3D-aware Synthesis

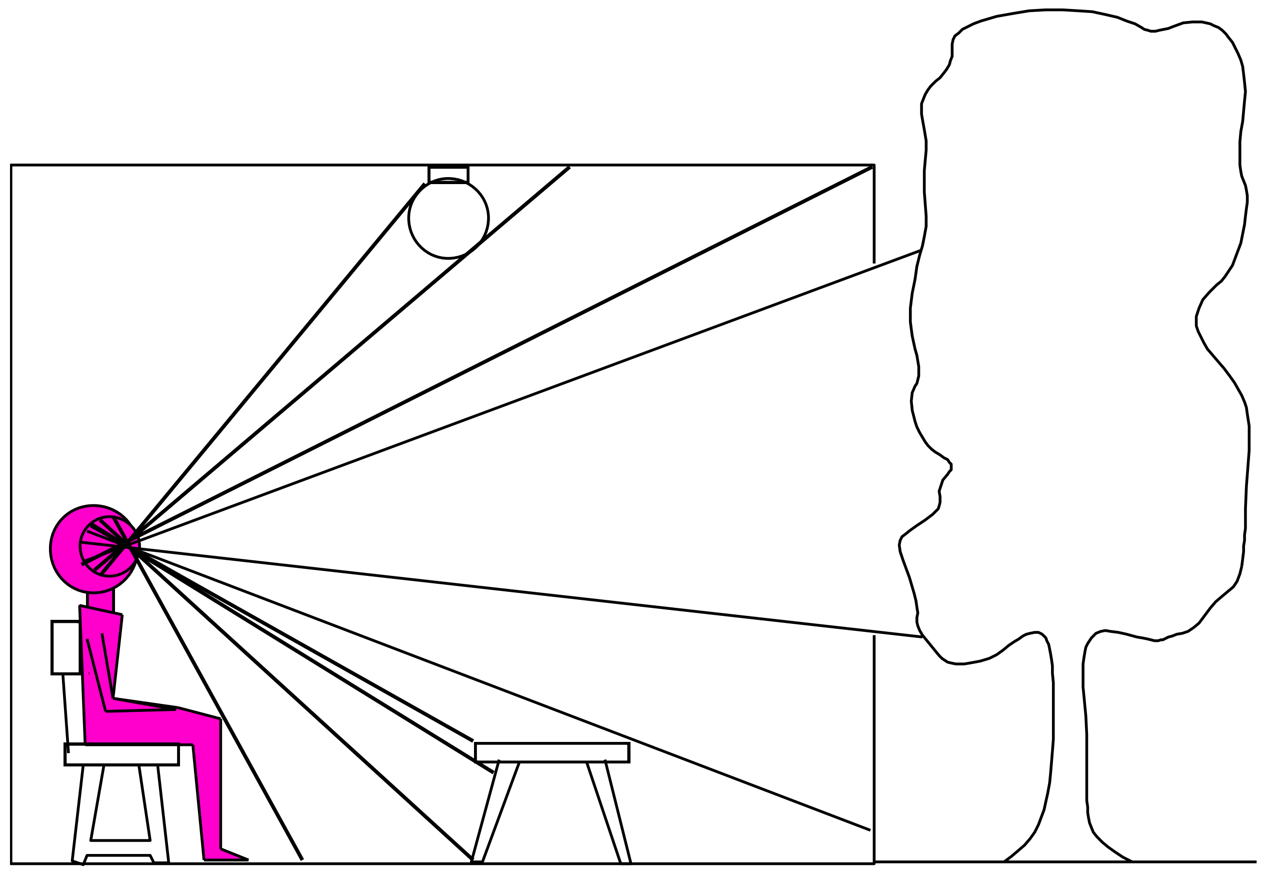
Jun-Yan Zhu

16-726, Spring 2025

Many slides from Alyosha Efros, Rick Szeliski, Michael Cohen
Paul Srinivasan, Ben Midlenhall, Jon Barron, Ren Ng

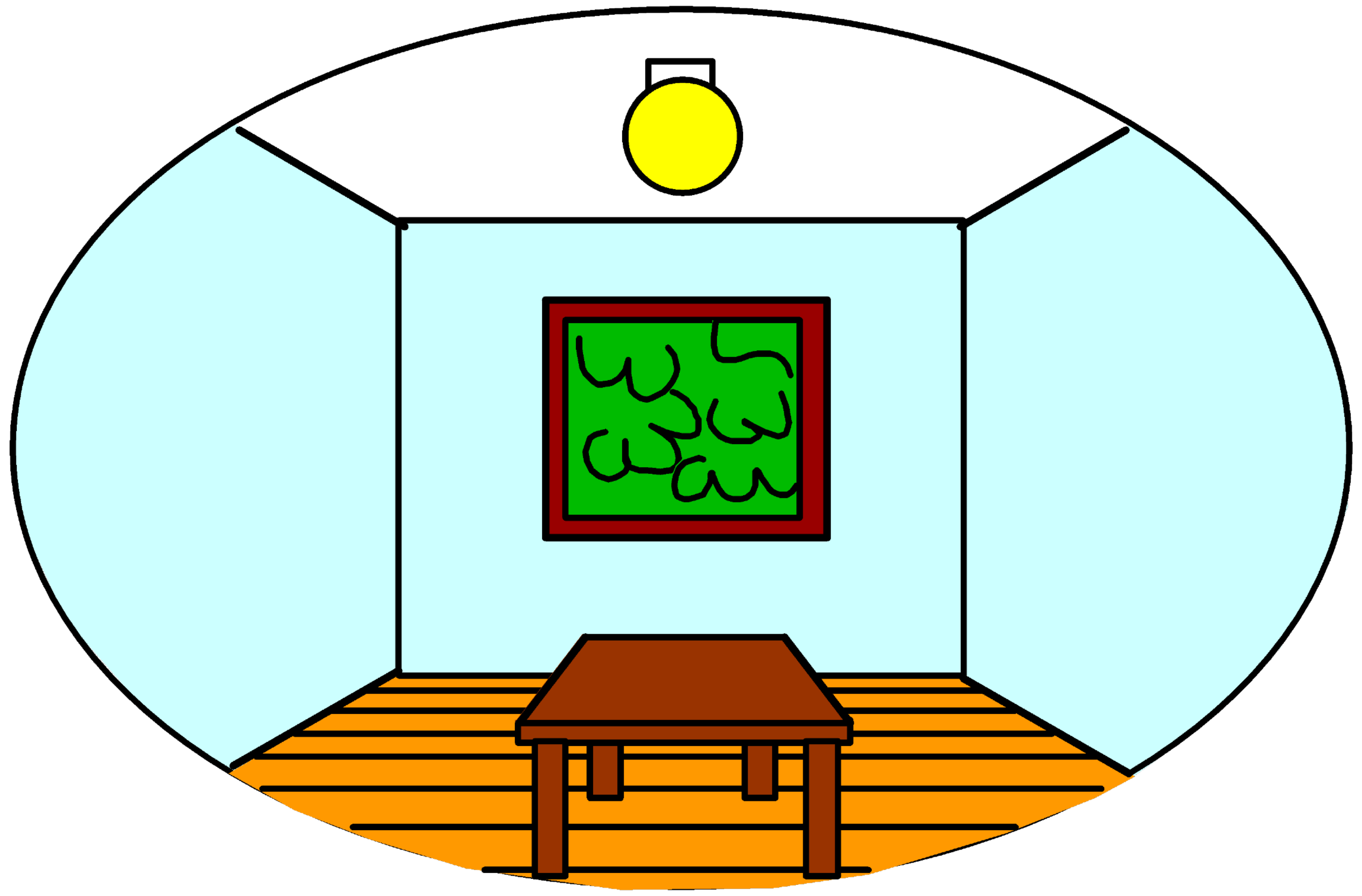
What do we see?

3D world



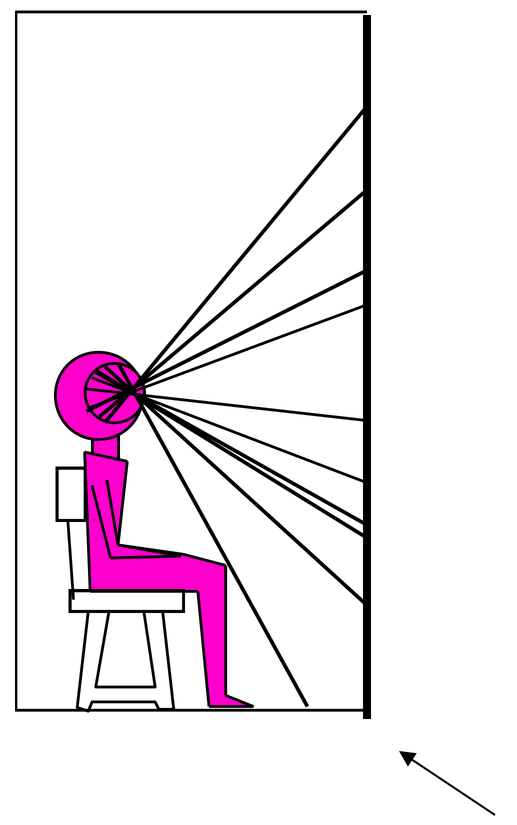
Point of observation

2D image

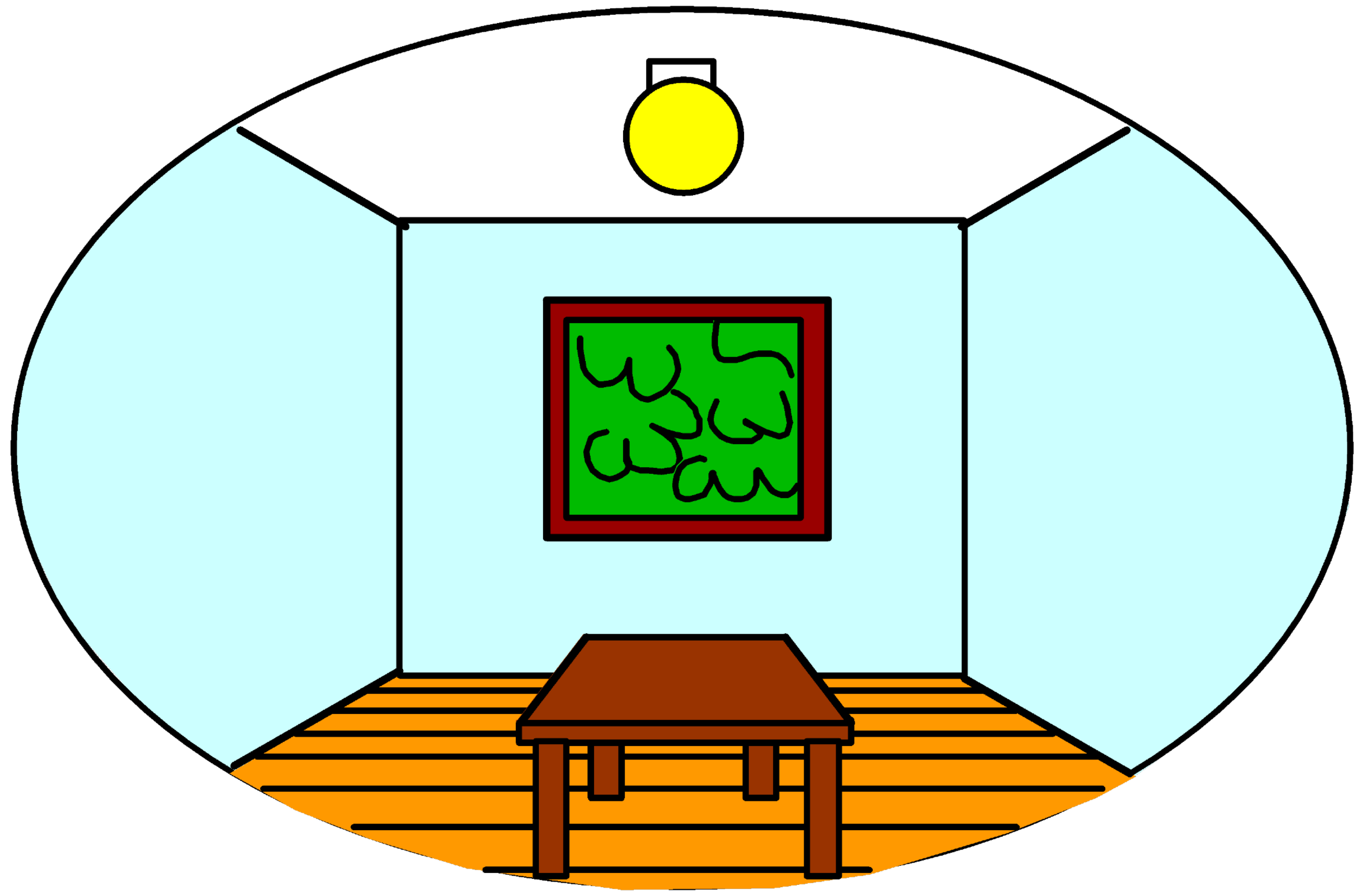


What do we see?

3D world



2D image



The Plenoptic Function

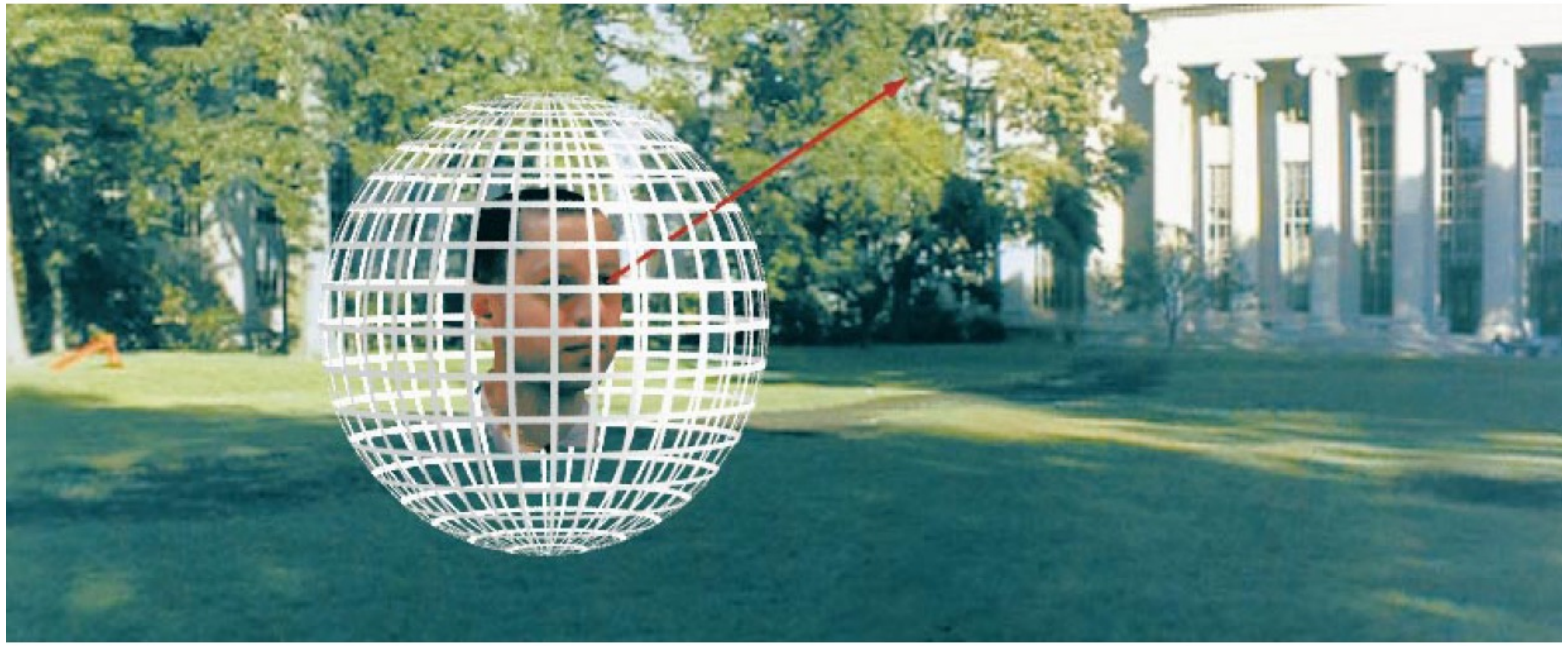


Figure by Leonard McMillan

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

- A: The Plenoptic Function (Adelson & Bergen)

- Let's start with a stationary person and try to parameterize <u>everything</u> that she or he can see…

from Alyosha Efros

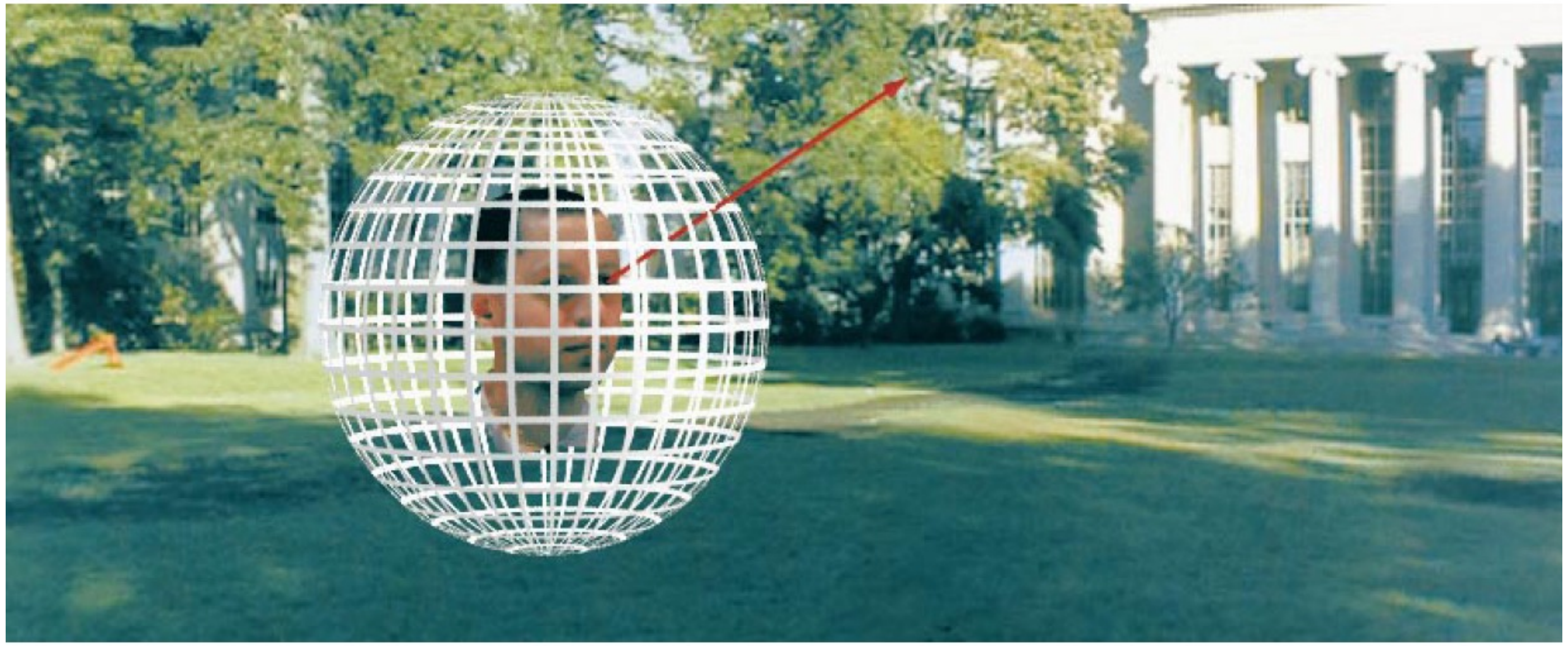
Grayscale snapshot

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

- is intensity of light
  - Seen from a single view point
  - At a single time
  - Averaged over the wavelengths of the visible spectrum
- (can also do *P(x,y),* but spherical coordinate are nicer)

from Alyosha Efros

Color snapshot

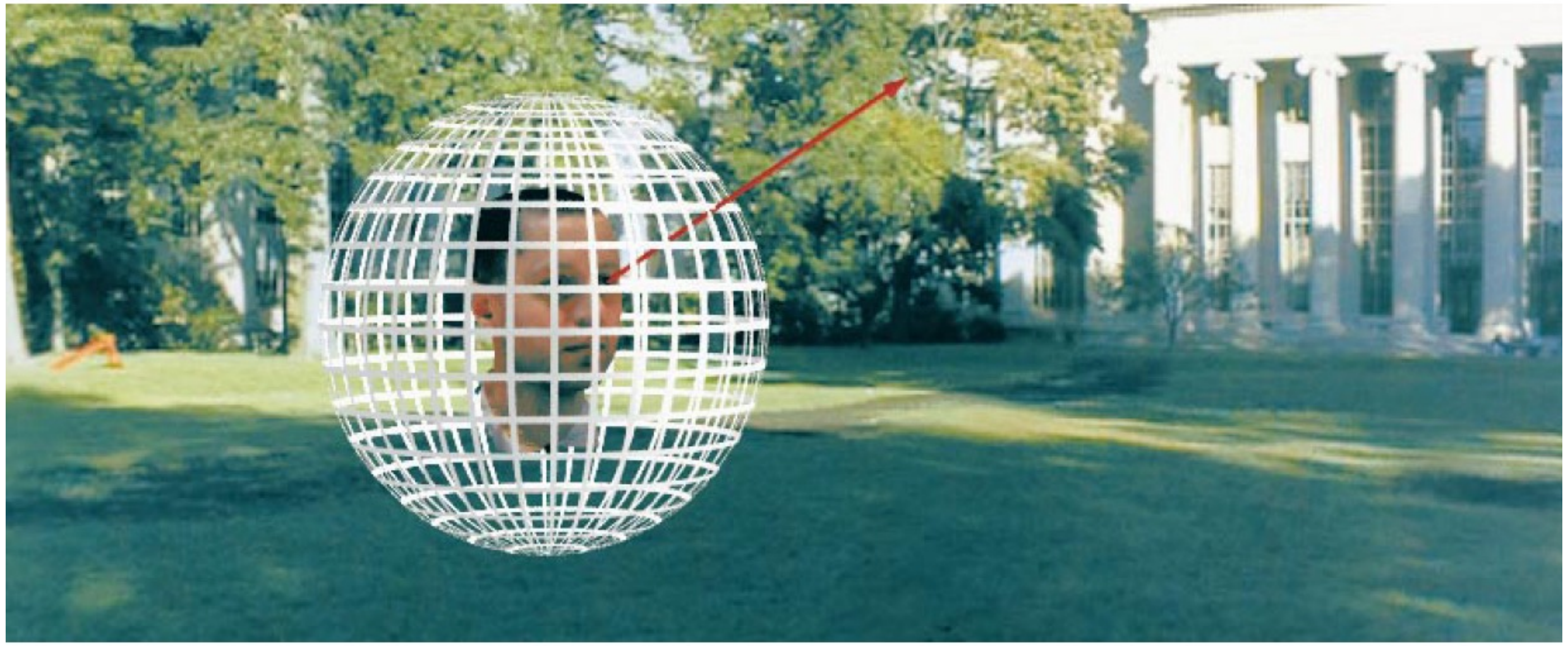


$$P(\theta,\phi,\lambda)$$

- is intensity of light
  - Seen from a single view point
  - At a single time
  - As a function of wavelength

