Neural Radiance Fields

Jon Barron

Google Research

About me

UC Berkeley
PhD Student
2008-2013
Advisor: Jitendra Malik

Google Research: Perception
Research Scientist
2013-Now

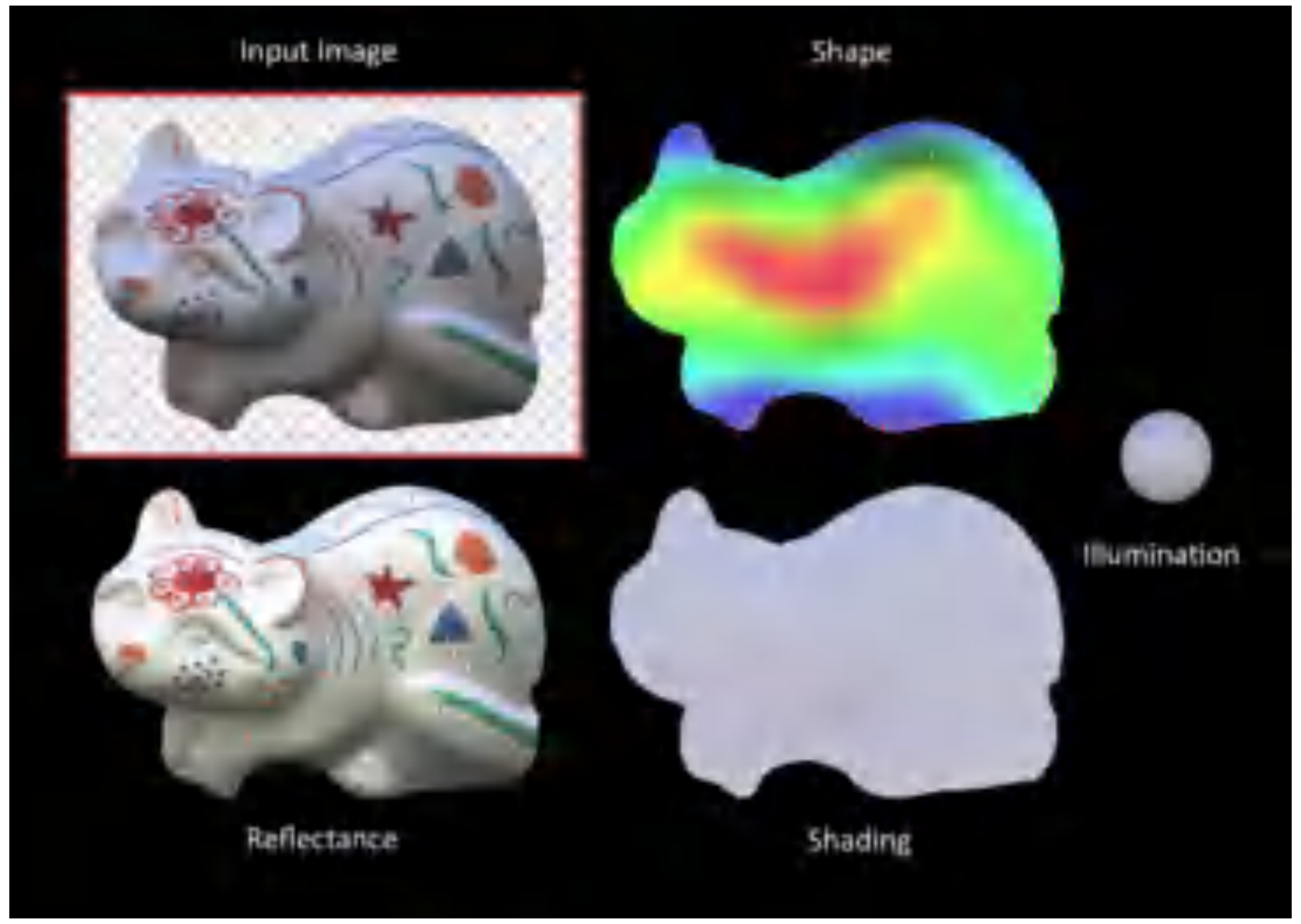
Team:


Peter
Hedman


Pratul
Srinivasan


Ben
Mildenhall

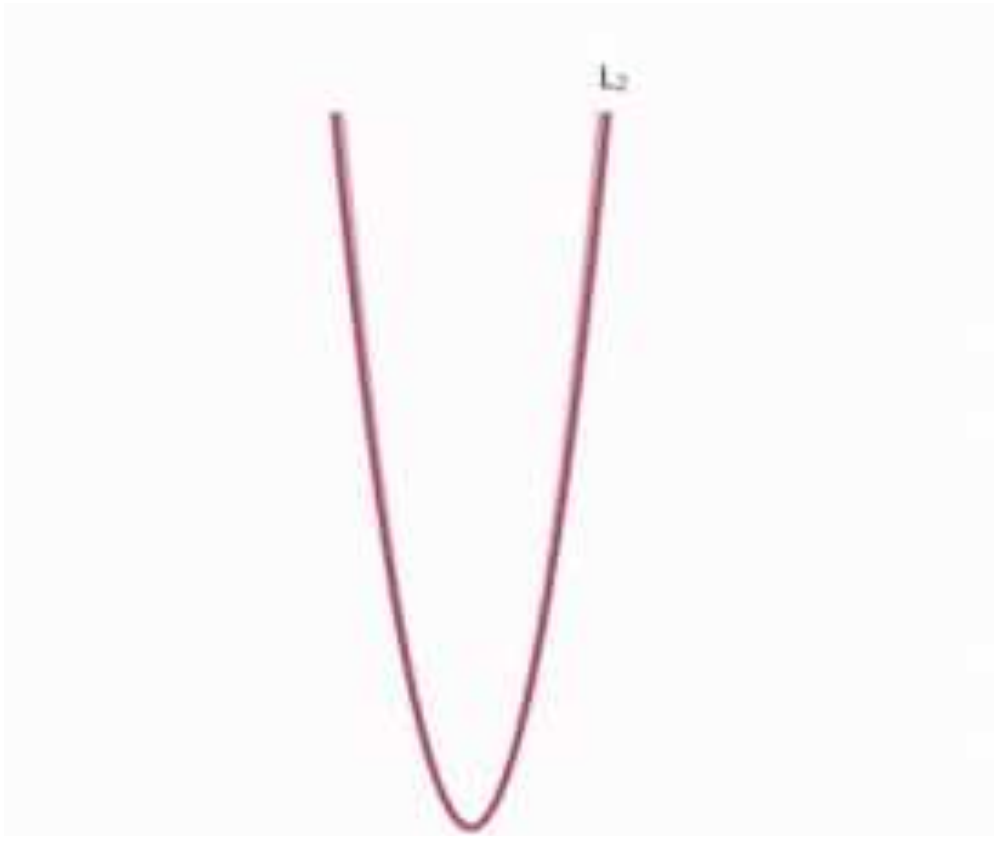
Research Interests



Inverse Rendering

Color Constancy

Smooth Motion / Depth Estimation

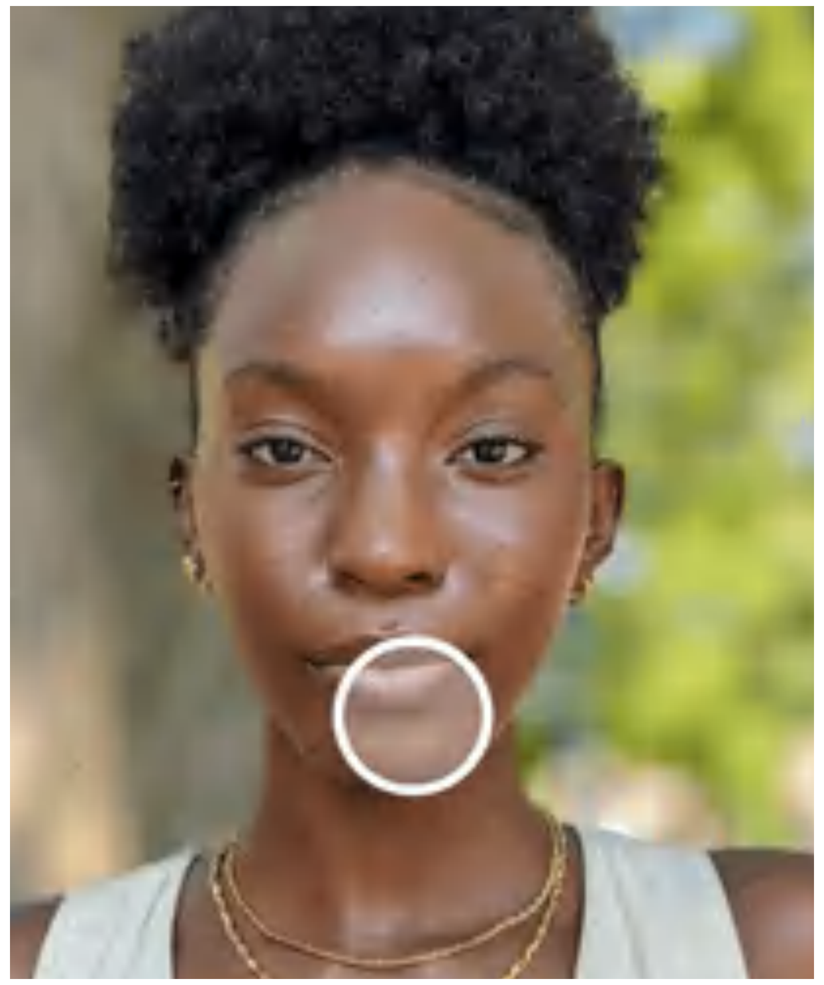


Loss Functions

Research Impact


HDR+ / Night Sight


Lens Blur / Portrait Mode


Portrait Light


Google Glass


Jump

What is graphics?

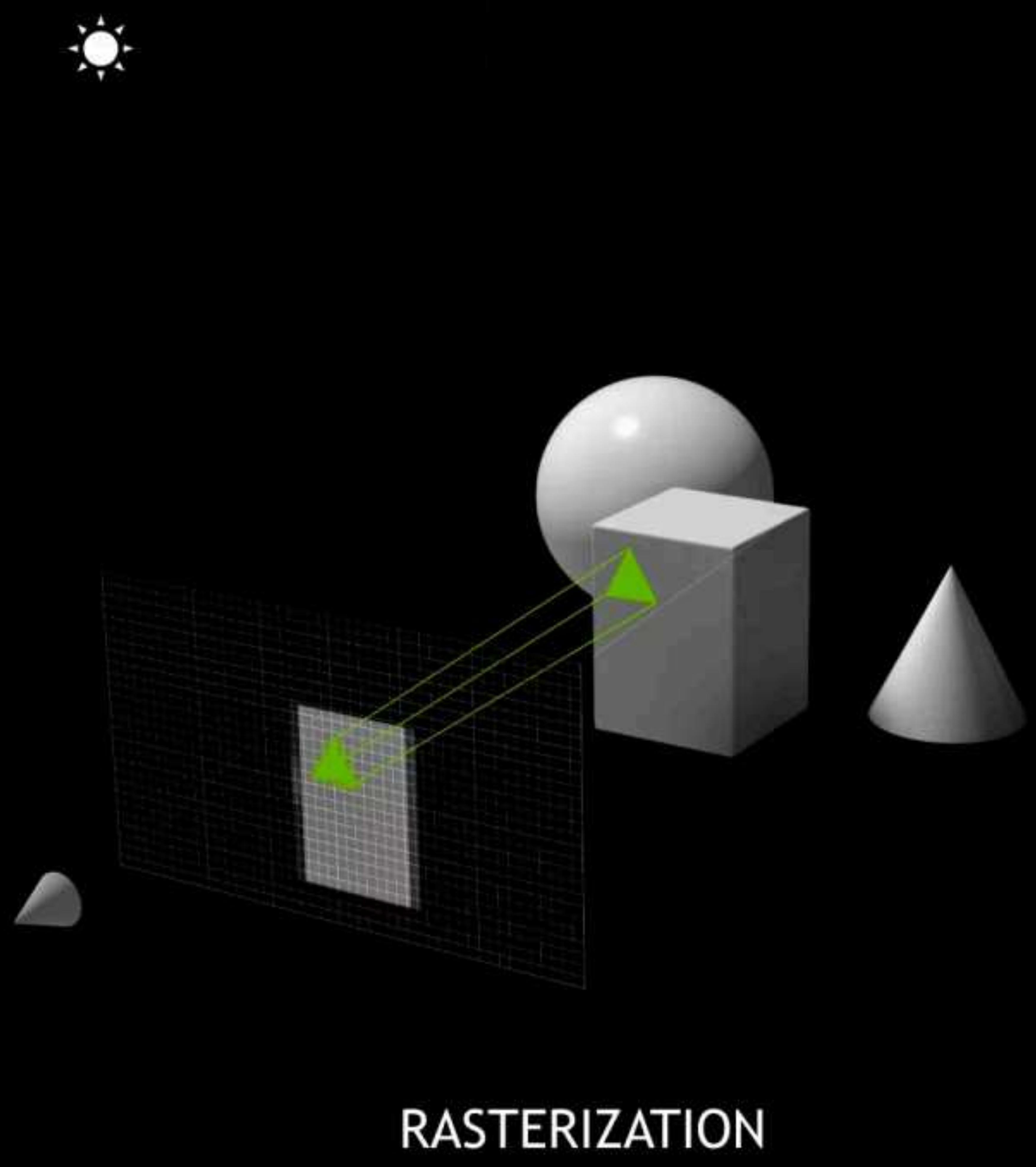
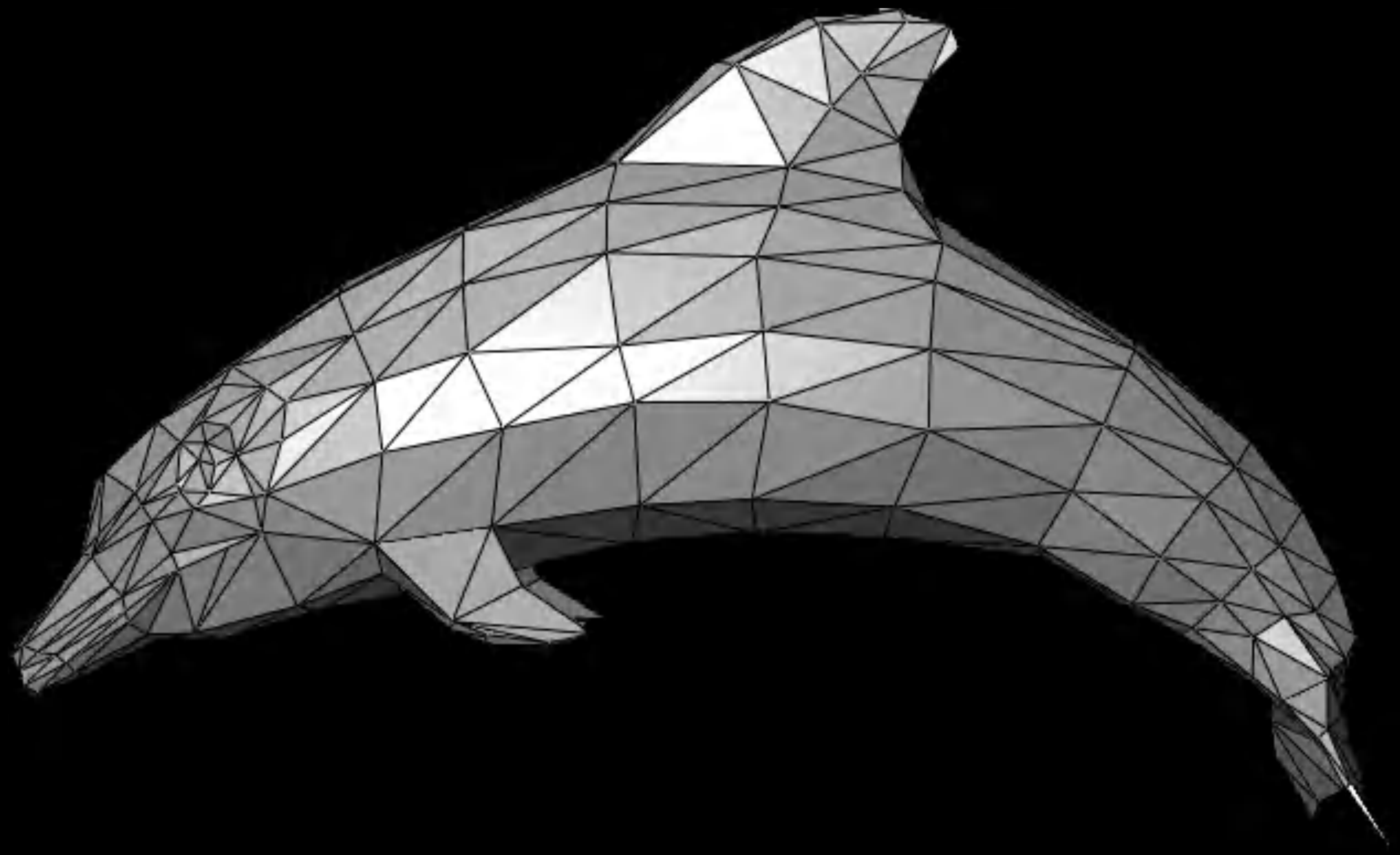


Image: Nvidia

What is graphics?



