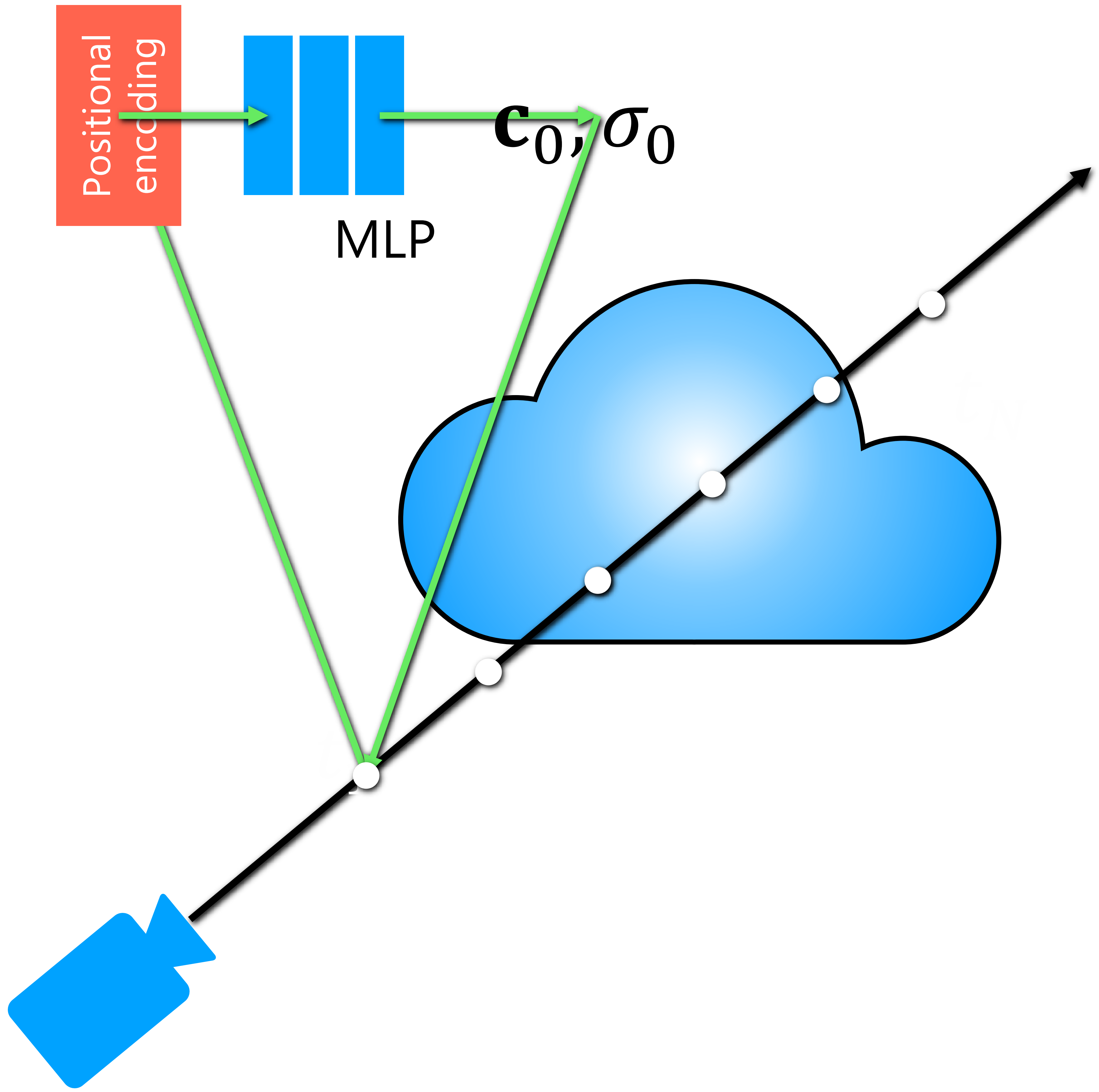
How do we store the values of $\mathbf{c}, \sigma$ at each point in space



