Neural Radiance Fields

CS194-26/294-26: Intro to Computer Vision and Computational Photography
Angjoo Kanazawa
UC Berkeley Fall 2022

Lots of content from ECCV 2022 Tutorial on Neural Volumetric Rendering for Computer Vision

Capturing Reality

Earliest cave painting (45,500 years old) in Sulawesi, Indonesia

Capturing Reality

Monet's Cathedral series: study of light 1893-1894

Capturing Reality

First self-portrait Cornelius 1839

First Movie - Muybridge 1878

Capturing Reality – in 3D

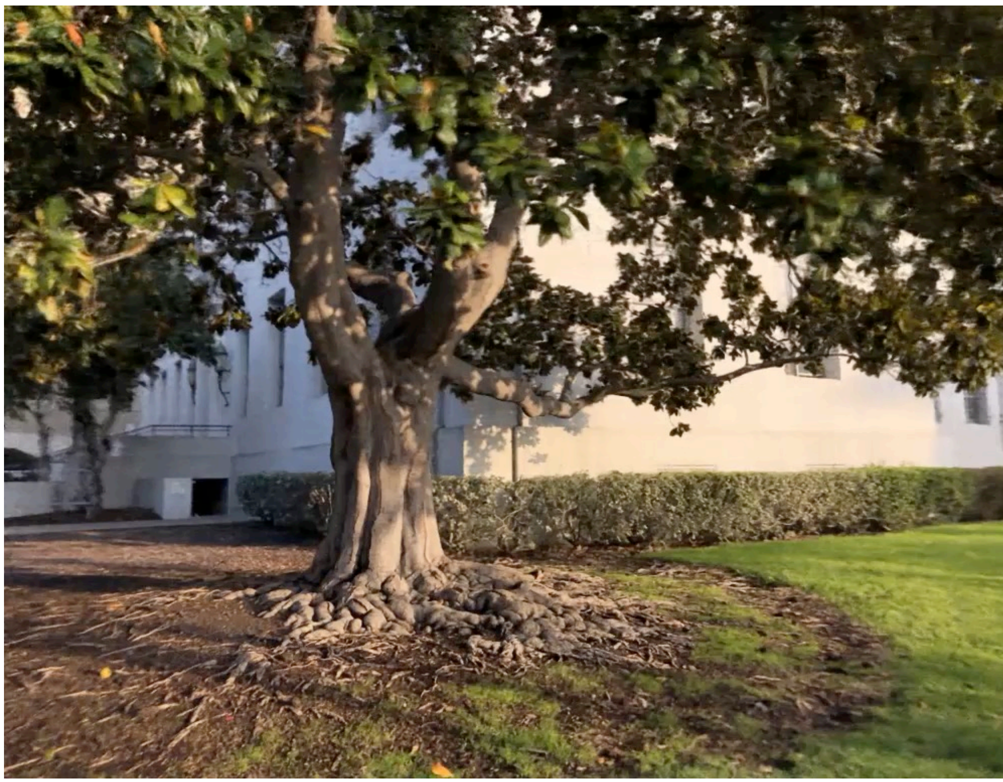
Building Rome in a Day, Agarwal et al. ICCV 2009

Capturing Reality – in 3D

Google Earth 2016~

What is next?

2020: Neural Radiance Field (NeRF)



Mildenhall*, Srinivasan*, Tancik*, Barron, Ramamoorthi, Ng, ECCV 2020

It has been two years

- Original NeRF paper: 1598 citations in 2 years

Handling Appearance Changes

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

Real-time Rendering

Video from PlenOctrees [Yu et al. CVPR 2021]

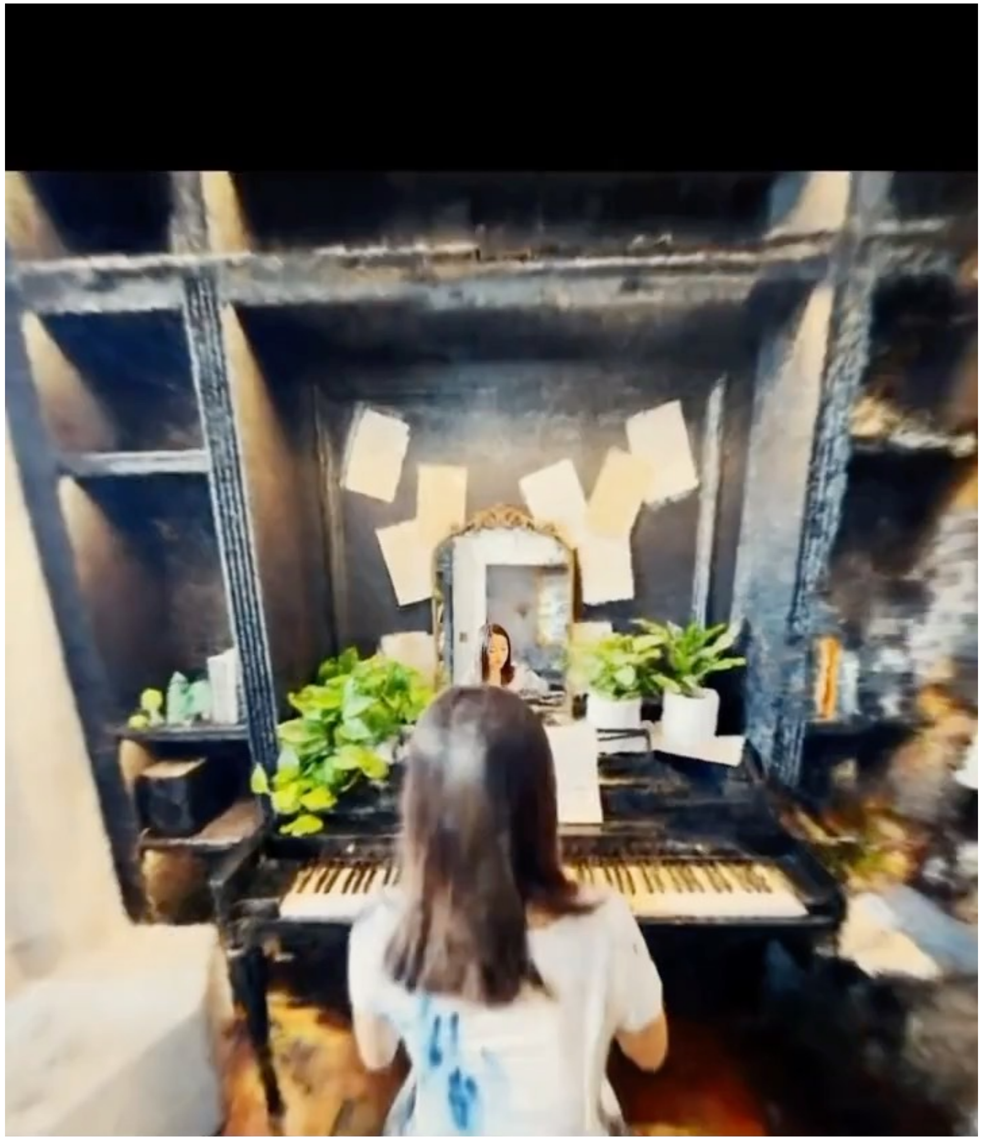
Real-time Inference

INSTANT NEURAL GRAPHICS PRIMITIVES WITH A MULTIRESOLUTION HASH ENCODING

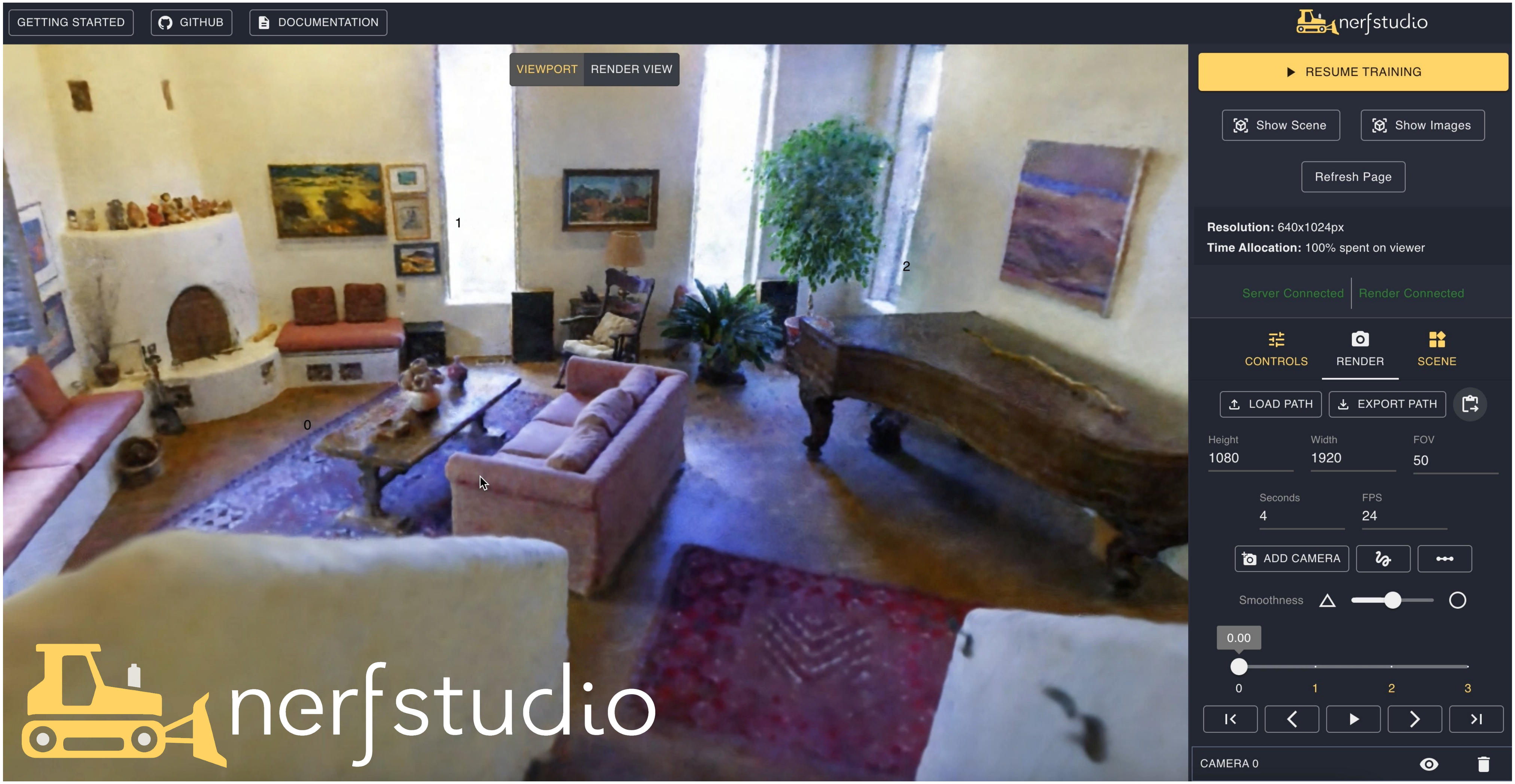
Thomas Müller   Alex Evans   Christoph Schied   Alexander Keller

https://nvlabs.github.io/instant-ngp

NVIDIA®



@karenxcheng, with InstantNGP [Müller et al., SIGGRAPH 2022]

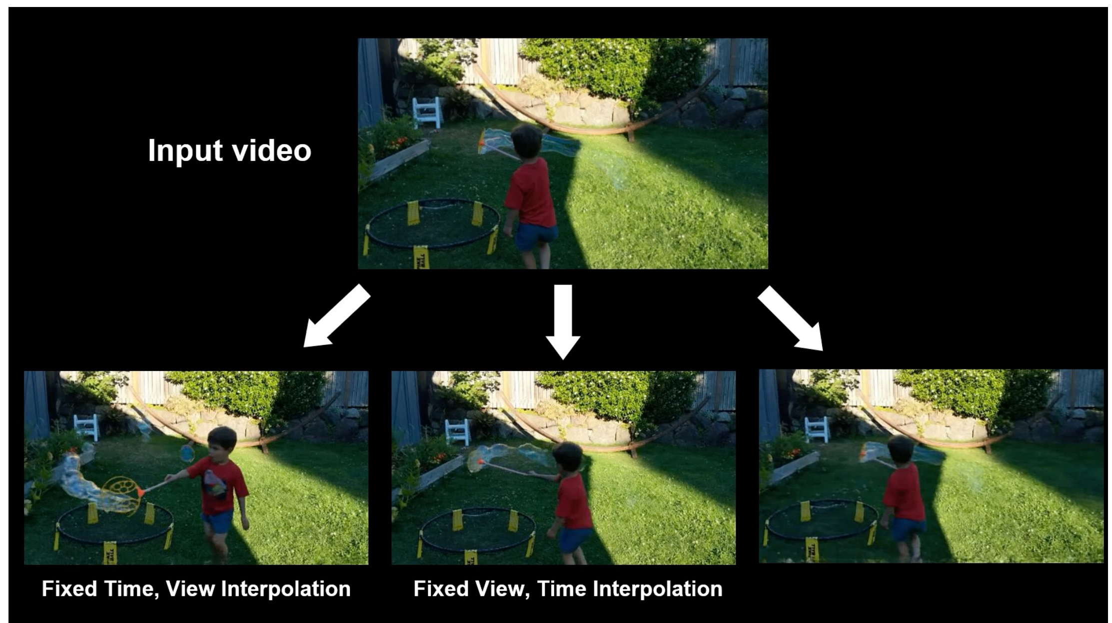

GETTING STARTED
GITHUB
DOCUMENTATION
nerfstudio
VIEWPORT
RENDER VIEW
RESUME TRAINING
Show Scene
Show Images
Refresh Page
Resolution: 640x1024px
Time Allocation: 100% spent on viewer
Server Connected
Render Connected
CONTROLS
RENDER
SCENE
LOAD PATH
EXPORT PATH
Height
1080
Width
1920
FOV
50
Seconds
4
FPS
24
ADD CAMERA
Smoothness
0.00
0
1
2
3
CAMERA 0
nerfstudio

Dynamic NeRFs

HyperNeRF [Park et al., SigAsia 2021]

Nerfies [Park et al., ICCV 2021]

[Xian et al., CVPR 2021]



NSFF [Li et al., CVPR 2021]

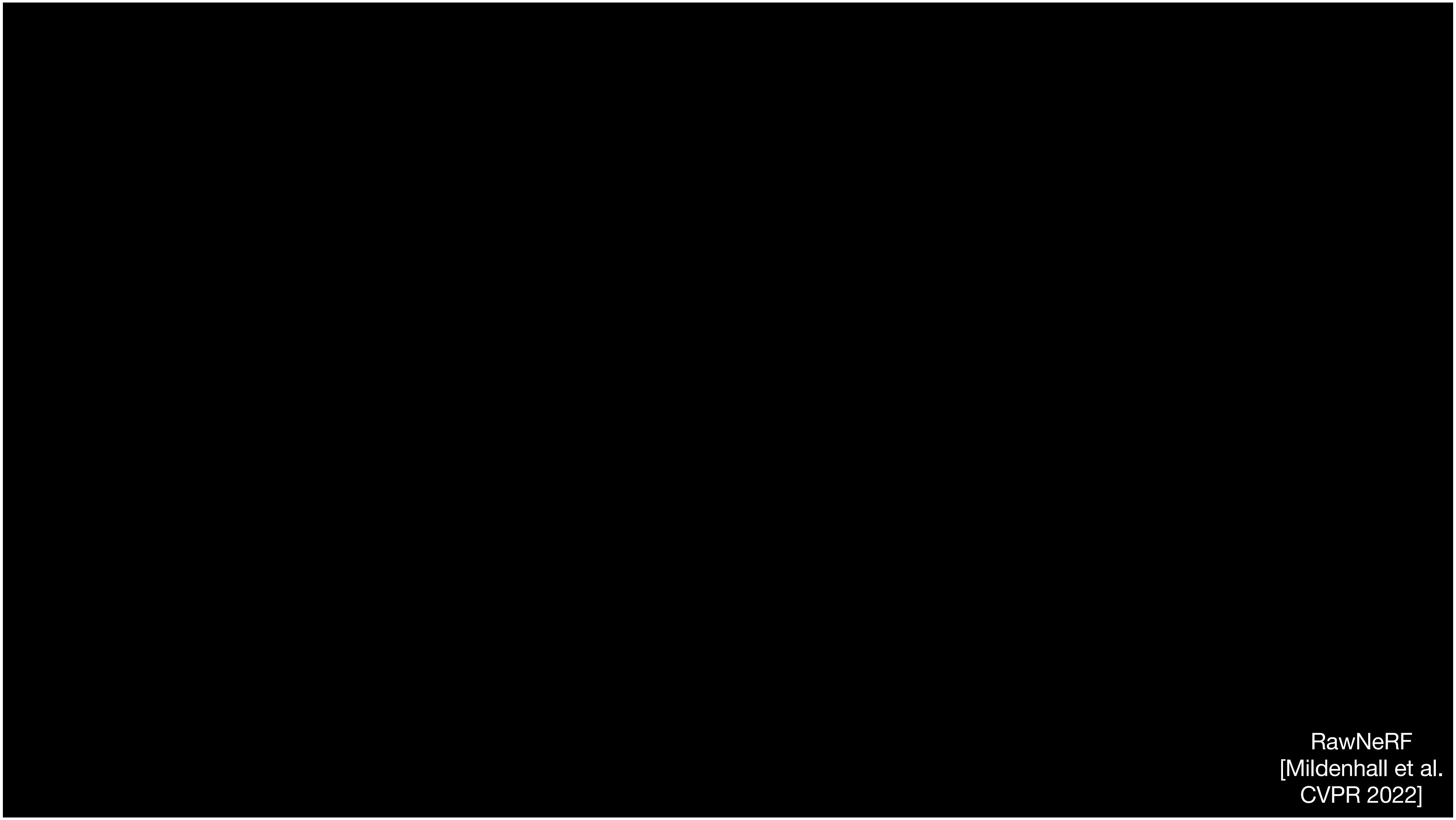
Generative 3D Faces

EG3D: Efficient Geometry-aware 3D Generative Adversarial Networks, Chan et al. CVPR 2022


Wang et al. SIGGRAPH 2022
© Disney

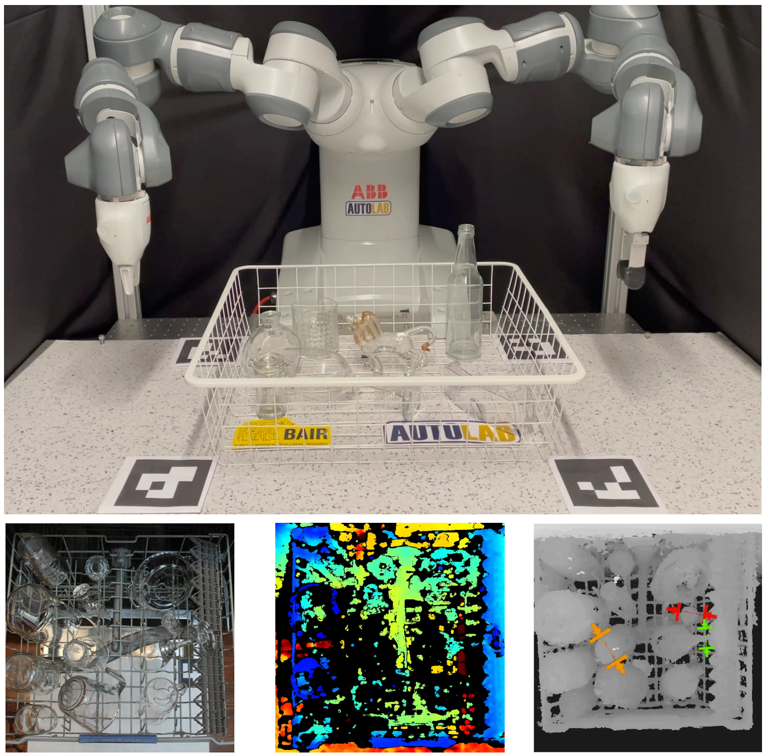
City-Scale NeRFs

BlockNeRF
[Tancik et al.
CVPR 2022]

