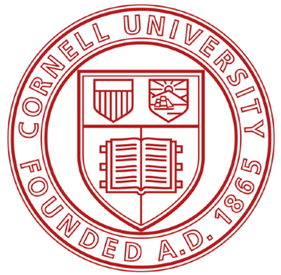
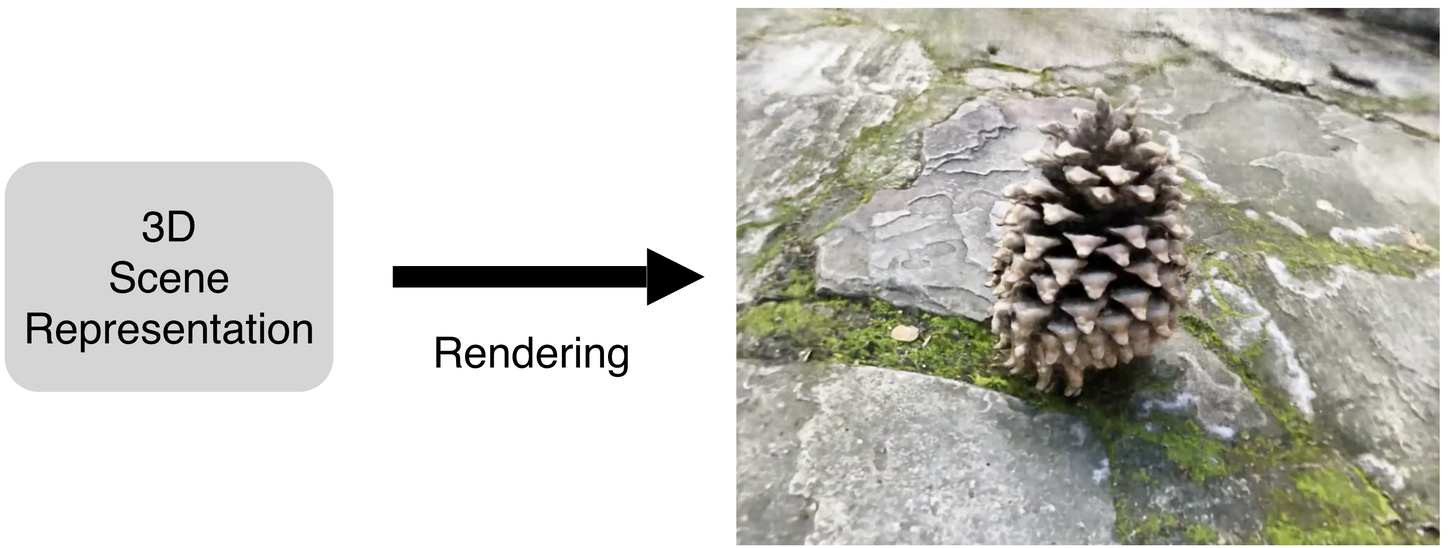
Lecture 22: Neural fields and learning-based 3D

CS 5670: Introduction to Computer Vision



Includes slides from V. Sitzmann and B. Mildenhall

Rendering in computer graphics



Source: N. Snavely, adapted from Pratul Srinivasan

Rendering in computer graphics

Inverse Rendering

3D
Scene
Representation

Source: N. Snavely, adapted from Pratul Srinivasan

Rendering in computer graphics

Inverse Rendering

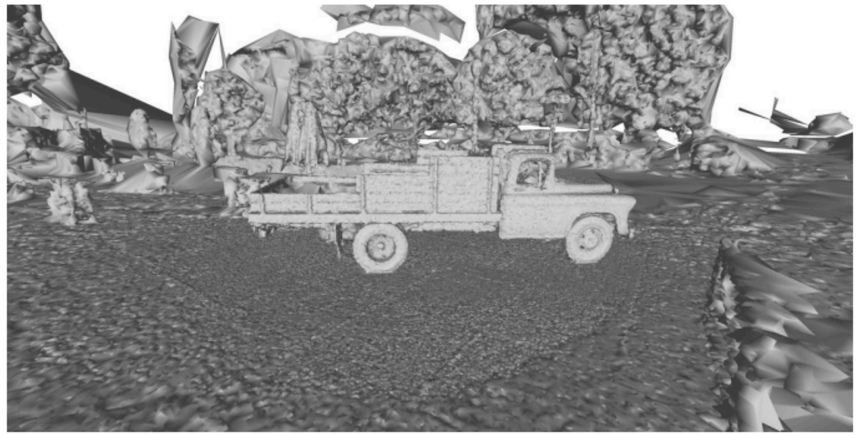
3D
Scene
Representation

Rendering

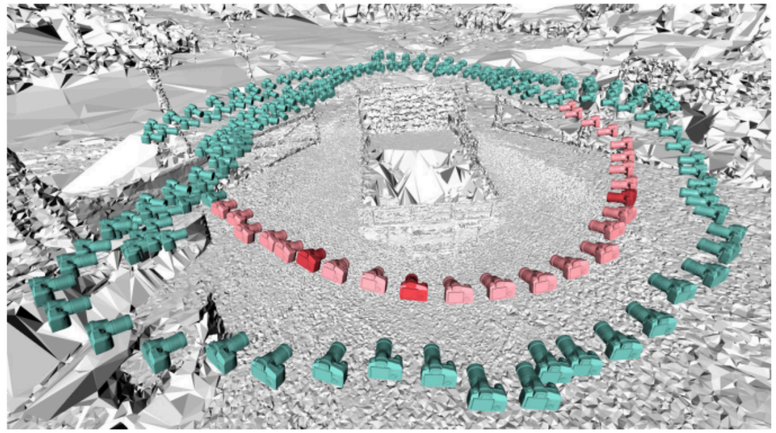
Source: N. Snavely, adapted from Pratul Srinivasan

Idea #1: Image-based rendering

Point cloud



Proxy geometry (a mesh)

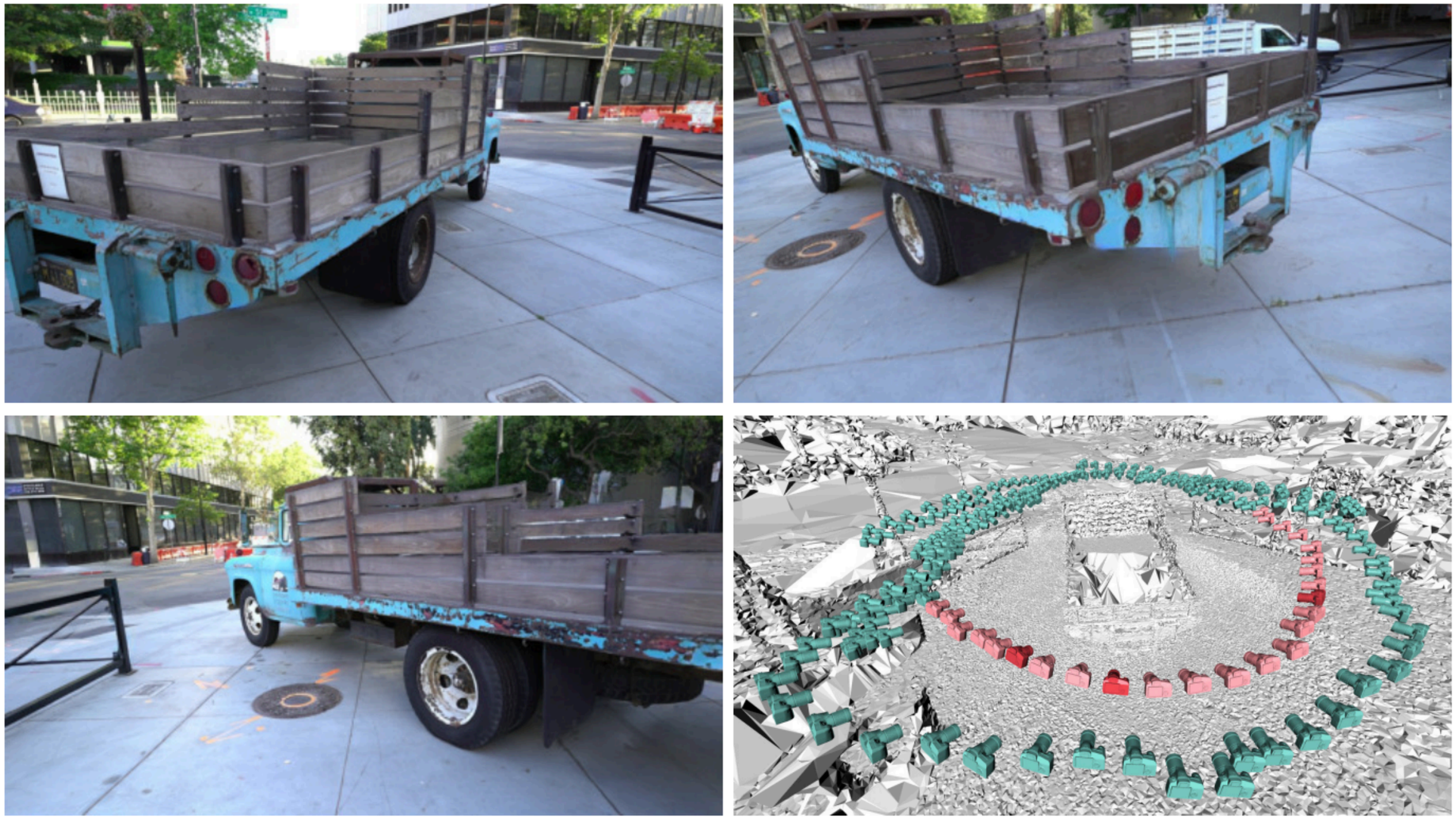


Reconstruction

To synthesize a new view, select colors from existing views using proxy geometry.

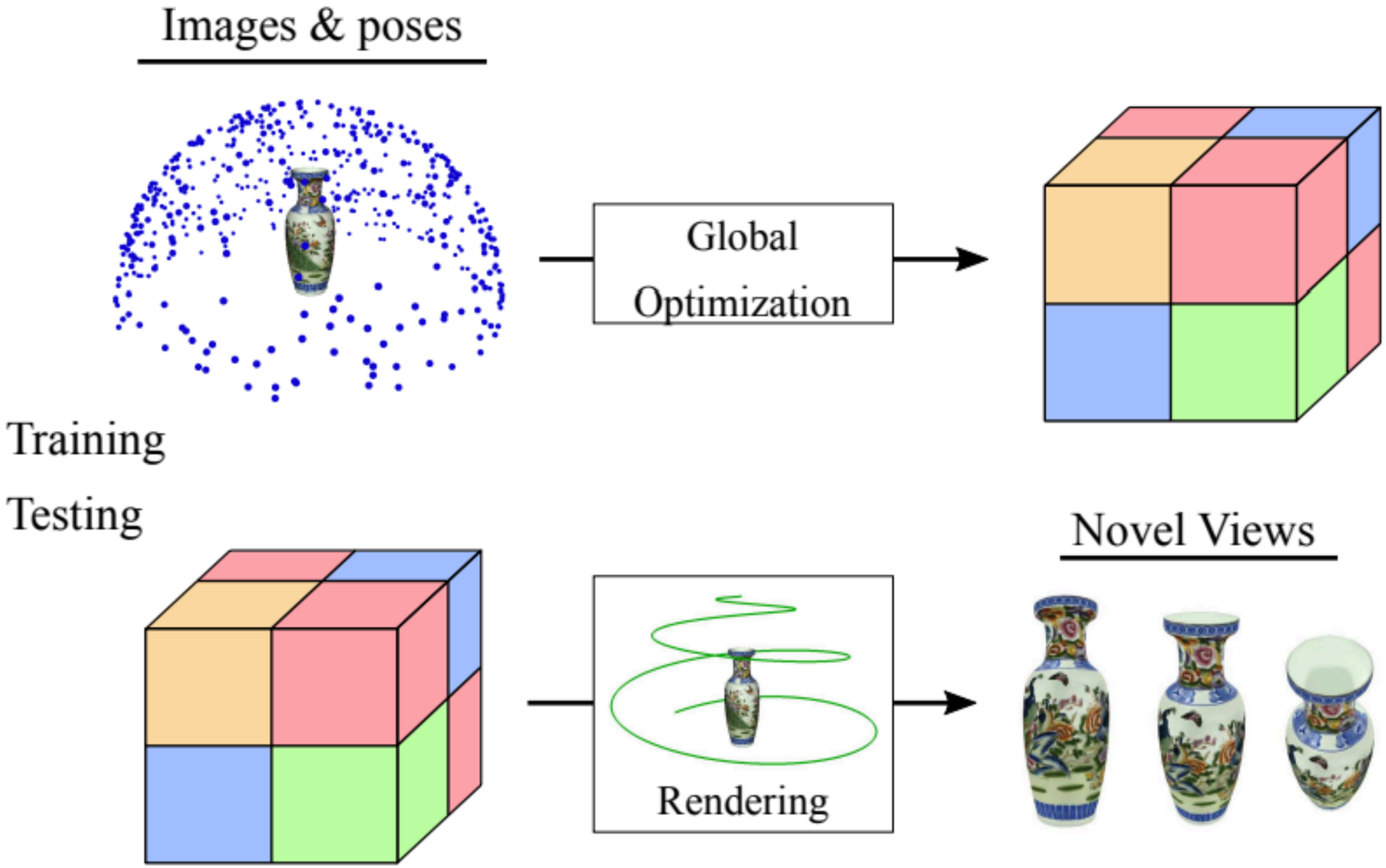
Figure source: [Riegler and Koltun, 2020]

Idea #1: Image-based rendering



[Riegler and Koltun, 2020]

Idea #2: voxel representation



[Source: Sitzmann et al., "DeepVoxels", 2019]

Idea #2: voxel representation

Input views

Position Viewing direction Color Density

$$V[x, y, z, \theta, \phi] = (R, G, B, \sigma)$$

Idea #2: voxel representation

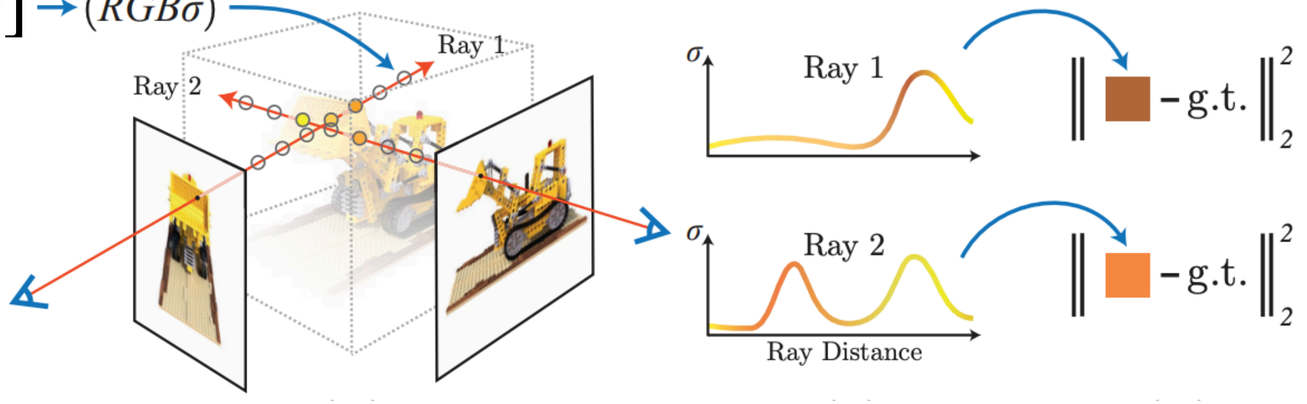
Input views

Position Viewing direction Color Density

$$V[x, y, z, \theta, \phi] = (R, G, B, \sigma)$$

Training:

$$V[x, y, z, \theta, \phi] \rightarrow (RGB\sigma)$$



Problem: A huge table! $\mathcal{O}(D^3A^2)$

Idea #3: neural radiance field (NeRF)