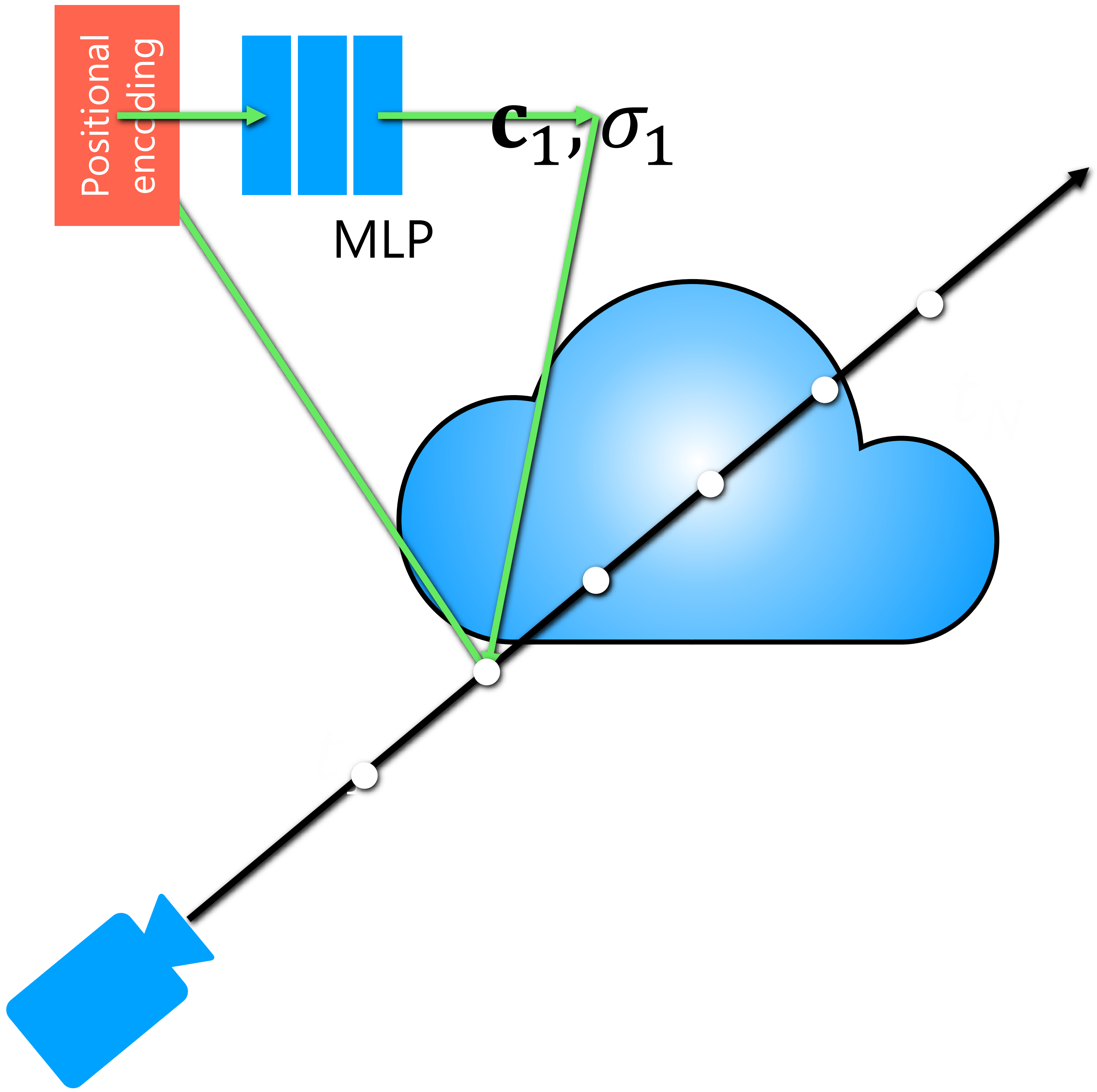
How do we store the values of $\mathbf{c}, \sigma$ at each point in space



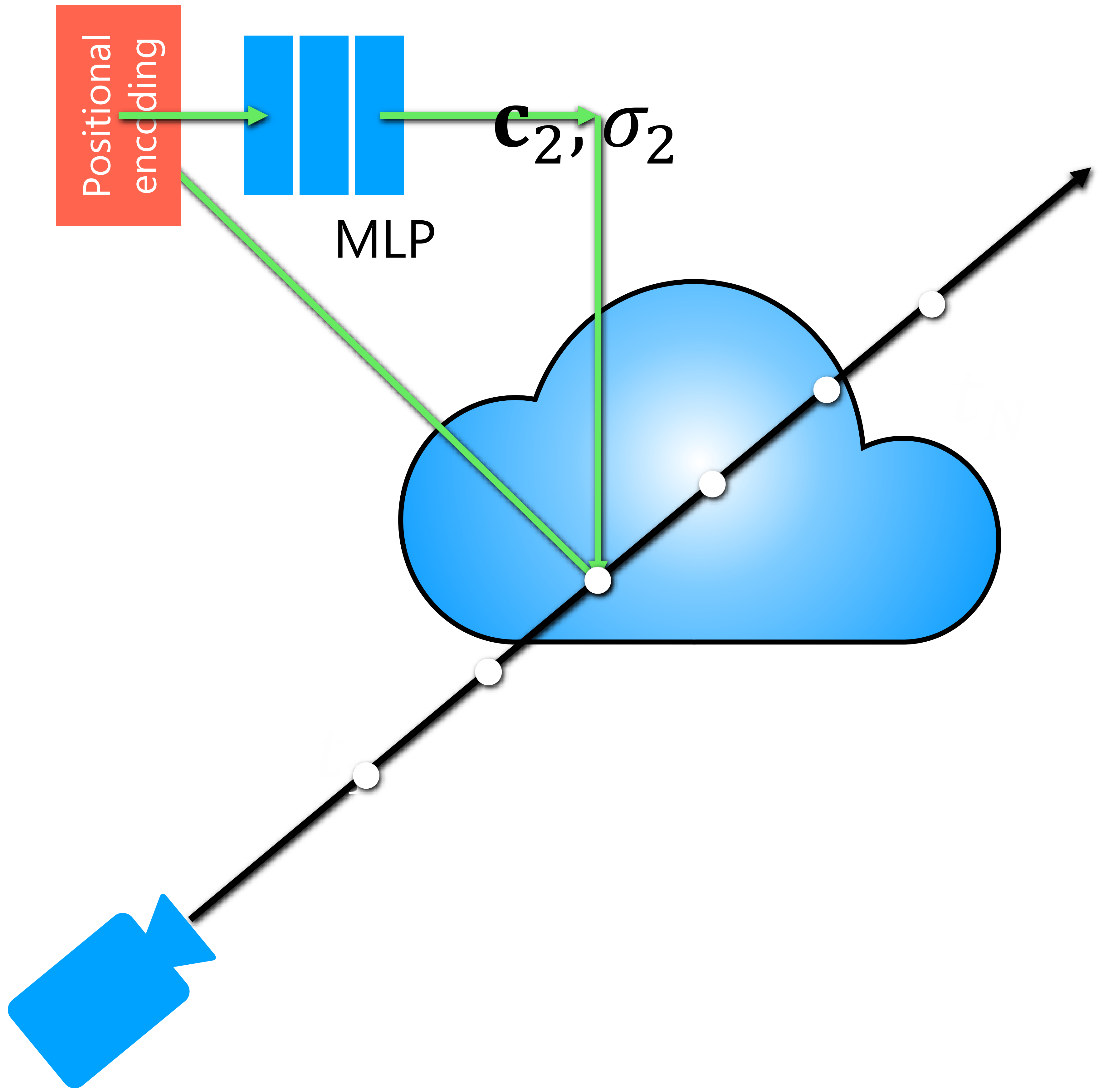
How do we store the values of $\mathbf{c}, \sigma$ at each point in space