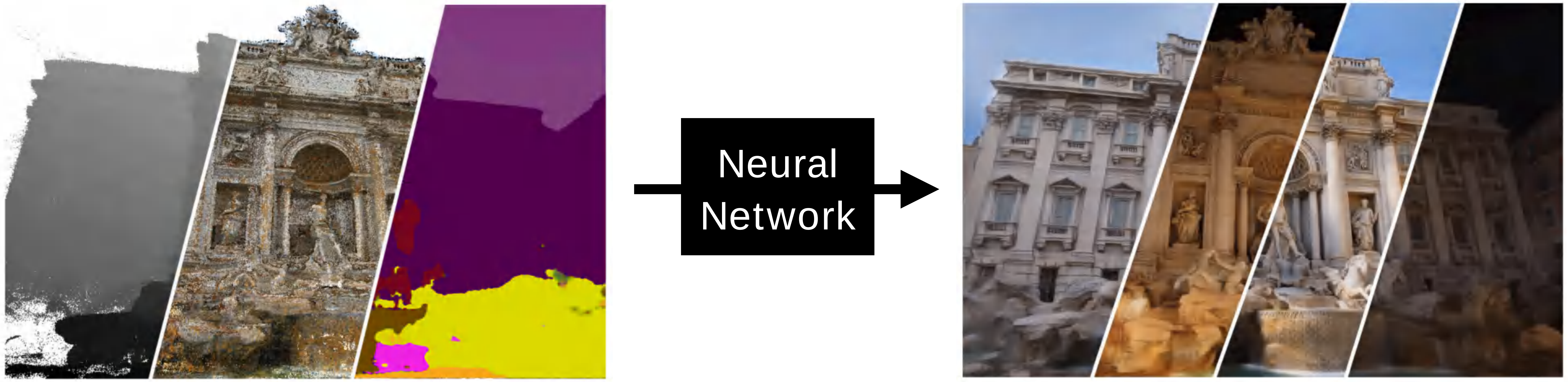
Mesh Rendering

Chrschn, Public domain, via Wikimedia Commons

Is this "neural rendering"?

Paradigm 1:

"The neural network is a black box that directly renders pixels"

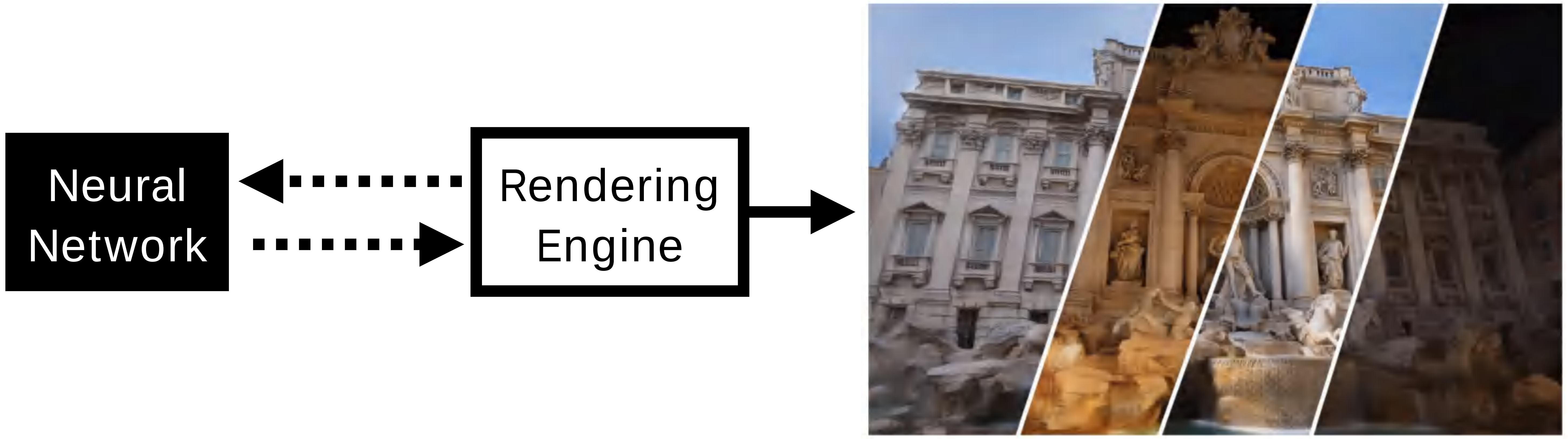


Neural Rerendering in the Wild, Meshry et al. CVPR 2019

Paradigm A:

"The neural network is a black box that models the geometry of the world, and a (non-learned) graphics engine renders it"

"Scene Representation"
"Implicit Representations"



NeRF in the Wild, Martin-Brualla et al. CVPR 2021

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Ben Mildenhall*

UC Berkeley

Pratul Srinivasan*

UC Berkeley

Matt Tancik*

UC Berkeley

Jon Barron

Google Research

Ravi Ramamoorthi

UC San Diego

Ren Ng

UC Berkeley

Problem: View Interpolation

Inputs: sparsely sampled images of scene

Outputs: *new* views of same scene

tancik.com/nerf

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)

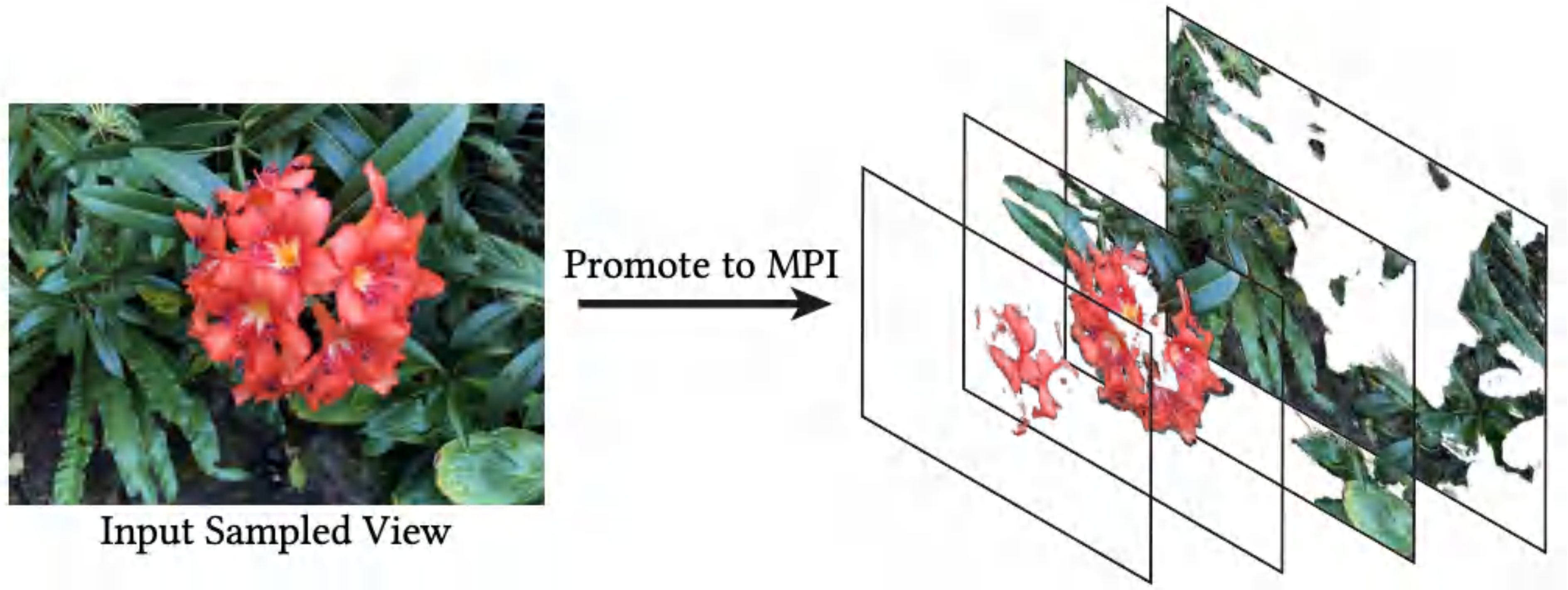
Pushing the Boundaries… (Srinivasan et al. 2019)

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

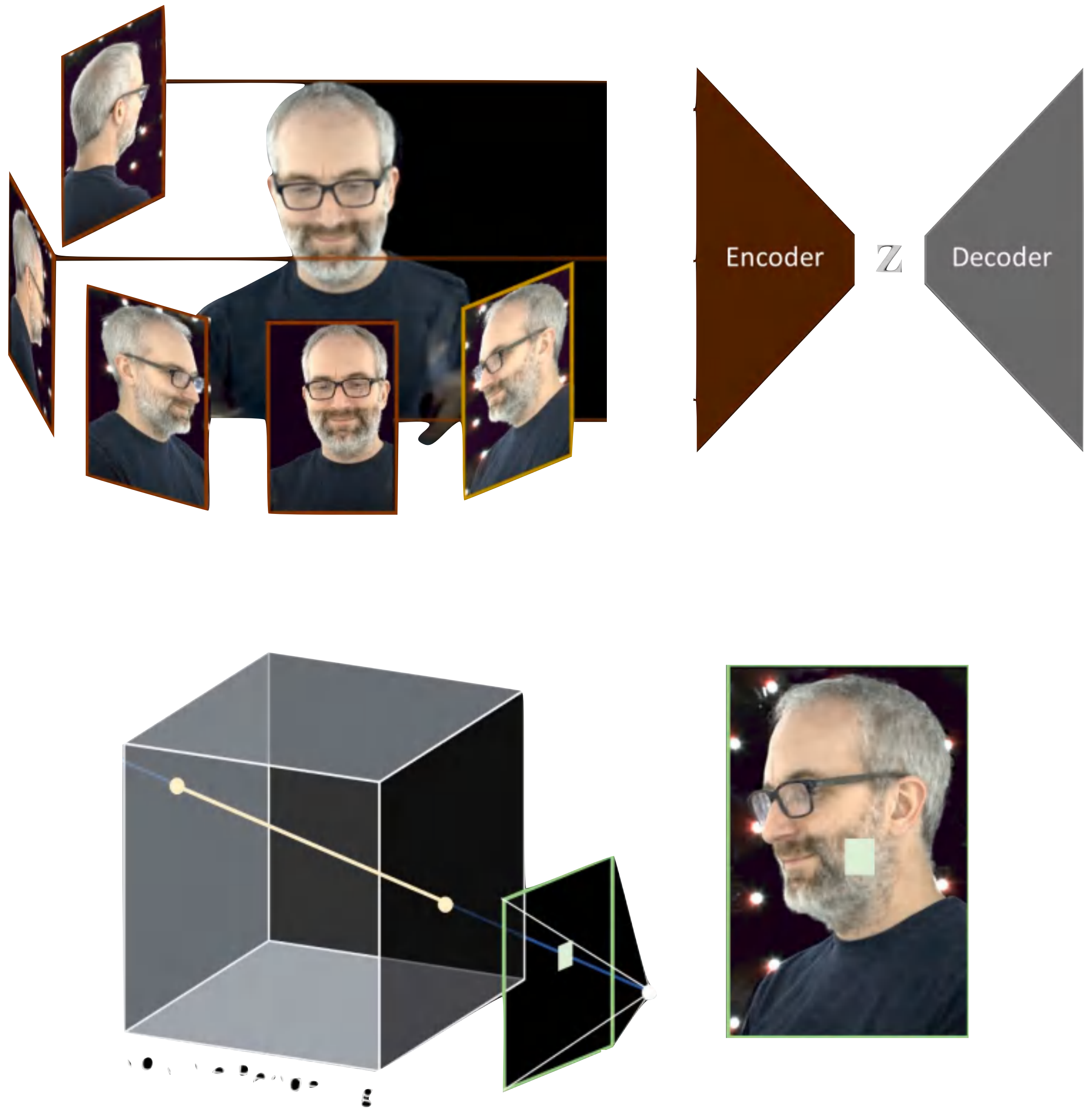
Soft 3D
(Penner & Zhang 2017)
Culmination of non-deep stereo matching techniques

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)

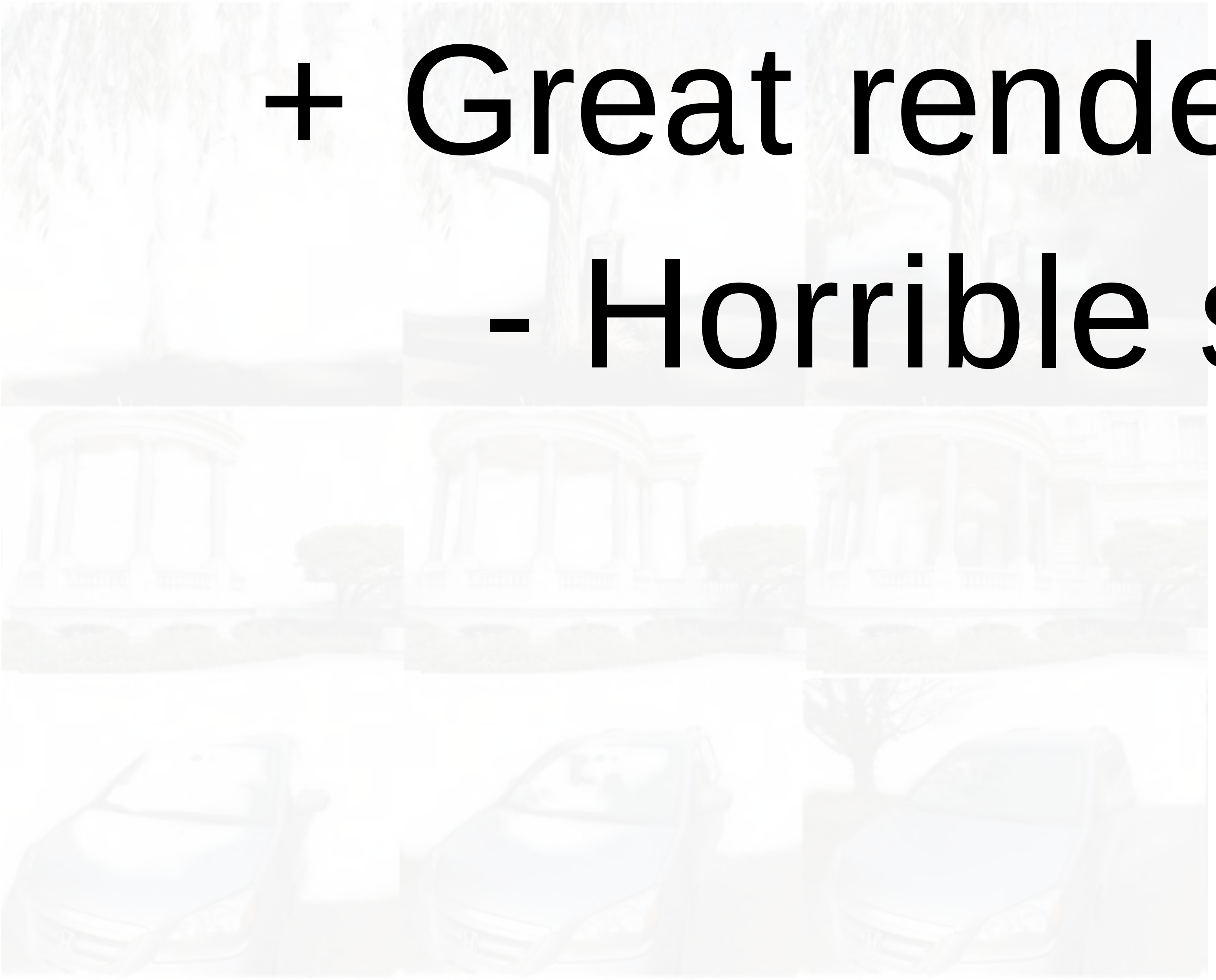
Typical deep learning pipelines - images go into a 3D CNN, big RGBA 3D volume comes out

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

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)

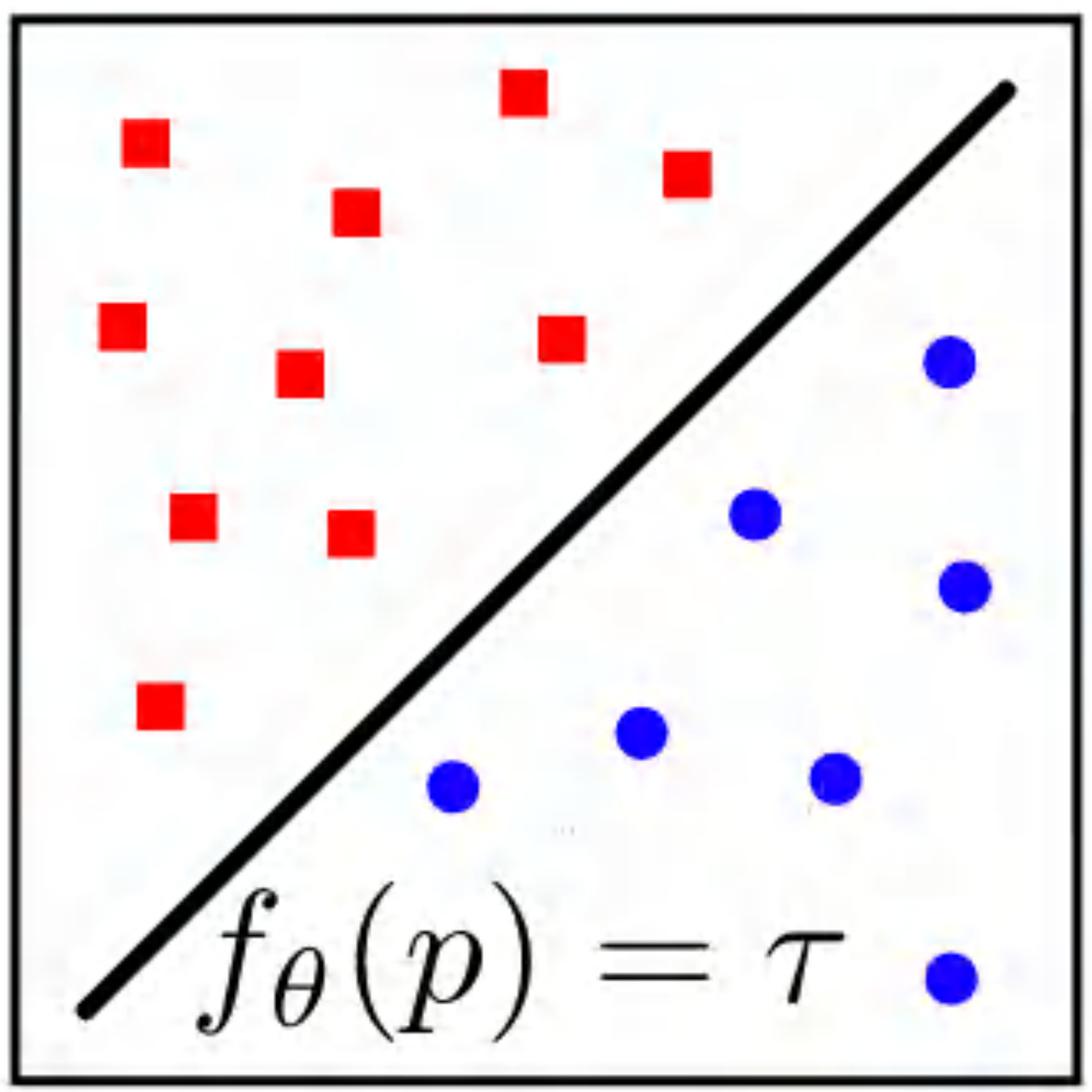
Typical deep learning pipelines - images go into a 3D CNN, big RGBA 3D volume comes out

