ADVANCES IN COMPUTER VISION

MODULE 1:
Geometry

NEURAL RENDERING AND FIELD REPRESENTATIONS



Prof. Vincent Sitzmann
Prof. Frédo Durand

Novel View Synthesis

Motivation Eye Candy: NERF

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Ben Mildenhall[1⋆] Pratul P. Srinivasan[1⋆] Matthew Tancik[1⋆]
Jonathan T. Barron[2] Ravi Ramamoorthi[3] Ren Ng[1]

[1]UC Berkeley [2]Google Research [3]UC San Diego

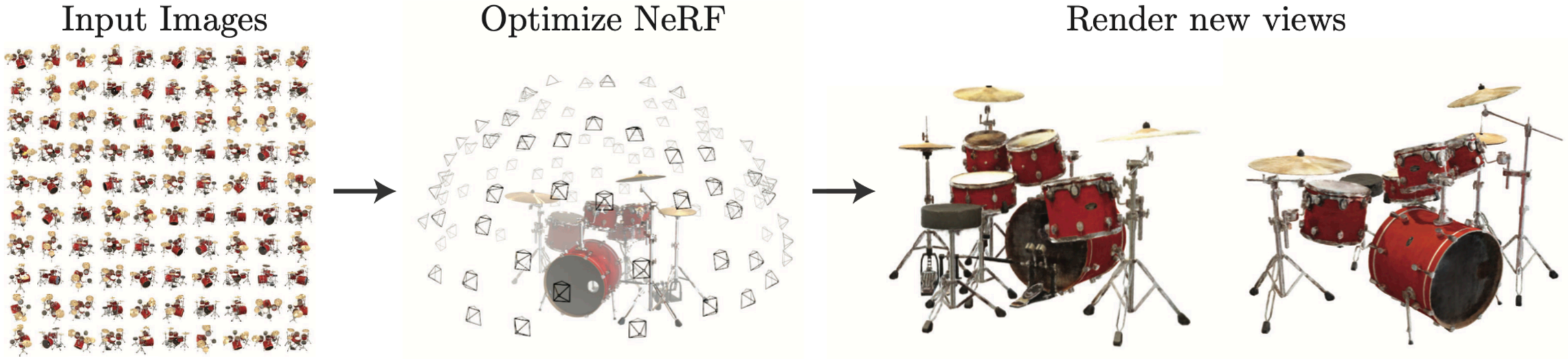
https://www.matthewtancik.com/nerf

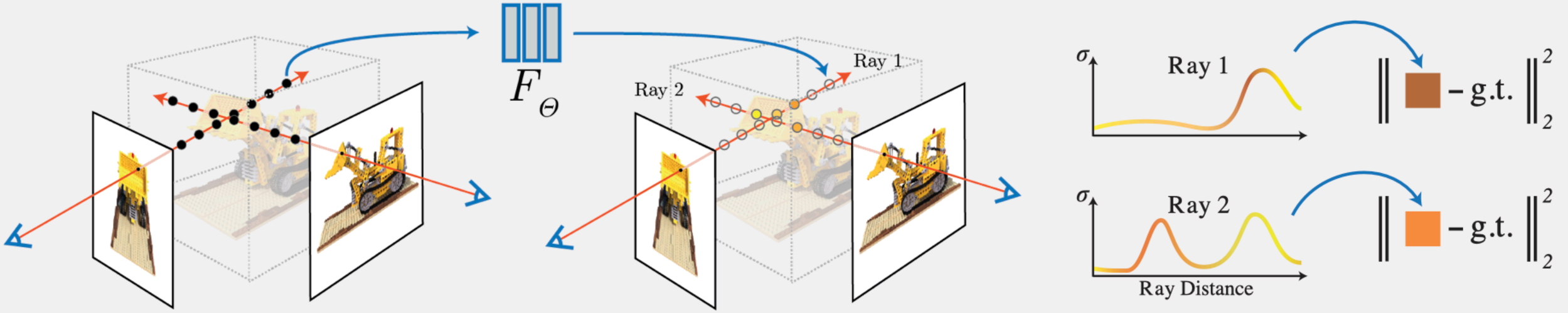
Scenario : 100s of images as input (known cameras), synthesize views for novel 3D viewpoints

Neural network (x, y, z) => RGB+opacity

Rendered with ray marching

Network overfit to single scene





We synthesize views by querying 5D coordinates along camera rays and use classic volume rendering techniques to project the output colors and densities into an image. Because volume rendering is naturally differentiable, the only input required to optimize our representation is a set of images with known camera poses. We describe how to effectively optimize neural radiance fields to render photorealistic novel views of scenes with complicated geometry and appearance, and demonstrate results that outperform prior work on neural rendering and view synthesis.

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Ben Mildenhall* UC Berkeley
Pratul P. Srinivasan* UC Berkeley
Matthew Tancik* UC Berkeley
Jonathan T. Barron Google Research
Ravi Ramamoorthi UC San Diego
Ren Ng UC Berkeley

* Denotes Equal Contribution







NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Ben Mildenhall* UC Berkeley
Pratul P. Srinivasan* UC Berkeley
Matthew Tancik* UC Berkeley
Jonathan T. Barron Google Research
Ravi Ramamoorthi UC San Diego
Ren Ng UC Berkeley

* Denotes Equal Contribution

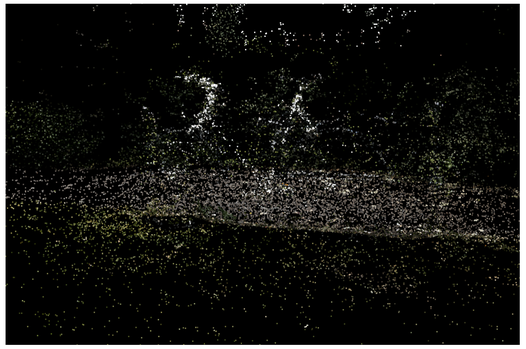






3D Gaussian Splatting

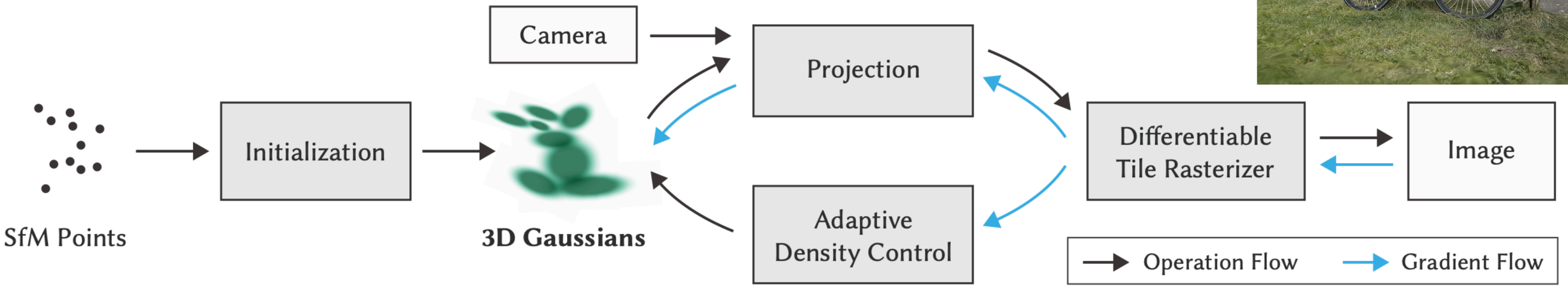
- Intuition: ray-casting through 3D is wasteful (mostly empty space)
- Represent the scene as a collection of points with size: 3D Gaussians (mean & covariance) + color & opacity
- Interestingly, builds on deep learning lessons, but removes deep learning







Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*



For NeRF Synthetic dataset we start from random points



Timelapse

For NeRF Synthetic dataset we start from random points



Timelapse

Implicit/iso Surfaces

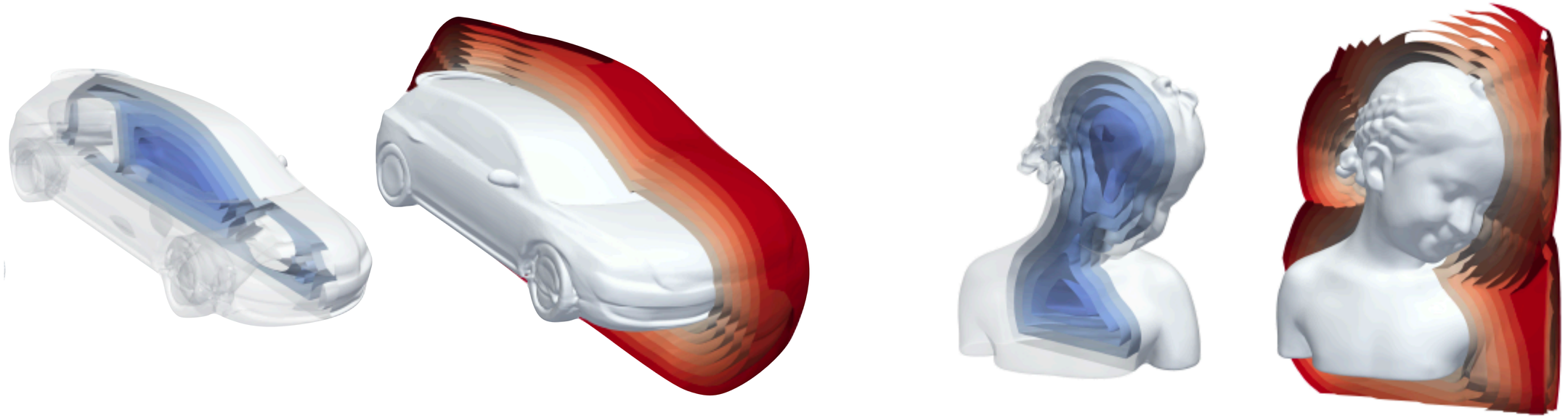


Image credits: IGR: Implicit Geometric Regularization for Learning Shapes. *Gropp et. al*

Neuralangelo: High-Fidelity Neural Surface Reconstruction

Zhaoshuo Li[1,2] Thomas Müller[1] Alex Evans[1] Russell H. Taylor[2] Mathias Unberath[2]
Ming-Yu Liu[1] Chen-Hsuan Lin[1]

DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation

Jeong Joon Park[1,3] Peter Florence [2,3] Julian Straub[3] Richard Newcombe[3] Steven Lovegrove[3]

- Reconstruction as isosurface of neural field (xyz -> distance to surface)



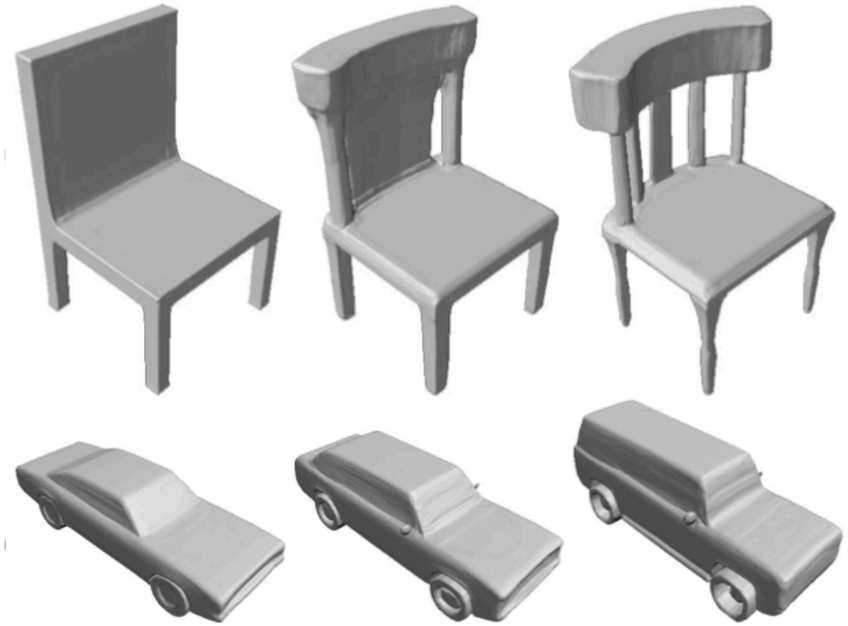


Figure 1. We present **Neuralangelo**, a framework for high-fidelity 3D surface reconstruction from RGB images using neural volume rendering, even without auxiliary data such as segmentation or depth. Shown in the figure is an extracted 3D mesh of a courthouse.

Overview and rambling

3D space is sparse

Most of 3D space is empty of solid stuff

That's why we can navigate in the world and do stuff.

Interaction is usually with the surface of things
(manipulation, locomotion on ground)

In a (mathy) sense, 3D is just an ambient space, the relevant stuff is 2D

That's why 2D vision systems work

In contrast 2D images are dense

Images compress/project/do dimensionality reduction of the sparse 3D space

And that mapping, while not trivial, is not that bad. Mostly one 3D point per 2D pixel. (As oppose to, e.g. CT computerized tomography). Mostly smooth (except boundaries)

Again, that's why 2D vision systems are so effective

Counterpoint: interesting parts of 2D images are sparse (edges, corners)

Plus curse of dimensionality: 3D is expensive, which makes sparsity important

Dealing with sparsity of 3D space

Ignore it: full volumetric representations
Volumetric sparse/adaptive/hierarchical

2D surfaces/clouds embedded in 3D
    Special case of implicit representations

2(.5)D view-based

2(.5)D Ground plan

Note about 2.5D: two flavors: just depth, or multiple layers

Make it even worse but densely redundant: 4D light fields
Make it even worse/ignore by by just keeping a collection of 2D images

Historical perspective

People used to try to make 2D sparse: edges, corners

Now we do the opposite : make 3D dense (field)

Big challenges for 3D-> 2D and 2D->3D

Differentiability

- Differentiability at all
- Sparse spiky gradient even when we have them

Loss localization and Lagrangian options:
Easy loss is localized: Eulerian on RGB in pixel-space,
plays poorly with Lagrangian representations because stuff needs to move non-locally

- Only OK if close initialization and first-order Taylor applies.
- Solutions like coarse-to-fine help, but still tricky

Lagrangian loss (e.g. error between location of a vertex and ground truth) would require correspondence

Easier to go from Lagrangian to Eulerian

- Traditional 3D graphics : 3D Lagrangian triangles -> 2D Eulerian pixels

More generally, tension 2D/3D for learning

The world is 3D

3D can provide absolute grounding

Sparse, dimensionality gap (The world is largely empty, sparse, we mostly see 2D surfaces)

- That's why human/animal/computer vision can work. The mapping is not that bad. It's not CT !
- Naturally/traditionally Lagrangian, both to deal with dimensionality gap and that's how things move in the world.

But 3D to 2D mapping is (a bit) messy, not differentiable

2D input available at scale (cameras are wonderful)

2D views are dense(r)

Dense 2D is cheap enough that we can handle it
(memory, compute)

Naturally Eulerian (largely because we can handle the full dense data)

But 2D views are more subjective/ accidental, and lose some of the 3D info

3D/NVS Major themes/dimensions/concerns

From explicit 3D reconstruction to internal representation to no 3D at all.

2D images, 3D space, 5D space of position + direction, 4D Line/ray space

Differentiability, fuzziness

Dealing with sparsity and scalability

Localization, Eulerian vs. Lagrangian

Analytical/Neural vs discrete and sample based

World-space vs camera space (esp. for single view methods)

Baked-in illumination vs. fully relightable (former for NVS)

Per-scene overfitting vs (auto) encoder per class/generic

How many input views?

Explicit data structure vs. Network as data structure

Warning: A complex landscape

- **Various forms of tasks :**
  - Reconstruction (output is 3D)
  - Novel View Synthesis (output is 2D)
- **Various scenarios**
  - Overfit to a single scene (NERF, Gaussian splatting) vs amortized training (DUST3R, VGGT)
  - General vs class specific, capture vs generative
  - How much prior is learned (training from a dataset of many scenes, not just the one to be reconstructed)
  - Baked-in illumination or full relightable model
  - Number of input views: from 1 to 100s
- **Various 3D representations** (when used)
  - Volumetric/fuzzy vs hard surface vs 4D line/ray space
  - Various degrees and combinations of neural vs traditional data structure,
  - Various levels of localization/colocation vs global
  - World space or camera space
  - Various levels of coverage, from single view to full 360 all the way to infinity
  - Or skip middleman, no explicit representation, just input images

3D Inverse Graphics Taxonomy (Vincent's version)

Field Parameterization

- MLP / Transformer
- Point Cloud Hybrid
- Voxel Grid, Voxel Grid Hybrid
- Tri-Plane, Ground Plan Hybrid

Geometry & Appearance Model

- 3D Density + Surface Light Field
- Light Field
- 3D Level Set + Surface Light Field
- Hybrid Levelset / Density + Surface Light Field

Forward Map

- Volume Rendering
- Sphere Tracing
- Light Field
- Hybrid Volume Rendering / Sphere Tracing

Inference Model

- Per-Scene Optimization
- Prior-based, encoder
- Unconditional Generative
- Prior-based, auto-decoder

FIELD REPRESENTATION

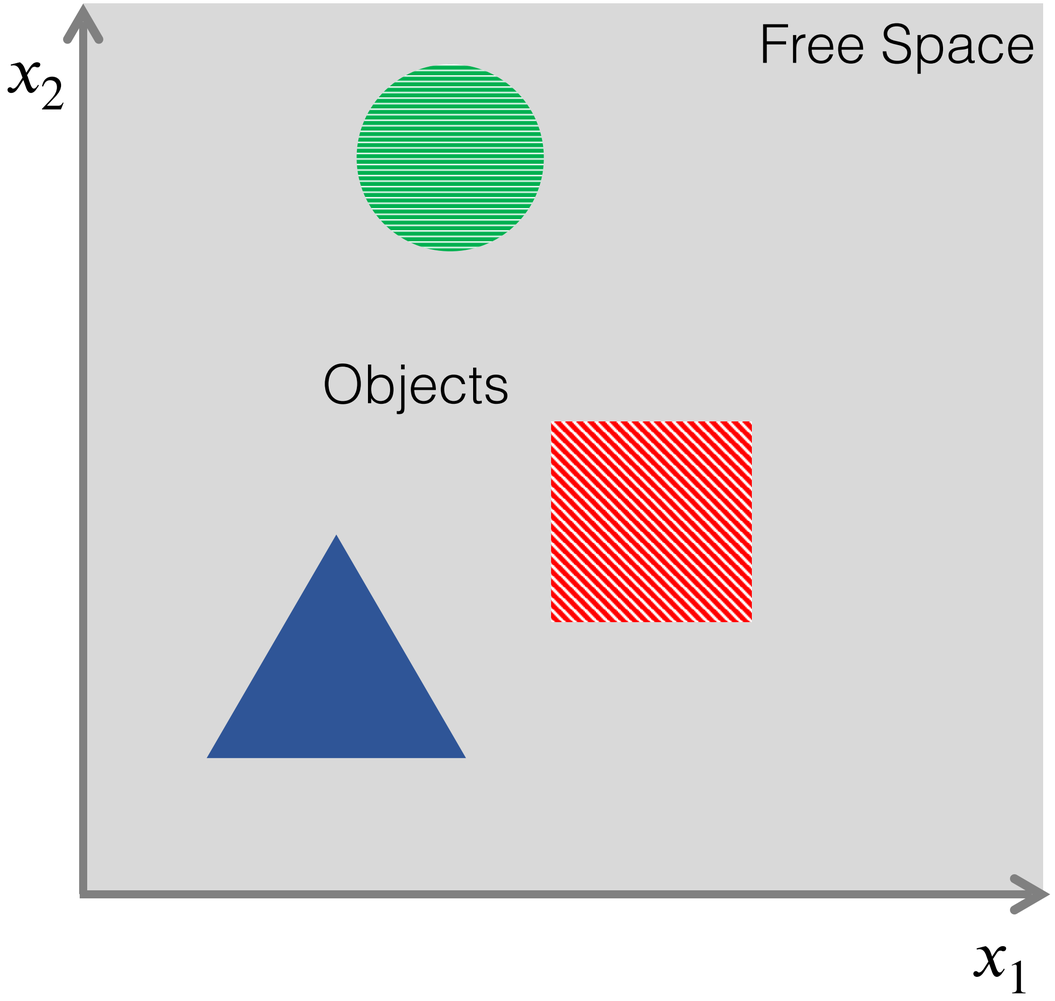
Definition: Field (physics)

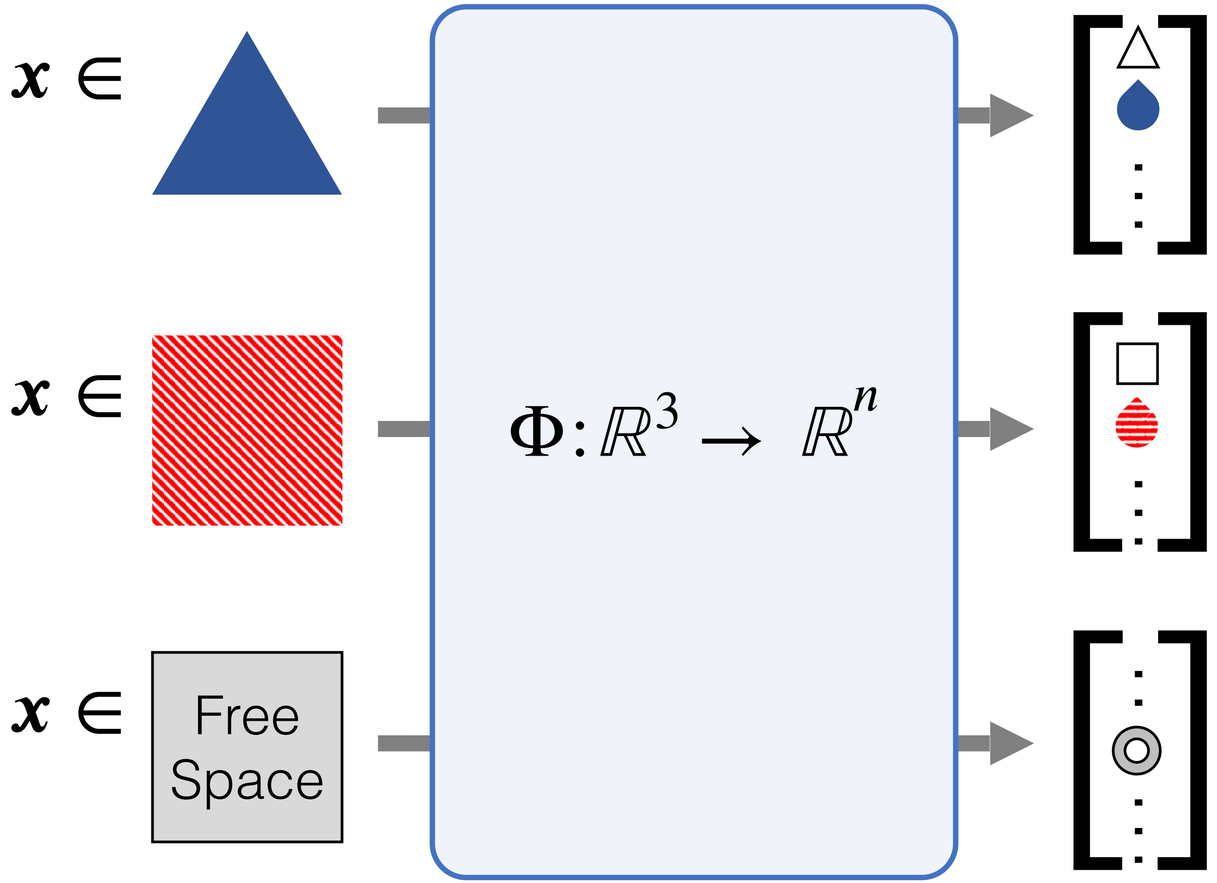
> A physical quantity represented by a scalar, vector, or tensor, that has a value for each point in space and time.

Will use term “field” for any function that takes a space, time, or space-time coordinate as input to map it to some quantity at that coordinate.

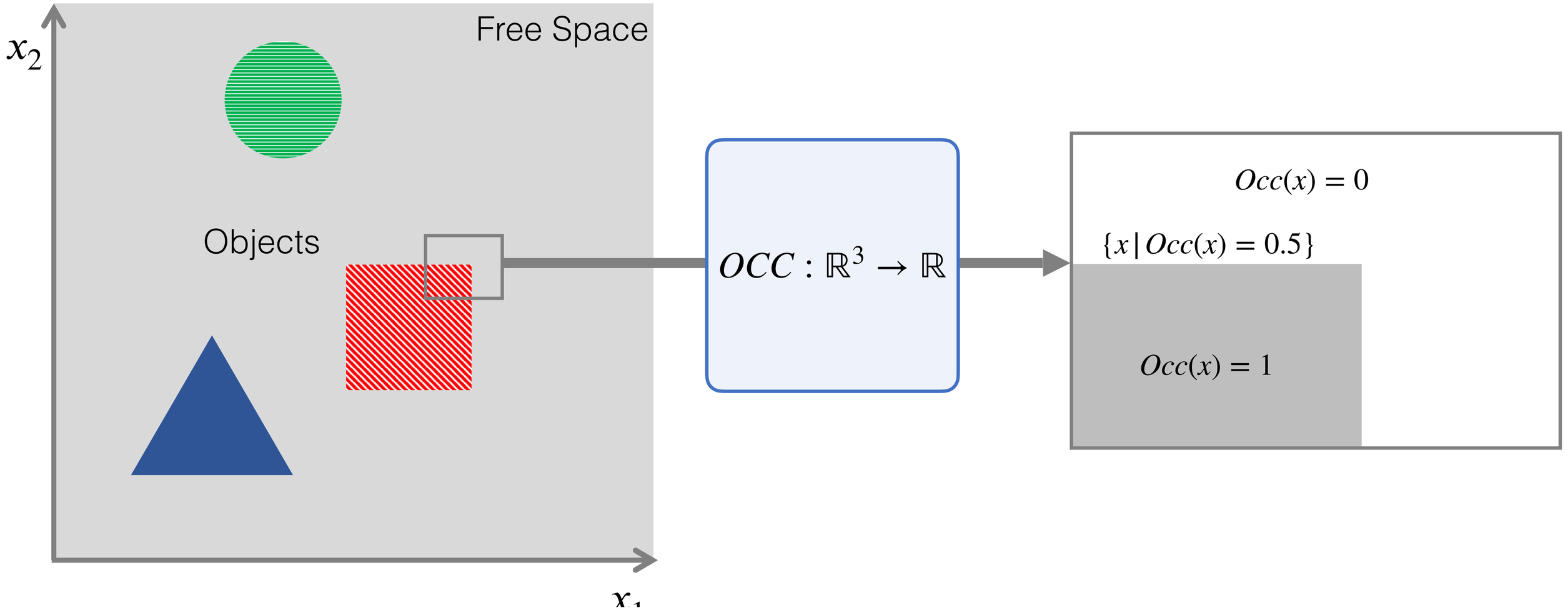
$$\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}^n$$

How to parameterize surfaces?





The Occupancy Field



Occupancy Networks: Learning 3D Reconstruction in Function Space, Mescheder et al.
IM-Net: Learning Implicit Fields for Generative Shape Modeling, Chen et al.

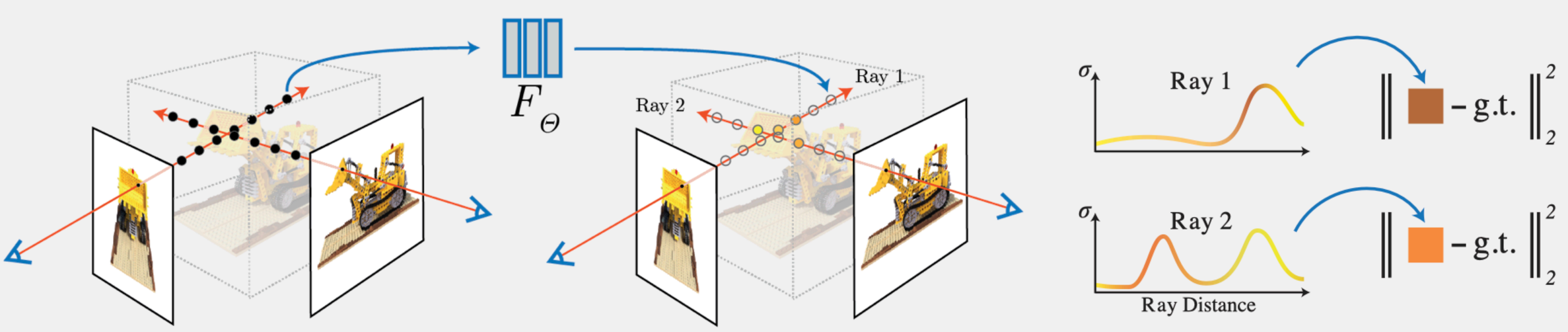
NERF is more or less an occupancy (well opacity) field + color

Basic model : color + opacity for each x, y, z

x,y,z -> r,g, b, alphas

Optionally directional, but let's not worry about it for now.

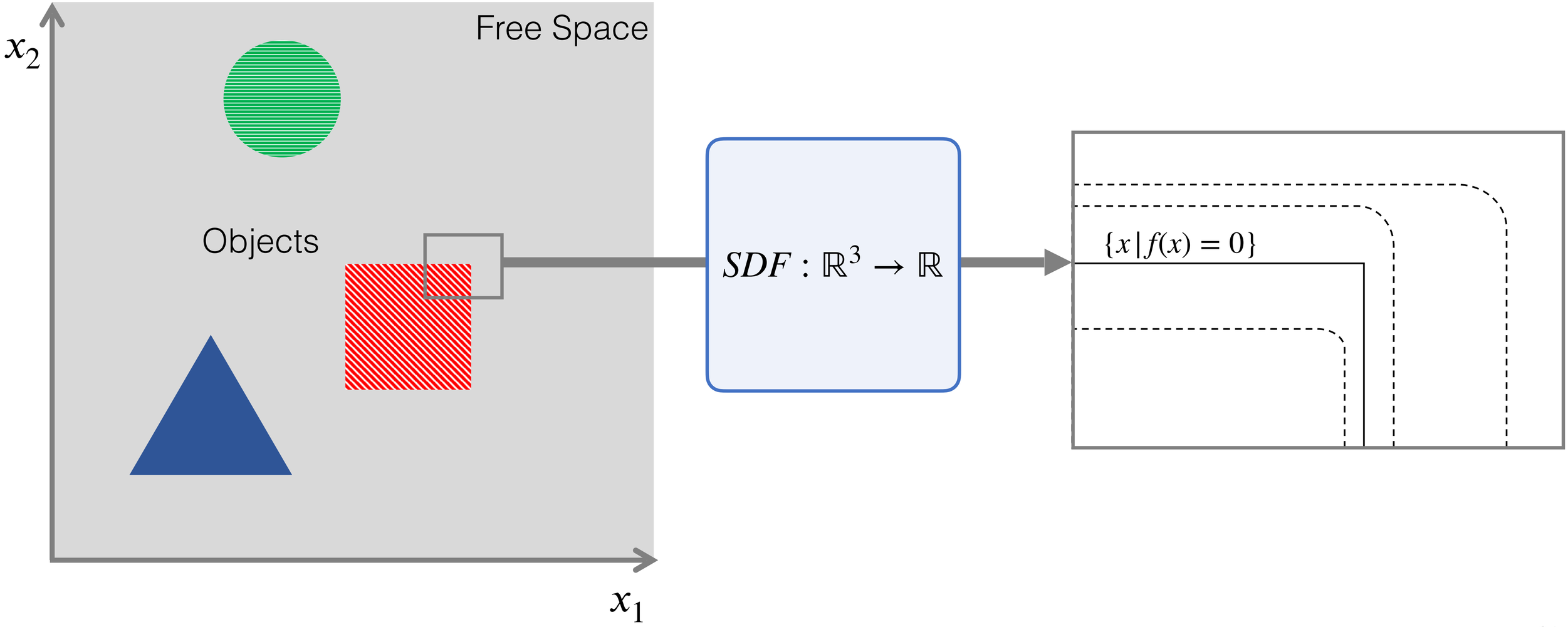
X, y, z, theta, phi -> r, g, b, alpha



We synthesize views by querying 5D coordinates along camera rays and use classic volume rendering techniques to project the output colors and densities into an image. Because volume rendering is naturally differentiable, the only input required to optimize our representation is a set of images with known camera poses. We describe how to effectively optimize neural radiance fields to render photorealistic novel views of scenes with complicated geometry and appearance, and demonstrate results that outperform prior work on neural rendering and view synthesis.

