COS 426: Computer Graphics

Neural Rendering

Felix Heide

PRINCETON UNIVERSITY

…. so far so good: Computer Graphics

Source: (Project Sol Part 2) https://www.youtube.com/watch?v=pNmhJx8yPLk

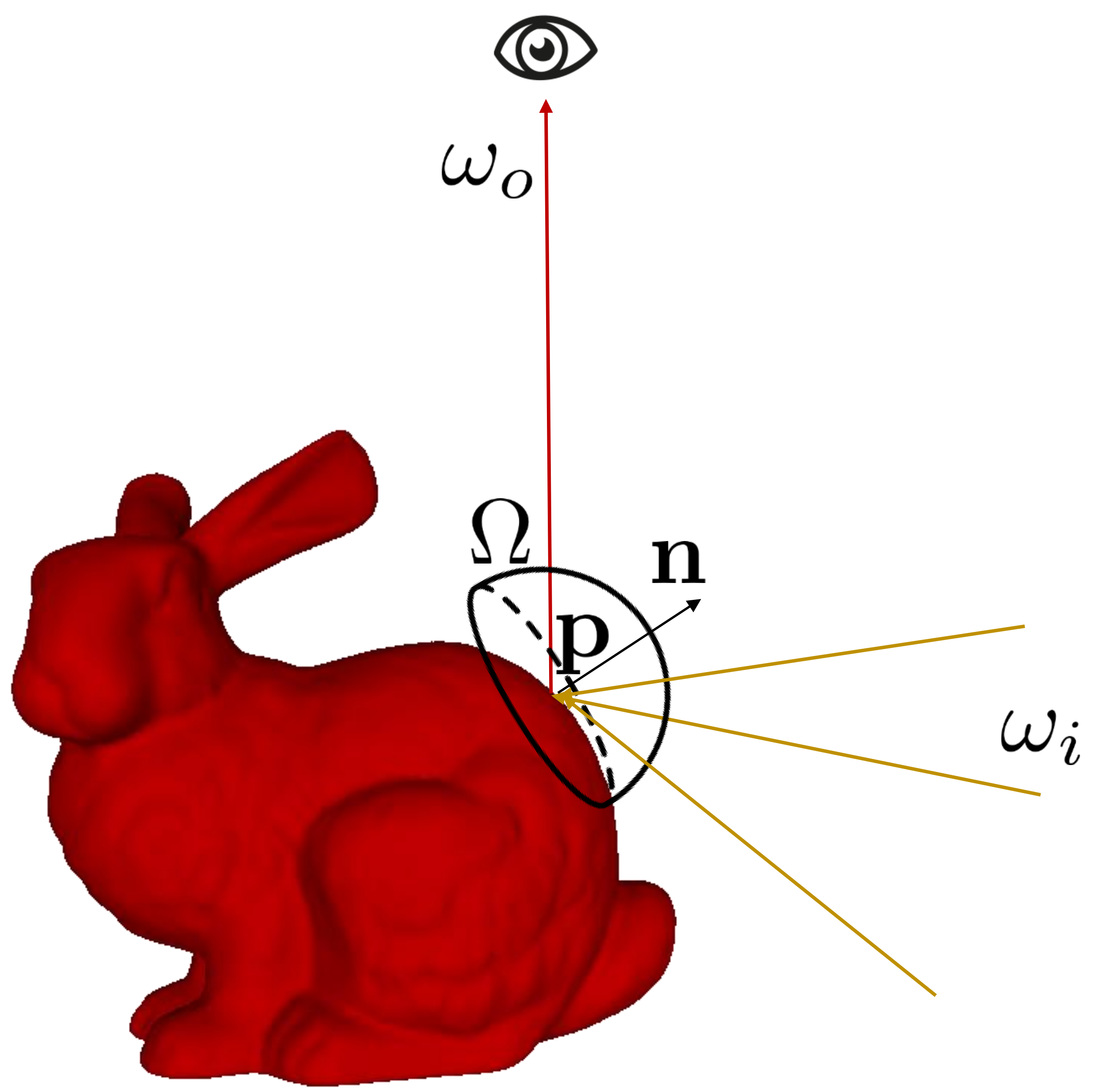
Rendering Equation

$$\boxed{L_{\mathrm{o}}(\mathbf{p},\, \omega_{\mathrm{o}})} = \boxed{L_{\mathrm{e}}(\mathbf{p},\, \omega_{\mathrm{o}})} + \boxed{\int_{\Omega} \boxed{L_{\mathrm{i}}(\mathbf{p},\, \omega_{\mathrm{i}})} \boxed{f_{\mathrm{r}}(\mathbf{p},\, \omega_{\mathrm{i}},\, \omega_{\mathrm{o}})\, (\omega_{\mathrm{i}} \cdot \mathbf{n})}\, \mathrm{d}\,\omega_{\mathrm{i}}}$$

Outgoing radiance

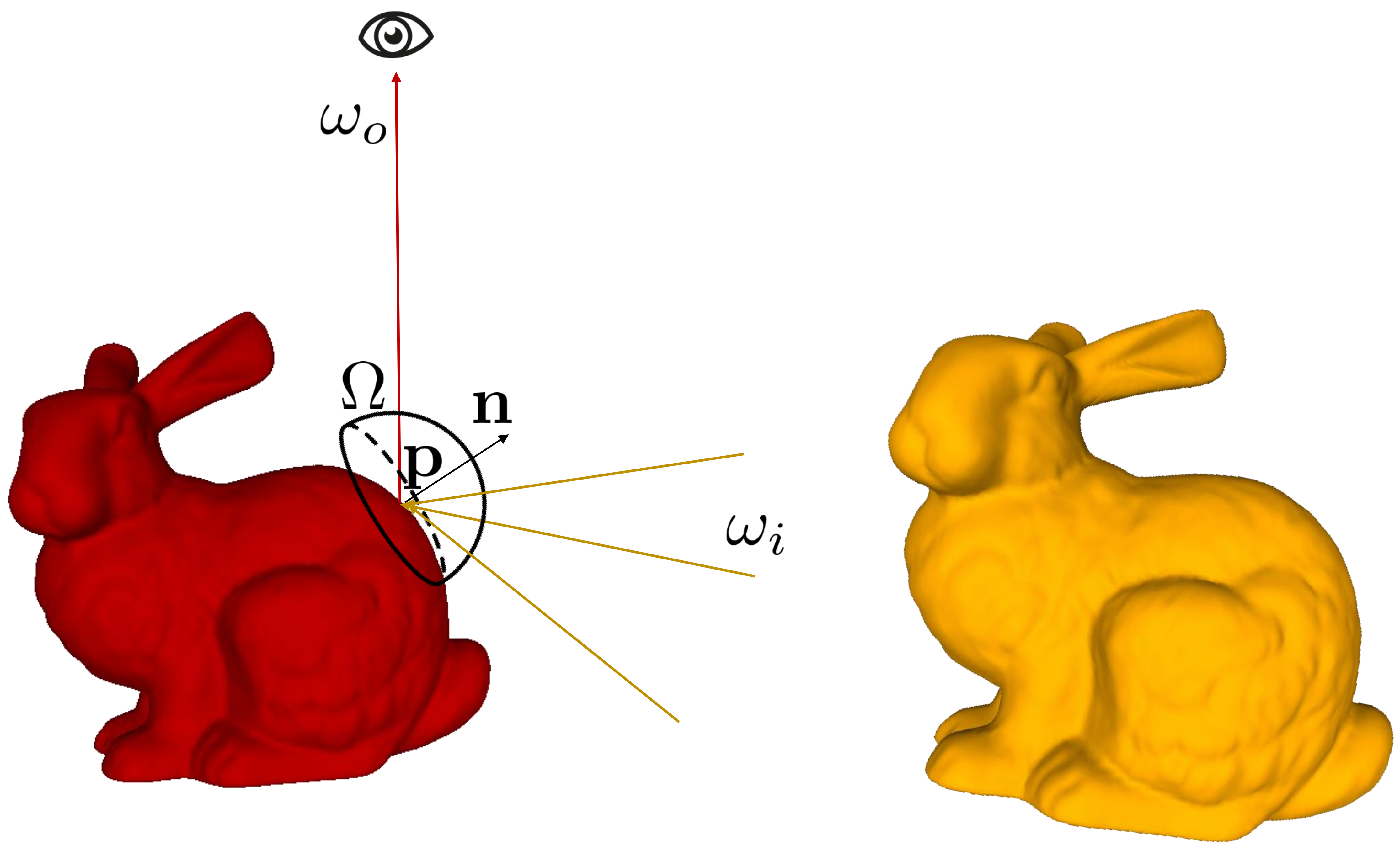
Incident radiance

BRDF

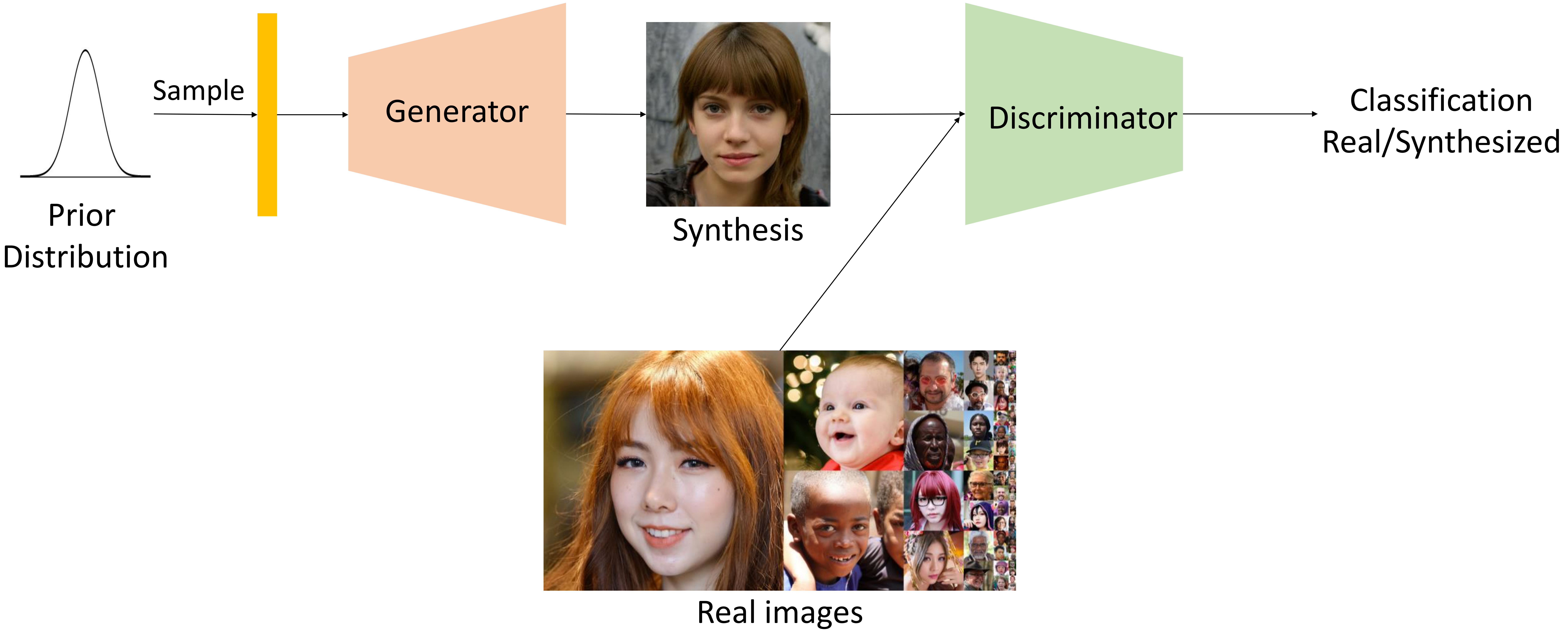


Rendering Equation

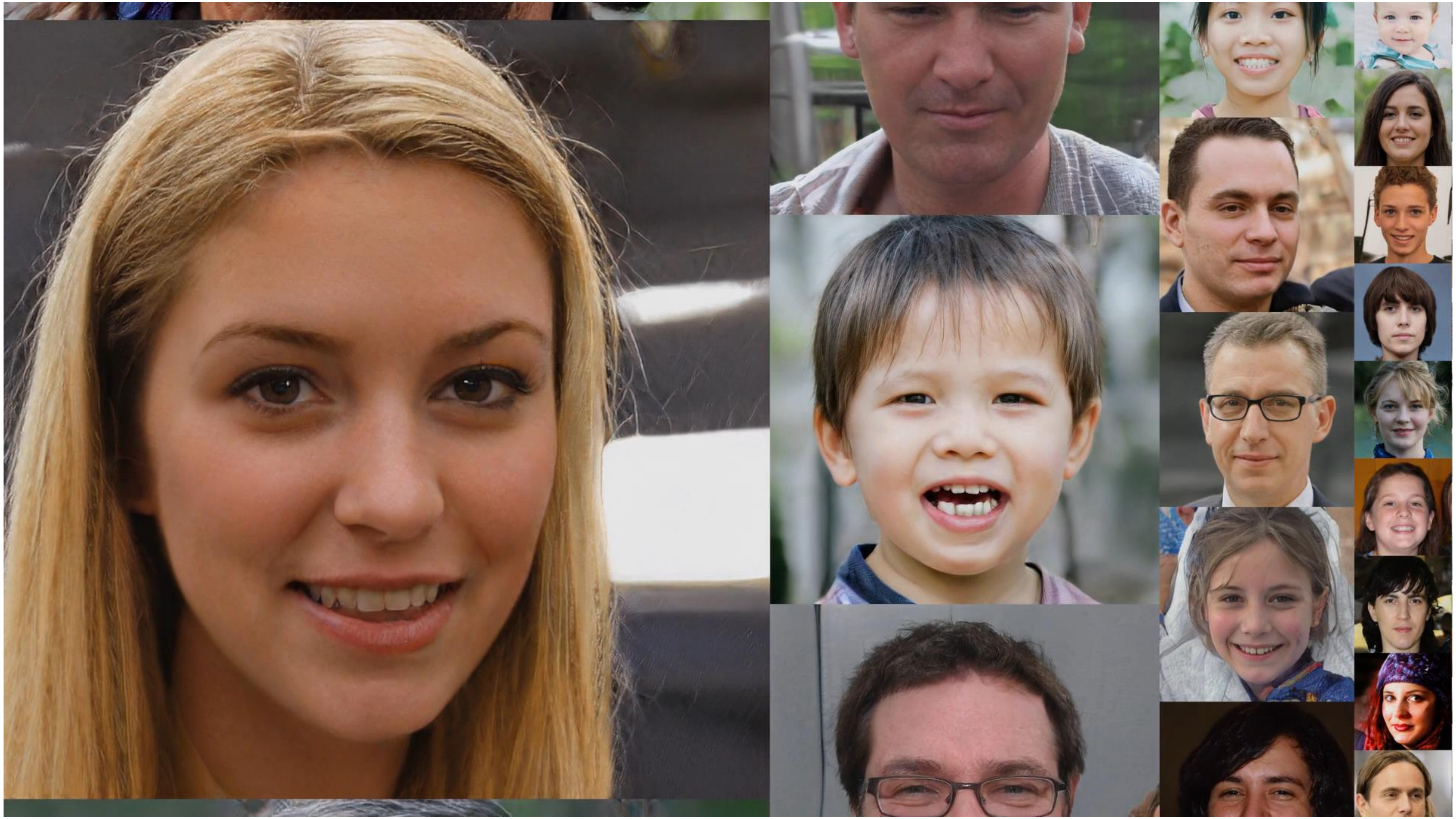
$$L_{\mathrm{o}}(\mathbf{p},\,\omega_{\mathrm{o}}) \;=\; L_{\mathrm{e}}(\mathbf{p},\,\omega_{\mathrm{o}}) \;+\; \int_{\Omega} \boxed{L_{\mathrm{i}}(\mathbf{p},\,\omega_{\mathrm{i}})}\, f_{\mathrm{r}}(\mathbf{p},\,\omega_{\mathrm{i}},\,\omega_{\mathrm{o}})\,(\omega_{\mathrm{i}}\,\cdot\,\mathbf{n})\;\mathrm{d}\,\omega_{\mathrm{i}}$$



Generative Adversarial Networks

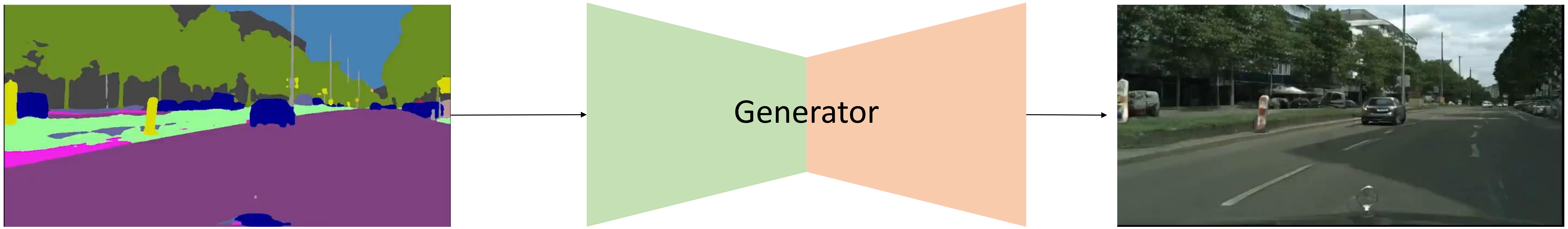


Generative Adversarial Networks



StyleGAN [Karras et al., 2019]

Conditional Generative Models



Vid2Vid [Wang et al., 2019]

Two Alternatives of Realistic Image Synthesis

Photo-realistic Rendering (CG)

Cons:

- Requires lots of manual work
  - Building of high-quality assets
  - Setting up the scene
- Long render times

Pros:

- Full control of scene parameter:
  - Camera, light sources, motion, geometry, appearance

Generative Machine Learning (ML)

Cons:

- Requires lots of training data
- No fine-grained semantic control of the scene parameters, e.g., motion or illumination

Pros:

- Fully automatic training
- Interactive inference/rendering

Neural Rendering to the rescue!