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)

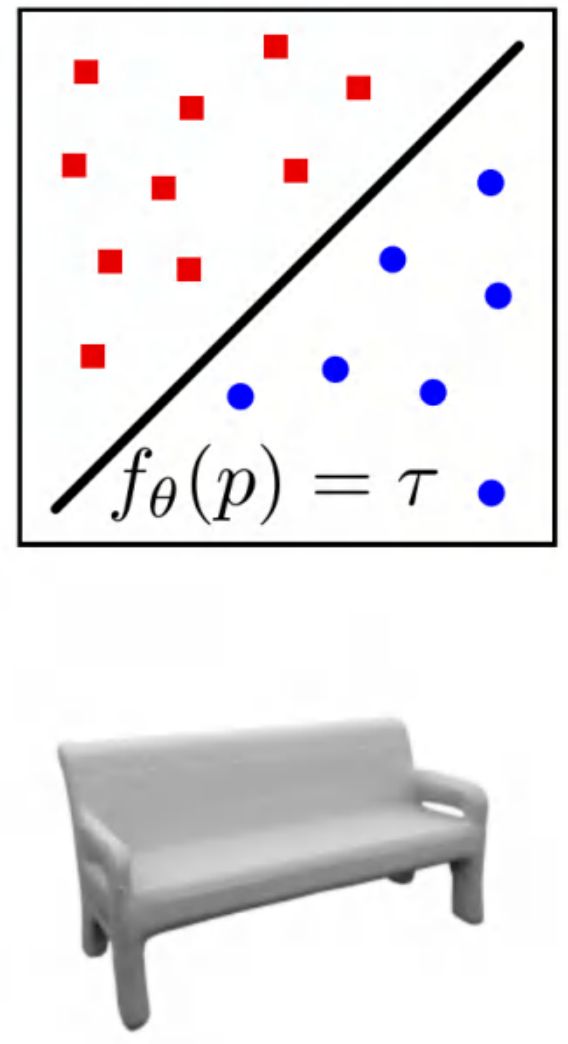
Neural networks as a continuous shape representation



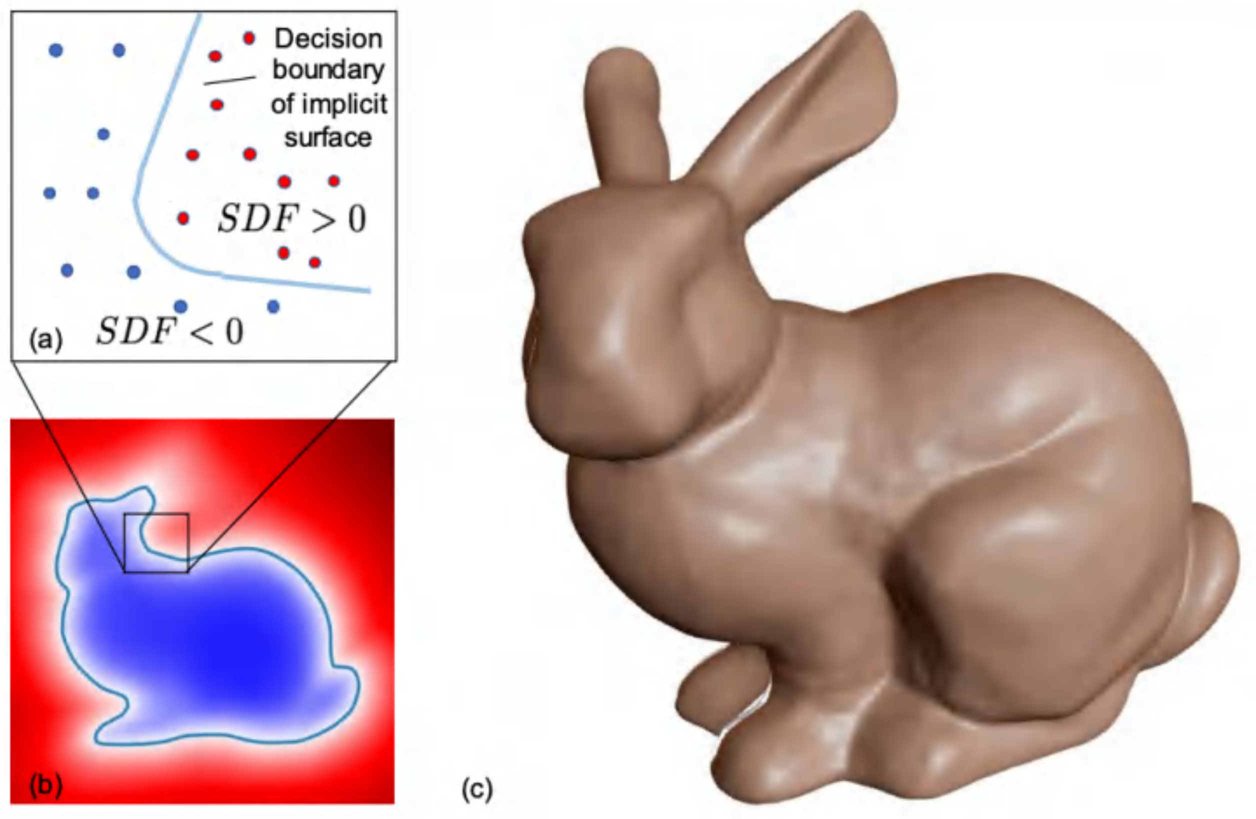
Occupancy Networks, Mescheder et al. CVPR 2019

Neural networks as a continuous shape representation

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

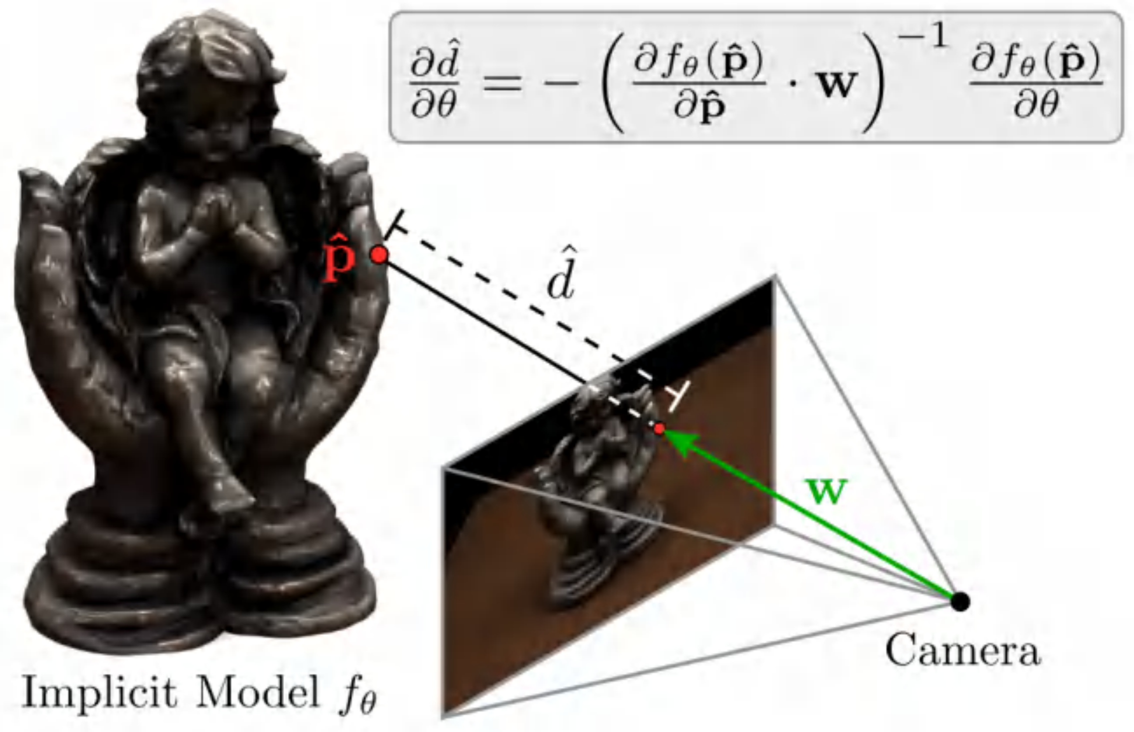


DeepSDF
(Park et al. 2019)
$(x, y, z) \rightarrow$ distance



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

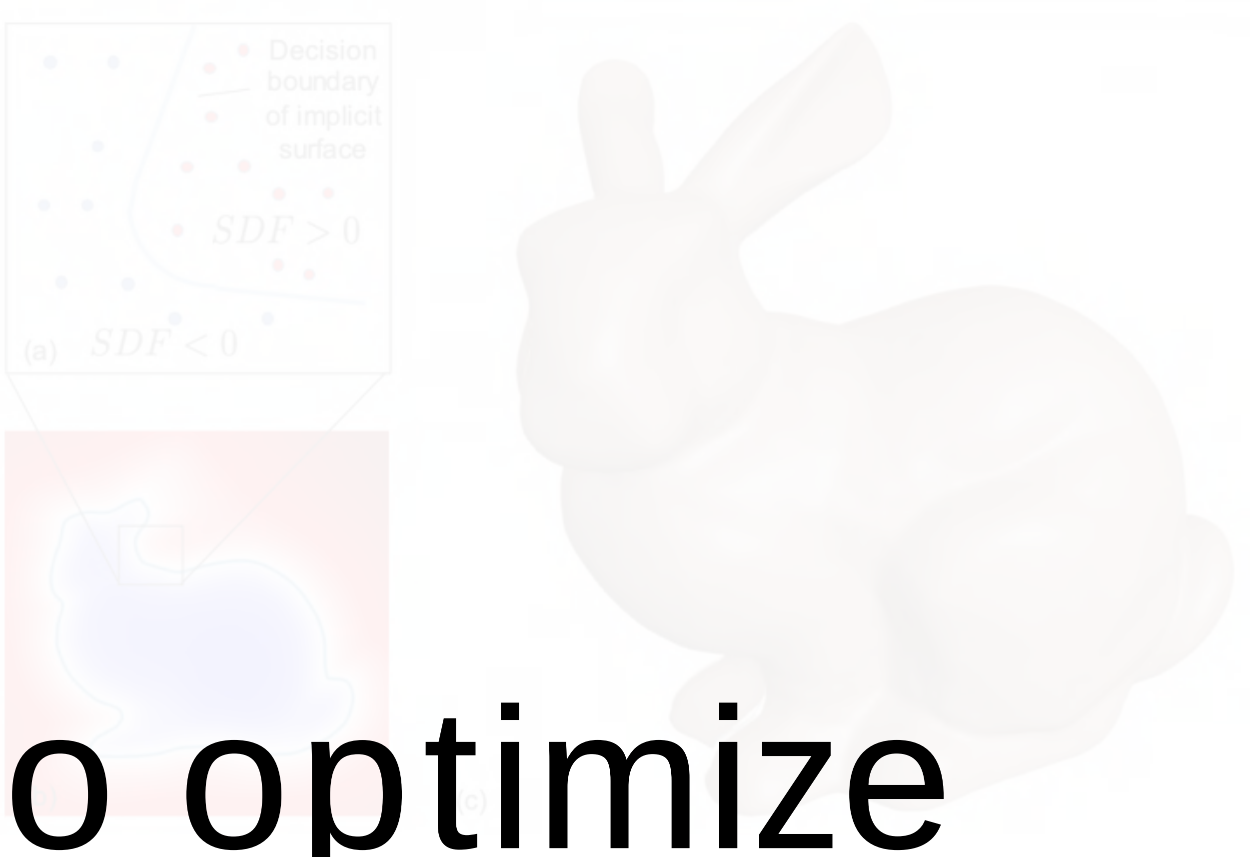
Differentiable Volumetric Rendering
(Niemeyer et al. 2020)
$(x, y, z) \rightarrow$ color, occ.



Neural networks as a continuous shape representation

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

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



- 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)

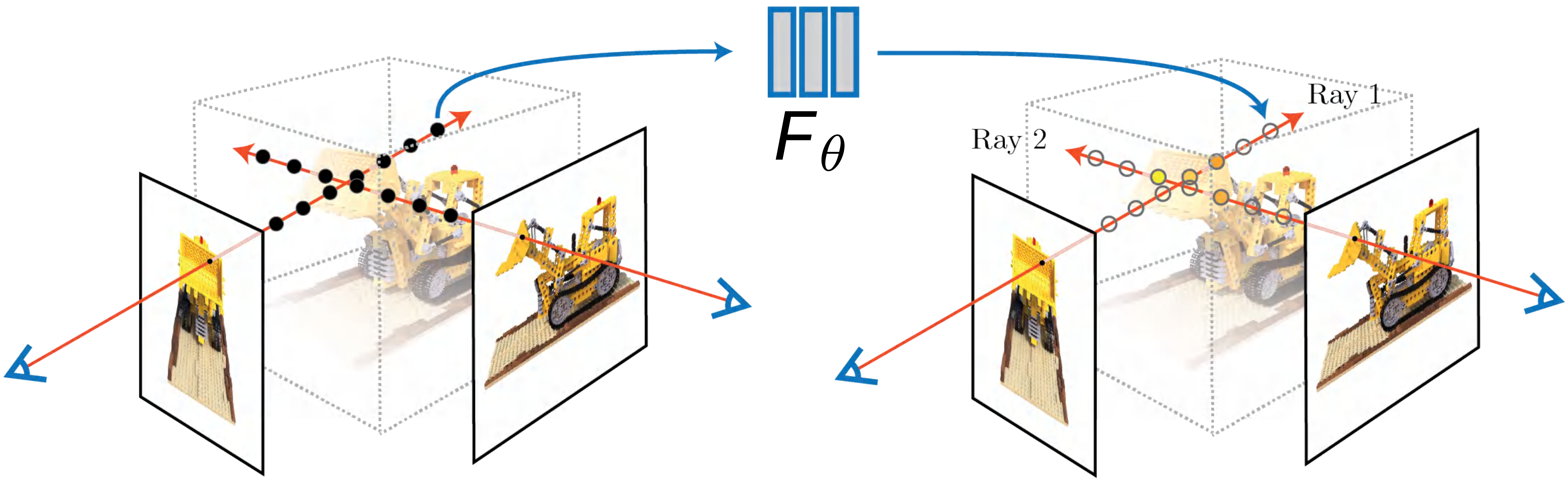
$$(\underbrace{x, y, z}_{\text{Spatial location}}, \underbrace{\theta, \phi}_{\text{Viewing direction}}) \rightarrow$$

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

$$F_\theta$$

Fully-connected
neural network
9 layers,
256 channels

Generate views with traditional volume rendering



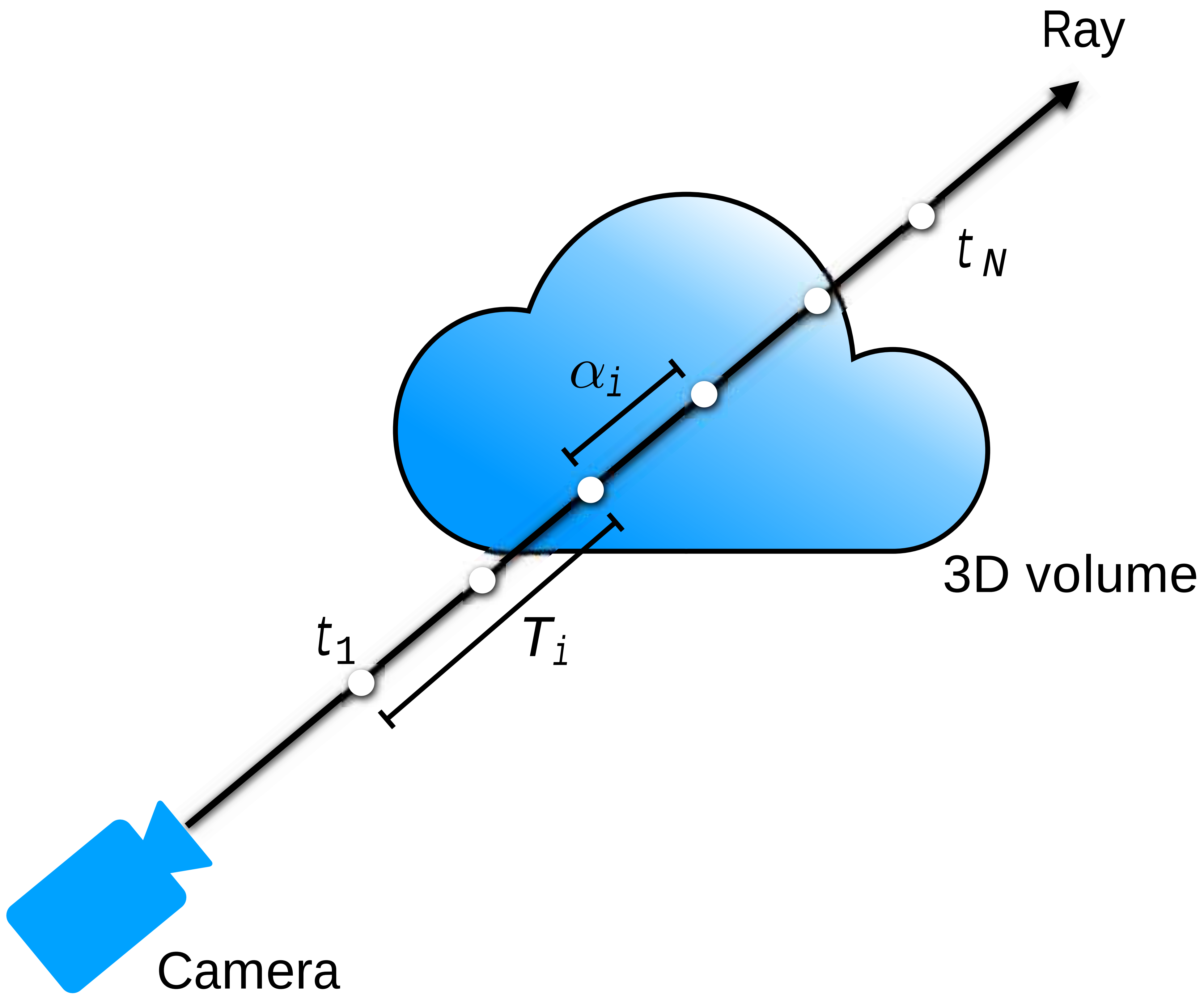
Volume rendering is trivially differentiable

