Folien zur Vorlesung am 03.06.2025

3D Computer Vision

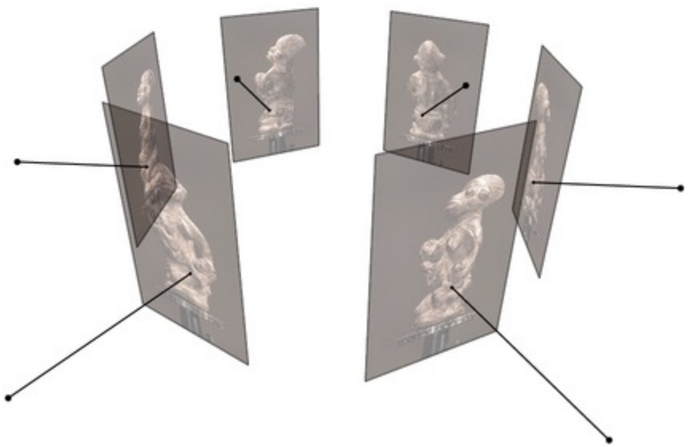
NERF - NEURAL RADIANCE FIELDS

Recommended Reading

- Slides from Noah Snavely (Cornell), which states that many slides are based on material from Pratul Srinivasan (Google)
- Original NeRF project and paper (Two Minutes Papers).

Last time: Multi View Stereo (MVS)

- Compute depth map per image
- Fuse the depth maps into a 3D model



Figures by Carlos Hernandez

Computer vision as inverse rendering

3D
Scene
Representation

Rendering

Computer vision as inverse rendering

Inverse Rendering

3D
Scene
Representation

Rendering

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

Inverse Rendering

Neural Radiance Field

Rendering

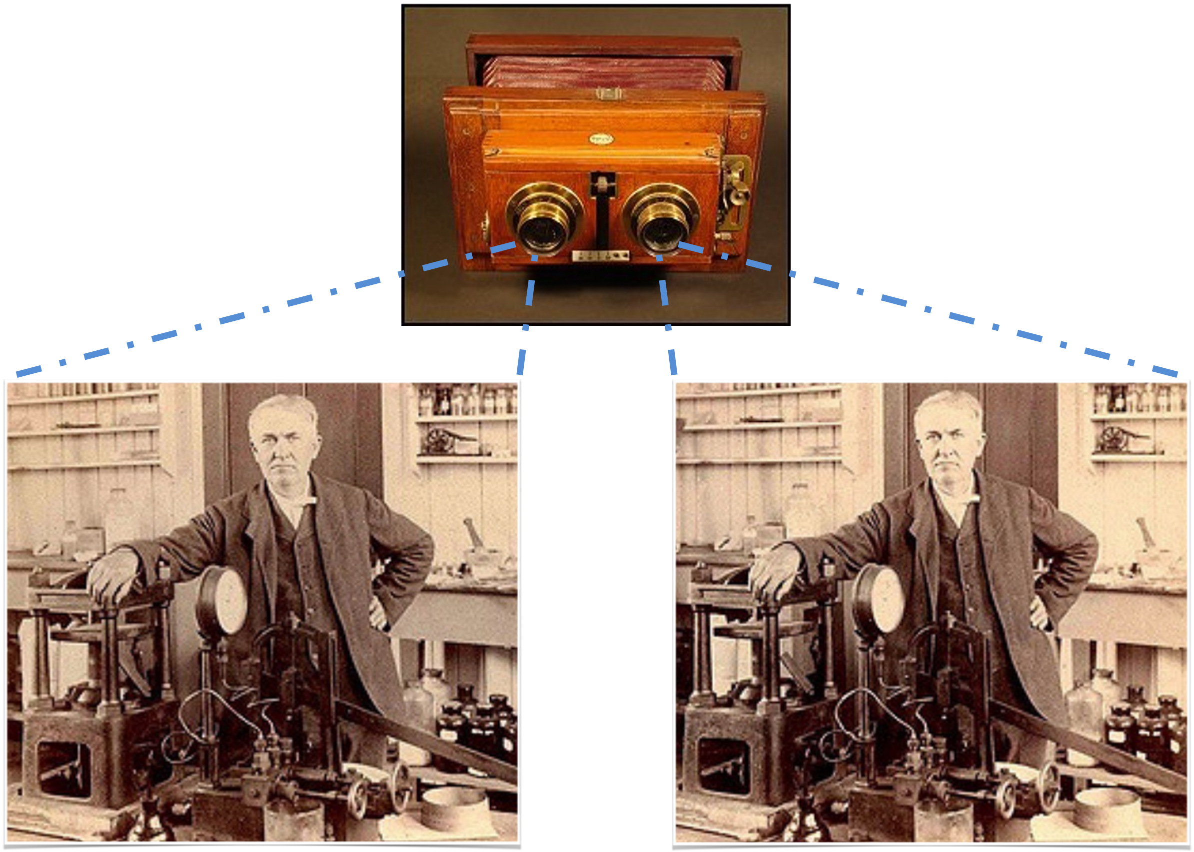
Deep learning for 3D reconstruction

- 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

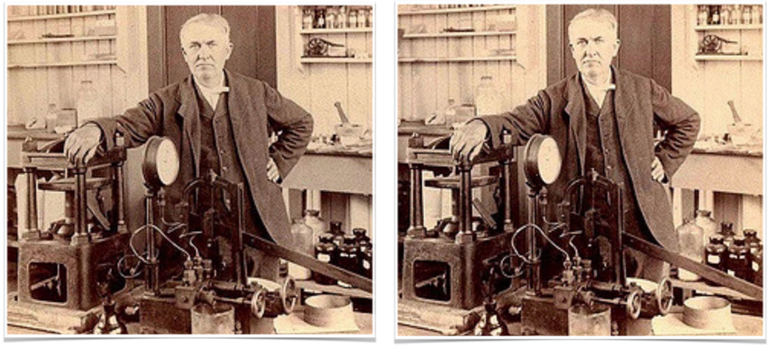
NeRF and related methods – Key ideas

- 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

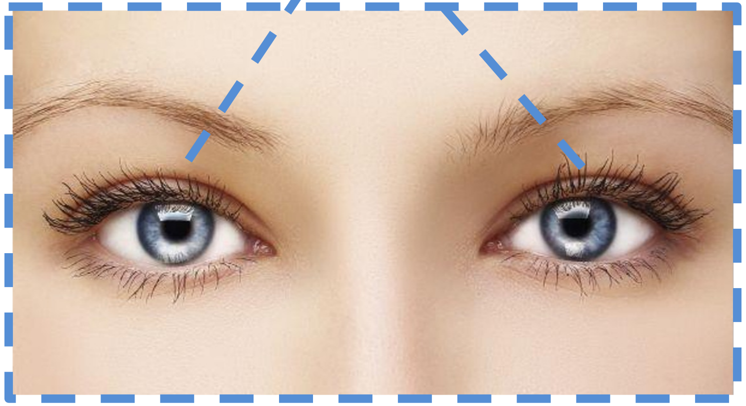
Stereo Photography

Queen Victoria at World Fair, 1851

Stereo Photography

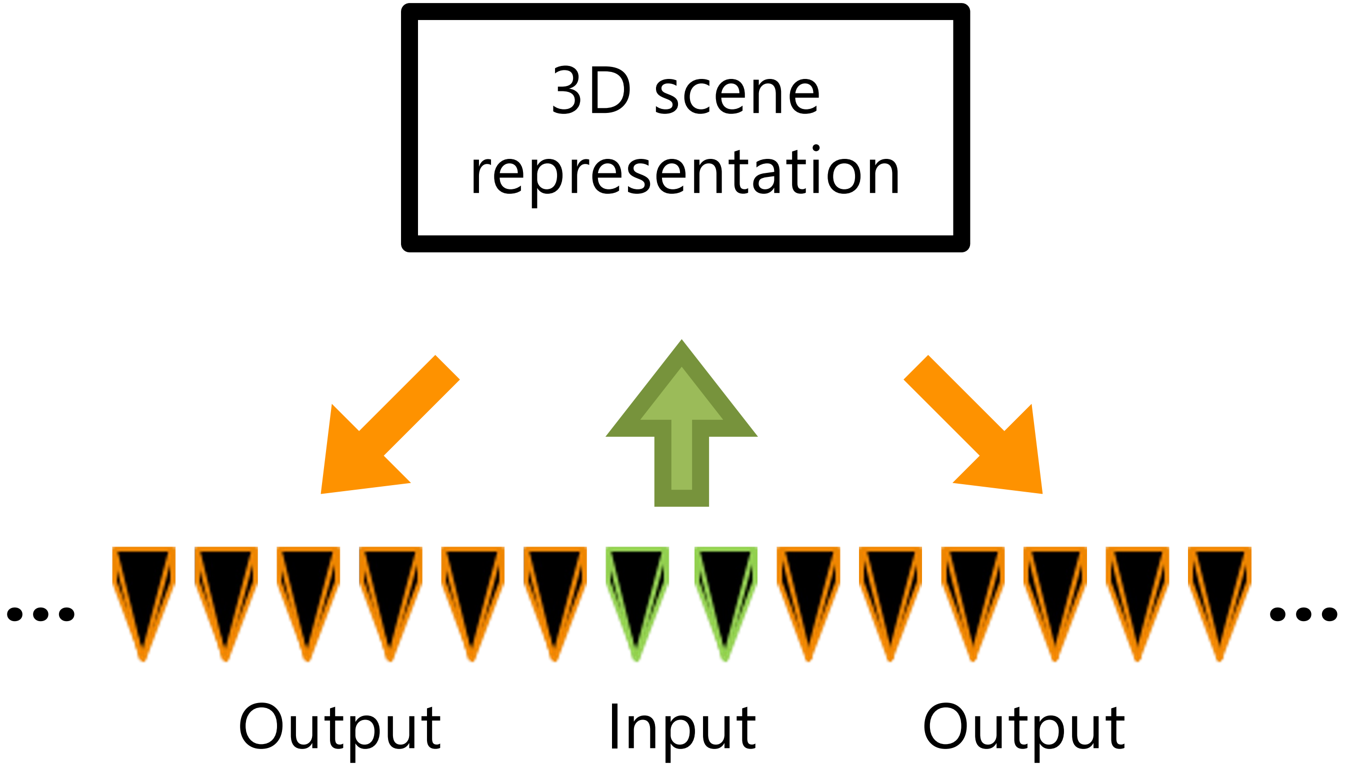
Issue: Narrow Baseline

~6.5 cm



~1.5 cm

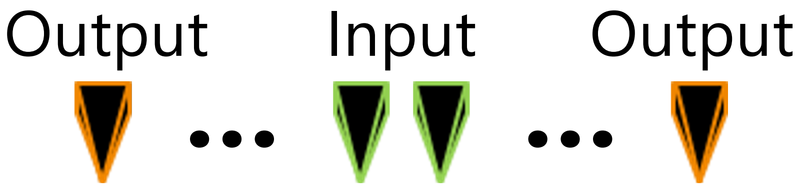
Problem Statement



Challenges

Extrapolation

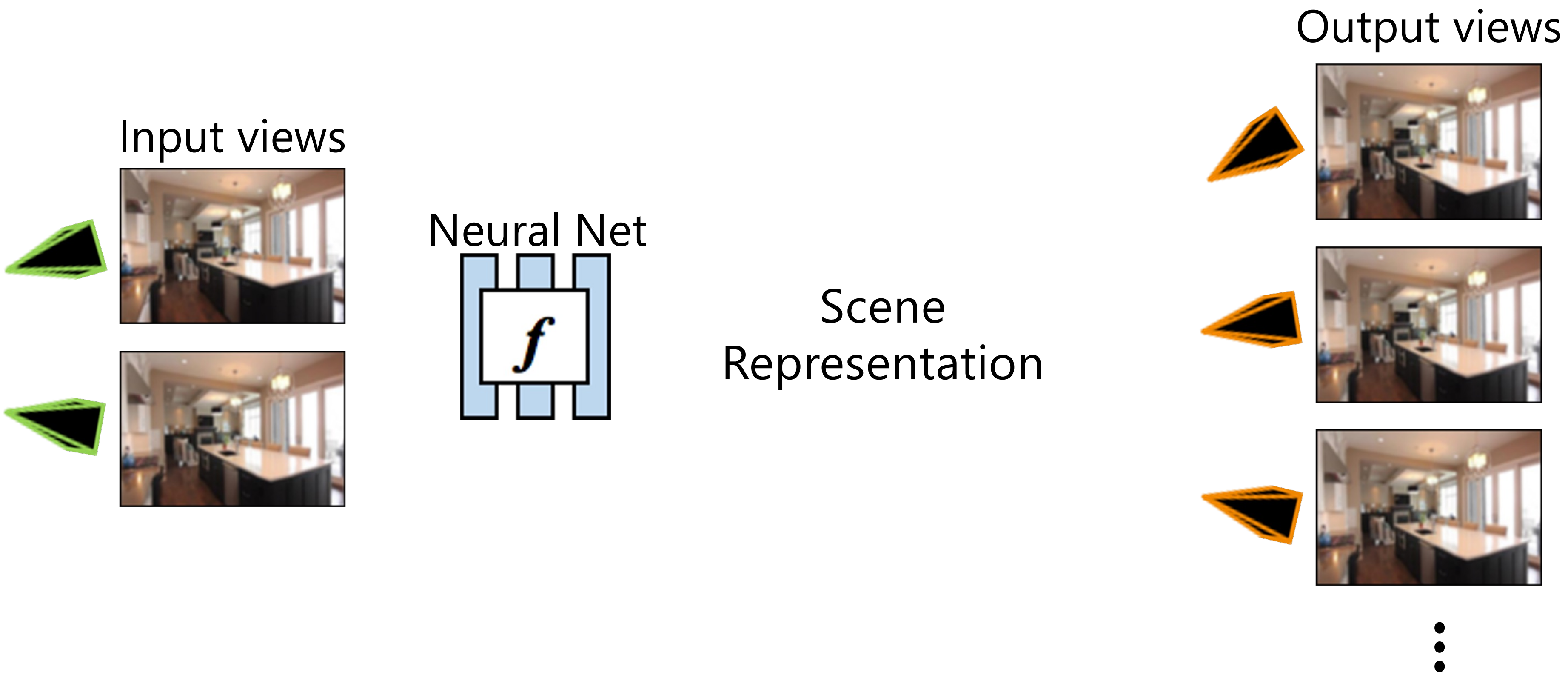
Large disocclusion



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
https://tinghuiz.github.io/projects/mpi/

