Quiz 8 (on Canvas)
Closed book / closed note

Ends at 1:08pm

CS5670: Computer Vision

Inverse Graphics & Neural Radiance Fields (NeRFs)

NeRF Slides adapted from material courtesy of Pratul Srinivasan

Announcements

- Project 5 released today, due Wednesday, May 3 (8pm)
  - To be done in groups of 2
- Sample final exam online – see Ed Stem
- Final exam in-class on May 9

Project 5 Demo

Rendering in computer graphics

3D
Scene
Representation

Rendering

Adapted from material from Pratul Srinivas

Computer vision as inverse rendering

Inverse Rendering →

3D
Scene
Representation

→ Rendering

Adapted from material from Pratul Srinivas

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

Inverse Rendering

Neural Radiance Field

Rendering

Adapted from material from Pratul Srinivas

Deep learning for 3D reconstruction

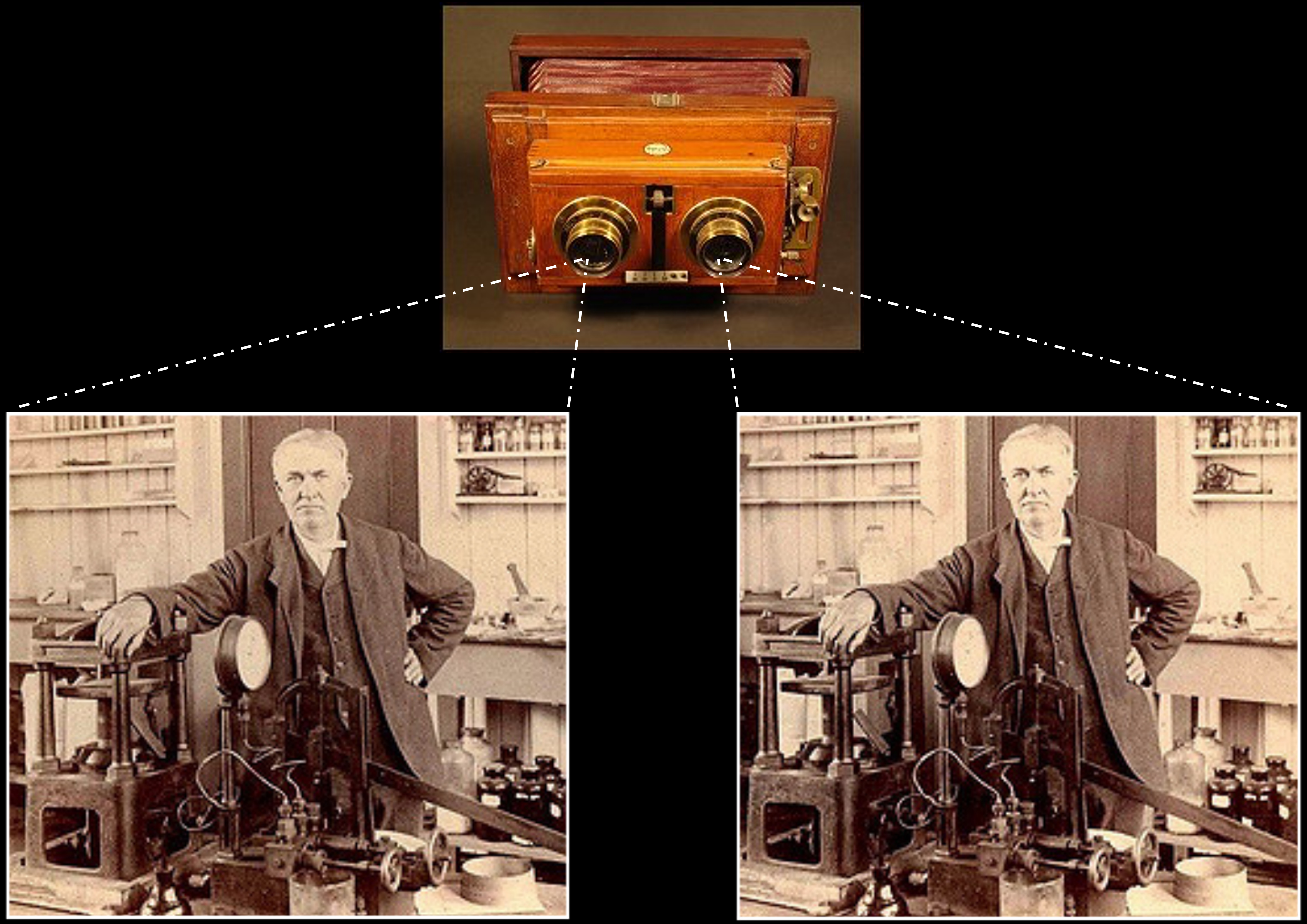
- Previously: we reconstruct geometry by running stereo or multi-view stereo on a set of images
  - "Classical" approach

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

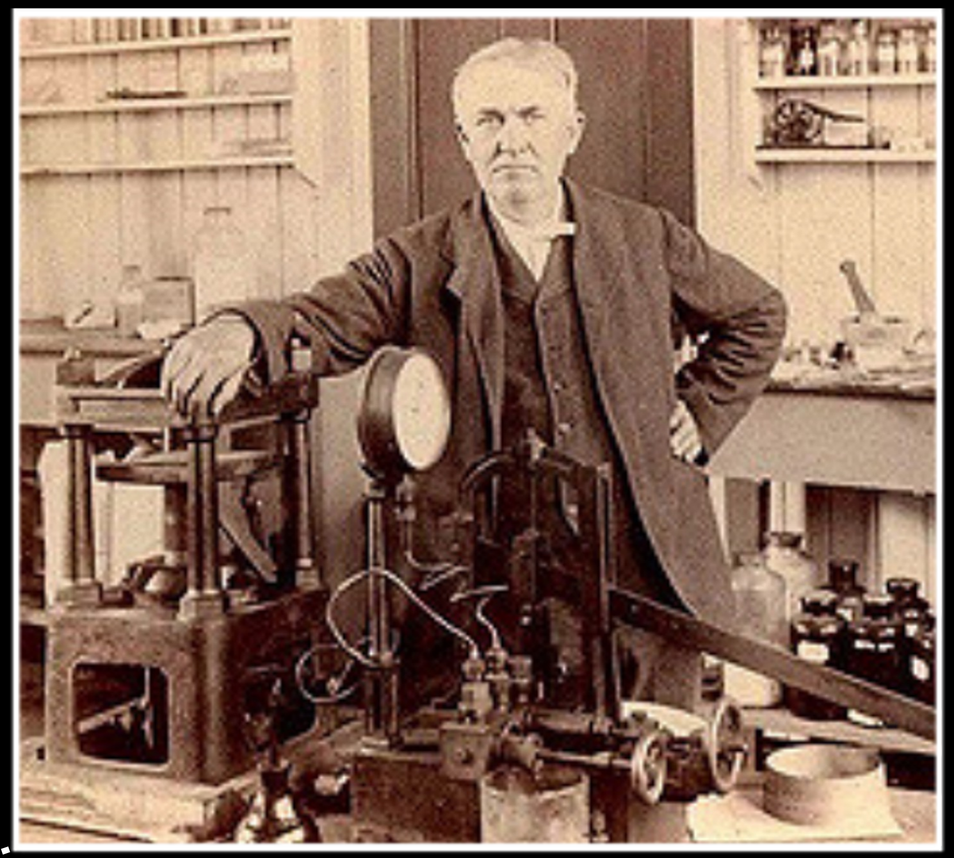
NeRF and related methods – Key ideas

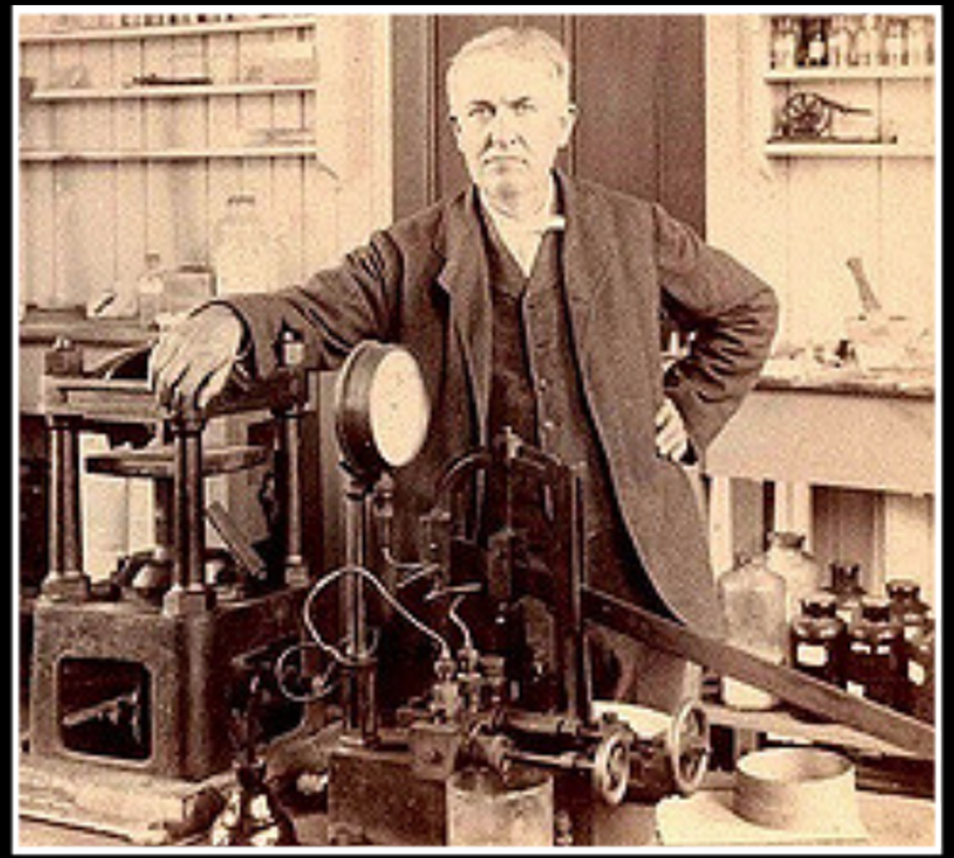
- We need to create a loss function and a scene representation that we can optimize using gradient descent to reconstruct the scene
- *Differentiable rendering*

Side Topic: Stereo Photography

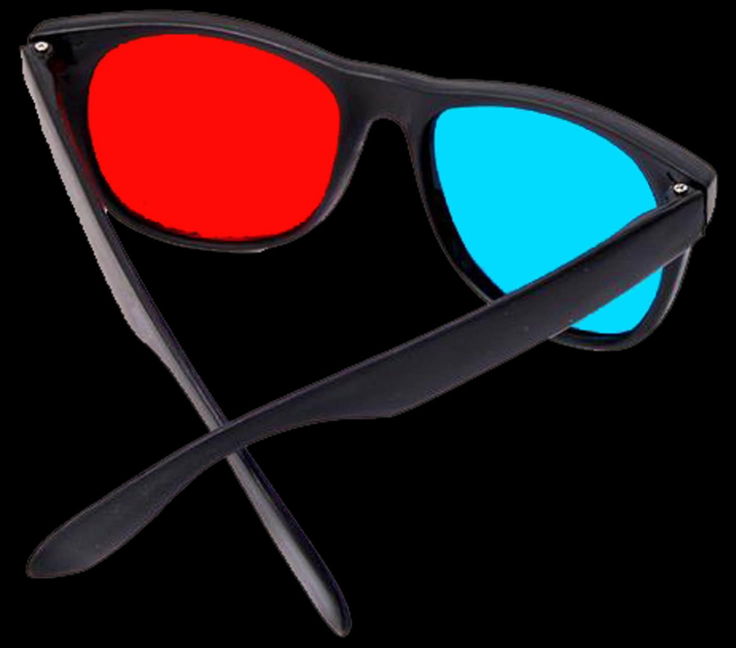


Stereo Photography





Viewing Devices



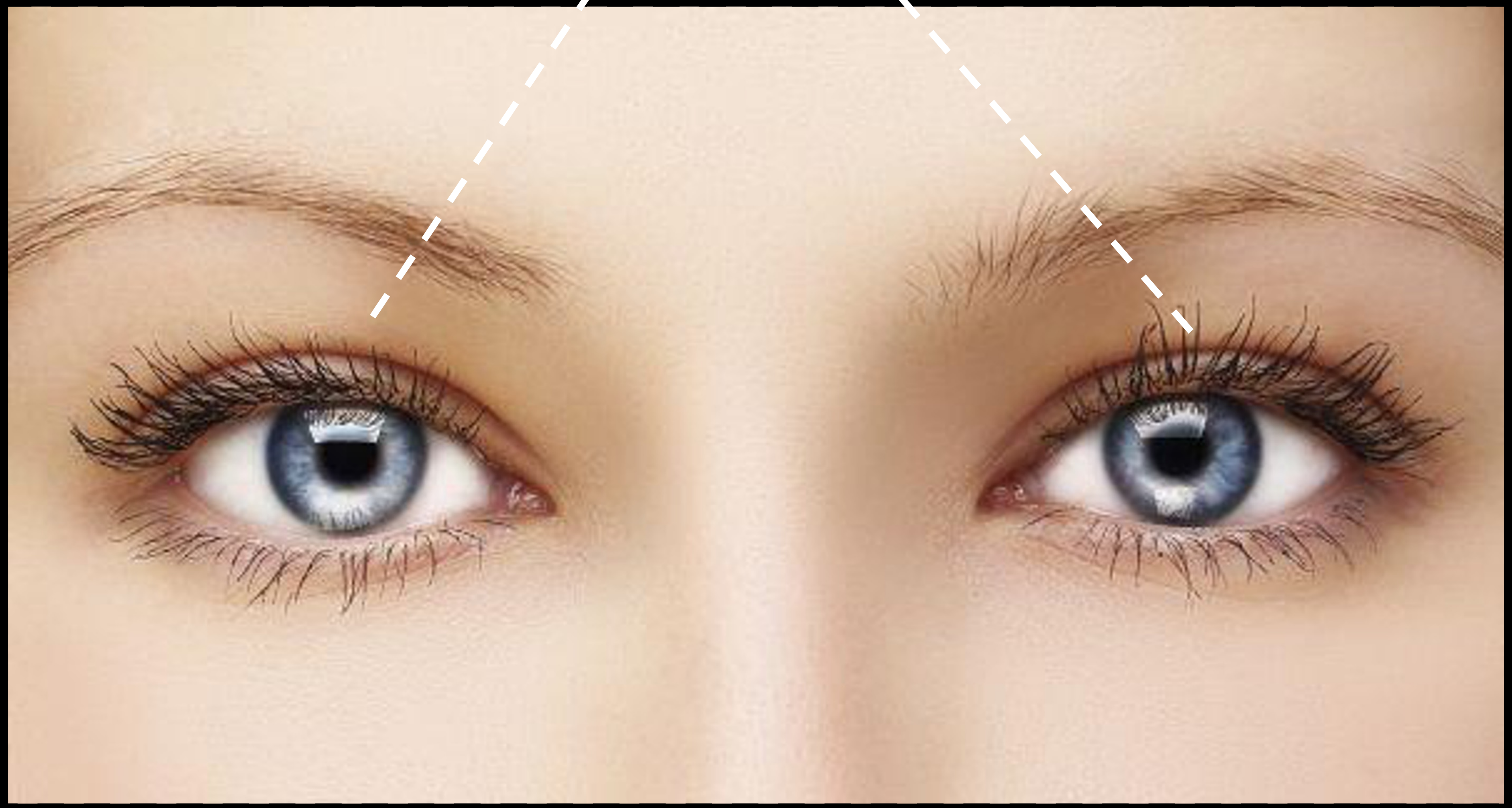
Stereo Photography

Queen Victoria at World Fair, 1851

Stereo Photography

Issue: Narrow Baseline

~6.5 cm



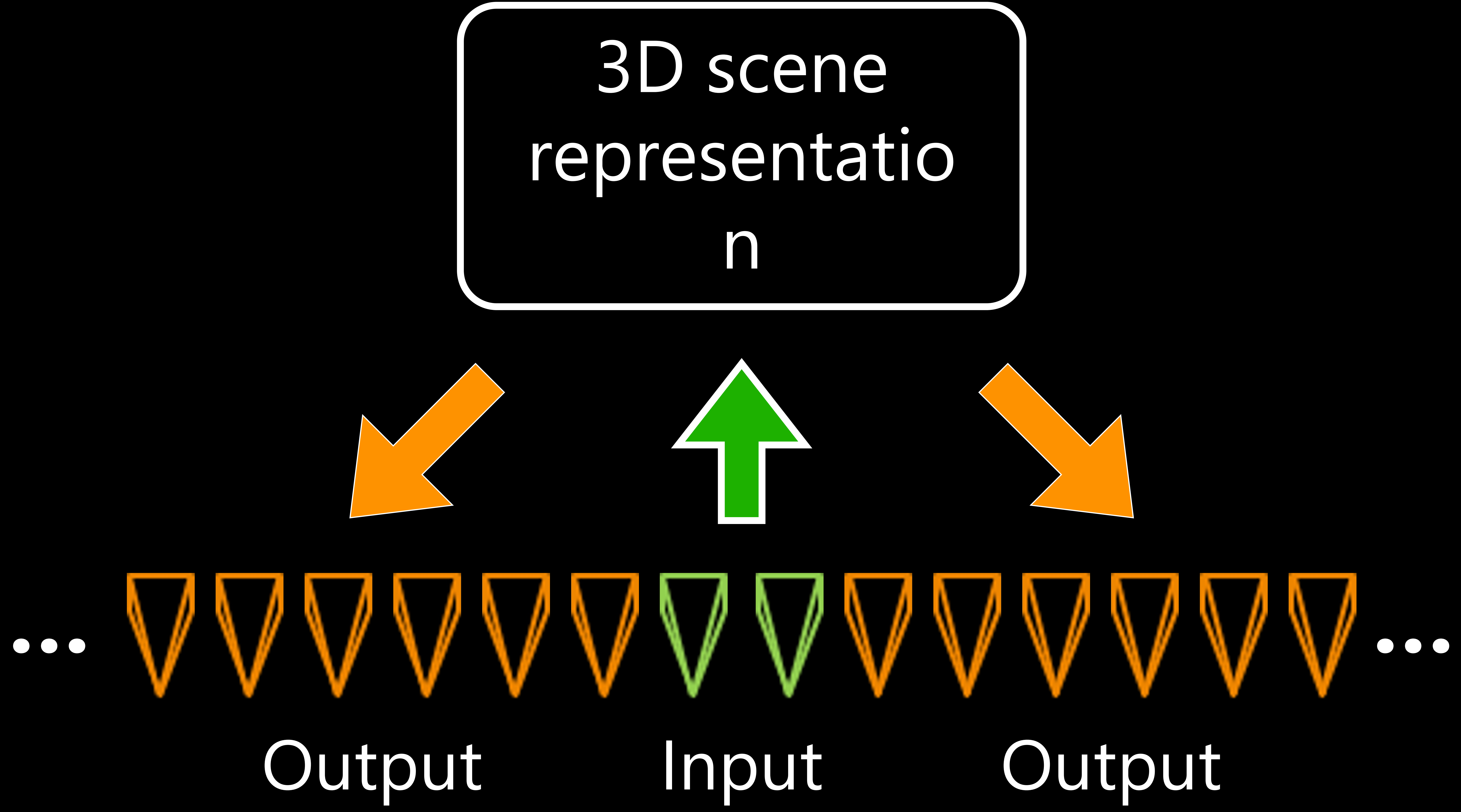
~1.5 cm


Left
MAZZER


Right
MAZZER

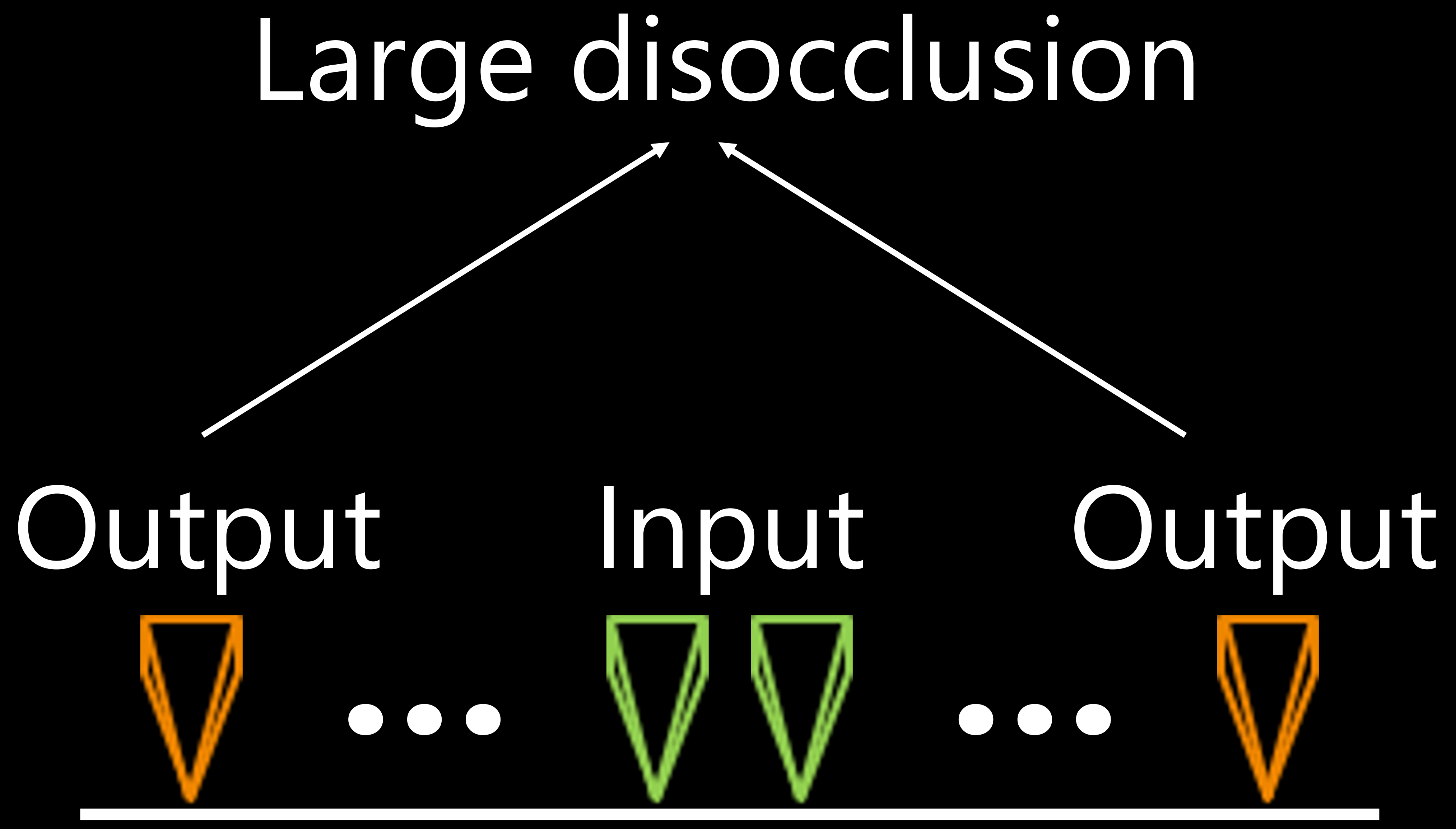
Output

Problem Statement



Challenges

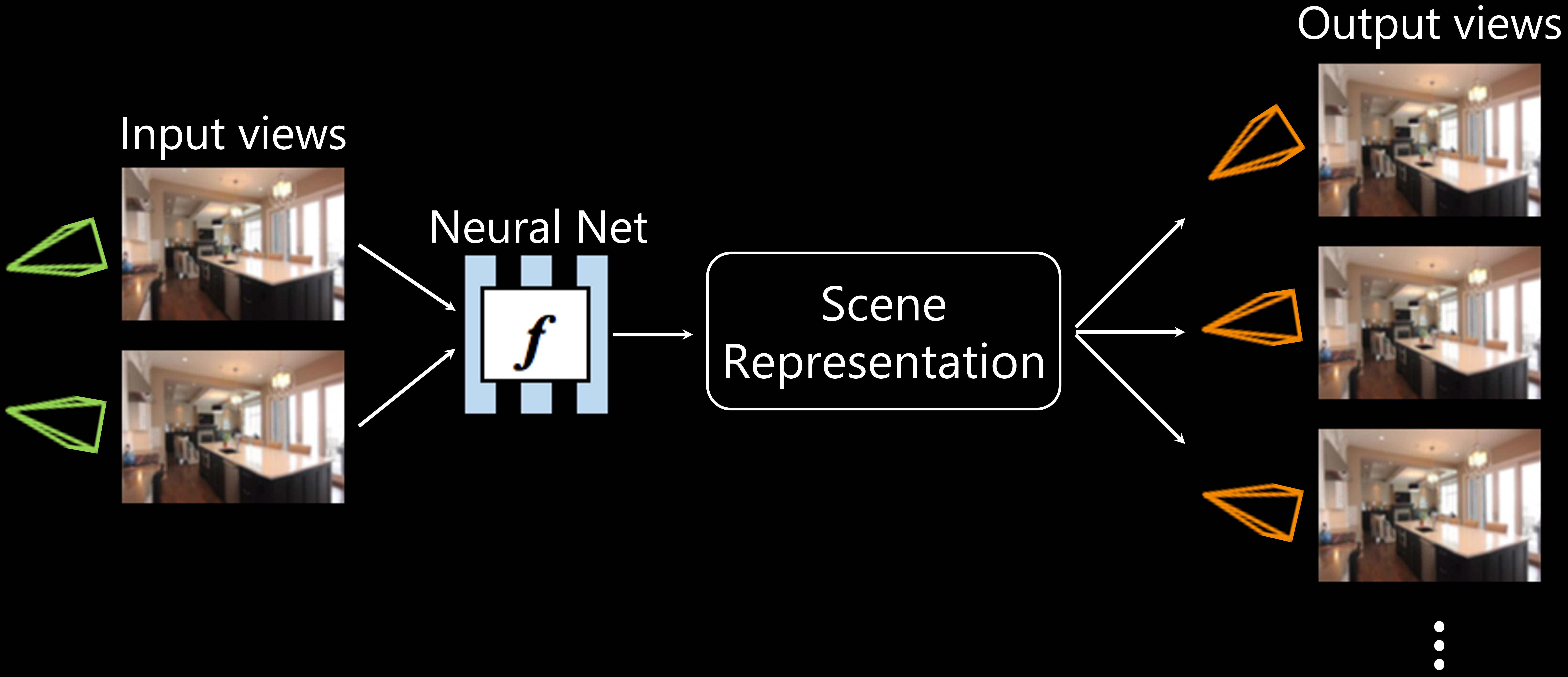
Extrapolation



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)