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

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

weights

colors



Volume rendering is trivially differentiable

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

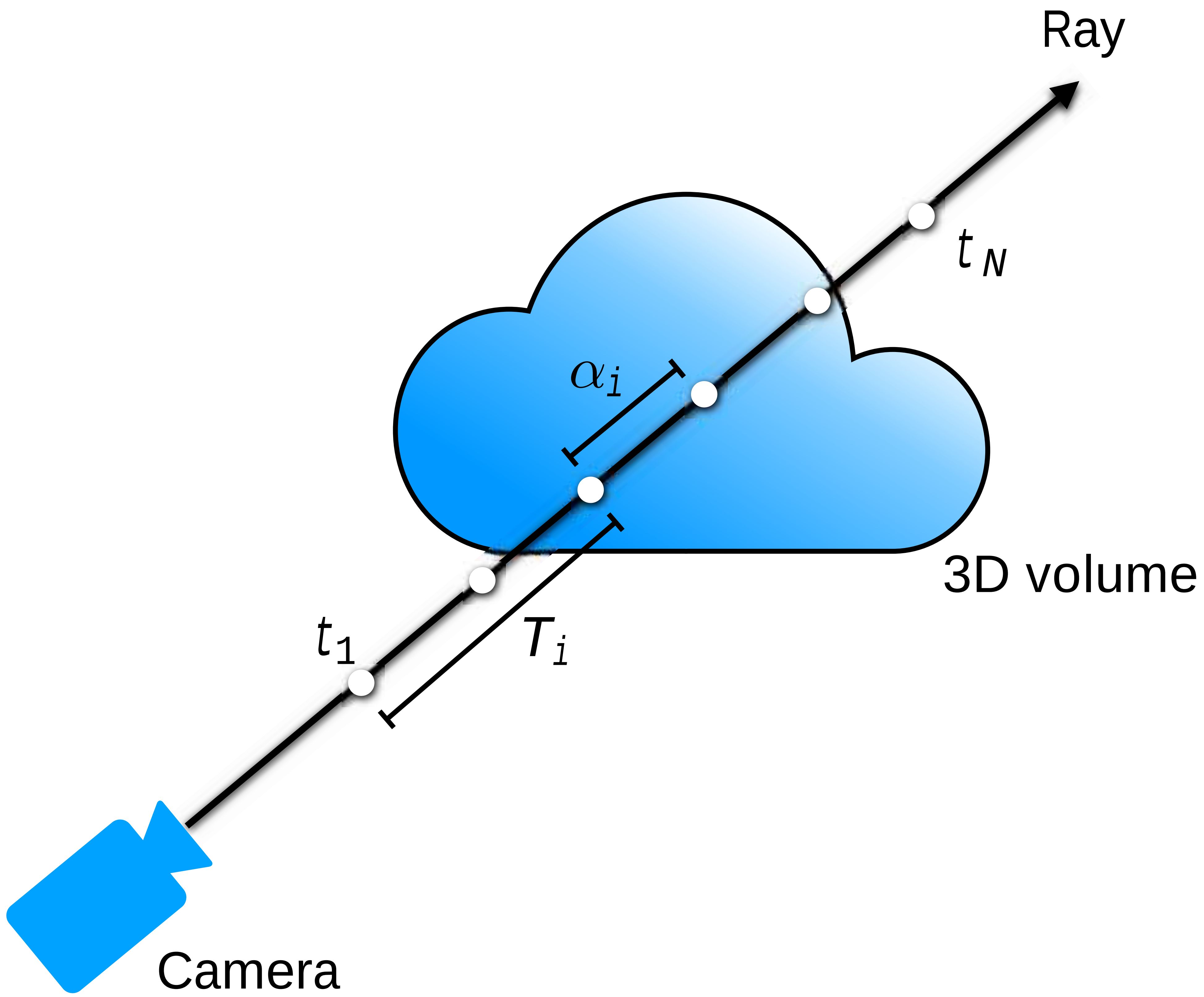
$$C \approx \sum_{i=1}^{N} T_i \alpha_i c_i$$

weights

colors

How much light is blocked earlier along ray:

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



Volume rendering is trivially differentiable

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

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

colors

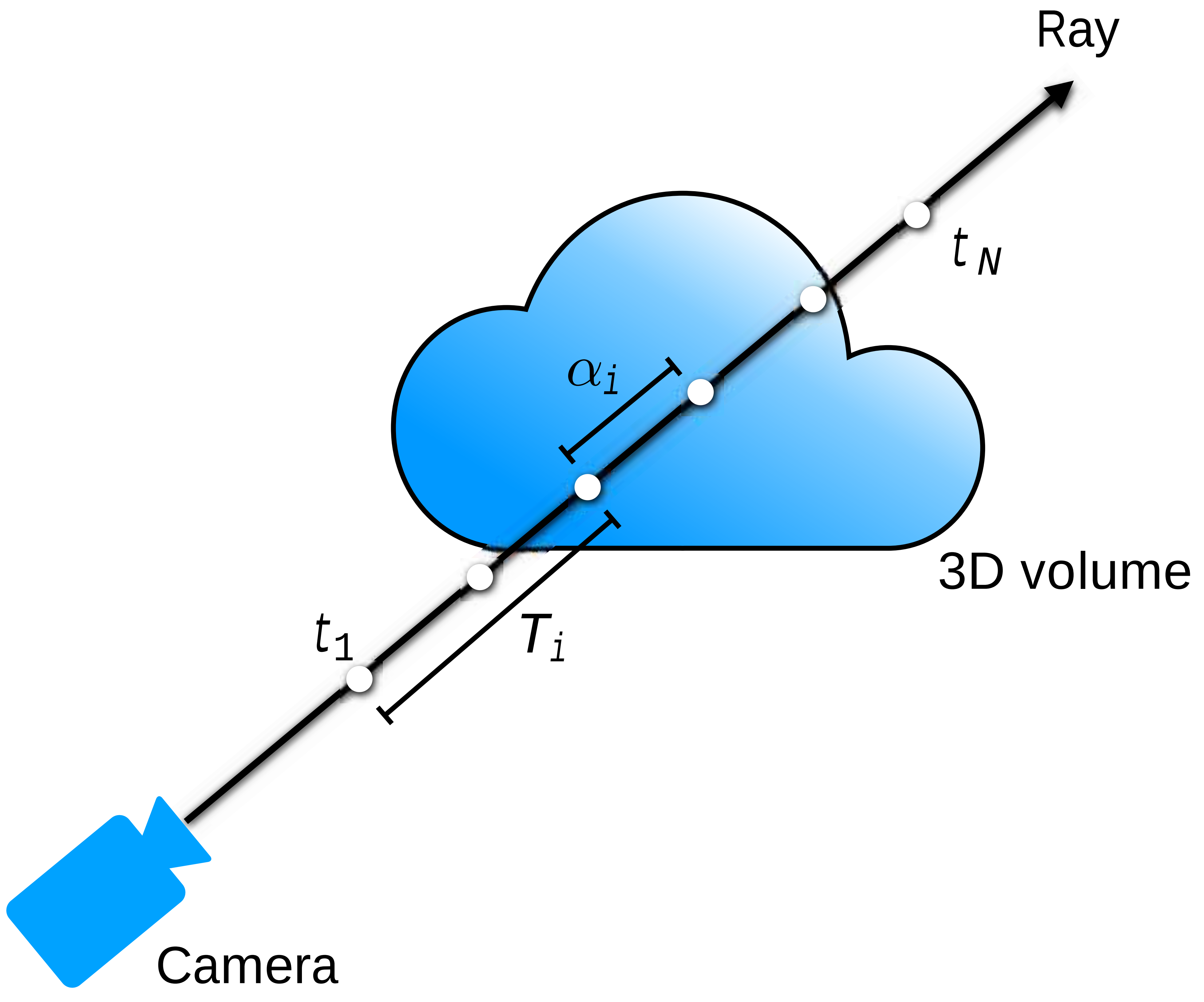
weights

How much light is blocked earlier along ray:

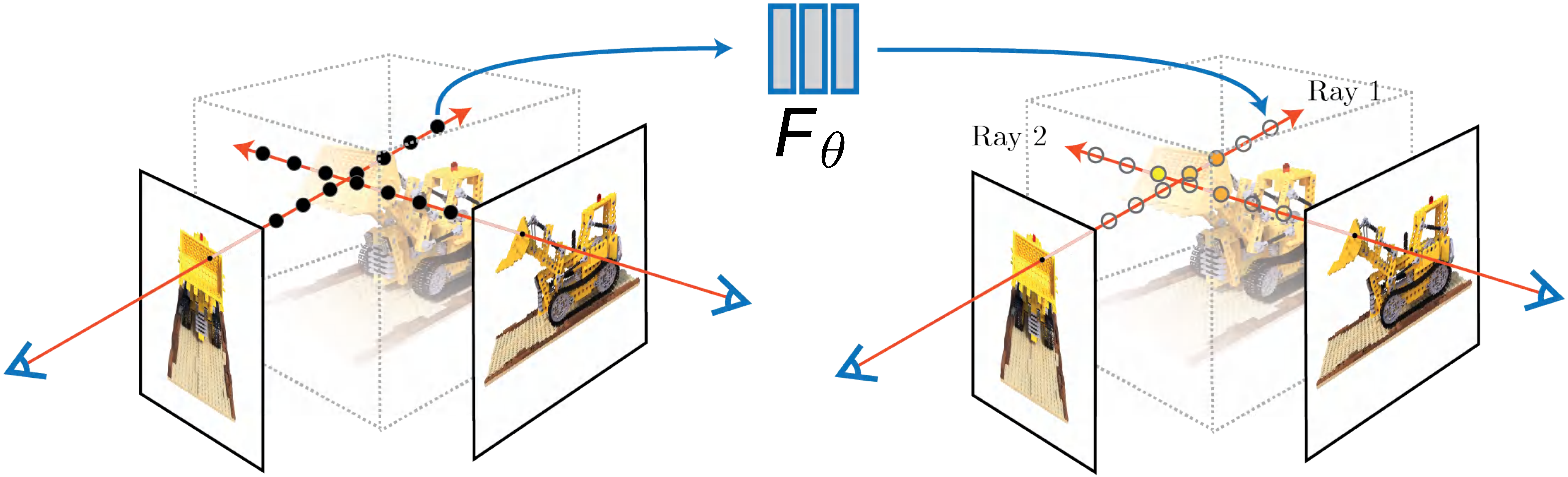
$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j)$$

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

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



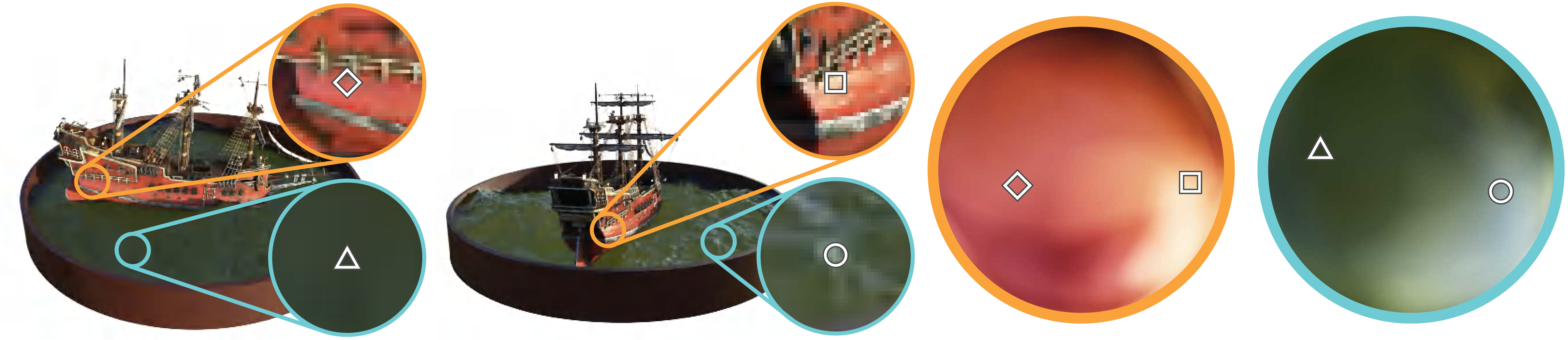
Optimize with gradient descent on rendering loss



$$\min_\theta \sum_i || \text{render}_i(F_\theta) - I_i||^2$$

Training network to reproduce all input views of the scene

Viewing directions as input



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

Results

vs. Prior Work (Implicit / MLP)

SRN [Sitzmann et al. 2019]

NeRF

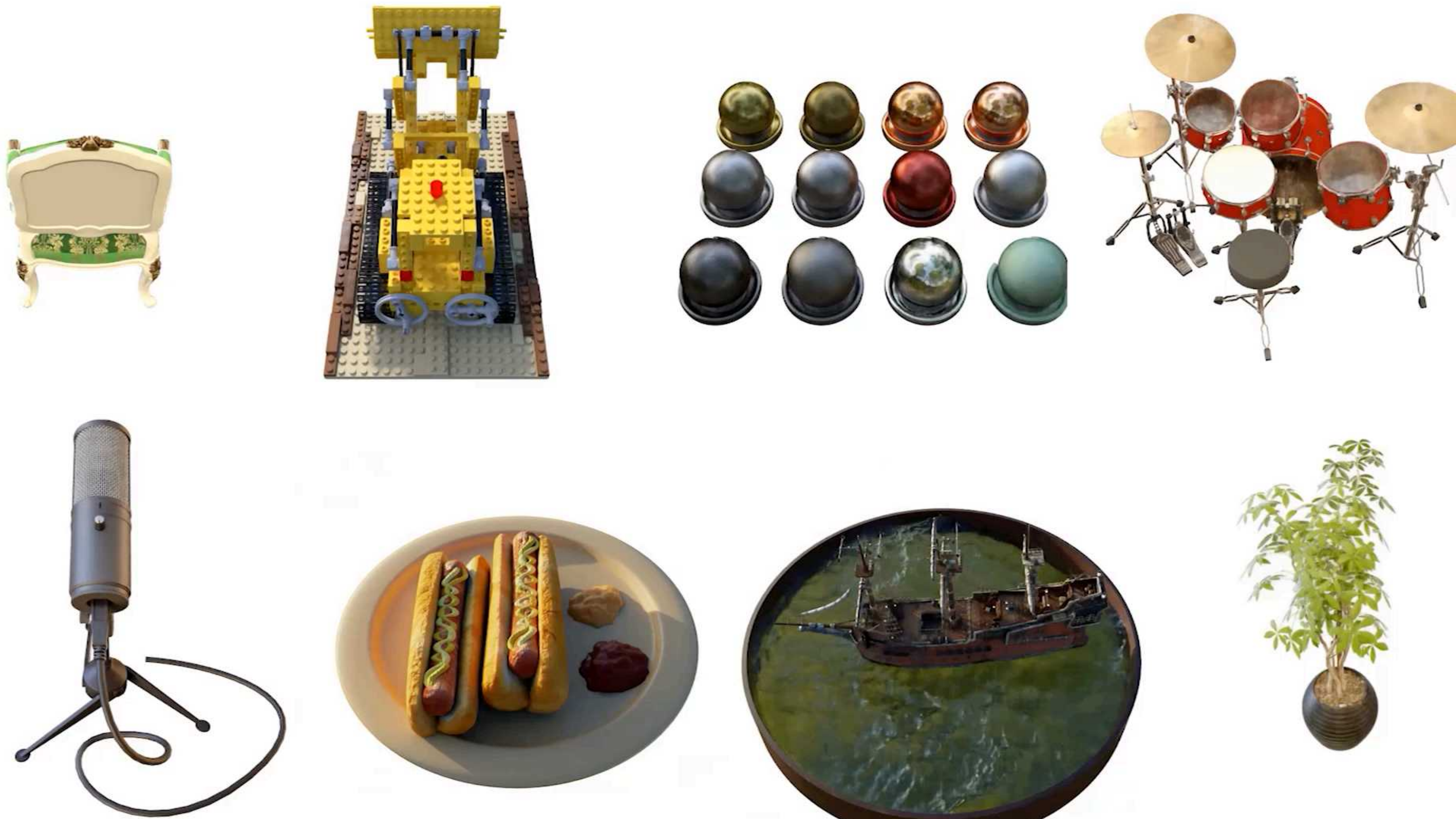
Nearest Input

vs. Prior Work (Implicit / MLP)

SRN [Sitzmann et al. 2019]