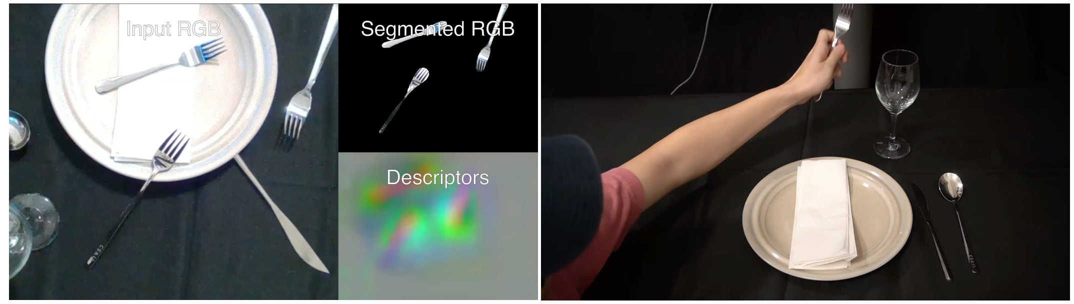

RawNeRF
[Mildenhall et al.
CVPR 2022]

RawNeRF
[Mildenhall et al.
CVPR 2022]

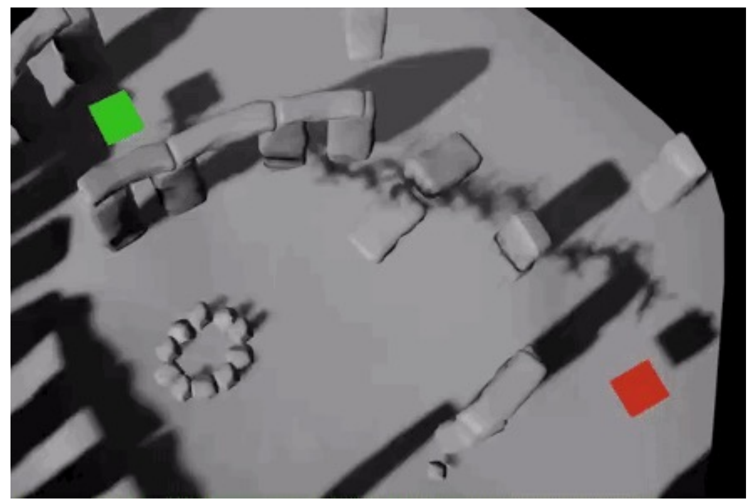
Robotics



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



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



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

Generating 3D scenes with diffusion models

DreamFusion [Poole et al. arXiv 2022]

Goals of these lectures

- In 2 years, 1840 citations (as of November 28th) will not cover all these papers
- Visit the fundamentals in Neural Volumetric Rendering by abstracting away recent developments
- Provide first principles + background for you to go and read these papers & play around with the tools

Menu

1. Birds Eye View & Background
2. Volumetric Rendering Function
3. Encoding and Representing 3D Volumes
4. Signal Processing Considerations
5. Challenges & Pointers

Capture of UC Berkeley redwoods with nerfstudio

Birds Eye View & Background

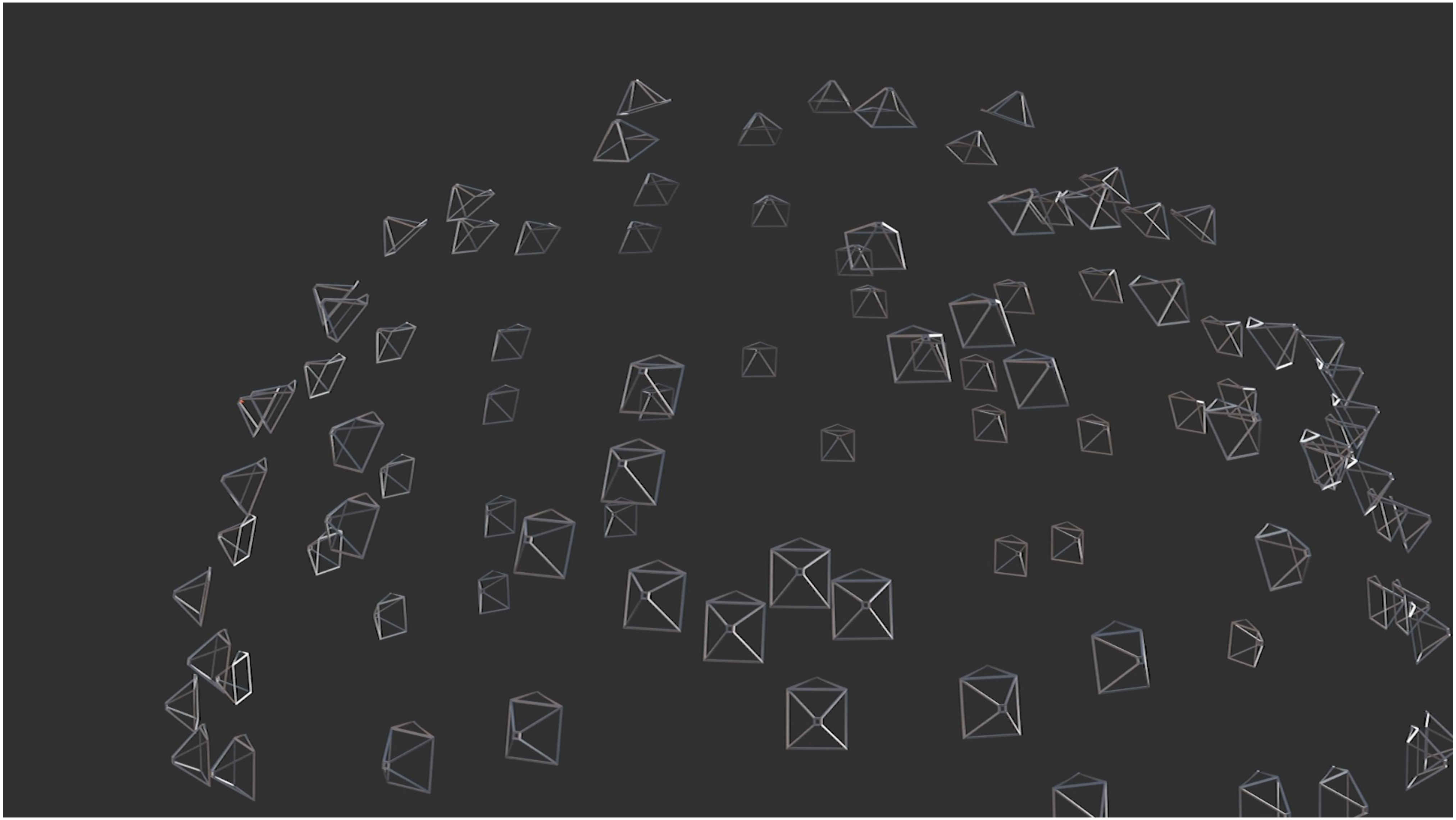
Birds Eye View

- What is NeRF?
- How is it different or similar to existing approaches?
- What is its historical context?

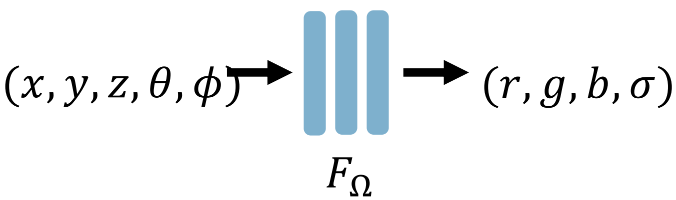
Problem Statement

Input:
A set of calibrated Images

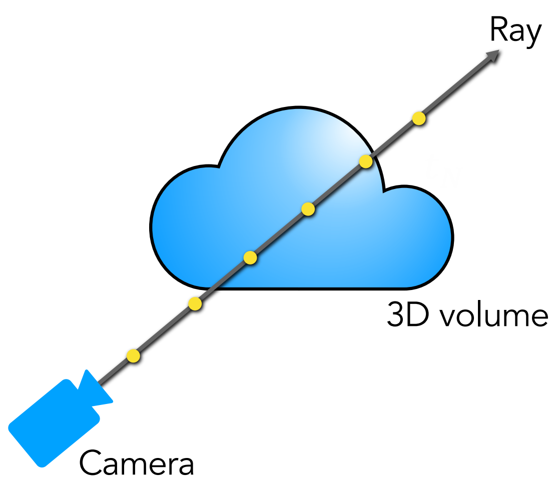
Output:
A 3D scene representation that renders novel views



Three Key Components

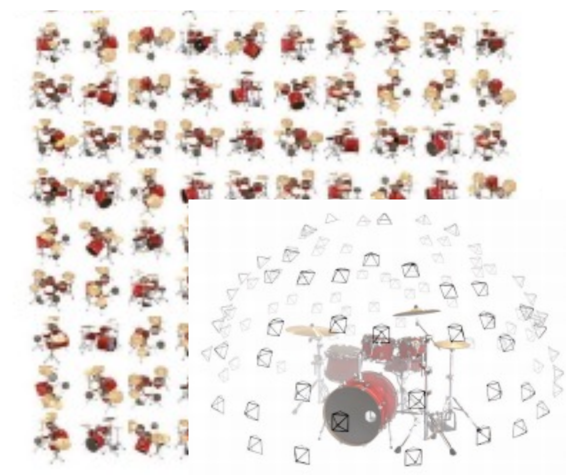


Neural Volumetric 3D Scene Representation



Differentiable Volumetric Rendering Function

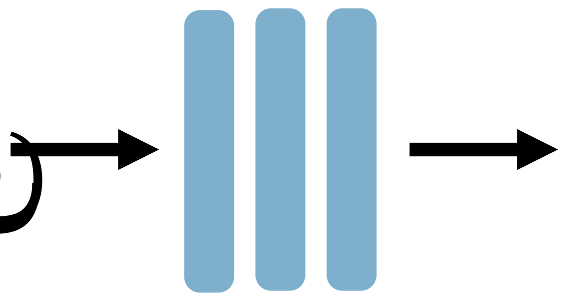
Objective: Synthesize all training views



Optimization via Analysis-by-Synthesis

Representing a 3D scene as a continuous 5D function

$$(\underbrace{x, y, z}_{\text{Spatial location}}, \underbrace{\theta, \phi}_{\text{Viewing direction}})$$



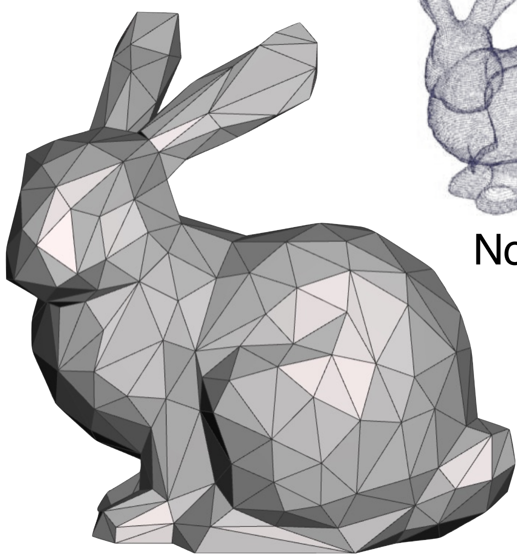
F_{Ω}

MLP
9 layers,
256 channels

$$(\underbrace{r, g, b}_{\text{Output color}}, \underbrace{\sigma}_{\text{Output density}})$$

What kind of a 3D representation is this?

It is not a Mesh

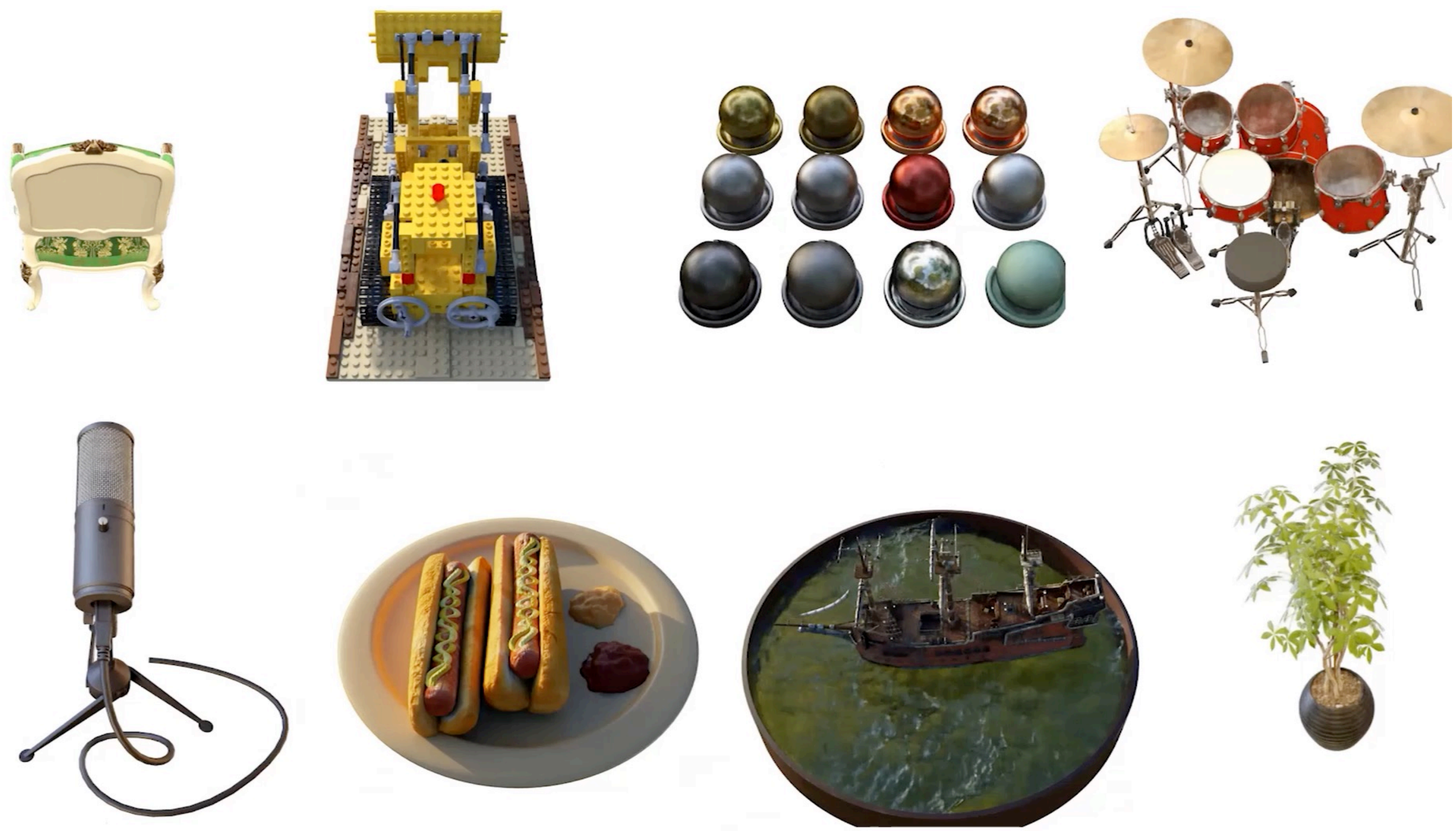


Not a point cloud
either

It is volumetric

It’s *continuous* voxels made of shiny transparent cubes

What is the problem that is being solved?



Plenoptic Function

Figure by Leonard McMillan

Q: What is the set of all things that we can ever see?

A: The Plenoptic Function (Adelson & Bergen ‘91)

Let’s start with a stationary person and try to parameterize <u>everything</u> that they can see…

Slide credit: Alyosha Efros

Grayscale Snapshot

$$P(\theta,\phi)$$

- is intensity of light
  - Seen from a single position (viewpoint)
  - At a single time
  - Averaged over the wavelengths of the visible spectrum

Slides from Alyosha Efros

Color snapshot

$$P(\theta,\phi,\lambda)$$

- is intensity of light
  - Seen from a single position (viewpoint)
  - At a single time
  - As a function of wavelength

Slides from Alyosha Efros

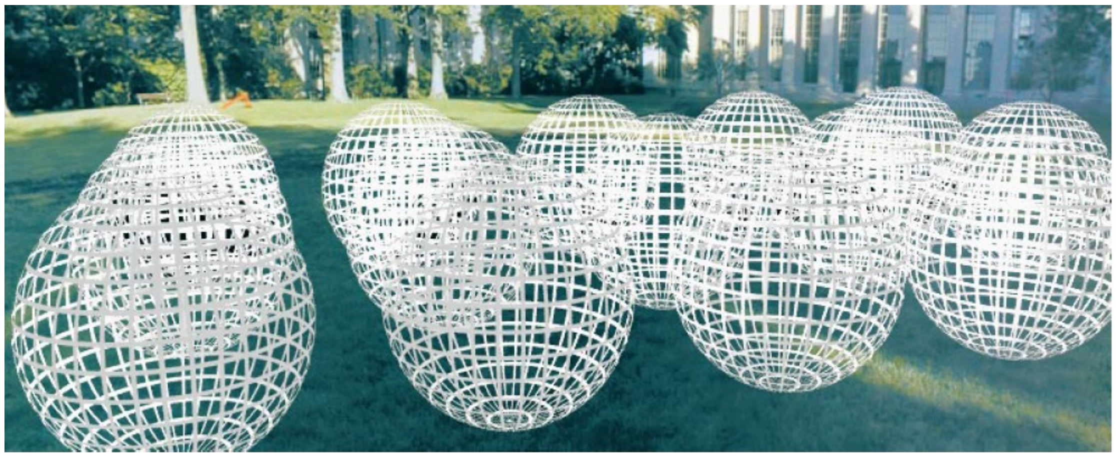
A movie

$$P(\theta,\phi,\lambda,t)$$

- is intensity of light
  - Seen from a single position (viewpoint)
  - Over time
  - As a function of wavelength

Slides from Alyosha Efros

A holographic movie

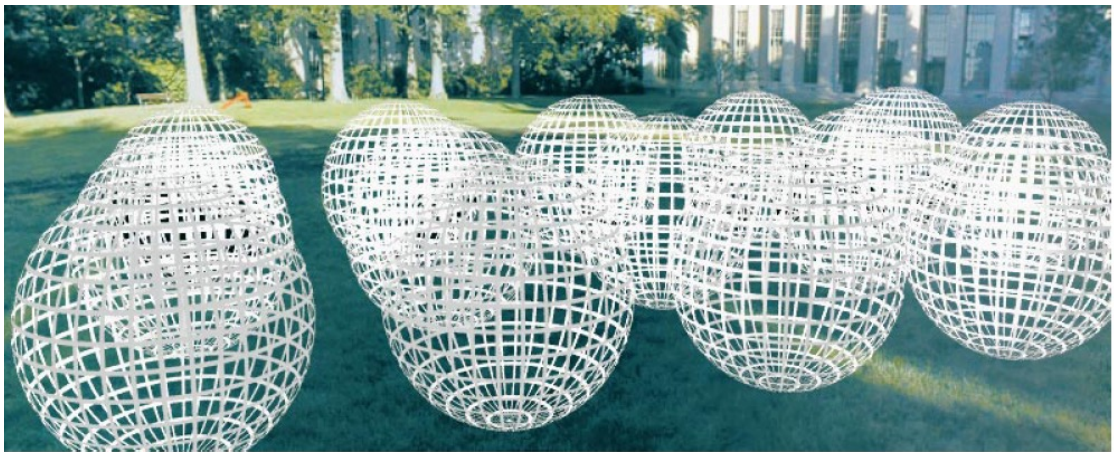


$$P(\theta,\phi,\lambda,t,V_X,V_Y,V_Z)$$

- is intensity of light
  - Seen from ANY position and direction
  - Over time
  - As a function of wavelength

Slides from Alyosha Efros

The plenoptic function

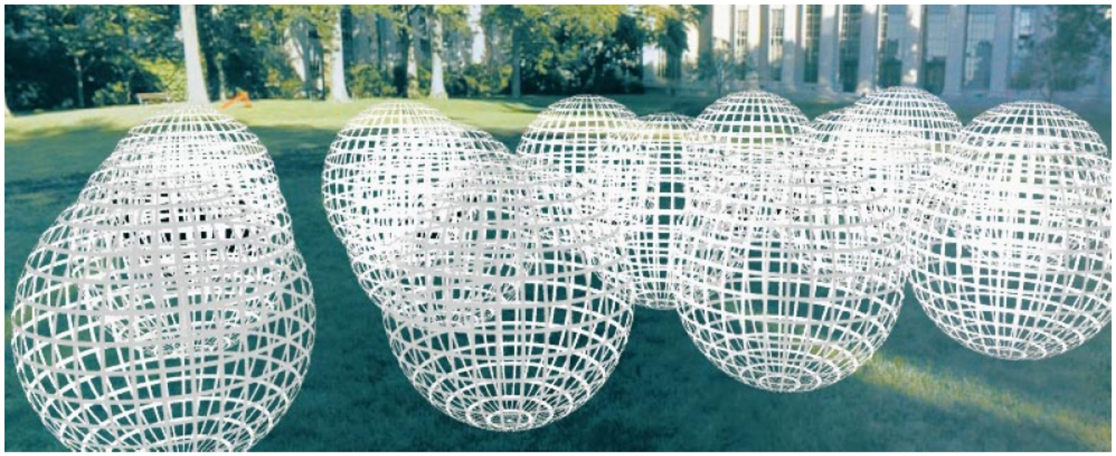


$$P(\theta,\phi,\lambda,t,V_X,V_Y,V_Z)$$

7D function, that can reconstruct every position & direction,
at every moment, at every wavelength
= it recreates the entirety of our visual reality!