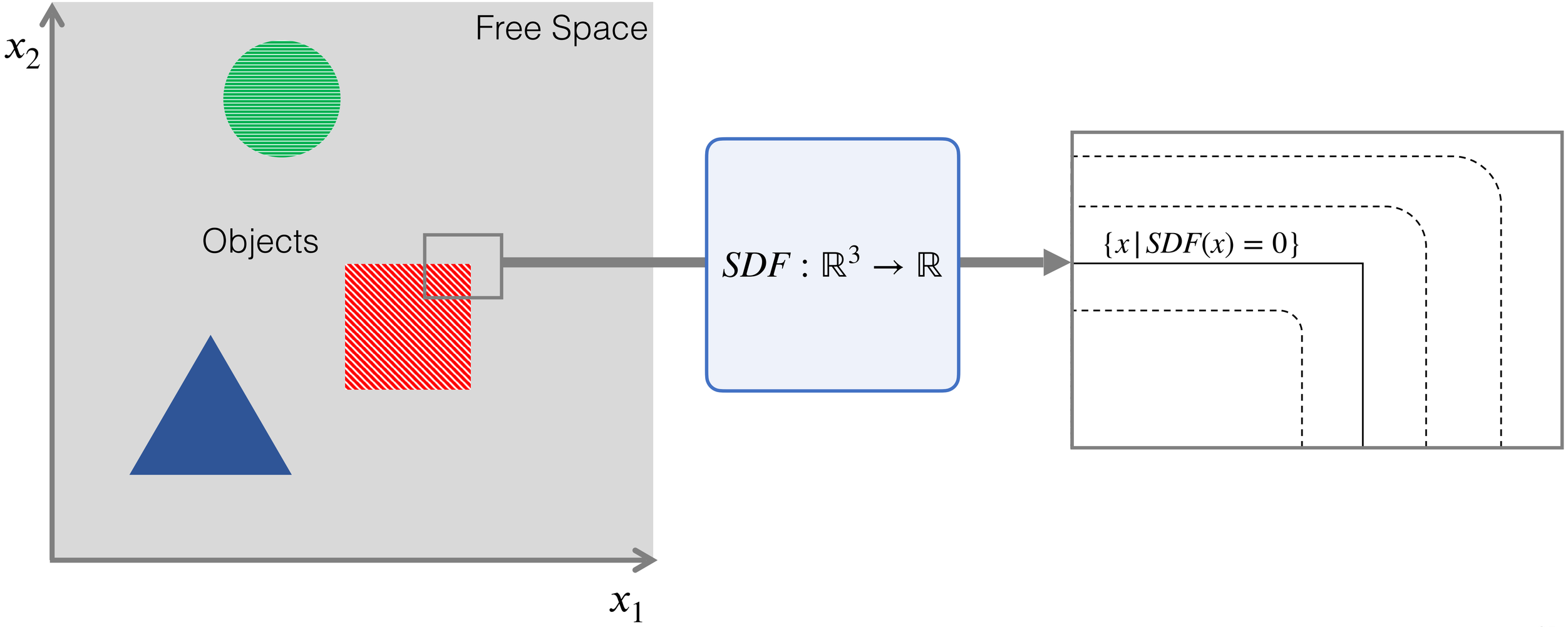
Option 2: Implicit surface

Iso surface of a field



Special case of implicit: Signed Distance Field

Field represents distance to surface

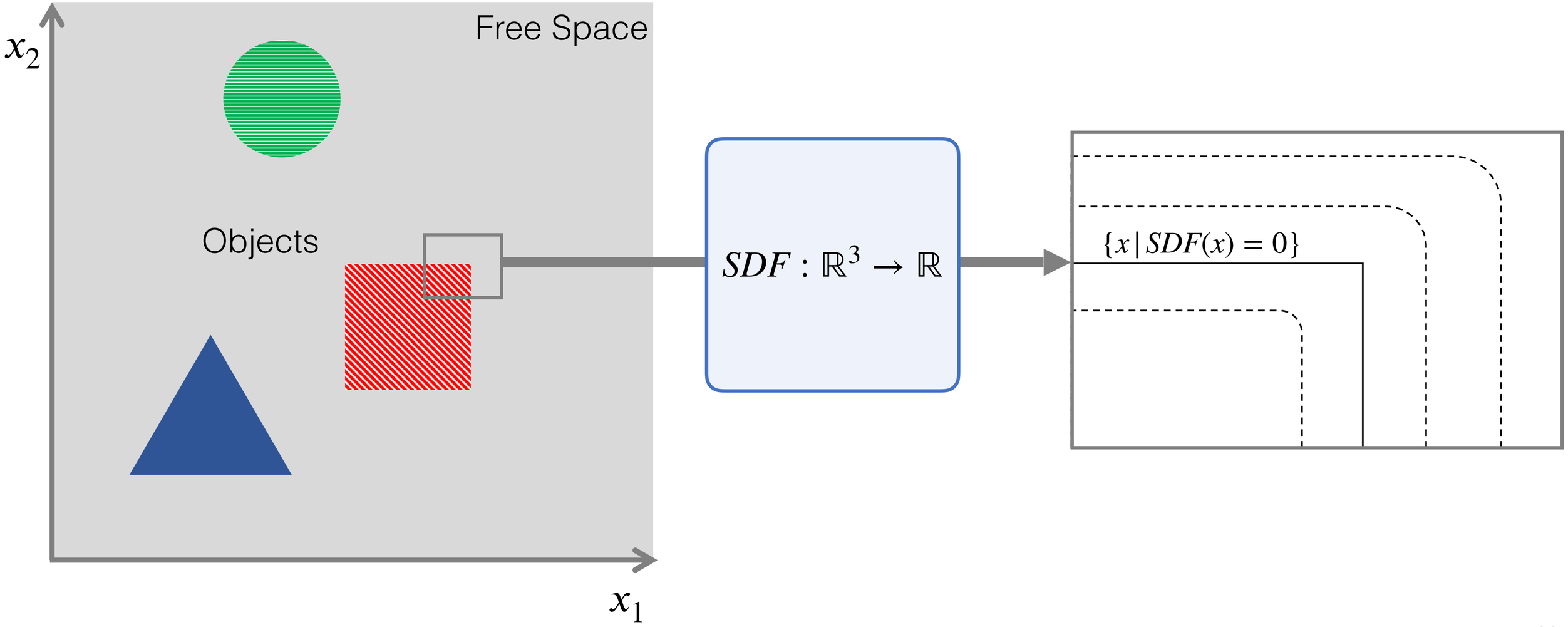


DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation, Park et al. 2019

The Signed Distance Field

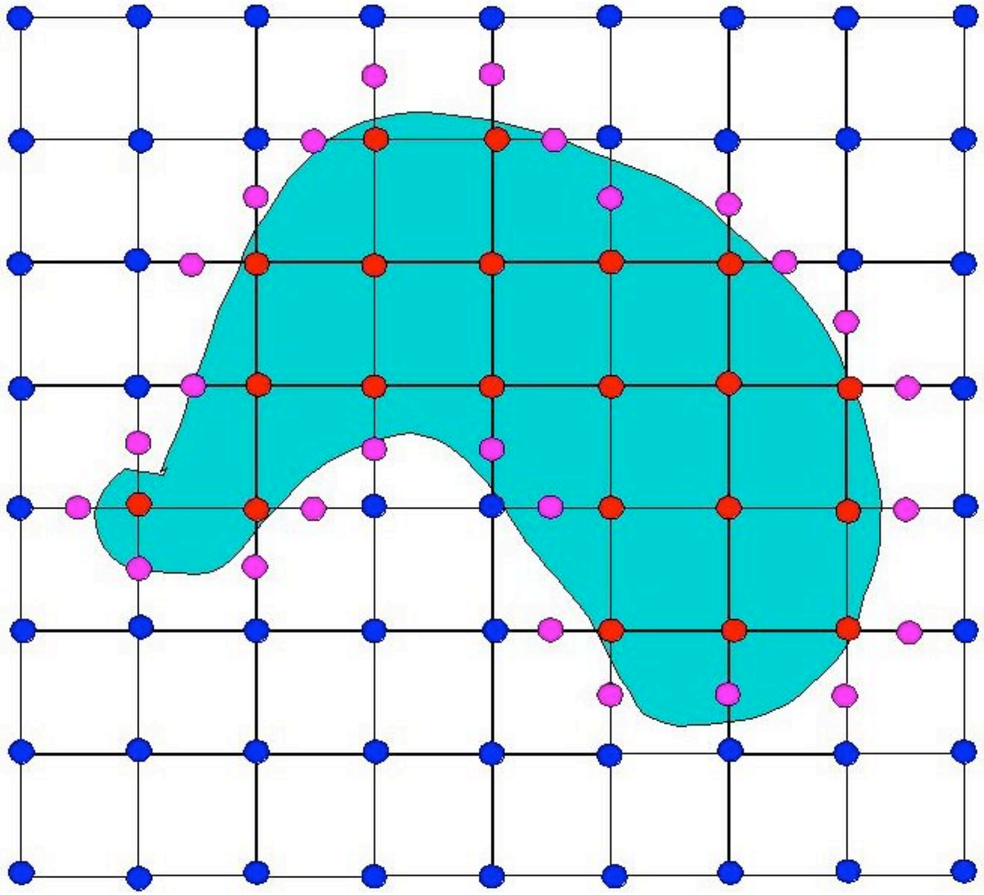
Make a lot of stuff easier

But needs to respect a so-called Eikonal equation $|\nabla f| = 1$

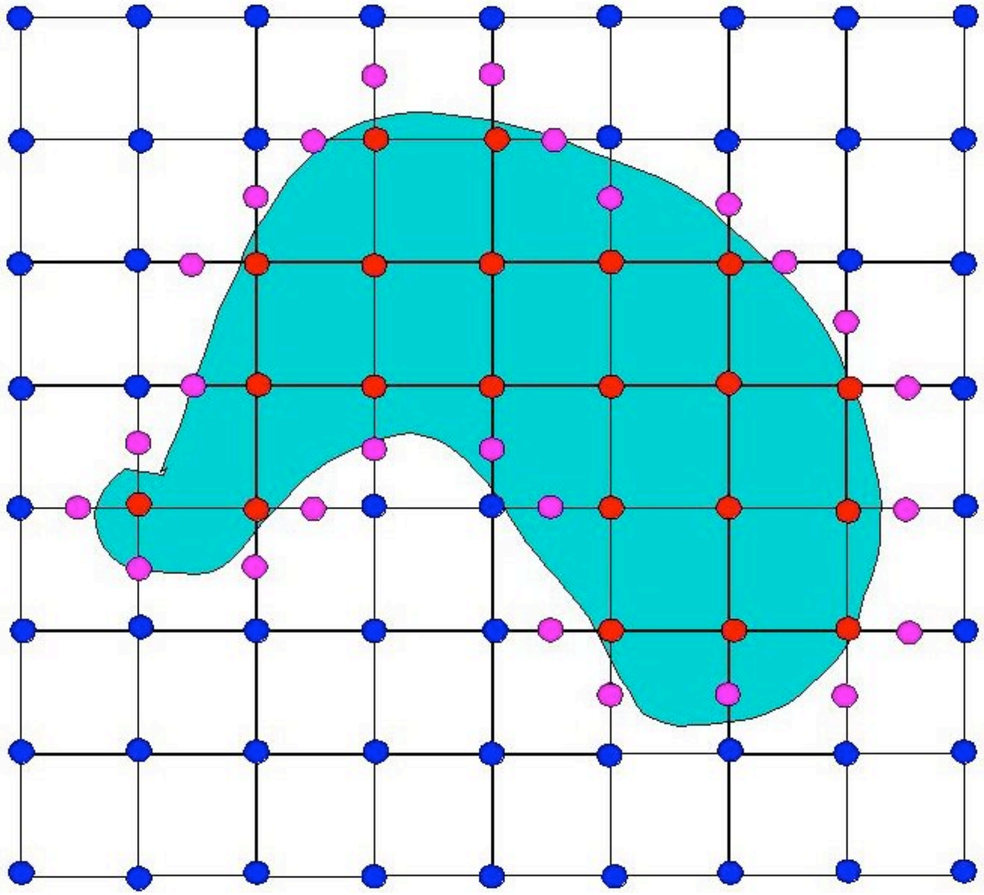


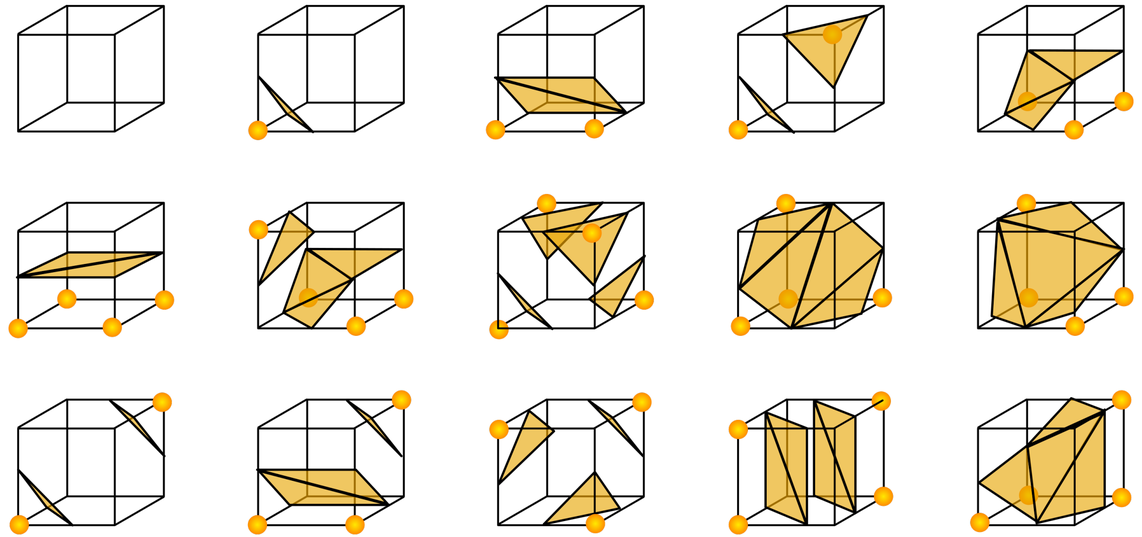
DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation, Park et al. 2019

Marching Cubes: Extracting 2D Mesh from 3D Implicit Representation

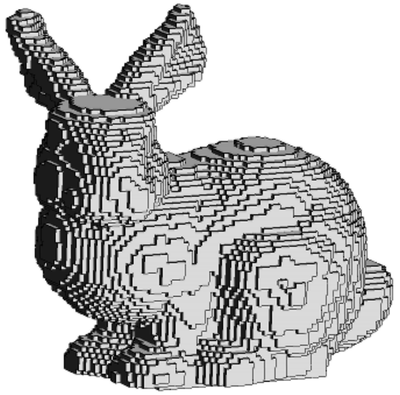


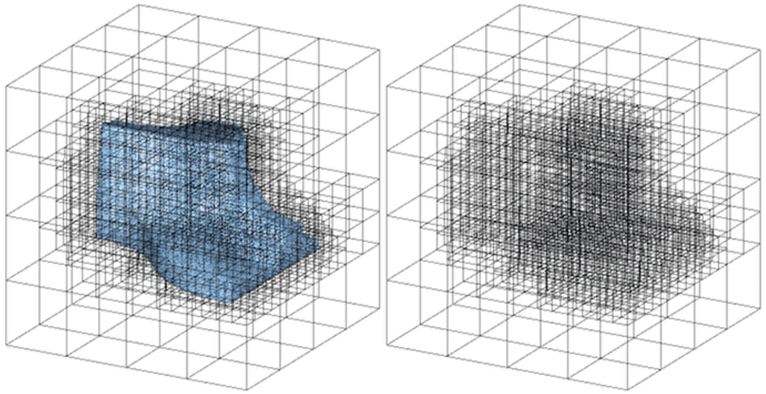
Marching Cubes: Extracting 2D Mesh from 3D Implicit Representation



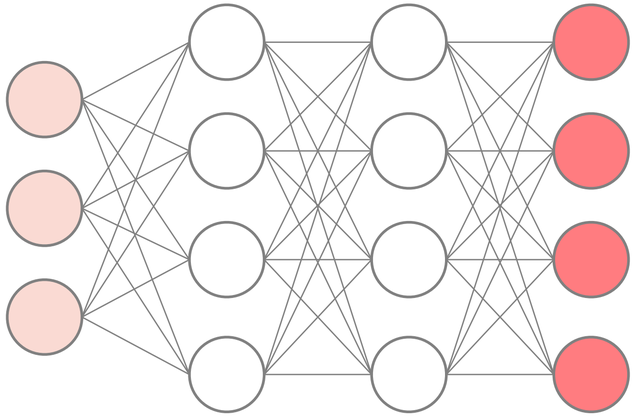


Parameterizing Fields

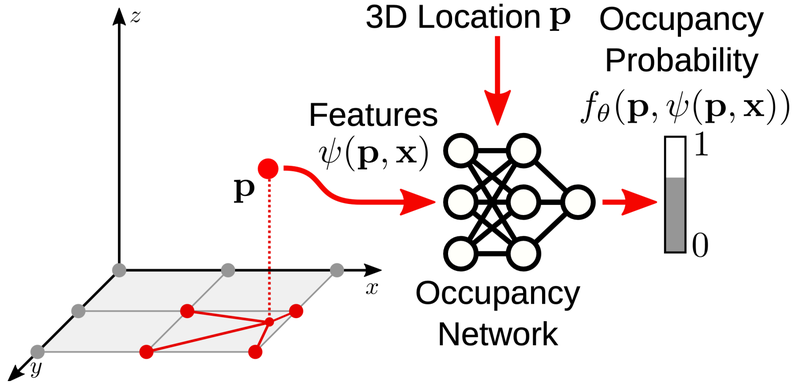


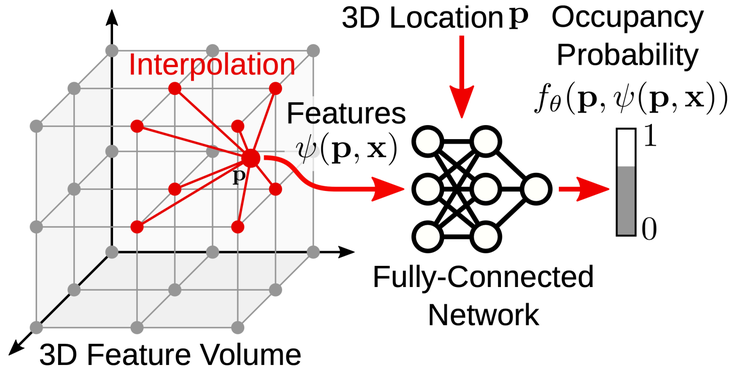


Discrete Parameterizations



Continuous Parameterizations

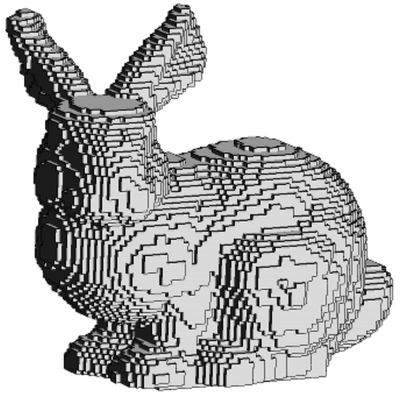


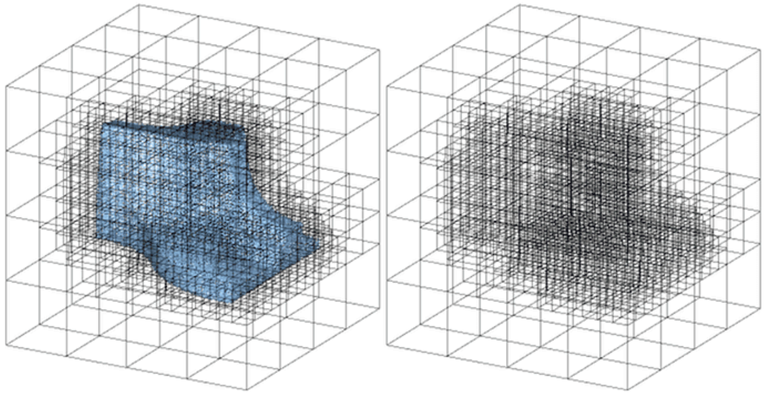


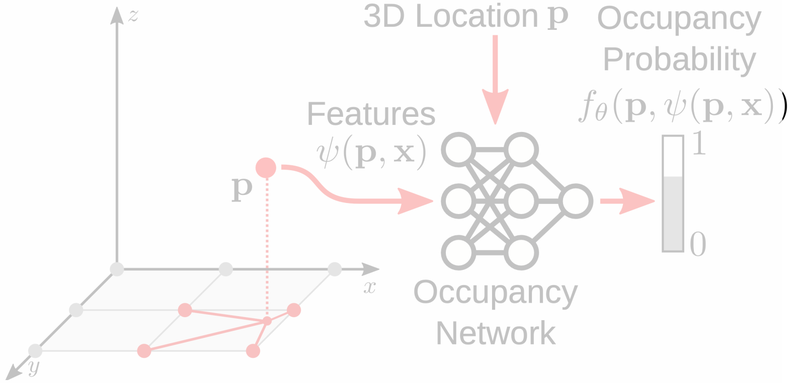
Hybrid Parameterizations

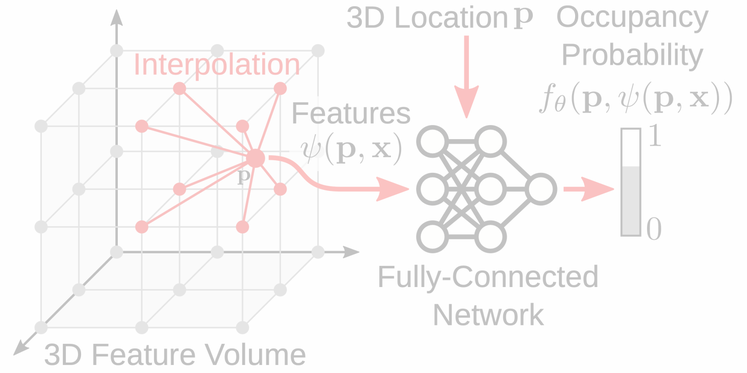
Discrete and grid-based

Voxel Grids









Discrete Parameterizations

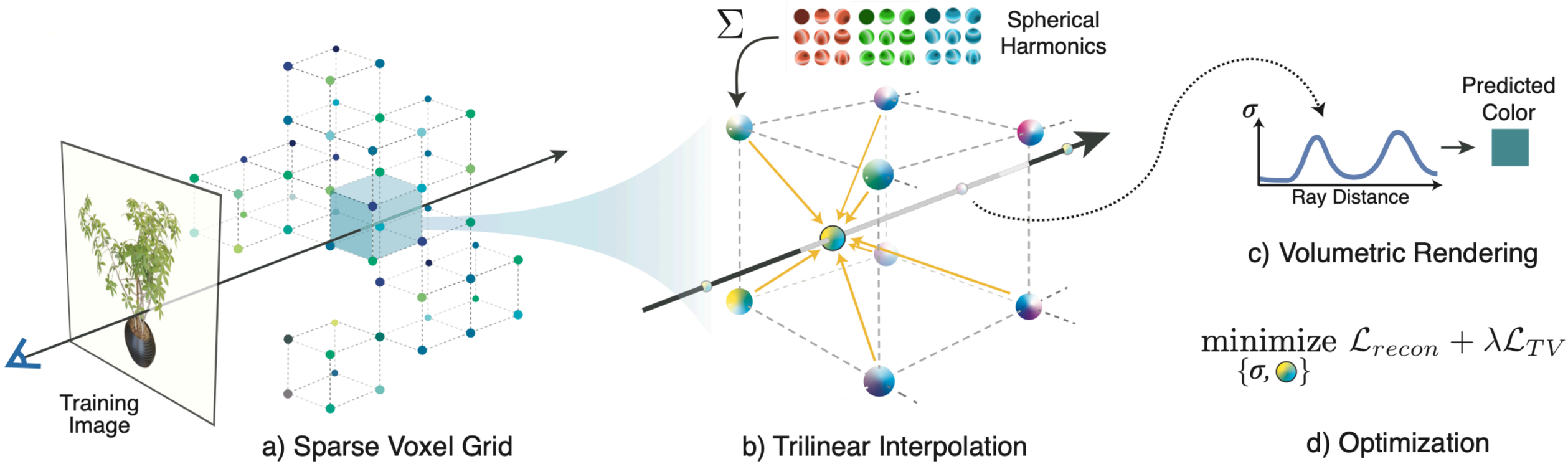
Continuous Parameterizations

Hybrid Parameterizations

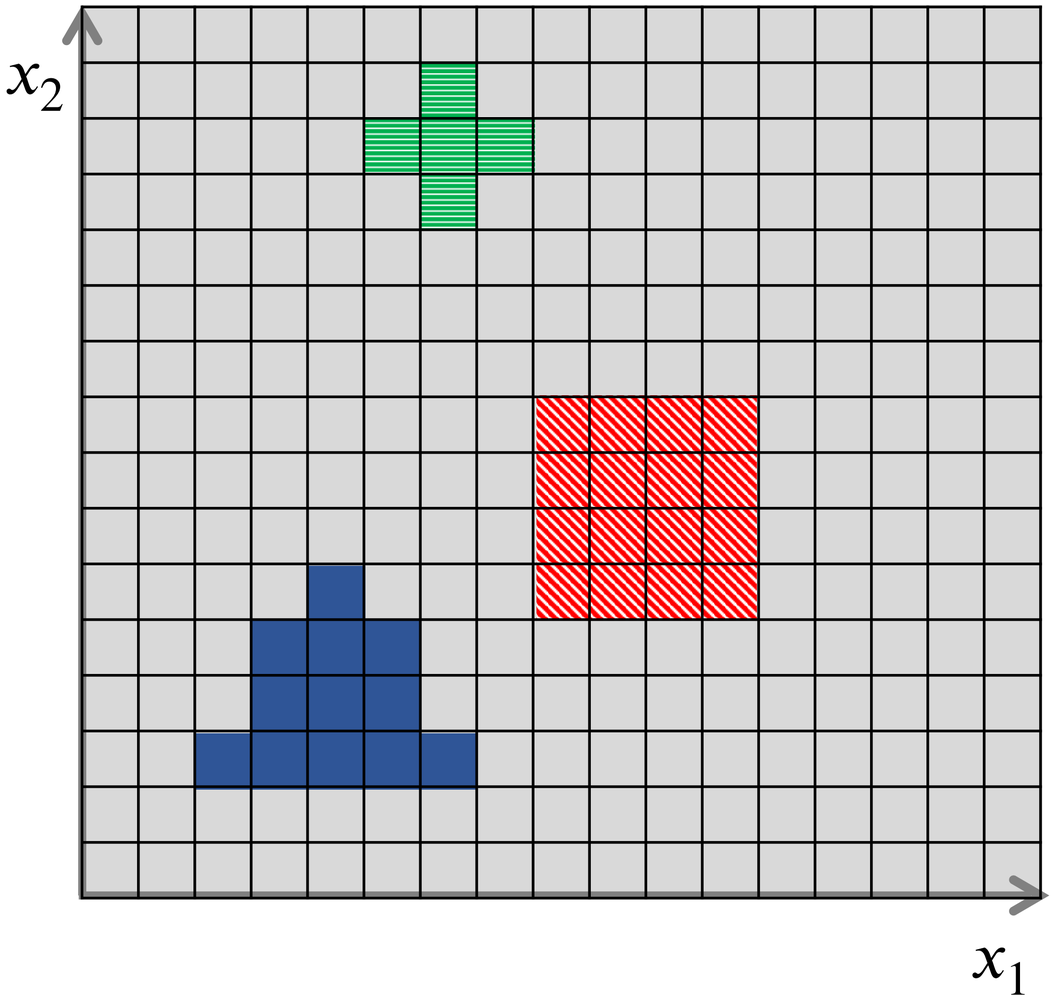
Plenoxels: No neural networks

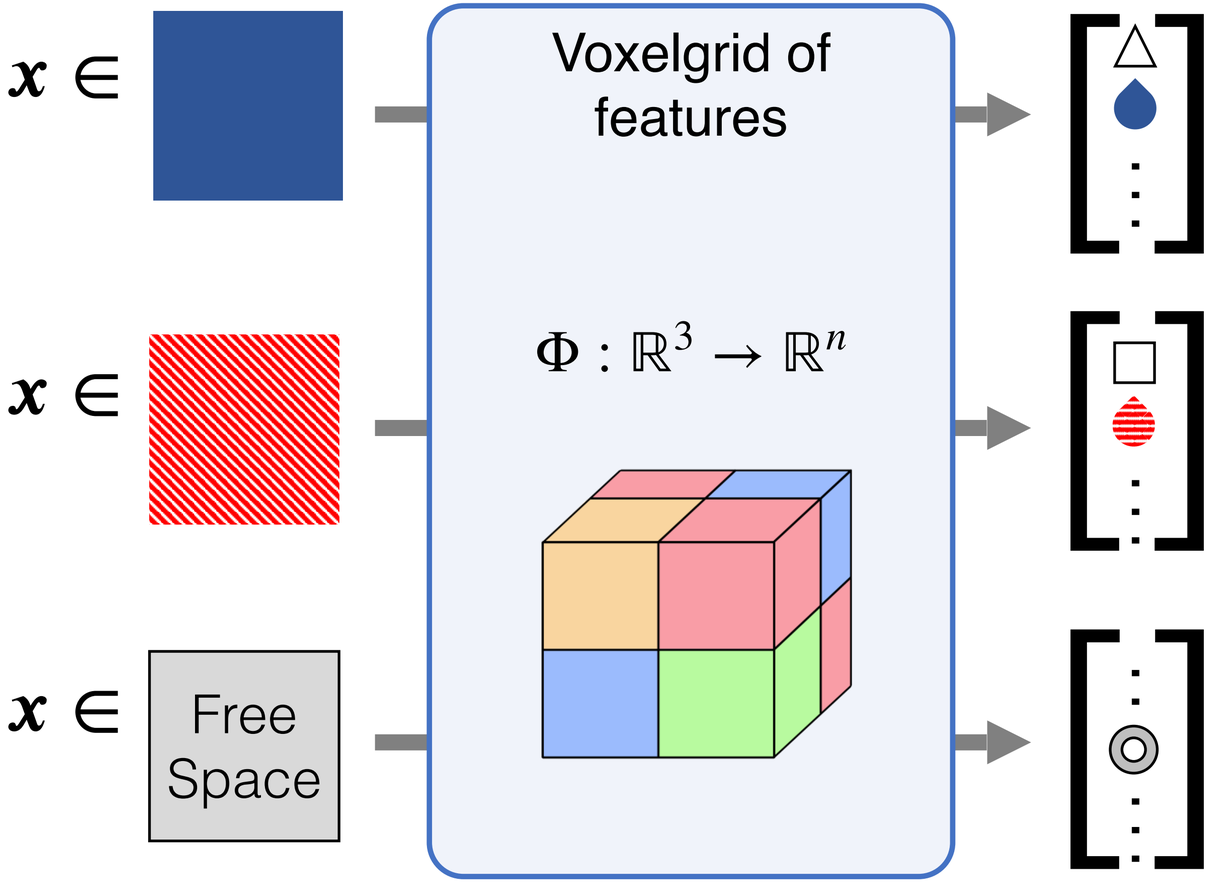
Plenoxels: Radiance Fields without Neural Networks

Alex Yu* Sara Fridovich-Keil* Matthew Tancik Qinhong Chen
Benjamin Recht Angjoo Kanazawa

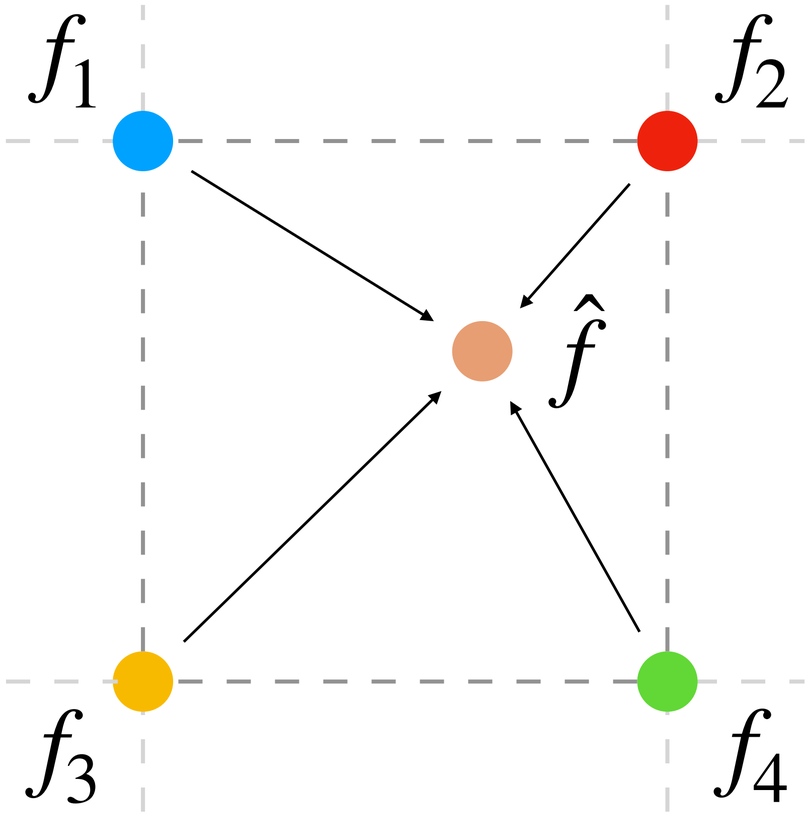


Voxel Grids


x_2
x_1


$x \in$
$x \in$
$x \in$
Free Space
Voxelgrid of features
$\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}^n$

Interpolation: Querying *continuous* values



Interpolation in a 2D voxel grid.

- Only stores values at vertex locations, i.e. corners
- Values at intermediate coordinates are defined via interpolation w/ some kernel k

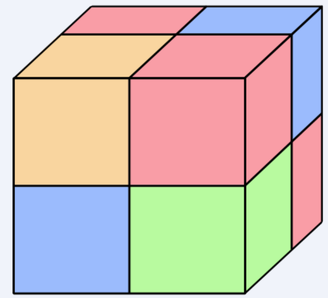
$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{N} f_i k(\mathbf{x}, \mathbf{x}_i)$$

Voxel grids

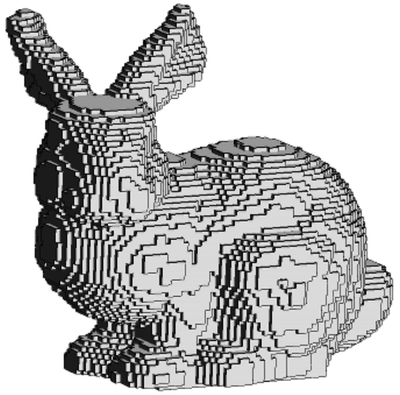
- For d spatial dimensions and resolution n, memory grows $O(n^d)$
- Fast sampling: d-linear interpolation
  = index into array 2^d times & weighted sum.
- Convenient processing as it exposes *locality.* Can easily run convolutions, nearest-neighbor lookups, etc.
- Intractable in higher dimensions :/

Voxelgrid of features

$$\Phi : \mathbb{R}^3 \to \mathbb{R}^n$$



Multi-Resolution Representations



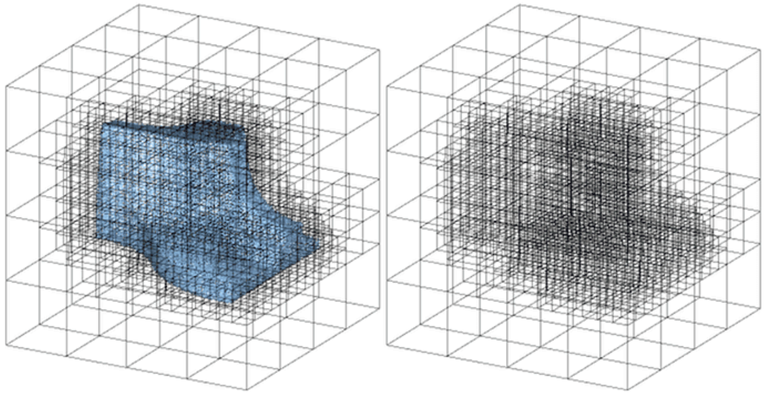
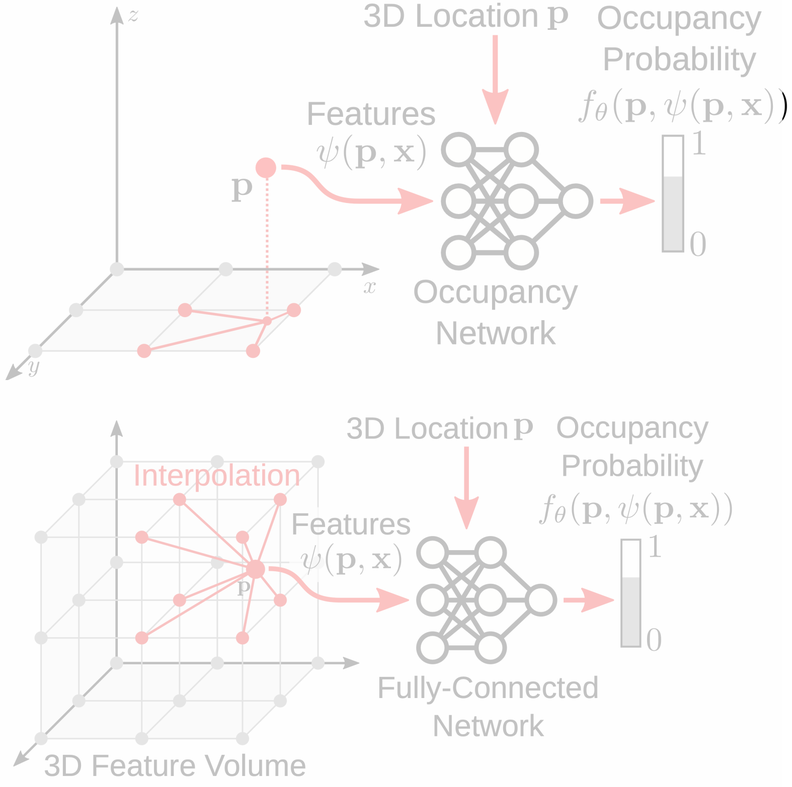


Image Source: https://doc.cgal.org/latest/Orthtree/index.html

Discrete Parameterizations

Continuous Parameterizations



Hybrid Parameterizations

Octrees

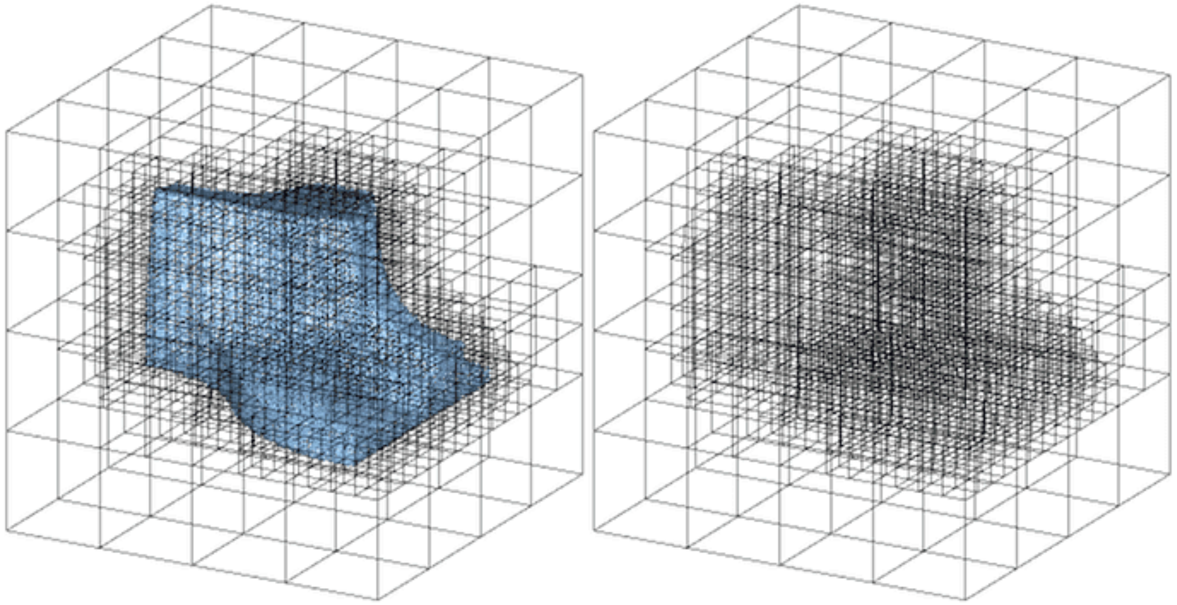


Image Source: https://doc.cgal.org/latest/Orthtree/index.html

Divide volume into 8 voxels

If any box has more than *N* points in it, divide it into 8 voxels

Do not divide voxels with zero points.