NeRF

Nearest Input

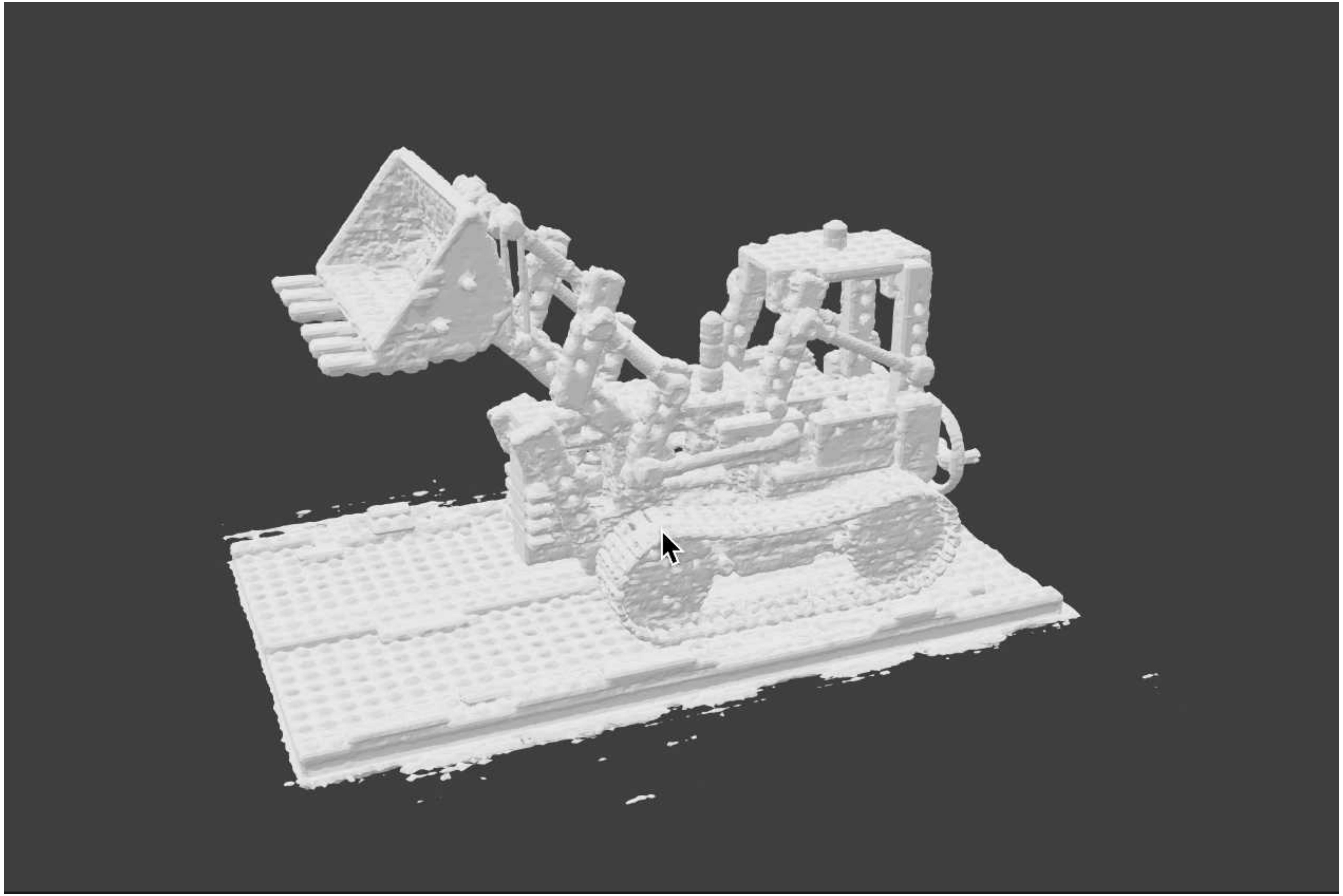


View-Dependent Effects

Detailed Geometry & Occlusion

Detailed Geometry & Occlusion

Meshable



Baking Neural Radiance Fields for Real-Time View Synthesis

arXiv 2021

Peter Hedman Pratul P. Srinivasan Ben Mildenhall Jonathan T. Barron Paul Debevec

Google Research

Paper

Video

Demos

http://nerf.live/

Naive implementation produces blurry results

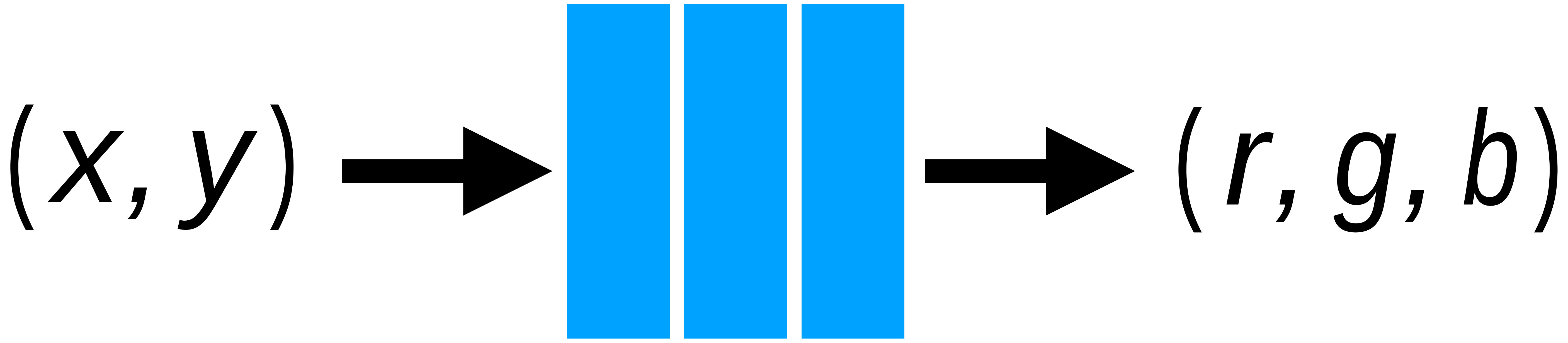
NeRF (Naive)

Naive implementation produces blurry results

NeRF (Naive)

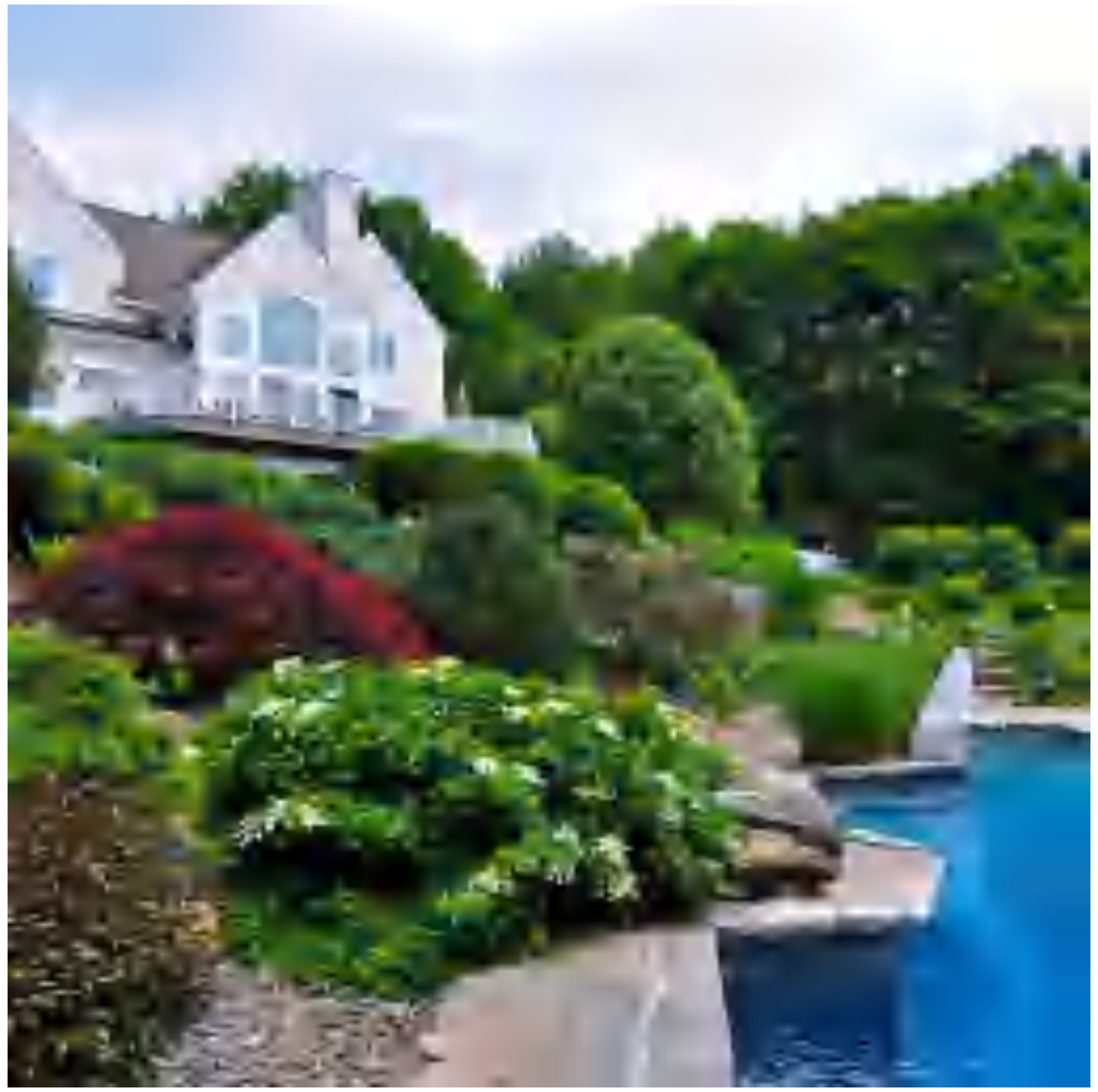
NeRF (with positional encoding)

Toy problem: memorizing a 2D image

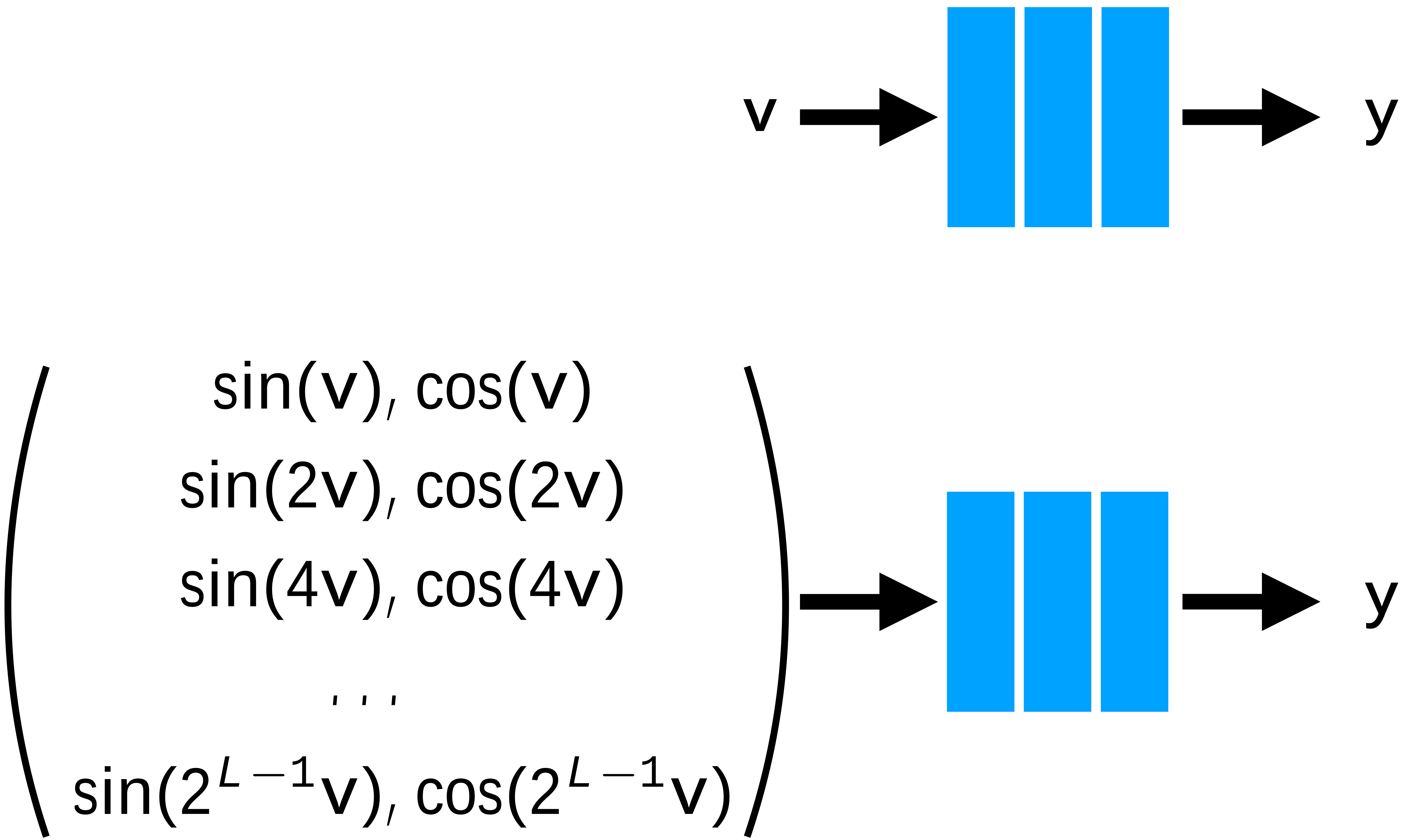


Toy problem: memorizing a 2D image

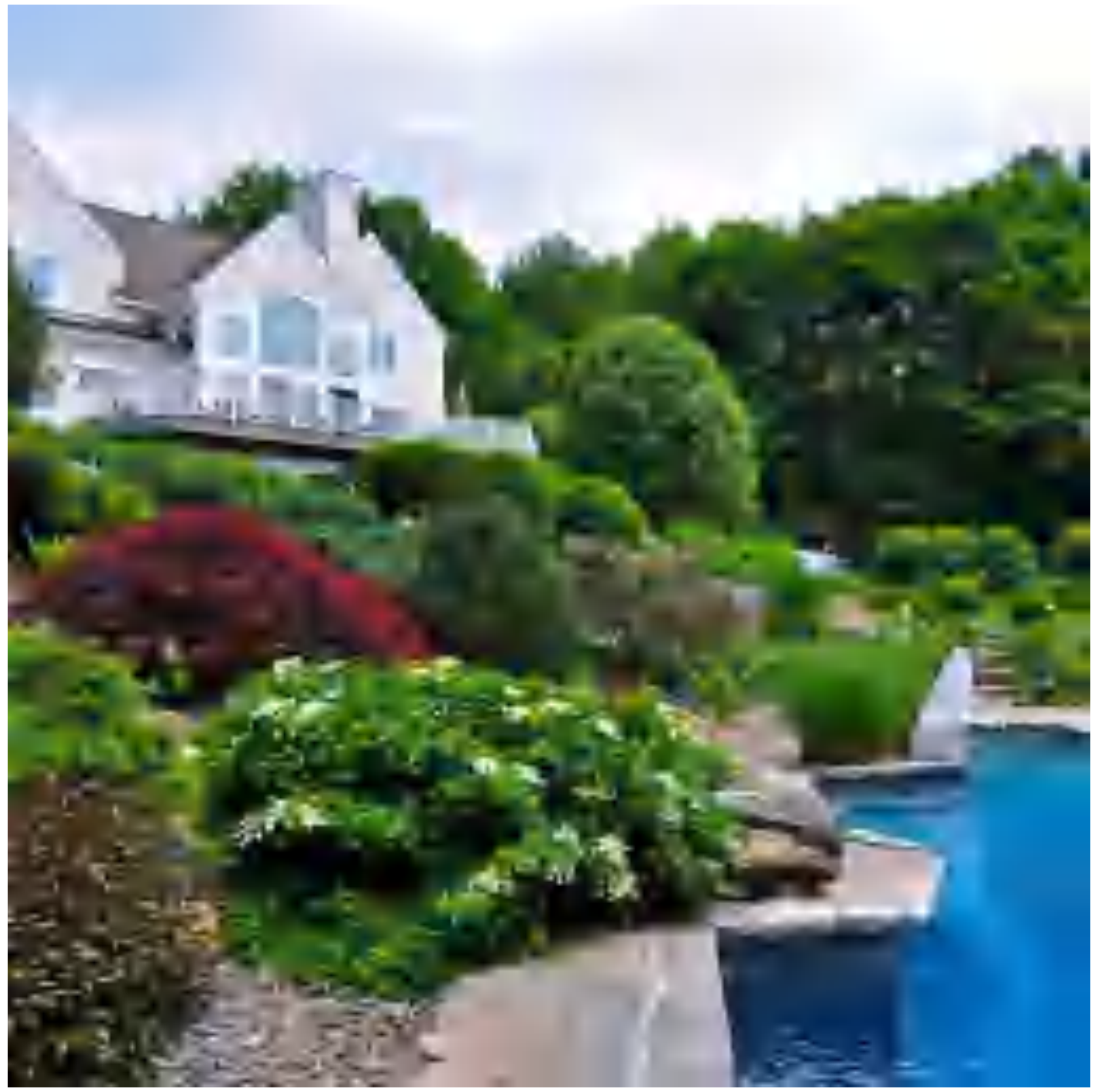
Ground truth image



Standard fully-connected net

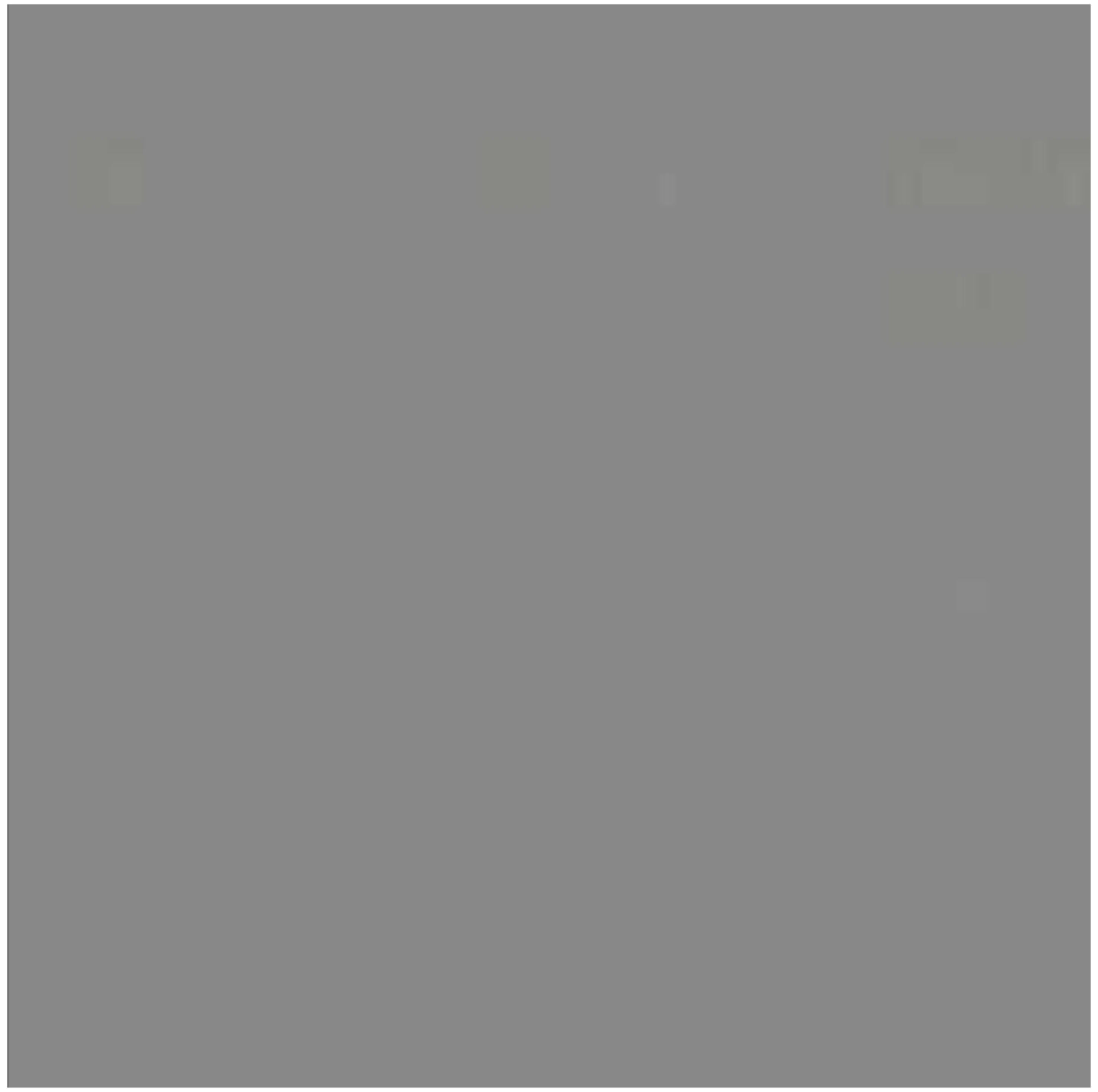

v
y
sin(v), cos(v)
sin(2v), cos(2v)
sin(4v), cos(4v)
...
$\sin(2^{L-1}v), \cos(2^{L-1}v)$
y