Slides from Alyosha Efros

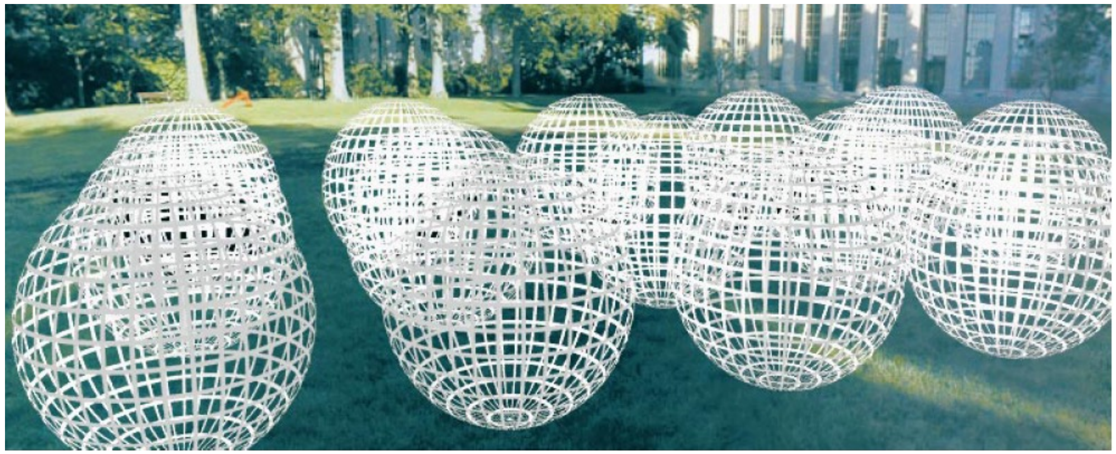
Goal: Plenoptic Function from a set of images



- Objective: Recreate the visual reality
- All about recovering photorealistic pixels, not about recording 3D point or surfaces
  —Image Based Rendering

aka **Novel View Synthesis**

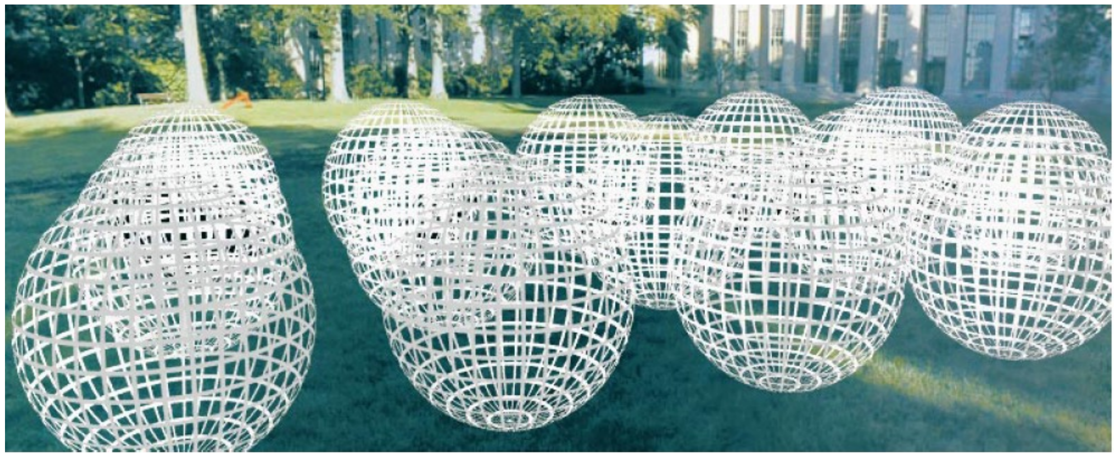
Goal: Plenoptic Function from a set of images



It is a conceptual device

Adelson & Bergen do not discuss how to solve this

Plenoptic Function



Look familiar 🙂?

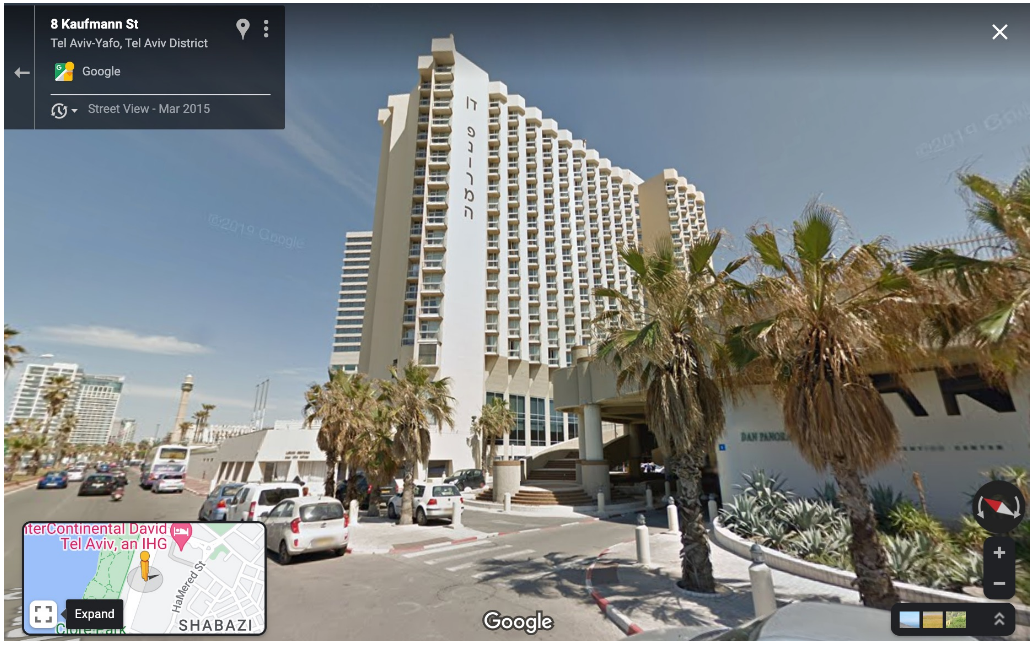
7D function:
2 – direction
1 – wavelength
1 – time
3 – location

$$P(\theta,\phi,\cancel{\lambda},\cancel{t},V_X,V_Y,V_Z) \longrightarrow P(\theta,\phi,V_X,V_Y,V_Z)$$

Let’s simplify:

1. Remove the time
2. Remove the wavelength & let the function output RGB colors

An example of a sparse plenoptic function



If street view was super dense
(360 view from any view point)
then it is the Plenoptic Function

Levoy and Hanrahan, SIGGRAPH 1996
Gortler et al. SIGGRAPH 1996

Lightfield / Lumigraph

- An approach for modeling the Plenoptic Function
- Take a lot of pictures from many views
- Interpolate the rays to render a novel view

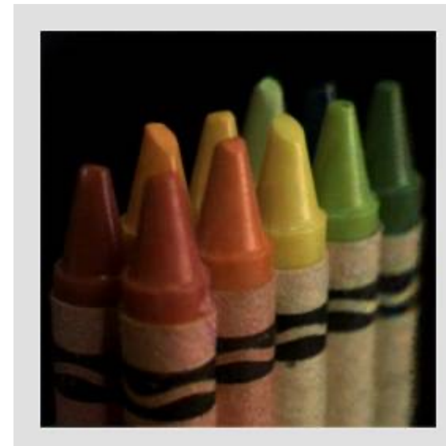
Stanford Gantry
128 cameras

Lytro camera

Levoy and Hanrahan, SIGGRAPH 1996
Gortler et al. SIGGRAPH 1996

Lightfield / Lumigraph

- An approach for modeling the Plenoptic Function
- Take a lot of pictures from many views
- Interpolate the rays to render a novel view



Stanford Gantry
128 cameras

Lytro camera

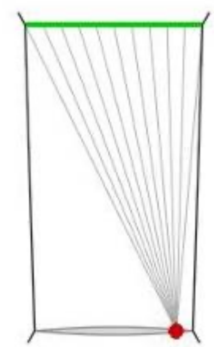


Figure from Marc Levoy

Levoy and Hanrahan, SIGGRAPH 1996
Gortler et al. SIGGRAPH 1996

Lightfield / Lumigraph

- An approach for modeling the Plenoptic Function
- Take a lot of pictures from many views
- Interpolate the rays to render a novel view

Stanford Gantry
128 cameras

Lytro camera

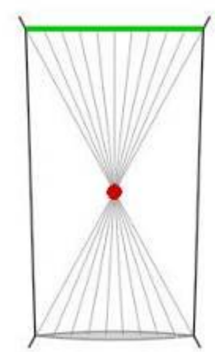
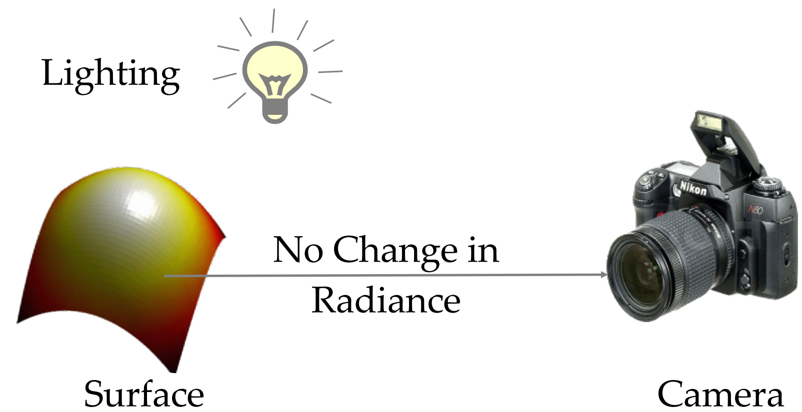


Figure from Marc Levoy

Levoy and Hanrahan, SIGGRAPH 1996
Gortler et al. SIGGRAPH 1996

Lightfield / Lumigraph

Lightfields assume that the ray shooting out from a pixel is never occluded.



Because of this it only models the plenoptic surface:

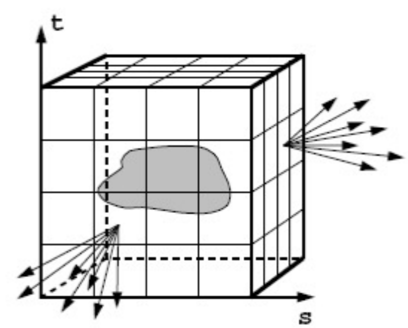


Figure 1: The surface of a cube holds all the radiance information due to the enclosed object.

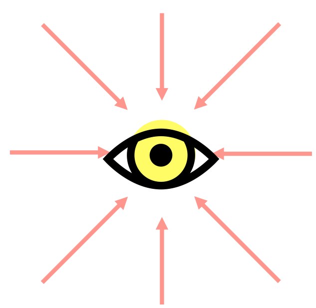
How NeRF models the Plenoptic Function

Look familiar 🙂?

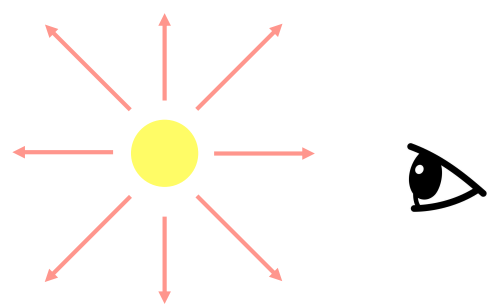
$$P(\theta,\phi,V_X,V_Y,V_Z)$$

NeRF takes the same input as the Plenoptic Function!

A subtle difference:



Plenoptic Function

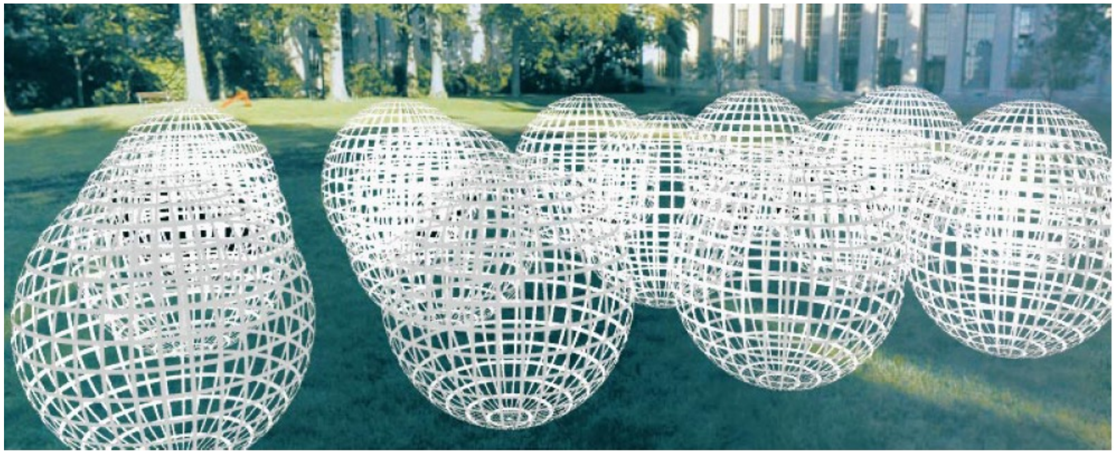


NeRF

So NeRF requires the integration along the viewing ray to compute the Plenoptic Function

Bottom line: it models the full plenoptic function!

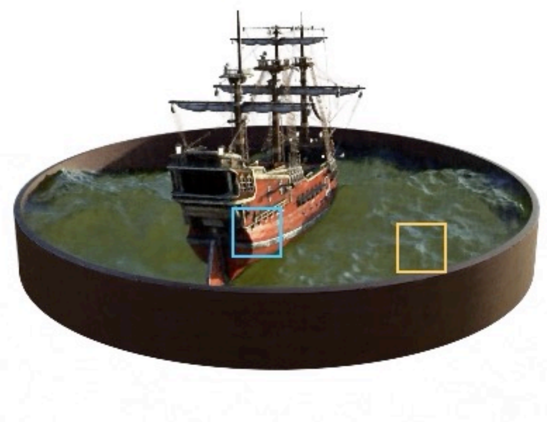
5D function

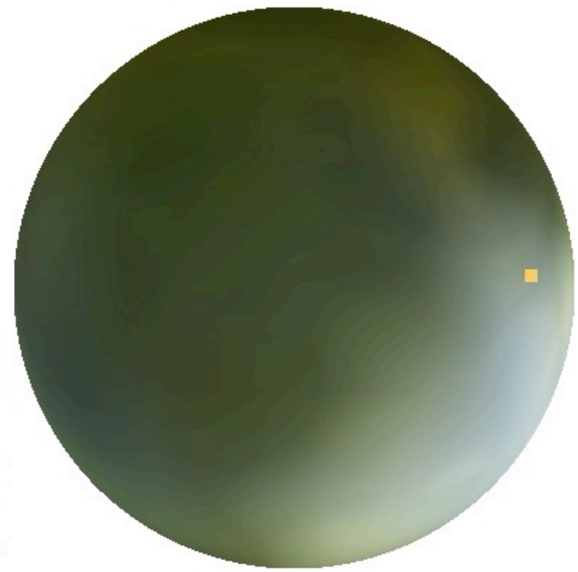


- For every location (3D), all possible views (2D) ✨🧊✨
- NeRF models this space with a continuous view-dependent volume with opacity
- The color emitted by every point is composited to render a pixel
- Unlike a light field, the entire 5D plenoptic function can be modeled (you can fly through the world)

Visualizing the 2D function on the sphere

Outgoing radiance distribution
for point on side of ship





Outgoing radiance distribution
for point on water's surface

Baking in Light

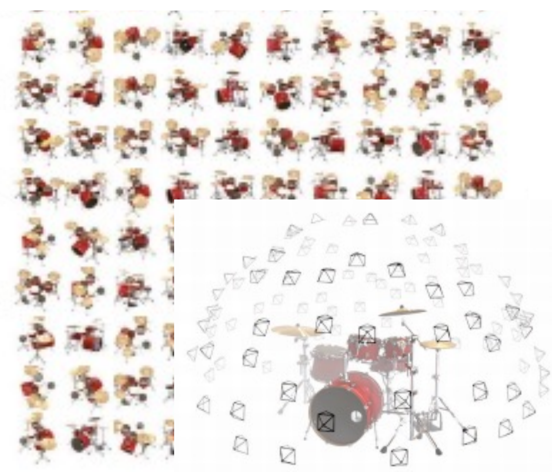
- NeRF can capture non-Lambertian (specular, shiny surfaces) because it models the color in a view-dependent manner
- This is hard to do with meshes unless you model the physical materials & lighting interactions
- But, with Image Based Rendering — All lighting effects are baked in

NeRF in a Slide

Volumetric 3D Scene Representation

Differentiable Volumetric Rendering Function

Objective: Reconstruct all training views



Optimization via Analysis-by-Synthesis

Unmentioned caveat so far

- Training a NeRF requires a **calibrated** camera!!!!

- Need to know the camera parameters: extrinsic (viewpoint) & intrinsics (focal length, distortion, etc)

How do we get this from images?

Structure from Motion

Or Photogrammetry (1850~)
Long history in Computer Vision

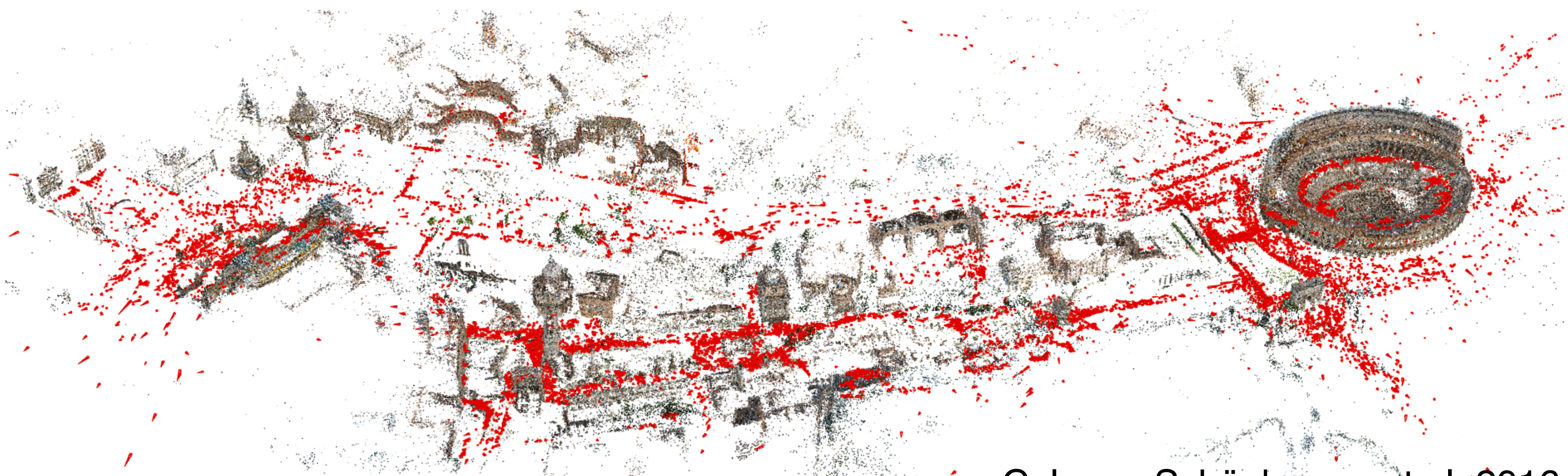
Proc. R. Soc. Lond. B. **203**, 405–426 (1979)
Printed in Great Britain

The interpretation of structure from motion

By S. Ullman
Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 545 Technology Square (Room 808), Cambridge, Massachusetts 02139 U.S.A.