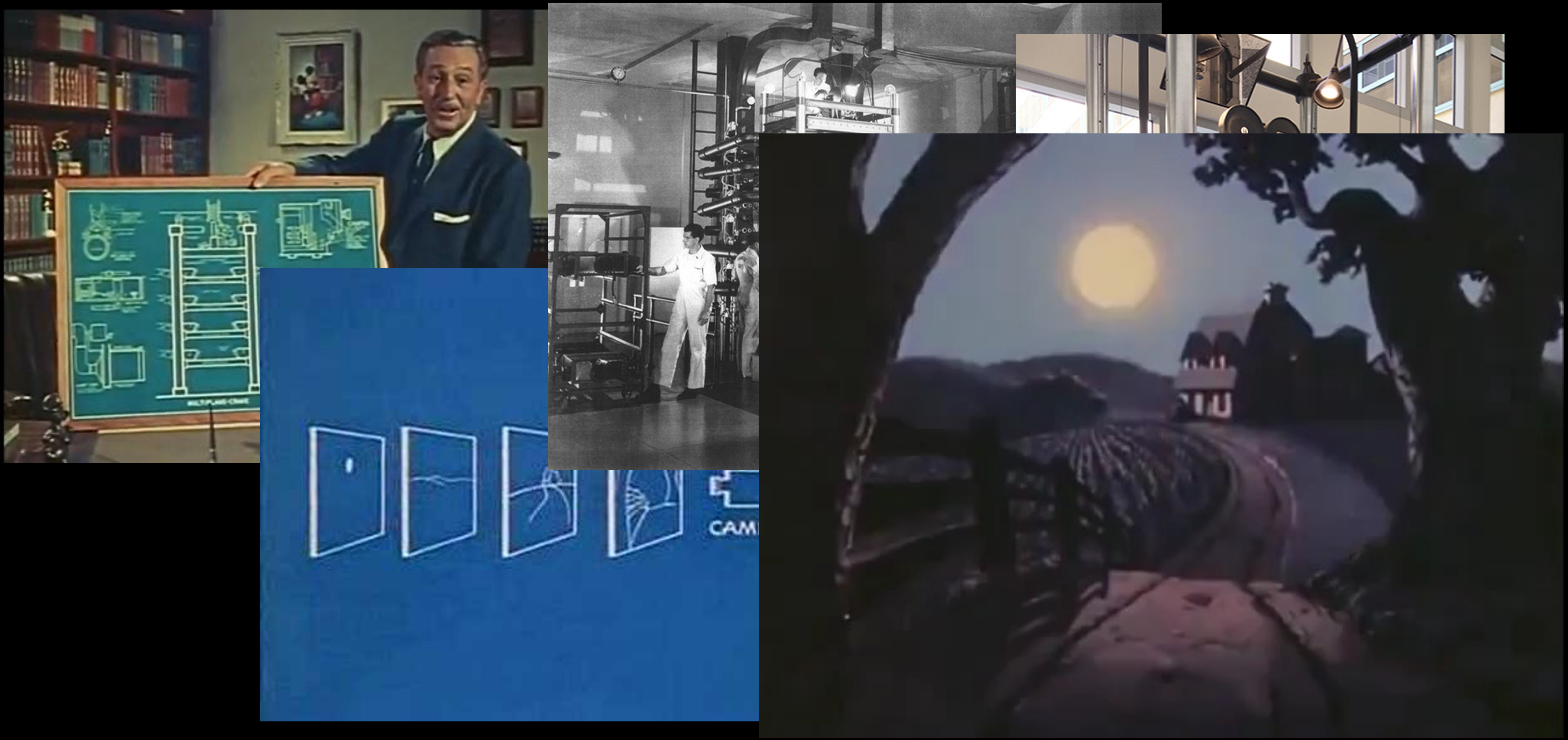
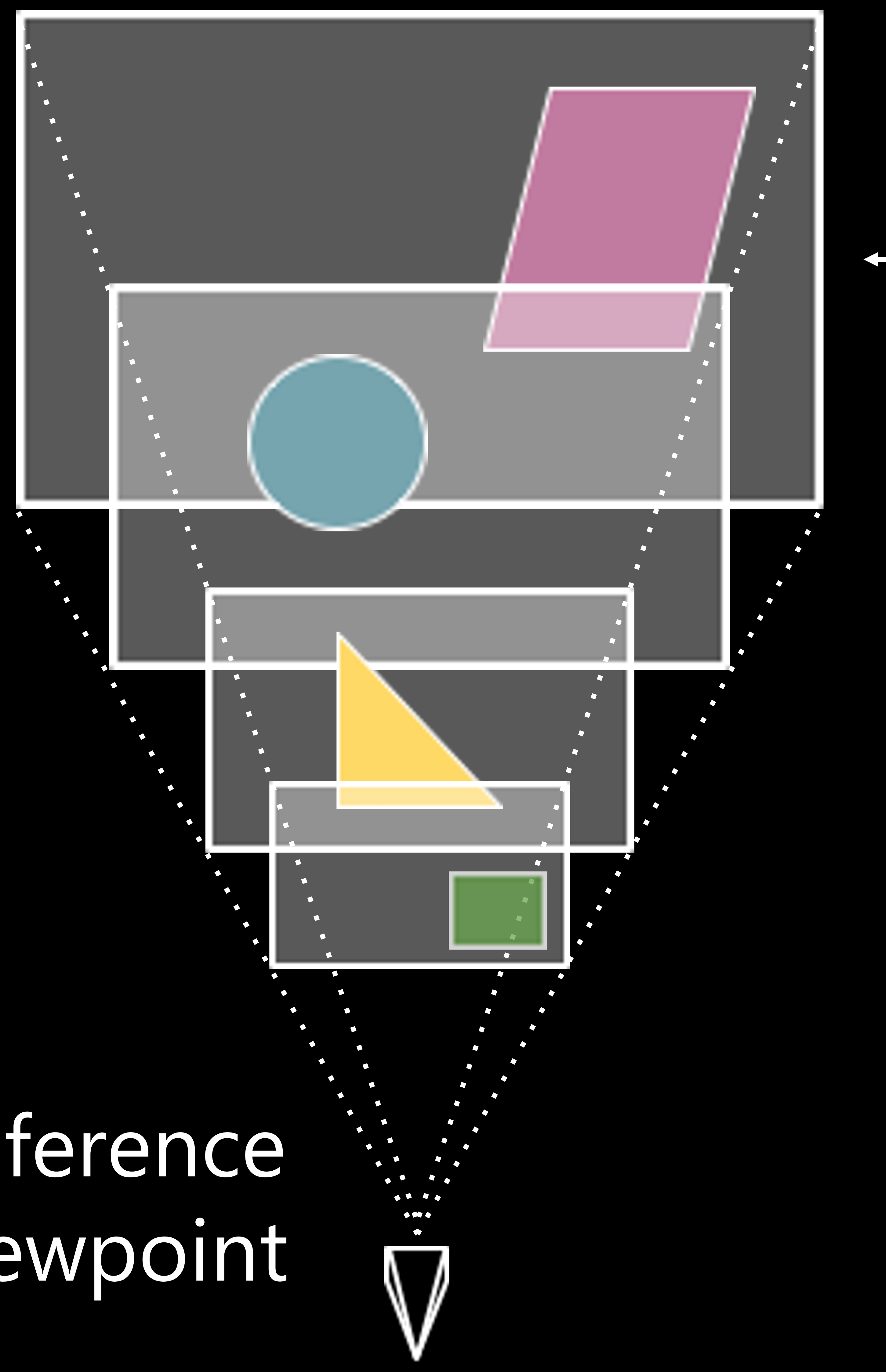


Image credits: Disney https://www.youtube.com/watch?v=kN-eCBAOw60 (from 1957)

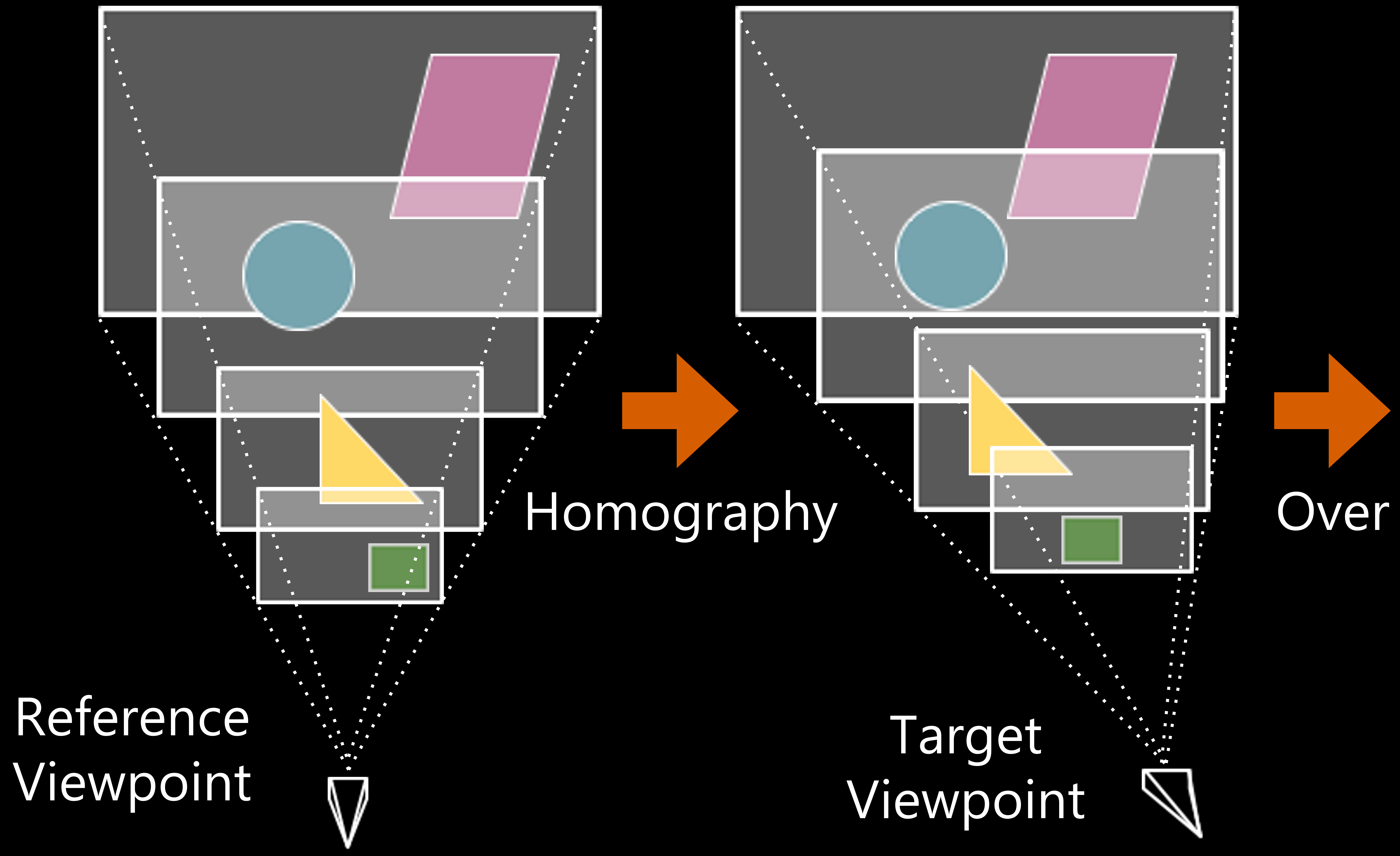
Multiplane Images (MPIs)



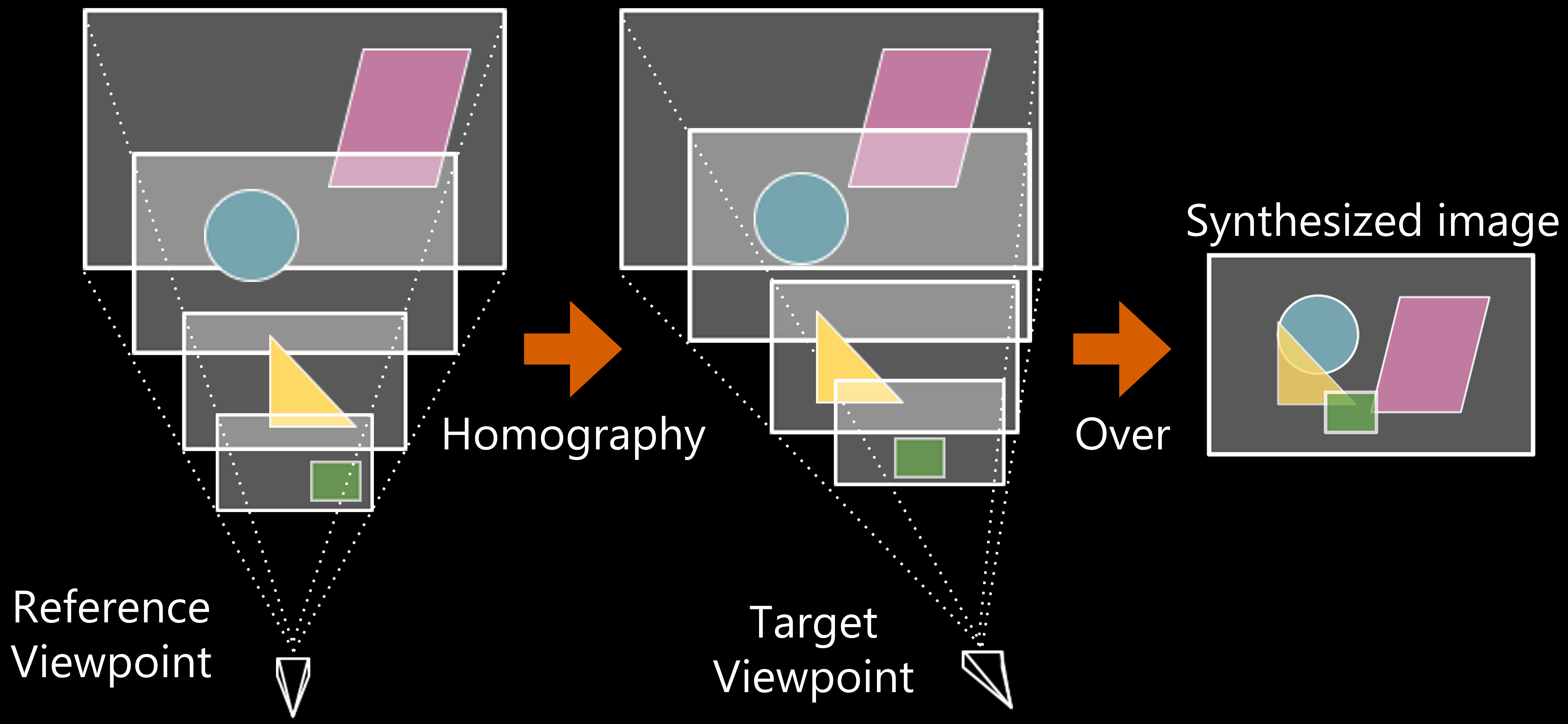
Each plane is at a fixed depth and encoded by an RGBA image

Reference Viewpoint

View Synthesis using Multiplane Images

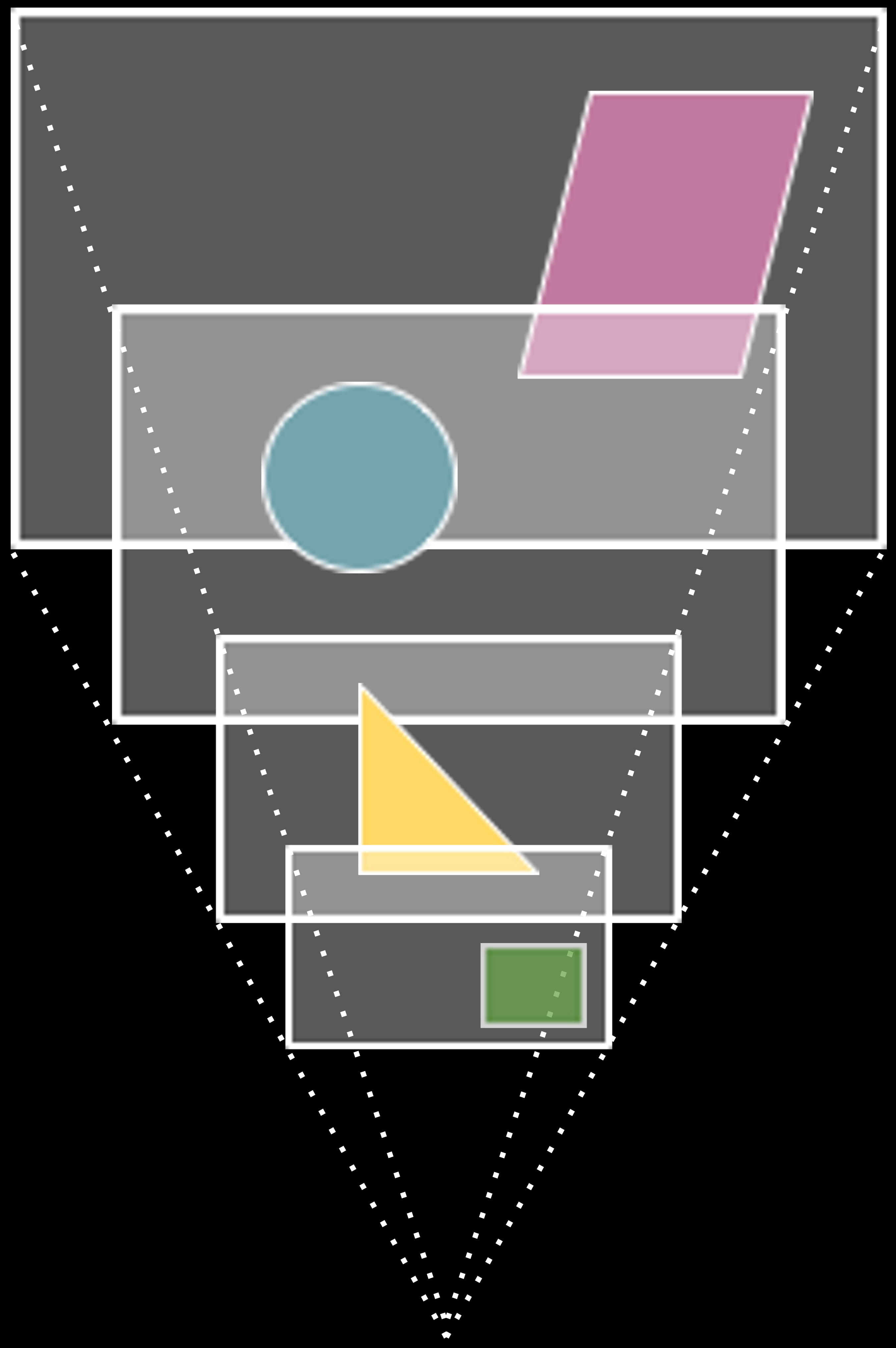


View Synthesis using Multiplane Images



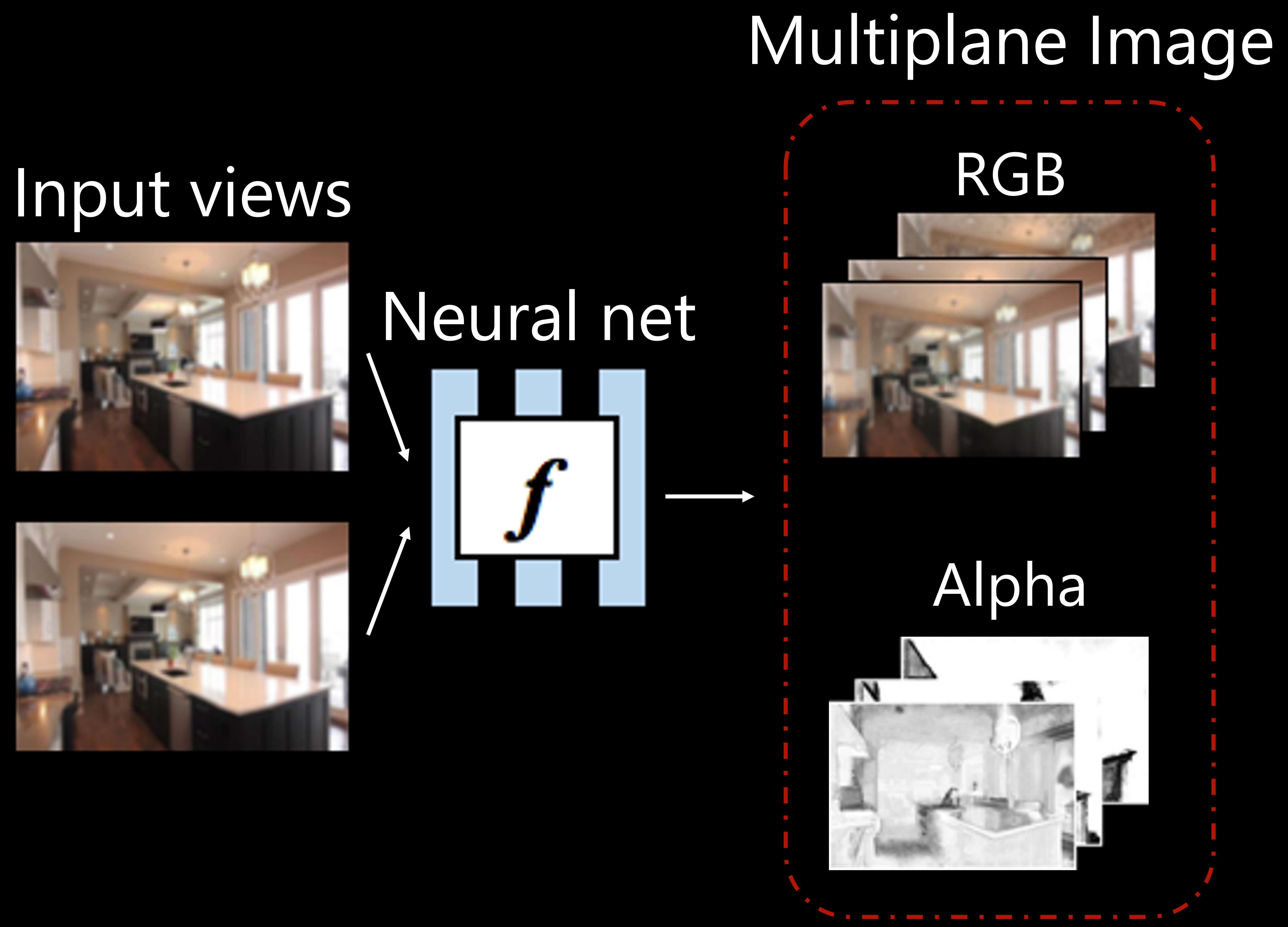

MAZZER

Properties of Multiplane Images

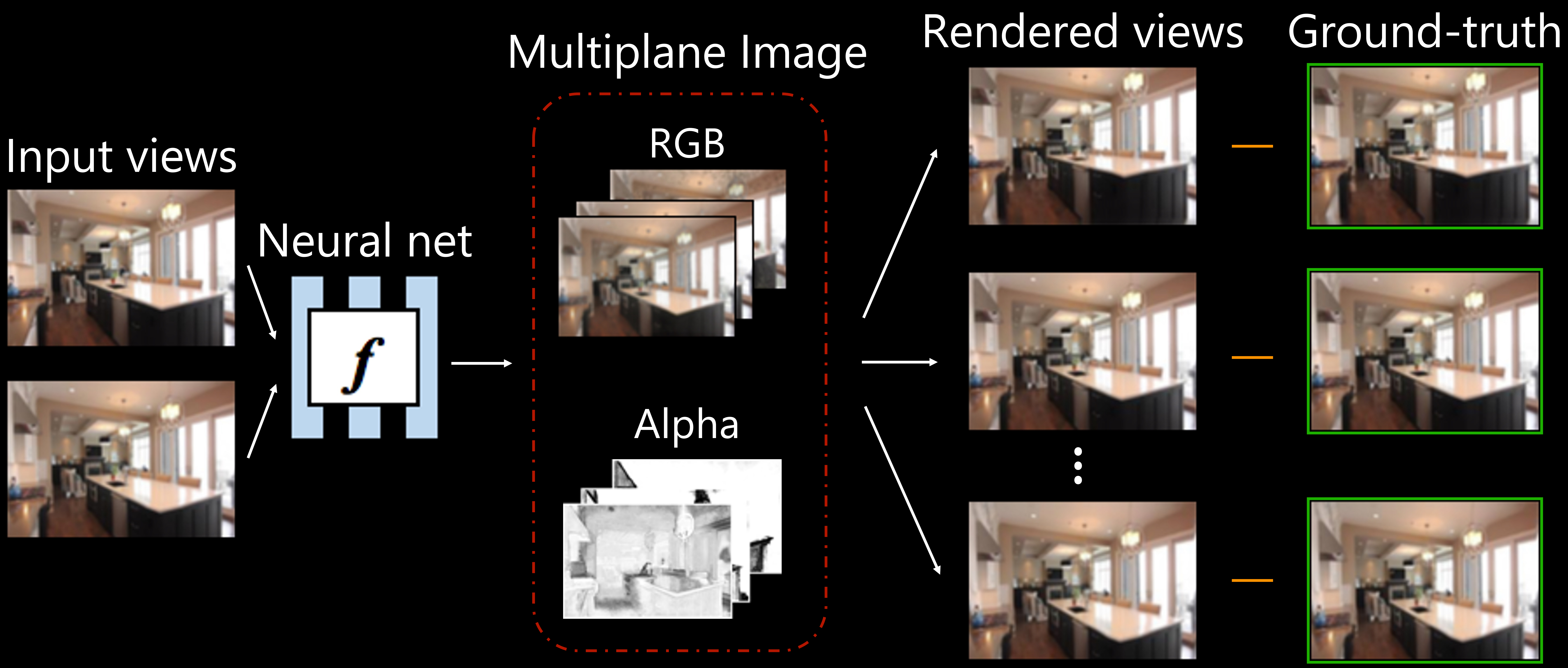


- Models disocclusion
- Models soft edges and non-Lambertian effects
- Efficient for view synthesis
- Differentiable rendering