Input views

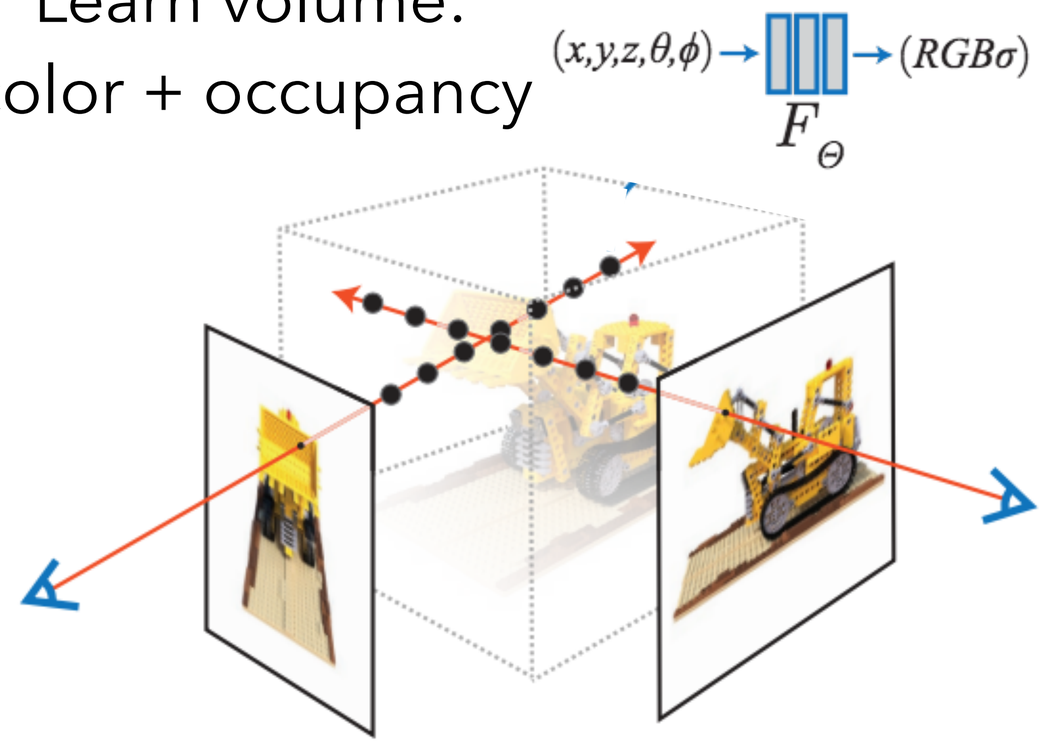
$$\rightarrow \quad F_\Theta(x, y, z, \theta, \phi) = (R, G, B, \sigma)$$

- Represent using a **neural radiance field**.
- Function that maps a (x, y, z, θ, ϕ) to a color and density.
- Typically parameterized as a multi-layer perceptron (MLP)
- Goal: find parameters Θ for MLP that explain the images

Idea #3: neural radiance field (NeRF)

3D scene

Learn volume:
color + occupancy



Viewpoints

[Mildenhall*, Srinivasan*, Tanick*, et al., Neural radiance fields, 2020

Volume Rendering

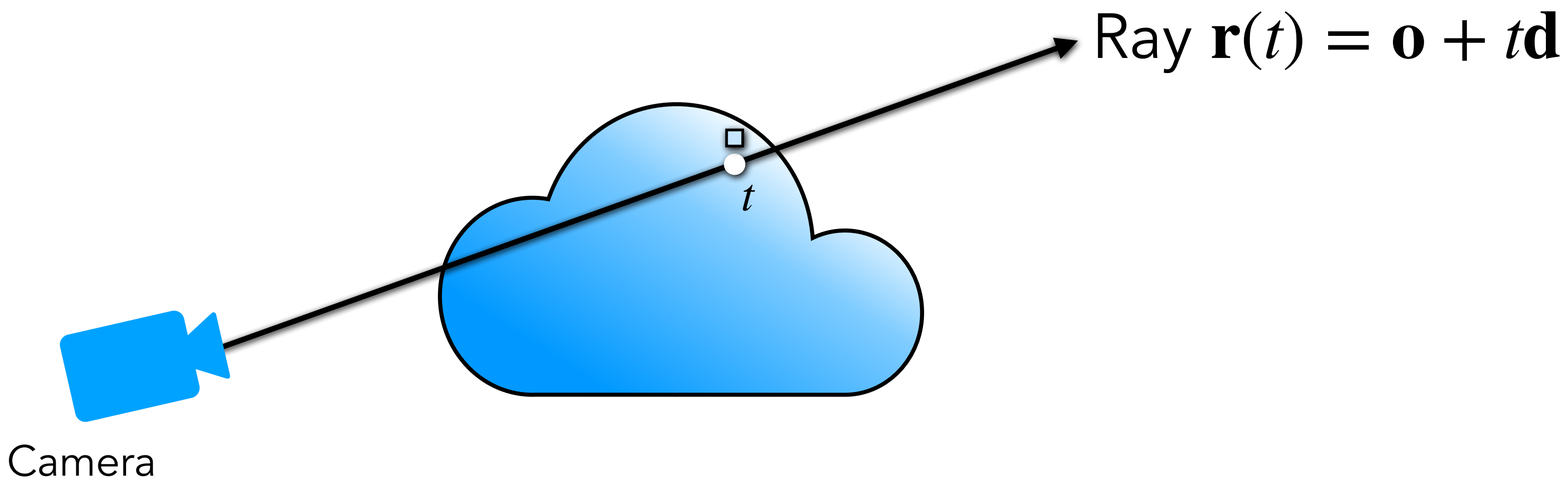
Scene is a cloud of tiny colored particles

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

Neural Radiance Fields, Mildenhall et al. 2021

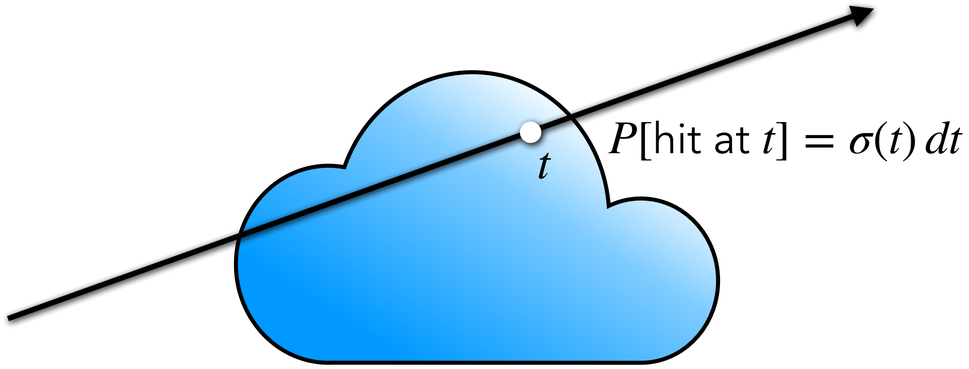
Slide credit: Ben Mildenhall, via V. Sitzmann

Volume Rendering



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

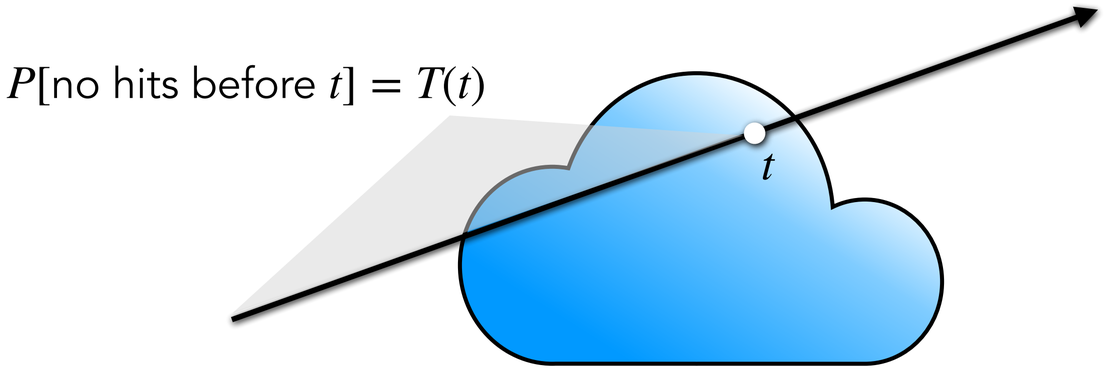
Volume Rendering



Probability that ray stops in a small interval around t is $\sigma(t)\,dt$.

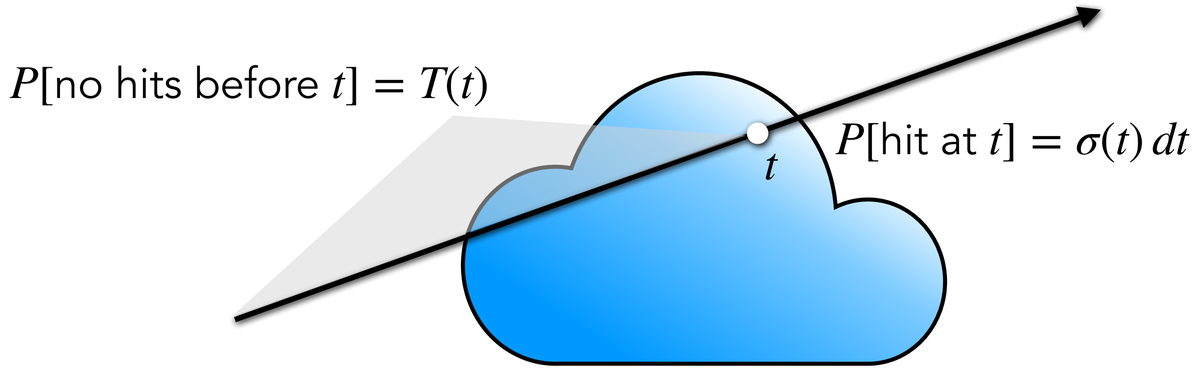
σ is known as the **Volume Density.**

Volume Rendering



To determine if t is the *first* hit, need to know $T(t)$:
probability that the ray didn't hit any particles earlier.
$T(t)$ is called **Transmittance.**

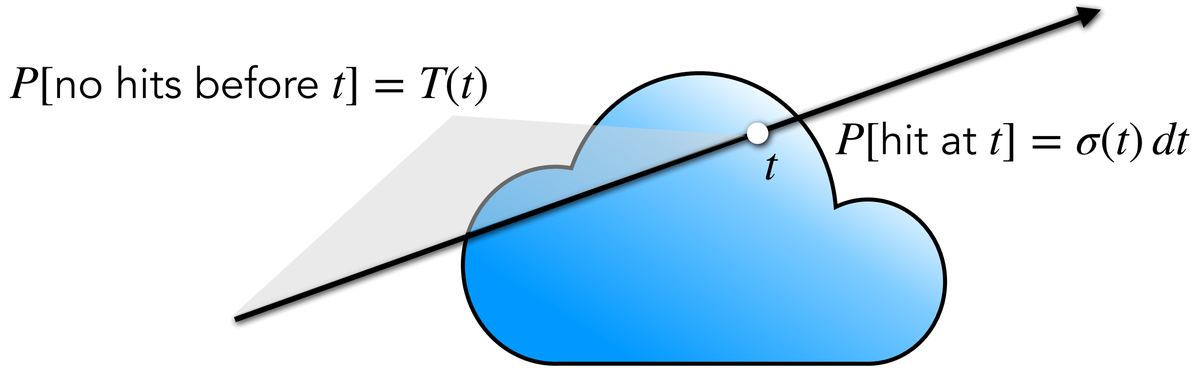
Volume Rendering



σ and T are related via

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

Volume Rendering



σ and T are related via

$$P[\quad T(t+dt) \quad] = P[\quad T(t) \quad] \times P[\quad 1 - \sigma(t)dt \quad]$$

Volume Rendering

Can show: no hits before t is equal to integral over density up until t.

$$T(t) = \exp\left(-\int_0^t \sigma(a)da\right)$$

$1 - T(t)$ can be seen as cumulative distribution function of probability that ray hits something before reaching t.

Then $T'(t) = T(t)\sigma(t)$ is probability that ray stops *exactly* at t .

Volume Rendering

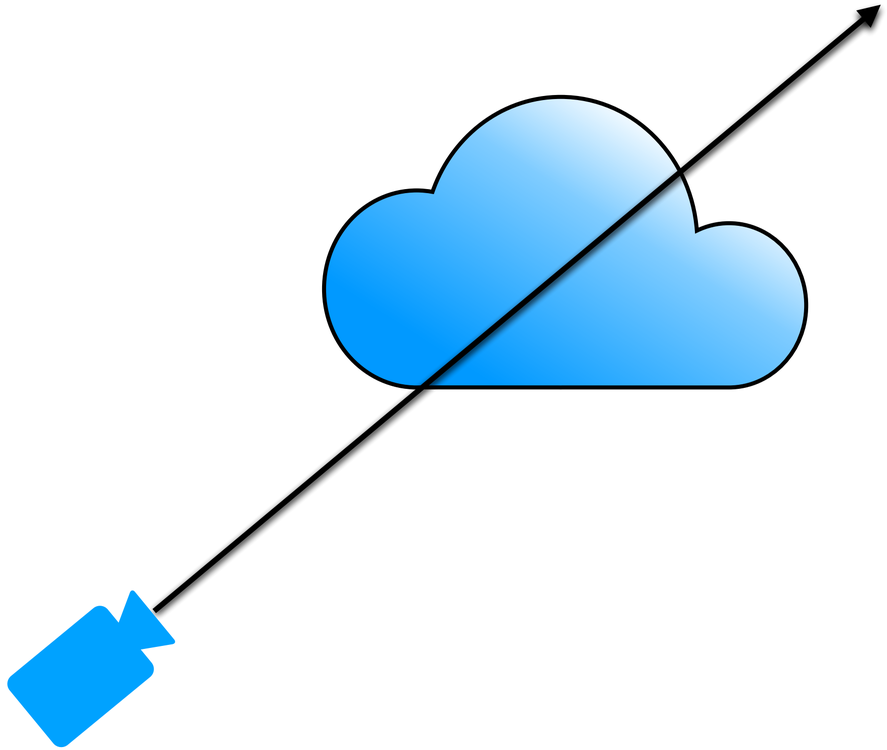
Then $T'(t) = T(t)\sigma(t)$ is probability that ray stops *exactly* at t .