Octrees

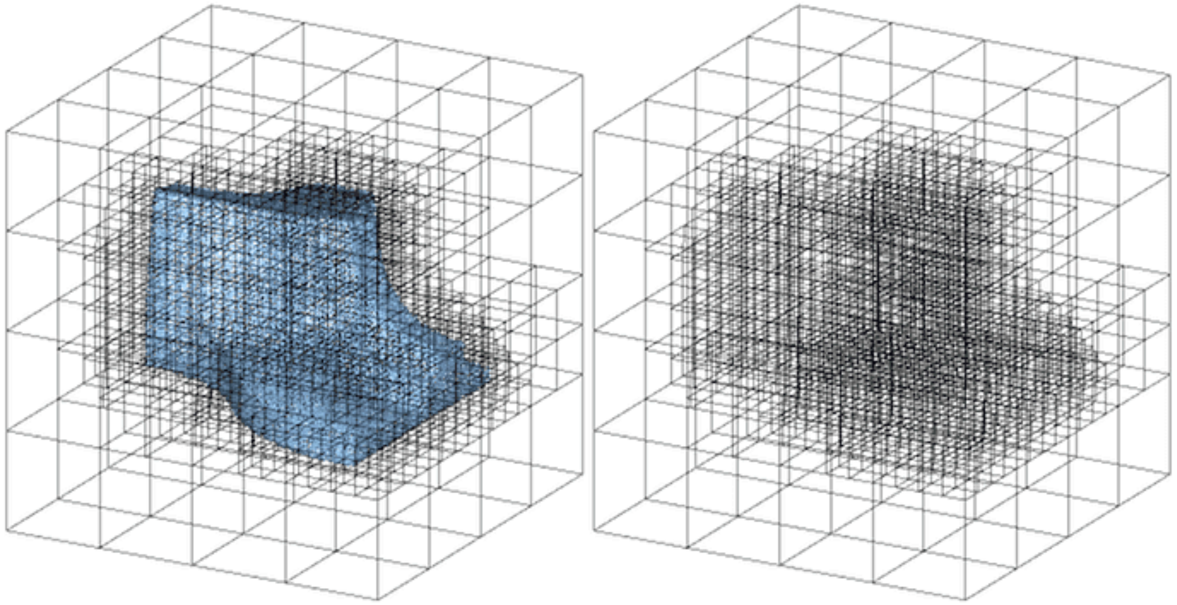


Image Source: https://doc.cgal.org/latest/Orthtree/index.html

Saves memory (less resolution in empty parts)

Sub-division is not differentiable: can’t easily build neural network that outputs Octree

To sample point, have to traverse multi-resolution hierarchy. Still fast, though.

Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics

Sarah F. Frisken, Ronald N. Perry, Alyn P. Rockwood, and Thouis R. Jones
MERL – Mitsubishi Electric Research Laboratory

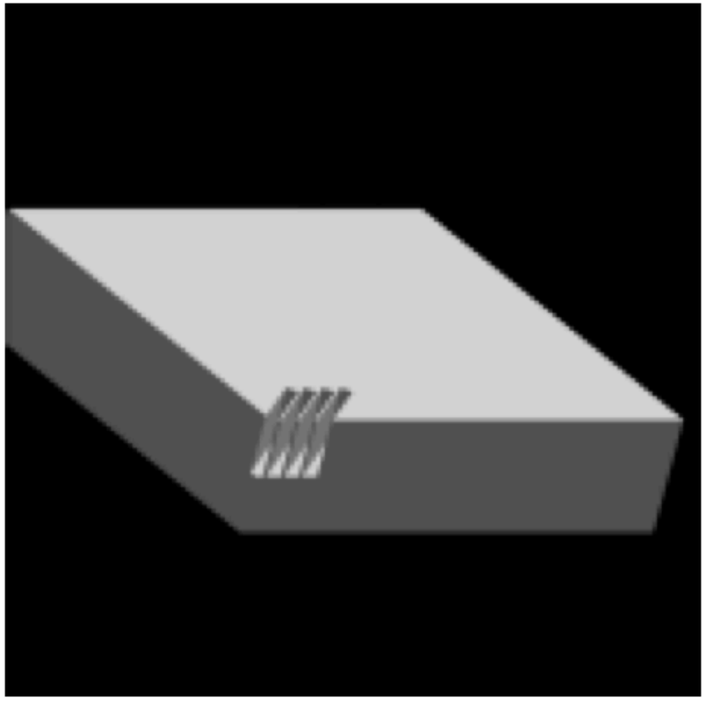


Figure 1. An ADF showing fine detail carved on a rectangular slab with a flat-edged chisel.



Figure 2. Artistic carving of a high order surface with a rounded chisel.

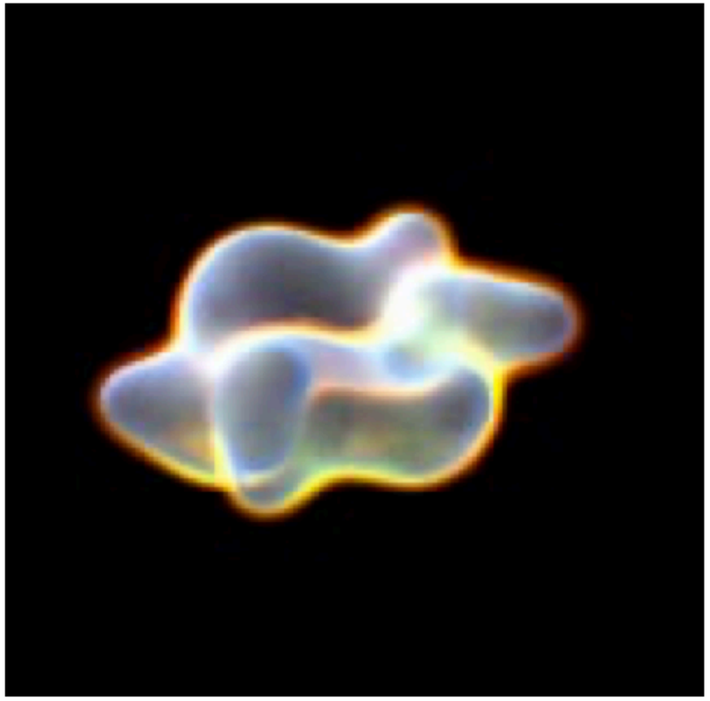
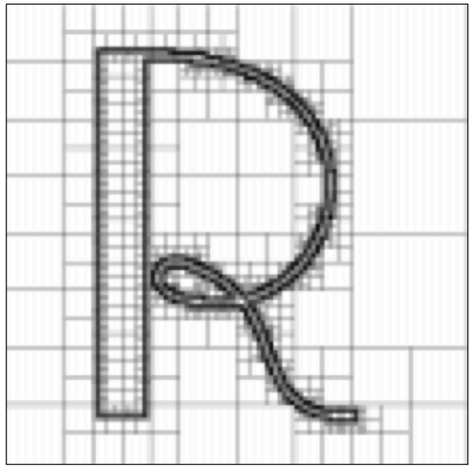
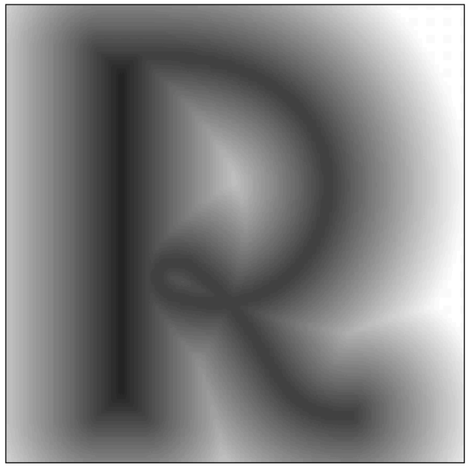


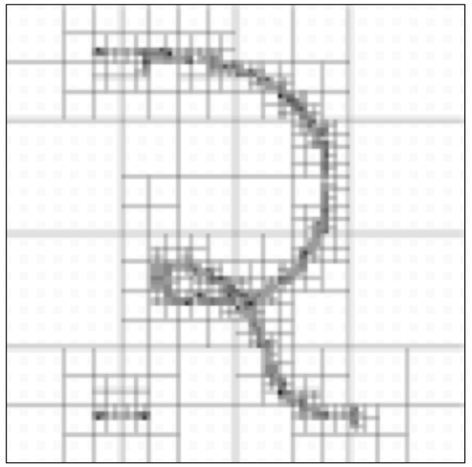
Figure 3. A semi-transparent electron probability distribution of a cyclohexane molecule.





Figures 4a "R" and **4b** 3-color quadtree containing 23,573 cells.





Figures 4c Distance field of "R" and **4d** ADF containing 1713 cells.

Error Tolerance	*Triangle Count*	*ADF Cell Count*	*ADF Sample Count*
6.25×10^{-5}	32,768	16,201	24,809
3.13×10^{-5}	131,072	44,681	67,405
1.56×10^{-6}	2,097,152	131,913	164,847

Table 1. Comparison of triangle count for a sphere (r = 0.4) to ADF size.

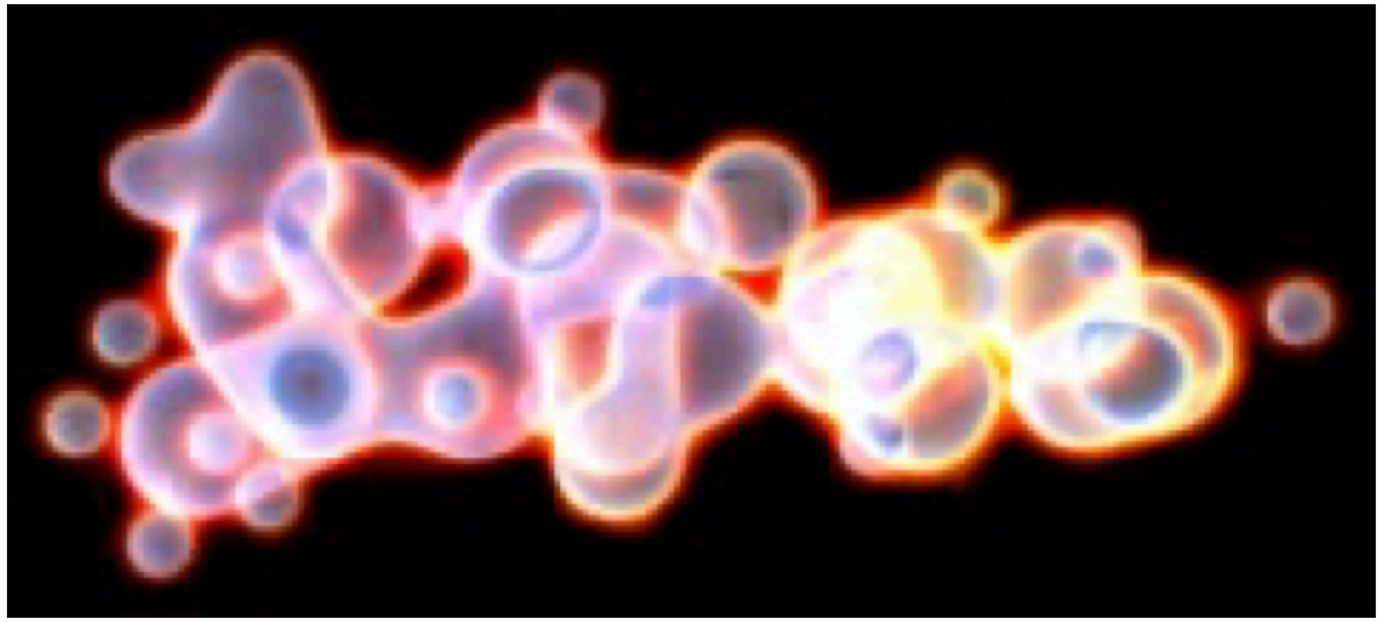


Figure 7. An ADF cocaine molecule volume rendered in a haze of turbulent mist. The mist was generated using a color function dependent on distance from the molecule surface.

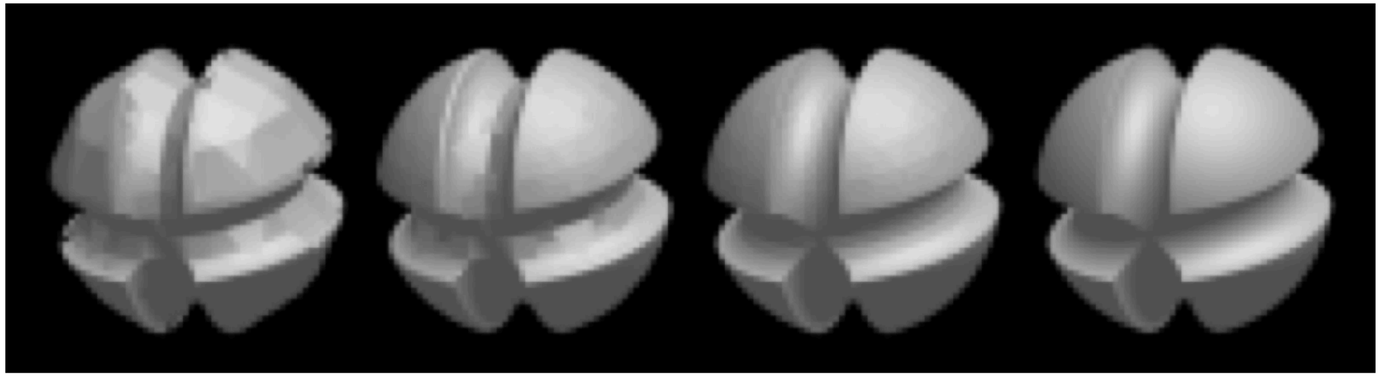
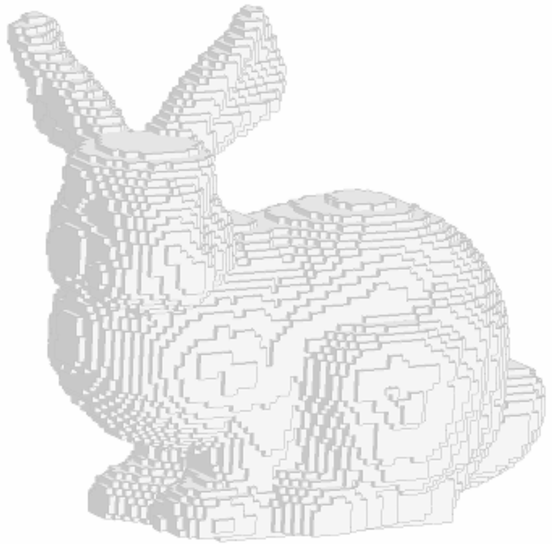


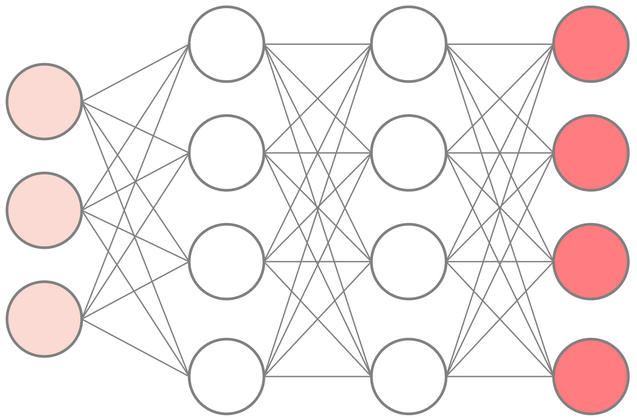
Figure 9. Four LOD models with varying amounts of error rendered from an ADF octree.

Neural fields

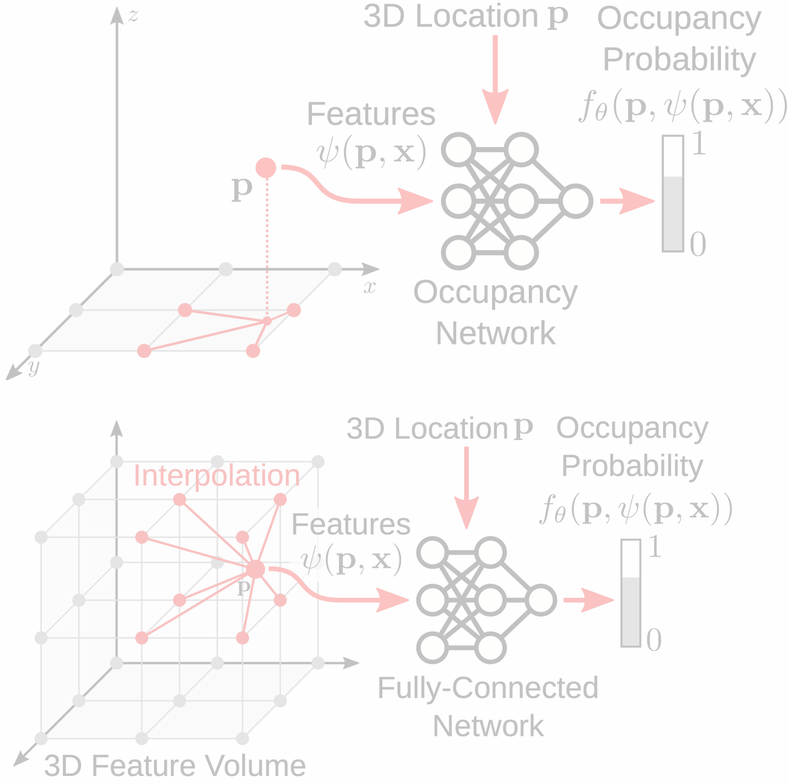
Neural Fields



Discrete Parameterizations

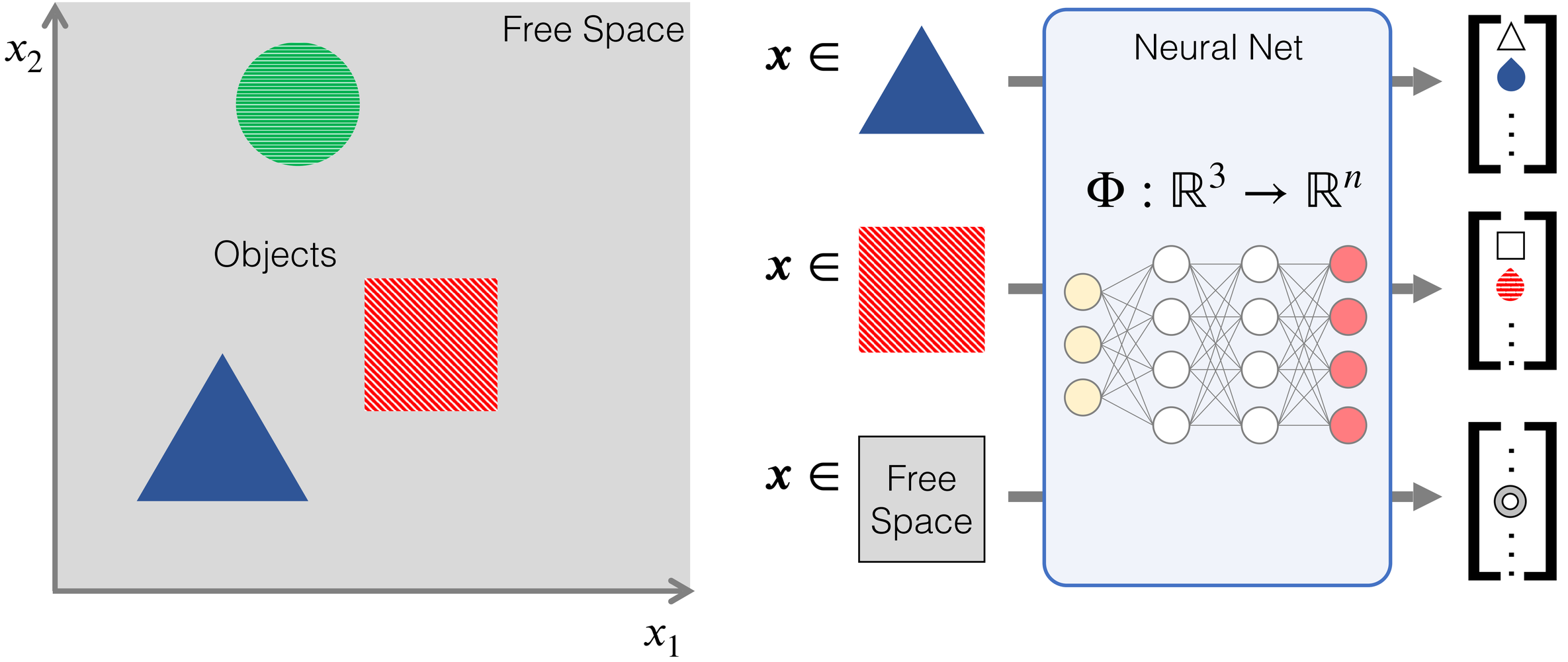


Neural Fields



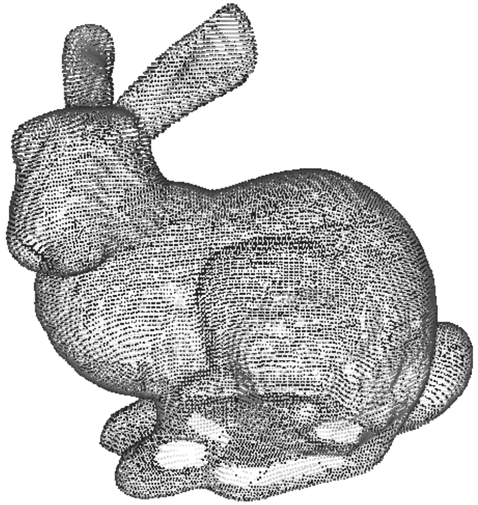
Hybrid Parameterizations

Neural fields: Parameterize Field as Neural Network!

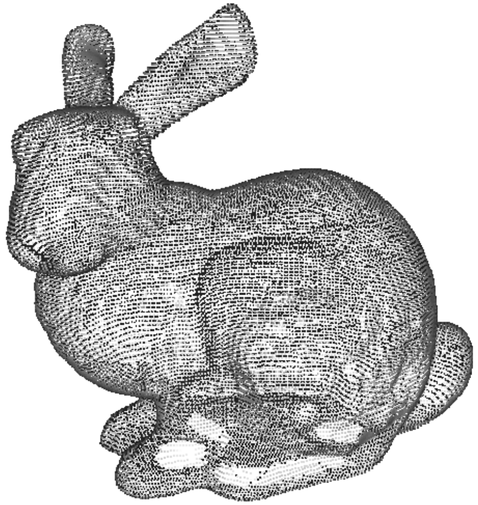


Occupancy Networks: Learning 3D Reconstruction in Function Space, Mescheder et al.
IM-Net: Learning Implicit Fields for Generative Shape Modeling, Chen et al.
DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation, Park et al. 2019
Sitzmann et al: Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations, NeurIPS 2019.

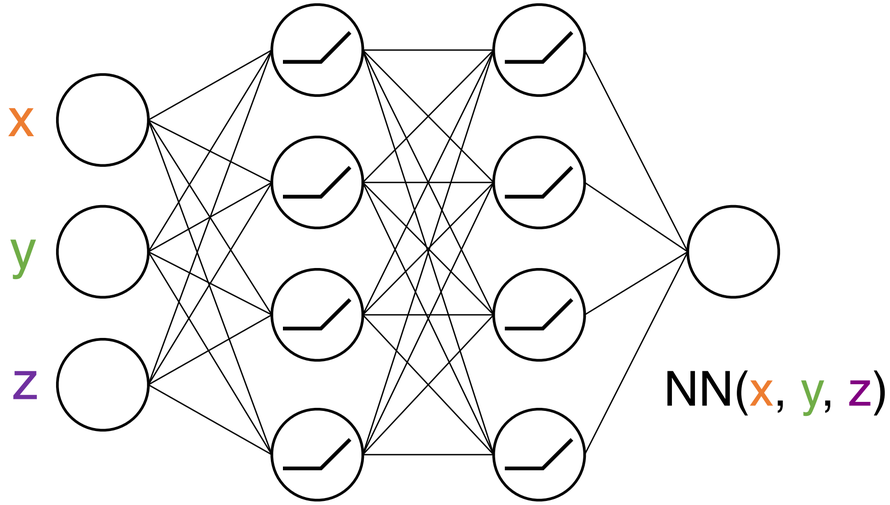
Shapes



Shapes



ReLU MLP



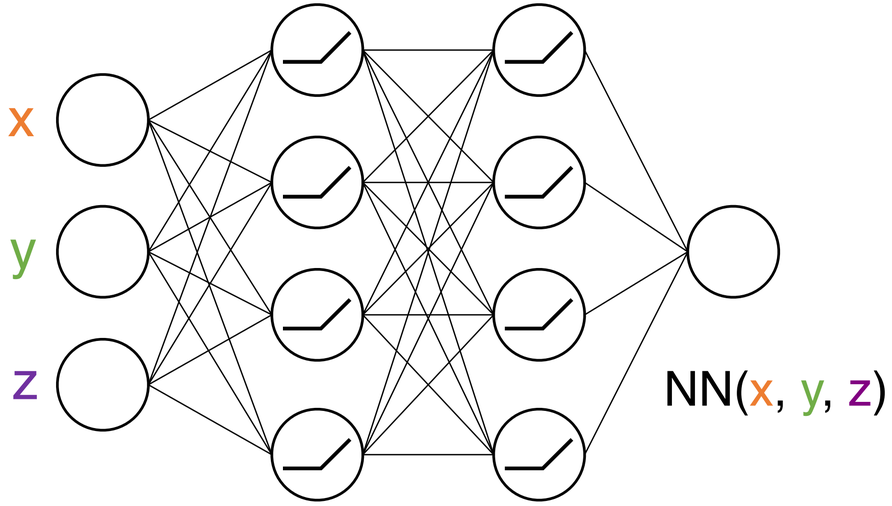


Mescheder et al. 2018
Park et al. 2018
Chen et al. 2018

Shapes



ReLU MLP





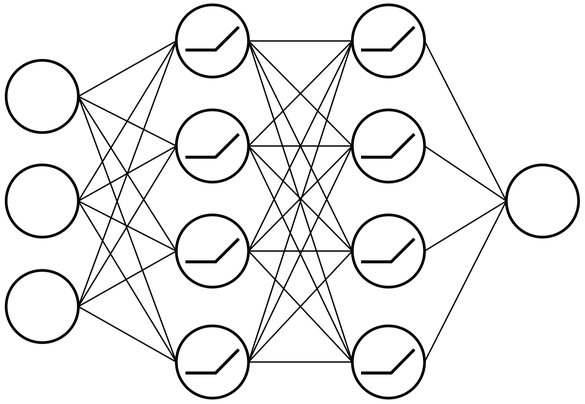
Mescheder et al. 2018
Park et al. 2018
Chen et al. 2018

Resolution is limited!

Shapes



ReLU MLP





Resolution is limited!

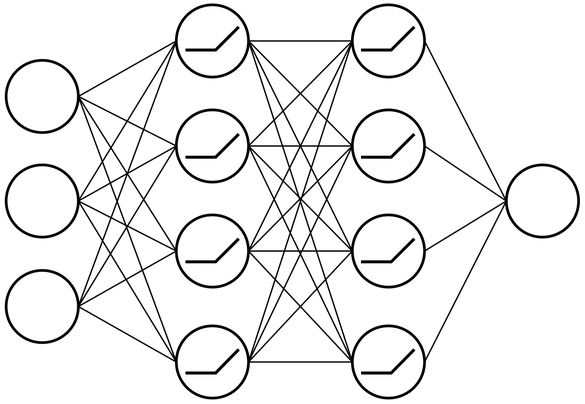
Images

Shapes





ReLU MLP







Resolution is limited!

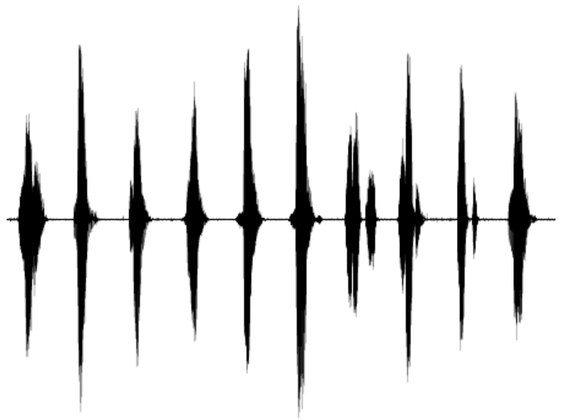
Images

Shapes

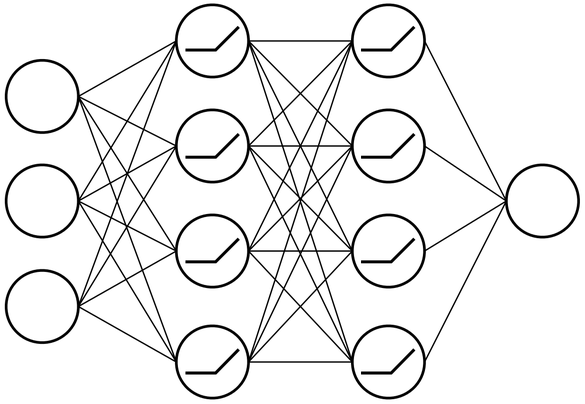
Audio





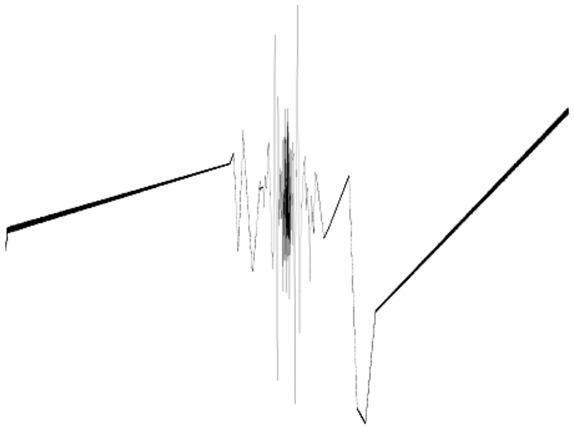


ReLU MLP





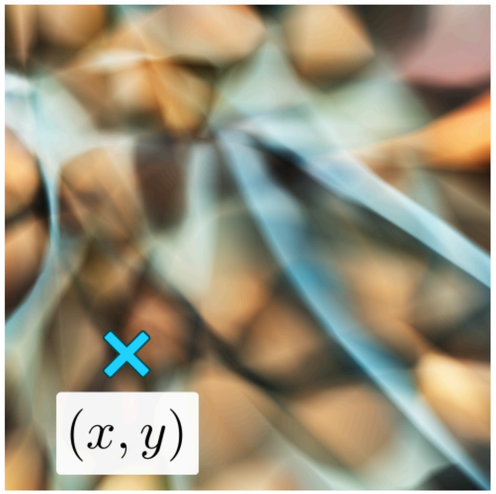




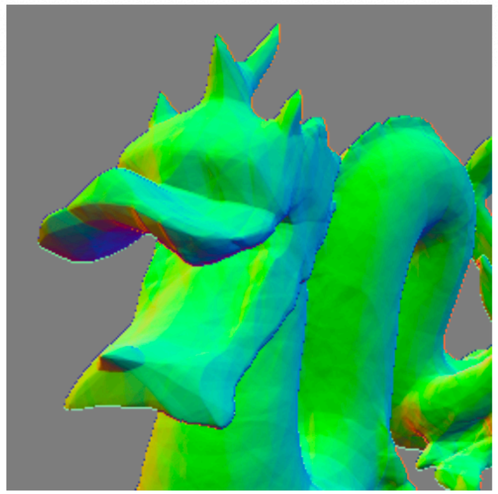
Back to Sine Waves 1: Positional encoding

NERF

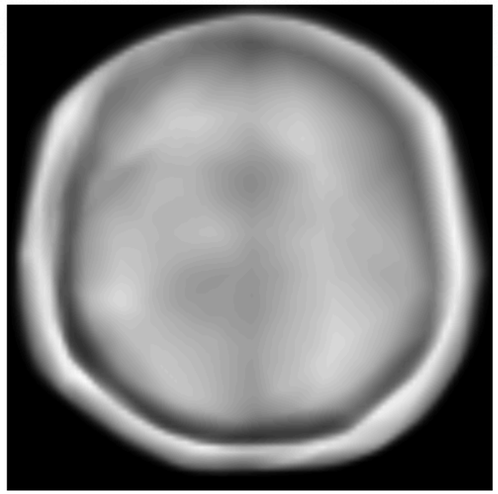
- Naive solution: use x, y, z directly
- Solutions are too blurry



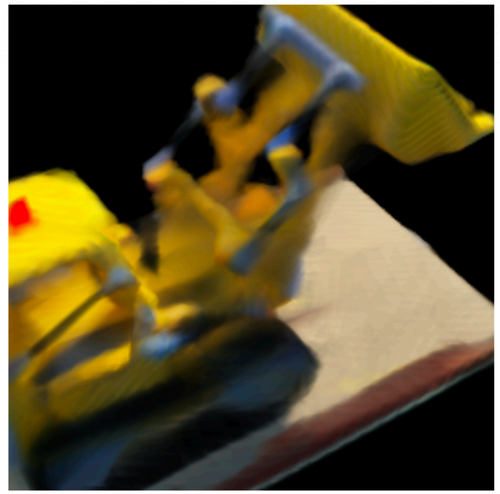
(b) Image regression
$(x,y) \rightarrow$ RGB



(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

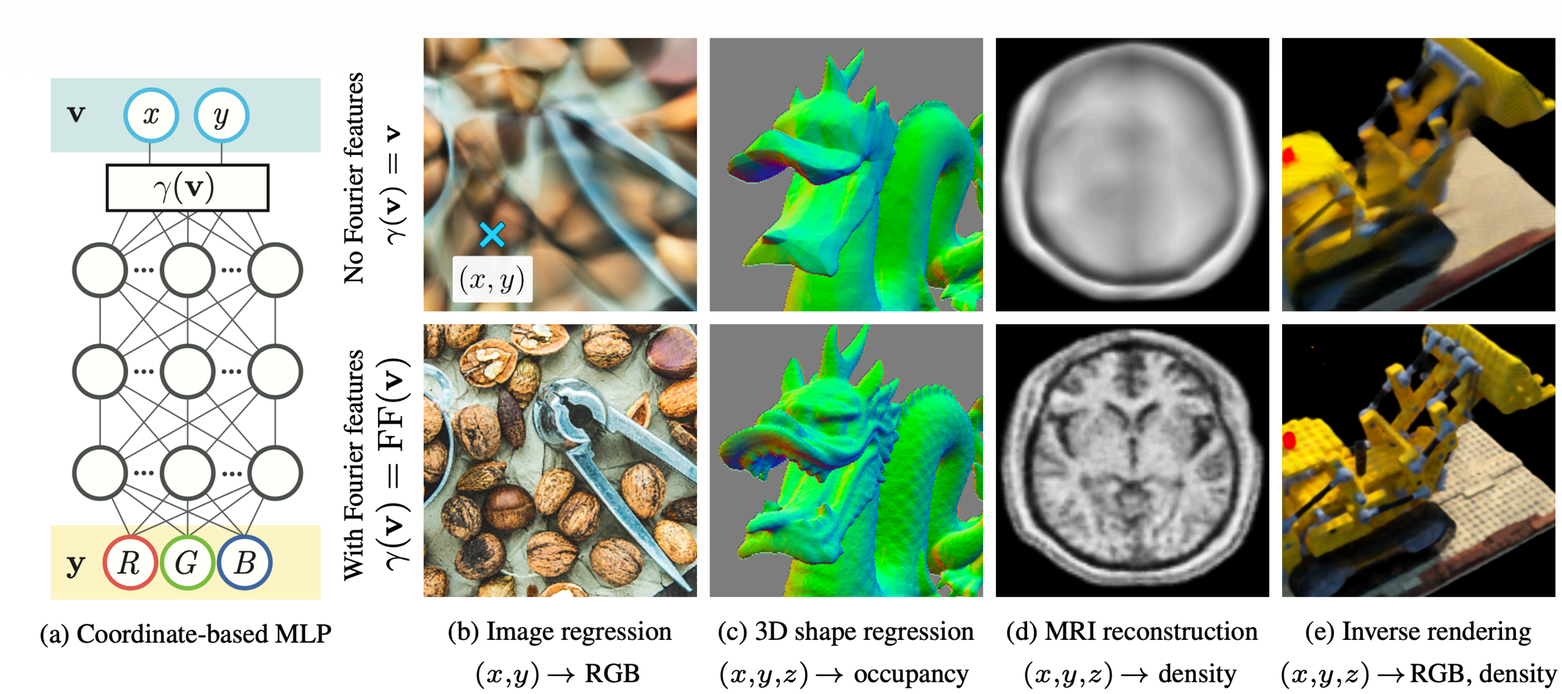
https://arxiv.org/pdf/2006.10739

NERF with sine positional encoding

- Before entering network, turn x, y, z into a bunch of sine/cosine values (similar to transformers)

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

  - Pretty much a version of the so-called kernel trick (careful, the kernel word gets used a lot)



(a) Coordinate-based MLP

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

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

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

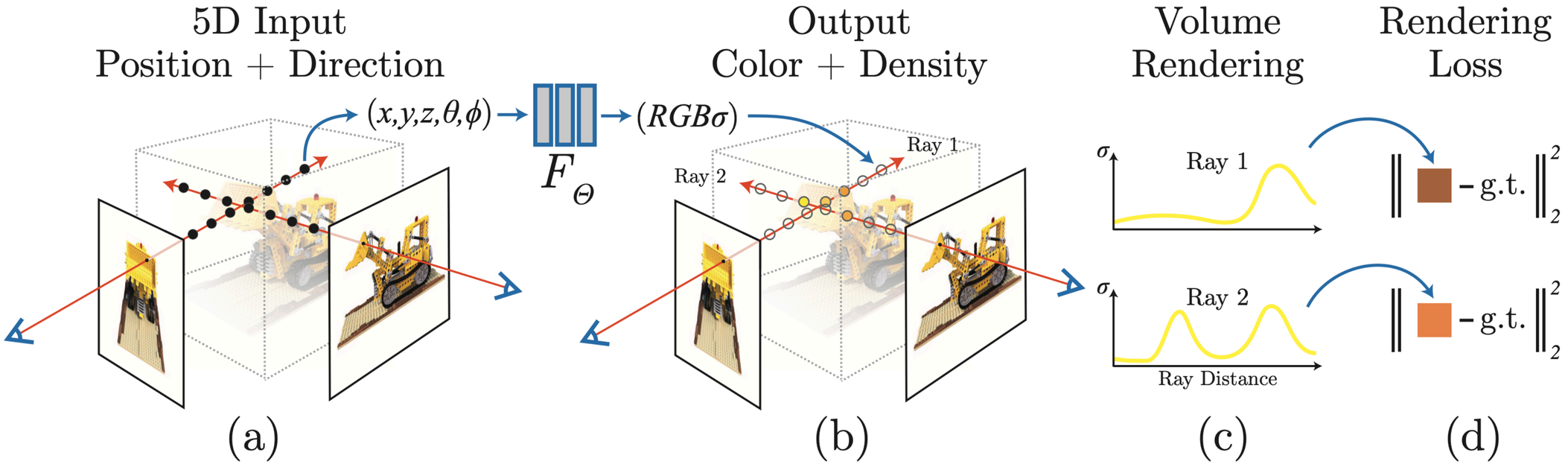
(e) Inverse rendering
$(x,y,z) \rightarrow$ RGB, density