Learning Multiplane Images

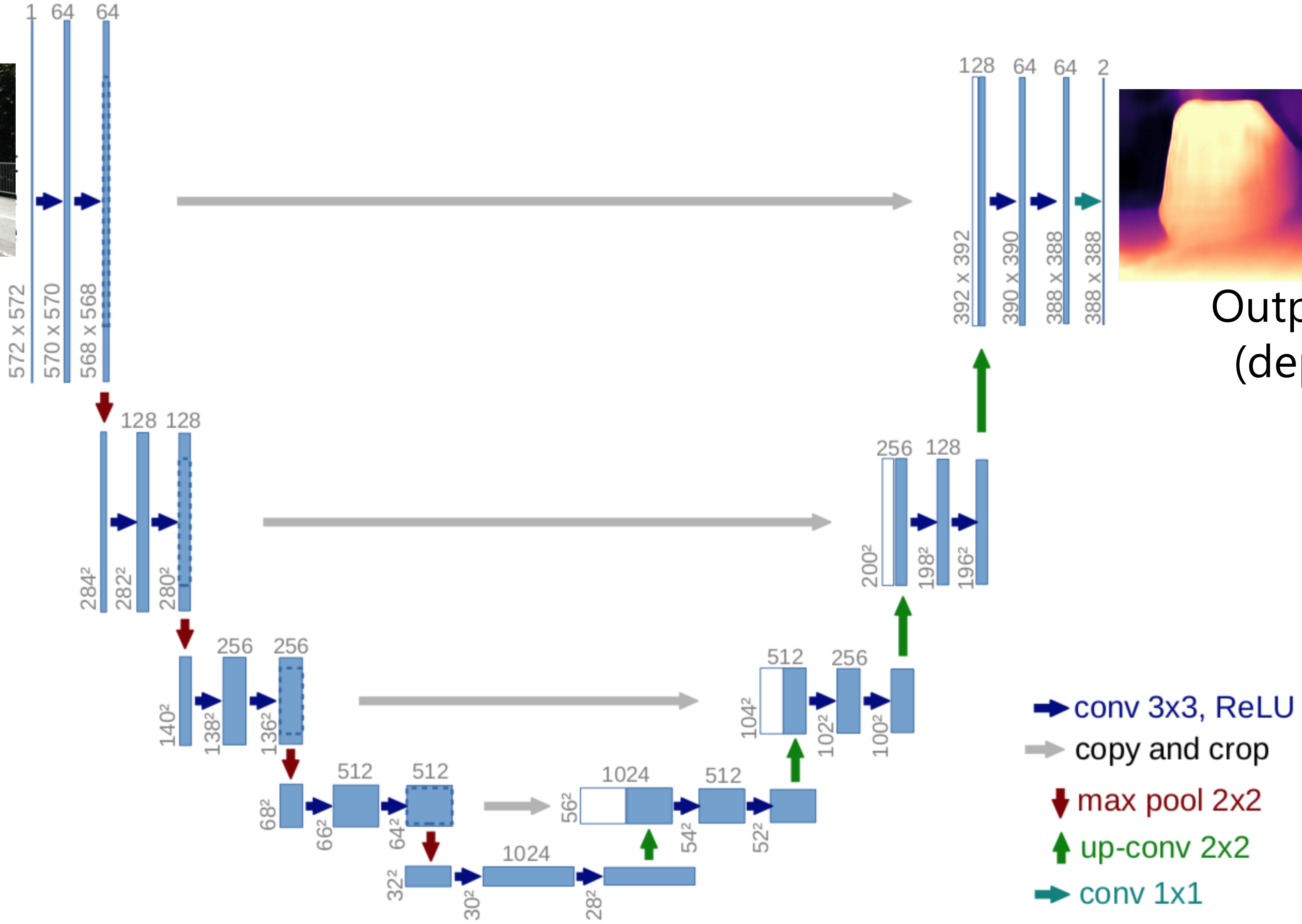


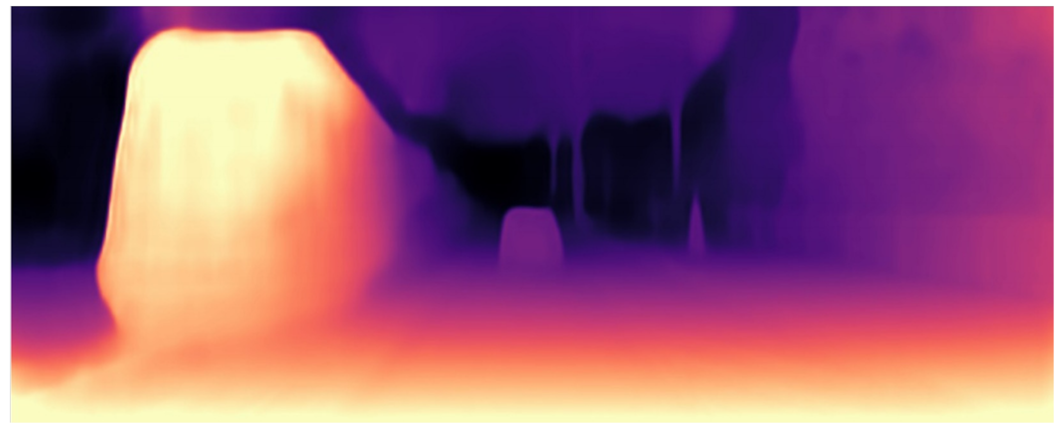
Learning Multiplane Images



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

Input RGB Image





Output image
(depth map)

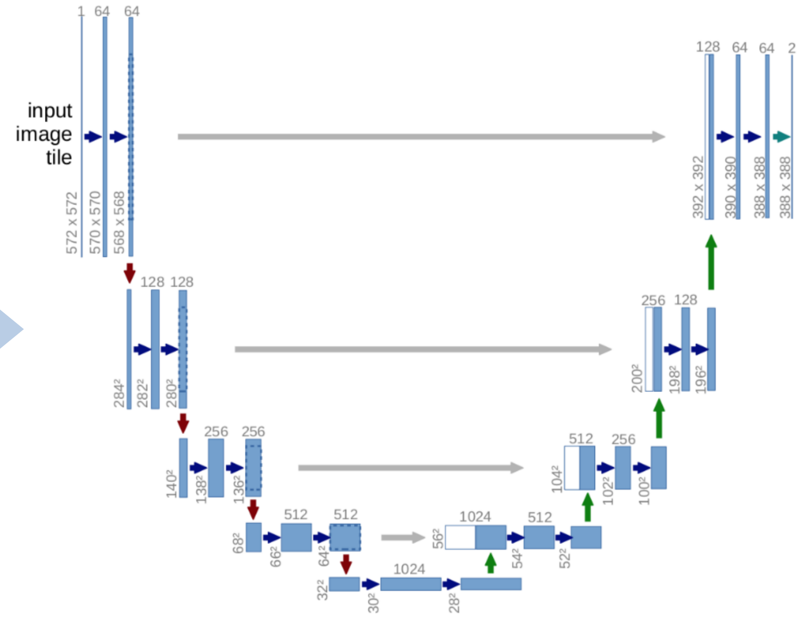
Image Pair → Multiplane Image

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

Input pair

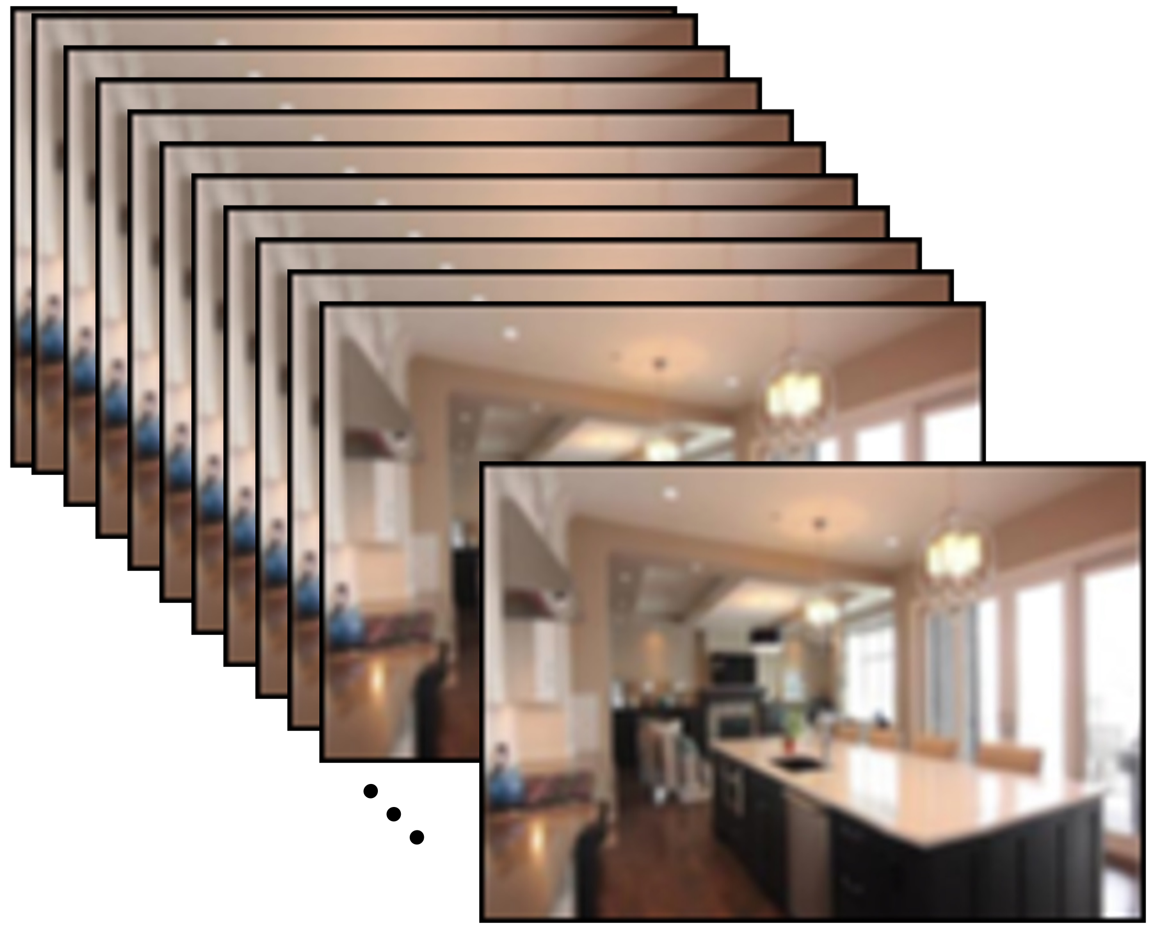
Plane sweep volume

(32 RGB planes)



U-Net CNN

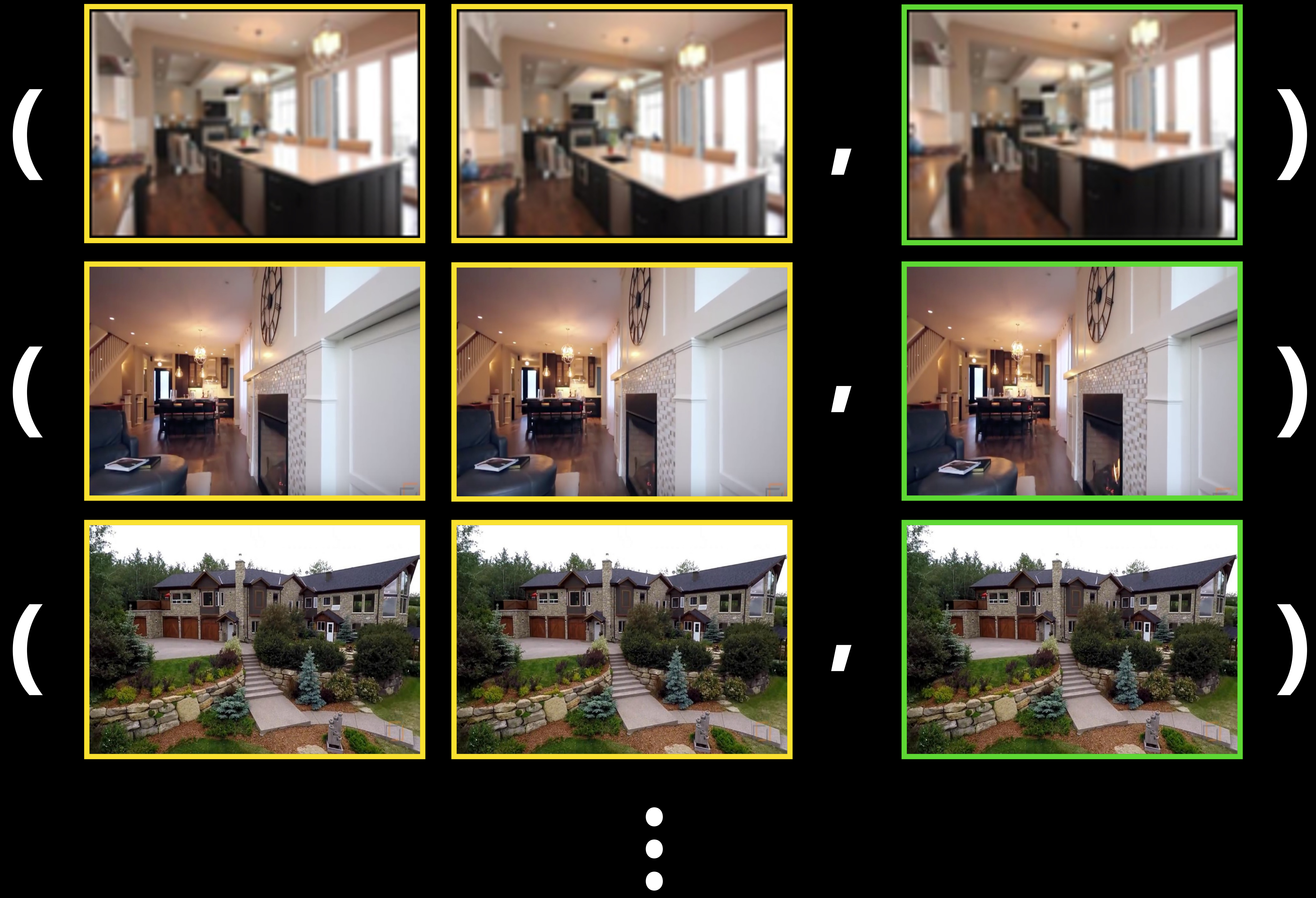
MPI Planes



(32 RGBA images)

Training Data

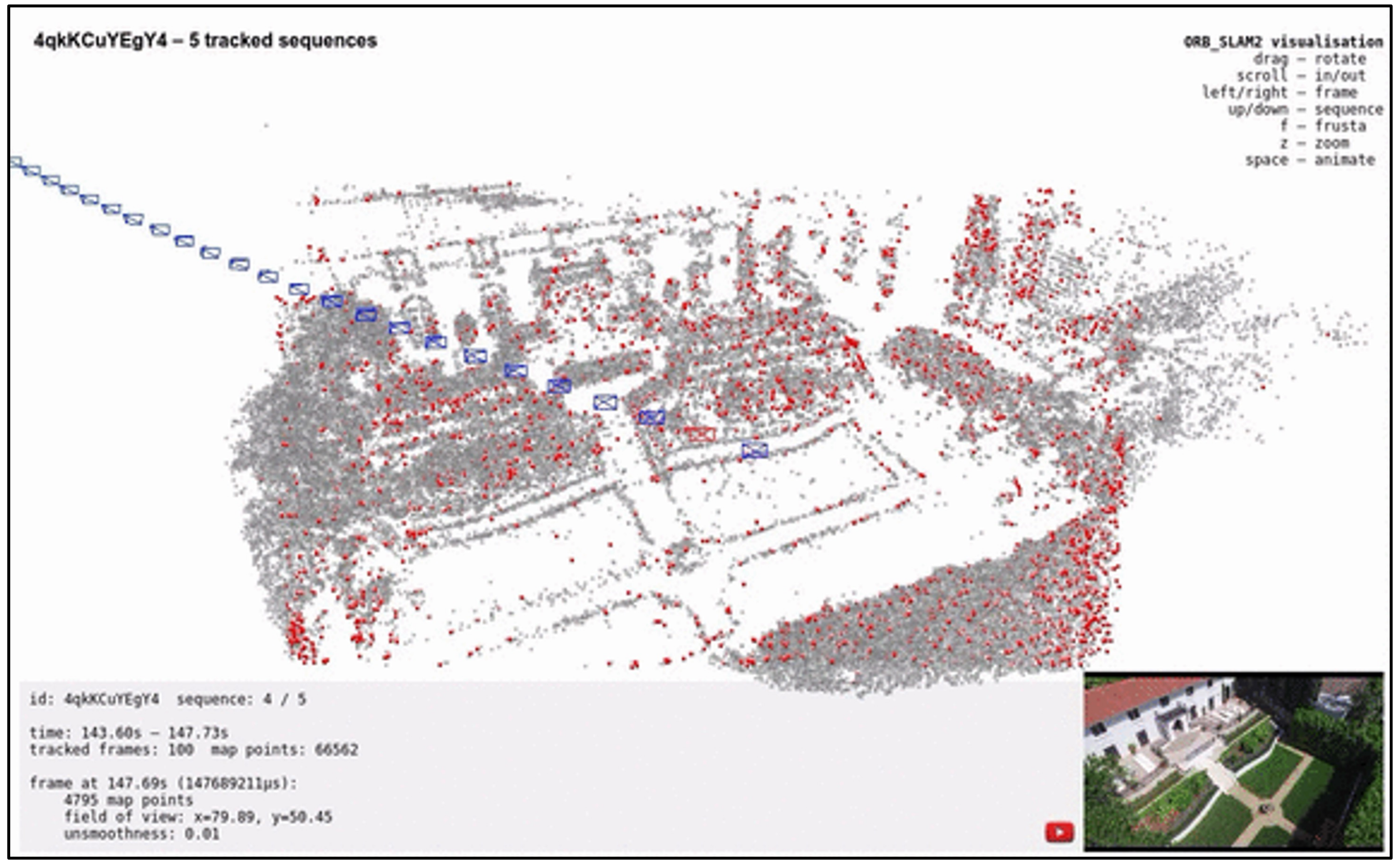
Input views   Target view



Need massive set of triplets with known camera poses

RealEstate10K

SLAM



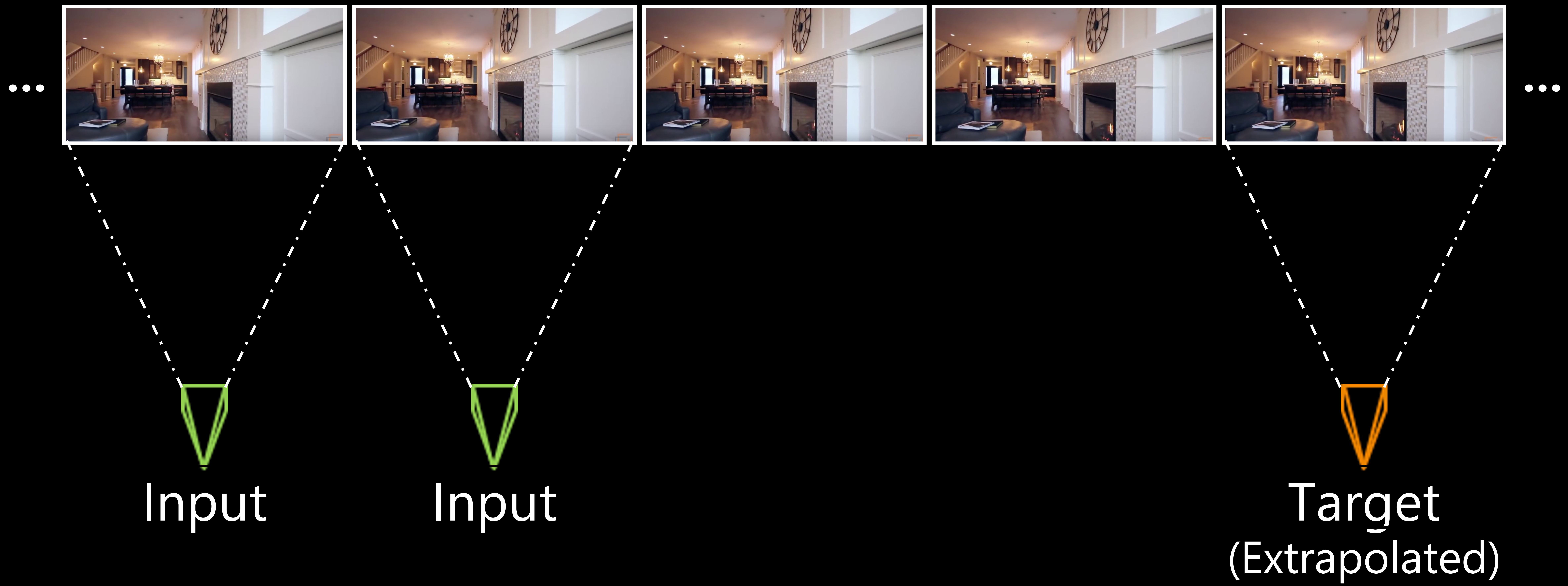
Running SLAM / SfM on YouTube videos at scale

RealEstate10K dataset

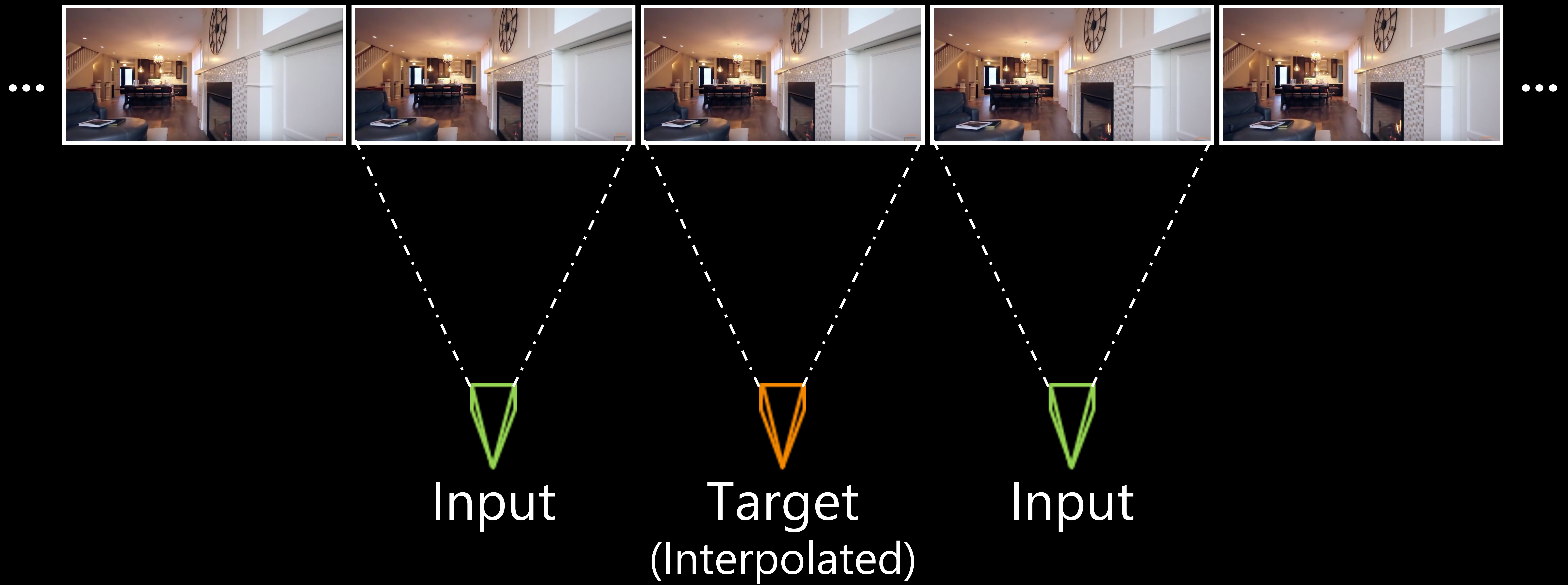
10 million frames from **80,000 video clips** from **10,000 videos**