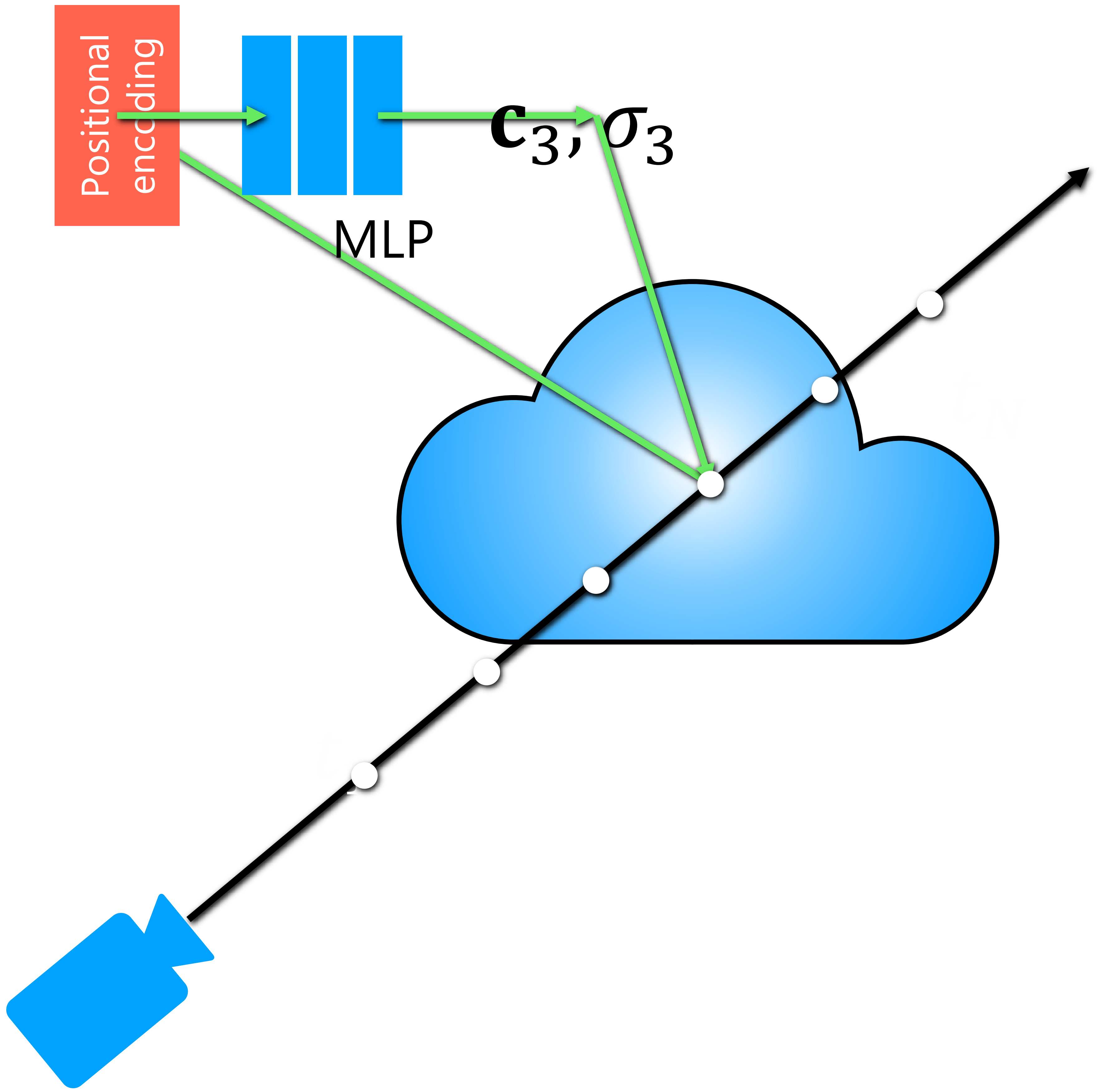
How do we store the values of $\mathbf{c}, \sigma$ at each point in space