Computer vision as inverse rendering

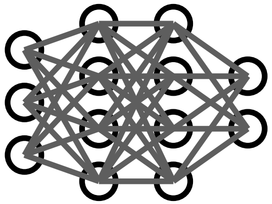
Inverse Rendering

3D
Scene
Representation

Rendering

Paradigm 1: Feedforward

Use a “Feedforward” network for inverse rendering.



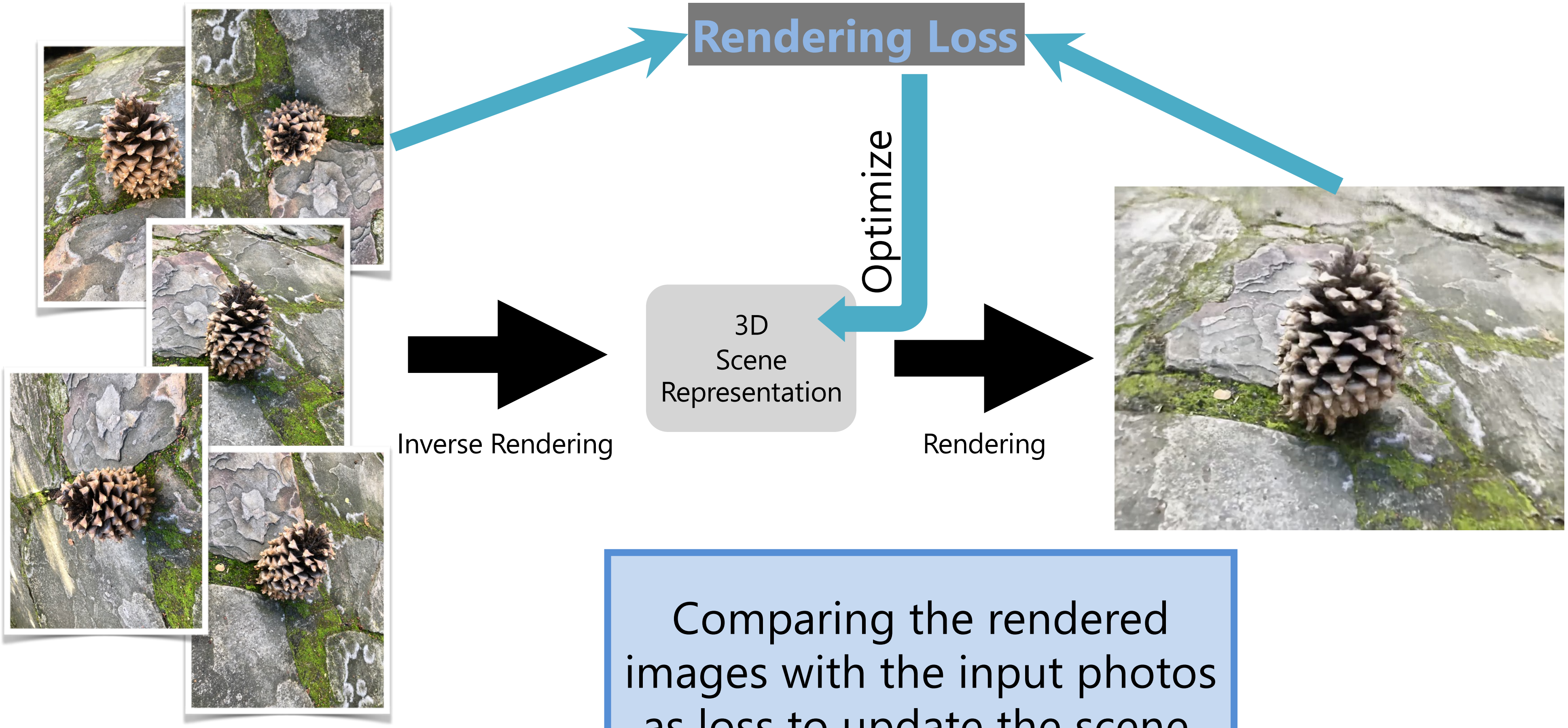
Inverse Rendering

3D Scene Representation

Rendering

→ Fast inference for small networks

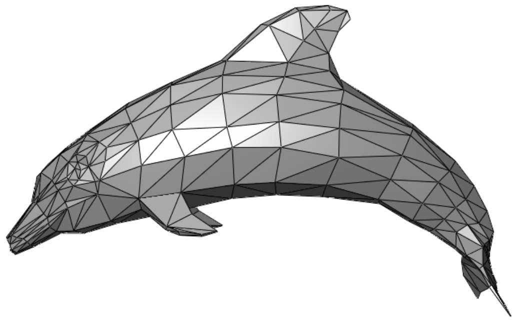
Paradigm 2: “Render-and-compare”



Comparing the rendered images with the input photos as loss to update the scene.

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

Multiplane Camera (1937)

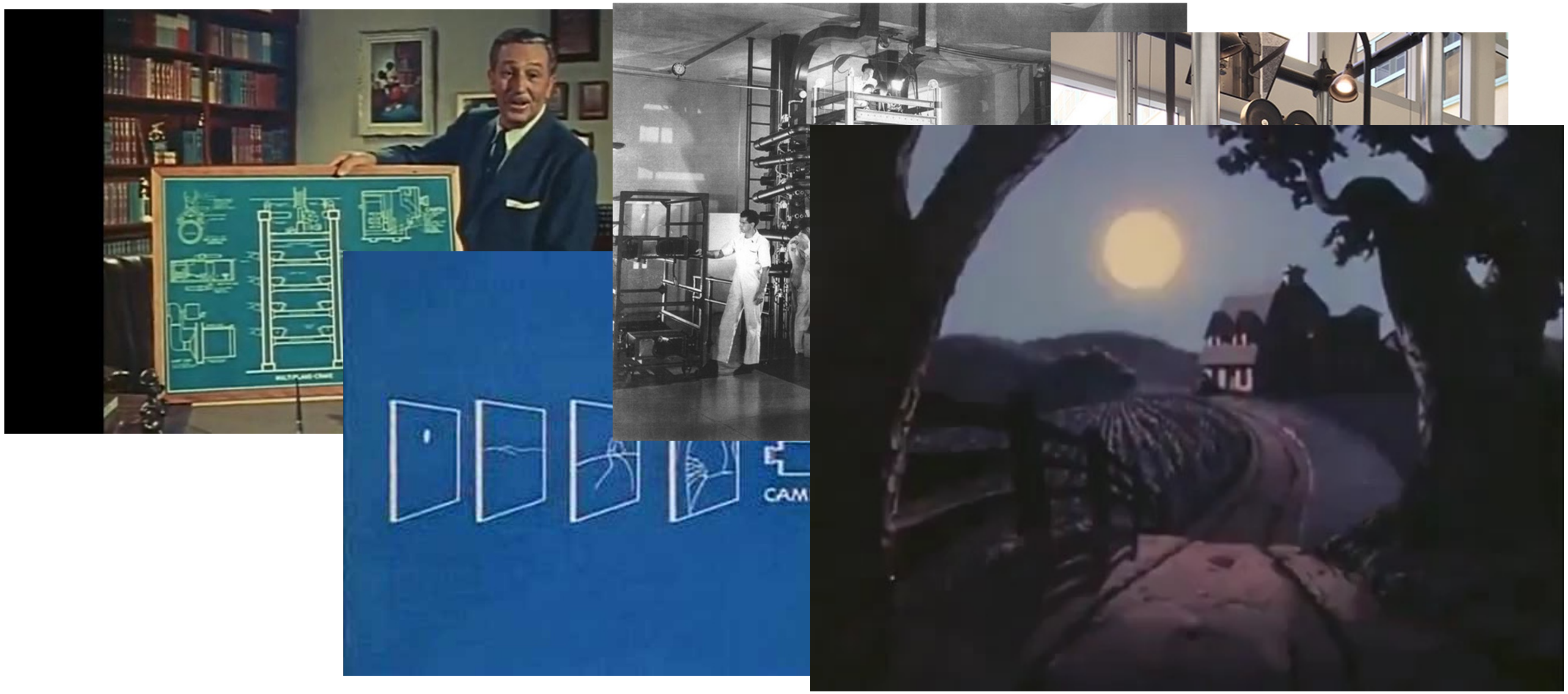
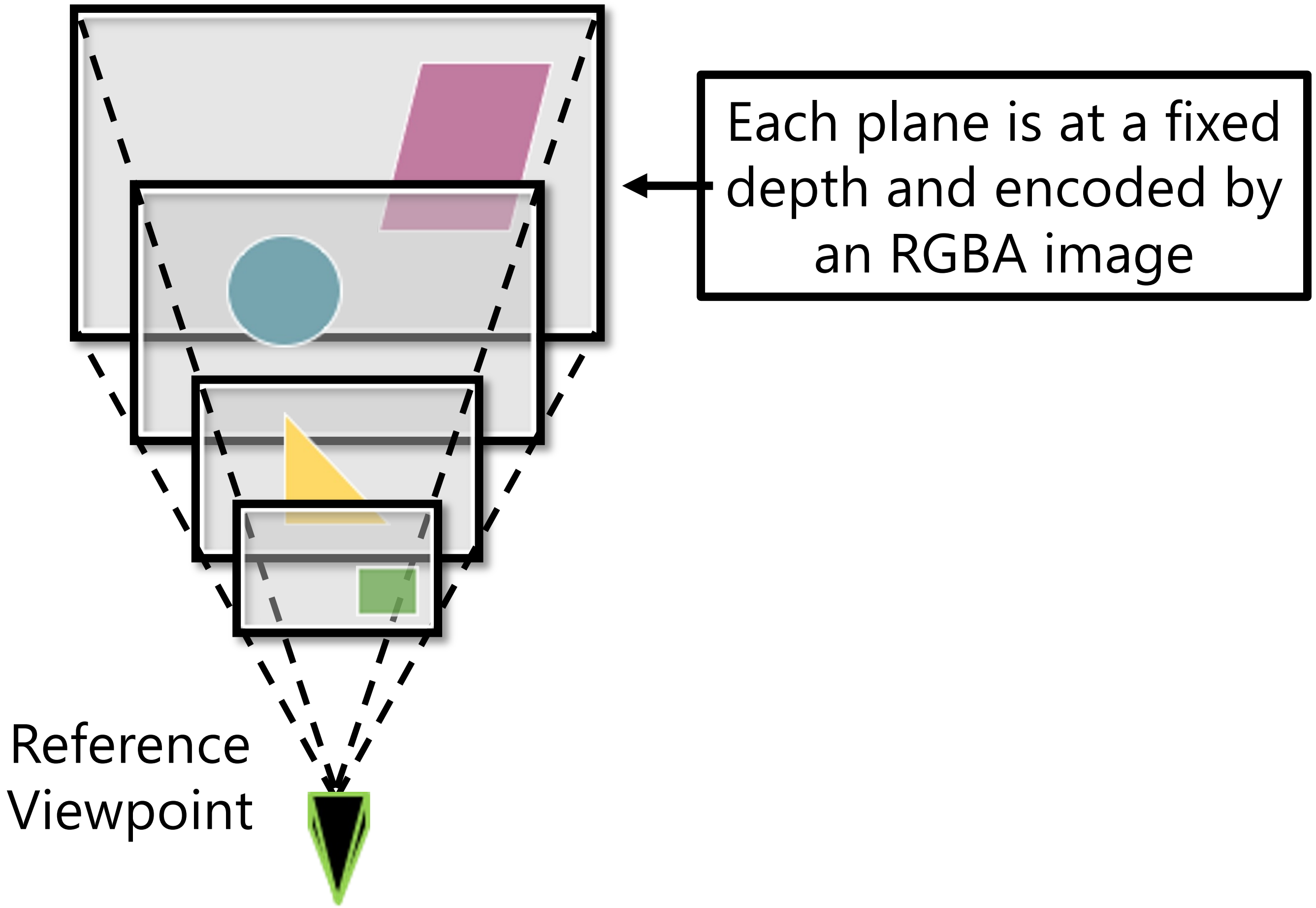