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

Sampling Training Examples



Sampling Training Examples



Results


Left

Right

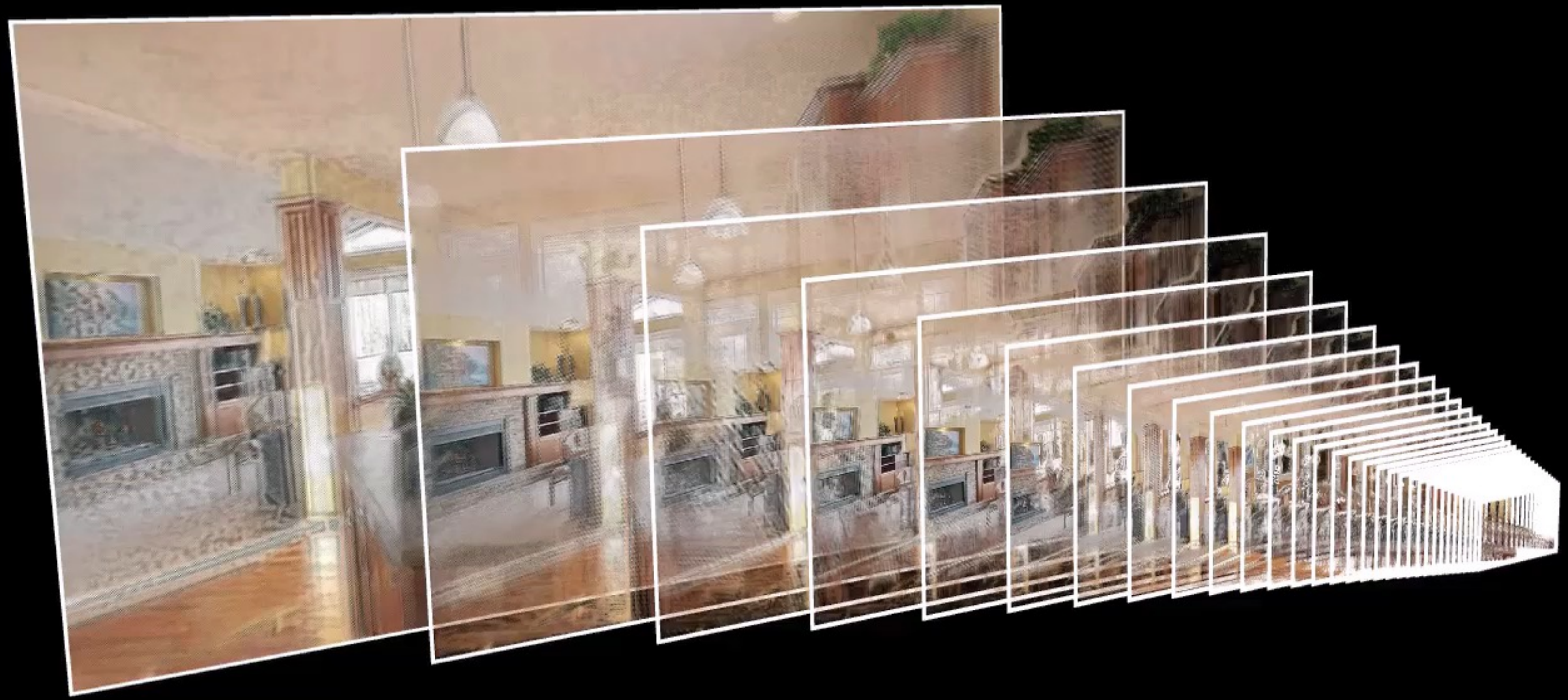
Output

Image 1

Image 2

Multi-plane Image (MPI)

Reference input view



Computer vision as inverse rendering

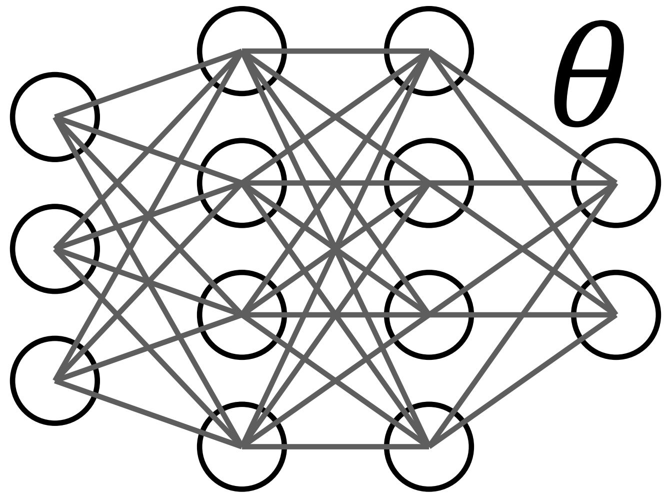
Inverse Rendering

3D
Scene
Representation

Rendering

Adapted from material from Pratul Srinivas

Paradigm 1: "Feedforward" inverse rendering



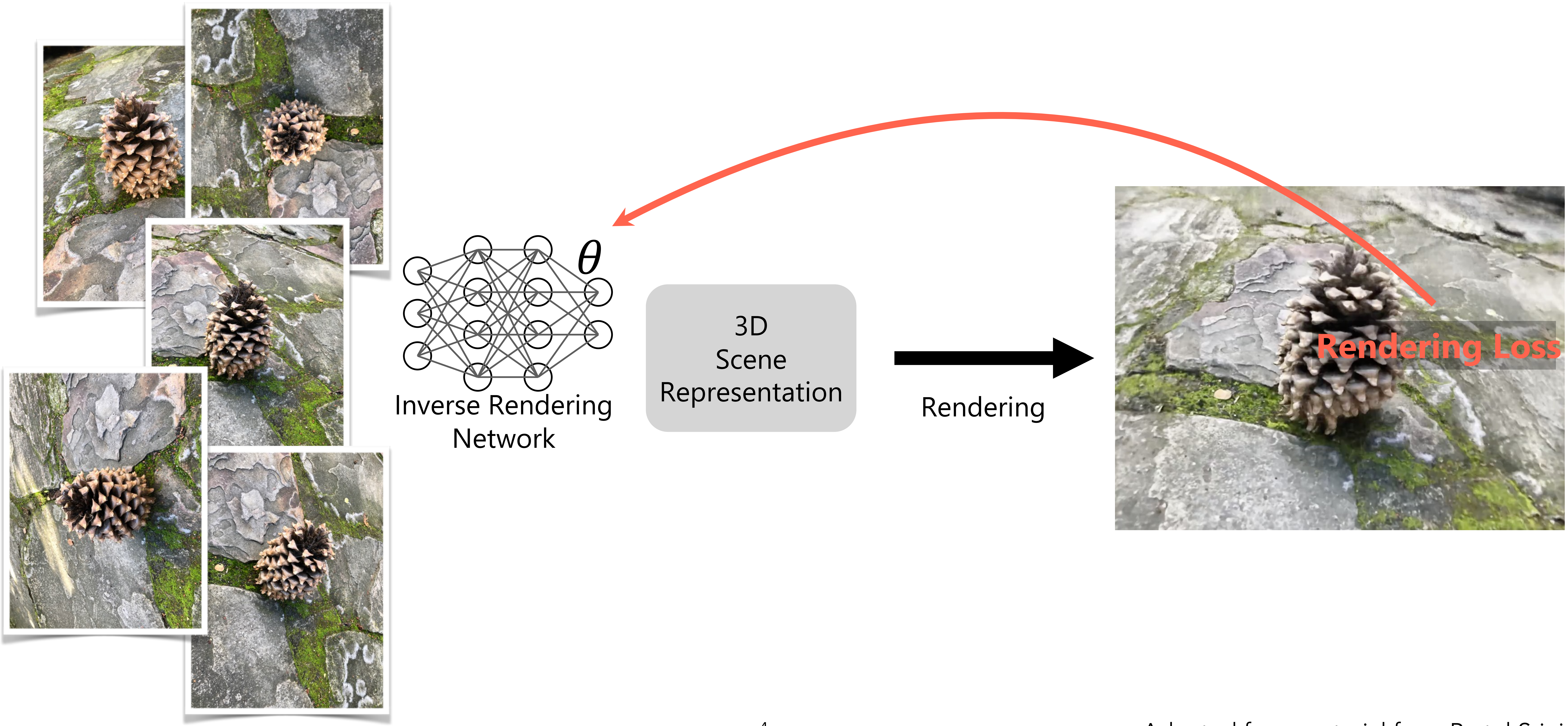
Inverse Rendering Network

3D
Scene
Representation

Rendering

Adapted from material from Pratul Srinivas

Paradigm 1: “Feedforward” inverse rendering



Adapted from material from Pratul Srinivas

Paradigm 2: “Render-and-compare”

Inverse Rendering

3D Scene Representation θ

Rendering

Adapted from material from Pratul Srinivas

Paradigm 2: “Render-and-compare”

Inverse Rendering

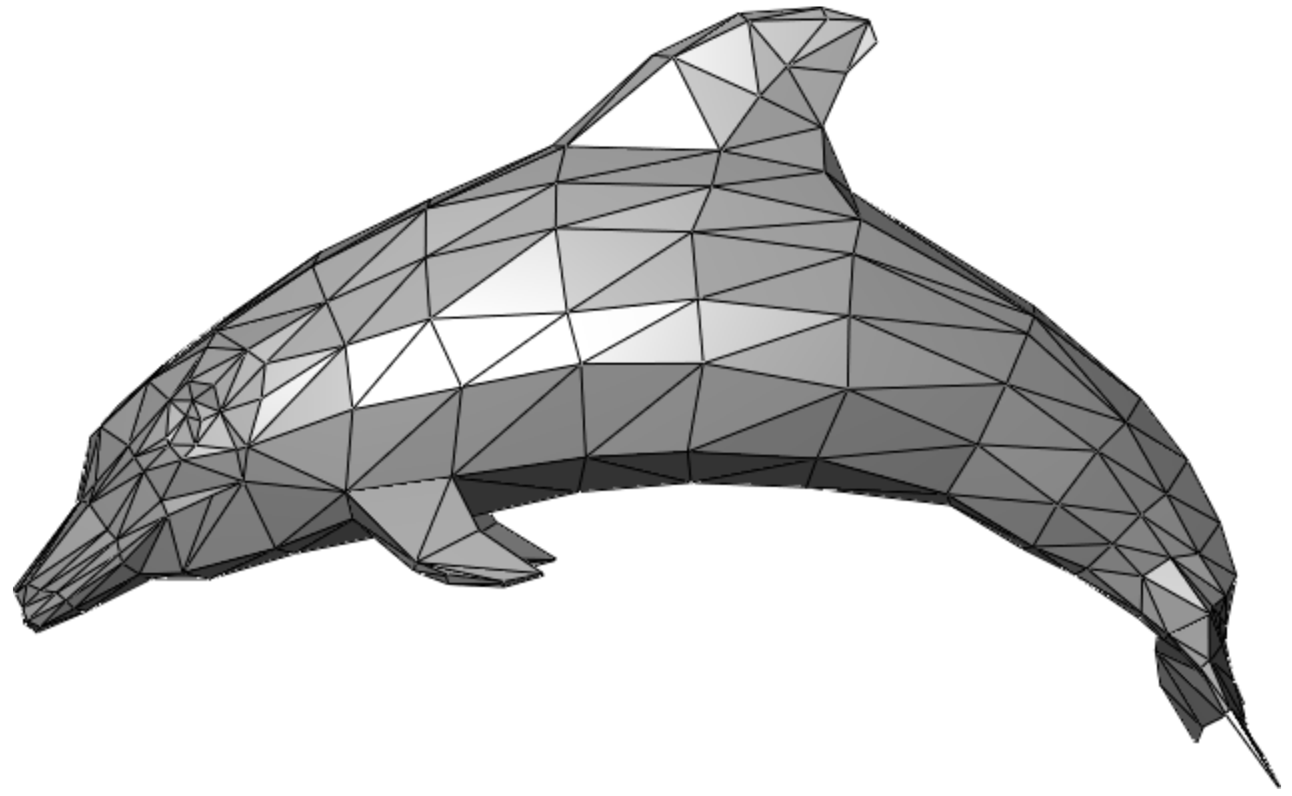
3D
Scene
Representation θ

Rendering

Adapted from material from Pratul Srinivas

What representation to use?

- Could use triangle meshes, but hard to differentiate during rendering

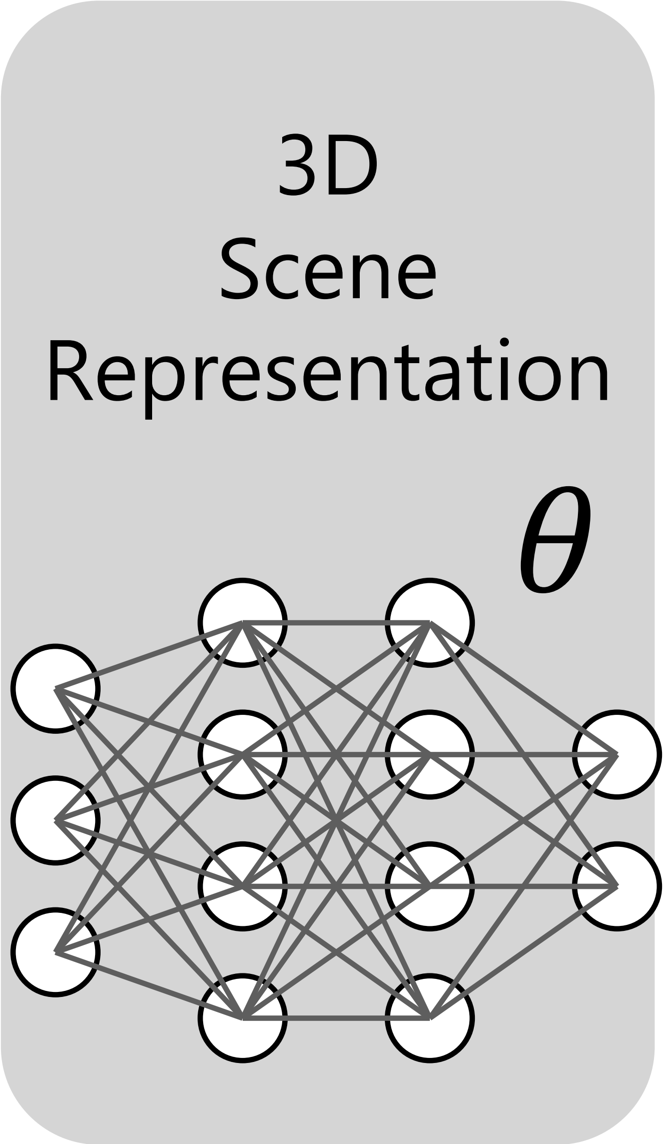


- Multiplane images (MPIs) are easy to differentiate, but only allow for rendering a small range of views

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

Paradigm 2: “Render-and-compare”

Inverse Rendering



Rendering

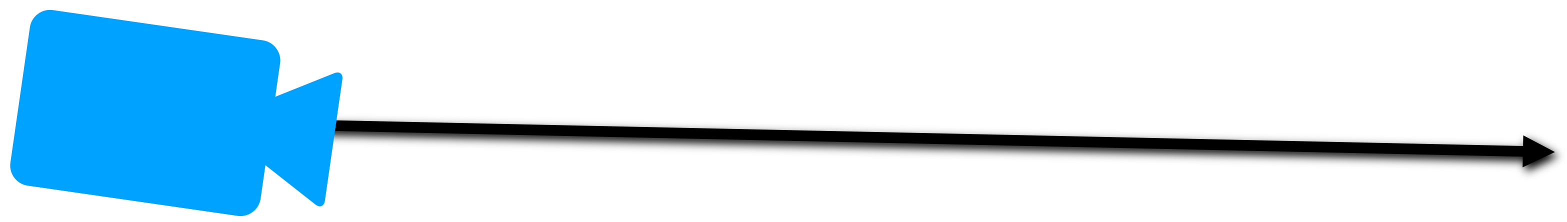
Adapted from material from Pratul Srinivas

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?

Adapted from material from Pratul Srinivas

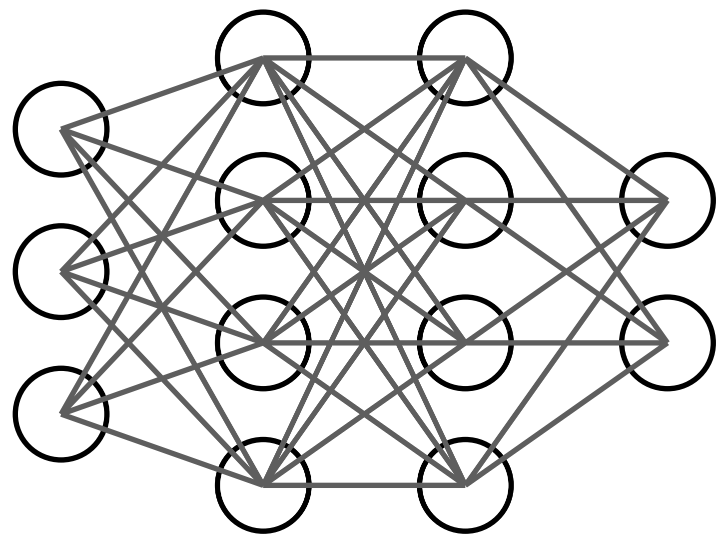
Neural Volumetric Rendering

continuous, differentiable
rendering model without
concrete ray/surface intersections

Adapted from material from Pratul Srinivas

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)

Adapted from material from Pratul Srinivas

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

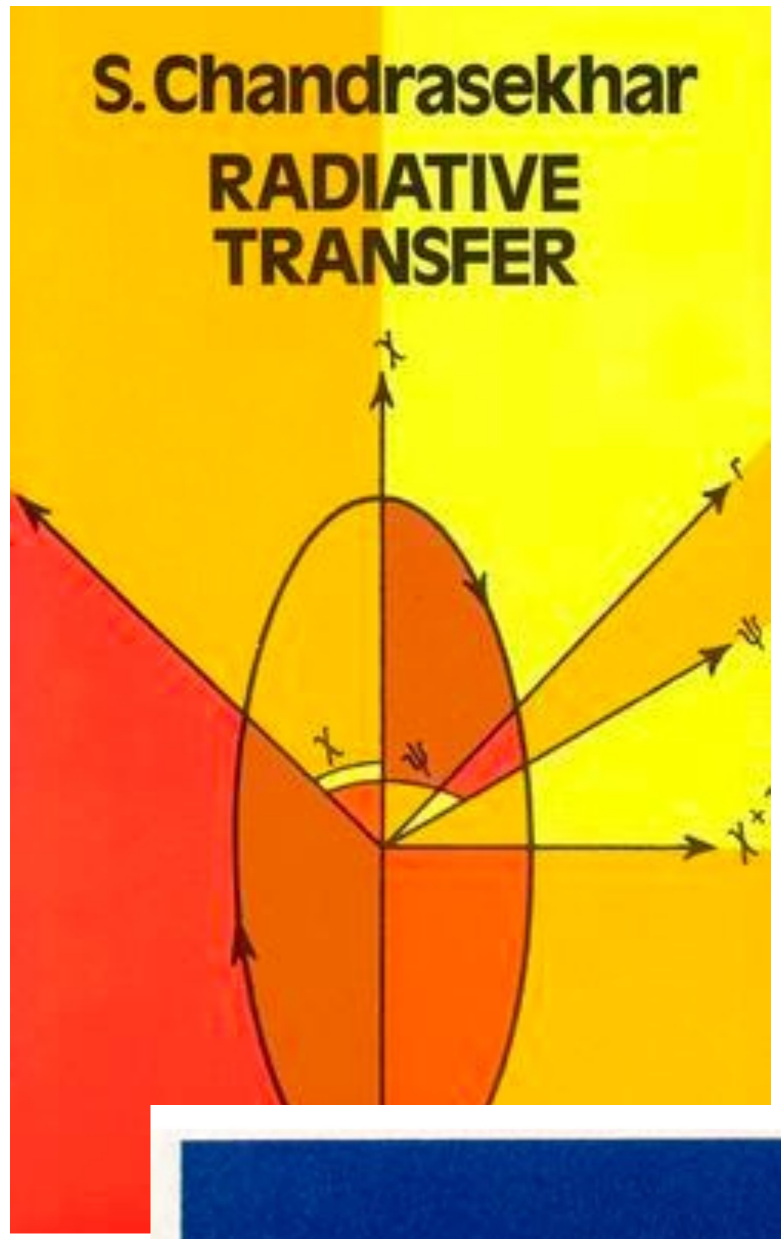
3D reconstruction viewable from any angle

NeRF Overview

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

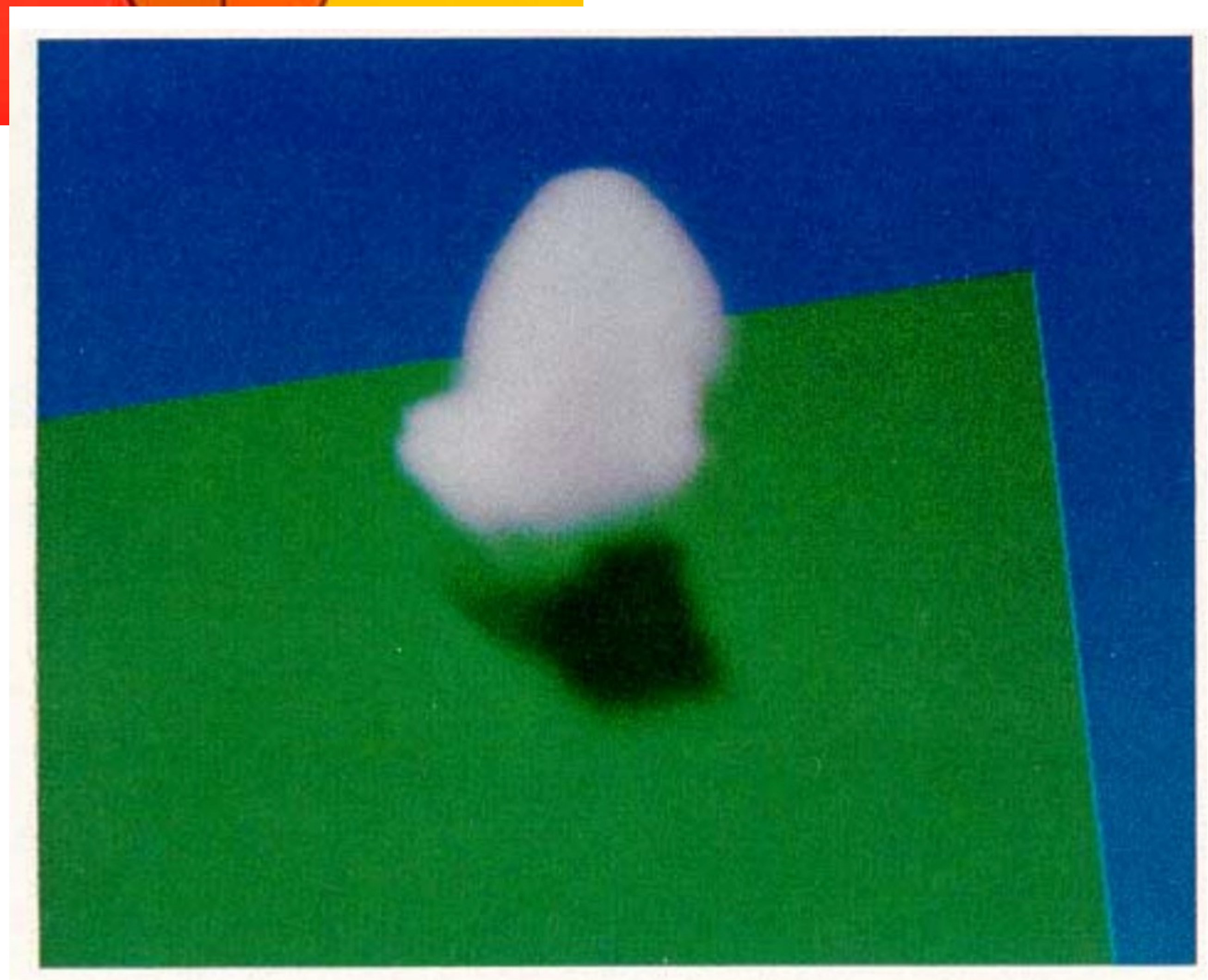
NeRF Overview

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



Traditional volumetric rendering

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

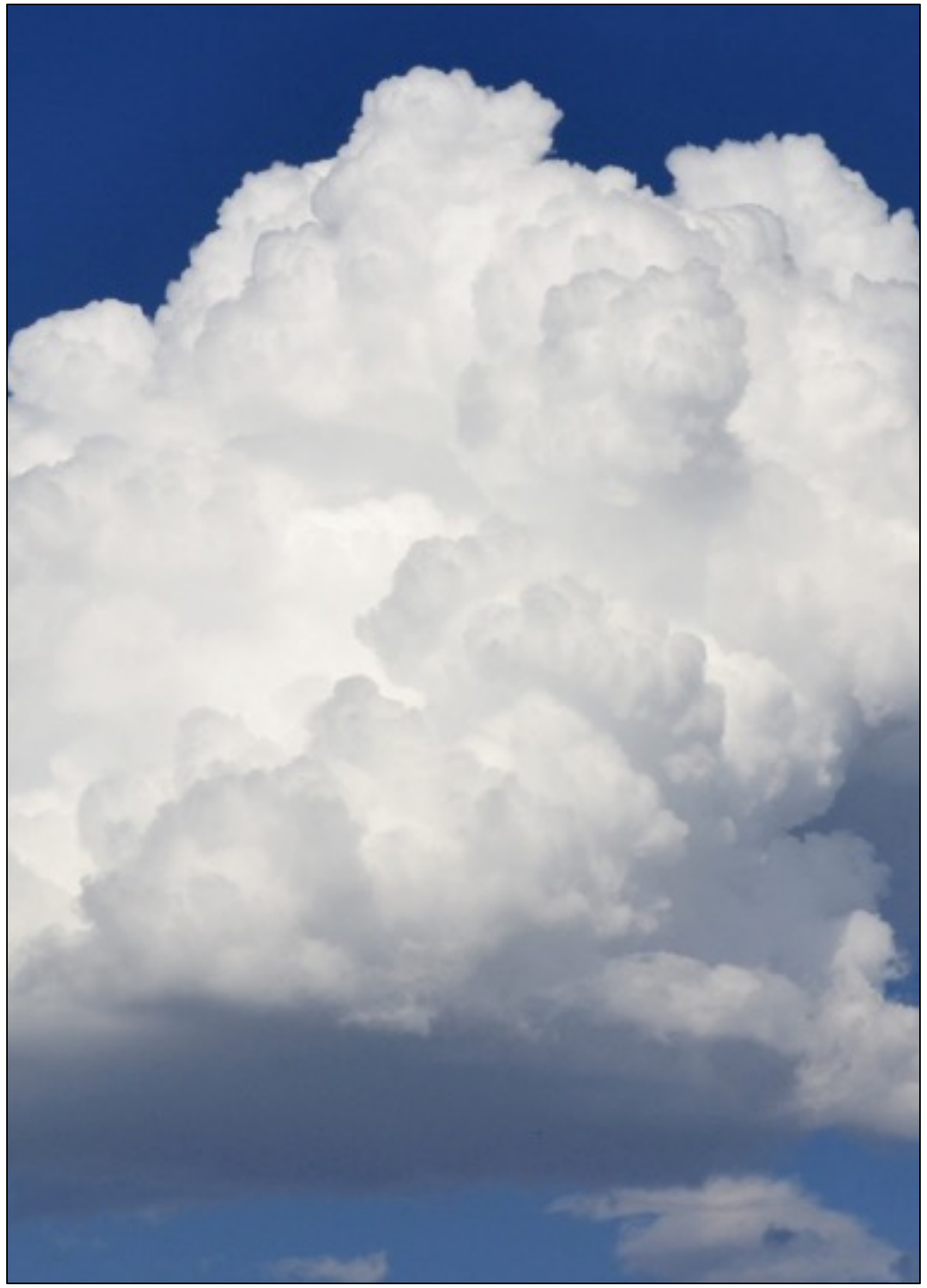
Adapted from material from Pratul Srinivas