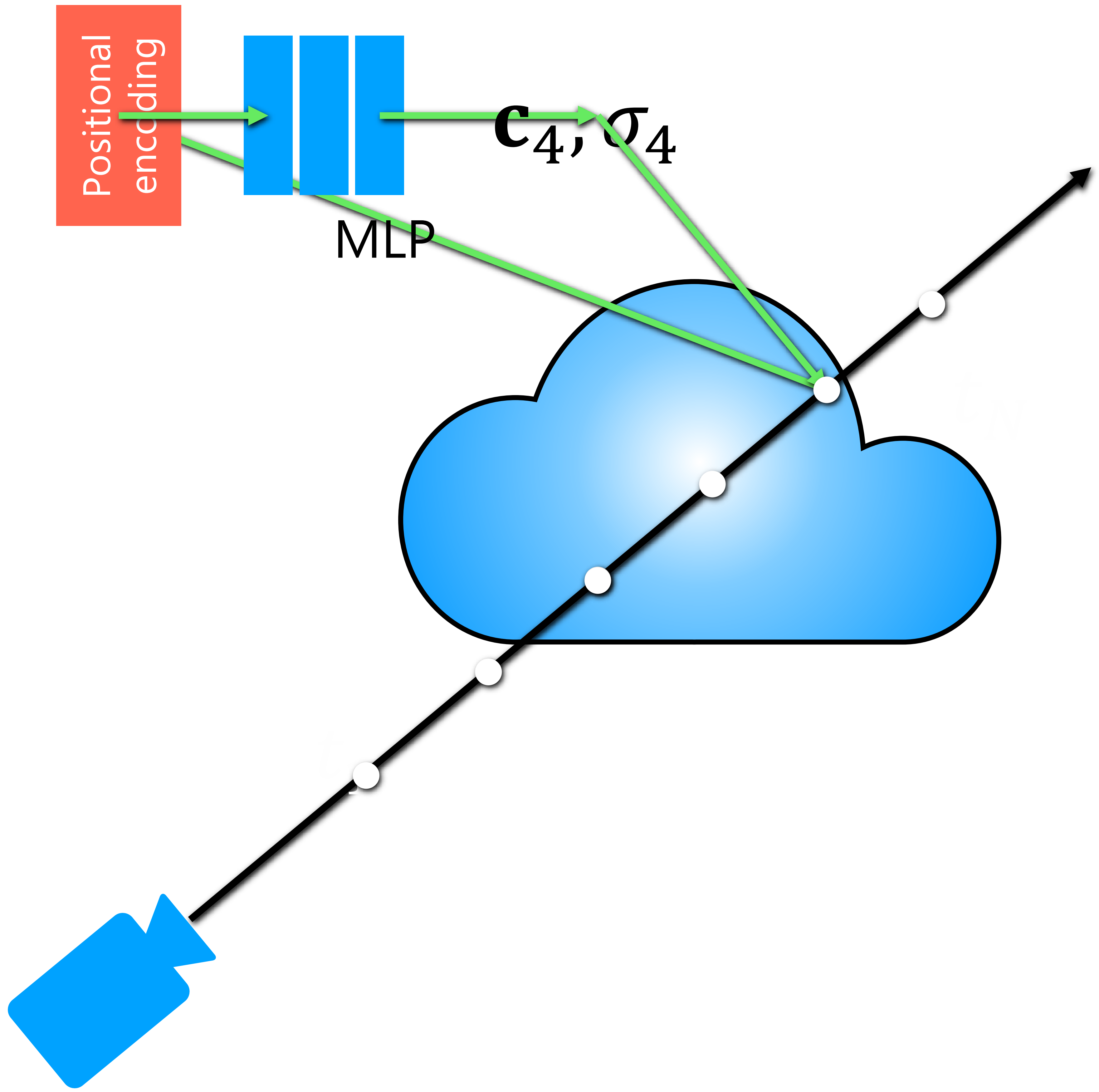
How do we store the values of $\mathbf{c}, \sigma$ at each point in space



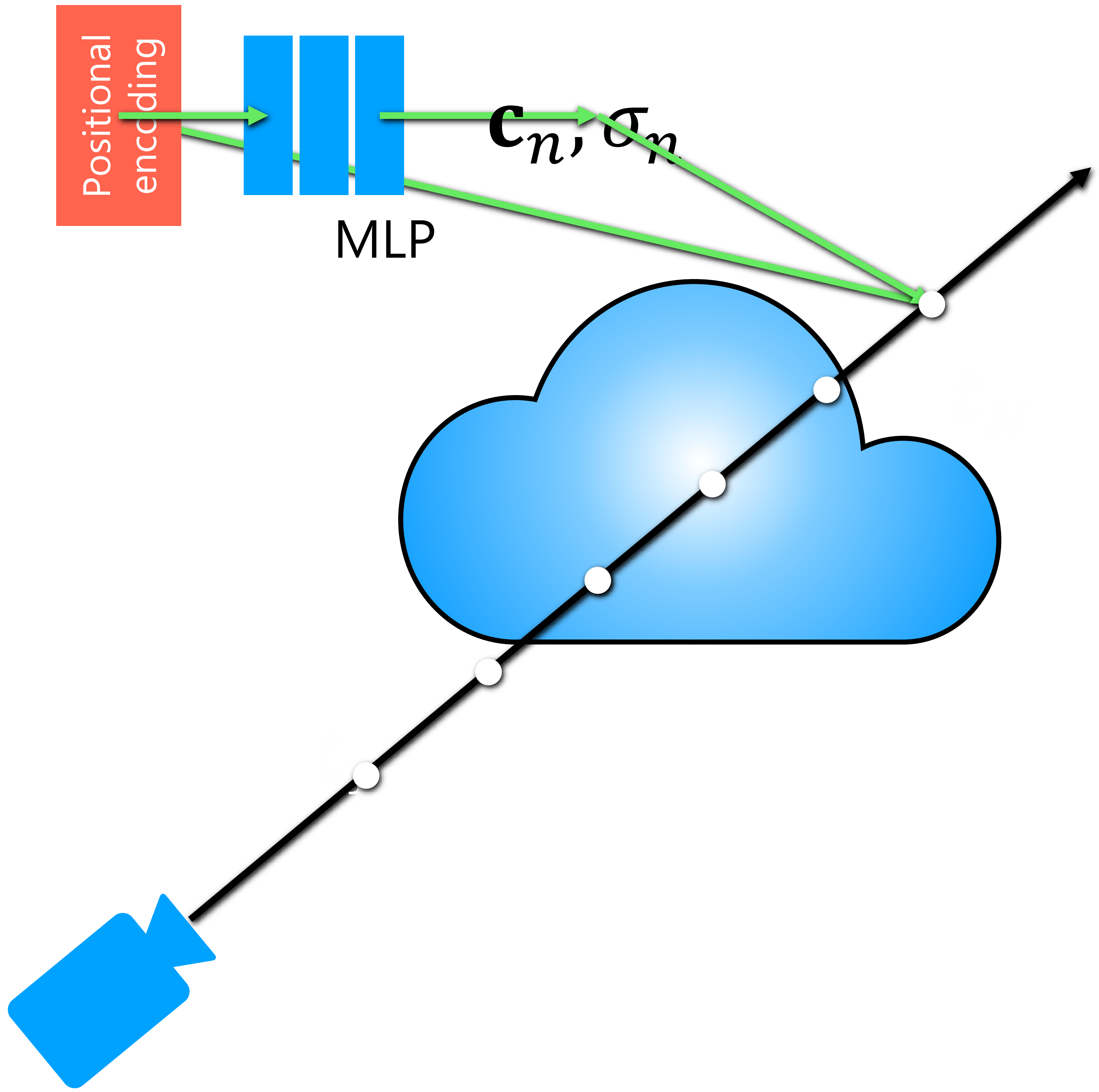
How do we store the values of $\mathbf{c}, \sigma$ at each point in space