NERF

- Positional ending for both the location x y, z and the unit vector of direction

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

- L=10 for position, 4 for direction



Back to Sine Waves 2: Activations (SIREN)

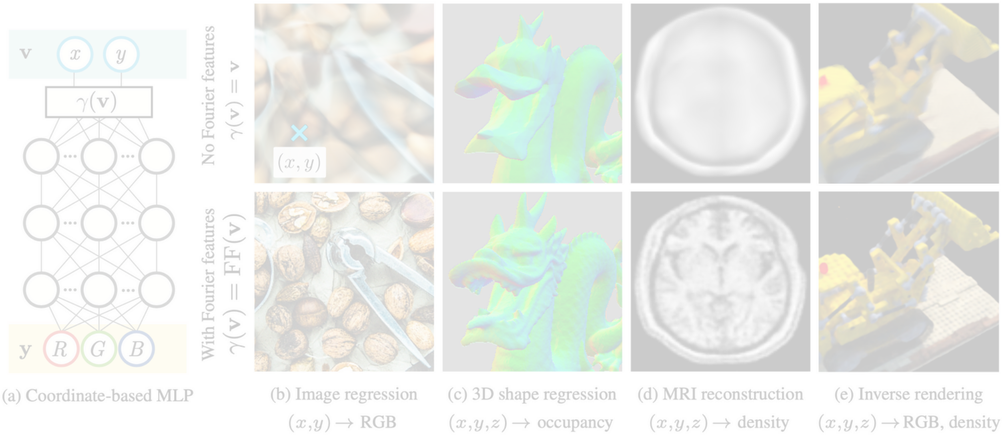
SIREN: Neural Implicit Representations With Periodic Activation Functions

Sitzmann & Martel et al. NeurIPS 2020



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

Tancik, Srinivasan, Mildenhall NeurIPS 2020



(a) Coordinate-based MLP

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

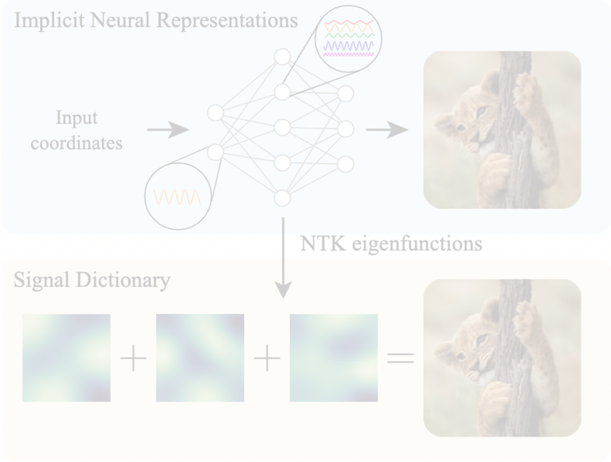
(c) 3D shape regression $(x,y,z) \rightarrow$ occupancy

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

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

A Structured Dictionary Perspective on Implicit Neural Representations

Yüce & Ortiz-Jiménez et al. CVPR 2022



Gaussian Activated Neural Radiance Fields for High Fidelity Reconstruction & Pose Estimation

Chng et al. ECCV 2022

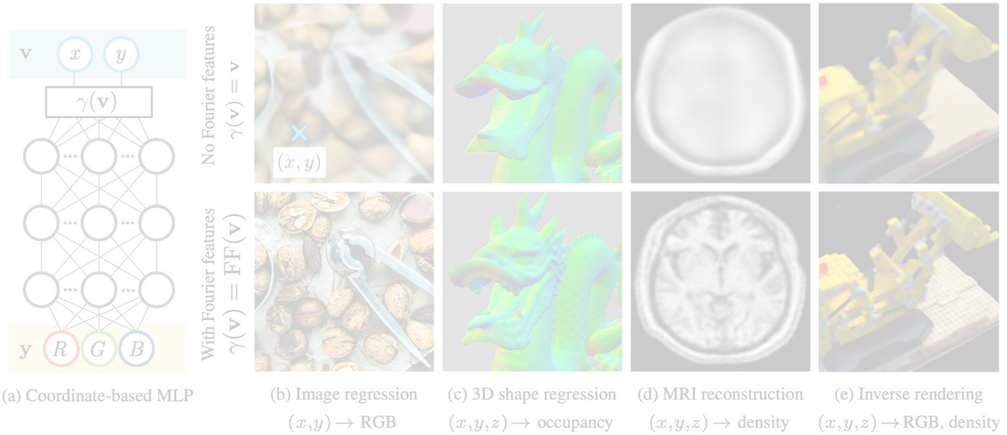
SIREN: Neural Implicit Representations With Periodic Activation Functions

Sitzmann & Martel et al. NeurIPS 2020



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

Tancik, Srinivasan, Mildenhall NeurIPS 2020



(a) Coordinate-based MLP

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

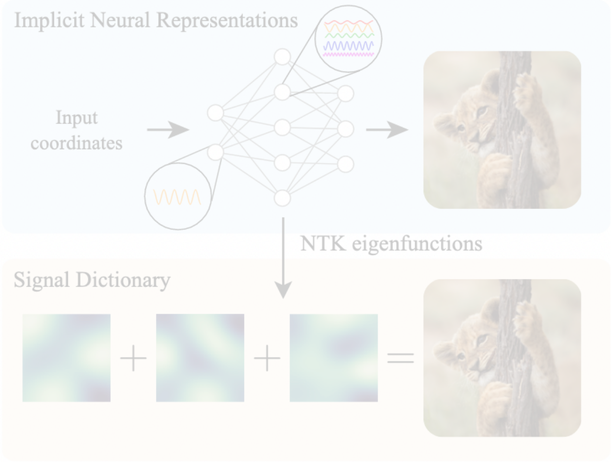
(c) 3D shape regression $(x,y,z) \rightarrow$ occupancy

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

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

A Structured Dictionary Perspective on Implicit Neural Representations

Yüce & Ortiz-Jiménez et al. CVPR 2022



Gaussian Activated Neural Radiance Fields for High Fidelity Reconstruction & Pose Estimation

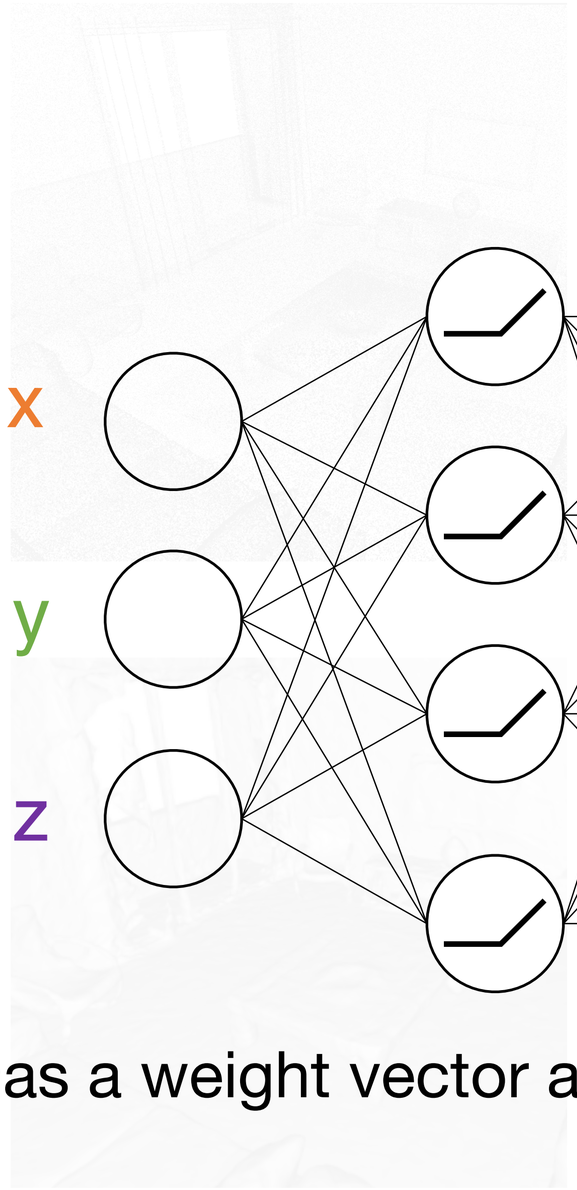
Chng et al. ECCV 2022

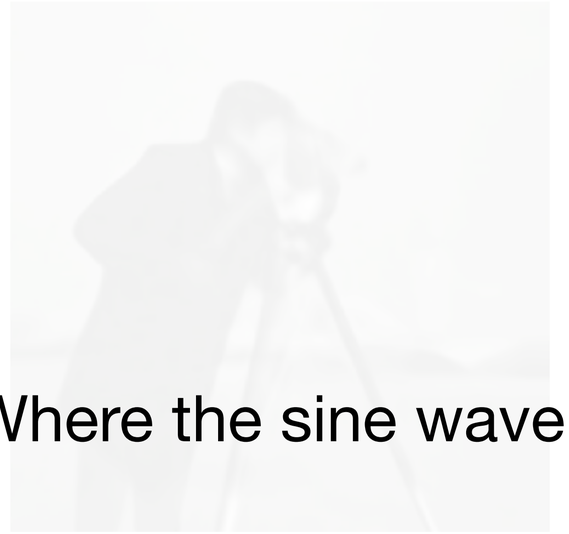
Images

Shapes

Audio

Quantities defined by a differential equation



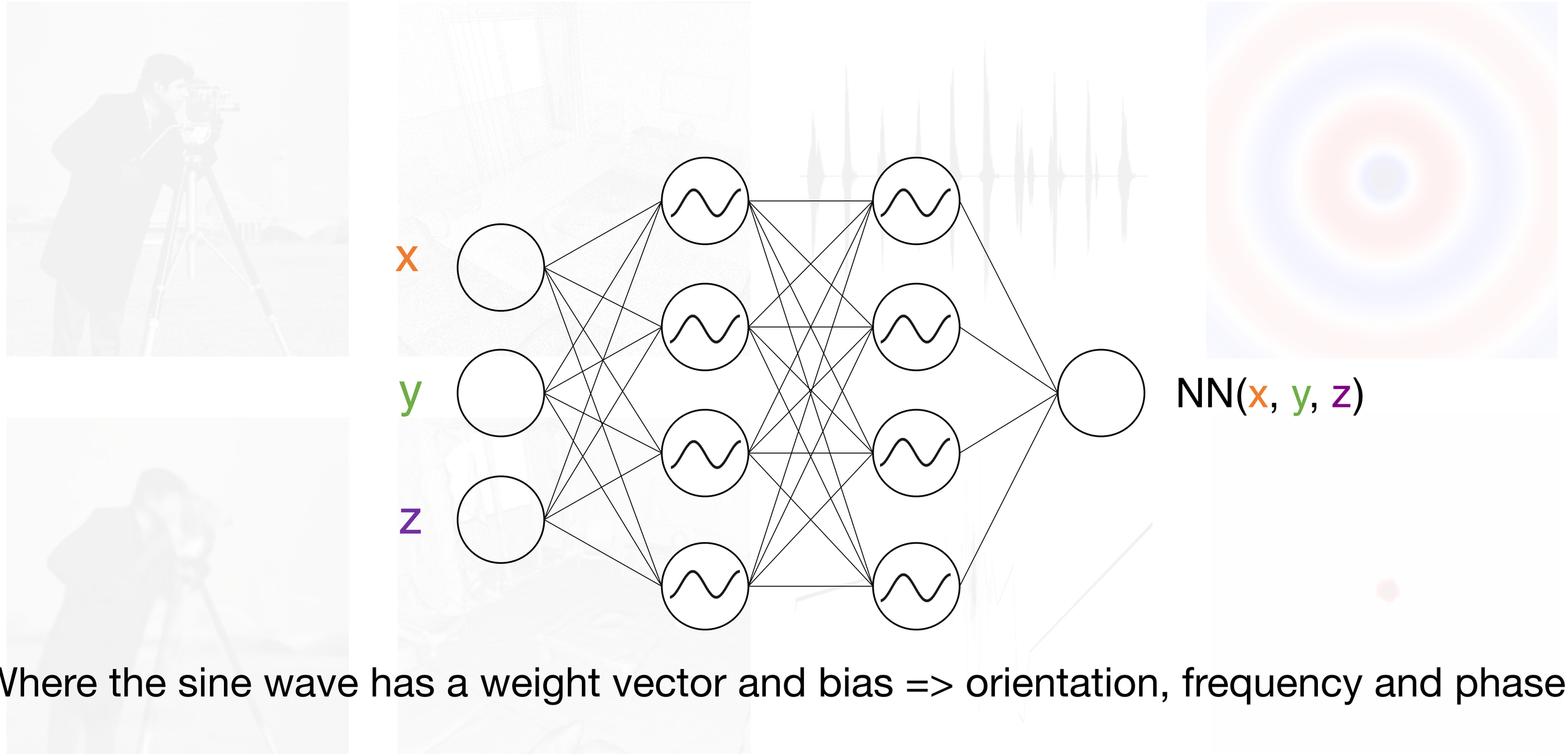


NN(x, y, z)

Where the sine wave has a weight vector and bias => orientation, frequency and phase

Sitzmann et al.: Implicit neural representations with periodic activation functions, NeurIPS 2020

SIREN: Sinusoidal Representation Networks



Where the sine wave has a weight vector and bias => orientation, frequency and phase

Sitzmann et al.: Implicit neural representations with periodic activation functions, NeurIPS 2020

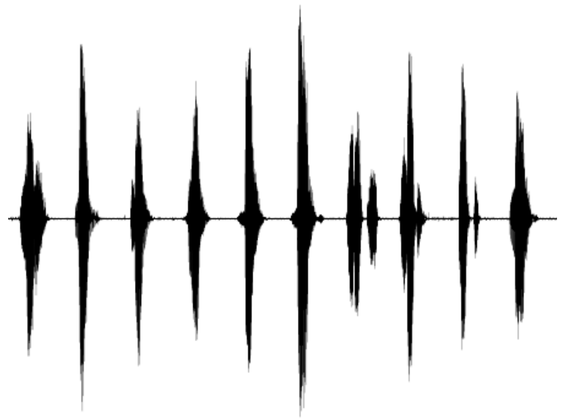
Images

Shapes

Audio

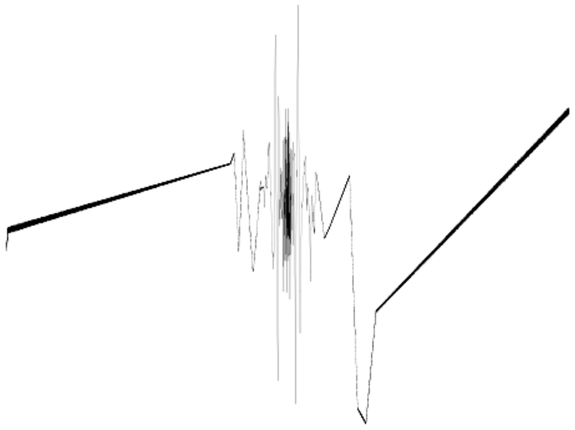












Images

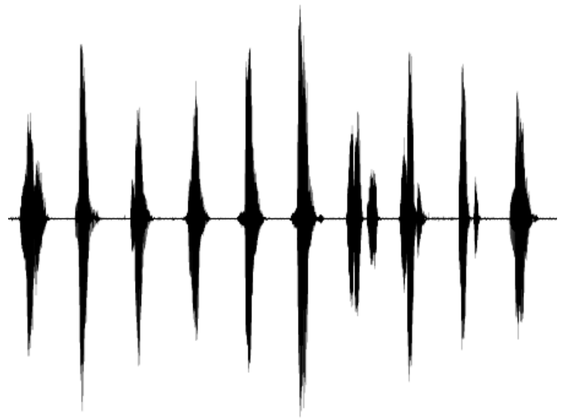
Shapes

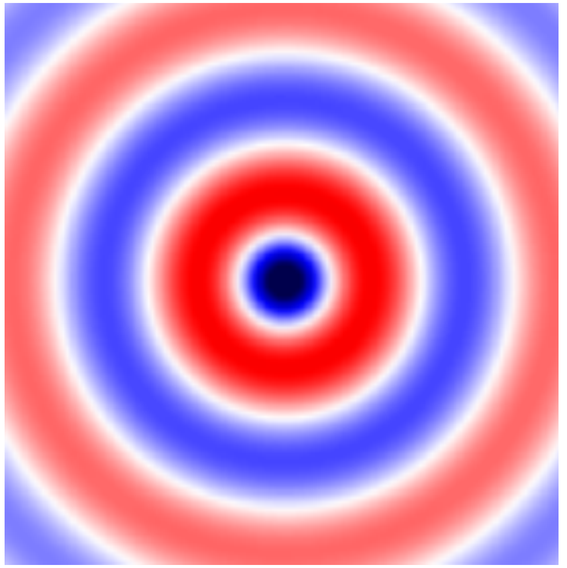
Audio

Quantities defined by a differential equation



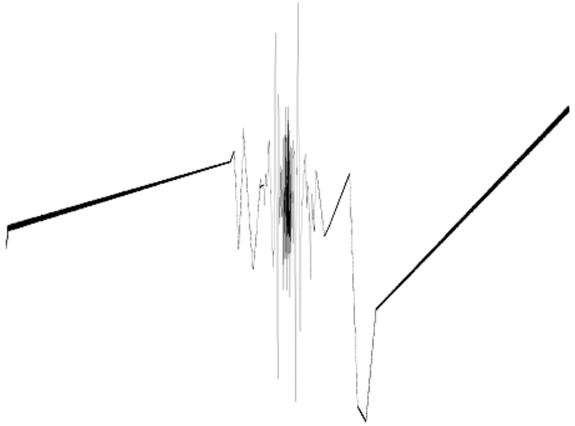














Images

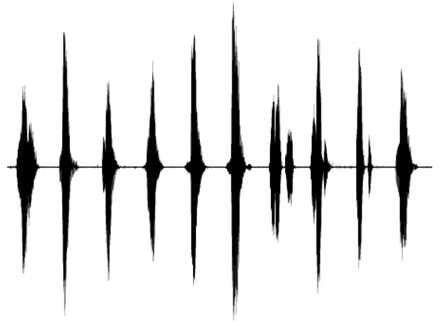
Shapes

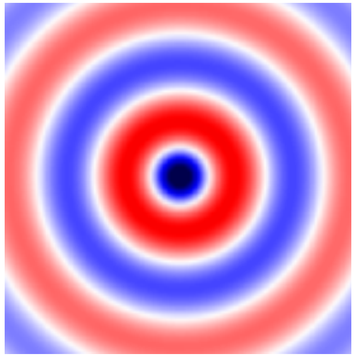
Audio

Quantities defined by a differential equation

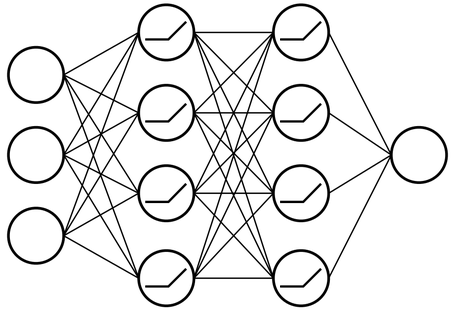






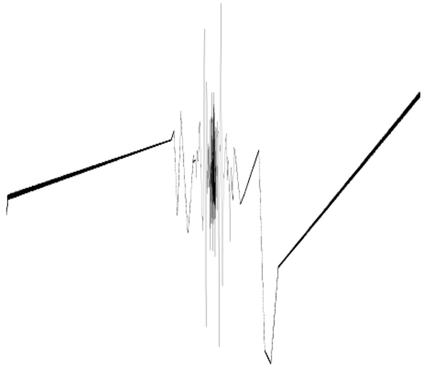


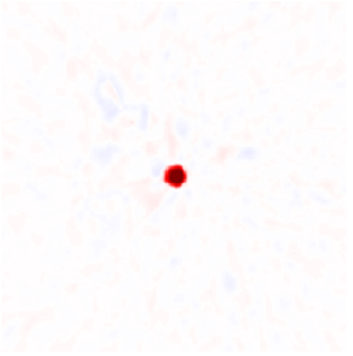
ReLU MLP



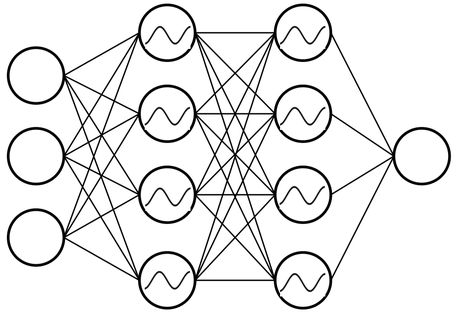








SIREN



Images

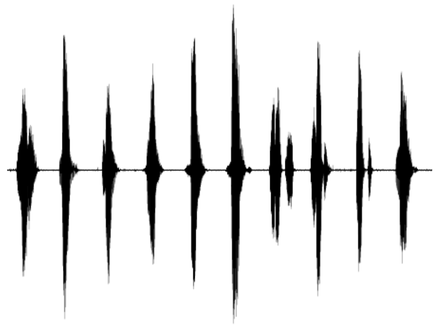
Shapes

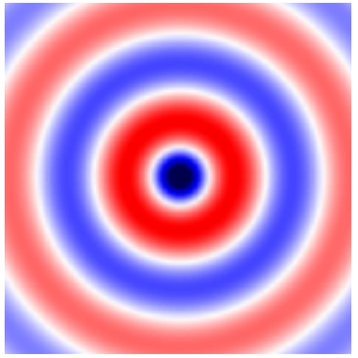
Audio

Quantities defined by a differential equation

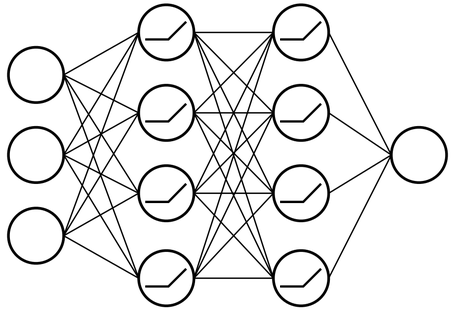






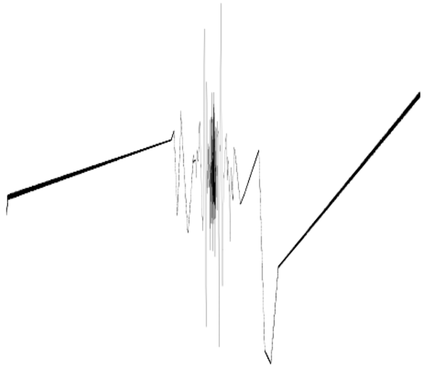


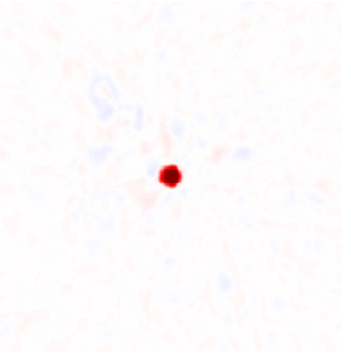
ReLU MLP



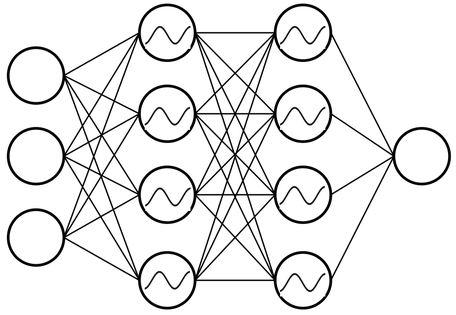






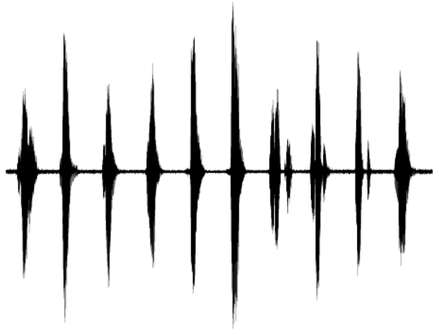


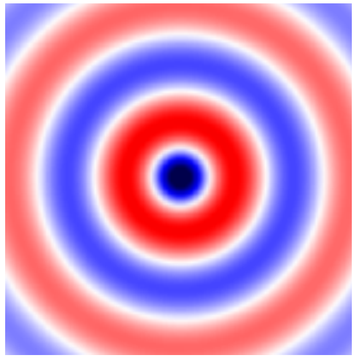
SIREN





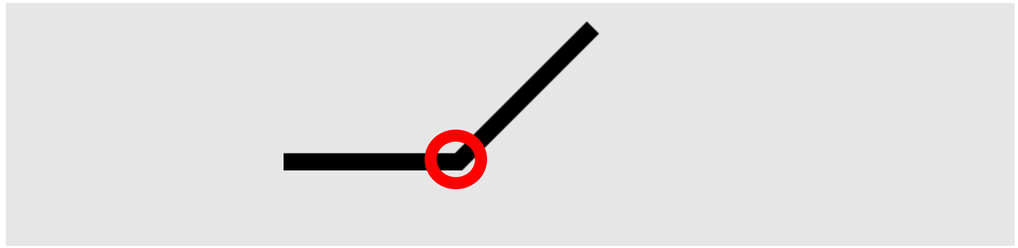


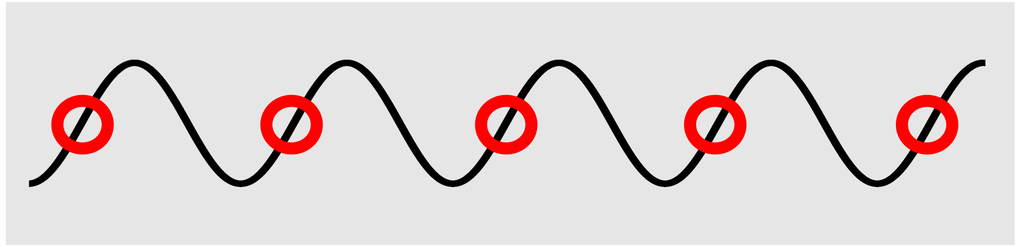




Locality

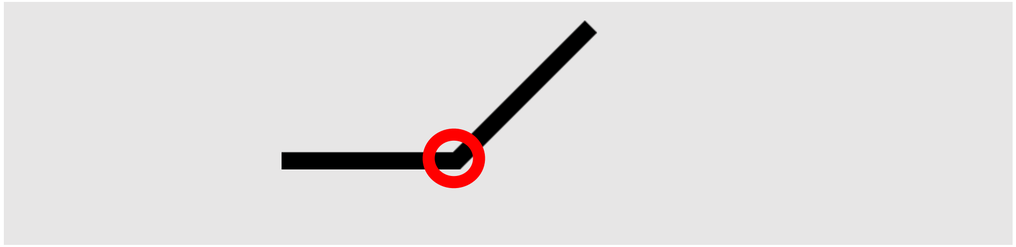
$\delta/\delta x$ $\delta^2/\delta x^2$

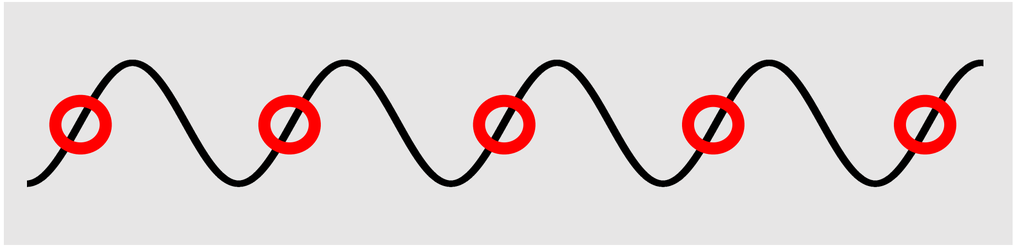




Periodicity allows SIREN to replicate activations across input domain

Locality

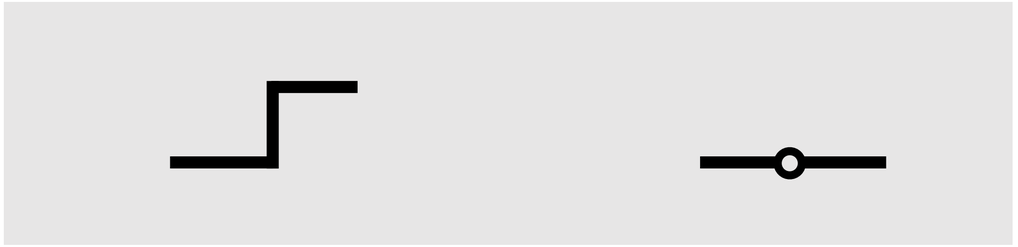


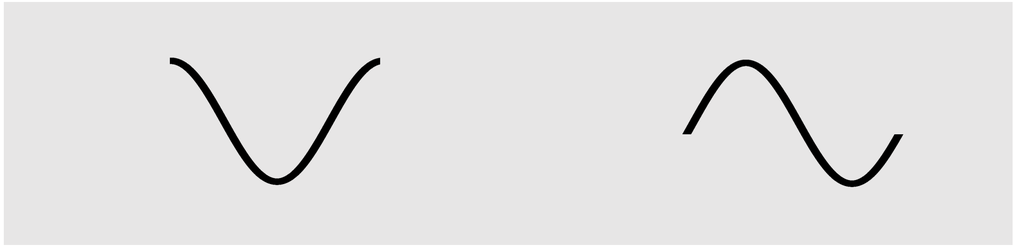


Periodicity allows SIREN to replicate activations across input domain

Derivatives

$\delta/\delta x$ $\delta^2/\delta x^2$



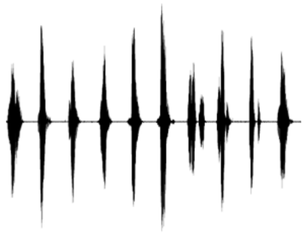


All derivatives exist, are nonzero and bounded by 1

Images



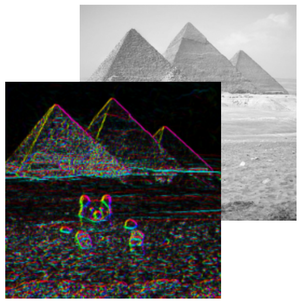
Audio



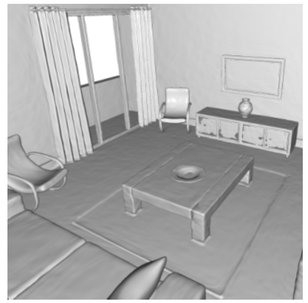
Videos



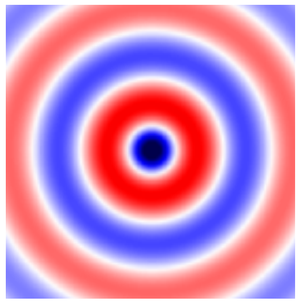
Poisson equation



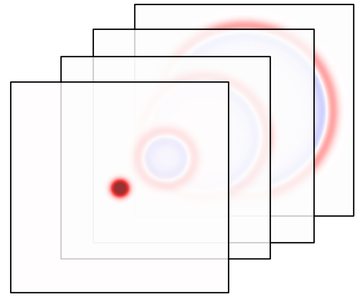
Eikonal equation



Helmholtz equation



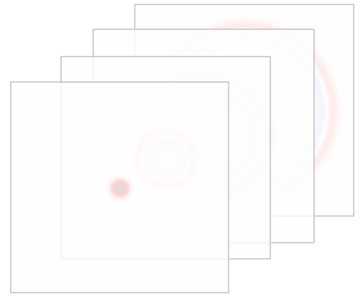
Wave equation



Images | Audio | Videos | Poisson equation | Eikonal equation | Helmholtz equation | Wave equation

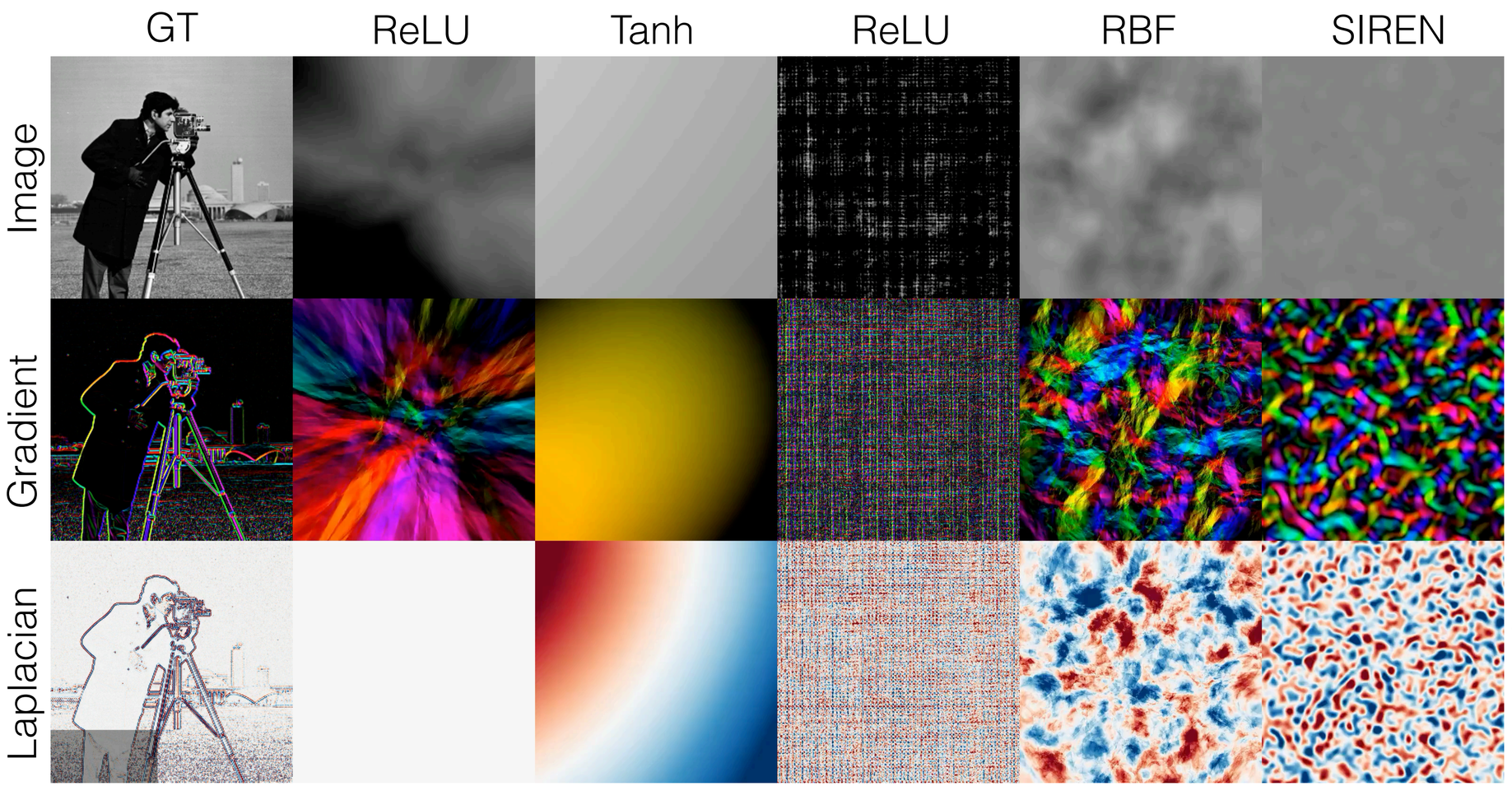


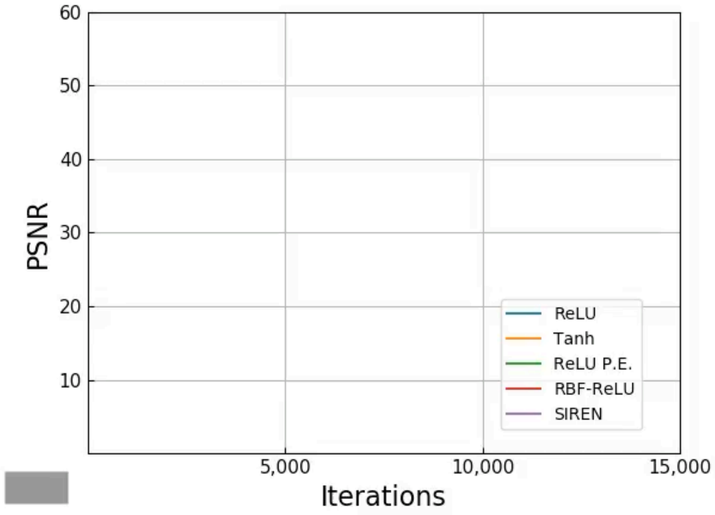




Input	Output supervised by	Implicit Formulation Find $\boldsymbol{\Phi}$ that minimizes $\mathscr{L}$
$\mathbf{x} \in \mathbb{R}^2$	$f(\mathbf{x}) \in \mathbb{R}^3$	$\mathcal{L}_{\text{img}} = \int_{\Omega} \lVert \Phi(\mathbf{x}) - f(\mathbf{x}) \rVert \, \mathrm{d}\mathbf{x}$
spatial coords.	*RGB values*	