Full volumetric rendering formulation

Absorption

http://commons.wikimedia.org

Scattering



http://coclouds.com

Emission

http://wikipedia.org

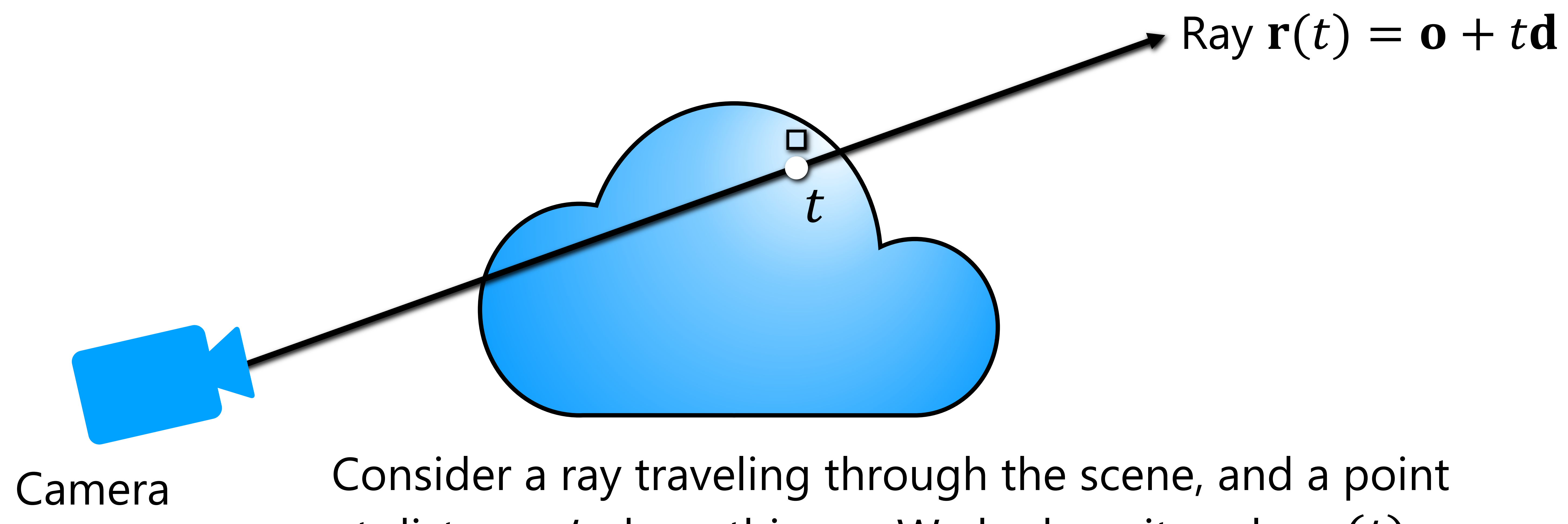
Slide credit: Novak et al 2018, Monte Carlo methods for physically based volume rendering

Volumetric formulation for NeRF

Scene is a cloud of colored fog

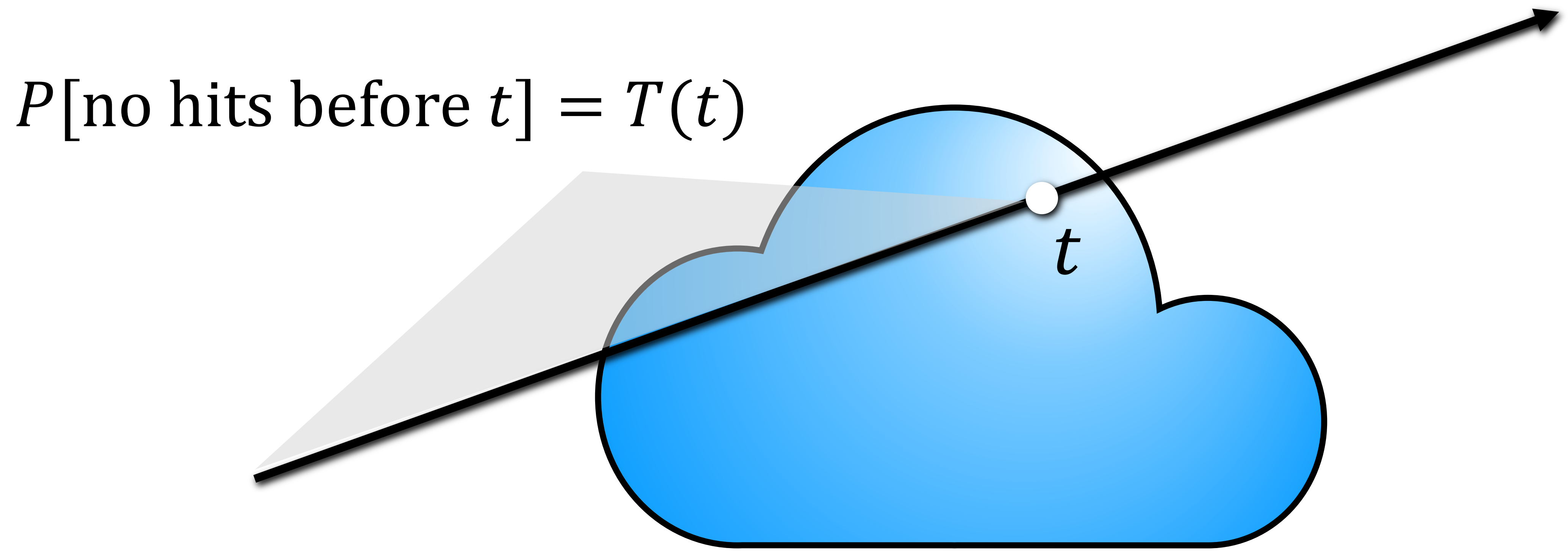
Max and Chen 2010, Local and Global Illumination in the Volume Rendering Integral

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) $\alpha(t)$ from a neural network

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"

Adapted from material from Pratul Srinivas

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

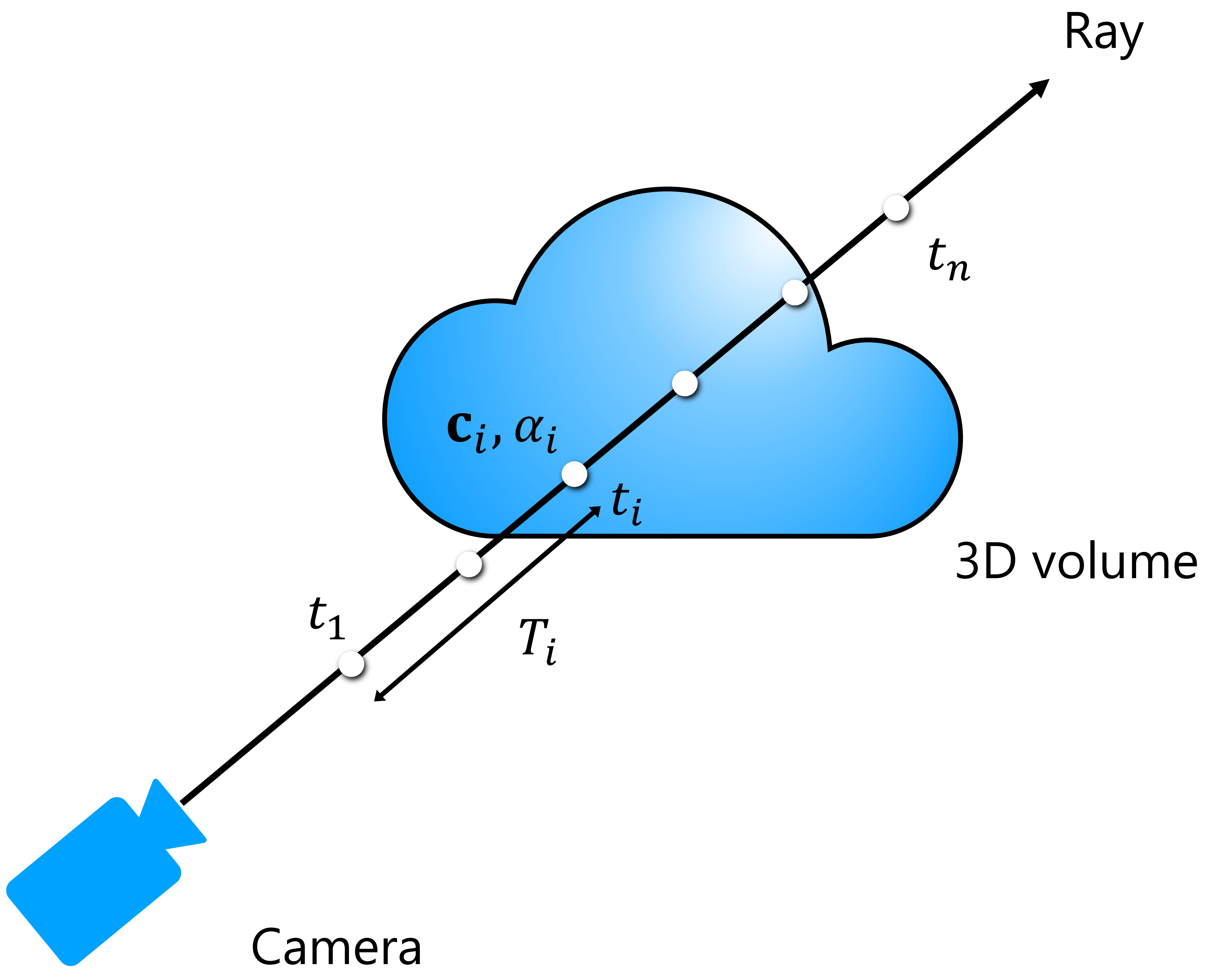
weights

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



Adapted from material from Pratul Srinivas

Volume rendering estimation: integrating color along a ray

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

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

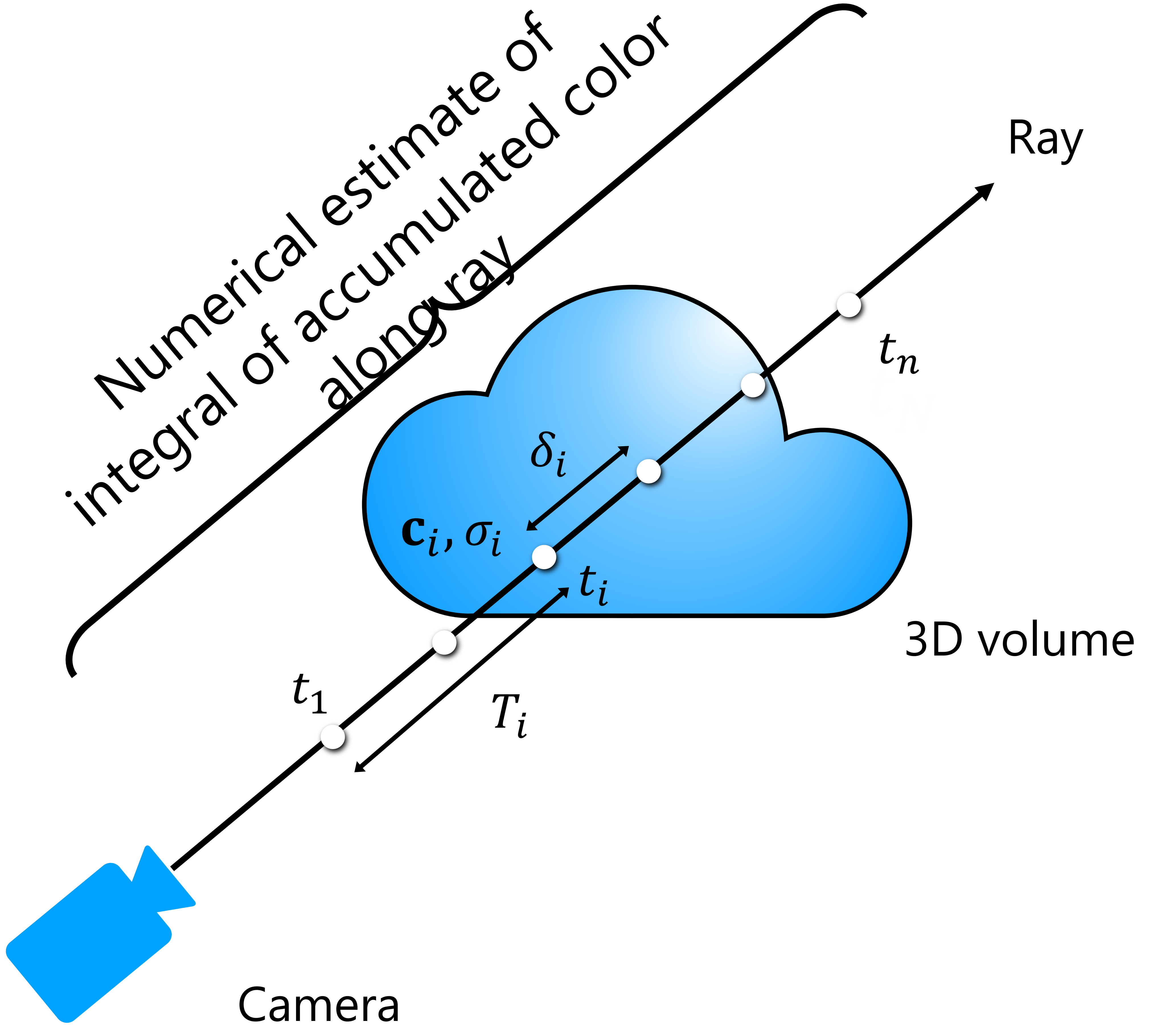
final rendered color along ray

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):

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

Adapted from material from Pratul Srinivas

Volume rendering estimation: integrating color along a ray

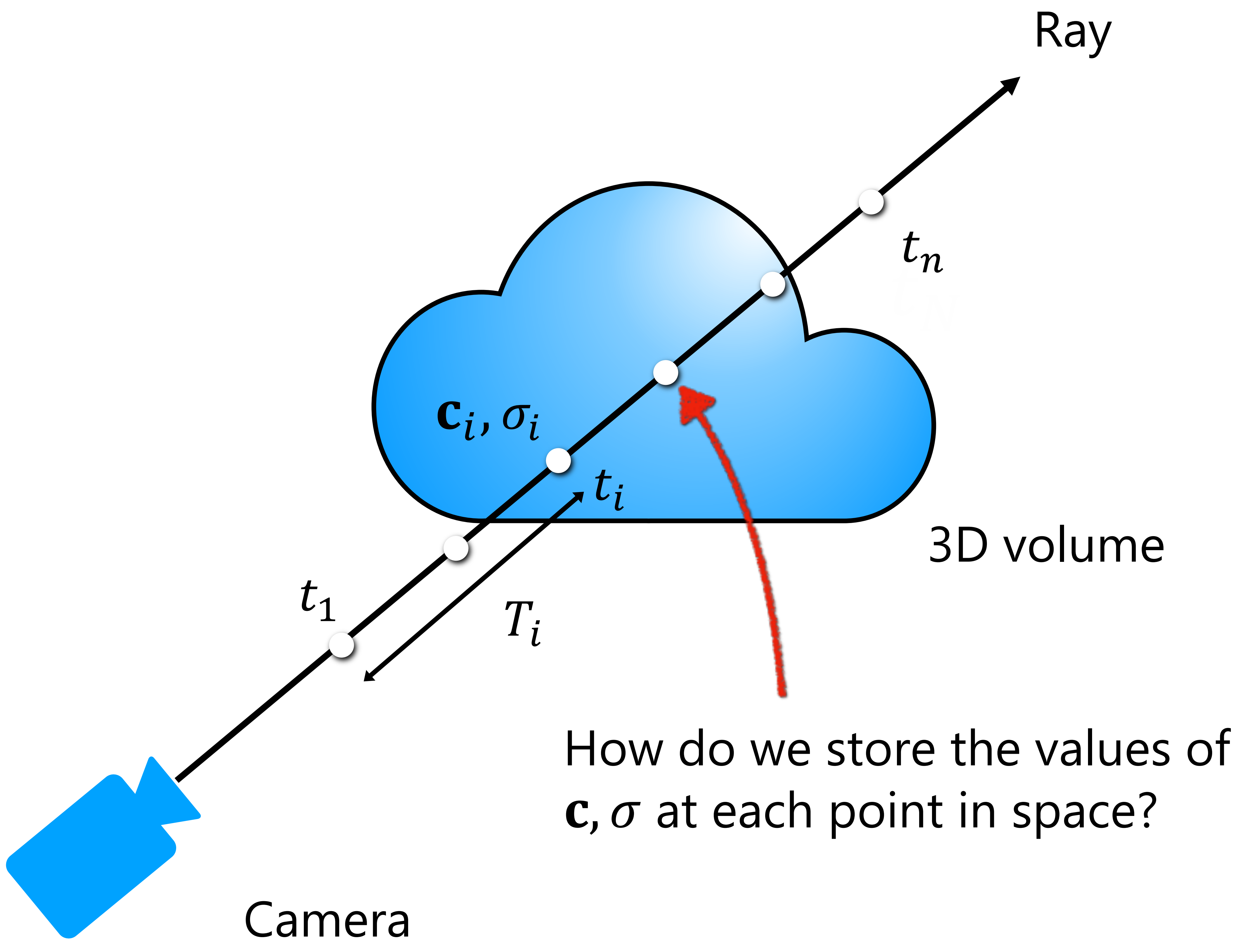
Rendering model for ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$:

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

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



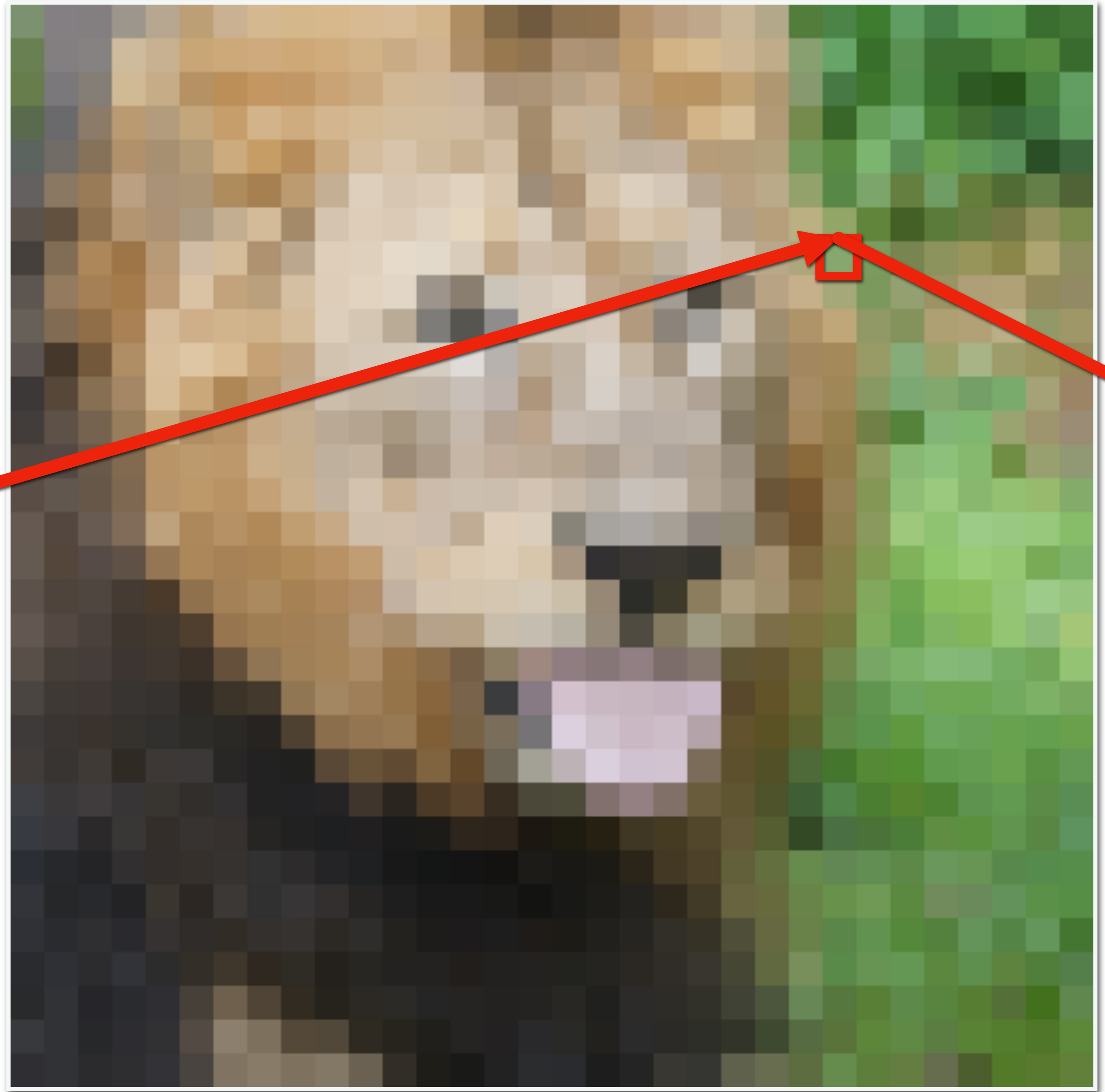
Adapted from material from Pratul Srinivas

NeRF Overview

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

Toy problem: storing 2D image data

(x, y)

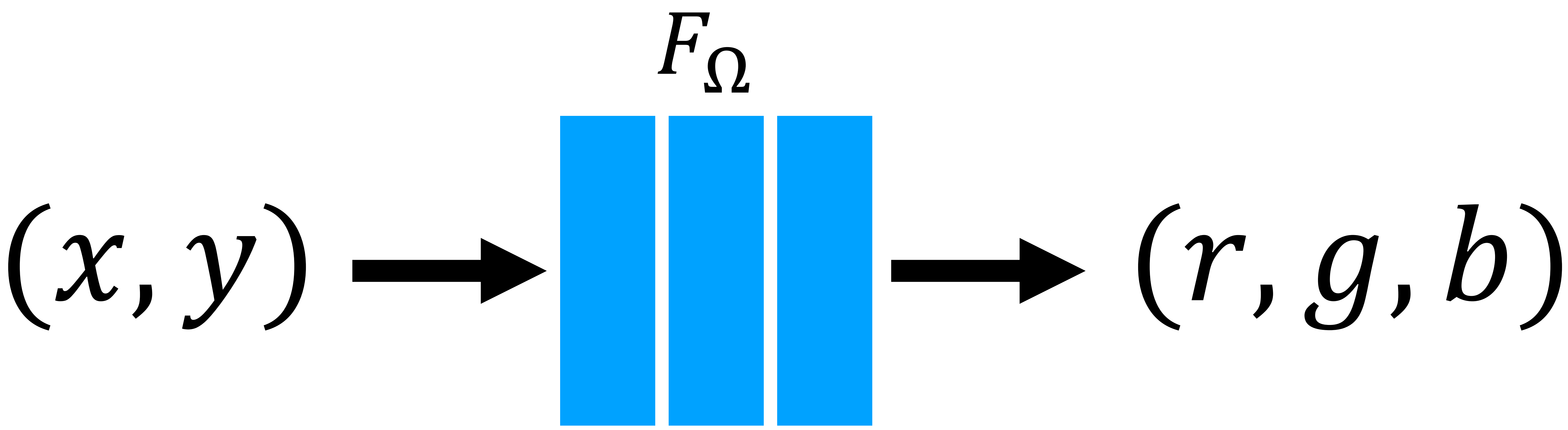


(r, g, b)