So the expected color returned by the ray will be

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

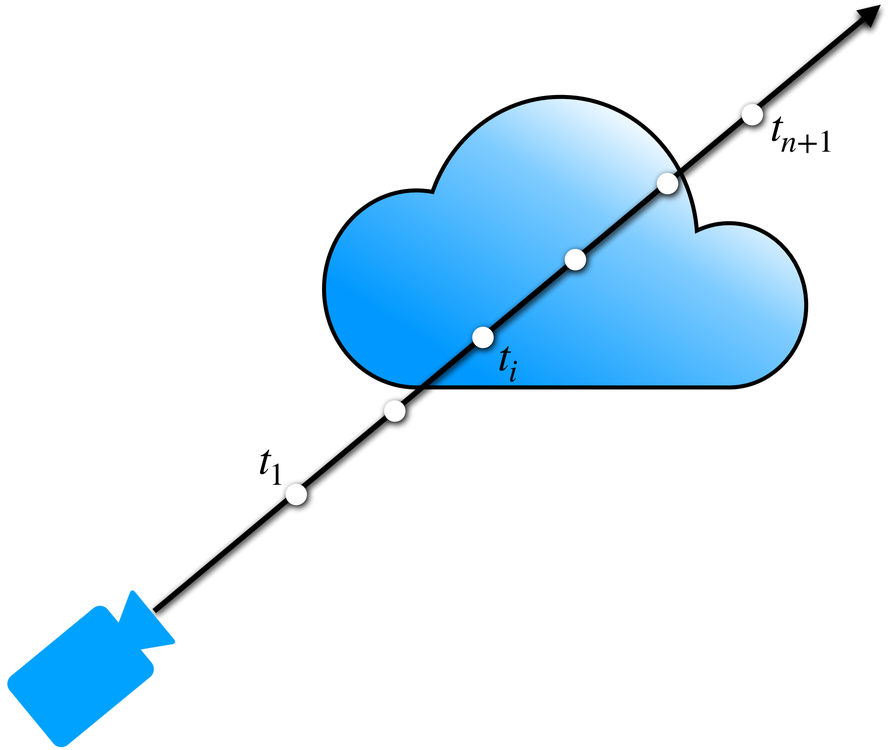
Note the nested integral!

Approximating the nested integral



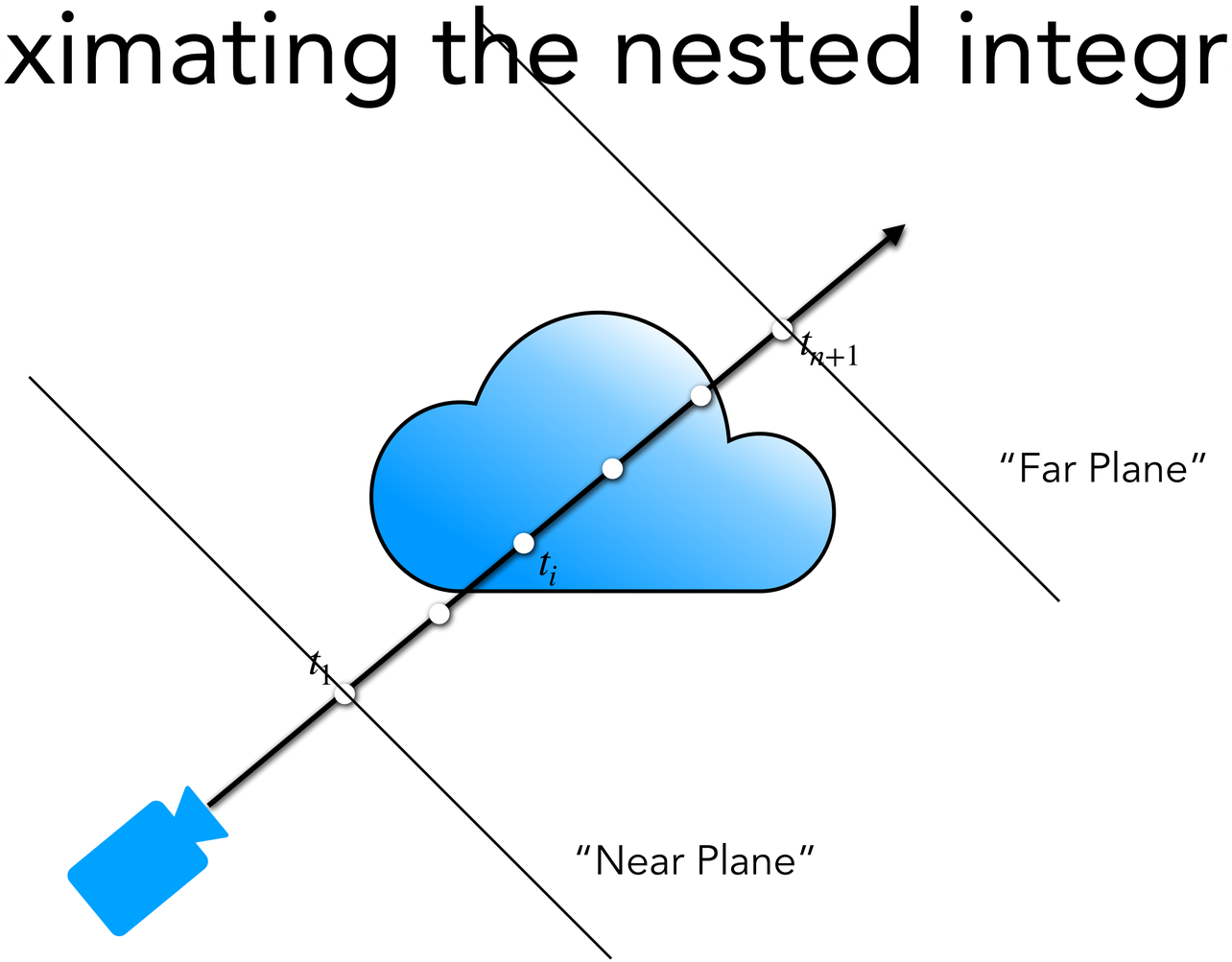
We use quadrature to approximate the nested integral,

Approximating the nested integral



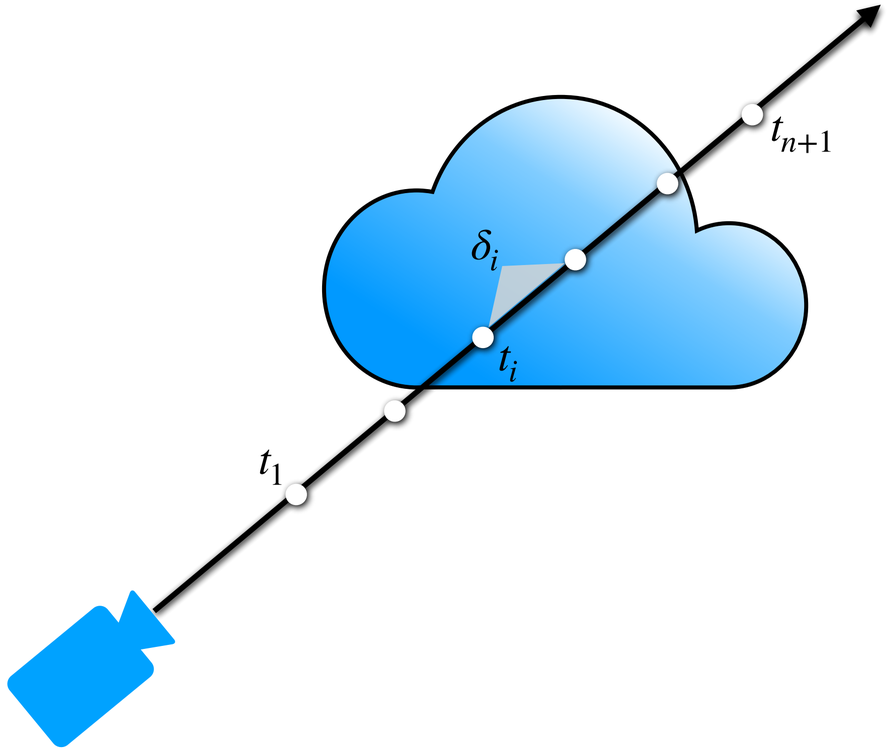
We use quadrature to approximate the nested integral,
splitting the ray up into n segments with endpoints $\{t_1, t_2, \ldots, t_{n+1}\}$

Approximating the nested integral



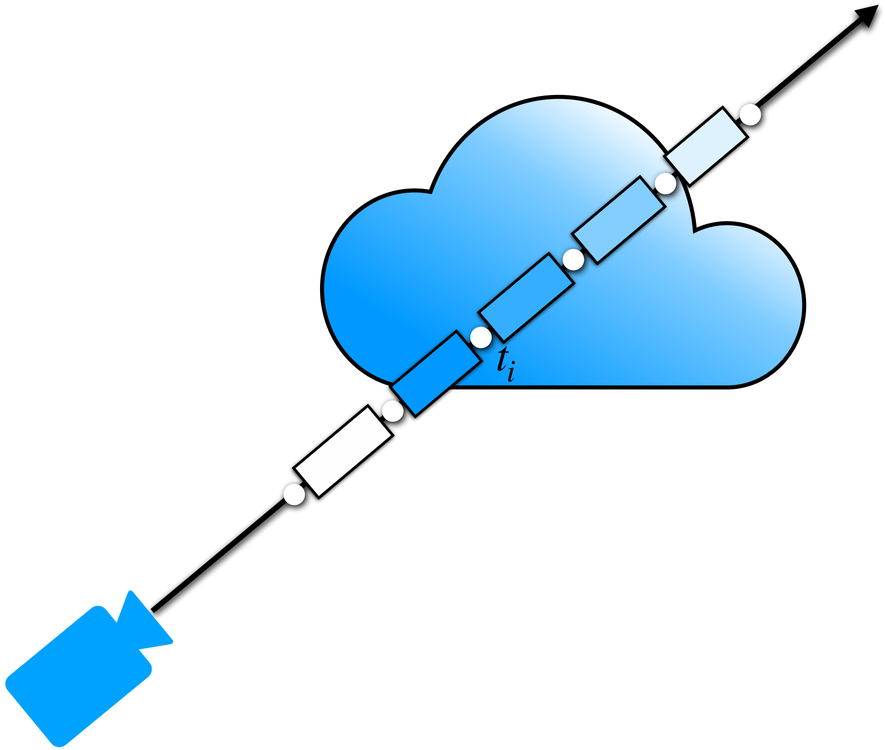
We use quadrature to approximate the nested integral,
splitting the ray up into n segments with endpoints $\{t_1, t_2, \ldots, t_{n+1}\}$

Approximating the nested integral



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

Summary: volume rendering integral estimate

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

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

colors

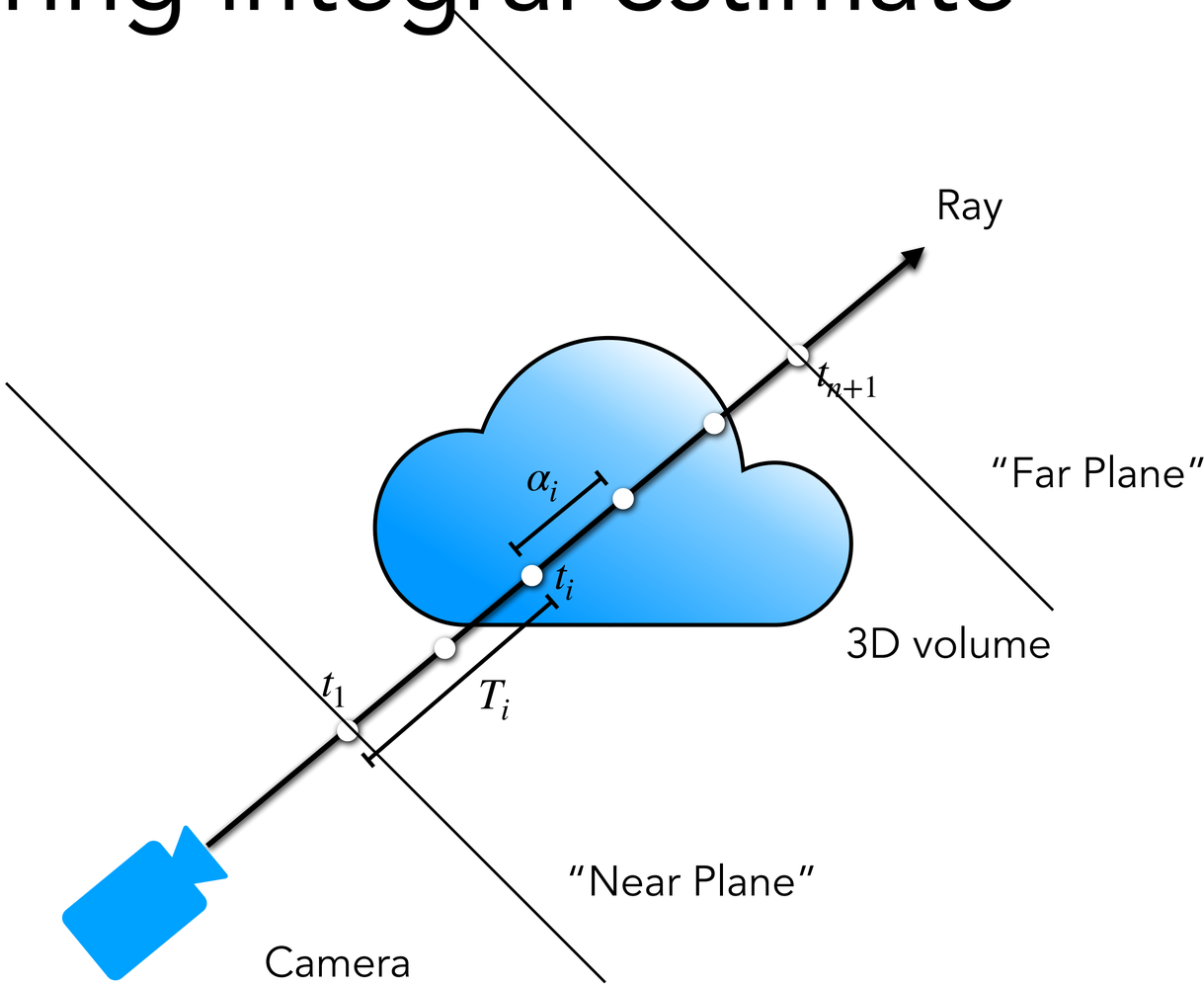
weights

How much light is blocked earlier along ray:

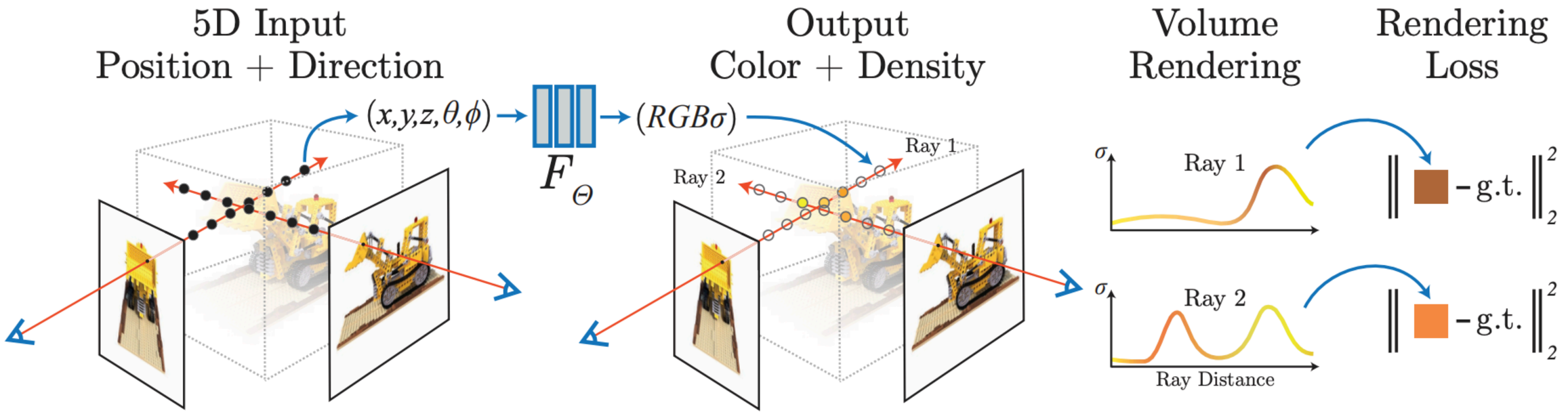
$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

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

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

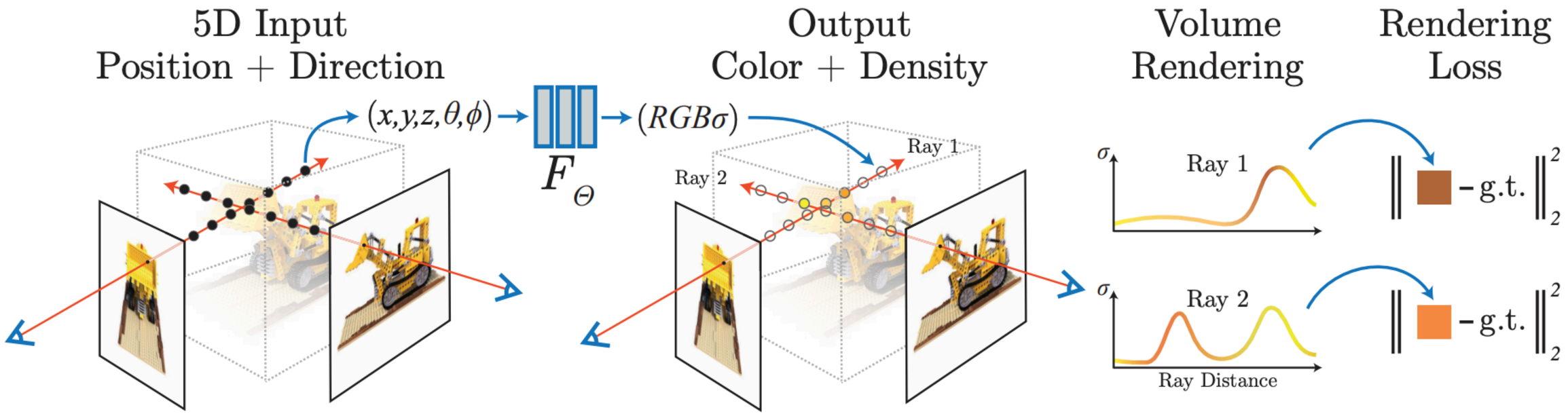


Learning a NeRF



[Source: Mildenhall et al., "NeRF", 2020]

Neural rendering



$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

Ray: $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ For color c and density σ.

[Source: Mildenhall et al., "NeRF", 2020]

Neural rendering

Color for ray $\mathbf{r}$

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

Ray: $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ For color c and density σ.

A point distance t along $\mathbf{r}$, centered at

[Source: Mildenhall et al., "NeRF", 2020]

Neural rendering

Color for ray $\mathbf{r}$

Weight

Color at 3D point $\mathbf{r}(t)$ & direction $\mathbf{d}$

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

For color c and density σ.

Ray: $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$

[Source: Mildenhall et al., "NeRF", 2020]

Neural rendering

Density at point $\mathbf{r}(t)$

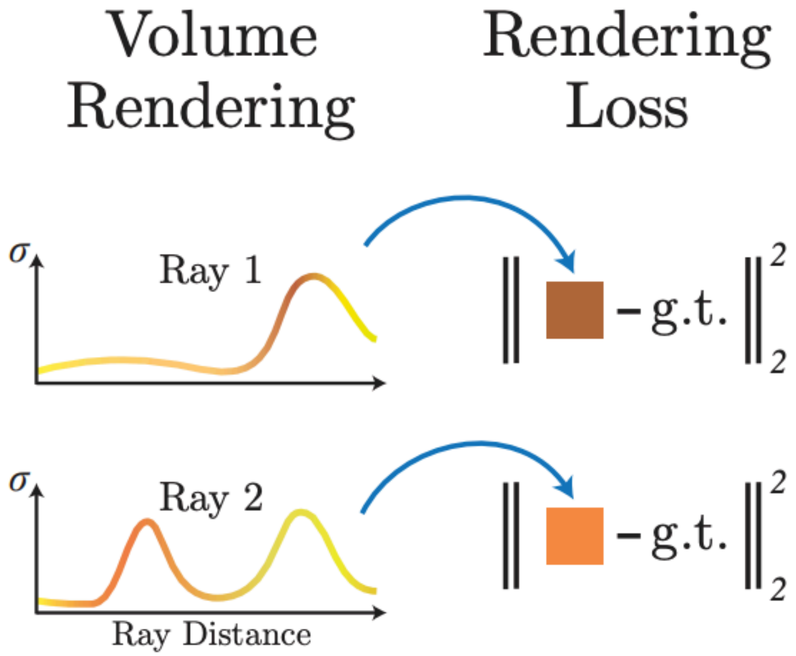
Probability that ray hasn been absorbed

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

Ray: $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ For color c and density σ.

[Source: Mildenhall et al., "NeRF", 2020]

Loss function



$$\mathscr{L} = \sum_{\mathbf{r}\in\mathscr{R}} \|C(\mathbf{r}) - C_{gt}(\mathbf{r})\|_2^2$$

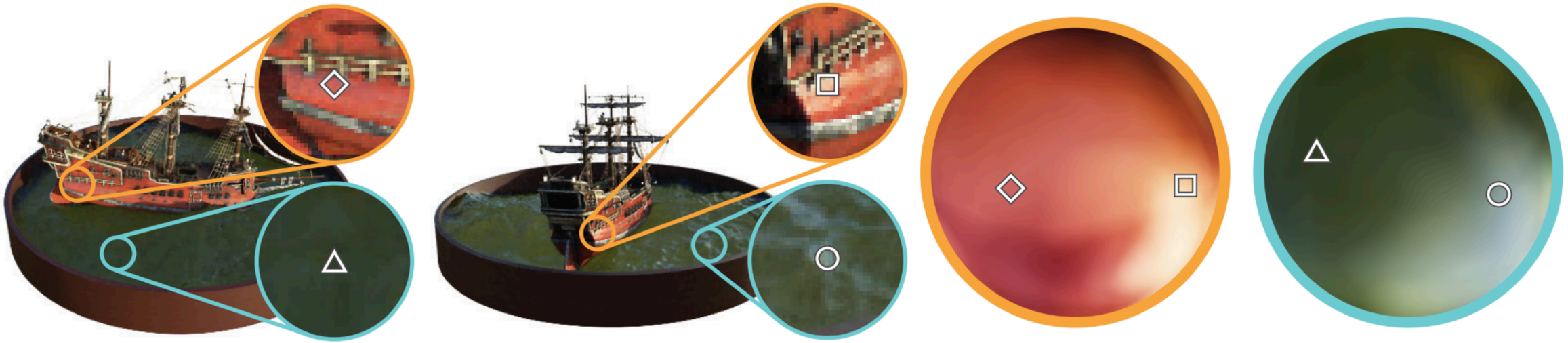
Minimize difference between predicted and observed colors.

In practice: coarse-to-fine and other tricks.

[Source: Mildenhall et al., "NeRF", 2020]

Implementation details

Why is it good to be view-dependent?



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

[Source: Mildenhall et al., “NeRF”, 2020]

Representing the inputs

Input views

$$\rightarrow \quad F_\Theta(x, y, z, \theta, \phi) = (R, G, B, \sigma)$$

- In theory, could just plug in 4 inputs x, y, z, θ, ϕ
- However, this leads to blurry results.
- Neural nets show a bias toward low frequency functions [Tancik et al., 2020]

Fourier features

Input views

$$\rightarrow \quad F_\Theta(x, y, z, \theta, \phi) = (R, G, B, \sigma)$$

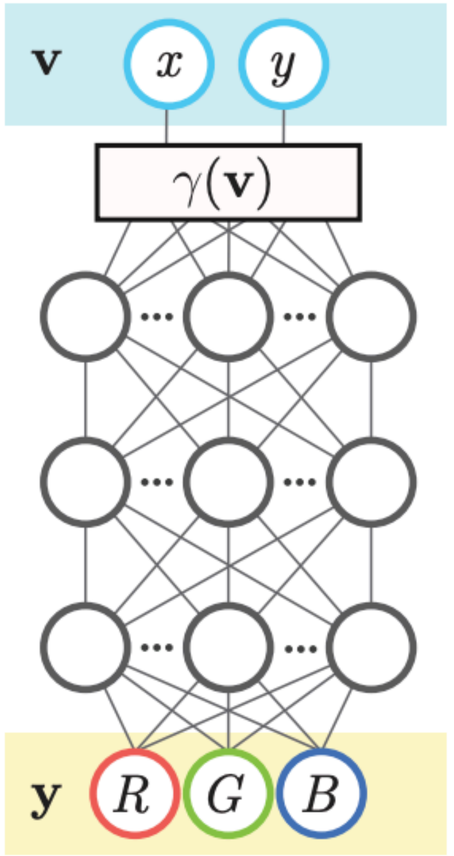
- Use a **positional encoding**. Given a scalar p, compute:

$$\gamma(p) = \left(\sin\left(2^0 \pi p\right), \cos\left(2^0 \pi p\right), \cdots, \sin\left(2^{L-1} \pi p\right), \cos\left(2^{L-1} \pi p\right)\right)$$

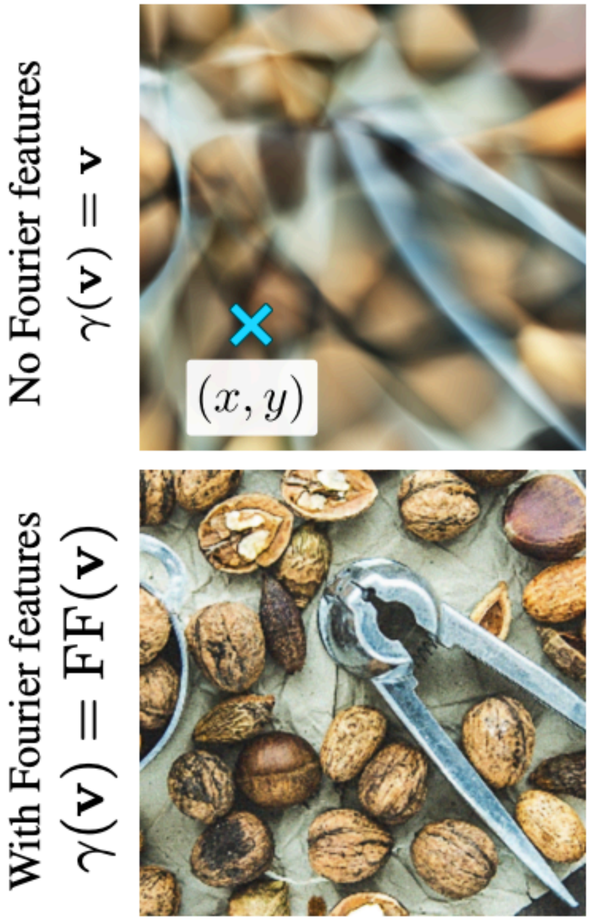
- Plug in the coordinate to sinusoids at different frequencies (e.g. L = 10).

[Source: Mildenhall et al., "NeRF", 2020]

Why does this happen?