NeRF is AFTER Structure from Motion

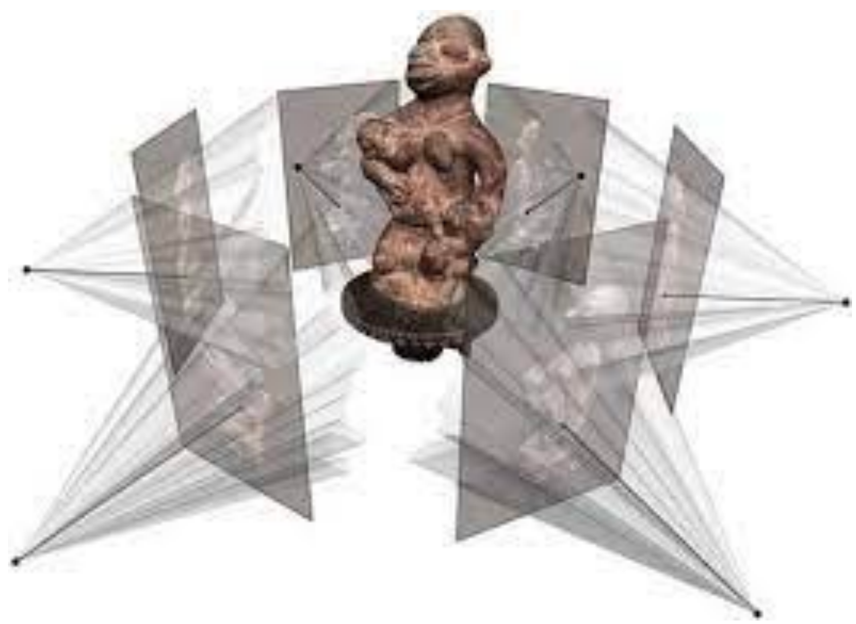
- In order to train NeRF you need to run SfM/SLAM on the images to estimate the camera parameters
- In this sense, the problem category is same as that of **Multi-view Stereo**



Colmap: Schönberger et al. 2016

Multi-view Stereo

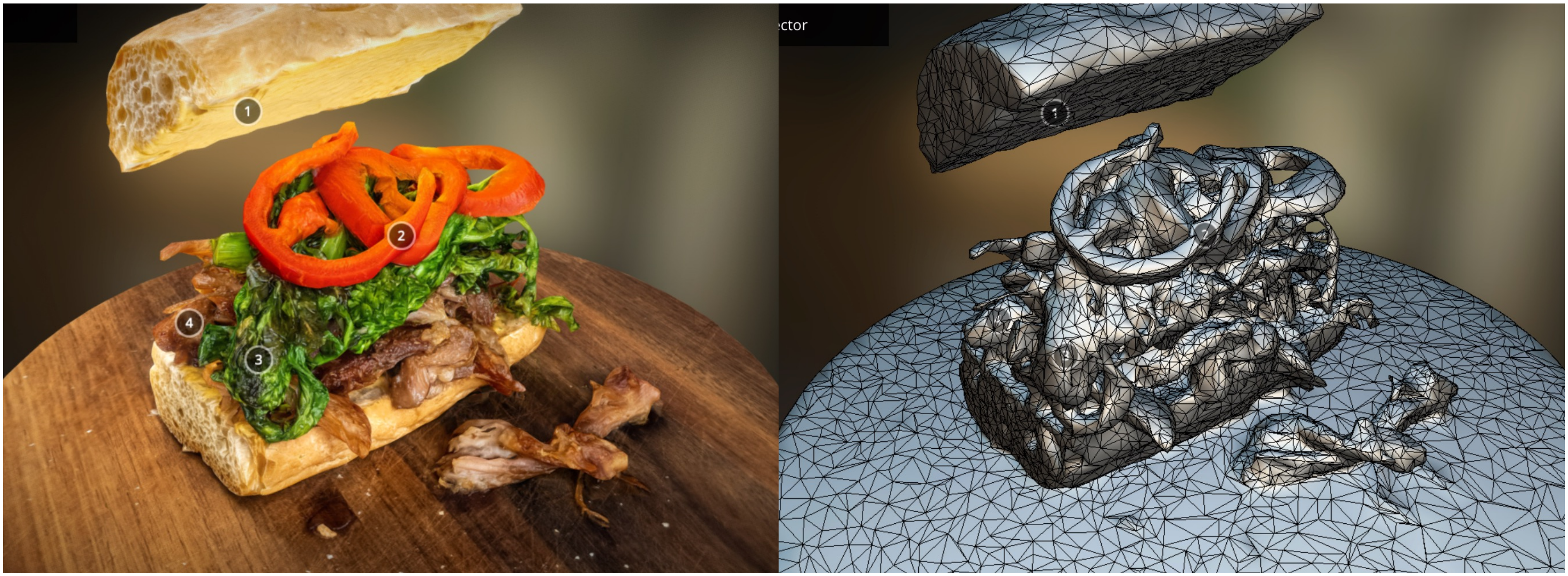
- Problem: Given calibrated cameras, recover highly detailed 3D **surface** model
- Dense photogrammetry, often the output is textured meshes



Figures by Carlos Hernandez, Yasutaka Furukawa

Multi-View Stereo

Solutions to MVS is what you see for any existing 3D scanning system, ie sketchfab, or what’s in your video game



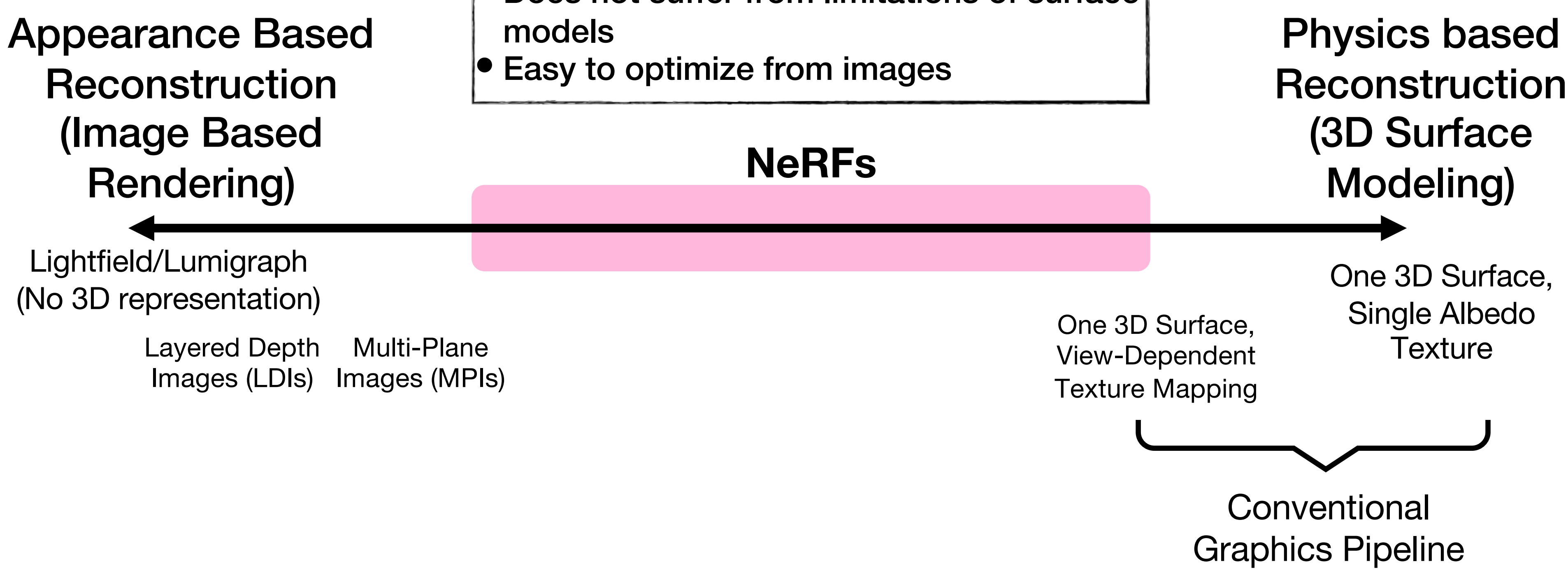
Multi-View Stereo

Because they often model surfaces, struggles on Thin / Amorphus / Shiny objects


nerfstudio

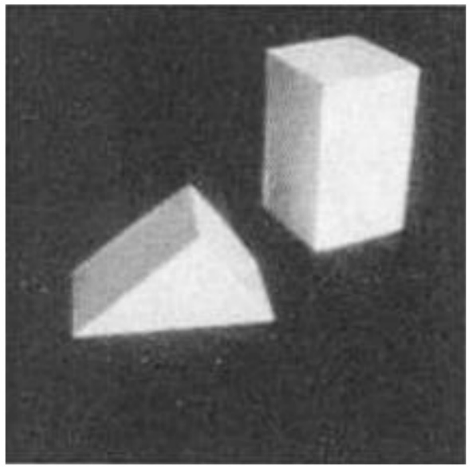
Where NeRF stands

- can do Image Based Rendering well, while also being a 3D representation
- Does not suffer from limitations of surface models
- Easy to optimize from images

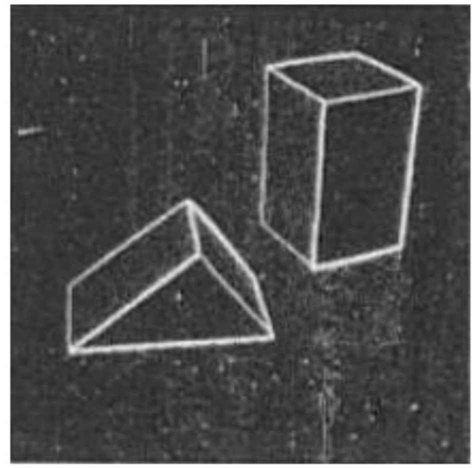


Analysis-by-Synthesis

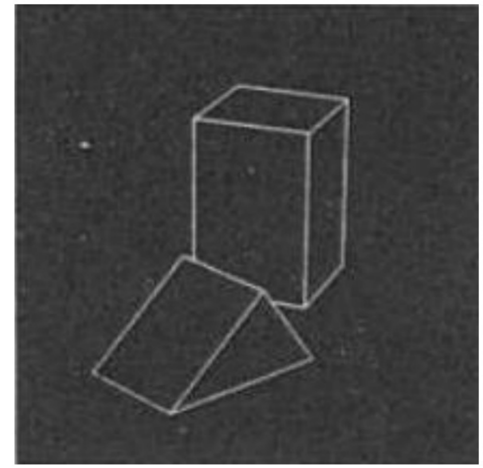
Larry Roberts
“Father of Computer Vision”



Input image



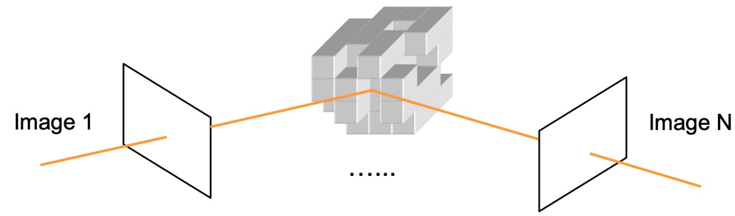
2x2 gradient operator



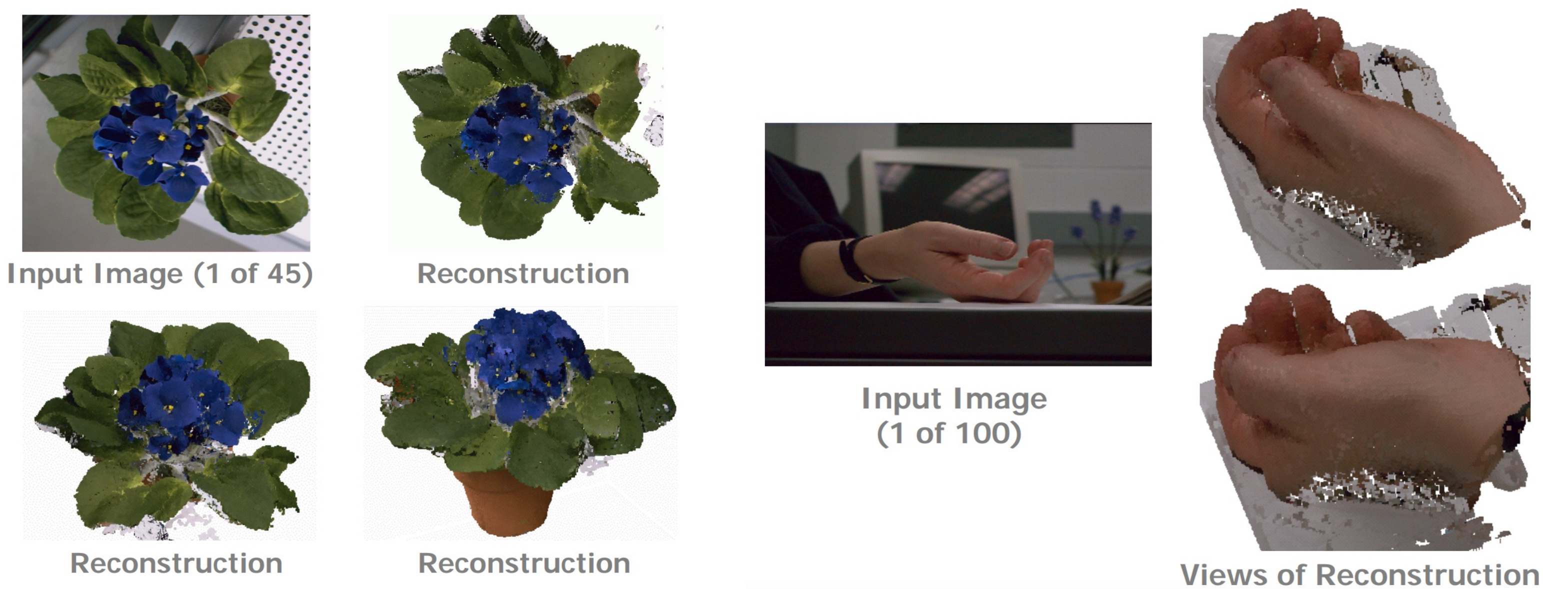
computed 3D model
rendered from new viewpoint

- History goes way back to the **first** Computer Vision paper!
  Roberts: Machine Perception of Three-Dimensional Solids, MIT, 1963

Power of Analysis-by-Synthesis

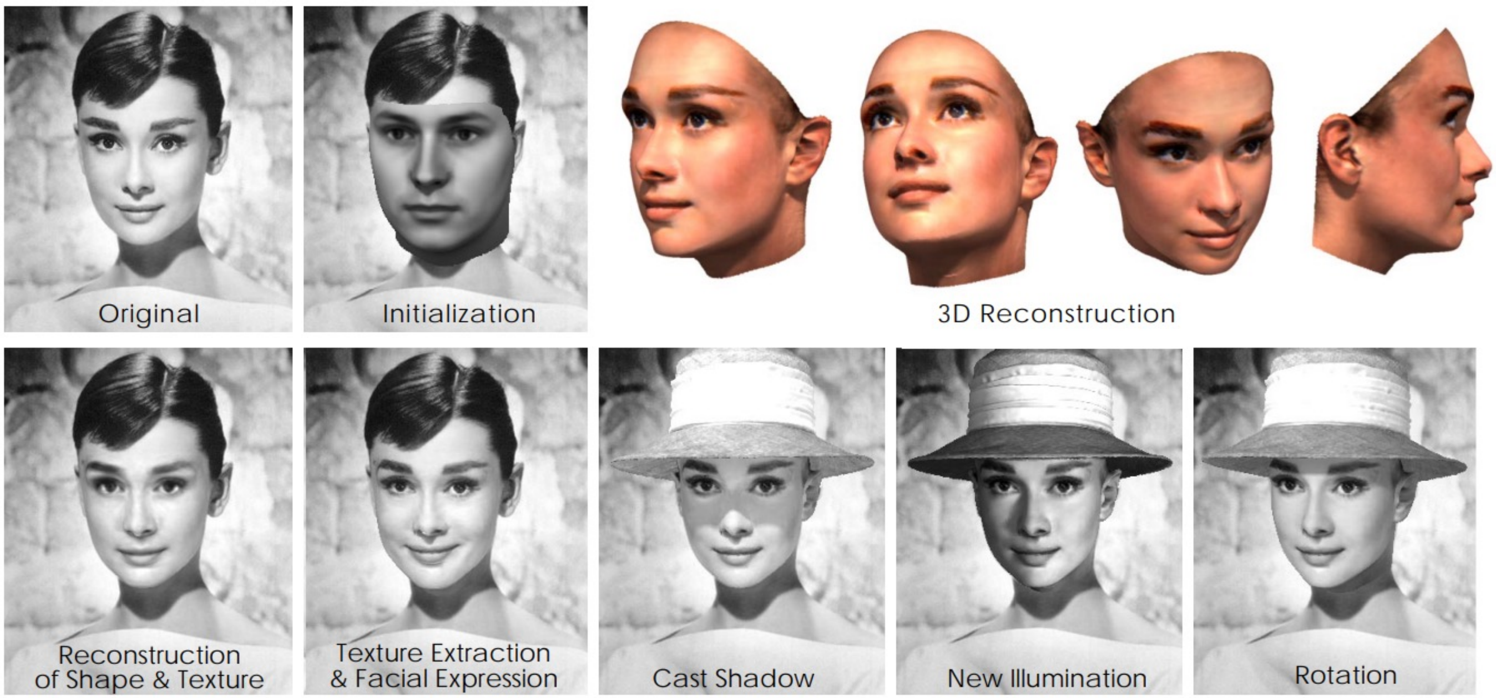


- Space Carving: A MVS method that used Colored voxels
- But the optimization method was bottom up then.
- Key is optimization via Analysis-by-Synthesis [Plenoxels, Yu et al. 2022]



Kultulakos and Seitz, A Theory of Shape by Space Carving IJCV 2000

Analysis-by-Synthesis



Blanz & Vetter 1999

- With custom differentiable renders

Analysis by Synthesis Requires Differentiable Renderers

Next: Deep dive into Volumetric Rendering Function

Where we are

1. Birds Eye View & Background
2. **Volumetric Rendering Function**
3. Encoding and Representing 3D Volumes
4. Signal Processing Considerations
5. Challenges & Pointers

Volume Rendering

"... in 10 years, all rendering will be volume rendering."
Jim Kajiya at SIGGRAPH '91

Neural Volumetric Rendering

Neural Volumetric **Rendering**

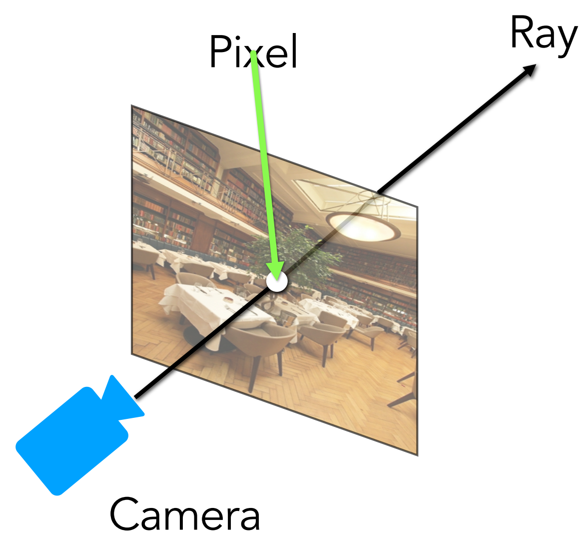
computing color along rays
through 3D space

What color is this pixel?