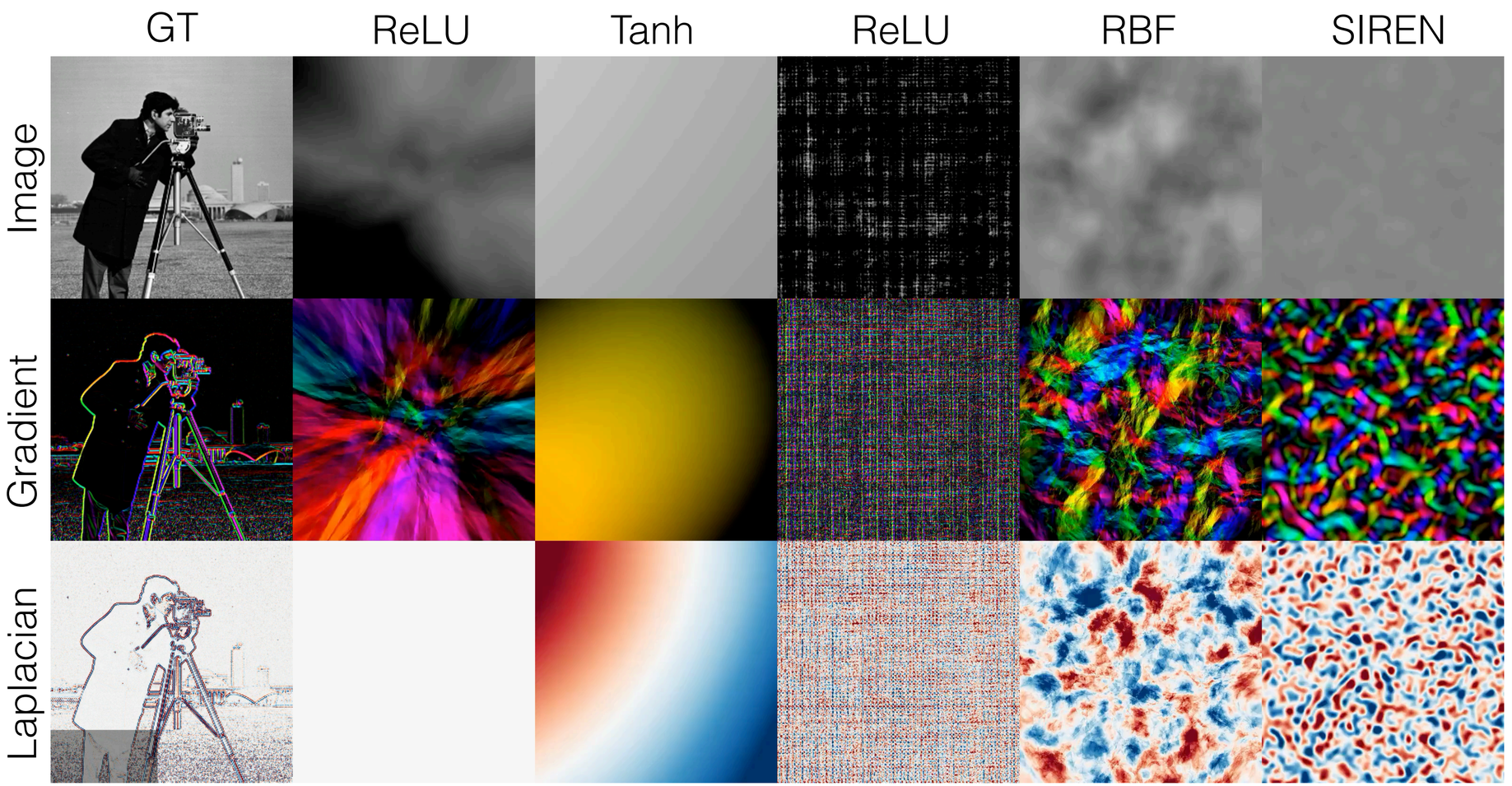
Representing Images

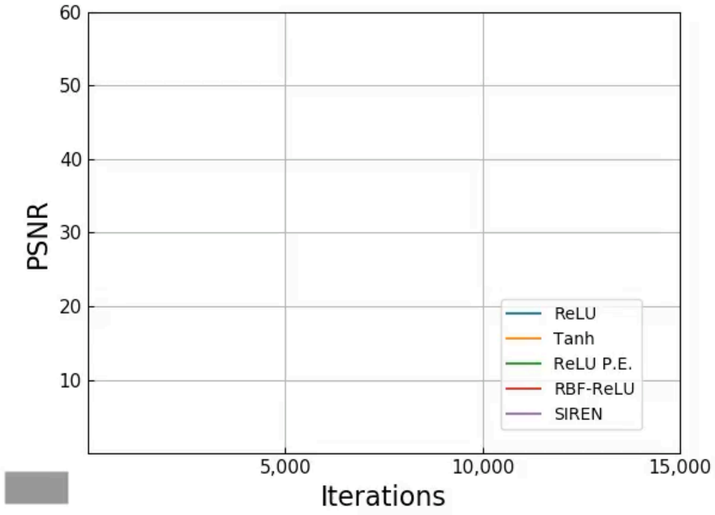




[1] Nerf [Mildenhall et al. 2020]

Representing Images

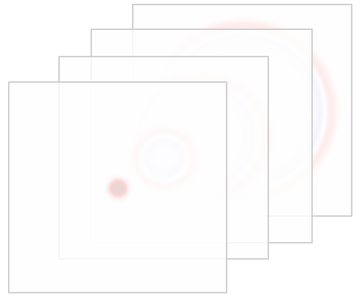




[1] Nerf [Mildenhall et al. 2020]

Images | Audio | Videos | Poisson equation | Eikonal equation | Helmholtz equation | Wave equation



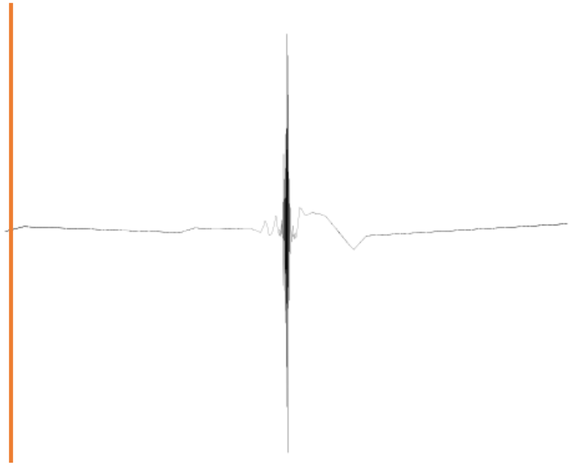
Input	Output supervised by	Implicit Formulation Find $\boldsymbol{\Phi}$ that minimizes $\mathscr{L}$
$t \in \mathbb{R}$	$f(t) \in \mathbb{R}$	$\mathcal{L}_{\text{audio}} = \int_{\Omega} \lVert \Phi(t) - f(t) \rVert \, \mathrm{d}t$
time point	*amplitude*	

Representing Audio – Music

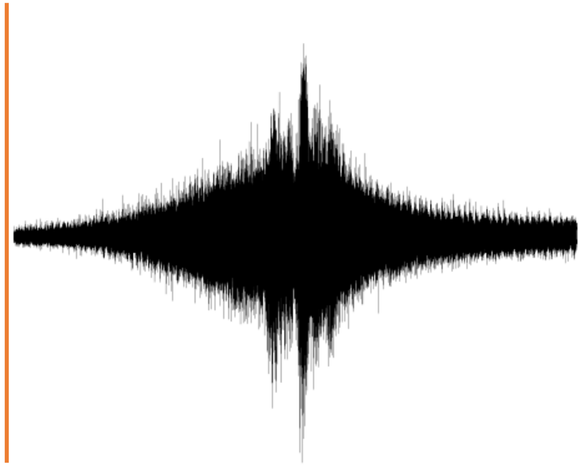
Ground Truth



ReLU MLP



ReLU w/ positional encoding



SIREN

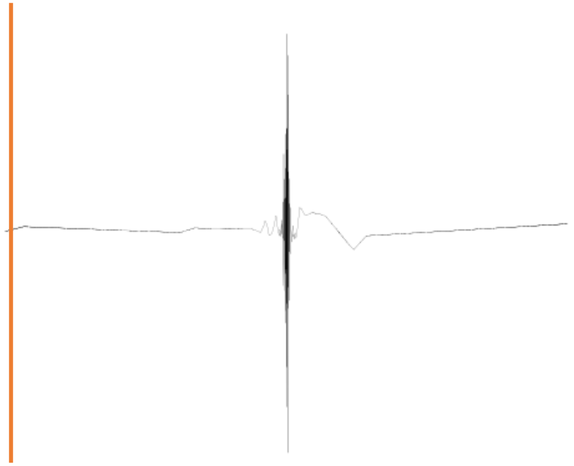


Representing Audio – Music

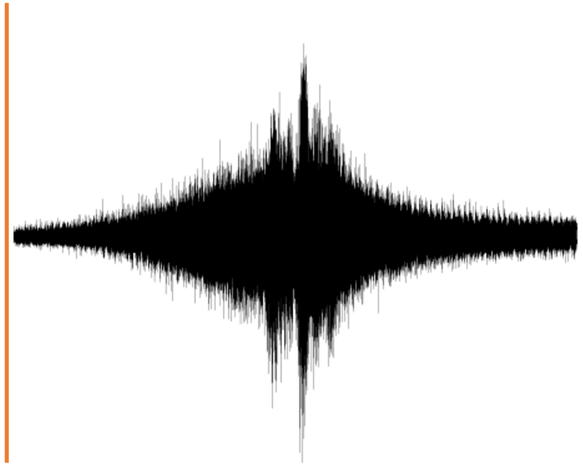
Ground Truth



ReLU MLP



ReLU w/ positional encoding



SIREN

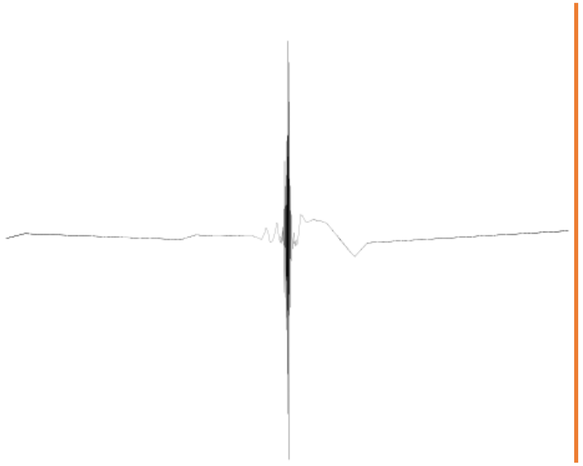


Representing Audio – Music

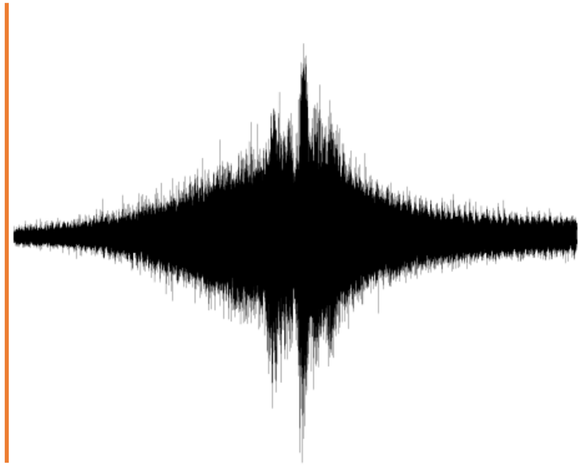
Ground Truth



ReLU MLP



ReLU w/ positional encoding



SIREN

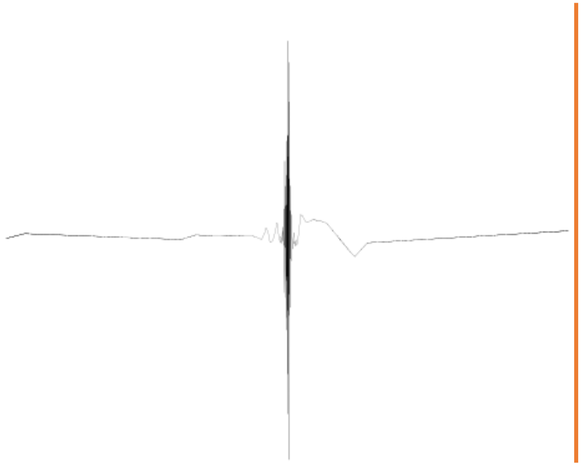


Representing Audio – Music

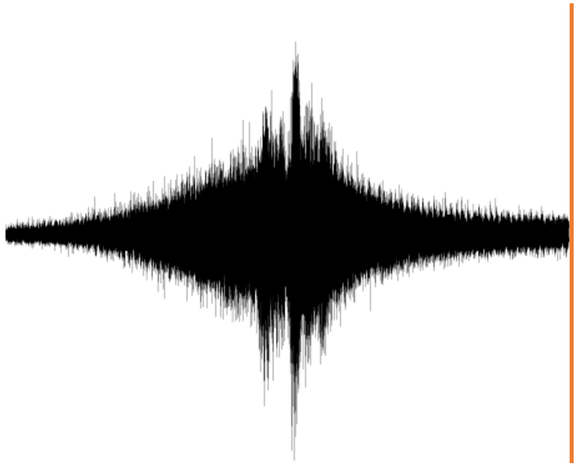
Ground Truth



ReLU MLP



ReLU w/ positional encoding



SIREN

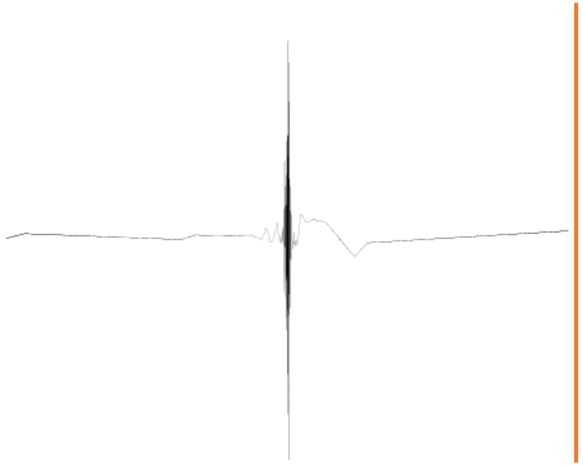


Representing Audio – Music

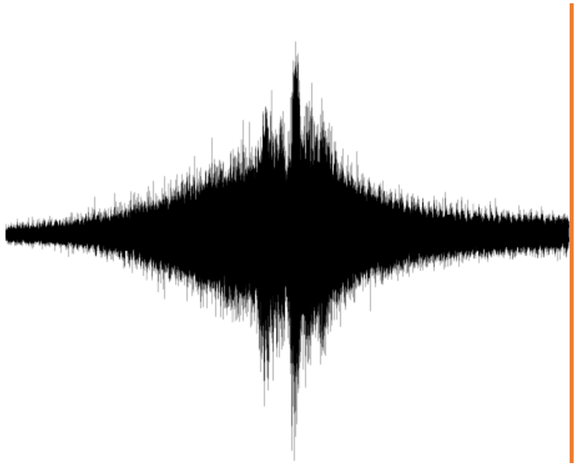
Ground Truth



ReLU MLP



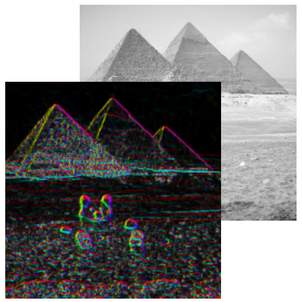
ReLU w/ positional encoding

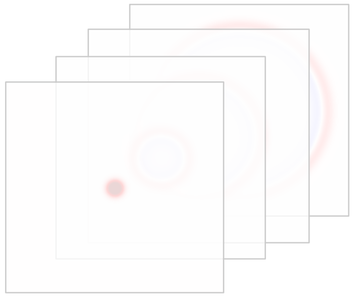


SIREN



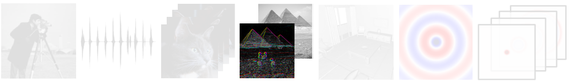
Images | Audio | Videos | Poisson equation | Eikonal equation | Helmholtz equation | Wave equation

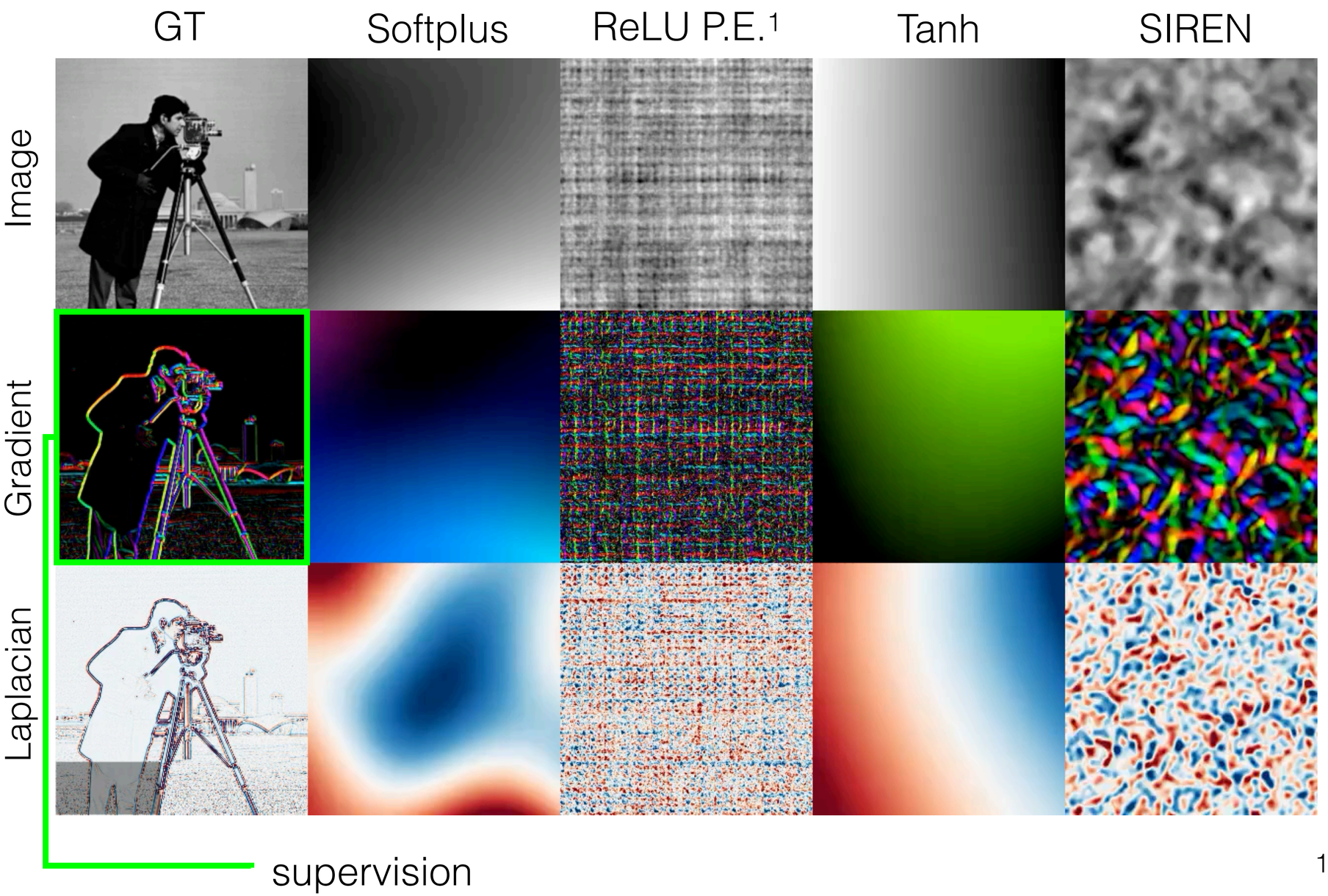



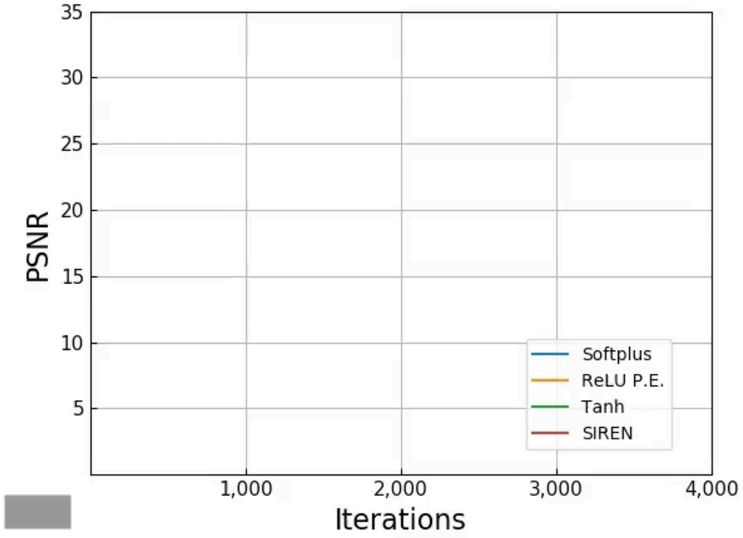


Input	Output supervised by	Implicit Formulation Find $\boldsymbol{\Phi}$ that minimizes $\mathscr{L}$
$\mathbf{x} \in \mathbb{R}^2$	$f(\mathbf{x}) \in \mathbb{R}^2$	$\mathcal{L}_{\text{Poisson}} = \int_{\Omega} \|\nabla\Phi(\mathbf{x}) - \nabla f(\mathbf{x})\| \, \mathrm{d}\mathbf{x}$
spatial coord.	*gray level*	

Poisson's Equation

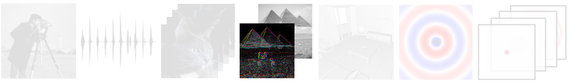


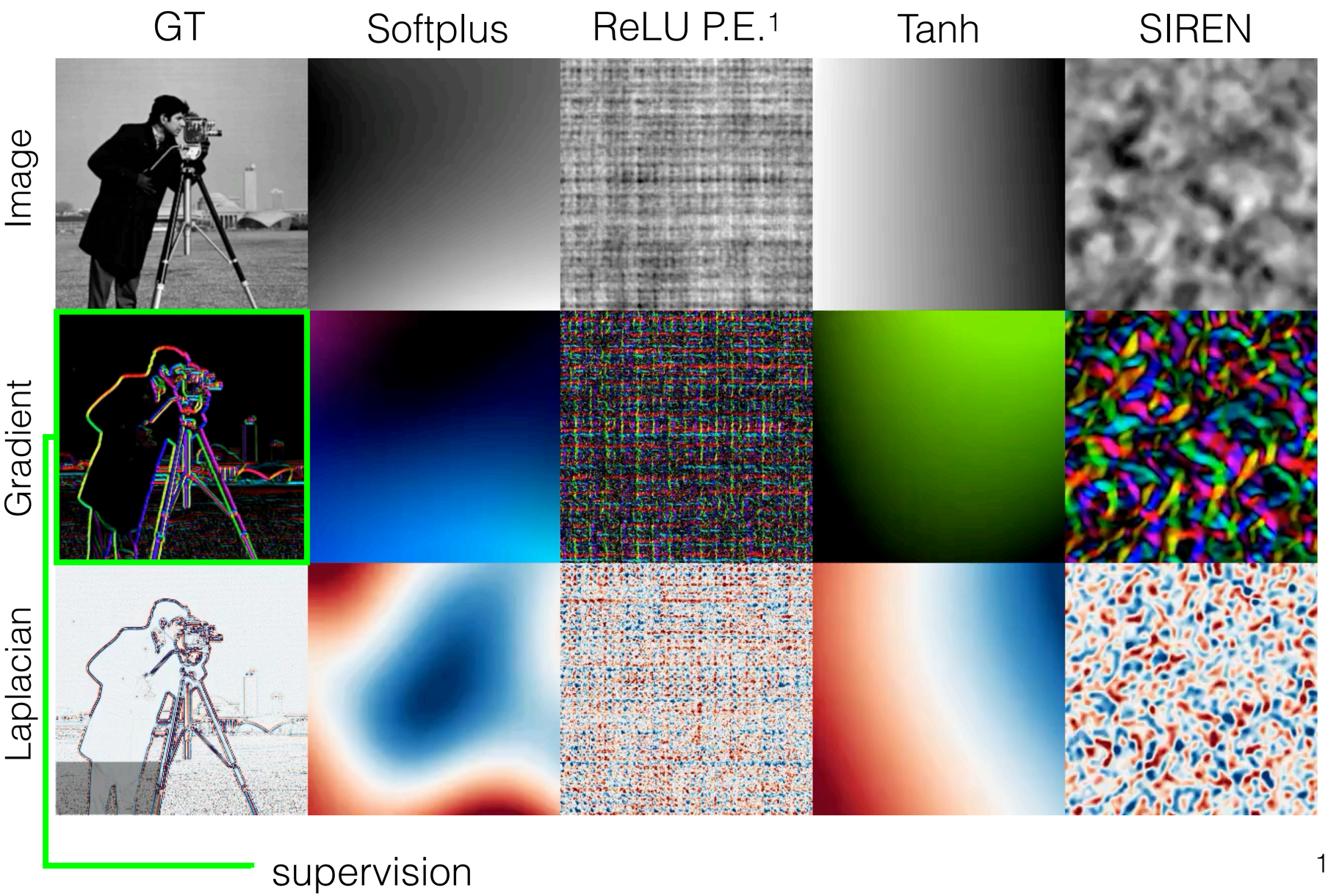


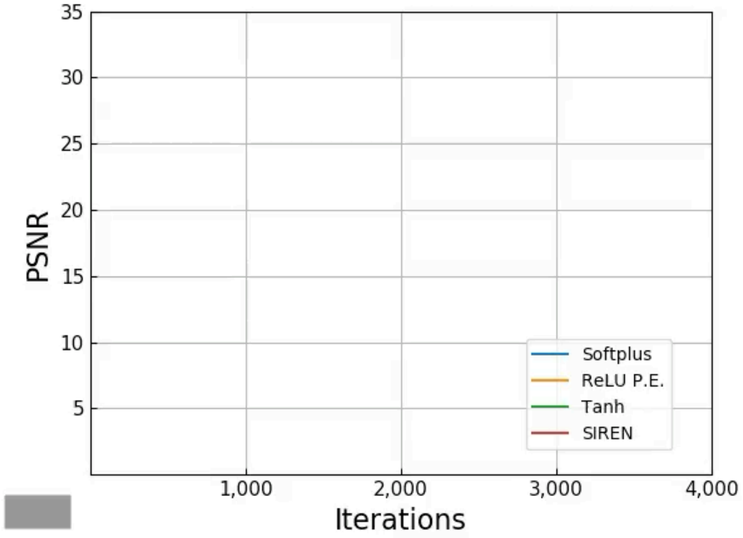


[1] Nerf [Mildenhall et al. 2020]

Poisson's Equation





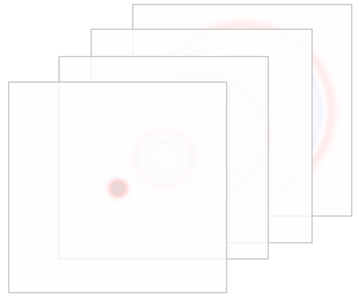


[1] Nerf [Mildenhall et al. 2020]

Images | Audio | Videos | Poisson equation | Eikonal equation | Helmholtz equation | Wave equation







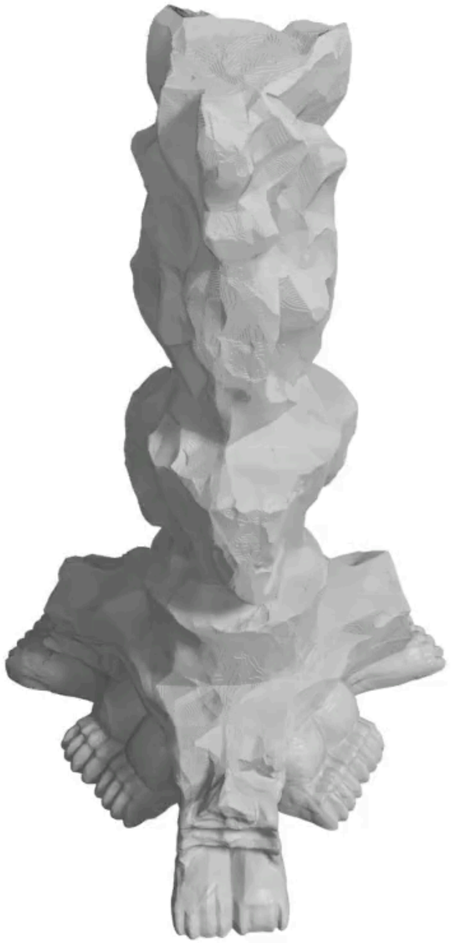
Input	Output supervised by	Implicit Formulation Find $\boldsymbol{\Phi}$ that minimizes $\mathscr{L}$
$\mathbf{x} \in \mathbb{R}^3$	$f(\mathbf{x}) \in \mathbb{R}$	$\mathcal{L}_{\text{Eikonal}} = \int_{\Omega_0} \lvert\Phi(\mathbf{x})\rvert + (1 - \langle\nabla\Phi(\mathbf{x}), \nabla f(\mathbf{x})\rangle)\, \mathrm{d}\mathbf{x}$ $+ \int_{\Omega} \lVert\lvert\nabla\Phi(\mathbf{x})\rvert - 1\rVert\, \mathrm{d}\mathbf{x}$
spatial coord.	*signed distance*	

Implicit Geometric Regularization for Learning Shapes [Gropp et al. 202

3D Shapes - solving the Eikonal equation



ReLU



SIREN

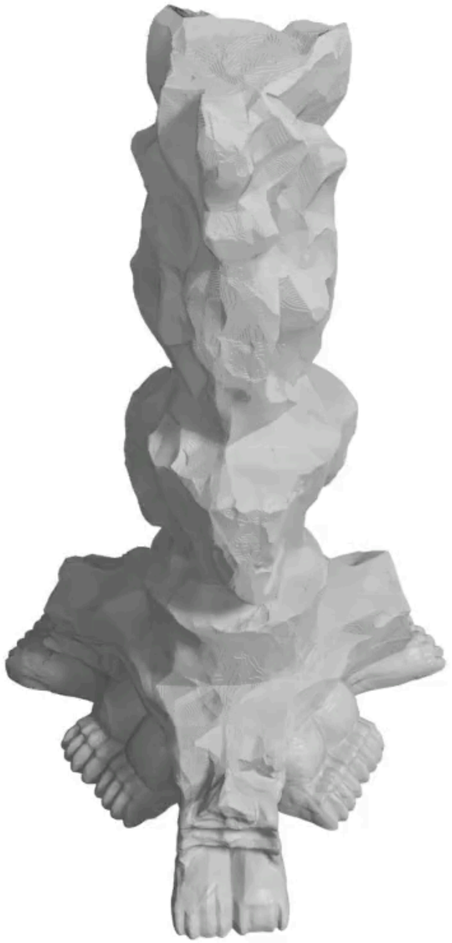


5 layers, 256 hidden units

3D Shapes - solving the Eikonal equation



ReLU



SIREN



5 layers, 256 hidden units

SIREN



ReLU

SIREN

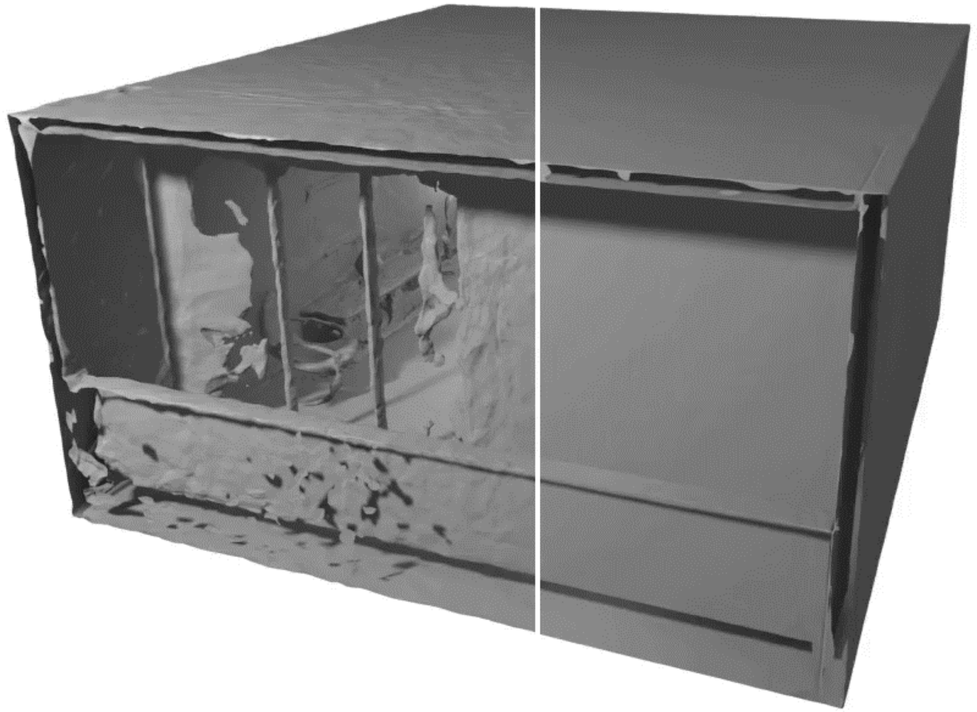
SIREN



ReLU

SIREN

SIREN



~1MB compared to
110MB of full mesh!

ReLU

SIREN

SIREN Discussion

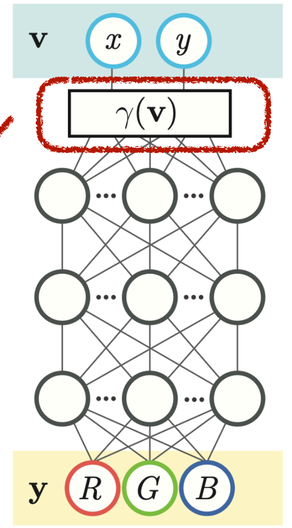
- Great when you need (higher-order) derivatives
- Finicky to train
- Initialization is key

- For NVS, localize approaches (hybrid or primitives/based) have come to dominate.

Tangent kernels

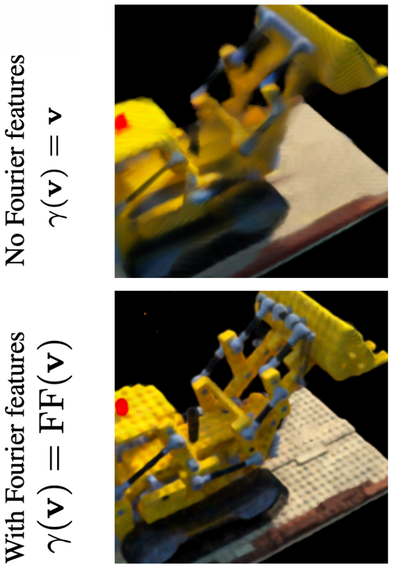
Goal

- **Why does sine/cosine positional encoding help NERF?**
- **Understand how training data leads to predictions**
- **Ideally, for a given prediction at x, get a sense of which training datapoints influence the results**
  - **Various approaches including influence functions**
    - See how the training optimum and subsequent prediction would change if a datapoint changed. Leads to second derivatives calculation after network has been trained. Pretty general but not super easy to make work
  - **And Neural Tangent Kernels**
    - More of a first-order Taylor approach around initialization parameters to see the influence of training data on parameters (and therefore predictions)
    - Only valid if parameters don't stray too far. Insightful for regular very wide MLP, not for stuff like transformers



(a) Coordinate-based MLP

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



(e) Inverse rendering

$(x, y, z) \rightarrow$ RGB, density

Background: Kernel Machines

Given "training" set $\{(x_i, y_i)\}_i$, a kernel machine makes predictions on a point $\mathbf{x}$ by interpolating labels $\mathbf{y}_i$ in the training set according to pairwise weights between $\mathbf{x}_i$ to $\mathbf{x}$ as measured by a kernel function:

$$f(\mathbf{x}) = \sum_{i=1}^{n} (\mathbf{K}^{-1}\mathbf{y})_i k(\mathbf{x}_i, \mathbf{x})$$

Where $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, k is kernel function with $k \geq 0$.

