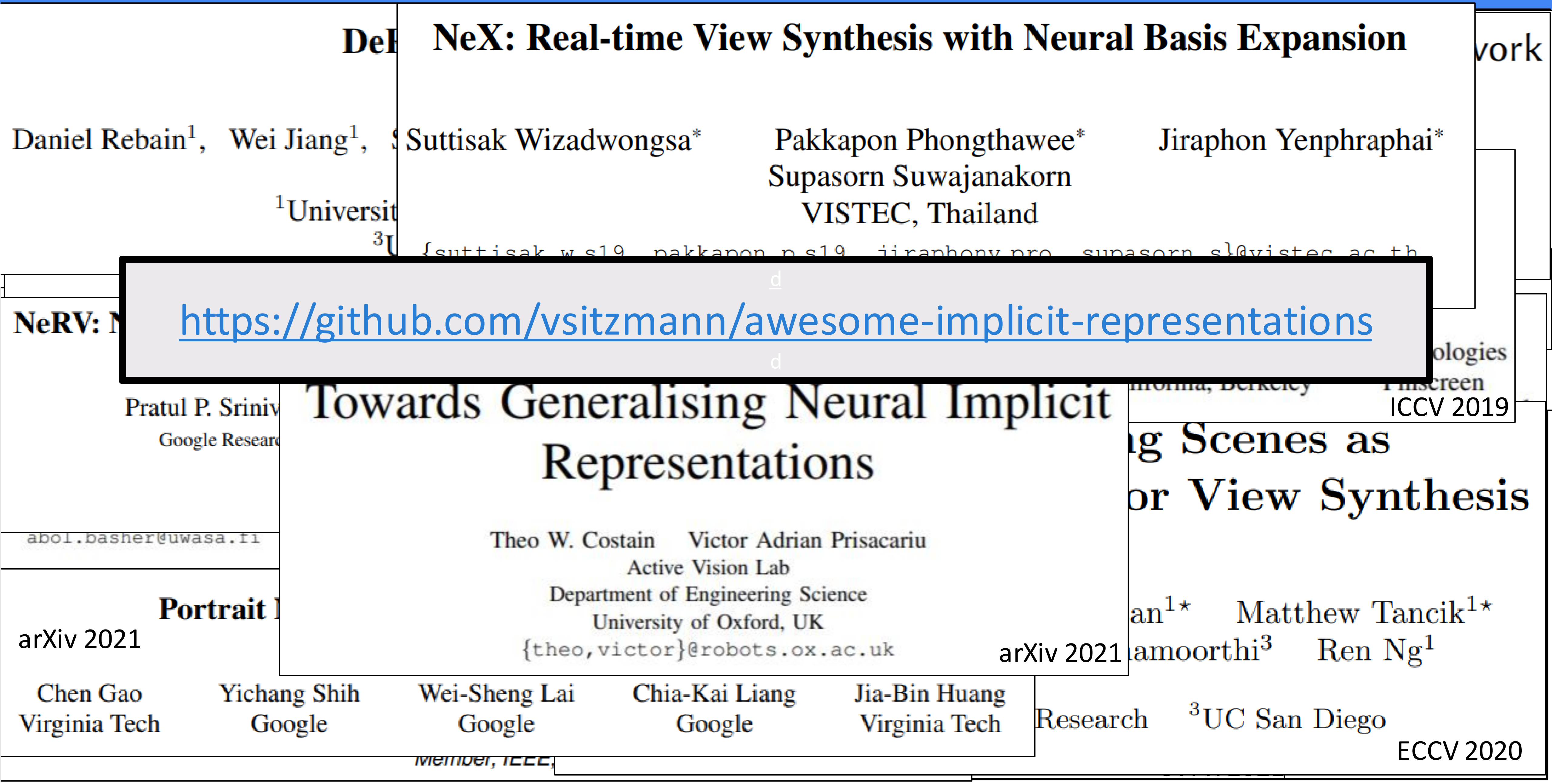
Neural Implicit Representations

Dolev Ofri and Eyal Naor

June 2021

A Rapidly Growing Research Field



Outline

Intro

NeRF

Fourier Feat.

SIREN

NeX

Explicit vs implicit

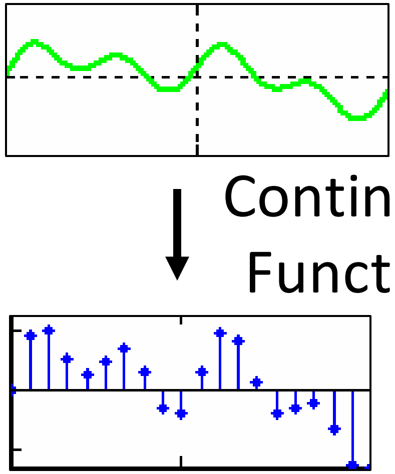
3D reconstruction examples

Explicit vs Implicit Representations

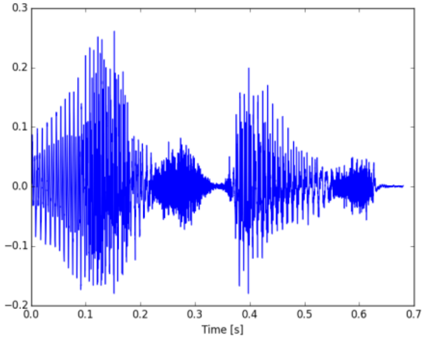
Explicit Representations

2D Representations

1D Representations

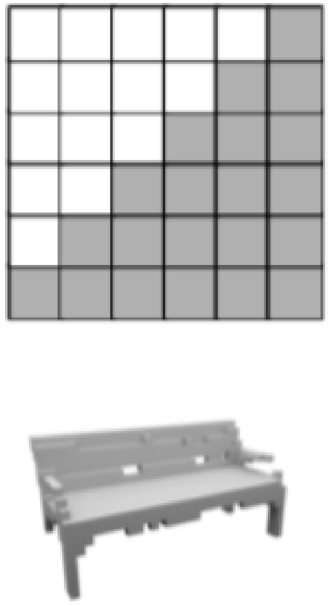


Continuous Functions

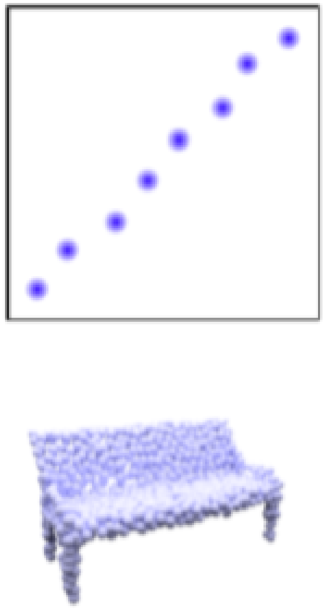


Audio Signals

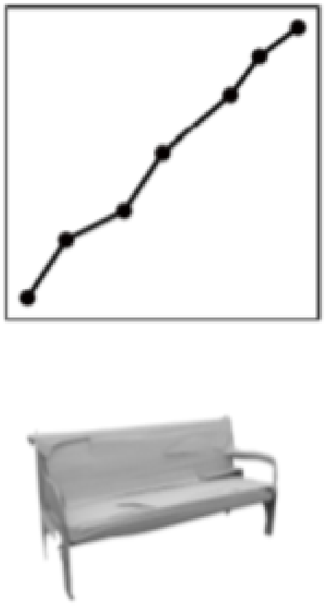
3D Representations



Voxels



Points



Mesh

Implicit Representations

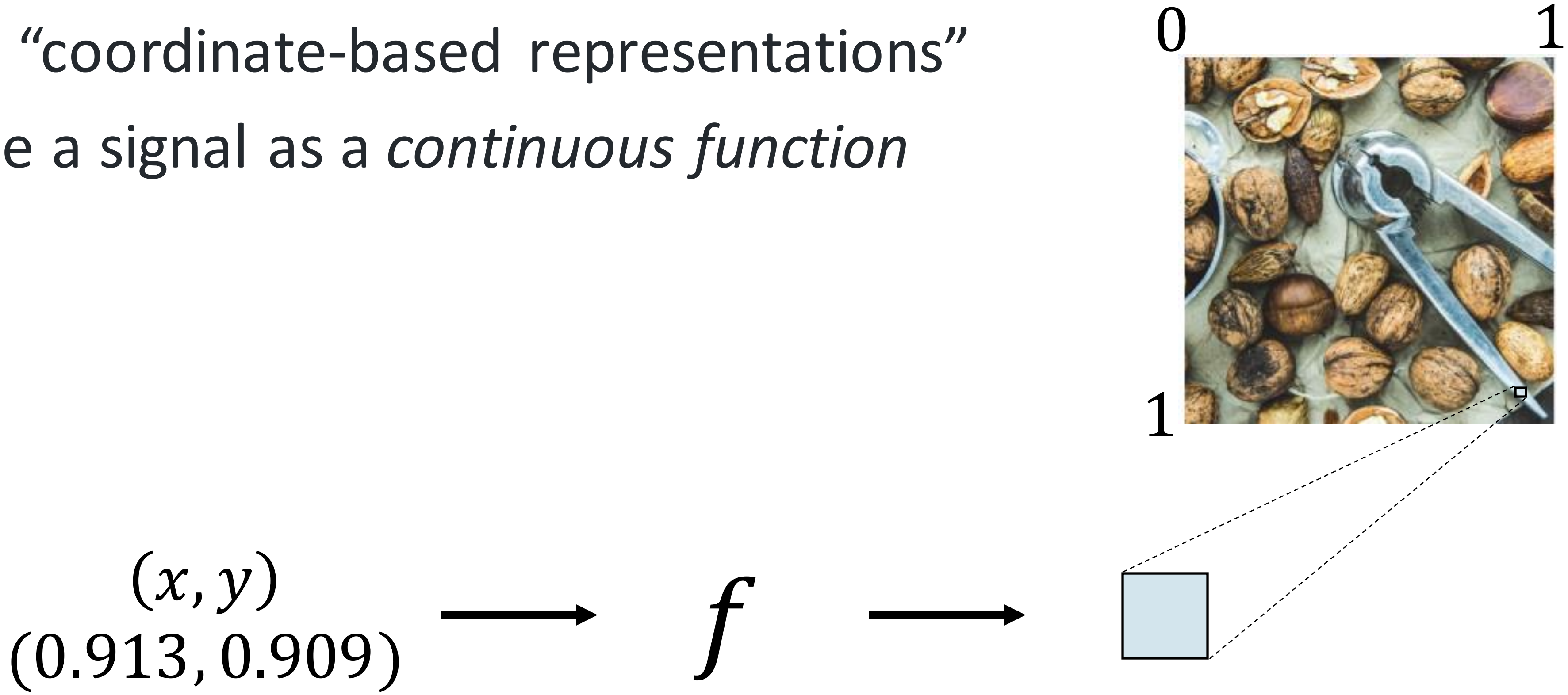
- Also called “coordinate-based representations”

Implicit Representations

- Also called “coordinate-based representations”
- Parametrize a signal as a *continuous function*

Implicit Representations

- Also called “coordinate-based representations”
- Parametrize a signal as a *continuous function*



$$\begin{matrix}(x, y) \\ (0.913, 0.909)\end{matrix} \longrightarrow f \longrightarrow$$

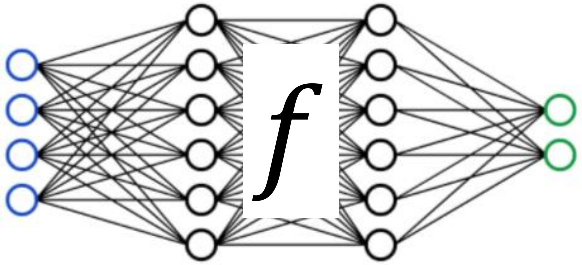
Implicit Representations

- Also called “coordinate-based representations”
- Parametrize a signal as a *continuous function*
- Exact mathematical function is unknown

$$f = ?$$

Implicit Representations

- Also called “coordinate-based representations”
- Parametrize a signal as a *continuous function*
- **Neural** Implicit Representations: use a neural network!

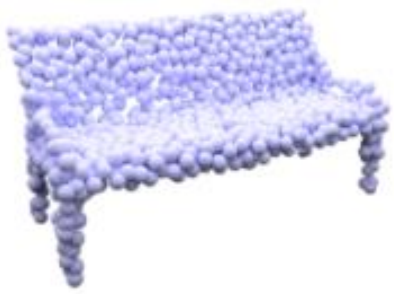


Implicit Representations

Main advantages:

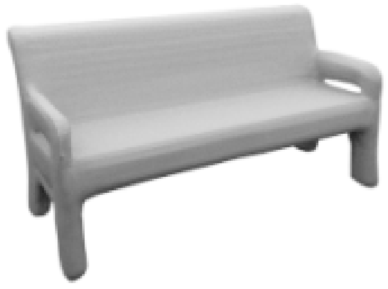
- Arbitrary resolution
- Memory efficient

Voxels



Points

Mesh



Continuous Function

Implicit Representations

Main advantages:

- Arbitrary resolution
- Memory efficient

Uses:

- Super resolution
- Geometry representation / 3D reconstruction
- …

Implicit Representations

Main advantages:

- Arbitrary resolution
- Memory efficient

Uses:

- Super resolution
- Geometry representation / 3D reconstruction
- …

Occupancy Networks

Learning 3D Reconstruction in Function Space

Lars Mescheder, Michael Oechsle,
Michael Niemeyer, Sebastian Nowozin,
Andreas Geiger

CVPR 2019

DeepSDF

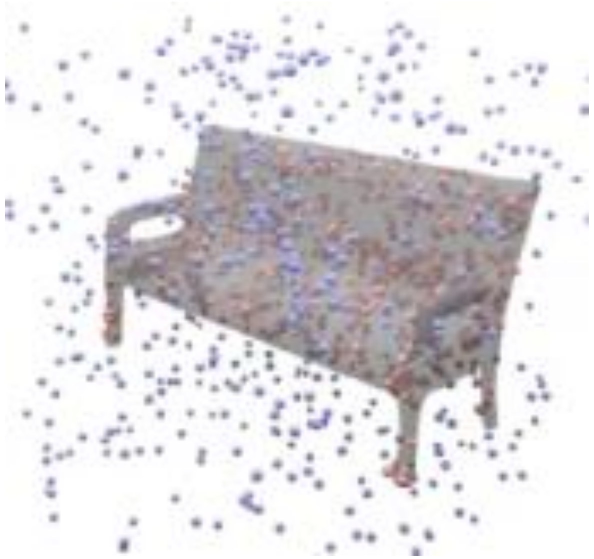
Learning Continuous Signed Distance Functions for Shape Representation

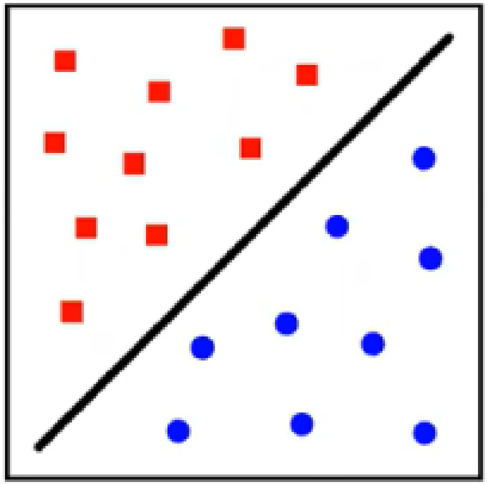
Jeong Joon Park, Peter Florence,
Julian Straub, Richard Newcombe,
Steven Lovegrove

CVPR 2019

Occupancy Networks

- **Decision boundary**





DeepSDF

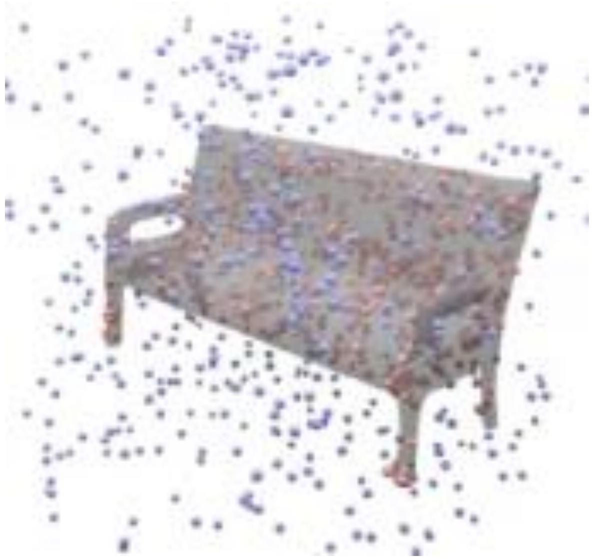
Learning Continuous Signed Distance Functions for Shape Representation

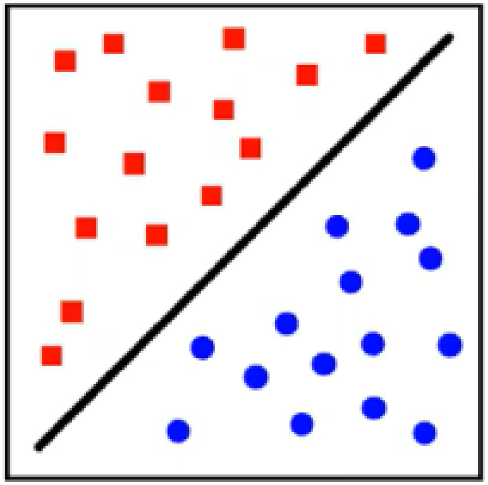
Jeong Joon Park, Peter Florence,
Julian Straub, Richard Newcombe,
Steven Lovegrove

CVPR 2019

Occupancy Networks

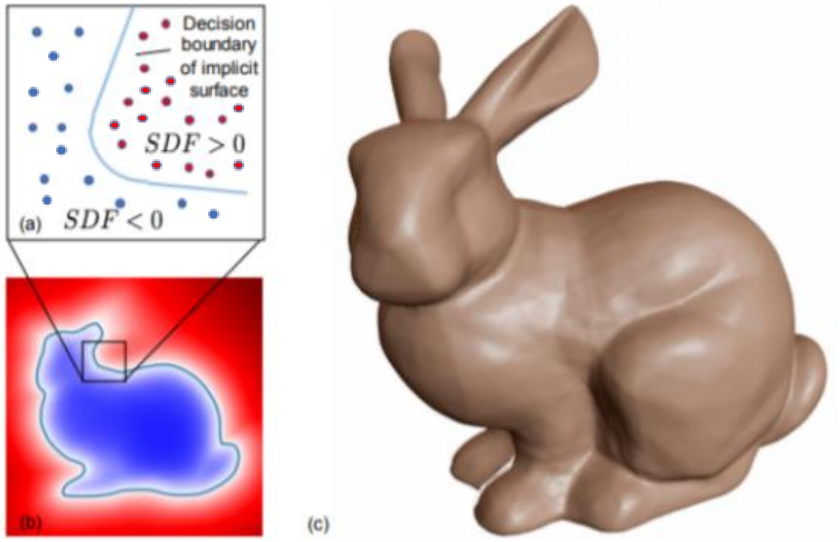
- **Decision boundary**





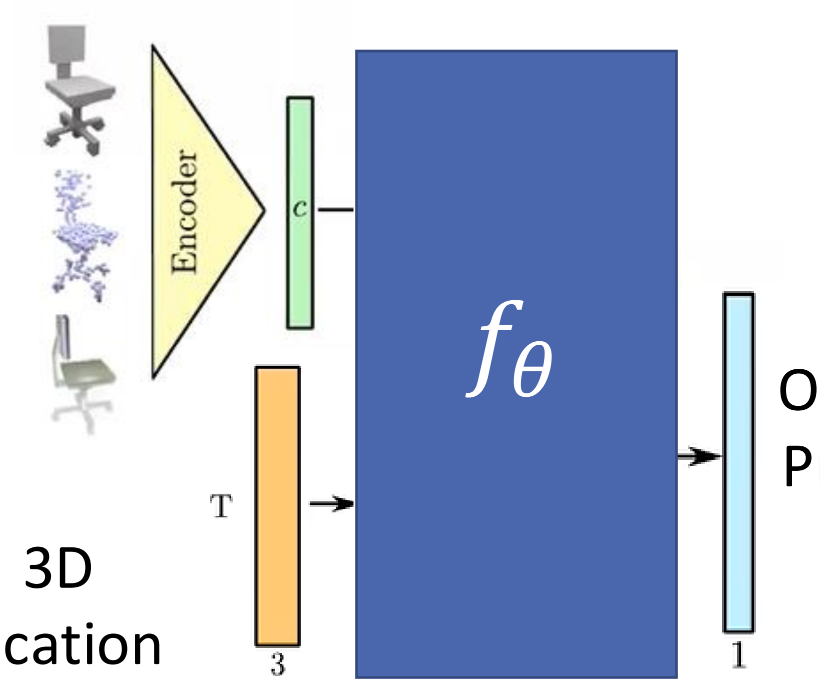
DeepSDF

- **Signed Distance Function (SDF)**



Occupancy Networks

- **Decision boundary**

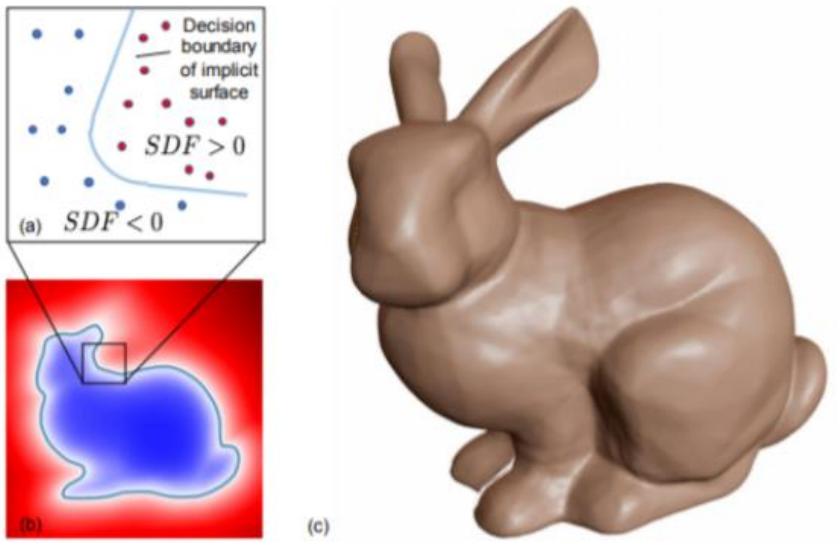


3D location

Occupancy Probability [0,1]

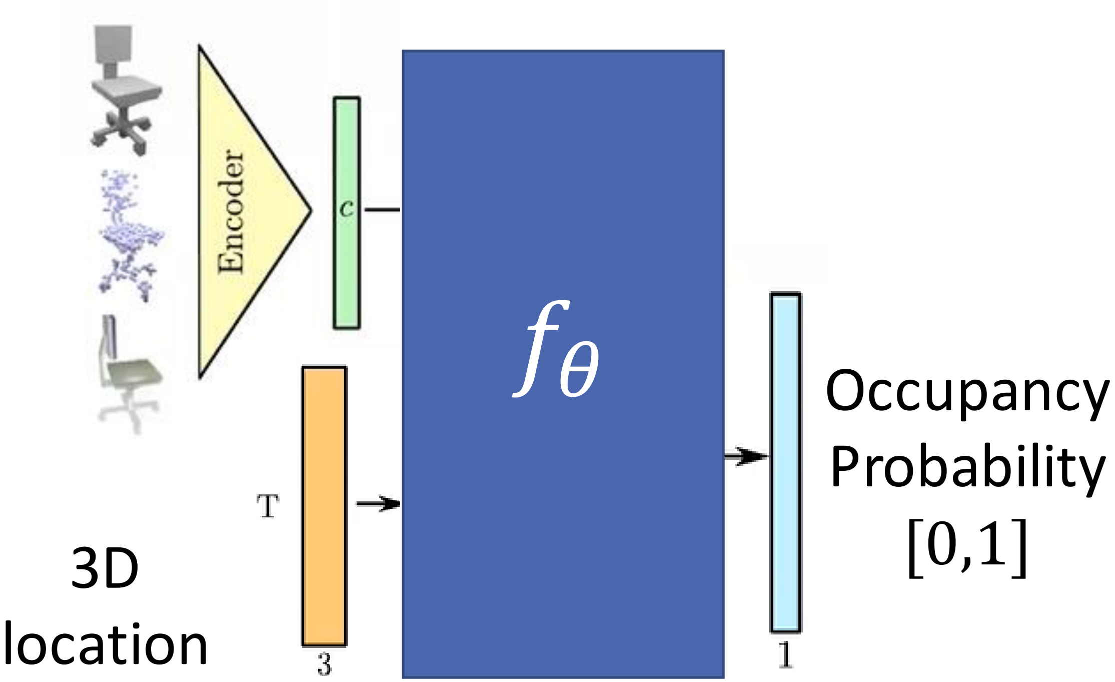
DeepSDF

- **Signed Distance Function (SDF)**



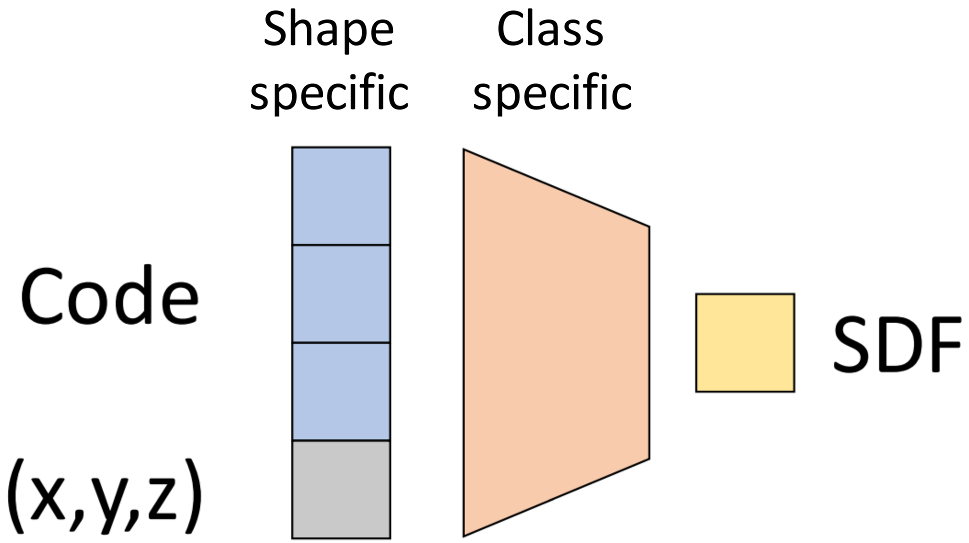
Occupancy Networks

- **Decision boundary**



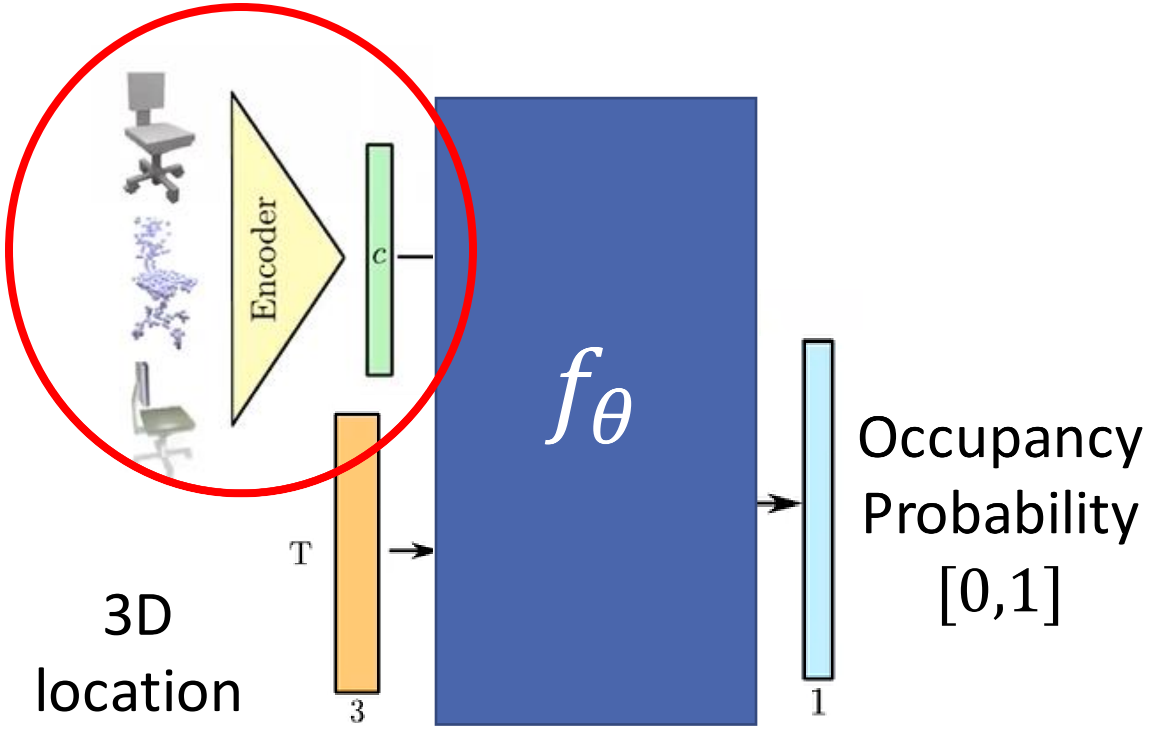
DeepSDF

- **Signed Distance Function (SDF)**



Occupancy Networks

- **Decision boundary**

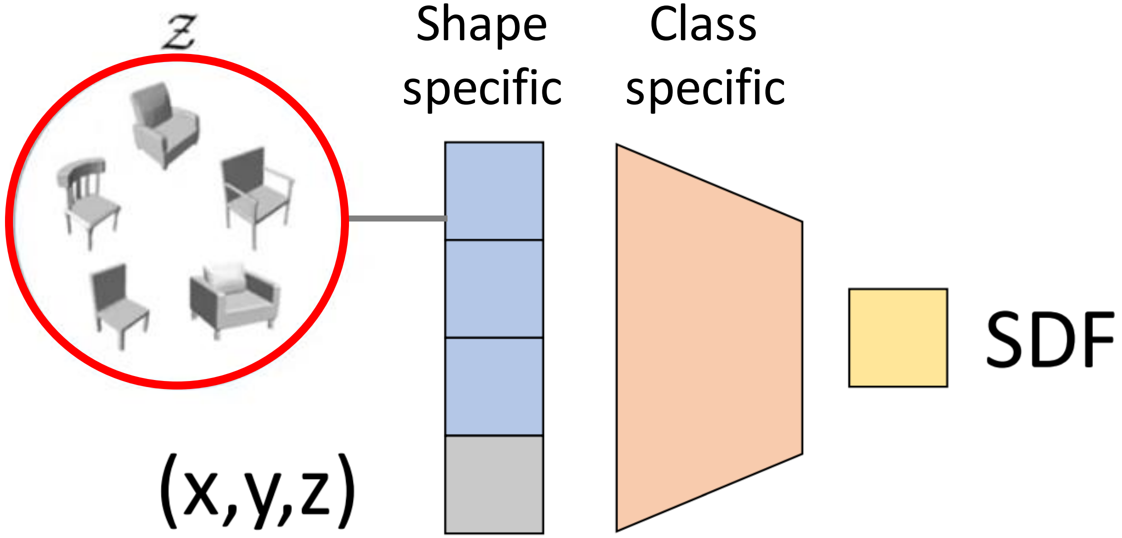


3D location

Occupancy Probability [0,1]

DeepSDF

- **Signed Distance Function (SDF)**



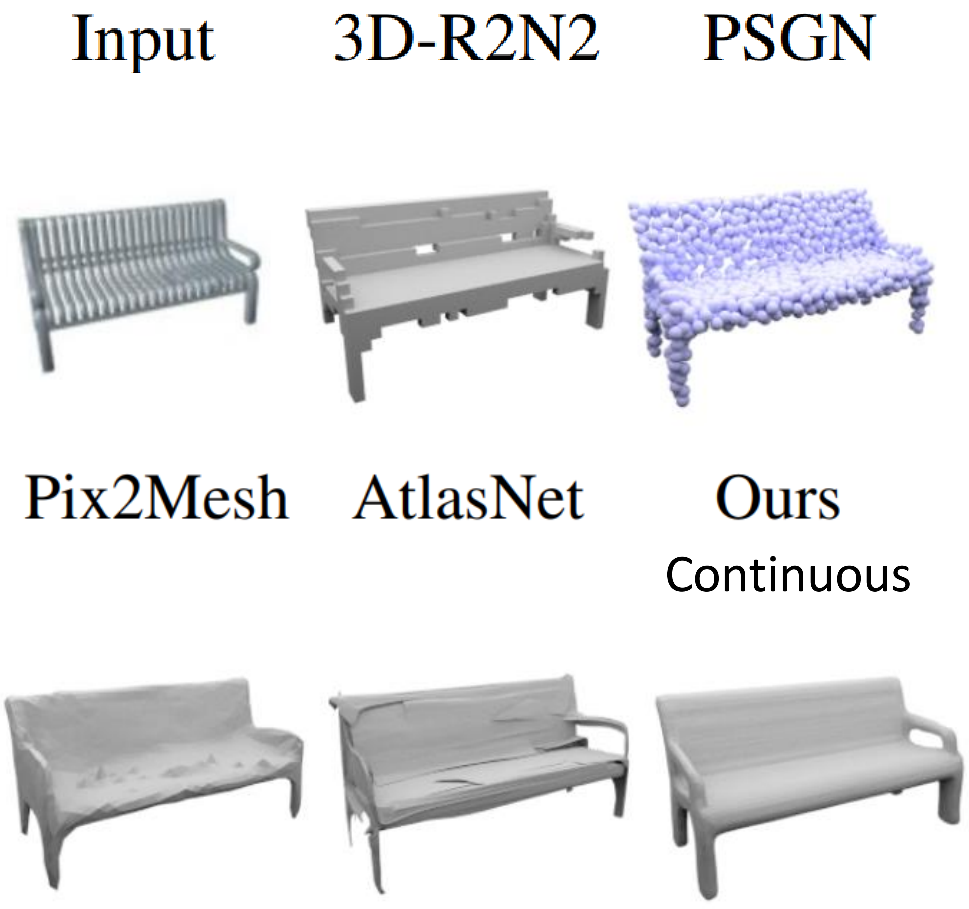
Shape specific

Class specific

SDF

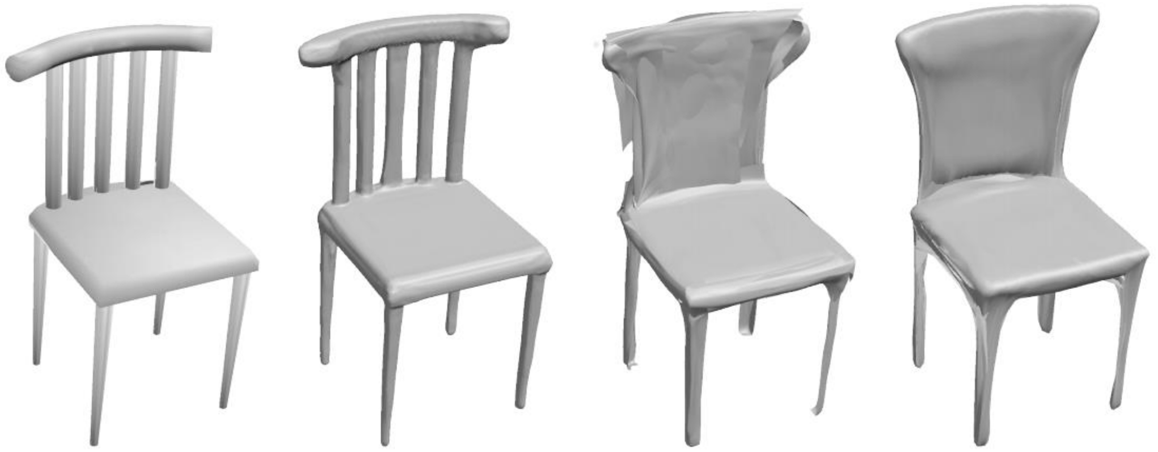
(x,y,z)

Occupancy Networks



Continuous

DeepSDF



(a) Ground-truth **(b)** Our Result **(c)** [22]-25 patch **(d)** [22]-sphere

Continuous

AtlasNet

Scene Representation

Scene Representation

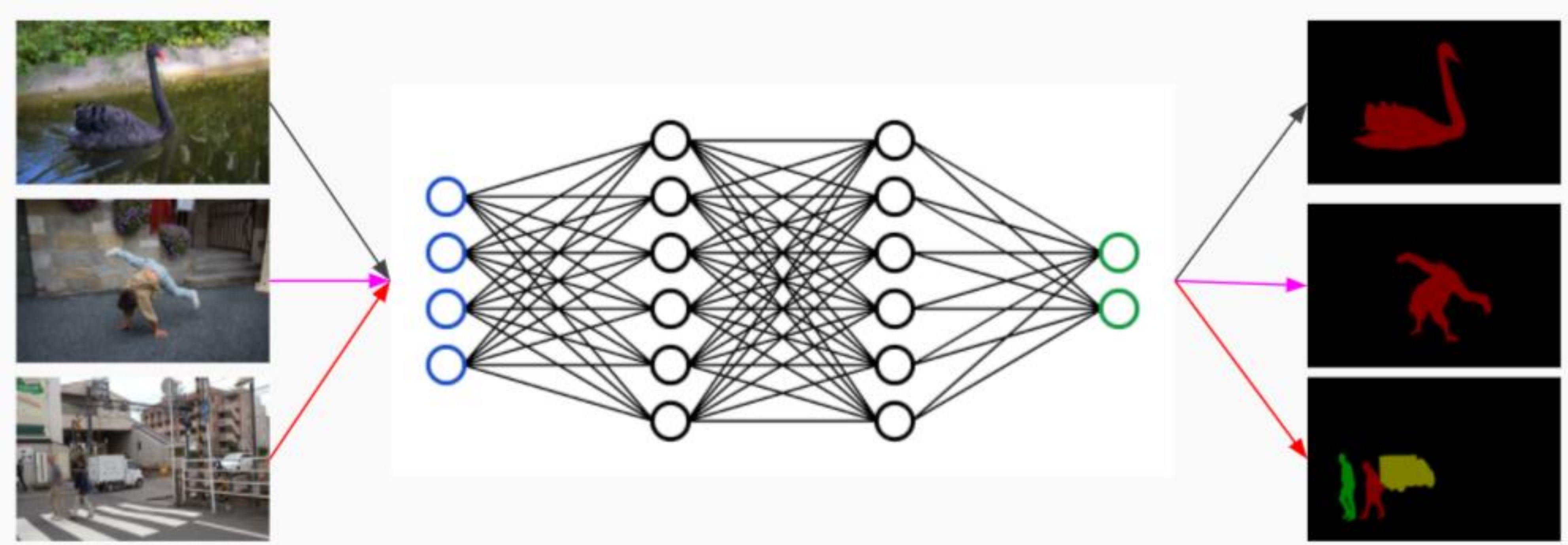
“Classic DL”: The Net == The Task

Single net, Single task

Scene Representation

“Classic DL”: The Net == The Task

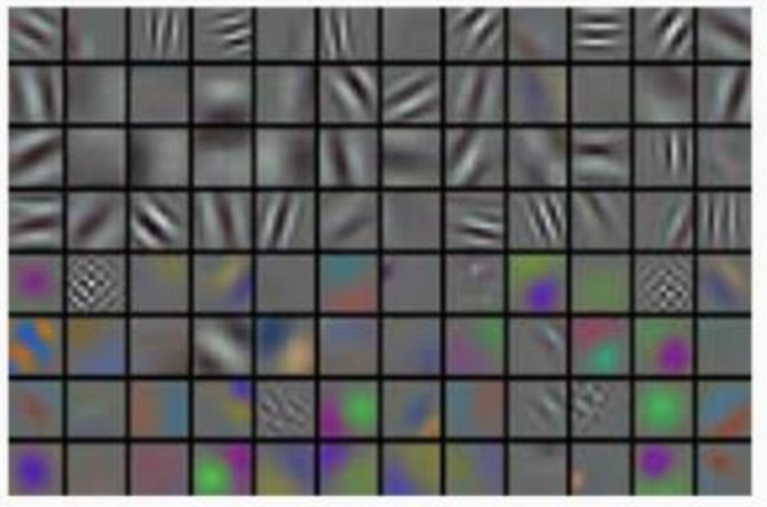
Single net, Single task



Scene Representation

“Classic DL”: The Net == The Task

Single net, Single task

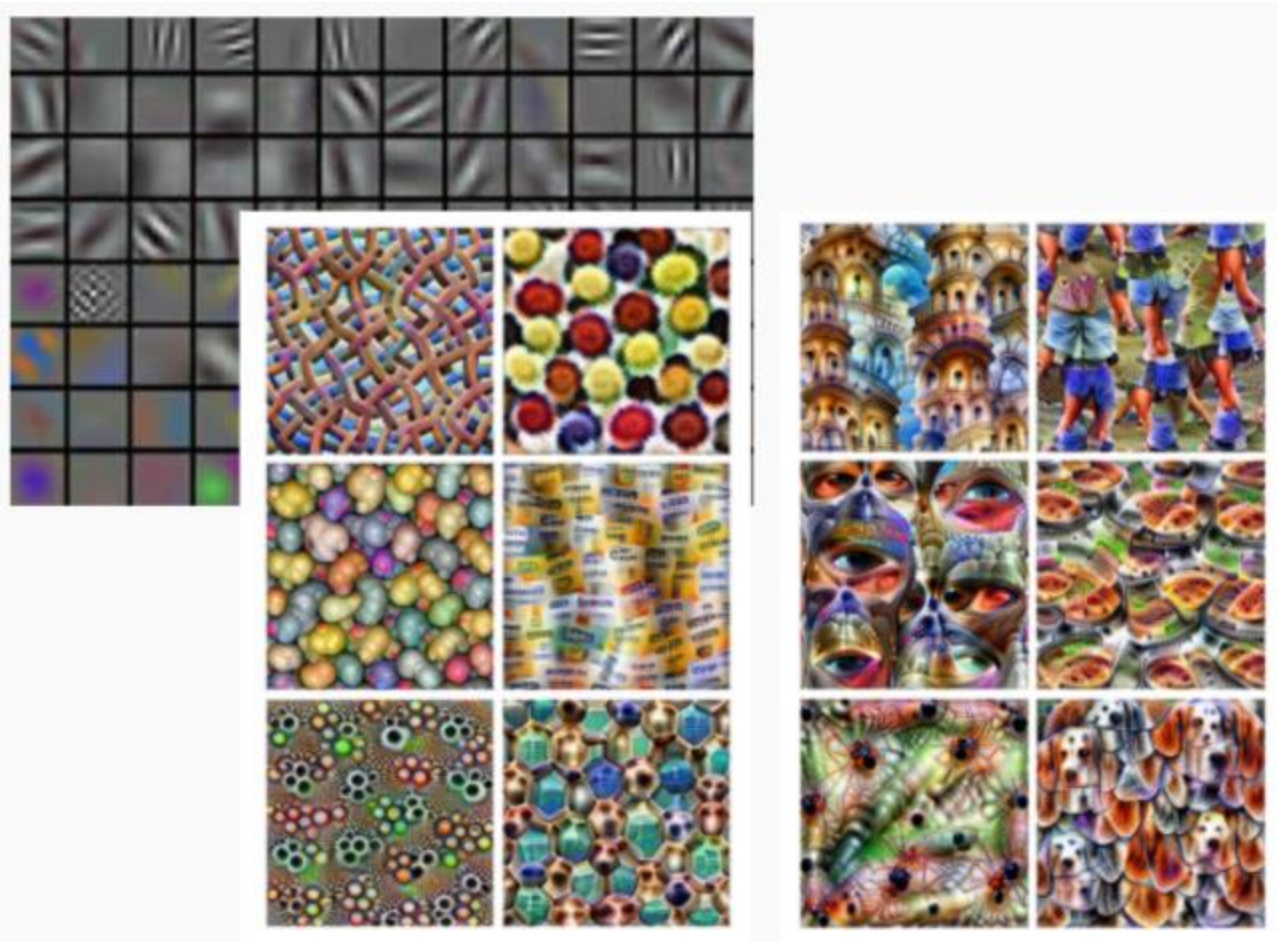


The network weights “hold” what’s needed for the task.

Scene Representation

“Classic DL”: The Net == The Task

Single net, Single task

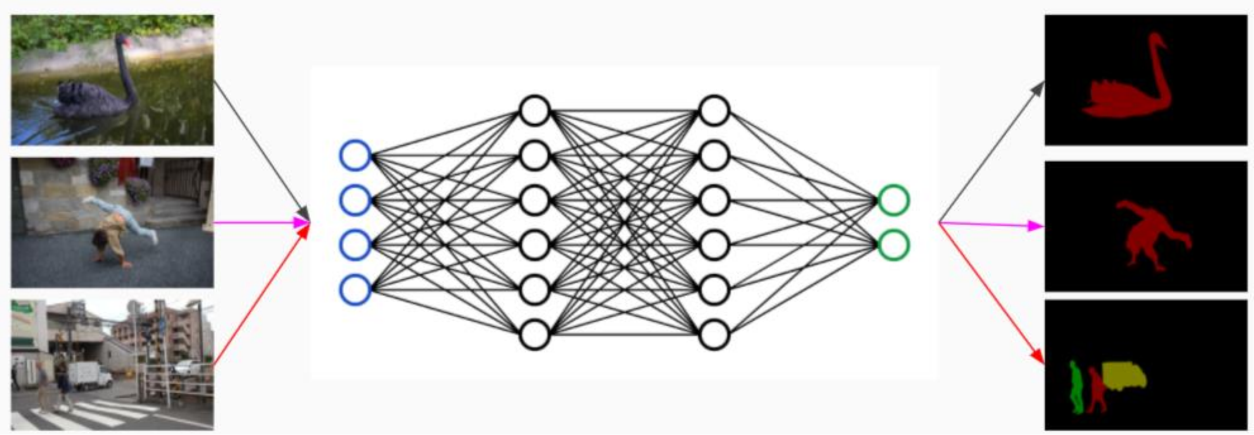


The network weights “hold” what’s needed for the task.