from Alyosha Efros

A movie

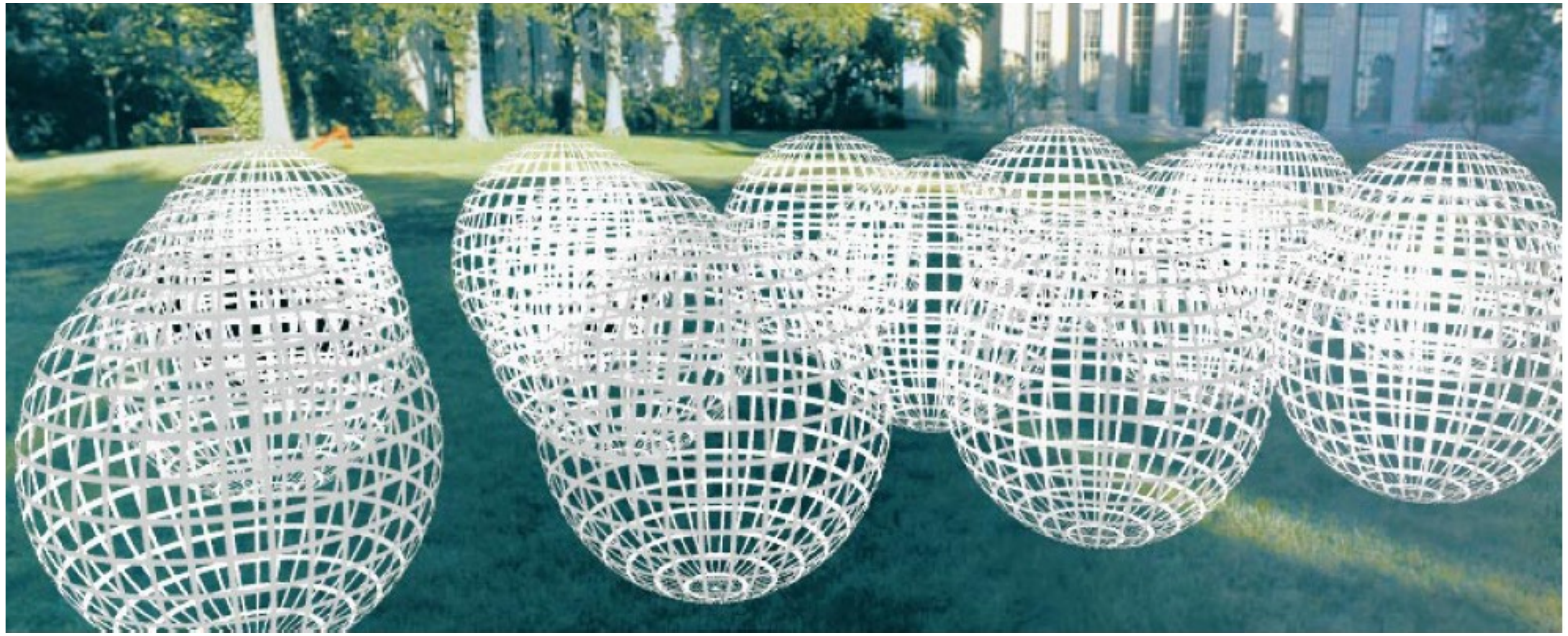


$$P(\theta,\phi,\lambda,t)$$

- is intensity of light
  - Seen from a single view point
  - Over time
  - As a function of wavelength

from Alyosha Efros

Holographic movie

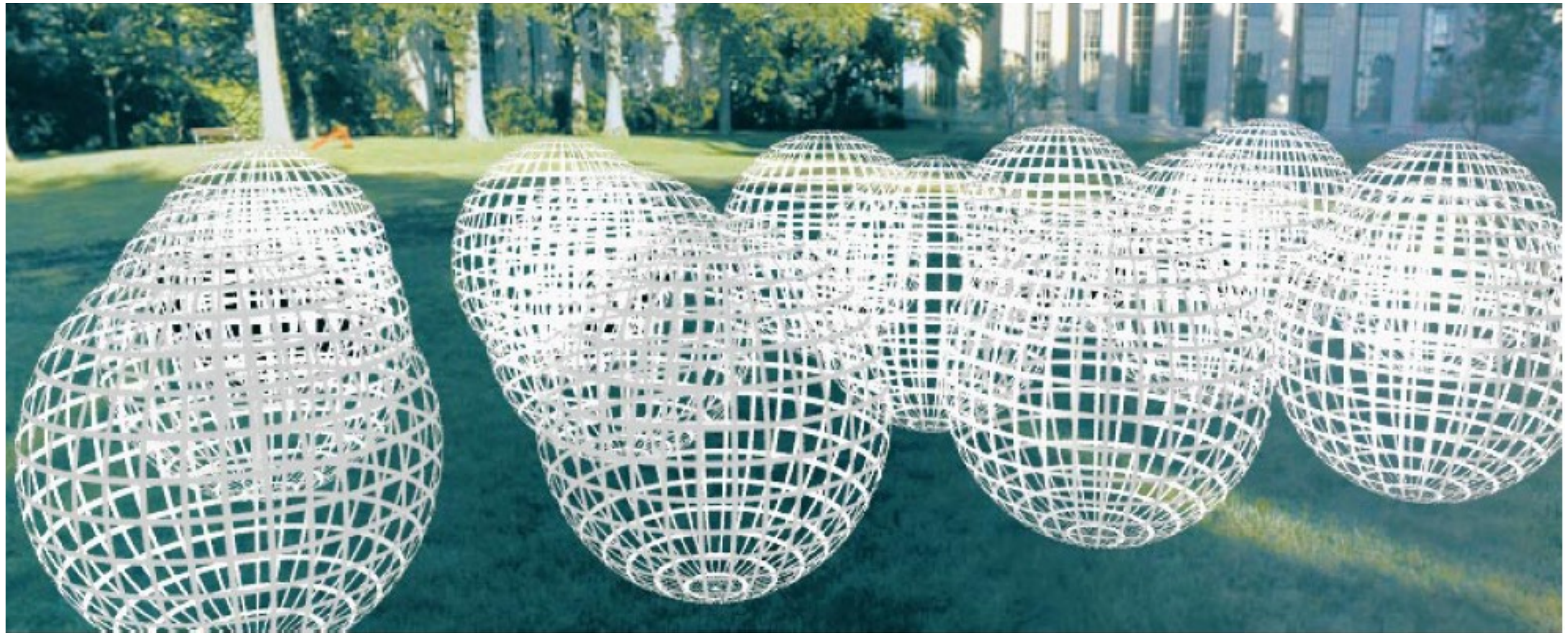


$$P(\theta,\phi,\lambda,t,V_X,V_Y,V_Z)$$

- is intensity of light
  - Seen from ANY viewpoint
  - Over time
  - As a function of wavelength

from Alyosha Efros

The Plenoptic Function



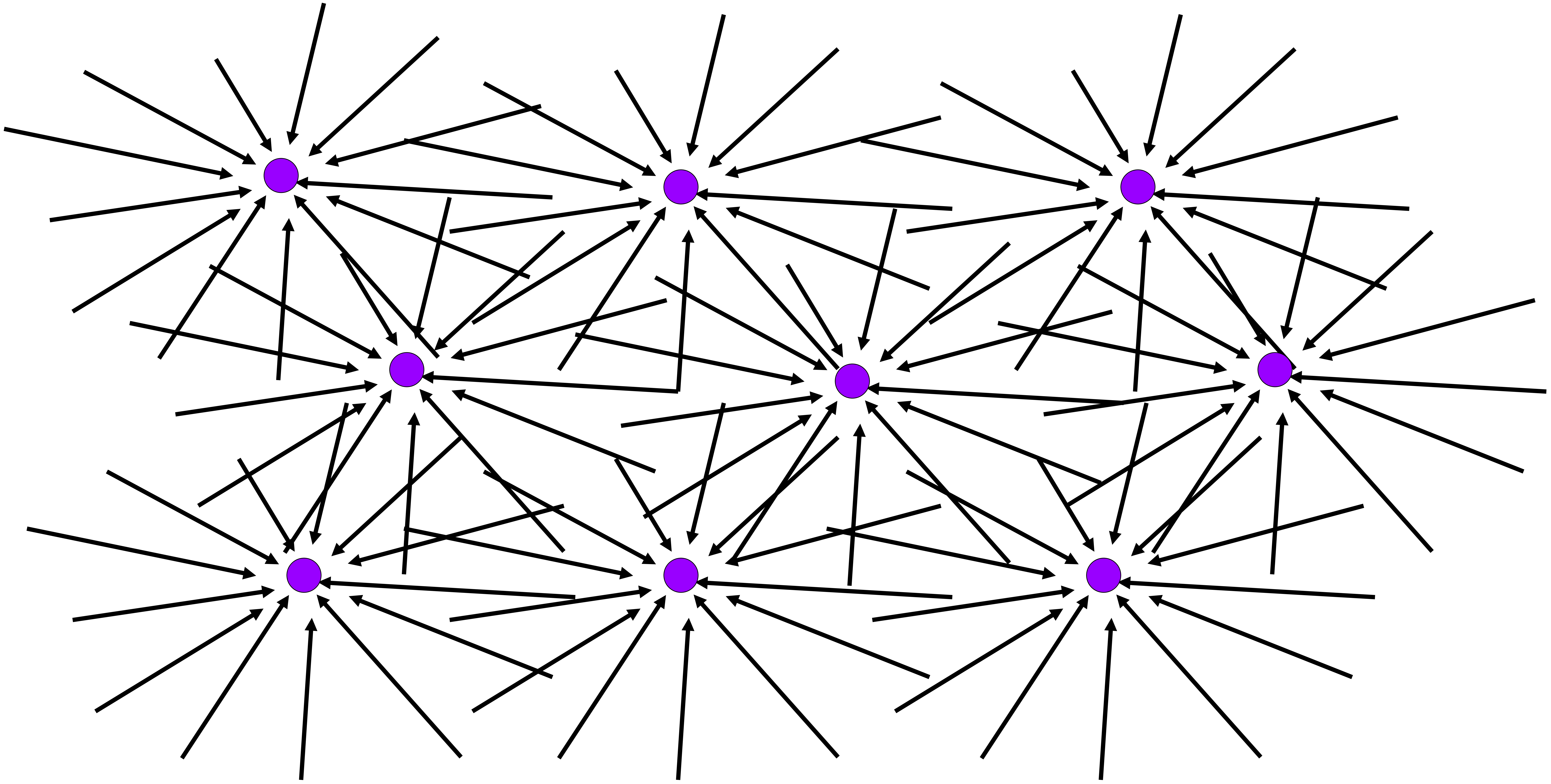
$$P(\theta,\phi,\lambda,t,V_X,V_Y,V_Z)$$

- Can reconstruct every possible view, at every moment, from every position, at every wavelength

- Contains every photograph, every movie, everything that anyone has ever seen! it completely captures our visual reality! Not bad for a function…

from Alyosha Efros

Sampling Plenoptic Function (top view)



Just lookup -- Quicktime VR

QuickTime VR

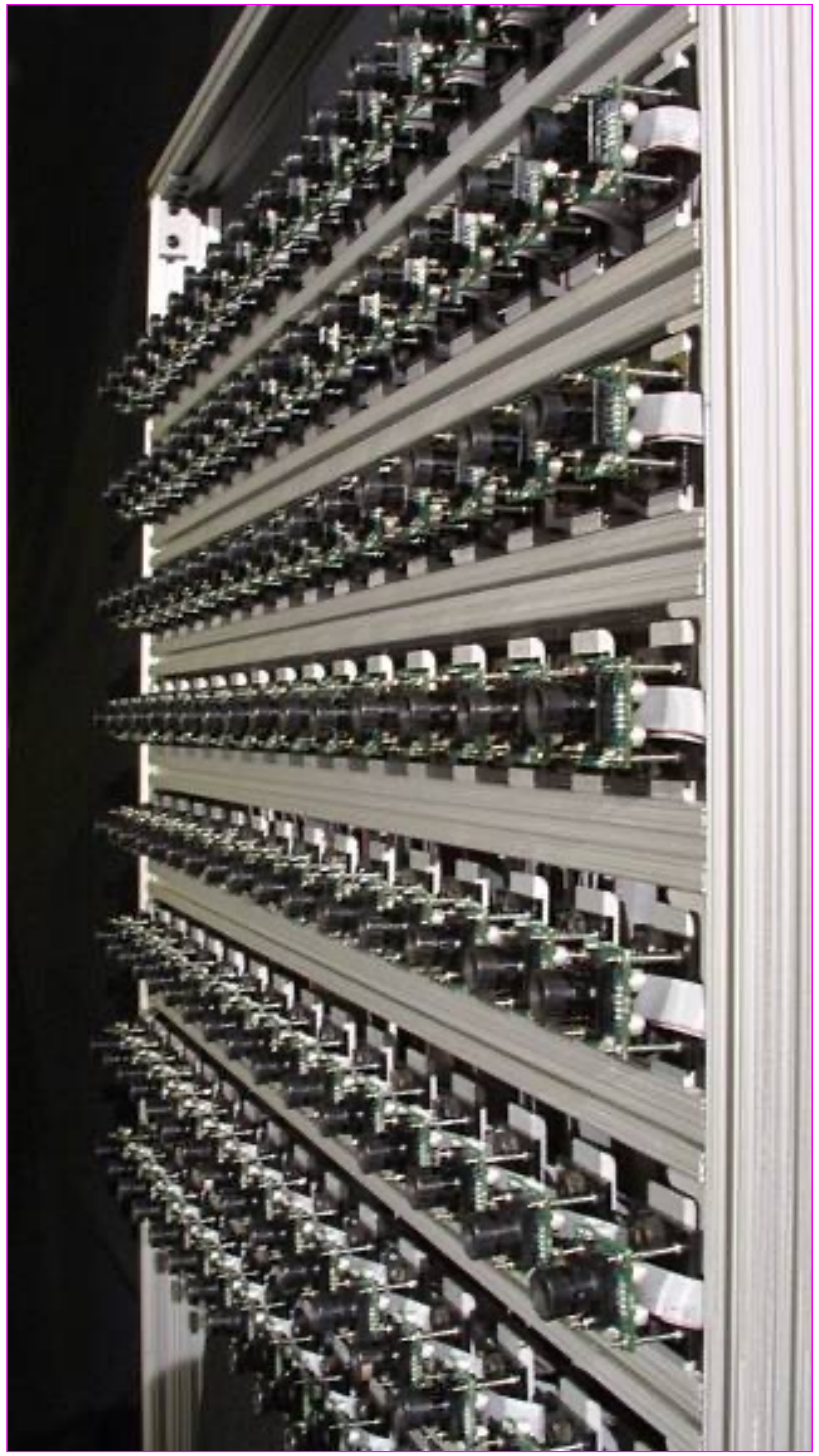
Panoramic image

Perspective Warp

QuickTime VR

Quicktime VR: An image-based approach to virtual environment navigation. Shenchang Eric Chen. SIGGRAPH 1995

Stanford multi-camera array



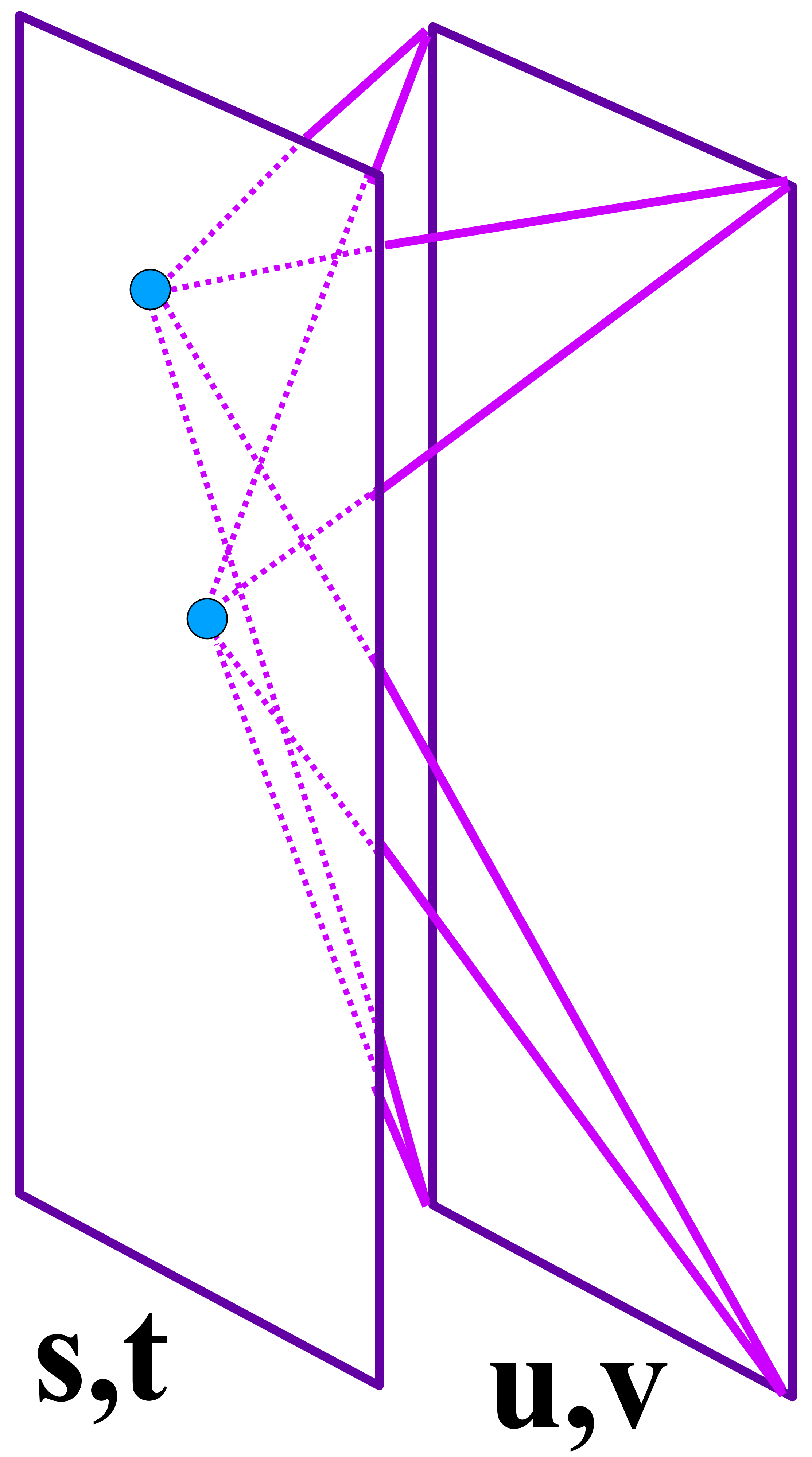
- 640 × 480 pixels × 30 fps × 128 cameras
- synchronized timing
- continuous streaming
- flexible arrangement