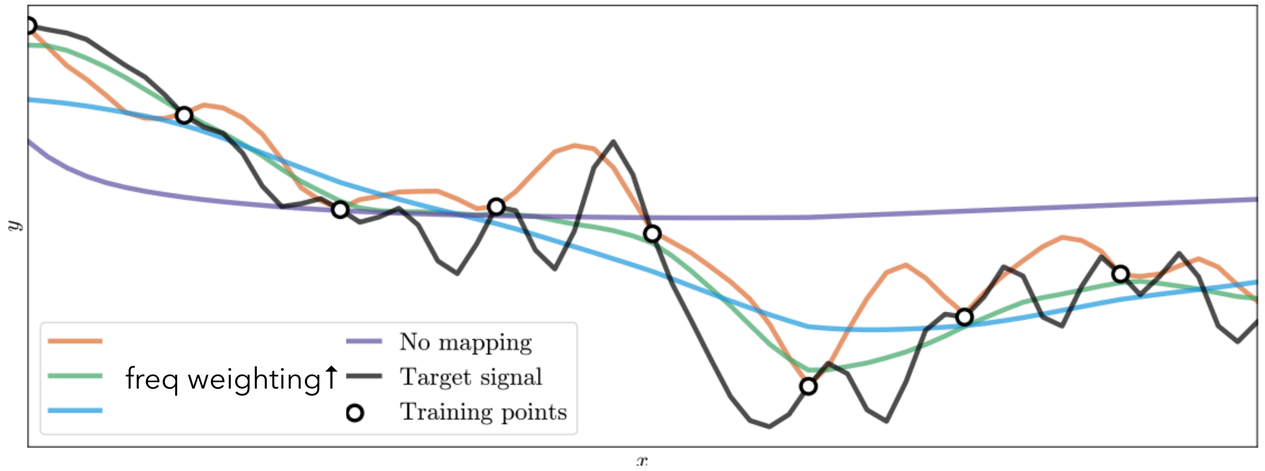
(a) Coordinate-based MLP



(b) Image regression

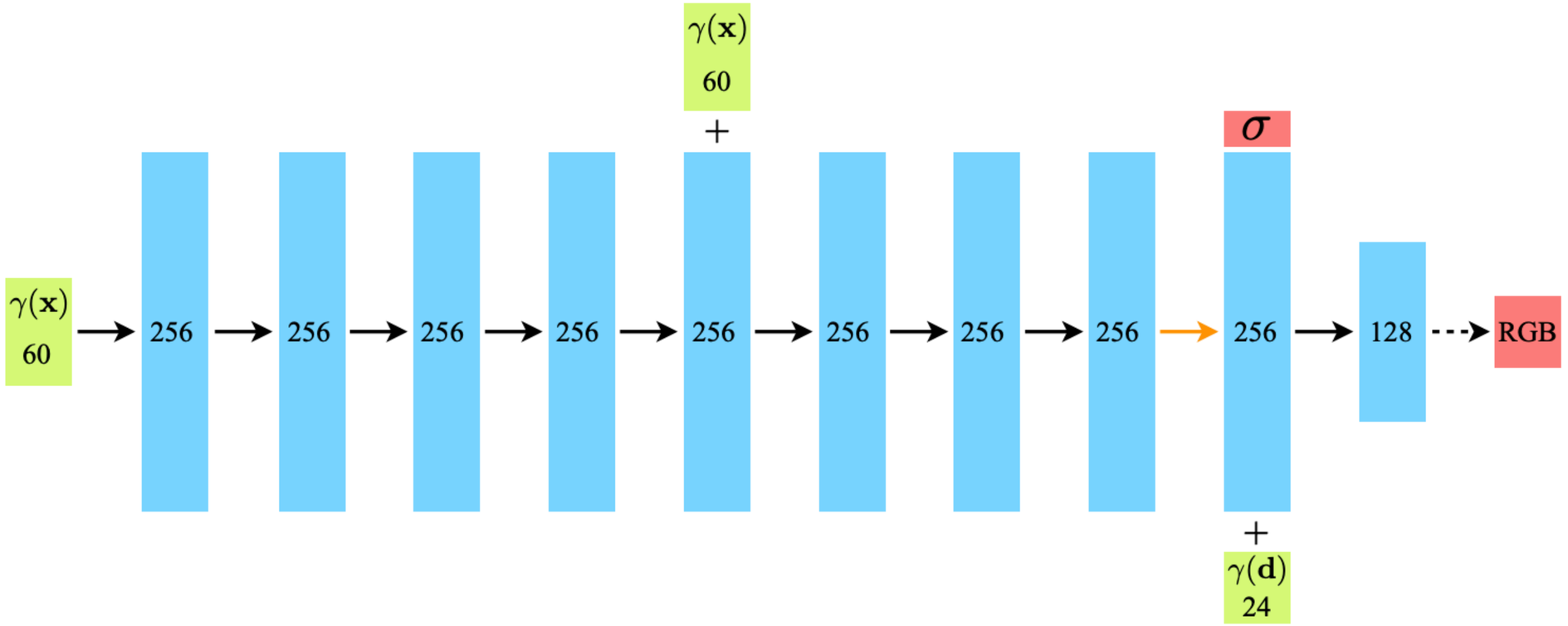
$(x,y) \rightarrow \mathrm{RGB}$

- Neural nets have trouble learning high frequency functions
- This mapping explicitly represents different frequencies (forces net to pay more attention high frequencies)



See [Tancik et al., "Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains", 2020]

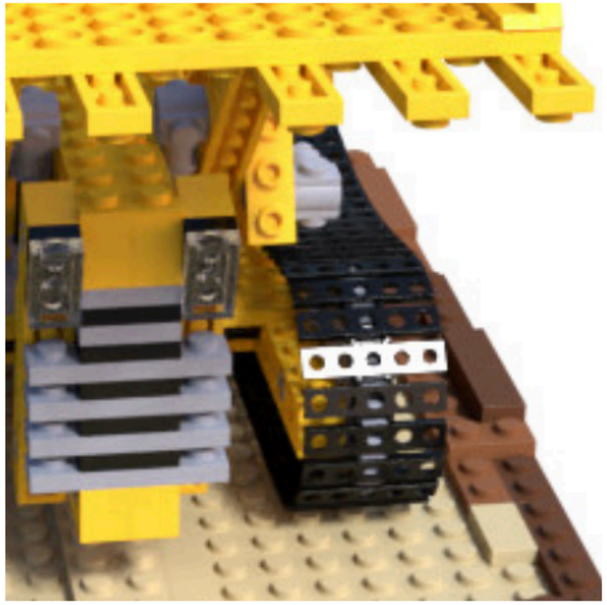
MLP architecture



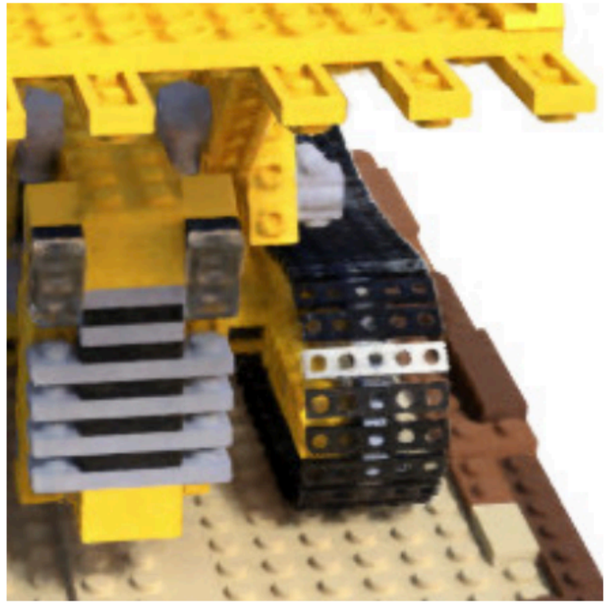
$$\gamma(p) = \left(\sin\left(2^0\pi p\right), \cos\left(2^0\pi p\right), \cdots, \sin\left(2^{L-1}\pi p\right), \cos\left(2^{L-1}\pi p\right)\right)$$

[Source: Mildenhall et al., "NeRF", 2020]

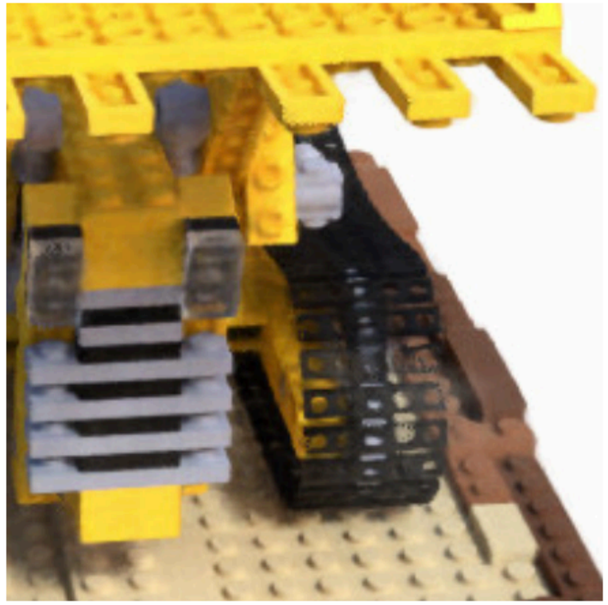
Results for a novel viewpoint



Ground Truth



Complete Model

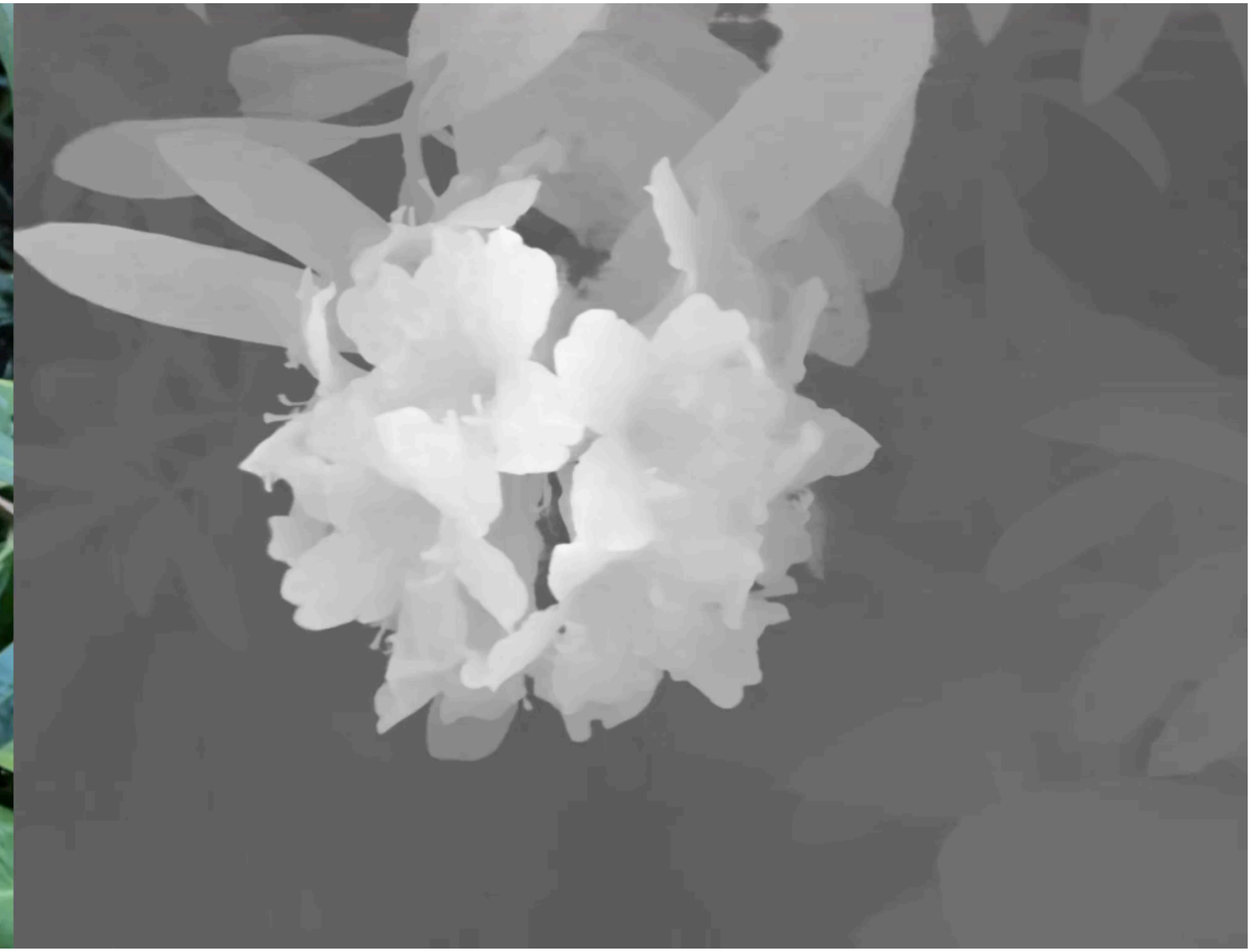


No View Dependence

No Positional Encoding

[Source: Mildenhall et al., “NeRF”, 2020]

Results



[Mildenhall*, Srinivasan*, Tanick*, et al. 2020

Results

[Mildenhall*, Srinivasan*, Tanick*, et al. 2020

Results

[Mildenhall*, Srinivasan*, Tanick*, et al. 2020

NeRF with other tricks

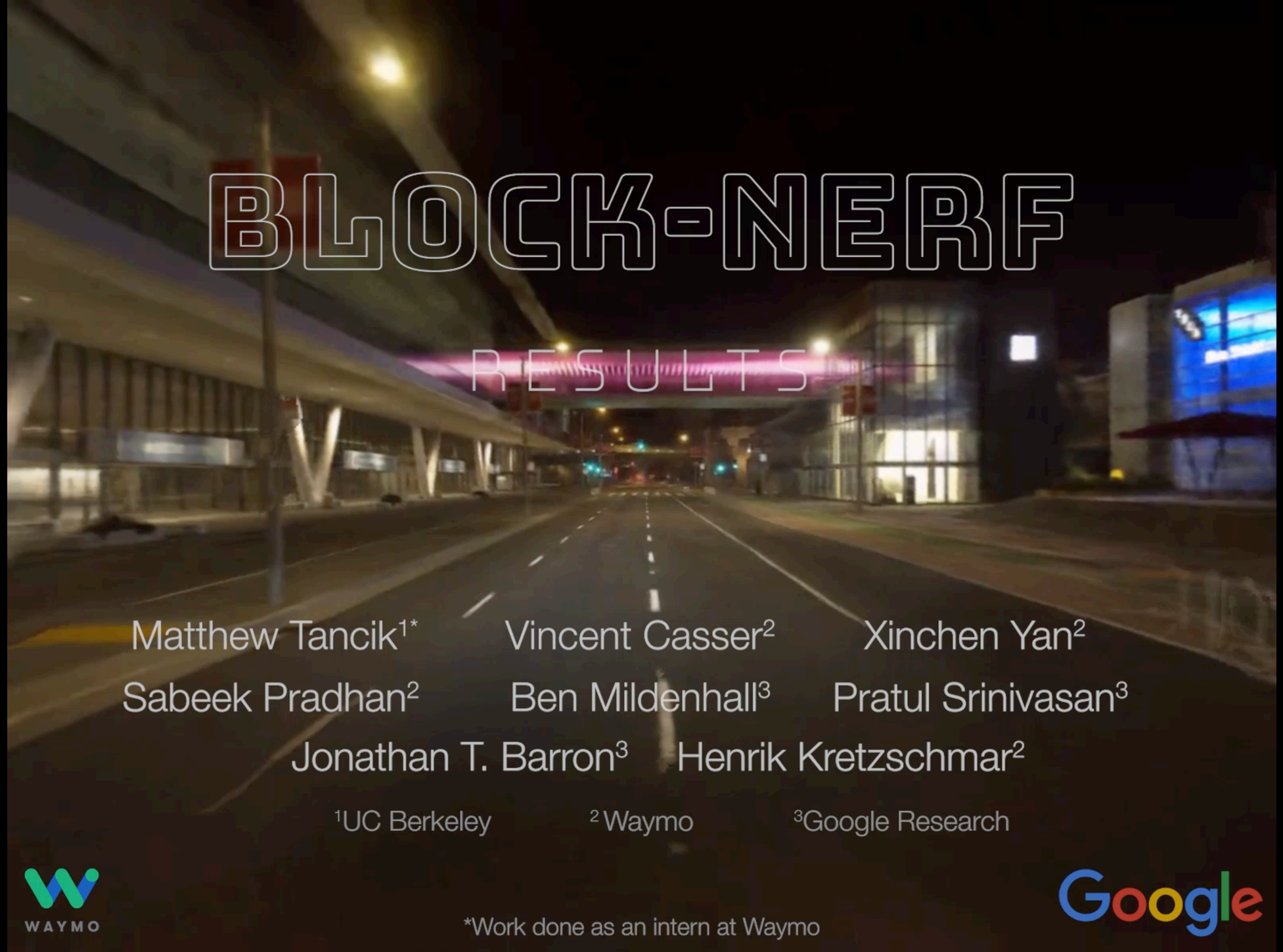
[Barron et al., “Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields. 2023]

Extension: internet photo collections

[Martin-Brualla, Radwan et al. “NeRF in the Wild”, 2020]

Extension: internet photo collections

[Martin-Brualla, Radwan et al. "NeRF in the Wild", 2020]



Matthew Tancik[1*] Vincent Casser[2] Xinchen Yan[2]
Sabeek Pradhan[2] Ben Mildenhall[3] Pratul Srinivasan[3]
Jonathan T. Barron[3] Henrik Kretzschmar[2]