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

Adapted from material from Pratul Srinivas

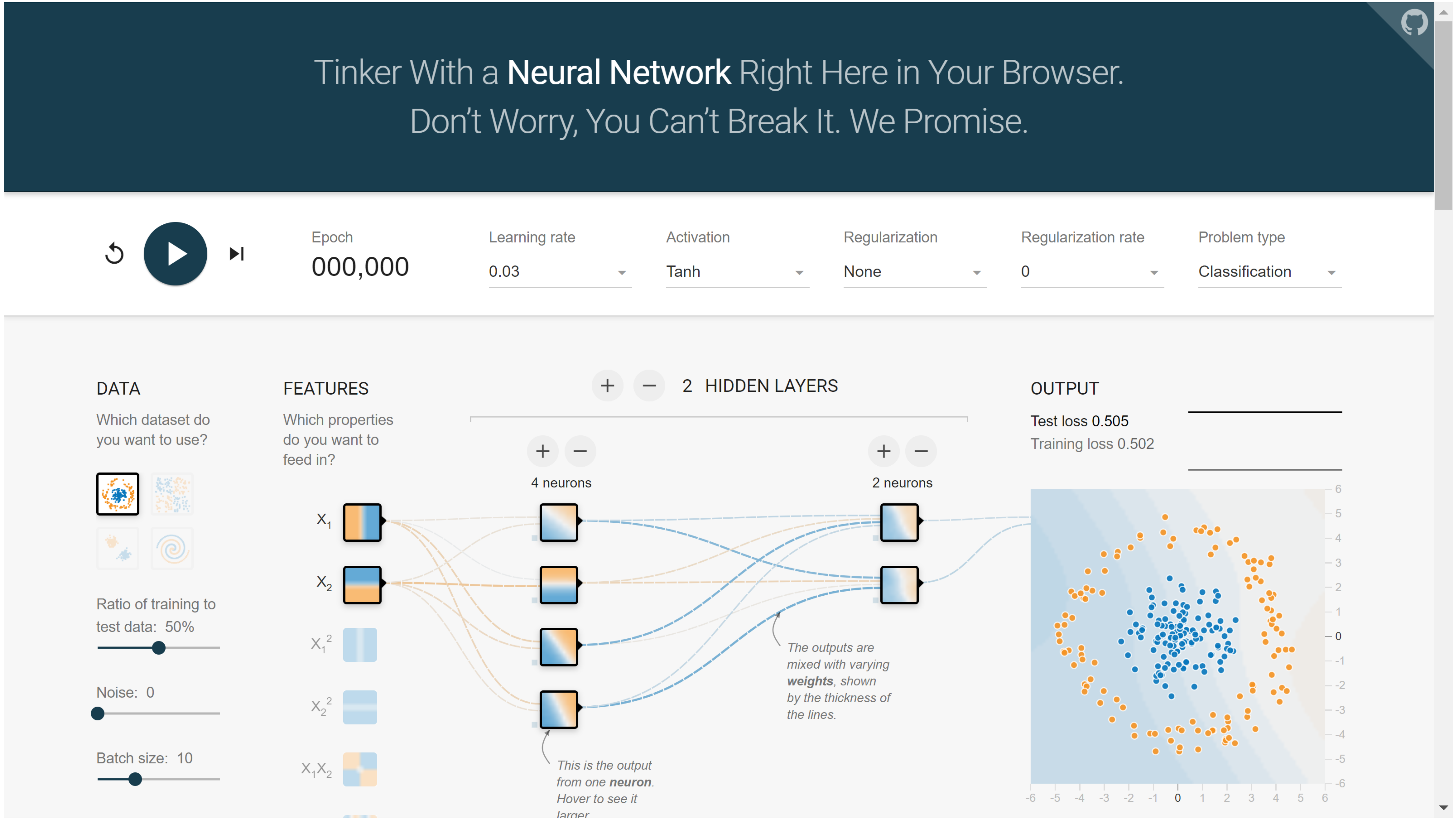
Toy problem: storing 2D image data



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

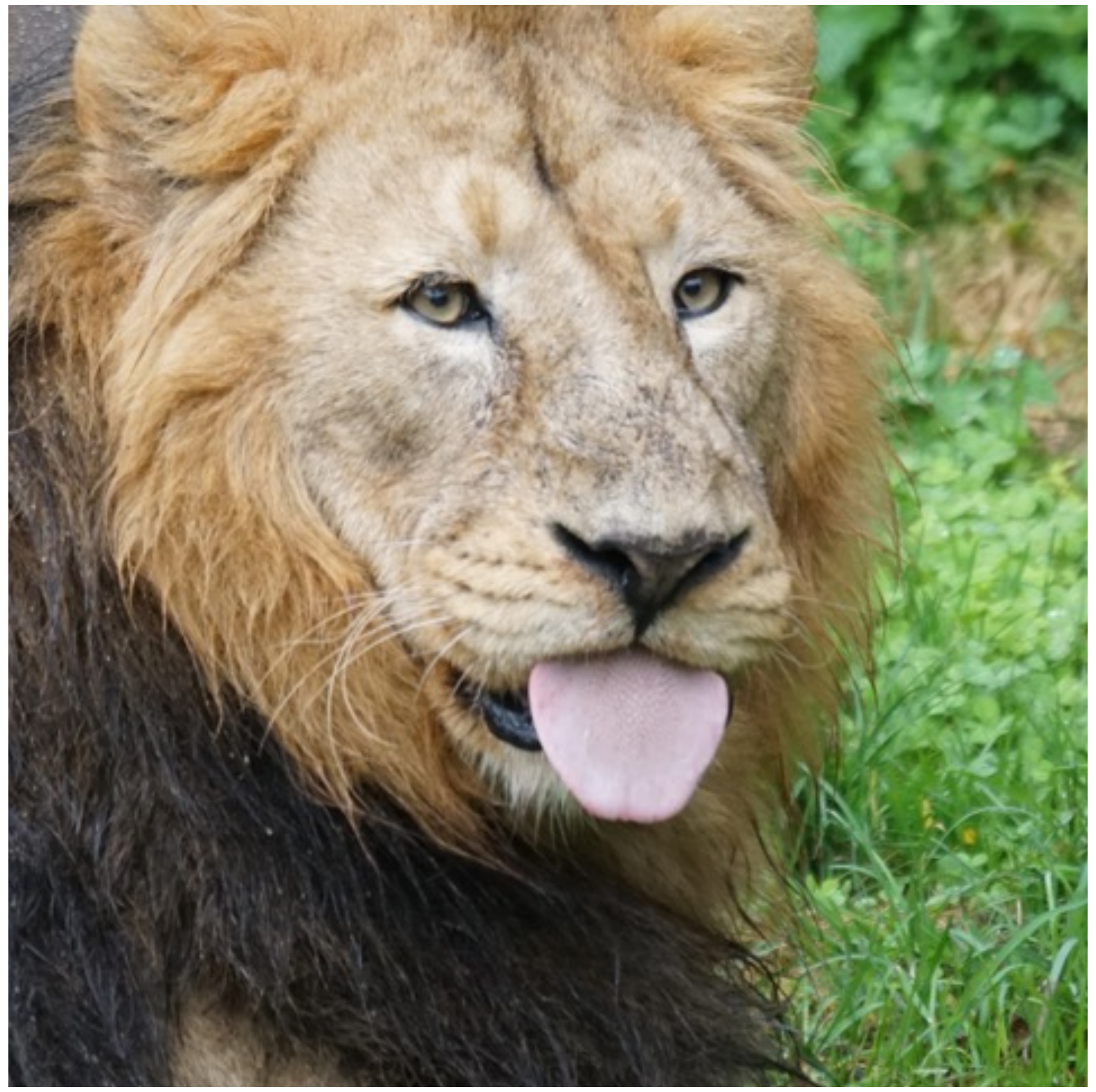
Adapted from material from Pratul Srinivas

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

Adapted from material from Pratul Srinivas

Problem:

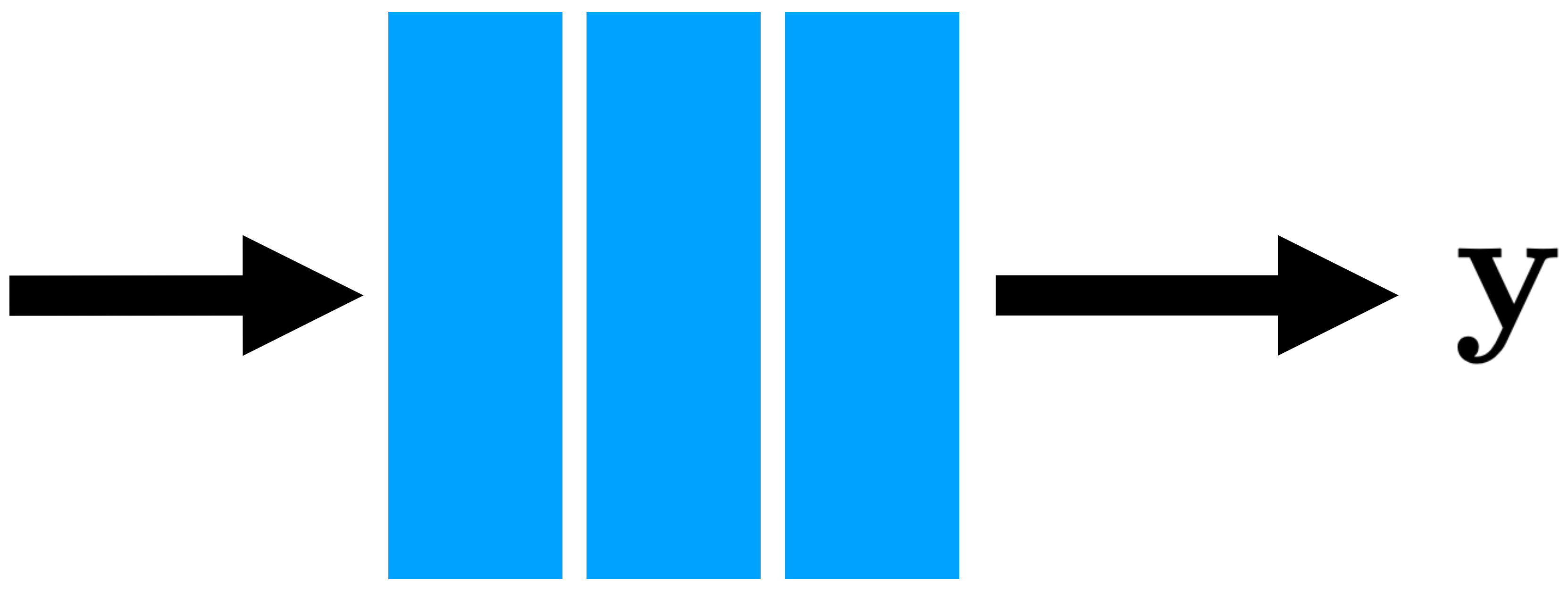
"Standard" coordinate-based MLPs cannot represent high frequency functions

Solution:

Pass input coordinates through a high frequency mapping first

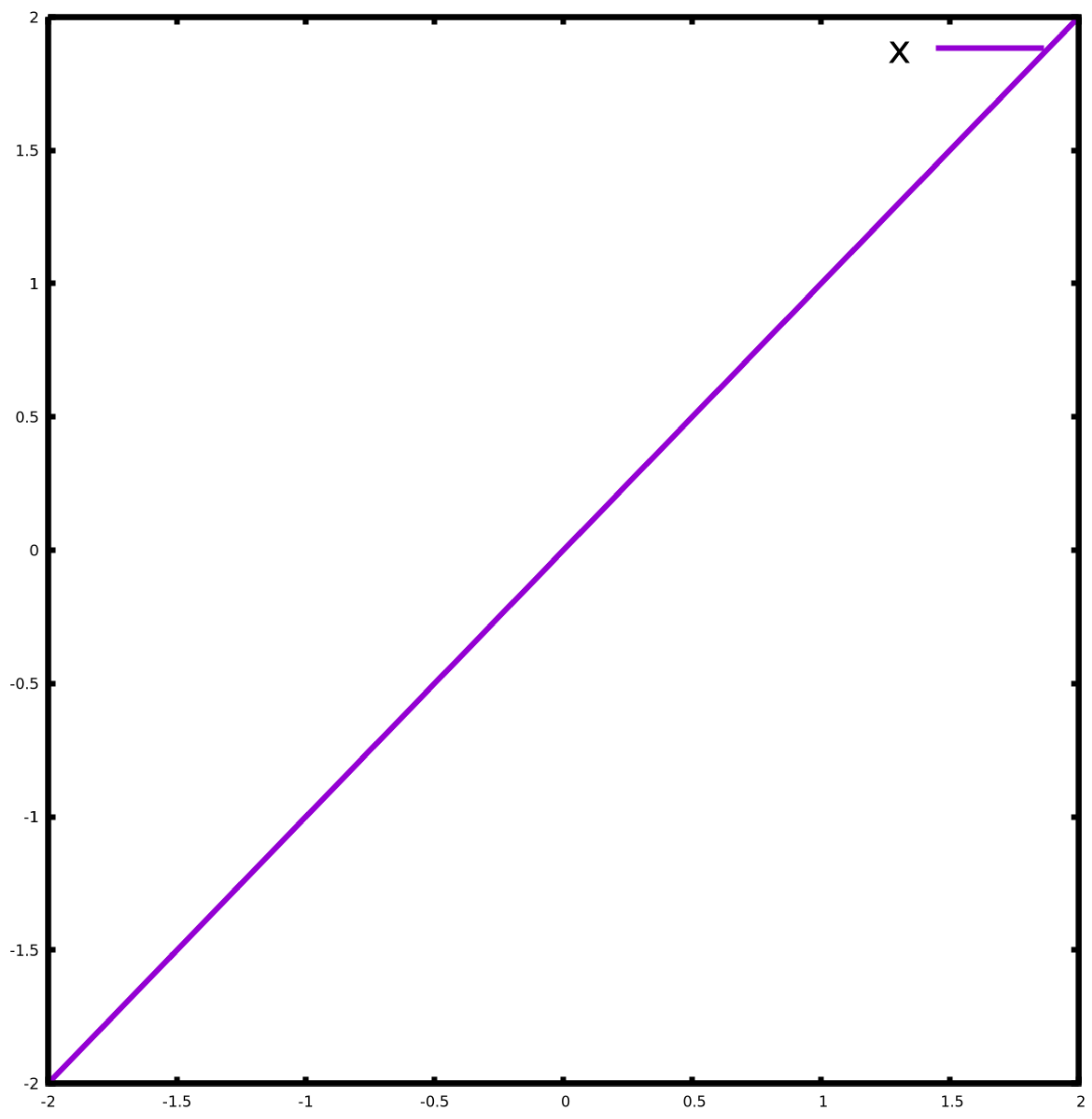
Example mapping: “positional encoding”

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

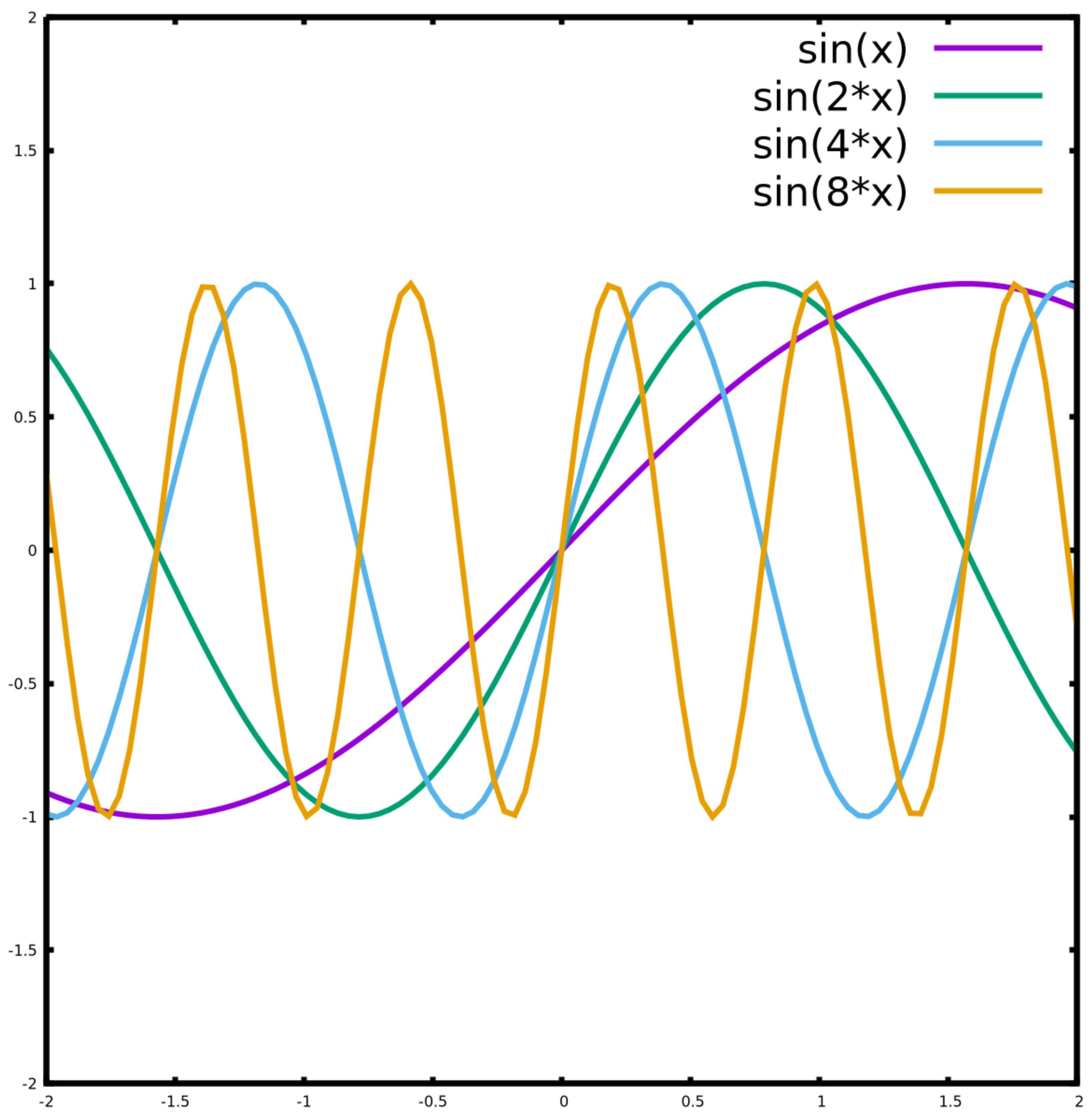


Adapted from material from Pratul Srinivas

Positional encoding



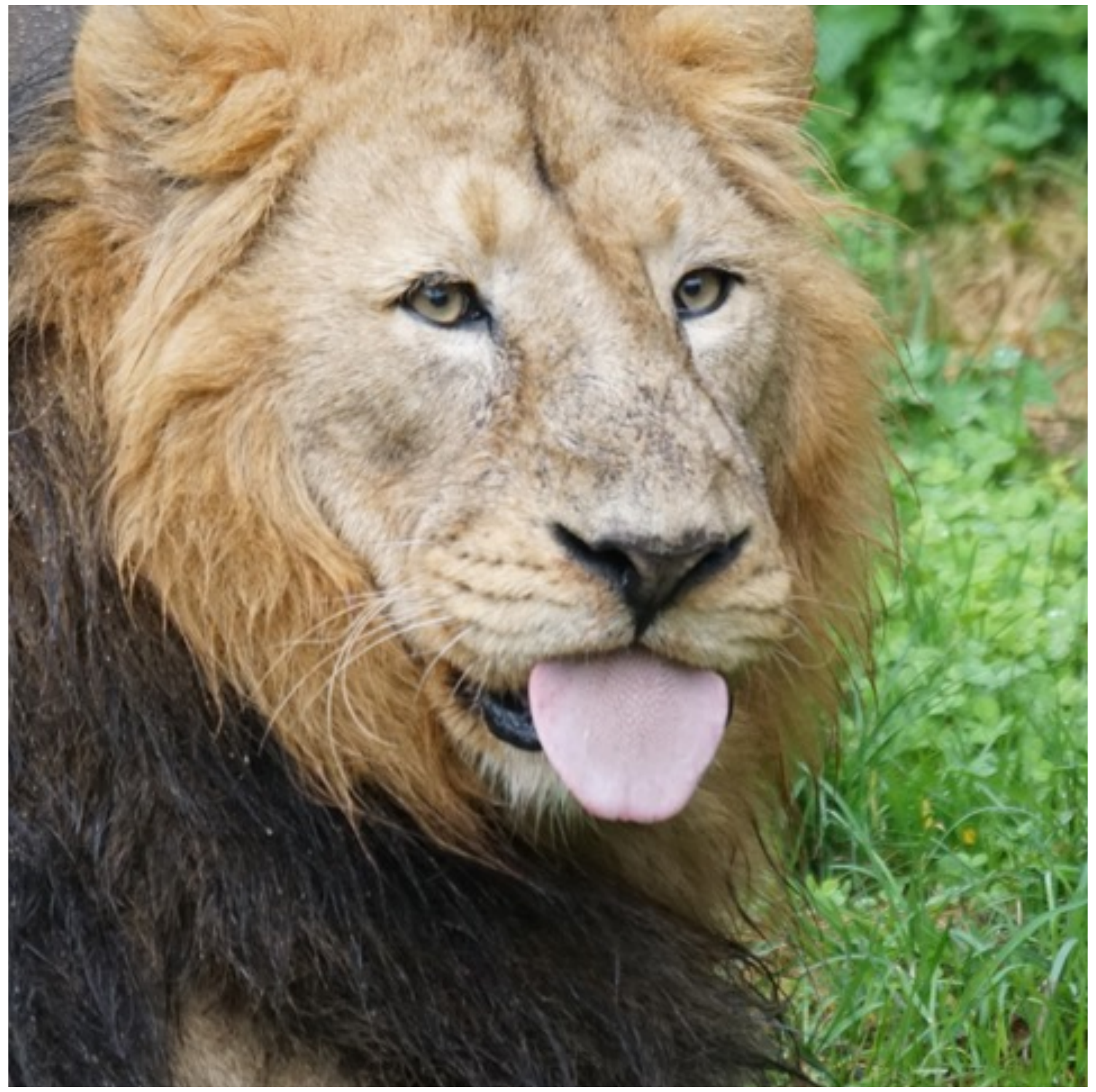
Raw encoding of a number ***x***



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

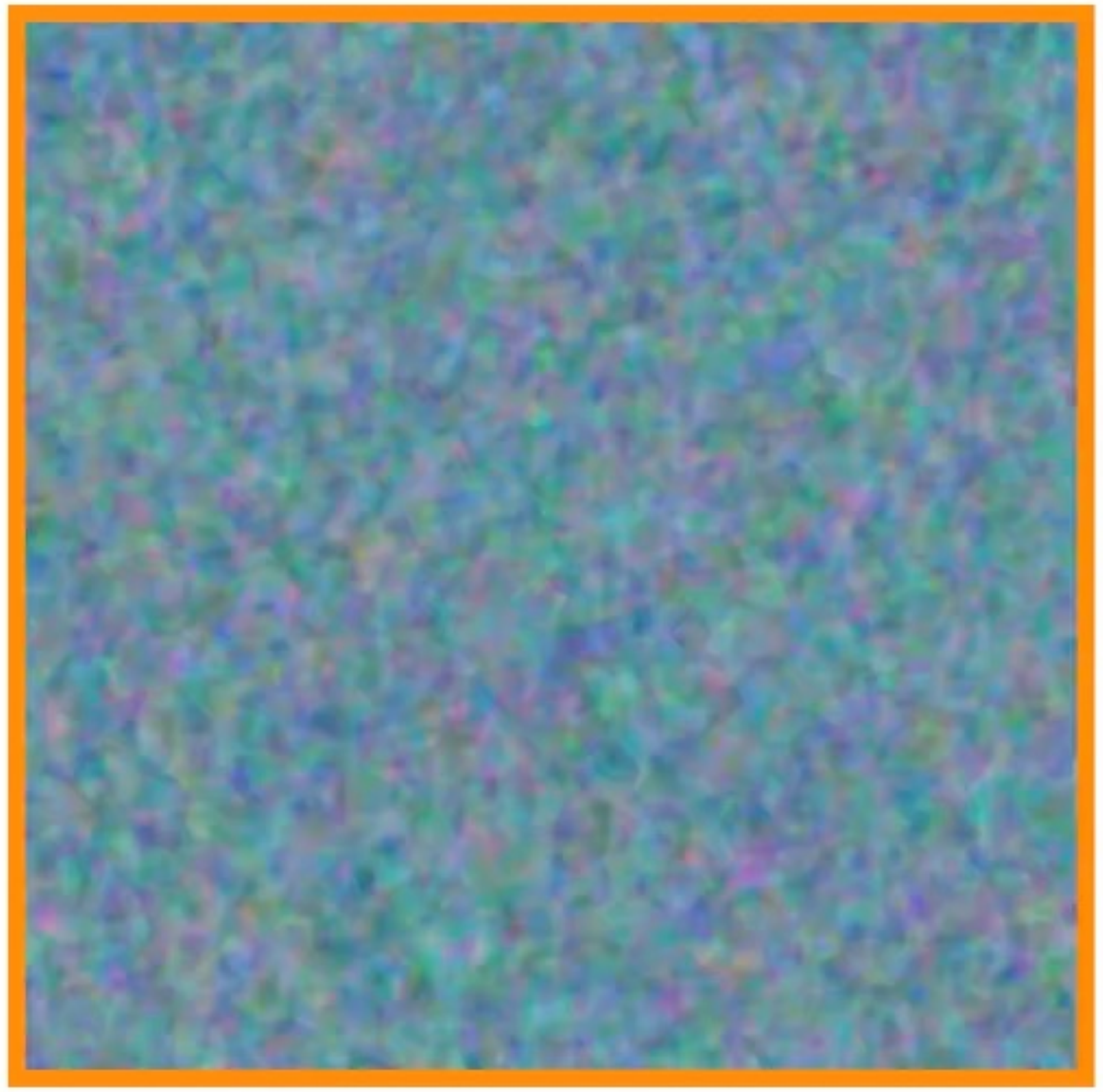
Adapted from material from Pratul Srinivas

Problem solved!