Scene Representation

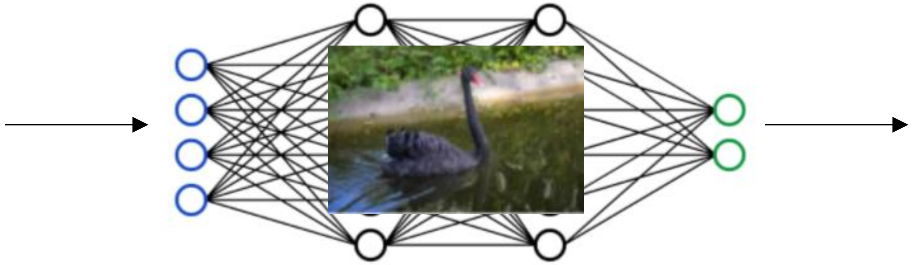
“Classic DL”: The Net == The Task

Single net, Single task



NeRF: **The Net == The Scene**

Single net, Single scene



NeRF
Representing Scenes as Neural Radiance Fields for View Synthesis

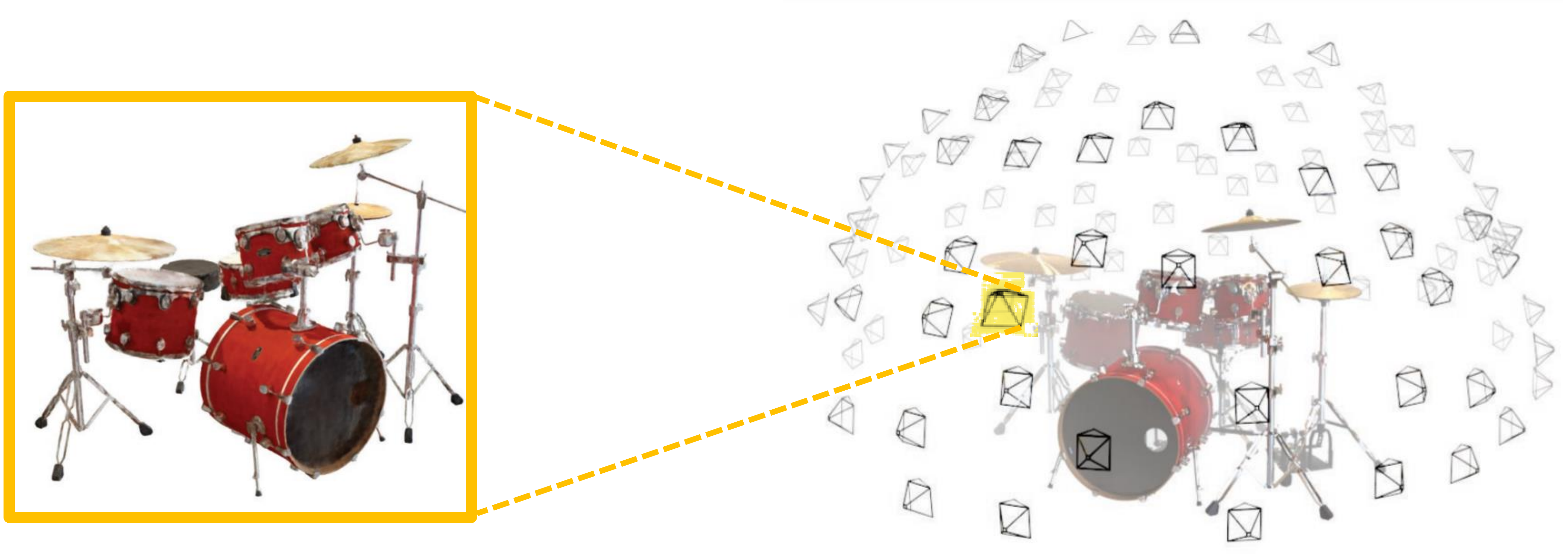
Ben Mildenhall, Pratul Srinivasan, Matt Tancik, Jon Barron, Ravi Ramamoorth, Ren Ng

ECCV 2020, Best Paper Honorable Mention

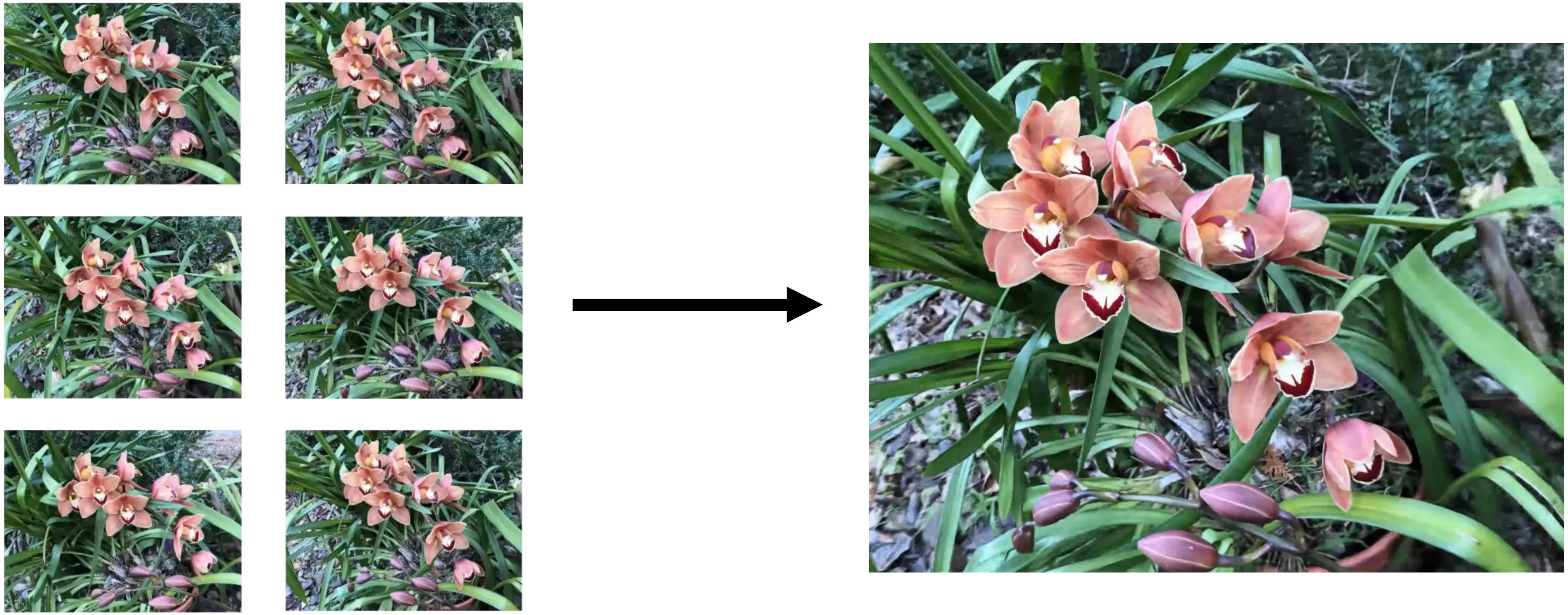
Slides on NeRF are based on Yoni Kansten’s slides

Task: Render New Views

Task: Render New Views



Task: Render New Views



Inputs: sparsely sampled images of scene

Output: includes new rendered views

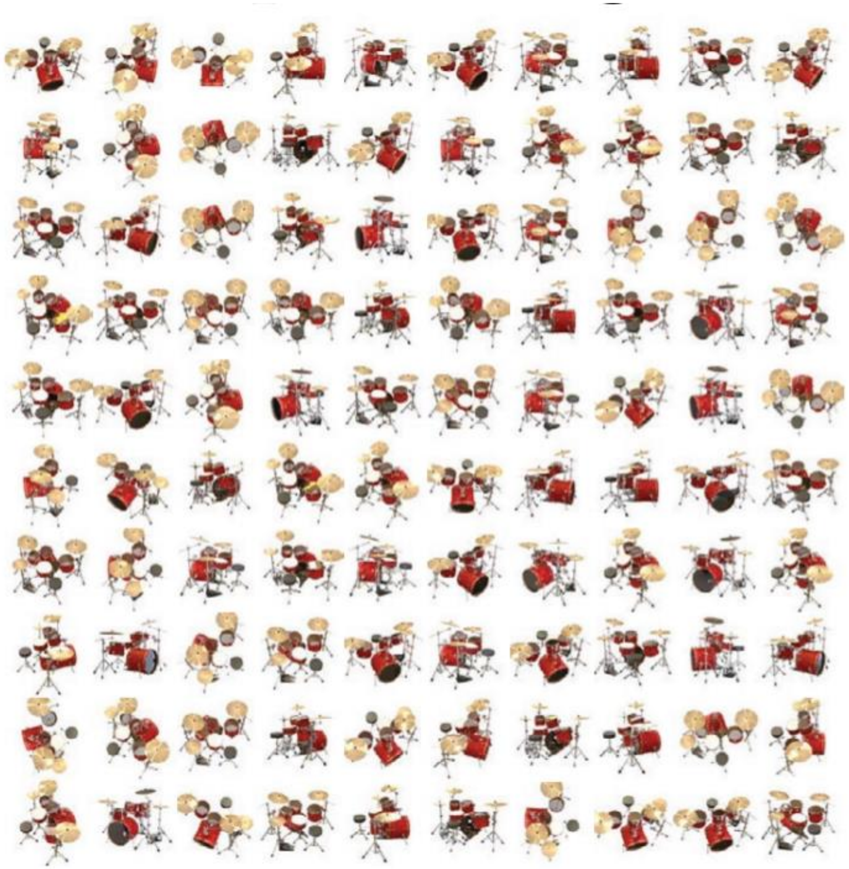
matthewtancik.com/nerf

Inputs

Multiview Images of a single scene

Inputs

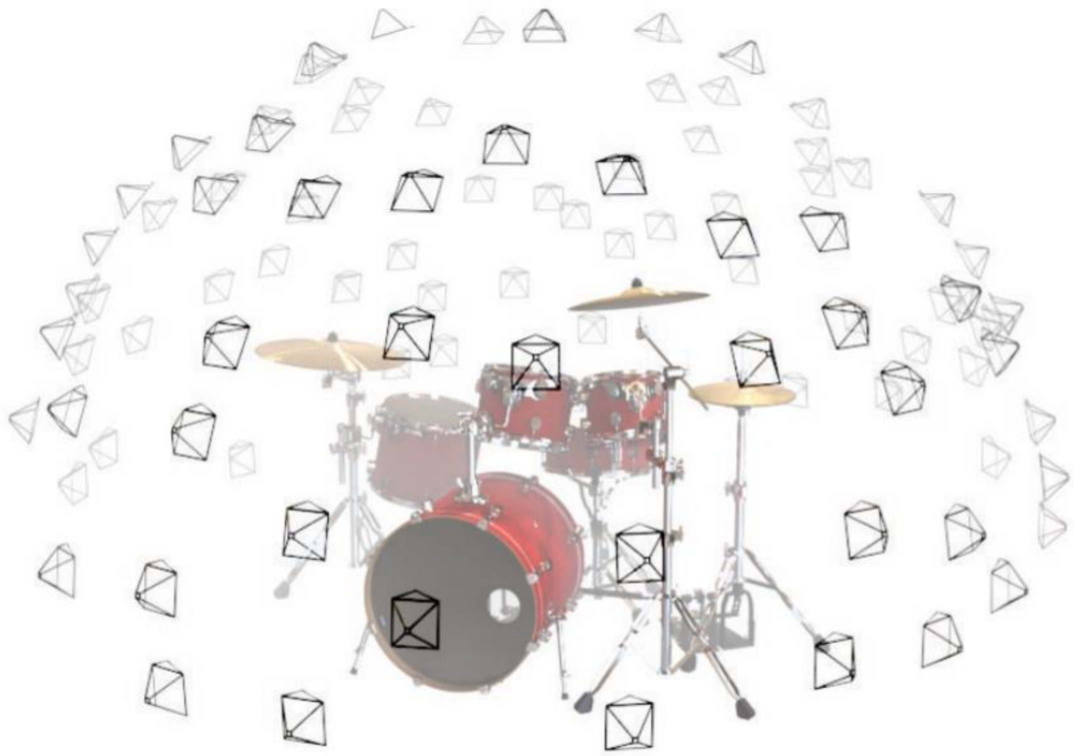
Multiview Images of a single scene



Inputs

Multiview Images of a single scene

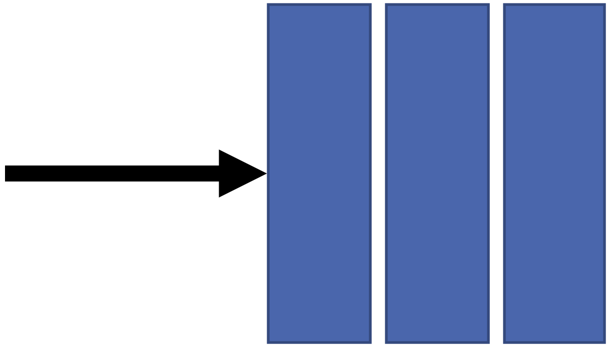
Camera poses



Scene representation

Scene representation

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

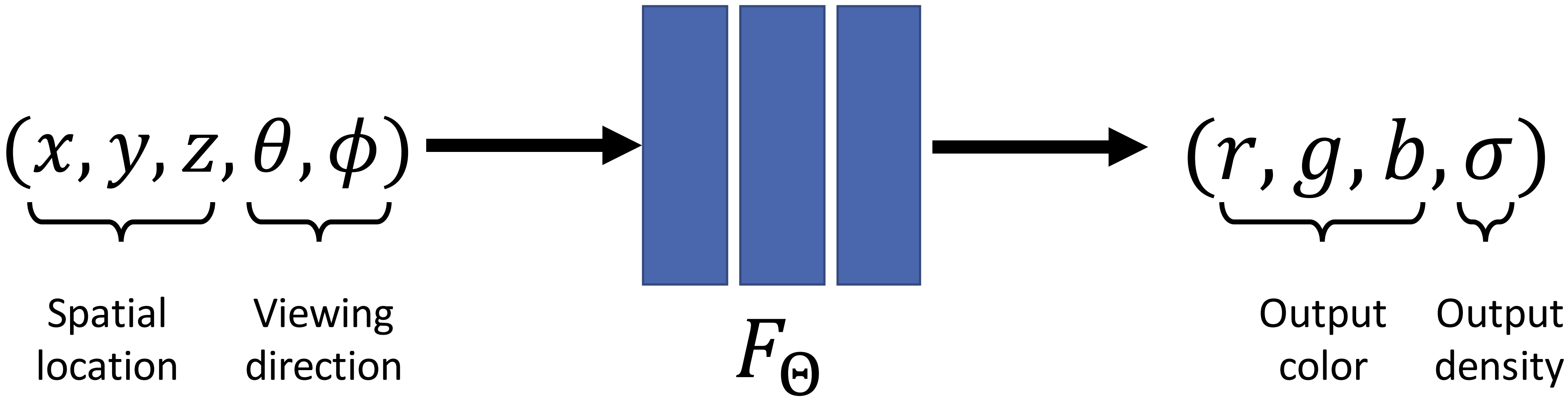


F_{Θ}

Multi-Layered Perceptron (MLP)
9 layers
256 channels

Slide credit: Jon Barron’s talk

Scene representation



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

Multi-Layered Perceptron (MLP)
9 layers
256 channels

Slide credit: Jon Barron’s talk

Scene representation

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

F_Θ

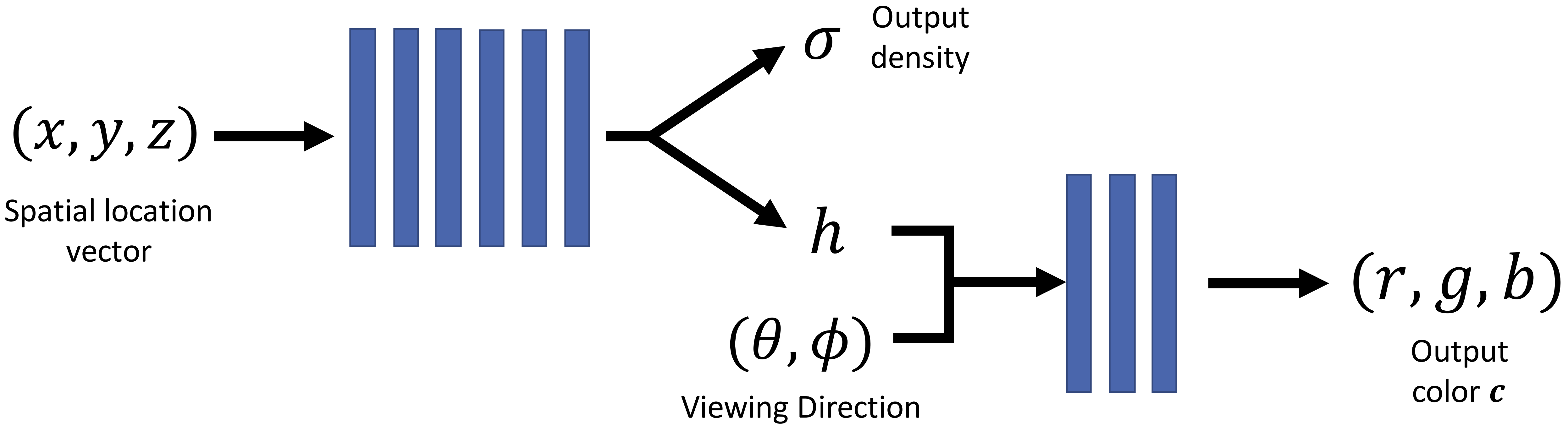
Multi-Layered Perceptron (MLP)
9 layers
256 channels

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

Input is only coordinates

No latent code

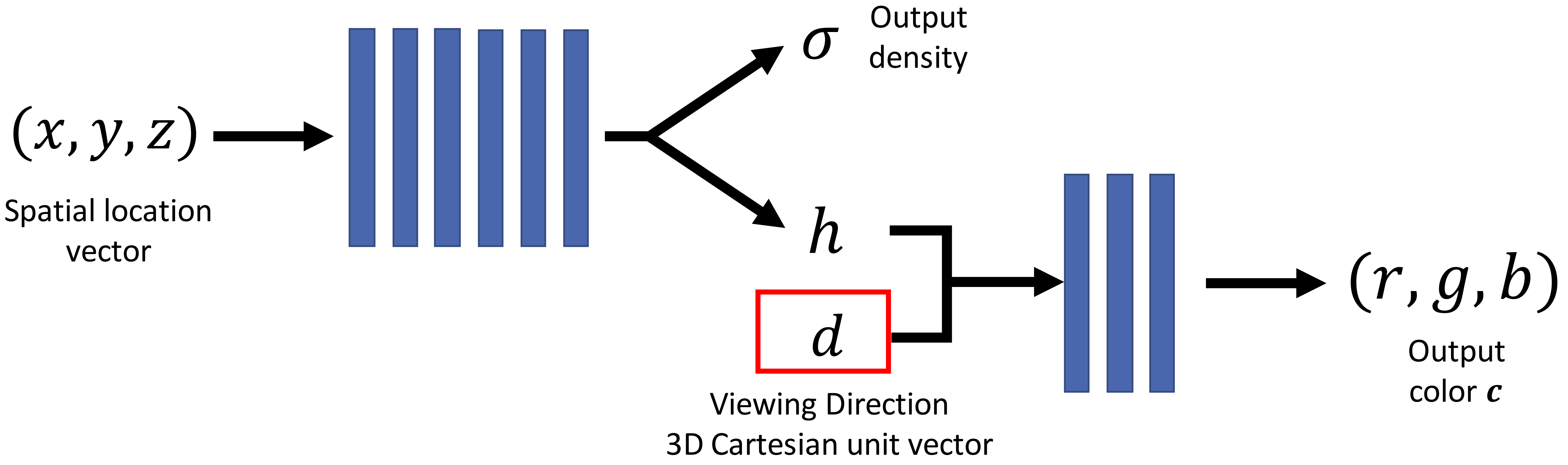
Scene representation



σ (spatial location)

$\boldsymbol{c}$ (spatial location, viewing direction)

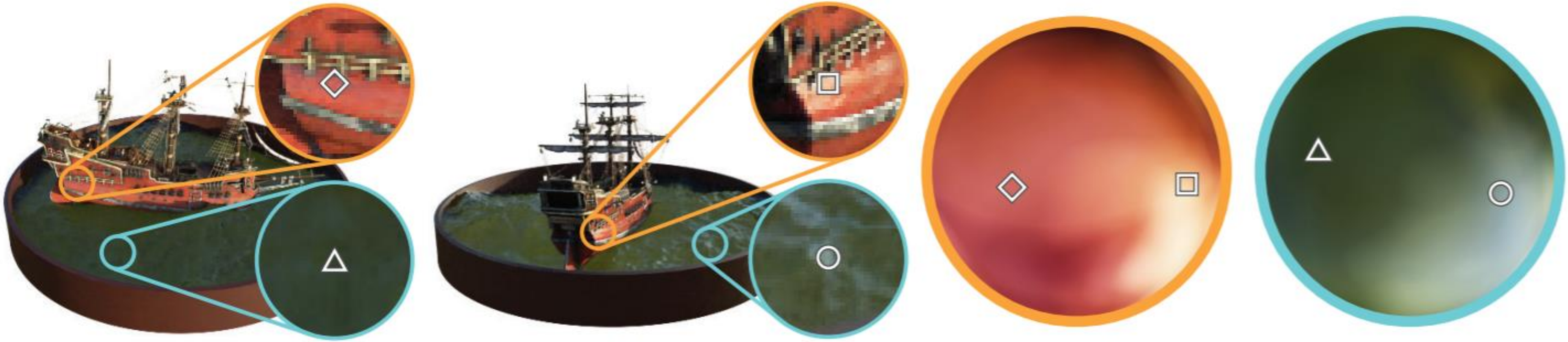
Scene representation



σ (spatial location)

$\boldsymbol{c}$ (spatial location, viewing direction)

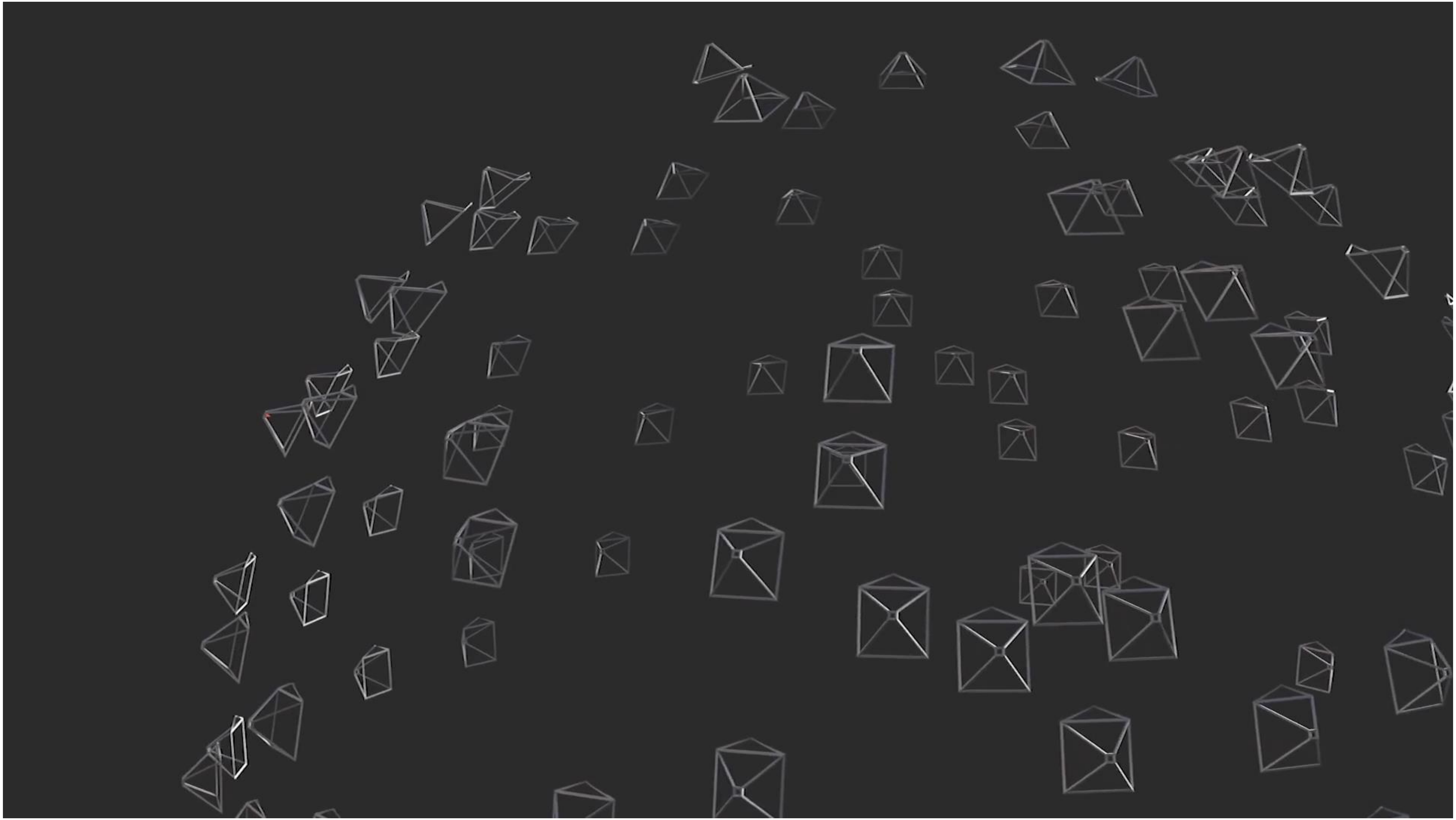
Viewing Directions as Input



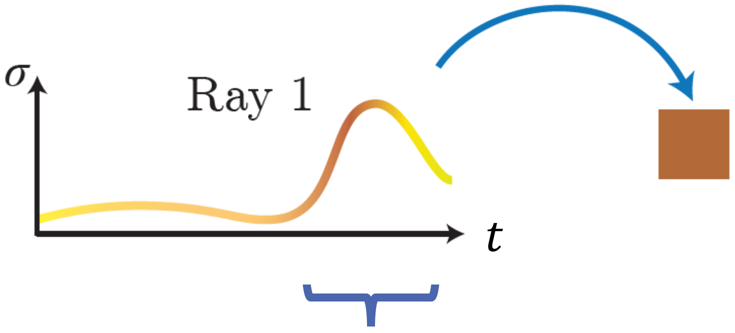
(a) View 1

(b) View 2

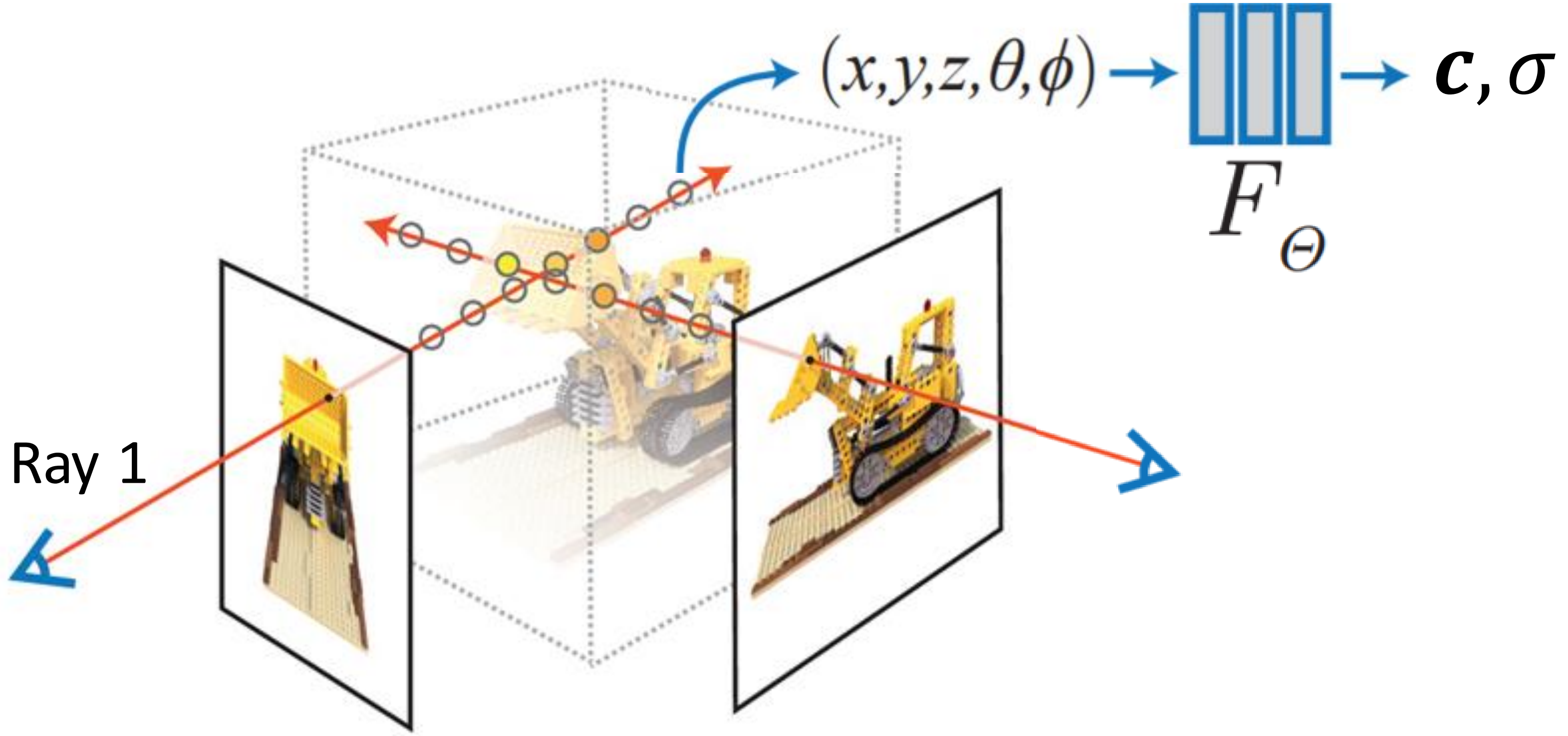
(c) Radiance Distributions



Volume rendering



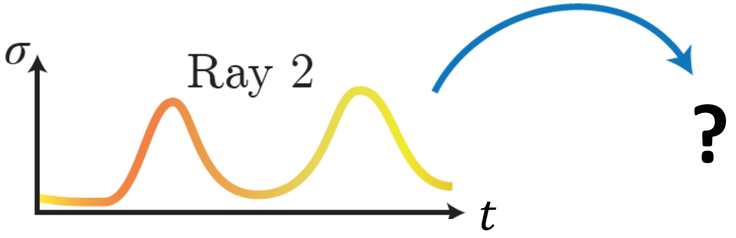
The ray hit something

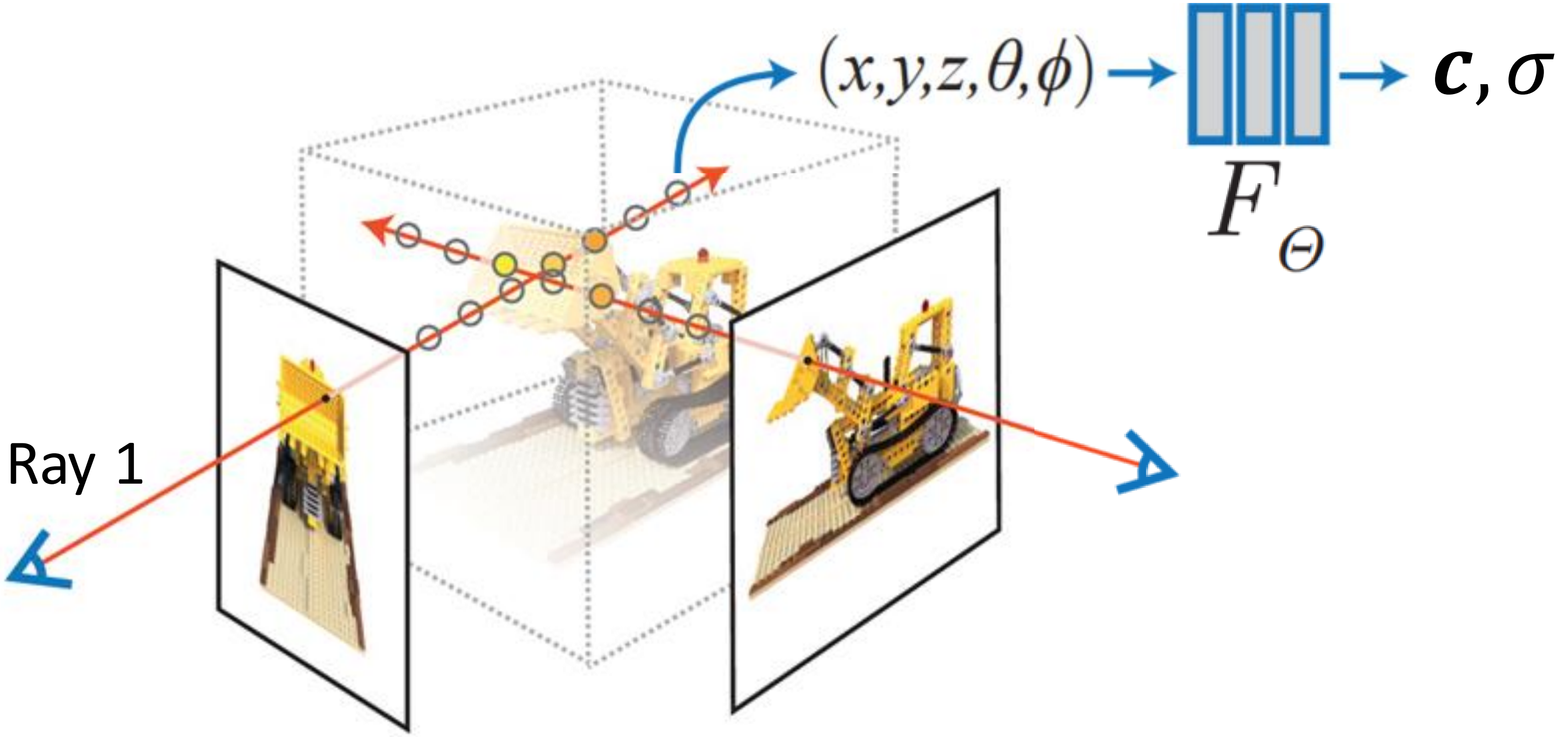


$\boldsymbol{r}(t)$ – camera ray $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$

σ – volume density

Volume rendering

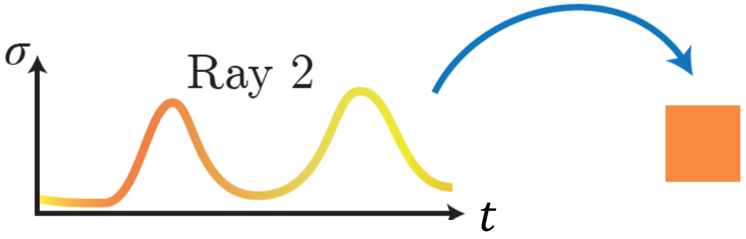


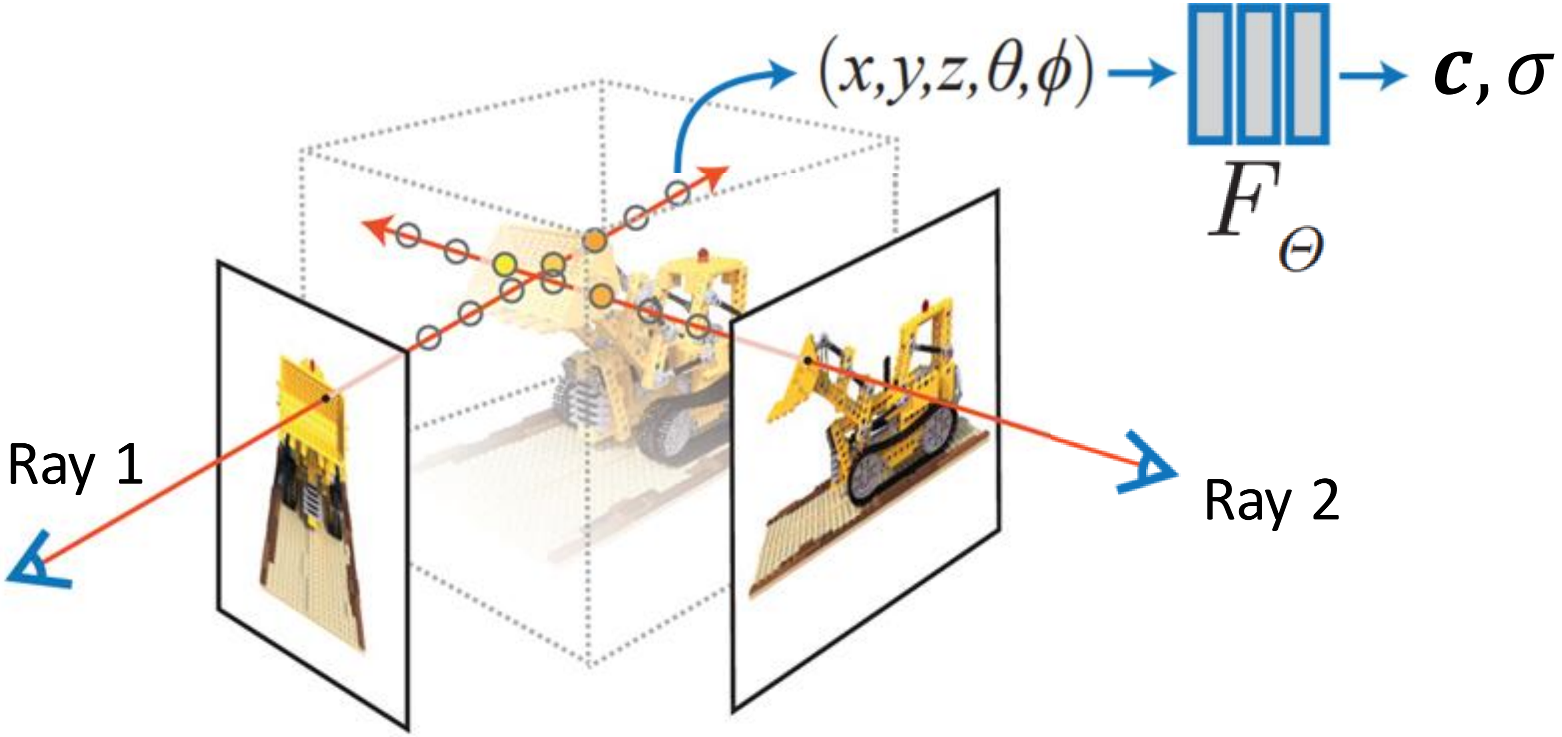


$\boldsymbol{r}(t)$ – camera ray $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$

σ – volume density

Volume rendering



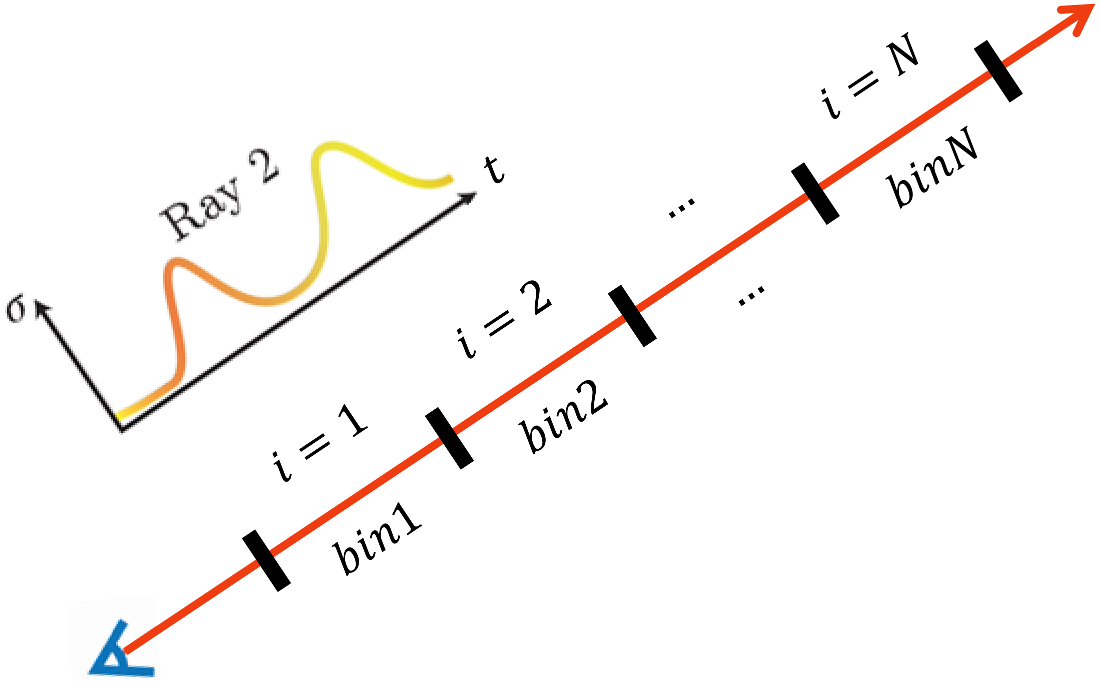


$\boldsymbol{r}(t)$ – camera ray $\boldsymbol{r}(t) = \boldsymbol{o} + t\boldsymbol{d}$

σ – volume density

Volume rendering

$$C(r) = \sum_{i=1}^{N} T_i \alpha_i c_i$$



Camera
Ray

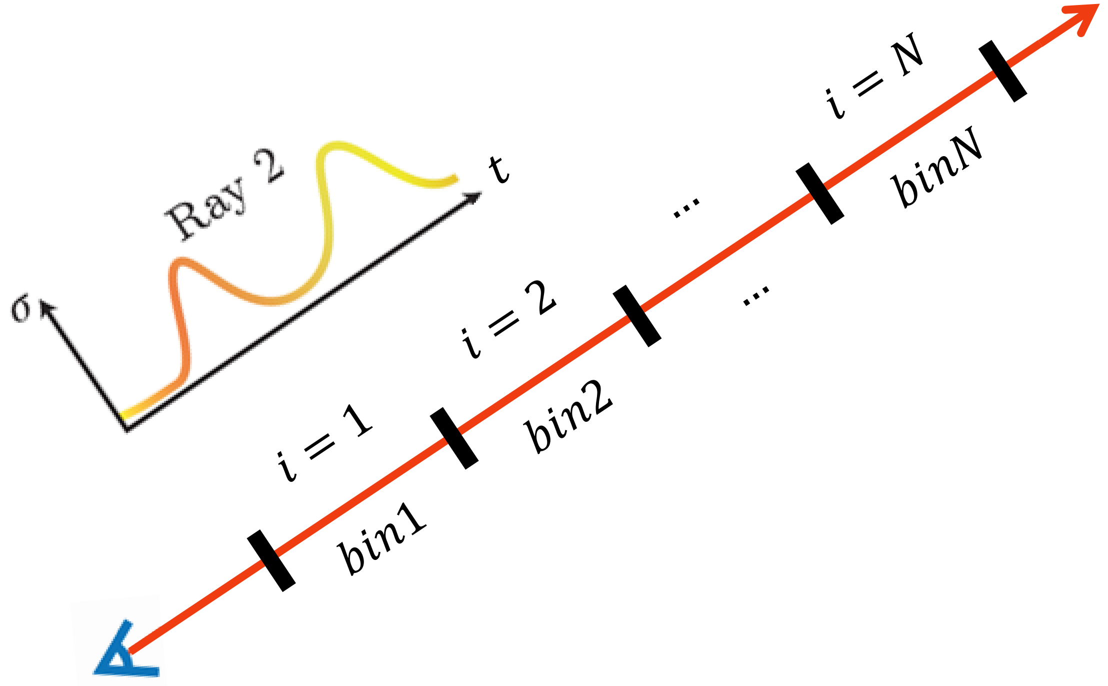
σ – volume density

Volume rendering

$$C(r) = \sum_{i=1}^{N} T_i \alpha_i c_i$$

Are you present?

$$\alpha_i = 1 - \mathrm{e}^{-\sigma_i \delta_i}$$



Camera Ray

σ – volume density

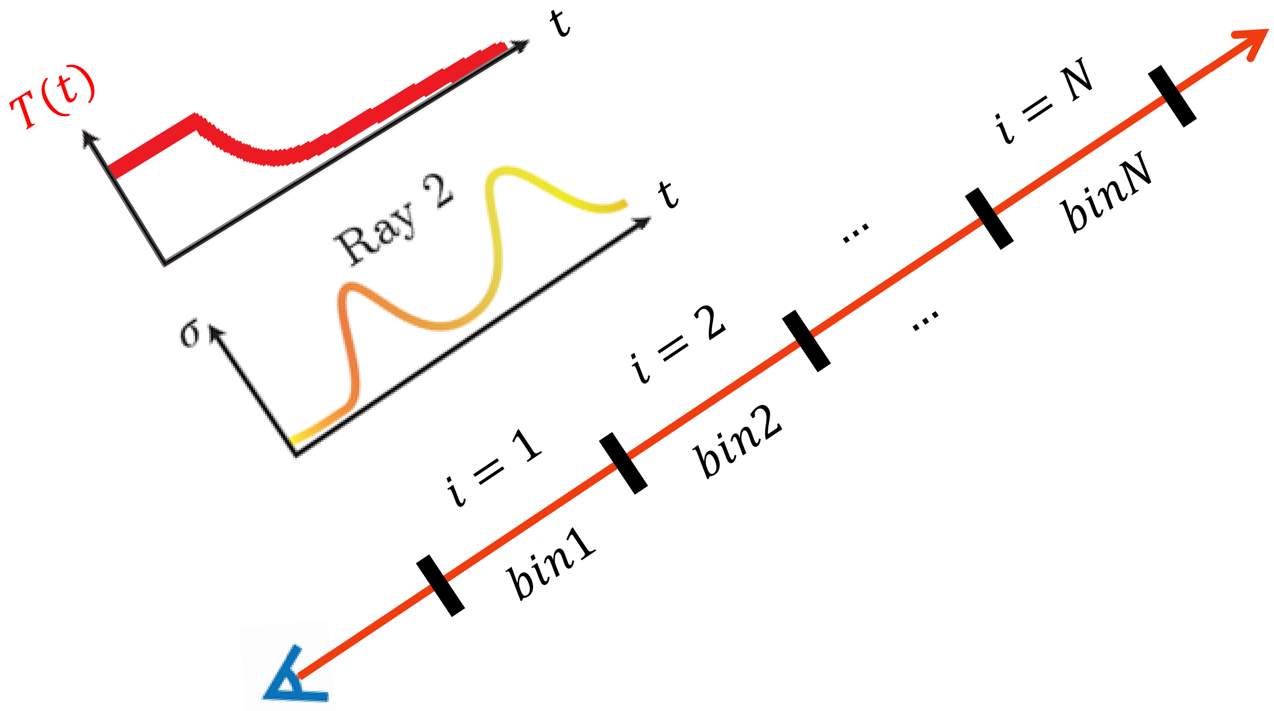
Volume rendering

$$C(r) = \sum_{i=1}^{N} T_i \alpha_i c_i$$

Are you visible?

Are you present?

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



Camera Ray

σ – volume density

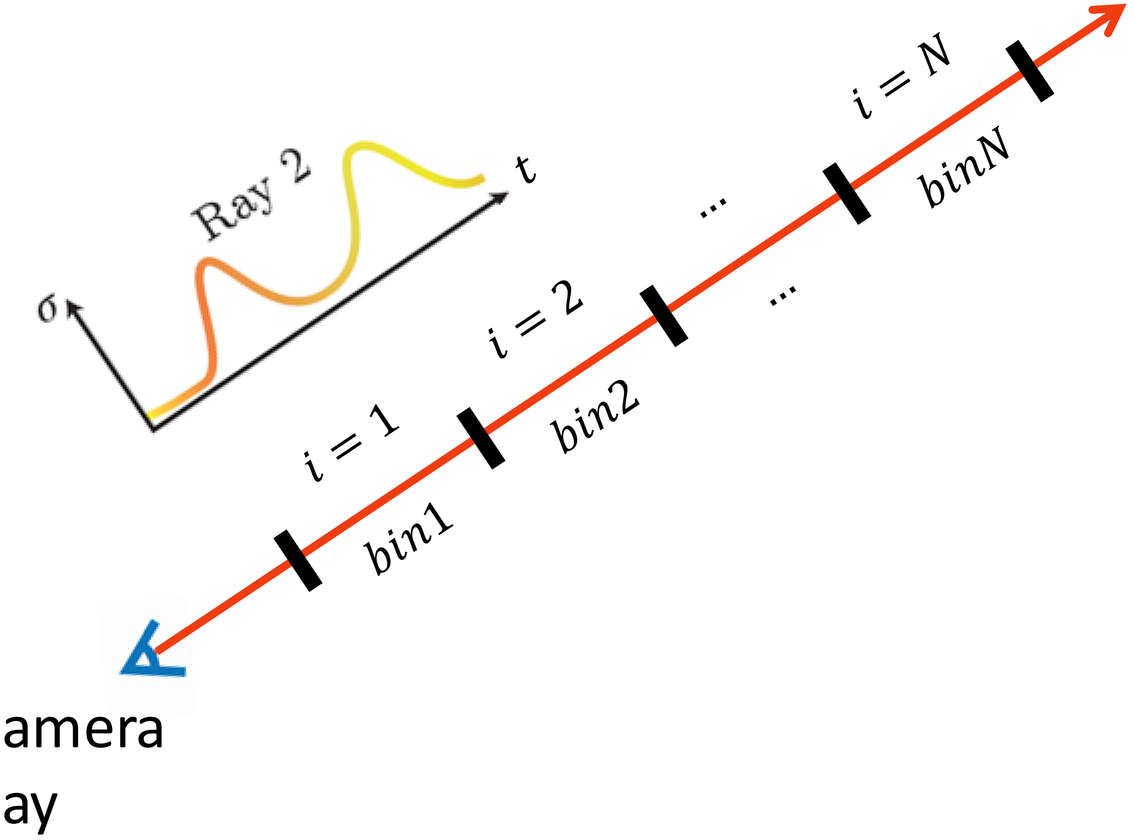
Volume rendering

$$C(r) = \sum_{i=1}^{N} T_i \alpha_i c_i$$

Are you visible?

Are you present?

What is your color?