Neural Rendering - Graphics vs. Learning

Scene Description:

- Light sources
- Geometry
- Materials
- Motion
- Camera

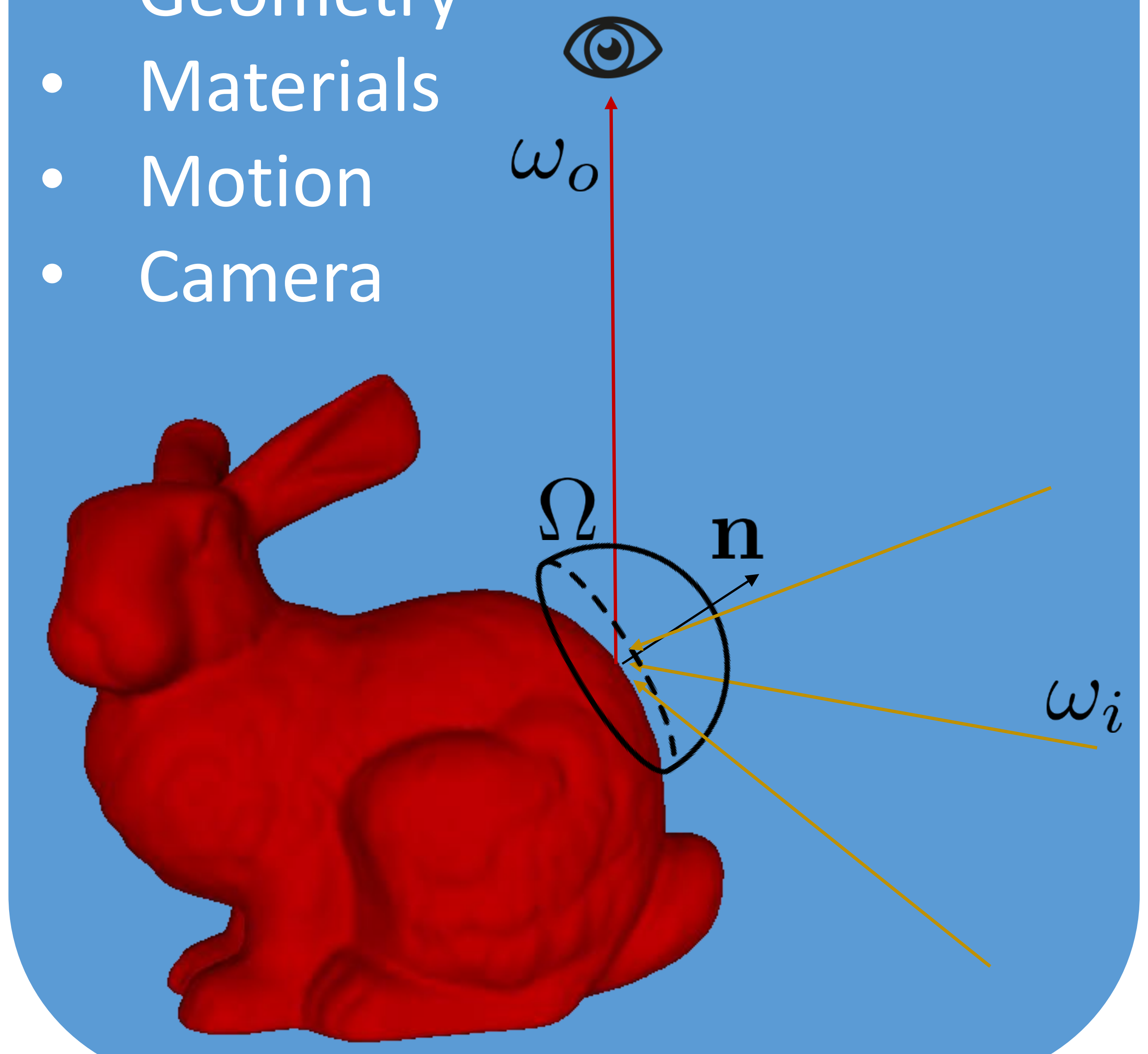
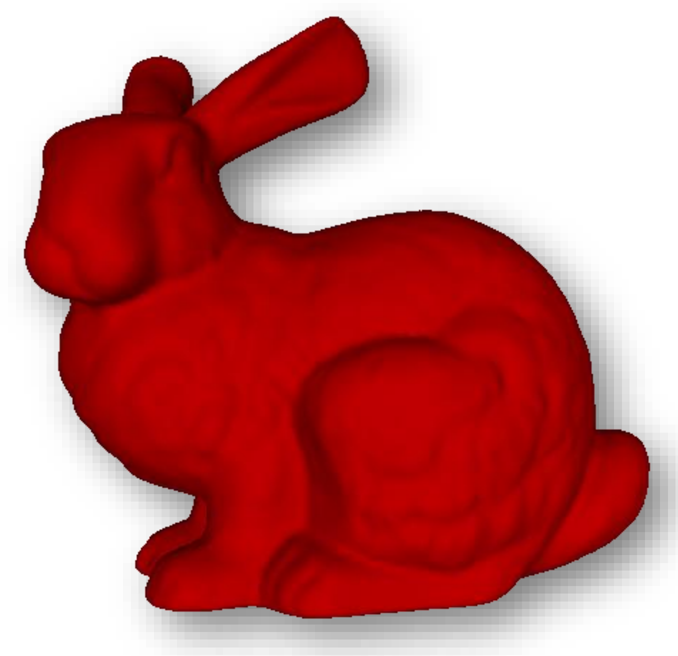


Image = CG(Scene)



Can we learn (part of the) scene representation and/or (part of the) "CG" function?

Real Images

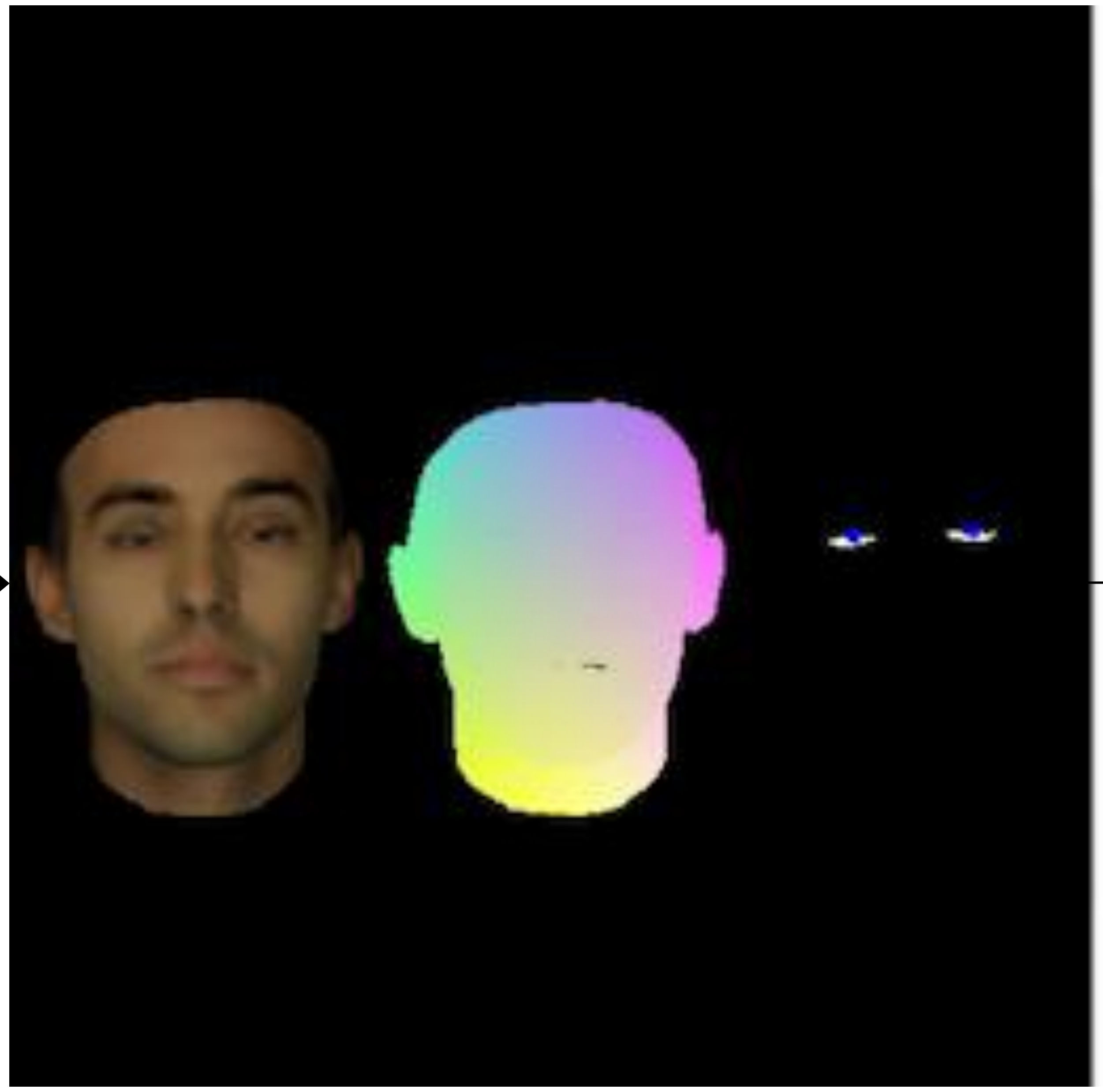
Neural Rendering to the rescue!

CG Modules

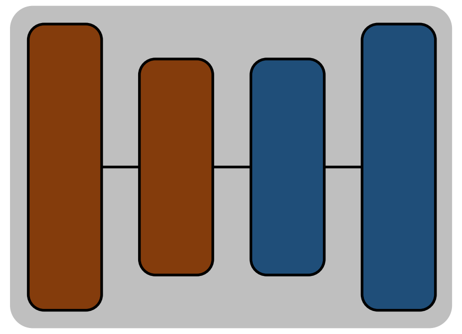
Computer Graphics Module: Rasterization is used to synthesize the input to the network.

Rendering Parameters

Rasterization



Rendering



Rendering-to-video translation network

Face Animation

Deep Video Portraits [Kim et al., 2018]

CG Modules

Computer Graphics Module: *Differentiable* Volume Renderer

Neural Volumes [Lombardi et al., 2019]

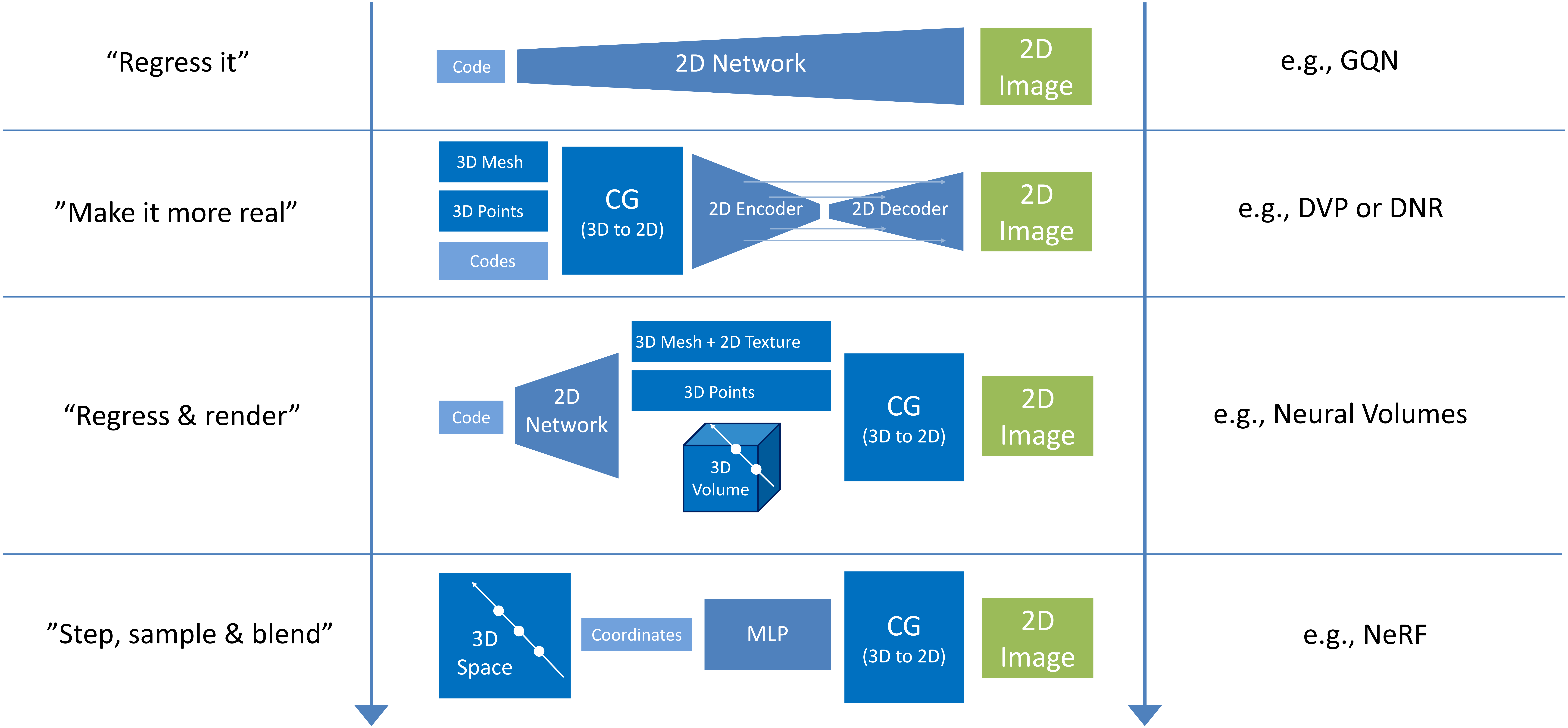
Neural Rendering Definition:

"(Deep) neural networks for image or video generation that

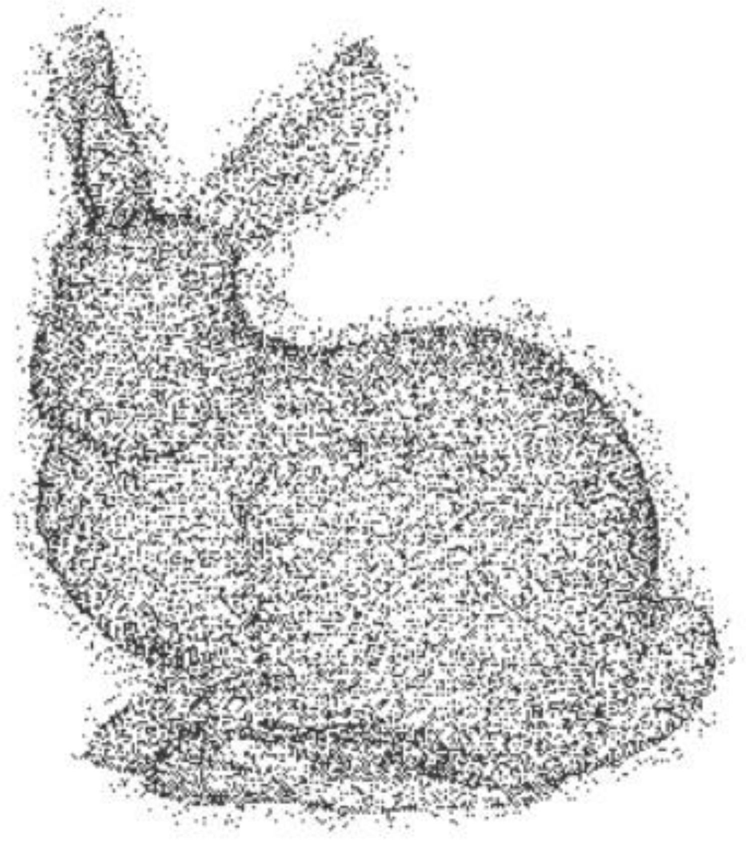
enable explicit or implicit control of scene properties"

Allows us to create photorealistic assets

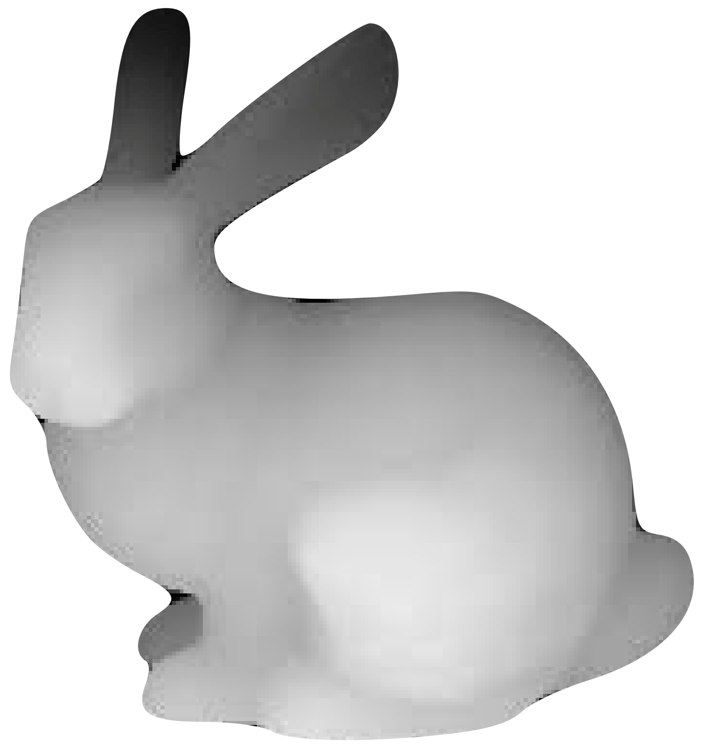
Neural Rendering Zoo



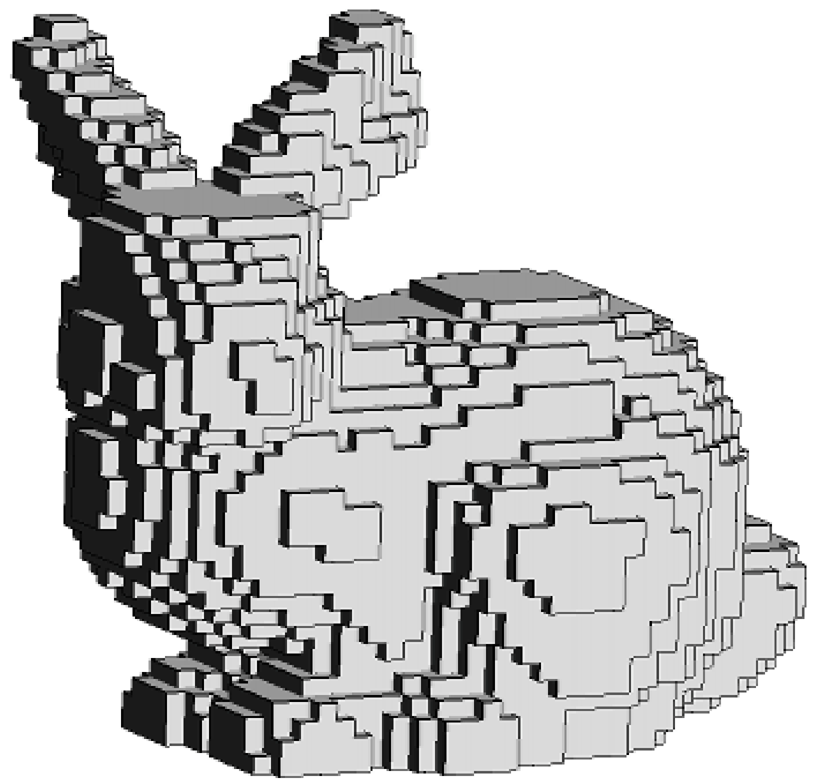
What's the right scene representation?