[1]UC Berkeley [2]Waymo [3]Google Research

WAYMO

*Work done as an intern at Waymo

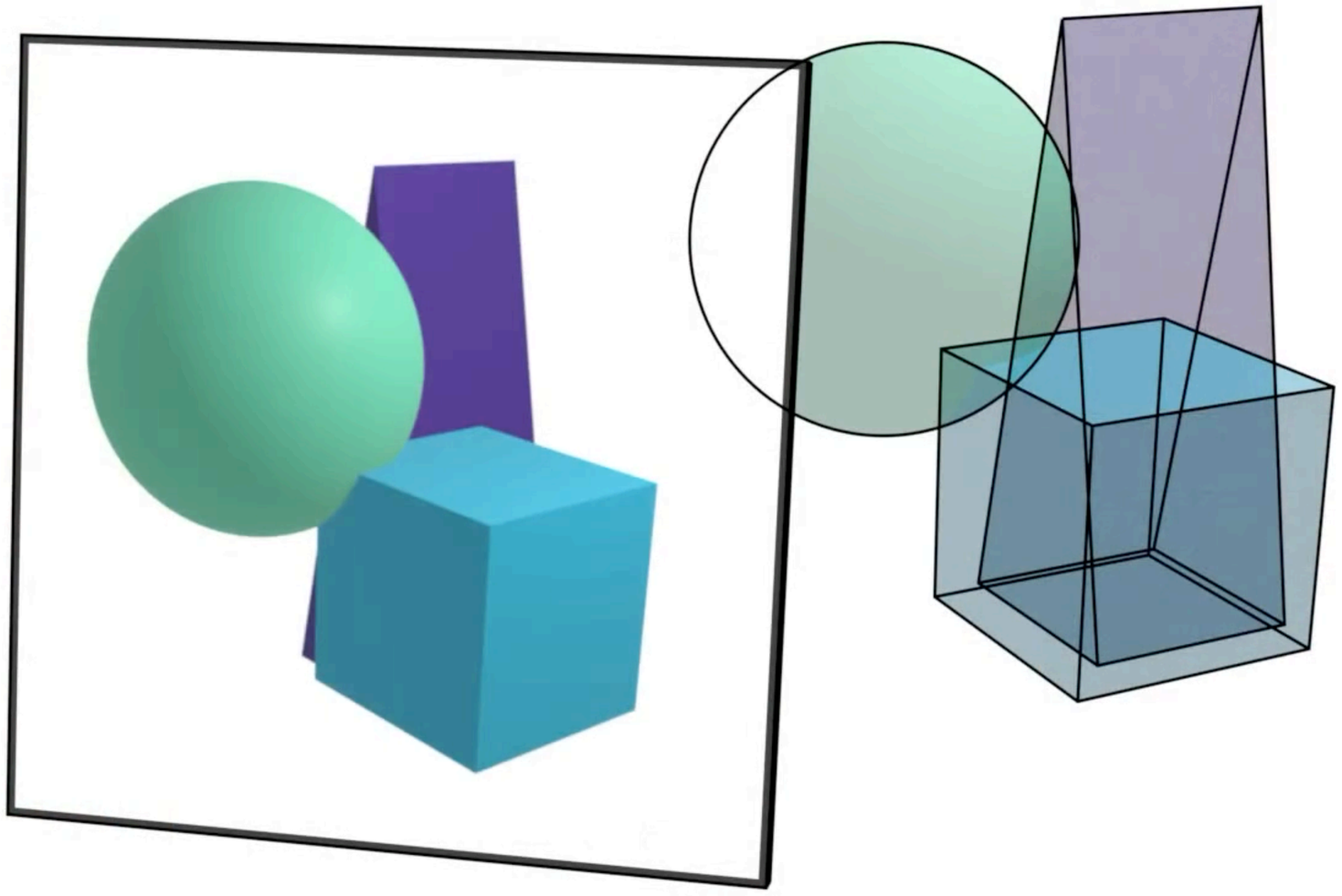
Google

What happens if we use fewer views?

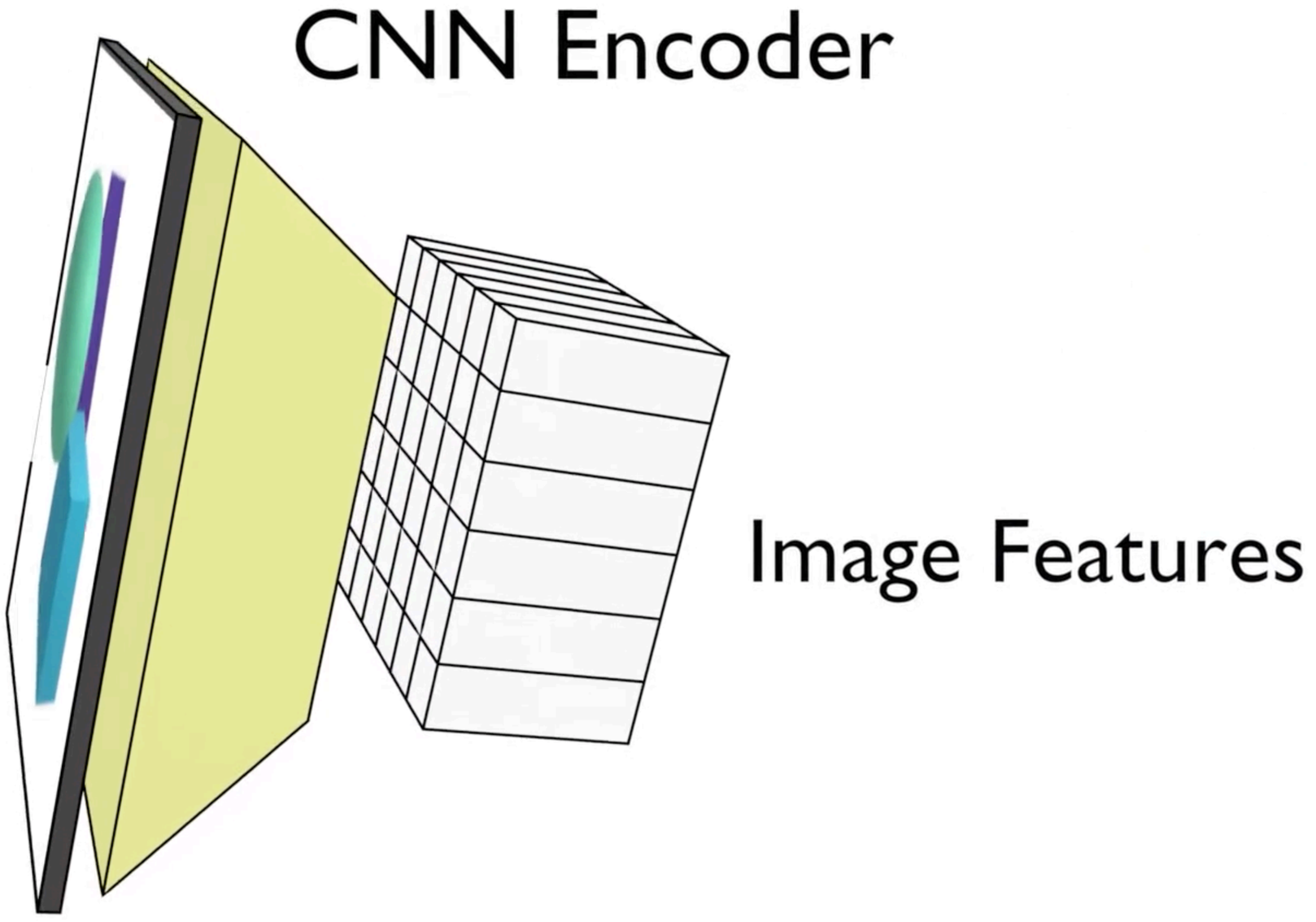
NeRF

Why does this happen?

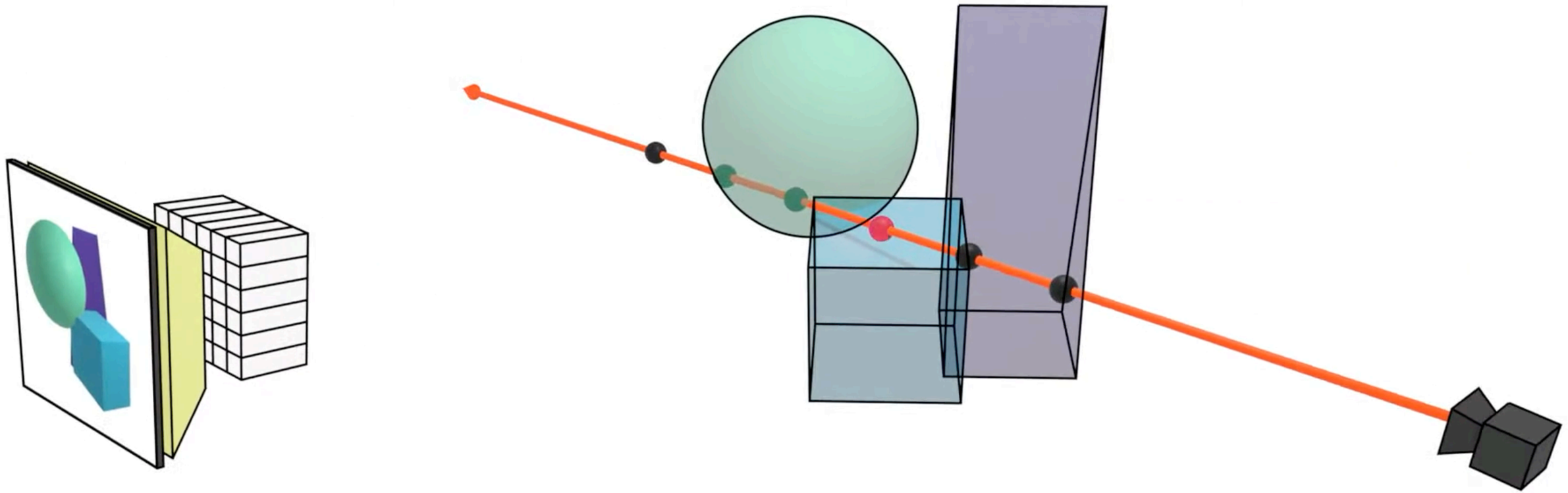
[Yu et al., “pixelNeRF: Neural Radiance Fields from One or Few Images”, 2021]



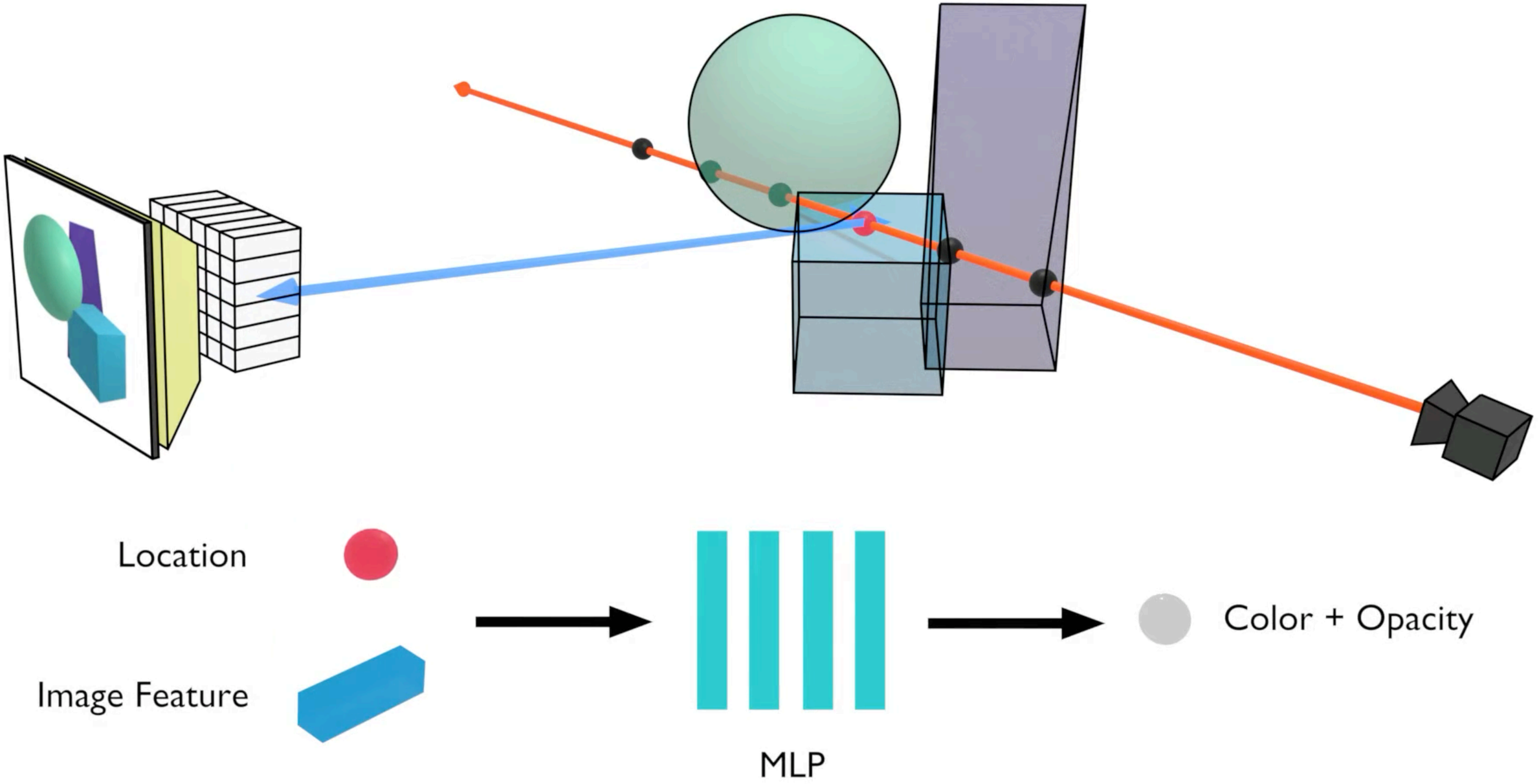
[Yu et al., “pixelNeRF: Neural Radiance Fields from One or Few Images”, 2021]