Cameras and rays

Cameras and rays

- We need the mathematical mapping from (*camera*, *pixel*) → *ray*
- Then can abstract underlying problem as learning the function *ray* → *color* (the "plenoptic function")



Recap Coordinate frames: World-to-Camera Transforms

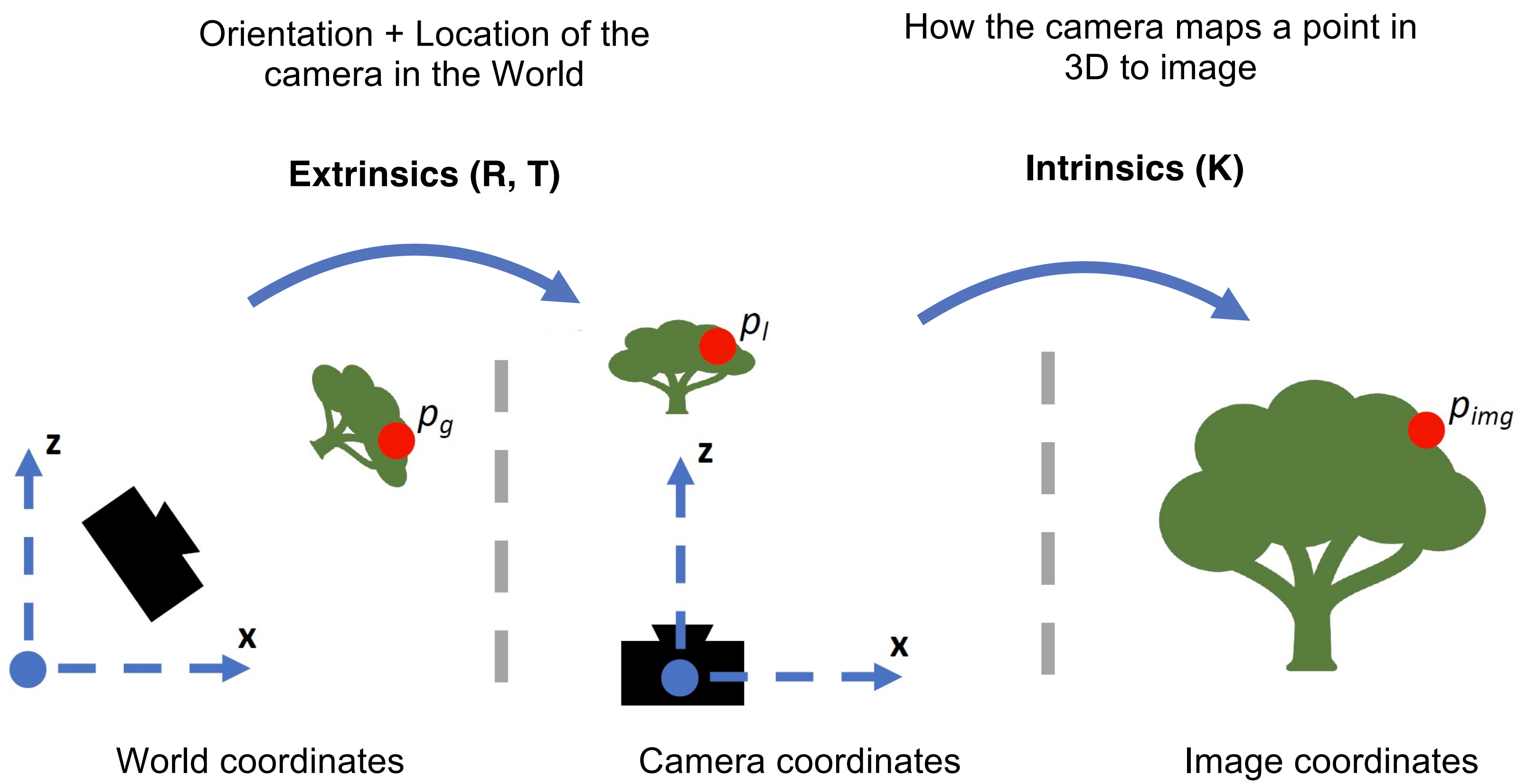


Figure credit: Peter Hedman

Recap Coordinate frames: Camera-to-World Transforms

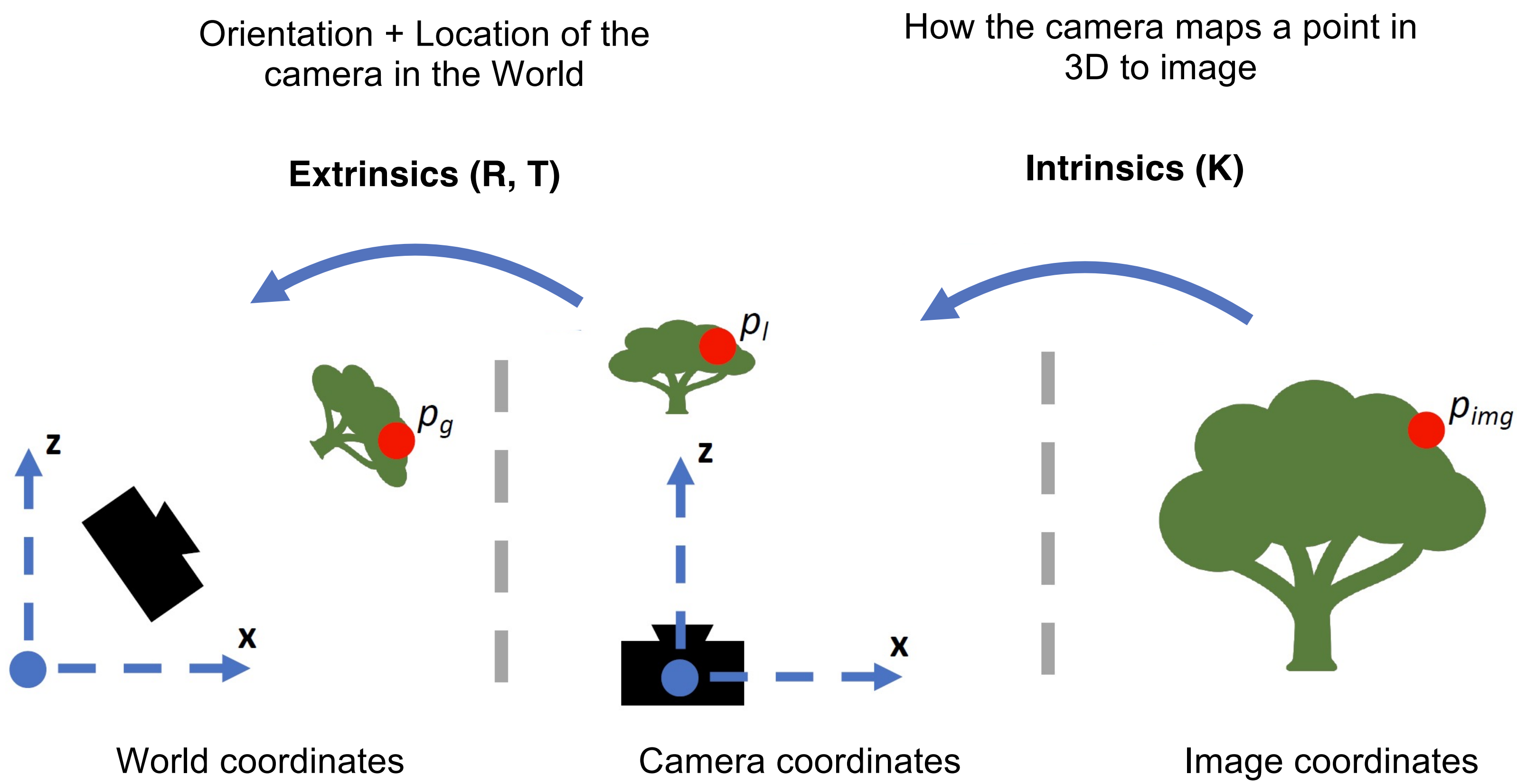
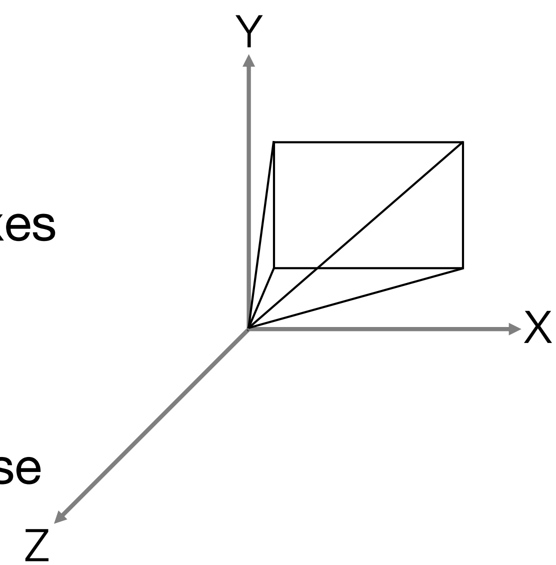


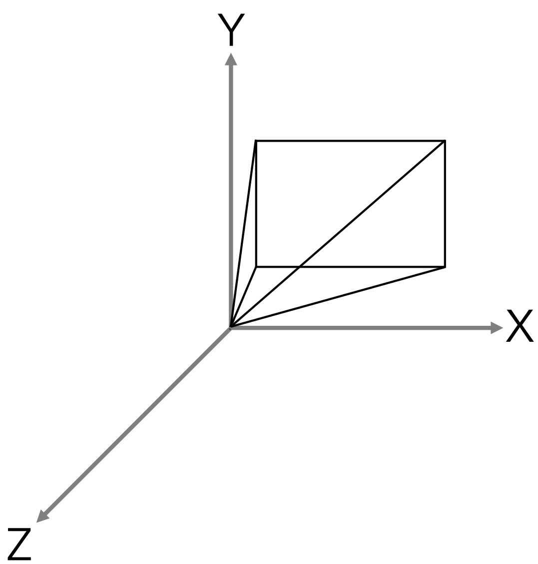
Figure credit: Peter Hedman

Camera pose - pixel to camera

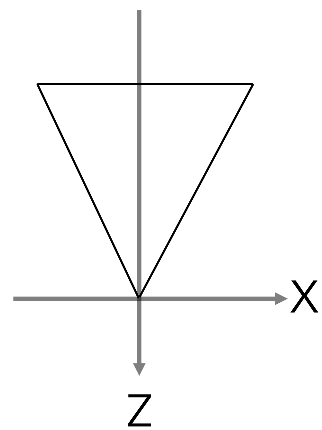
- Mapping from (*camera*, *pixel*) to ray in camera coordinate frame
- This coordinate system has camera situated at origin, with right/up/backwards aligned to x/y/z axes
  - Axis convention varies in different codebases :(
- “Inverse intrinsic matrix” in a computer vision sense



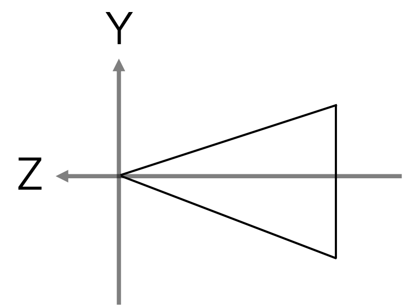
Camera pose - pixel to camera



3D view

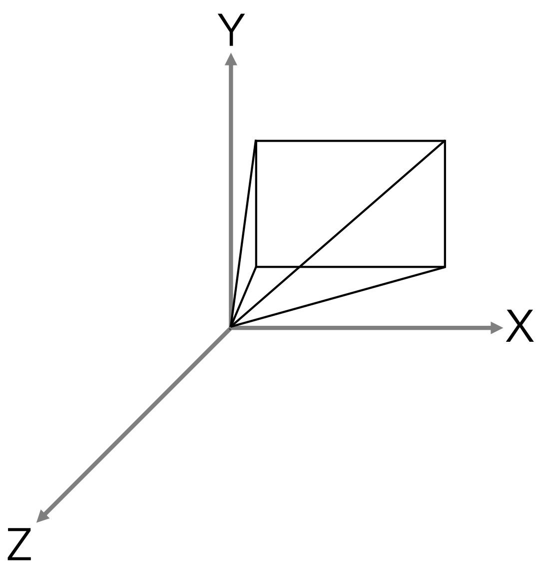


Top view
(looking along Y)

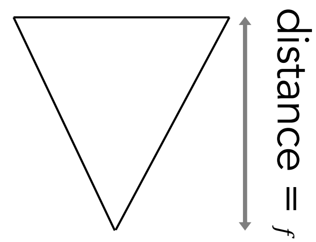


Side view
(looking along X)

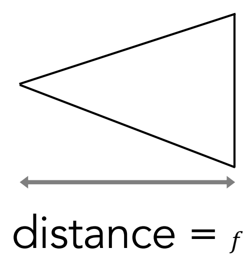
Camera pose - pixel to camera



3D view



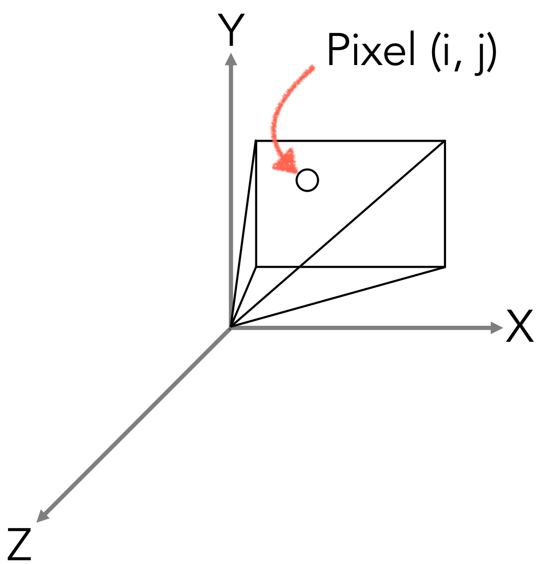
Top view
(looking along Y)



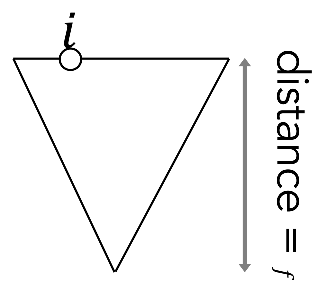
Side view
(looking along X)

Camera pose - pixel to camera

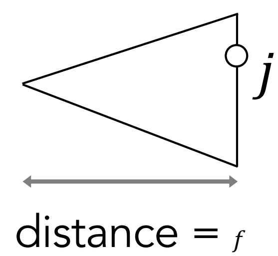
Coordinates are (i, j) in pixel space



3D view



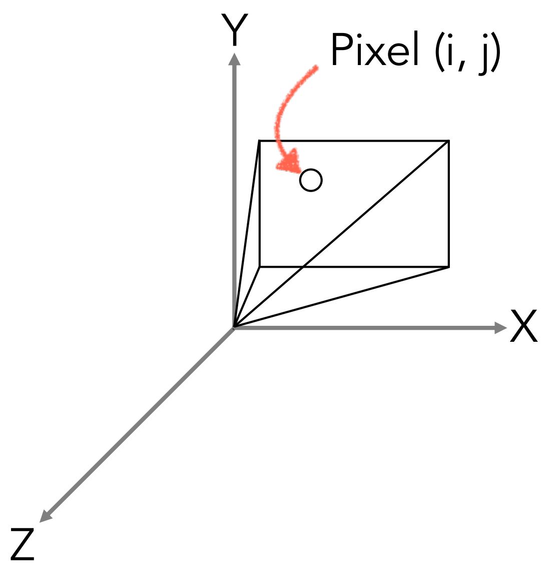
Top view
(looking along Y)



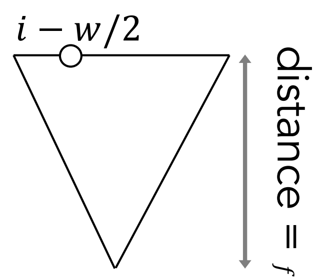
Side view
(looking along X)

Camera pose - pixel to camera

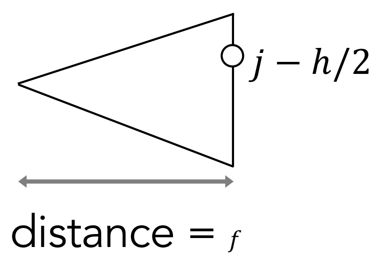
Recenter using pixel coordinates of image center



3D view



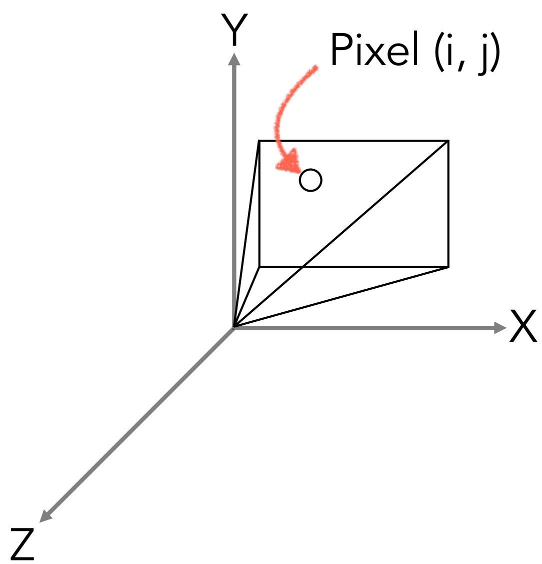
Top view
(looking along Y)



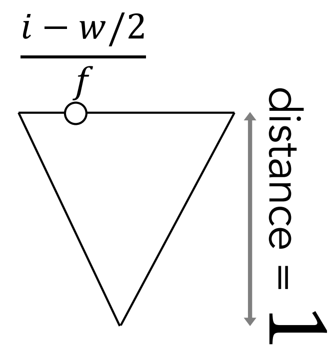
Side view
(looking along X)

Camera pose - pixel to camera

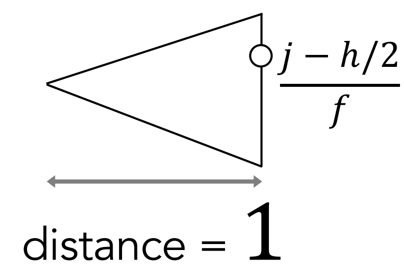
Rescale frustum by focal length f so that image plane is at distance 1



3D view



Top view
(looking along Y)

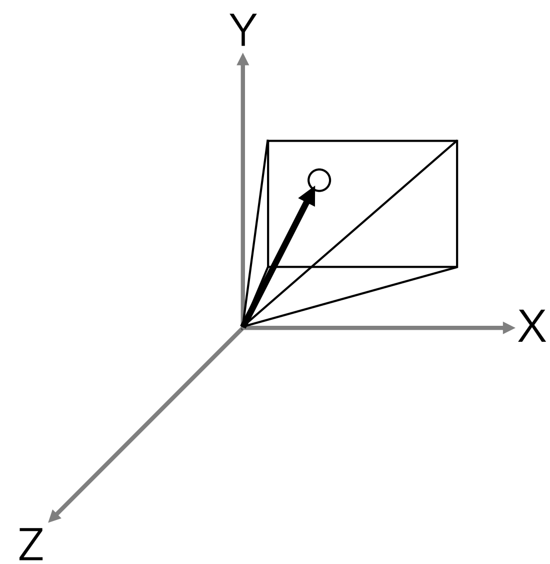


Side view
(looking along X)

Camera pose - pixel to camera

Full mapping is $(i, j) \rightarrow \left(\frac{i-w/2}{f}, \frac{j-h/2}{f}, -1\right)$ to get 3D coordinates for a point on the image plane.

Camera space ray points from origin toward this point.