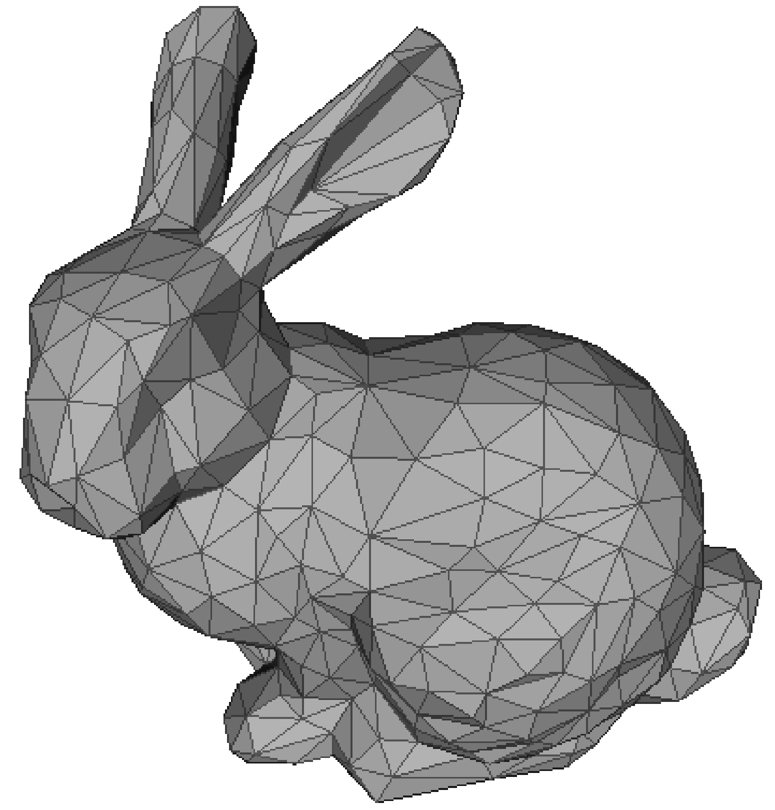
Point cloud



Depth map



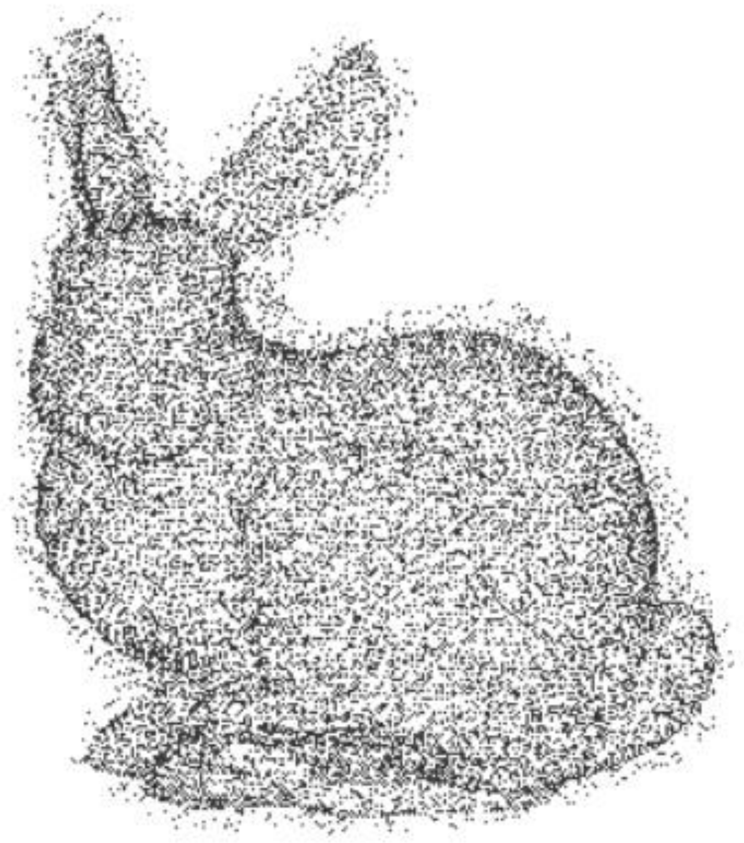
Voxel grid



Triangle mesh

What’s the right scene representation?

- **Do sensors output data in this representation?**


Point cloud

Depth map

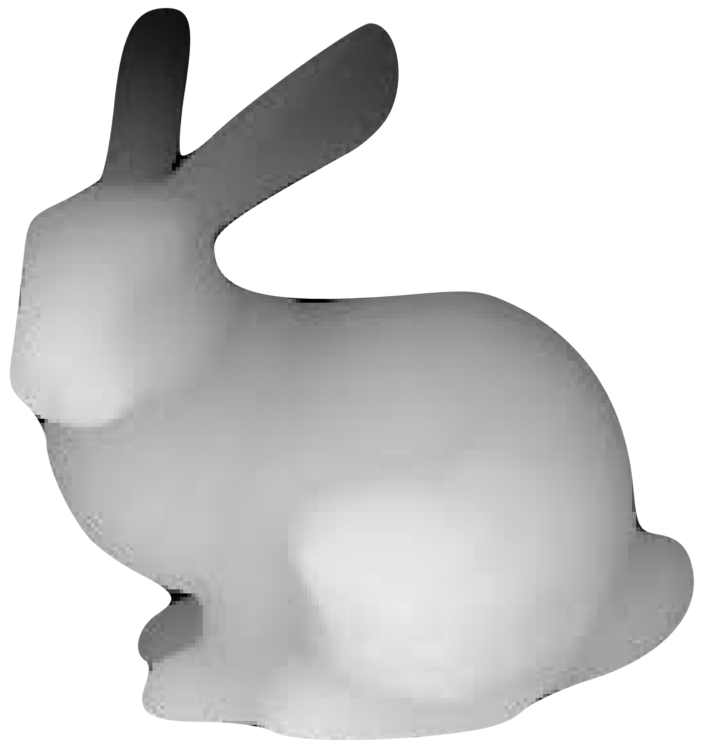
Voxel grid

Triangle mesh

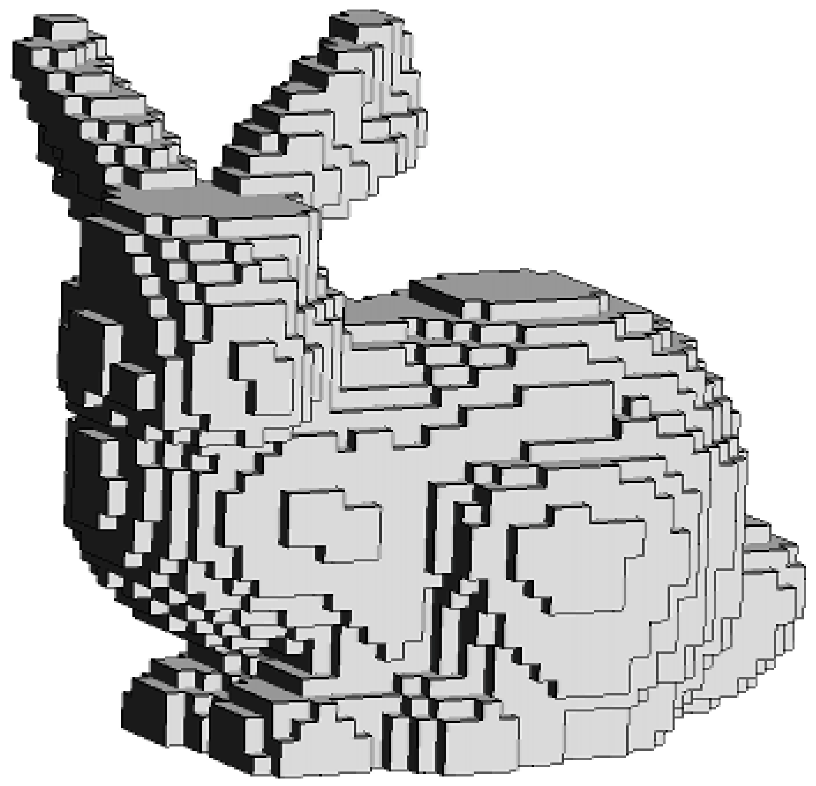
What's the right scene representation?

- Do sensors output data in this representation?
- Can we process/generate content in this representation?

Point cloud



Depth map



Voxel grid

Triangle mesh

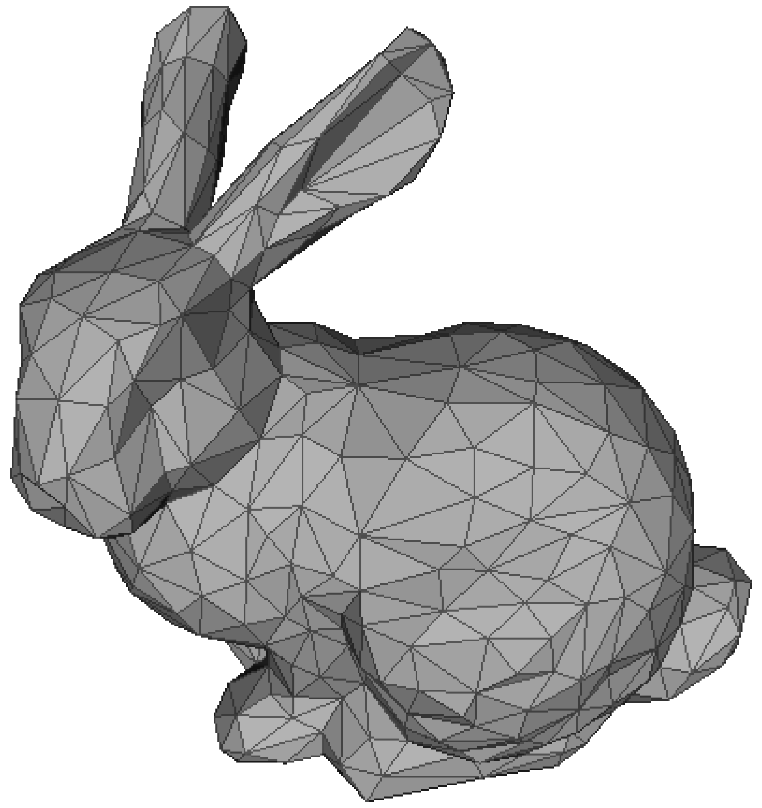
What's the right scene representation?

- Do sensors output data in this representation?
- Can we process/generate content in this representation?
- Can we easily render this representation?

Point cloud

Depth map

Voxel grid



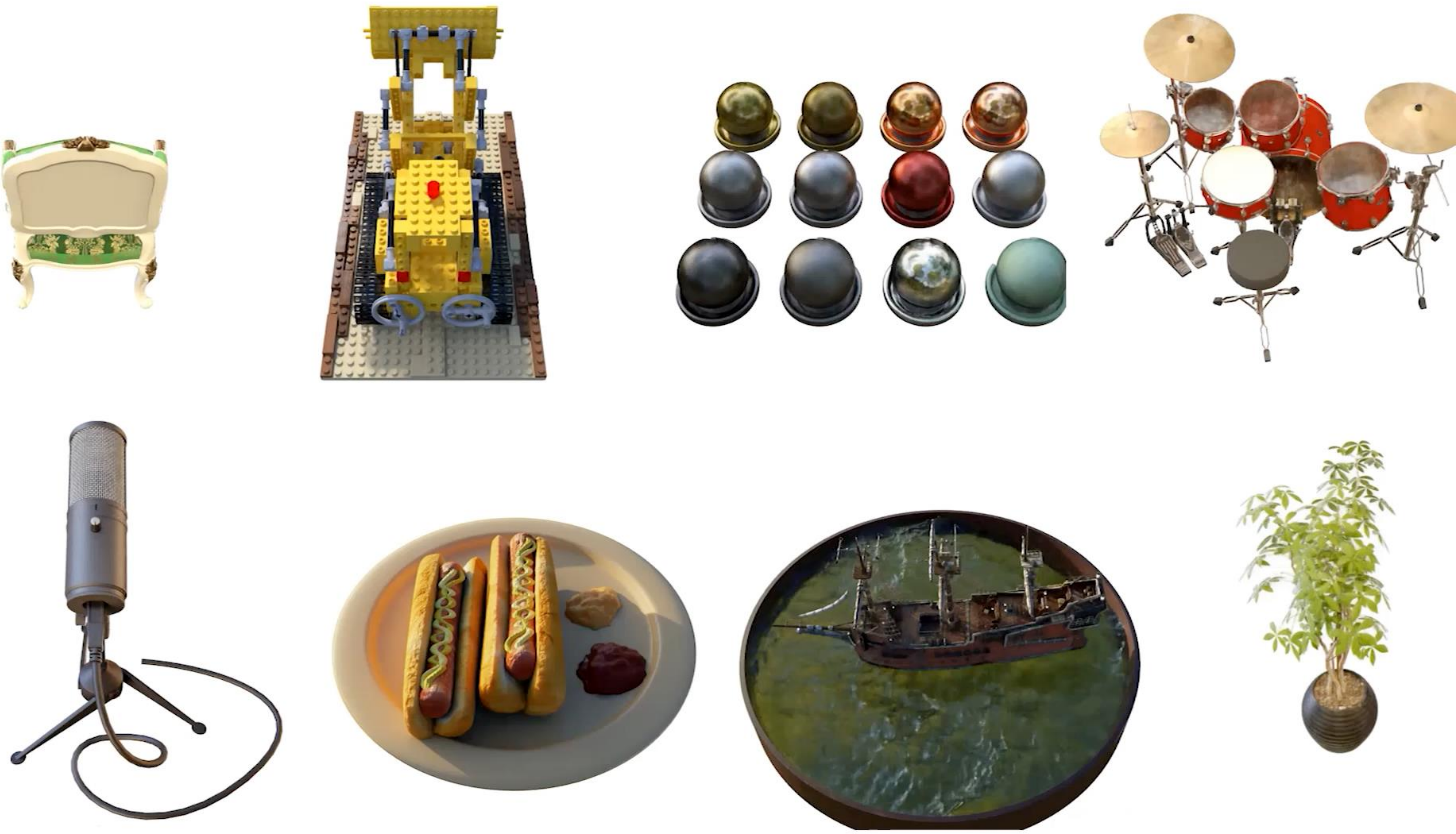
Triangle mesh

What’s the right scene representation for 3D vision?

- Sensors don’t give us nice data
- World is not dense in 3D space, so maybe representation shouldn’t be
- Doing computation directly on “efficient” representations is hard
- What's missing? Efficient representation that’s easy to optimize with gradient-based methods!

NeRF (neural radiance fields):

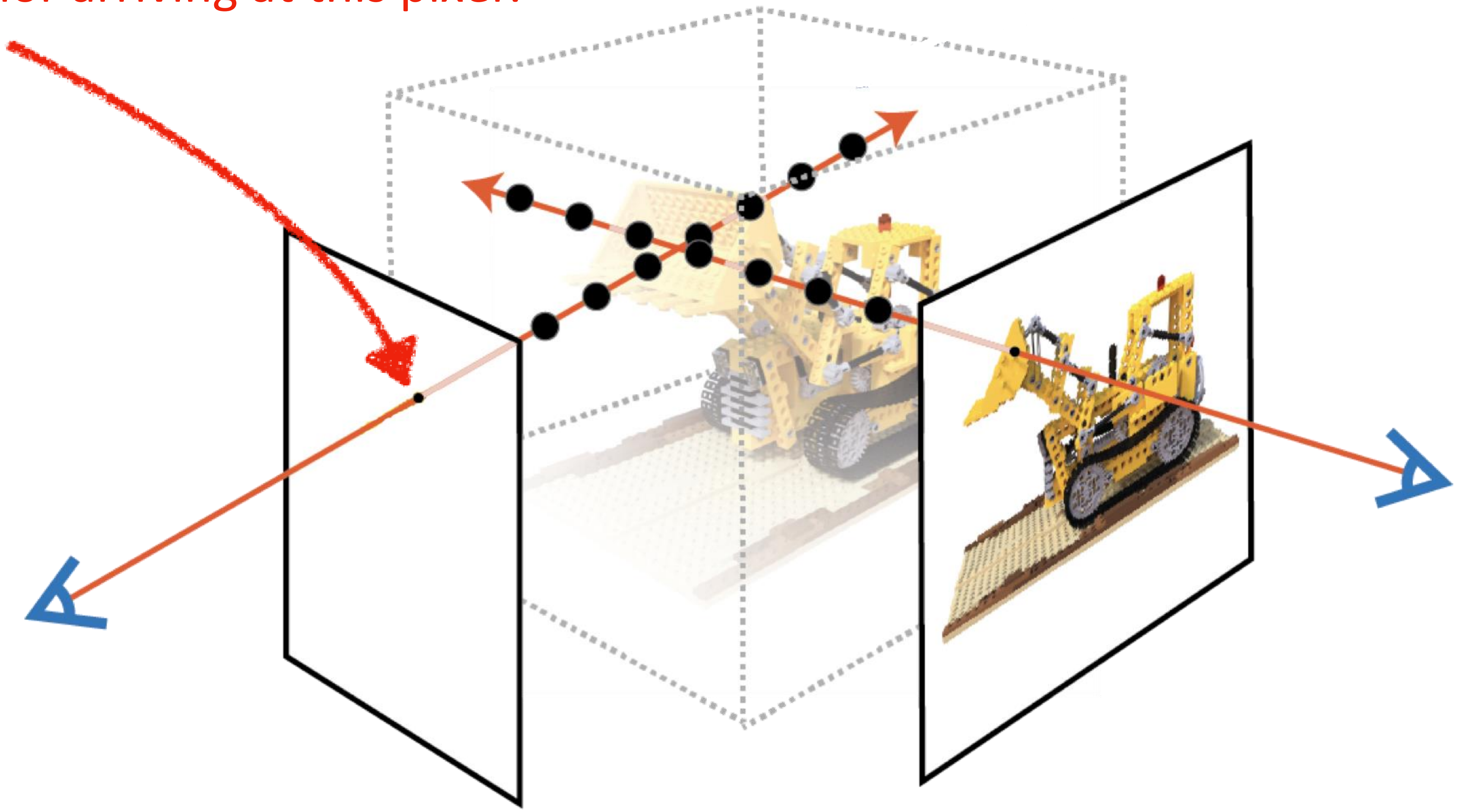
Neural networks as a volume representation, using volume rendering to do view synthesis.$(x, y, z, \theta, \phi) \rightarrow color, opacity$



Neural Volumetric Rendering

Neural Volumetric Rendering

What's the radiance/color arriving at this pixel?