Lumigraph - Organization

Hold s,t constant

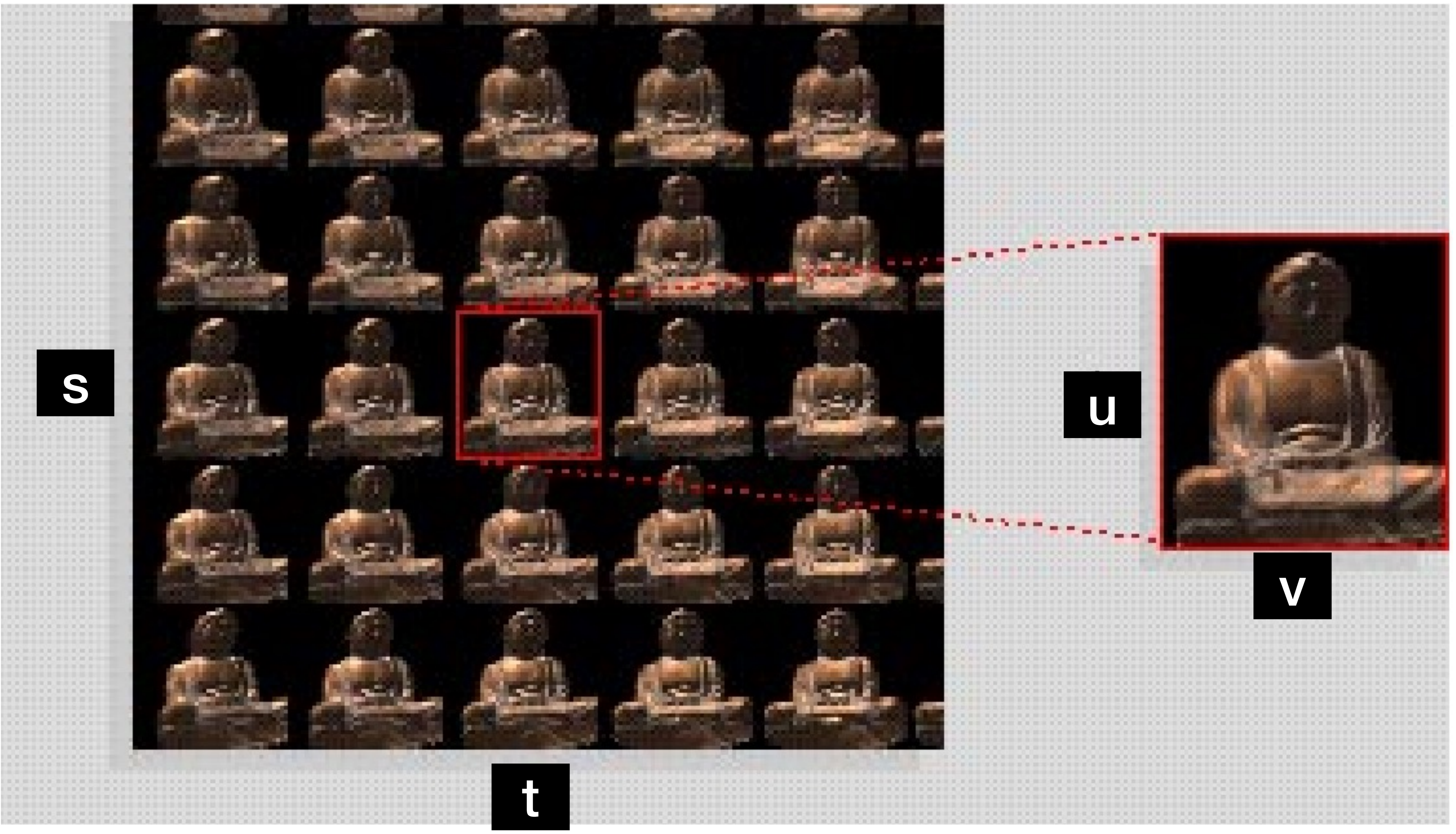
Let u,v vary

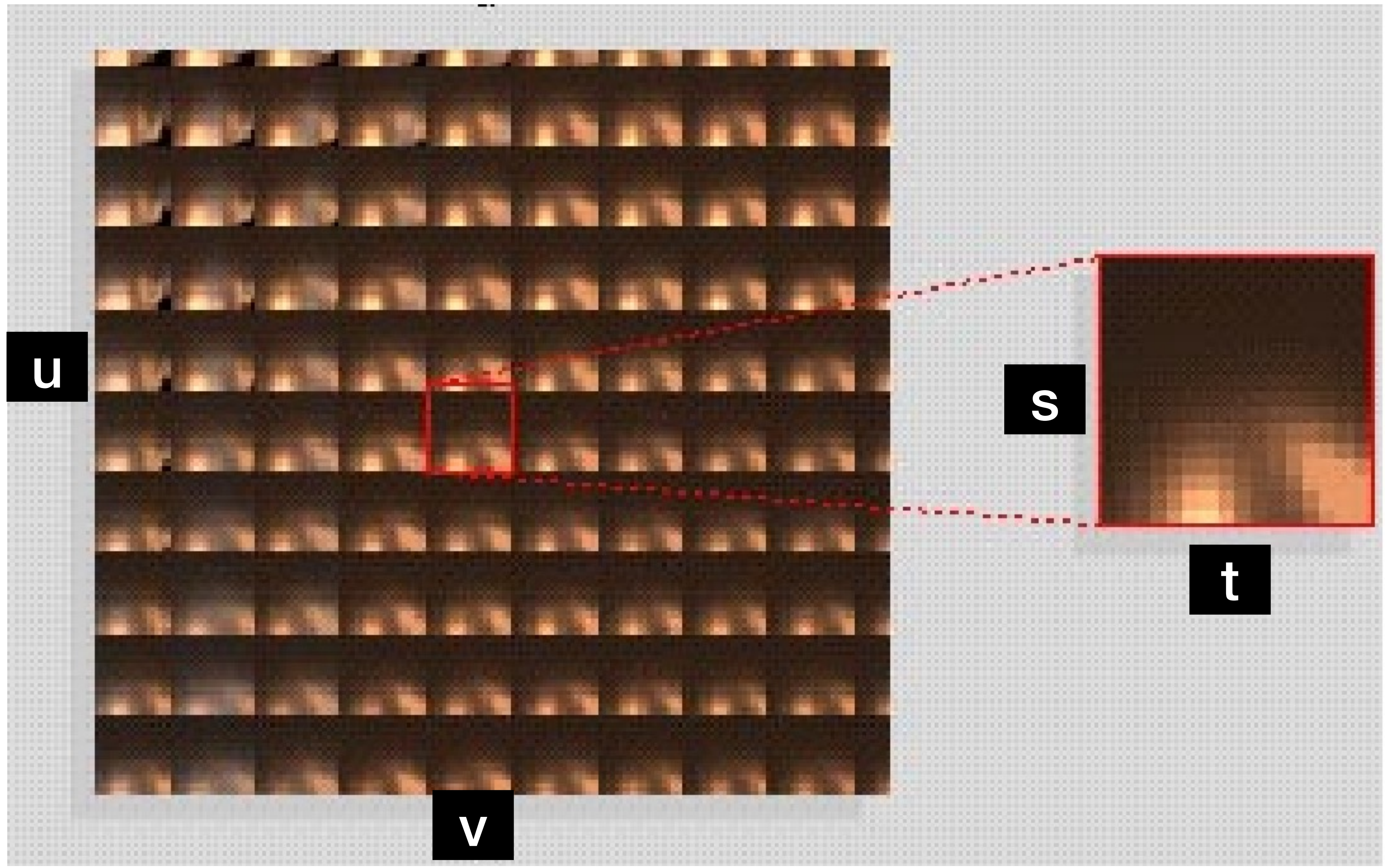
An image



Slide by Rick Szeliski and Michael Cohen

Lumigraph / Lightfield

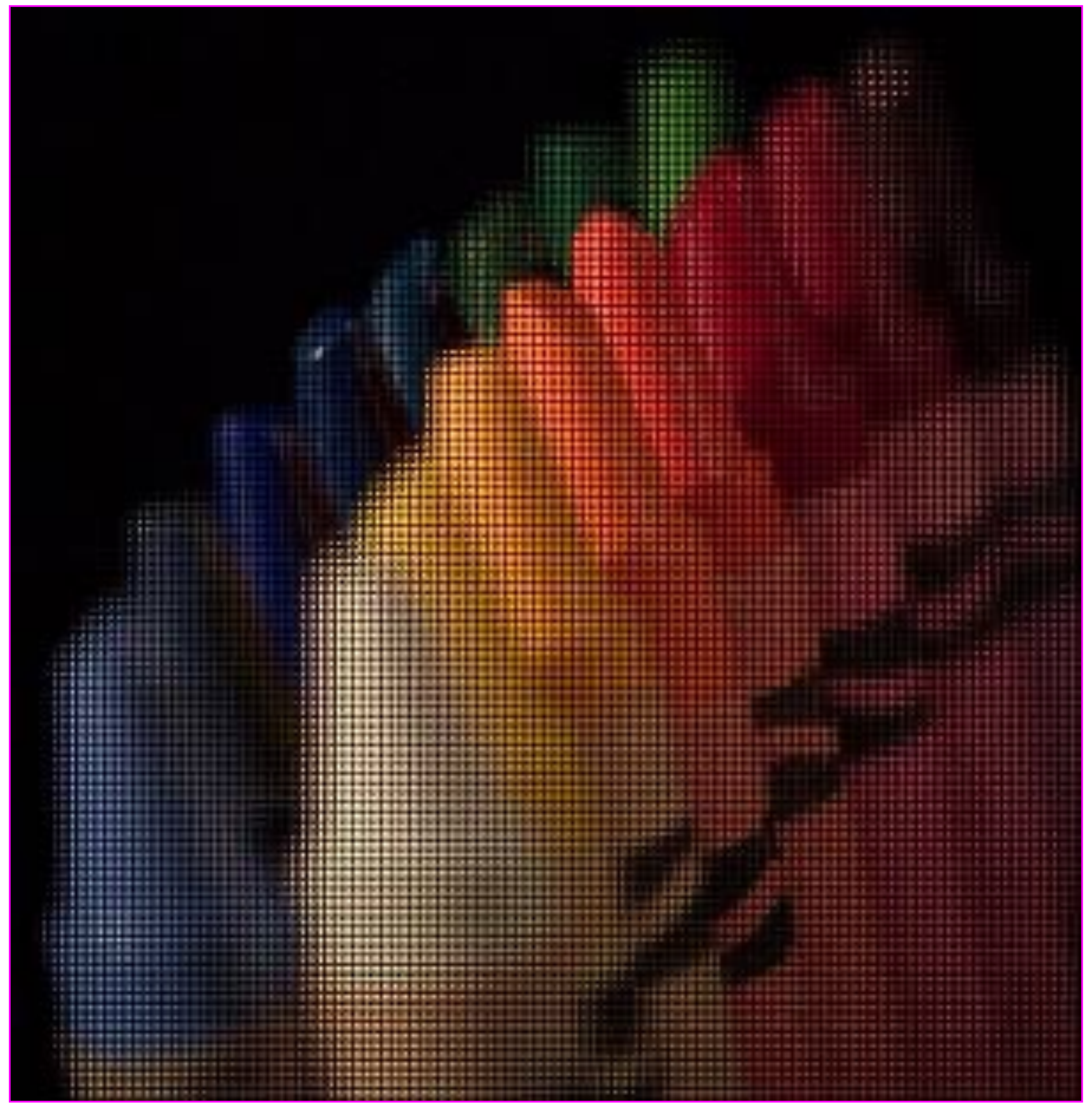




from Marc Levoy and Pat Hanrahan

Light field photography using a handheld plenoptic camera

Ren Ng, Marc Levoy, Mathieu Brédif,
Gene Duval, Mark Horowitz and Pat Hanrahan



Ren Ng

Light field photography using a handheld plenoptic camera

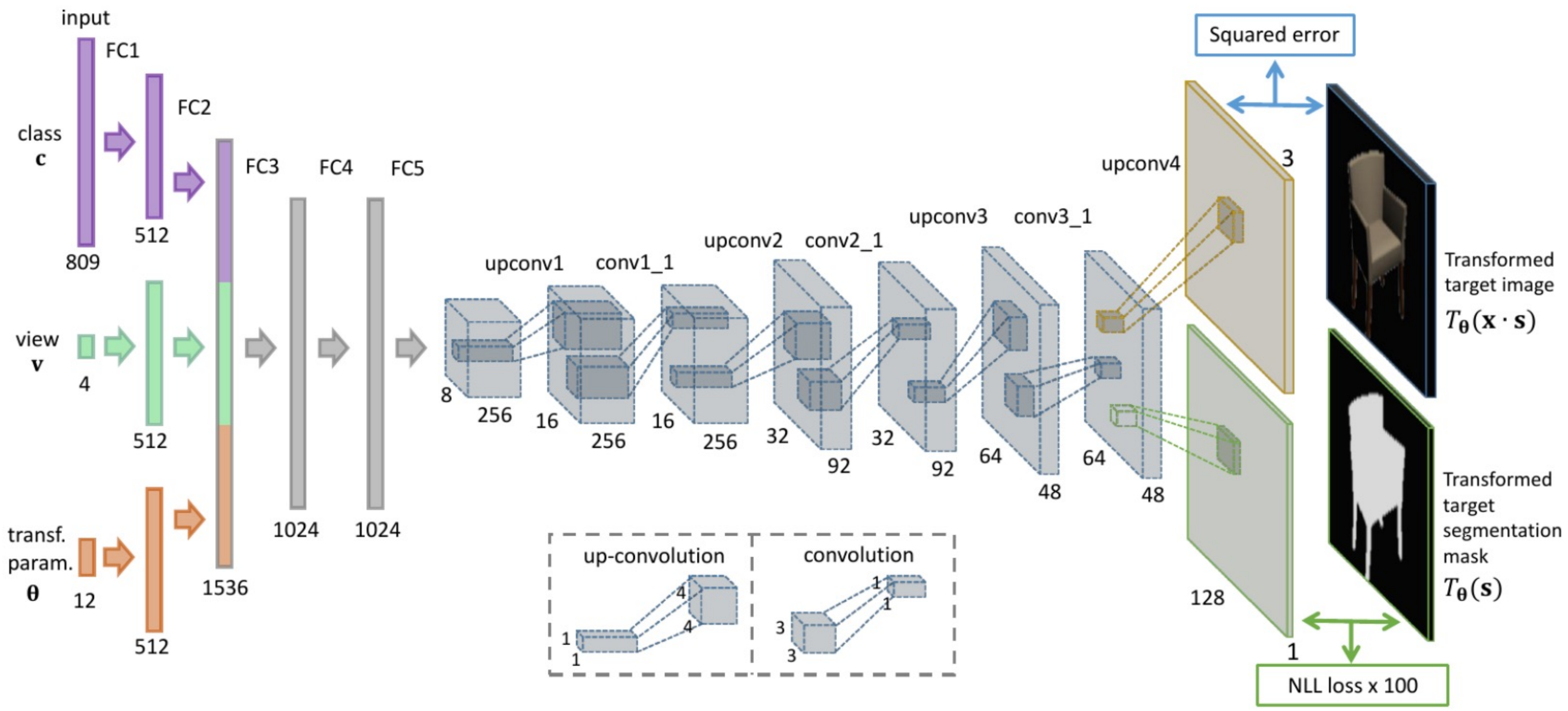
Refocusing

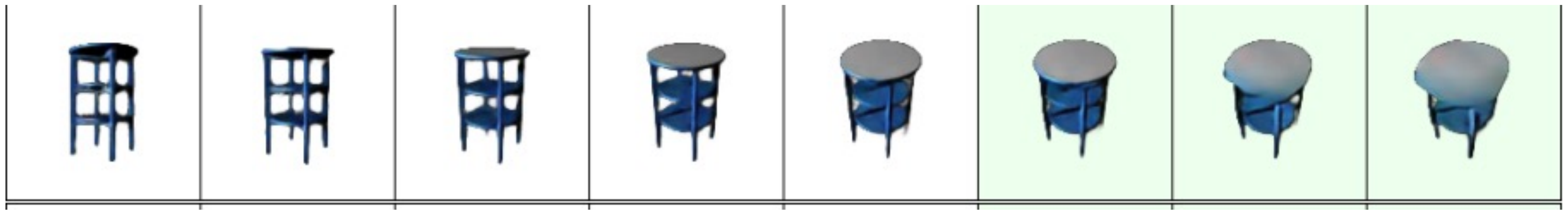
Novel View Synthesis

http://lightfield-forum.com/en/

Deep Learning for View Synthesis

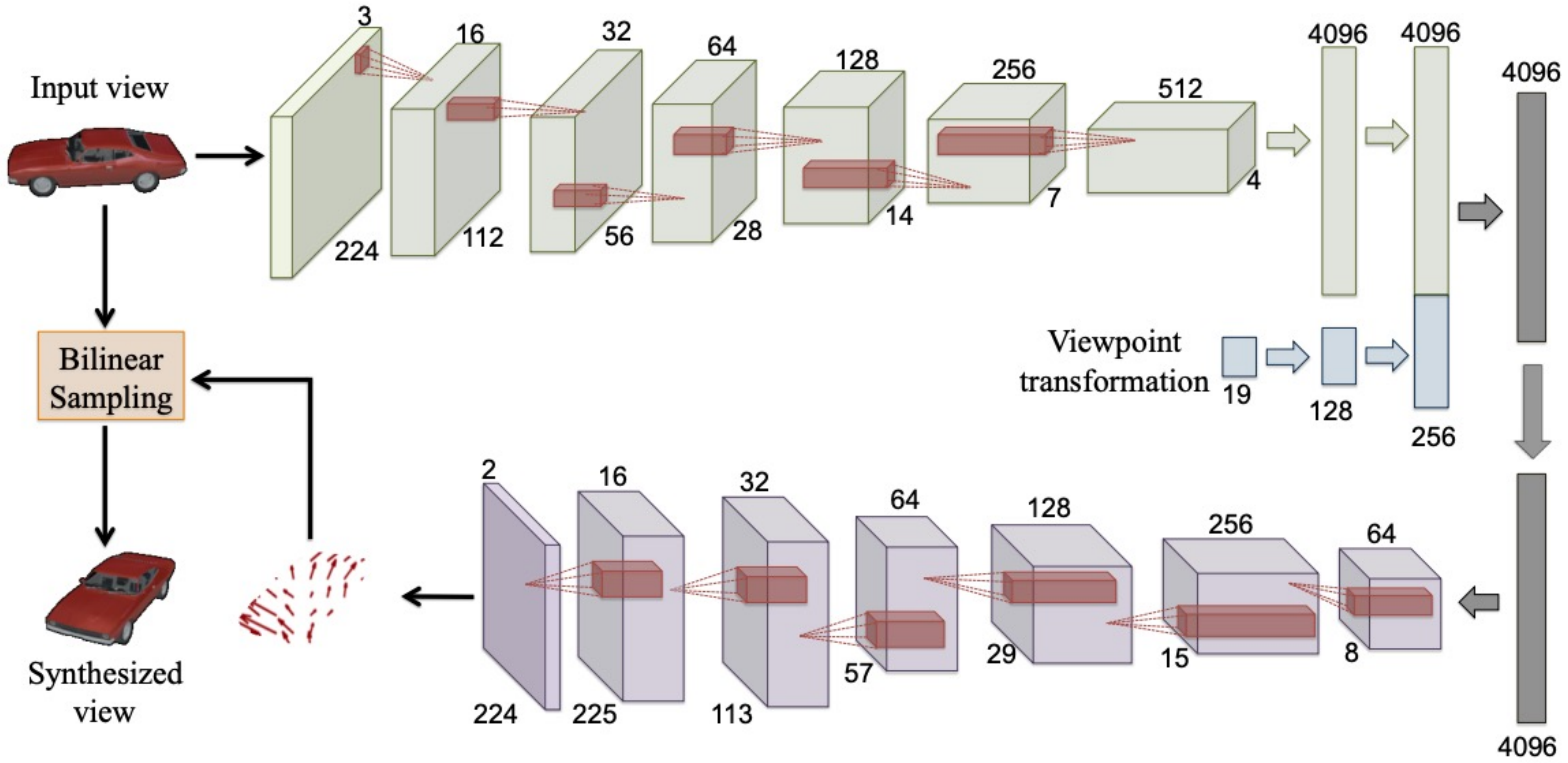
Generating Chairs with CNNs





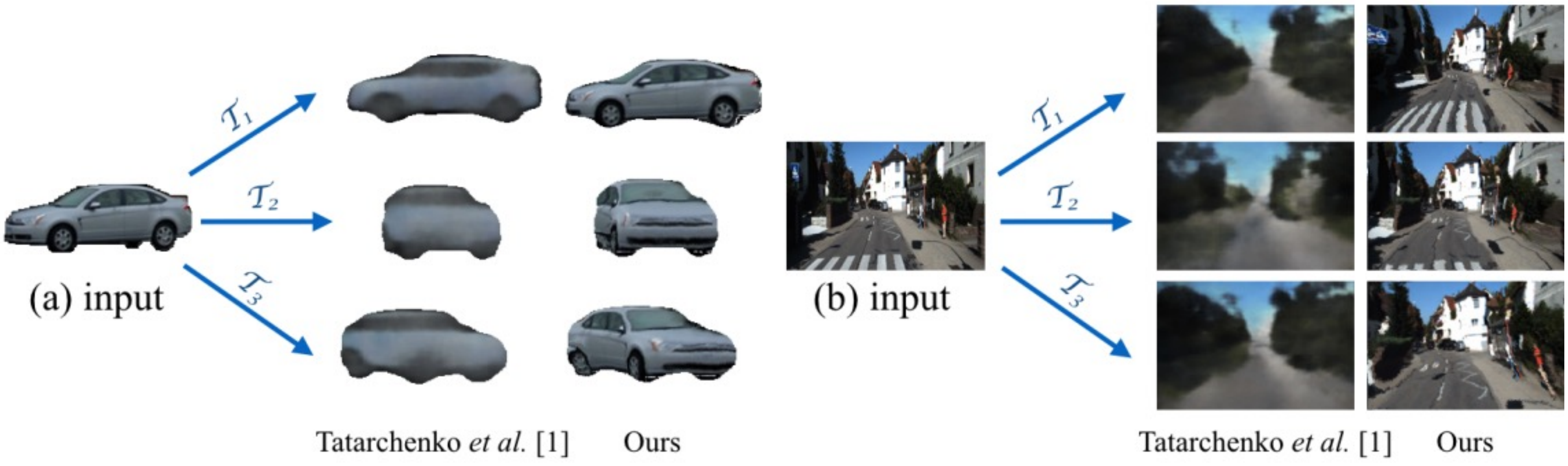
Dosovitskiy et al. Learning to Generate Chairs, Tables and Cars with Convolutional Networks
PAMI 2017 (CVPR 2015)

View Synthesis with Dense Correspondence



View Synthesis by Appearance Flow
Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, Alexei A. Efros
ECCV 2016

View Synthesis with Dense Correspondence



View Synthesis by Appearance Flow
Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, Alexei A. Efros
ECCV 2016

Lots of recent progress using deep learning for view synthesis!