The Neural Tangent Kernel (Jacot et al. 2018)

$$f(\mathbf{x}) = \sum_{i=1}^{n} (\mathbf{K}^{-1}\mathbf{y})_i k(\mathbf{x}_i, \mathbf{x})$$

It turns out (Jacot et al. 2018) that in the limit of an infinitely wide MLP $f(x)$ with initialization θ_0, test-time predictions are made according to a kernel:

$$k_{NTK}(\mathbf{x}_1, \mathbf{x}_2) = \langle \nabla_\theta f_{\theta_0}(\mathbf{x}_1), \nabla_\theta f_{\theta_0}(\mathbf{x}_2) \rangle$$

The Neural Tangent Kernel (Jacot et al. 2018)

$$f(\mathbf{x}) = \sum_{i=1}^{n} (\mathbf{K}^{-1}\mathbf{y})_i k(\mathbf{x}_i, \mathbf{x})$$

It turns out (Jacot et al. 2018) that in the limit of an infinitely wide MLP $f(x)$ with initialization θ_0, test-time predictions are made according to a kernel:

$$k_{NTK}(\mathbf{x}_1, \mathbf{x}_2) = \langle \nabla_\theta f_{\theta_0}(\mathbf{x}_1), \nabla_\theta f_{\theta_0}(\mathbf{x}_2) \rangle$$

In other words: “Neural” implicit representations interpolate the training set according to a relatively simple rule. No “intelligence”, no AI, no magic.

The Neural Tangent Kernel (Jacot et al. 2018)

$$f(\mathbf{x}) = \sum_{i=1}^{n} (\mathbf{K}^{-1}\mathbf{y})_i k(\mathbf{x}_i, \mathbf{x})$$

It turns out (Jacot et al. 2018) that in the limit of an infinitely wide MLP $f(x)$ with initialization θ_0, test-time predictions are made according to a kernel:

$$k_{NTK}(\mathbf{x}_1, \mathbf{x}_2) = \langle \nabla_\theta f_{\theta_0}(\mathbf{x}_1), \nabla_\theta f_{\theta_0}(\mathbf{x}_2) \rangle$$

What we can do is design the kernel and pick the space of $(\mathbf{x}, \mathbf{y})$,
the space in which our neural network will interpolate.

The Neural Tangent Kernel (Jacot et al. 2018)

$$f(\mathbf{x}) = \sum_{i=1}^{n} (\mathbf{K}^{-1}\mathbf{y})_i k(\mathbf{x}_i, \mathbf{x})$$

It turns out (Jacot et al. 2018) that in the limit of an infinitely wide MLP $f(x)$ with initialization θ_0, test-time predictions are made according to a kernel:

$$k_{NTK}(\mathbf{x}_1, \mathbf{x}_2) = \langle \nabla_\theta f_{\theta_0}(\mathbf{x}_1), \nabla_\theta f_{\theta_0}(\mathbf{x}_2) \rangle$$

We do **not** know of an equivalent rule for transformers to date - work in progress!

The Neural Tangent Kernel (Jacot et al. 2018)

$$f(\mathbf{x}) = \sum_{i=1}^{n} (\mathbf{K}^{-1}\mathbf{y})_i k(\mathbf{x}_i, \mathbf{x})$$

It turns out (Jacot et al. 2018) that in the limit of an infinitely wide MLP $f(x)$ with initialization θ_0, test-time predictions are made according to a kernel:

$$k_{NTK}(\mathbf{x}_1, \mathbf{x}_2) = \langle \nabla_\theta f_{\theta_0}(\mathbf{x}_1), \nabla_\theta f_{\theta_0}(\mathbf{x}_2) \rangle$$

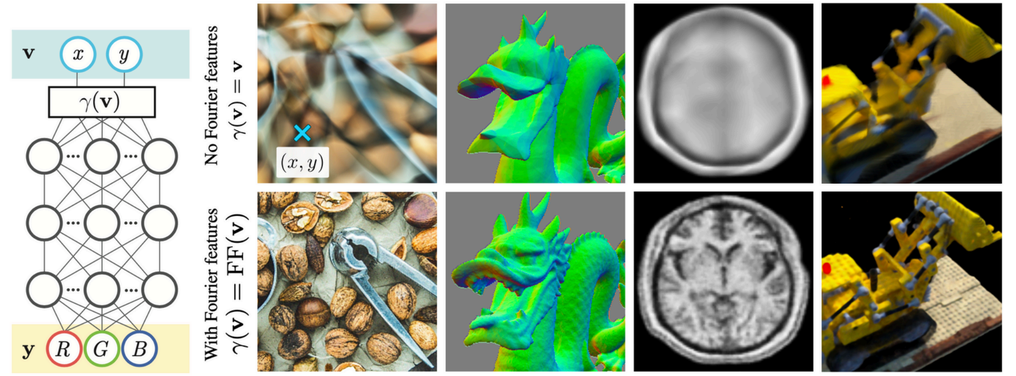
What does this mean for neural fields?

SIREN: Neural Implicit Representations With Periodic Activation Functions

Sitzmann & Martel et al. NeurIPS 2020

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

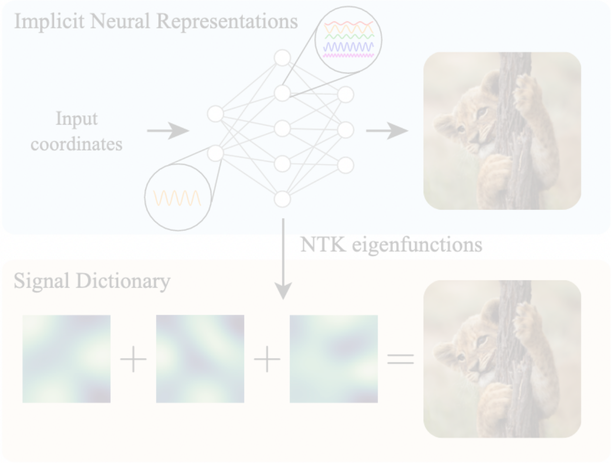
Tancik, Srinivasan, Mildenhall et al, NeurIPS 2020



(a) Coordinate-based MLP | (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

A Structured Dictionary Perspective on Implicit Neural Representations

Yüce & Ortiz-Jiménez et al. CVPR 2022



Gaussian Activated Neural Radiance Fields for High Fidelity Reconstruction & Pose Estimation

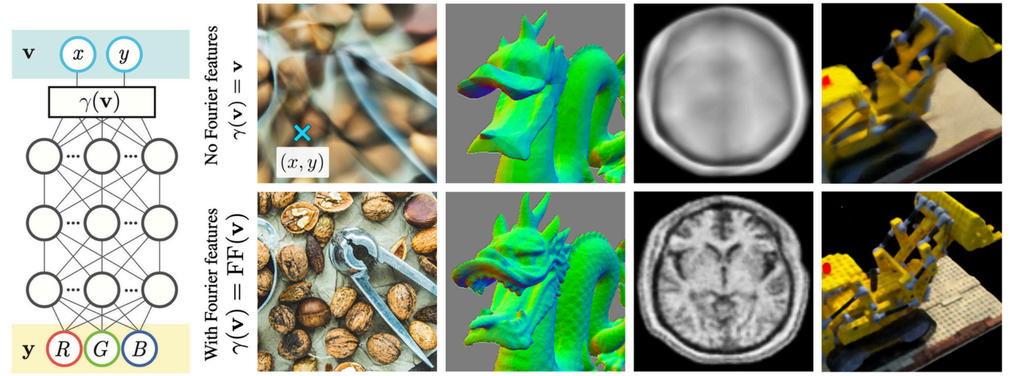
Chng et al. ECCV 2022

SIREN: Neural Implicit Representations With Periodic Activation Functions

Sitzmann & Martel et al. NeurIPS 2020

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

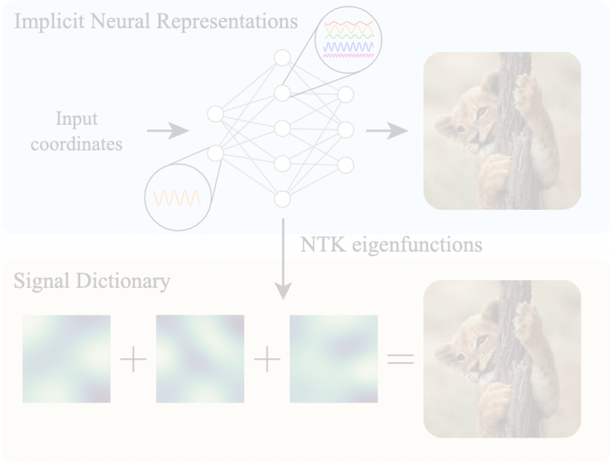
Tancik, Srinivasan, Mildenhall et al, NeurIPS 2020



(a) Coordinate-based MLP | (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

A Structured Dictionary Perspective on Implicit Neural Representations

Yüce & Ortiz-Jiménez et al. CVPR 2022



Gaussian Activated Neural Radiance Fields for High Fidelity Reconstruction & Pose Estimation

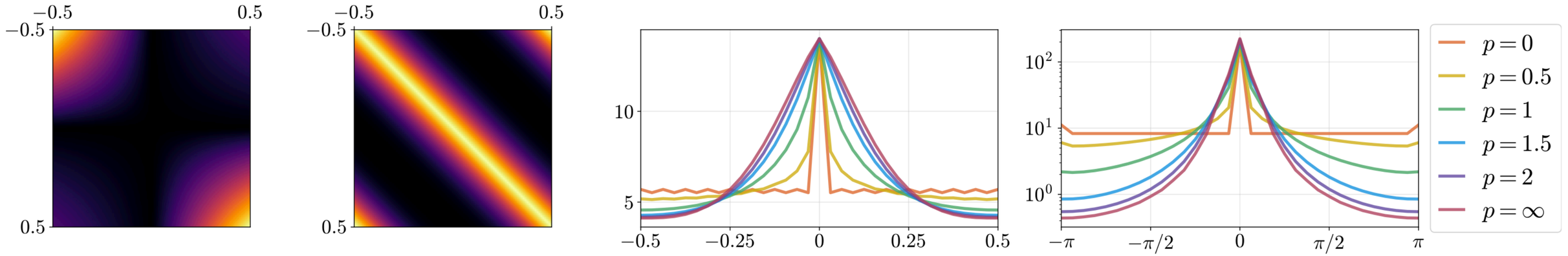
Chng et al. ECCV 2022

Kernel regression

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \left(\mathbf{K}^{-1}\mathbf{y}\right)_i k(\mathbf{x}_i, \mathbf{x}),$$

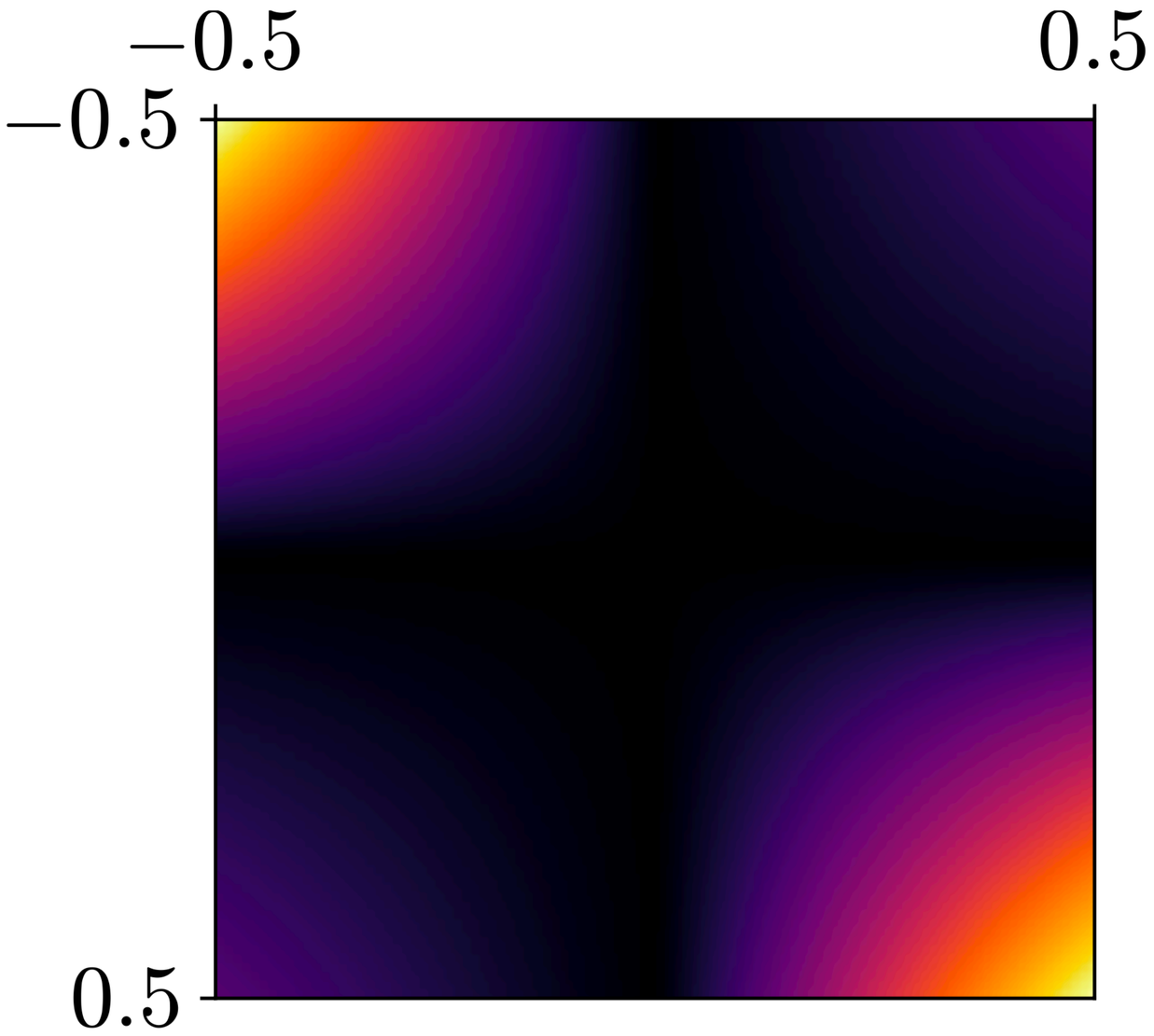
Neural tangent kernel

$$k_{\text{NTK}}(\mathbf{x}_i, \mathbf{x}_j) = \mathbb{E}_{\theta\sim\mathcal{N}} \left\langle \frac{\partial f(\mathbf{x}_i;\theta)}{\partial\theta}, \frac{\partial f(\mathbf{x}_j;\theta)}{\partial\theta} \right\rangle$$



(a) No mapping NTK (b) Basic mapping NTK (c) NTK spatial (d) NTK Fourier spectrum

Figure 2: Adding a Fourier feature mapping can improve the poor conditioning of a coordinate-based MLP's neural tangent kernel (NTK). (a) We visualize the NTK function $k_{\text{NTK}}(x_i, x_j)$ (Eqn. 2) for a 4-layer ReLU MLP with one scalar input. This kernel is not shift-invariant and does not have a strong diagonal, making it poorly suited for kernel regression in low-dimensional problems. (b) A basic input mapping $\gamma(v) = [\cos 2\pi v, \sin 2\pi v]^{\mathrm{T}}$ makes the composed NTK $k_{\text{NTK}}(\gamma(v_i), \gamma(v_j))$ shift-invariant (stationary). (c) A Fourier feature input mapping (Eqn. 5) can be used to tune the composed kernel's width, where we set $a_j = 1/j^p$ and $b_j = j$ for $j = 1, \ldots, n/2$. (d) Higher frequency mappings (lower p) result in composed kernels with wider spectra, which enables faster convergence for high-frequency components (see Figure 3).

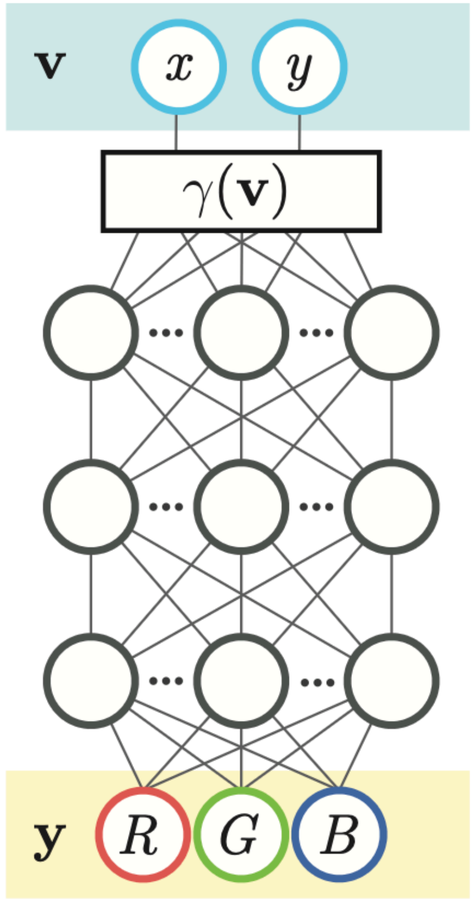


NTK kernel $k_{NTK}(x_i, x_j)$ for a 4-layer ReLU MLP with one scalar input.

From “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains”, Tancik, Srinivasan, Mildenhall et al.

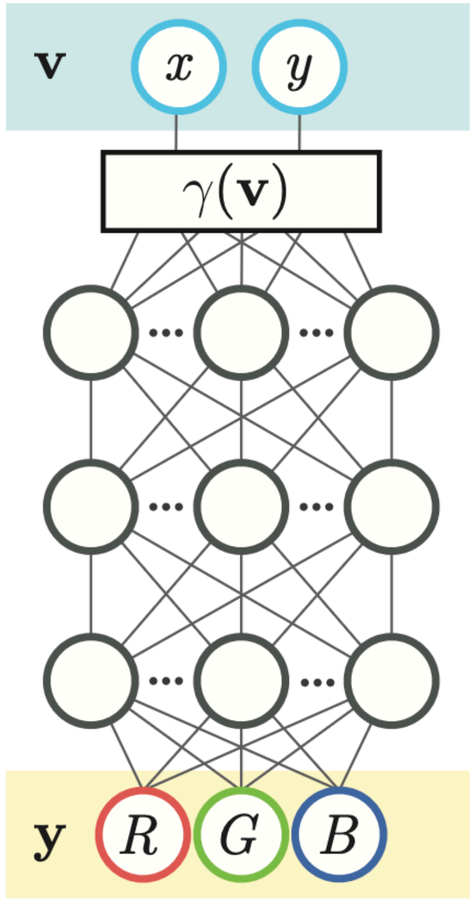
NTK of ReLU neural net is quite non-local.

Feature Embeddings



Reconstructing continuous distributions of 3D protein…, Zhong et al.; Neural Radiance Fields, Mildenhall et al.

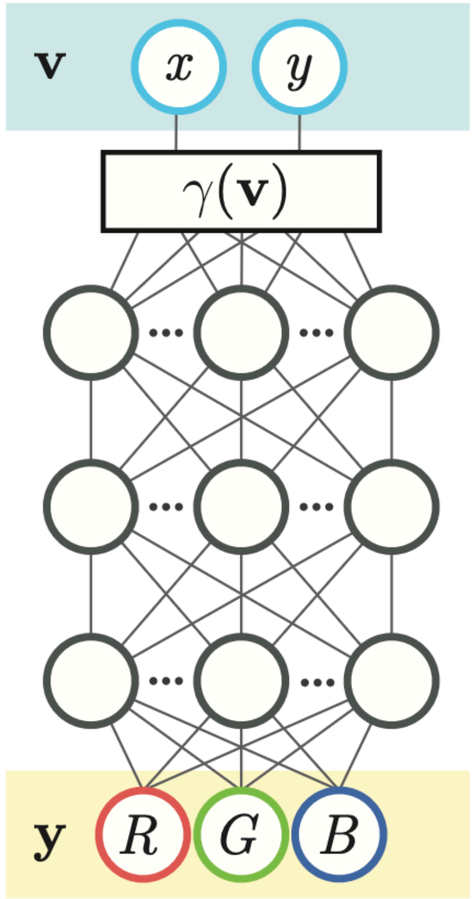
Feature Embeddings



$$\gamma(\mathbf{x}) = [\gamma_1(\mathbf{x}), \ldots, \gamma_n(\mathbf{x})]$$

Reconstructing continuous distributions of 3D protein…, Zhong et al.; Neural Radiance Fields, Mildenhall et al.

Feature Embeddings



$$\gamma(\mathbf{x}) = [\gamma_1(\mathbf{x}), \ldots, \gamma_n(\mathbf{x})]$$

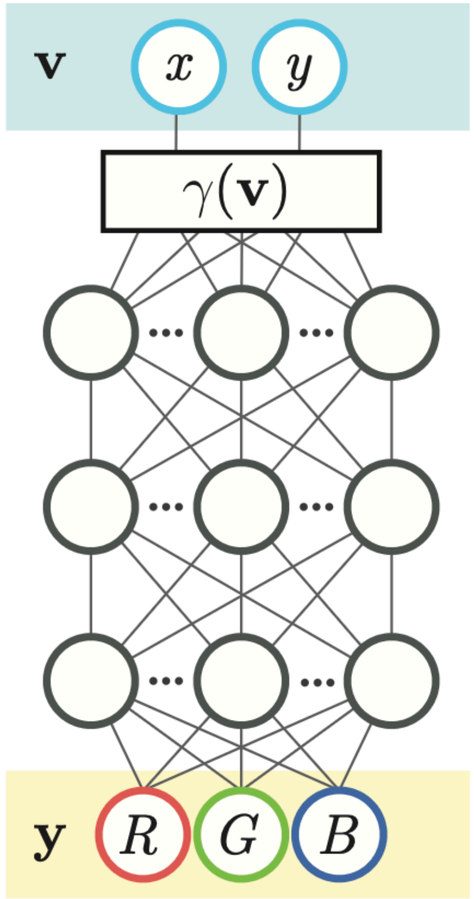
$$\gamma_{2i}(\mathbf{x}) = \sin(2^{i-1}\pi x_i)$$

$$\gamma_{(2i+1)}(\mathbf{x}) = \cos(2^{i-1}\pi x_i)$$

Sinusoidal Embeddings
Zhong et al. ICLR 2020
Mildenhall et al., ECCV 2020

Reconstructing continuous distributions of 3D protein…, Zhong et al.; Neural Radiance Fields, Mildenhall et al.

Feature Embeddings



$$\gamma(\mathbf{x}) = [\gamma_1(\mathbf{x}), \ldots, \gamma_n(\mathbf{x})]$$

$$\gamma_{2i}(\mathbf{x}) = \sin(2^{i-1}\pi x_i)$$
$$\gamma_{(2i+1)}(\mathbf{x}) = \cos(2^{i-1}\pi x_i)$$

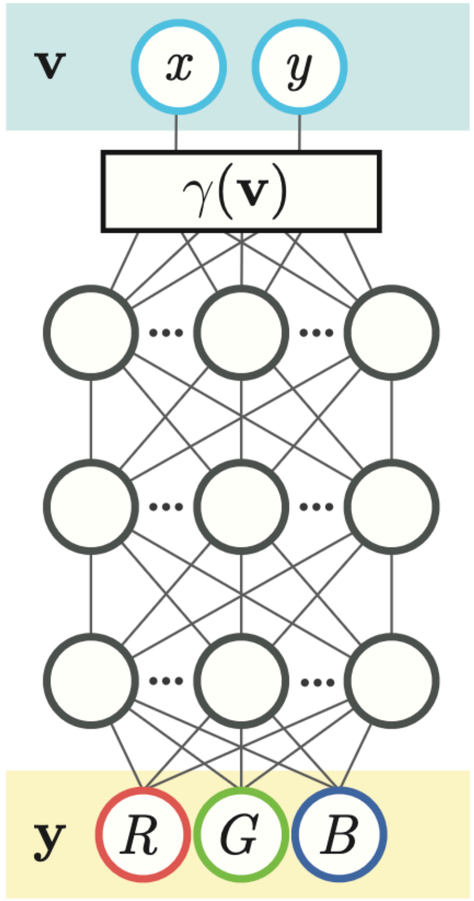
Sinusoidal Embeddings
Zhong et al. ICLR 2020
Mildenhall et al., ECCV 2020

$$\gamma(\mathbf{x}) = \exp(-\frac{\|t - x\|^2}{2\sigma^2})$$

Gaussian Embeddings
Zheng et al., arXiv 2021

Reconstructing continuous distributions of 3D protein…, Zhong et al.; Neural Radiance Fields, Mildenhall et al.

Feature Embeddings



$$\gamma(\mathbf{x}) = [\gamma_1(\mathbf{x}), \ldots, \gamma_n(\mathbf{x})]$$

$$\gamma_{2i}(\mathbf{x}) = \sin(2^{i-1}\pi x_i)$$
$$\gamma_{(2i+1)}(\mathbf{x}) = \cos(2^{i-1}\pi x_i)$$

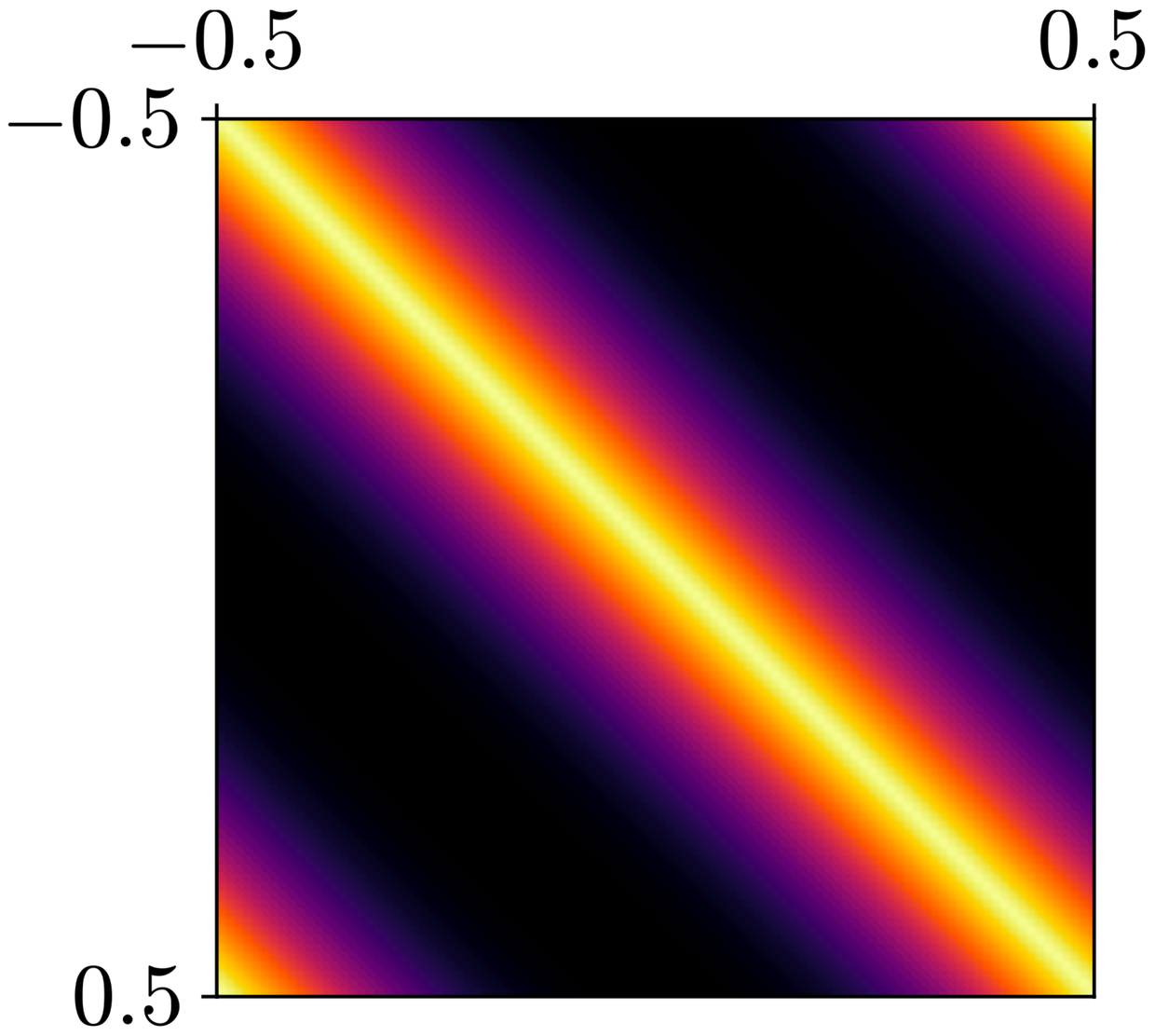
Sinusoidal Embeddings
Zhong et al. ICLR 2020
Mildenhall et al., ECCV 2020

$$\gamma(\mathbf{x}) = \exp(-\frac{\|t - x\|^2}{2\sigma^2})$$

Gaussian Embeddings
Zheng et al., arXiv 2021

• • •

Reconstructing continuous distributions of 3D protein…, Zhong et al.; Neural Radiance Fields, Mildenhall et al.



NTK kernel $k_{NTK}(\gamma(x_i), \gamma(x_j))$ for a 4-layer ReLU MLP with one scalar input and **Sinusoidal Feature Mapping**

$$\gamma(x) = [\cos(2\pi x), \sin(2\pi x)].$$

From “Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains”, Tancik, Srinivasan, Mildenhall et al.

NTK is local (note diagonal).

SIREN



ReLU

SIREN

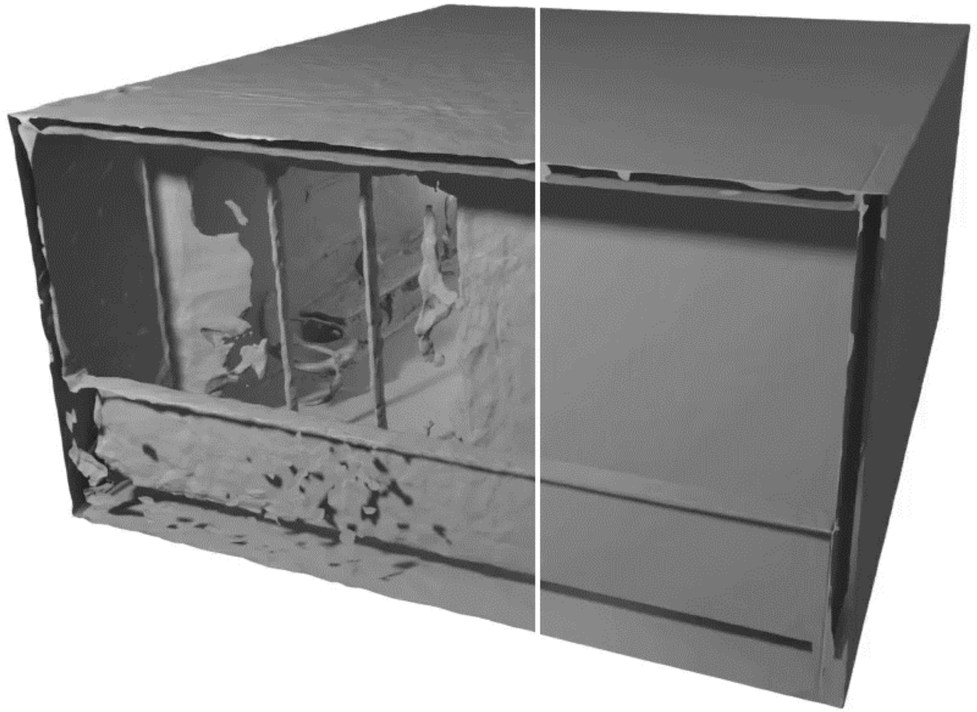
SIREN



ReLU

SIREN

SIREN

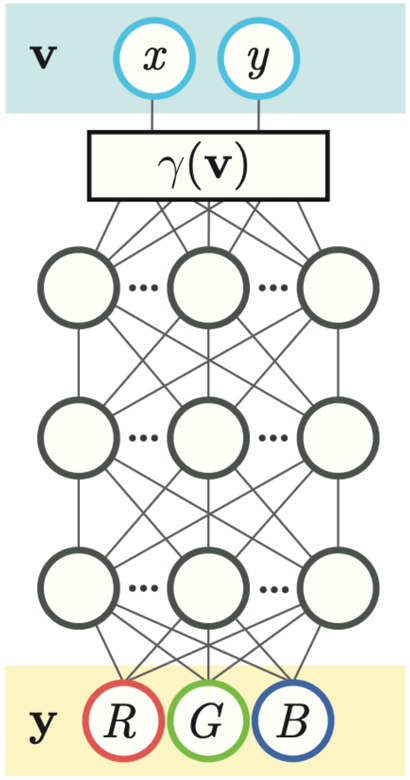


~1MB compared to
110MB of full mesh!