Ground truth image



Standard fully-connected net

With Positional Encoding



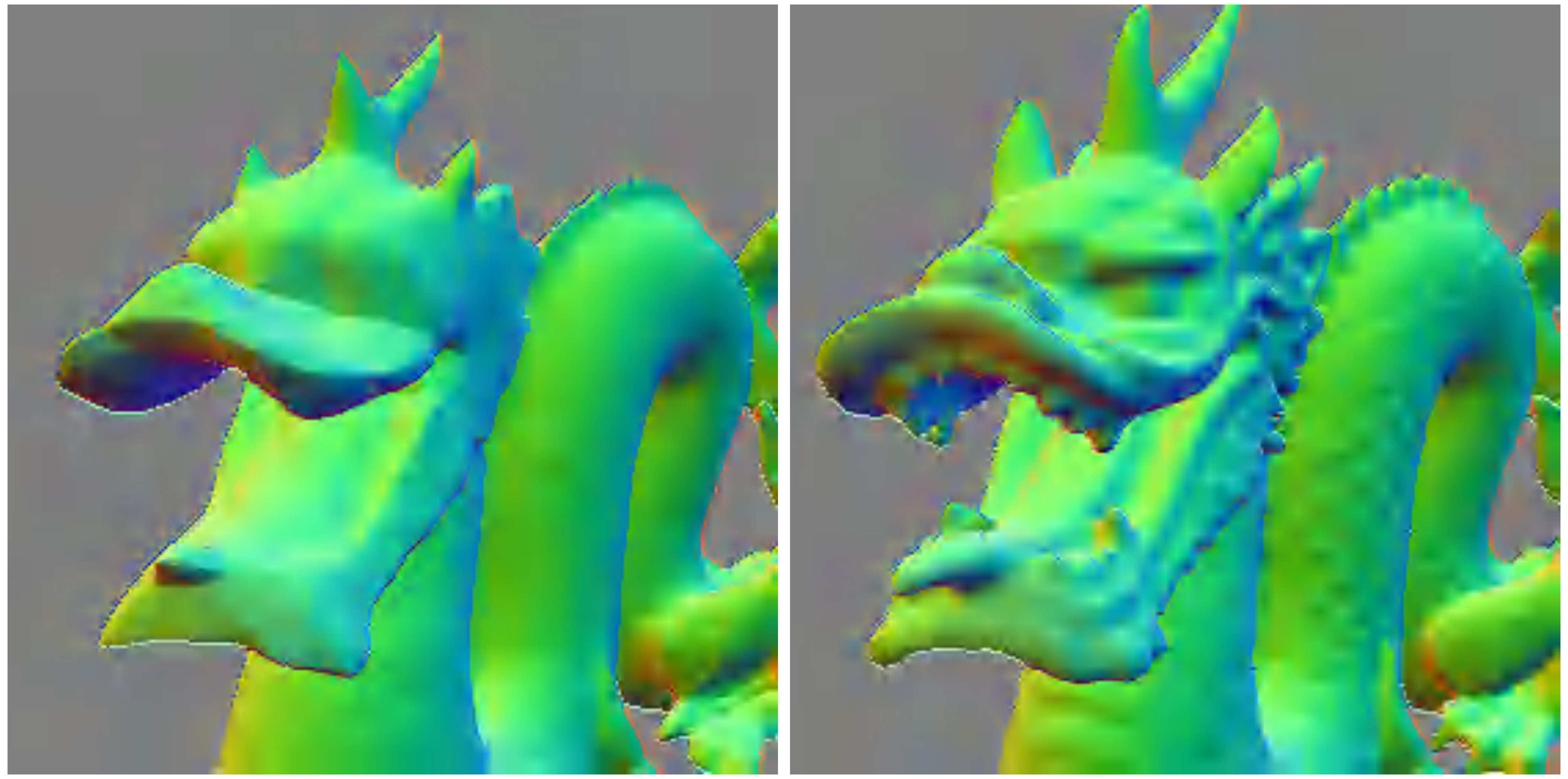
Positional encoding also directly improves our scene representation!

NeRF (Naive)

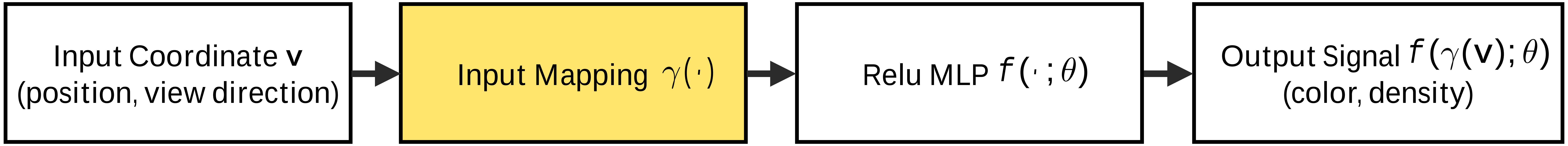
NeRF (with positional encoding)

Why?

Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains



Matthew Tancik*, Pratul Srinivasan*, Ben Mildenhall*,
Sara Fridovich-Keil, Nithin Ragahavan, Utkarsh Singhal,
Ravi Ramamoorthi, Jonathan T. Barron, Ren Ng



Positional Encoding [1]: $\gamma(\mathbf{v}) = \left[\cos(2^0\mathbf{v}), \sin(2^0\mathbf{v}), \ldots, \cos(2^{L-1}\mathbf{v}), \sin(2^{\boxed{L}-1}\mathbf{v})\right]$

Random Fourier Features [2]: $\gamma(\mathbf{v}) = [\cos(\mathbf{B}\mathbf{v}), \sin(\mathbf{B}\mathbf{v})]$ $\quad \mathbf{B} \sim N(0, \boxed{\sigma}^2)$

[1] Vaswani et al.. NeurIPS, 2017
[2] Rahimi & Recht. NeurIPS, 2007

Neural Tangent Kernel

$$f(\mathrm{x};\theta) \approx \sum_i (\mathrm{K}^{-1}\mathrm{y})_i k(\mathrm{x}_i, \mathrm{x})$$

Under certain conditions,
neural networks are kernel regression(!)

$$k(\mathrm{x}_i, \mathrm{x}_j) = h_{\mathrm{NTK}}(\langle \mathrm{x}_i, \mathrm{x}_j \rangle)$$

$$h_{\mathrm{NTK}} : \mathrm{R} \to \mathrm{R}$$

ReLU MLPs correspond to a “dot product” kernel

Jacot et al., NeurIPS, 2018, Arora, et al., ICML, 2019, Basri et al., 2020., Du et al., ICLR, 2019., Lee et al., NeurIPS, 2019

Dot Product of Fourier Features

$$
\begin{aligned}
\langle \gamma(\mathbf{v}_1), \gamma(\mathbf{v}_2) \rangle &= \sum_j \left(\cos(\mathbf{b}_j^\mathsf{T} \mathbf{v}_1) \cos(\mathbf{b}_j^\mathsf{T} \mathbf{v}_2) + \sin(\mathbf{b}_j^\mathsf{T} \mathbf{v}_1) \sin(\mathbf{b}_j^\mathsf{T} \mathbf{v}_2) \right) \\
&= \sum_j \cos\left(\mathbf{b}_j^\mathsf{T} (\mathbf{v}_1 - \mathbf{v}_2) \right) \qquad \text{(cosine difference trig identity)} \\
&\triangleq h_\gamma(\mathbf{v}_1 - \mathbf{v}_2)
\end{aligned}
$$

Fourier Features $\rightarrow$ *stationary* kernel

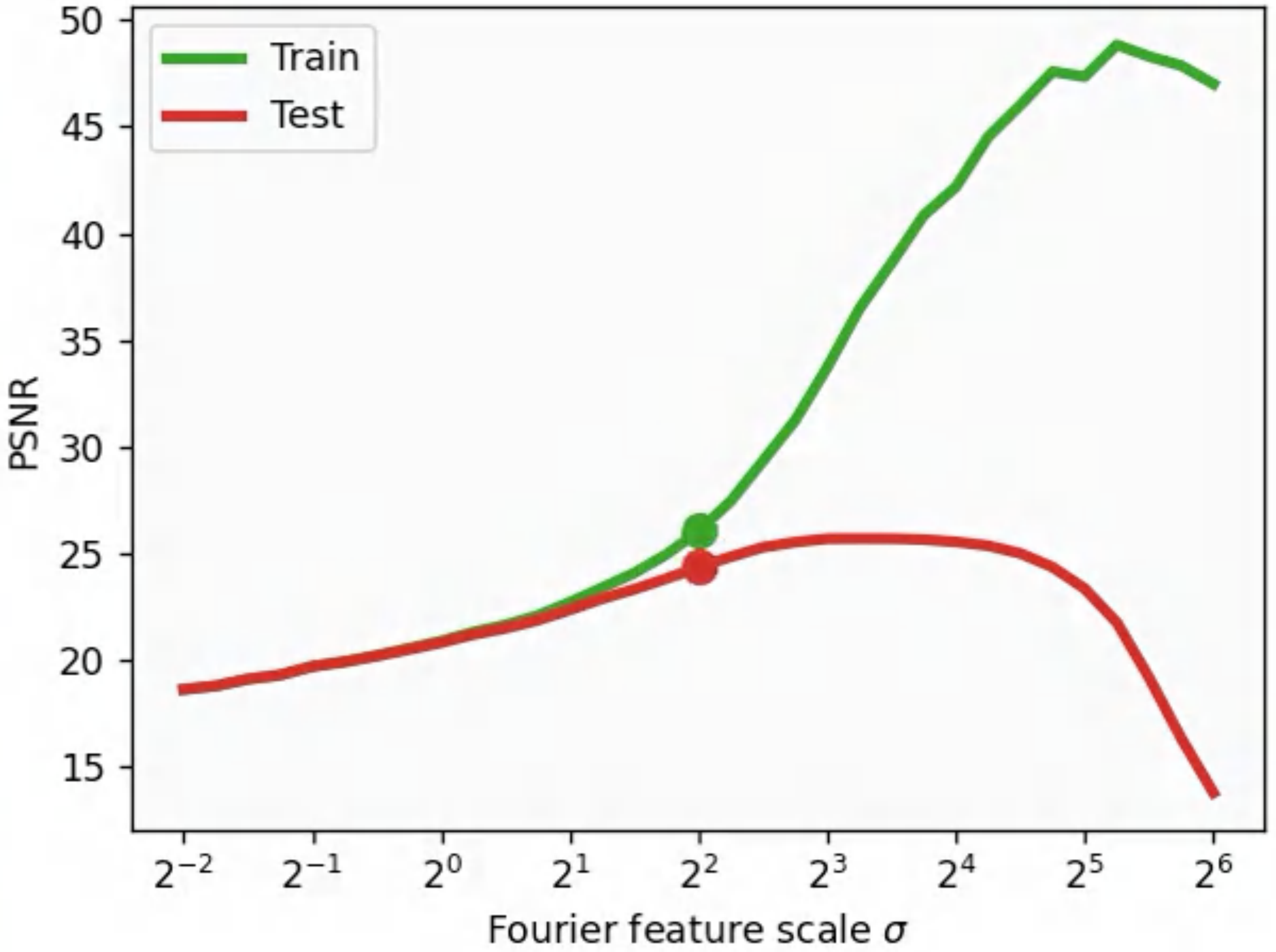
Resulting *composed* NTK is stationary

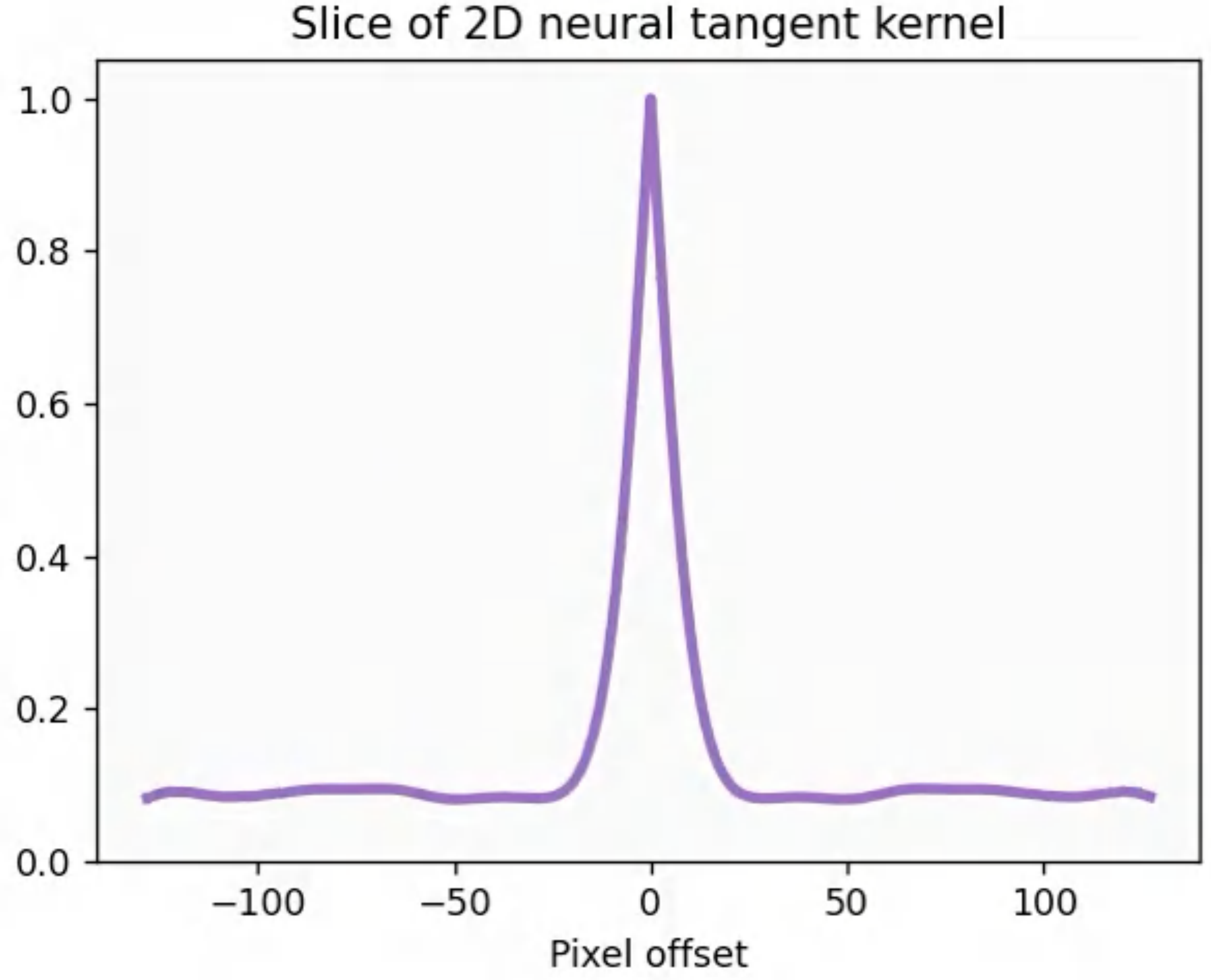
$$h_{\mathrm{NTK}}\Big(\langle\gamma(\mathbf{v})_i, \gamma(\mathbf{v})_j\rangle\Big) = h_{\mathrm{NTK}}(h_\gamma(\mathbf{v}_i - \mathbf{v}_j))$$

Resulting network regression function is a *convolution*

$$\hat{f} = (h_{\mathrm{NTK}} \circ h_\gamma) * \sum_{i=1}^{n} w_i \delta_{\mathbf{v}_i}$$