Image credits: Disney https://www.youtube.com/watch?v=kN-eCBAOw60 (from 1957)

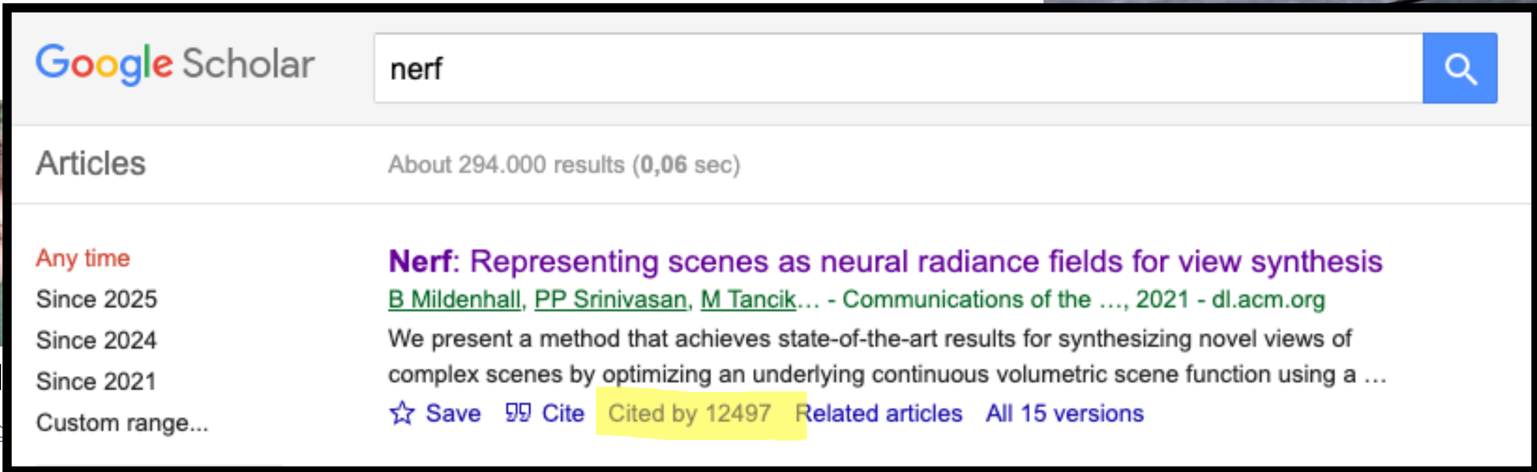
Multiplane Images (MPIs)



NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis
ECCV 2020

Ben Milden

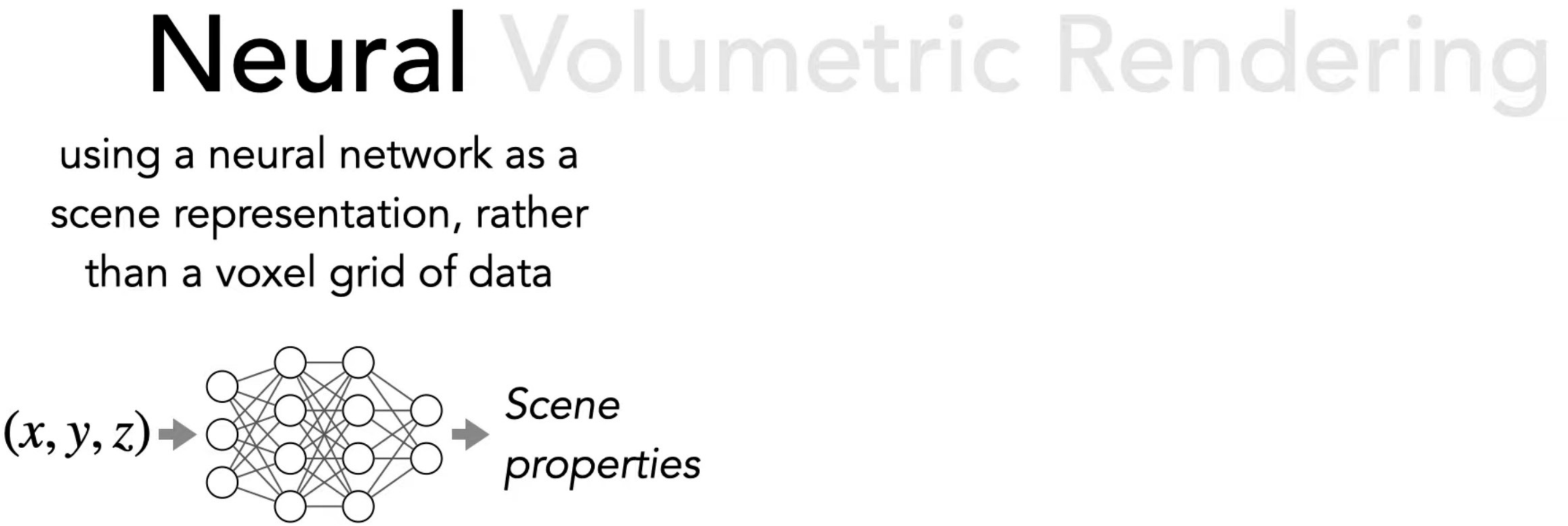
UC Berkel



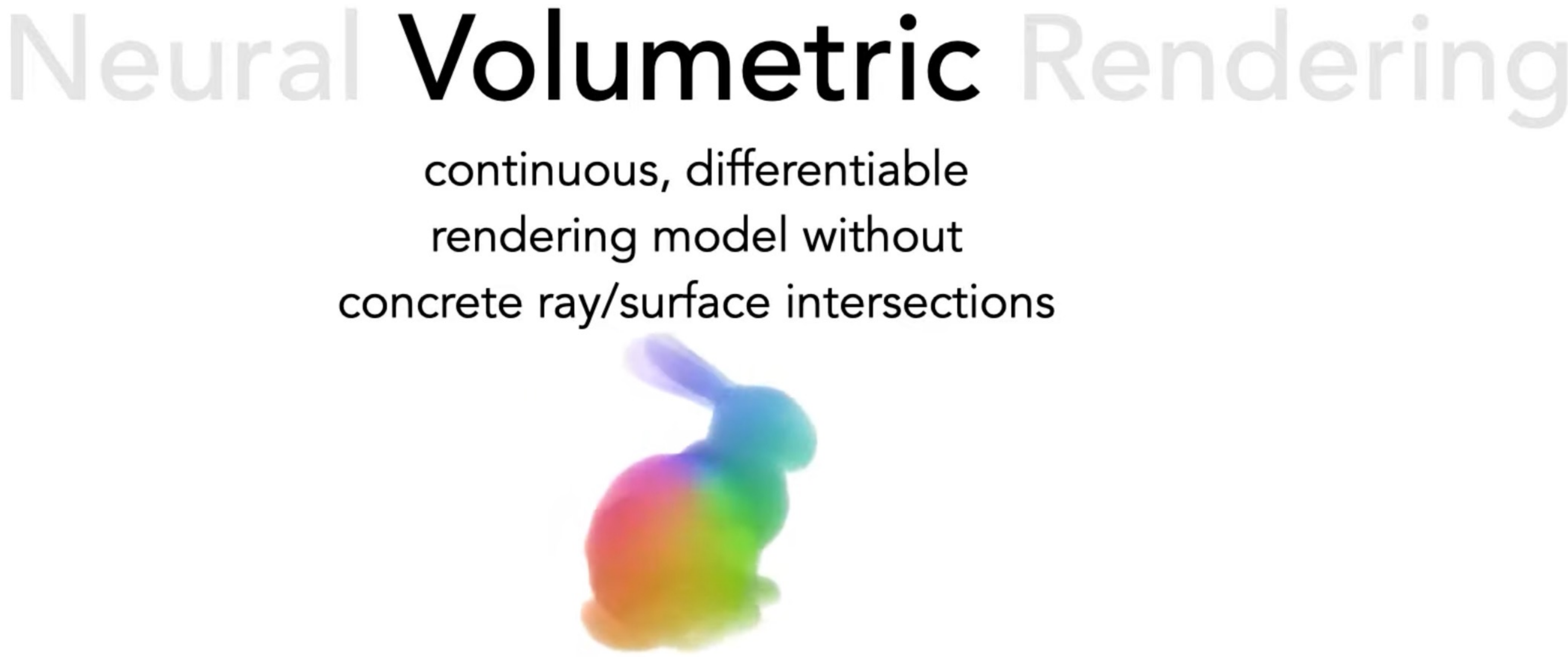
Ren Ng

UC Berkeley

Neural Volumetric Rendering



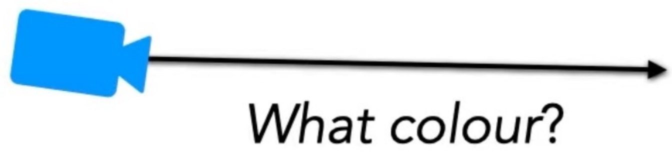
Neural Volumetric Rendering



Neural Volumetric Rendering

Neural Volumetric **Rendering**

querying the radiance value
along rays through 3D space



NeRF means

DIFFERENTIABLE RENDERING WITH A NEURAL VOLUMETRIC REPRESENTATION

Results

From the original NeRF paper: https://www.matthewtancik.com/nerf

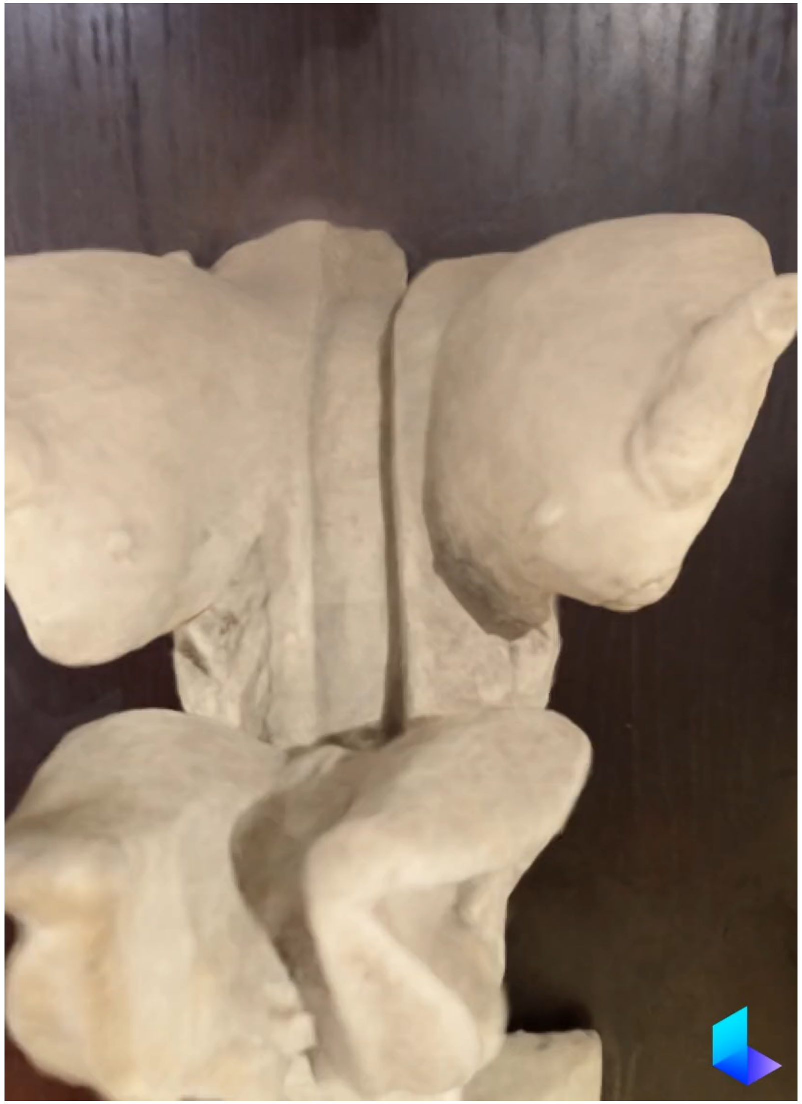
Results

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

Results

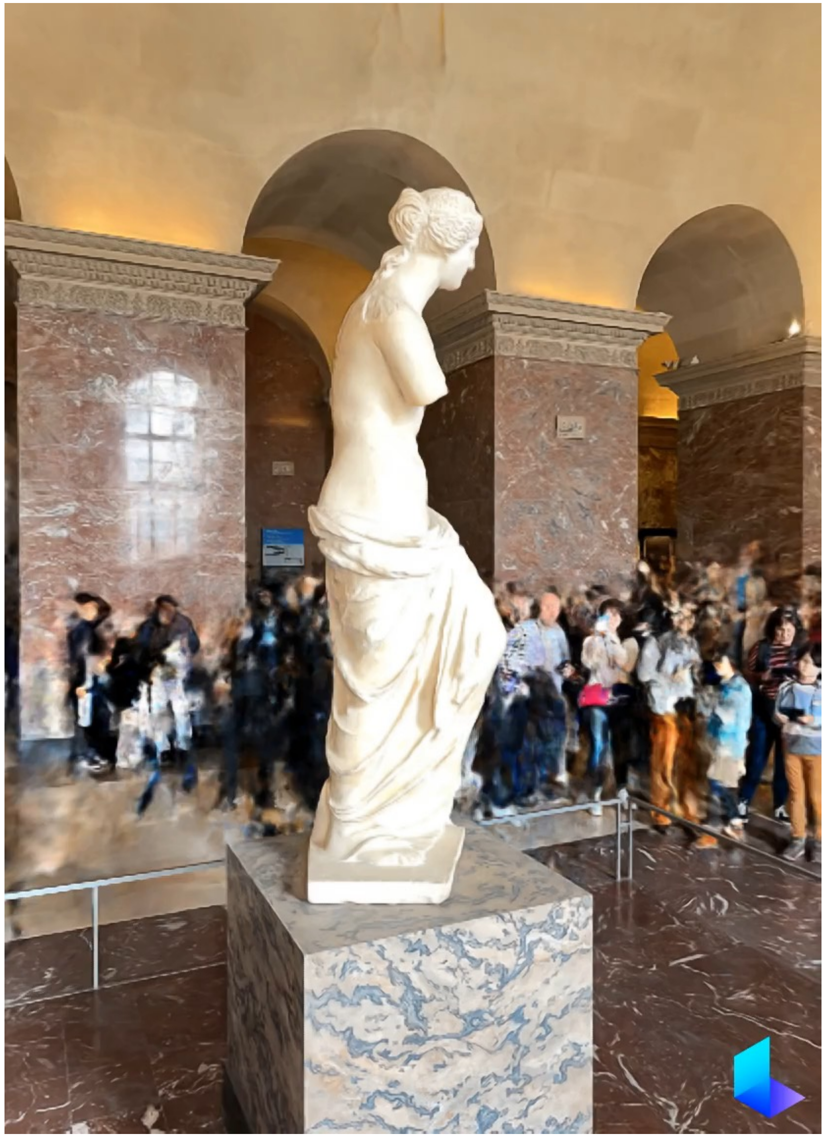
Barron et al 2021, Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields

Results



Made with LumaAI by Uwe Hahne, Ulm 2023

Results



Made with LumaAI by Uwe Hahne, Paris 2023

Results

Barron et al 2023, Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields

Idea

Optimize a NeRF model

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

3D reconstruction viewable from any angle

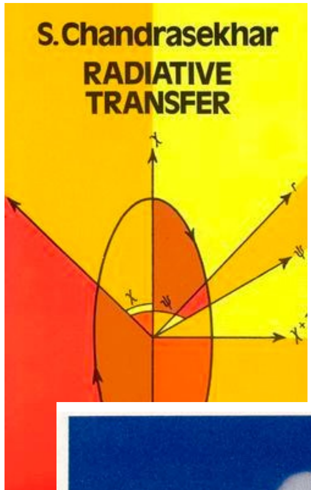
NeRF Overview

- 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

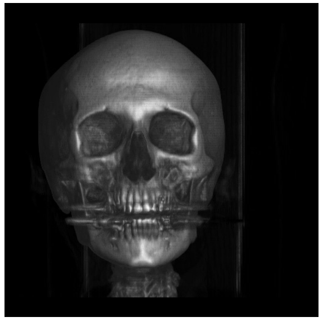



Ray tracing simulated cumulus cloud [Kajiya]

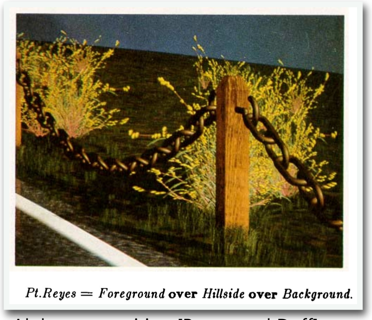
Chandrasekhar 1950, Radiative Transfer
Kajiya 1984, Ray Tracing Volume Densities

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

Traditional volumetric rendering



Medical data visualisation [Levoy]



Alpha compositing [Porter and Duff]

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

- 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

Traditional volumetric rendering

- 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


Physically-based Monte Carlo rendering [Novak et al]

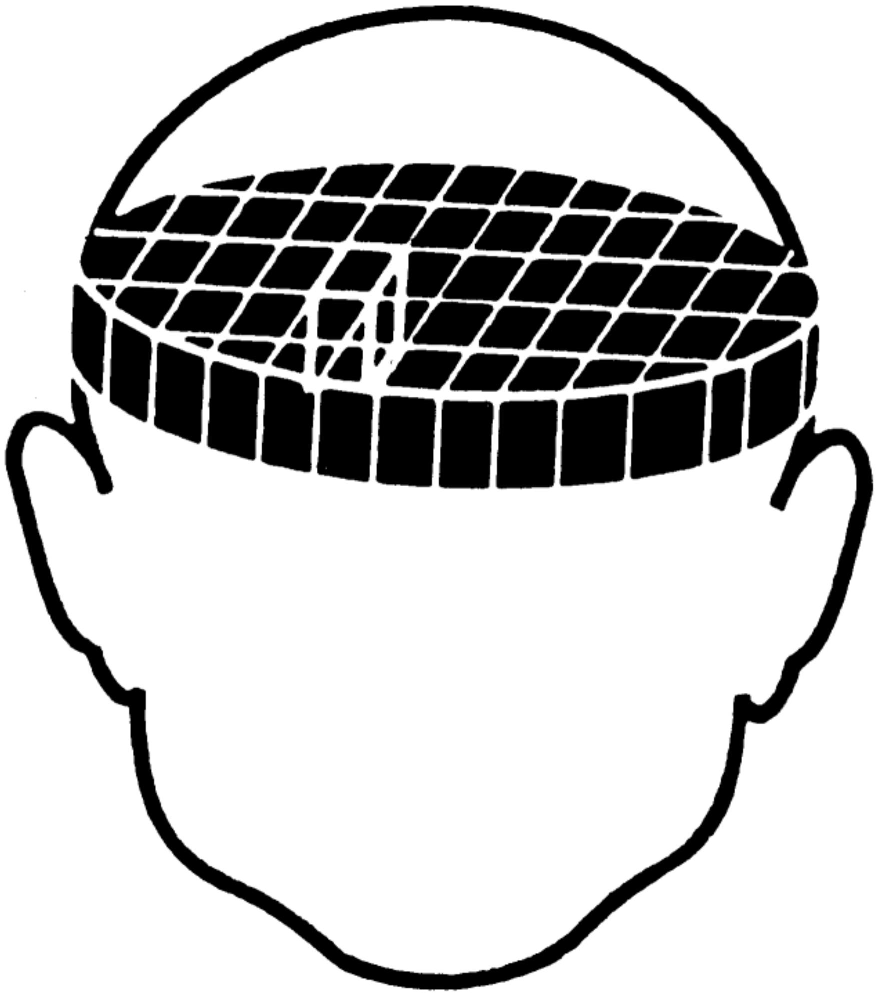
Chandrasekhar 1950, Radiative Transfer
Kajiya 1984, Ray Tracing Volume Densities
Levoy 1988, Display of Surfaces from Volume Data
Max 1995, Optical Models for Direct Volume Rendering
Porter and Duff 1984, Compositing Digital Images
Novak et al 2018, Monte Carlo methods for physically based volume rendering

Side topic: Medical imaging (CT)

Image from Toshiba: http://www.trafita.it/images/stories/TOSHIBA/toshiba16.pdf

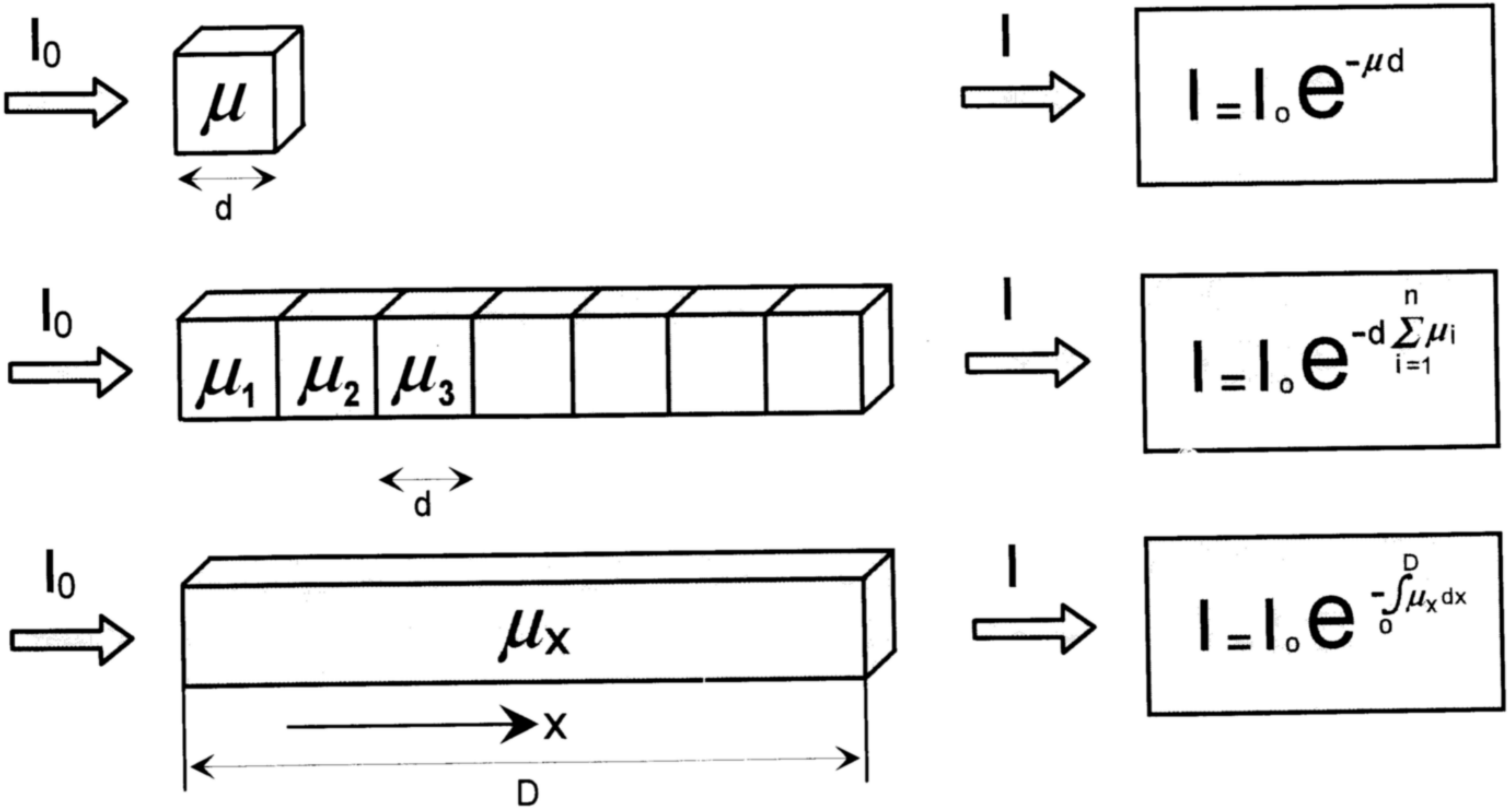
Medical Volume Image Rendering

Using voxel representation



X-ray absorption coefficient

μ: Value of one voxel



Projection profiles

- For each measuring beam j a measuring signal P_j is obtained as a projection of all picture elements along the measuring beam j in the body cross section.