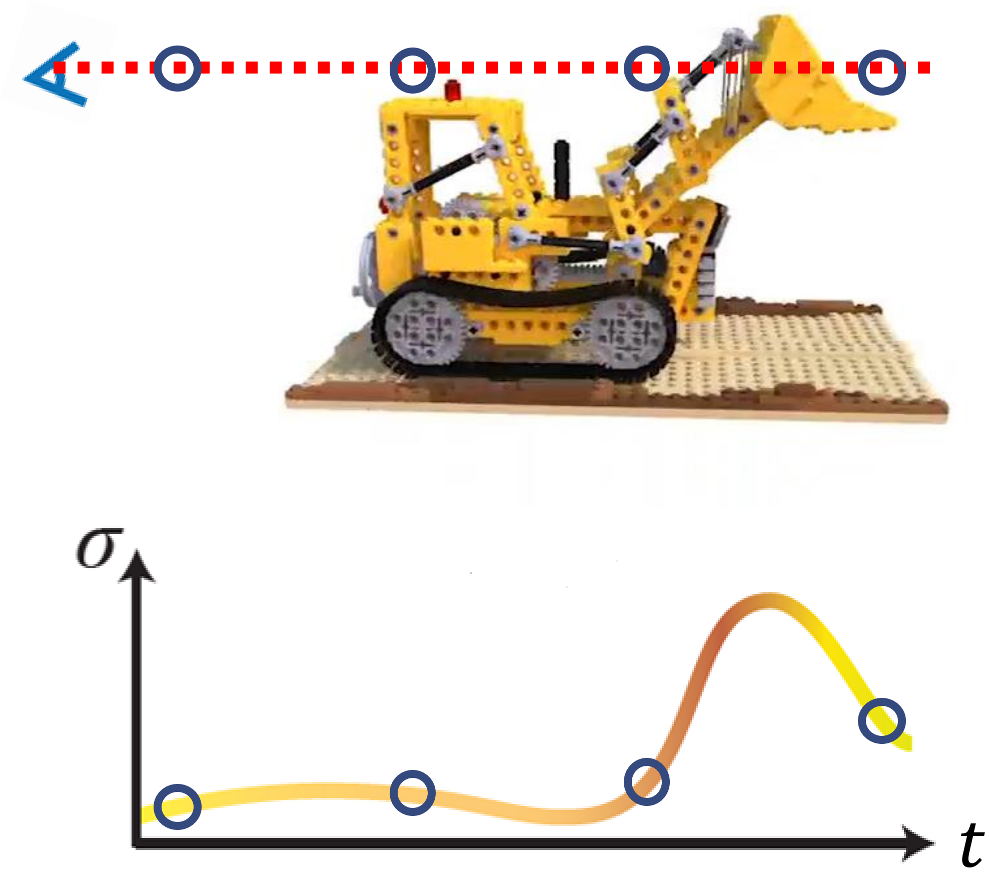


σ – volume density

The Sampling Method

Uniform sampling with a **small** N

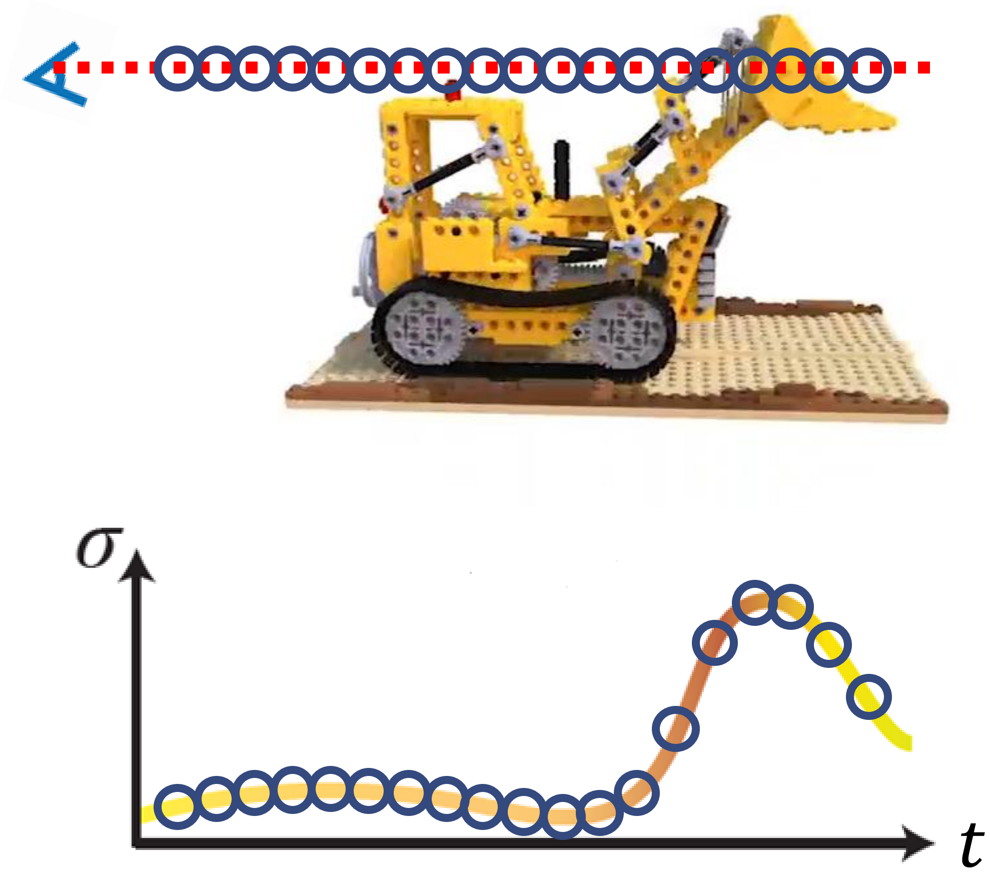
→ Low accuracy



The Sampling Method

Uniform sampling with a **large** N

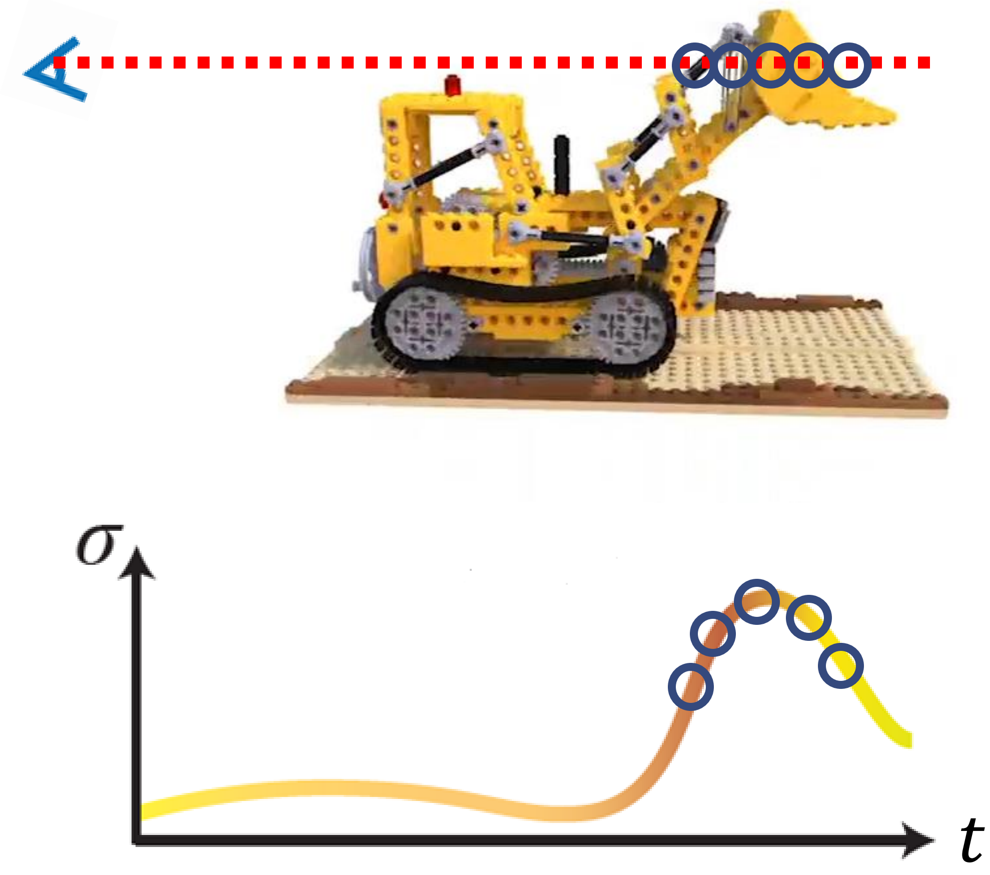
→ Inefficient



The Sampling Method

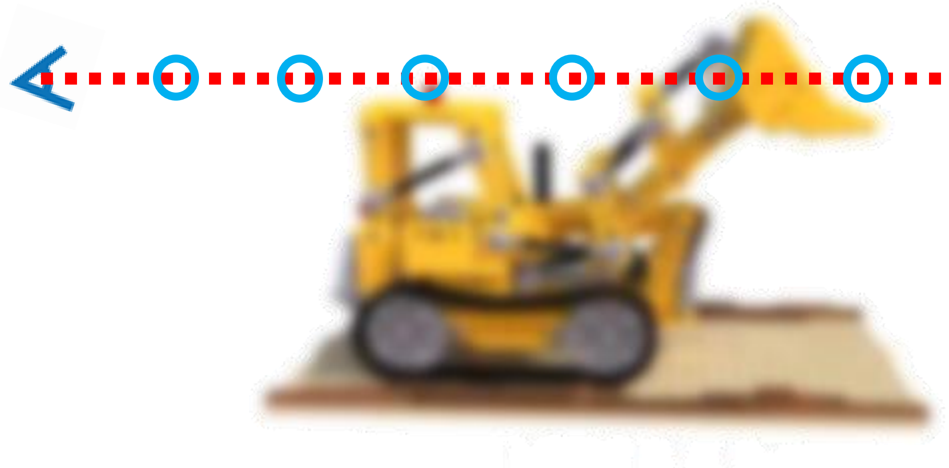
Non-uniform sampling

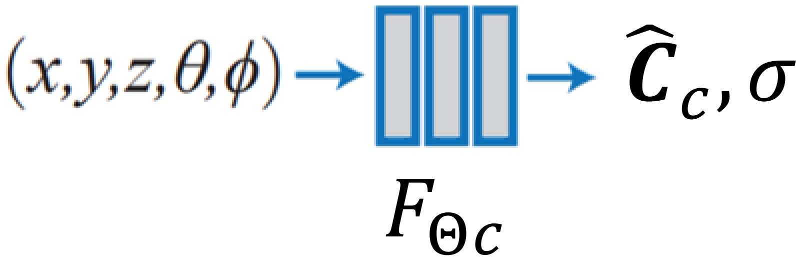
→ How/where?



Hierarchical Volume Rendering

Uniform samples





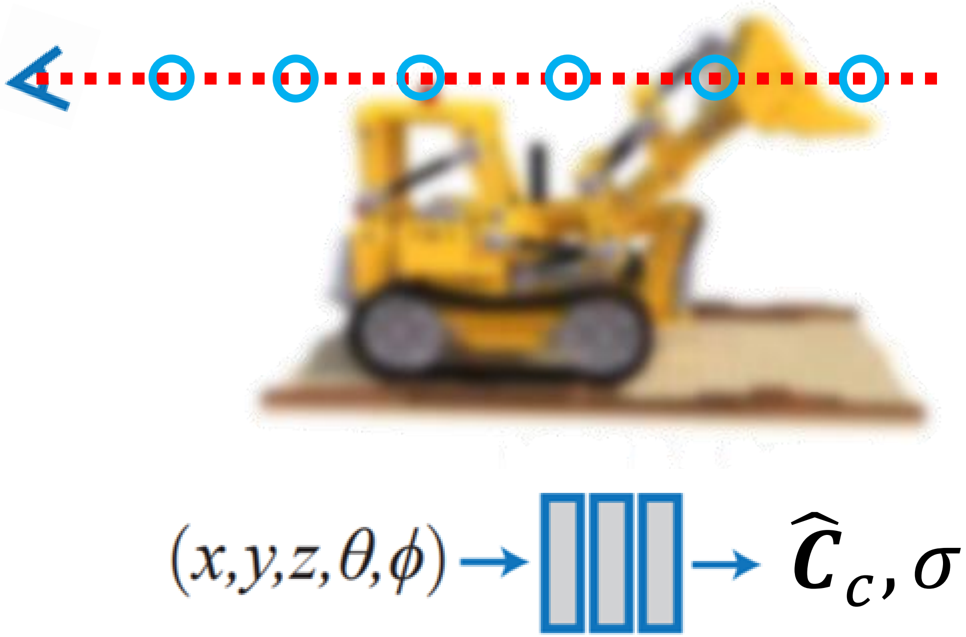
$(x, y, z, \theta, \phi) \rightarrow \quad \rightarrow \hat{C}_c, \sigma$

$F_{\Theta c}$

Coarse NeRF

Hierarchical Volume Rendering

Uniform samples

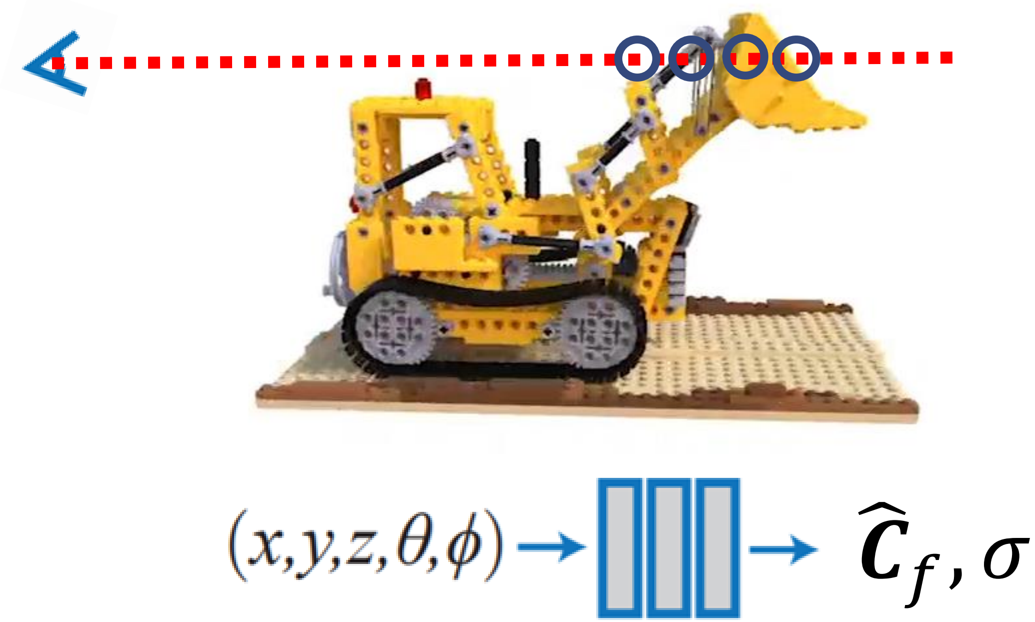


$(x,y,z,\theta,\phi) \rightarrow \boxed{\;} \rightarrow \widehat{\boldsymbol{C}}_c, \sigma$

$F_{\Theta c}$

Coarse NeRF

Non-uniform samples



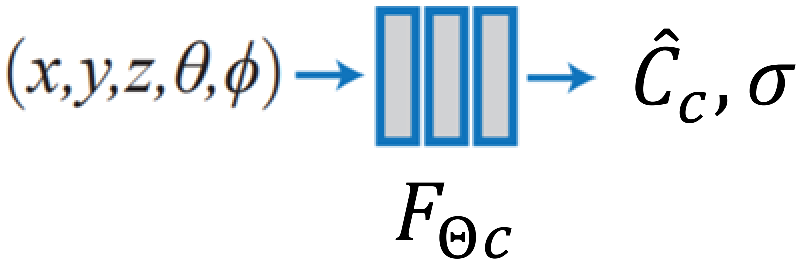
$(x,y,z,\theta,\phi) \rightarrow \boxed{\;} \rightarrow \widehat{\boldsymbol{C}}_f, \sigma$

$F_{\Theta f}$

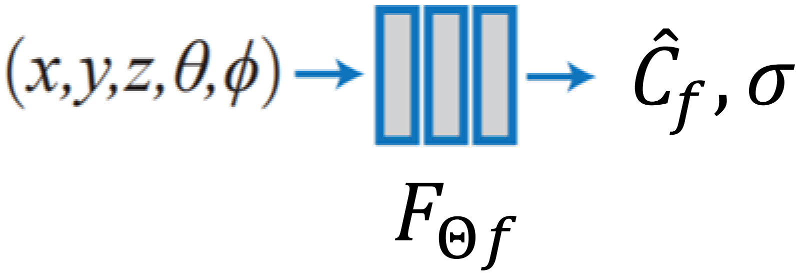
Fine NeRF

Hierarchical Volume Rendering

Train two networks



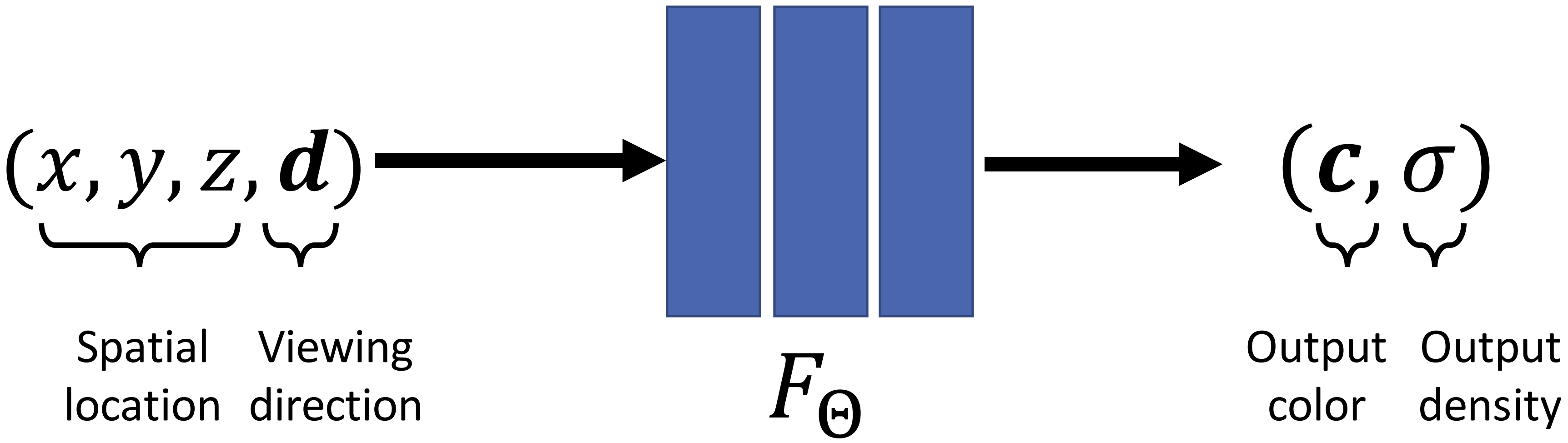
Coarse NeRF



Fine NeRF

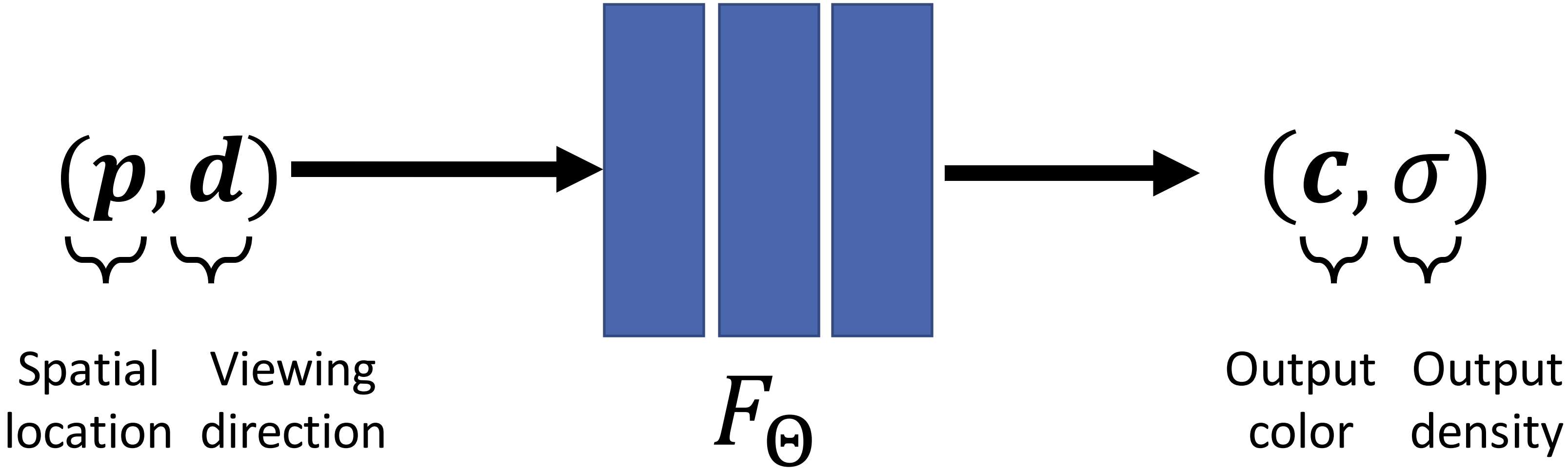
$$Loss = \sum_{r \in \mathcal{R}} \left(\left\| \hat{C}_c(r) - C(r) \right\|_2^2 + \left\| \hat{C}_f(r) - C(r) \right\|_2^2 \right)$$

What else?



$(\underbrace{x, y, z}_{\text{Spatial location}}, \underbrace{\mathbf{d}}_{\text{Viewing direction}}) \longrightarrow F_\Theta \longrightarrow (\underbrace{\mathbf{c}}_{\text{Output color}}, \underbrace{\sigma}_{\text{Output density}})$

What else?



$(\underbrace{\boldsymbol{p}}_{}, \underbrace{\boldsymbol{d}}_{})$

Spatial location

Viewing direction

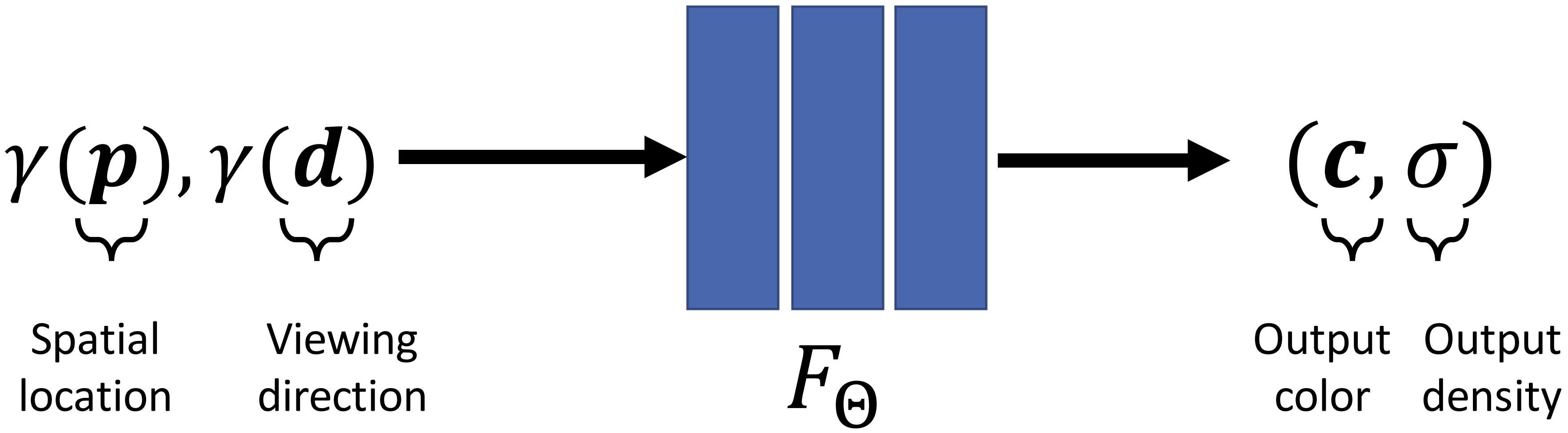
F_{Θ}

$(\underbrace{\boldsymbol{c}}_{}, \underbrace{\sigma}_{})$

Output color

Output density

Positional encoding



$\gamma(\underbrace{\boldsymbol{p}}), \gamma(\underbrace{\boldsymbol{d}})$

Spatial location

Viewing direction

F_Θ

$(\underbrace{\boldsymbol{c}}, \underbrace{\sigma})$

Output color

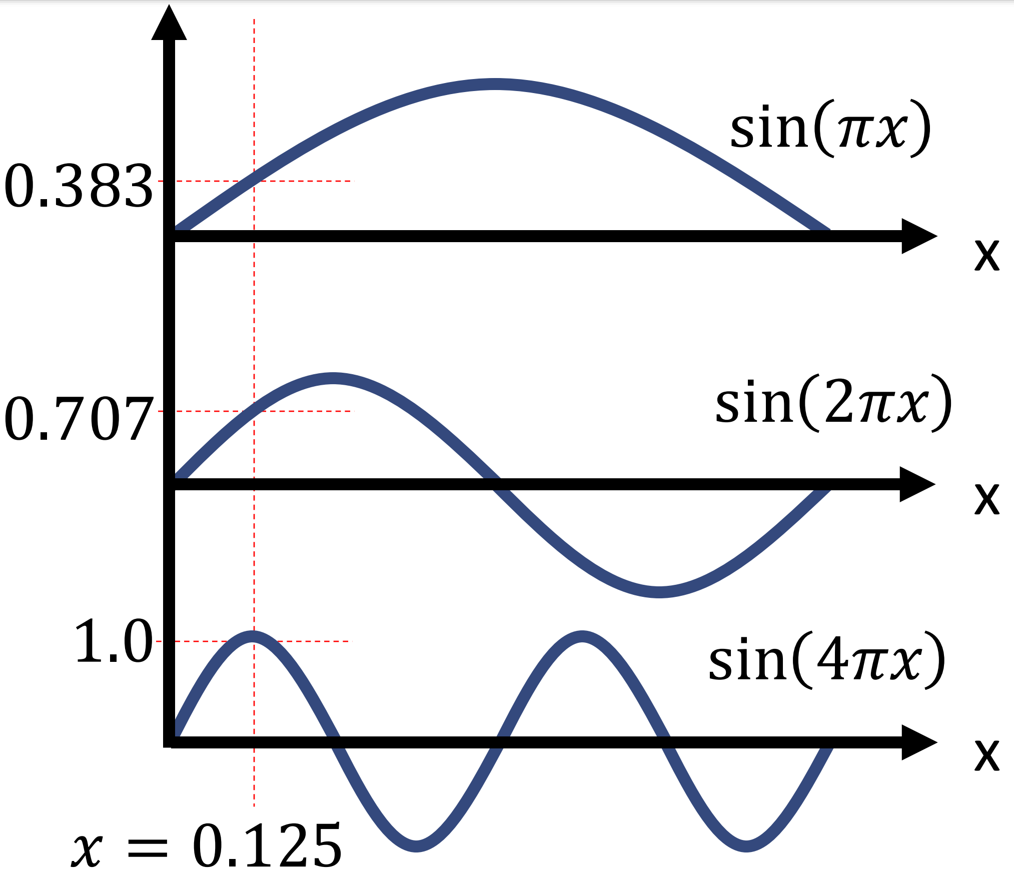
Output density

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

* Vaswani et al. NeurIPS, 2017

Positional encoding – 1D

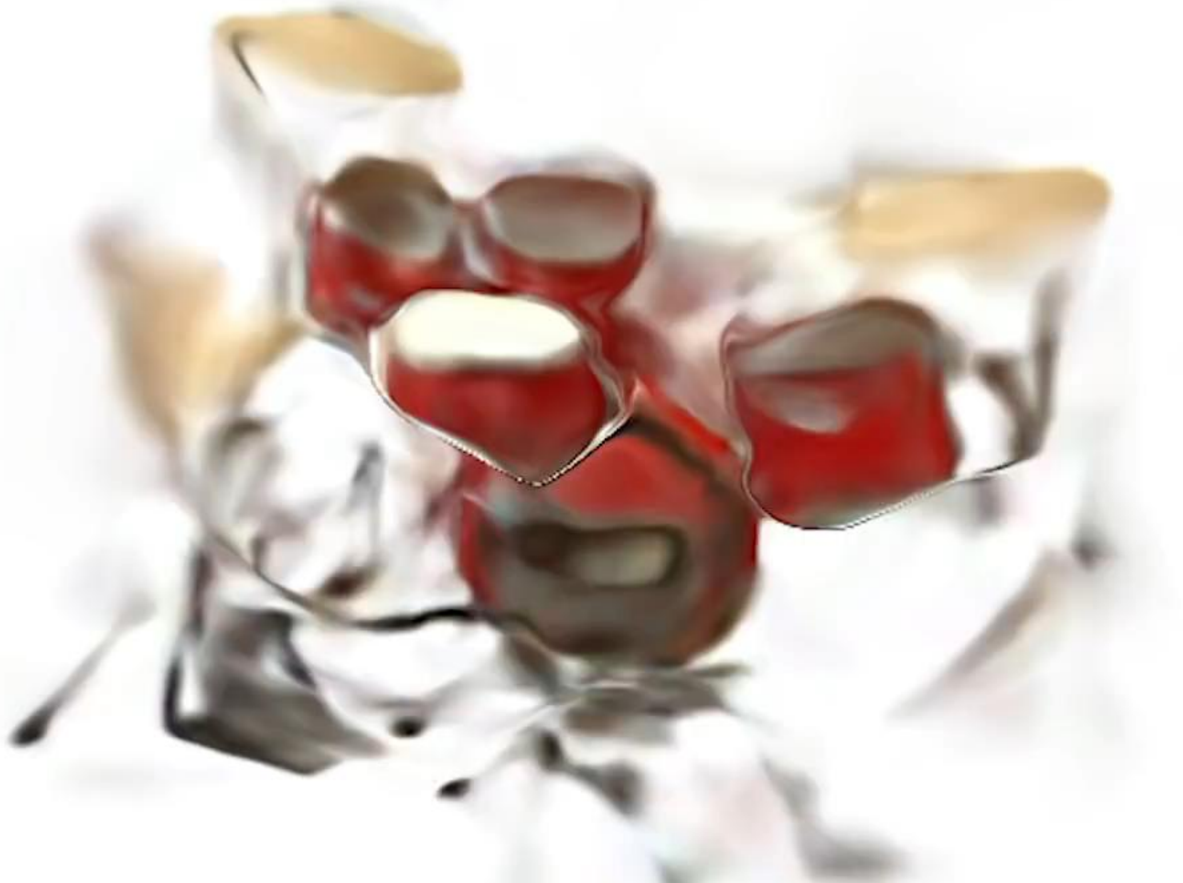
$$\gamma(x = 0.125) = (0.383, 0.707, 1.0)$$



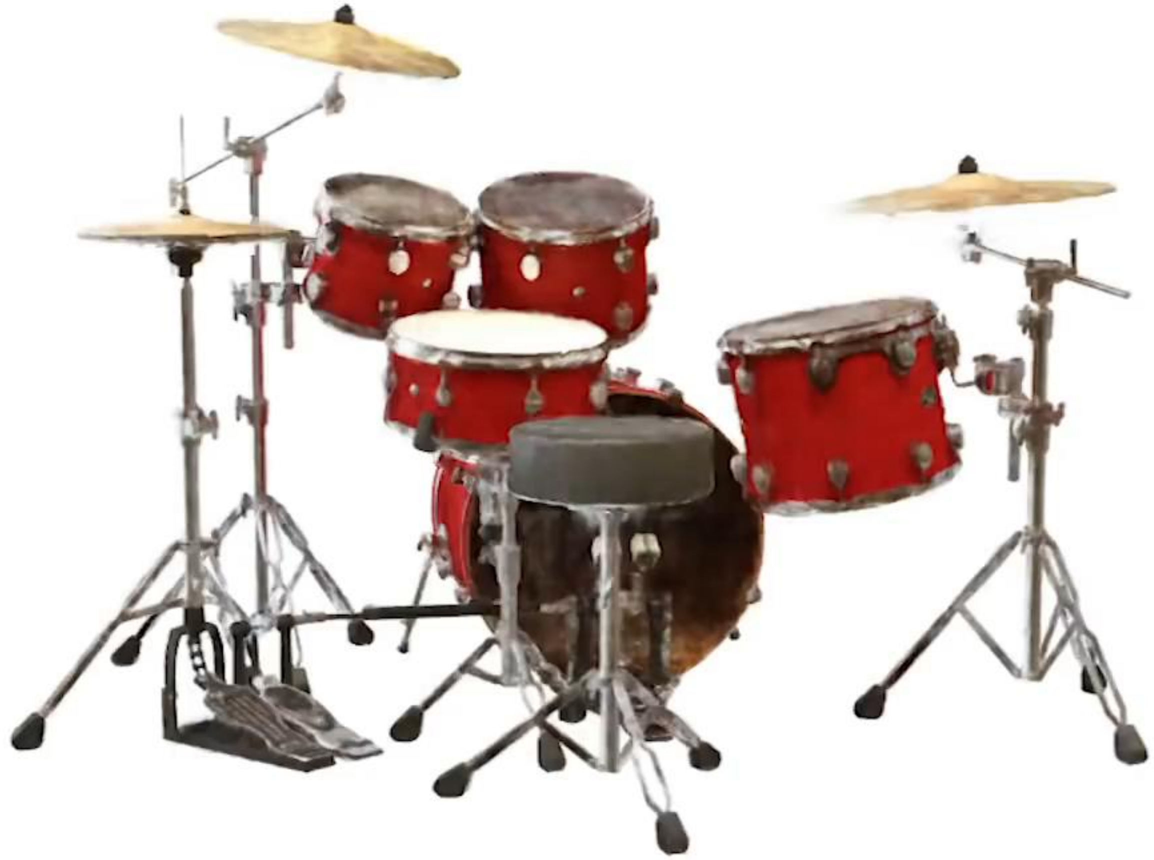
$$\gamma(\boldsymbol{p}) = (\sin(2^0\pi\boldsymbol{p}), \cos(2^0\pi\boldsymbol{p}), \ldots, \sin(2^{L-1}\pi\boldsymbol{p}), \cos(2^{L-1}\pi\boldsymbol{p}))$$

Results
Synthetic Scenes

SRN [Sitzmann 2019]



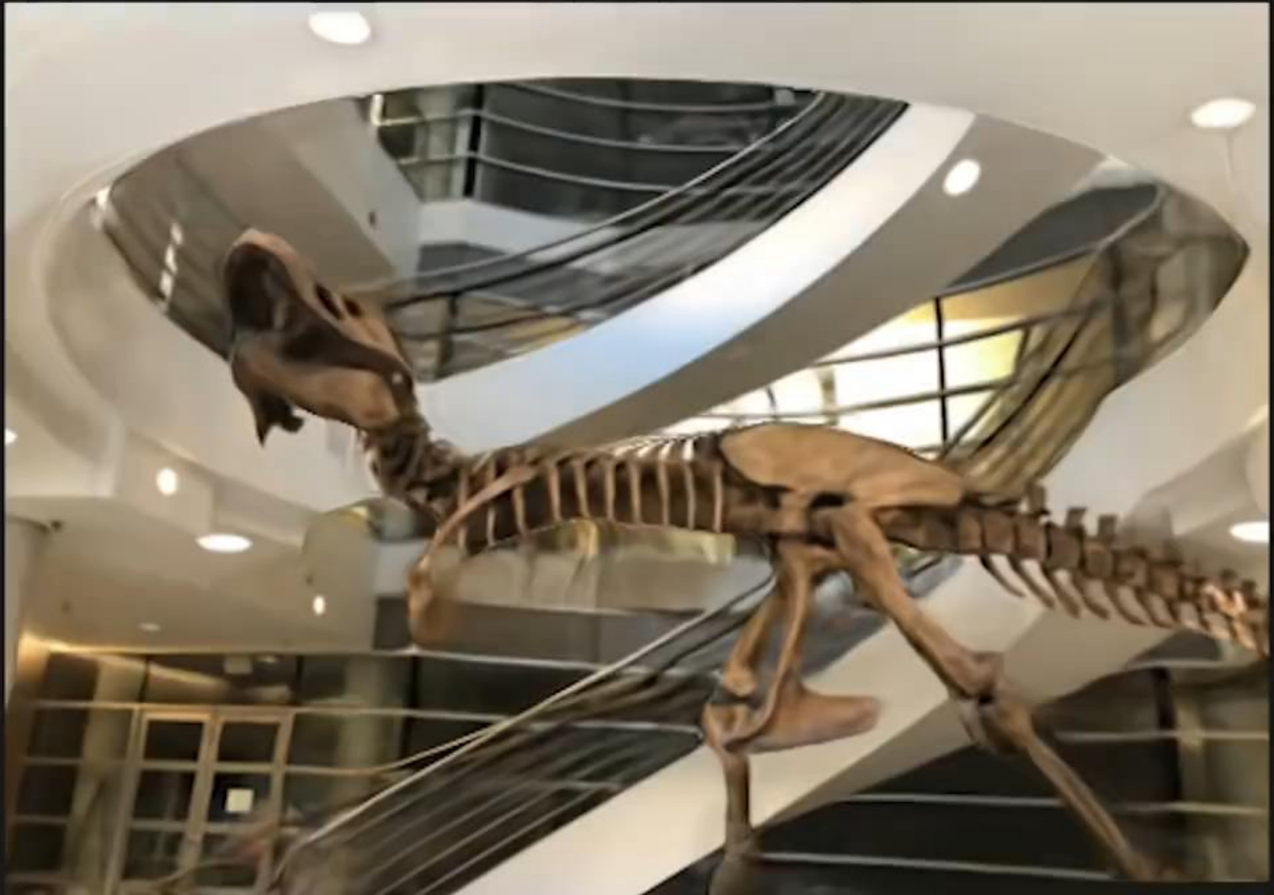
NeRF



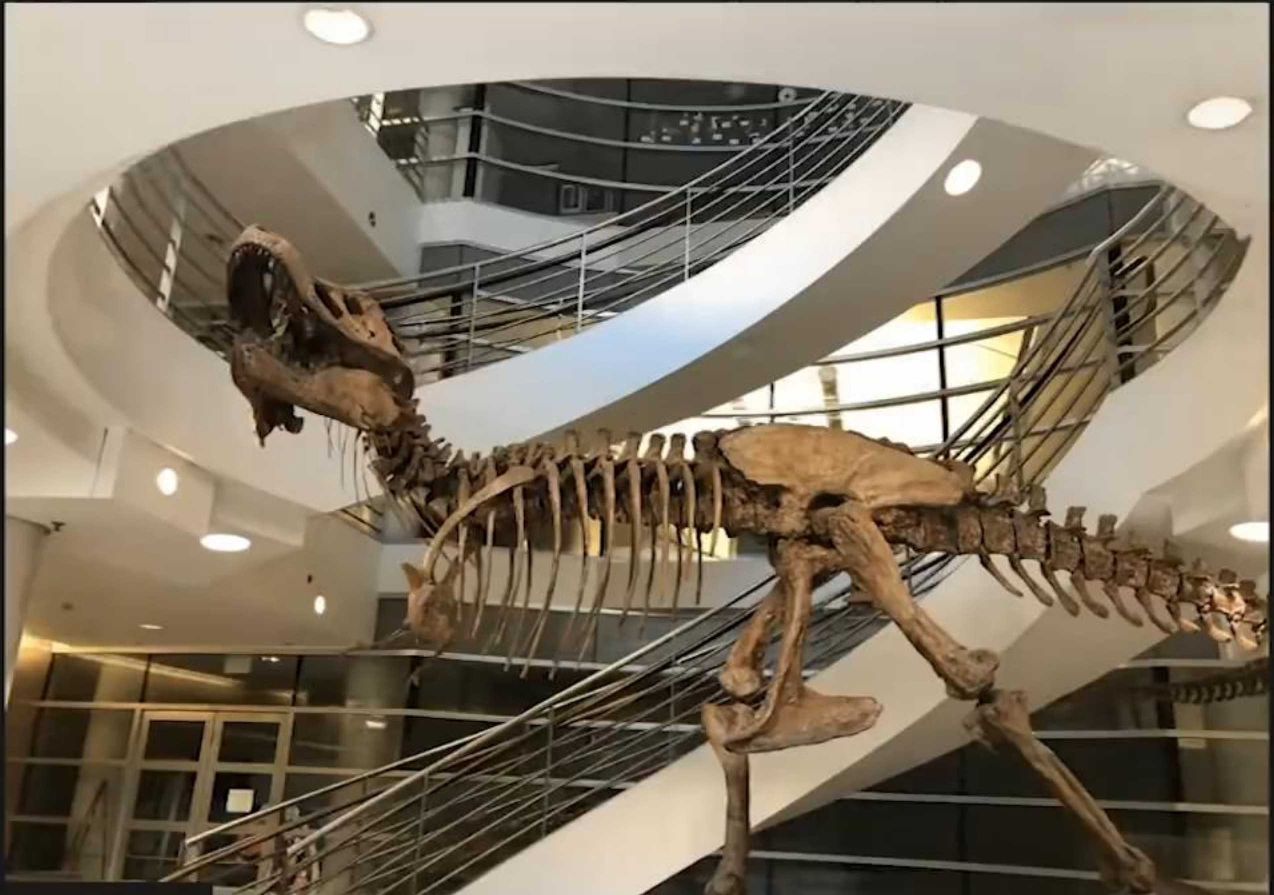
Nearest Input

Results
Real Scenes

SRN [Sitzmann 2019]