Ground truth image

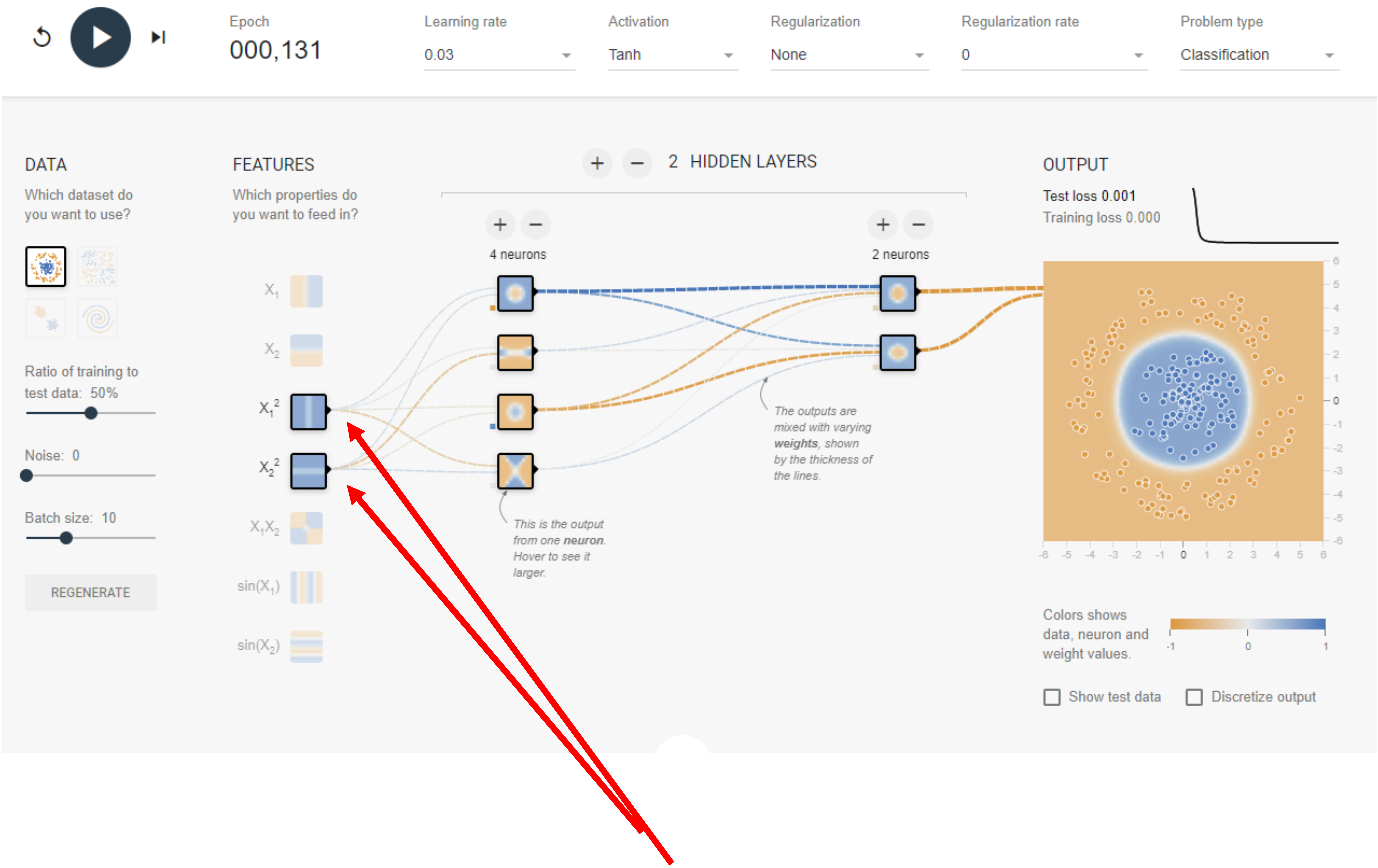
Neural network output without high frequency mapping



Neural network output with high frequency mapping

Adapted from material from Pratul Srinivas

Sometimes a better input encoding is all you need



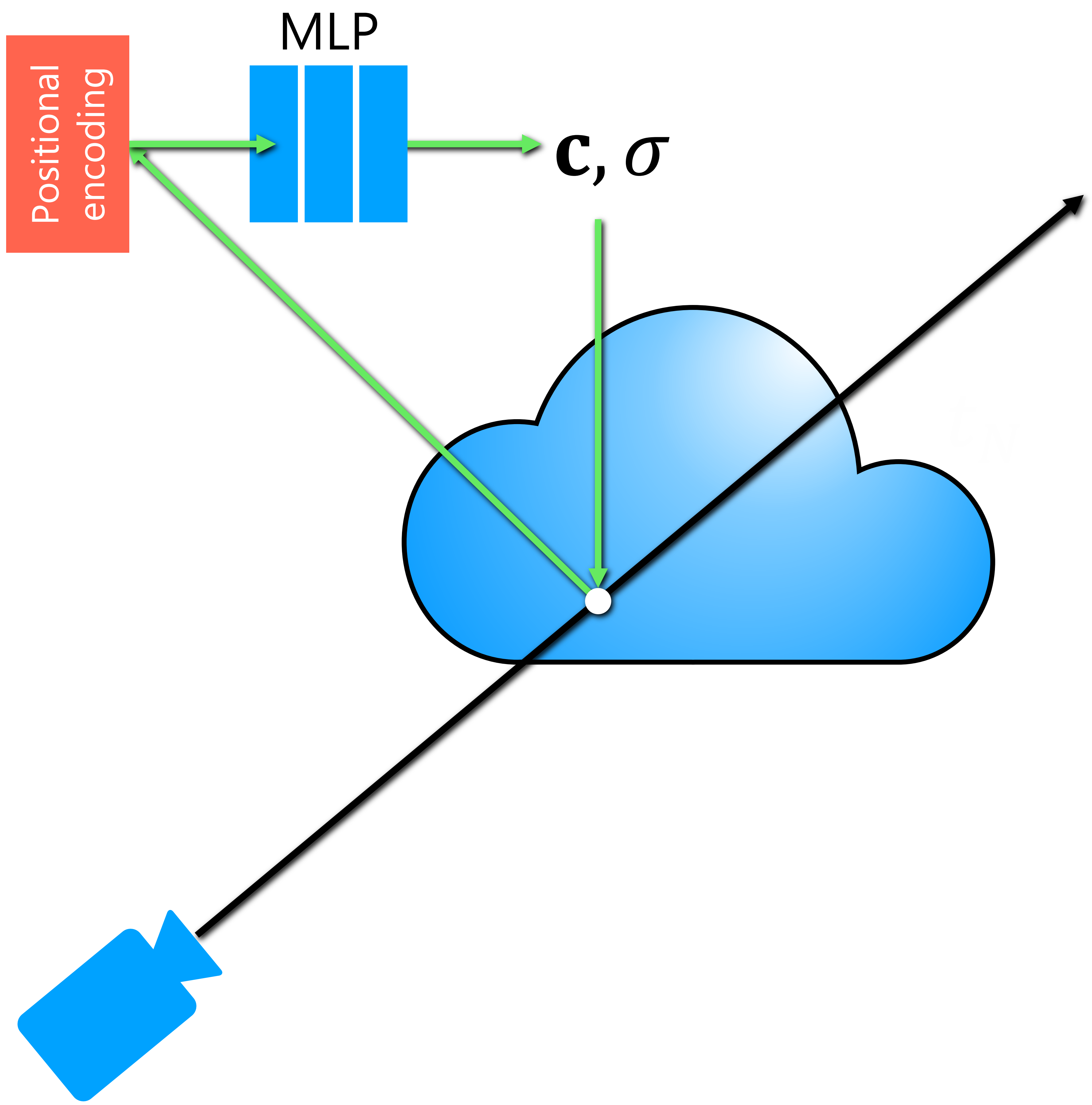
Recall “squared” encoding in TensorFlow Playground

NeRF Overview

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

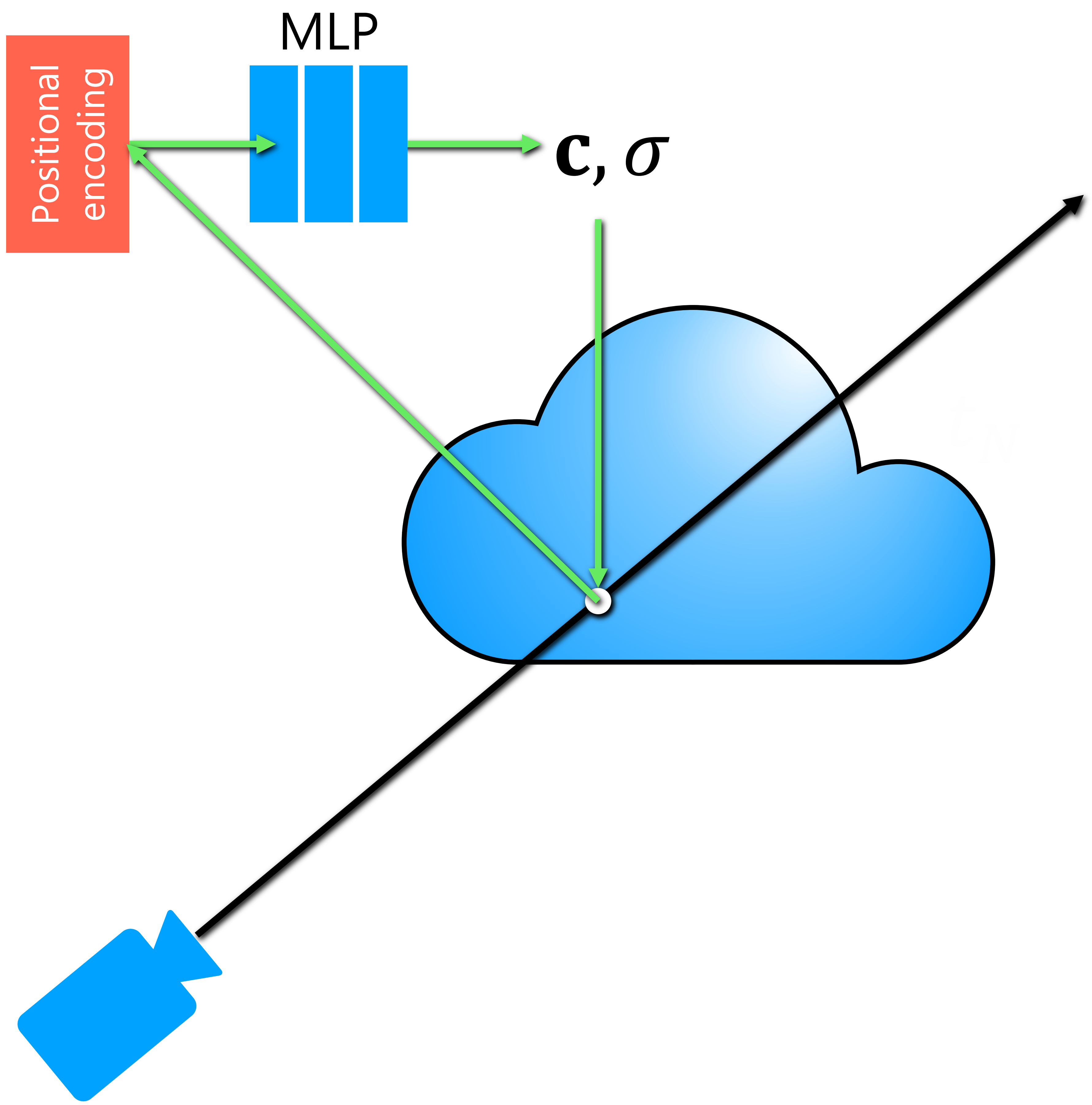
NeRF = volume rendering + coordinate-based network

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



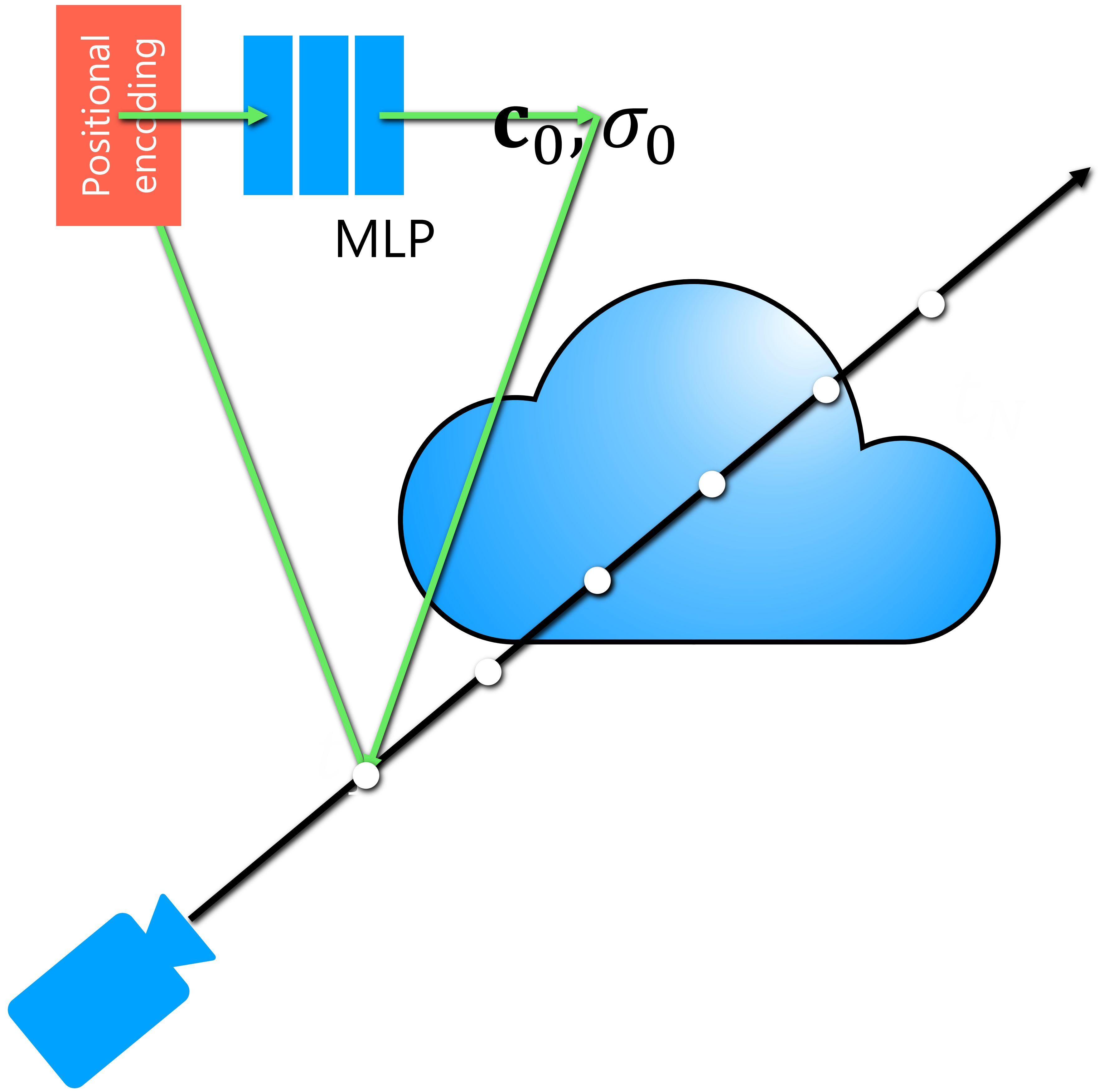
Adapted from material from Pratul Srinivas

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



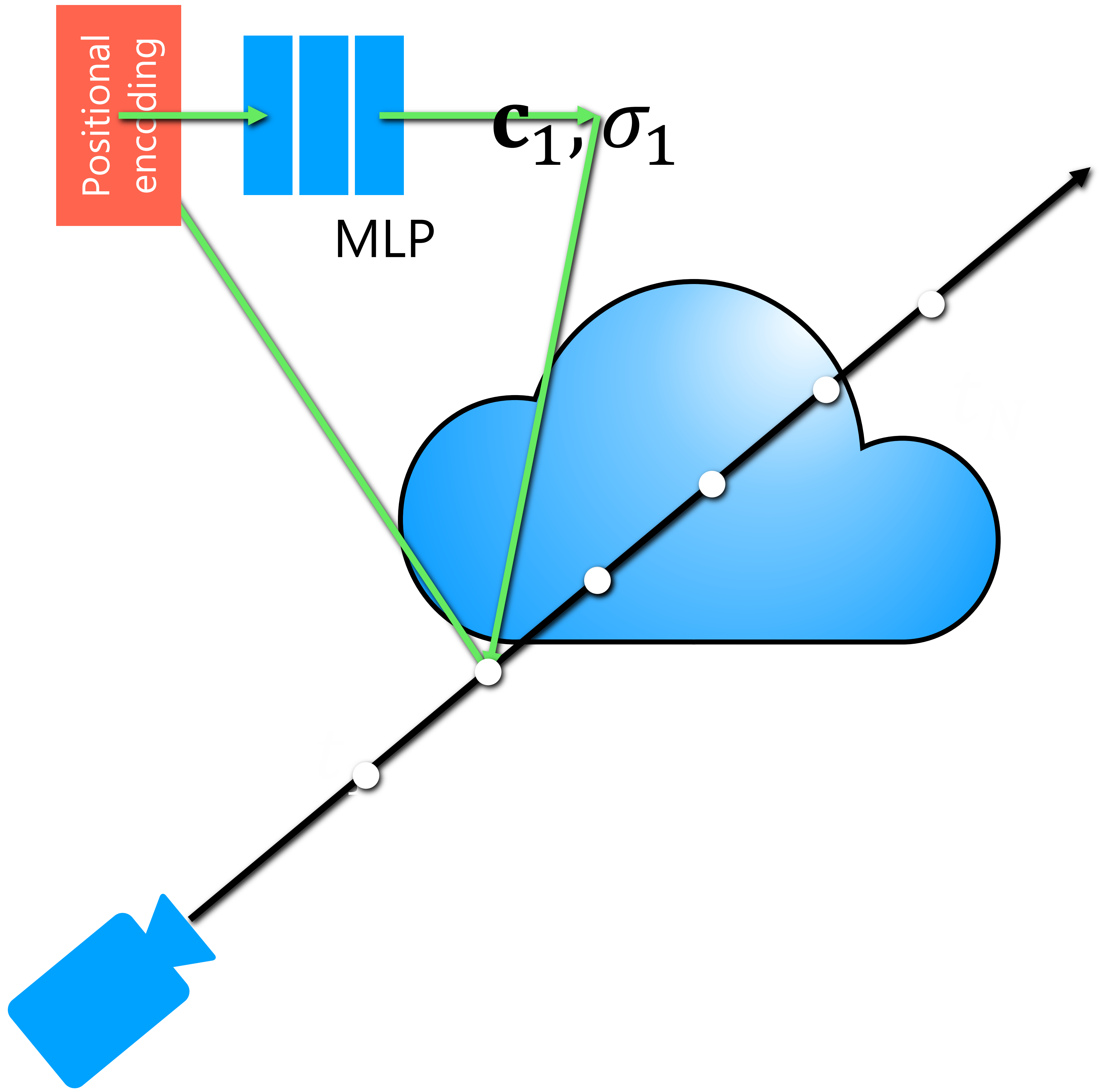
Adapted from material from Pratul Srinivas

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



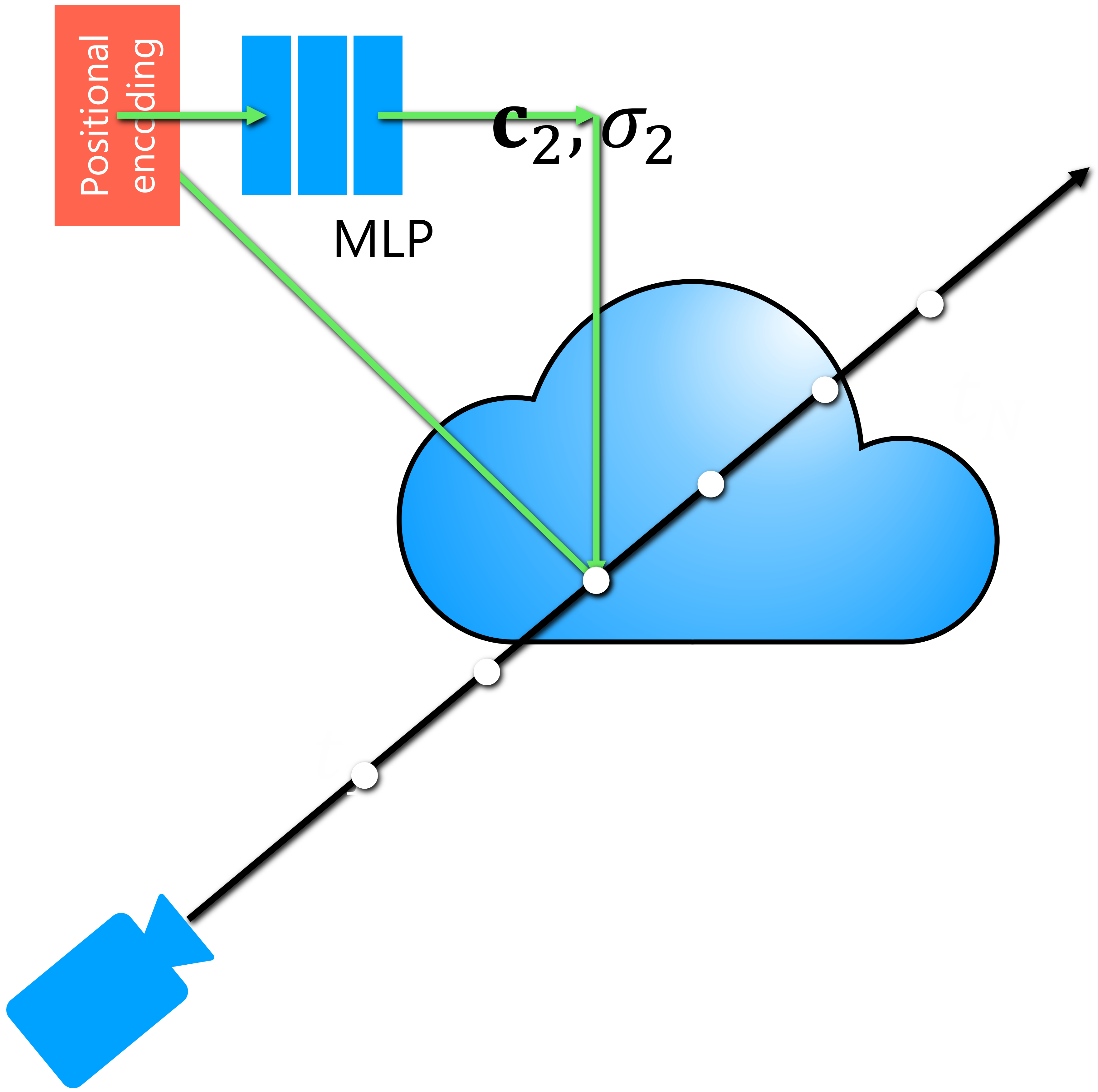
Adapted from material from Pratul Srinivas