Algebraic reconstruction

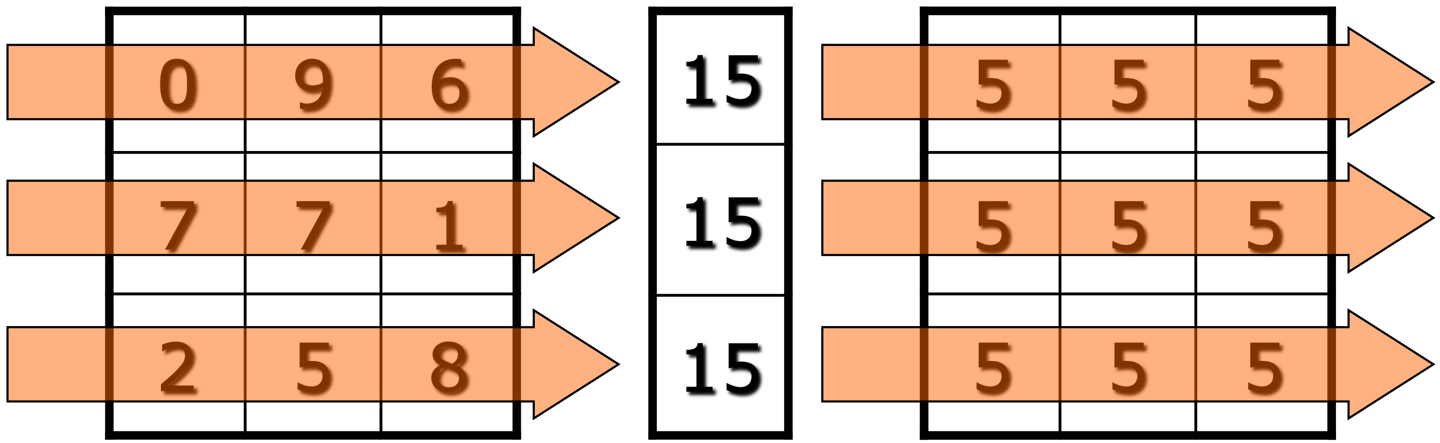
- Unknown original is approximated iteratively in 3 steps:
  1. estimation, 2. correction, 3. iteration

1. first approximation is derived from starting direction
2. creation of correction profiles
3. iterate over all directions

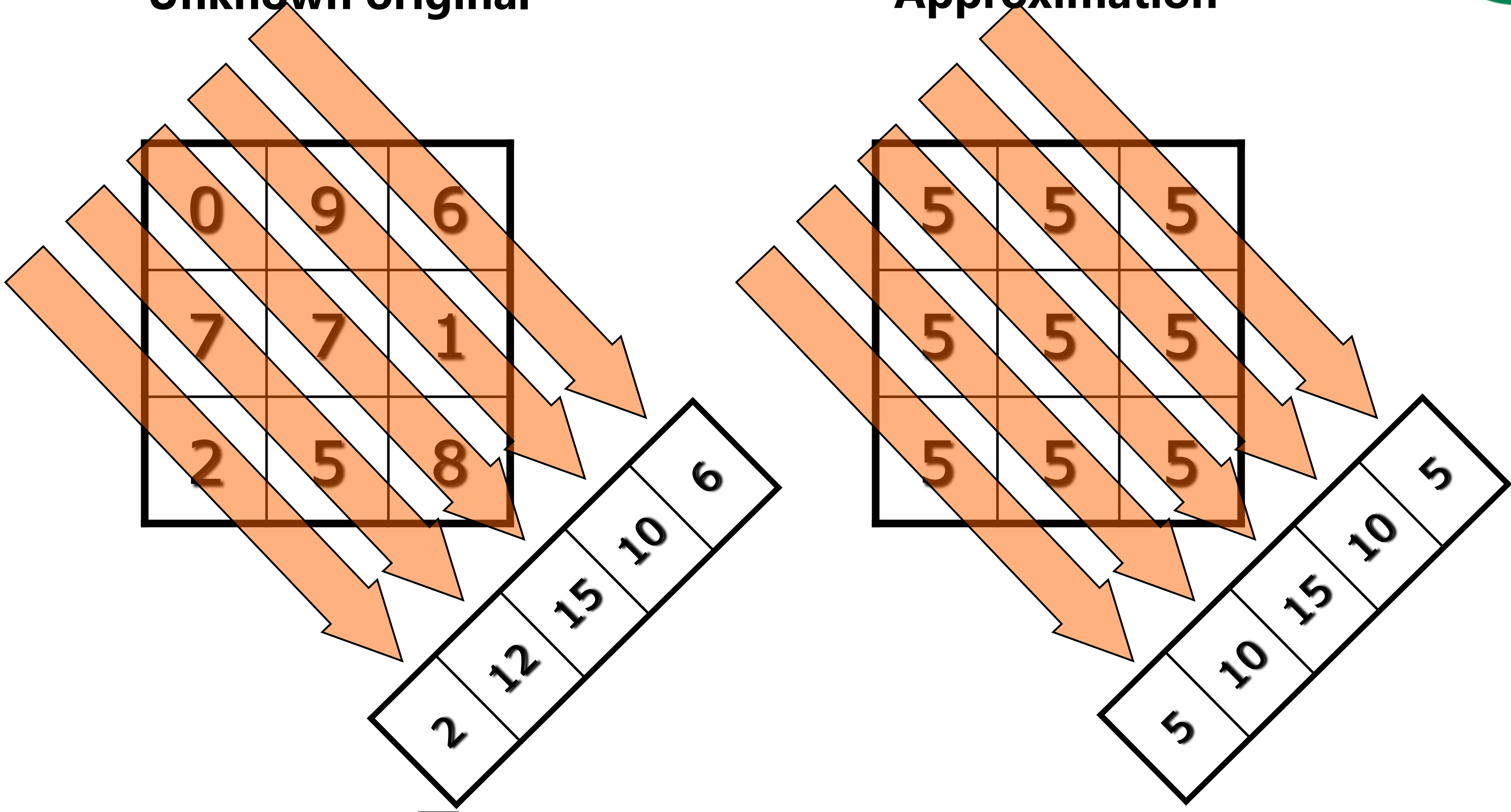
   termination criterion: by presetting an error measure or maximum number of iterations