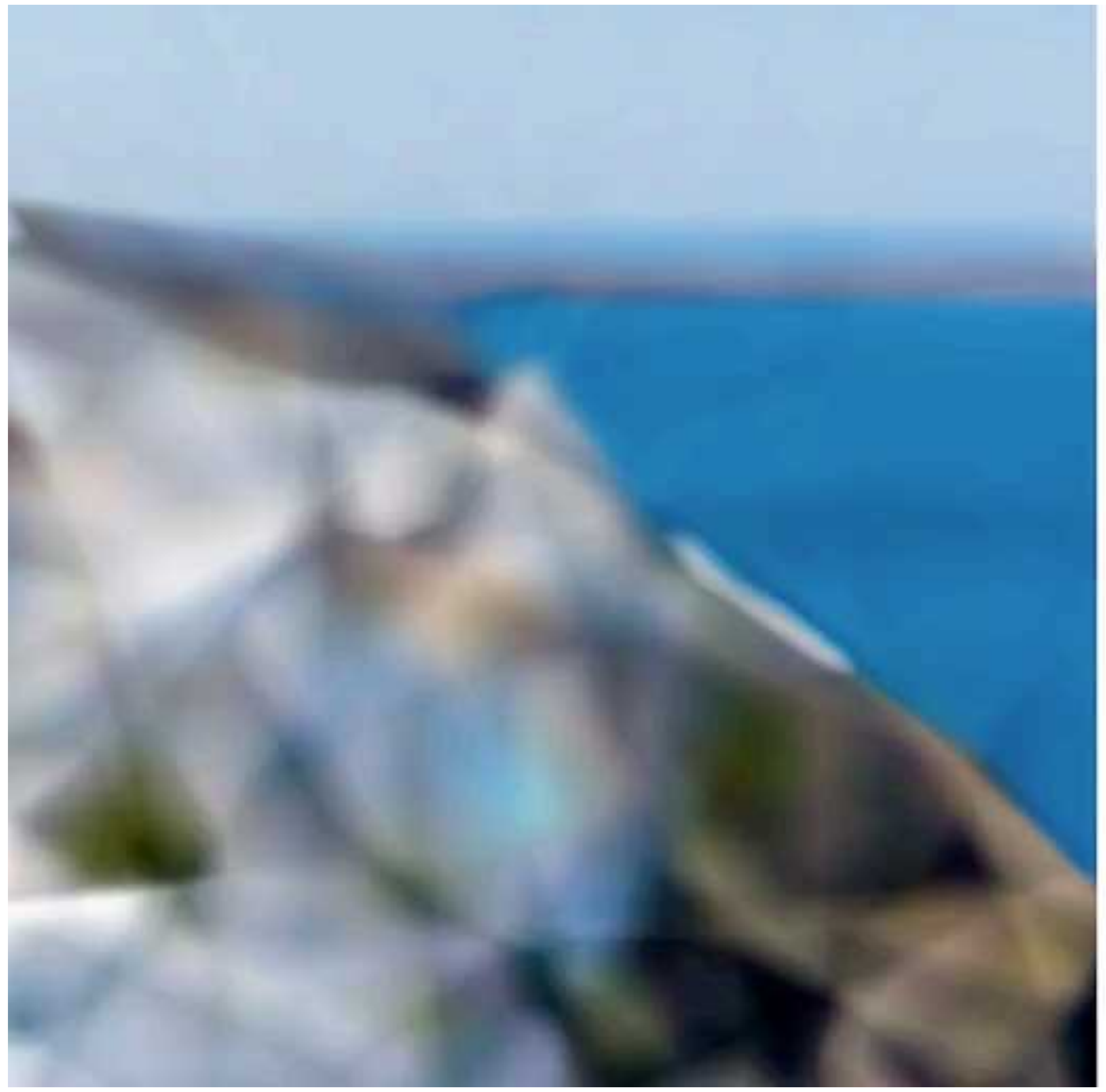
Mapping bandwidth controls underfitting / overfitting

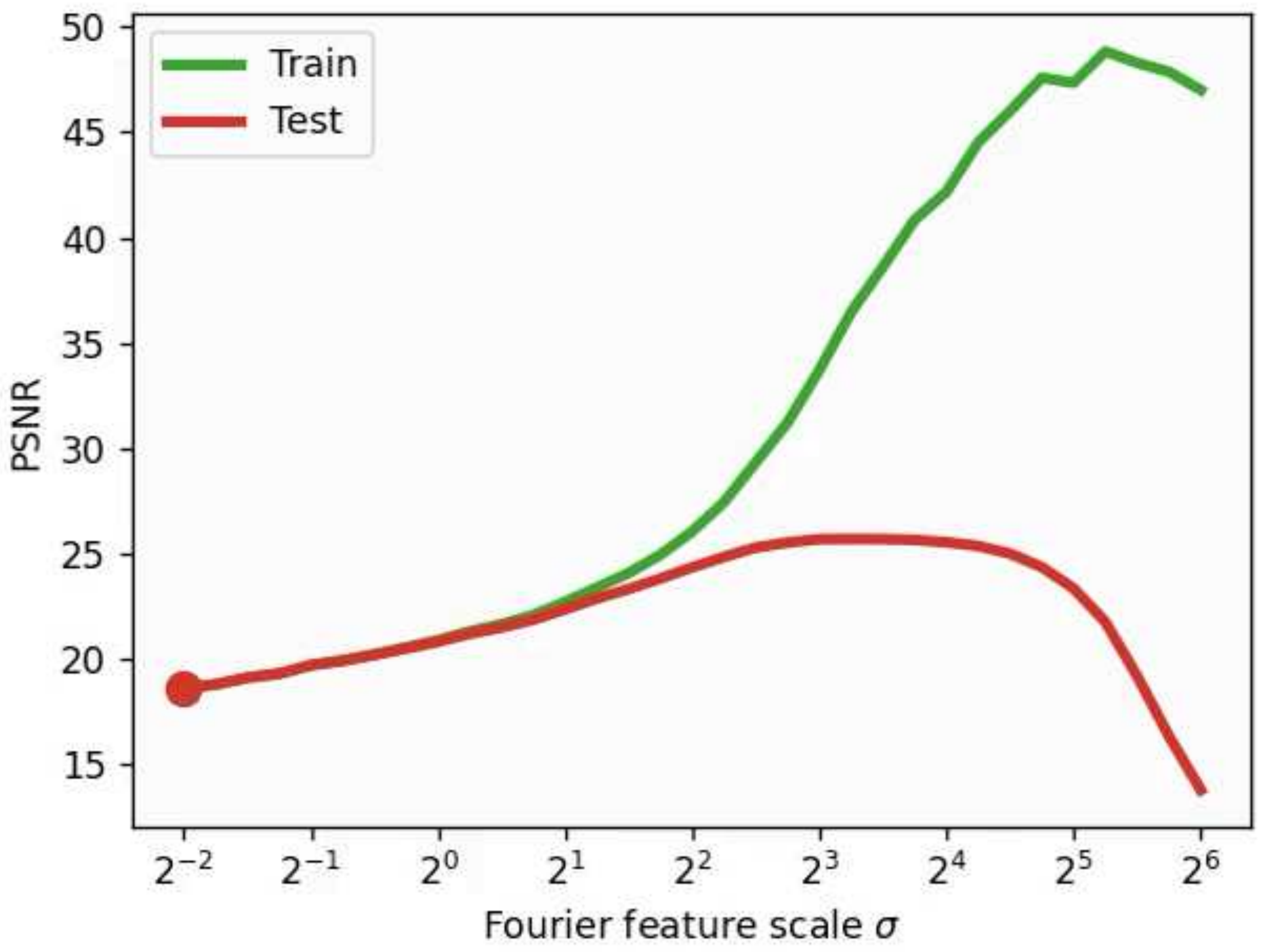


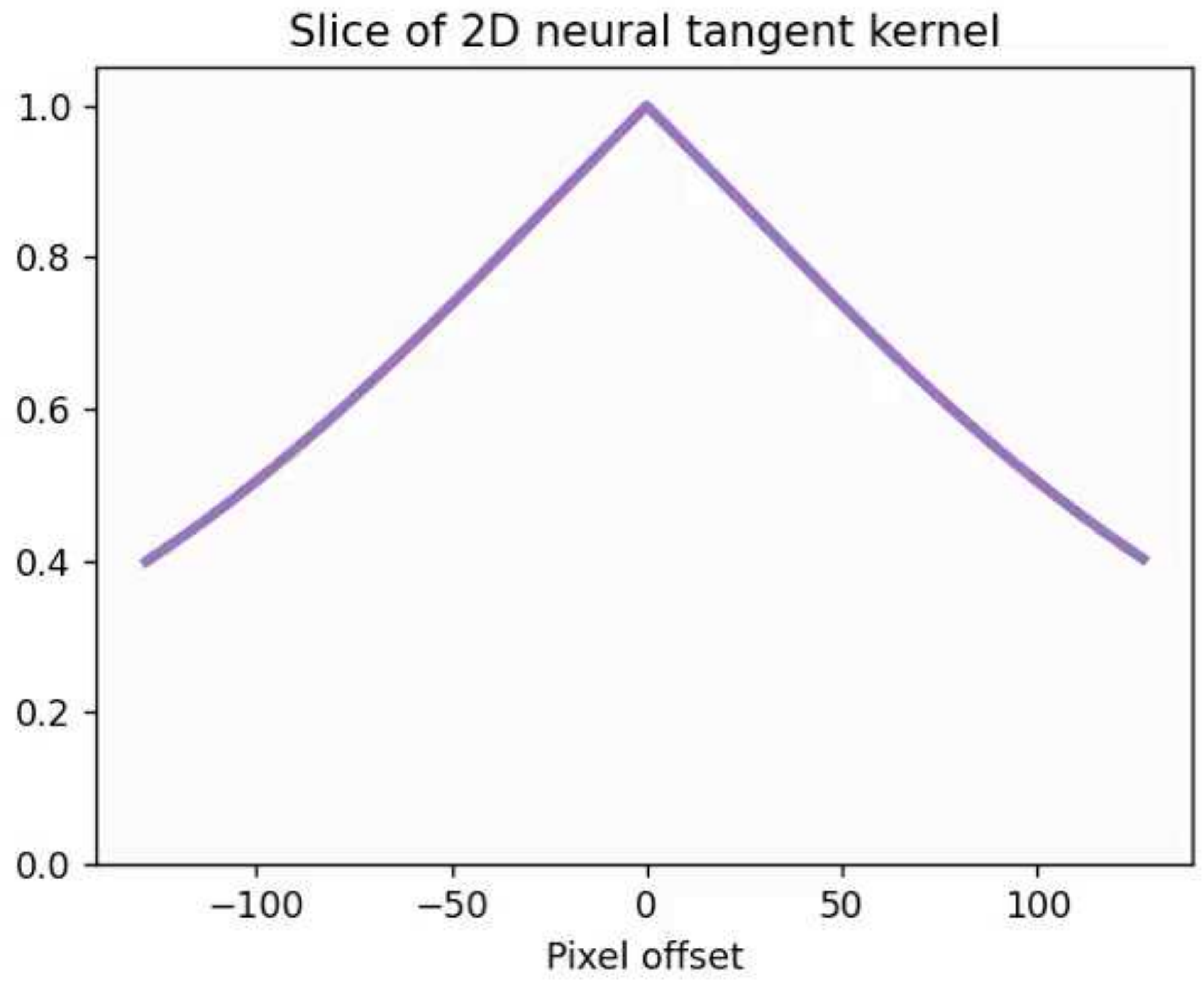


$\gamma(\mathbf{v}) = [\cos(\mathbf{B}\mathbf{v}), \sin(\mathbf{B}\mathbf{v})]$ $\qquad$ $\mathbf{B} \sim N(0, \boxed{\sigma}^2)$