Wiles CVPR 2020

Choi ICCV 2019

Flynn CVPR 2019

The following slides deck is from
Ben Mildenhall*, Pratul Srinivasan*, Matthew Tancik*, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng

The problem of novel view interpolation

Inputs: sparsely sampled images of scene

Outputs: *new* views of same scene

RGB-alpha volume rendering for view synthesis

Soft 3D
(Penner & Zhang 2017)
Culmination of non-deep stereo matching techniques

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

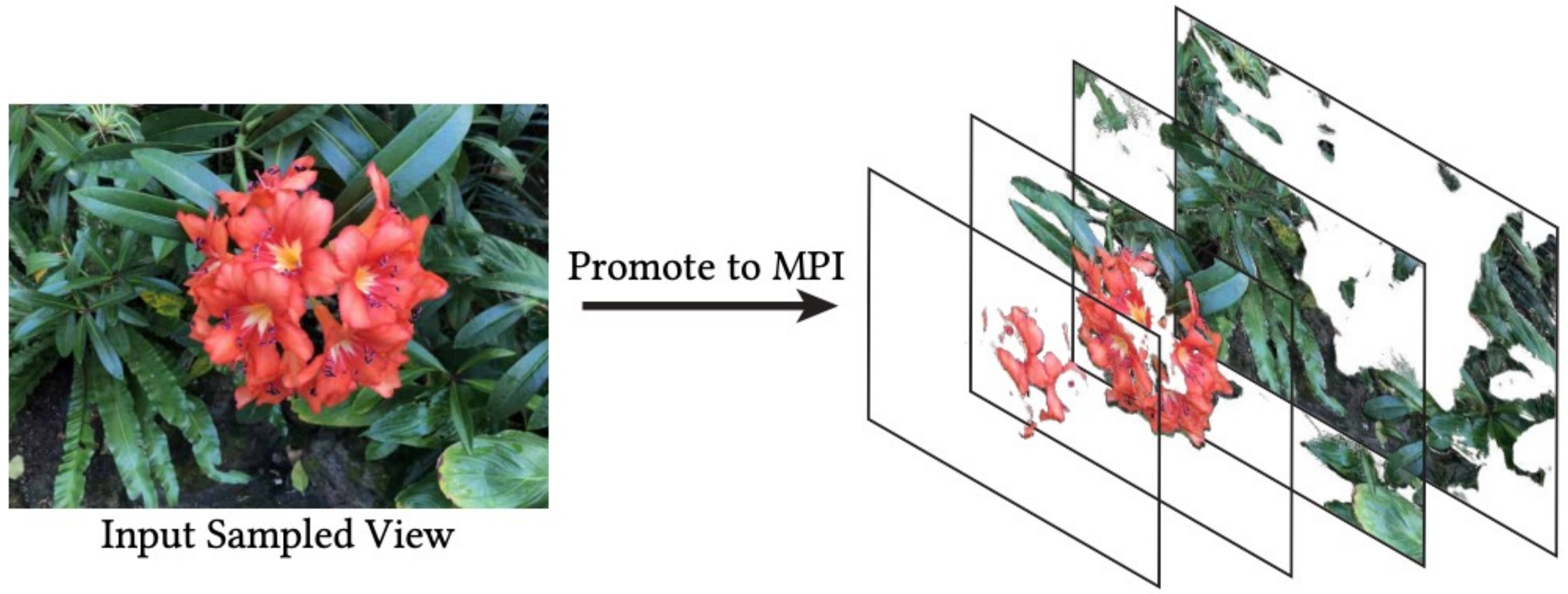
RGB-alpha volume rendering for view synthesis

Soft 3D
(Penner & Zhang 2017)
Culmination of non-deep stereo matching techniques

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

RGB-alpha volume rendering for view synthesis

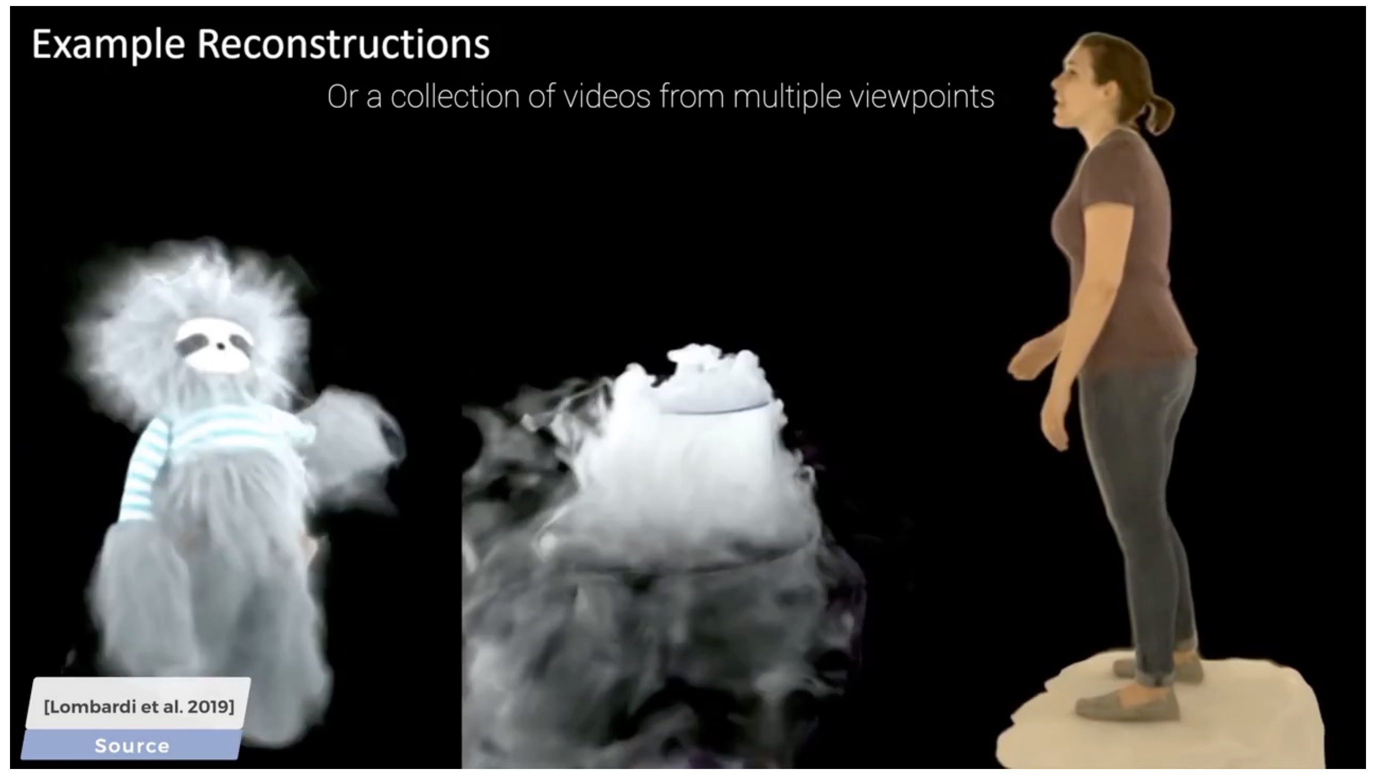
Soft 3D
(Penner & Zhang 2017)
Culmination of non-deep stereo matching techniques

Multiplane image methods

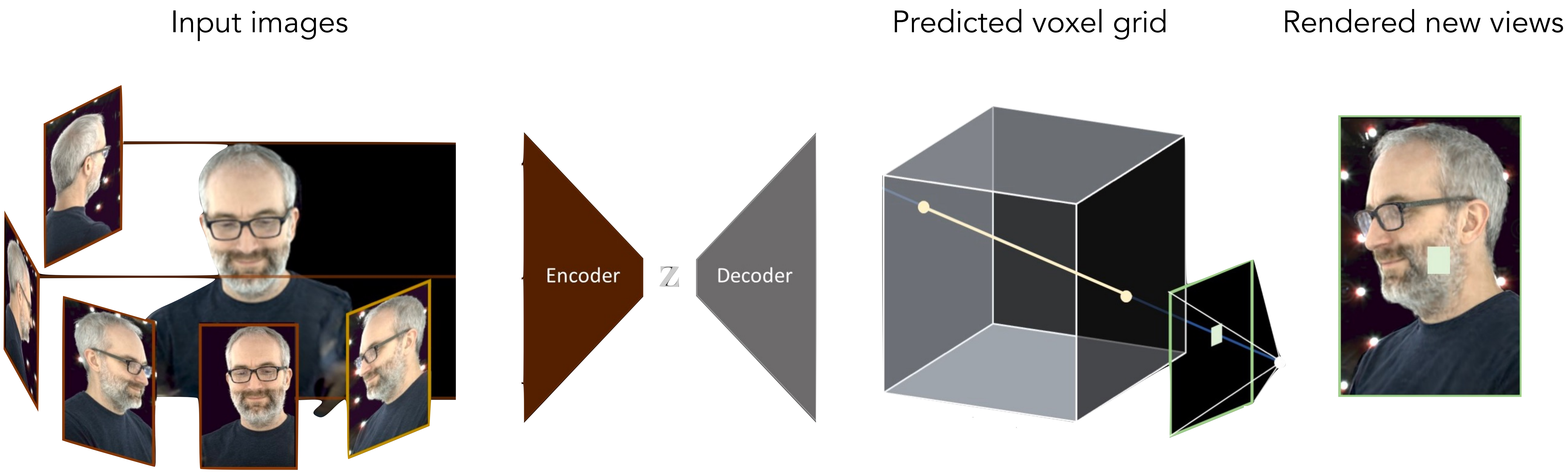
Stereo Magnification (Zhou et al. 2018)
Pushing the Boundaries… (Srinivasan et al. 2019)
Local Light Field Fusion (Mildenhall et al. 2019)
DeepView (Flynn et al. 2019)
Single-View… (Tucker & Snavely 2020)

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



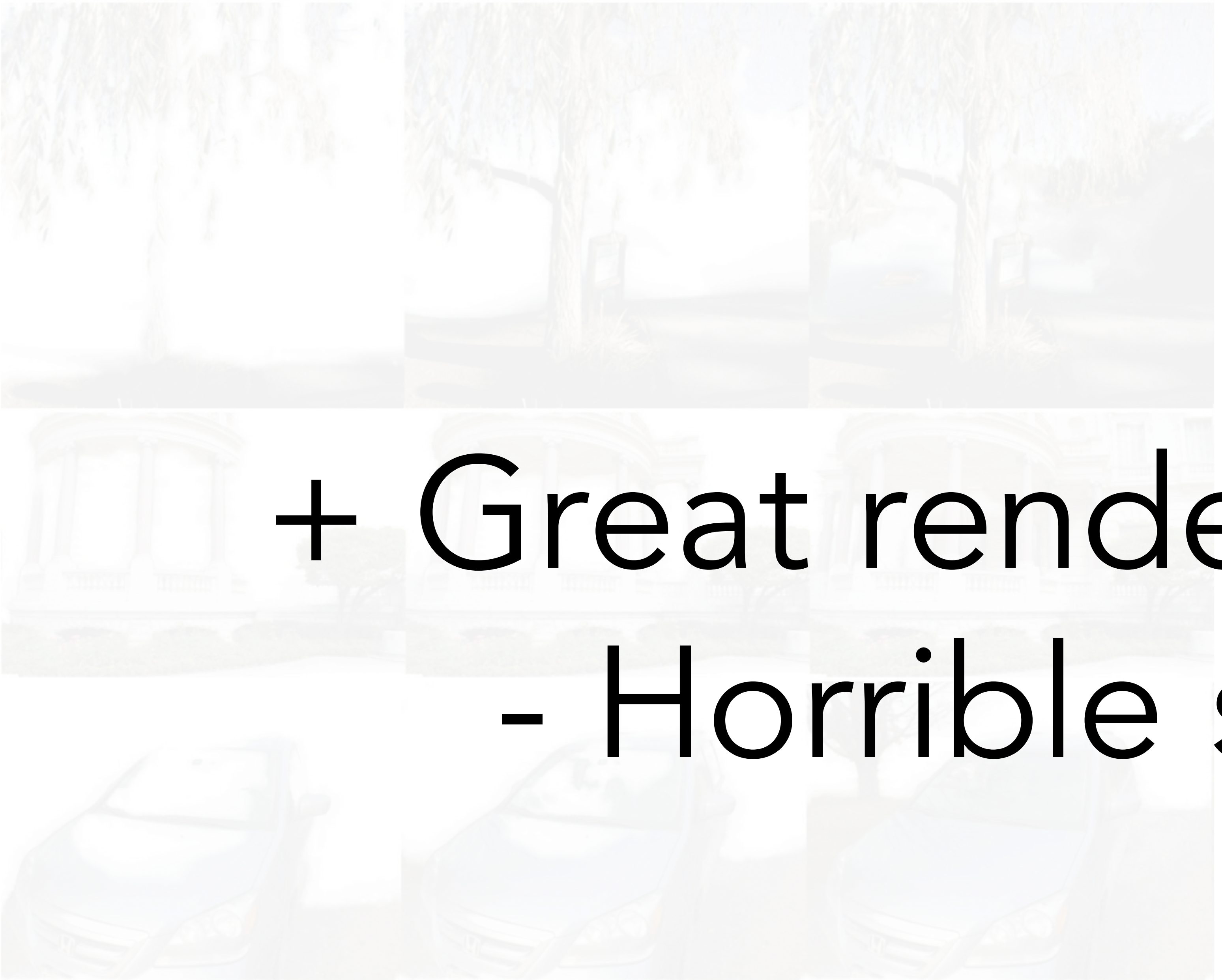
RGB-alpha volume rendering for view synthesis



Neural Volumes, Lombardi et al. 2019

RGB-alpha volume rendering for view synthesis

Soft 3D
(Penner & Zhang 2017)
Culmination of non-deep stereo matching techniques

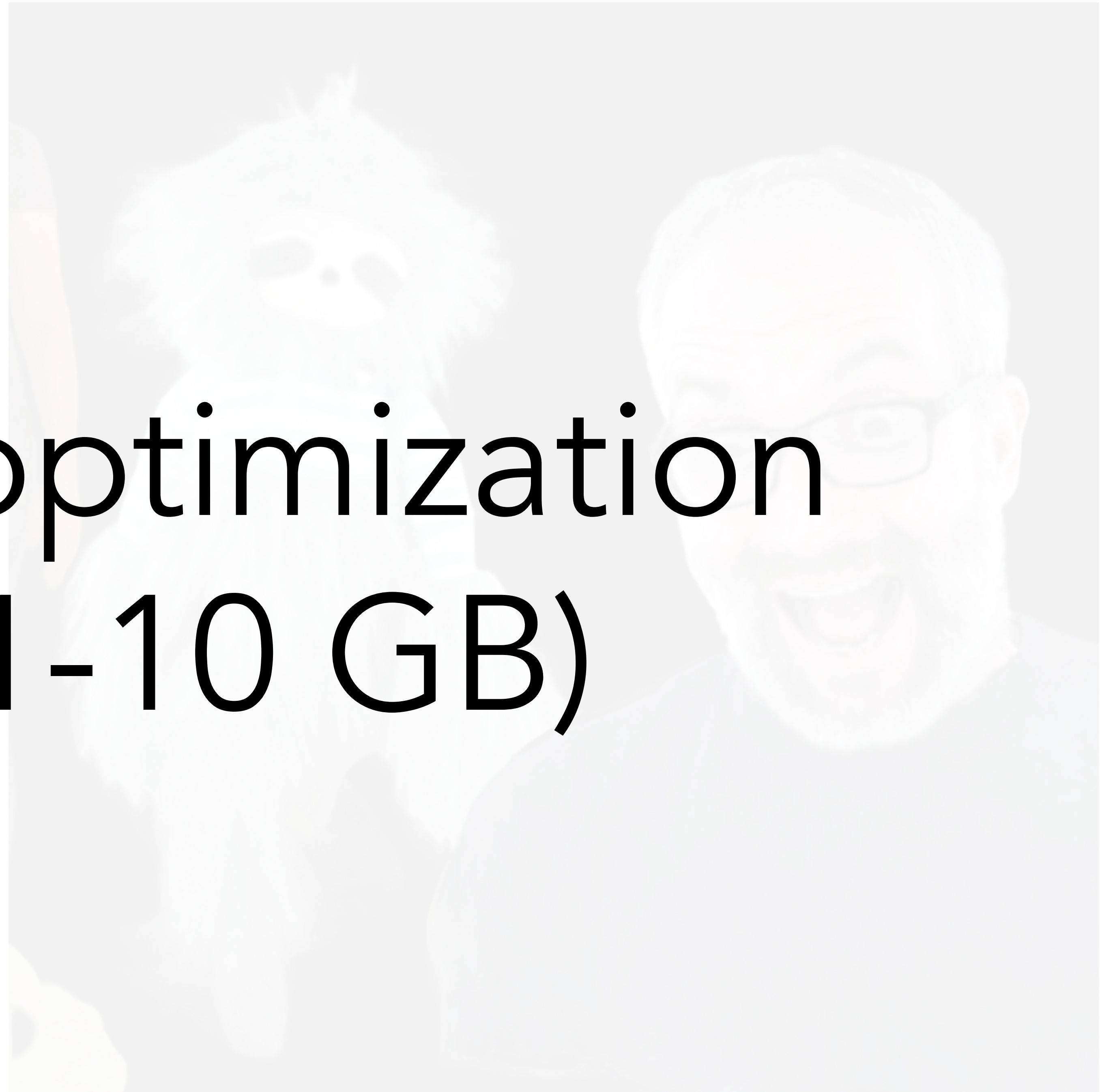


Multiplane image methods

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)

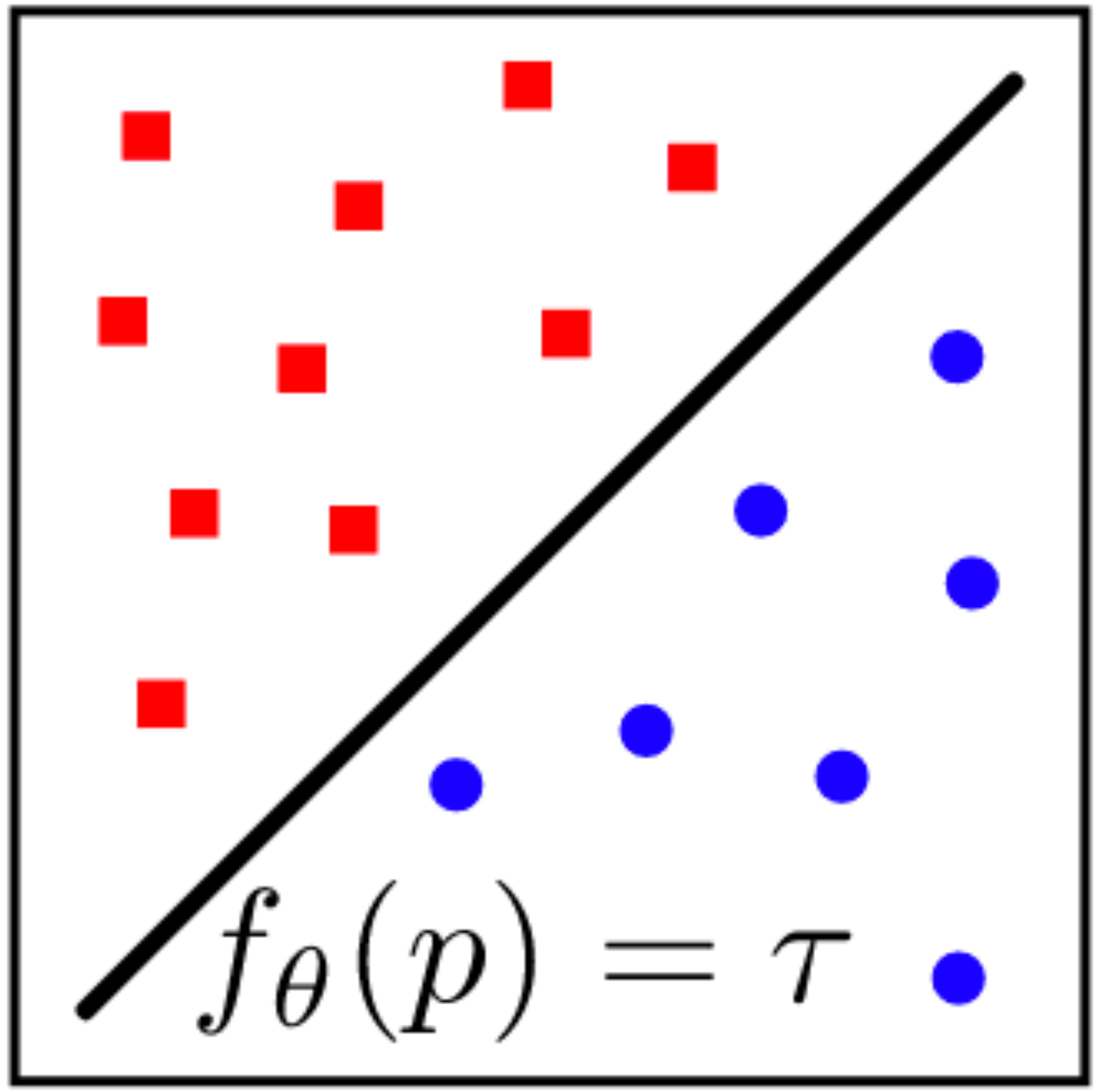
Typical deep learning pipelines - images go into a 3D CNN, big RGBA 3D volume comes out

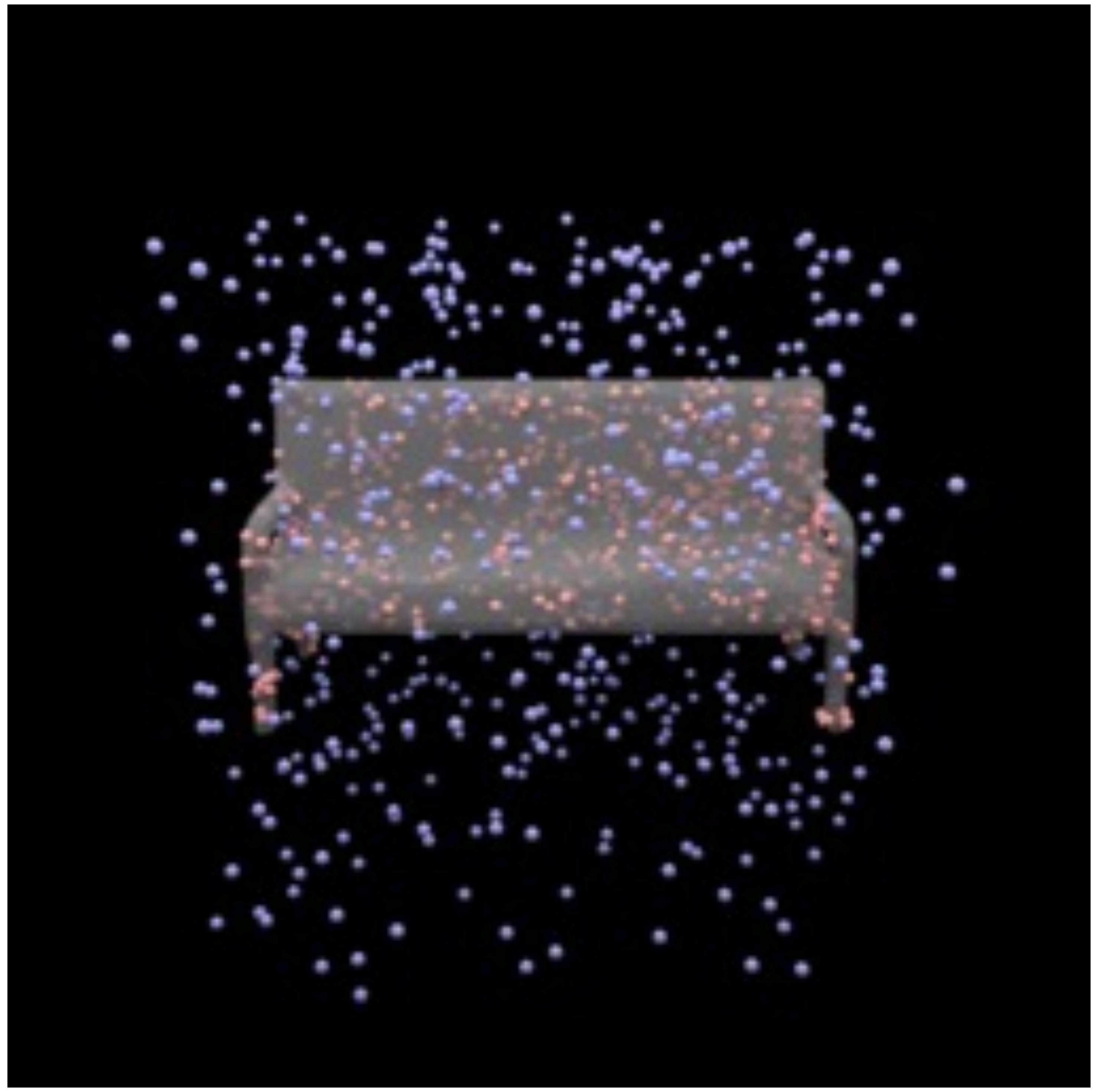
Neural Volumes
(Lombardi et al. 2019)
Direct gradient descent to optimize an RGBA volume, regularized by a 3D CNN