NeRF

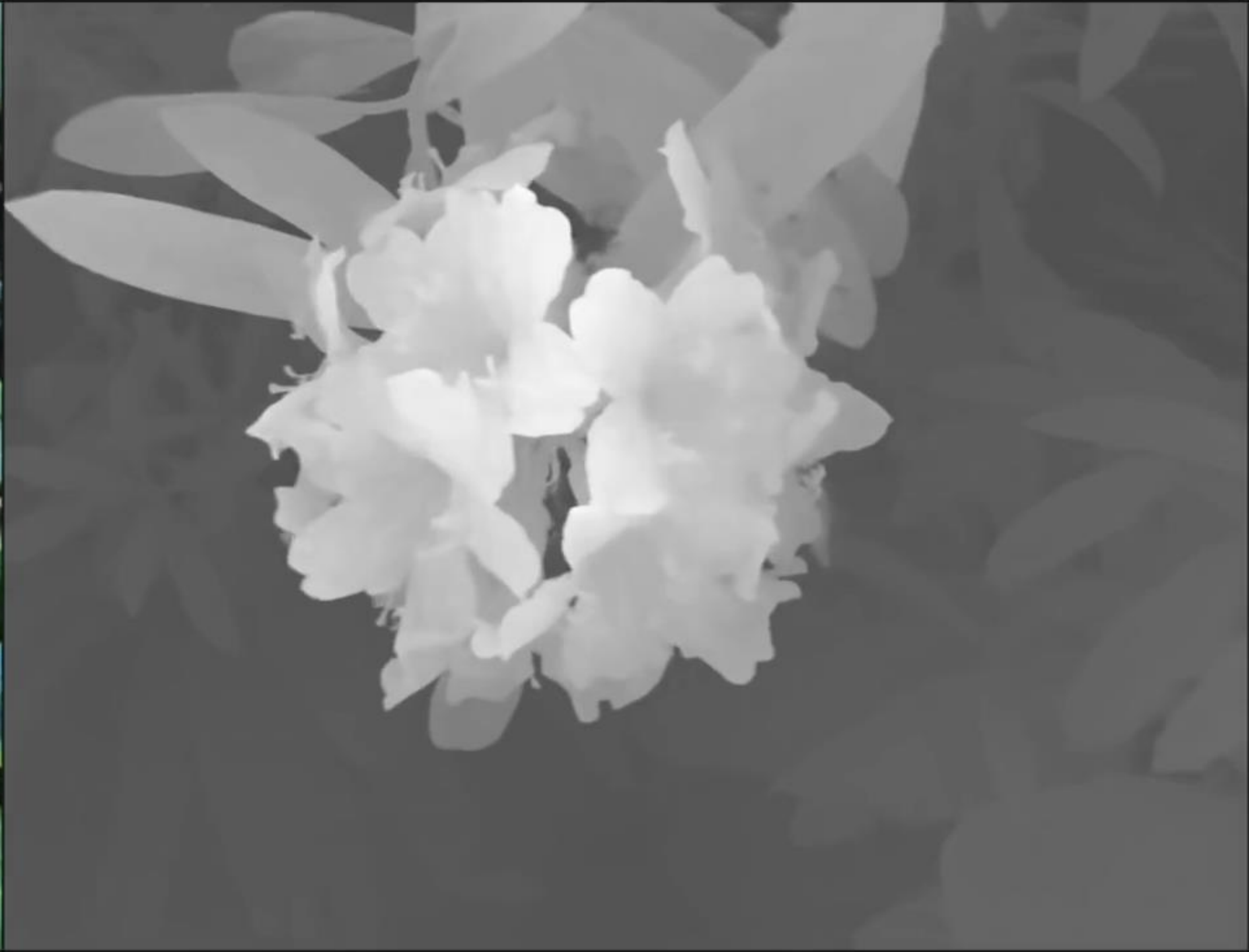


Nearest Input

Results
Representation Benefits

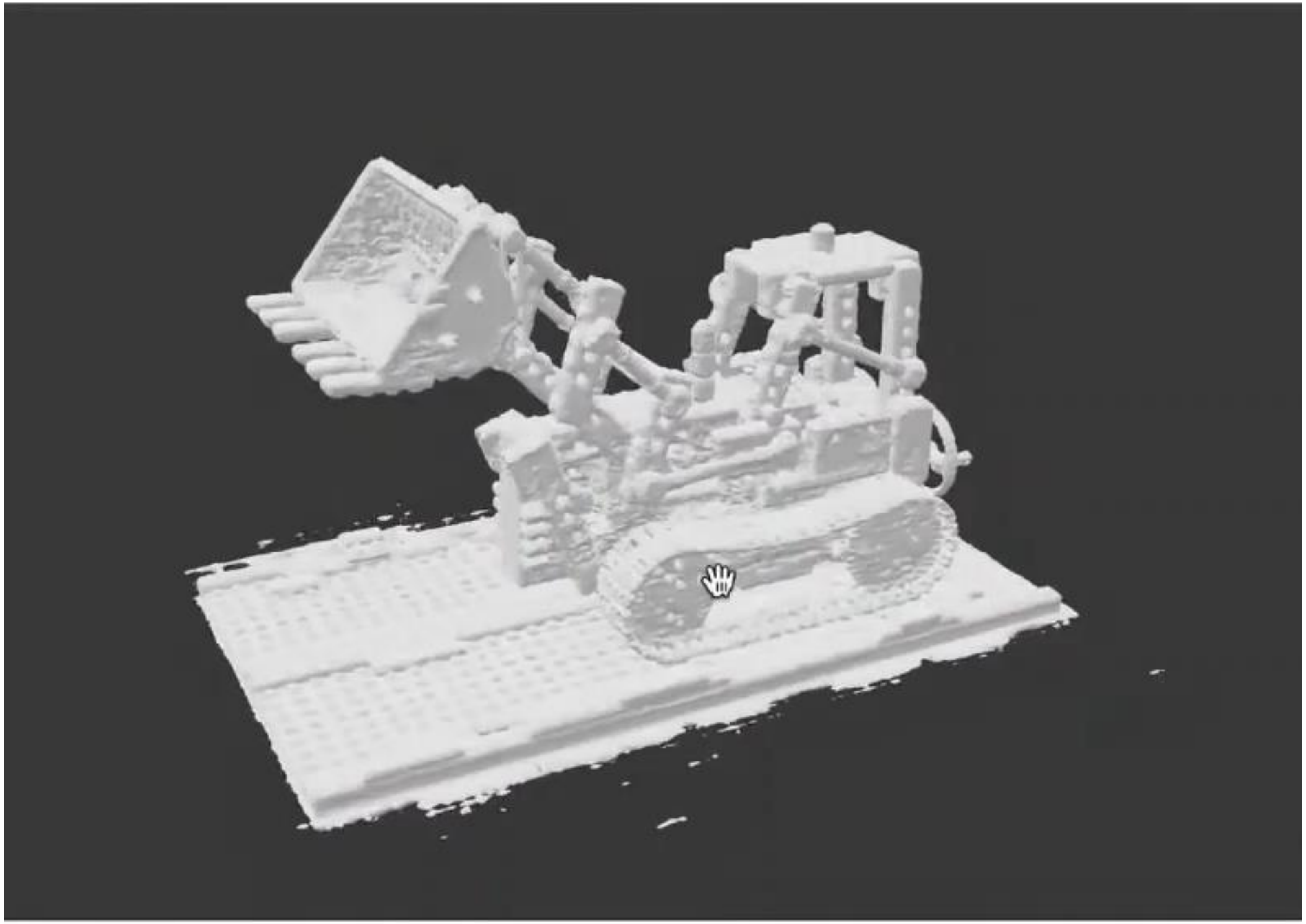
Depth Maps

Rendered Camera Path

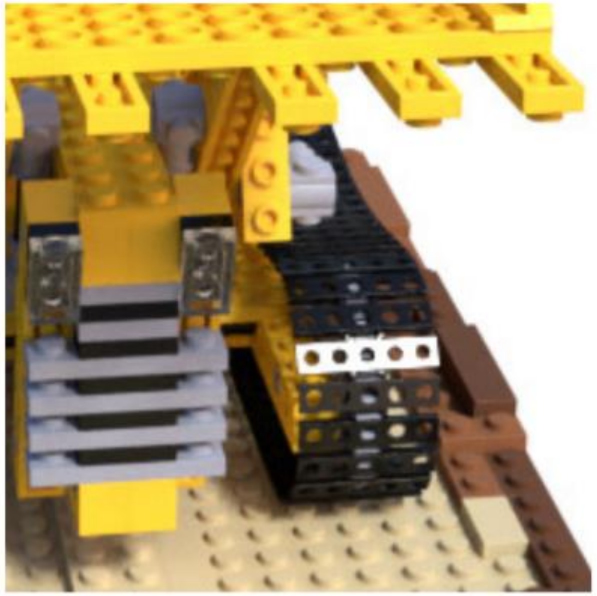


Expected Ray Termination Depth

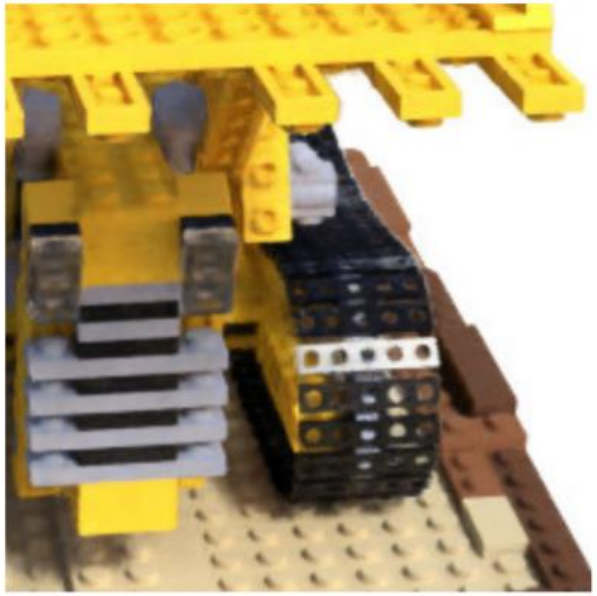
Meshable



Ablation study

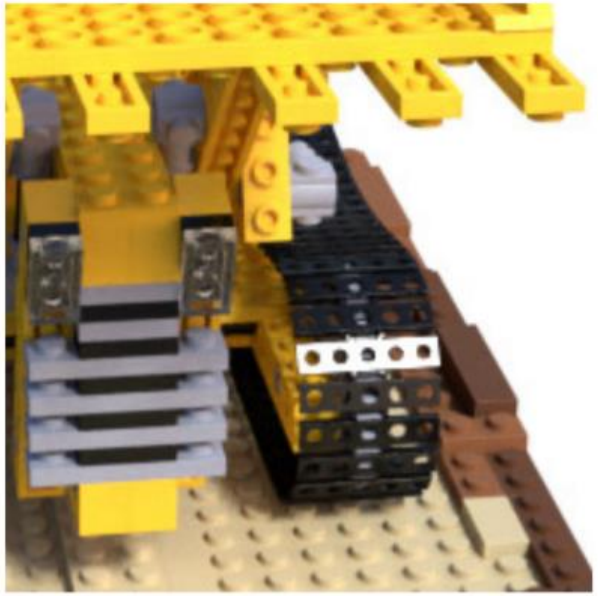


Ground Truth

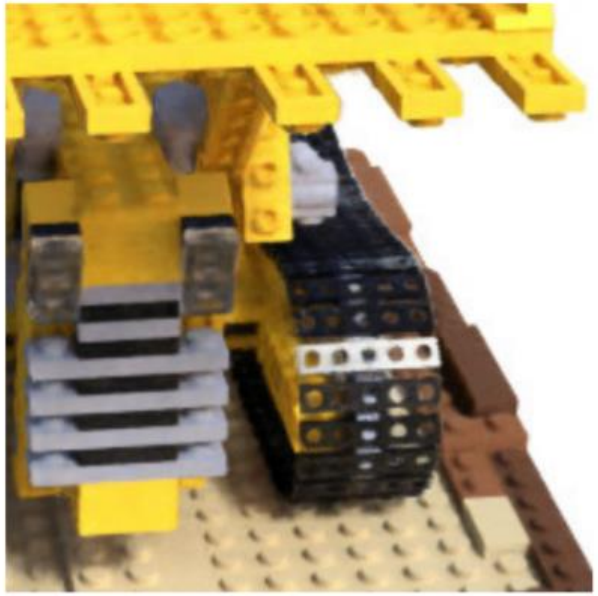


Complete Model

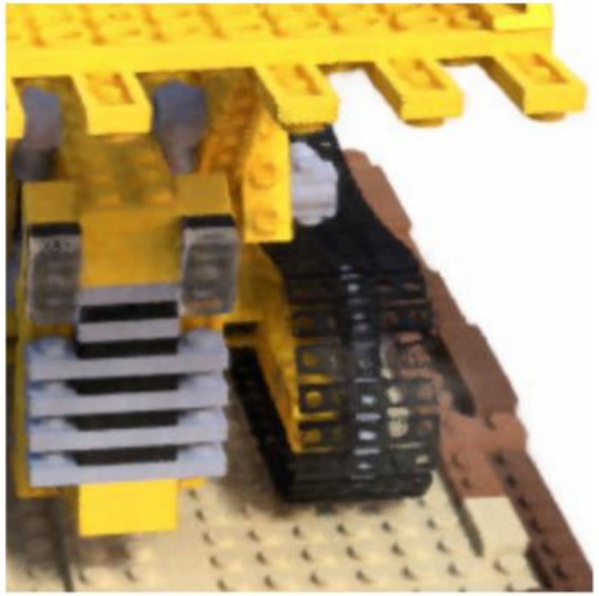
Ablation study



Ground Truth

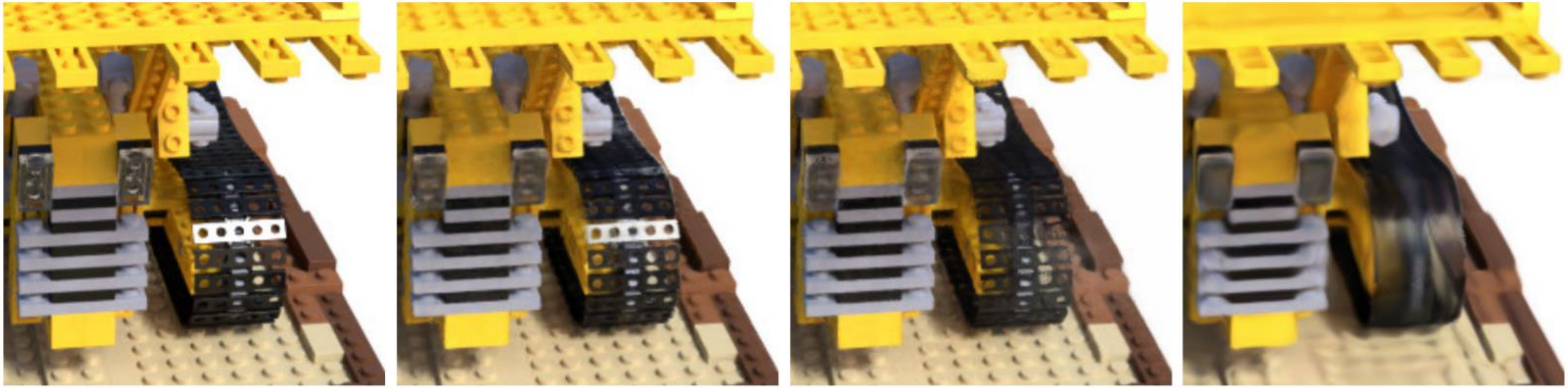


Complete Model



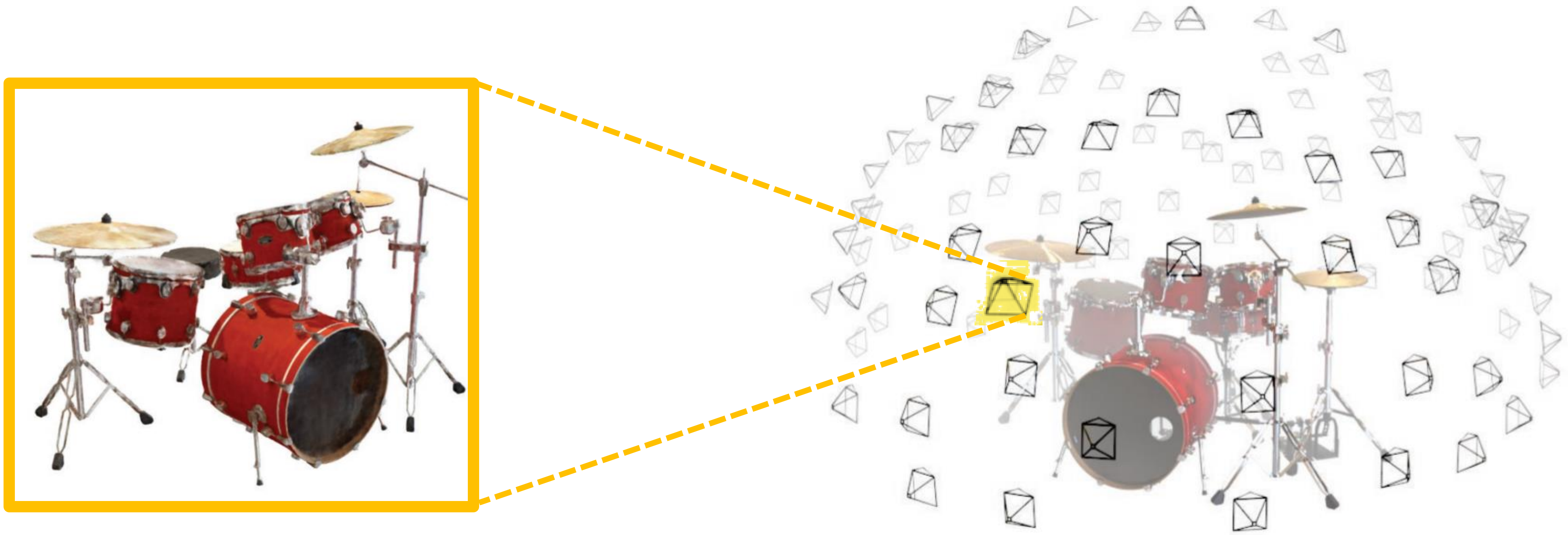
No View Dependence

Ablation study

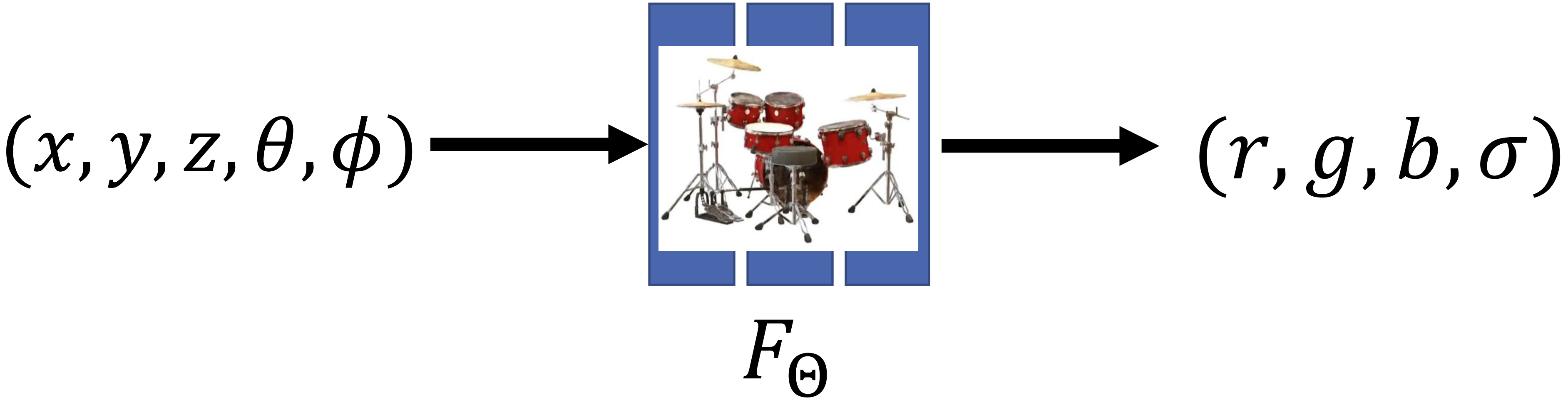


Ground Truth Complete Model No View Dependence No Positional Encoding

NeRF: Summary



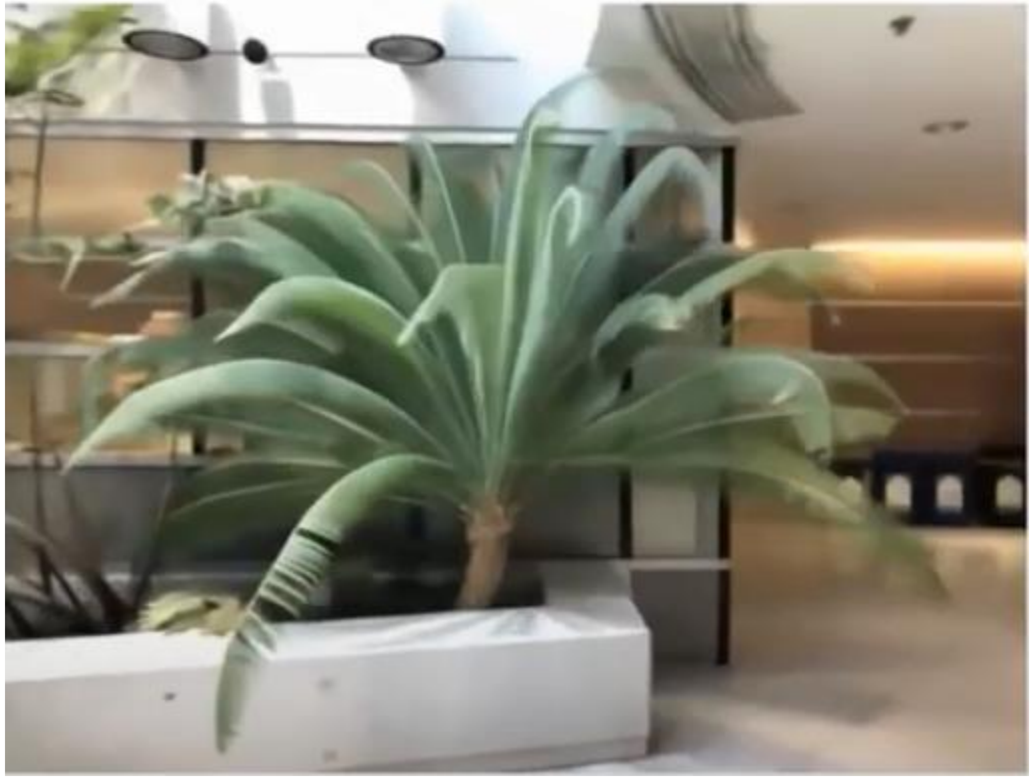
NeRF: Summary



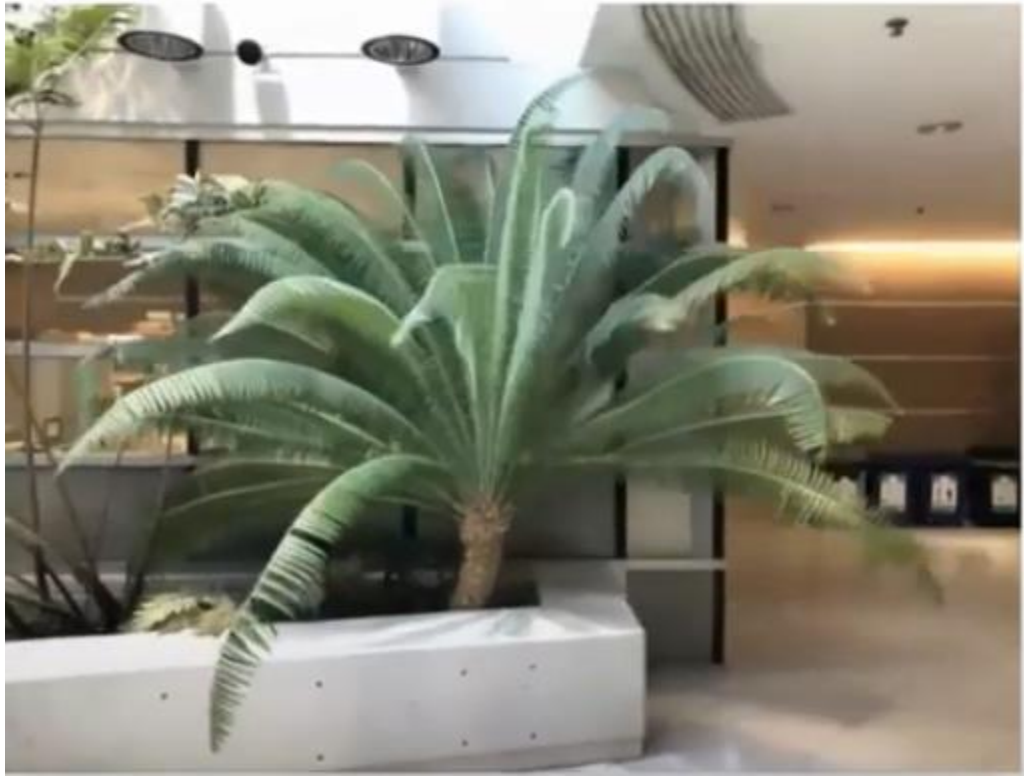
$(x, y, z, \theta, \phi) \rightarrow F_\Theta \rightarrow (r, g, b, \sigma)$

MLP
Architecture

Importance of Positional Encoding

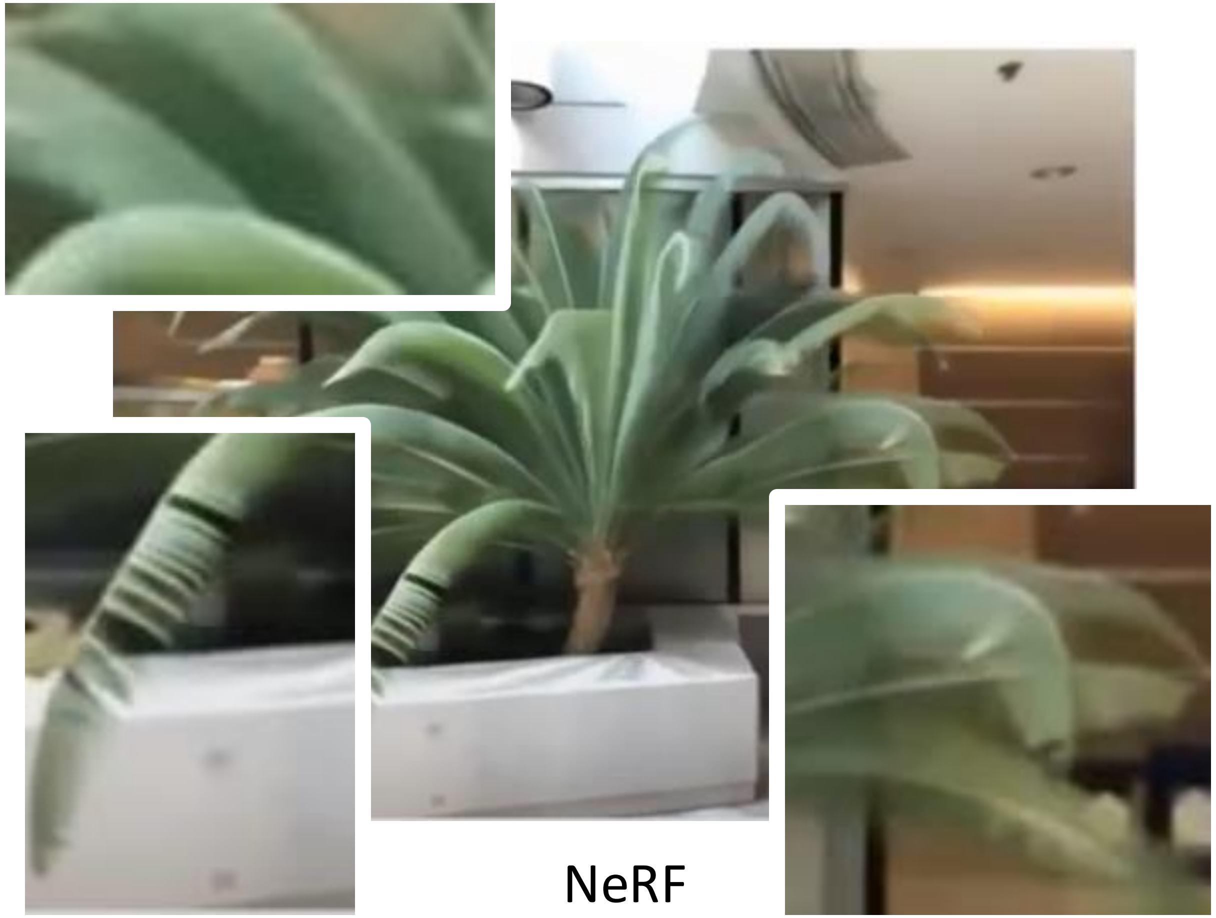


NeRF
No positional encoding



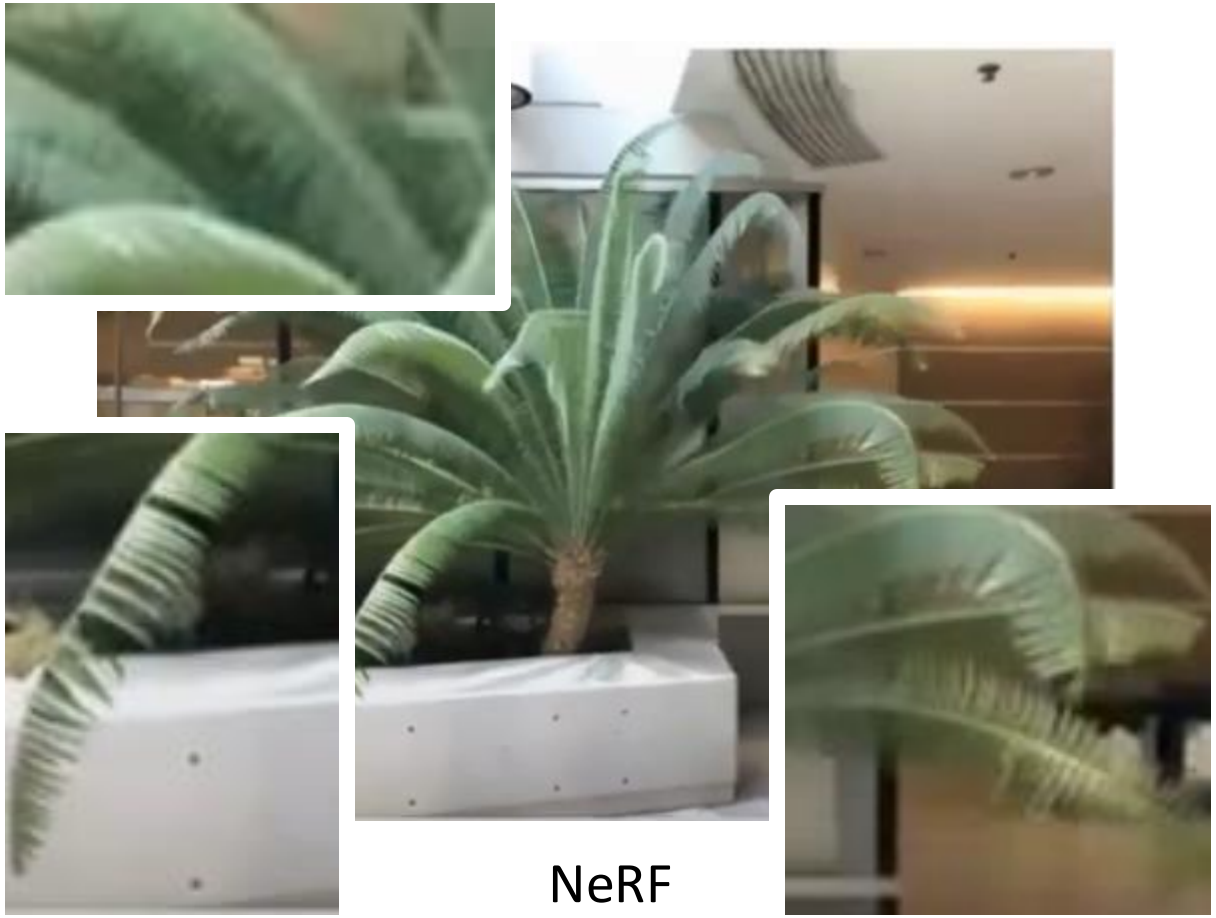
NeRF
With positional encoding

Importance of Positional Encoding



NeRF

No positional encoding



NeRF

With positional encoding

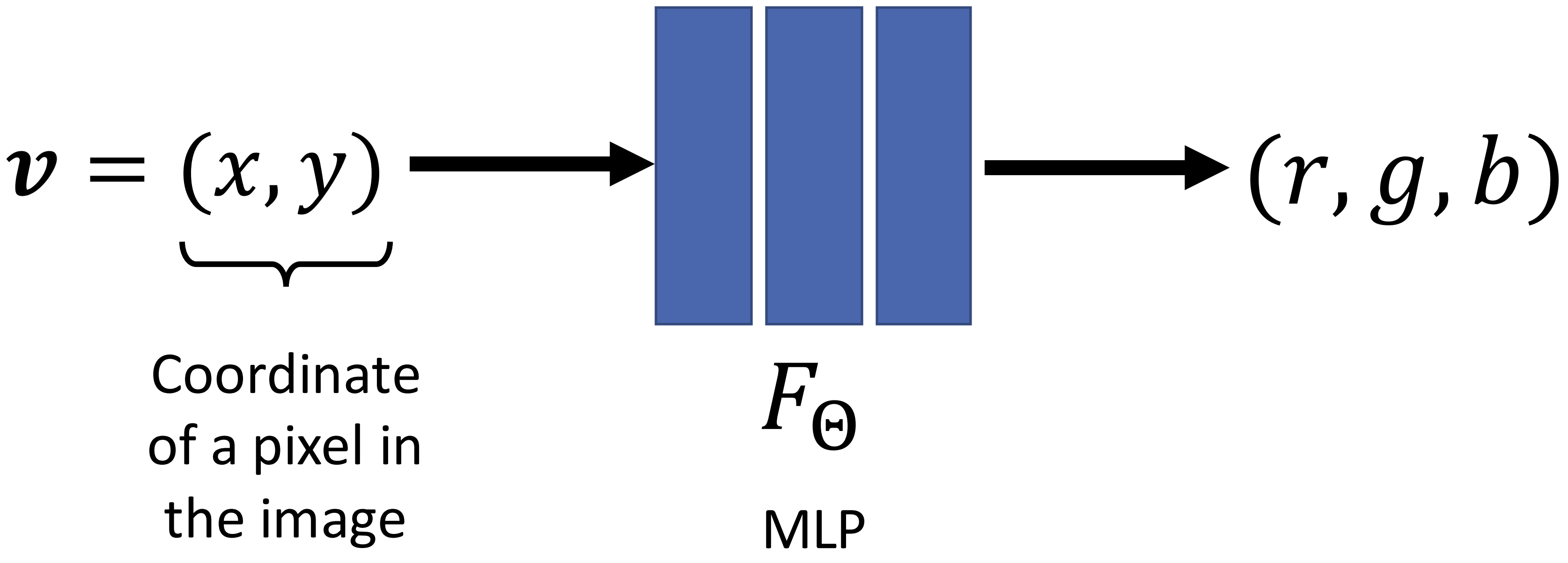
Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains

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

NeurIPS 2020

Problem Setting

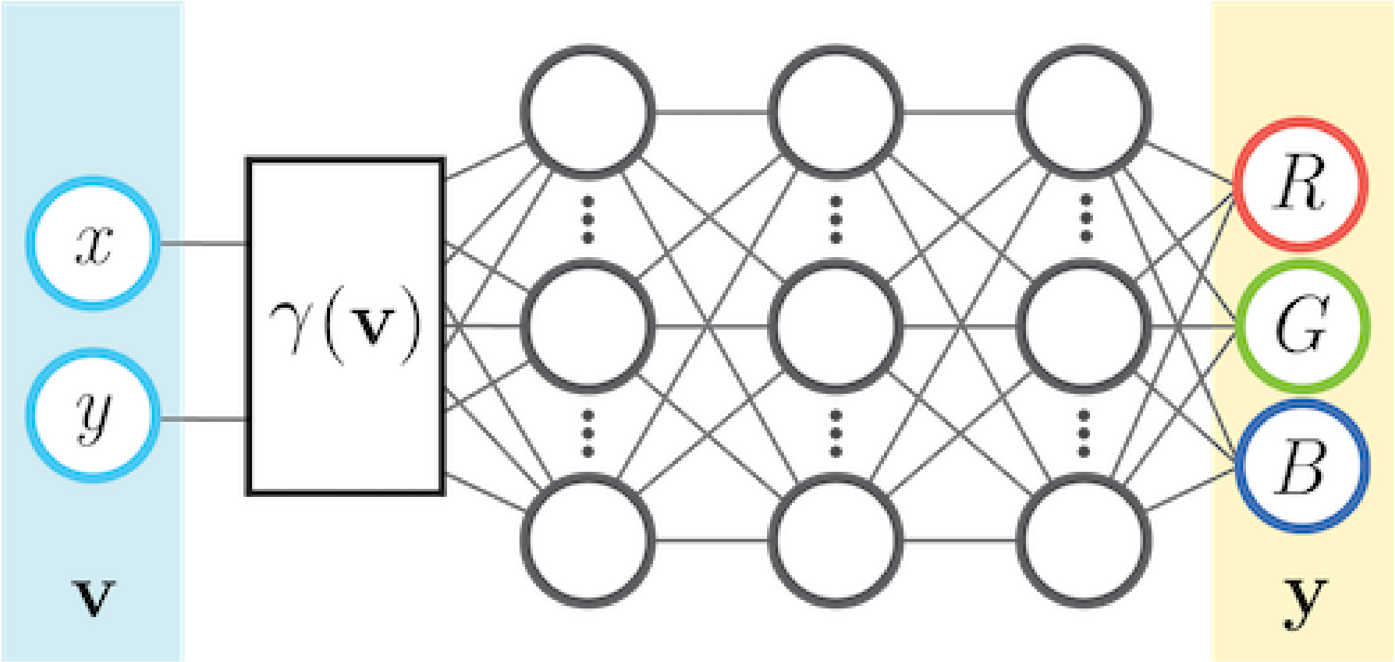
A simpler example: representing a 2D image



$$\boldsymbol{v} = \underbrace{(x, y)}_{\text{Coordinate of a pixel in the image}} \longrightarrow F_{\Theta} \longrightarrow (r, g, b)$$

Problem Setting

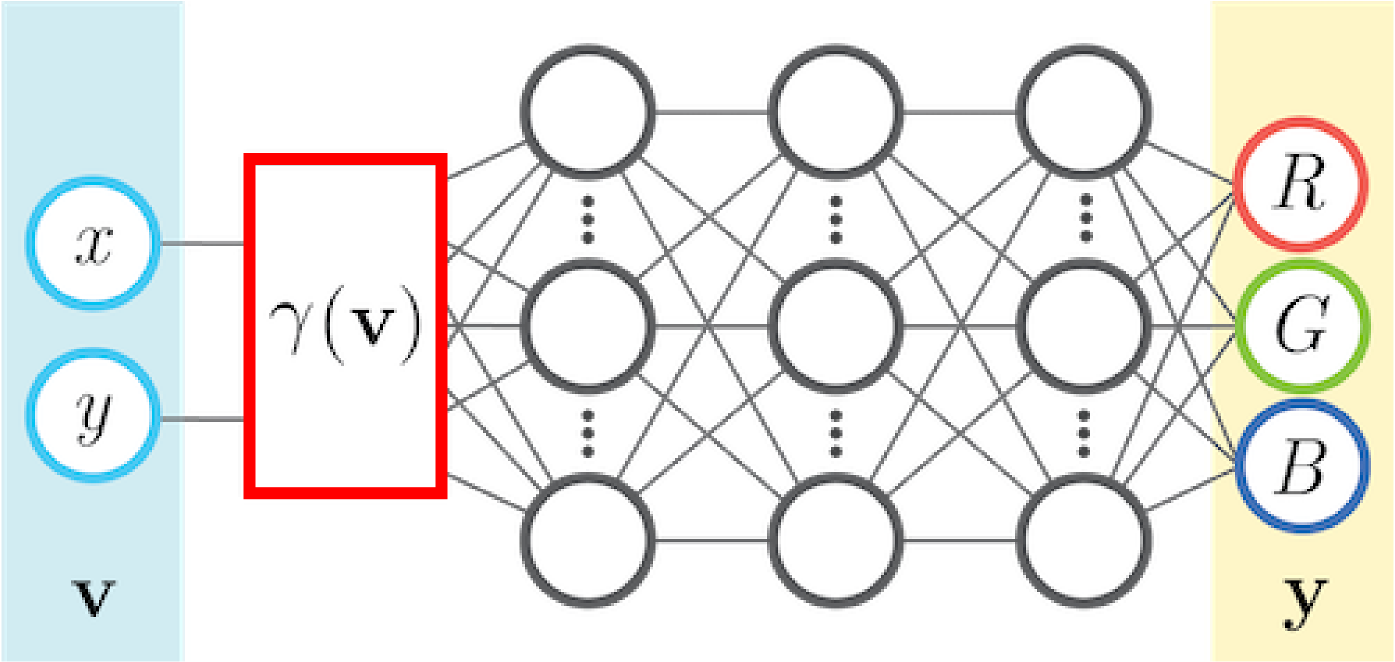
A simpler example: representing a 2D image



In NeRF: $\gamma(\boldsymbol{v}) = (\sin(2^0\pi\boldsymbol{v}), \cos(2^0\pi\boldsymbol{v}), \dots, \sin(2^{L-1}\pi\boldsymbol{v}), \cos(2^{L-1}\pi\boldsymbol{v}))$

Problem Setting

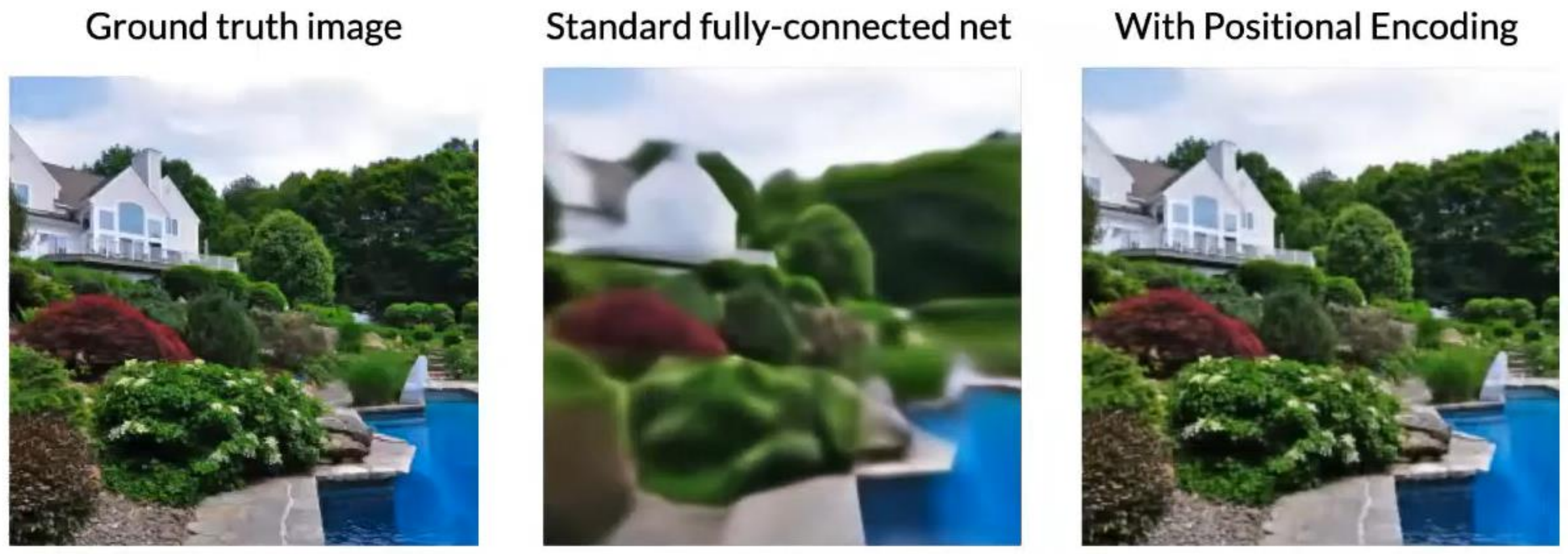
A simpler example: representing a 2D image



In NeRF: $\gamma(\boldsymbol{v}) = (\sin(2^0 \pi \boldsymbol{v}), \cos(2^0 \pi \boldsymbol{v}), \ldots, \sin(2^{L-1} \pi \boldsymbol{v}), \cos(2^{L-1} \pi \boldsymbol{v}))$

Positional Encoding – With or Without?

Feeding a 2D image to a simple MLP doesn’t work



Ground truth image

Standard fully-connected net

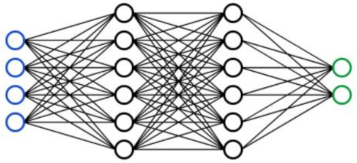
With Positional Encoding

Tools

- Theorical:
  Input mapping using Fourier features works – why?

- + Experimental:
  Dive into different mappings and check what’s important

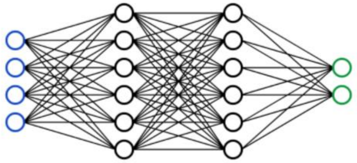
Theory:
Neural Tangent Kernel (NTK)



Defined
architecture +
training data

Jacot et al., NeurIPS 2018; Arora et al., ICML 2019; Basri et al. 2020; Du et al., ICML 2019; Lee et al., NeurIPS 2019 and more

Theory:
Neural Tangent Kernel (NTK)

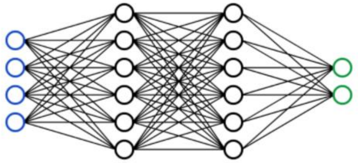


*Defined architecture + training data

*under certain conditions

Jacot et al., NeurIPS 2018; Arora et al., ICML 2019; Basri et al. 2020; Du et al., ICML 2019; Lee et al., NeurIPS 2019 and more

Theory:
Neural Tangent Kernel (NTK)



*Defined architecture + training data

*under certain conditions

→

NTK method

→

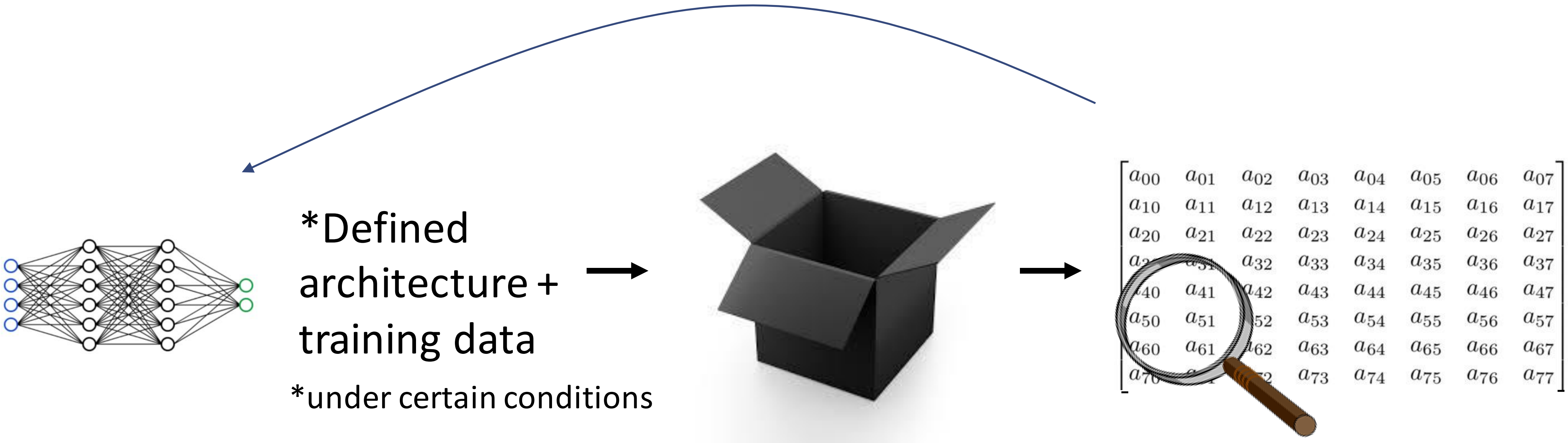
$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} & a_{04} & a_{05} & a_{06} & a_{07} \\ a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} \\ a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} \\ a_{30} & a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & a_{37} \\ a_{40} & a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} \\ a_{50} & a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} \\ a_{60} & a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} & a_{67} \\ a_{70} & a_{71} & a_{72} & a_{73} & a_{74} & a_{75} & a_{76} & a_{77} \end{bmatrix}$$

K_{NTK}

$n \times n$

Jacot et al., NeurIPS 2018; Arora et al., ICML 2019; Basri et al. 2020; Du et al., ICML 2019; Lee et al., NeurIPS 2019 and more

Theory:
Neural Tangent Kernel (NTK)



*Defined architecture + training data →

*under certain conditions

→

NTK method

K_{NTK}

$n \times n$

Jacot et al., NeurIPS 2018; Arora et al., ICML 2019; Basri et al. 2020; Du et al., ICML 2019; Lee et al., NeurIPS 2019 and more

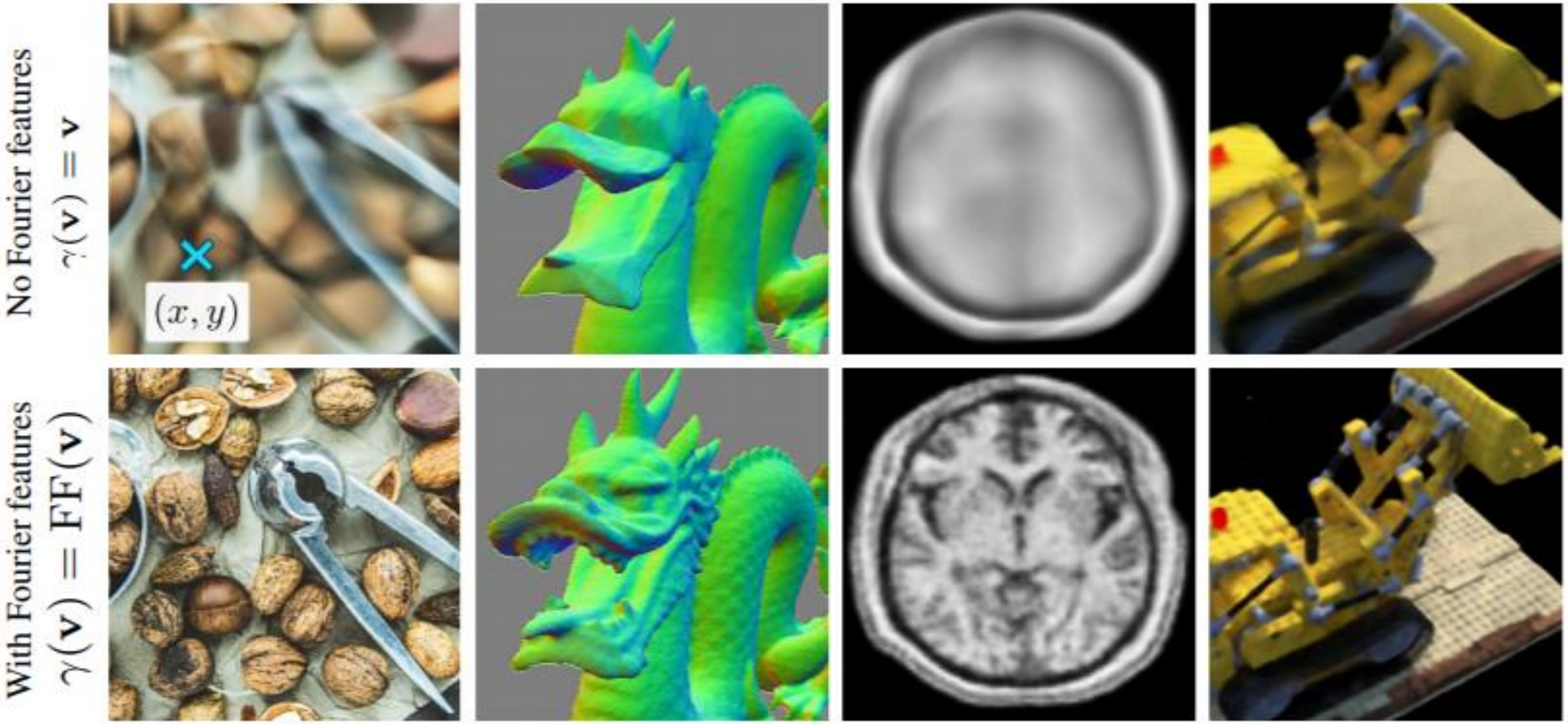
Theory:
Neural Tangent Kernel (NTK)

Used the NTK method to show:

- No input mapping → "spectral bias"
- Can overcome this bias using Fourier feature mapping

Jacot et al., NeurIPS 2018; Arora et al., ICML 2019; Basri et al. 2020; Du et al., ICML 2019; Lee et al., NeurIPS 2019 and more

Different Experiment Domains


No Fourier features
$\gamma(\mathbf{v}) = \mathbf{v}$
With Fourier features
$\gamma(\mathbf{v}) = \mathrm{FF}(\mathbf{v})$
(x, y)

(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

Input Mappings

Basic:

$$\gamma(\boldsymbol{v}) = [\cos(2\pi\boldsymbol{v}), \sin(2\pi\boldsymbol{v})]$$

Input Mappings

Basic:

$$\gamma(\boldsymbol{v}) = [\cos(2\pi\boldsymbol{v}), \sin(2\pi\boldsymbol{v})]$$

Positional Encoding:

$$\gamma(\boldsymbol{v}) = \left[\ldots, a_j\cos\left(2\pi\sigma^{j/m}\boldsymbol{v}\right), a_j\sin\left(2\pi\sigma^{j/m}\boldsymbol{v}\right), \ldots\right], \ j = 0, \ldots, m-1$$

m – number of frequencies

Input Mappings

Basic:

$$\gamma(\boldsymbol{v}) = [\cos(2\pi\boldsymbol{v}), \sin(2\pi\boldsymbol{v})]$$

Positional Encoding:

$$\gamma(\boldsymbol{v}) = \left[\dots, a_j \cos\left(2\pi\sigma^{j/m}\boldsymbol{v}\right), a_j \sin\left(2\pi\sigma^{j/m}\boldsymbol{v}\right), \dots\right], \; j = 0, \dots, m-1$$

m – number of frequencies

Gaussian Random Fourier Features (RFF)*:

$$\gamma(\boldsymbol{v}) = [\cos(2\pi\boldsymbol{B}\boldsymbol{v}), \sin(2\pi\boldsymbol{B}\boldsymbol{v})], \qquad \boldsymbol{B} \sim N(0, \sigma^2), \qquad \boldsymbol{B} \in \mathbb{R}^{m \times d}$$

*Rahimi & Recht. NeurIPS, 2007

Input Mappings

Basic:

$$\gamma(\boldsymbol{v}) = [\cos(2\pi\boldsymbol{v}), \sin(2\pi\boldsymbol{v})]$$

Positional Encoding:

$$\gamma(\boldsymbol{v}) = \left[\ldots, a_j \cos\left(2\pi\sigma^{j/m}\boldsymbol{v}\right), a_j \sin\left(2\pi\sigma^{j/m}\boldsymbol{v}\right), \ldots\right],\ j = 0, \ldots, m-1$$

m – number of frequencies

Gaussian Random Fourier Features (RFF)*:

$$\gamma(\boldsymbol{v}) = [\cos(2\pi\boldsymbol{B}\boldsymbol{v}), \sin(2\pi\boldsymbol{B}\boldsymbol{v})], \qquad \boldsymbol{B}\sim N(0,\sigma^2), \qquad \boldsymbol{B} \in \mathbb{R}^{m\times d}$$

*Rahimi & Recht. NeurIPS, 2007

Input Mappings

Basic:

$$\gamma(\boldsymbol{v}) = [\cos(2\pi\boldsymbol{v}), \sin(2\pi\boldsymbol{v})]$$

Positional Encoding:

$$\gamma(\boldsymbol{v}) = \left[\ldots, a_j \cos\left(2\pi\sigma^{j/m}\boldsymbol{v}\right), a_j \sin\left(2\pi\sigma^{j/m}\boldsymbol{v}\right), \ldots\right], \; j = 0, \ldots, m-1$$

m – number of frequencies

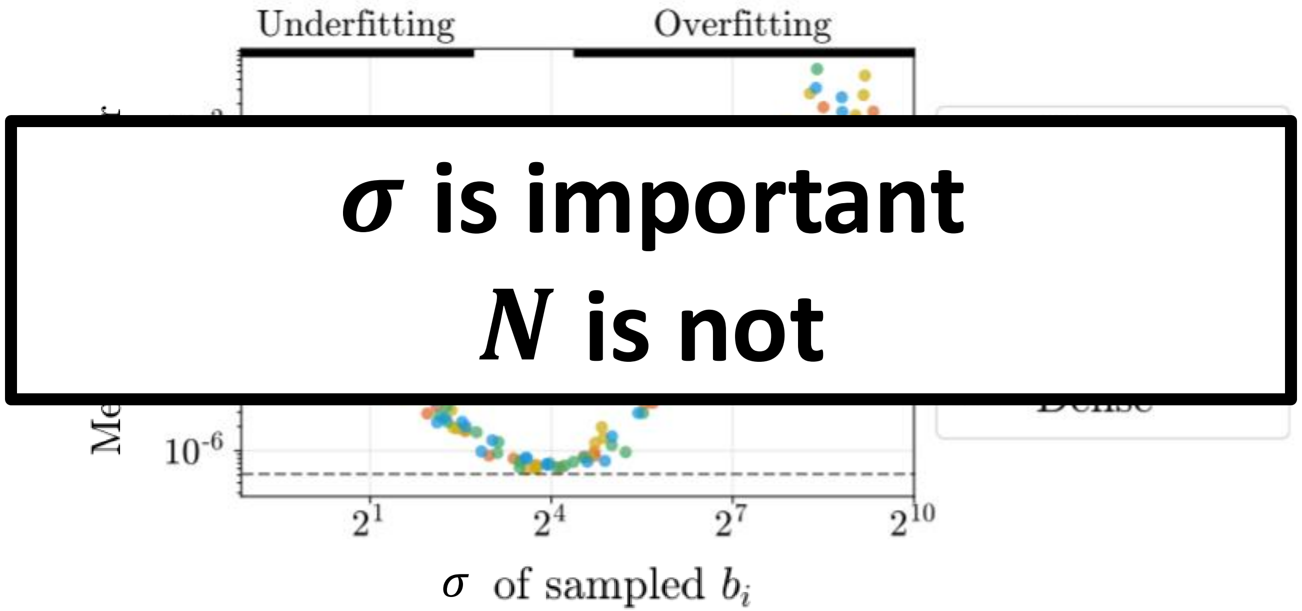
Gaussian Random Fourier Features (RFF)*:

$$\gamma(\boldsymbol{v}) = [\cos(2\pi\boldsymbol{B}\boldsymbol{v}), \sin(2\pi\boldsymbol{B}\boldsymbol{v})], \qquad \boldsymbol{B} \sim \boxed{\mathcal{N}}(0, \boxed{\sigma^2}), \qquad \boldsymbol{B} \in \mathbb{R}^{m \times d}$$

*Rahimi & Recht. NeurIPS, 2007

Distribution Types and Mapping Bandwidth

Gaussian RFF: 1D experiment



$\boldsymbol{\sigma}$ is important

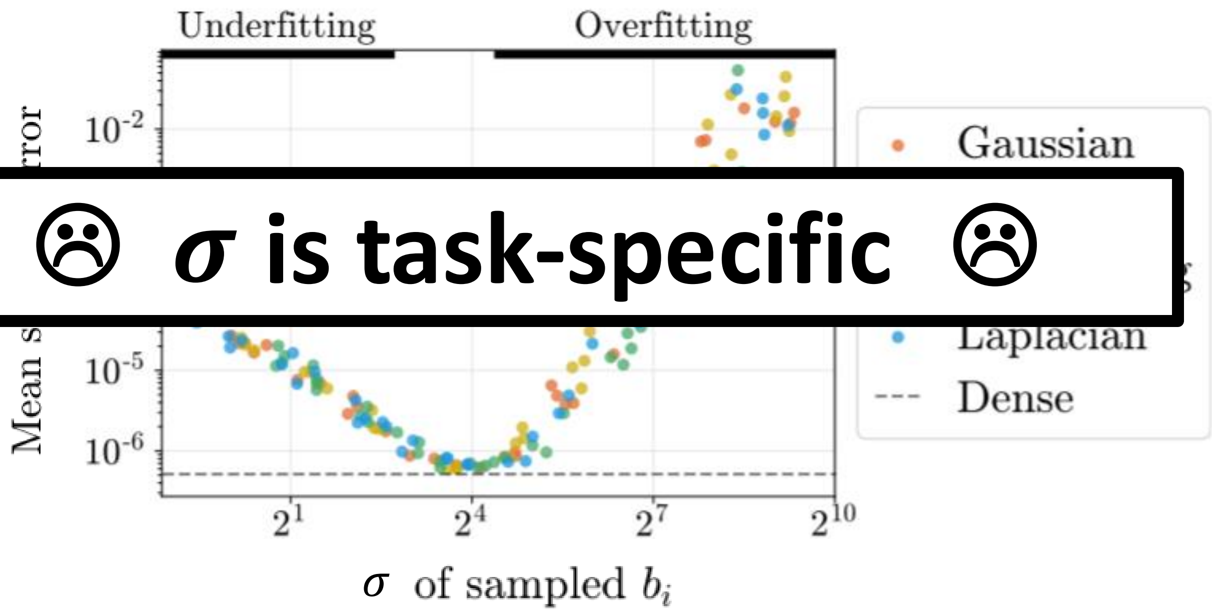
$\boldsymbol{N}$ is not

Data sampled from $1/f^{1.5}$

$$\gamma(\boldsymbol{v}) = [\cos(2\pi \boldsymbol{B}\boldsymbol{v}), \sin(2\pi \boldsymbol{B}\boldsymbol{v})], \qquad \boldsymbol{B} \sim N(0, \sigma^2), \qquad \boldsymbol{B} \in \mathbb{R}^{m \times d}$$

Distribution Types and Mapping Bandwidth

Gaussian RFF: 1D experiment

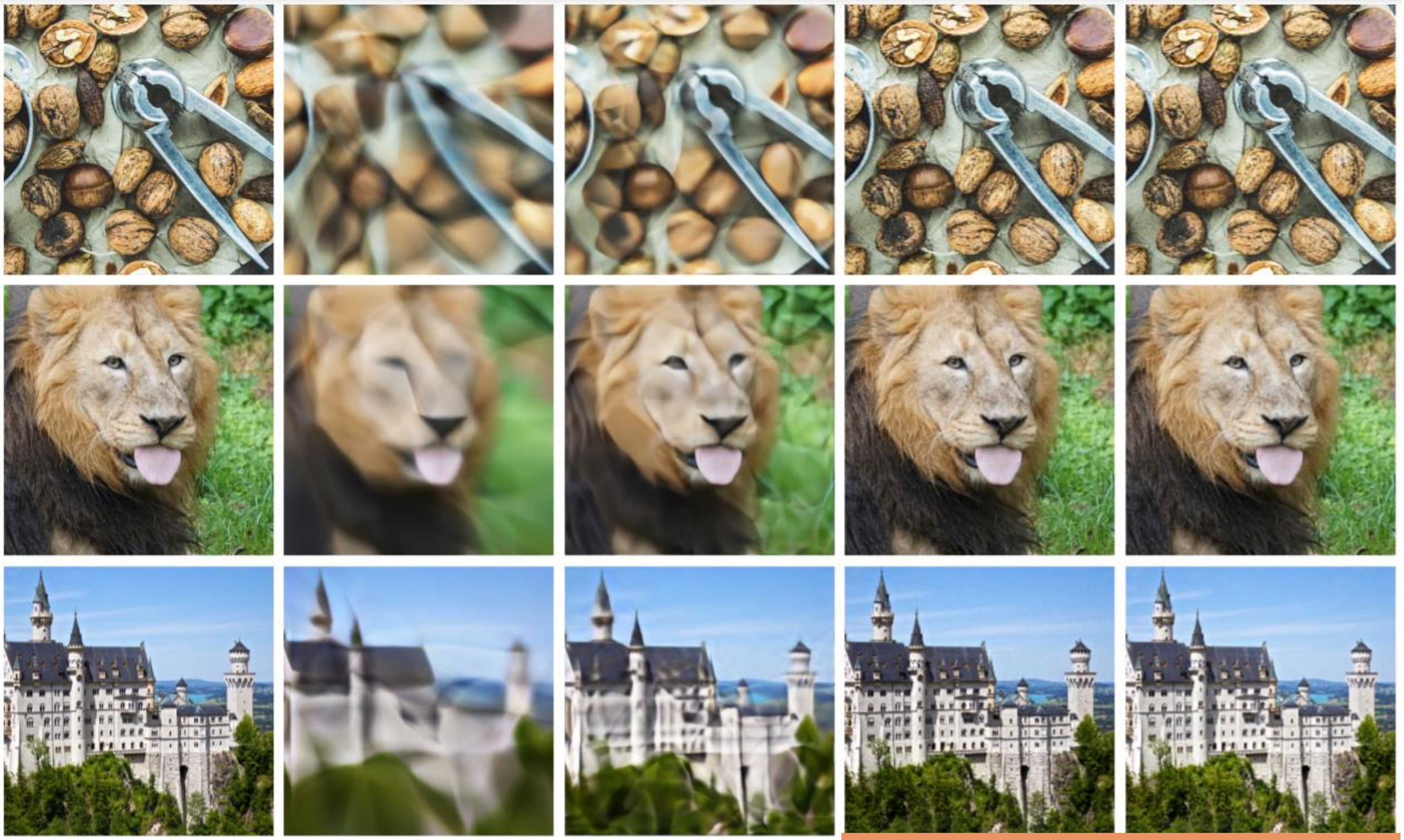


☹ $\boldsymbol{\sigma}$ is task-specific ☹

Data sampled from $1/f^{1.5}$

$$\gamma(\boldsymbol{v}) = [\cos(2\pi \boldsymbol{B}\boldsymbol{v}), \sin(2\pi \boldsymbol{B}\boldsymbol{v})], \qquad \boldsymbol{B} \sim N(0, \sigma^2), \qquad \boldsymbol{B} \in \mathbb{R}^{m \times d}$$

Which Mapping is Best Visually?