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



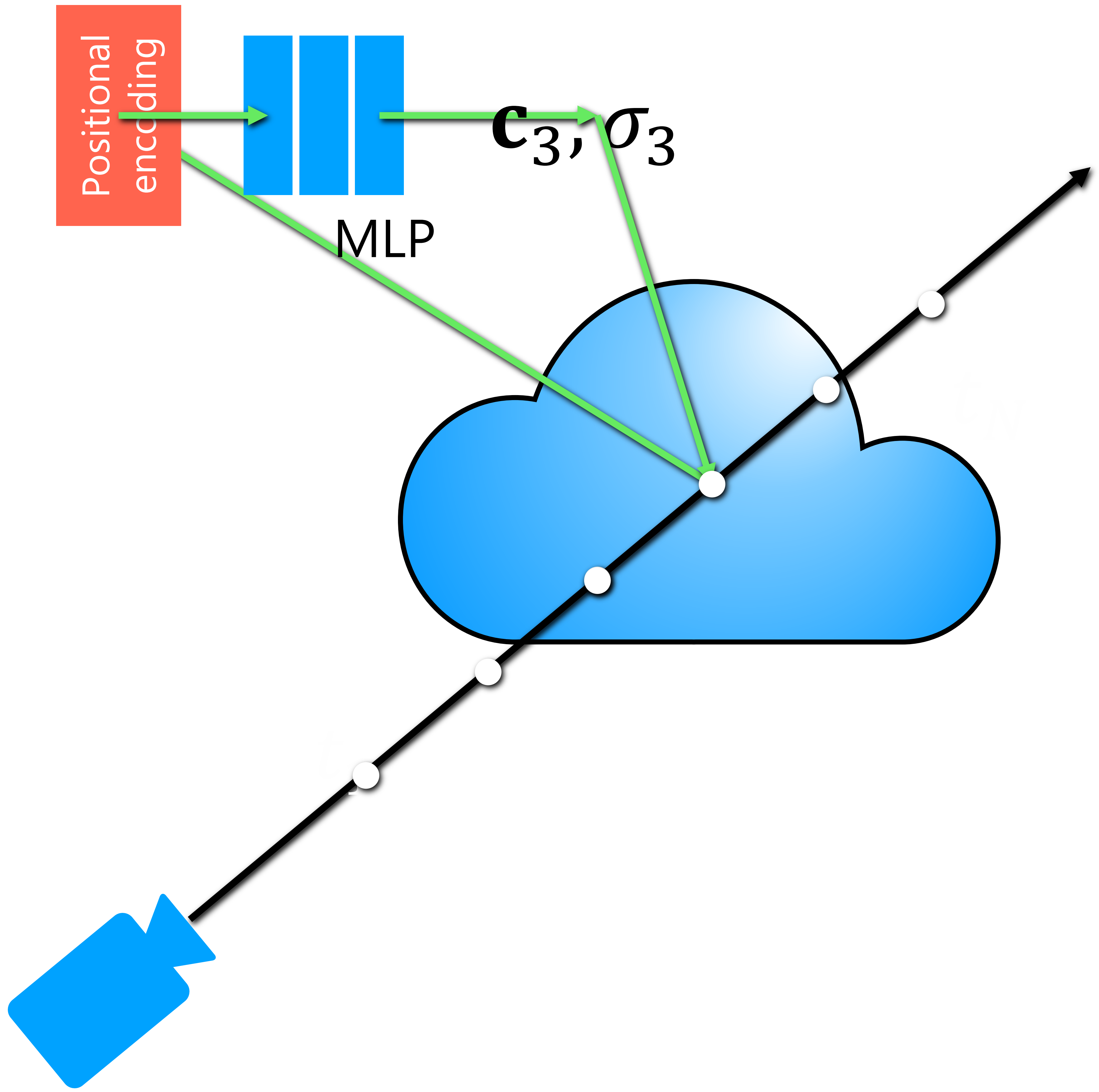
Adapted from material from Pratul Srinivas

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



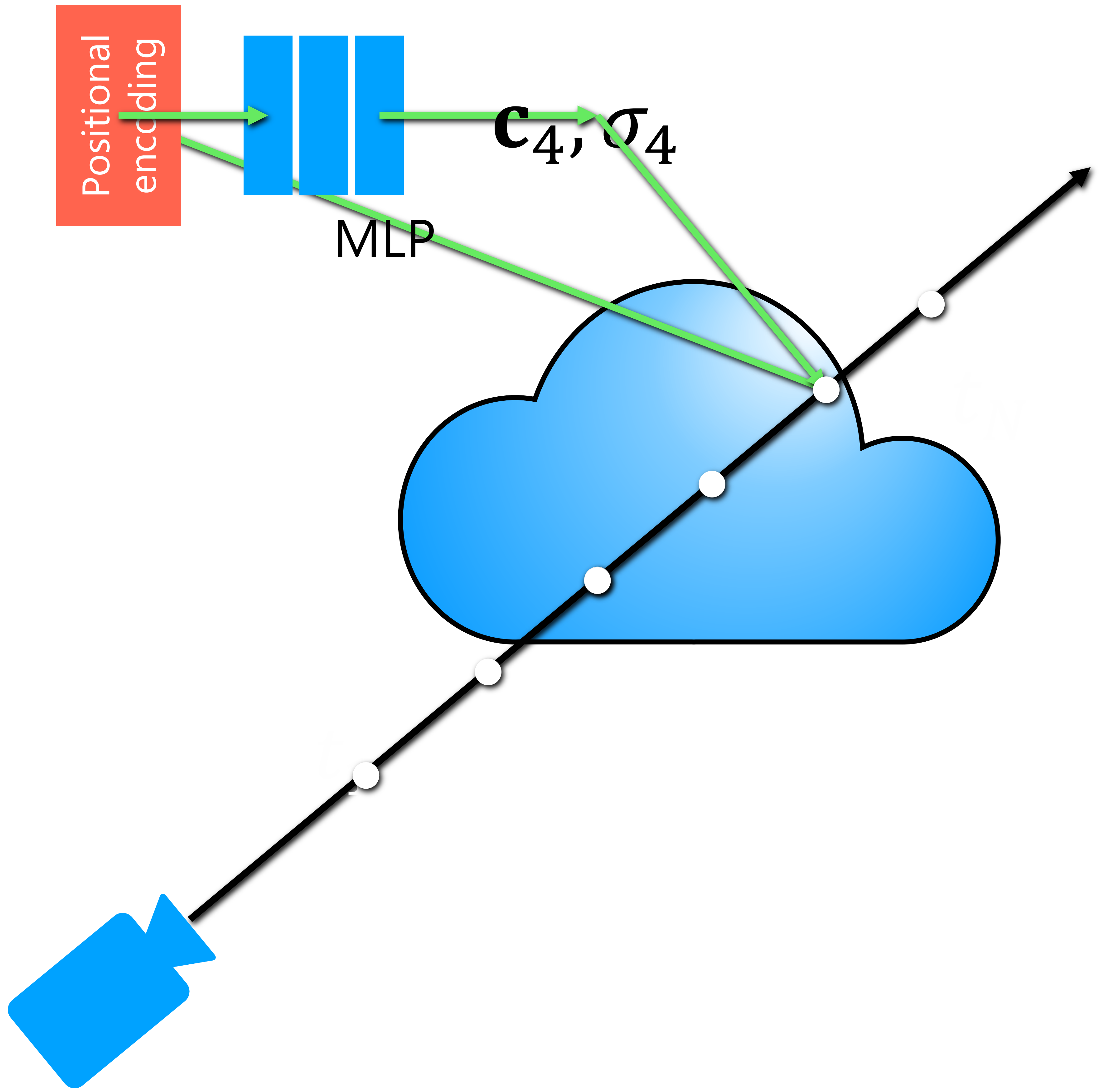
Adapted from material from Pratul Srinivas

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



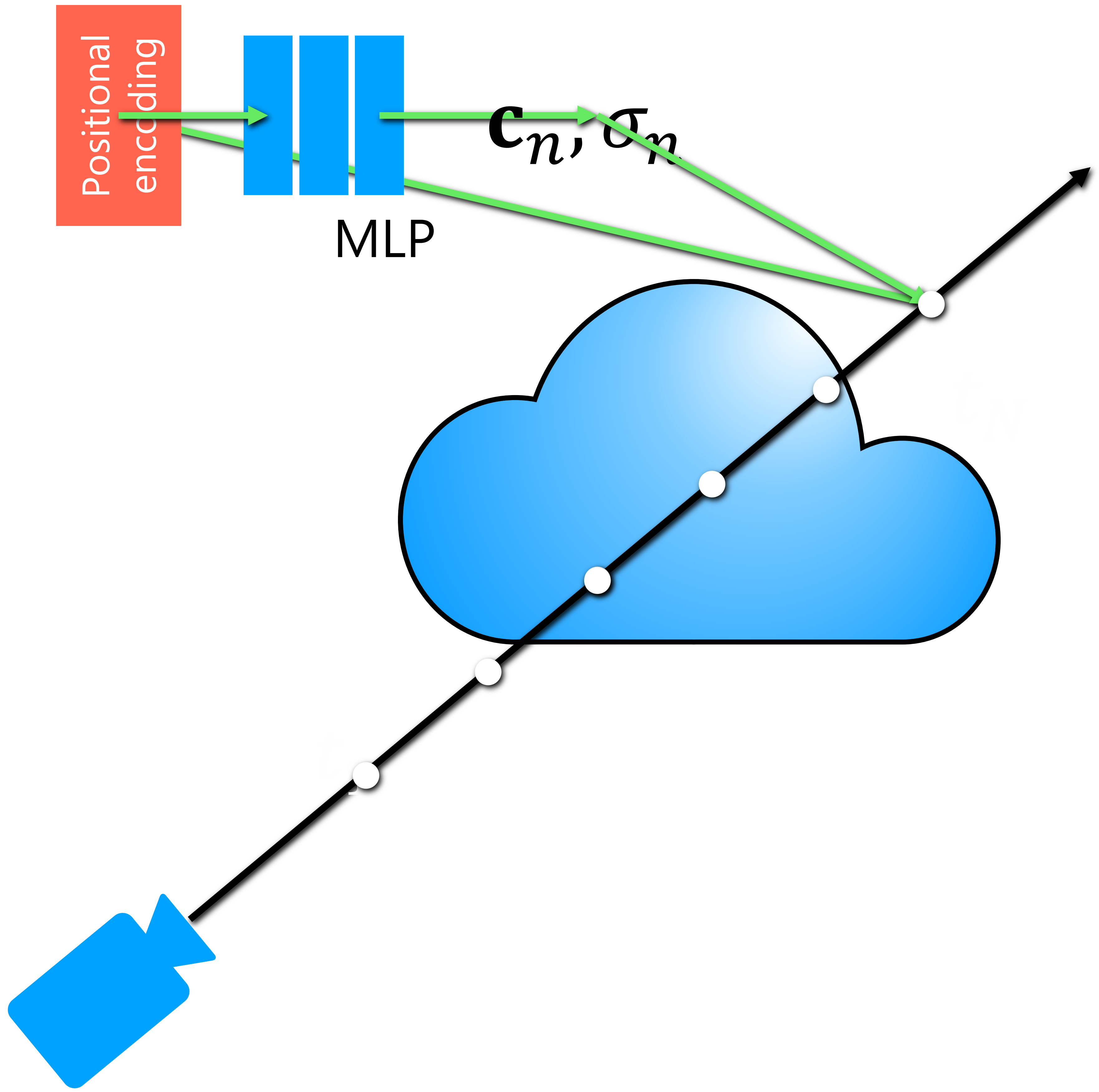
Adapted from material from Pratul Srinivas

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



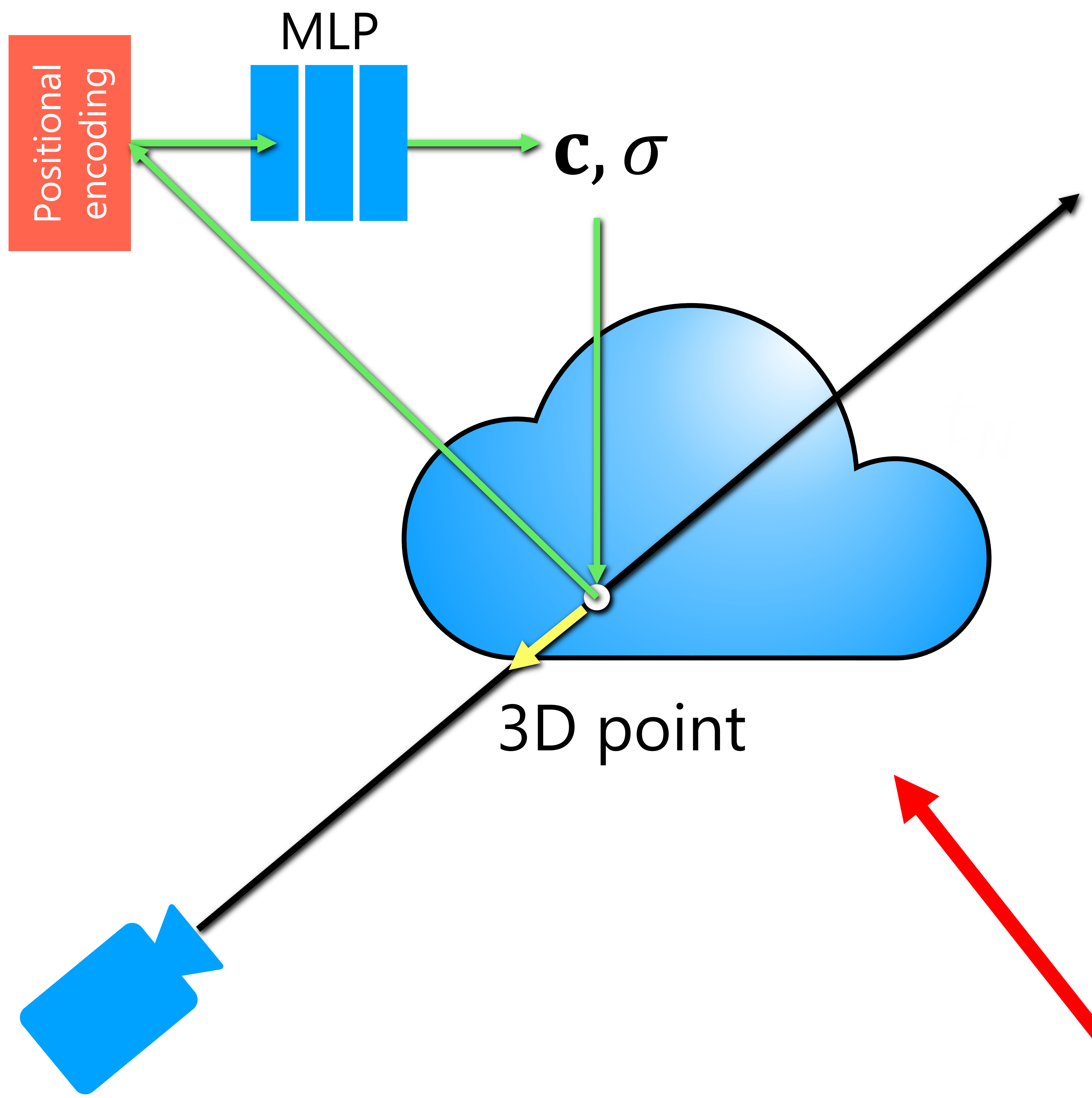
Adapted from material from Pratul Srinivas

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



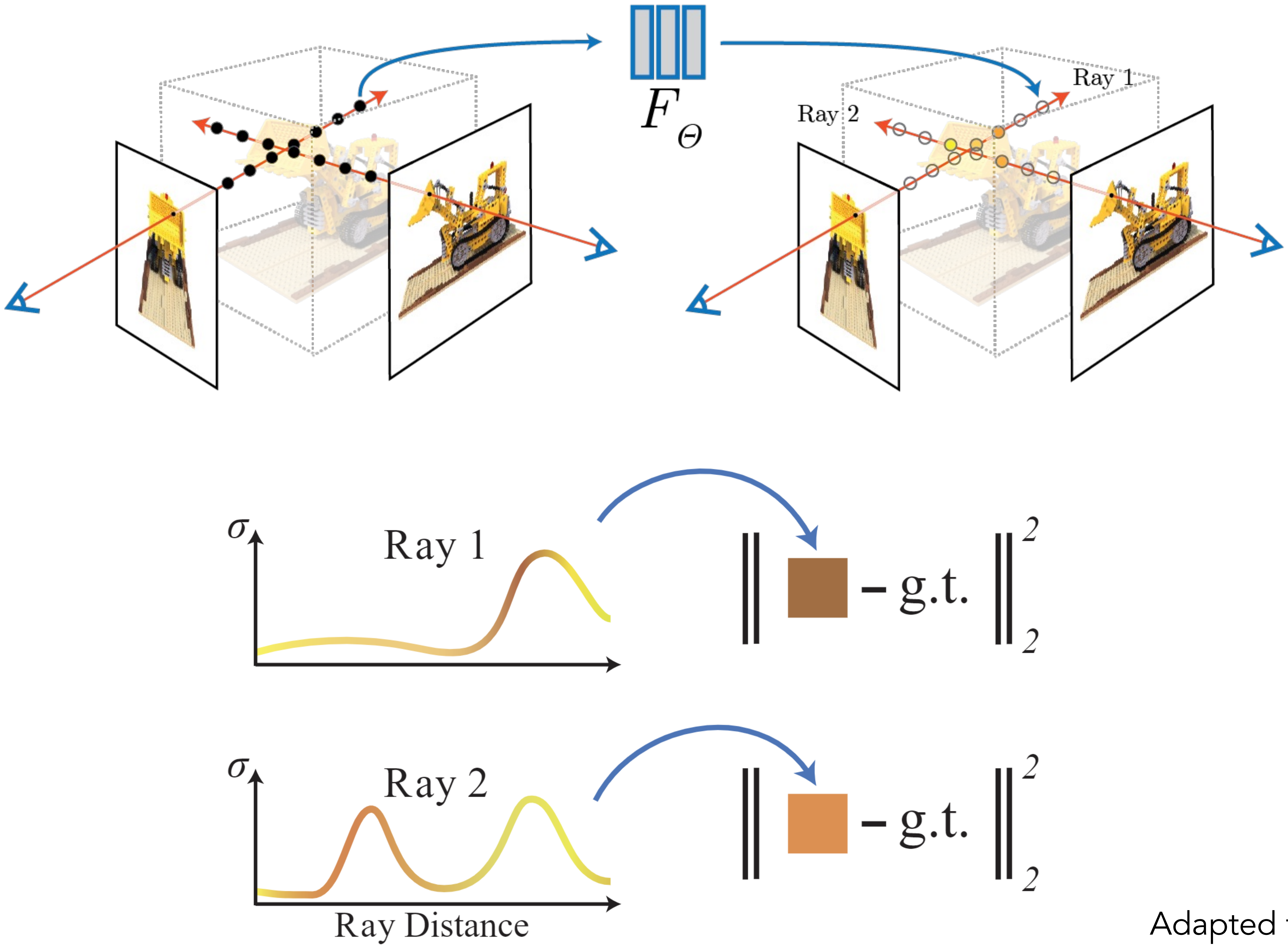
Adapted from material from Pratul Srinivas

Extension: view-dependent field



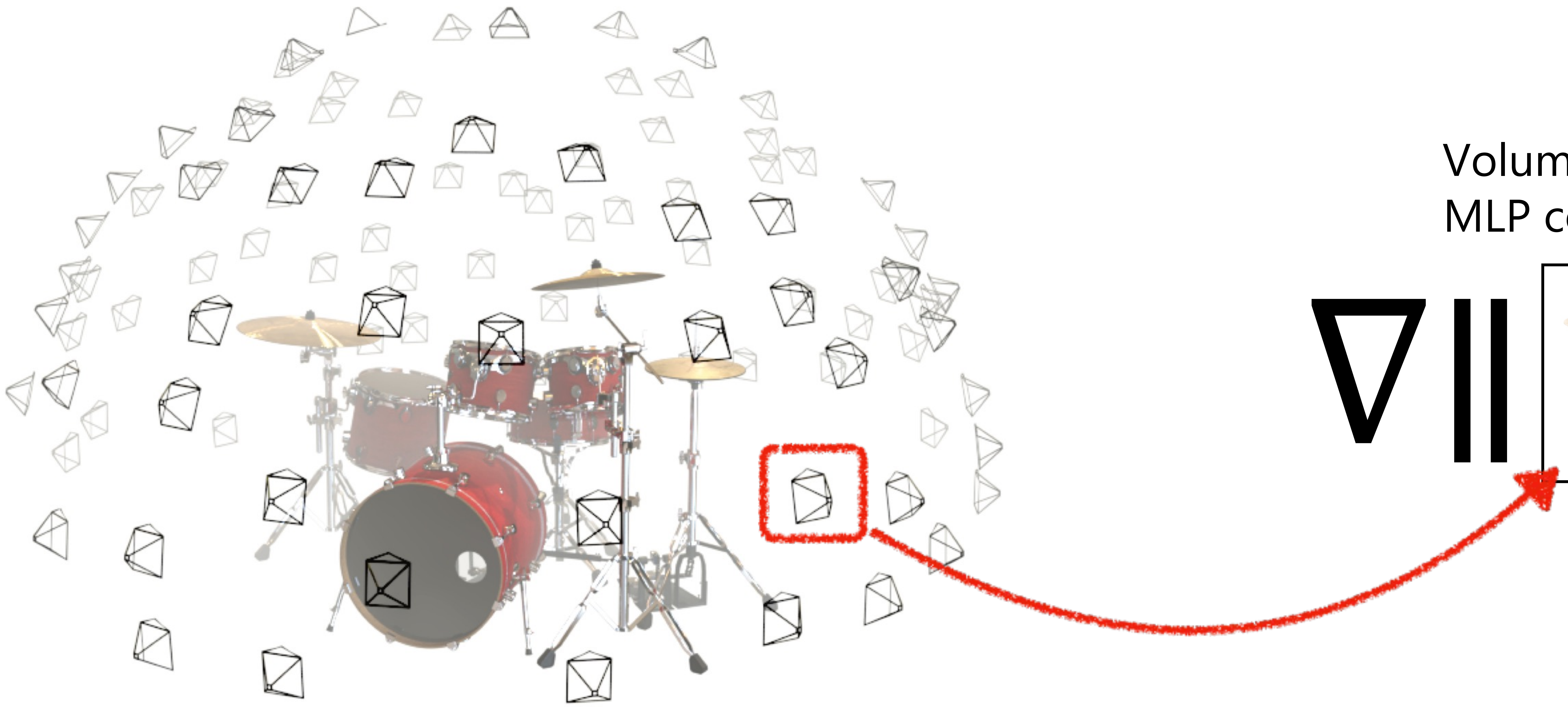
Include the ray direction in the input to the MLP → allows for capturing and rendering view-dependent effects (e.g., shiny surfaces)

Putting it all together



Adapted from material from Pratul Srinivas

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



Volume rendering of MLP colors/densities

Ground truth image

$\nabla \|$  $-$  $\|^2$

Adapted from material from Pratul Srinivas

Results

NeRF encodes convincing view-dependent effects using directional dependence

Adapted from material from Pratul Srinivas

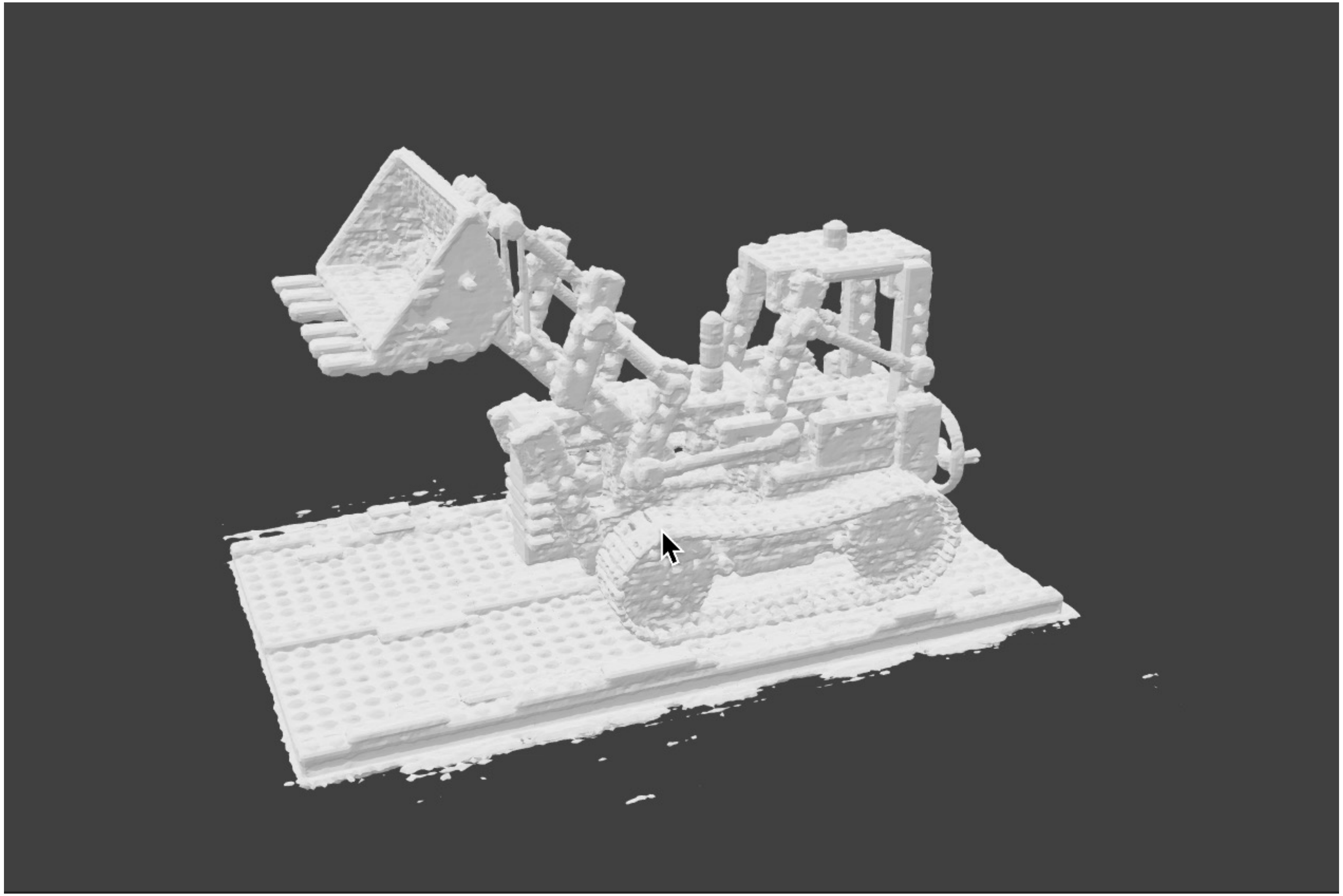
NeRF encodes convincing view-dependent effects using directional dependence

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

Input images of
an object

Reconstructed shape,
albedo, and materials

Reconstructed models inserted into scene with new lighting

Zhang, Luan, Li, Snavely. CVPR

Questions?