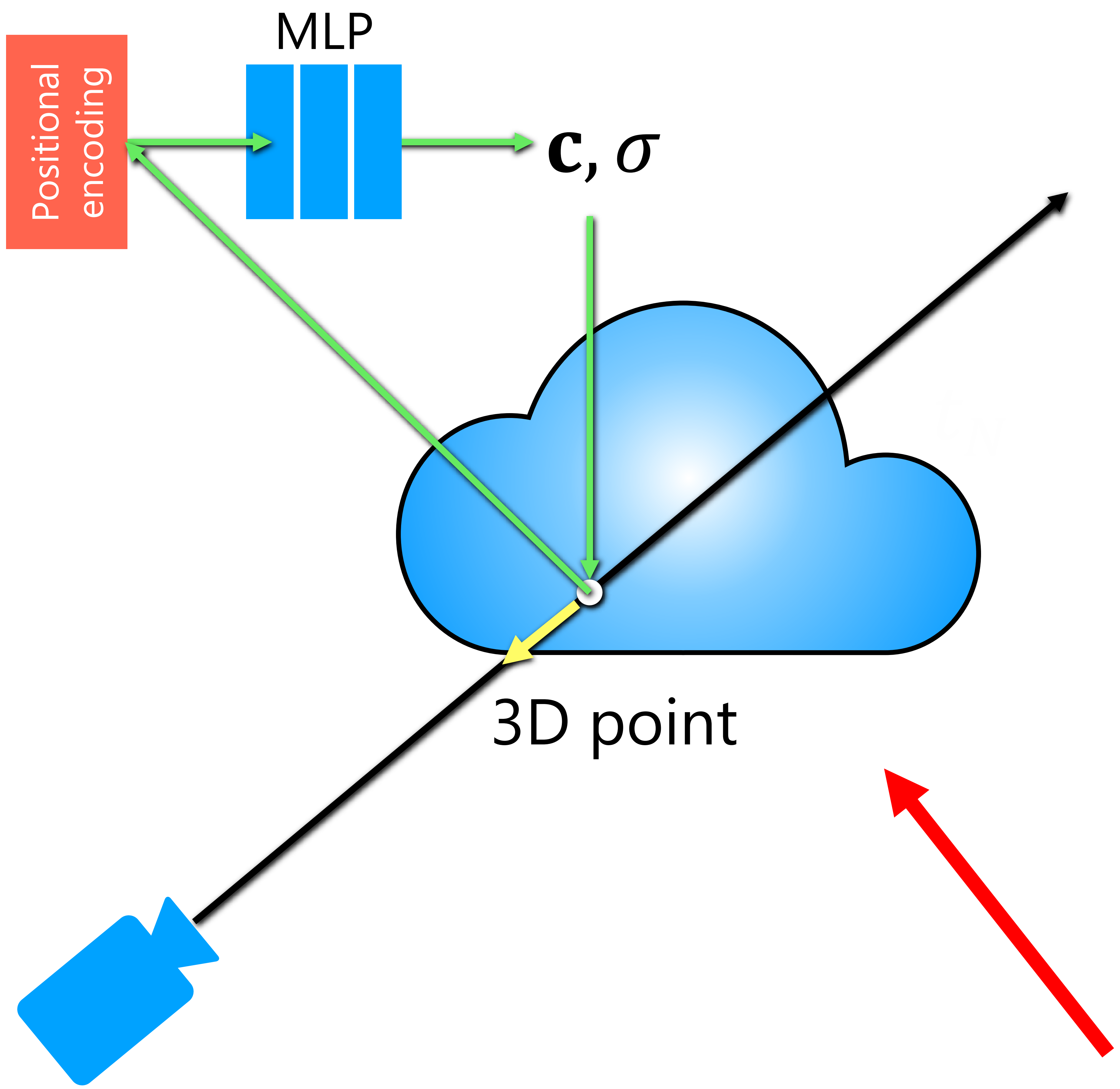
How do we store the values of $\mathbf{c}, \sigma$ at each point in space