+ Great rendering model: good for optimization
- Horrible storage requirements (1-10 GB)

Neural networks as a continuous shape representation

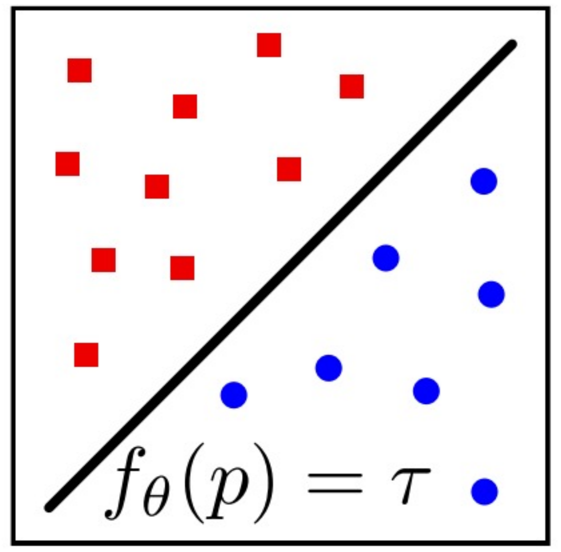




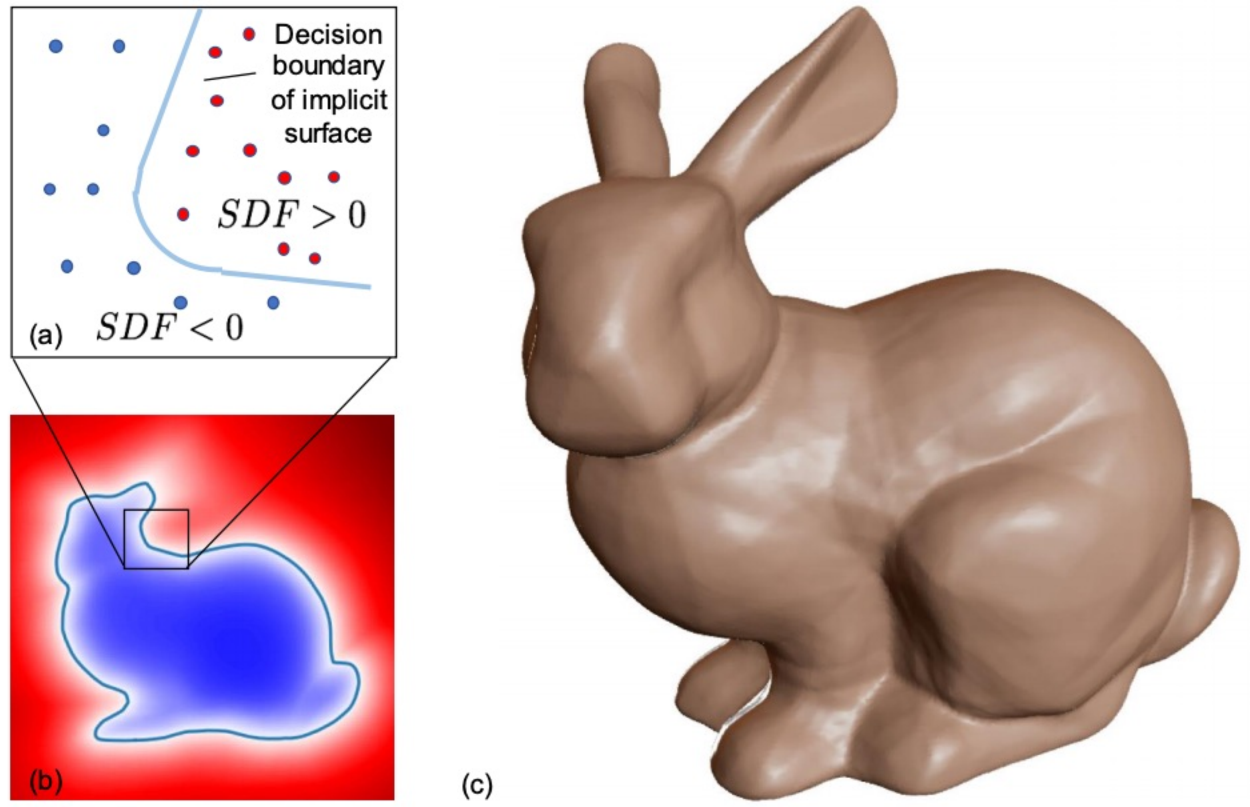
Occupancy Networks, Mescheder et al. CVPR 2019

Neural networks as a continuous shape representation

Occupancy Networks
(Mescheder et al. 2019)
$(x, y, z) \rightarrow occupancy$

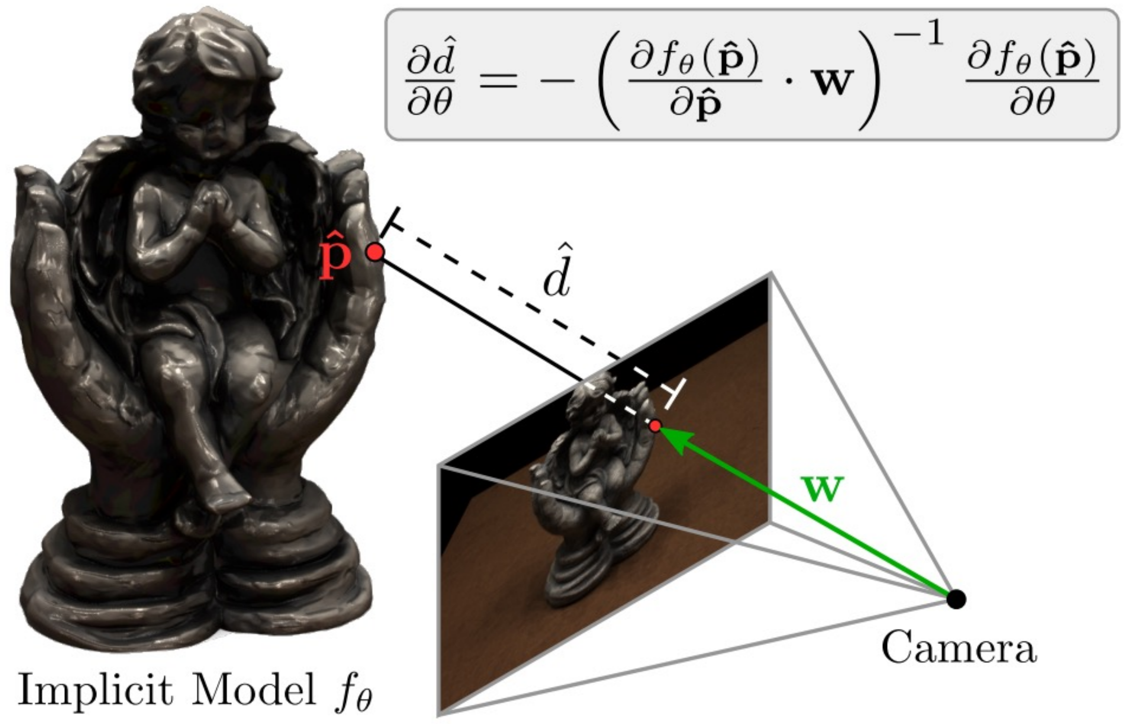


DeepSDF
(Park et al. 2019)
$(x, y, z) \rightarrow distance$



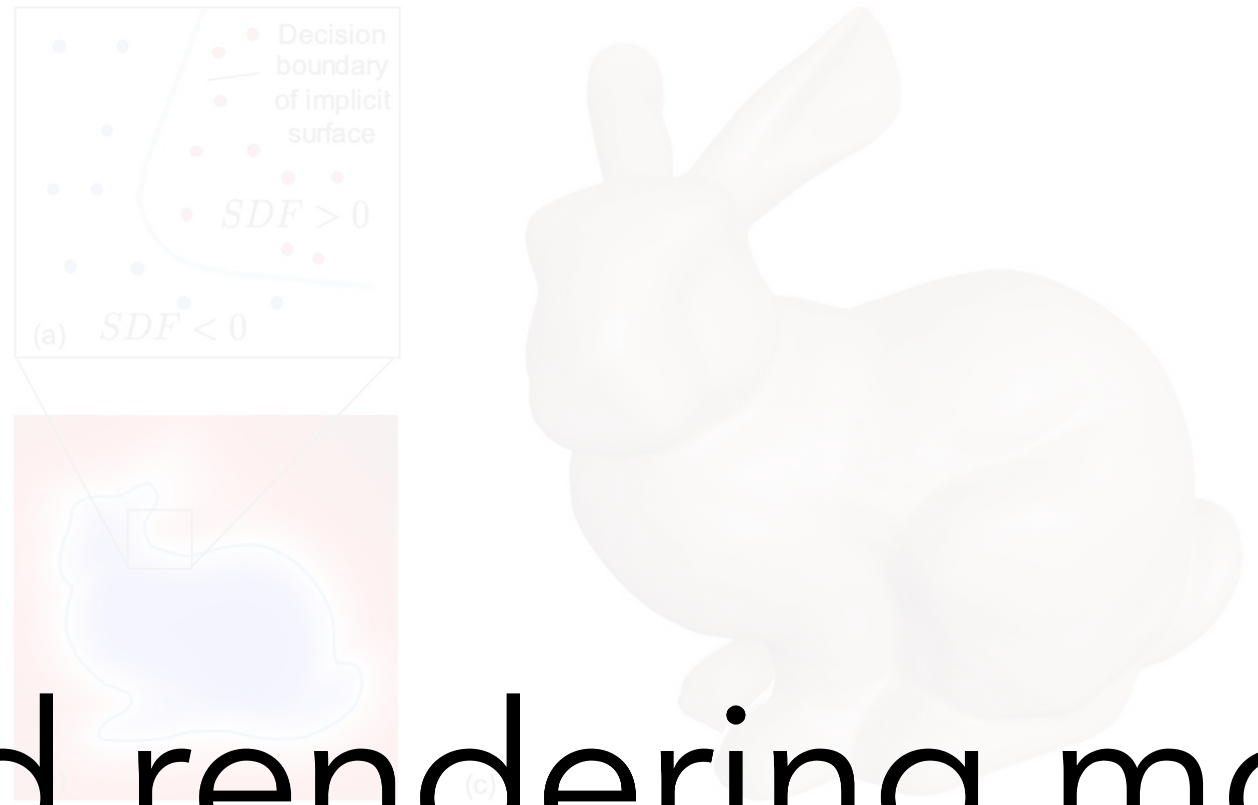
Scene Representation Networks
(Sitzmann et al. 2019)
$(x, y, z) \rightarrow latent\ vec.\ (color,\ dist.)$

Differentiable Volumetric Rendering
(Niemeyer et al. 2020)
$(x, y, z) \rightarrow color,\ occ.$



Neural networks as a shape representation

DeepSDF
(Park et al. 2019)
$(x, y, z) \rightarrow distance$



Occupancy Networks
(Mescheder et al. 2019)
$(x, y, z) \rightarrow occupancy$

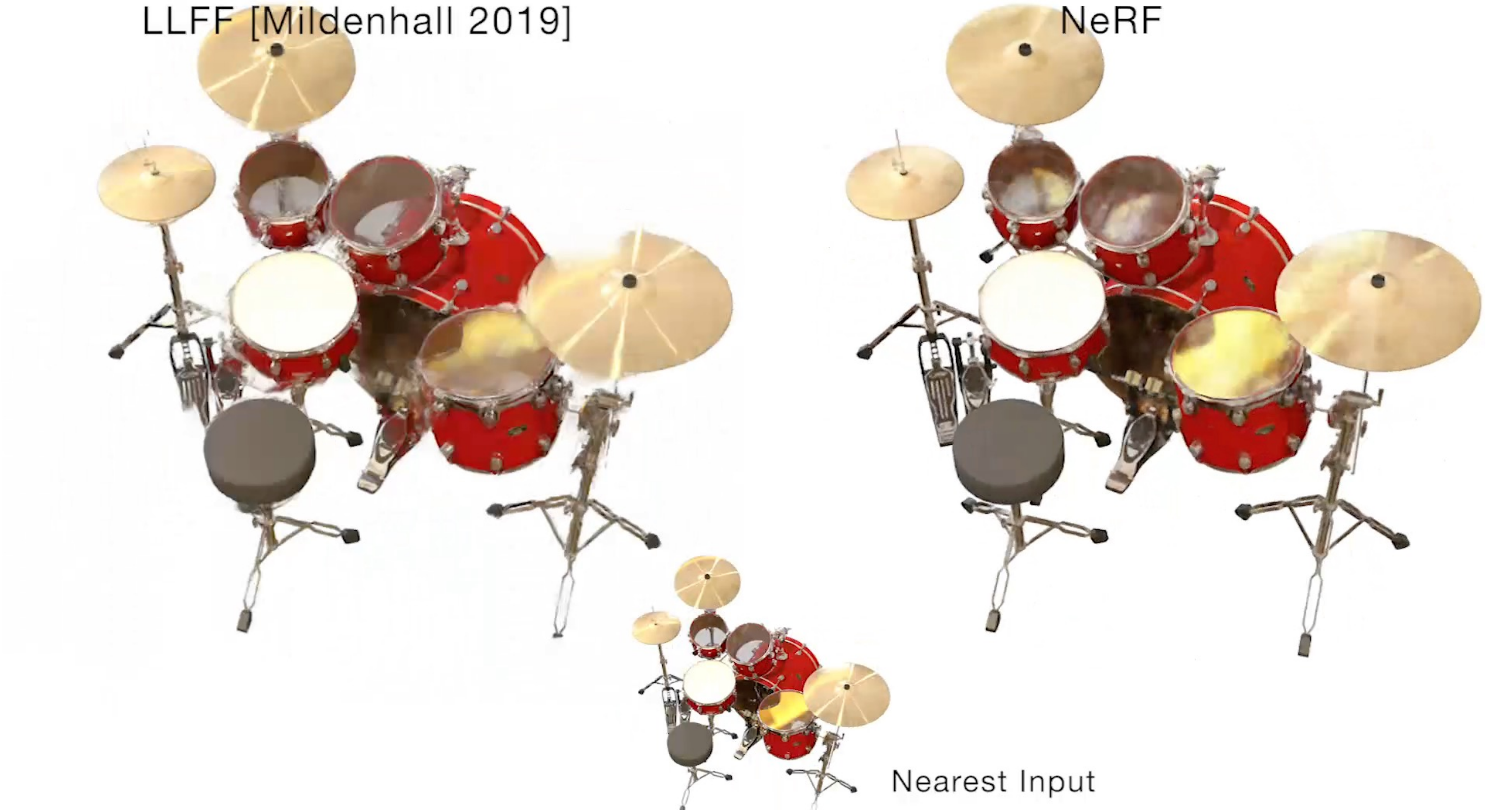
- Limited rendering model: difficult to optimize
+ Highly compressible (1-10 MB)

Scene Representation Networks
(Sitzmann et al. 2019)
$(x, y, z) \rightarrow latent\ vec.\ (color, dist.)$

Differentiable Volumetric Rendering
(Niemeyer et al. 2020)
$(x, y, z) \rightarrow color, occ.$

NeRF (neural radiance fields):
Neural networks as a *volume* representation, using volume rendering to do view synthesis.$(x, y, z, \theta, \phi) \rightarrow color, opacity$

NeRF achieves state-of-the-art results on an extremely difficult problem



NeRF achieves state-of-the-art results on an extremely difficult problem

NeRF achieves state-of-the-art results on an extremely difficult problem

Key points

- Continuous neural network as a volumetric scene representation (5D = xyz + direction)
- Use volume rendering model to synthesize new views
- Optimize using rendering loss for one scene (no prior training)
- One extra trick for passing coordinates into network to get high frequency details

Key points

- Continuous neural network as a volumetric scene representation (5D = xyz + direction)
- Use volume rendering model to synthesize new views
- Optimize using rendering loss for one scene (no prior training)
- One extra trick for passing coordinates into network to get high frequency details

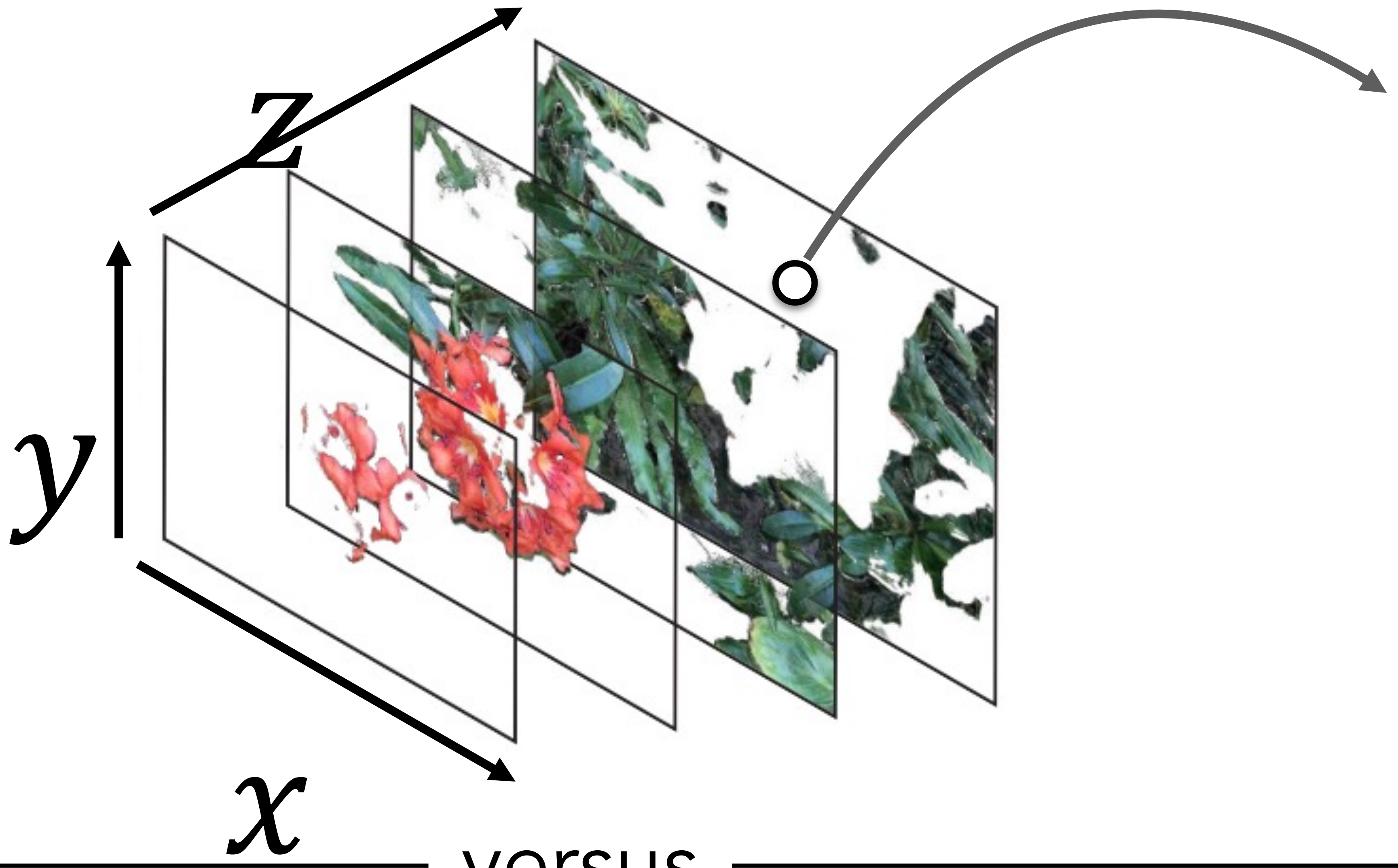
Representing a scene as a continuous 5D function

$$(\underbrace{x, y, z}_{\text{Spatial location}}, \underbrace{\theta, \phi}_{\text{Viewing direction}}) \longrightarrow F_{\Omega} \longrightarrow (\underbrace{r, g, b}_{\text{Output color}}, \underbrace{\sigma}_{\text{Output density}})$$