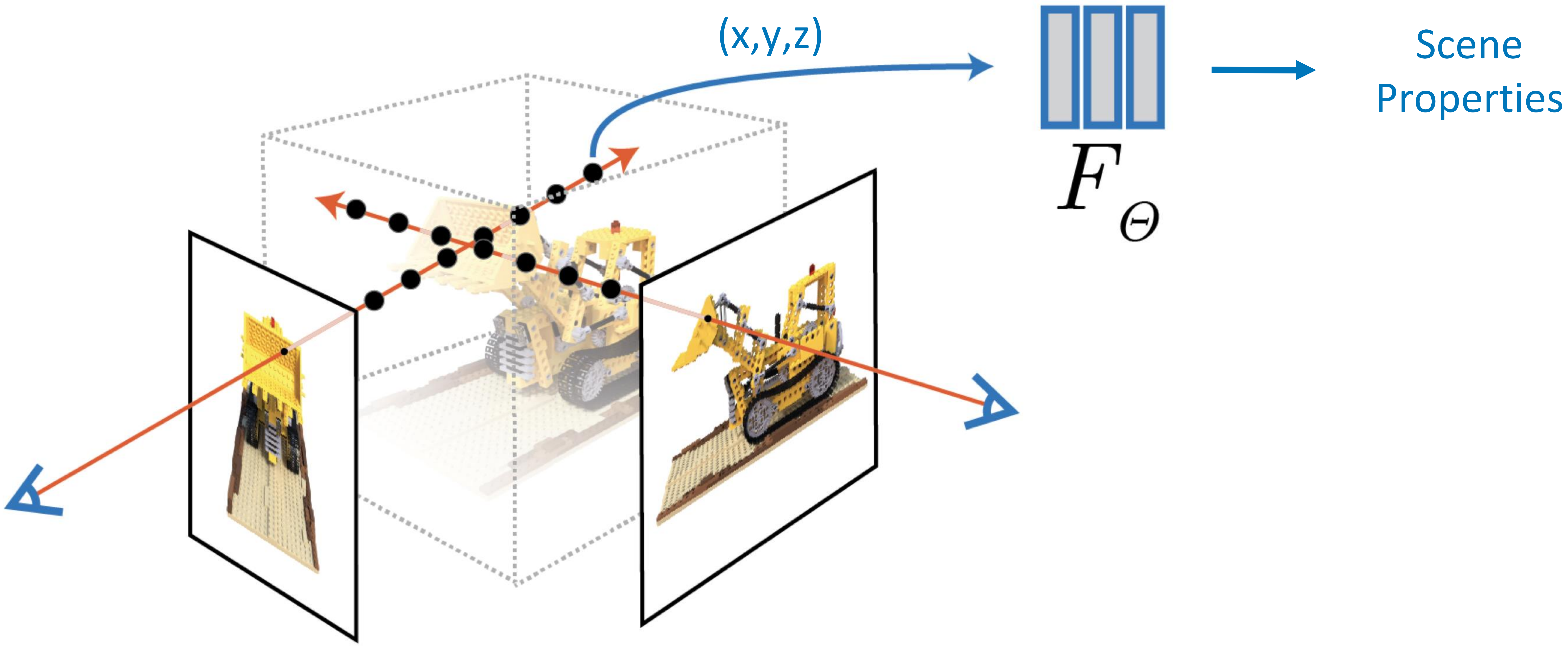


Neural Volumetric Rendering

- “Soft” volumetric functions better suited for gradient-based optimization

Neural Volumetric Rendering

- (Coordinate-based) neural network represents scene as continuous function



NeRF: neural volumetric rendering for view synthesis

Inputs: sparsely sampled images of scene

Outputs: new views of same scene


NeRF in the Wild, Martin-Brualla et al.

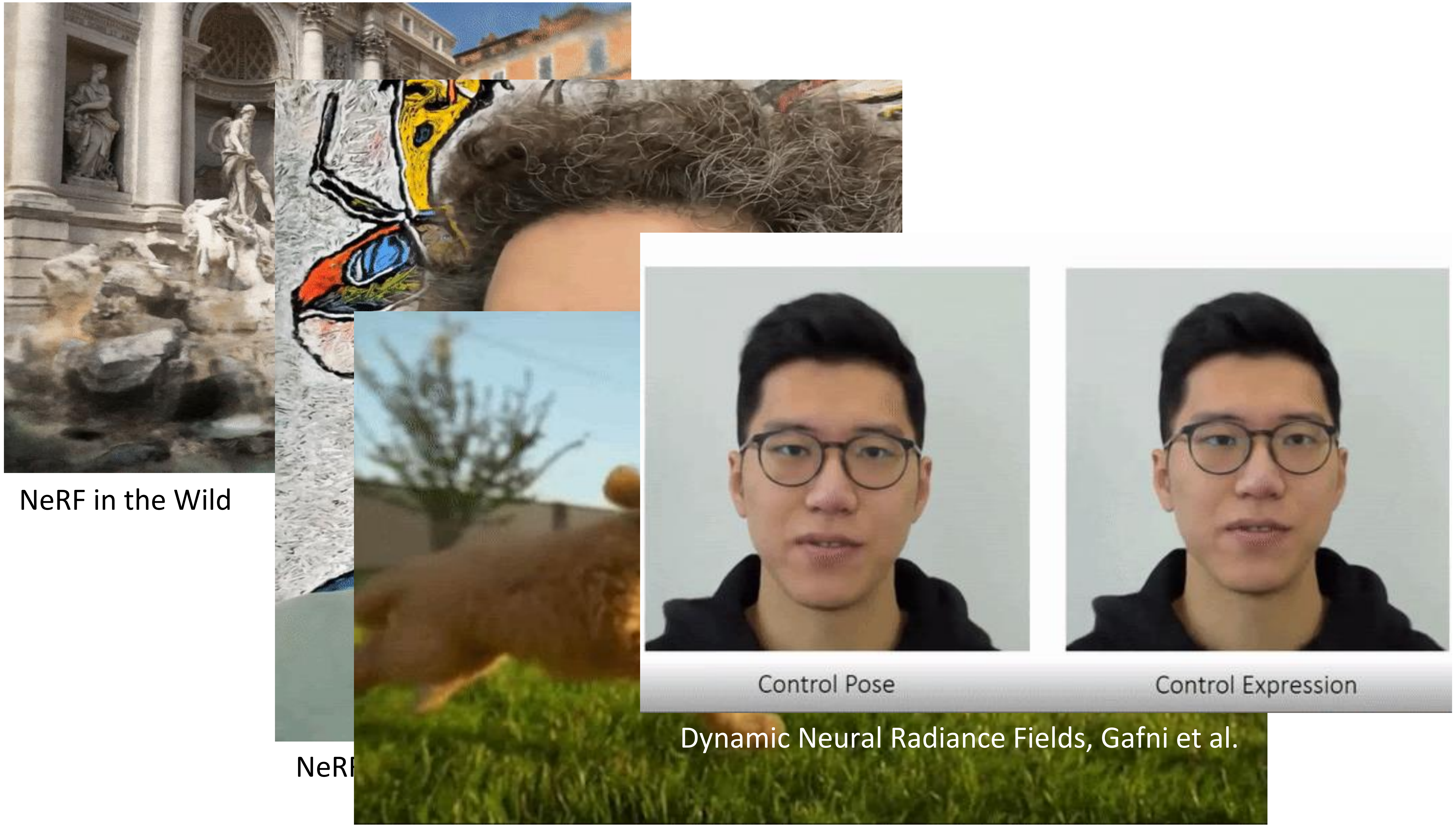
NeRF in the Wild, M

NeRFies, Park et al.

NeRF in the Wild

NeRF

Neural Scene Flow Fields, Li et al.



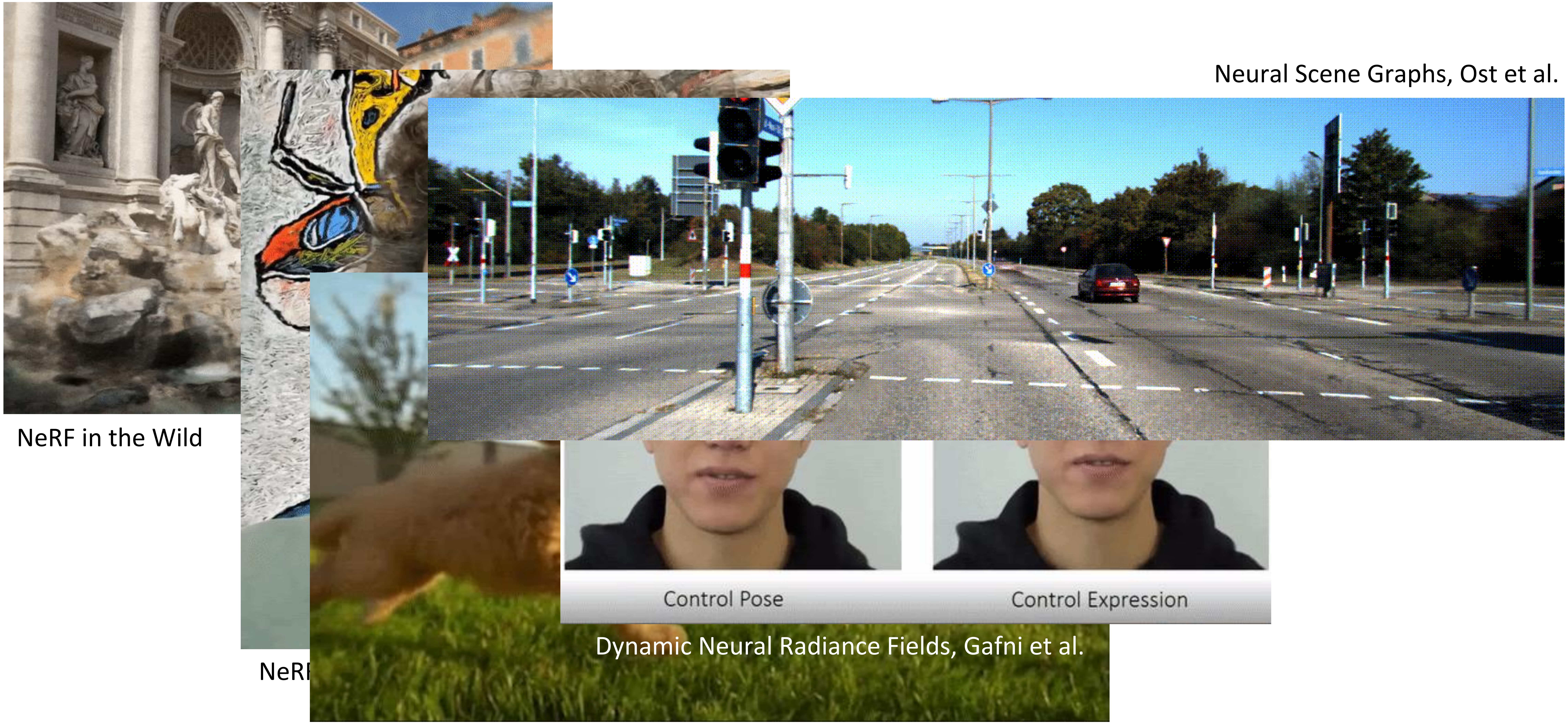
NeRF in the Wild

NeRF

Dynamic Neural Radiance Fields, Gafni et al.

Neural Scene Flow Fields, Li et al.

Neural Scene Graphs, Ost et al.



NeRF in the Wild

NeRF

Dynamic Neural Radiance Fields, Gafni et al.

Neural Scene Flow Fields, Li et al.

Representing a scene as a continuous 5D function

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

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

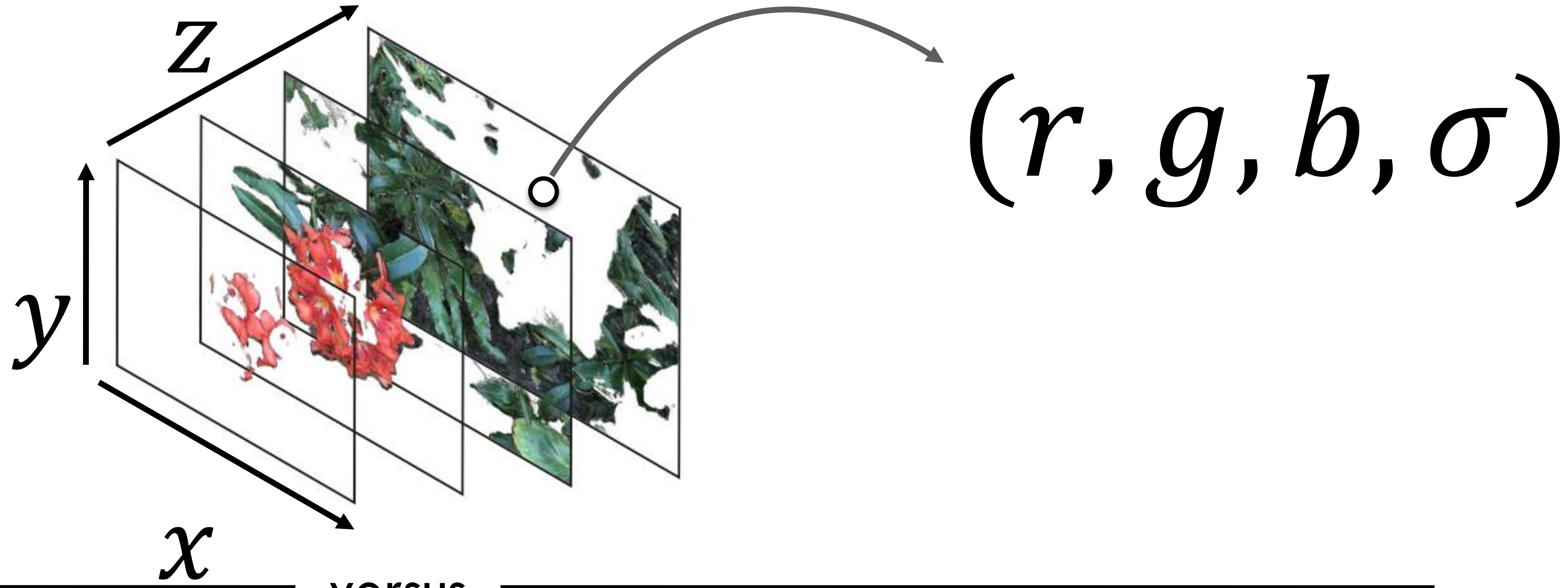
F_Ω

Fully-connected neural network

9 layers,

256 channels

Neural network replaces large N-d array: tradeoff between storage and computation