Mapping bandwidth controls underfitting / overfitting

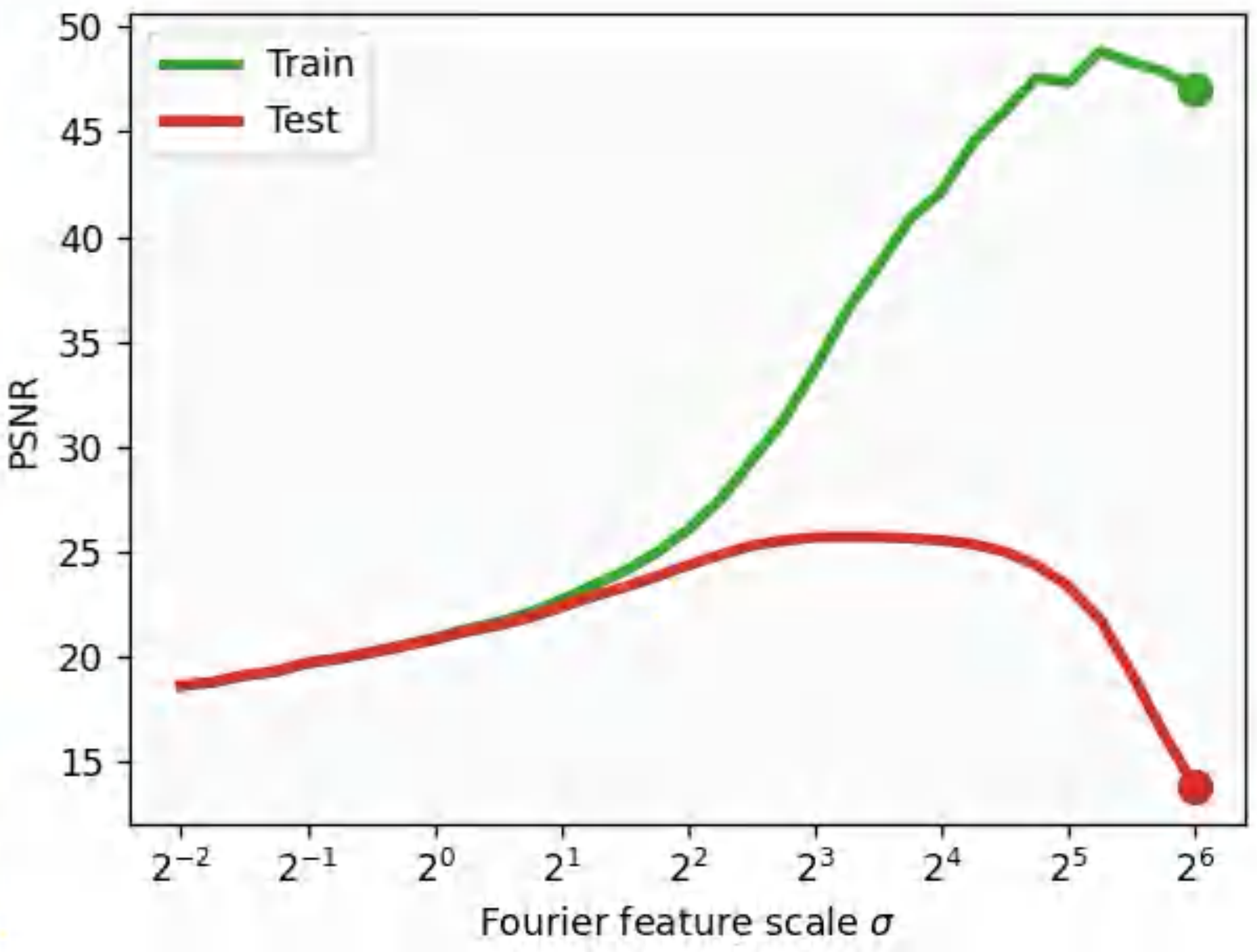


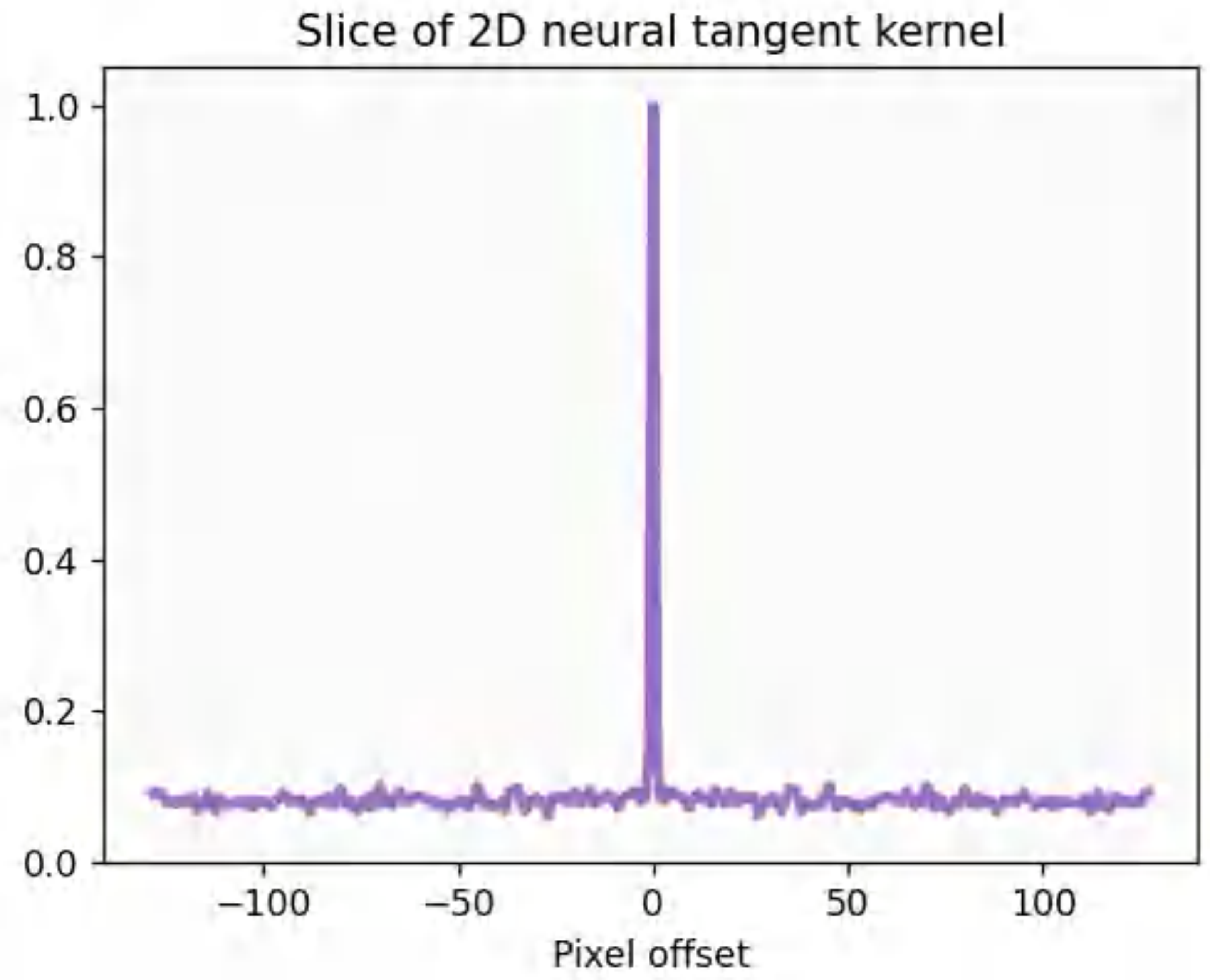




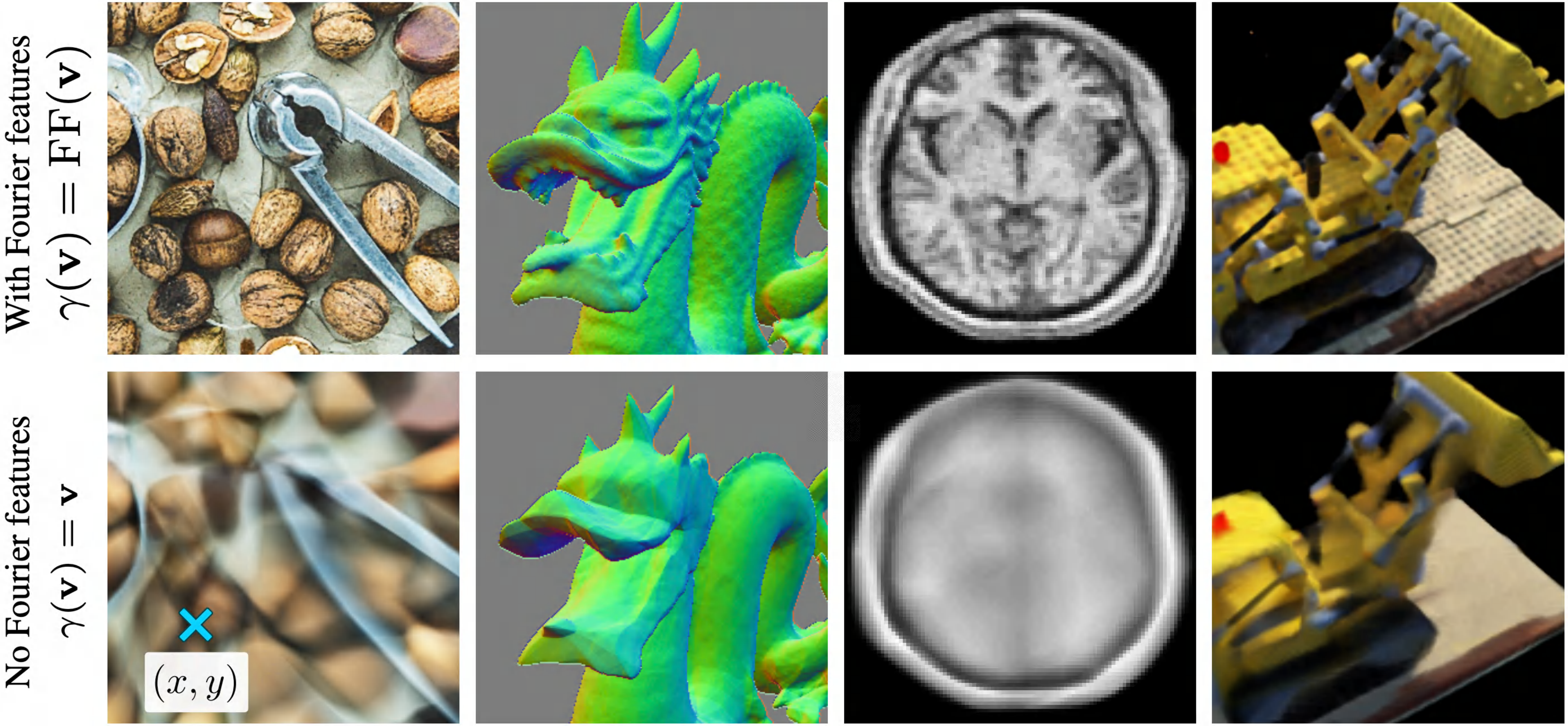
$$\gamma(\mathbf{v}) = [\cos(\mathbf{B}\mathbf{v}), \sin(\mathbf{B}\mathbf{v})] \qquad \mathbf{B} \sim \mathcal{N}(0, \boxed{\sigma}^2)$$

Mapping bandwidth controls underfitting / overfitting





$$\gamma(\mathbf{v}) = [\cos(\mathbf{B}\mathbf{v}), \sin(\mathbf{B}\mathbf{v})] \qquad \mathbf{B} \sim \mathcal{N}(0, \boxed{\sigma}^2)$$



(b) Image regression $(x,y) \rightarrow$ RGB

(c) 3D shape regression $(x,y,z) \rightarrow$ occupancy

(d) MRI reconstruction $(x,y,z) \rightarrow$ density

(e) Inverse rendering $(x,y,z) \rightarrow$ RGB, density

Try It!

```
B = SCALE * np.random.normal(shape=(input_dims, NUM_FEATURES))
x = np.concatenate([np.sin(x @ B), np.cos(x @ B)], axis=-1)
x = nn.Dense(x, features=256)
```

Coordinate based neural representation

$\neq$

a magic black box that learns things and generalizes

Coordinate based neural representation

$\neq$

a magic black box that learns things and generalizes

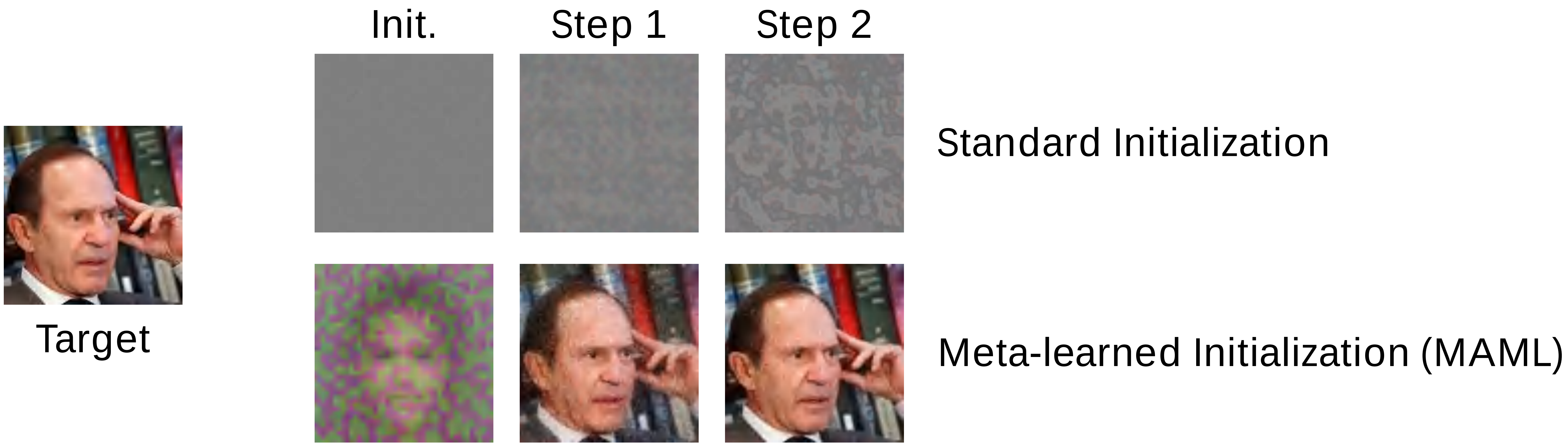
Coordinate based neural representation

=

a tiny n-dimensional lookup table with extremely high resolution

Learned Initializations for Optimizing Coordinate-Based Neural Representations

Matthew Tancik*[1] Ben Mildenhall*[1] Terrance Wang[1] Divi Schmidt[1]
Pratul P. Srinivasan[2] Jonathan T. Barron[2] Ren Ng[1]



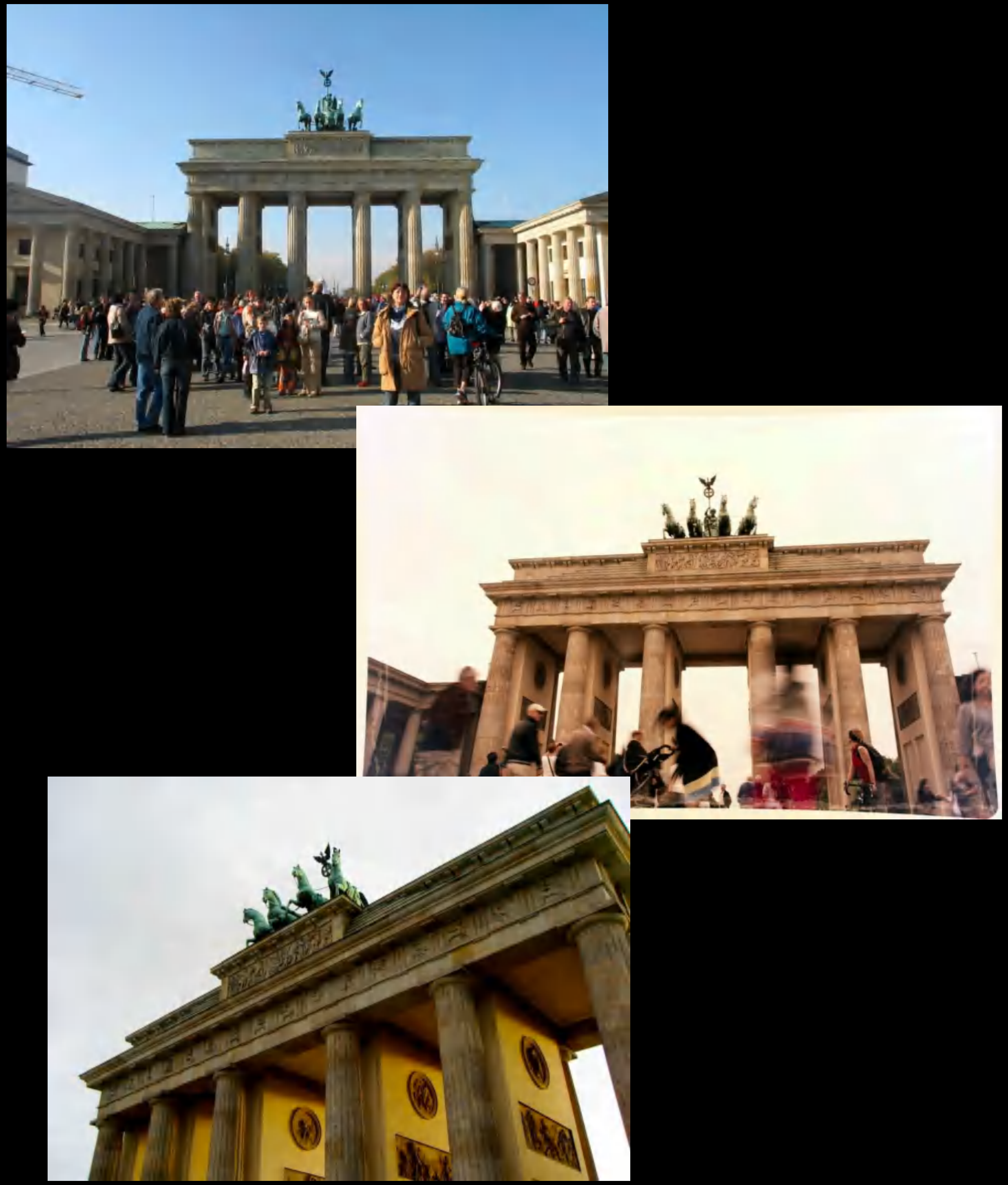
NeRF in the Wild: Neural Radiance Fields for Uncontrolled Photo Collections

CVPR 2021

Ricardo Martin-Brualla*, Noha Radwan*, Mehdi Sajjadi*,
Jonathan T. Barron, Alexey Dosovitskiy, Daniel Duckworth

Google Brain Berlin & Google Research

https://nerf-w.github.io/

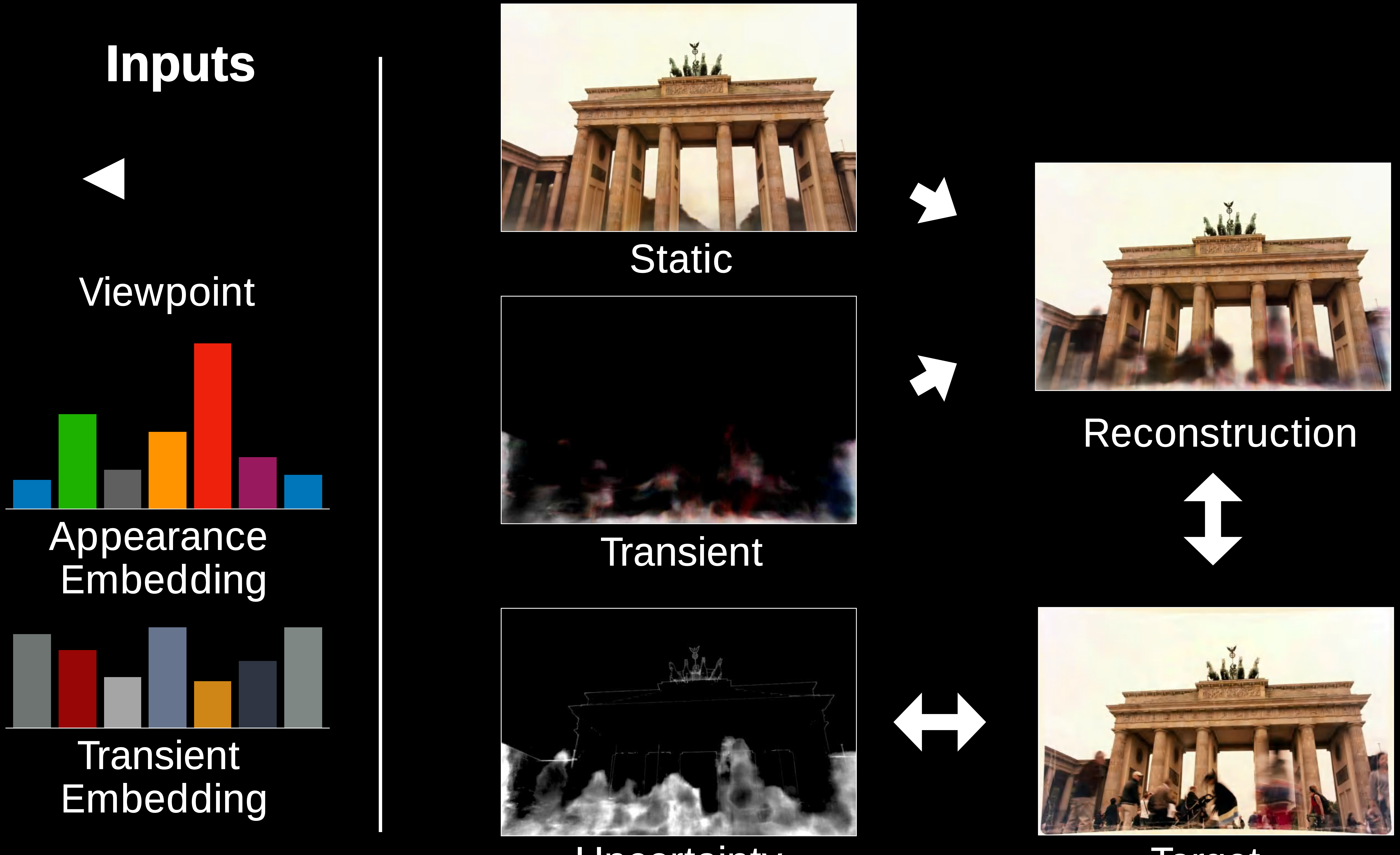


Unconstrained photo collection

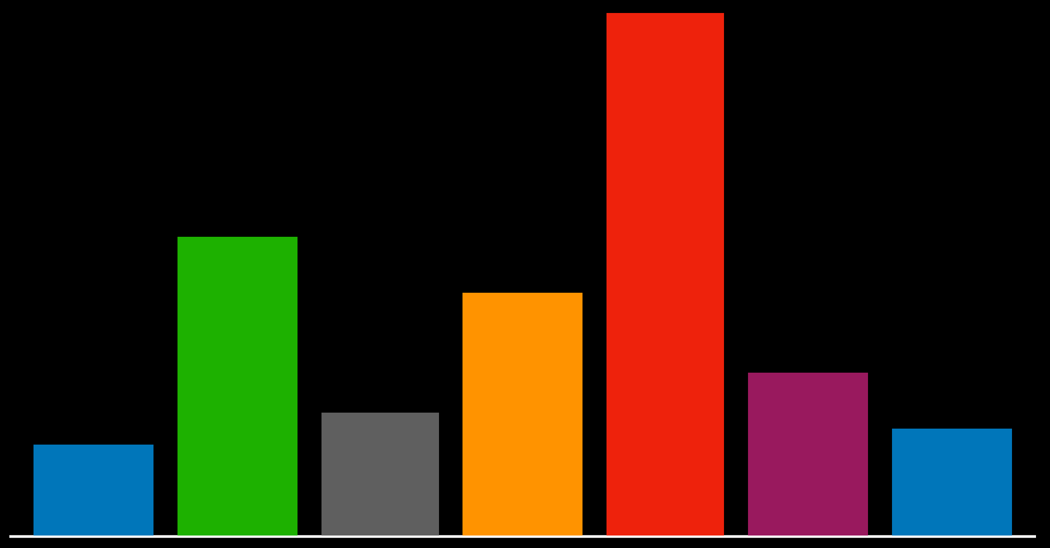
Novel views + Novel appearance




12 OKTOBER 1492
LATEINAMERIKA WURDE
NICHT ENTDECKT.
ES WURDE UND WIRD
BESETZT UND AUSGEPLÜNDERT


Inputs
Viewpoint
Appearance Embedding
Transient Embedding
Static
Transient
Uncertainty
Reconstruction
Target

Viewpoint



Appearance
Embedding

NeRF

Ours

“NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”,
Mildenhall, Srinivasan, Tancik et. al., ECCV 2020

Ours

Neural Rendering in the Wild

“Neural Rerendering In the Wild”, Meshry et. al., CVPR 2019

Thanks!

http://jonbarron.info

https://twitter.com/jon_barron