Unknown original

Approximation



Unknown original **Approximation**

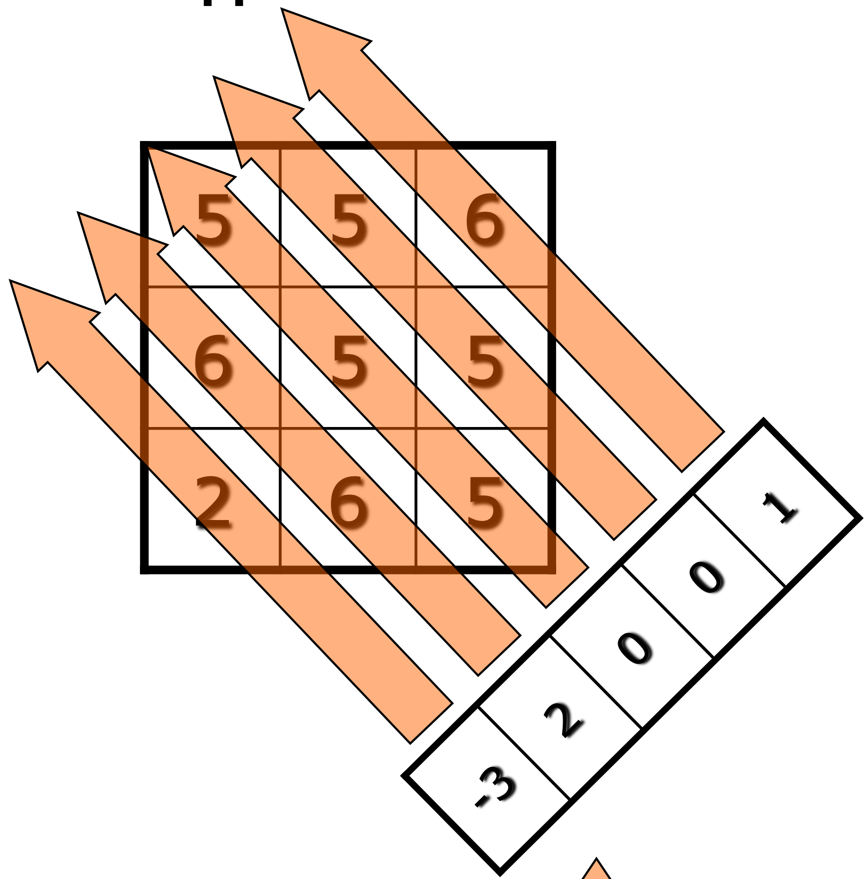


2	12	15	10	6
5	10	15	10	5
-3	2	0	0	1

Unknown original

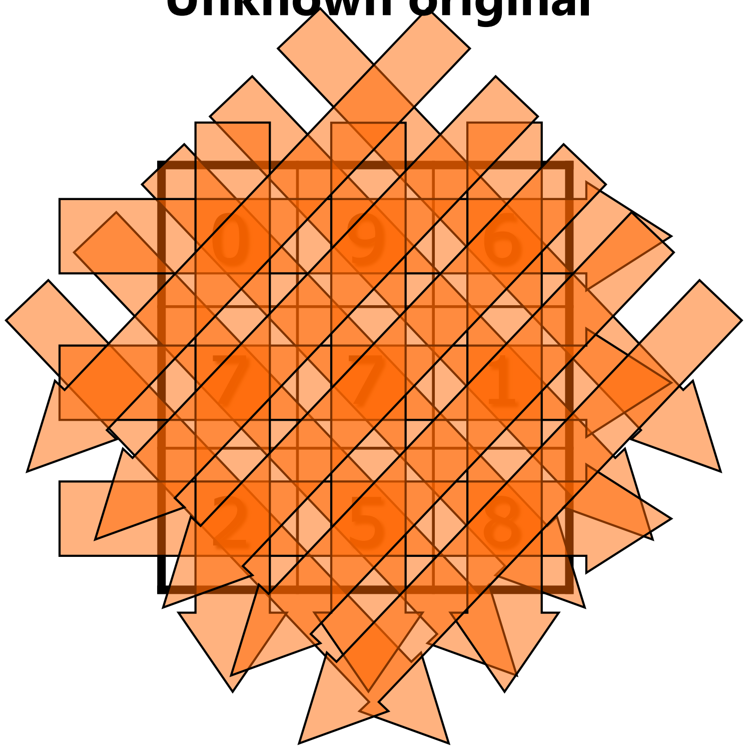
0	9	6
7	7	1
2	5	8

Approximation

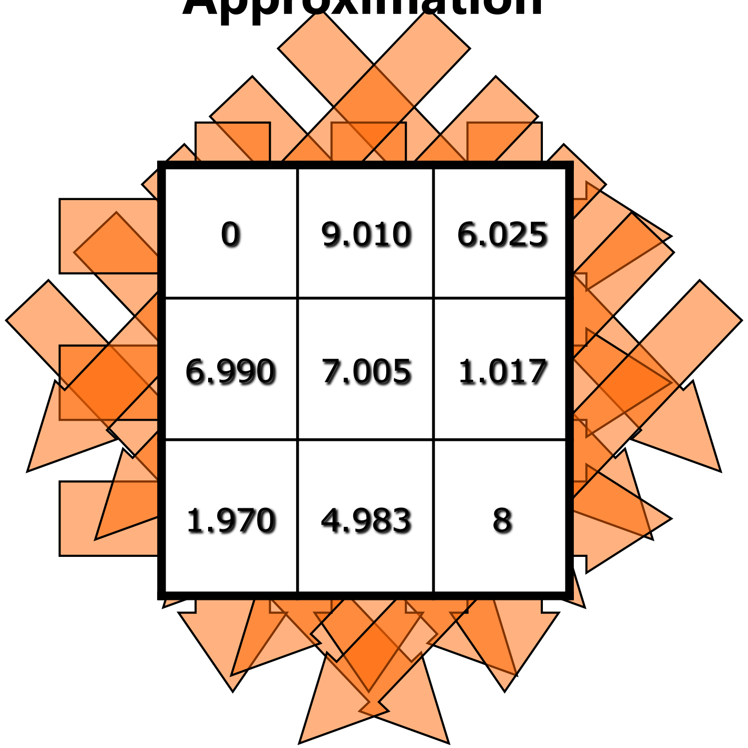


2	12	15	10	6
5	10	15	10	5
-3	2	0	0	1

Unknown original

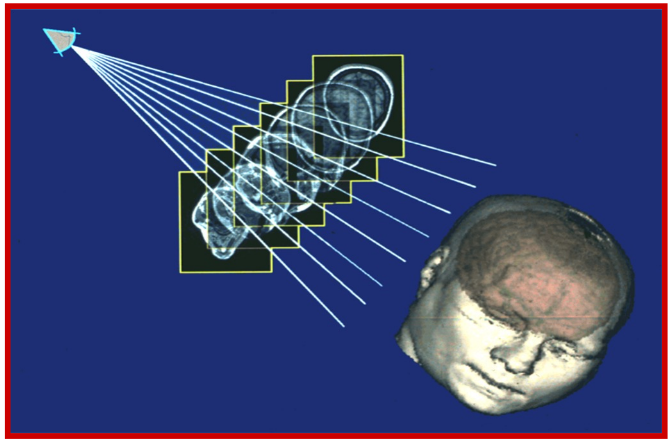


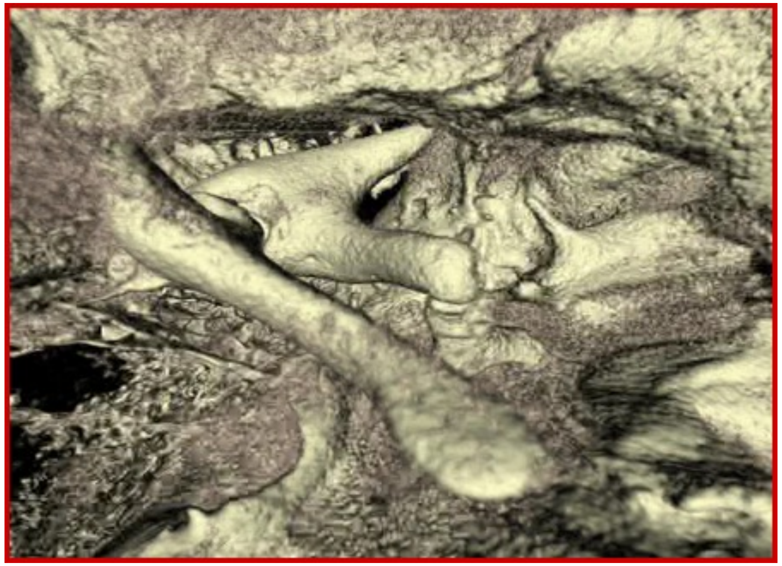
Approximation

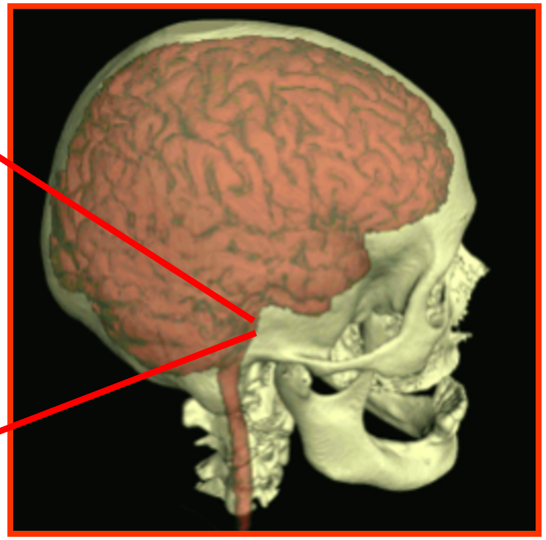


0	9.010	6.025
6.990	7.005	1.017
1.970	4.983	8

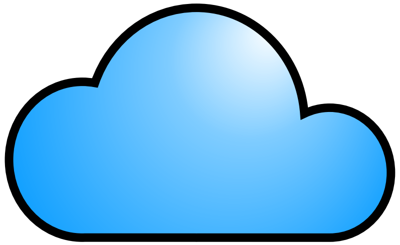
Medical Volume Image Rendering







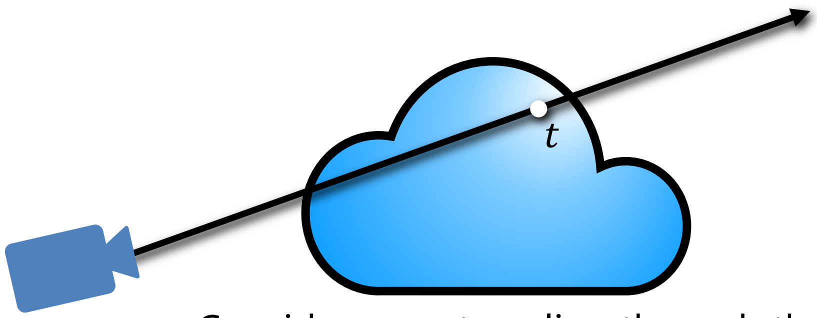
Volumetric formulation for NeRF



Scene is a cloud of colored fog

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

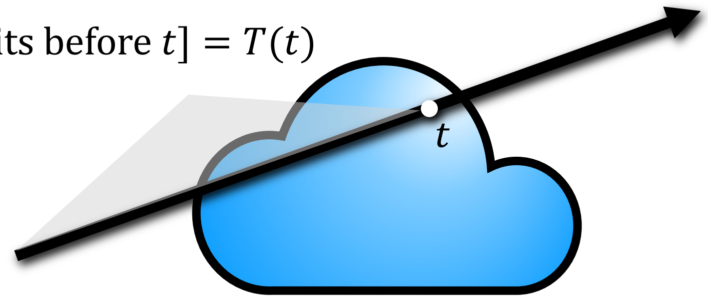
Volumetric formulation for NeRF



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) $\boldsymbol{\alpha}(t)$

Volumetric formulation for NeRF

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



But t may also be blocked by earlier points along the ray.
$T(t)$: probability that the ray didn't hit any particles earlier.
$T(t)$ is called "transmittance"

Volume rendering estimation: integrating color along a ray

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

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

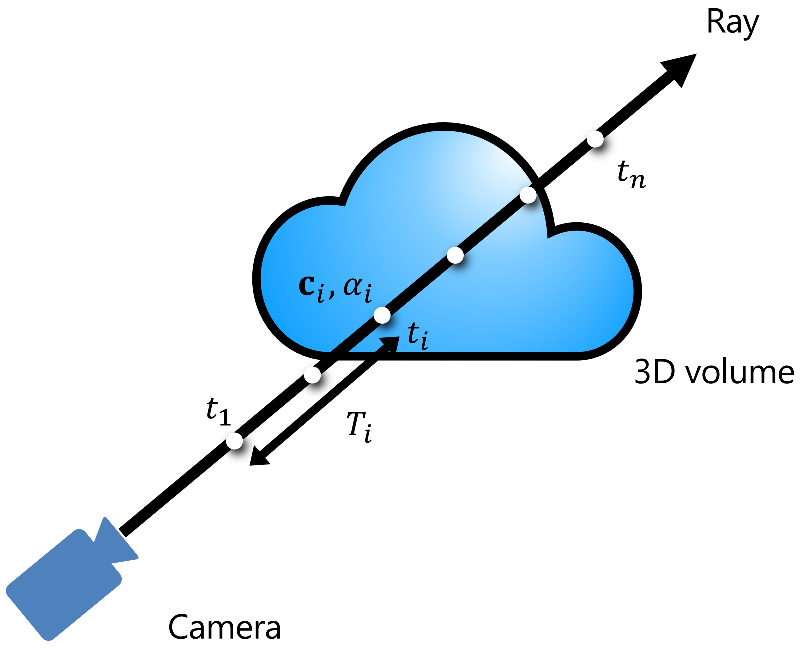
final rendered color along ray

weights

colors

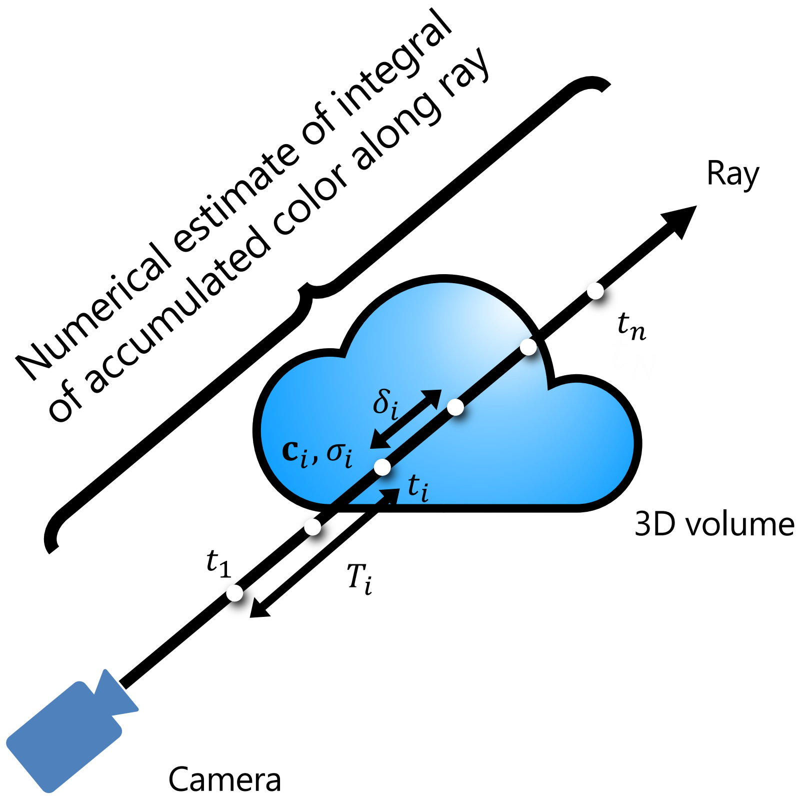
How much light is blocked earlier along ray:

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



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

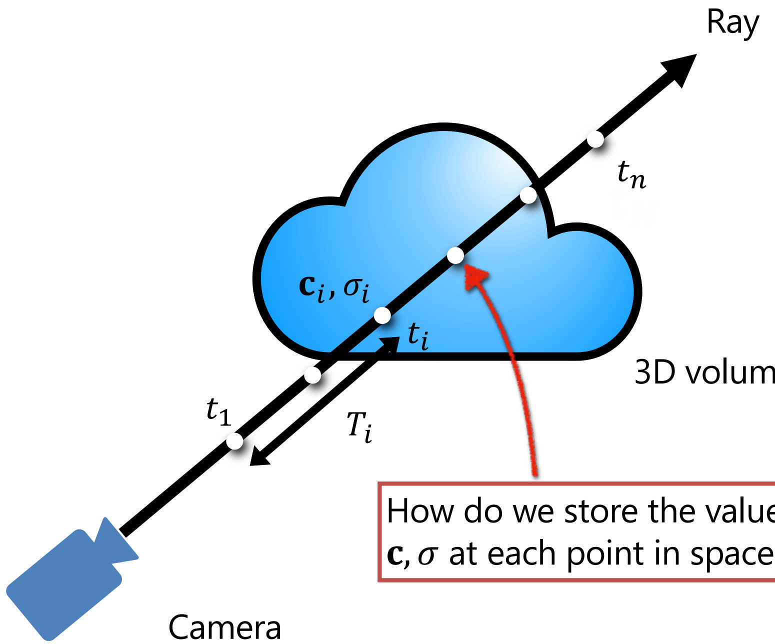
Volume rendering estimation: integrating color along a ray



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



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

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)

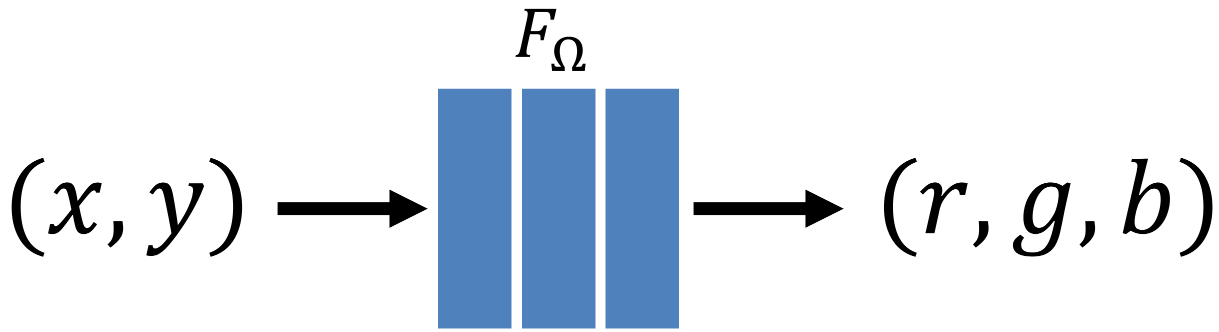
Toy problem: storing 2D image data

(x, y)

(r, g, b)

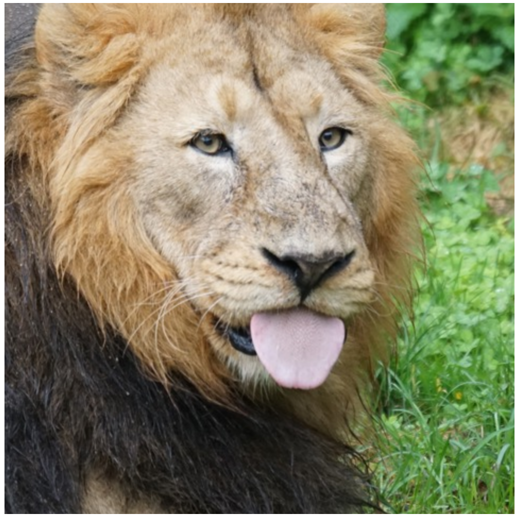
Usually we store an image as a
2D grid of RGB color values

Toy problem: storing 2D image data



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

Naive approach fails!