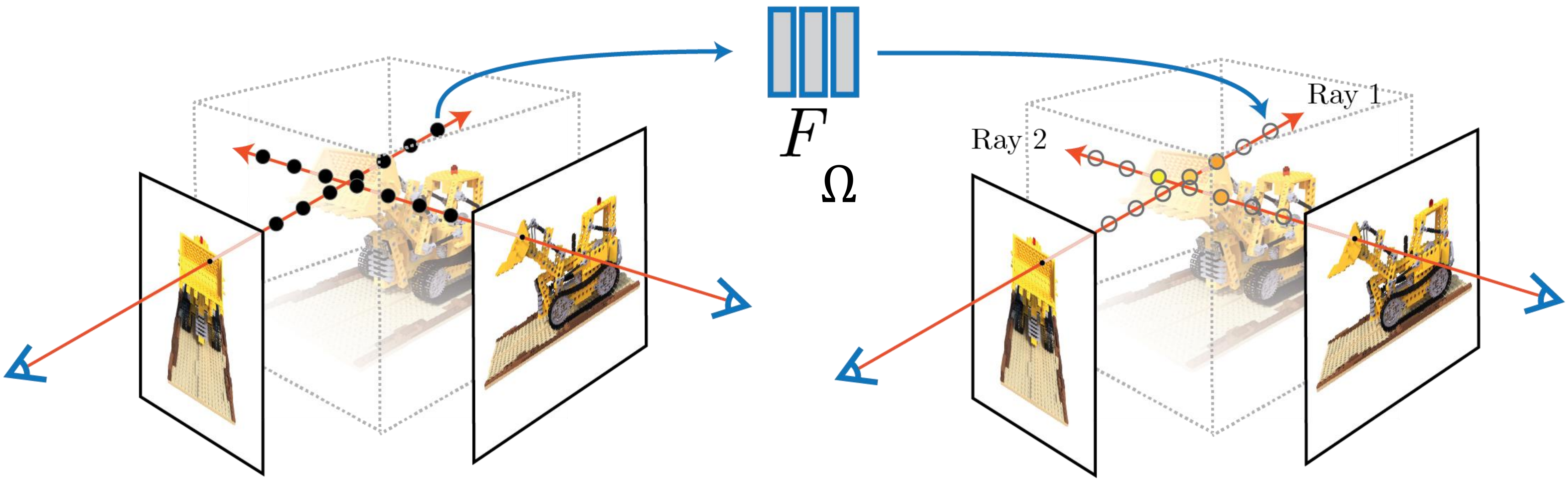
versus

$$(x, y, z, \theta, \phi) \rightarrow$$

$$\rightarrow (r, g, b, \sigma)$$

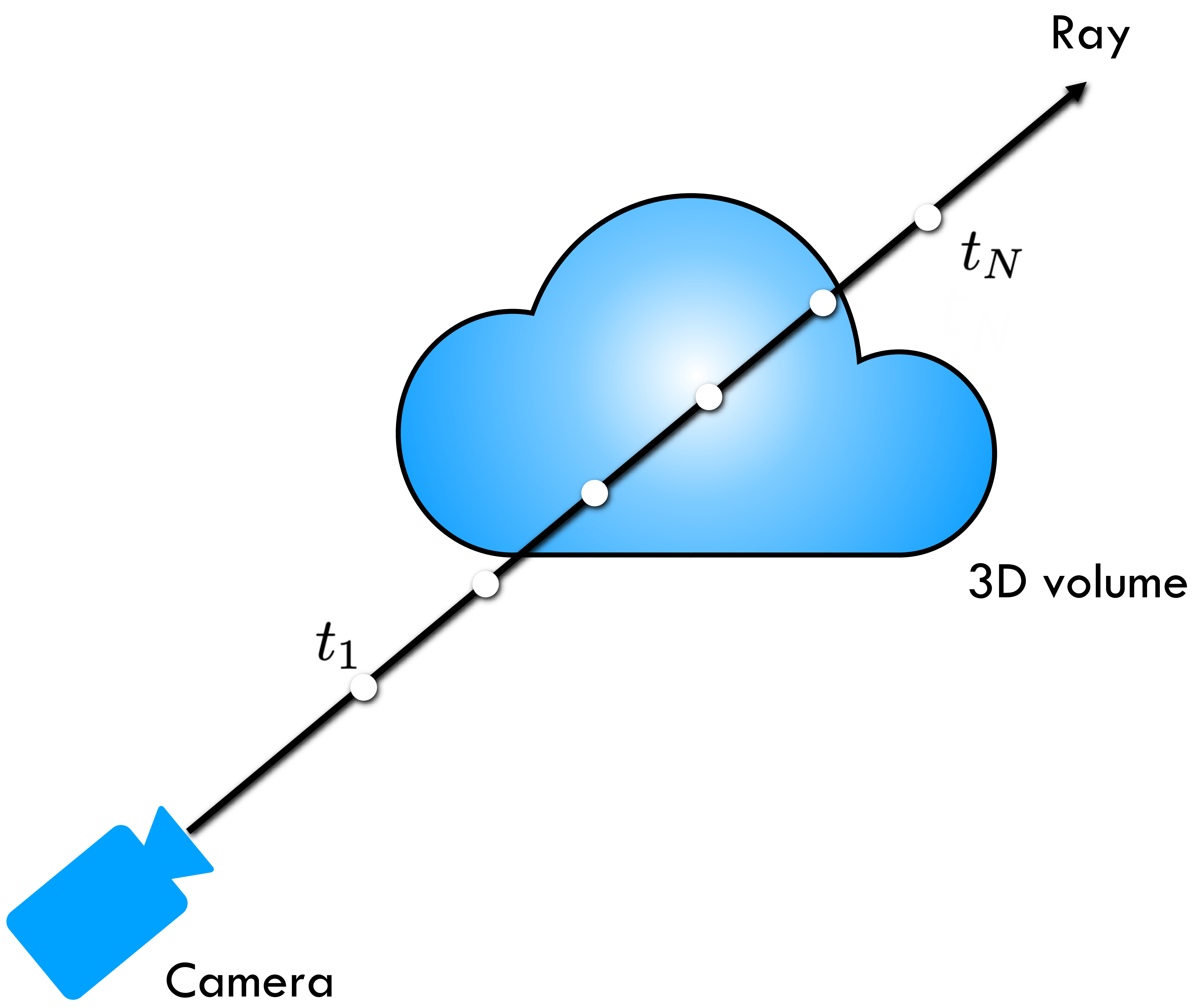
$$F_{\Omega}$$

Generate views with traditional volume rendering



Generate views with traditional volume rendering

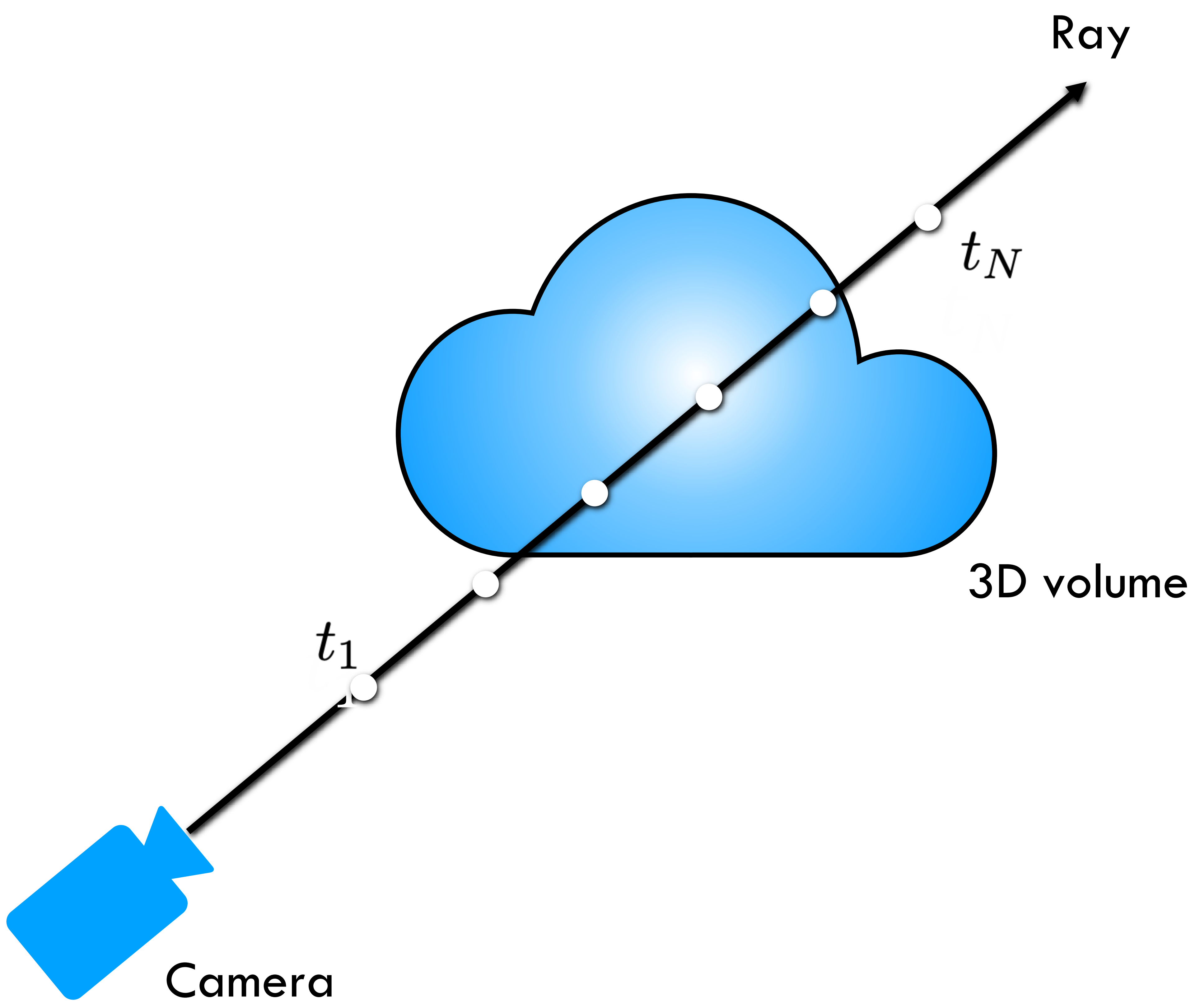
Rendering model for ray r(t) = o + td:



Generate views with traditional volume rendering

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

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



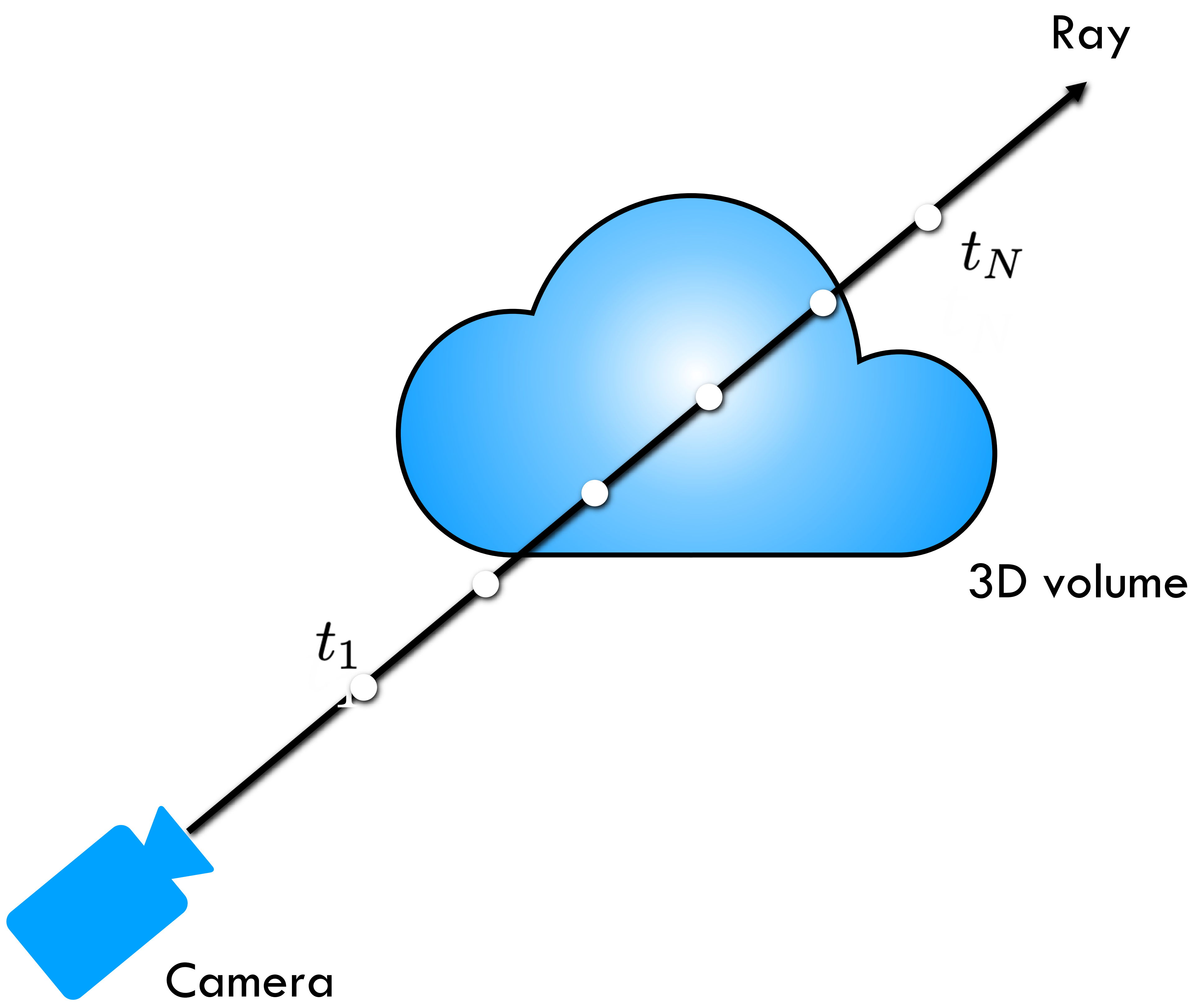
Generate views with traditional volume rendering

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

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

colors

weights



Generate views with traditional volume rendering

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

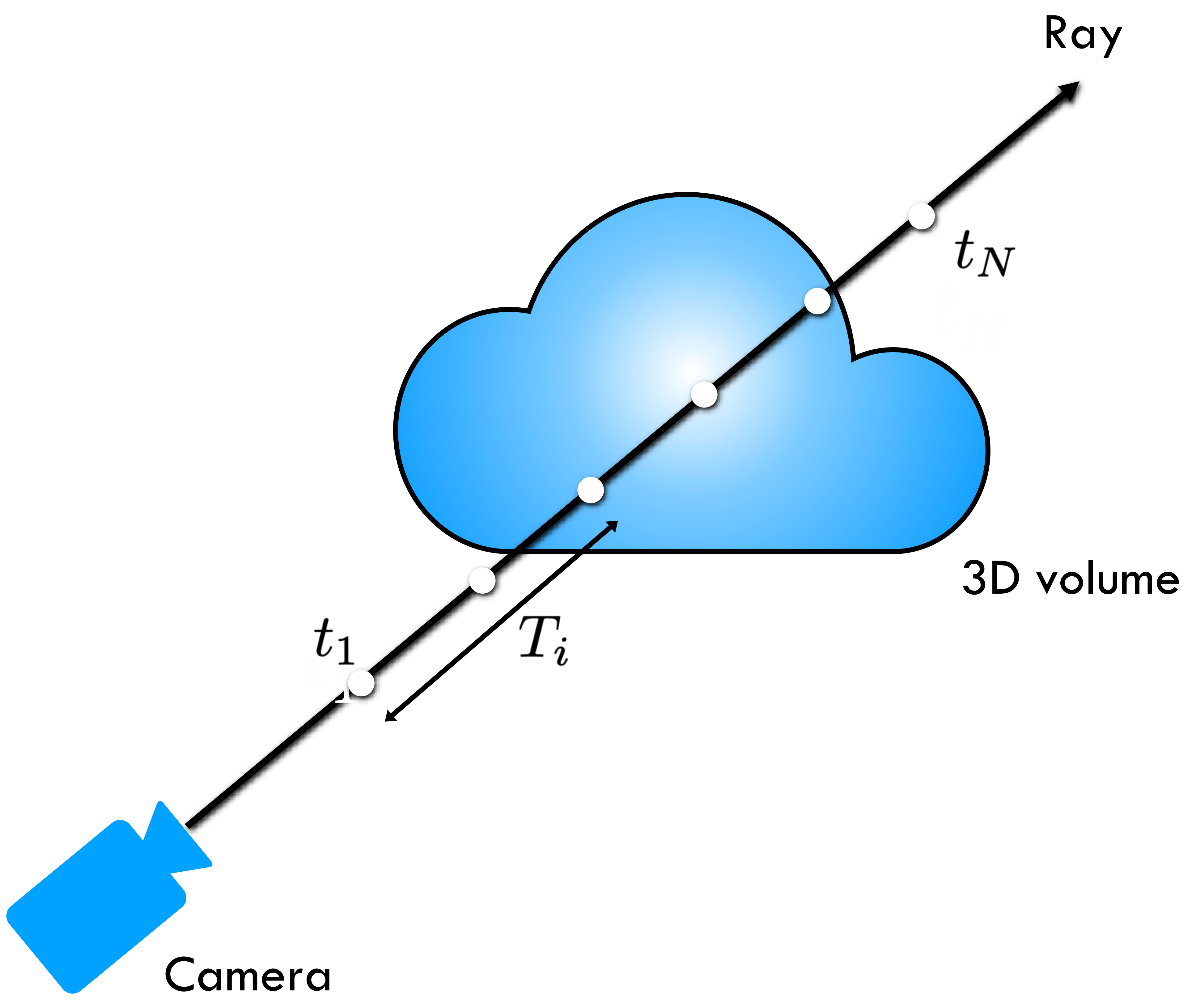
$$C \approx \sum_{i=1}^{N} \boxed{T_i}\, \alpha_i c_i$$

colors

weights

How much light is blocked earlier along ray:

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



Generate views with traditional volume rendering

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

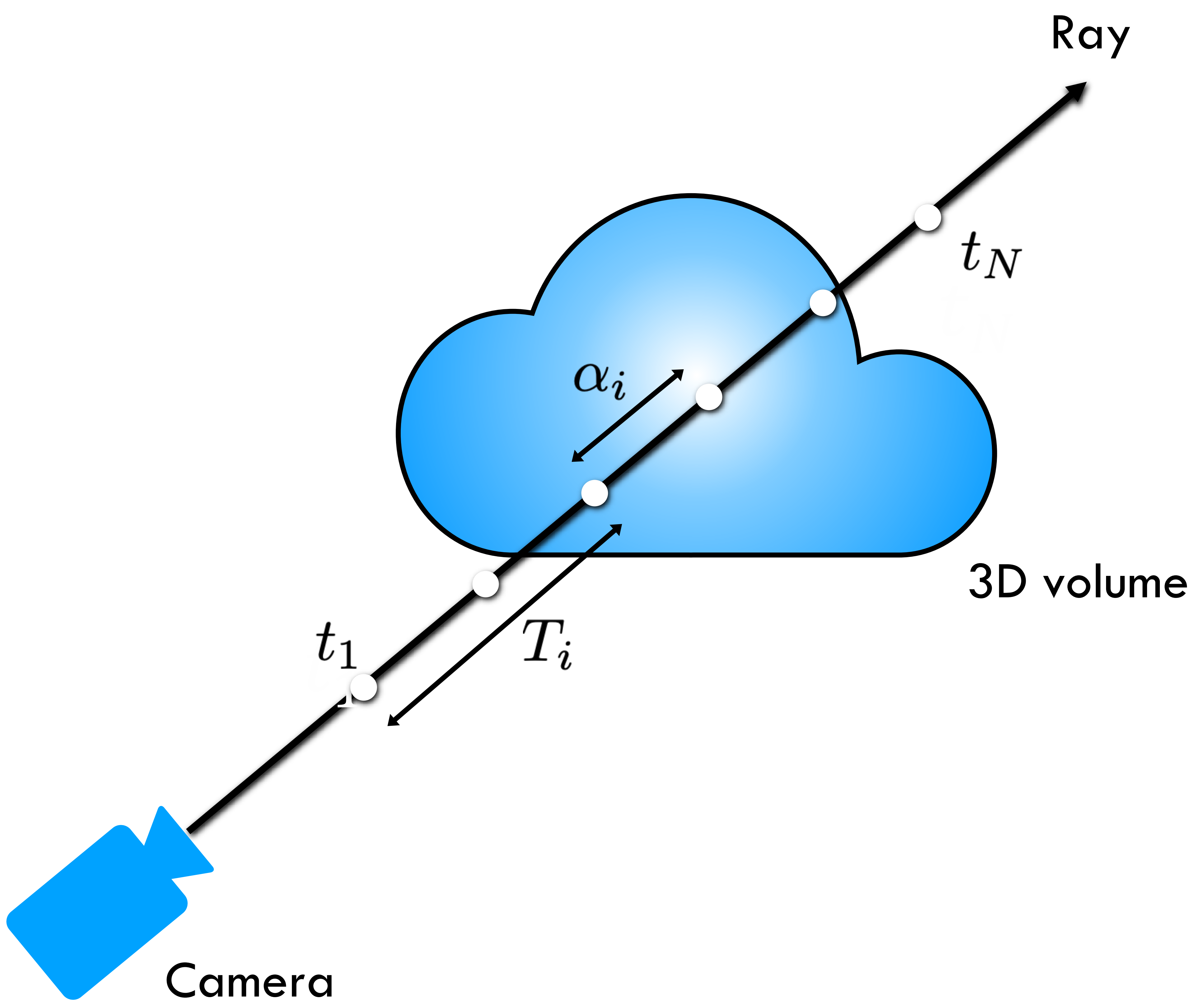
$$C \approx \sum_{i=1}^{N} \underbrace{T_i \boxed{\alpha_i}}_{\text{weights}} \underbrace{c_i}_{\text{colors}}$$