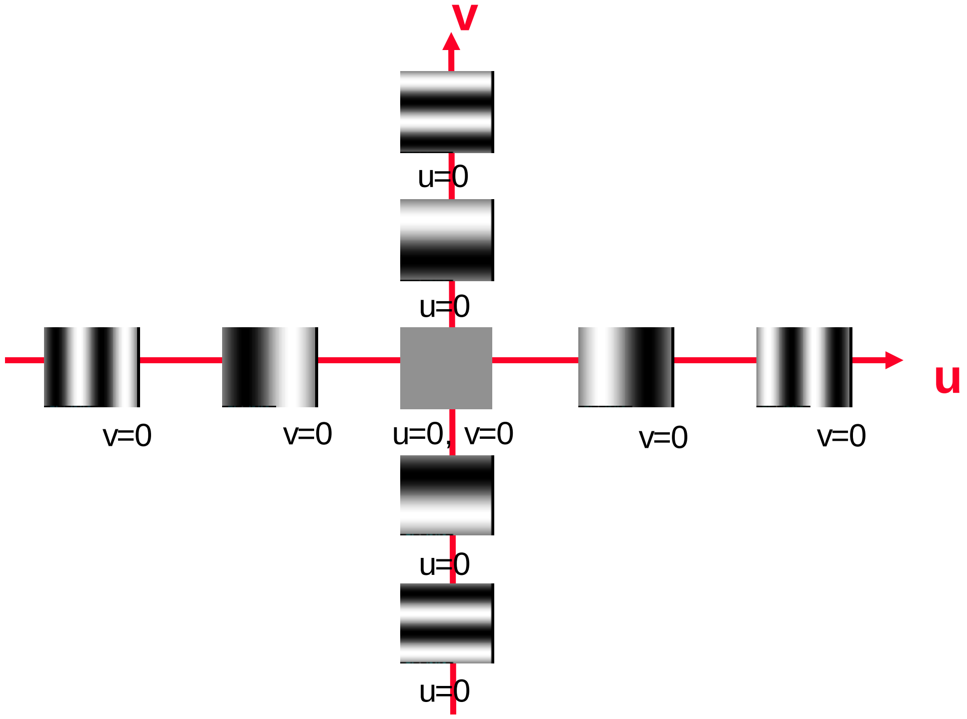


(a) Ground Truth (b) No mapping (c) Basic (d) Positional enc. (e) Gaussian

On/Off-Axis Frequencies

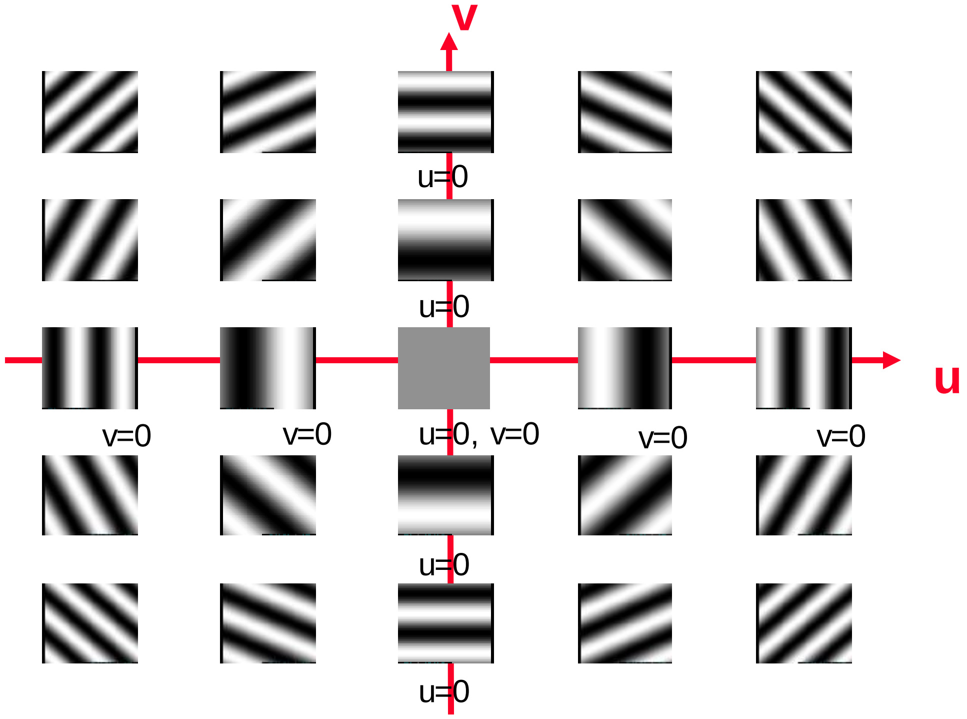
Positional Encoding:

$$\left(\sin\left(2\pi\sigma^{j\backslash\mathrm{m}}{\color{red}x}\right), \sin\left(2\pi\sigma^{j\backslash\mathrm{m}}{\color{red}y}\right)\right)$$



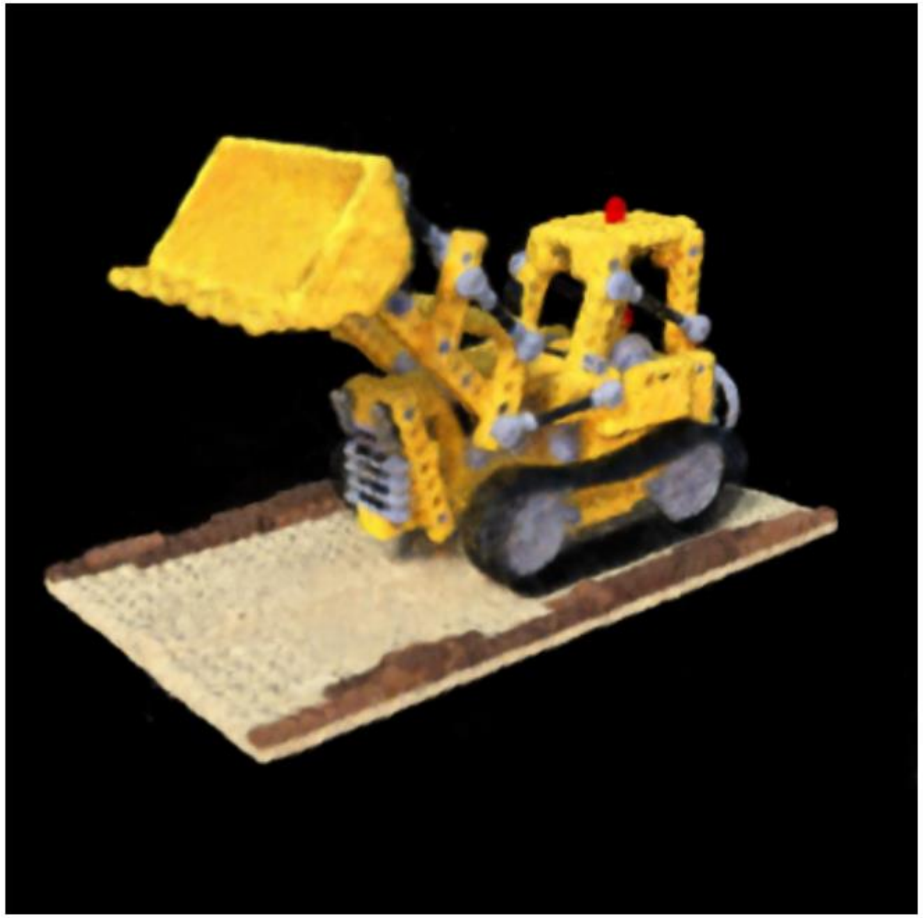
Gaussian: $\boldsymbol{B} \in \mathbb{R}^{m\times d}$

$$\sin\left(2\pi\left(b_{i1}{\color{red}x} + b_{i2}{\color{red}y}\right)\right)$$

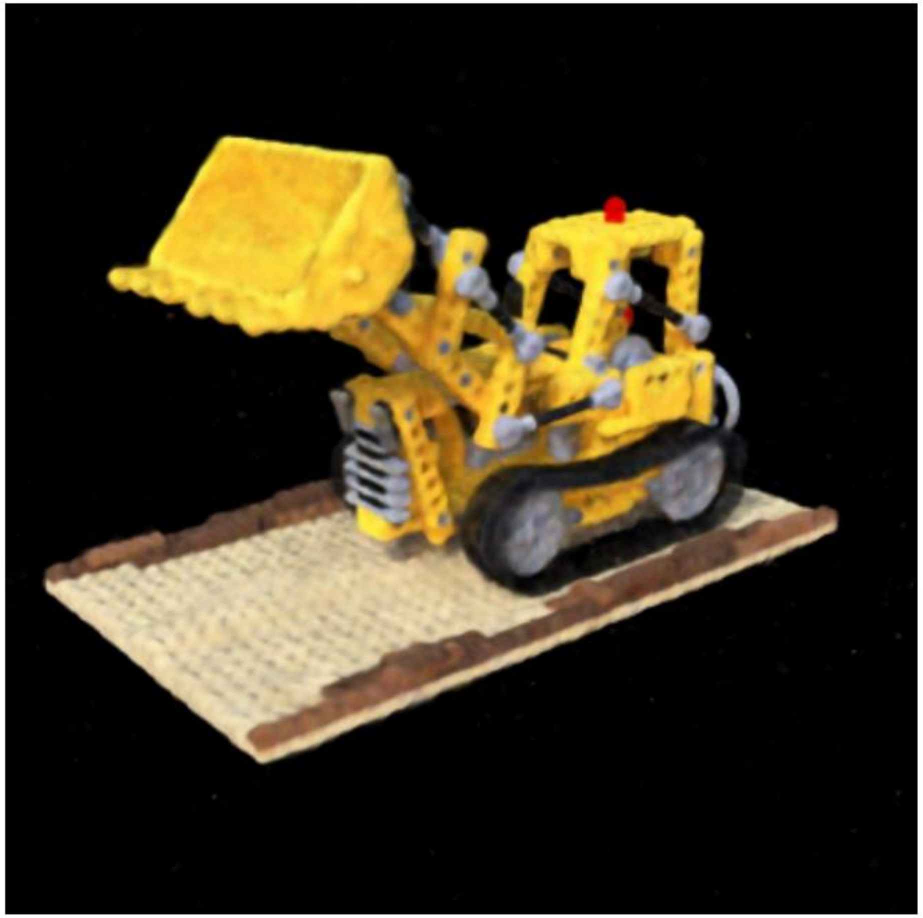


Images credit: Michal Irani, Intro to Comp. Vision

PE vs Gaussian Comparison

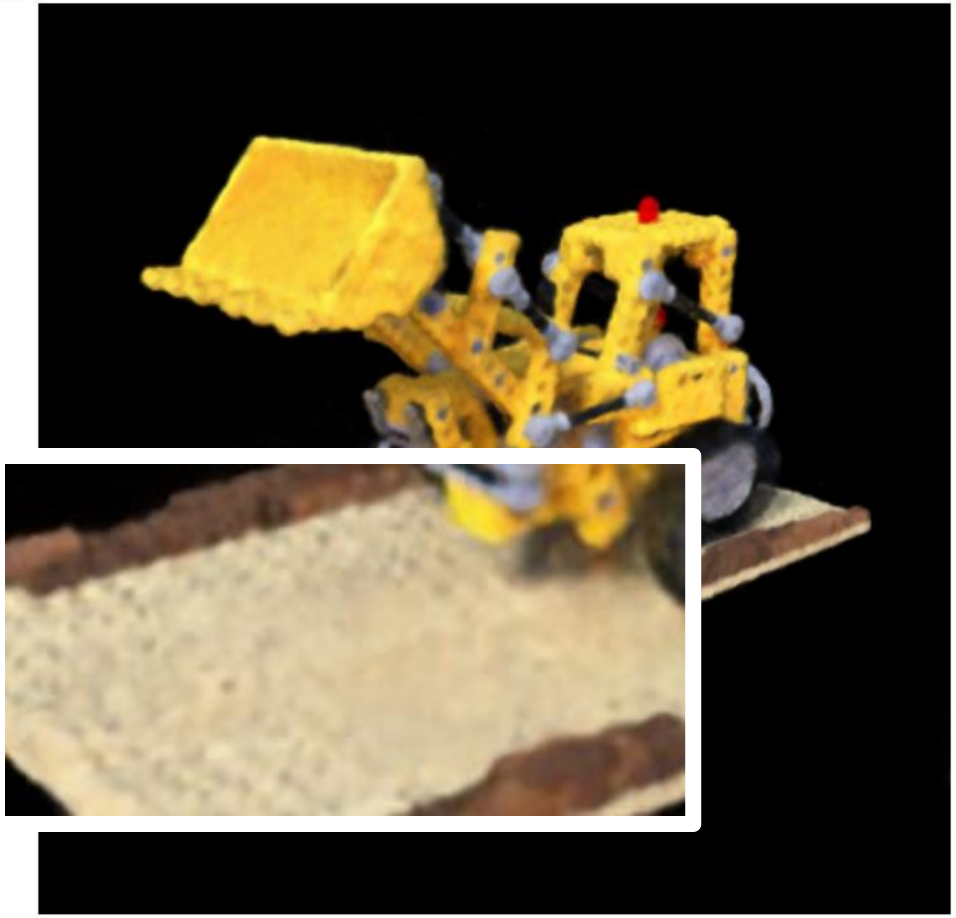


Positional Encoding

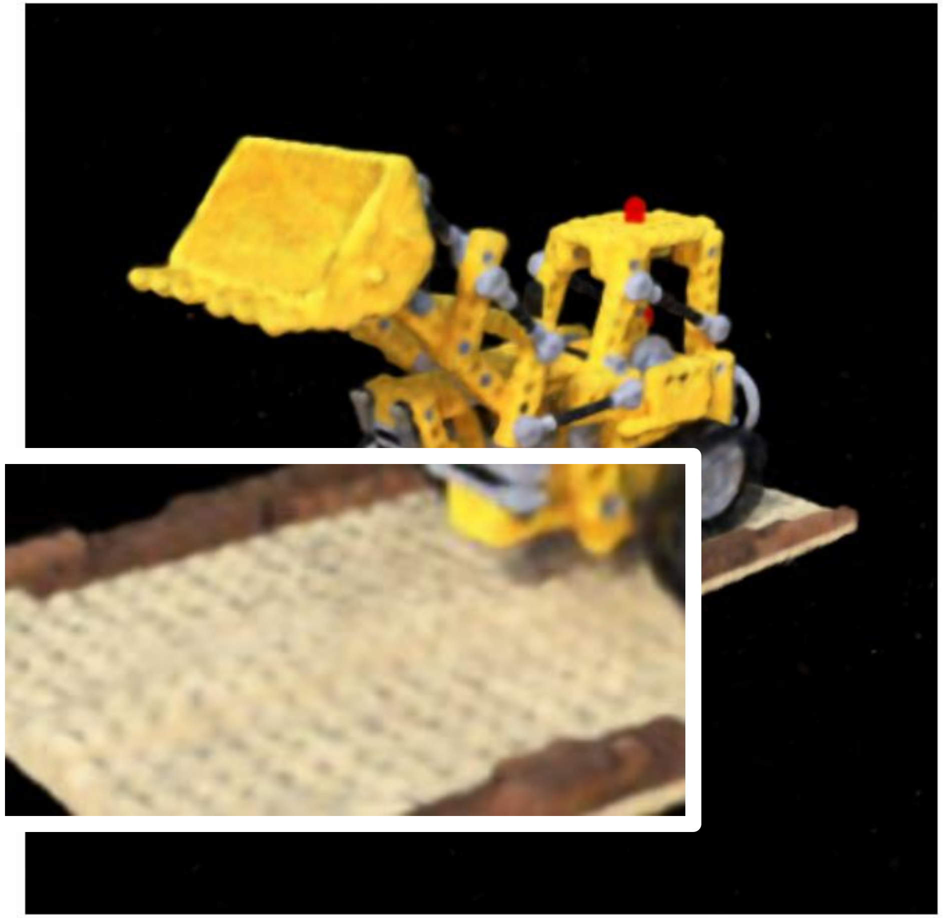


Gaussian

PE vs Gaussian Comparison

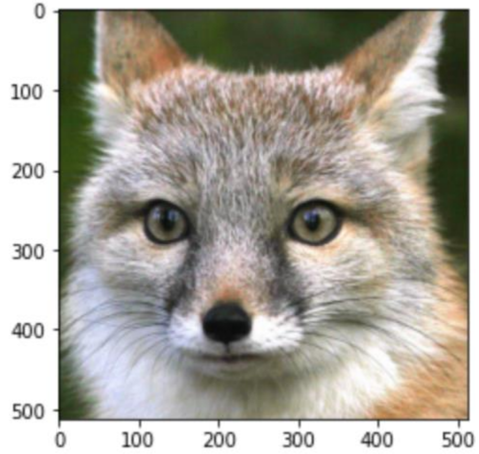


Positional Encoding



Gaussian

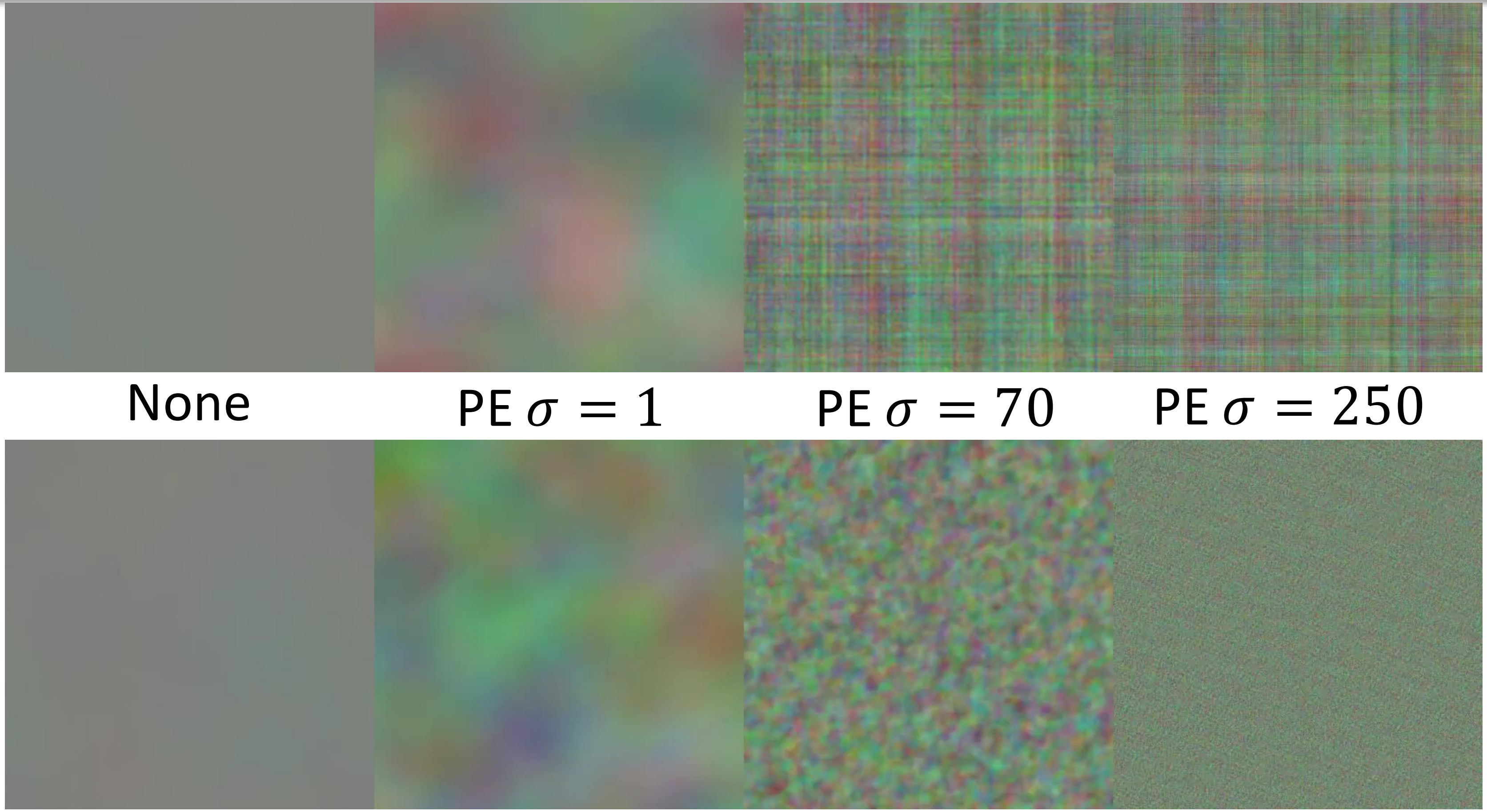
Try It Yourself!



GT

Underfitting

Overfitting

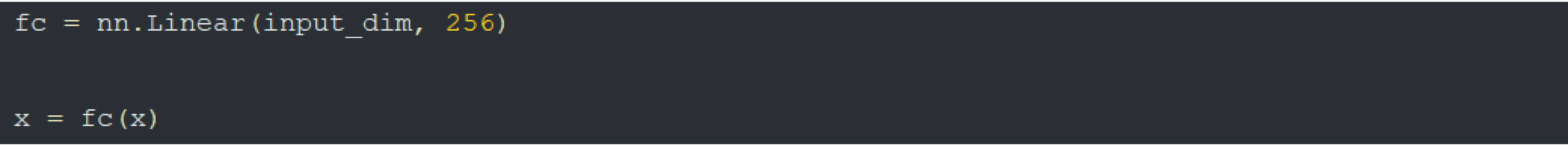


None | PE $\sigma = 1$ | PE $\sigma = 70$ | PE $\sigma = 250$

Basic | Gauss $\sigma = 1$ | Gauss $\sigma = 10$ | Gauss $\sigma = 100$

https://colab.research.google.com/github/tancik/fourier-feature-networks/blob/master/Demo.ipynb (extended)

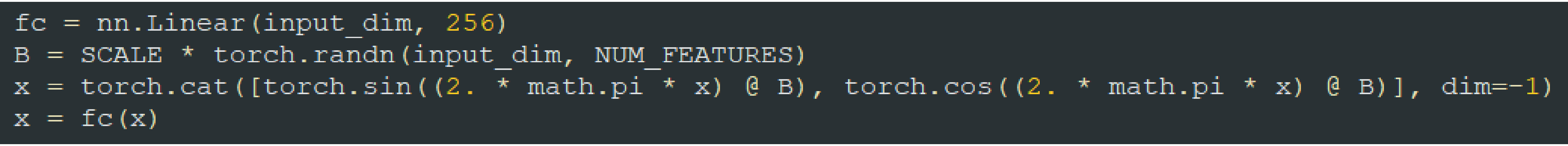
Add to Your Code!



```
fc = nn.Linear(input_dim, 256)


x = fc(x)
```

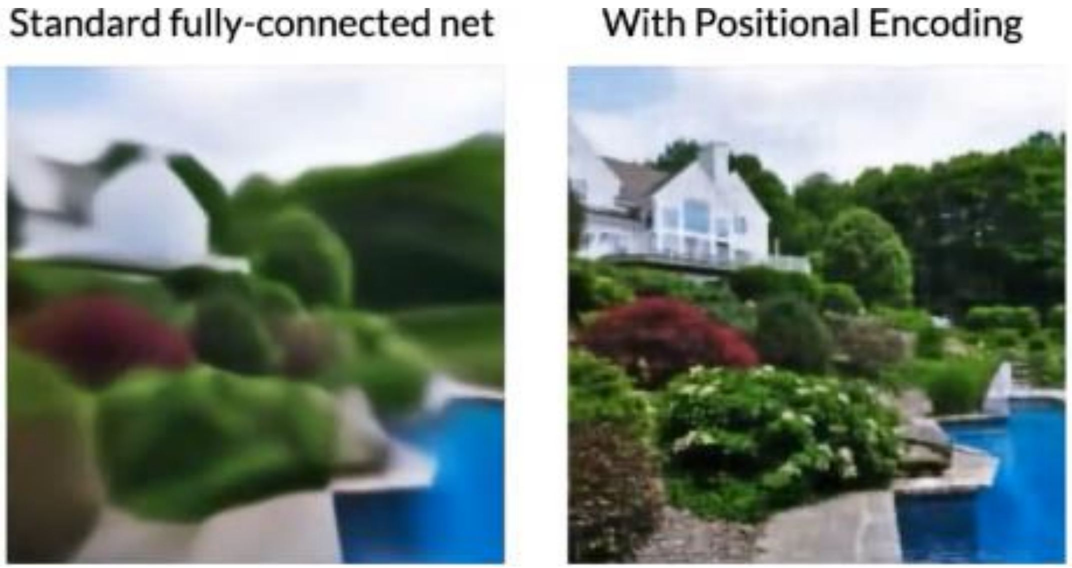
Add to Your Code!



```
fc = nn.Linear(input_dim, 256)
B = SCALE * torch.randn(input_dim, NUM_FEATURES)
x = torch.cat([torch.sin((2. * math.pi * x) @ B), torch.cos((2. * math.pi * x) @ B)], dim=-1)
x = fc(x)
```

Summary

Input mapping helps the network learn fine details / high frequencies!

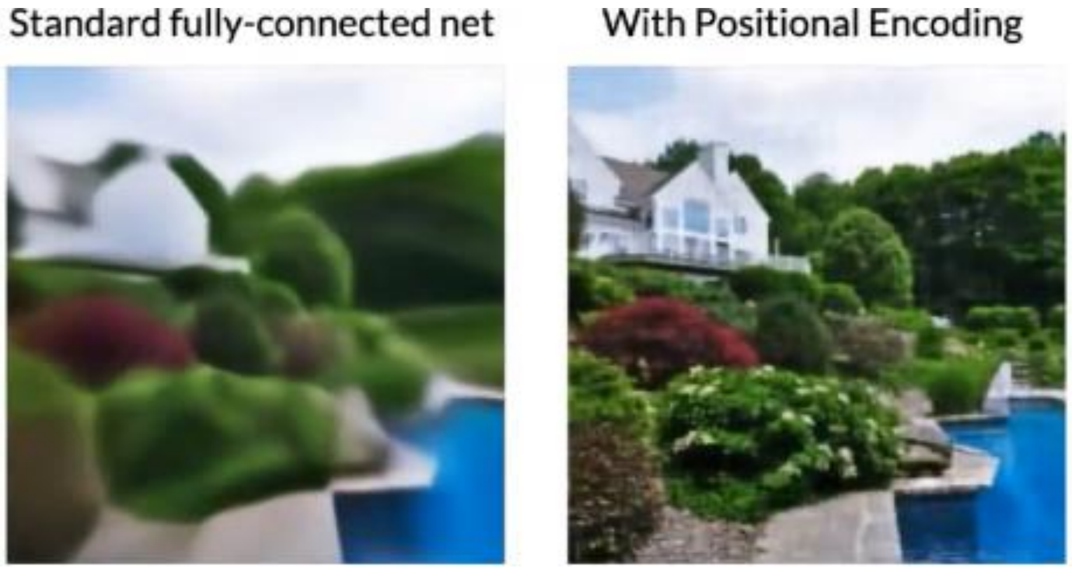


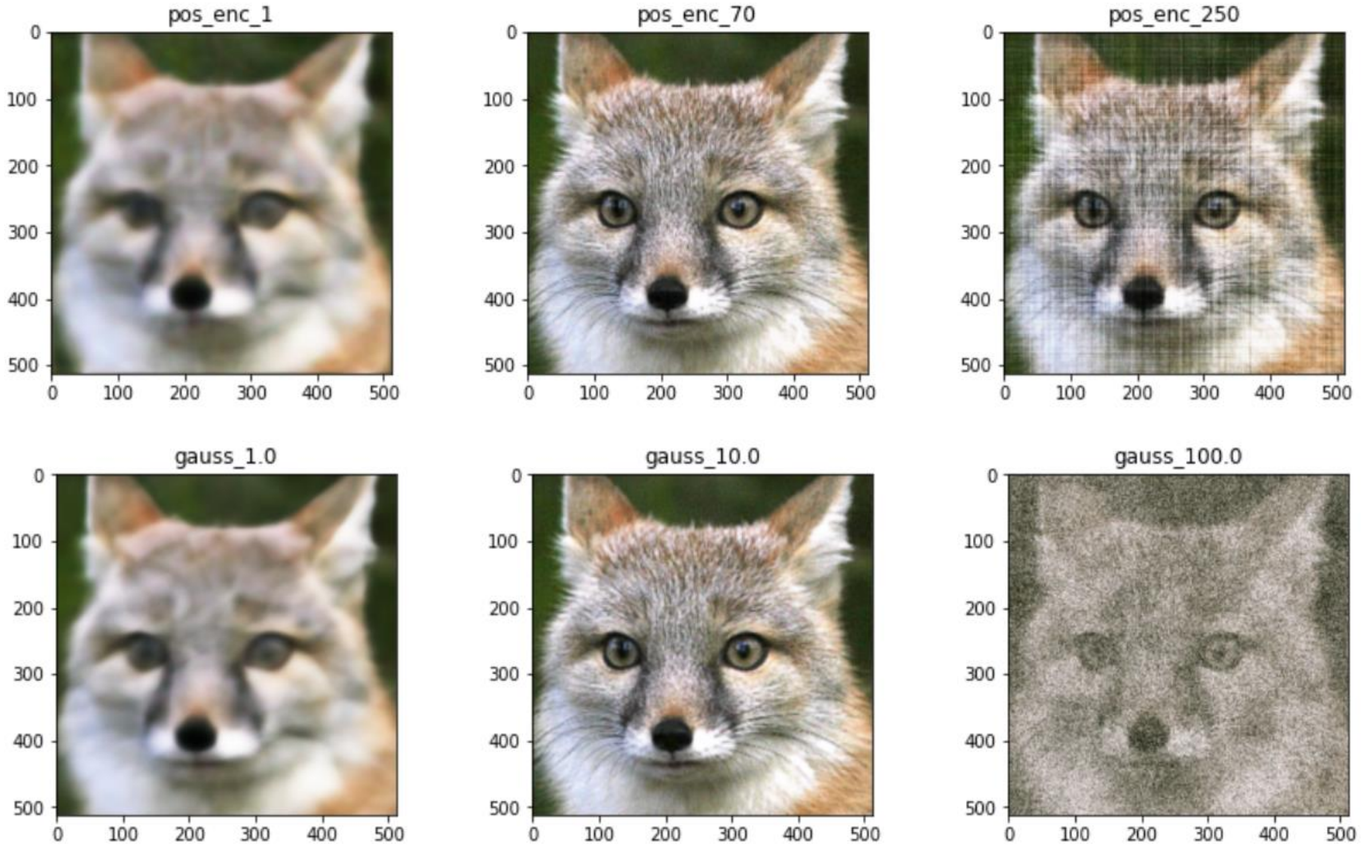
Standard fully-connected net

With Positional Encoding

Summary

Input mapping helps the network learn fine details / high frequencies!





Any Questions?

Welcome back

Implicit Neural Representations with Periodic Activation Functions

Vincent Sitzmann*, Julien N. P. Marte*, Alexander W. Bergman, David B. Lindell, Gordon Wetzstein

NeurIPS 2020

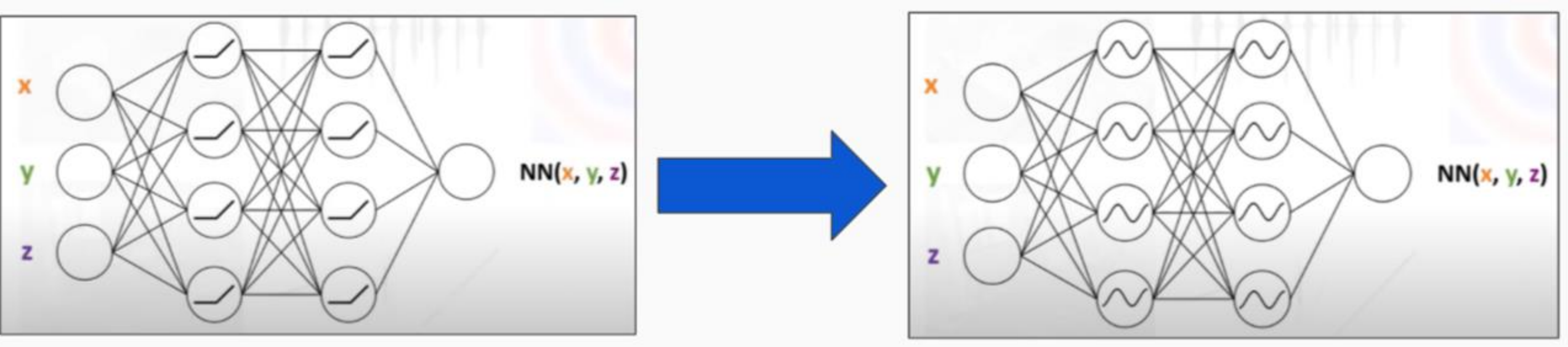
Implicit Neural Representations with Periodic Activation Functions, aka SIRENs - SInusoidal REpresentation Networks

Vincent Sitzmann*, Julien N. P. Marte*, Alexander W. Bergman, David B. Lindell, Gordon Wetzstein

NeurIPS 2020

SIRENs - SInusoidal REpresentation Networks

The gist: Neural Internal Representation with sinusoidal activation functions.



SIRENs - SInusoidal REpresentation Networks

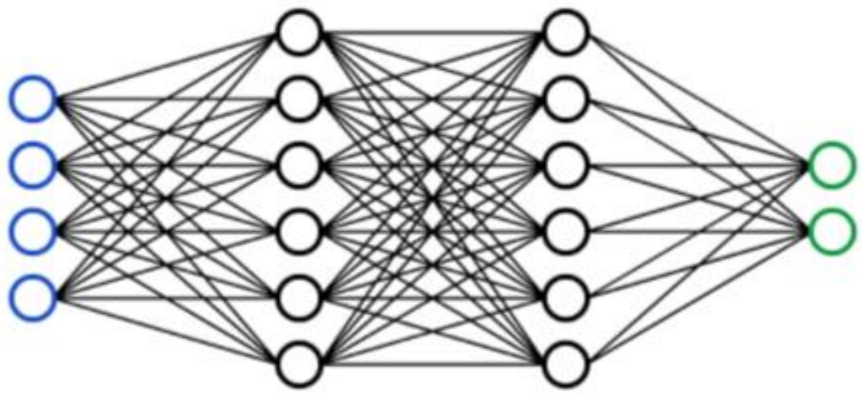
The gist: Neural Internal Representation with sinusoidal activation functions.

The interesting part: opens a door for new applications/implementations.

SIRENs - Motivation

Until now - the network is trained directly by the wanted function.

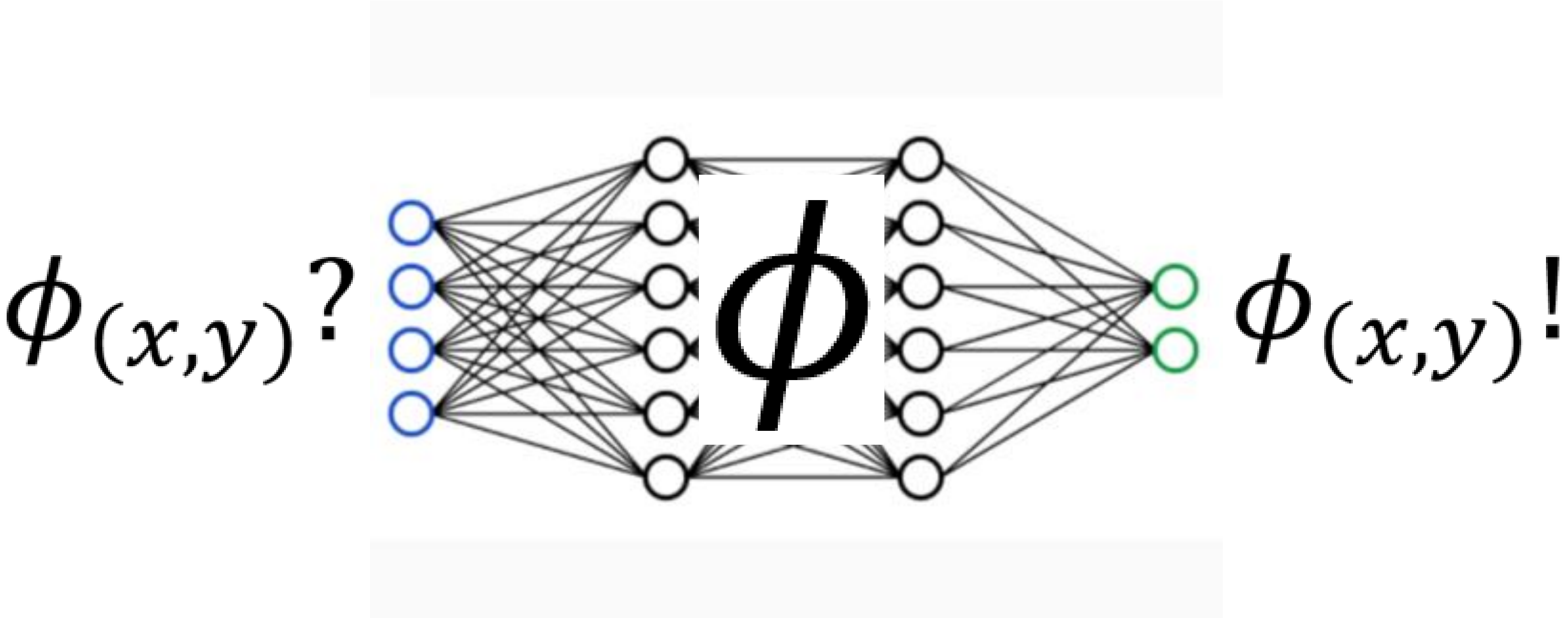
$\phi_{(x,y)}?$



$\phi_{(x,y)}!$

SIRENs - Motivation

Until now - the network is trained directly by the wanted function.



SIRENs - Motivation

Until now - the network is trained directly by the wanted function.

But for some tasks - the input’s **derivatives** are essential.

SIRENs - Motivation

Until now - the network is trained directly by the wanted function.

But for some tasks - the input’s **derivatives** are essential.

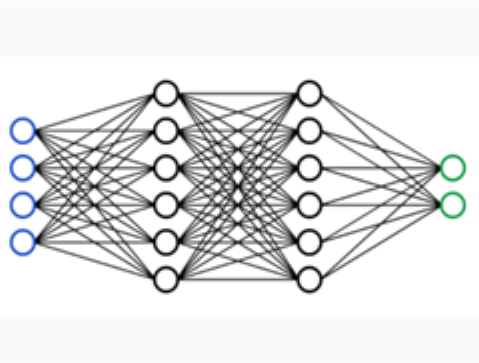
$$\mathcal{L}_{(\phi,\nabla\phi)} = \|\phi(x) - f(x)\|^2 + \|\nabla\phi(x) - \nabla f(x)\|^2$$

SIRENs - Motivation

Until now - the network is trained directly by the wanted function.

But for some tasks - the input's **derivatives** are essential.

$$\mathcal{L}_{(\phi,\nabla\phi)} = \|\phi(x) - f(x)\|^2 + \|\nabla\phi(x) - \nabla f(x)\|^2$$



$\nabla\phi_{(x,y)}?$ $\nabla\phi_{(x,y)}!$

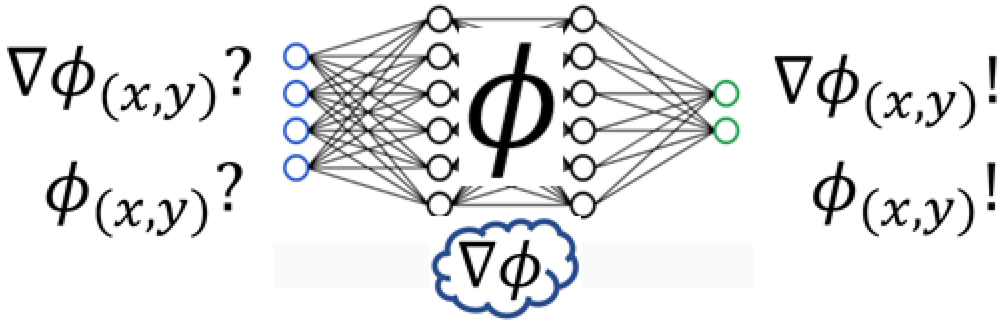
$\phi_{(x,y)}?$ $\phi_{(x,y)}!$

SIRENs - Motivation

Until now - the network is trained directly by the wanted function.

But for some tasks - the input’s **derivatives** are essential.

$$\mathcal{L}_{(\phi,\nabla\phi)} = \|\phi(x) - f(x)\|^2 + \|\nabla\phi(x) - \nabla f(x)\|^2$$



SIRENs - Motivation

Until now - the network is trained directly by the wanted function.

But for some tasks - the input’s **derivatives** are essential.

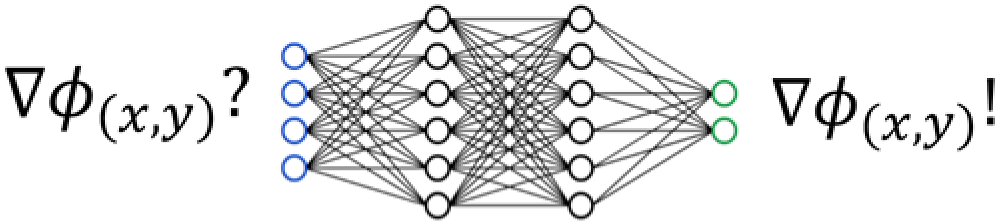
$$\boxed{\frac{\partial^2\Phi}{\partial t^2} - c^2\Delta\Phi = 0}$$

SIRENs - Motivation

Until now - the network is trained directly by the wanted function.

But for some tasks - the input’s **derivatives** are essential.

$$\boxed{\frac{\partial^2 \Phi}{\partial t^2} - c^2 \Delta \Phi = 0}$$

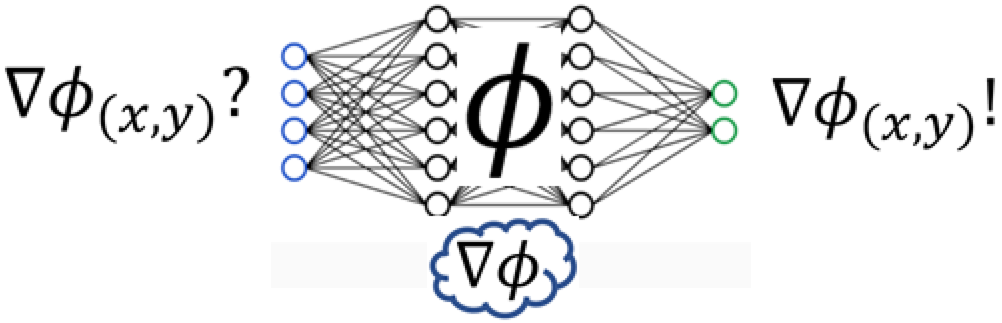


SIRENs - Motivation

Until now - the network is trained directly by the wanted function.

But for some tasks - the input’s **derivatives** are essential.

$$\boxed{\frac{\partial^2 \Phi}{\partial t^2} - c^2 \Delta \Phi = 0}$$



SIRENs - Motivation

Until now - the network is trained directly by the wanted function.

But for some tasks - the input’s **derivatives** are essential.

Are they also represented well?

SIRENs - Motivation

Until now - the network is trained directly by the wanted function.

But for some tasks - the input’s **derivatives** are essential.

Are they also represented well?

Obviously not.. So SIRENs will help!

SIRENs - Why do they work?

SIRENs - Why do they work?

The derivative of a SIREN is also a SIREN!

$$\frac{\partial}{\partial x}\sin(x) = \cos(x) = \sin\left(x + \frac{\pi}{2}\right)$$

SIRENs - Why do they work?