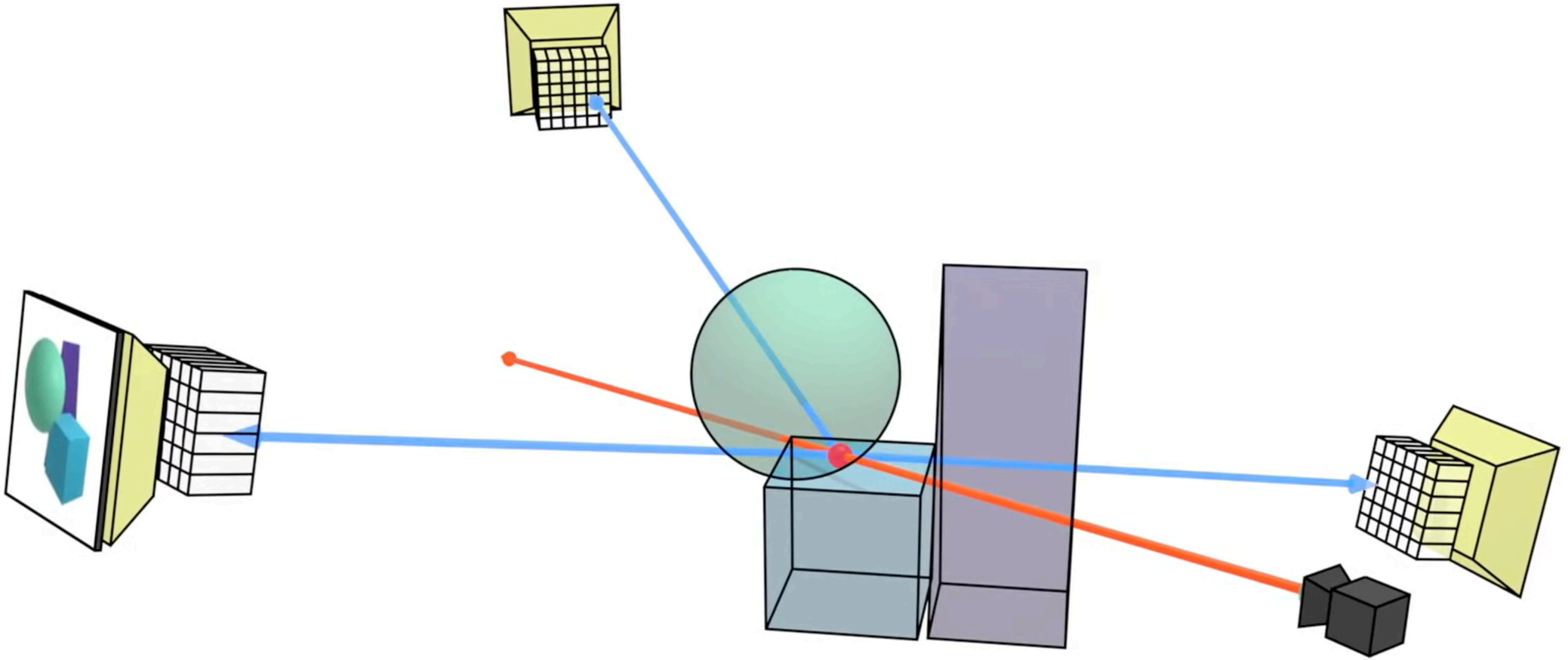
[Yu et al., “pixelNeRF: Neural Radiance Fields from One or Few Images”, 2021]



[Yu et al., “pixelNeRF: Neural Radiance Fields from One or Few Images”, 2021]



[Yu et al., “pixelNeRF: Neural Radiance Fields from One or Few Images”, 2021]



[Yu et al., “pixelNeRF: Neural Radiance Fields from One or Few Images”, 2021]

Input

Input

[Yu et al., “pixelNeRF: Neural Radiance Fields from One or Few Images”, 2021]

Can we generate 3D models?

Generating 3D models

Do we actually need 3D supervision?

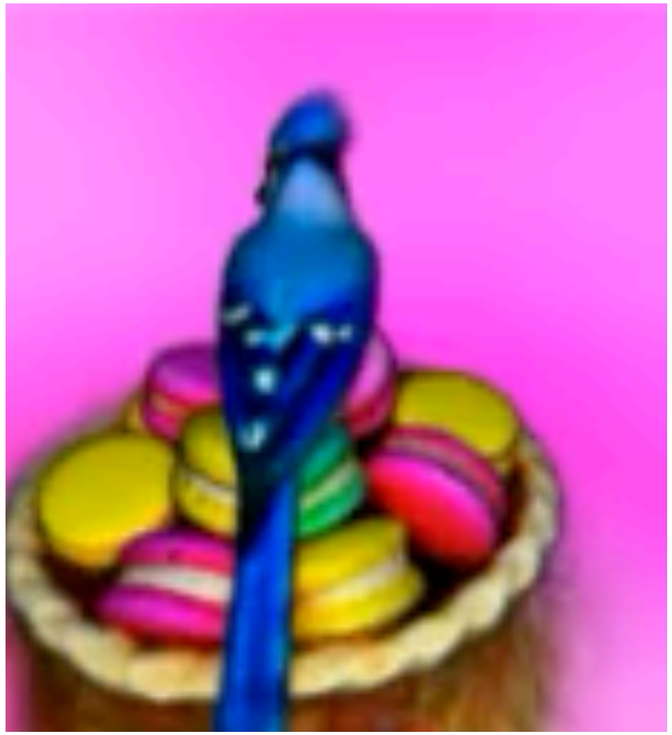
Source: [Poole et al., 2023]

We already have good text-to-image models

“A blue jay standing on a large basket of rainbow macarons.”

Source: [Poole et al., 2023]

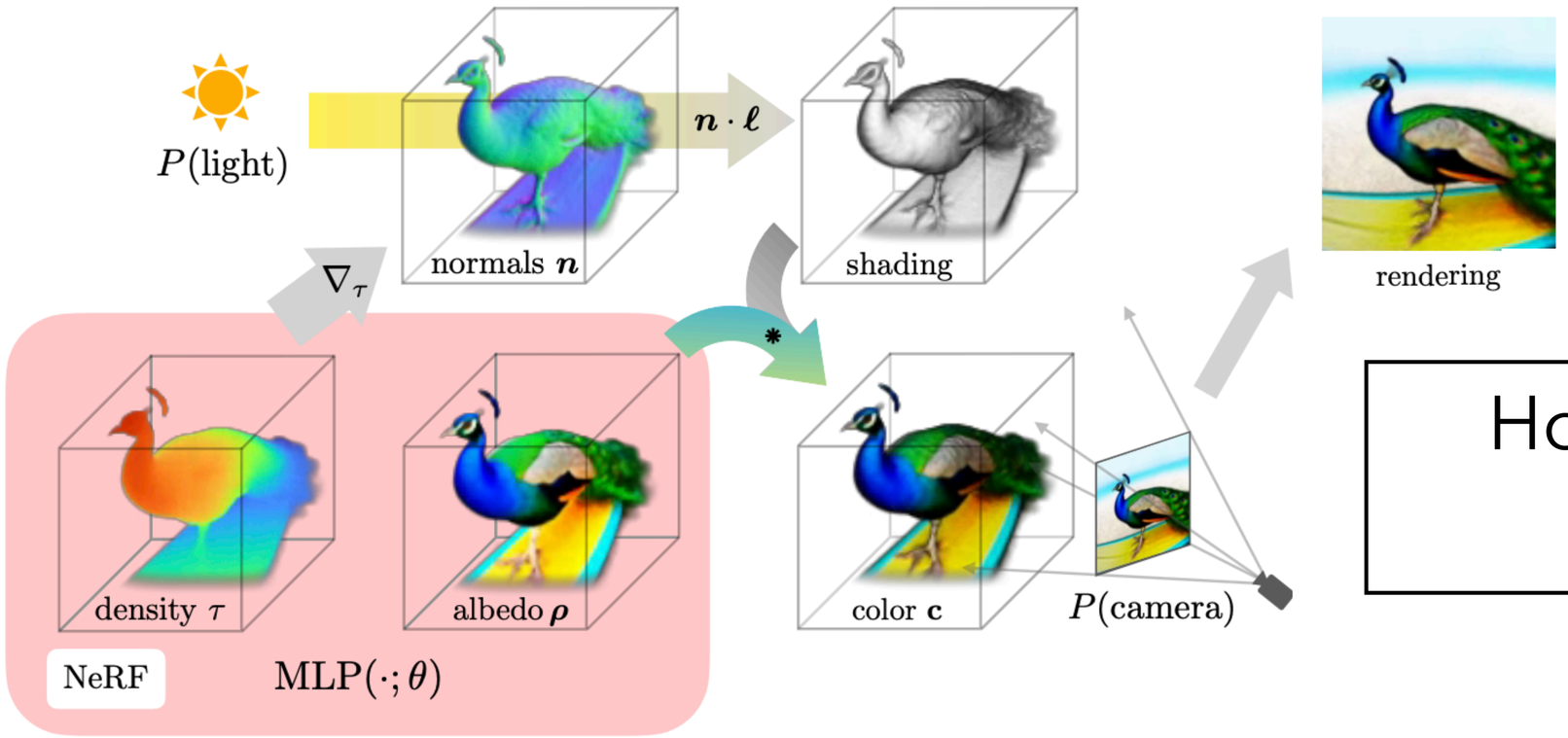
Could we fit a 3D model to this image?



Single image could “supervise” single viewpoint.

Source: [Poole et al., 2023]

DreamFusion: solve for a NeRF that "looks right" from every viewpoint

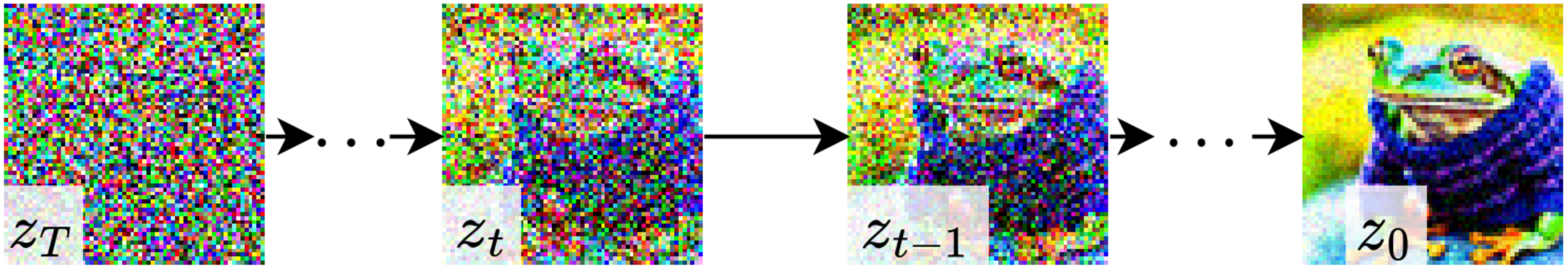


How do we measure "looks right"?

$p(\text{rendered image} \mid \text{"a DSLR photo of a peacock on a surfboard"})$

Source: [Poole et al., 2023]

Can we use a pretrained diffusion model?



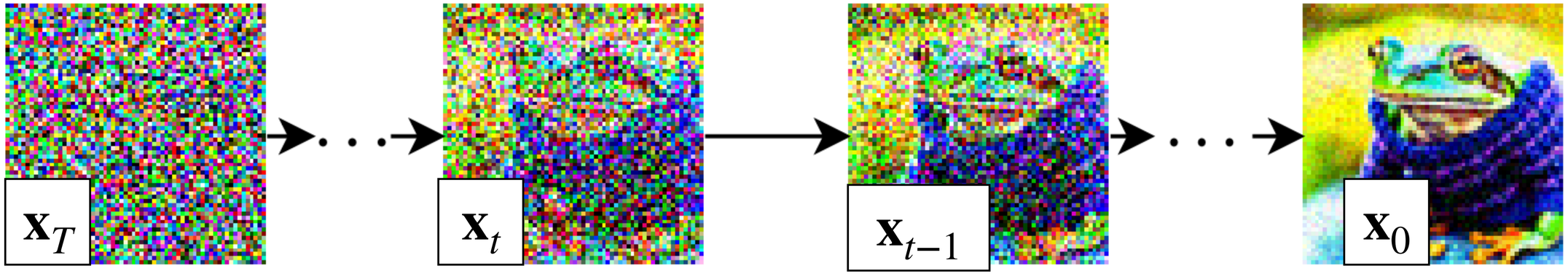
Diffusion: generate image by denoising

… but also it models $p(\text{image} \mid \text{text})$!

Source: [Poole et al., 2023]

Score distillation sampling

Reverse diffusion process:



Diffusion objective function:

Noise estimator

Noise

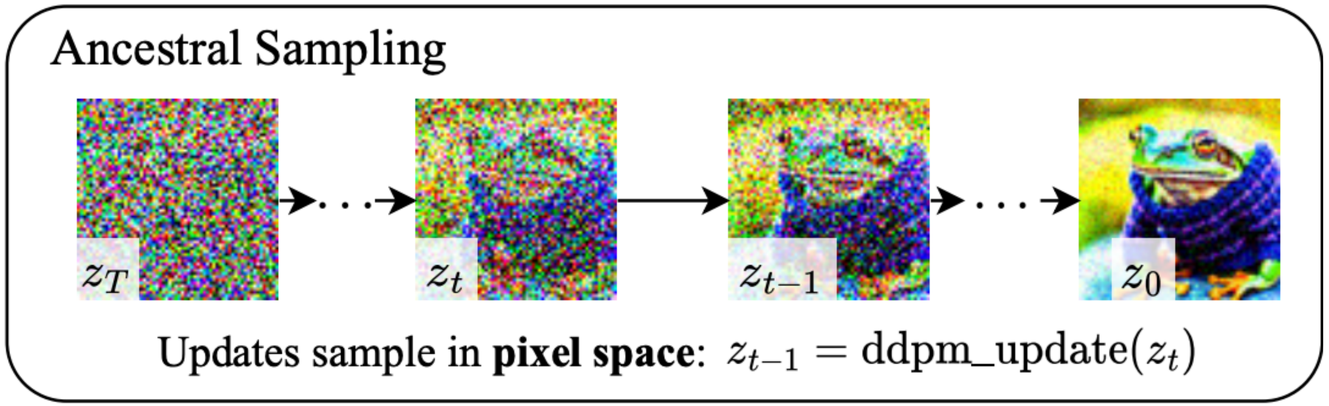
$$\mathcal{L}_{\text{Diff}}(\phi, \mathbf{x}) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\epsilon\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[w(t)\|\epsilon_\phi(\alpha_t\mathbf{x} + \sigma_t\epsilon; t) - \epsilon\|_2^2\right]$$

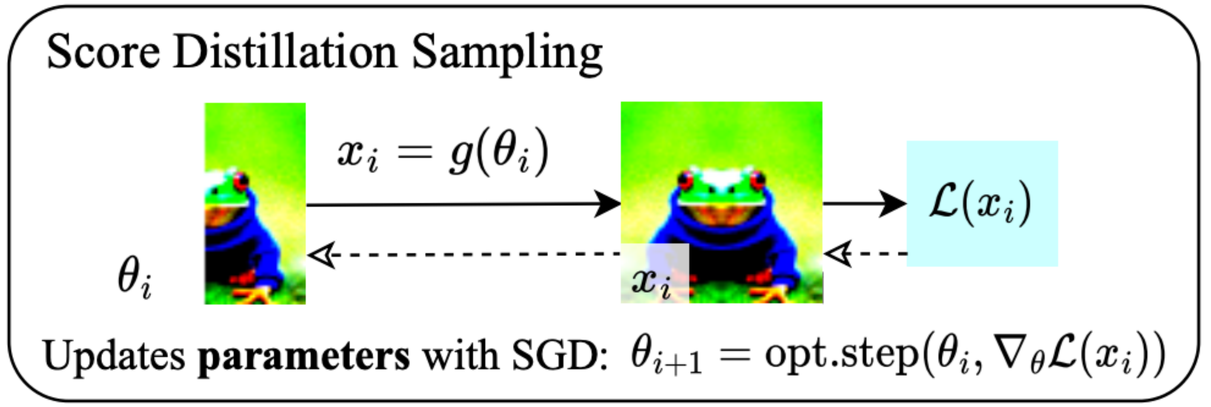
Noisy image

Provides a lower bound on $\log(p(\mathbf{x}))$ [Ho et al., 2020, Kingma et al., 2021]