How much light is blocked earlier along ray:

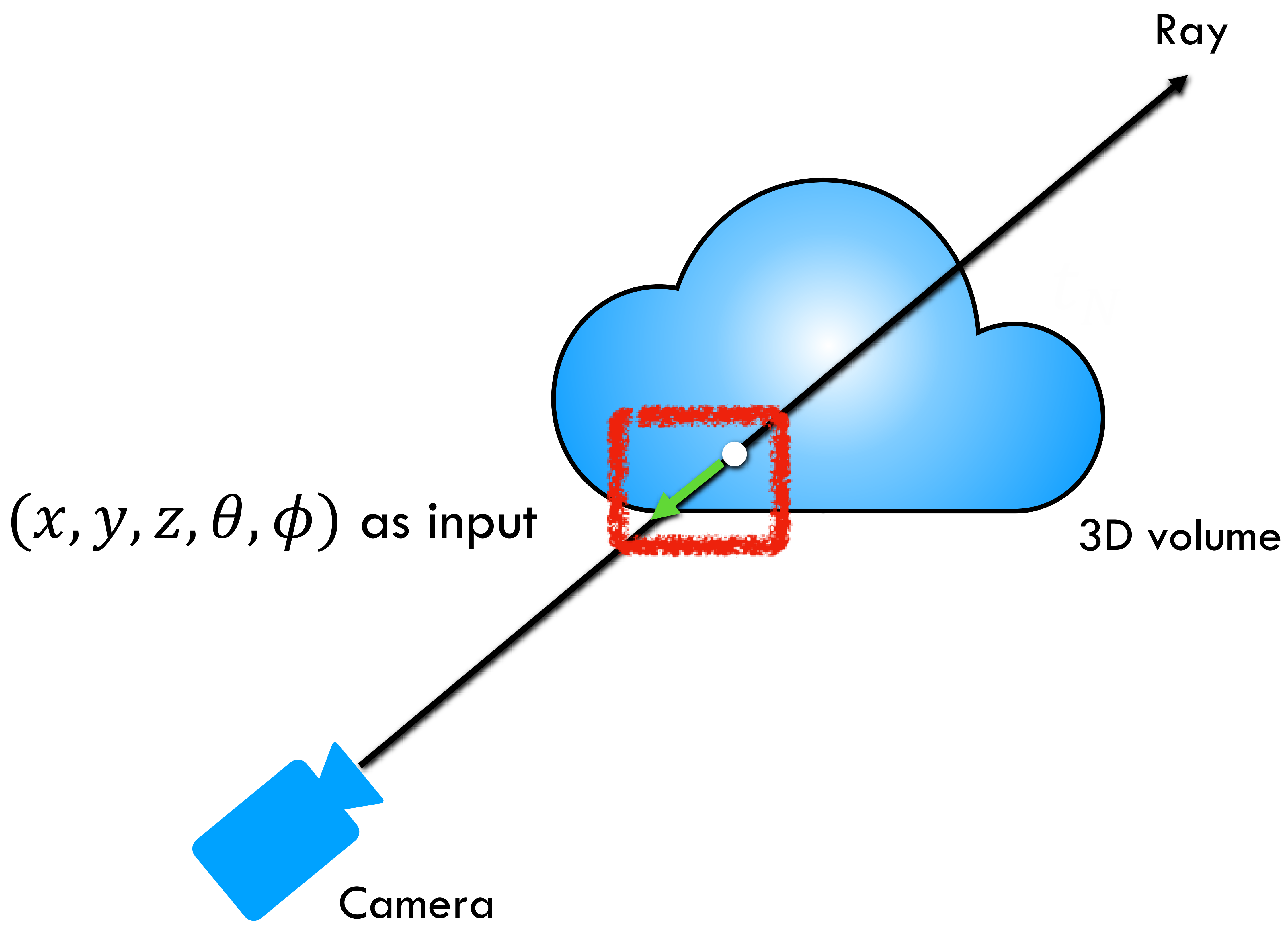
$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

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

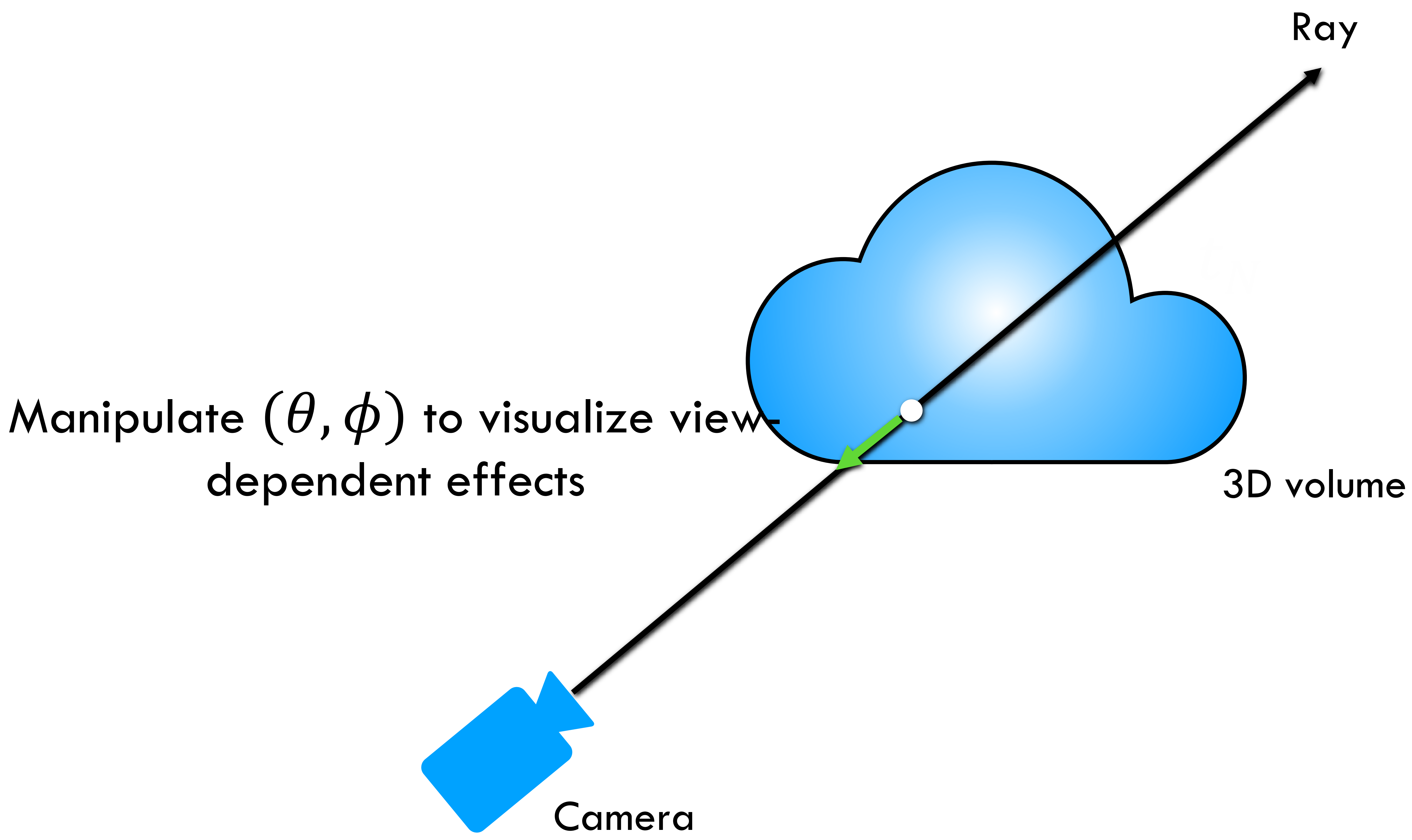
$$\boxed{\alpha_i} = 1 - e^{-\sigma_i \delta t_i}$$



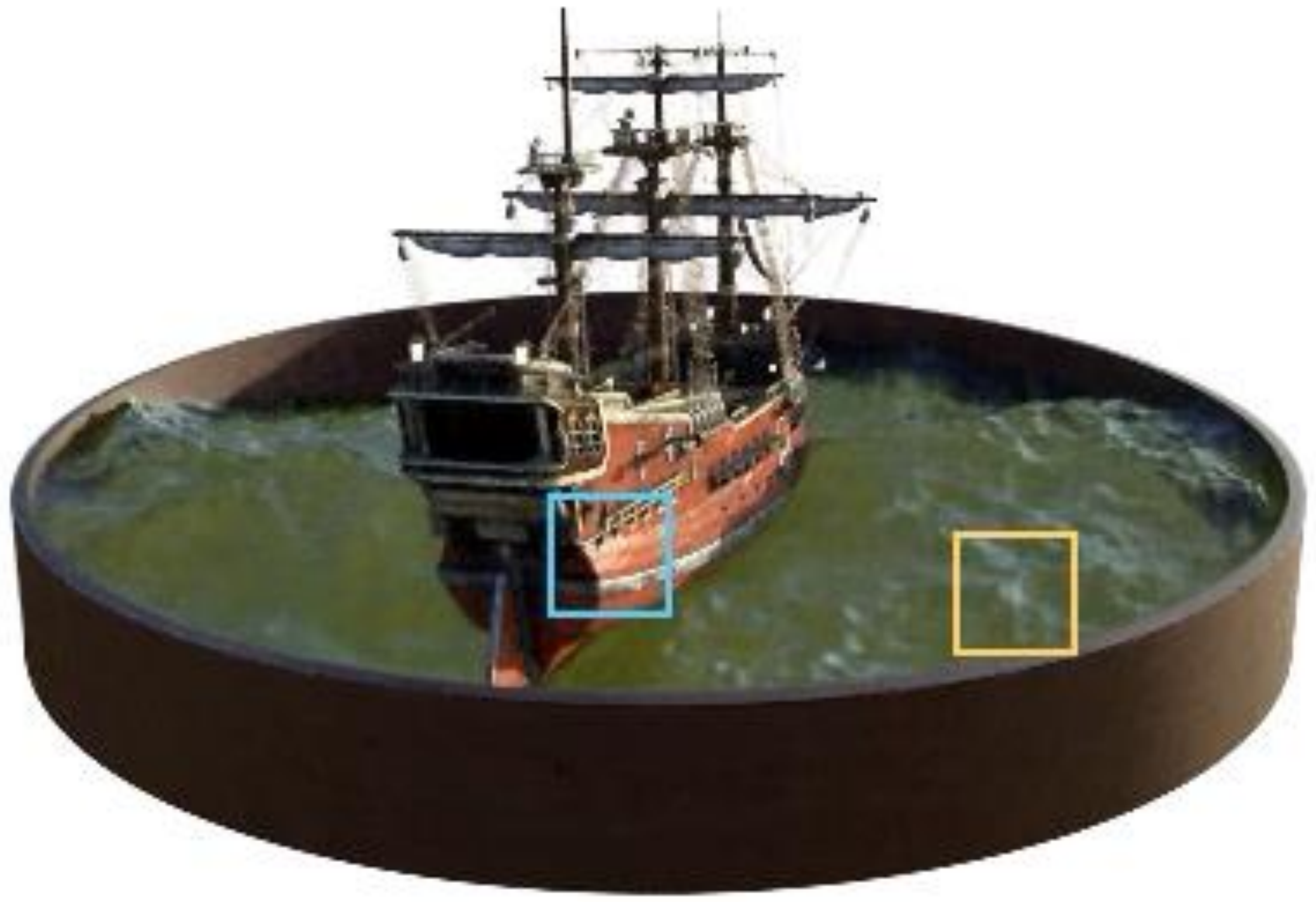
Viewing directions as input

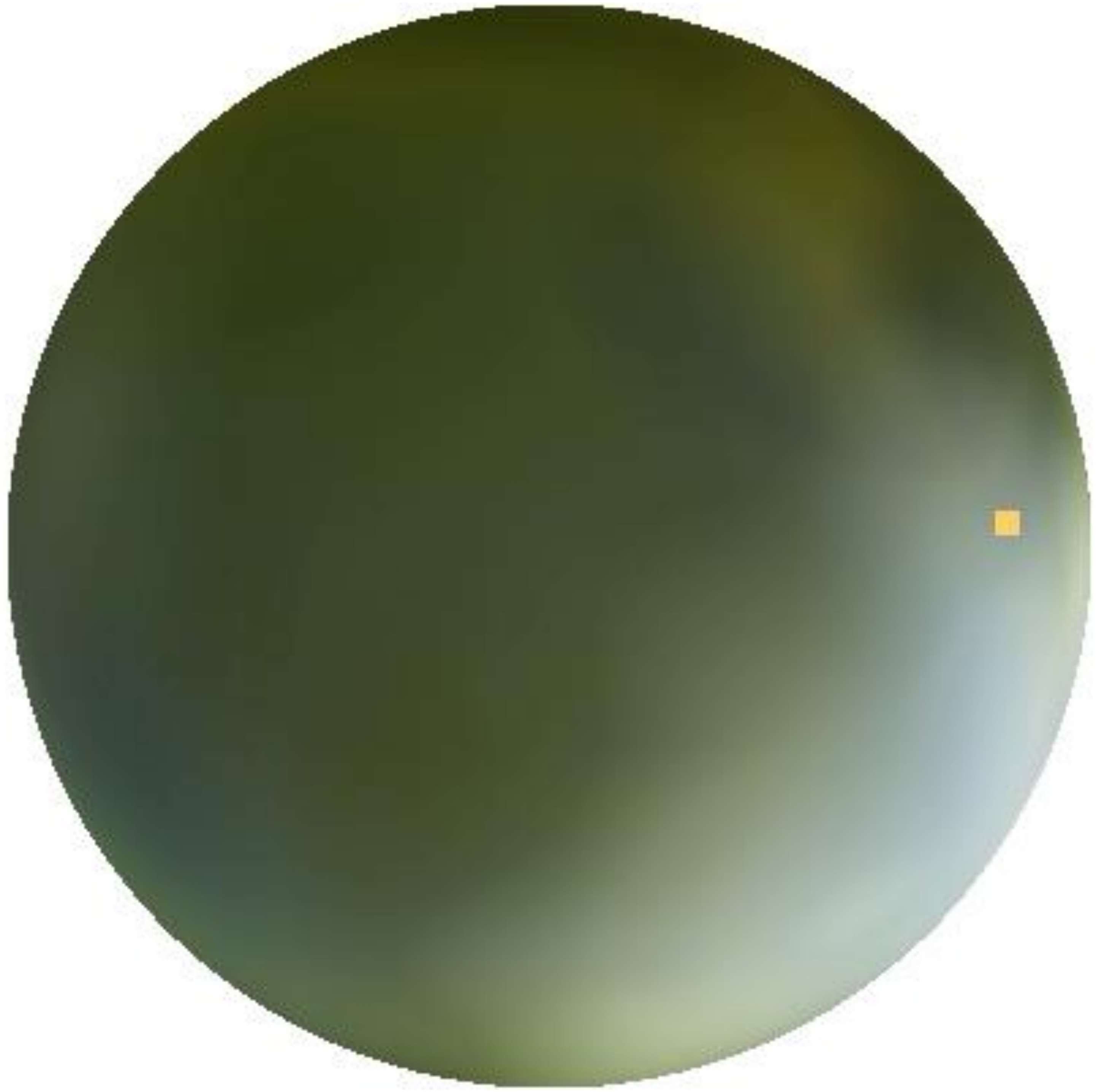


Viewing directions as input



Viewing directions as input





Radiance distribution for point on side of ship

Radiance distribution for point on water's surface

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.

weights

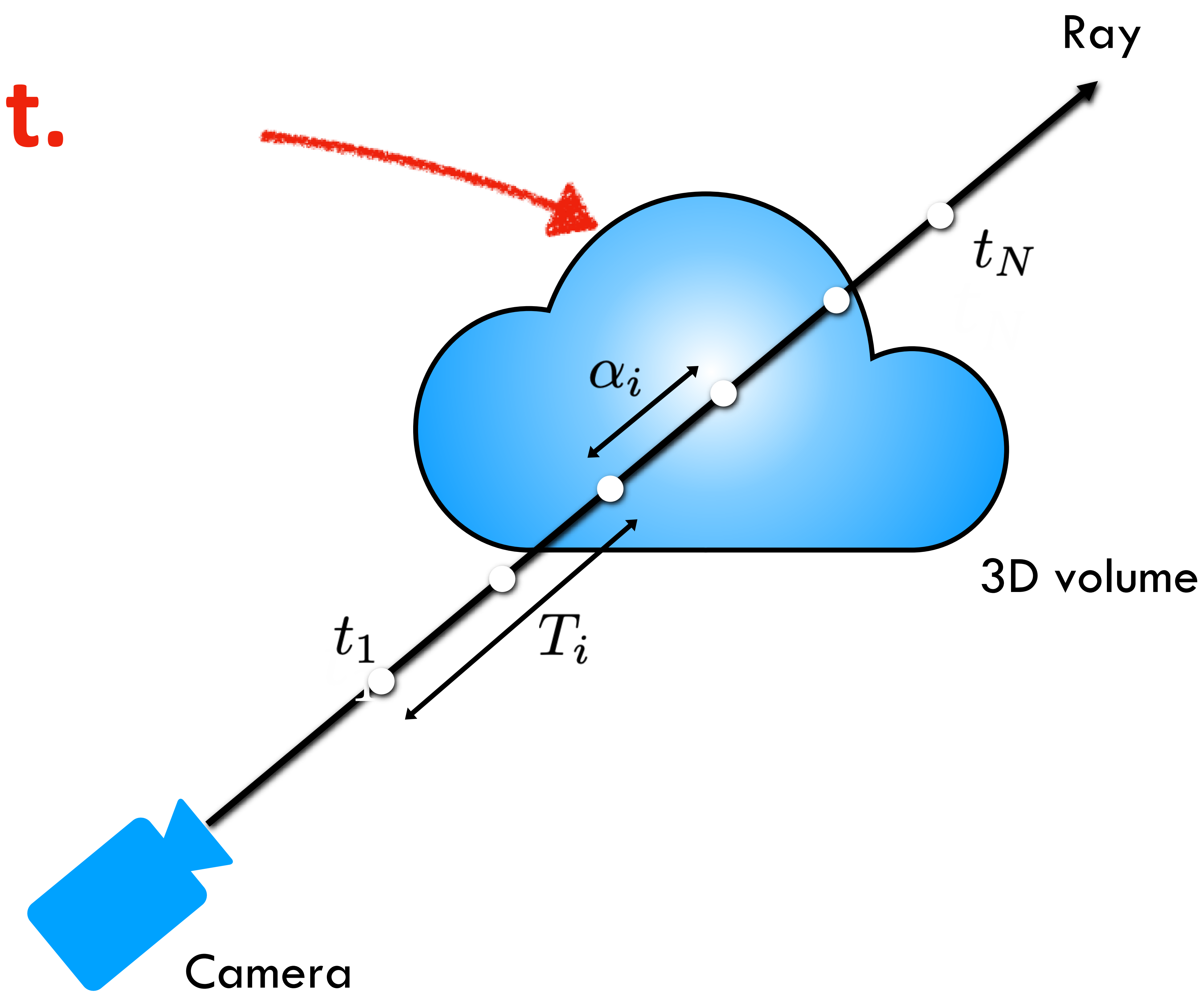
colors

How much light is blocked earlier along ray:

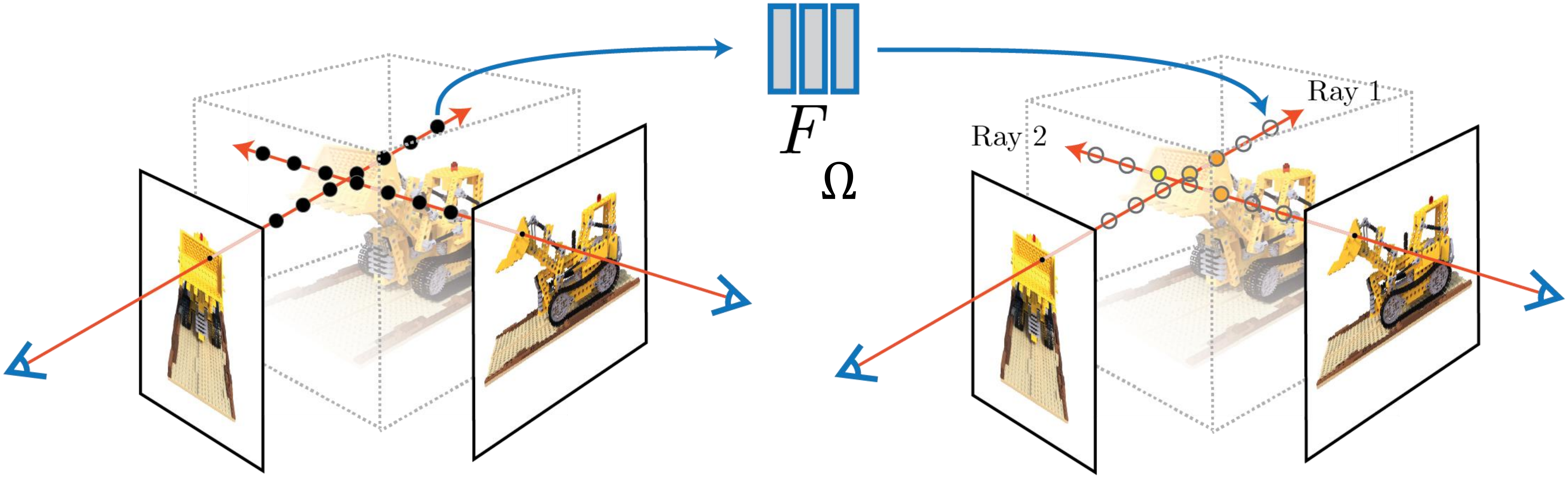
$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

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

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



Optimize with gradient descent on rendering loss



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

Optimize with gradient descent on rendering loss

Any differentiable scene representation F_Ω
could be used here

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

Training network to reproduce all input views of the scene

Naive implementation produces blurry results

NeRF (Naive)

Naive implementation produces blurry results

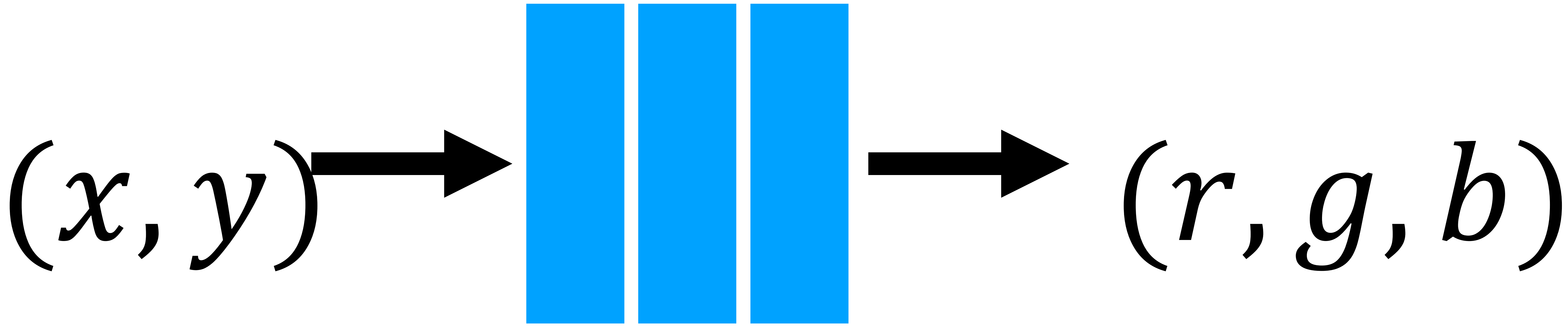
NeRF (Naive)

NeRF (with positional encoding)

Challenge:

How to get MLPs to represent higher frequency functions?

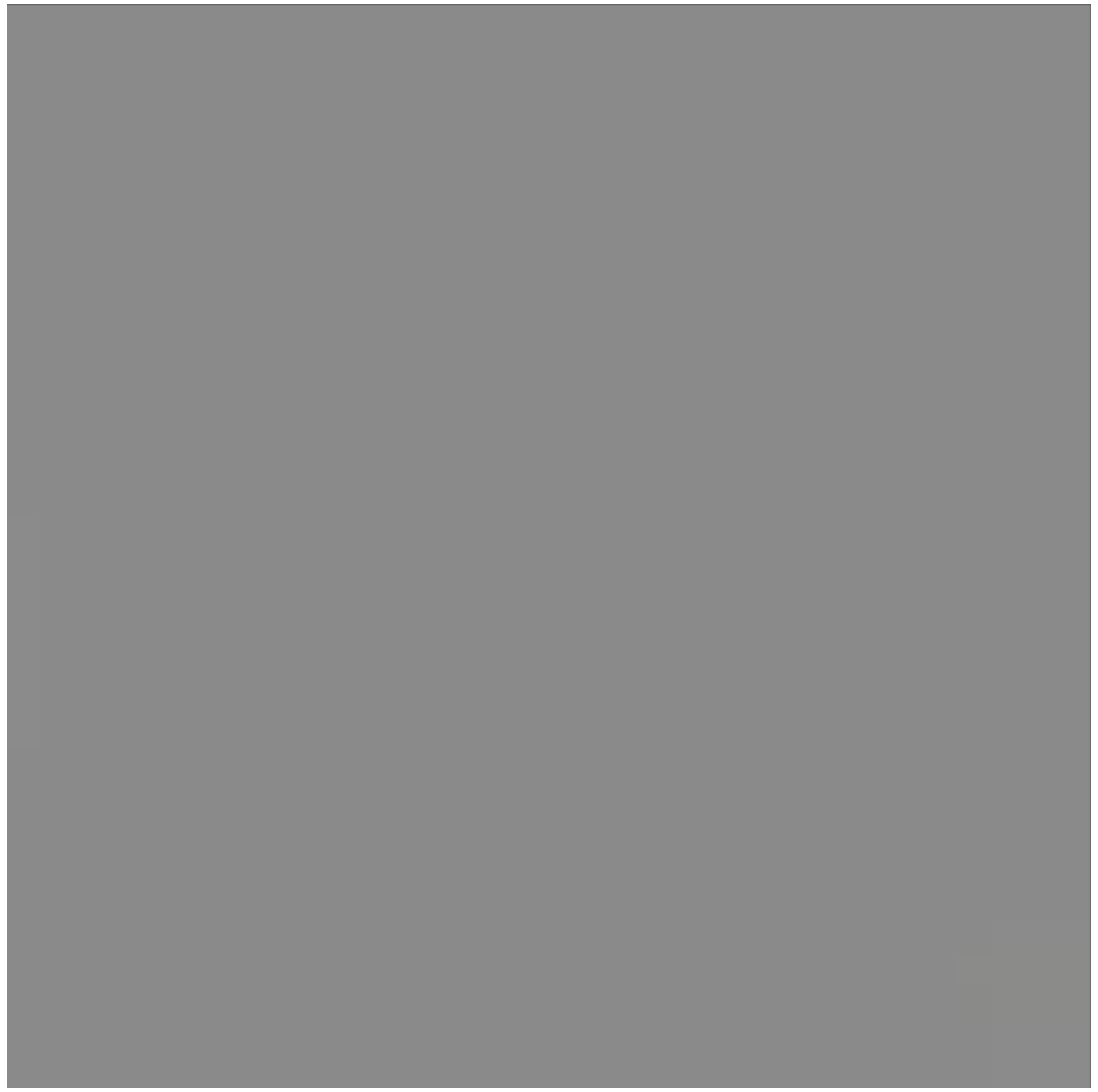
Simpler toy problem: memorizing a 2D image



Simple trick enables network to memorize images

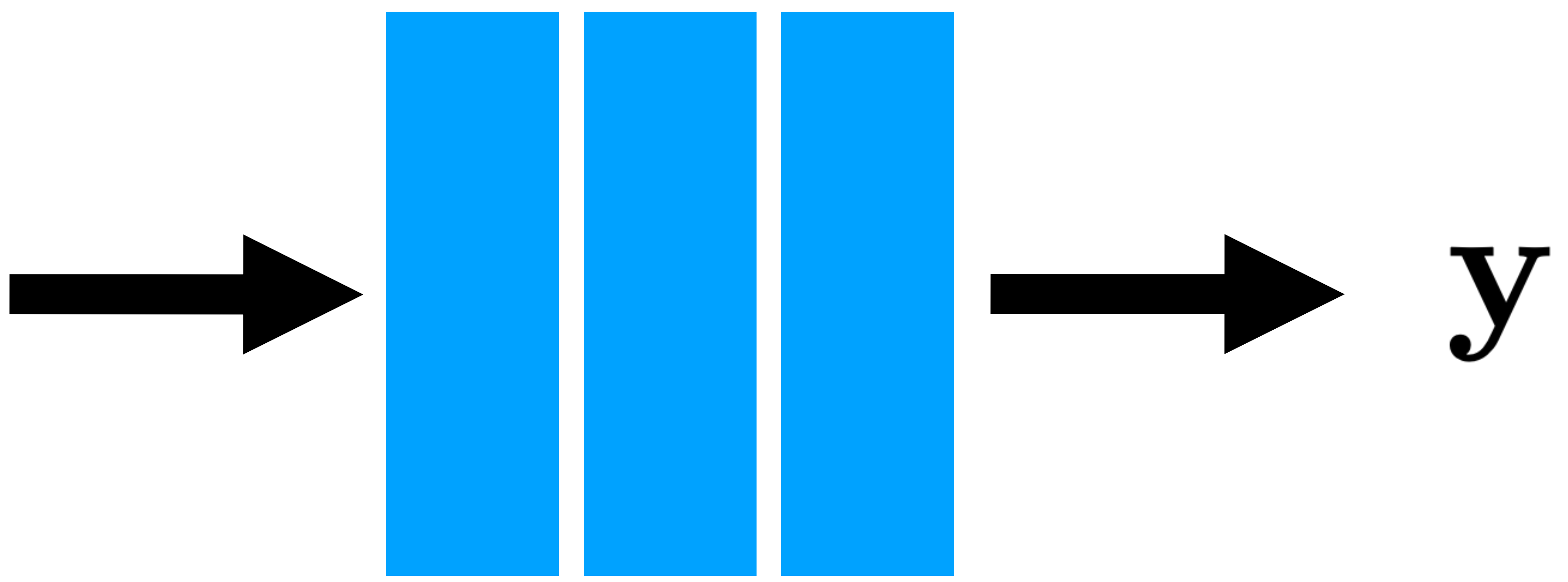
Ground truth image

Standard fully-connected net



Positional encoding

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



Training networks ≈ kernel regression

- Recent ML theory work shows that training neural network with gradient descent becomes the same as performing kernel regression as the width of each layer goes to infinity
- Can examine corresponding kernel function (the neural tangent kernel) to see why adding Fourier feature mapping allows MLPs to represent high frequency functions

Jacot et al., Neural Tangent Kernel: Convergence and generalization in neural networks, NeurIPS 2018

Kernel regression

- Method for fitting a continuous function to a set of data points $\{(x_i, y_i)\}$
- High level: add up a set of blobs (kernel functions), one centered at each input point, each with its own weight
- Weights are optimal in a least-squares sense: $\min_w \sum_i \| y_i - \hat{f}_w(x_i) \|^2$