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

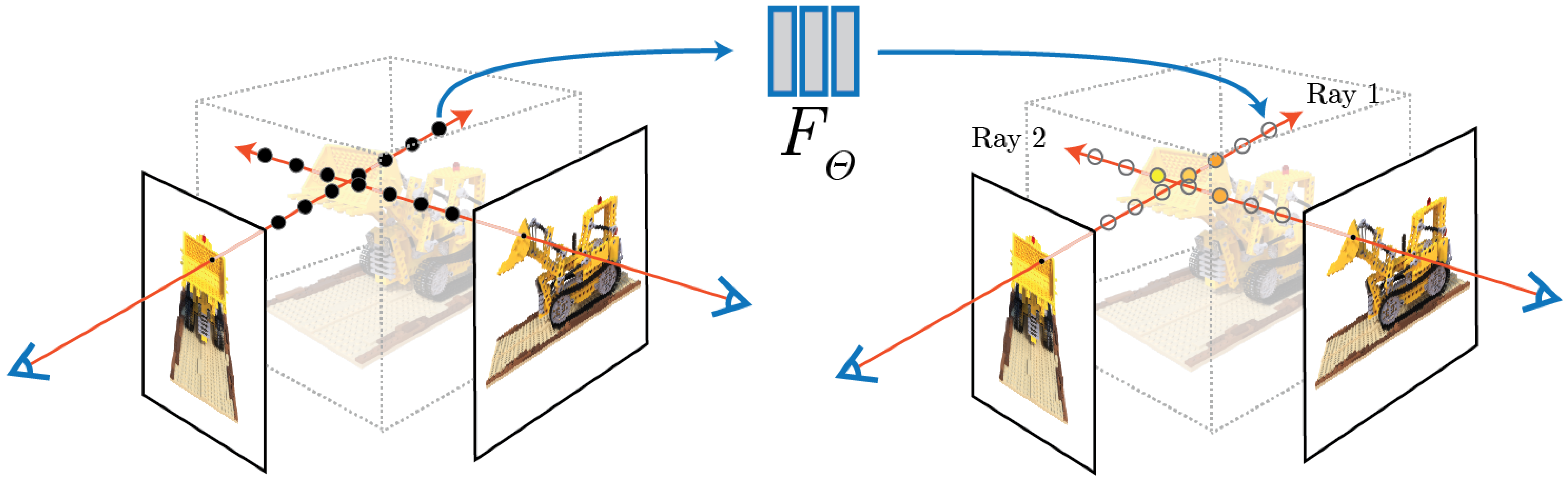


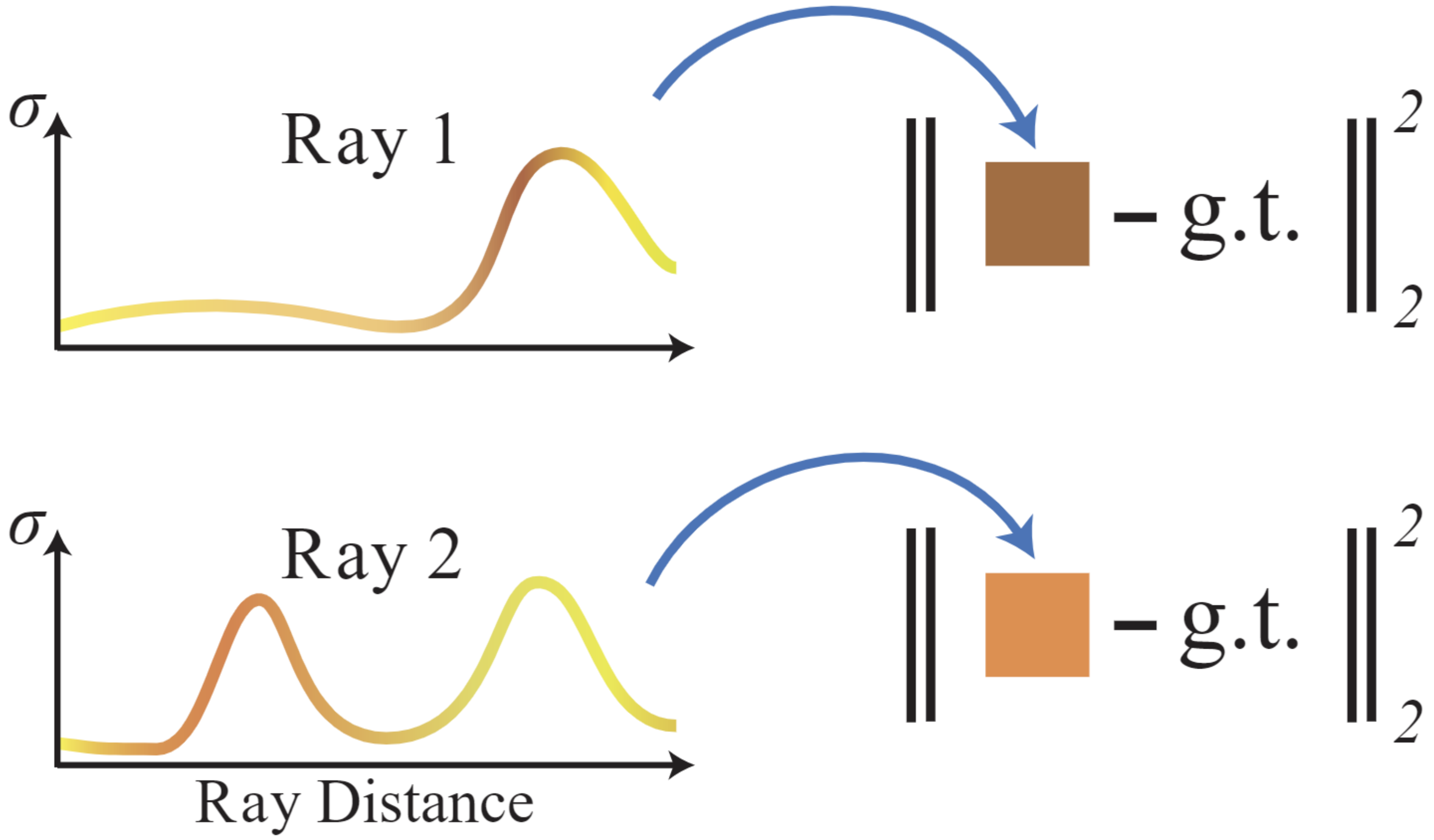
Extension: view-dependent field