Source: [Poole et al., 2023]

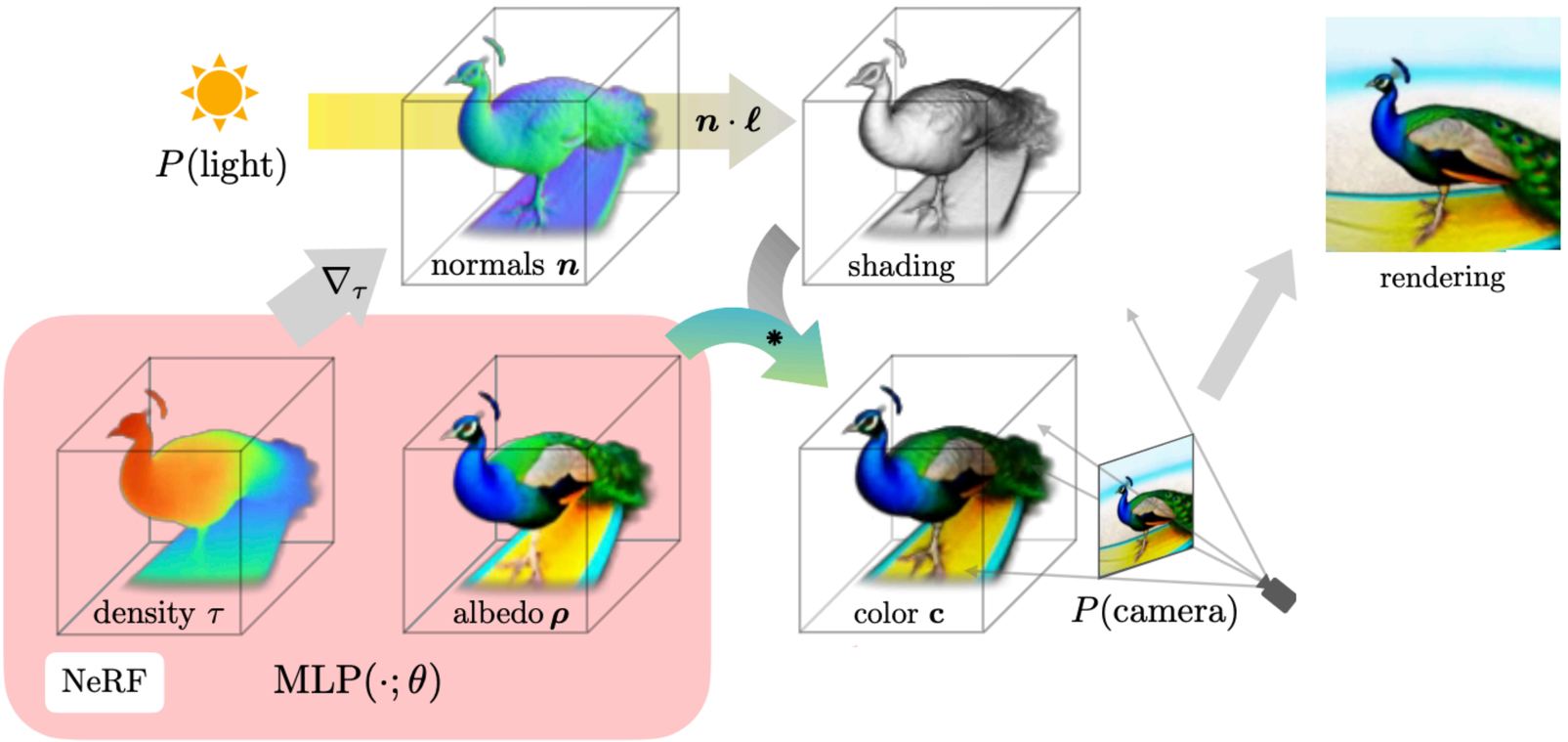
Score distillation sampling





Source: [Poole et al., 2023]

DreamFusion



Solve for a NeRF such that, when it is rendered, it has high probability under a text-to-image model.

Source: [Poole et al., 2023]

DreamFusion

Generate 3D from text yourself!

a DSLR photo of a squirrel | an intricate wooden carving of a squirrel | a highly detailed metal sculpture of a squirrel

[...] | wearing a kimono | wearing a medieval suit of armor | wearing a purple hoodie | wearing an elegant ballgown

[...] | reading a book | riding a motorcycle | playing the saxophone | chopping vegetables | sitting at a pottery wheel shaping a clay bowl | riding a skateboard | wielding a katana | eating a hamburger | dancing

DreamFusion

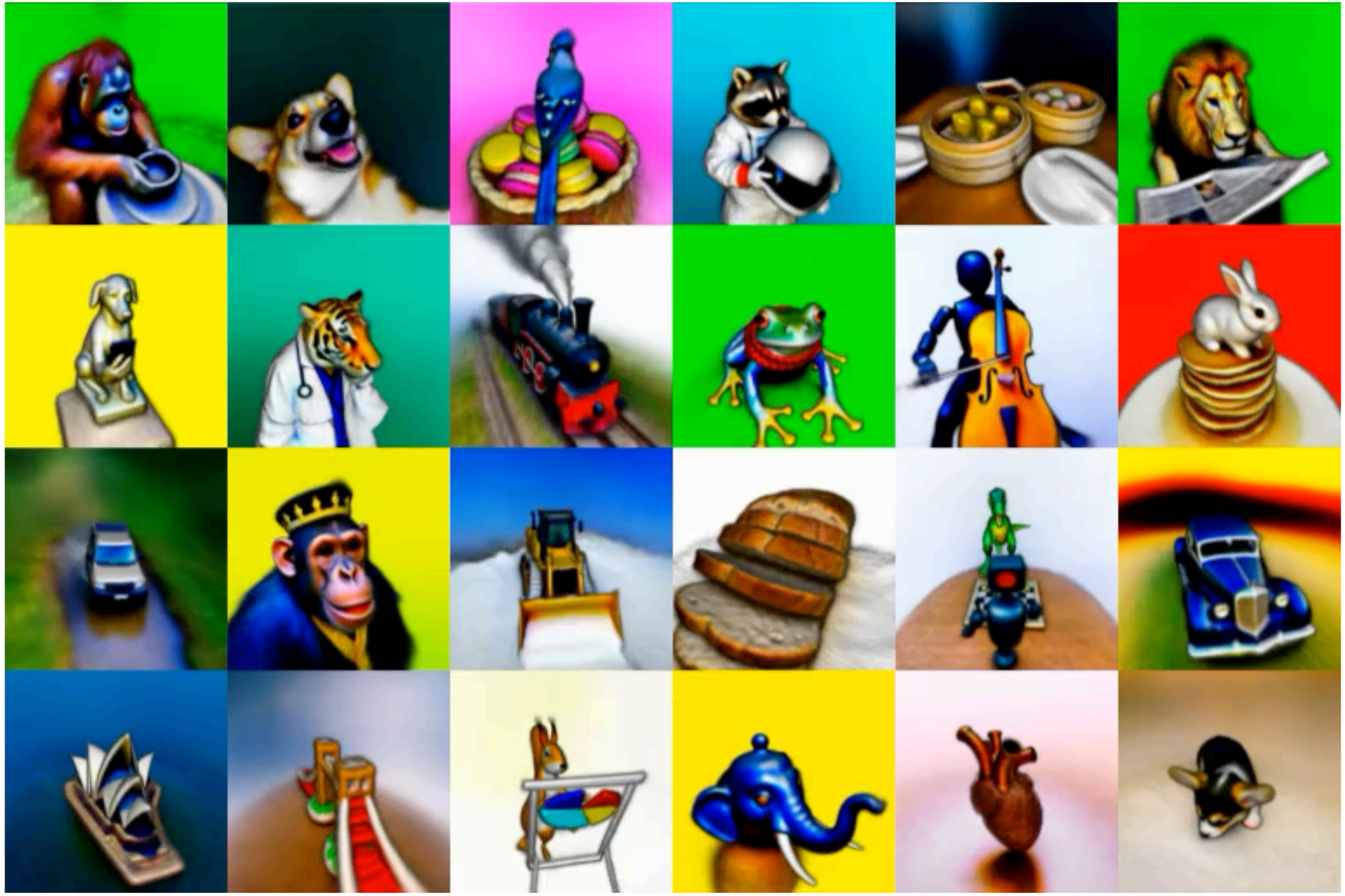
Generate 3D from text yourself!

a DSLR photo of a squirrel | an intricate wooden carving of a squirrel | a highly detailed metal sculpture of a squirrel

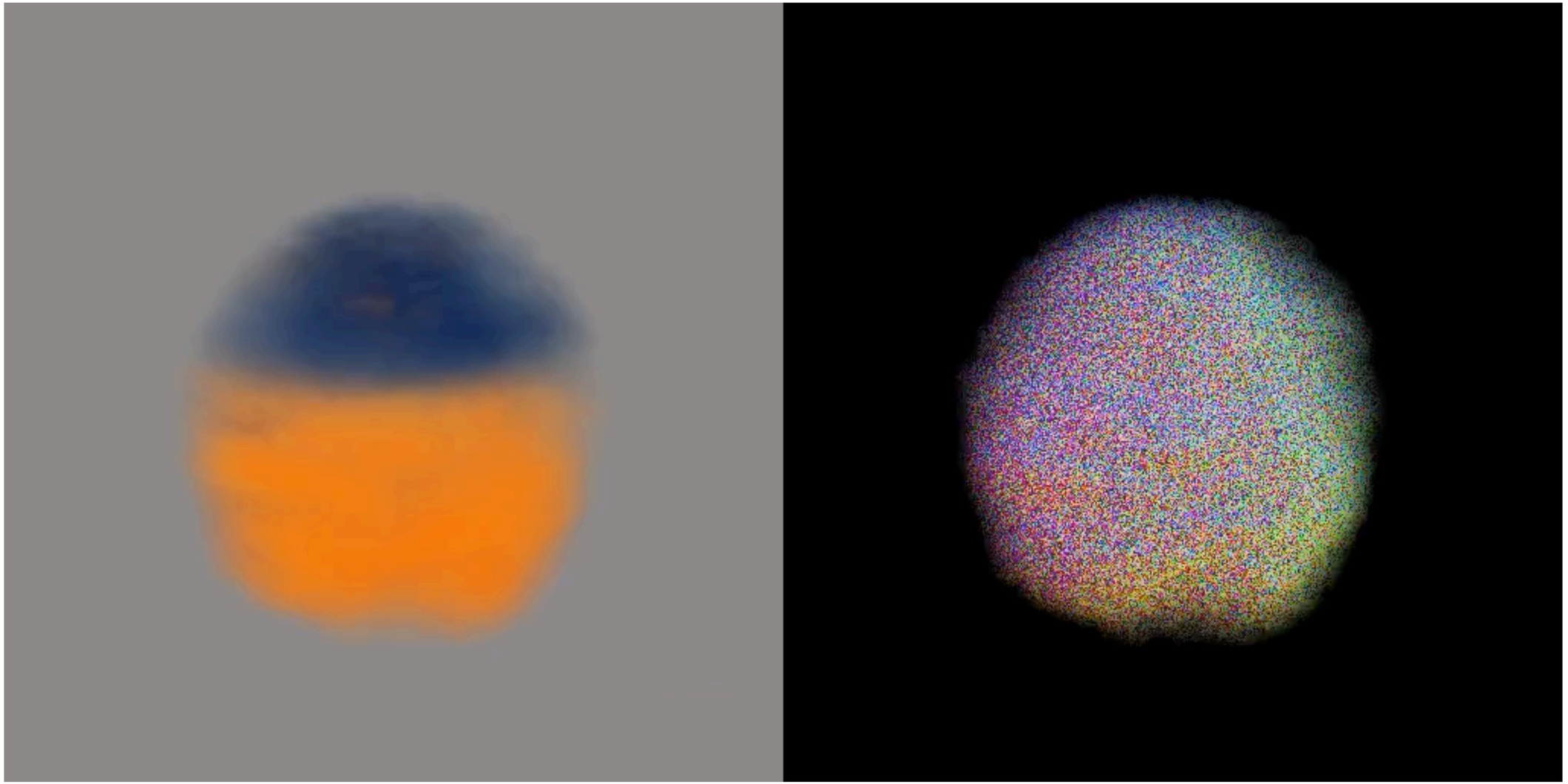
[...] | wearing a kimono | wearing a medieval suit of armor | wearing a purple hoodie | wearing an elegant ballgown

[...] | reading a book | riding a motorcycle | playing the saxophone | chopping vegetables | sitting at a pottery wheel shaping a clay bowl | riding a skateboard | wielding a katana | eating a hamburger | dancing

DreamFusion

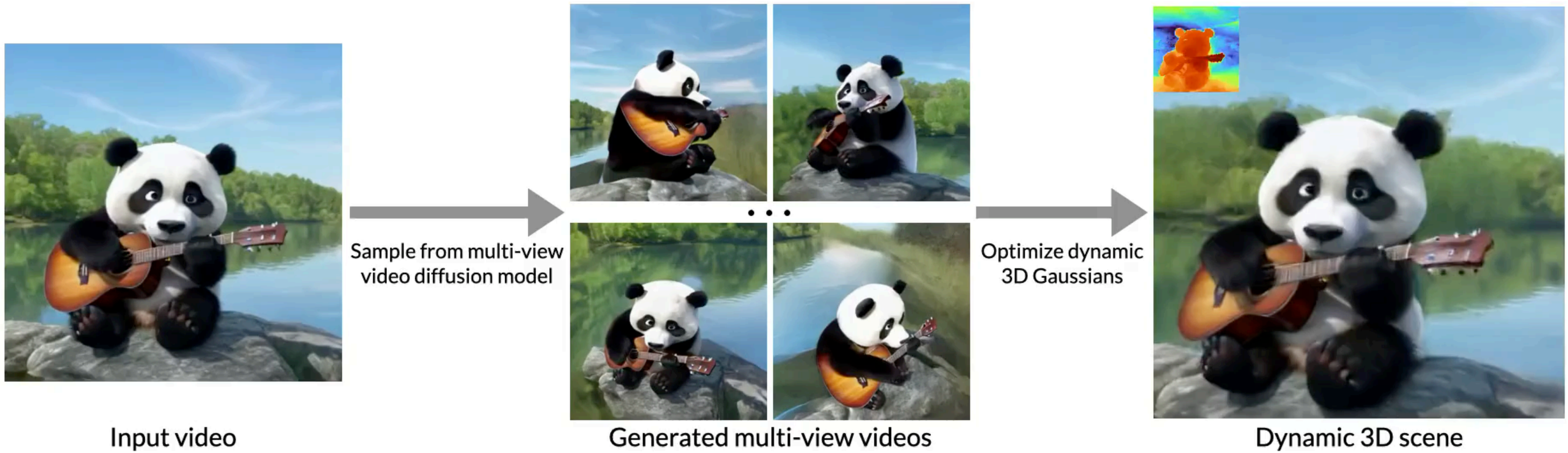


More recent text-to-3D



[Shi et al., MVDream, 2023]

Text-to-4D



[Wu et al., CAT4D, 2024]

[Wu et al., CAT4D, 2024]

Next class: light