$$\hat{f}_w(x) = \sum_{i=1}^{n} w_i k(x - x_i)$$

Blob centered at training input point x_i

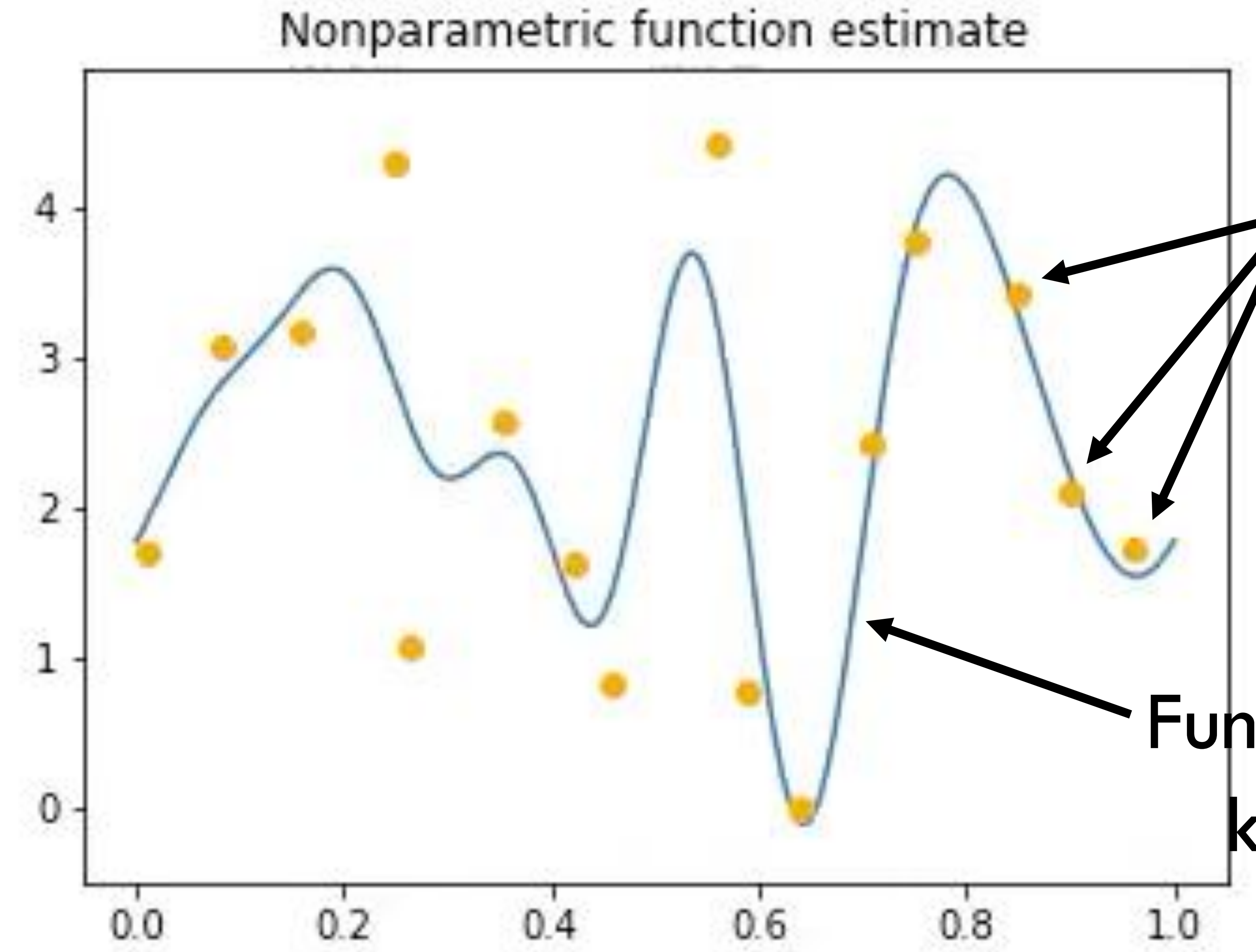
Estimated function

Weight corresponding to blob centered at x_i

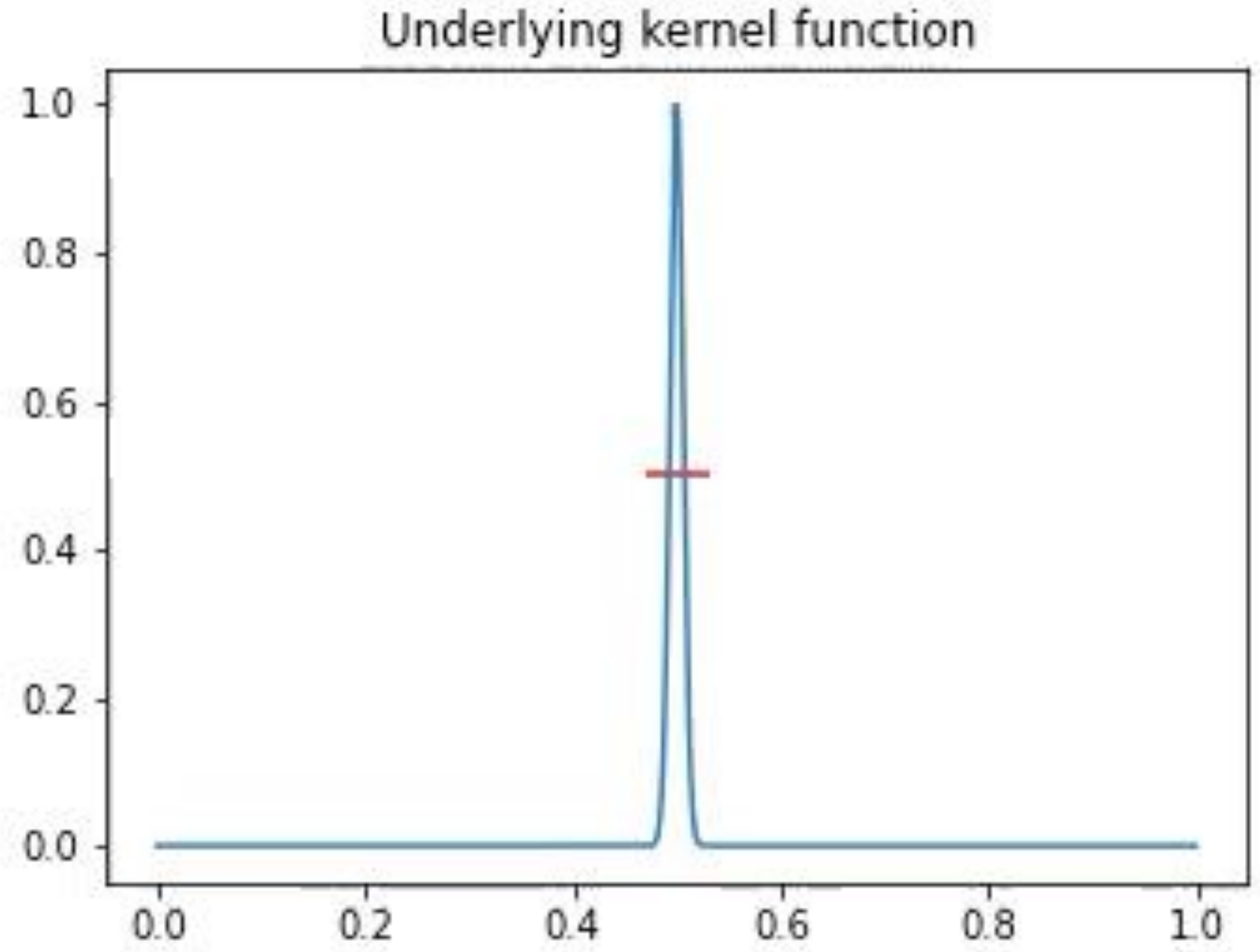
“Width” of kernel function is critical

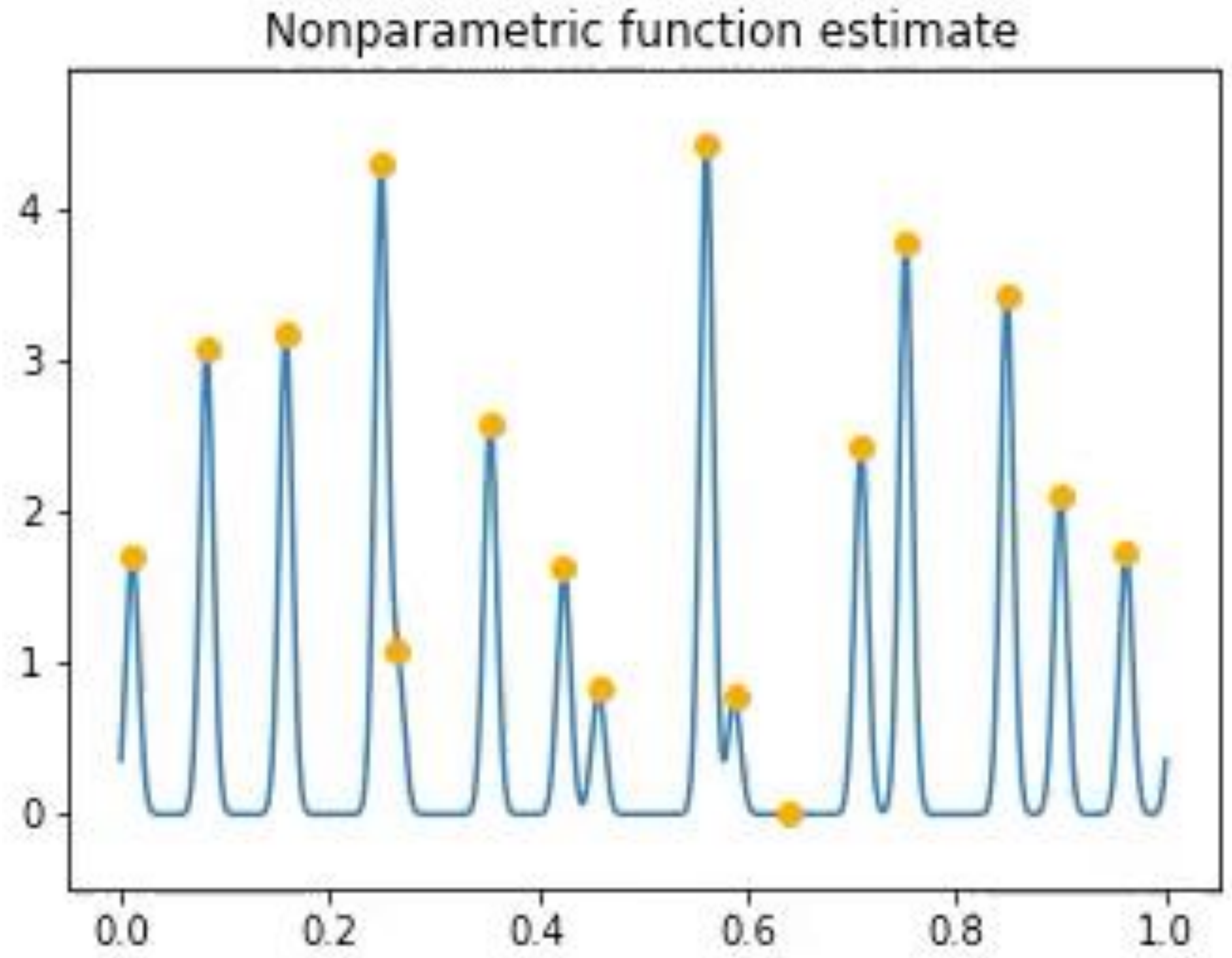
- If the kernel function is too wide, reconstruction is too smooth. If it’s too skinny, reconstruction does not interpolate correctly.
- Similar to picking the right reconstruction filter bandwidth in signal processing to avoid either blurring or aliasing.

"Width" of kernel function is critical



“Width” of kernel function is critical



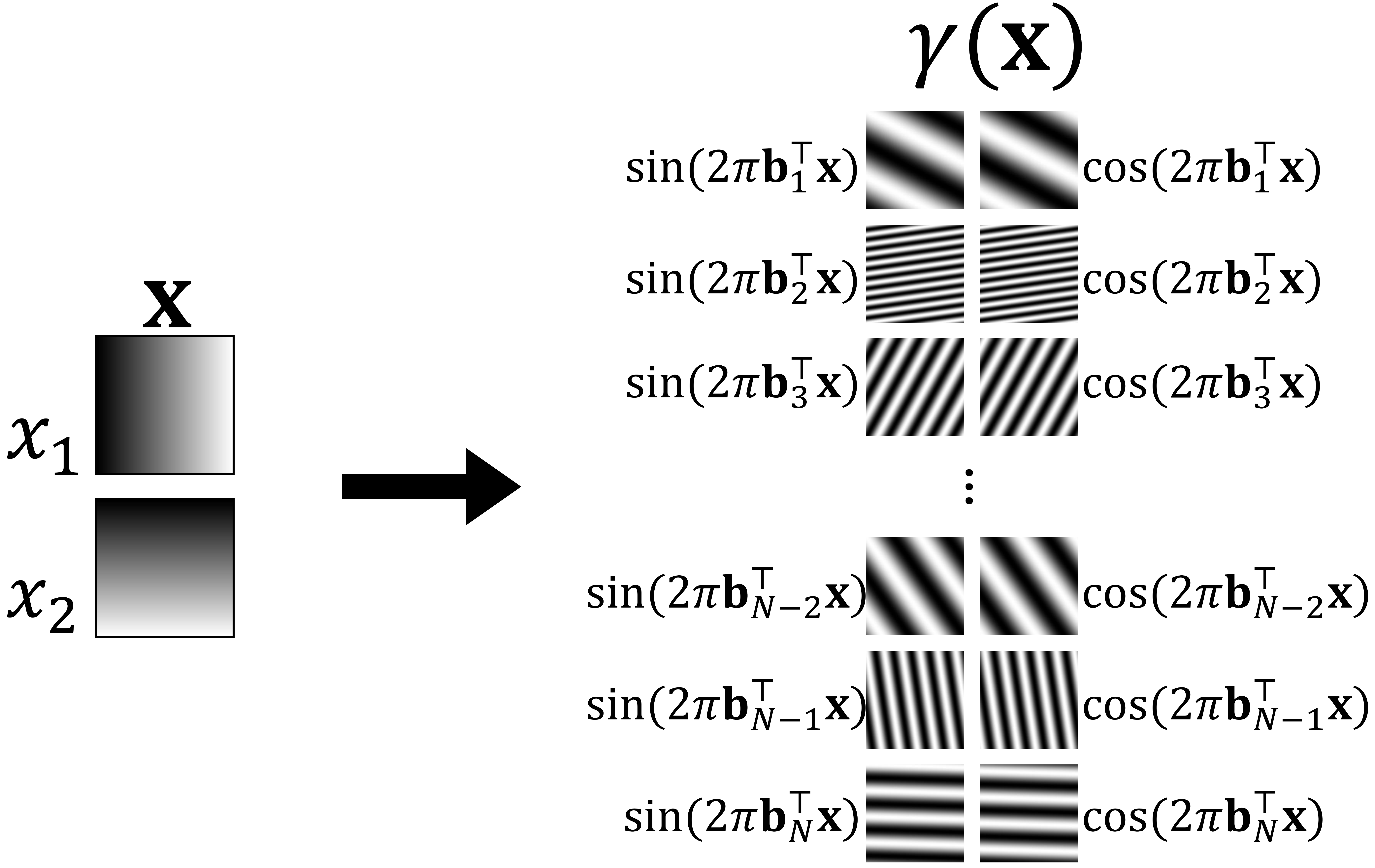


Training networks ≈ kernel regression

- Recent ML theory work shows that training neural network with gradient descent becomes the same as performing kernel regression as the width of each layer goes to infinity
- Using a Fourier feature mapping changes the corresponding kernel function (the neural tangent kernel), allowing MLPs to represent higher frequency functions

Jacot et al., Neural Tangent Kernel: Convergence and generalization in neural networks, NeurIPS 2018

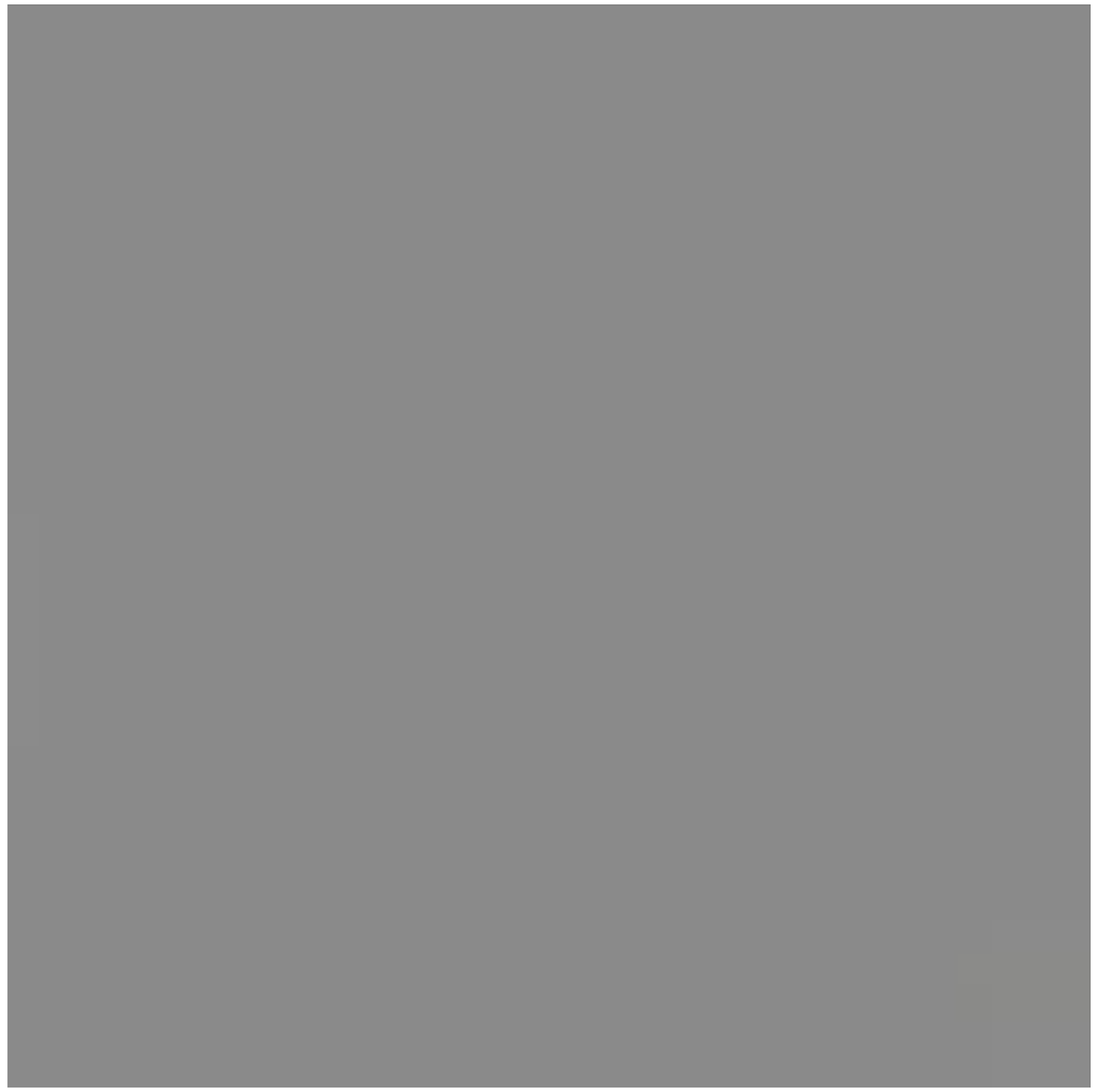
Fourier feature mapping: simple 2D example



Simple trick enables network to memorize images

Ground truth image

Standard fully-connected net



With “encoding”

Positional encoding also directly improves our scene representation!

NeRF (Naive)

NeRF (with positional encoding)

Check out https://bmild.github.io/fourfeat/ for more details

More detailed and consistent than prior work that represents scene as function encoded by MLP

SRN [Sitzmann et al. 2019]

NeRF

Nearest Input

NeRF encodes convincing view-dependent effects using directional dependence

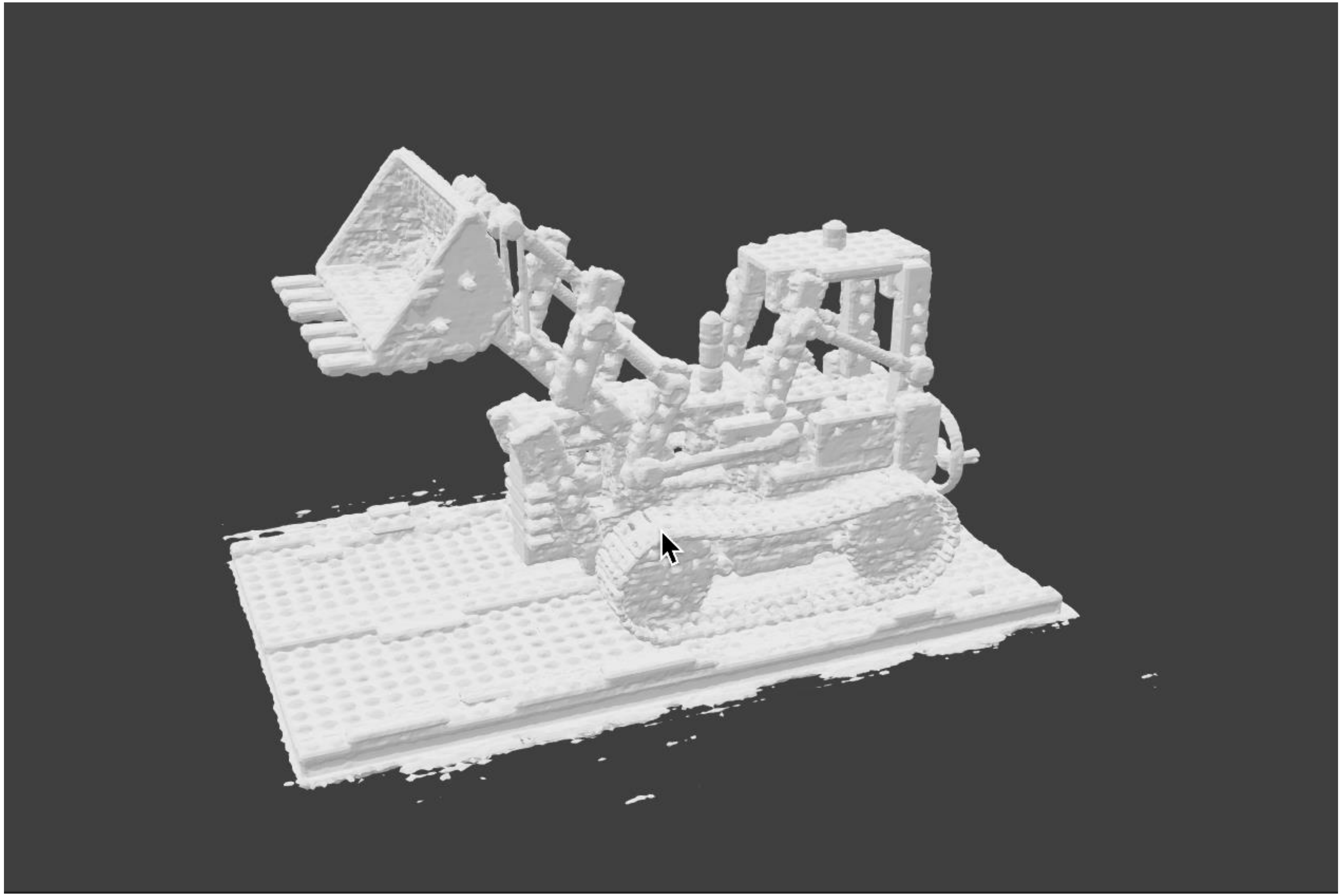
NeRF encodes convincing view-dependent effects using directional dependence

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



NeRF: 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)
- Apply positional encoding before passing coordinates into network to recover high frequency details