The derivative of a SIREN is also a SIREN!

$$\frac{\partial}{\partial x}\sin(x) = \cos(x) = \sin\left(x + \frac{\pi}{2}\right)$$

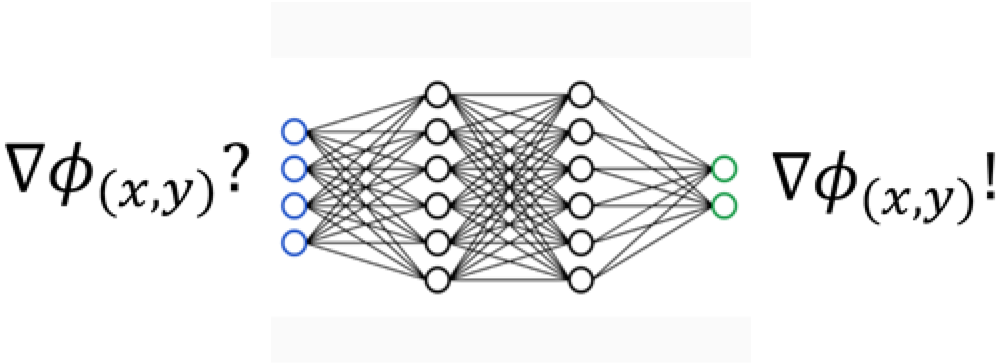
Enables supervising complicated signals.

SIRENs - Why do they work?

The derivative of a SIREN is also a SIREN!

$$\frac{\partial}{\partial x}\sin(x) = \cos(x) = \sin\left(x + \frac{\pi}{2}\right)$$

Enables supervising complicated signals.

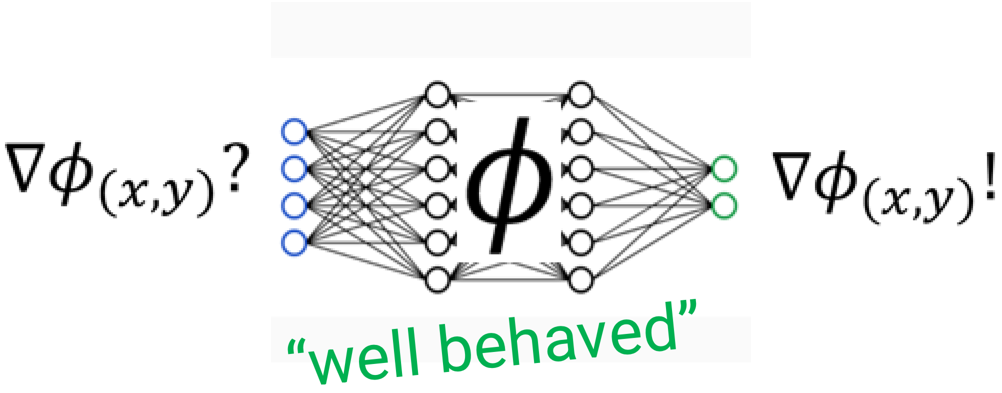


SIRENs - Why do they work?

The derivative of a SIREN is also a SIREN!

$$\frac{\partial}{\partial x}\sin(x) = \cos(x) = \sin\left(x + \frac{\pi}{2}\right)$$

Enables supervising complicated signals.



SIRENs - Initialization is crucial

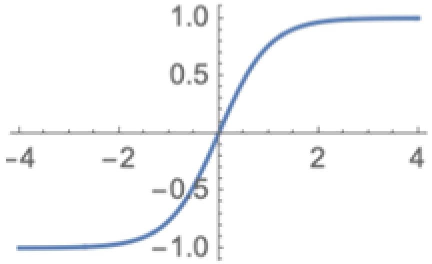
SIRENs - Initialization is crucial

Sinusoidal functions are not intuitively good activation functions

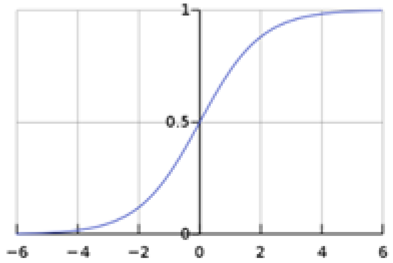
SIRENs - Initialization is crucial

Sinusoidal functions are not intuitively good activation functions

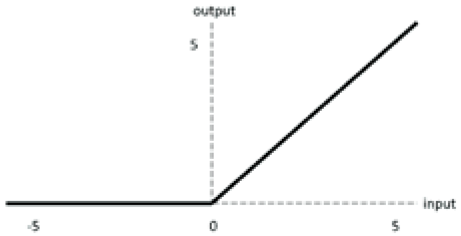
tanh



sigmoid



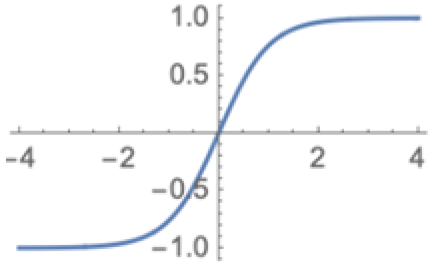
ReLU



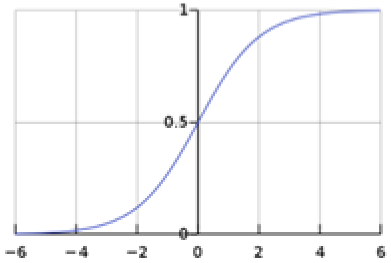
SIRENs - Initialization is crucial

Sinusoidal functions are not intuitively good activation functions

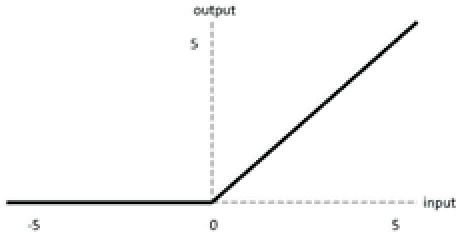
tanh



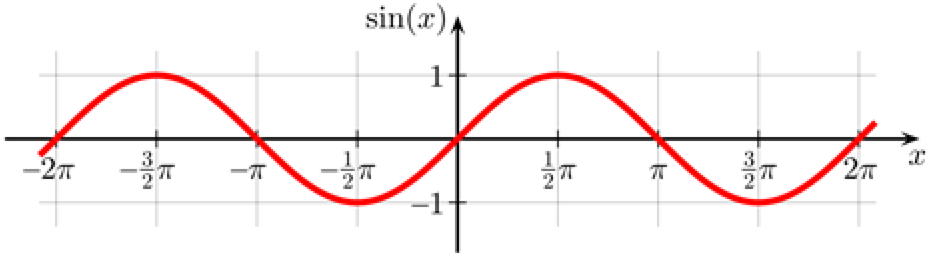
sigmoid



ReLU



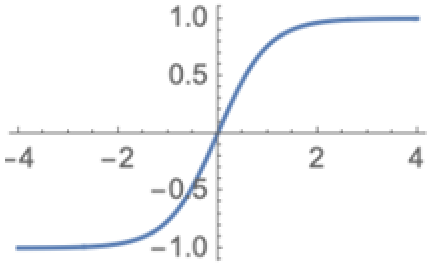
sin(x)



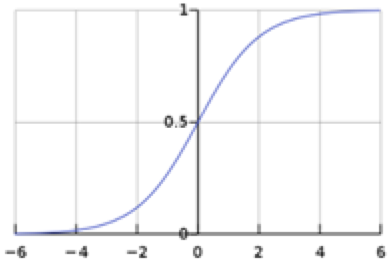
SIRENs - Initialization is crucial

Sinusoidal functions are not intuitively good activation functions

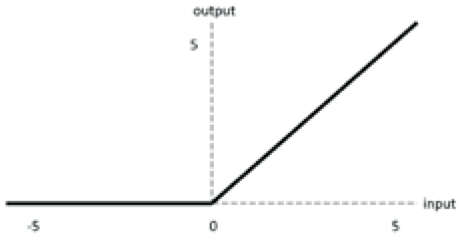
tanh



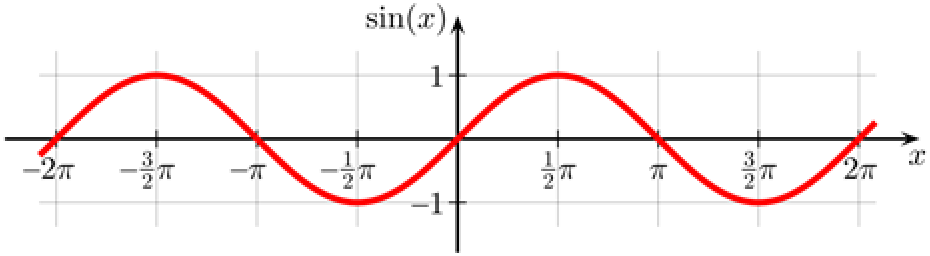
sigmoid



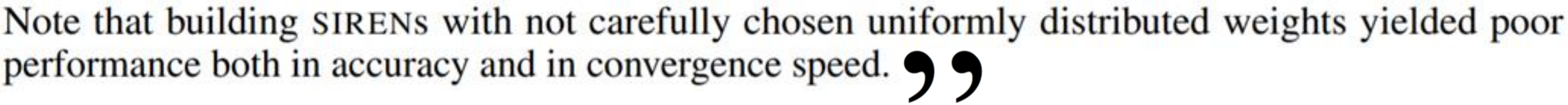
ReLU



sin(x)



To “behave well” and enable deep MLPs, initialization is crucial:



> “Note that building SIRENs with not carefully chosen uniformly distributed weights yielded poor performance both in accuracy and in convergence speed.”

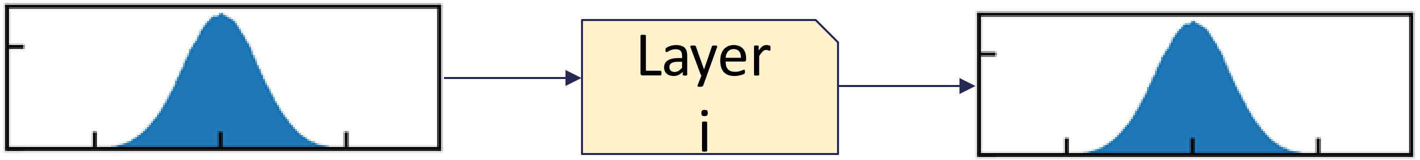
SIRENs - Initialization is crucial

Initialization scheme + explanation

SIRENs - Initialization is crucial

Many lemmas, bottom line:

Initializing all weights (except first layer) by uniform distribution in: $\left[-\sqrt{\frac{6}{fan\ in}}, \sqrt{\frac{6}{fan\ in}}\right]$

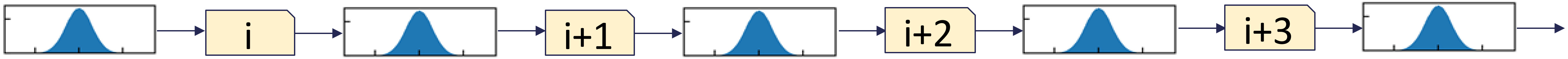


Layer i

SIRENs - Initialization is crucial

Many lemmas, bottom line:

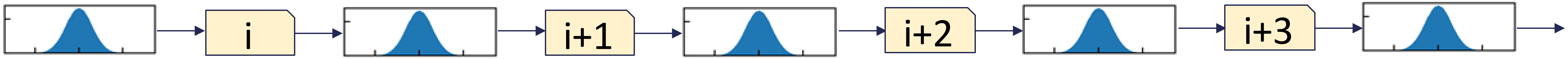
Initializing all weights (except first layer) by uniform distribution in: $\left[-\sqrt{\frac{6}{fan\ in}}, \sqrt{\frac{6}{fan\ in}}\right]$



SIRENs - Initialization is crucial

Many lemmas, bottom line:

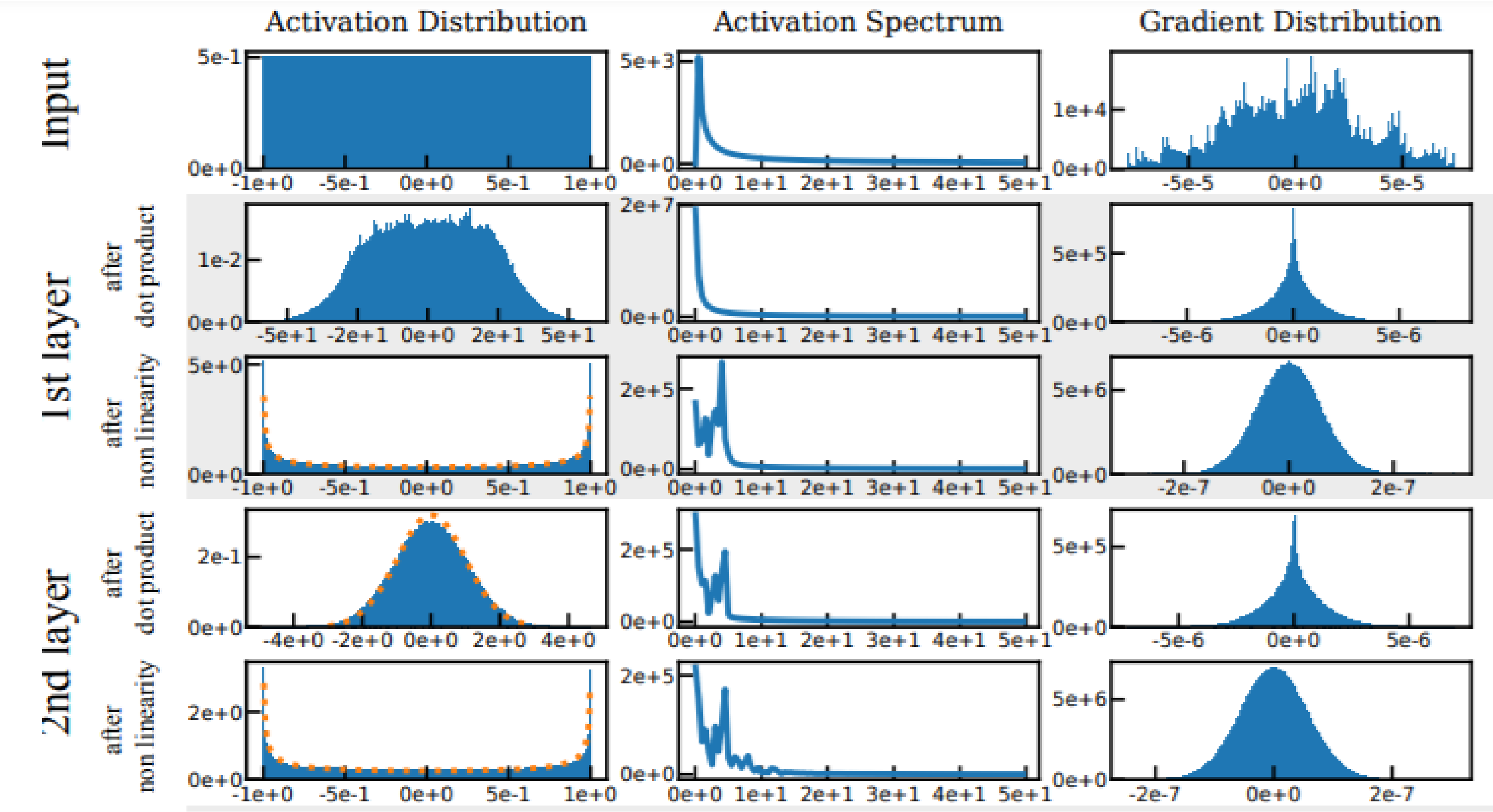
Initializing all weights (except first layer) by uniform distribution in: $\left[-\sqrt{\frac{6}{fan\ in}}, \sqrt{\frac{6}{fan\ in}}\right]$



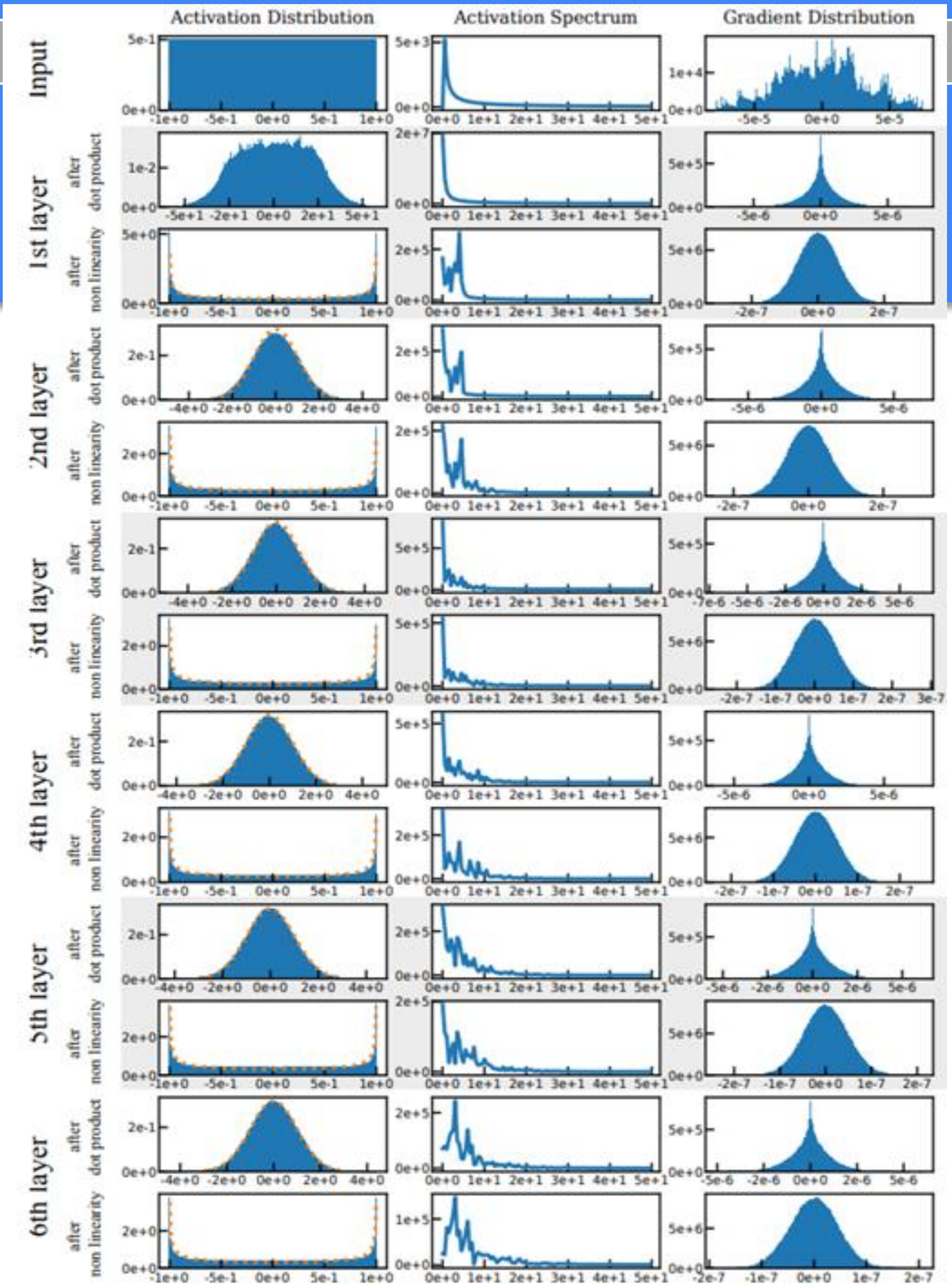
They claim (“beyond the scope of this paper”) - with this initialization -

“the frequency throughout the sine network grows only slowly”

SIRENs - Initialization is crucial



SIRENs - Initialization is crucial



SIRENs - Results

SIRENs - Results

Directly on signal

- Images, Videos, Audio

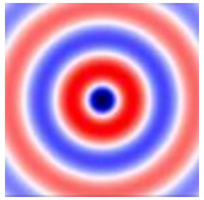
SIRENs - Results

Directly on signal

- Images, Videos, Audio

Only on derivatives

- Poisson (I)
- Helmholtz (I and II)



SIRENs - Results

Signal + derivatives

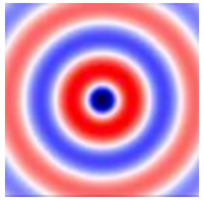
- SDF

Directly on signal

- Images, Videos, Audio

Only on derivatives

- Poisson (I)
- Helmholtz (I and II)



SIRENs - Results

Signal + derivatives

- SDF

Directly on signal

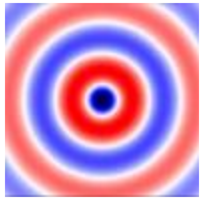
- Images, Videos, Audio

Spatial & temporal derivatives

- The Wave eq.

Only on derivatives

- Poisson (I)
- Helmholtz (I and II)



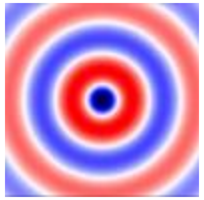
SIRENs - Results

Directly on signal

- Images, Videos, Audio

Only on derivatives

- Poisson (I)
- Helmholtz (I and II)



Signal + derivatives

- SDF

Spatial & temporal derivatives

- The Wave eq.

Can learn priors

- Inpainting: encoder→SIREN’s params

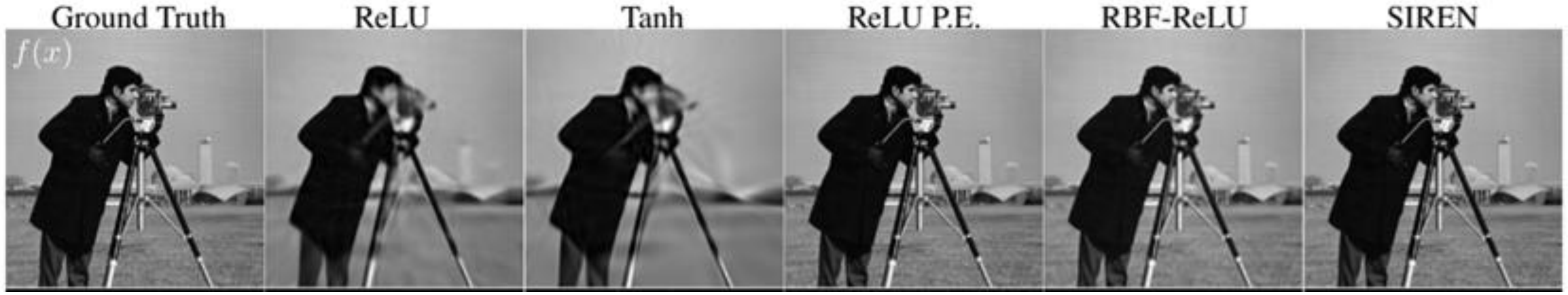
$$\tilde{\mathcal{L}} = \sum_i \|\Phi(\mathbf{x}_i) - f(\mathbf{x}_i)\|^2$$

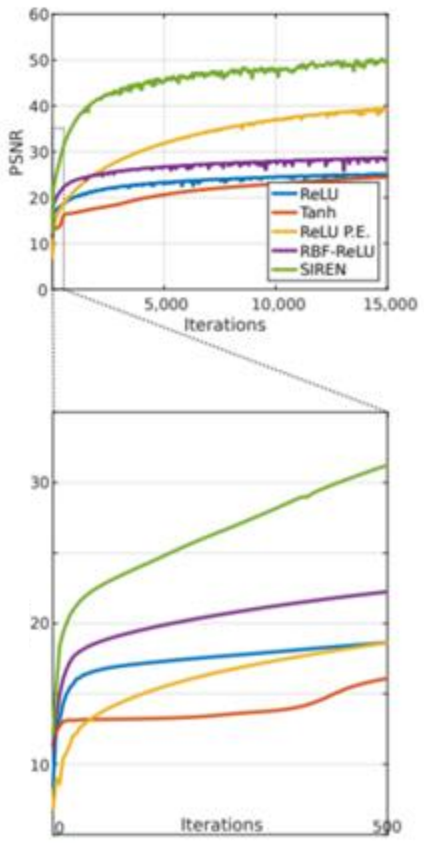
SIRENs - Directly on signal

$$\tilde{\mathcal{L}} = \sum_i \|\Phi(\mathbf{x}_i) - f(\mathbf{x}_i)\|^2$$

SIRENs - Directly on signal

Images

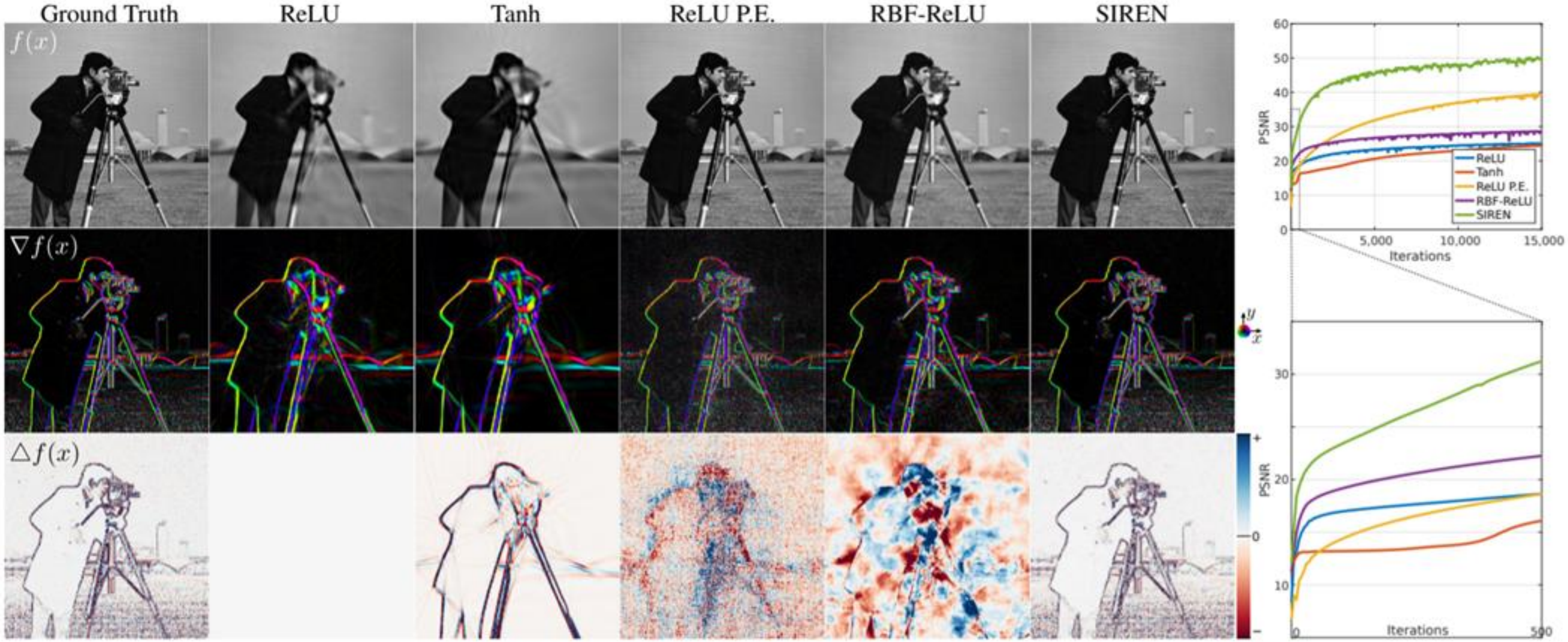




SIRENs - Directly on signal

$$\tilde{\mathcal{L}} = \sum_i \|\Phi(\mathbf{x}_i) - f(\mathbf{x}_i)\|^2$$

Images

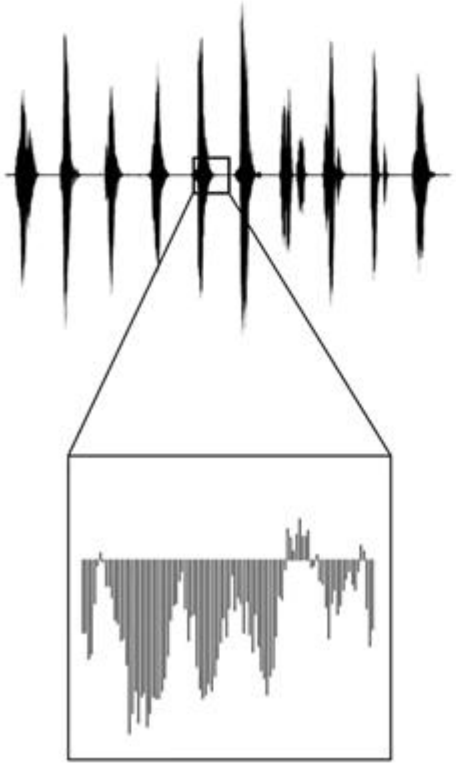


$$\tilde{\mathcal{L}} = \sum_i \|\Phi(\mathbf{x}_i) - f(\mathbf{x}_i)\|^2$$

SIRENs - Directly on signal

The data

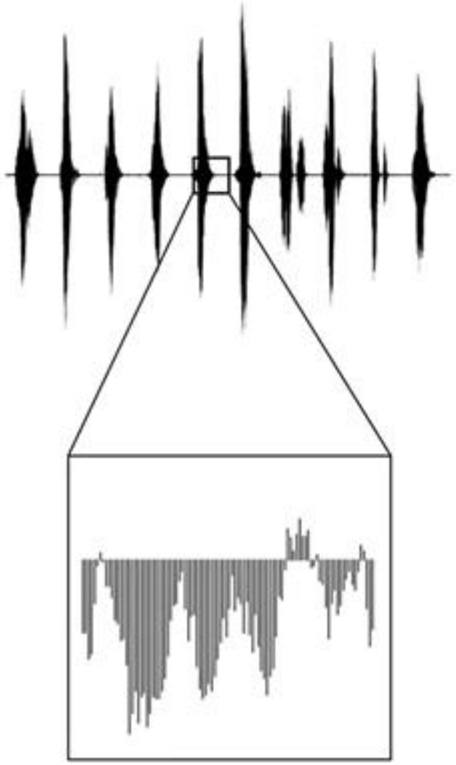
Audio



$$\tilde{\mathcal{L}} = \sum_i \|\Phi(\mathbf{x}_i) - f(\mathbf{x}_i)\|^2$$

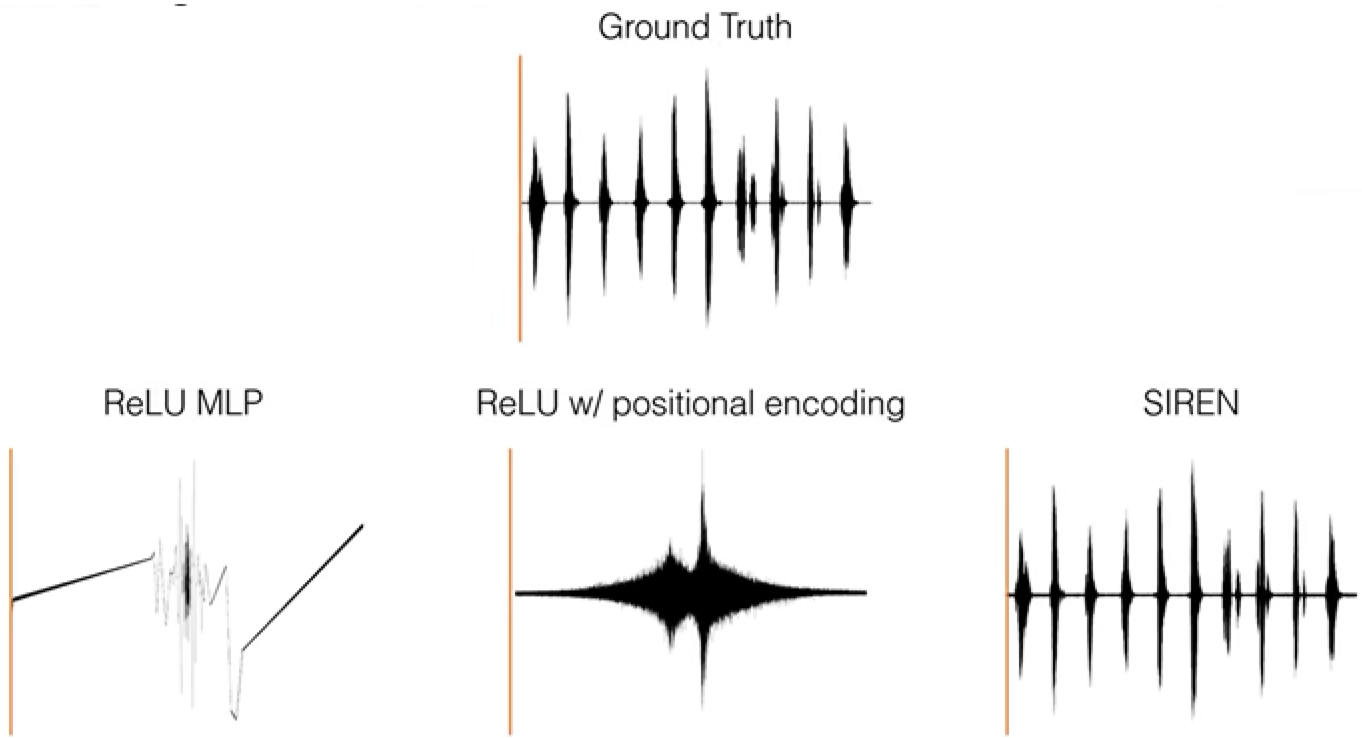
SIRENs - Directly on signal

The data



Audio

Results

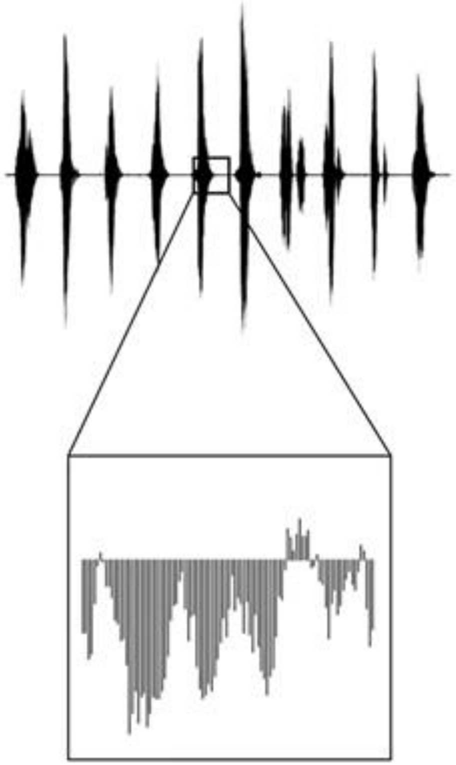


$$\tilde{\mathcal{L}} = \sum_i \|\Phi(\mathbf{x}_i) - f(\mathbf{x}_i)\|^2$$

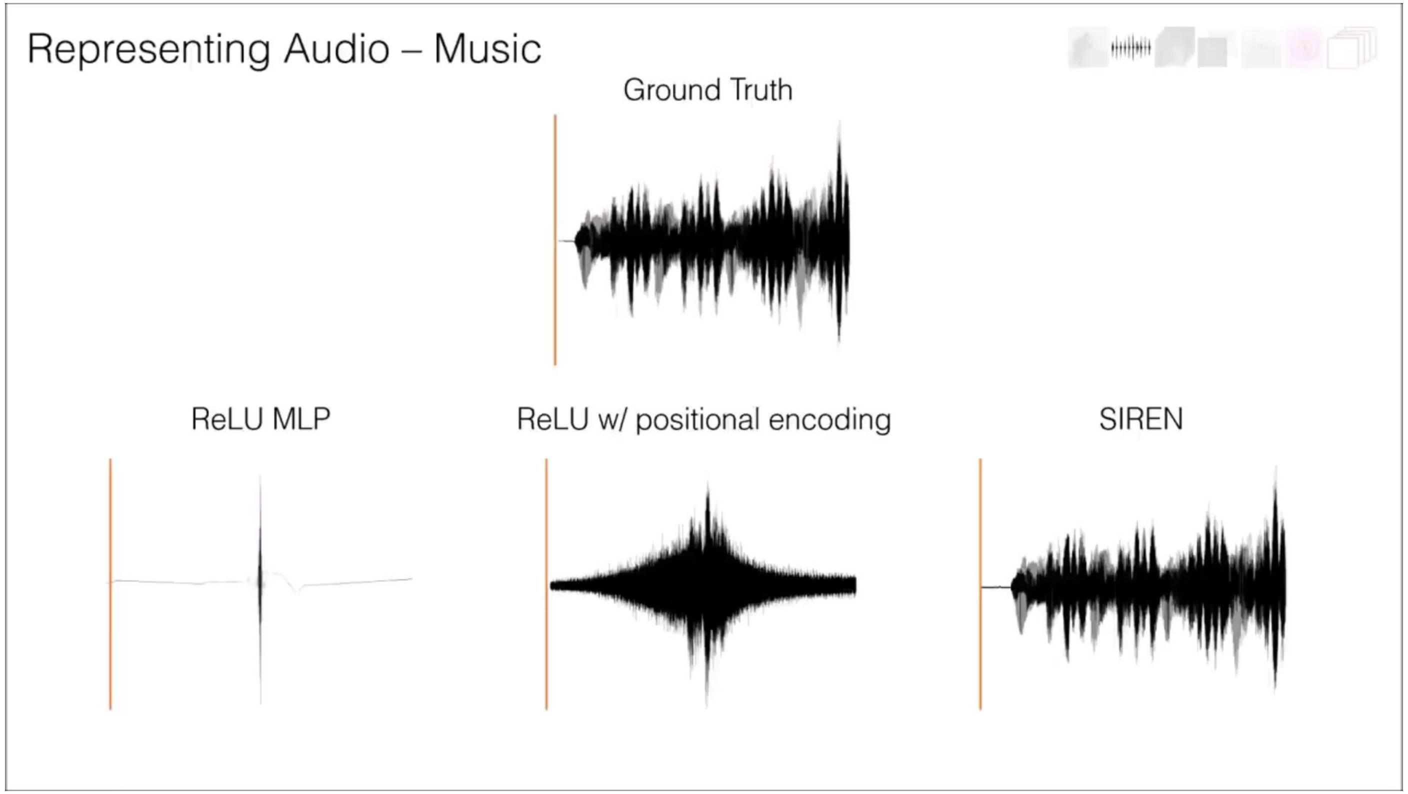
SIRENs - Directly on signal

The data

Audio



Results

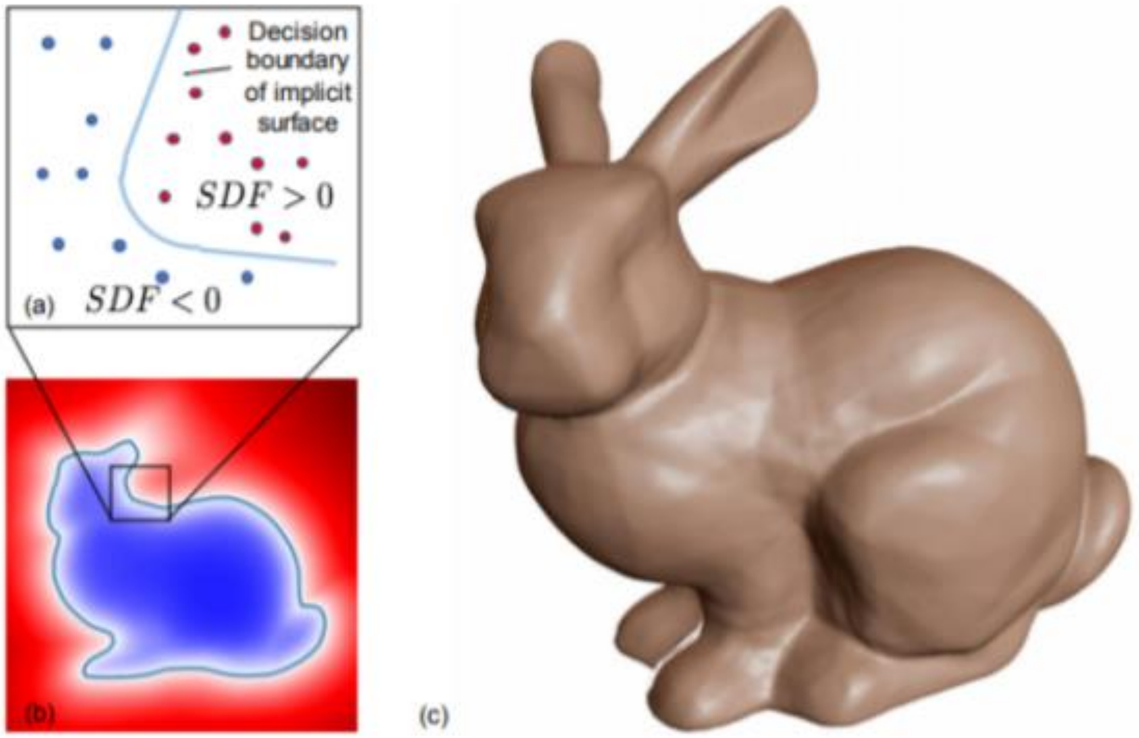


Representing Audio – Music

SIRENs - Signal + derivatives

SIRENs - Signal + derivatives

Signed Distance Function (S



SIRENs - Signal + derivatives

Signed Distance Function (SDF):

$$\mathcal{L}_{\mathrm{sdf}} = \int_{\Omega} \big\| \, |\nabla_{\mathbf{x}} \Phi(\mathbf{x})| - 1 \big\| d\mathbf{x} + \int_{\Omega_0} \|\Phi(\mathbf{x})\| + \big(1 - \langle \nabla_{\mathbf{x}} \Phi(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle\big) d\mathbf{x} + \int_{\Omega \setminus \Omega_0} \psi\big(\Phi(\mathbf{x})\big) d\mathbf{x}$$

SIRENs - Signal + derivatives

Signed Distance Function (SDF):

Everywhere **On border** **Not on border**

$$\mathcal{L}_{\mathrm{sdf}} = \int_{\Omega} \big\| \left| \nabla_{\mathbf{x}} \Phi(\mathbf{x}) \right| - 1 \big\| d\mathbf{x} + \int_{\Omega_0} \| \Phi(\mathbf{x}) \| + \big(1 - \langle \nabla_{\mathbf{x}} \Phi(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle \big) d\mathbf{x} + \int_{\Omega \setminus \Omega_0} \psi \big(\Phi(\mathbf{x}) \big) d\mathbf{x}$$

SIRENs - Signal + derivatives

Signed Distance Function (SDF):

$$\mathcal{L}_{\mathrm{sdf}} = \overset{\textbf{Everywhere}}{\int_{\Omega}} \big\| \, |\nabla_{\mathbf{x}} \Phi(\mathbf{x})| - 1 \big\| d\mathbf{x} + \overset{\textbf{On border}}{\int_{\Omega_0}} \underbrace{\|\Phi(\mathbf{x})\|}_{\text{SDF}\rightarrow 0} + \big(1 - \langle \nabla_{\mathbf{x}} \Phi(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle \big) d\mathbf{x} + \overset{\textbf{Not on border}}{\int_{\Omega \setminus \Omega_0}} \underbrace{\psi(\Phi(\mathbf{x}))}_{\text{Penalty on small SDF}} d\mathbf{x}$$

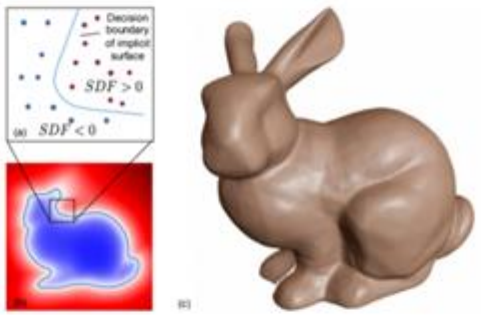
$$\psi(\mathbf{x}) = \exp(-\alpha \cdot |\Phi(\mathbf{x})|)$$

$$\alpha \gg 1$$

SIRENs - Signal + derivatives

Signed Distance Function (SDF):

$$\mathcal{L}_{\text{sdf}} = \overset{\textbf{Everywhere}}{\int_{\Omega}} \underbrace{\big\|\, |\nabla_{\mathbf{x}}\Phi(\mathbf{x})| - 1 \big\|}_{|\text{grad}|\to 1} d\mathbf{x} + \overset{\textbf{On border}}{\int_{\Omega_0}} \underbrace{\|\Phi(\mathbf{x})\|}_{\text{SDF}\to 0} + \big(1 - \langle \nabla_{\mathbf{x}}\Phi(\mathbf{x}), \mathbf{n}(\mathbf{x})\rangle\big)\, d\mathbf{x} + \overset{\textbf{Not on border}}{\int_{\Omega\setminus\Omega_0}} \underbrace{\psi(\Phi(\mathbf{x}))}_{\text{Penalty on small SDF}} d\mathbf{x}$$



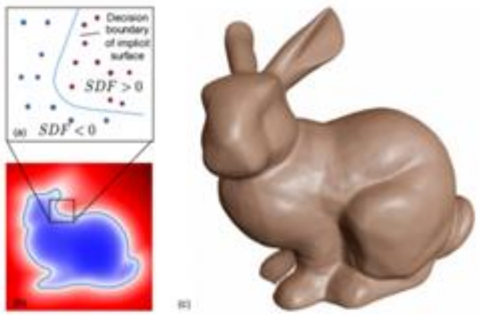
$$\psi(\mathbf{x}) = \exp(-\alpha \cdot |\Phi(\mathbf{x})|)$$

$$\alpha \gg 1$$

SIRENs - Signal + derivatives

Signed Distance Function (SDF):

$$\mathcal{L}_{\mathrm{sdf}} = \overset{\textbf{Everywhere}}{\int_{\Omega}} \underbrace{\big\| \left|\nabla_{\mathbf{x}}\Phi(\mathbf{x})\right| - 1 \big\|}_{|\text{grad}|\to 1} d\mathbf{x} + \overset{\textbf{On border}}{\int_{\Omega_0}} \underbrace{\|\Phi(\mathbf{x})\|}_{\text{SDF}\to 0} + \underbrace{\left(1 - \langle \nabla_{\mathbf{x}}\Phi(\mathbf{x}), \mathbf{n}(\mathbf{x})\rangle\right)}_{\text{grad} \parallel \text{normal}} d\mathbf{x} + \overset{\textbf{Not on border}}{\int_{\Omega\setminus\Omega_0}} \underbrace{\psi(\Phi(\mathbf{x}))}_{\text{Penalty on small SDF}} d\mathbf{x}$$

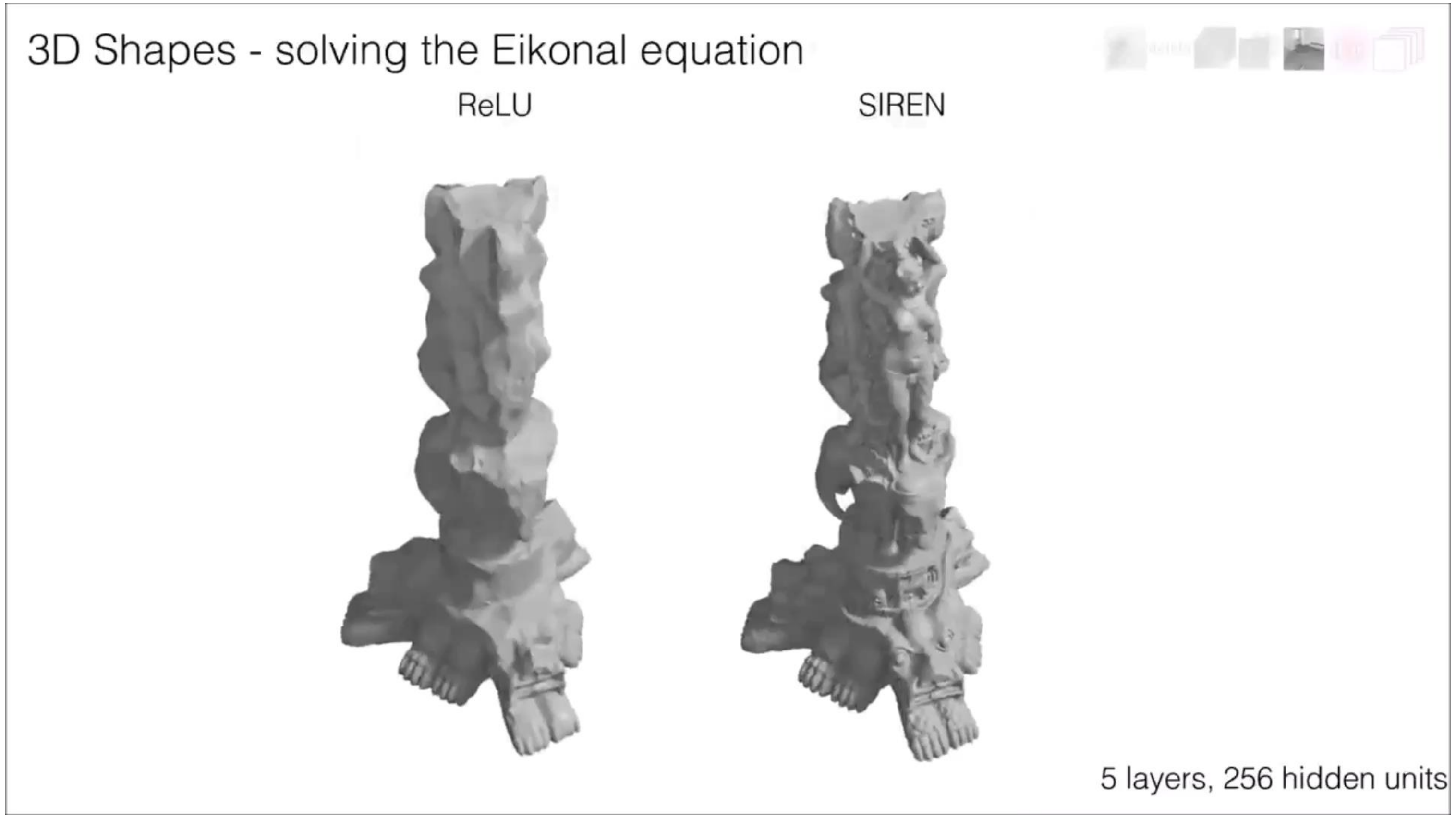


$$\psi(\mathbf{x}) = \exp(-\alpha \cdot |\Phi(\mathbf{x})|)$$

$$\alpha \gg 1$$

$$\mathcal{L}_{\mathrm{sdf}} = \int_{\Omega} \big\| \, |\nabla_{\mathbf{x}} \Phi(\mathbf{x})| - 1 \big\| d\mathbf{x} + \int_{\Omega_0} \| \Phi(\mathbf{x}) \| + \big(1 - \langle \nabla_{\mathbf{x}} \Phi(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle \big) d\mathbf{x} + \int_{\Omega \setminus \Omega_0} \psi(\Phi(\mathbf{x})) d\mathbf{x}$$

SIRENs - Signal + derivatives

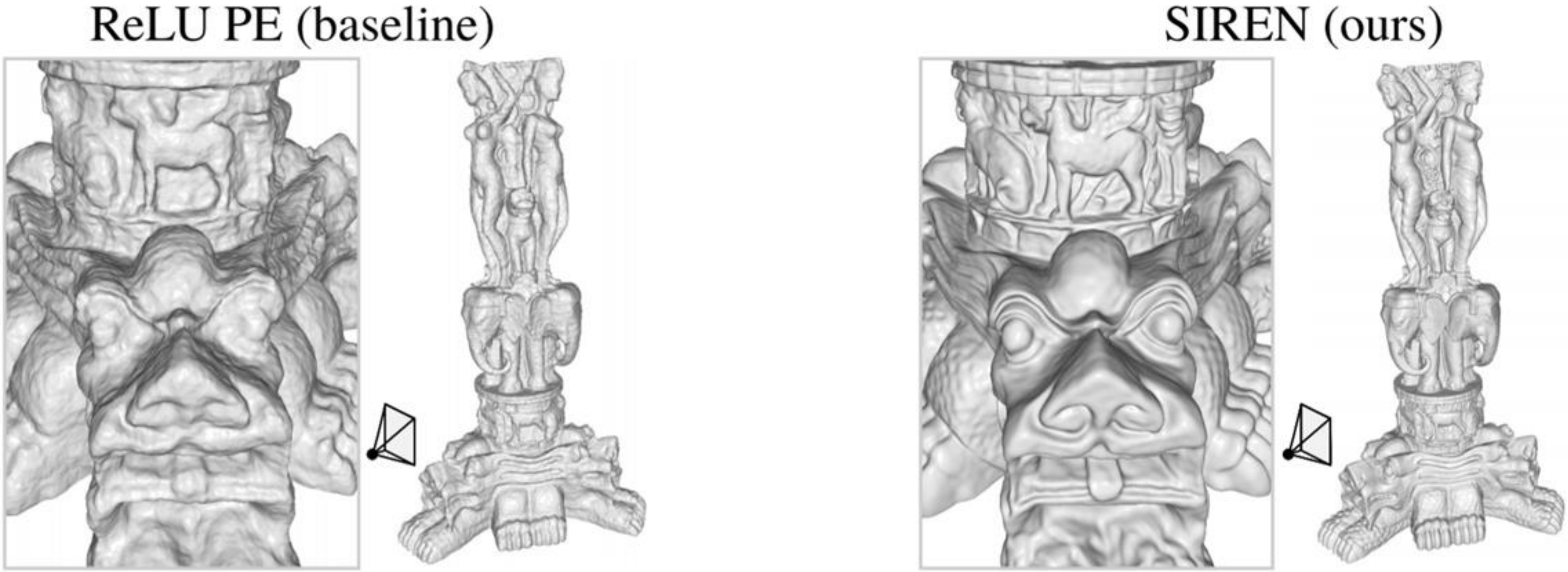


3D Shapes - solving the Eikonal equation

5 layers, 256 hidden units

$$\mathcal{L}_{\mathrm{sdf}} = \int_{\Omega} \big\| |\nabla_{\mathbf{x}}\Phi(\mathbf{x})| - 1 \big\| d\mathbf{x} + \int_{\Omega_0} \|\Phi(\mathbf{x})\| + \left(1 - \langle \nabla_{\mathbf{x}}\Phi(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle\right) d\mathbf{x} + \int_{\Omega \setminus \Omega_0} \psi(\Phi(\mathbf{x}))\, d\mathbf{x}$$

SIRENs - Signal + derivatives



ReLU PE (baseline)

SIREN (ours)

$$\mathcal{L}_{\mathrm{sdf}} = \int_{\Omega} \big\| \, |\nabla_{\mathbf{x}} \Phi(\mathbf{x})| - 1 \big\| d\mathbf{x} + \int_{\Omega_0} \|\Phi(\mathbf{x})\| + \big(1 - \langle \nabla_{\mathbf{x}} \Phi(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle\big) d\mathbf{x} + \int_{\Omega \setminus \Omega_0} \psi(\Phi(\mathbf{x})) d\mathbf{x}$$

SIRENs - Signal + derivatives

SIRENs - The wave equation

SIRENs - The wave equation

The system:

$$\boxed{\frac{\partial^2 \Phi}{\partial t^2} - c^2 \Delta \Phi = 0}$$

Input: (t, x, y)

SIRENs - The wave equation

The system:

$$\boxed{\frac{\partial^2 \Phi}{\partial t^2} - c^2 \Delta \Phi = 0}$$

Input: (t, x, y)

Initial conditions:

$$\boxed{\begin{aligned} &\frac{\partial \Phi(0, \mathbf{x})}{\partial t} = 0 \\ &\Phi(0, \mathbf{x}) = f(\mathbf{x}) \end{aligned}}$$

SIRENs - The wave equation

The system:

$$\boxed{\frac{\partial^2 \Phi}{\partial t^2} - c^2 \Delta \Phi = 0}$$

Input: (t, x, y)

Initial conditions:

$$\boxed{\begin{array}{l} \dfrac{\partial \Phi(0, \mathbf{x})}{\partial t} = 0 \\ \Phi(0, \mathbf{x}) = f(\mathbf{x}) \end{array}}$$

How to enforce?