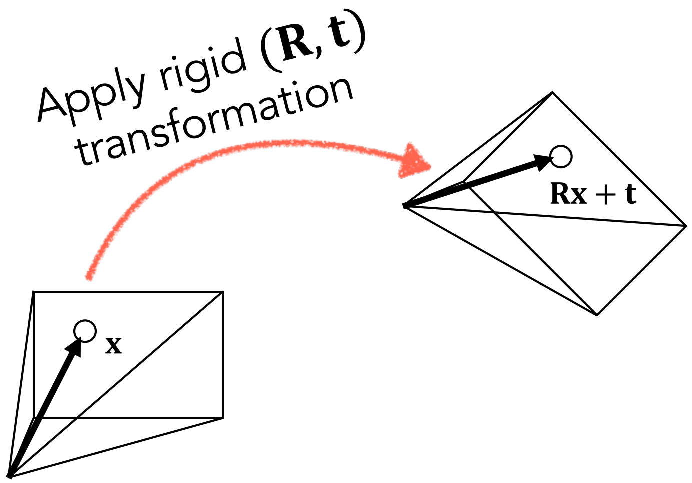


Camera pose - pixel to camera

- Omitted details
  - Half-pixel offset — add 0.5 to *i* and *j* so ray precisely hits pixel center
  - This is a *perfect* pinhole model — typically need to add a distortion model to correct for error found in real cameras

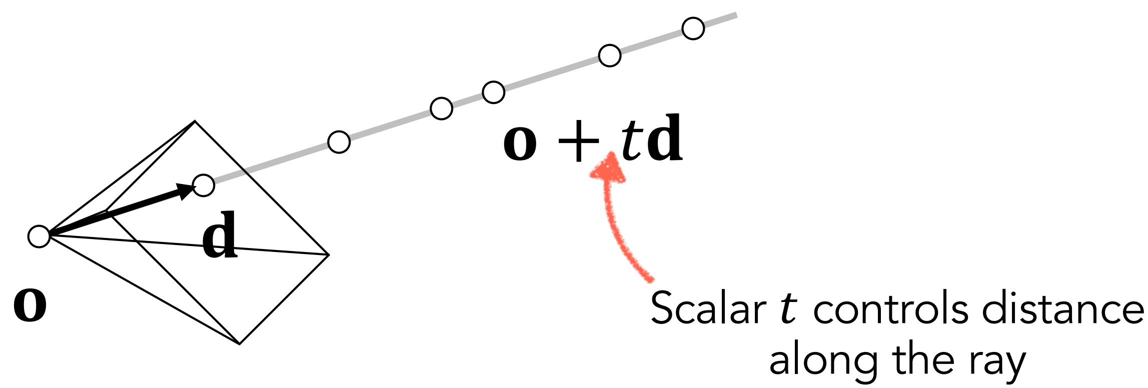
Camera pose - camera to world

- Simply apply rigid rotation and translation to origin and image plane points (six degrees of freedom).
- This positions the camera in “world space”.



Calculating points along a ray

In the world coordinate frame:

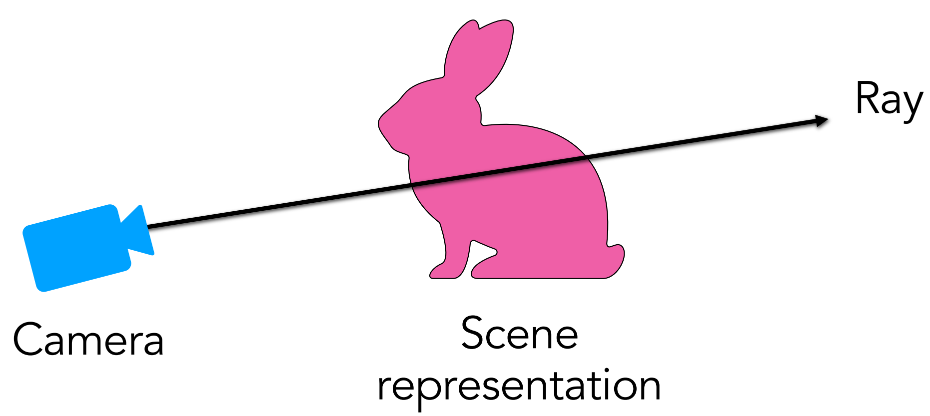


Neural Volumetric Rendering

Neural **Volumetric** Rendering

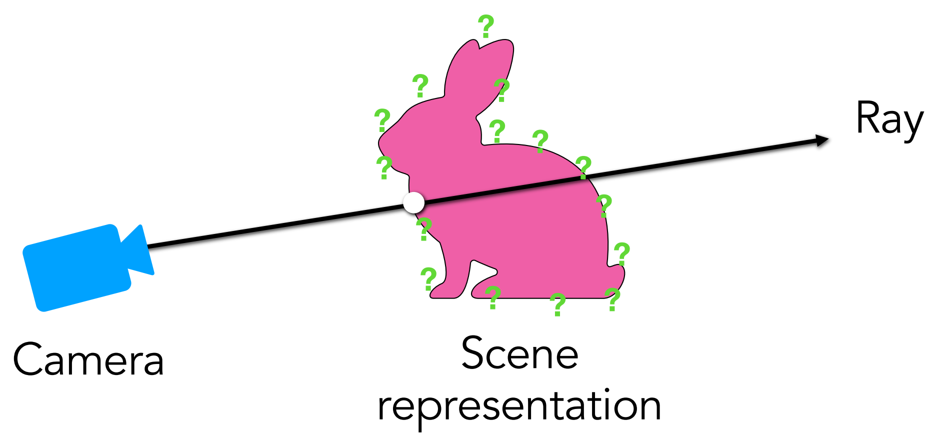
continuous, differentiable
rendering model without
concrete ray/surface intersections

Surface vs. volume rendering



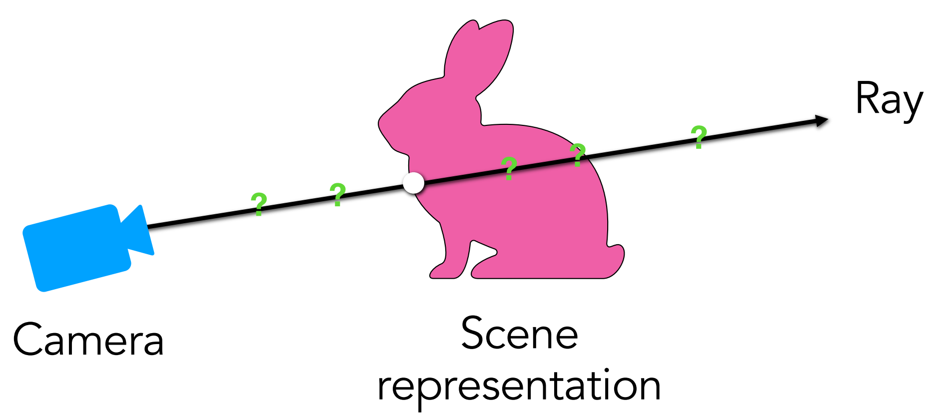
Want to know how ray interacts with scene

Surface vs. volume rendering



Surface rendering — loop over geometry, check for ray hits

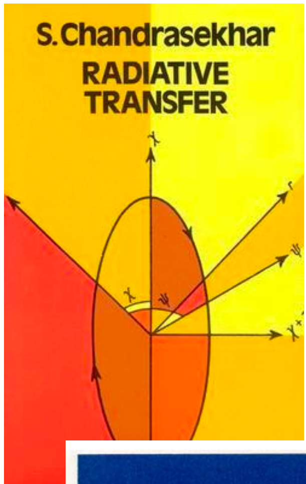
Surface vs. volume rendering



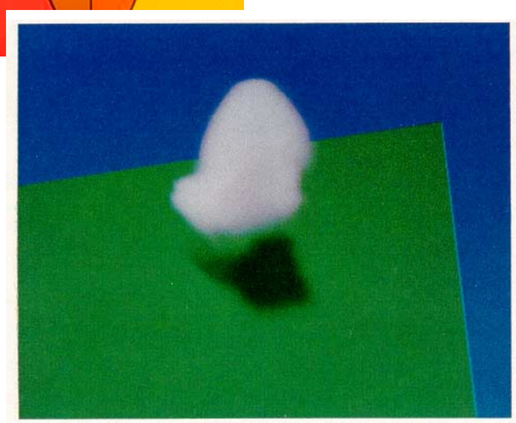
Volume rendering — loop over ray points, query geometry

History of volume rendering

Early computer graphics



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

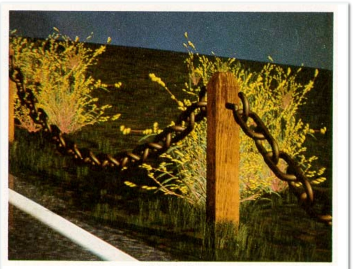


Ray tracing simulated cumulus cloud [Kajiya]

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

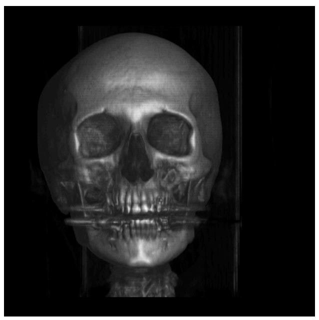
Alpha compositing

- Theory of volume rendering co-opted from physics in the 1980s: absorption, emission, out-scattering/in-scattering
- Alpha rendering developed for digital compositing in VFX movie production


Pt.Reyes = Foreground **over** Hillside **over** Background.
Alpha compositing [Porter and Duff]

Porter and Duff 1984, *Compositing Digital Images*

Volume rendering for visualization

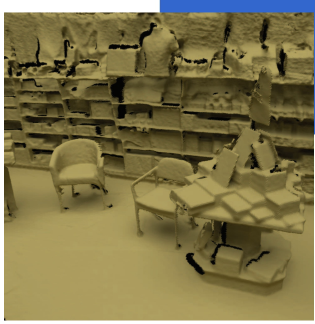


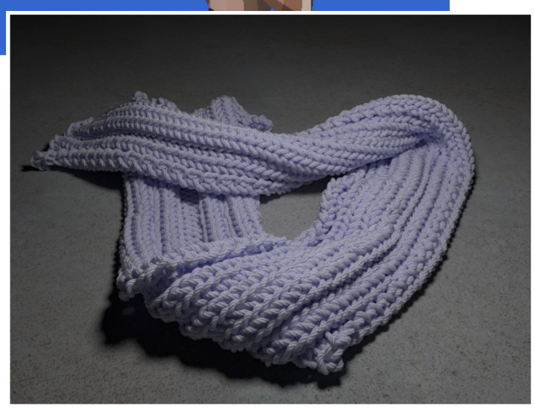
Medical data visualisation [Levoy]

- Theory of volume rendering co-opted from physics in the 1980s: absorption, emission, out-scattering/in-scattering
- Alpha rendering developed for digital compositing in VFX movie production
- Volume rendering applied to visualise 3D medical scan data in 1990s

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

Volume rendering for surfaces





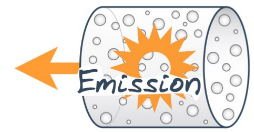
Geometry and materials can be stored per-voxel and used with standard surface rendering methods

- Sparse voxel octrees
- Voxel hashing
- Anisotropic radiative transfer

Volume rendering derivations

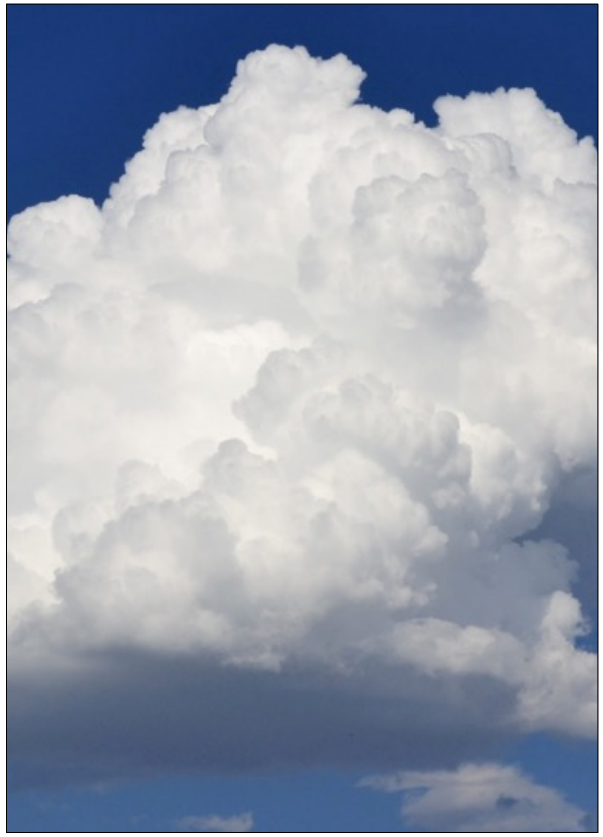
Absorption

Scattering



Emission

http://commons.wikimedia.org



http://wikipedia.org

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

Simplify

Absorption

http://commons.wikimedia.org

Scattering

Emission

http://wikipedia.org

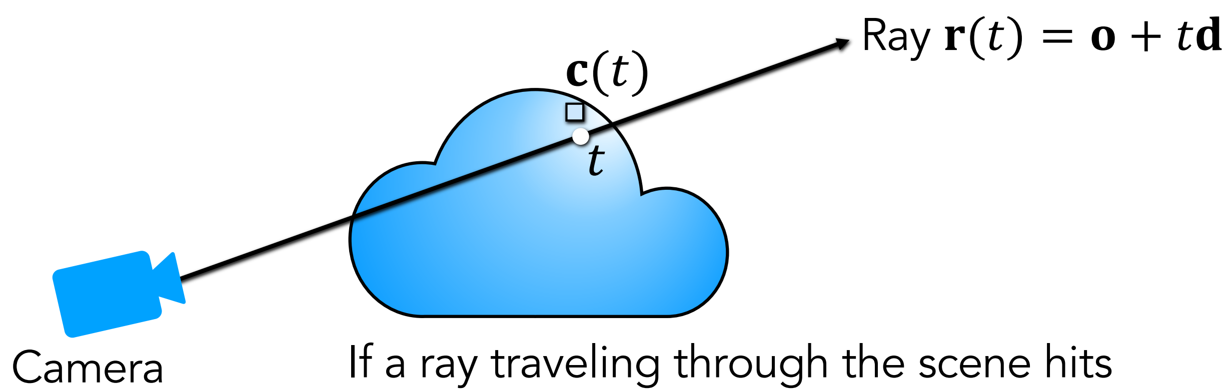
Slide credit: Novak et al 2018, *Monte Carlo methods for physically based volume rendering*

Volumetric formulation for NeRF

Scene is a cloud of tiny colored particles

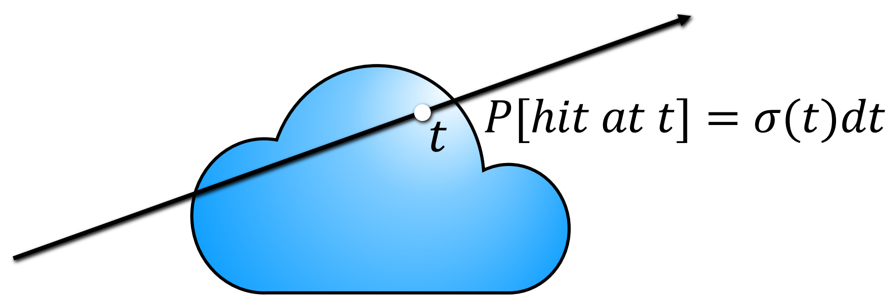
Max and Chen 2010, *Local and Global Illumination in the Volume Rendering Integral*

Volumetric formulation for NeRF



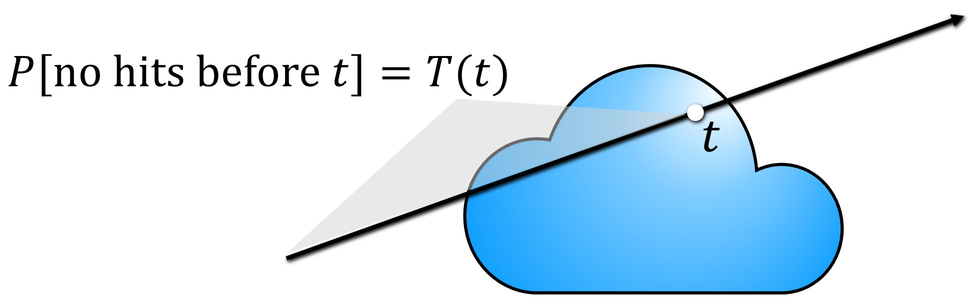
If a ray traveling through the scene hits a particle at distance t along the ray, we return its color $\mathbf{c}(t)$

What does it mean for a ray to “hit” the volume?



This notion is *probabilistic:* chance that ray hits a particle in a small interval around t is $\sigma(t)dt$.
σ is called the “volume density”

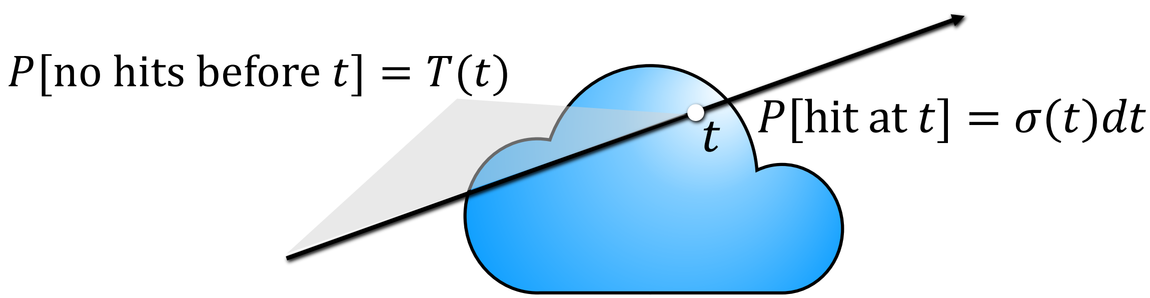
Probabilistic interpretation



To determine if t is the *first* hit along the ray, need to know $T(t)$: the probability that the ray makes it through the volume up to t.

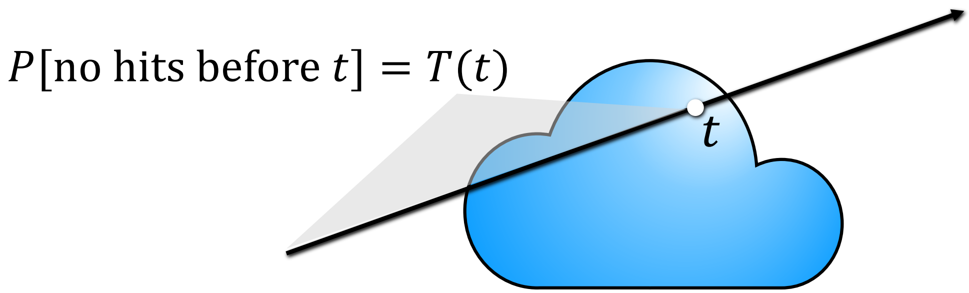
$T(t)$ is called "transmittance"

Probabilistic interpretation



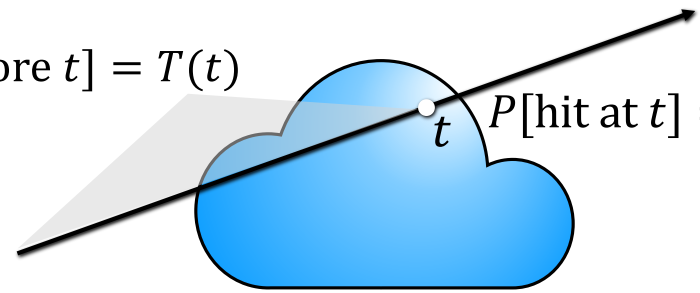
The product of these probabilities tells us how much you see the particles at t:
$P[\text{first hit at } t] = P[\text{no hit before } t] \times P[hit\ at\ t] = T(t)\sigma(t)dt$

Calculating T given σ



Let's write T as a function of σ ! How?