F_{Ω}

Fully-connected
neural network
9 layers,
256 channels

Neural network replaces large N-d array



$$(r, g, b, \alpha)$$

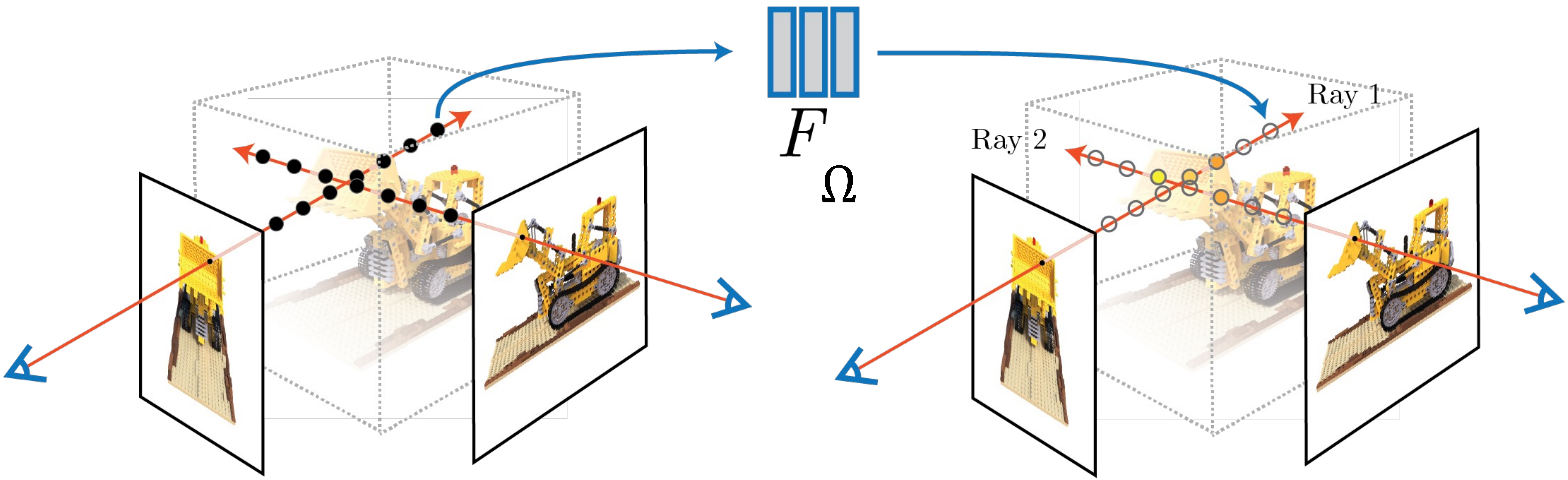
versus

$$(x, y, z, \theta, \phi) \rightarrow F_\Omega \rightarrow (r, g, b, \sigma)$$

Key points

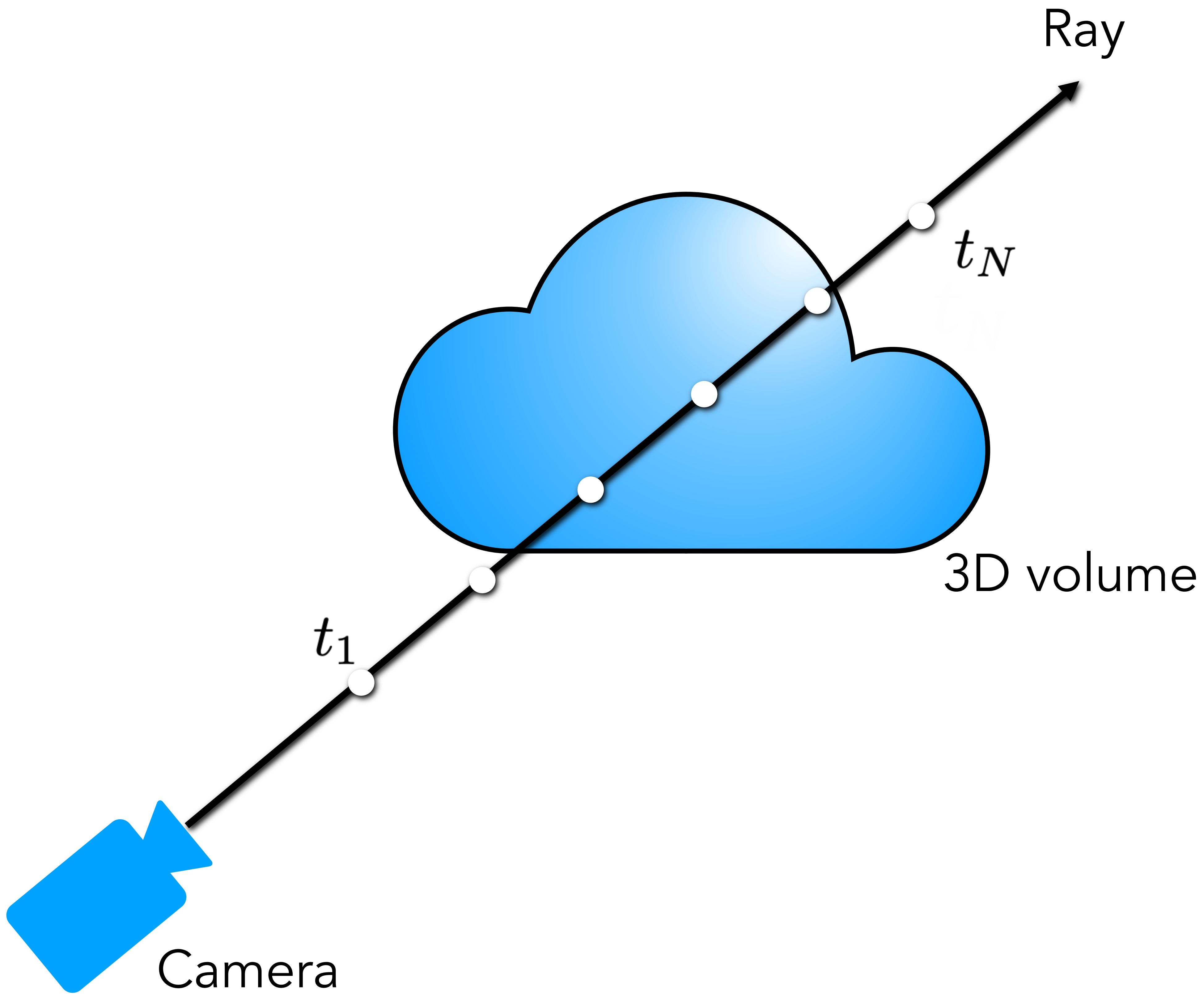
- Continuous neural network as a volumetric scene representation (5D = xyz + direction)
- Use volume rendering model to synthesize new views
- Optimize using rendering loss for one scene (no prior training)
- One extra trick for passing coordinates into network to get high frequency details

Generate views with traditional volume rendering



Generate views with traditional volume rendering

Rendering model for ray r(t) = o + td:



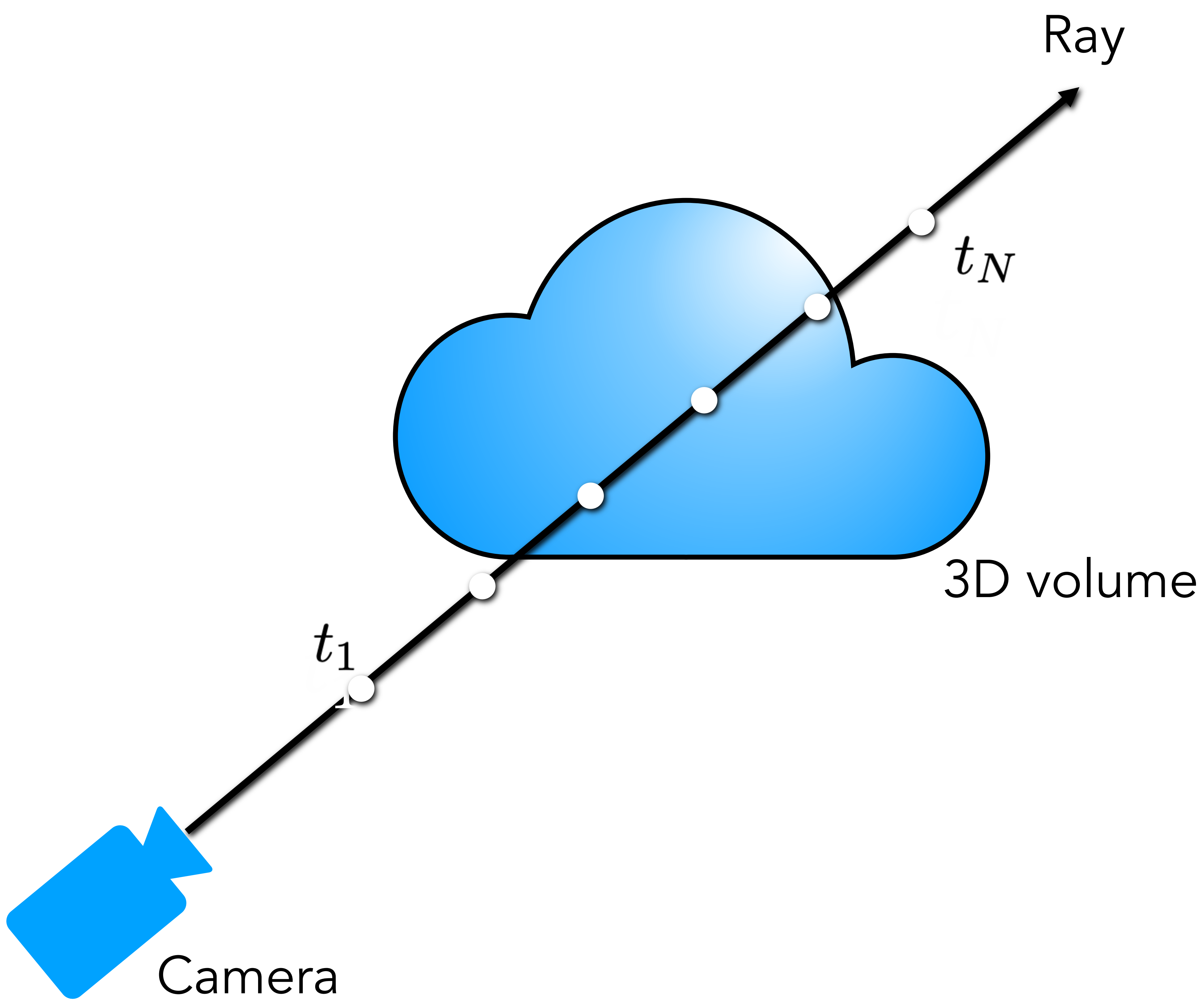
Generate views with traditional volume rendering

Rendering model for ray r(t) = o + td:

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

weights

colors



Generate views with traditional volume rendering

Rendering model for ray r(t) = o + td:

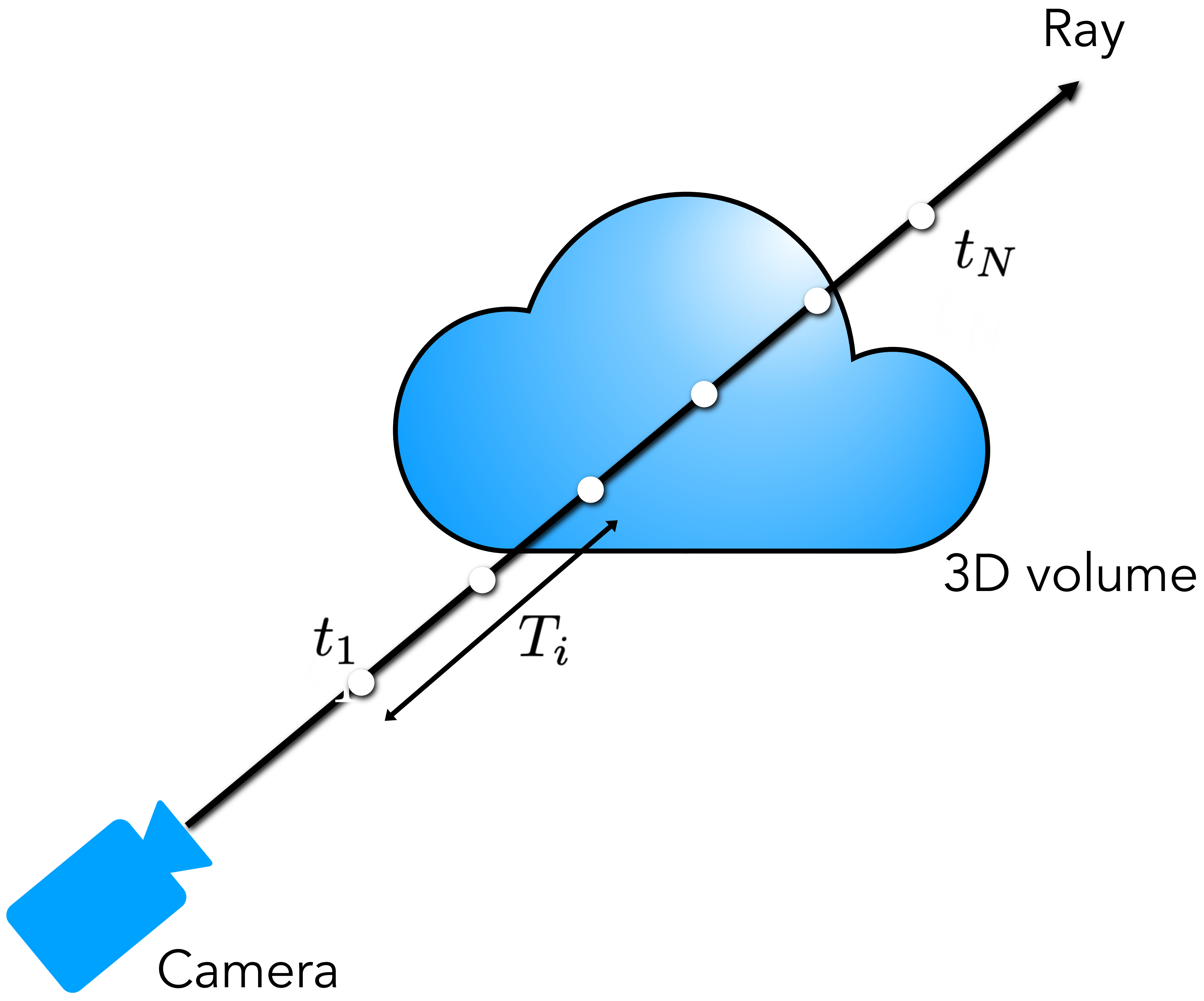
$$C \approx \sum_{i=1}^{N} T_i \alpha_i c_i$$

weights

colors

How much light is blocked earlier along ray:

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



Generate views with traditional volume rendering

Rendering model for ray r(t) = o + td:

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

weights

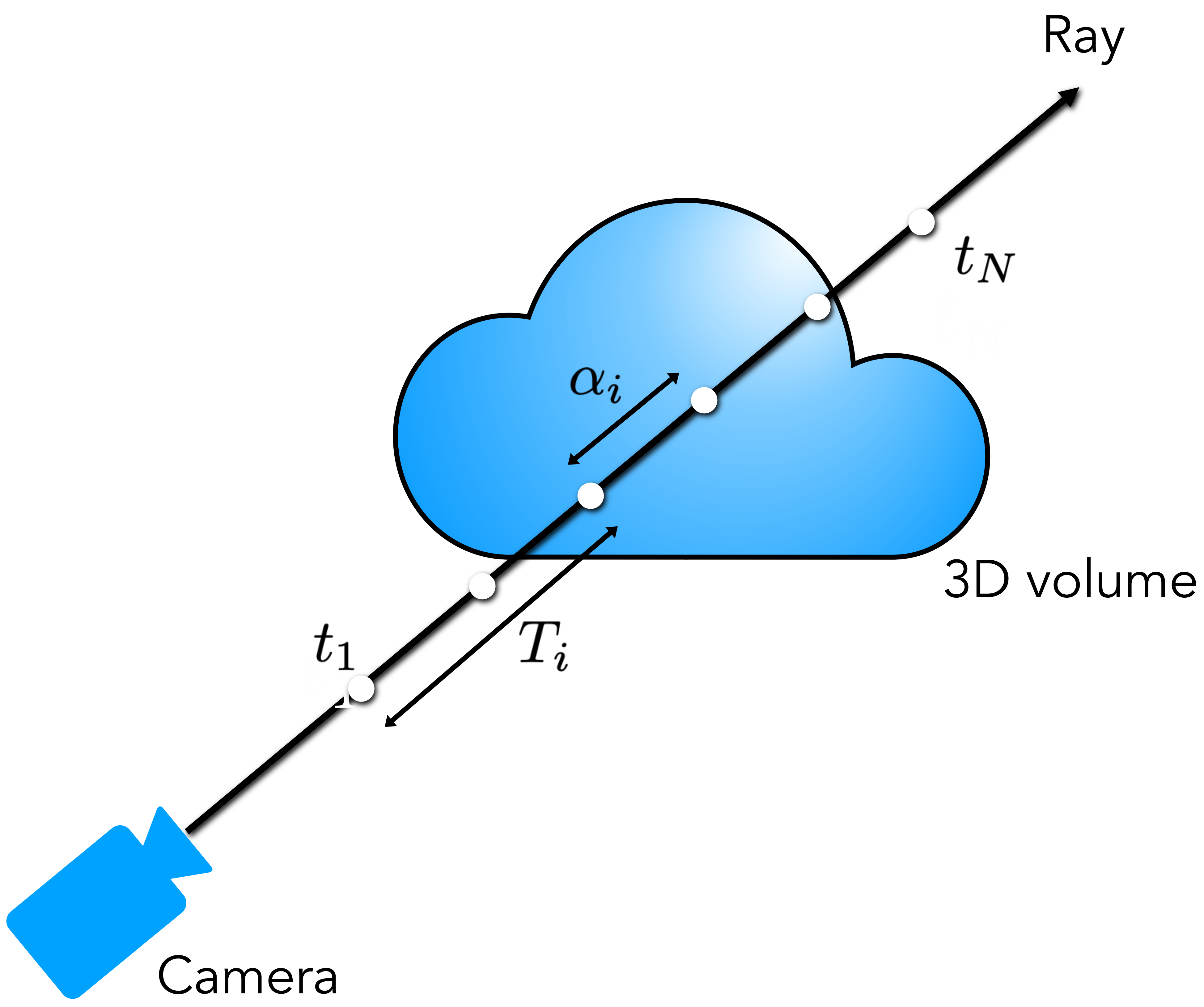
colors

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*:

$$\alpha_i = 1 - e^{-\sigma_i \delta t_i}$$



Sigma parametrization for continuous opacity

Rendering model for ray r(t) = o + td:

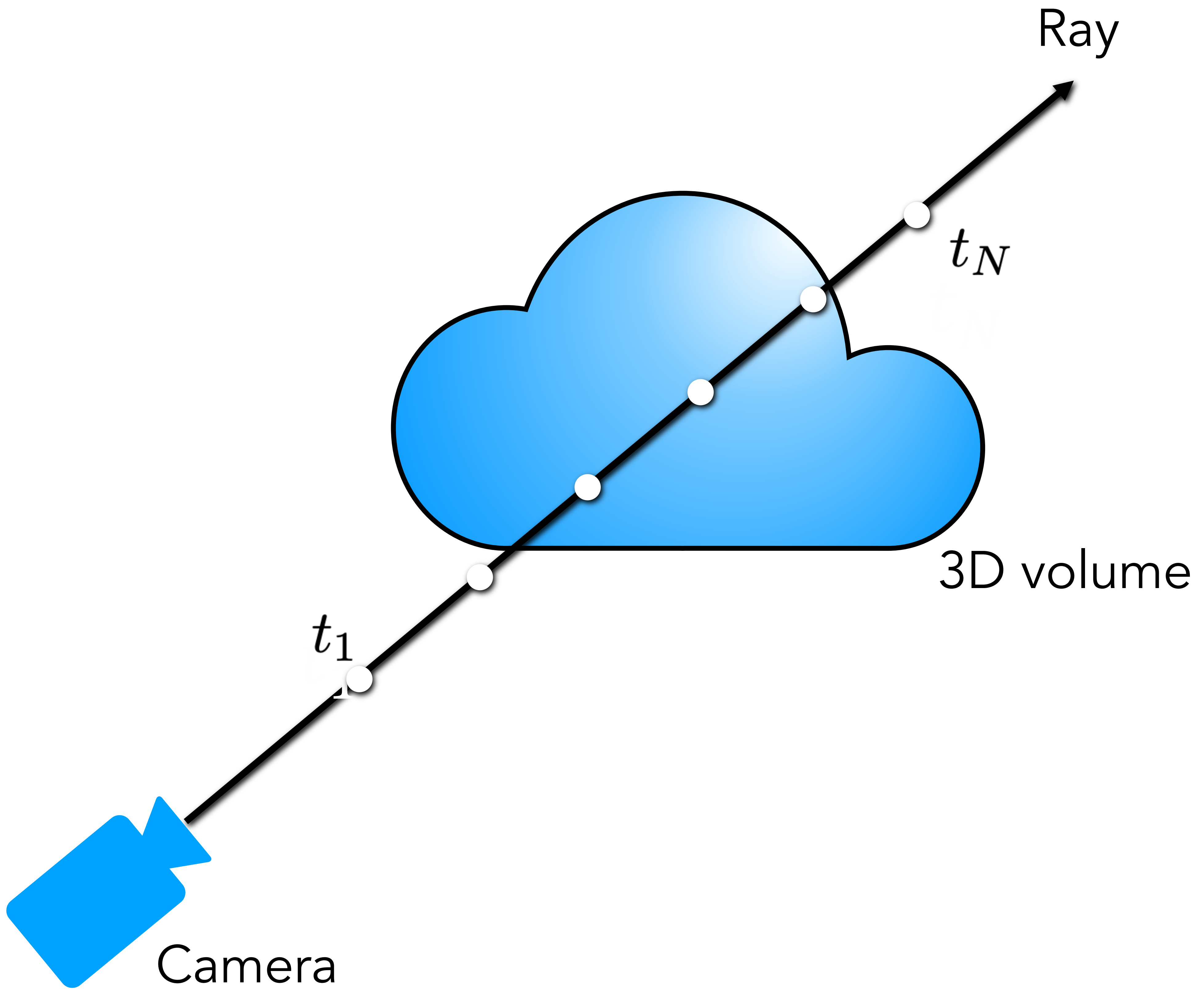
$$C \approx \sum_{i=1}^{N} \underbrace{T_i \alpha_i}_{\text{weights}} \underbrace{c_i}_{\text{colors}}$$

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*:

$$\boxed{\alpha_i = 1 - e^{-\sigma_i \delta t_i}}$$



Effective resolution is tied to distance between samples

Rendering model for ray r(t) = o + td:

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

weights

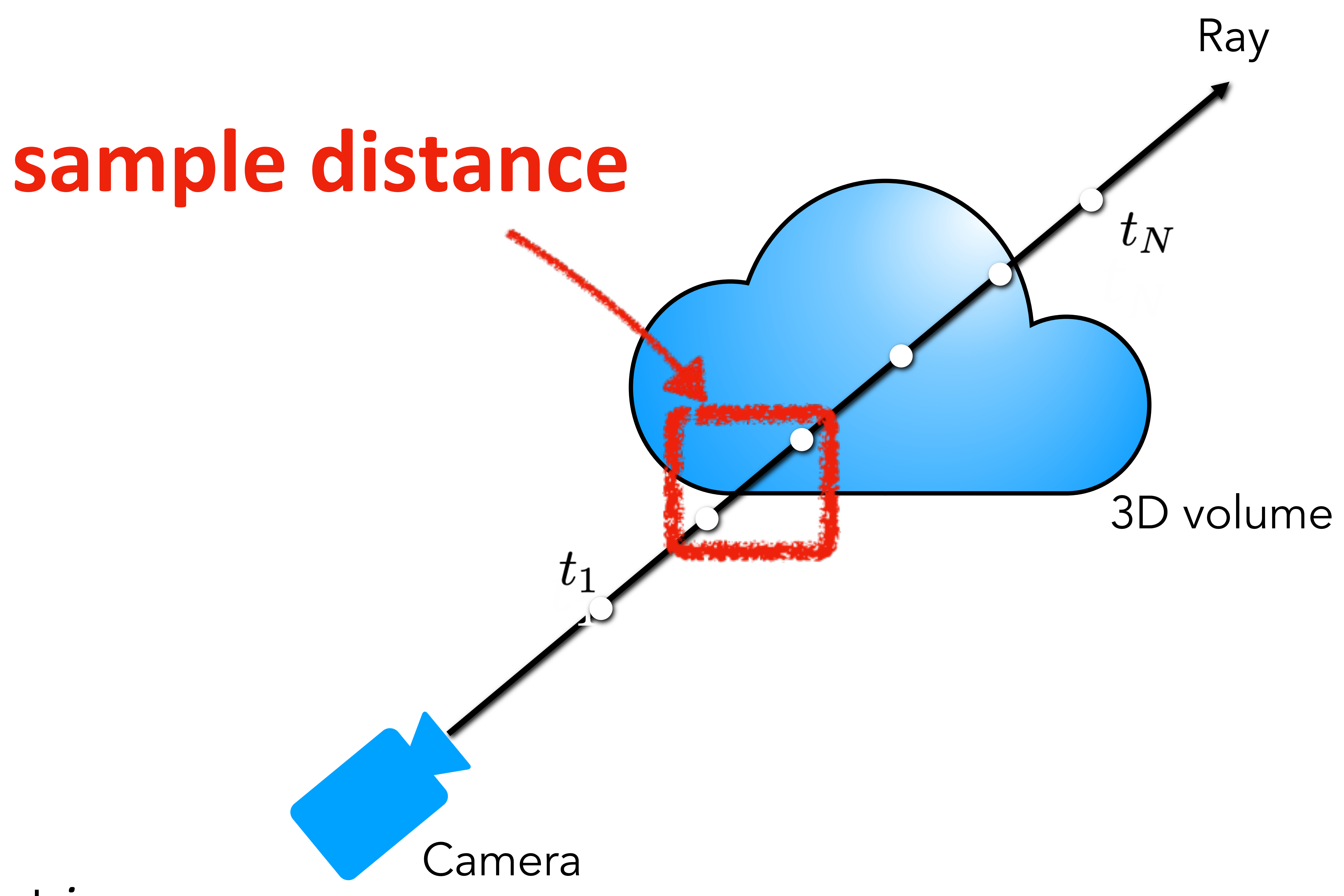
colors

How much light is blocked earlier along ray:

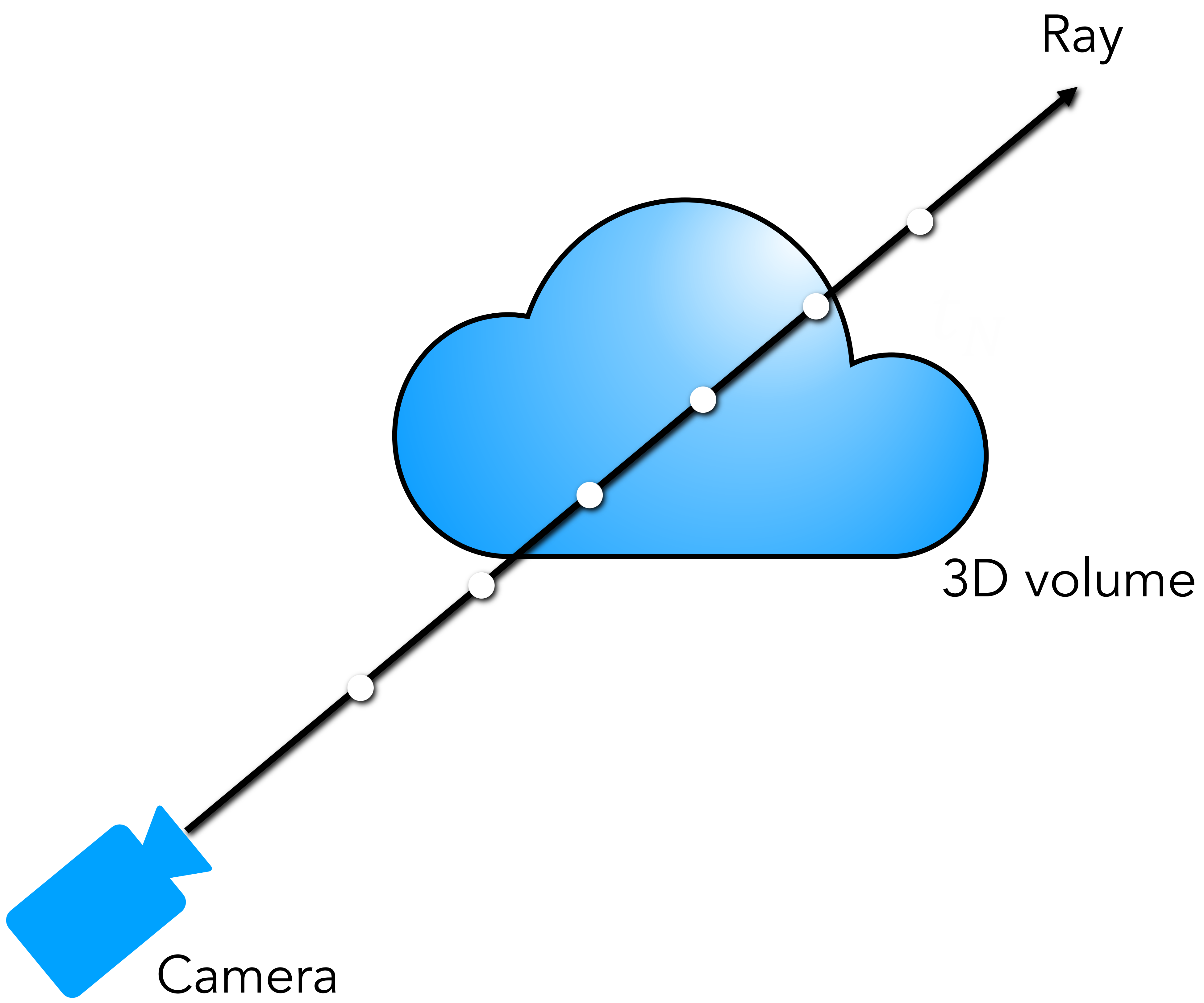
$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

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

$$\alpha_i = 1 - e^{-\sigma_i \delta t_i}$$



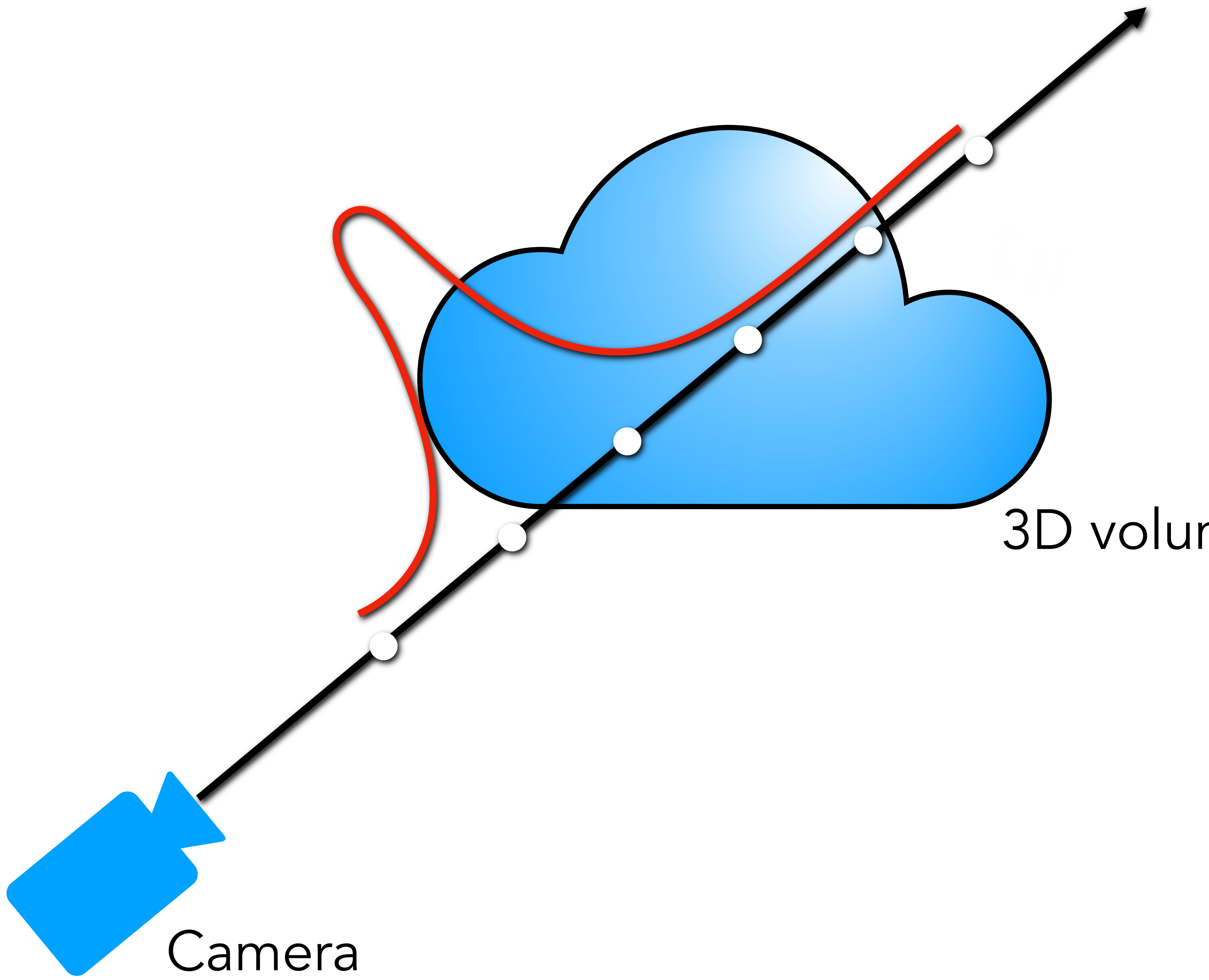
Can we allocate samples more efficiently? Two pass rendering



Two pass rendering: coarse

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

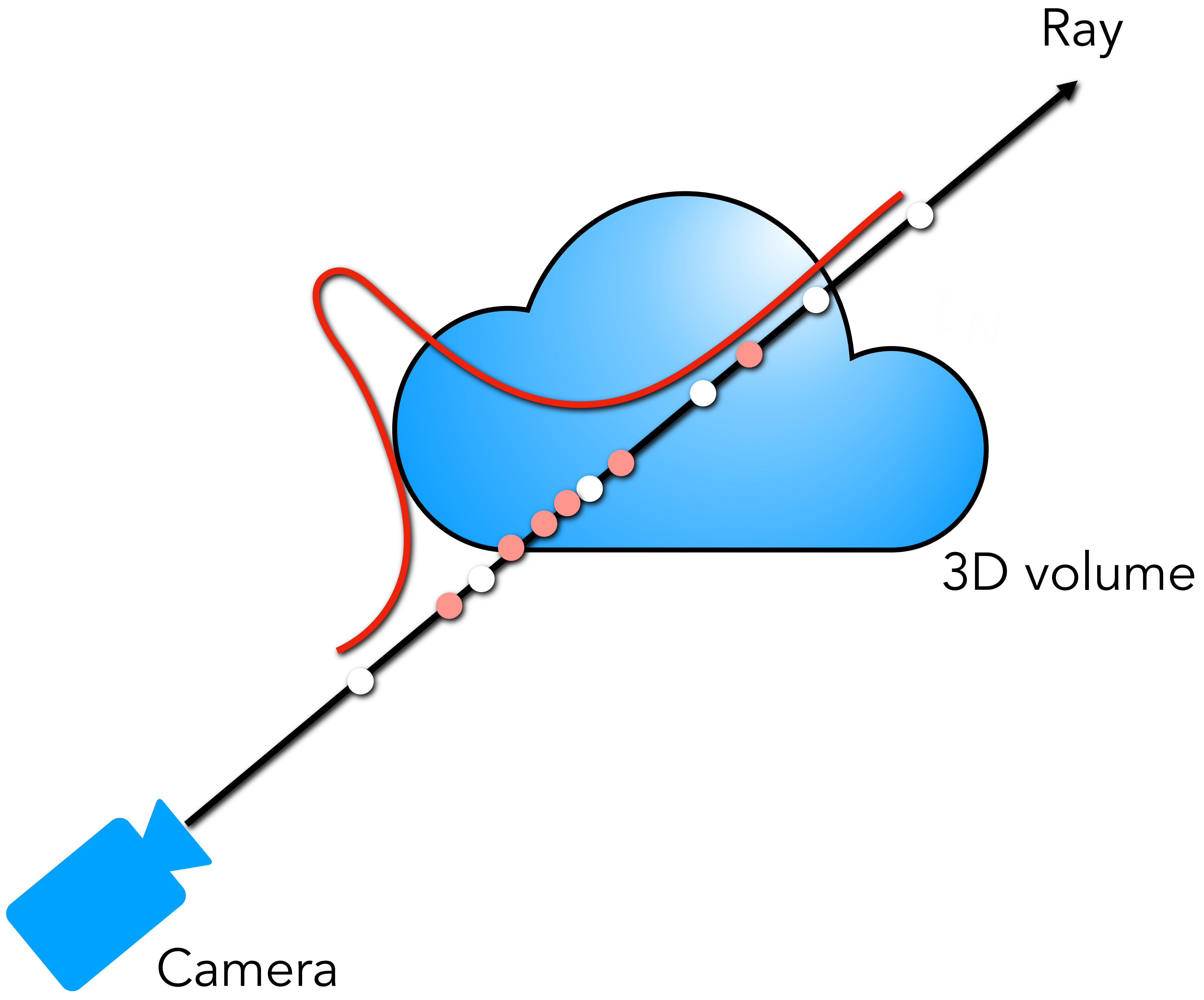
treat weights as probability distribution for new samples



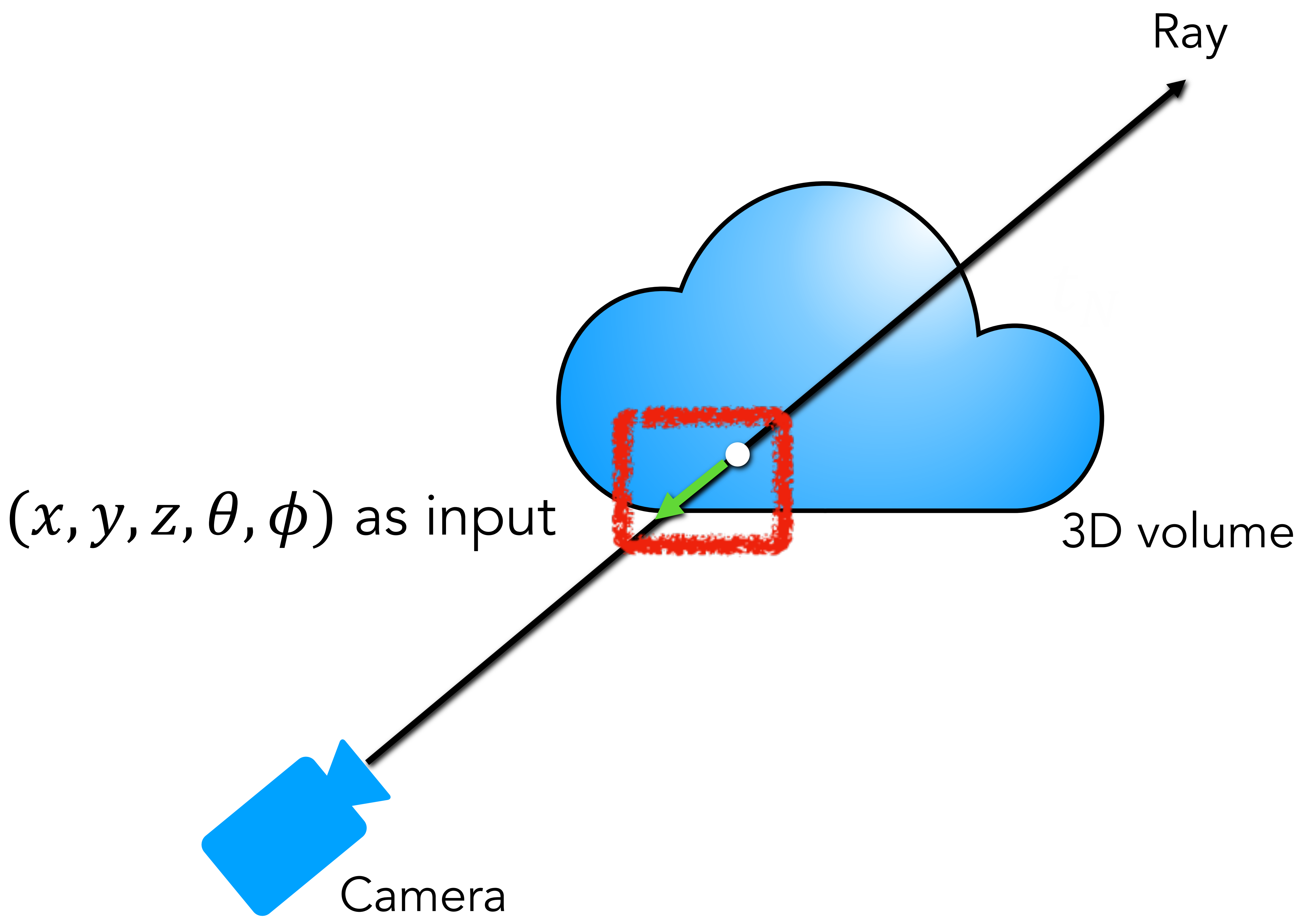
Two pass rendering: fine

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

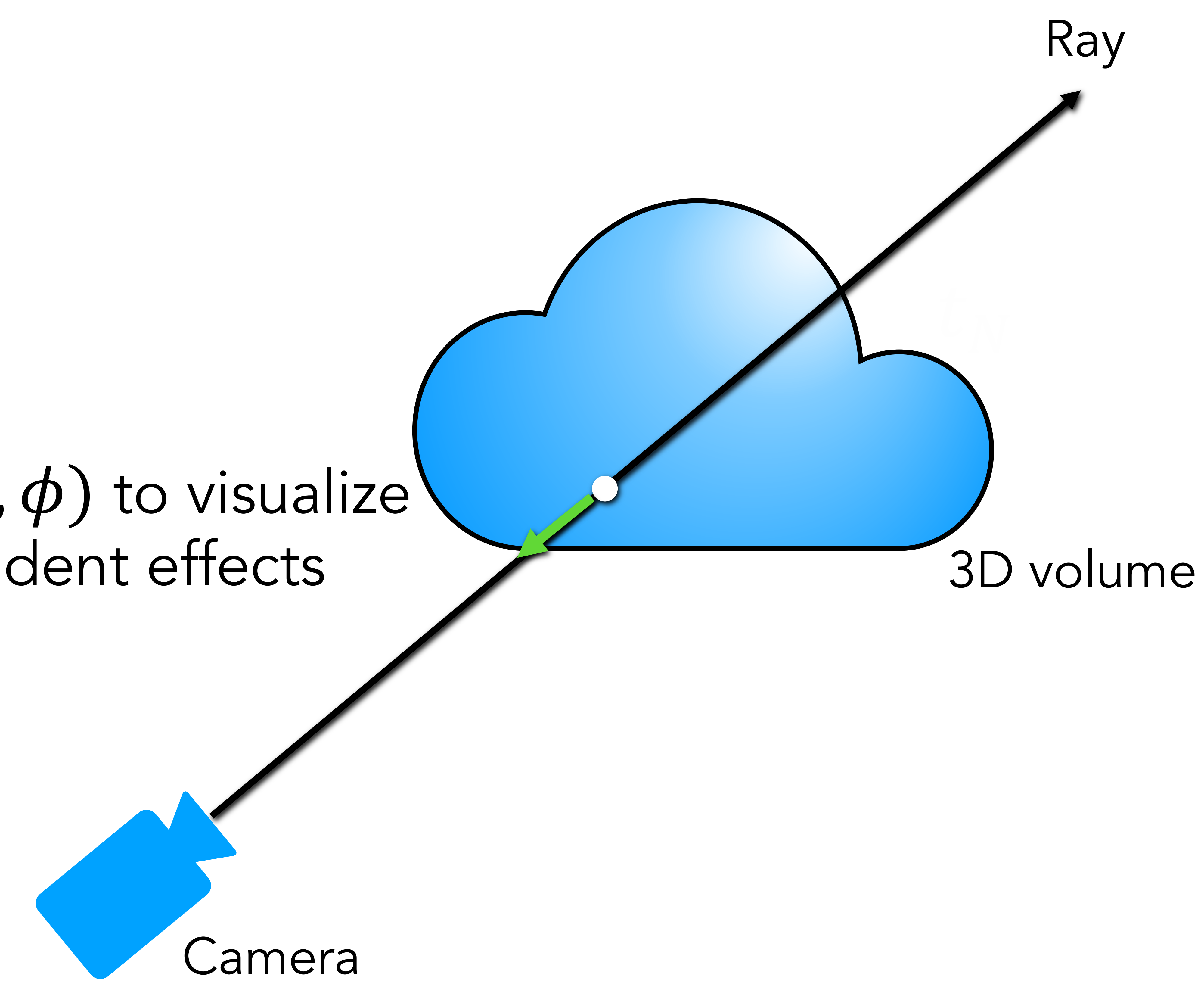
treat weights as probability distribution for new samples