SIRENs - The wave equation

The system:

$$\boxed{\frac{\partial^2 \Phi}{\partial t^2} - c^2 \Delta \Phi = 0}$$

Input: (t, x, y)

Initial conditions:

$$\boxed{\begin{aligned} \frac{\partial \Phi(0, \mathbf{x})}{\partial t} &= 0 \\ \Phi(0, \mathbf{x}) &= f(\mathbf{x}) \end{aligned}}$$

How to enforce? Inside the loss!

$$\boxed{L_{wave} = \int_{\Omega} \left\| \frac{\partial^2 \Phi}{\partial t^2} - c^2 \Delta \Phi \right\|_1}$$

SIRENs - The wave equation

The system:

$$\boxed{\frac{\partial^2\Phi}{\partial t^2} - c^2\Delta\Phi = 0}$$

Input: (t, x, y)

Initial conditions:

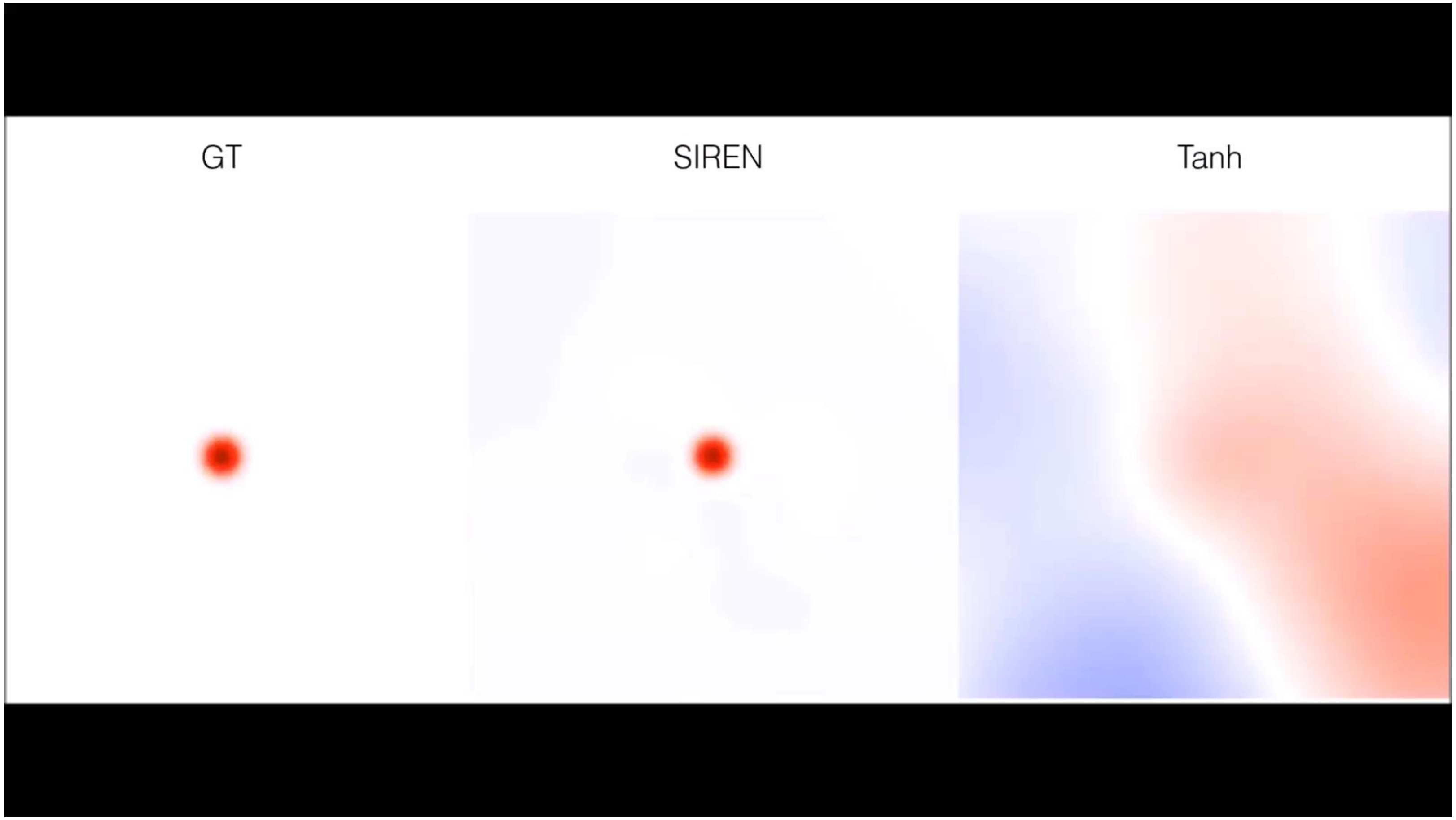
$$\boxed{\begin{aligned}\frac{\partial\Phi(0,\mathbf{x})}{\partial t} &= 0\\ \Phi(0,\mathbf{x}) &= f(\mathbf{x})\end{aligned}}$$

How to enforce? Inside the loss!

$$\boxed{L_{wave} = \int_{\Omega} \left\|\frac{\partial^2\Phi}{\partial t^2} - c^2\Delta\Phi\right\|_1 + \lambda_1(\boldsymbol{x}) \left\|\frac{\partial\Phi}{\partial t}\right\|_1 + \lambda_2(\boldsymbol{x})\|\Phi - f(\boldsymbol{x})\| d\boldsymbol{x}dt}$$

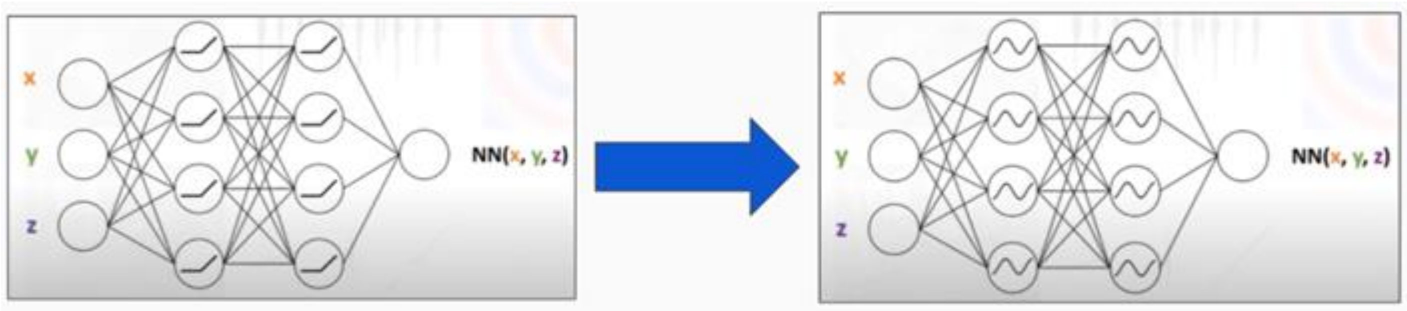
$\lambda \neq 0$ only when t=0

SIRENs - The wave equation



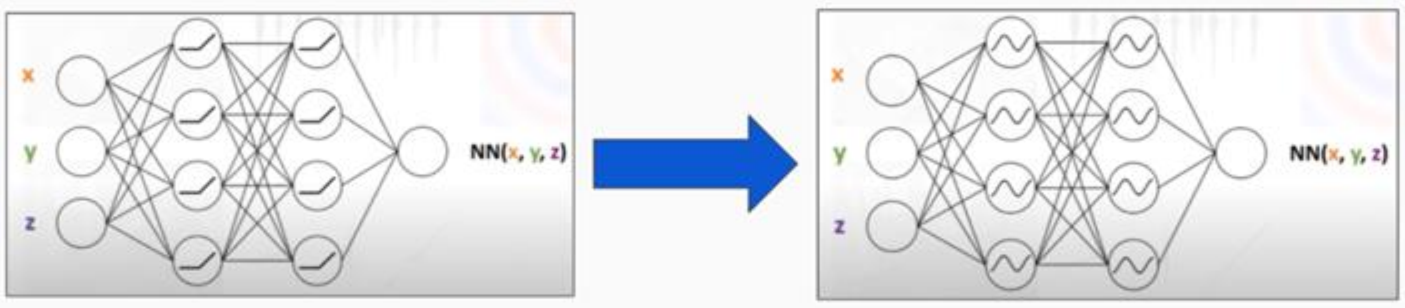
SIRENs - Summary

Simple gist

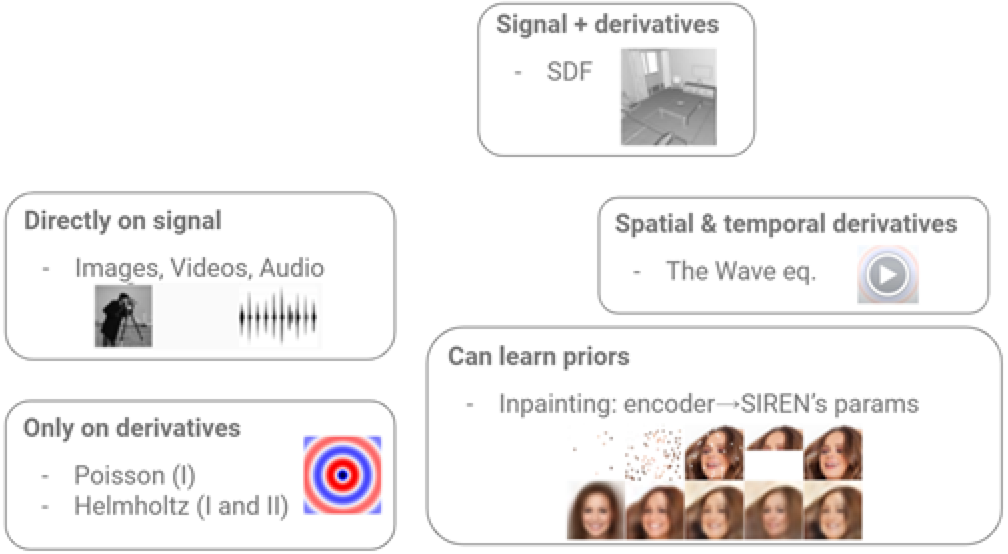


SIRENs - Summary

Simple gist



Impressive application potential



Directly on signal

- Images, Videos, Audio

Only on derivatives

- Poisson (I)
- Helmholtz (I and II)

Signal + derivatives

- SDF

Spatial & temporal derivatives

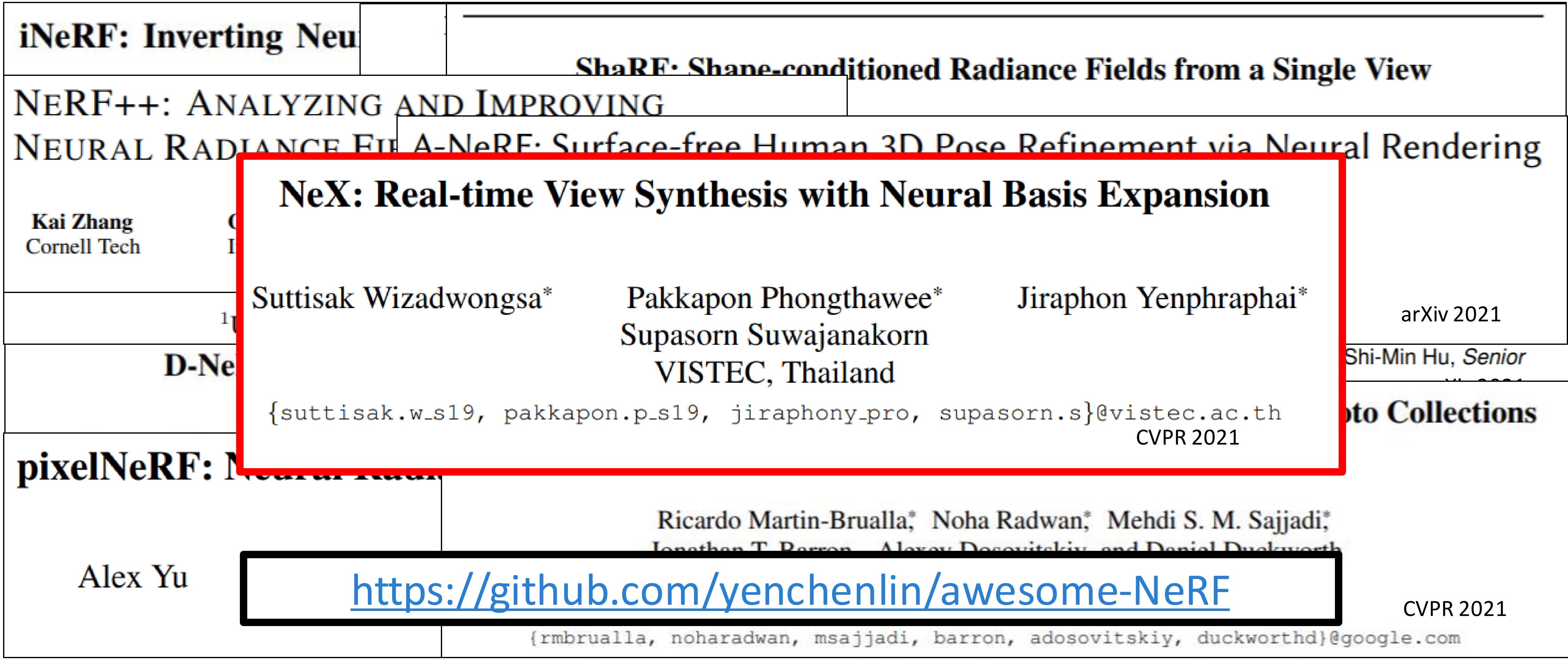
- The Wave eq.

Can learn priors

- Inpainting: encoder→SIREN's params

SIRENs - Questions?

A Rapidly Growing Research Field



iNeRF: Inverting Neu

ShaRF: Shape-conditioned Radiance Fields from a Single View

NeRF++: Analyzing and Improving Neural Radiance Fie

A-NeRF: Surface-free Human 3D Pose Refinement via Neural Rendering

Kai Zhang
Cornell Tech

arXiv 2021

NeX: Real-time View Synthesis with Neural Basis Expansion

Suttisak Wizadwongsa* Pakkapon Phongthawee* Jiraphon Yenphraphai*
Supasorn Suwajanakorn
VISTEC, Thailand
{suttisak.w_s19, pakkapon.p_s19, jiraphony_pro, supasorn.s}@vistec.ac.th

CVPR 2021

Shi-Min Hu, *Senior*

D-Ne

oto Collections

pixelNeRF: N

Alex Yu

Ricardo Martin-Brualla* Noha Radwan* Mehdi S. M. Sajjadi*
Jonathan T. Barron Alexey Dosovitskiy and Daniel Duckworth
{rmbrualla, noharadwan, msajjadi, barron, adosovitskiy, duckworthd}@google.com

https://github.com/yenchenlin/awesome-NeRF

CVPR 2021

NeX: Real-time View Synthesis with Neural Basis Expansion

Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, Supasorn Suwajanakorn

CVPR 2021

NeX - Real-time View Synthesis with Neural Basis Expansion

NeX - Contributions

NeX - Contributions

1. Real time rendering (new view synthesis)

NeX - Contributions

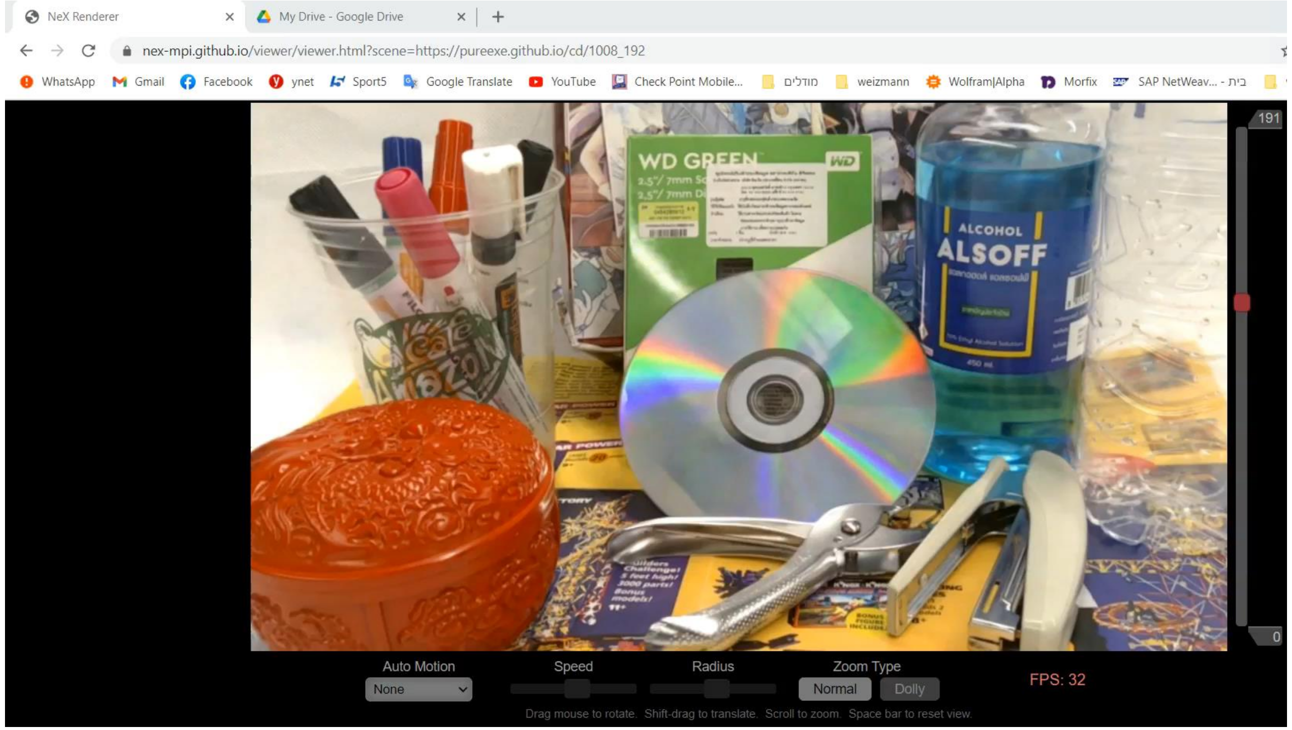
1. Real time rendering

On same NVIDIA RTX 2080Ti:
300 fps VS NeRF: **0.018** (55 spf)

NeX - Contributions

1. Real time rendering

On same NVIDIA RTX 2080Ti:
300 fps VS NeRF: **0.018** (55 spf)



Auto Motion: None

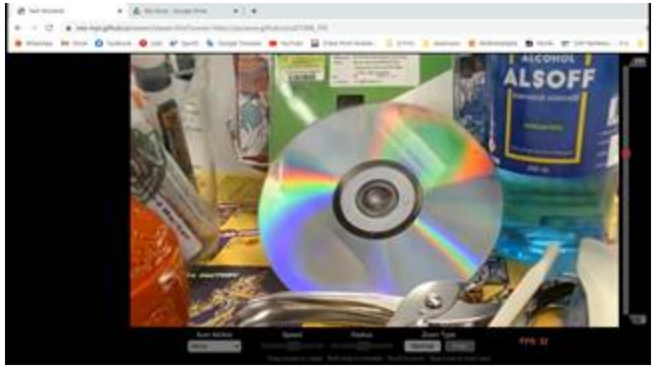
Zoom Type: Normal

FPS: 32

PC with Nvidia GeForce GTX 1650

NeX - Contributions

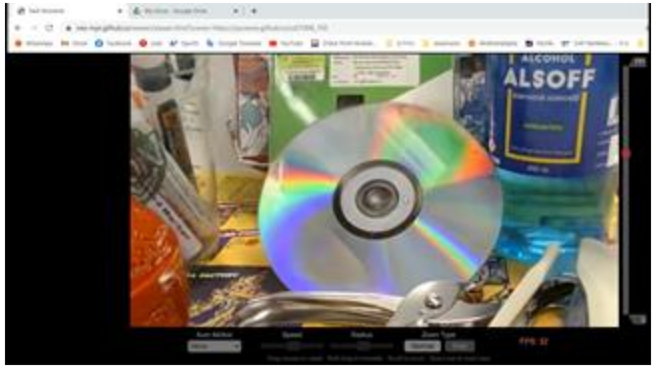
1. Real time rendering



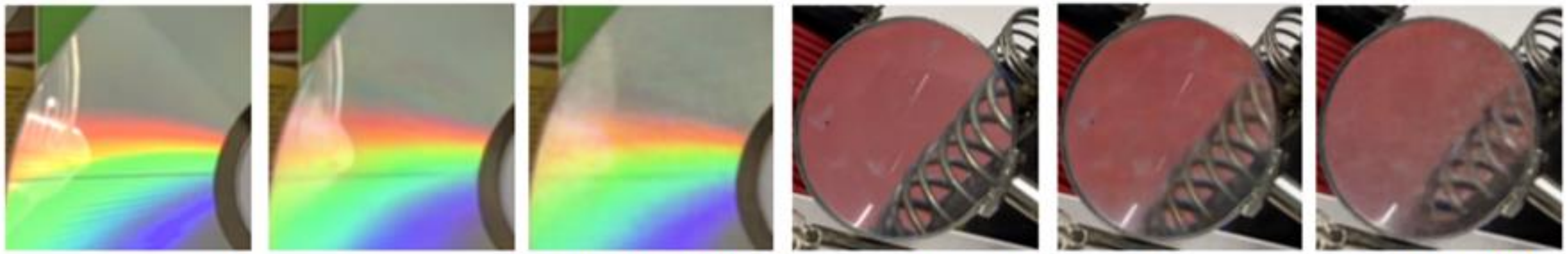
2. Better results on reflections/refractions (+ “Shiny” dataset)

NeX - Contributions

1. Real time rendering



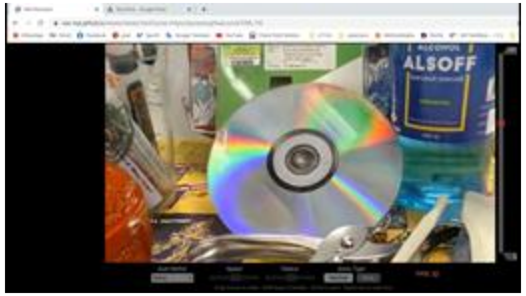
2. Better results on reflections/refractions (+ “Shiny” dataset)



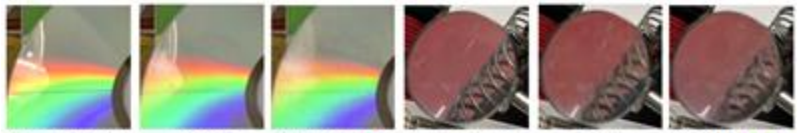
Ground truth Ours NeRF[22] Ground truth Ours NeRF[22]

NeX - Contributions

1. Real time rendering



2. Better results on reflections/refractions (+ “Shiny” dataset)



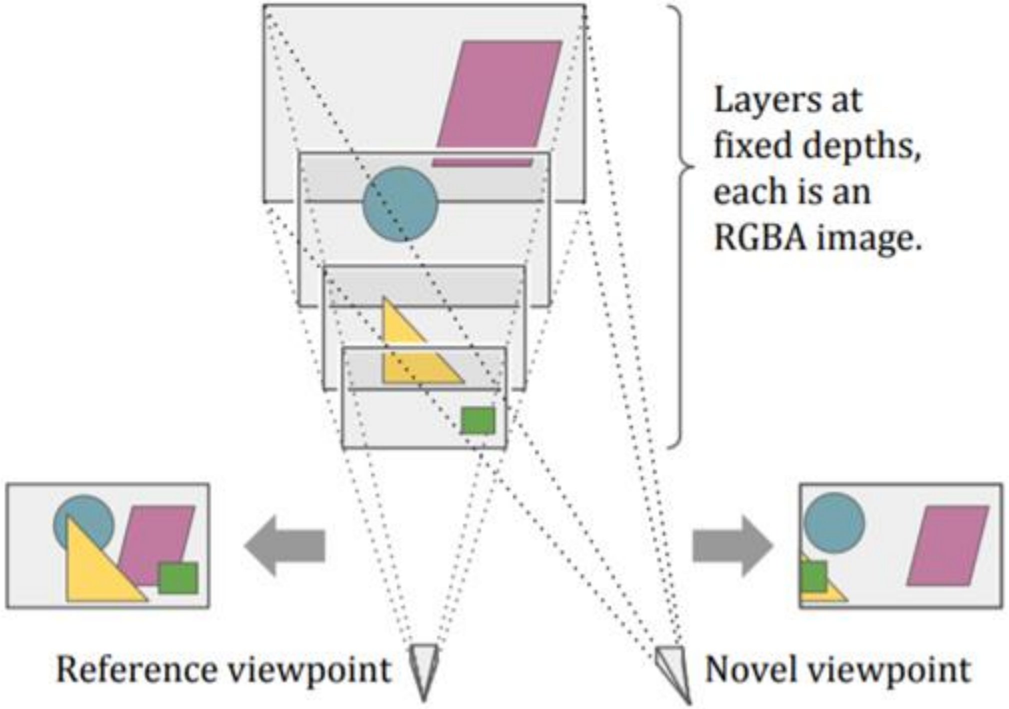
Ground truth Ours NeRF[22] Ground truth Ours NeRF[22]

3. Representation method: Implicit/Explicit & Learned Basis

NeX - Implementation

NeX - Implementation

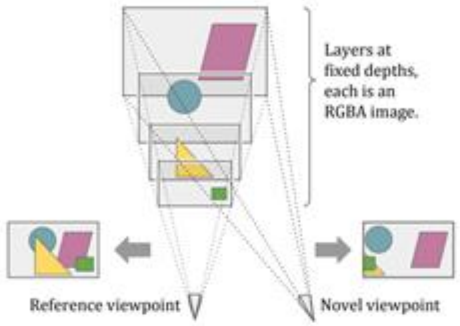
Use Multi-Plane Image (MPI)



Zhou, Tinghui, et al. "Stereo magnification: Learning view synthesis using multiplane images.", *ACM Transactions on Graphics 2018*

NeX - Implementation

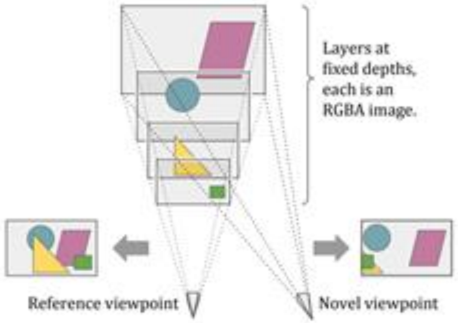
Use Multi-Plane Image (MPI)



For new angle: Homography

NeX - Implementation

Use Multi-Plane Image (MPI)



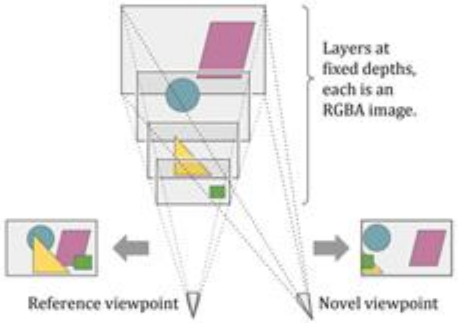
For new angle: Homography

Downside:

Only front facing scenes

NeX - Implementation

Use Multi-Plane Image (MPI)

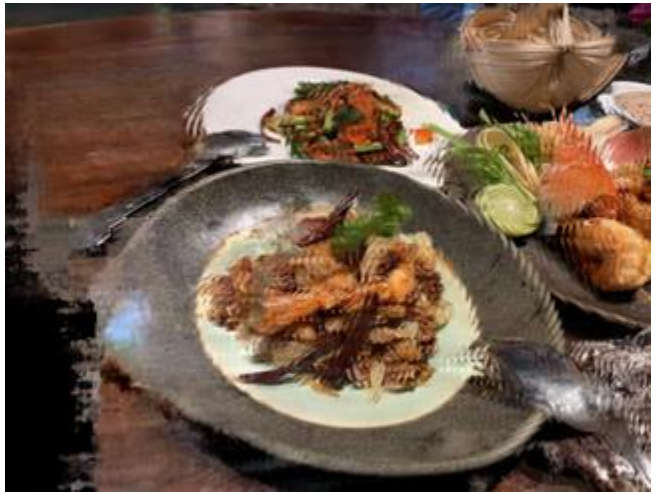


For new angle: Homography

Downside:

Only front facing scenes

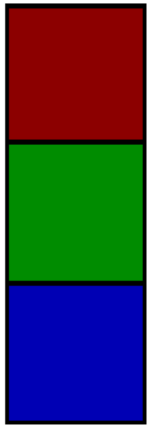
When too far:



NeX - Color representation

NeX - Color representation

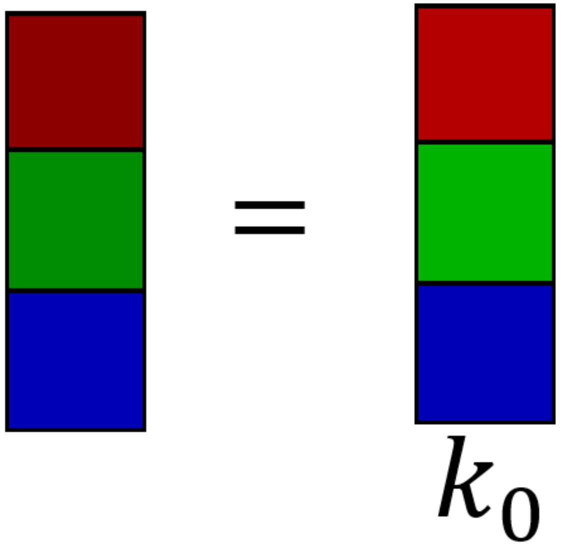
Each pixel’s RGB is “broken down”:



=

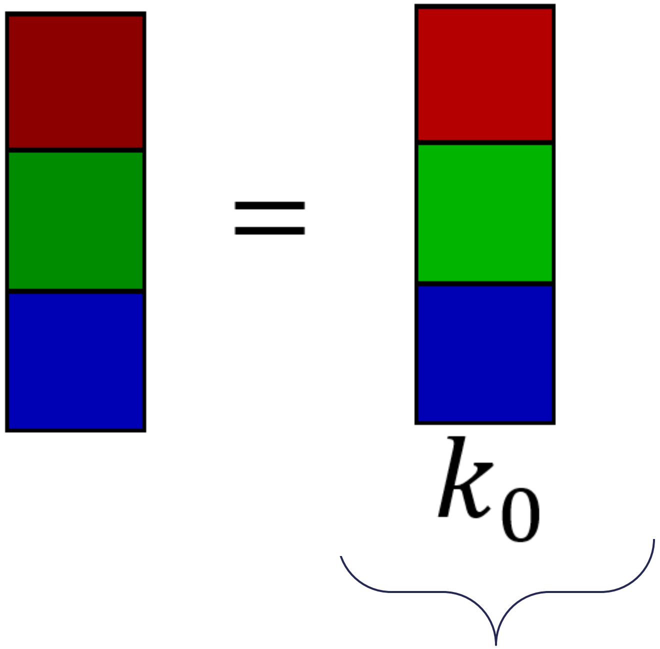
NeX - Color representation

Each pixel’s RGB is “broken down”:



NeX - Color representation

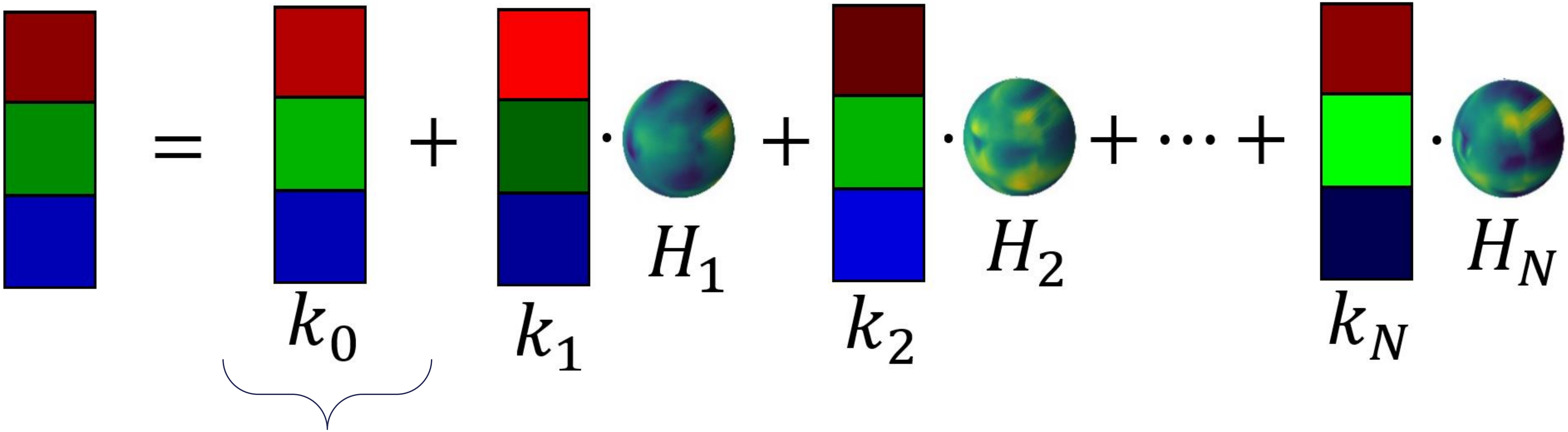
Each pixel’s RGB is “broken down”:



View independent
Explicitly

NeX - Color representation

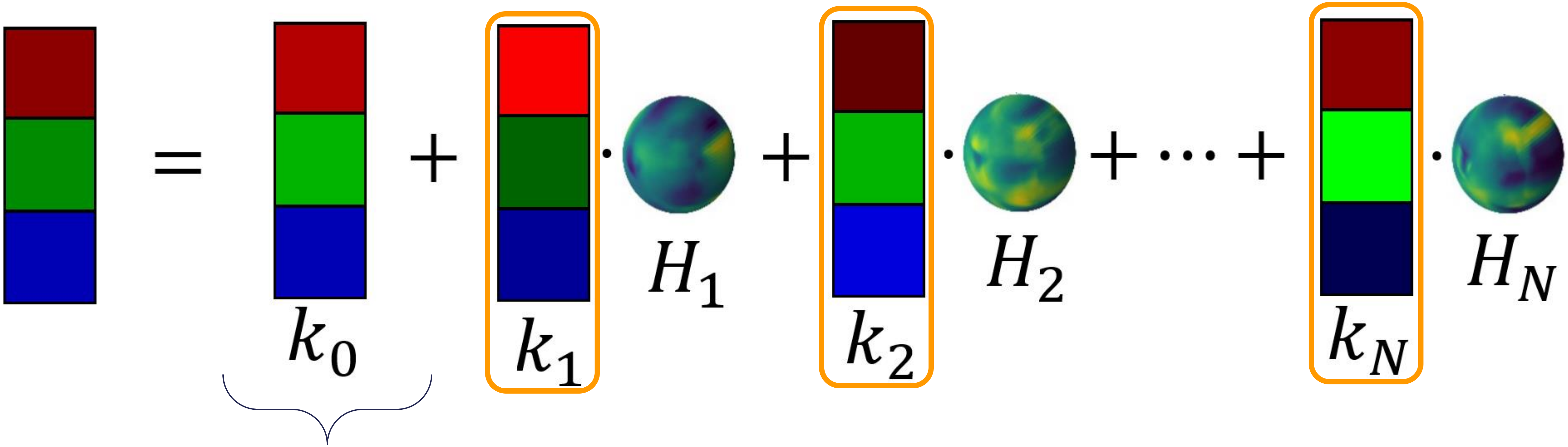
Each pixel’s RGB is “broken down”:



View independent
Explicitly

NeX - Color representation

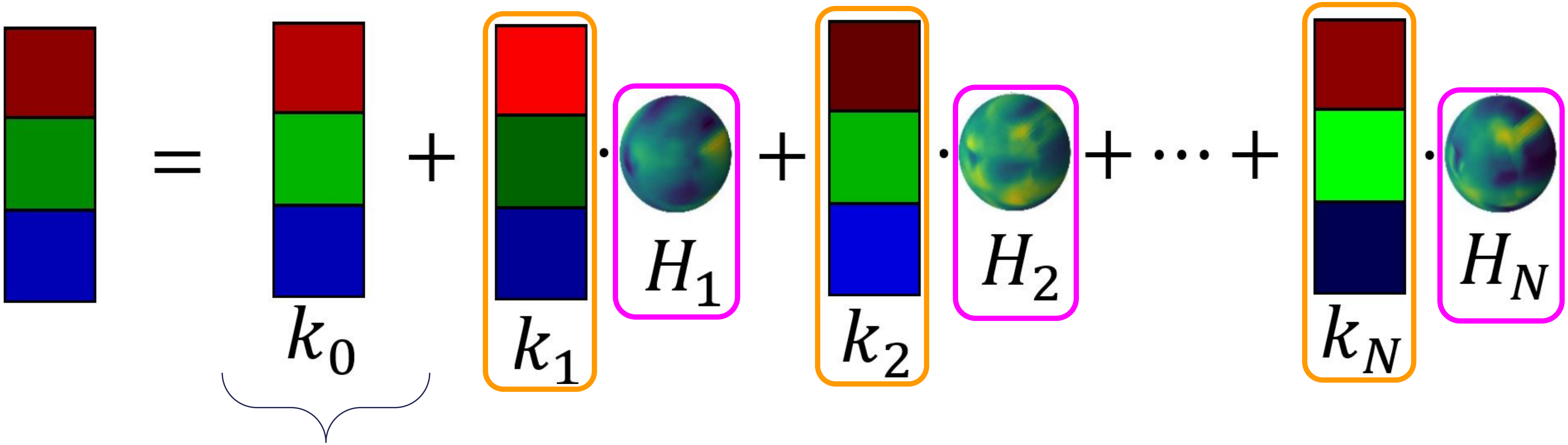
Each pixel’s RGB is “broken down”:



View independent
Explicitly

NeX - Color representation

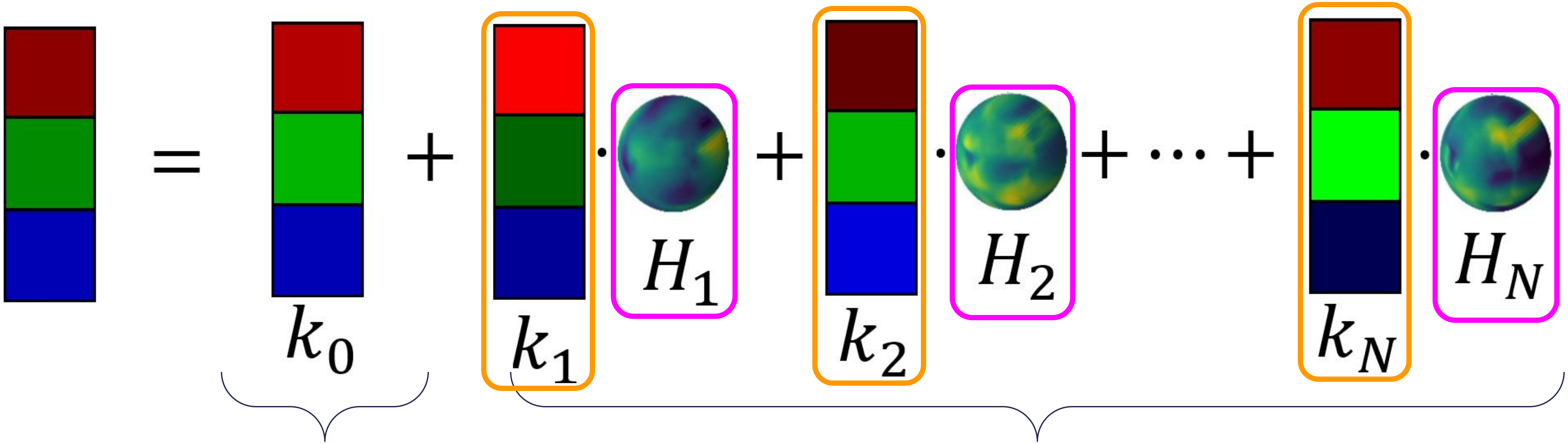
Each pixel’s RGB is “broken down”:



View independent
Explicitly

NeX - Color representation

Each pixel’s RGB is “broken down”:

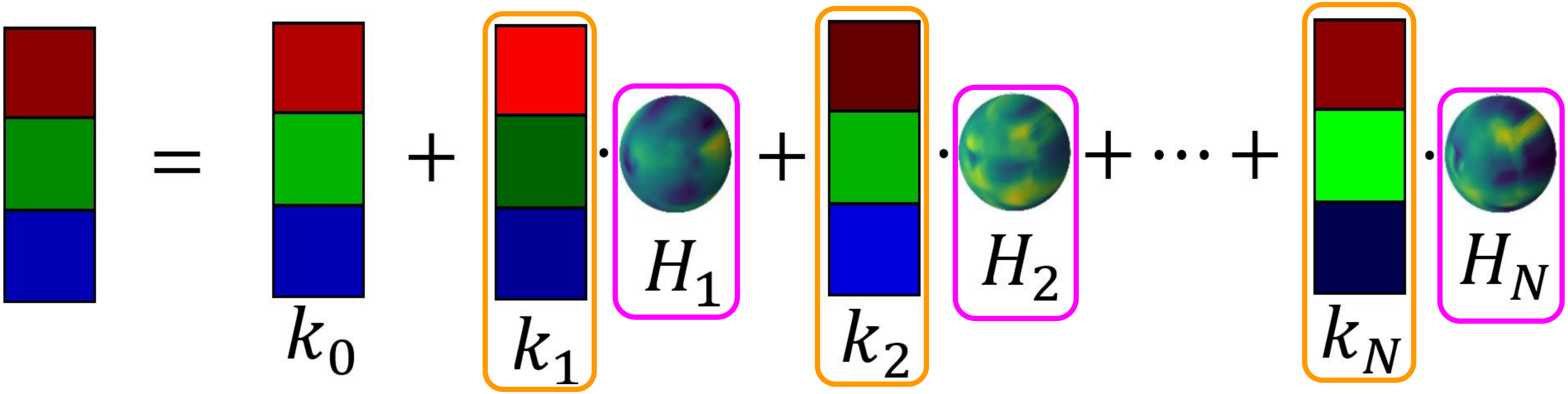

$= k_0 + k_1 \cdot H_1 + k_2 \cdot H_2 + \cdots + k_N \cdot H_N$

View independent
Explicitly

View dependent
Implicitly

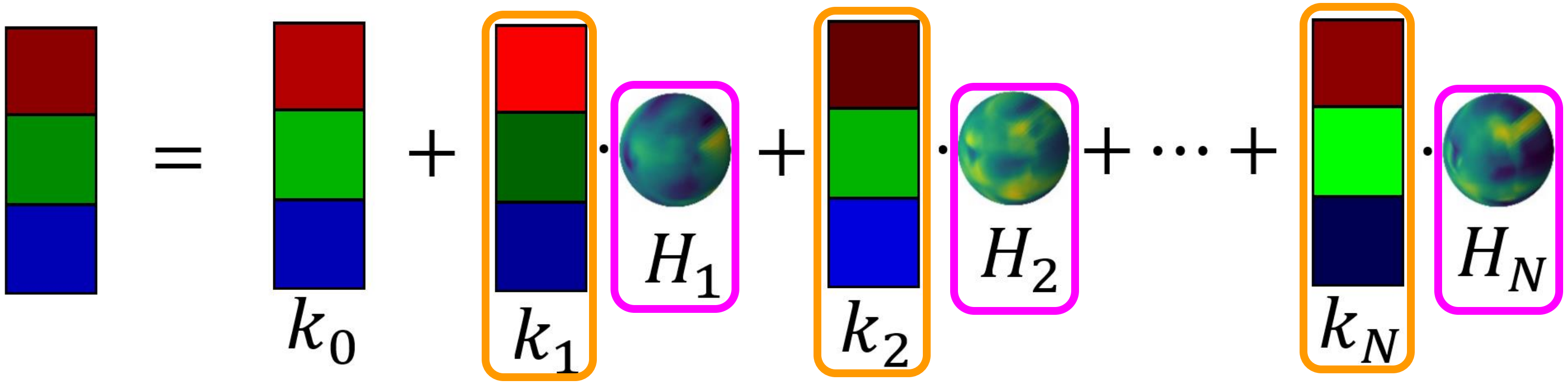
NeX - Color representation

Each pixel’s RGB is “broken down”:



$$C = K_0 + \boxed{\vec{K}} \cdot \boxed{\vec{H}_\phi(\mathcal{V}_i)}$$

NeX - Color representation - Questions?



$$C = K_0 + \boxed{\vec{K}} \cdot \boxed{\vec{H}_\phi(\mathcal{V}_i)}$$

Explicitly Learne

Implicitly Represented

NeX - Implicit/Explicit

NeX - Implicit/Explicit

In NeRF: entire scene represented implicitly in the MLP.