ReLU

SIREN

Fourier Features



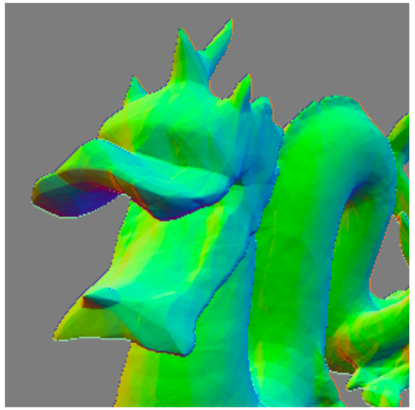
(a) Coordinate-based MLP

No Fourier features $\gamma(\mathbf{v}) = \mathbf{v}$

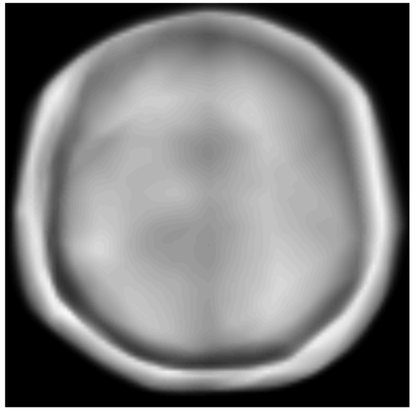
With Fourier features $\gamma(\mathbf{v}) = \mathrm{FF}(\mathbf{v})$



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



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

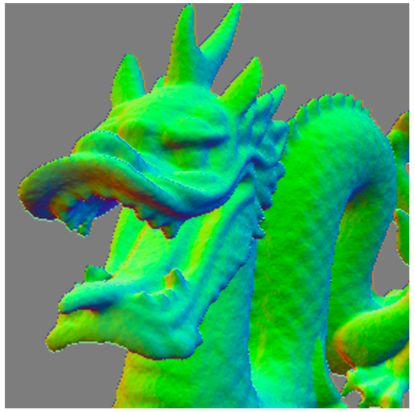


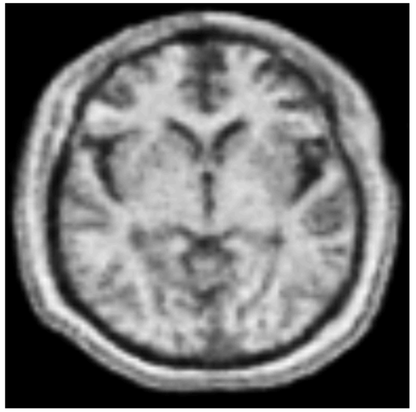
(d) MRI reconstruction
$(x,y,z) \rightarrow$ density

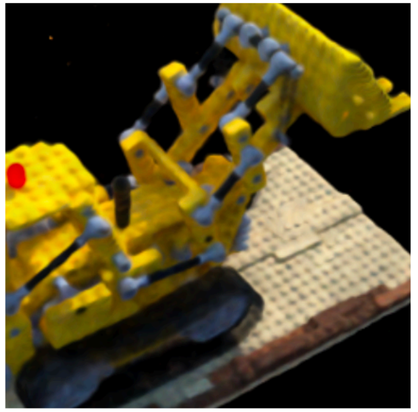


(e) Inverse rendering
$(x,y,z) \rightarrow$ RGB, density









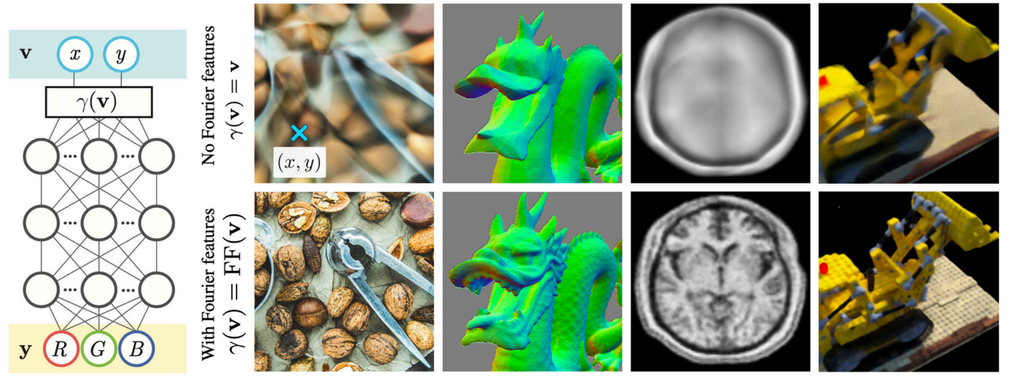
SIREN: Neural Implicit Representations With Periodic Activation Functions

Sitzmann & Martel et al. NeurIPS 2020



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

Tancik, Srinivasan, Mildenhall NeurIPS 2020



(a) Coordinate-based MLP

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

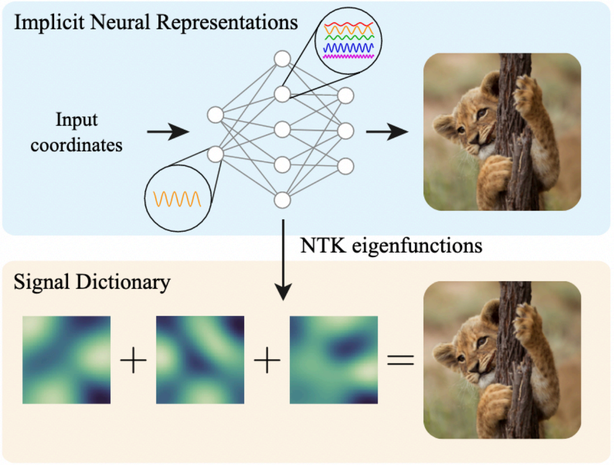
(c) 3D shape regression $(x,y,z) \rightarrow$ occupancy

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

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

A Structured Dictionary Perspective on Implicit Neural Representations

Yüce & Ortiz-Jiménez et al. CVPR 2022



Gaussian Activated Neural Radiance Fields for High Fidelity Reconstruction & Pose Estimation

Chng et al. ECCV 2022



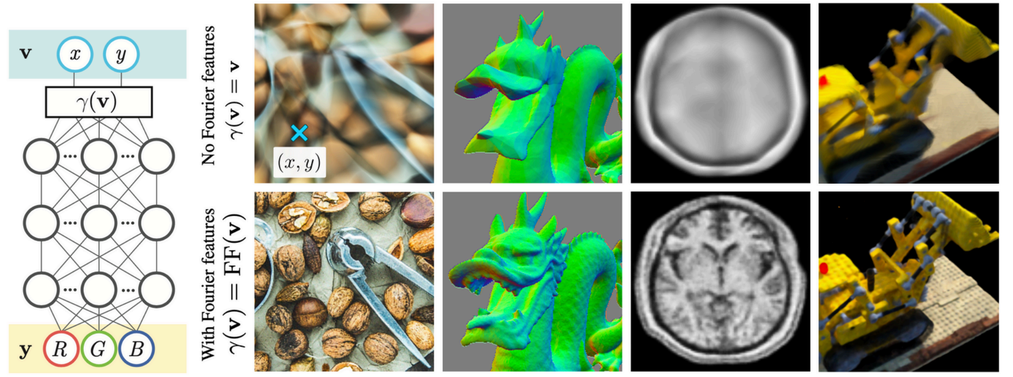
SIREN: Neural Implicit Representations With Periodic Activation Functions

Sitzmann & Martel et al. NeurIPS 2020



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

Tancik, Srinivasan, Mildenhall NeurIPS 2020



(a) Coordinate-based MLP

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

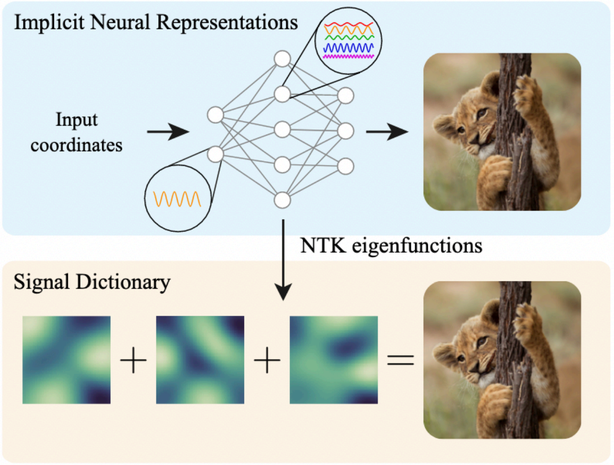
(c) 3D shape regression $(x,y,z) \rightarrow$ occupancy

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

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

A Structured Dictionary Perspective on Implicit Neural Representations

Yüce & Ortiz-Jiménez et al. CVPR 2022



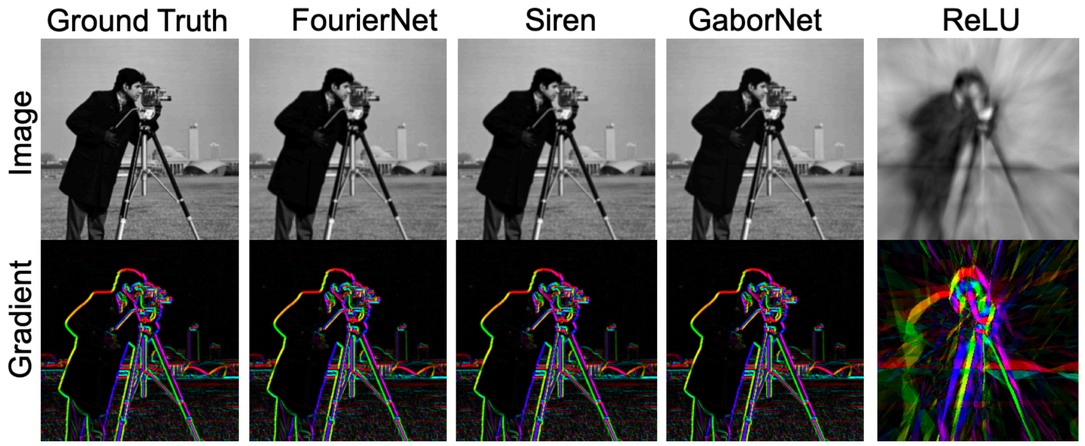
Gaussian Activated Neural Radiance Fields for High Fidelity Reconstruction & Pose Estimation

Chng et al. ECCV 2022

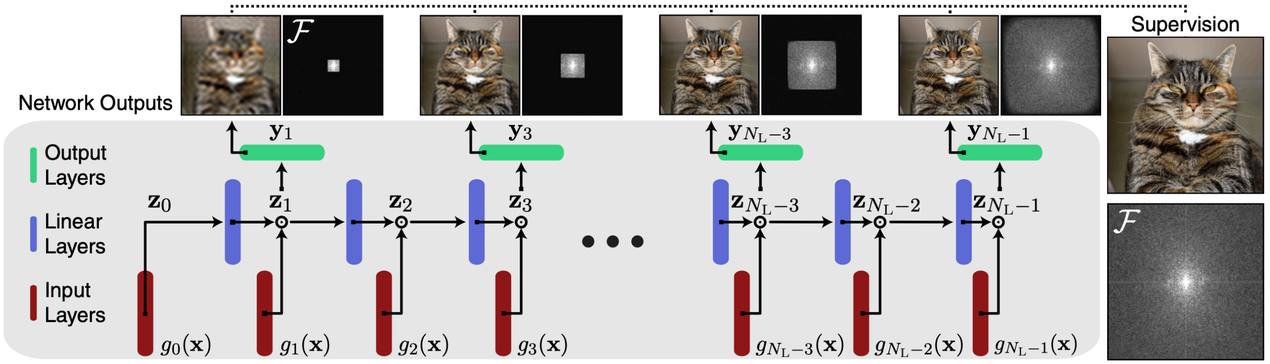


Neural (?) Fields with analytical (but not tractably computable) Fourier spectrum!

Multiplicative Filter Networks
Fathony et al. ICLR 2021

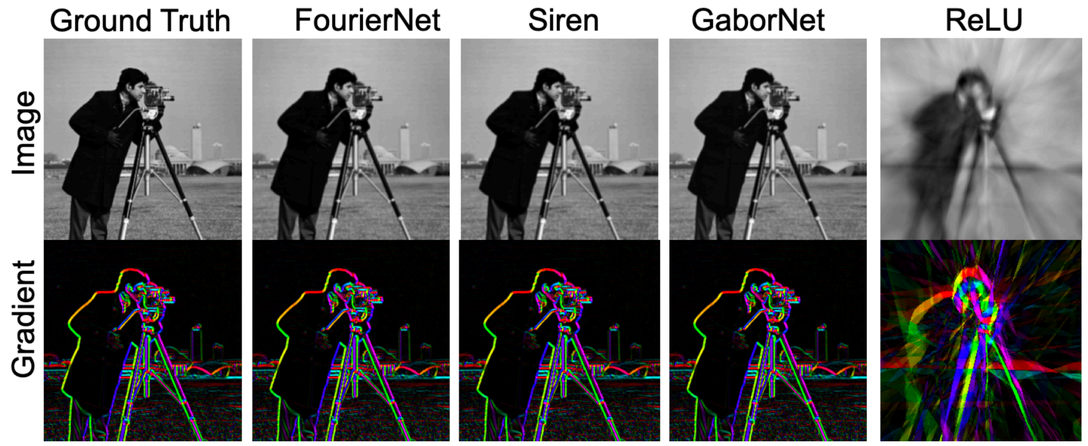


BACON: Band-limited coordinate networks for multiscale scene representation
Lindell et al. CVPR 2022

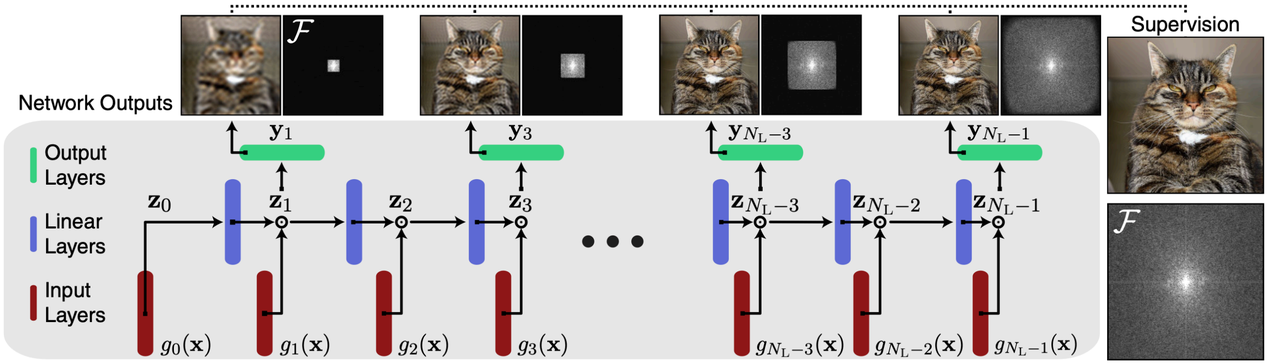


Neural (?) Fields with analytical (but not tractably computable) Fourier spectrum!

Multiplicative Filter Networks
Fathony et al. ICLR 2021



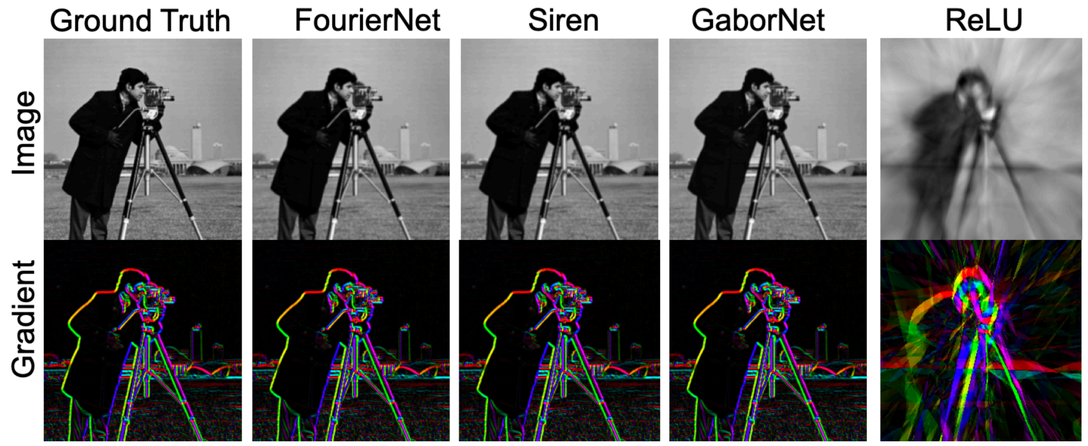
BACON: Band-limited coordinate networks for multiscale scene representation
Lindell et al. CVPR 2022



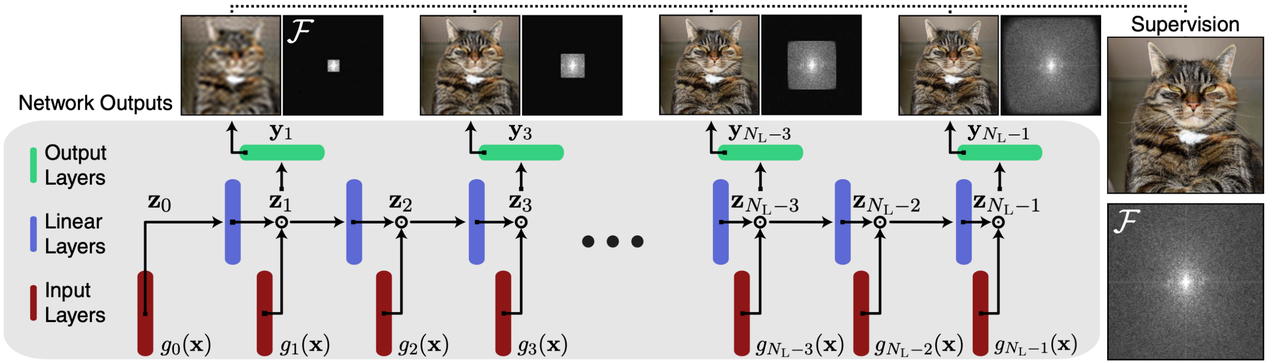
• • •

Neural (?) Fields with analytical (but not tractably computable) spectrum!

Multiplicative Filter Networks
Fathony et al. ICLR 2021



BACON: Band-limited coordinate networks for multiscale scene representation
Lindell et al. CVPR 2022



arXiv:2111.11426v4 [cs.CV] 5 Apr 2022

EUROGRAPHICS 2022
D. Meneveaux and G. Patanè
(Guest Editors)

Volume 41 (2022), Number 2
STAR – State of The Art Report

Neural Fields in Visual Computing and Beyond

Yiheng Xie[1,2] Towaki Takikawa[3,4] Shunsuke Saito[5] Or Litany[4] Shiqin Yan[1] Numair Khan[1] Federico Tombari[6,7]
James Tompkin[1] Vincent Sitzmann[8†] Srinath Sridhar[1†]

[1]Brown University [2]Unity Technologies [3]University of Toronto [4]NVIDIA [5]Meta Reality Labs Research [6]Google [7]Technical University of Munich
[8]Massachusetts Institute of Technology [†]*Equal advising*

https://neuralfields.cs.brown.edu/

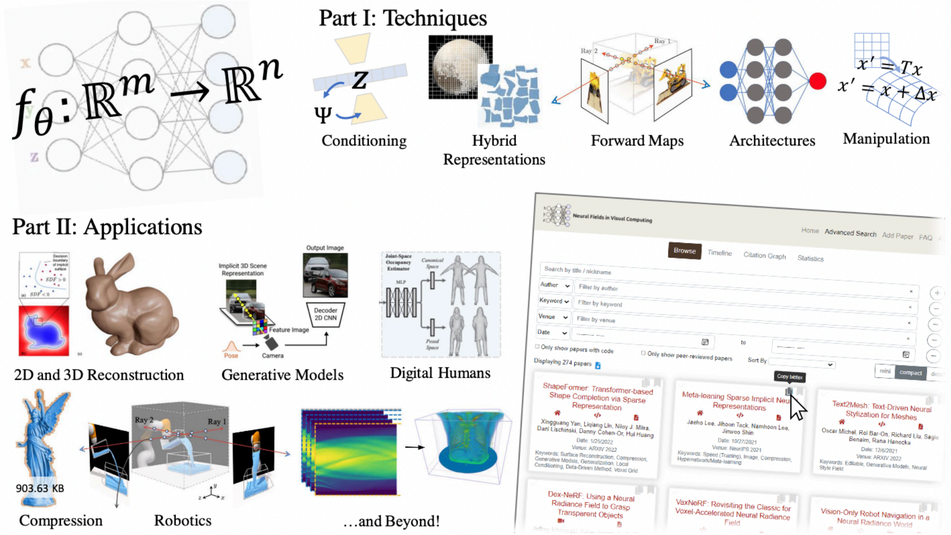
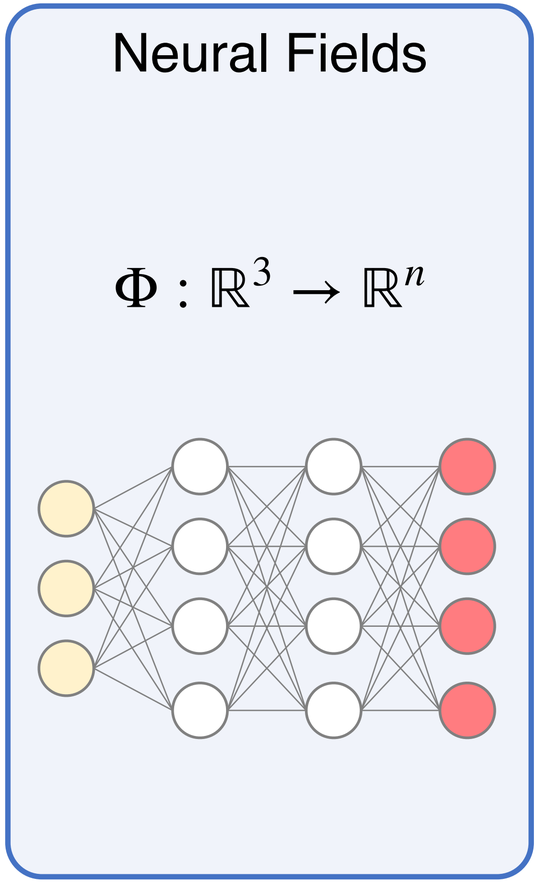


Figure 1: **Contribution of this report.** Following a survey of over 250 papers, we provide a review of (**Part I**) techniques in neural fields such as prior learning and conditioning, representations, forward maps, architectures, and manipulation, and of (**Part II**) applications in visual computing including 2D image processing, 3D scene reconstruction, generative modeling, digital humans, compression, robotics, and beyond. This report is complemented by a community-driven website with search, filtering, bibliographic, and visualization features.

Neural Fields

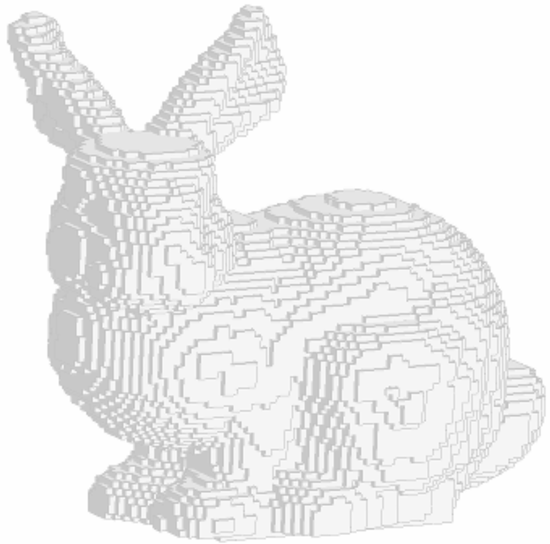
- Storage memory **does not grow with spatial resolution or number of spatial dimensions**.
- **Slow** sampling: Each query takes a forward pass through the neural net. Means GPU-memory intensive forward passes.
- Does **not** expose locality: Can't identify set of parameters / direction in parameter space that encodes particular spatial location.
- **Inconvenient processing**: cannot run convolutions.
- But: automatic adaptive resolution. Over optimization, will converge to assign more compute to higher-frequency areas.

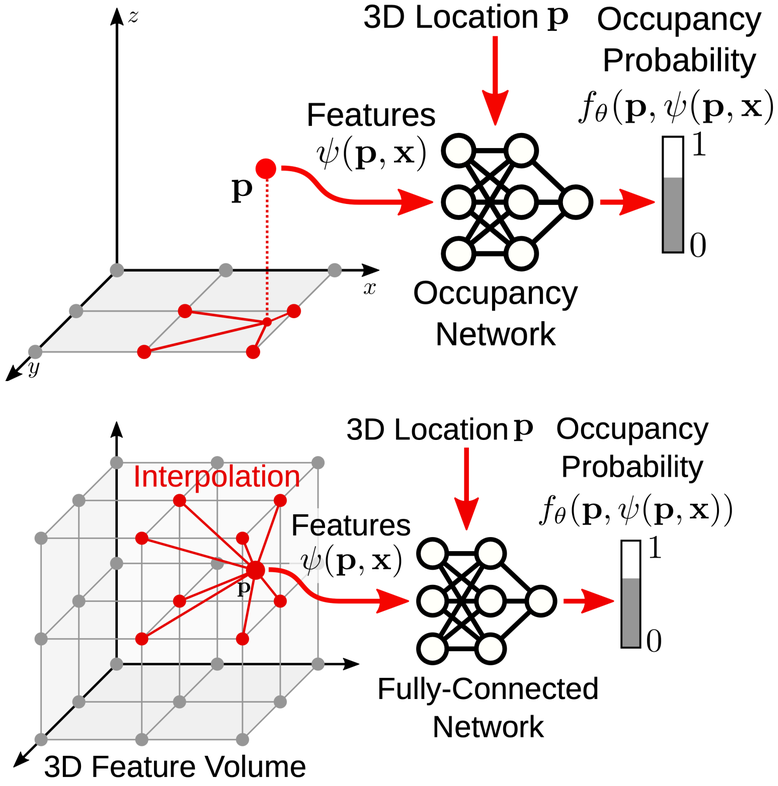


Note: Despite their name, has nothing to do with Machine Learning itself.
It's as "smart" as a voxelgrid.

Hybrid approaches

Hybrid discrete / continuous reps





Discrete Parameterizations

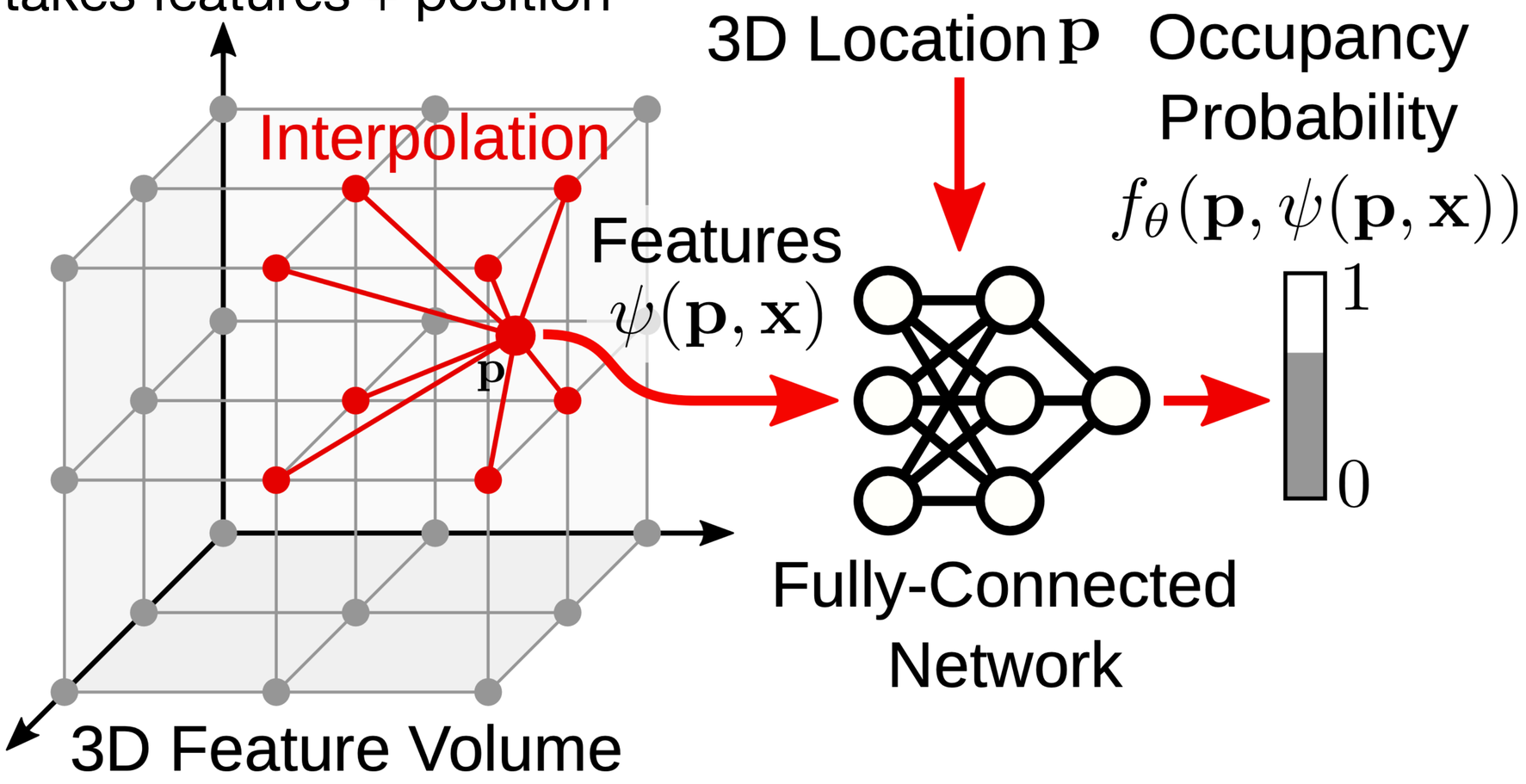
Continuous Parameterizations

Hybrid Parameterizations

Basic idea: Use Neural Net as Interpolation Kernel

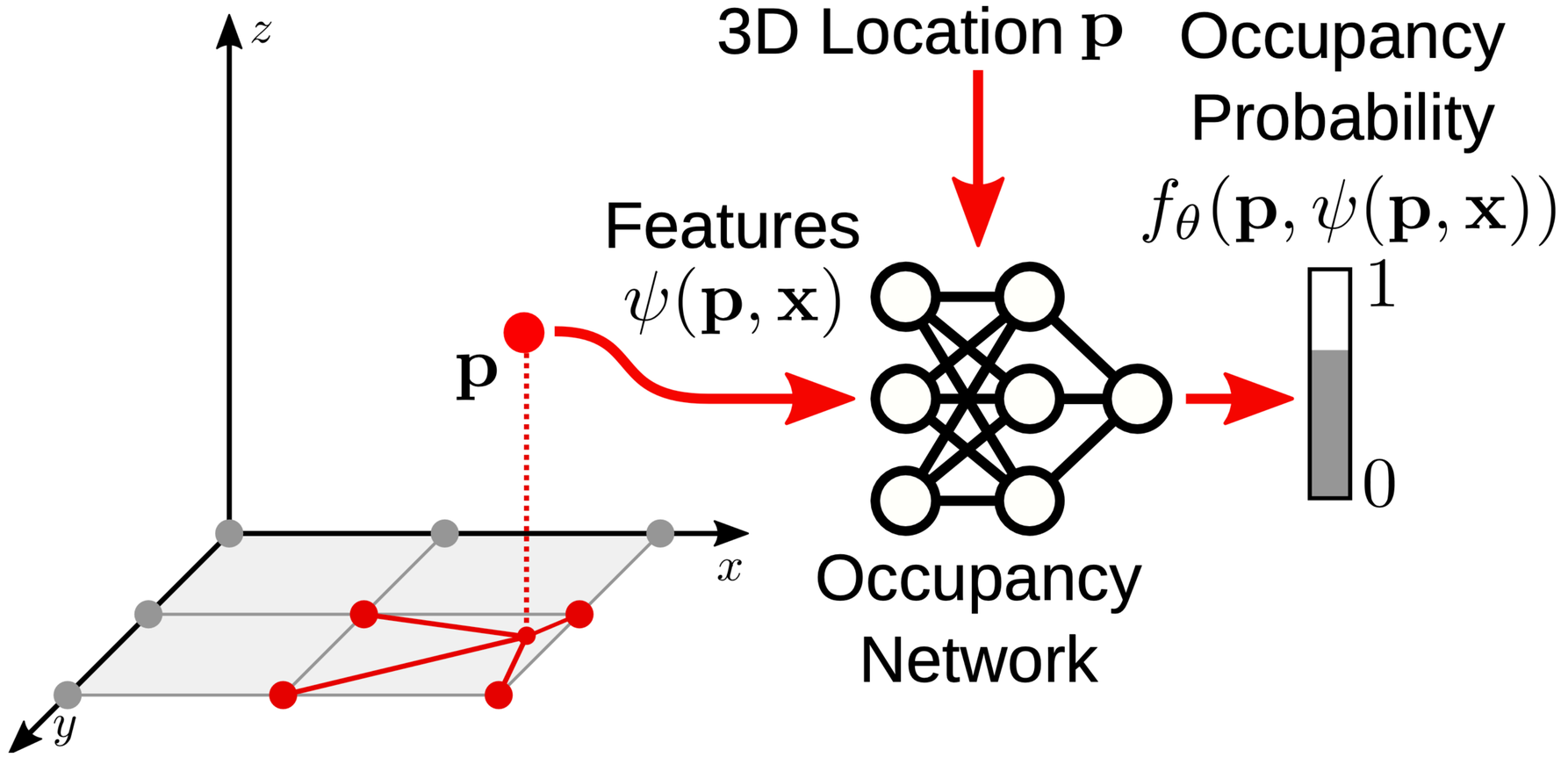
Store grid of trained features

Simple MLP takes features + position



Convolutional Occupancy Networks [Peng et al. 2020]
Local Implicit Grid Representations for 3D Scenes [Jiang et al. 2020]
Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion [Chibane et al. 2020]
Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction [Chabra et al. 2020]

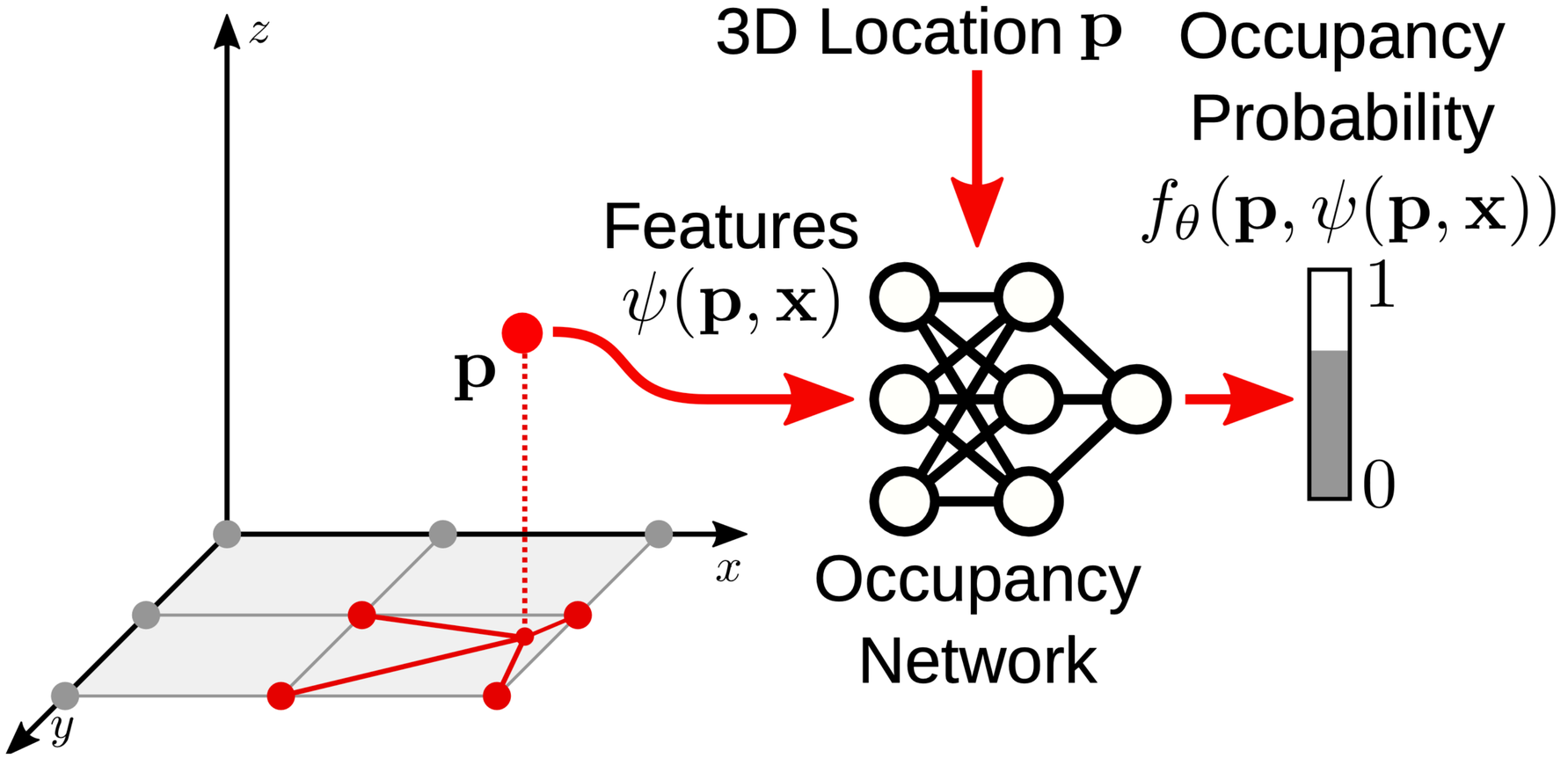
Ground plan & Orthographic Projection



Any assumptions made on scene? Any limitations?

Convolutional Occupancy Networks [Peng et al. 2020]

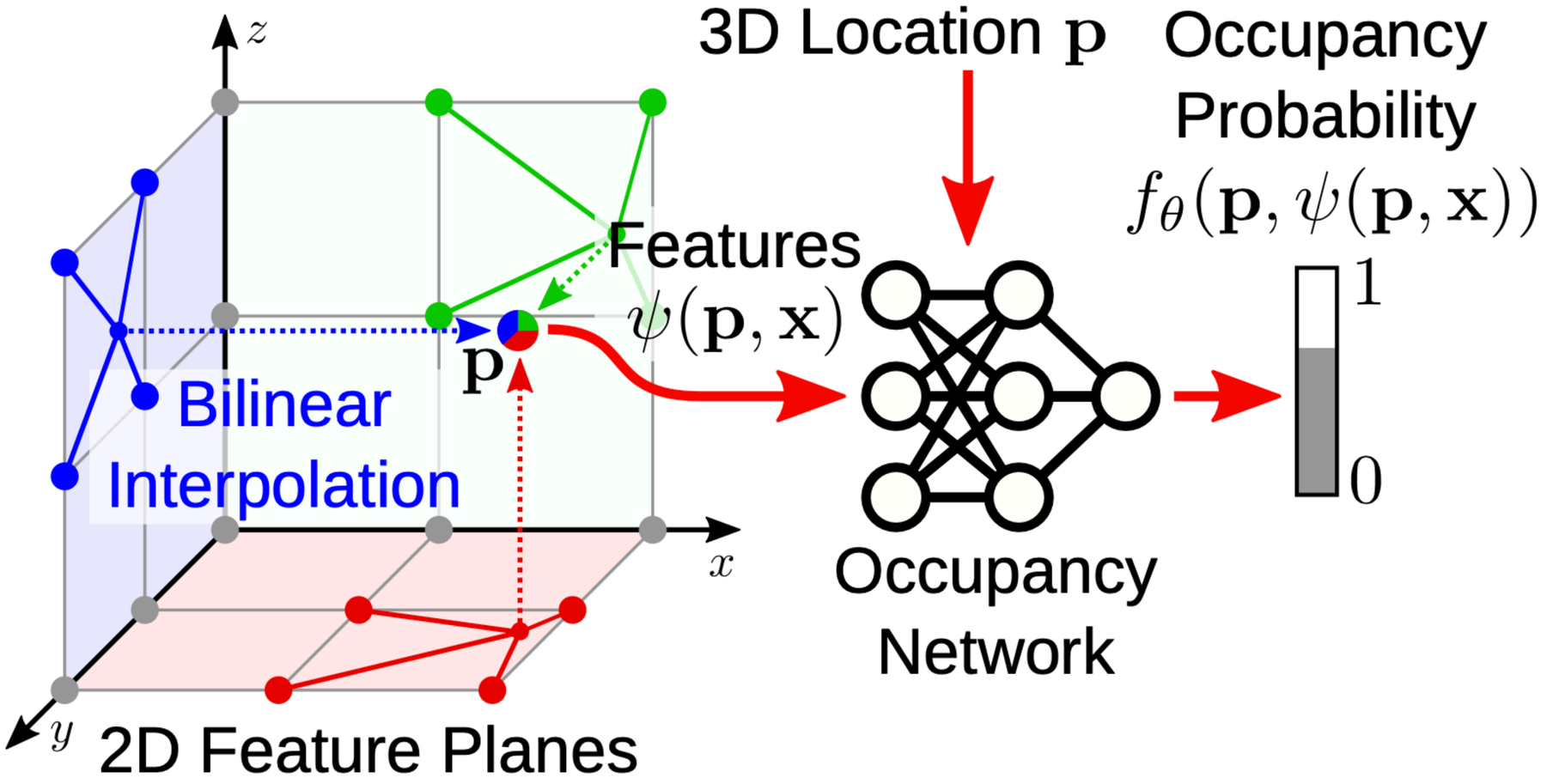
Ground plan & Orthographic Projection



No. You can still represent *any scene* - some info stored in NN, some in grid.