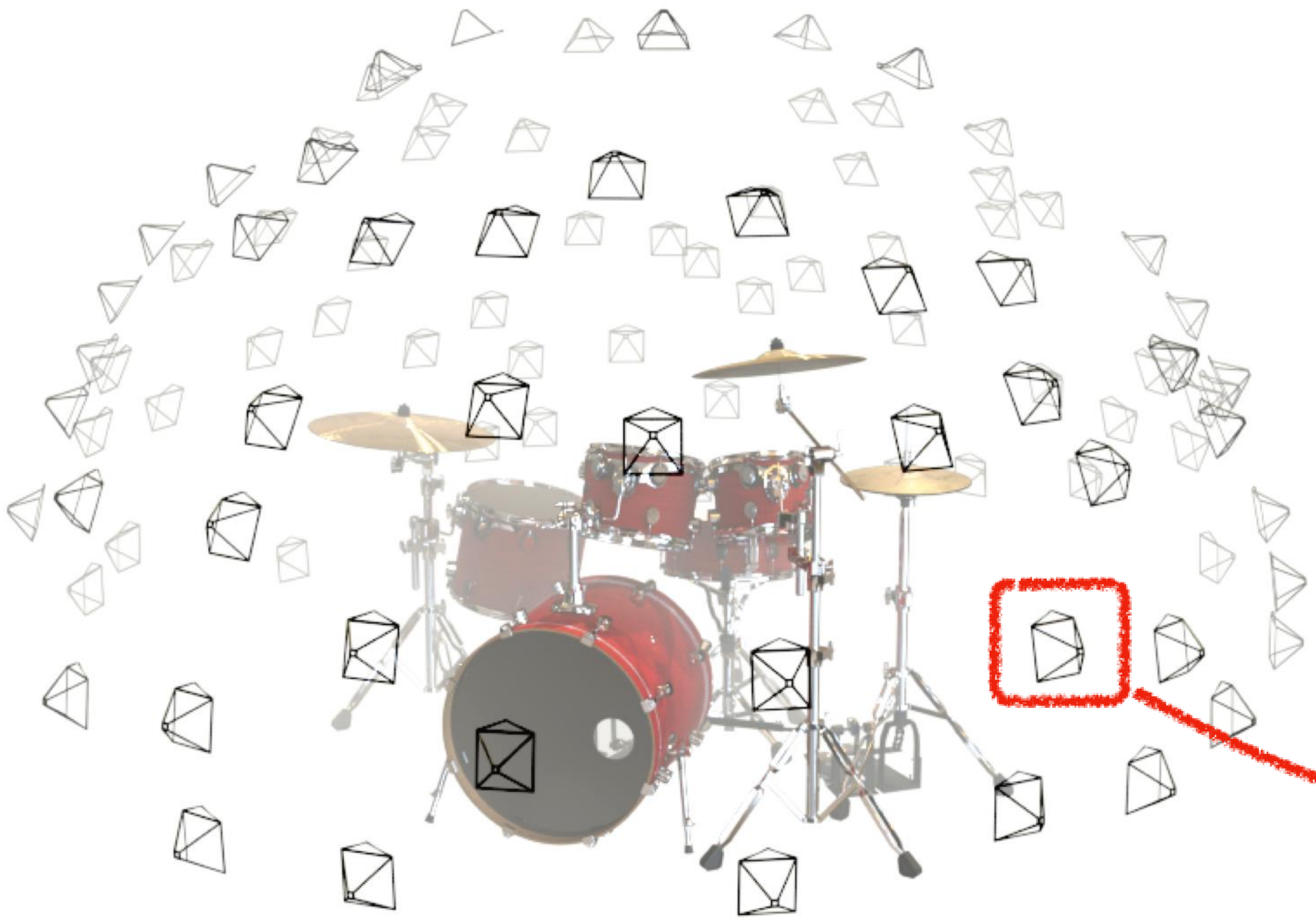
Include the ray direction in the input to the MLP → allows for capturing and rendering view-dependent effects (e.g., shiny surfaces)