Ground truth image

Neural network output fit
with gradient descent

Problem

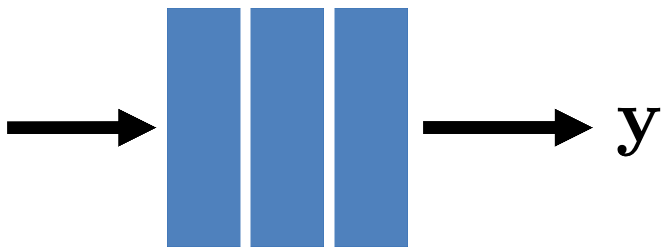
“Standard” coordinate-based MLPs cannot represent high frequency functions.

Solution

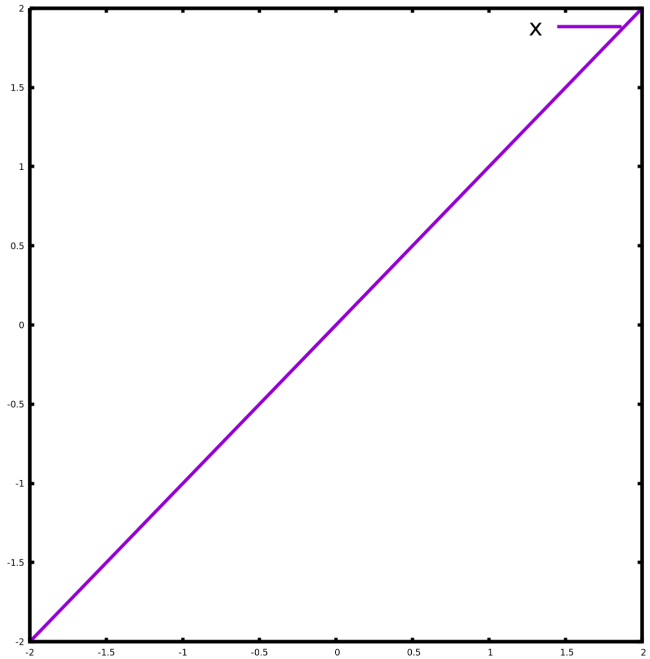
Pass input coordinates through a high frequency mapping first.

Example mapping: “positional encoding”

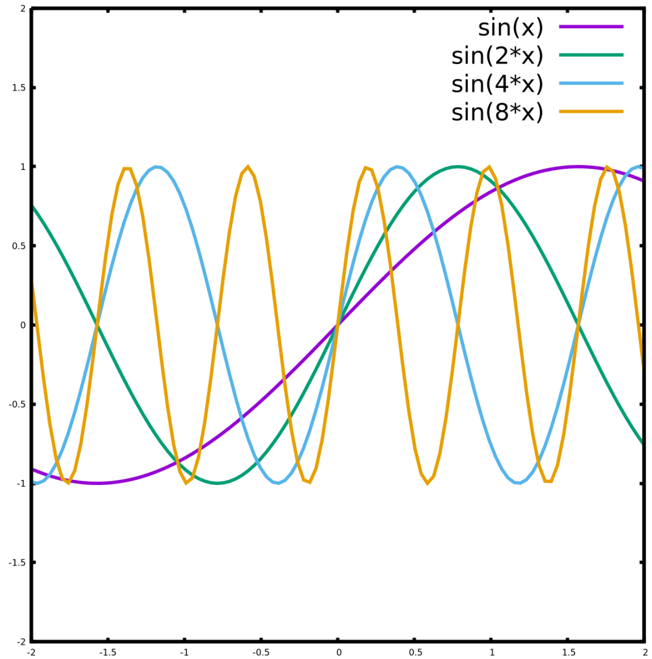
$$\begin{pmatrix} \sin(\mathbf{v}), \cos(\mathbf{v}) \\ \sin(2\mathbf{v}), \cos(2\mathbf{v}) \\ \sin(4\mathbf{v}), \cos(4\mathbf{v}) \\ \dots \\ \sin(2^{L-1}\mathbf{v}), \cos(2^{L-1}\mathbf{v}) \end{pmatrix}$$



Positional encoding

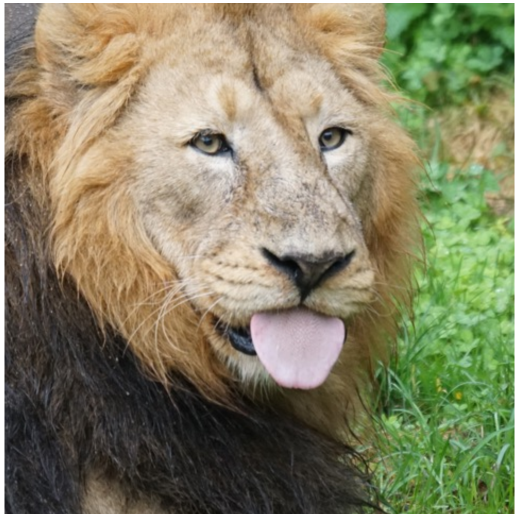


Raw encoding of a number ***x***



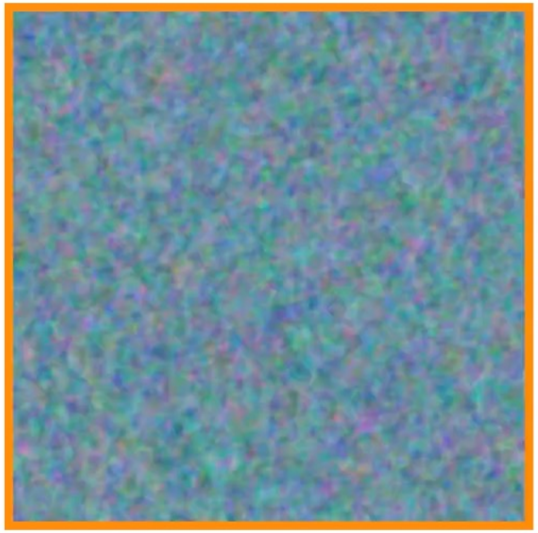
"Positional encoding" of a number ***x***

Problem solved!



Ground truth image

Neural network output without high frequency mapping



Neural network output with high frequency mapping

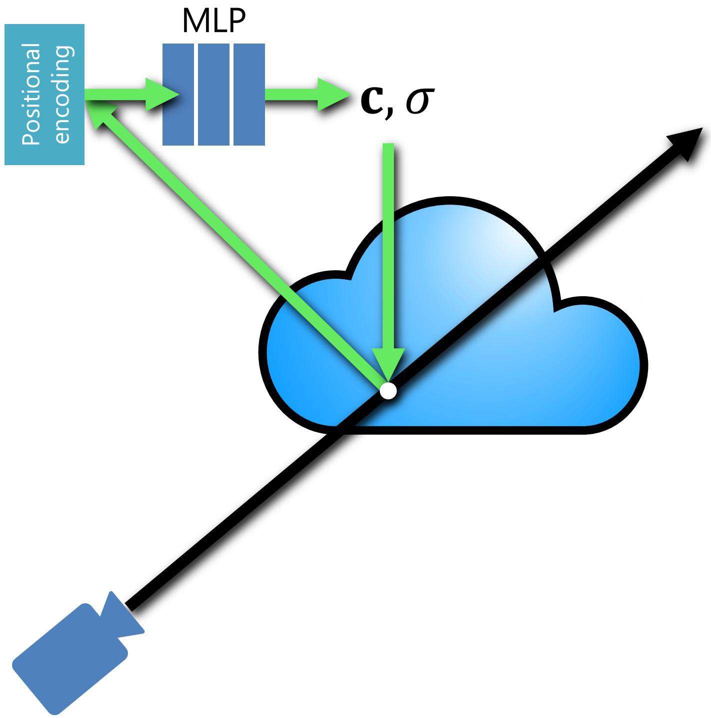
NeRF Overview

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

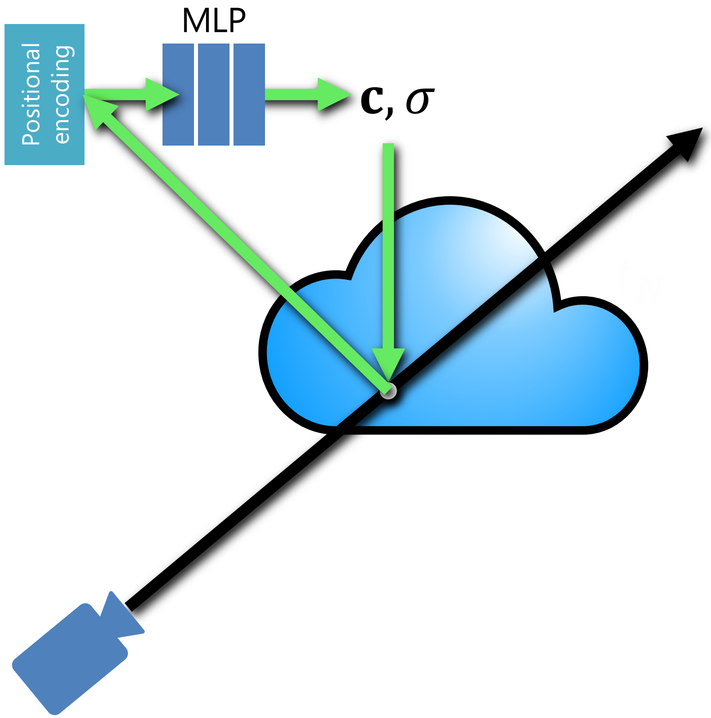
NeRF is a combination of

VOLUME RENDERING + COORDINATE-BASED NEURAL NETWORK

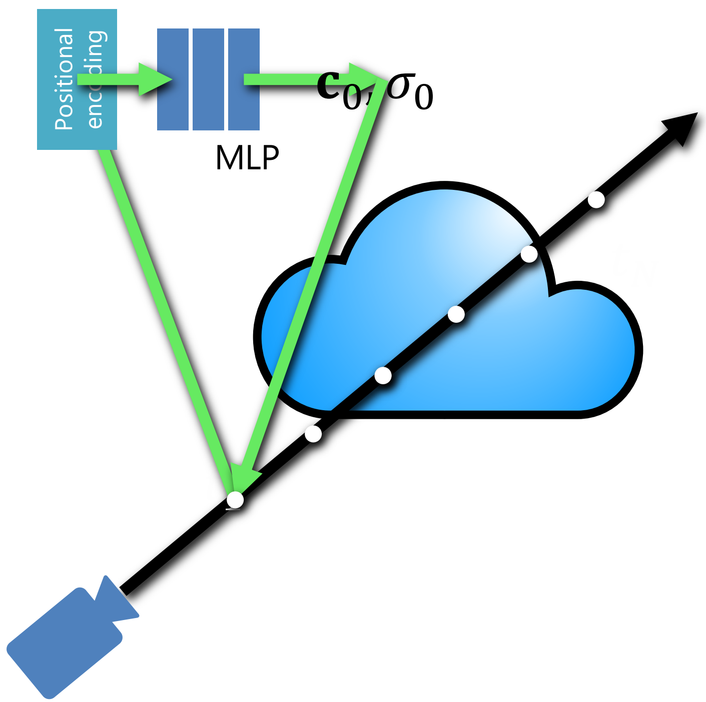
How do we store the values of $\mathbf{c}, \sigma$ at each point in space