Calculating T given σ



σ and T are related by the probabilistic fact that

$$\underbrace{P[\text{no hit before } t + dt]}_{T(t+dt)} = \underbrace{P[\text{no hit before } t]}_{T(t)} \times \underbrace{P[\text{no hit at } t]}_{(1-\sigma(t)dt)}$$

Calculating transmittance T

$$T(t + dt) = T(t)(1 - \sigma(t)dt)$$

$$T(t + dt) \qquad = \qquad T(t) \qquad (1 - \sigma(t)dt)$$

Solve for T

$$T(t + dt) = T(t)(1 - \sigma(t)dt)$$

Solve for T

$$T(t + dt) = T(t)(1 - \sigma(t)dt)$$

Taylor expansion for T⇒ $T(t) + T'(t)dt = T(t) - T(t)\sigma(t)dt$

Solve for T

$$T(t + dt) = T(t)(1 - \sigma(t)dt)$$

Taylor expansion for T⇒ $T(t) + T'(t)dt = T(t) - T(t)\sigma(t)dt$

Rearrange⇒ $\frac{T'(t)}{T(t)} dt = -\sigma(t)dt$

Solve for T

$$T(t+dt) = T(t)(1-\sigma(t)dt)$$

Taylor expansion for T⇒ $T(t) + T'(t)dt = T(t) - T(t)\sigma(t)dt$

Rearrange⇒ $\frac{T'(t)}{T(t)} dt = -\sigma(t)dt$

Integrate⇒ $\log T(t) = -\int_{t_0}^{t} \sigma(s)ds$

Solve for T

$$T(t+dt) = T(t)(1-\sigma(t)dt)$$

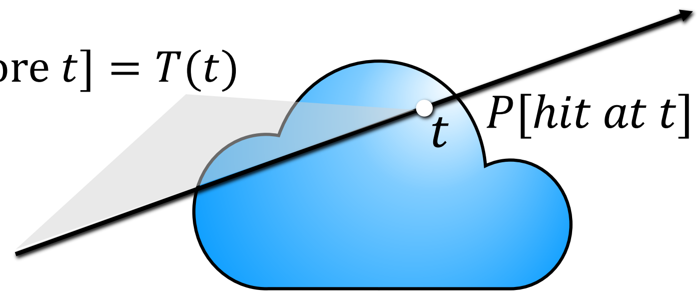
Taylor expansion for T⇒ $T(t) + T'(t)dt = T(t) - T(t)\sigma(t)dt$

Rearrange⇒ $\frac{T'(t)}{T(t)}dt = -\sigma(t)dt$

Integrate⇒ $\log T(t) = -\int_{t_0}^{t}\sigma(s)ds$

Exponentiate⇒ $T(t) = \exp\left(-\int_{t_0}^{t}\sigma(s)ds\right)$

PDF for ray termination



Finally, we can write the probability that a ray terminates at t as a function of only sigma

$$P[\text{first hit at } t] = P[\text{no hit before } t] \times P[\text{hit at } t]$$

$$= T(t)\sigma(t)dt$$

$$= \exp\left(-\int_{t_0}^{t} \sigma(s)ds\right) \sigma(t)dt$$

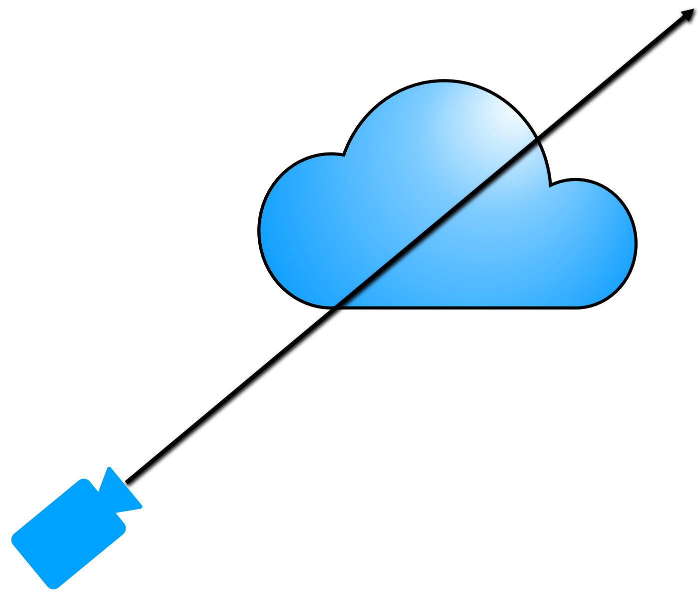
Expected value of color along ray

This means the expected color returned by the ray will be

$$\int_{t_0}^{t_1} T(t)\sigma(t)\mathbf{c}(t)dt$$

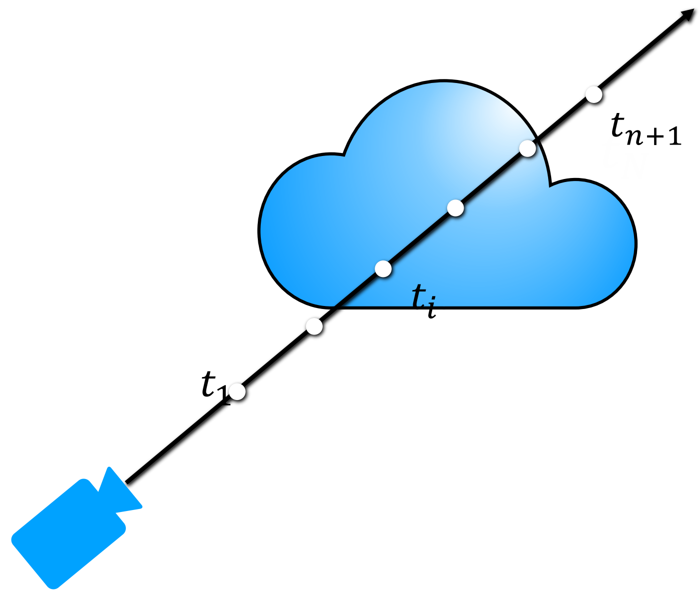
Note the nested integral!

Approximating the nested integral



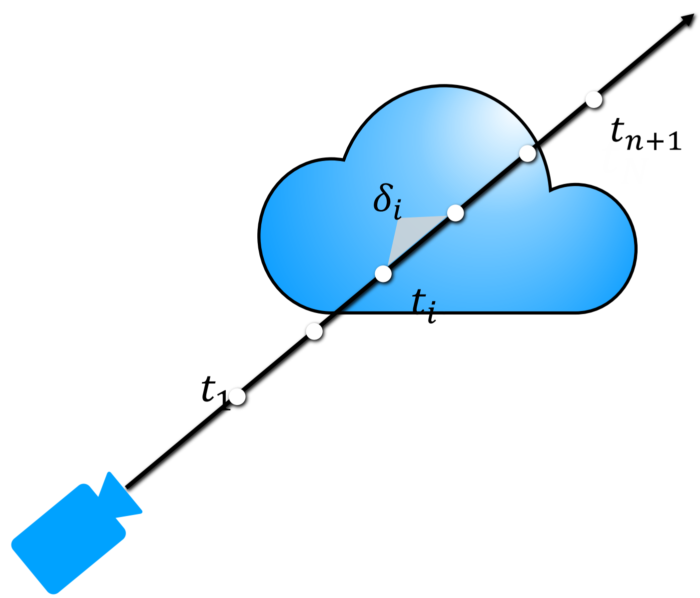
We use quadrature to approximate the nested integral,

Approximating the nested integral



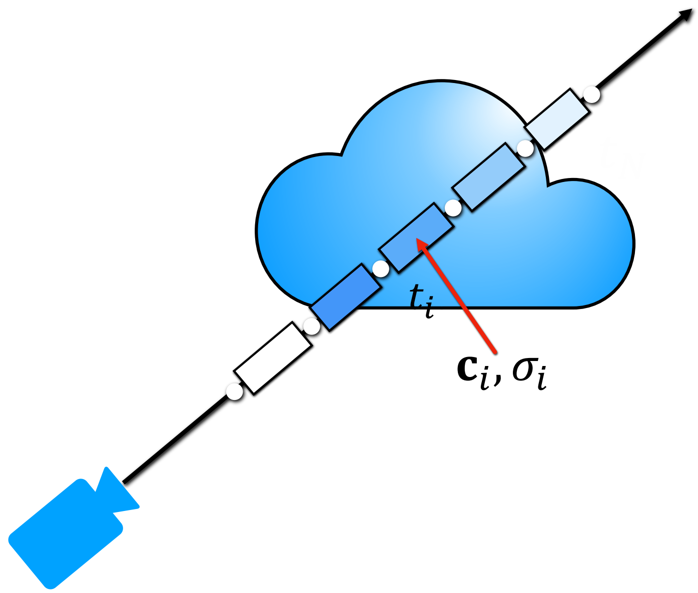
We use quadrature to approximate the nested integral, splitting the ray up into n segments with endpoints $\{t_1, t_2, \ldots, t_{n+1}\}$

Approximating the nested integral



We use quadrature to approximate the nested integral, splitting the ray up into n segments with endpoints $\{t_1, t_2, \dots, t_{n+1}\}$
with lengths $\delta_i = t_{i+1} - t_i$

Approximating the nested integral



We assume volume density and color are roughly constant within each interval

Deriving quadrature estimate

$\int T(t)\sigma(t)\mathbf{c}(t)dt \approx$

This allows us to break the outer integral

Deriving quadrature estimate

$$\int T(t)\sigma(t)\mathbf{c}(t)dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i dt$$

This allows us to break the outer integral into a sum of analytically tractable integrals

Deriving quadrature estimate

$$\int T(t)\sigma(t)\mathbf{c}(t)dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} \boxed{T(t)}\sigma_i \mathbf{c}_i dt$$

Caveat: piecewise constant density and color
do not imply constant transmittance!

Deriving quadrature estimate

$$\int T(t)\sigma(t)\mathbf{c}(t)dt \approx \sum_{i=1}^{n}\int_{t_i}^{t_{i+1}} \boxed{T(t)}\sigma_i \mathbf{c}_i dt$$

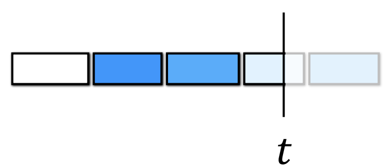
Caveat: piecewise constant density and color
do not imply constant transmittance!

Important to account for how early part of a segment blocks later part when σ_i is high

Evaluating T for piecewise constant density

For $t \in [t_i, t_{i+1}]$, $T(t) = \exp\left(-\int_{t_1}^{t_i} \sigma_i ds\right) \exp\left(-\int_{t_i}^{t} \sigma_i ds\right)$

We need to evaluate at continuous t values
that can lie *partway through* an interval

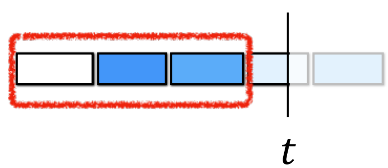


Evaluating T for piecewise constant density

For $t \in [t_i, t_{i+1}]$, $T(t) = \exp\left(-\int_{t_1}^{t_i} \sigma_i ds\right) \exp\left(-\int_{t_i}^{t} \sigma_i ds\right)$

$$\exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right) = T_i$$

"How much light is blocked by all previous segments?"

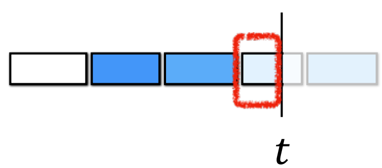


Evaluating T for piecewise constant density

For $t \in [t_i, t_{i+1}]$, $T(t) = \exp\left(-\int_{t_1}^{t_i} \sigma_i ds\right) \exp\left(-\int_{t_i}^{t} \sigma_i ds\right)$

"How much light is blocked partway through the current segment?"

$\exp(-\sigma_i(t - t_i))$



Deriving quadrature estimate

$$\int T(t)\sigma(t)\mathbf{c}(t)dt \approx \sum_{i=1}^{n}\int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i dt$$

Deriving quadrature estimate

$$\int T(t)\sigma(t)\mathbf{c}(t)dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i dt$$

Substitute $$= \sum_{i=1}^{n} T_i \sigma_i \mathbf{c}_i \int_{t_i}^{t_{i+1}} \exp(-\sigma_i(t - t_i))dt$$

Deriving quadrature estimate

$$\int T(t)\sigma(t)\mathbf{c}(t)dt \approx \sum_{i=1}^{n} \int_{t_i}^{t_{i+1}} T(t)\sigma_i \mathbf{c}_i dt$$

$$= \sum_{i=1}^{n} T_i \sigma_i \mathbf{c}_i \int_{t_i}^{t_{i+1}} \exp(-\sigma_i(t - t_i))dt$$

Integrate $= \sum_{i=1}^{n} T_i \sigma_i \mathbf{c}_i \frac{\exp(-\sigma_i(t_{i+1} - t_i)) - 1}{-\sigma_i}$

Deriving quadrature estimate

$$\int T(t)\sigma(t)\mathbf{c}(t)dt \approx \sum_{i=1}^{n}\int_{t_i}^{t_{i+1}} T(t)\sigma_i\mathbf{c}_i dt$$

$$= \sum_{i=1}^{n} T_i\sigma_i\mathbf{c}_i \int_{t_i}^{t_{i+1}} \exp(-\sigma_i(t - t_i))dt$$

$$= \sum_{i=1}^{n} T_i\sigma_i\mathbf{c}_i \frac{\exp(-\sigma_i(t_{i+1} - t_i)) - 1}{-\sigma_i}$$

Cancel σ_i $$= \sum_{i=1}^{n} T_i\mathbf{c}_i(1 - \exp(-\sigma_i\delta_i))$$

Connection to alpha compositing

$$= \sum_{i=1}^{n} T_i \mathbf{c}_i \underbrace{(1 - \exp(-\sigma_i \delta_i))}_{\text{segment opacity } \alpha_i}$$

Connection to alpha compositing

$$= \sum_{i=1}^{n} T_i \mathbf{c}_i \underbrace{(1 - \exp(-\sigma_i \delta_i))}_{\text{segment opacity } \alpha_i}$$

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

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

Summary: volume rendering integral estimate

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

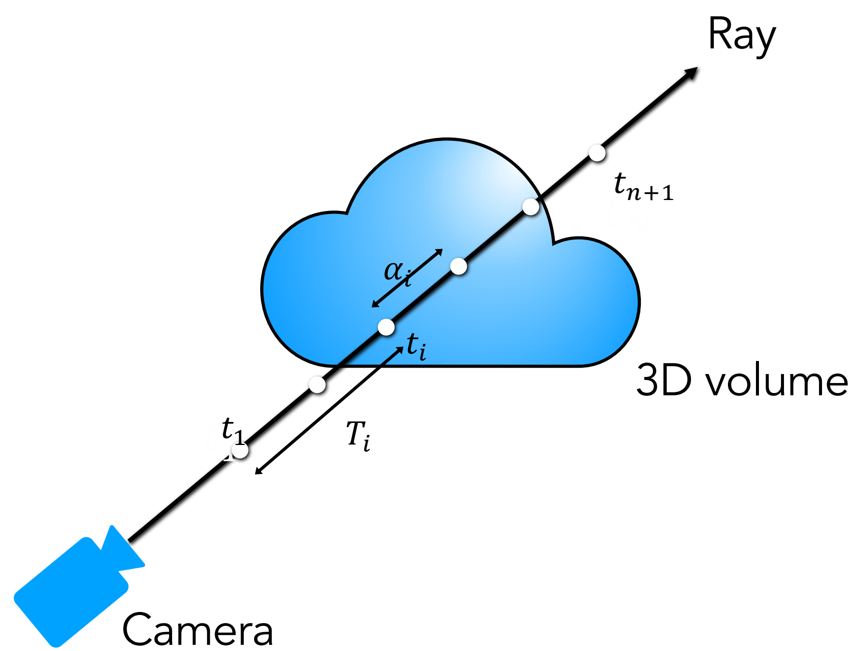
$$\mathbf{c} \approx \sum_{i=1}^{n} \underbrace{T_i \alpha_i}_{\text{weights}} \underbrace{\mathbf{c}_i}_{\text{colors}}$$

How much light is blocked earlier along ray:

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

How much light is contributed by ray segment *i*:

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



Volume rendering is trivially differentiable

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

$$\boxed{\mathbf{c} \approx \sum_{i=1}^{n} \underbrace{T_i \alpha_i}_{\text{weights}} \underbrace{\mathbf{c}_i}_{\text{colors}}}$$

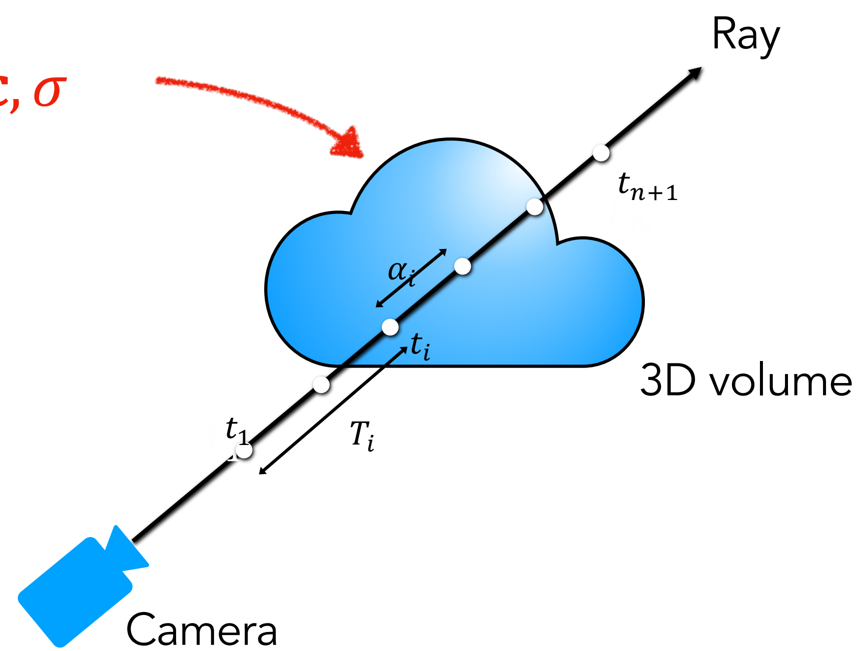
differentiable w.r.t. c, σ

How much light is blocked earlier along ray:

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

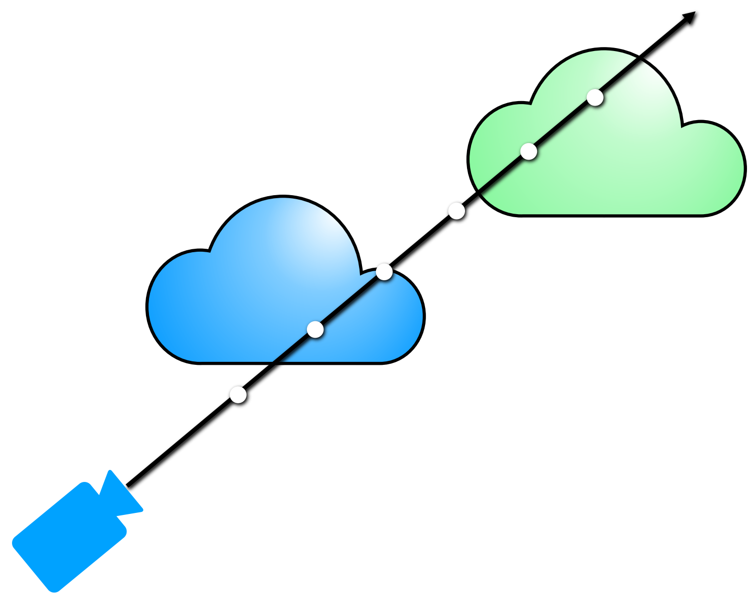
How much light is contributed by ray segment *i*:

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

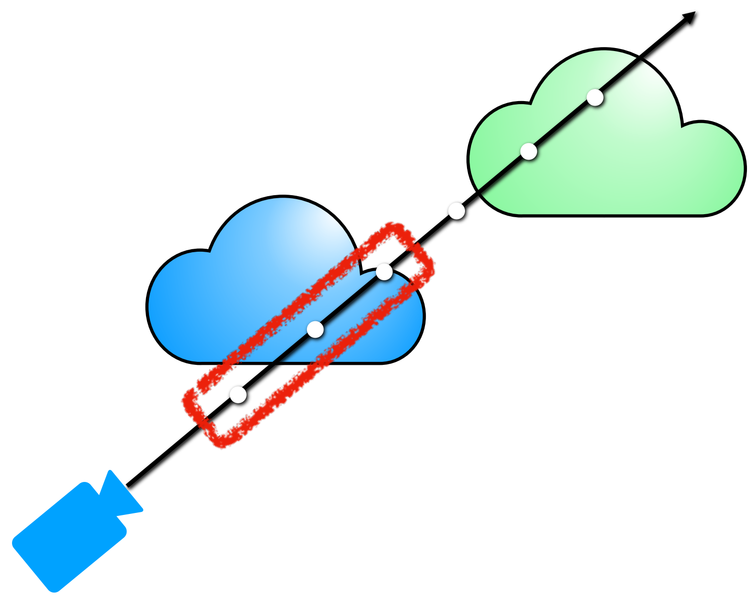


Further points on volume rendering

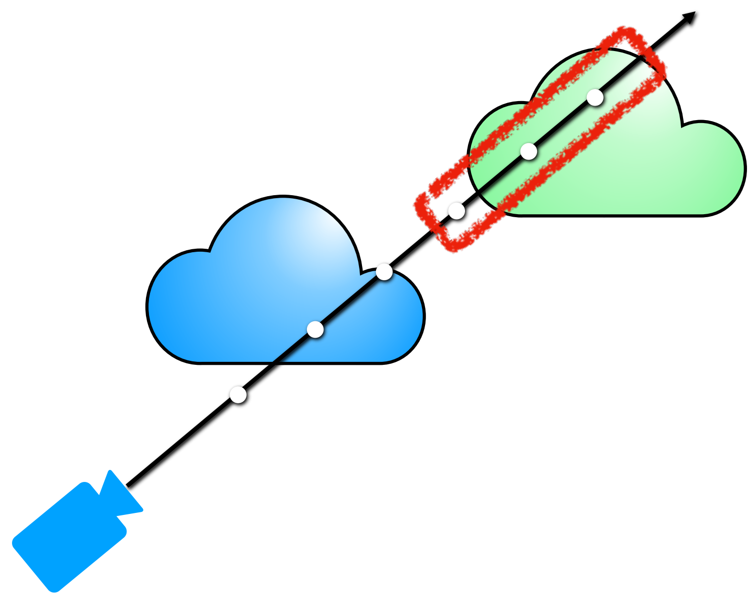
Alpha mattes and compositing



Alpha mattes and compositing



Alpha mattes and compositing



Alpha mattes and compositing

Mildenhall*, Srinivasan*, Tancik* et al 2020, *NeRF*
Poole et al 2022, *DreamFusion*
Tang et al 2022, *Compressible-composable NeRF via Rank-residual Decomposition*

Rendering weight PDF is important

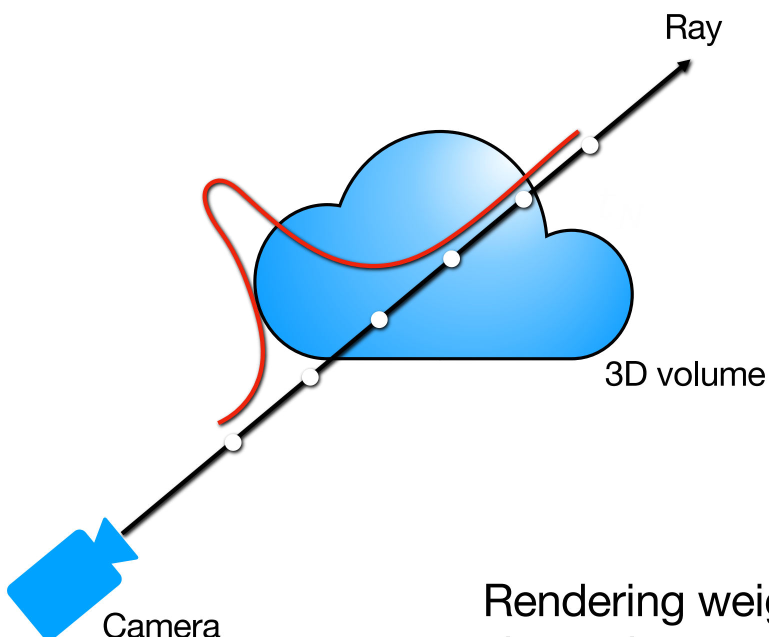
Remember, expected color is equal to

$$\int T(t)\sigma(t)\mathbf{c}(t)dt \approx \sum_i T_i \alpha_i \mathbf{c}_i$$

$T(t)\sigma(t)$ and $T_i\alpha_i$ are "rendering weights" — probability distribution along the ray (continuous and discrete, respectively)

Visual intuition — rendering weights not just 3D function

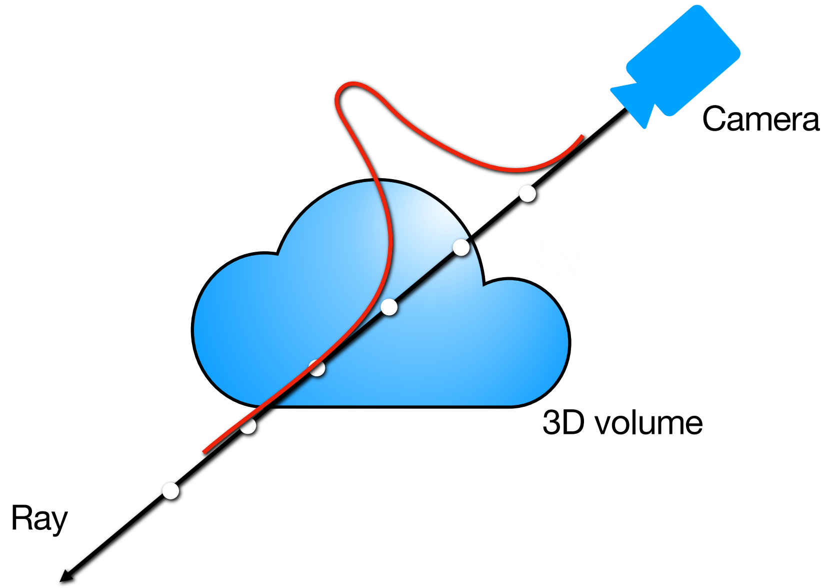
$$C \approx \sum_{i=1}^{N} \boxed{T_i \alpha_i} c_i$$



Rendering weights are not a 3D function — depends on ray, because of tranmisttance!

Visual intuition — rendering weights not just 3D function

$$C \approx \sum_{i=1}^{N} \boxed{T_i \alpha_i} c_i$$



Rendering weights are not a 3D function — depends on ray, because of tranmisttance!

Rendering weight PDF is important — depth

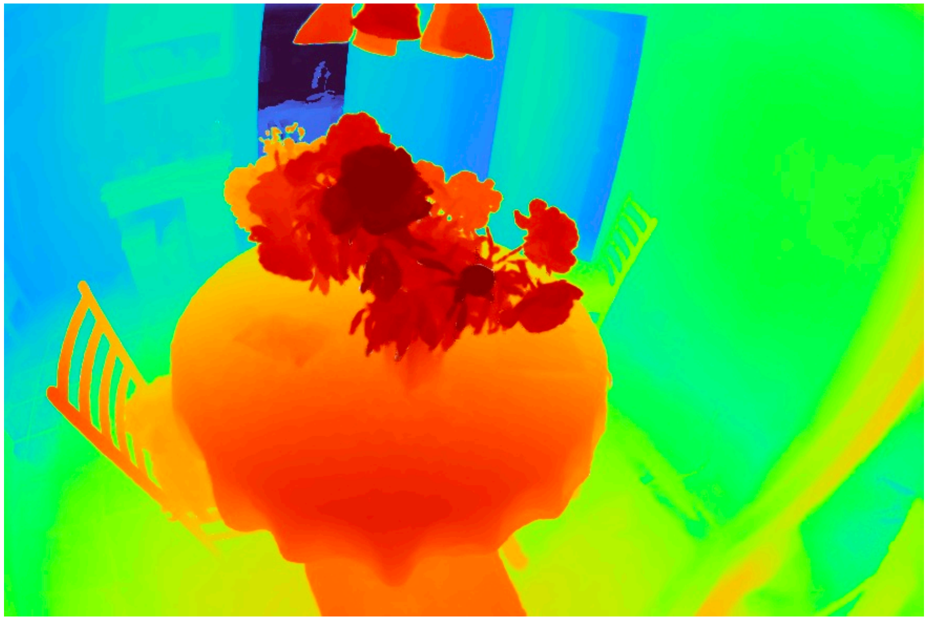
We can use this distribution to compute expectations for other quantities, e.g. "expected depth":

$$\bar{t} = \sum_i T_i \alpha_i t_i$$

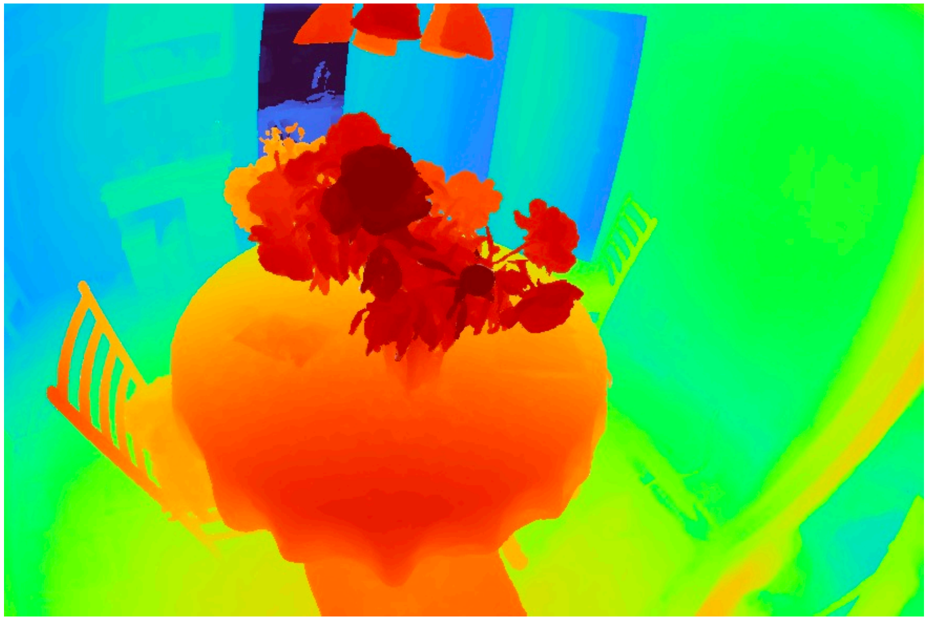
This is often how people visualise NeRF depth maps.

Alternatively, other statistics like mode or median can be used.

Rendering weight PDF is important — depth

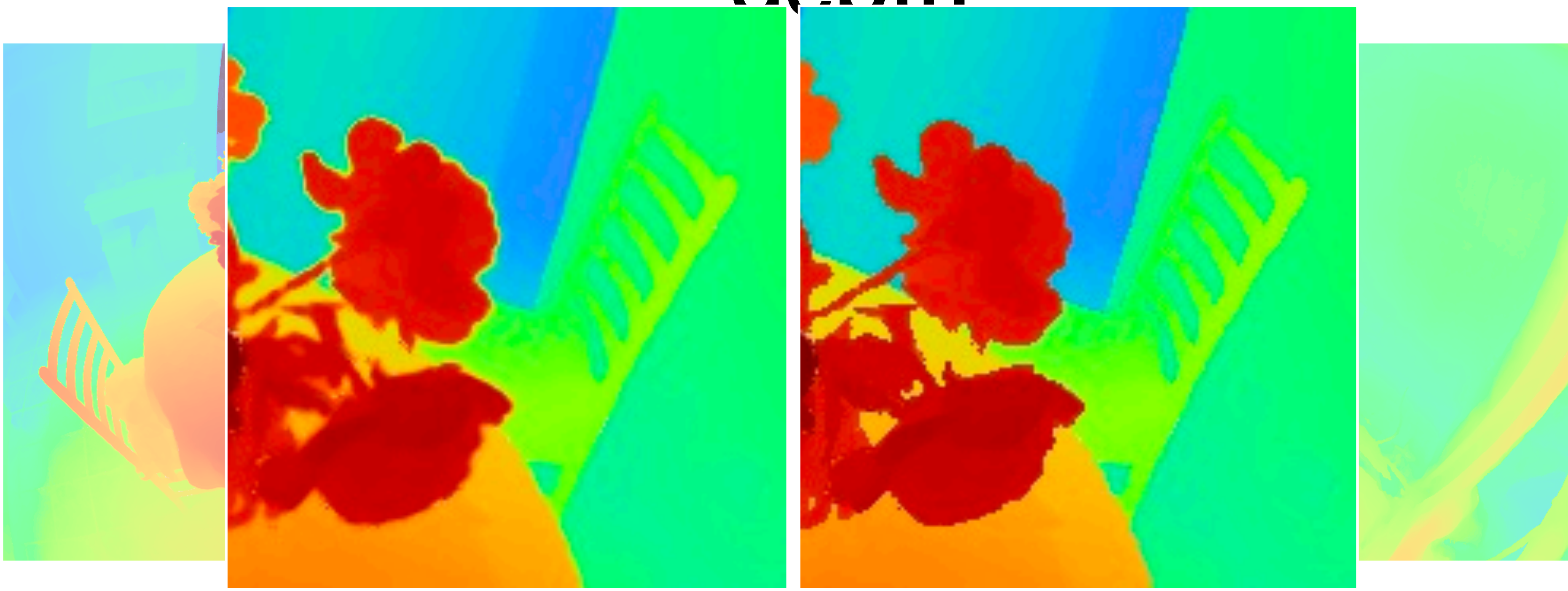


Mean depth



Median depth

Rendering weight PDF is important — depth



Mean depth

Median depth

Volume rendering other quantities

This idea can be used for any quantity we want to "volume render" into a 2D image.
If $\mathbf{v}$ lives in 3D space (semantic features, normal vectors, etc.)

$$\sum_i T_i \alpha_i \mathbf{v}_i$$

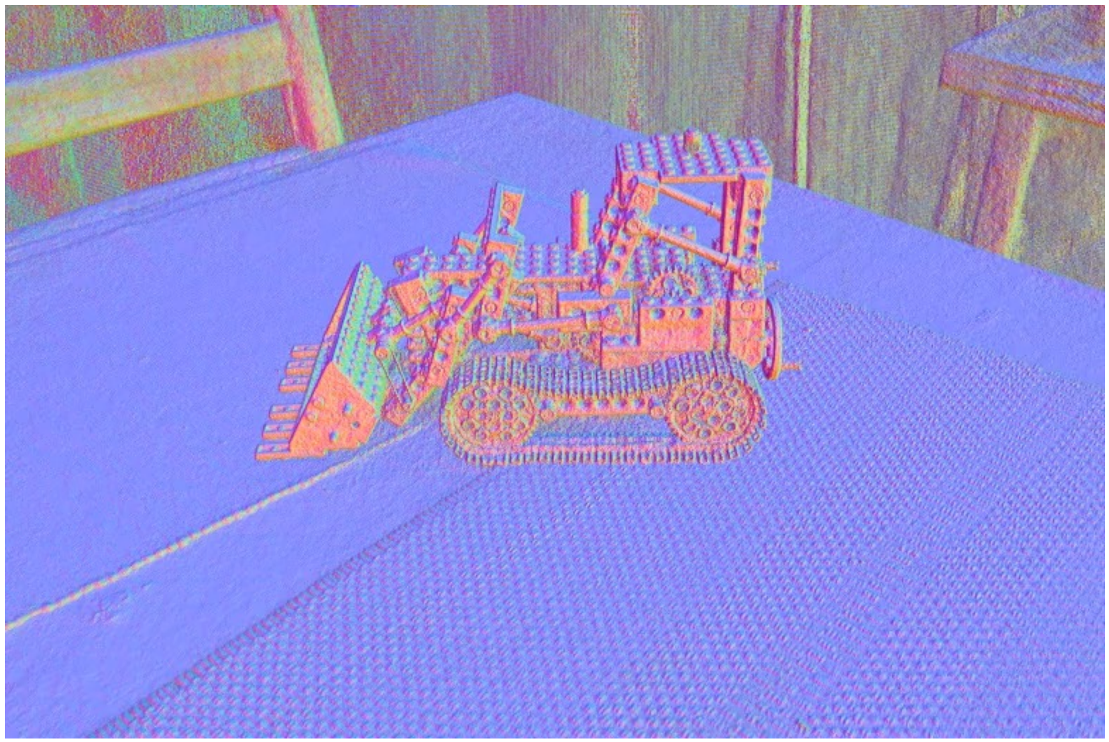
can be taken per-ray to produce 2D output images.

Volume rendering other quantities

Various recent works have used this idea to render higher-level semantic feature maps (e.g., *Feature Field Distillation* and *Neural Feature Fusion Fields*).

Kobayashi et al 2022, *Decomposing NeRF for Editing via Feature Field Distillation*

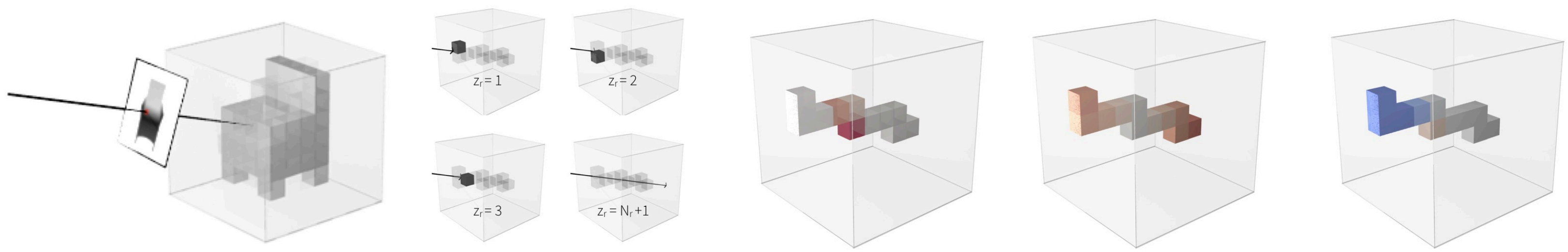
Density as geometry



Normal vectors (from analytic gradient of density)

Applications/optimizing differentiable volume rendering

Alpha compositing model in ML/computer vision



Differentiable ray consistency work used a forward model with “probabilistic occupancy” to supervise 3D-from-single-image prediction. Same rendering model as alpha compositing!

$$p(z_r = i) = \begin{cases} (1 - x_i^r) \prod_{j=1}^{i-1} x_j^r, & \text{if } i \leq N_r \\ \prod_{j=1}^{N_r} x_j^r, & \text{if } i = N_r + 1 \end{cases}$$

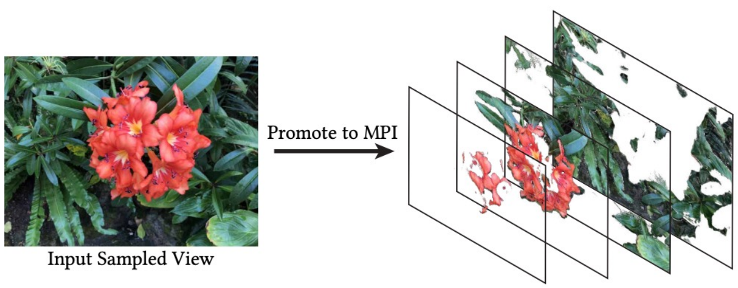
Tulsiani et al 2017, *Multi-view Supervision for Single-view Reconstruction via Differentiable Ray Consistency*

Volume rendering for view synthesis

Multiplane image methods

Stereo Magnification (Zhou et al. 2018)
Pushing the Boundaries… (Srinivasan et al. 2019)
Local Light Field Fusion (Mildenhall et al. 2019)
DeepView (Flynn et al. 2019)
Single-View… (Tucker & Snavely 2020)

Typical deep learning pipelines - images go into a 3D CNN, big RGBA 3D volume comes out



Neural Volumes

(Lombardi et al. 2019)
Direct gradient descent to optimize an RGBA volume, regularized by a 3D CNN