NeX - Implicit/Explicit

In NeRF: entire scene represented implicitly in the MLP.

In NeX: First order found **explicitly** by minimizing TV.

$$C = \boxed{K_0} + \boxed{\vec{K}} \cdot \boxed{\vec{H}_\phi}(\mathcal{V}_i)$$

Explicitly Learned

Implicitly Represented

NeX - Implicit/Explicit

In NeRF: entire scene represented implicitly in the MLP.

In NeX: First order found **explicitly** by minimizing TV.

$$C = \boxed{K_0} + \boxed{\vec{K}} \cdot \boxed{\vec{H}_\phi}(\mathcal{V}_i)$$

Explicitly Learned

Implicitly Represented

“… helps ease the network’s burden … and leads to sharper results”

NeX - Implicit/Explicit

In NeRF: entire scene represented implicitly in the MLP.

In NeX: First order found **explicitly** by minimizing TV.

$$C = \boxed{K_0} + \boxed{\vec{K}} \cdot \boxed{\vec{H}_\phi}(\mathcal{V}_i)$$

Explicitly Learned

Implicitly Represented

"... helps ease the network's burden ... and leads to sharper results"

(Reminds me of external+internal learning)

NeX - Learning the basis Functions

$$C = K_0 + \vec{K} \cdot \boxed{\vec{H}_\phi}(\mathcal{V}_i)$$

NeX - Learning the basis Functions

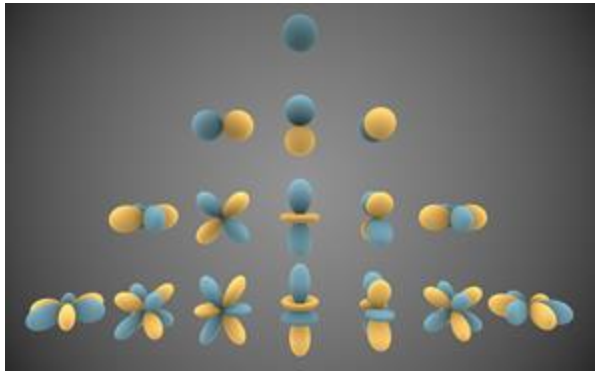
Why learn the basis functions?

$$C = K_0 + \vec{K} \cdot \boxed{\vec{H}_\phi}(\mathcal{V}_i)$$

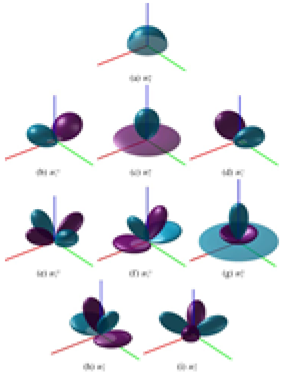
NeX - Learning the basis Functions

Why learn the basis functions?

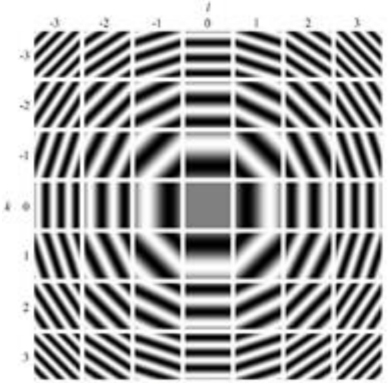
$$C = K_0 + \vec{K} \cdot \boxed{\vec{H}_\phi}(\mathcal{V}_i)$$



Spherical harmonics



Hemispherical harmonics

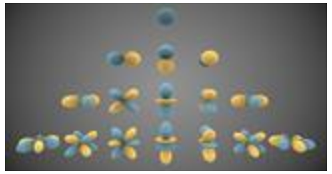


Fourier

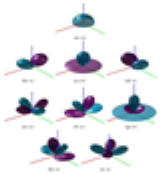
NeX - Learning the basis Functions

Why learn the basis functions?

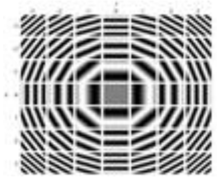
$$C = K_0 + \vec{K} \cdot \boxed{\vec{H}_\phi}(\mathcal{V}_i)$$



Spherical harmonics



Hemispherical harmonics



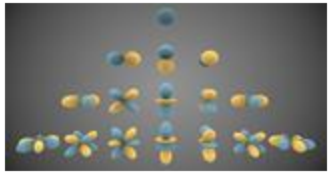
Fourier

1. Better results.. Higher frequencies with same rank order.

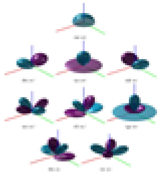
NeX - Learning the basis Functions

Why learn the basis functions?

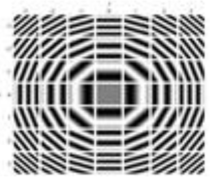
$$C = K_0 + \vec{K} \cdot \boxed{\vec{H}_\phi}(\mathcal{V}_i)$$



Spherical harmonics



Hemispherical harmonics



Fourier

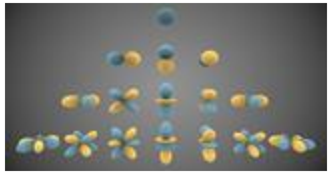
1. Better results.. **Higher frequencies** with same rank order.

$$\vec{H}_\phi(\mathcal{V}_i)$$

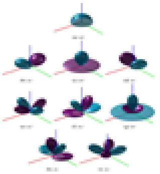
NeX - Learning the basis Functions

Why learn the basis functions?

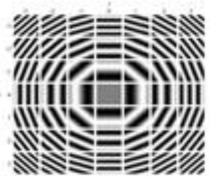
$$C = K_0 + \vec{K} \cdot \boxed{\vec{H}_\phi}(\mathcal{V}_i)$$



Spherical harmonics



Hemispherical harmonics



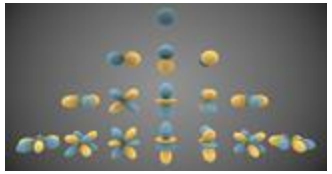
Fourier

1. Better results.. **Higher frequencies** with same rank order.
2. Since global - incorporates Image Prior.

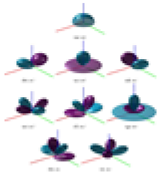
NeX - Learning the basis Functions

Why learn the basis functions?

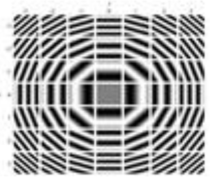
$$C = K_0 + \vec{K} \cdot \boxed{\vec{H}_\phi}(\mathcal{V}_i)$$



Spherical harmonics



Hemispherical harmonics



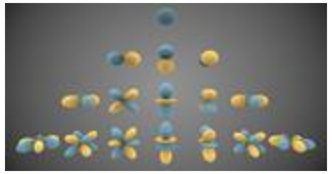
Fourier

1. Better results.. **Higher frequencies** with same rank order.
2. Since global - incorporates Image Prior.

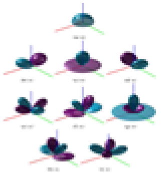
NeX - Learning the basis Functions

Why learn the basis functions?

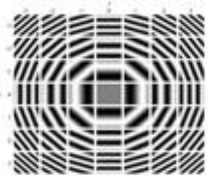
$$C = K_0 + \vec{K} \cdot \boxed{\vec{H}_\phi}(\mathcal{V}_i)$$



Spherical harmonics



Hemispherical harmonics

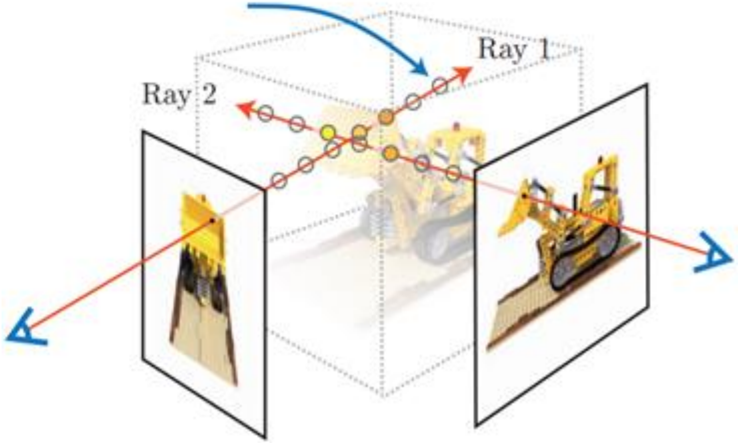


Fourier

1. Better results.. **Higher frequencies** with same rank order.
2. Since global - incorporates Image Prior.

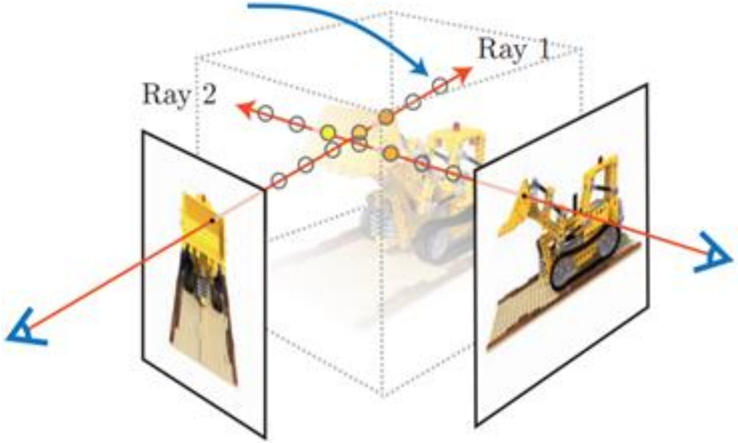
Less is more. Too many basis vectors → overfit

NeX - Real Time Rendering



NeX - Real Time Rendering

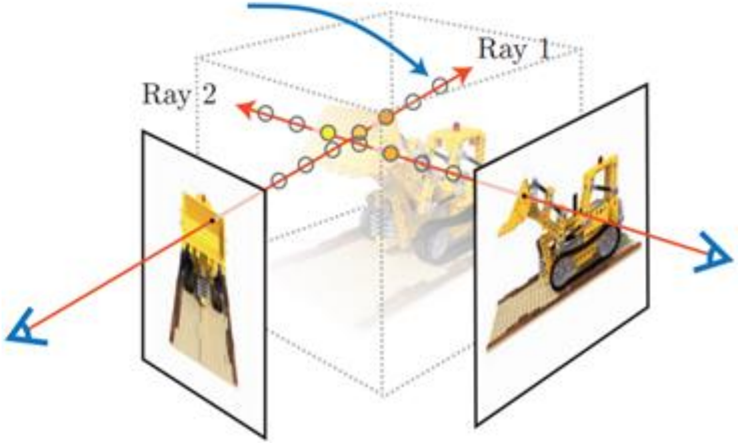
Why is NeRF rendering so slow?



NeX - Real Time Rendering

Why is NeRF rendering so slow?

For each new view synthesis:

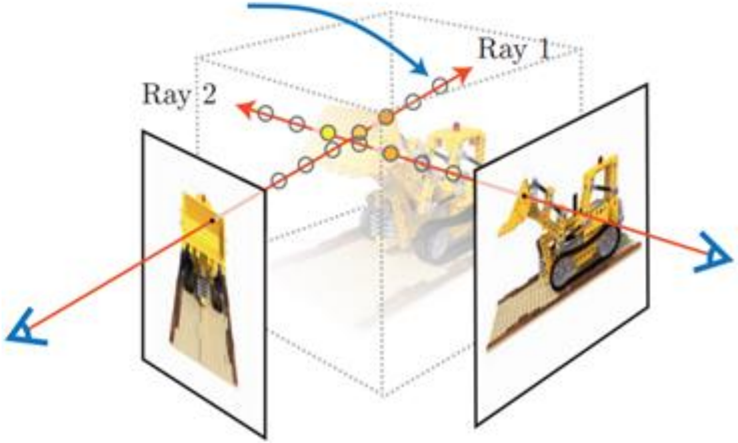


NeX - Real Time Rendering

Why is NeRF rendering so slow?

For each new view synthesis:

For each pixel:



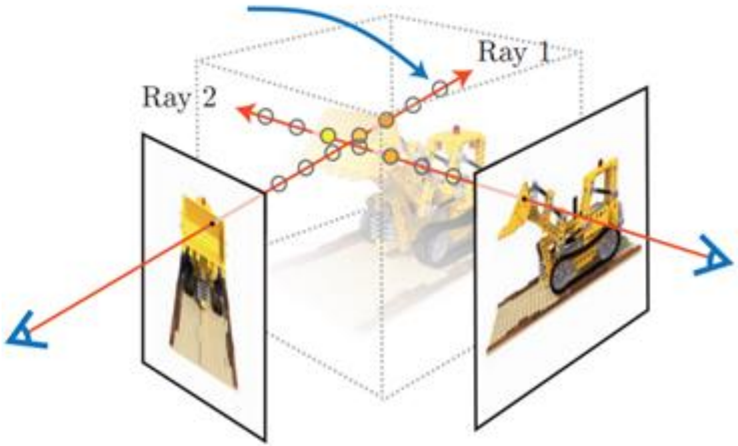
NeX - Real Time Rendering

Why is NeRF rendering so slow?

For each new view synthesis:

For each pixel:

Multiple forward passes on coarse → Where to look

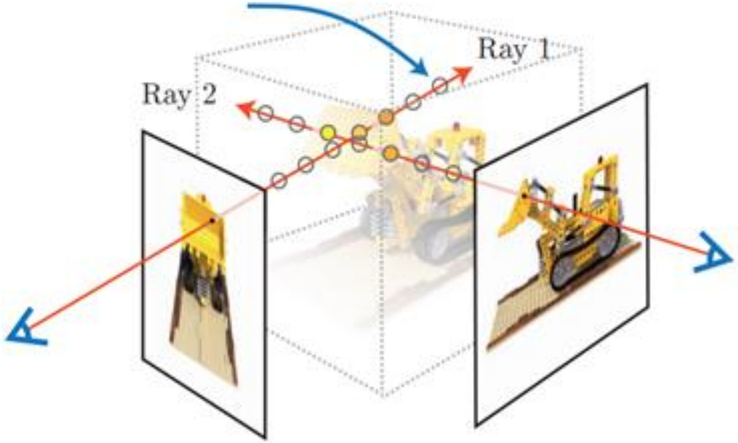


NeX - Real Time Rendering

Why is NeRF rendering so slow?

For each new view synthesis:

- **For each** pixel:
  - **Multiple** forward passes on coarse → Where to look
  - **Multiple** forward passes on fine → color & density



NeX - Real Time Rendering

Why is NeX faster?

NeX - Real Time Rendering

Why is NeX faster?

They split (x,y,d) from viewing angle

NeX - Real Time Rendering

Why is NeX faster?

They split (x,y,d) from viewing angle

$$C = K_0 + \vec{K} \cdot \vec{H}_\phi(\mathcal{V}_i)$$

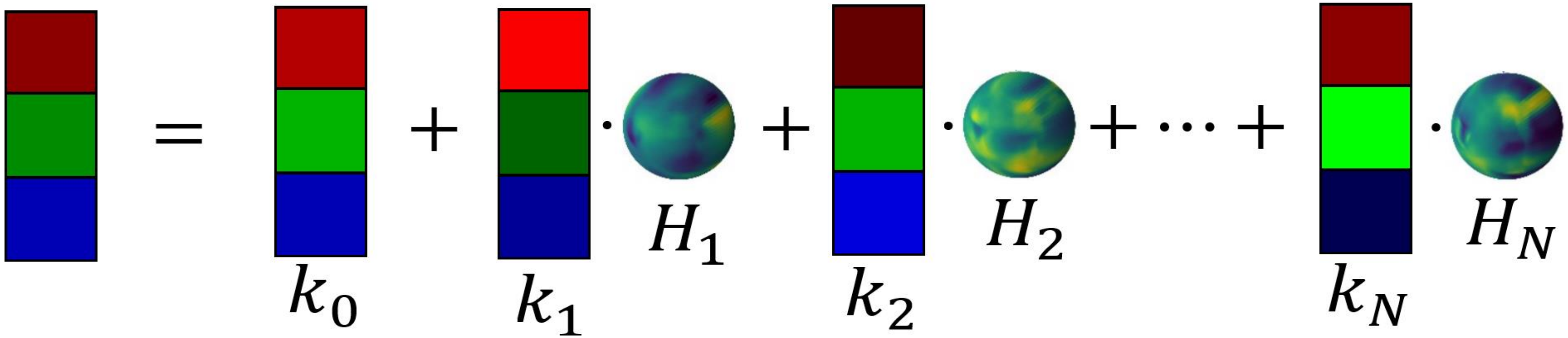
NeX - Real Time Rendering

Why is NeX faster?

They split (x,y,d) from viewing angle

$$C = K_0 + \vec{K} \cdot \vec{H}_\phi(\mathcal{V}_i)$$

1. One-time run for each pixel → magnitudes in an unknown basis



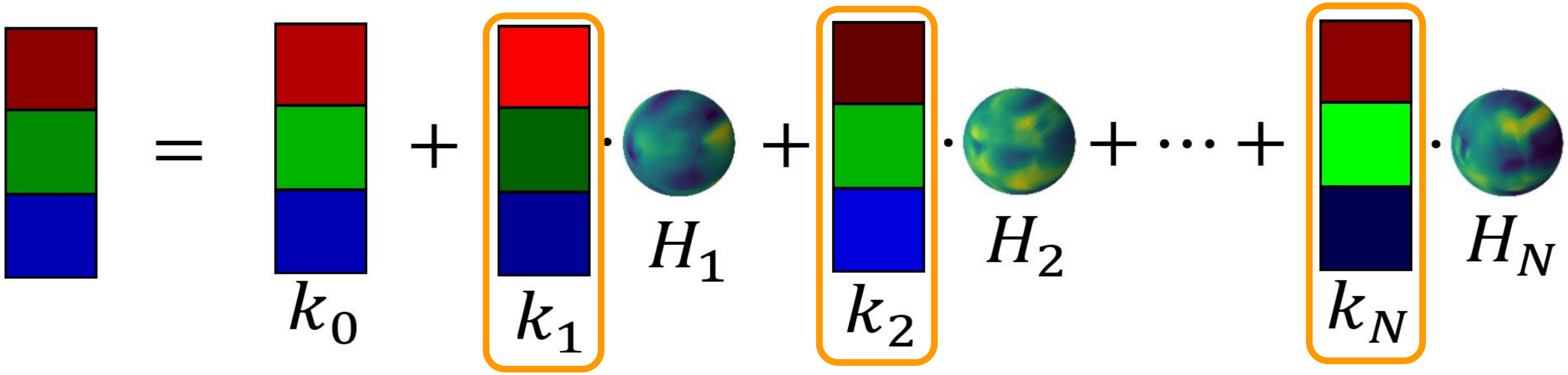
NeX - Real Time Rendering

Why is NeX faster?

They split (x,y,d) from viewing angle

$$C = K_0 + \boxed{\vec{K}} \cdot \vec{H}_\phi(\mathcal{V}_i)$$

1. One-time run for each pixel → magnitudes in an unknown basis



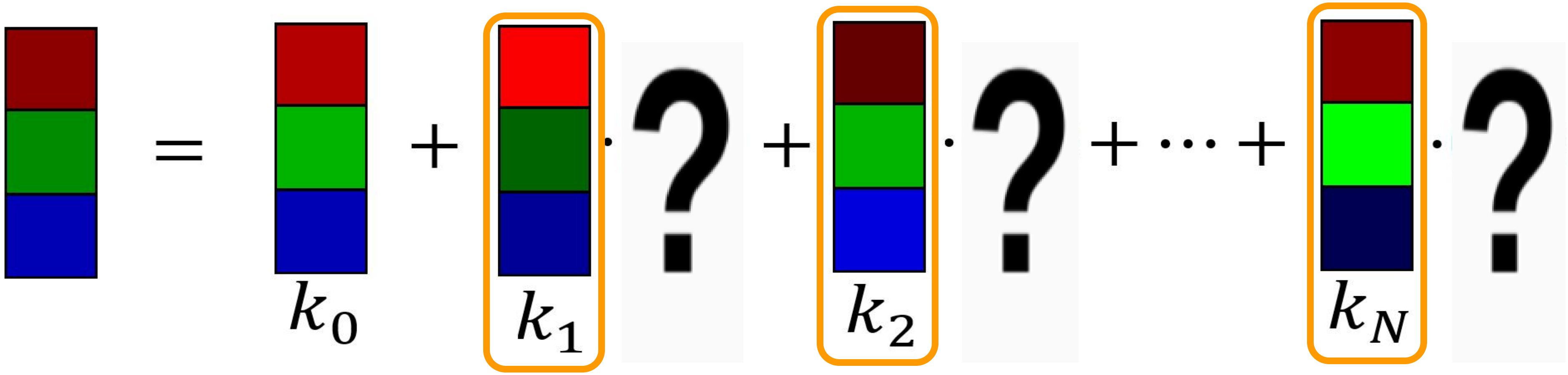
NeX - Real Time Rendering

Why is NeX faster?

They split (x,y,d) from viewing angle

$$C = K_0 + \boxed{\vec{K}} \cdot \vec{H}_\phi(\mathcal{V}_i)$$

1. One-time run for each pixel → magnitudes in an **unknown** basis



NeX - Real Time Rendering

Why is NeX faster?

They split (x,y,d) from viewing angle

$$C = K_0 + \boxed{\vec{K}} \cdot \vec{H}_\phi(\mathcal{V}_i)$$

1. One-time run for each pixel → magnitudes in an unknown basis
2. **In test time - single forward pass**: viewing angle → basis vectors.

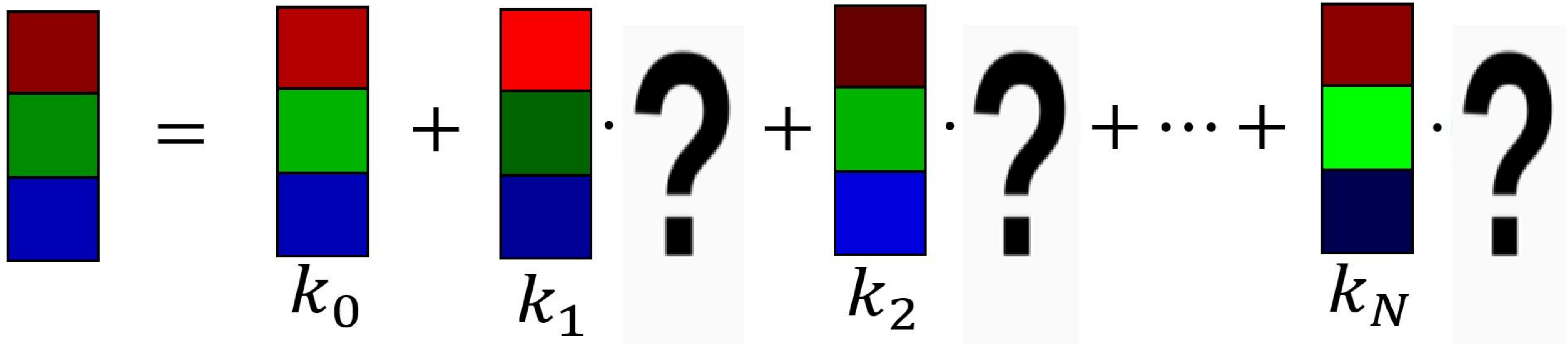
NeX - Real Time Rendering

Why is NeX faster?

They split (x,y,d) from viewing angle

$$C = K_0 + \boxed{\vec{K}} \cdot \vec{H}_\phi(\mathcal{V}_i)$$

1. One-time run for each pixel → magnitudes in an unknown basis
2. **In test time - single forward pass**: viewing angle → basis vectors.



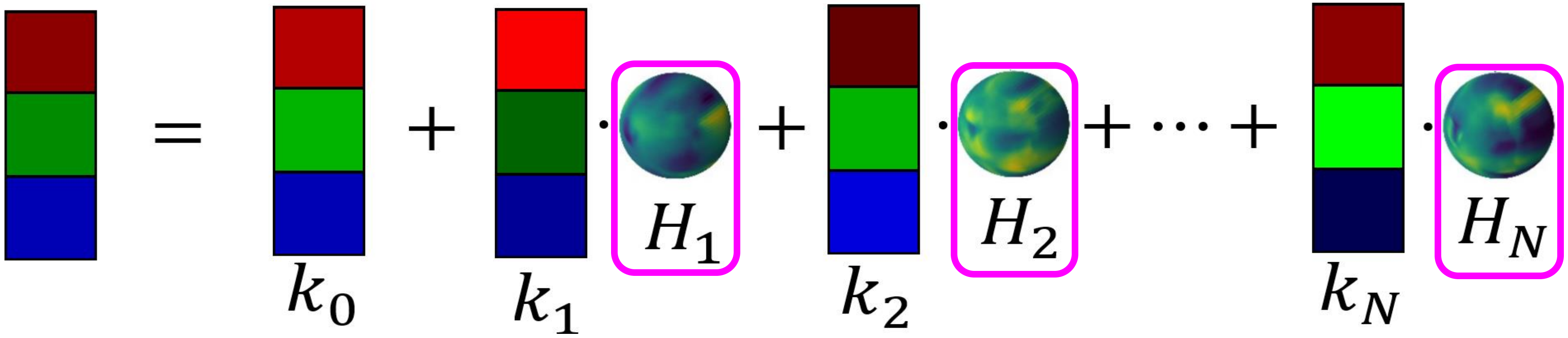
NeX - Real Time Rendering

Why is NeX faster?

They split (x,y,d) from viewing angle

$$C = K_0 + \boxed{\vec{K}} \cdot \boxed{\vec{H}_\phi(\mathcal{V}_i)}$$

1. One-time run for each pixel → magnitudes in an unknown basis
2. **In test time - single forward pass**: viewing angle → basis vectors.



NeX - Real Time Rendering

Why is NeX faster?

They split (x,y,d) from viewing angle

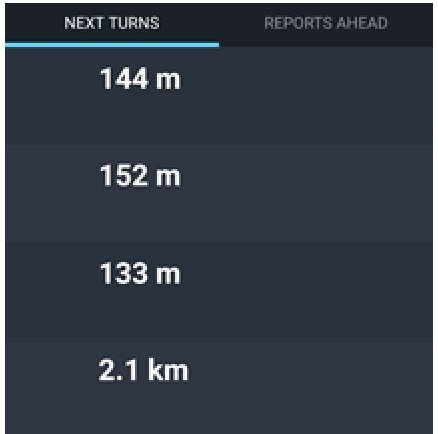
1. One-time run for each pixel → magnitudes in an unknown basis
2. **In test time - single forward pass**: viewing angle → basis vectors.

NeX - Real Time Rendering

Why is NeX faster?

They split (x,y,d) from viewing angle

1. One-time run for each pixel → magnitudes in an unknown basis
2. **In test time - single forward pass**: viewing angle → basis vectors.



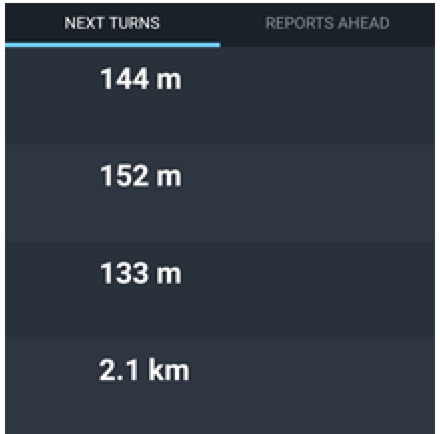
NEXT TURNS REPORTS AHEAD

144 m

152 m

133 m

2.1 km



NEXT TURNS REPORTS AHEAD

144 m

152 m

133 m

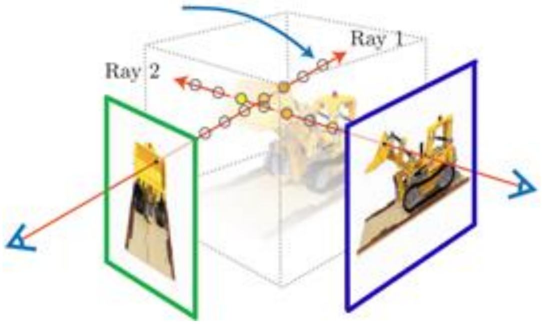
2.1 km

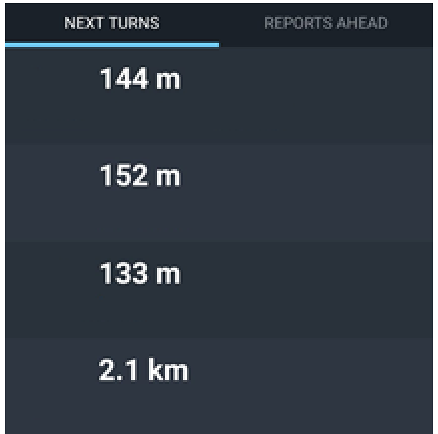
NeX - Real Time Rendering

Why is NeX faster?

They split (x,y,d) from viewing angle

1. One-time run for each pixel → magnitudes in an unknown basis
2. **In test time - single forward pass**: viewing angle → basis vectors.





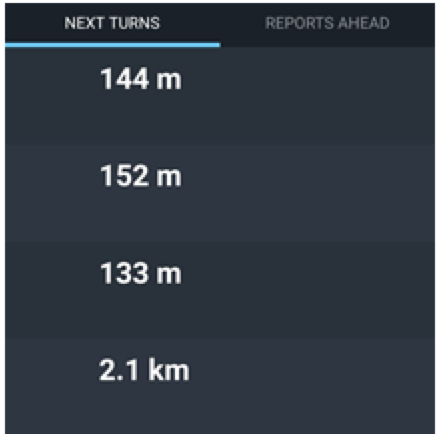
NEXT TURNS REPORTS AHEAD

144 m

152 m

133 m

2.1 km



NEXT TURNS REPORTS AHEAD

144 m

152 m

133 m

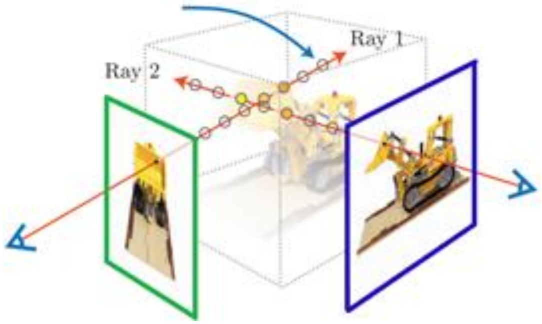
2.1 km

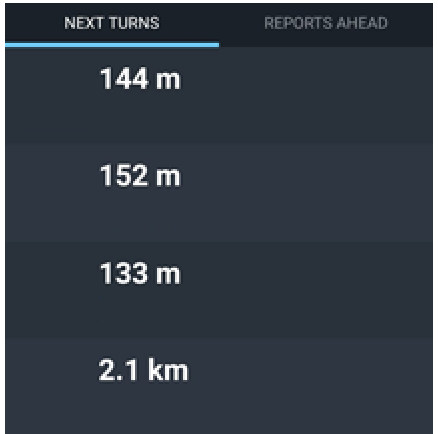
NeX - Real Time Rendering

Why is NeX faster?

They split (x,y,d) from viewing angle

1. One-time run for each pixel → magnitudes in an unknown basis
2. **In test time - single forward pass**: viewing angle → basis vectors.





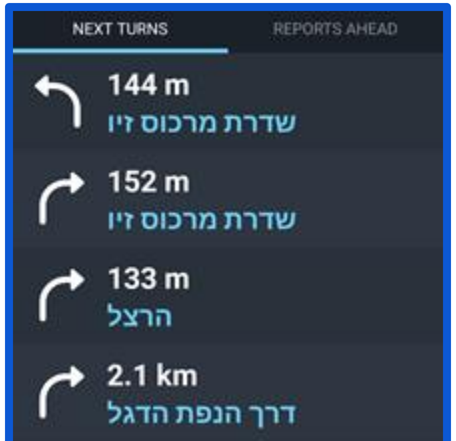
NEXT TURNS | REPORTS AHEAD

144 m

152 m

133 m

2.1 km



NEXT TURNS | REPORTS AHEAD

144 m
שדרת מרכוס זיו

152 m
שדרת מרכוס זיו

133 m
הרצל

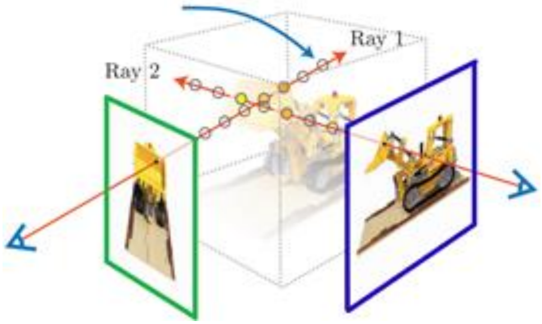
2.1 km
דרך הנפת הדגל

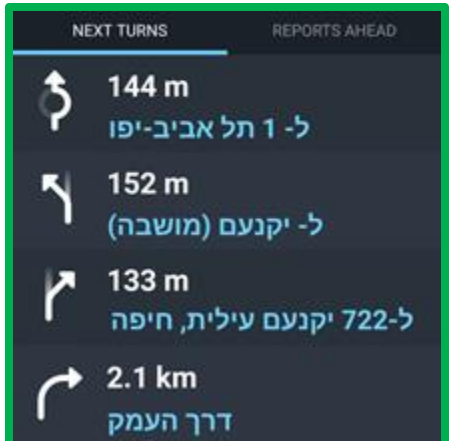
NeX - Real Time Rendering

Why is NeX faster?

They split (x,y,d) from viewing angle

1. One-time run for each pixel → magnitudes in an unknown basis
2. **In test time - single forward pass**: viewing angle → basis vectors.





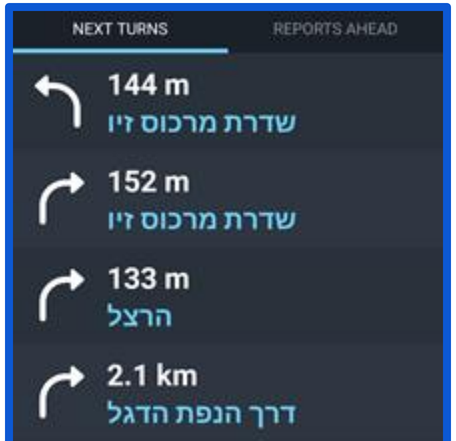
NEXT TURNS REPORTS AHEAD

144 m
ל- 1 תל אביב-יפו

152 m
ל- יקנעם (מושבה)

133 m
ל-722 יקנעם עילית, חיפה

2.1 km
דרך העמק



NEXT TURNS REPORTS AHEAD

144 m
שדרת מרכוס זיו

152 m
שדרת מרכוס זיו

133 m
הרצל

2.1 km
דרך הנפת הדגל

NeX - Short-term Nostalgia

Throwback:

NeX - Short-term Nostalgia

Throwback:

1. They use positional encoding (for both spatial coordinates and angles)

NeX - Short-term Nostalgia

Throwback:

1. They use positional encoding (for both spatial coordinates and angles)

2. They use gradients in their loss.

$$L_{\text{rec}}(\hat{I}_i, I_i) = ||\hat{I}_i - I_i||^2 + \omega||\nabla\hat{I}_i - \nabla I_i||_1$$

NeX - Short-term Nostalgia

Throwback:

1. They use positional encoding (for both spatial coordinates and angles)

2. They use gradients in their loss. Perhaps SIRENs would help?

$$L_{\text{rec}}(\hat{I}_i, I_i) = ||\hat{I}_i - I_i||^2 + \omega||\nabla\hat{I}_i - \nabla I_i||_1$$

NeX - (Our) disclaimers

NeX - (Our) disclaimers

A lot of hypertuning took place:

- α uses a sigmoid activation, and the others use tanh activations.
- Positional Encoding: (x,y) → 20 dims, d → 16 , angle → 12
- Scan for optimal number of basis functions
- To be lighter: Multiple planes (4) share color, differ in density

NeX - (Our) disclaimers

A lot of hypertuning took place:

- α uses a sigmoid activation, and the others use tanh activations.
- Positional Encoding: (x,y) → 20 dims, d → 16 , angle → 12
- Scan for optimal number of basis functions
- To be lighter: Multiple planes (4) share color, differ in density

Improvement from there? Or “deeper”?

NeX - (Our) disclaimers

Fishy comparisons:

1. NeRF is 360°, they are front-facing

NeX - (Our) disclaimers

Fishy comparisons:

1. NeRF is 360°, they are front-facing
2. One of comparisons w.o. NeRF:

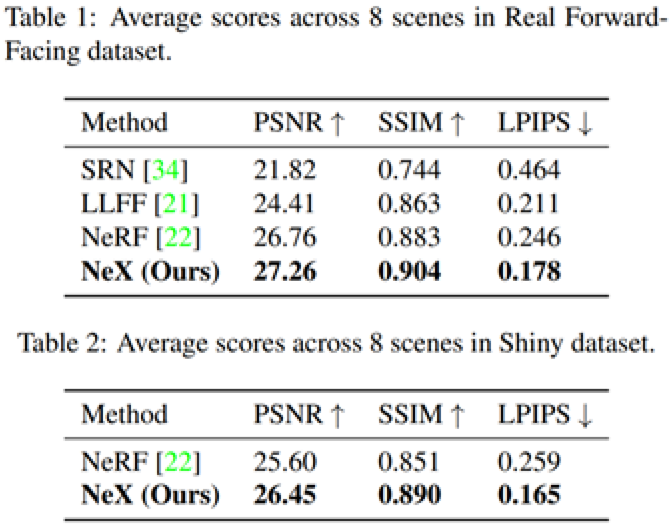


Table 1: Average scores across 8 scenes in Real Forward-Facing dataset.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
SRN [34]	21.82	0.744	0.464
LLFF [21]	24.41	0.863	0.211
NeRF [22]	26.76	0.883	0.246
NeX (Ours)	**27.26**	**0.904**	**0.178**

Table 2: Average scores across 8 scenes in Shiny dataset.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
NeRF [22]	25.60	0.851	0.259
NeX (Ours)	**26.45**	**0.890**	**0.165**

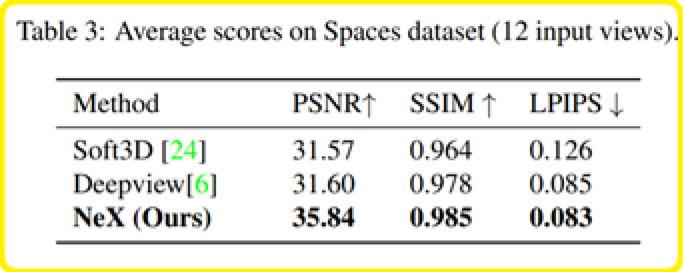


Table 3: Average scores on Spaces dataset (12 input views).

Method	PSNR↑	SSIM ↑	LPIPS ↓
Soft3D [24]	31.57	0.964	0.126
Deepview[6]	31.60	0.978	0.085
NeX (Ours)	**35.84**	**0.985**	**0.083**

NeX - Summary

NeX - Summary

Realtime new view synthesis.

NeX - Summary

Realtime new view synthesis.

Do so with “a step **back**” after NeRF

NeX - Summary

Realtime new view synthesis.

Do so with “a step **back**” after NeRF:

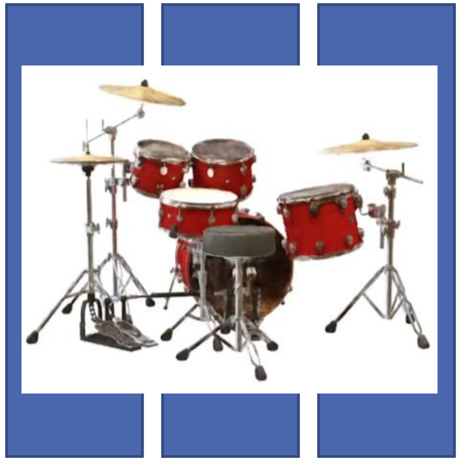
1. Some return to global
2. Some return to explicit representation

$$C = \boxed{K_0} + \vec{K} \cdot \boxed{\vec{H}_\phi(\mathcal{V}_i)}$$

NeX - Questions?

What did we see today?

Neural Implicit Representation – Representing data implicitly inside a NN



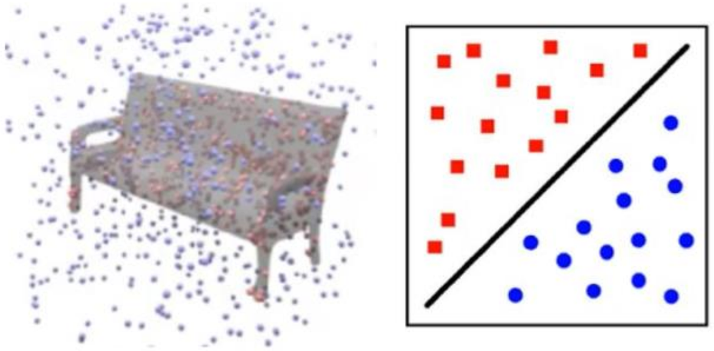
$$F_\Theta$$

What did we see today?

3D reconstruction: Implicit representation of functions

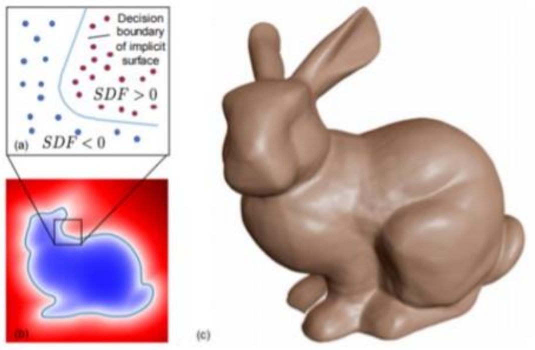
Occupancy Networks

- **Decision boundary**



DeepSDF

- **Signed Distance Function (SDF)**



What did we see today?

3D reconstruction: Implicit representation of a function

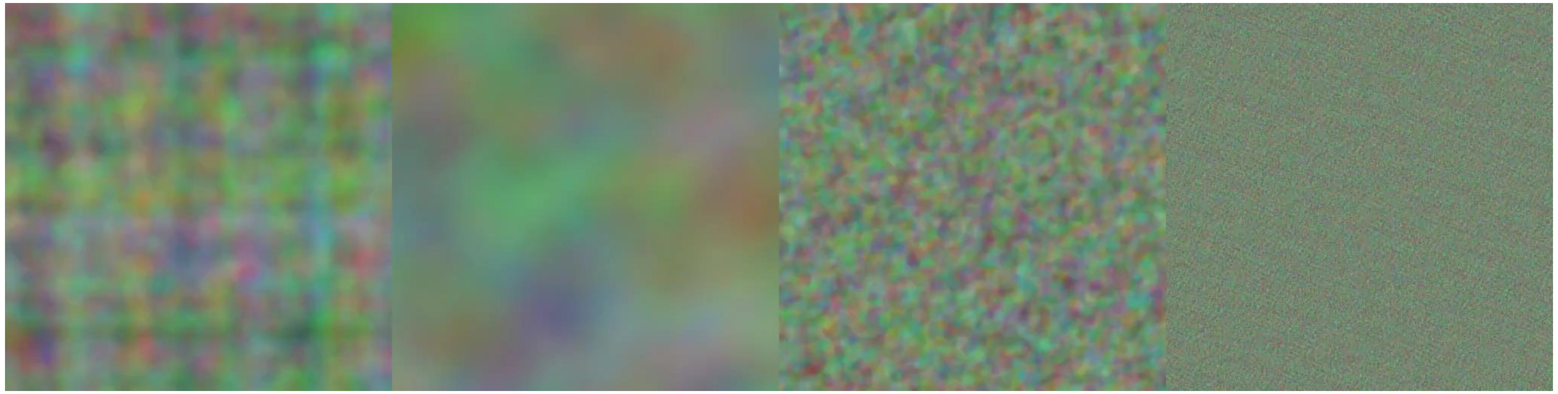
NeRF: Implicit representation of a scene

What did we see today?

3D reconstruction: Implicit representation of a function

NeRF: Implicit representation of a scene

Positional Encoding → Fourier Features



General PE

Gauss $\sigma = 1$

Gauss $\sigma = 10$

Gauss $\sigma = 100$

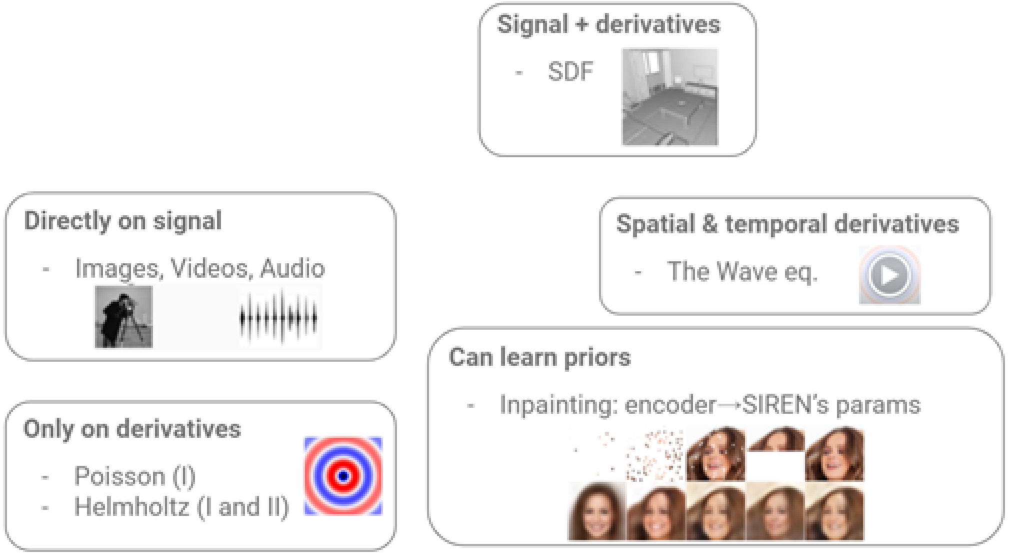
What did we see today?

3D reconstruction: Implicit representation of a function

NeRF: Implicit representation of a scene

Positional Encoding → Fourier Features

SIRENs: NIR with sine activations → new applications



Signal + derivatives

- SDF

Directly on signal

- Images, Videos, Audio

Spatial & temporal derivatives

- The Wave eq.

Only on derivatives

- Poisson (I)
- Helmholtz (I and II)

Can learn priors

- Inpainting: encoder→SIREN's params

What did we see today?

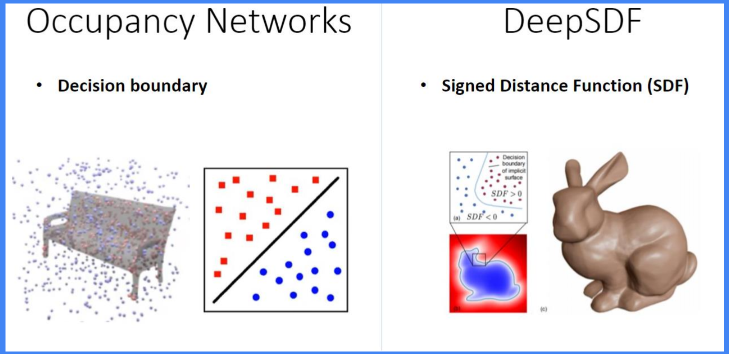
3D reconstruction: Implicit representation of a function

NeRF: Implicit representation of a scene

Positional Encoding → Fourier Features

SIRENs: NIR with sine activations → new applications

NeX: (one) Followup of NeRF



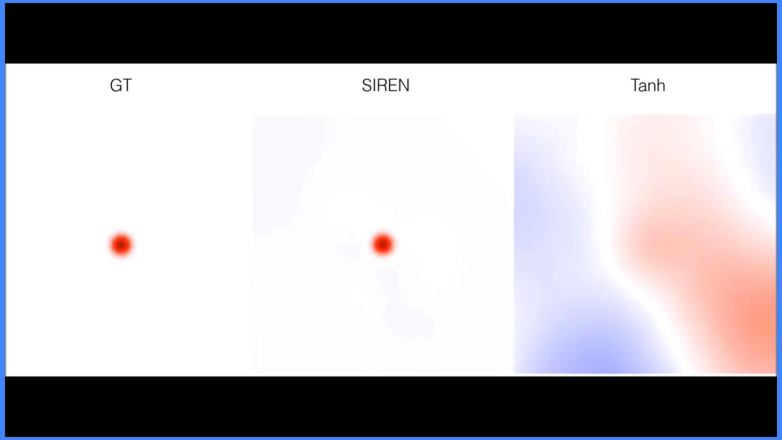
Occupancy Networks

- Decision boundary

DeepSDF

- Signed Distance Function (SDF)

Decision boundary of implicit surface
SDF > 0
SDF < 0
(a)



Questions?