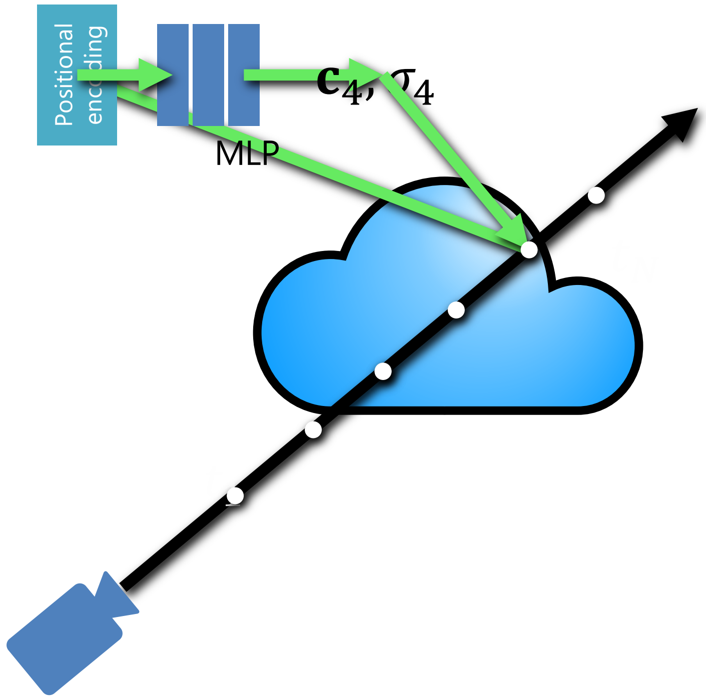
How do we store the values of $\mathbf{c}, \sigma$ at each point in space



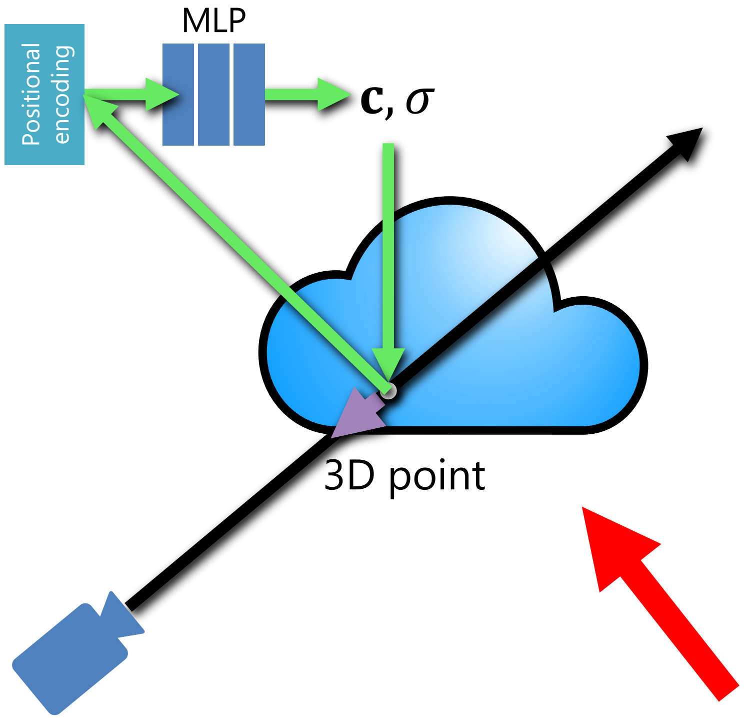
How do we store the values of $\mathbf{c}, \sigma$ at each point in space



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

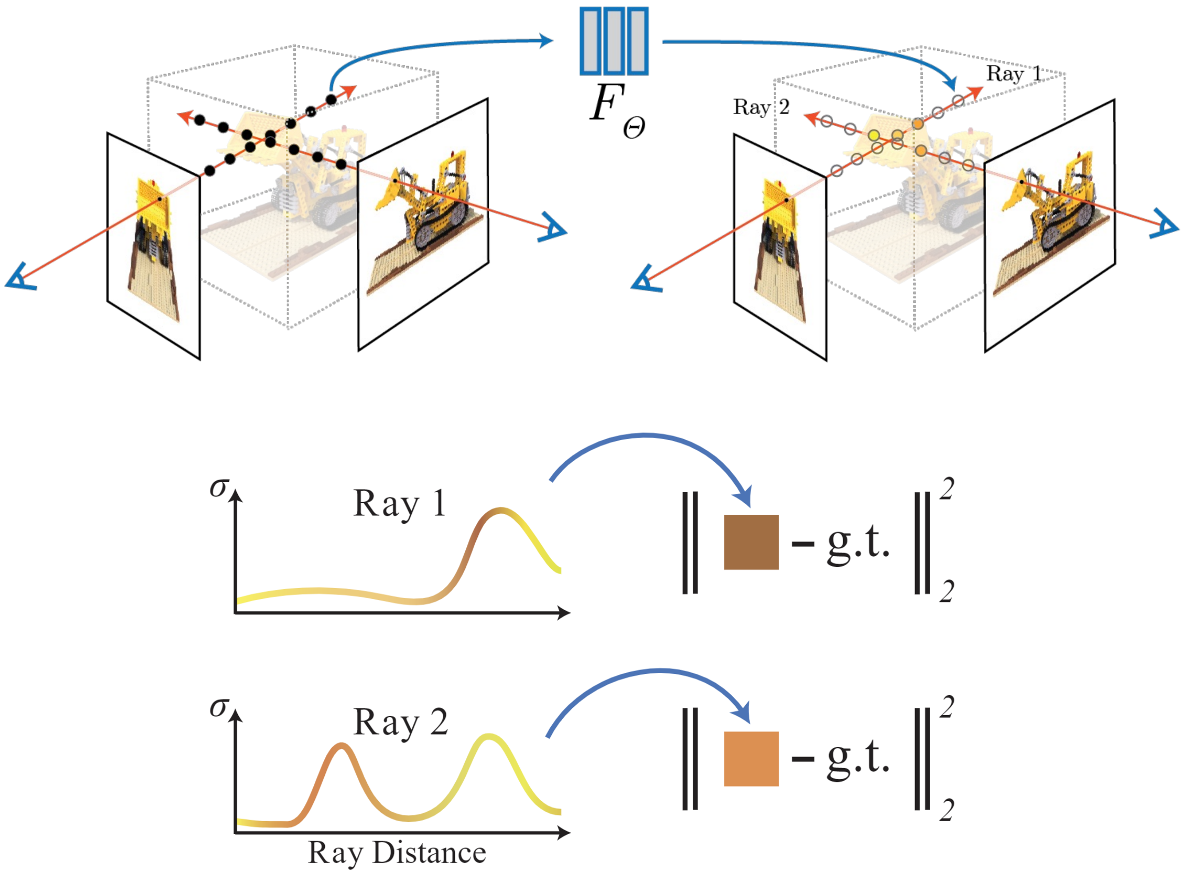


Extension: view-dependent field

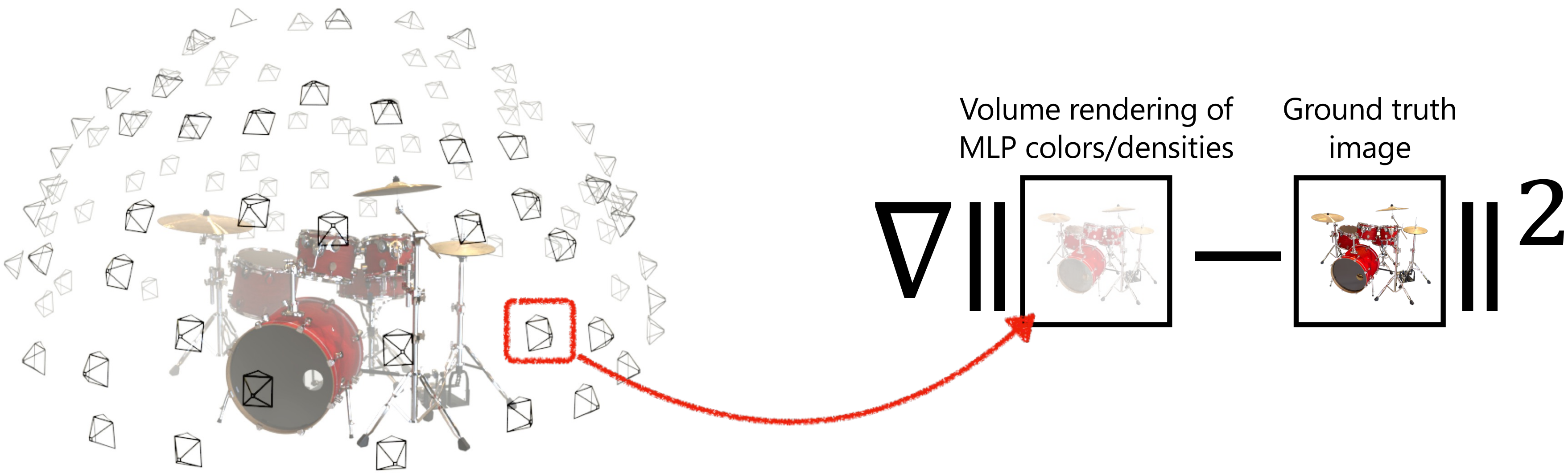


Include the ray direction in the input to the MLP → allows for capturing and rendering view-dependent effects (e.g., shiny surfaces)

Putting it all together



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



REPETITION AND RESULTS

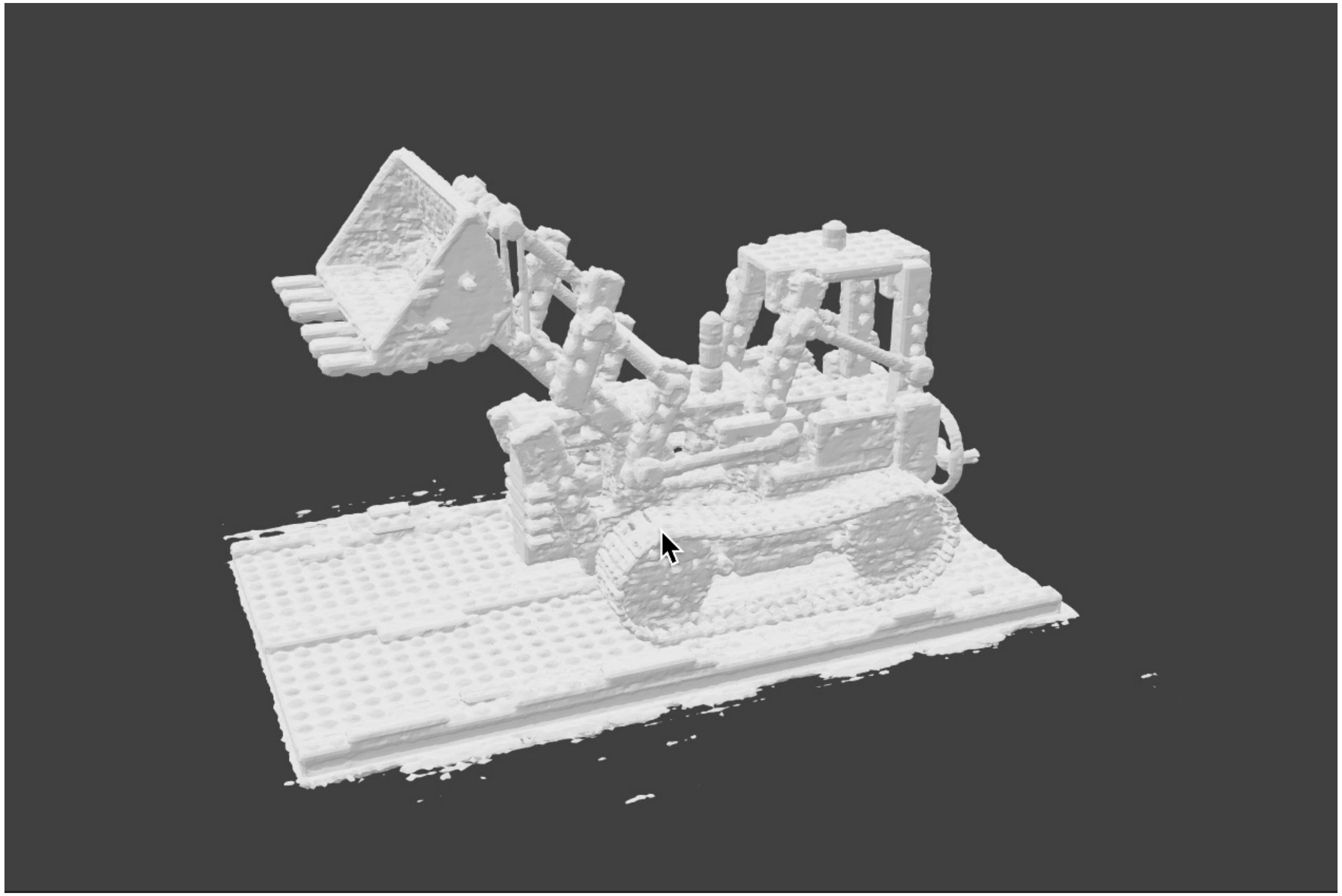
NeRF encodes convincing view-dependent effects using directional dependence

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