Viewing directions as input

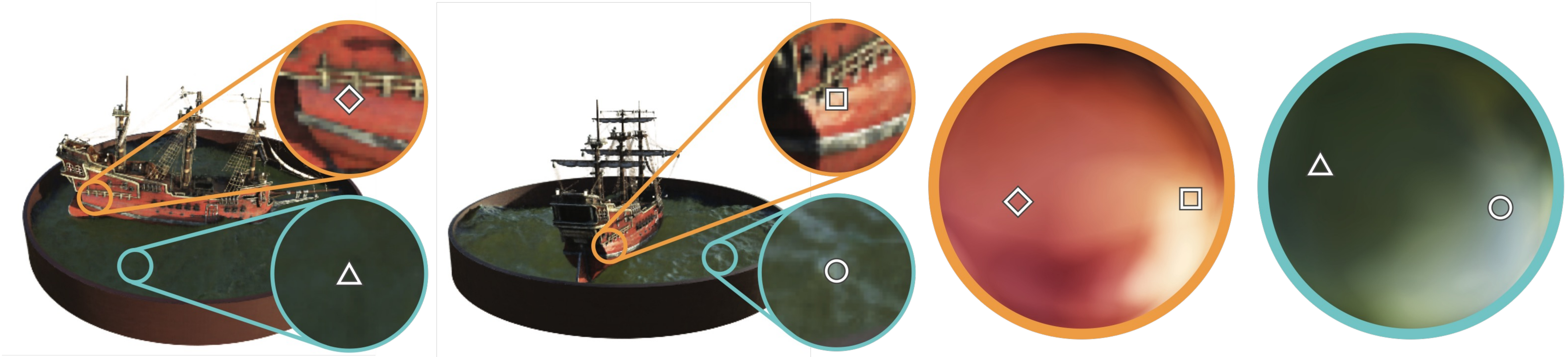


Viewing directions as input



Manipulate (θ, ϕ) to visualize view-dependent effects

Viewing directions as input



(a) View 1 (b) View 2 (c) Radiance Distributions

Key points

- Continuous neural network as a volumetric scene representation (5D = xyz + direction)
- Use volume rendering model to synthesize new views
- Optimize using rendering loss for one scene (no prior training)
- One extra trick for passing coordinates into network to get high frequency details

Volume rendering is trivially differentiable

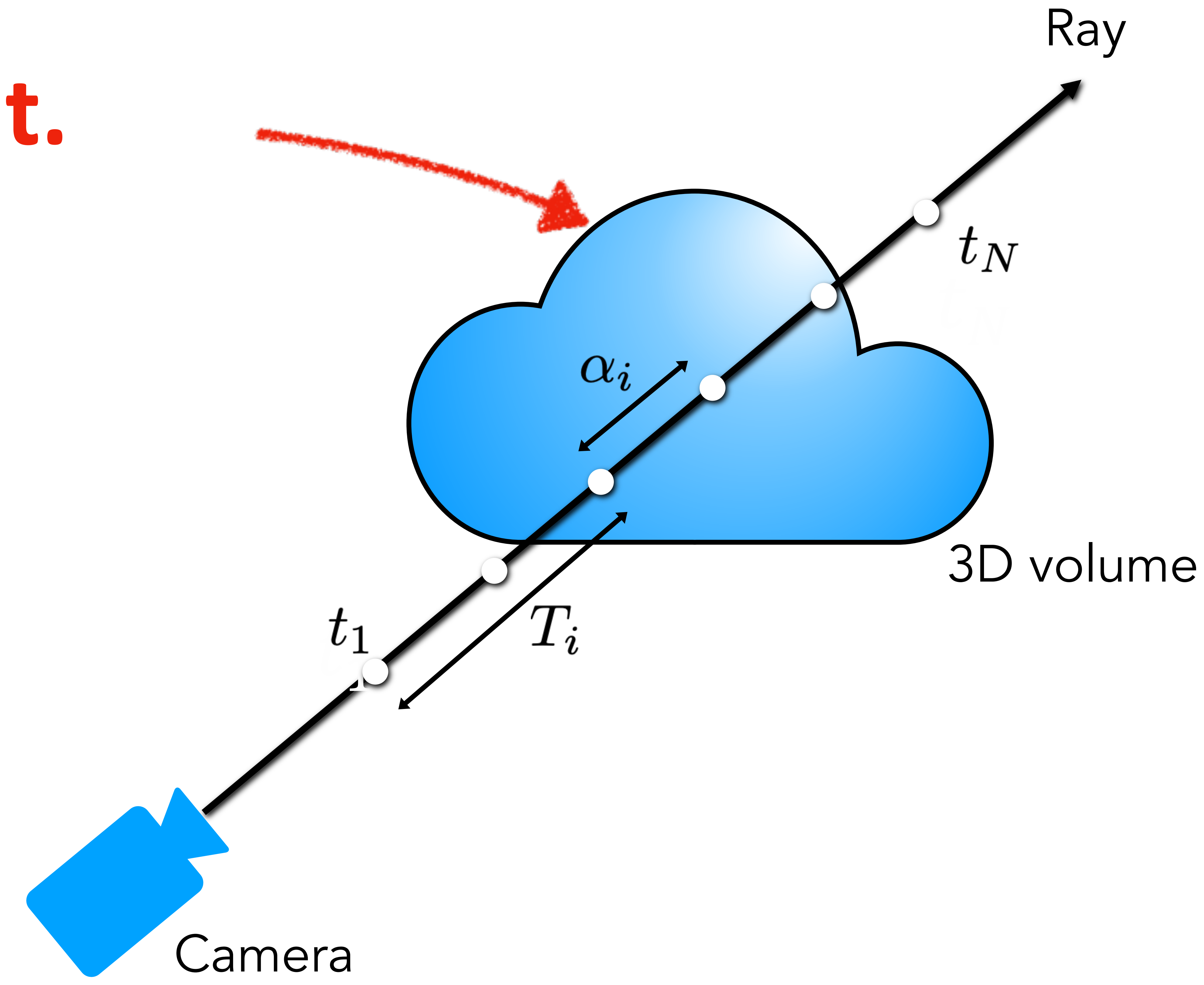
Rendering model for ray r(t) = o + td:

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

differentiable w.r.t.

colors

weights



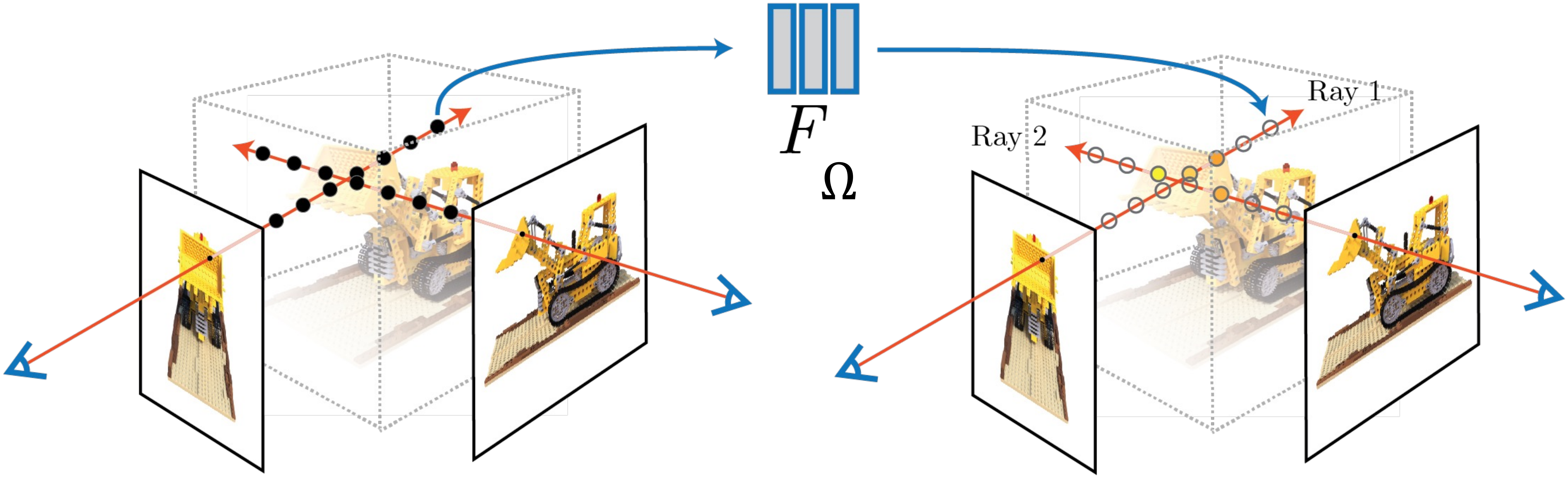
How much light is blocked earlier along ray:

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

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

$$\alpha_i = 1 - e^{-\sigma_i \delta t_i}$$

Optimize with gradient descent on rendering loss



$$\min_{\Omega} \sum_i \| \mathrm{render}^{(i)}(F_\Omega) - I_{\mathrm{gt}}^{(i)} \|^2$$

Training network to reproduce all input views of the scene

Naive implementation produces blurry results

NeRF (Naive)

Naive implementation produces blurry results

NeRF (Naive)

NeRF (with positional encoding)

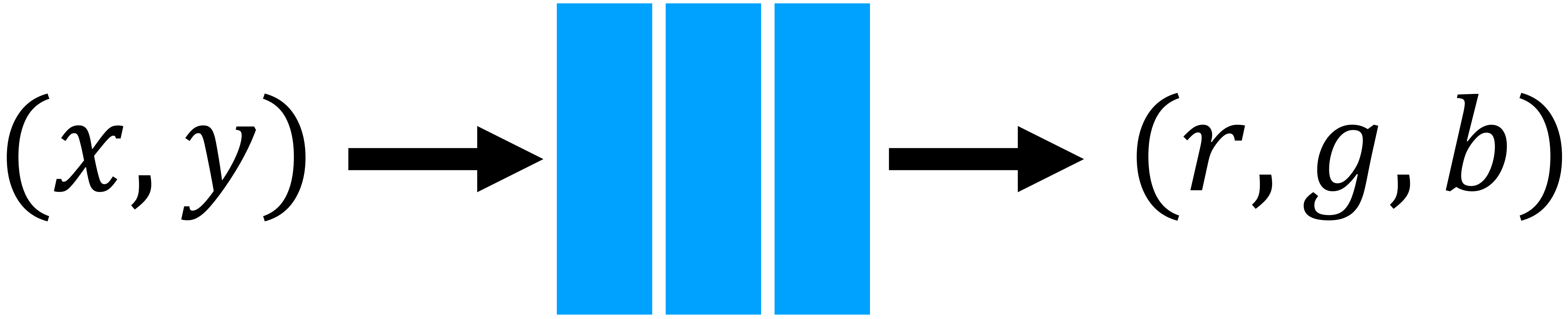
Key points

- Continuous neural network as a volumetric scene representation (5D = xyz + direction)
- Use volume rendering model to synthesize new views
- Optimize using rendering loss for one scene (no prior training)
- One extra trick for passing coordinates into network to get high frequency details

Challenge:

How to get MLPs to represent higher frequency functions?

Simpler toy problem: memorizing a 2D image

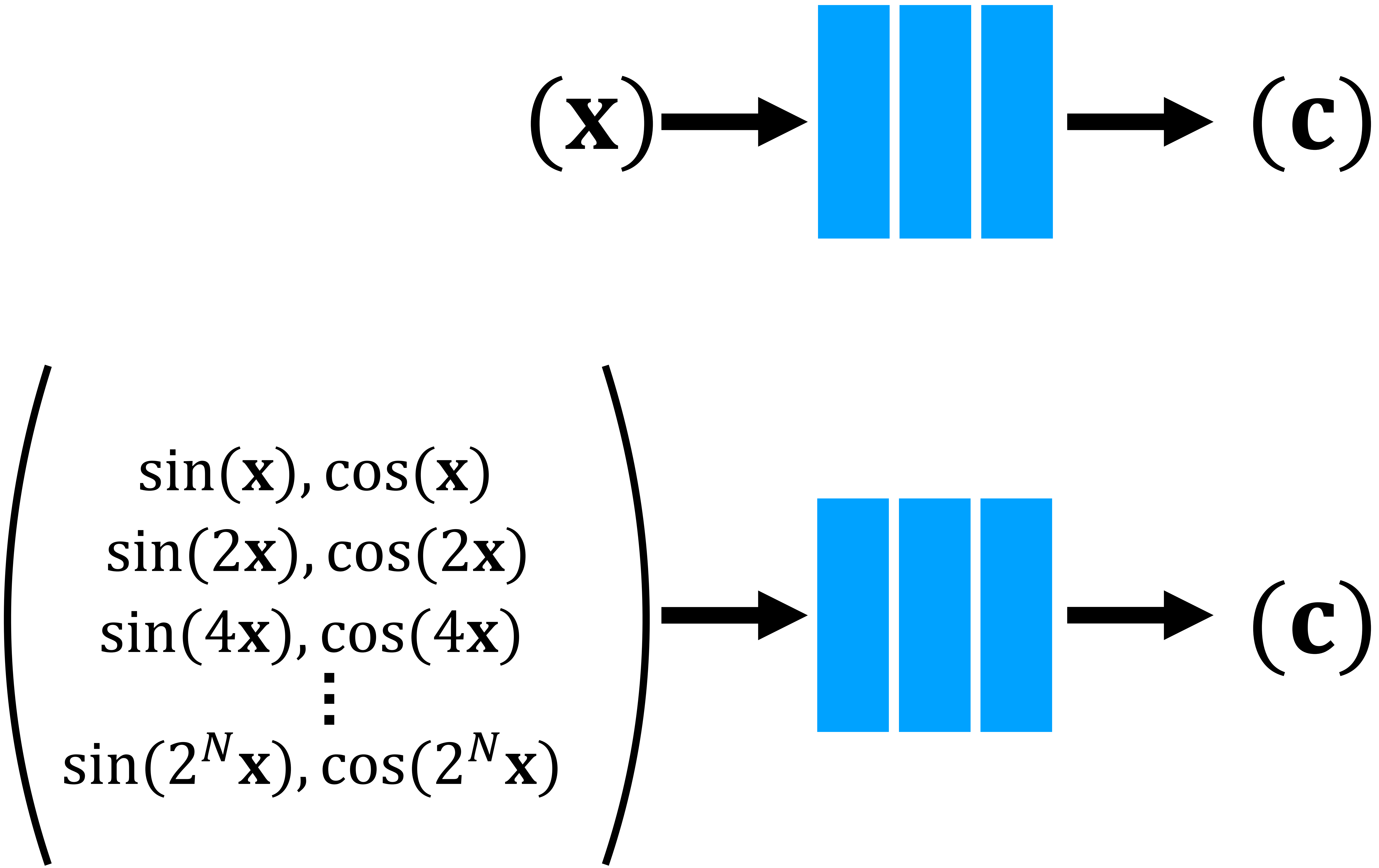


Simple trick enables network to memorize images

Ground truth image

Standard fully-connected net

Positional encoding: high frequency embedding of input coordinates



Simple trick enables network to memorize images

Ground truth image

Standard fully-connected net

With "embedding"

Positional encoding also directly improves our scene representation!

NeRF (Naive)

NeRF (with positional encoding)

Results

NeRF encodes convincing view-dependent effects using directional dependence

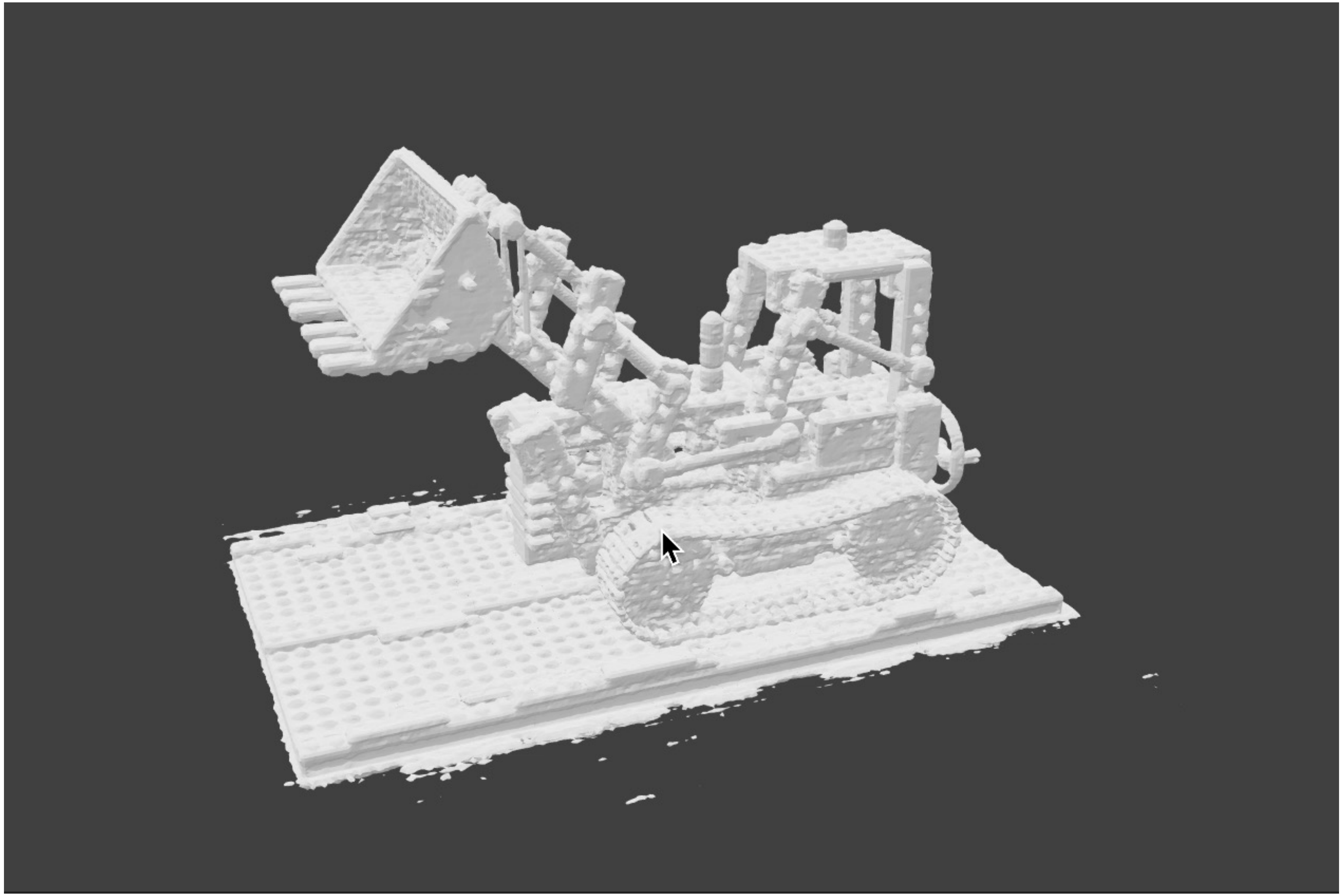
NeRF encodes convincing view-dependent effects using directional dependence


NeRF encodes detailed scene geometry with occlusion effects

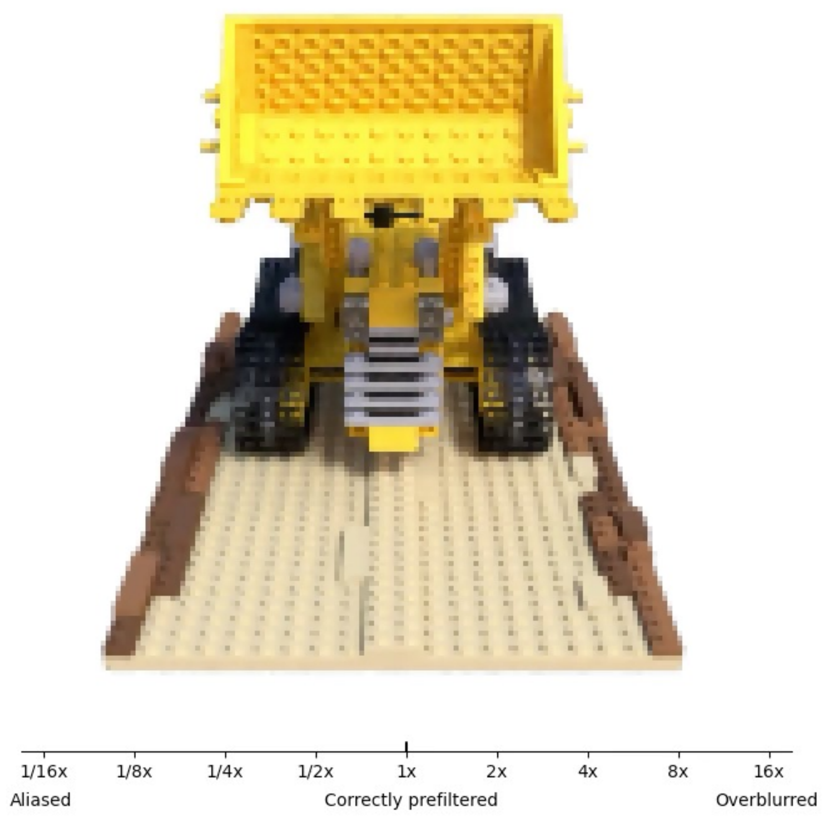
NeRF encodes detailed scene geometry with occlusion effects

NeRF encodes detailed scene geometry with occlusion effects

NeRF encodes detailed scene geometry



Thank You!



16-726, Spring 2025

https://learning-image-synthesis.github.io/

Video © Mip-NeRF [Barron et al., 2021]