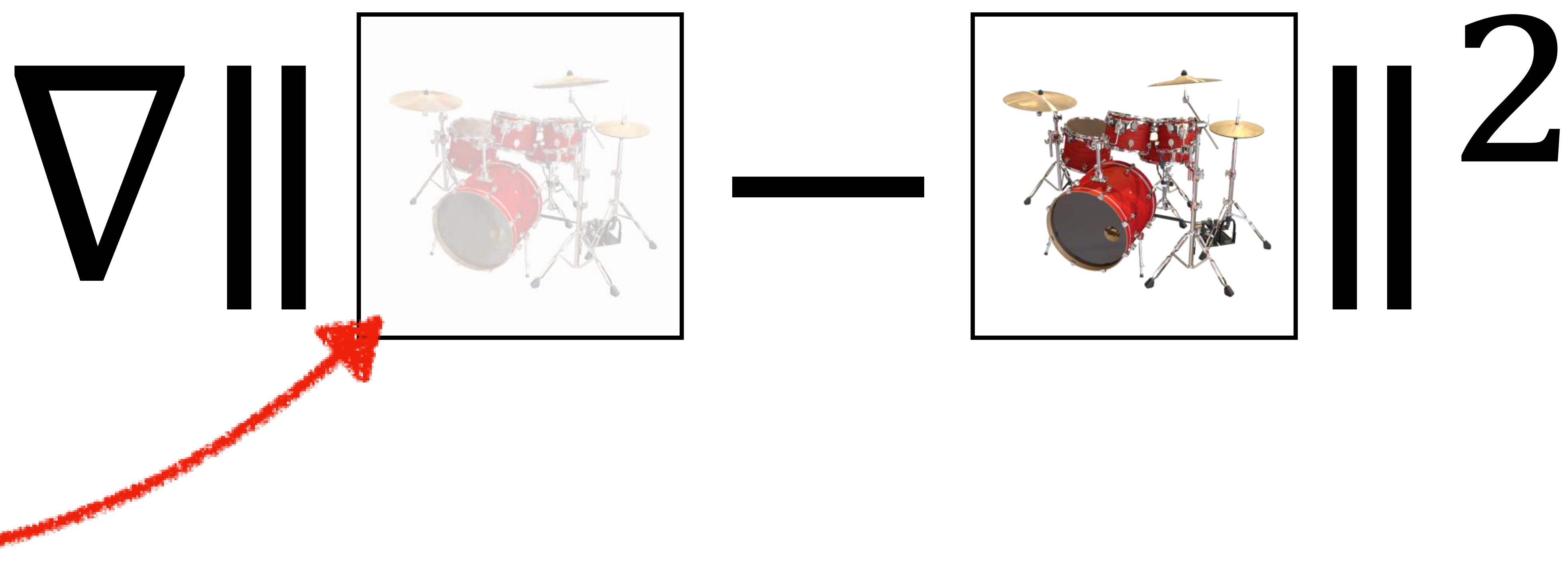
Putting it all together





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





Results

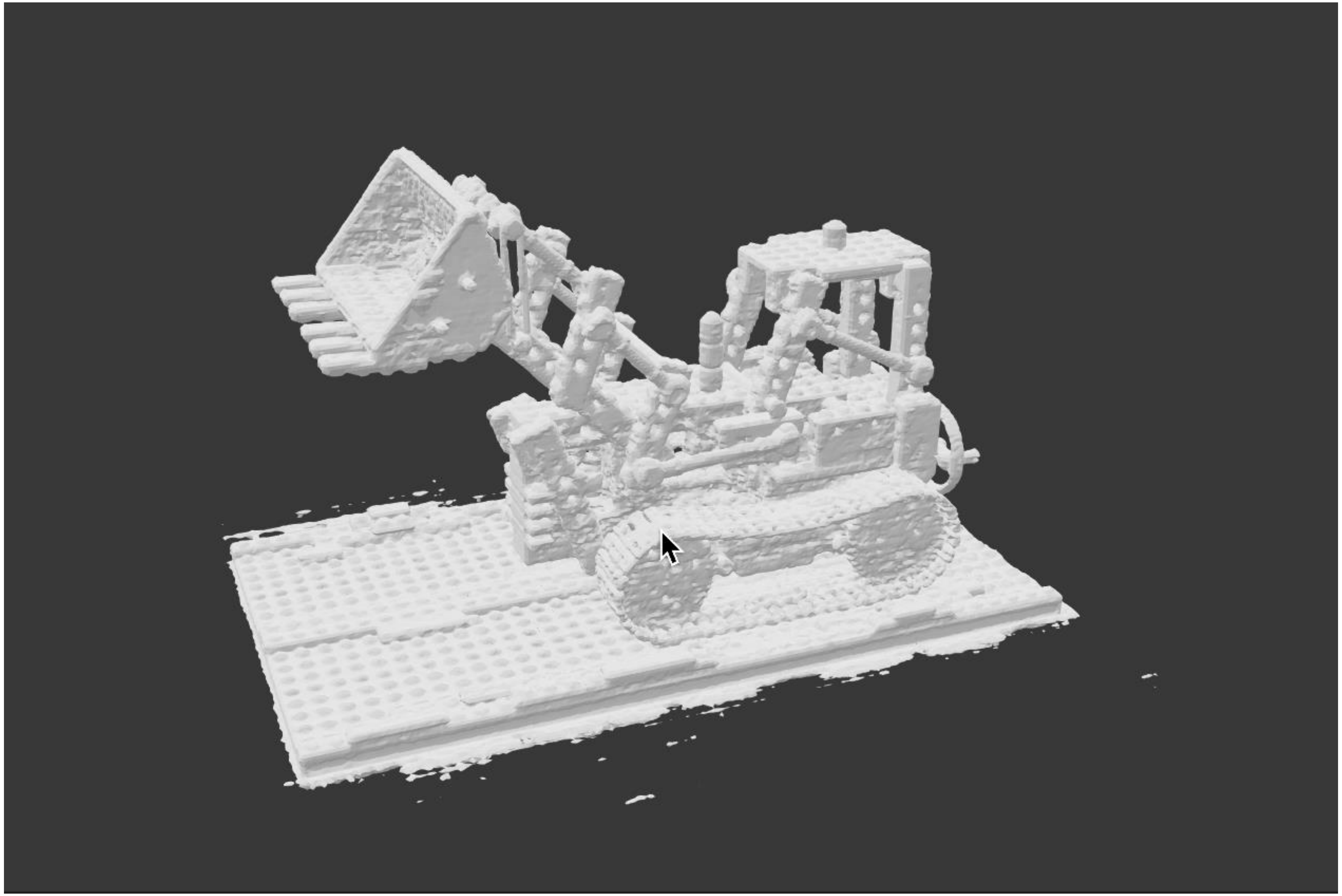
NeRF encodes convincing view-dependent effects using directional dependence

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

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

Questions?