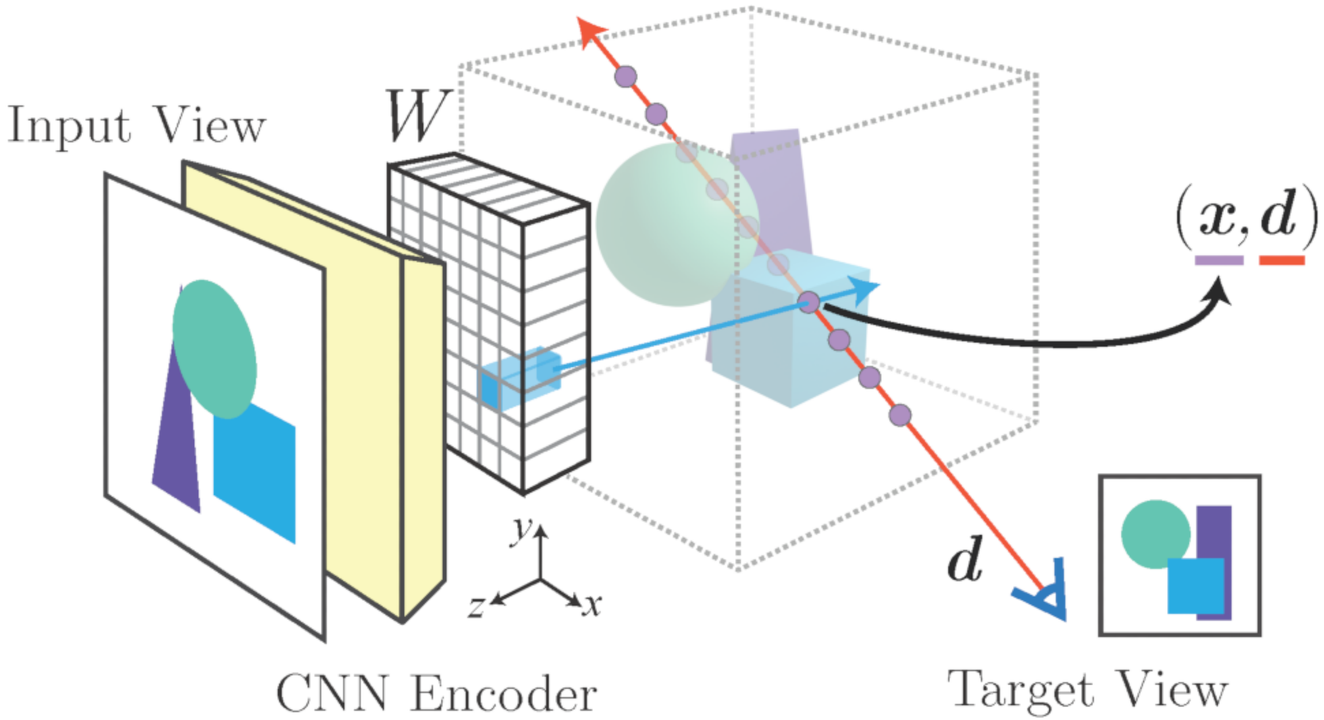
Convolutional Occupancy Networks [Peng et al. 2020]

Tri-Planes and Orthographic Projection



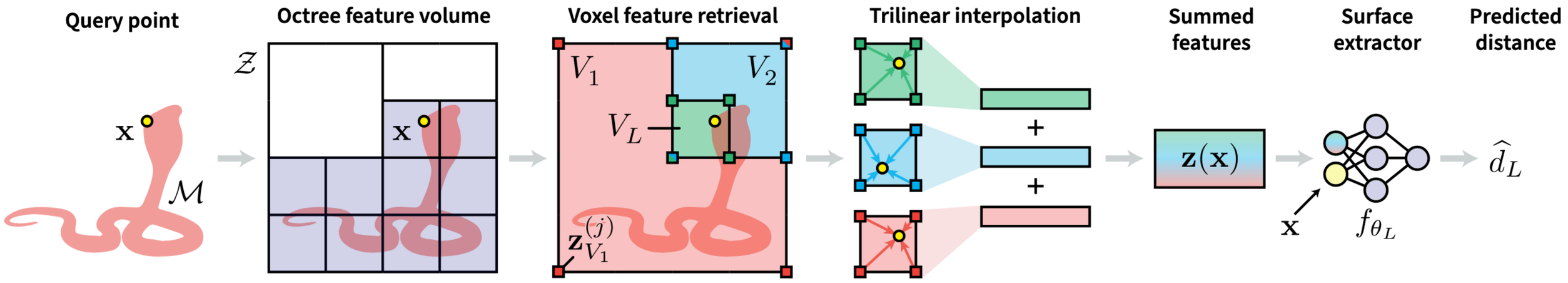
Convolutional Occupancy Networks [Peng et al. 2020]

Image Grids & Perspective Projection



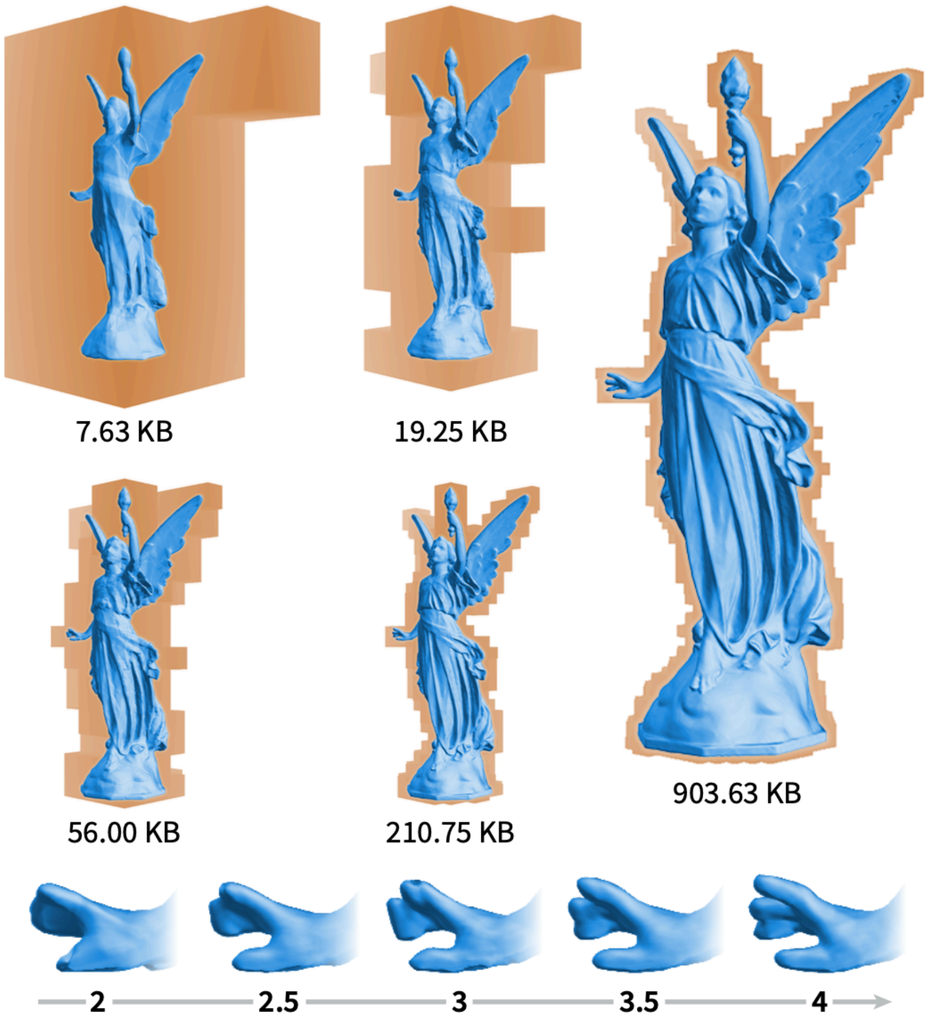
PixelNeRF (Yu et al. 2020)

Multi-scale Voxel Grids



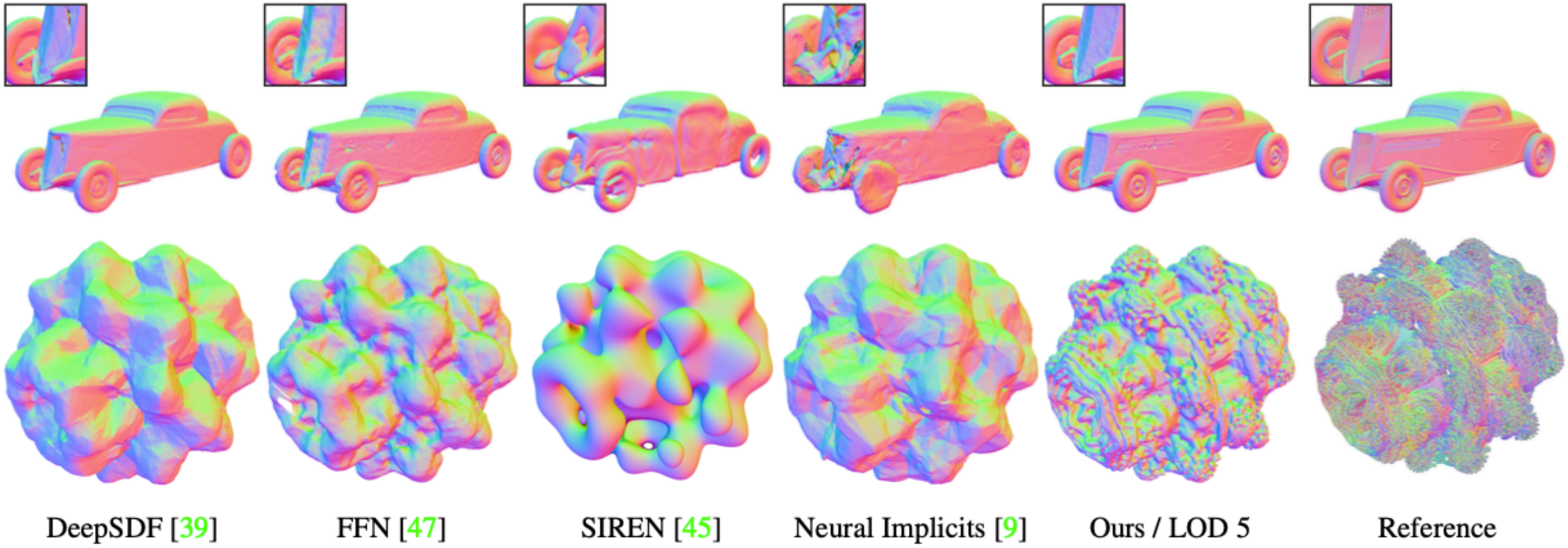
Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes, Takikawa, Litalien et al. 2021

Multi-scale Voxel Grids



Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes, Takikawa, Litalien et al. 2021

Multi-scale Voxel Grids



Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes, Takikawa, Litalien et al. 2021

Instant NGP

Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

THOMAS MÜLLER, NVIDIA, Switzerland
ALEX EVANS, NVIDIA, United Kingdom
CHRISTOPH SCHIED, NVIDIA, USA
ALEXANDER KELLER, NVIDIA, Germany

https://nvlabs.github.io/instant-ngp

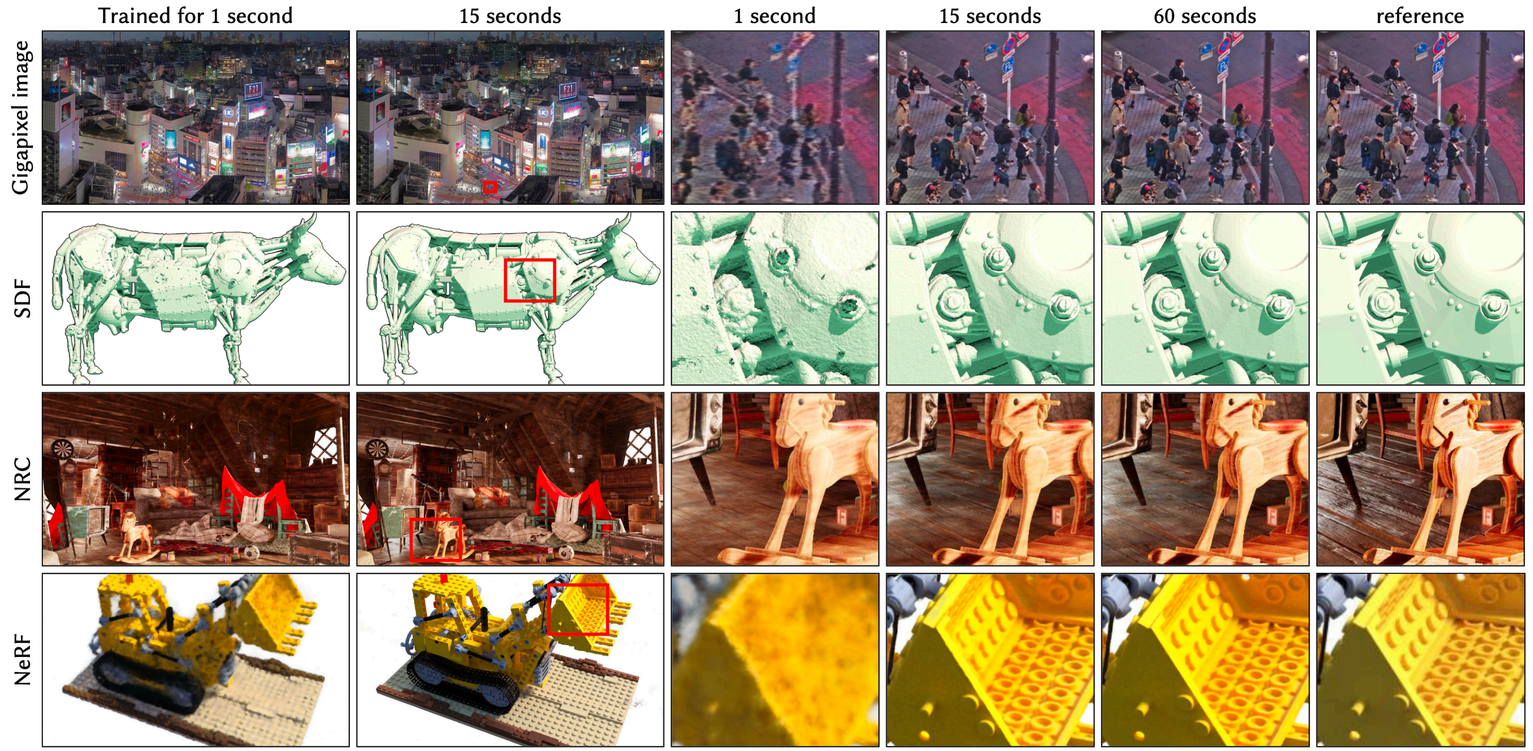


Fig. 1. We demonstrate instant training of neural graphics primitives on a single GPU for multiple tasks. In *Gigapixel image* we represent a gigapixel image by a neural network. *SDF* learns a signed distance function in 3D space whose zero level-set represents a 2D surface. Neural radiance caching (*NRC*) [Müller et al. 2021] employs a neural network that is trained in real-time to cache costly lighting calculations. Lastly, *NeRF* [Mildenhall et al. 2020] uses 2D images and their camera poses to reconstruct a volumetric radiance-and-density field that is visualized using ray marching. In all tasks, our encoding and its efficient implementation provide clear benefits: rapid training, high quality, and simplicity. Our encoding is task-agnostic: we use the same implementation and hyperparameters across all tasks and only vary the hash table size which trades off quality and performance. Photograph ©Trevor Dobson (CC BY-NC-ND 2.0)

https://nvlabs.github.io/instant-ngp/

Instant NGP

- L levels of sparse grids at different resolutions stores feature vectors
  - Each level is independent. Features will be concatenated
  - Each level has fixed size
- MLP computes field given interpolated featuresTo query a grid at location x:
  - Discretize x to 8 corners at grid resolution
  - For each corner, read feature in hash table
  - Big trick for performance : live with collisions, hope the MLP will handle it. One thing that helps: collisions are uncorrelated at different scales

Showing only 2 levels

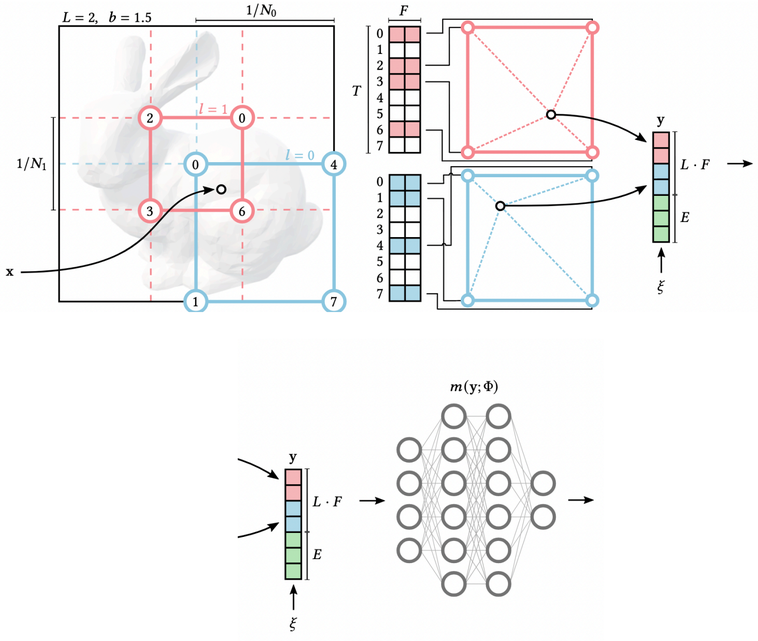
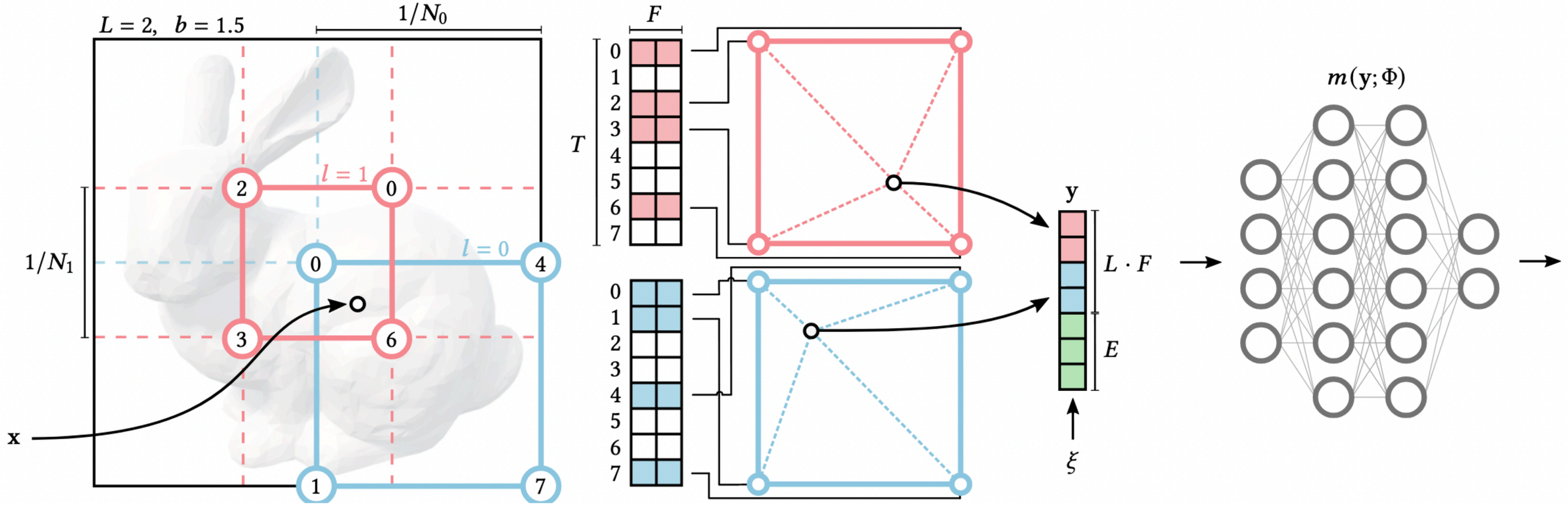


Table 1. Hash encoding parameters and their ranges in our results. Only the hash table size T and max. resolution $N_{\max}$ need to be tuned to the task.

Parameter	Symbol	Value
Number of levels	L	16
Max. entries per level (hash table size)	T	2^{14} to 2^{24}
Number of feature dimensions per entry	F	2
Coarsest resolution	$N_{\min}$	16
Finest resolution	$N_{\max}$	512 to 524288

Multi-scale Voxel Grids + Hash Table



Instant Neural Graphics Primitives with a Multiresolution Hash Encoding, Müller et al. 2022

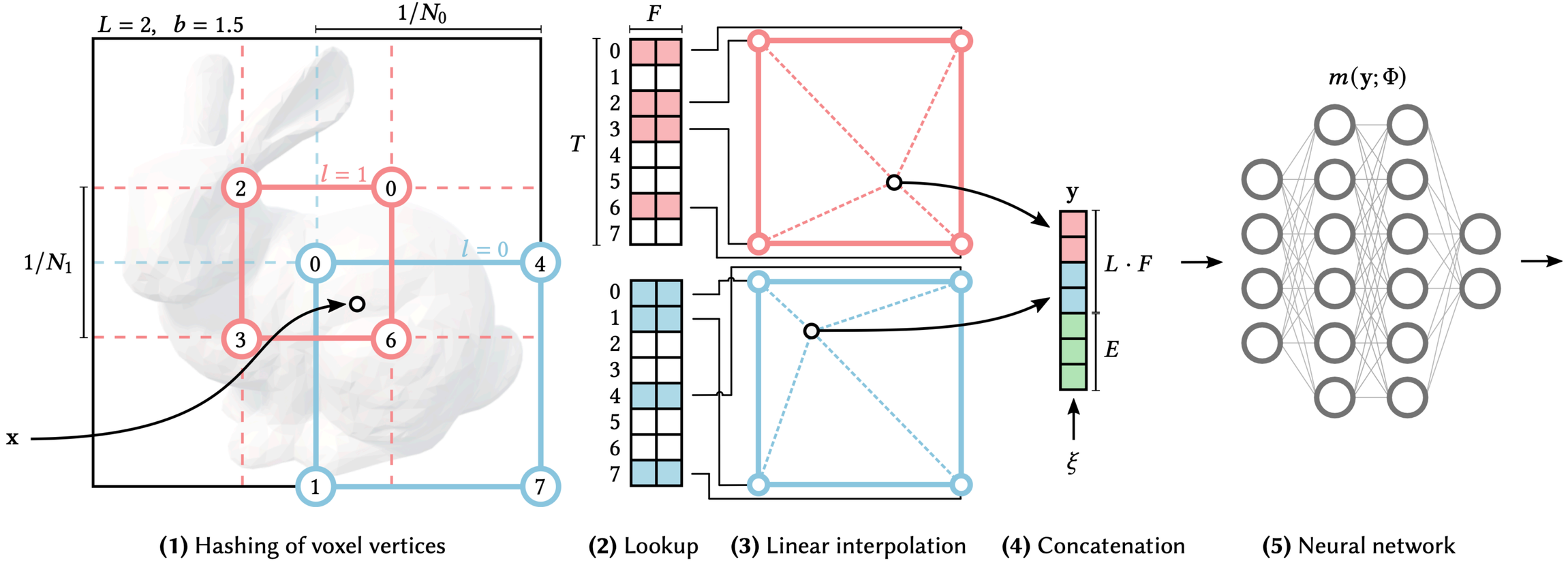
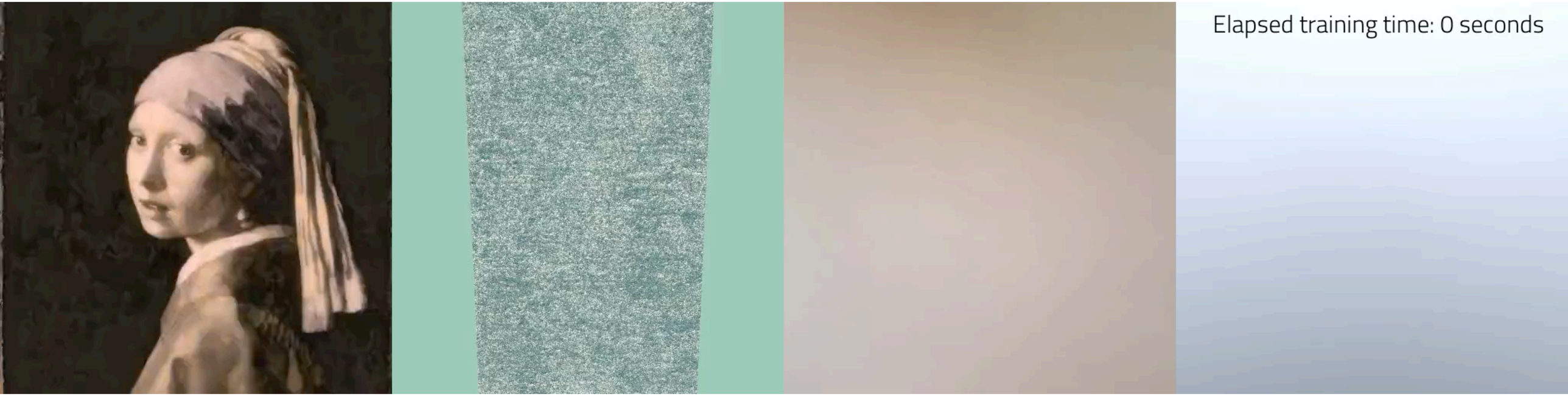


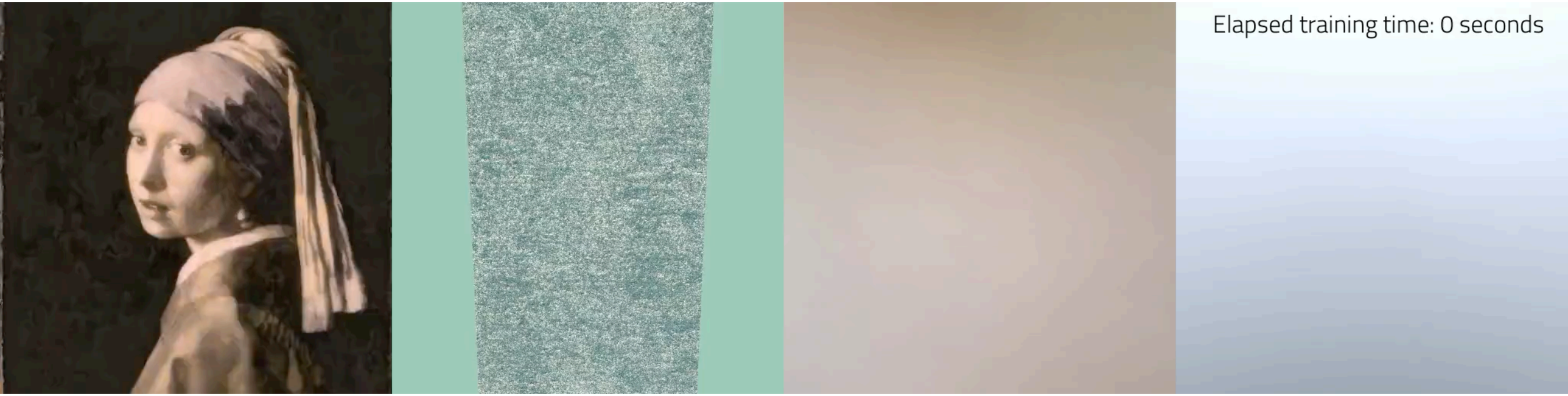
Fig. 3. Illustration of the multiresolution hash encoding in 2D. **(1)** for a given input coordinate $\mathbf{x}$, we find the surrounding voxels at L resolution levels and assign indices to their corners by hashing their integer coordinates. **(2)** for all resulting corner indices, we look up the corresponding F-dimensional feature vectors from the hash tables θ_l and **(3)** linearly interpolate them according to the relative position of $\mathbf{x}$ within the respective l-th voxel. **(4)** we concatenate the result of each level, as well as auxiliary inputs $\xi \in \mathbb{R}^E$, producing the encoded MLP input $y \in \mathbb{R}^{LF+E}$, which **(5)** is evaluated last. To train the encoding, loss gradients are backpropagated through the MLP **(5)**, the concatenation **(4)**, the linear interpolation **(3)**, and then accumulated in the looked-up feature vectors.

Multi-scale Voxel Grids + Hash Table



Instant Neural Graphics Primitives with a Multiresolution Hash Encoding, Müller et al. 2022

Multi-scale Voxel Grids + Hash Table



Instant Neural Graphics Primitives with a Multiresolution Hash Encoding, Müller et al. 2022

INSTANT NEURAL GRAPHICS PRIMITIVES WITH A MULTIRESOLUTION HASH ENCODING

Thomas Müller Alex Evans Christoph Schied Alexander Keller

https://nvlabs.github.io/instant-ngp



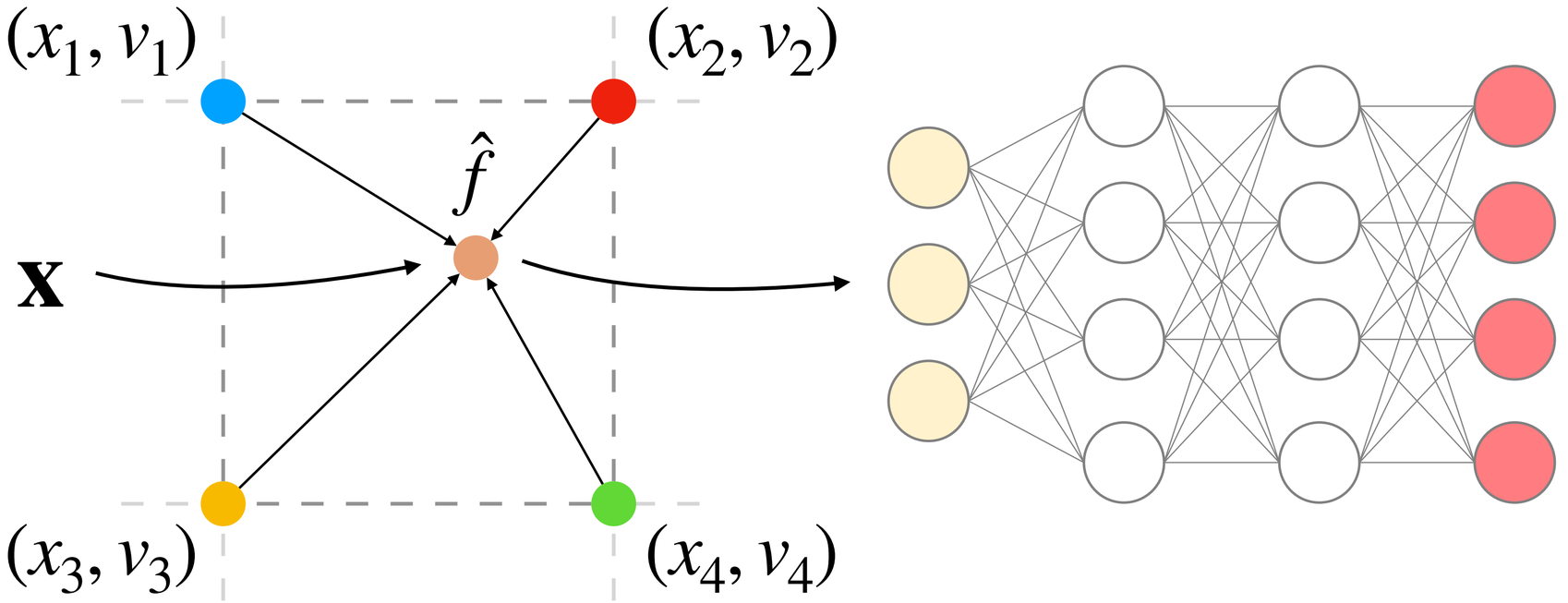
INSTANT NEURAL GRAPHICS PRIMITIVES WITH A MULTIRESOLUTION HASH ENCODING

Thomas Müller Alex Evans Christoph Schied Alexander Keller

https://nvlabs.github.io/instant-ngp

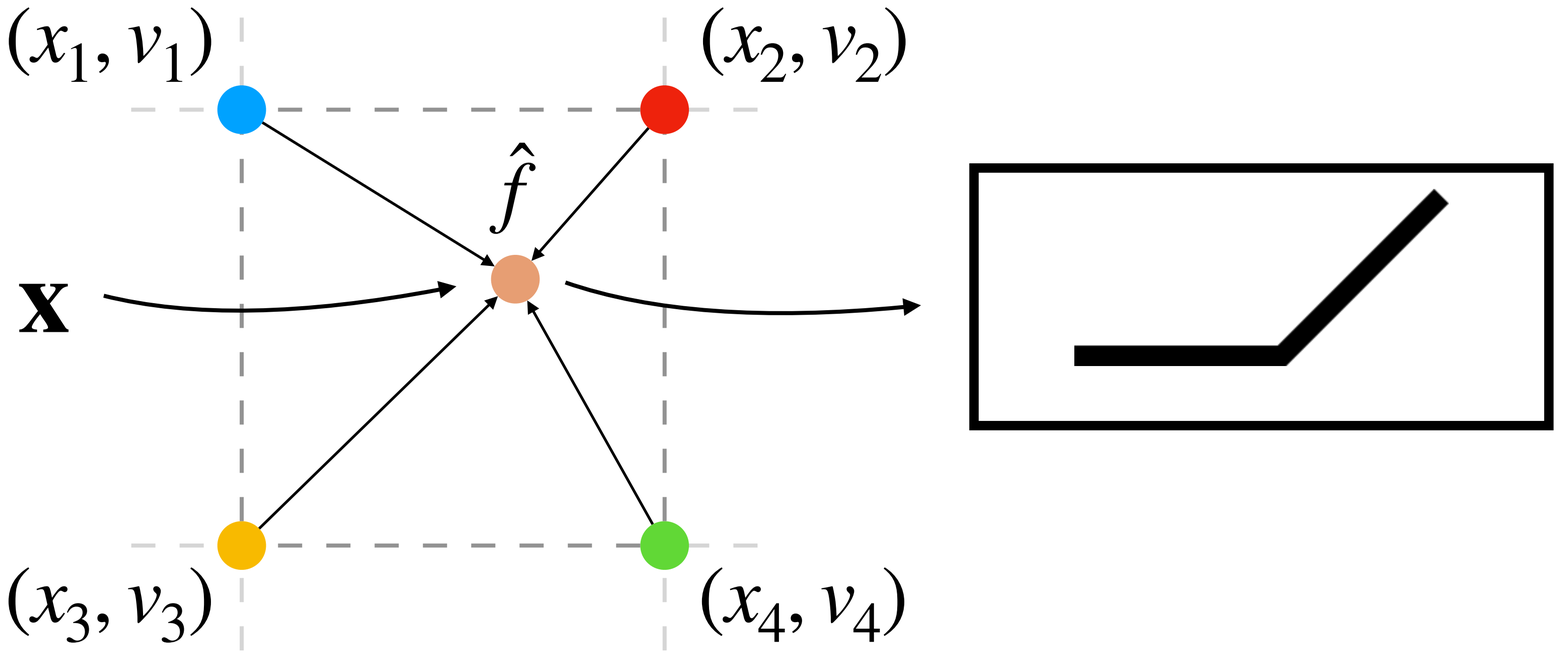


ReLU Fields: How much MLP do you need?



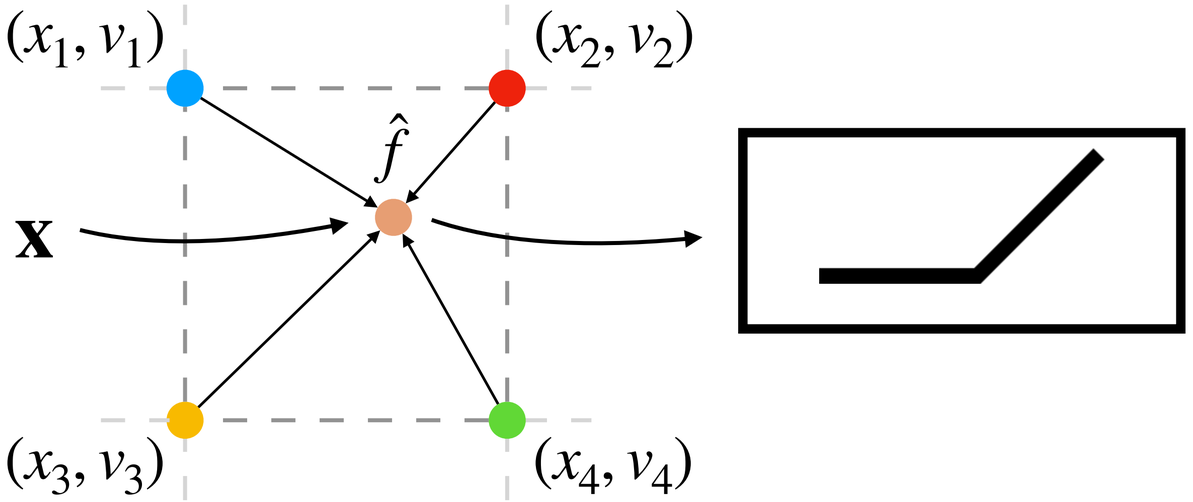
ReLU Fields: The Little Non-linearity That Could (Karnewar et al. 2022)

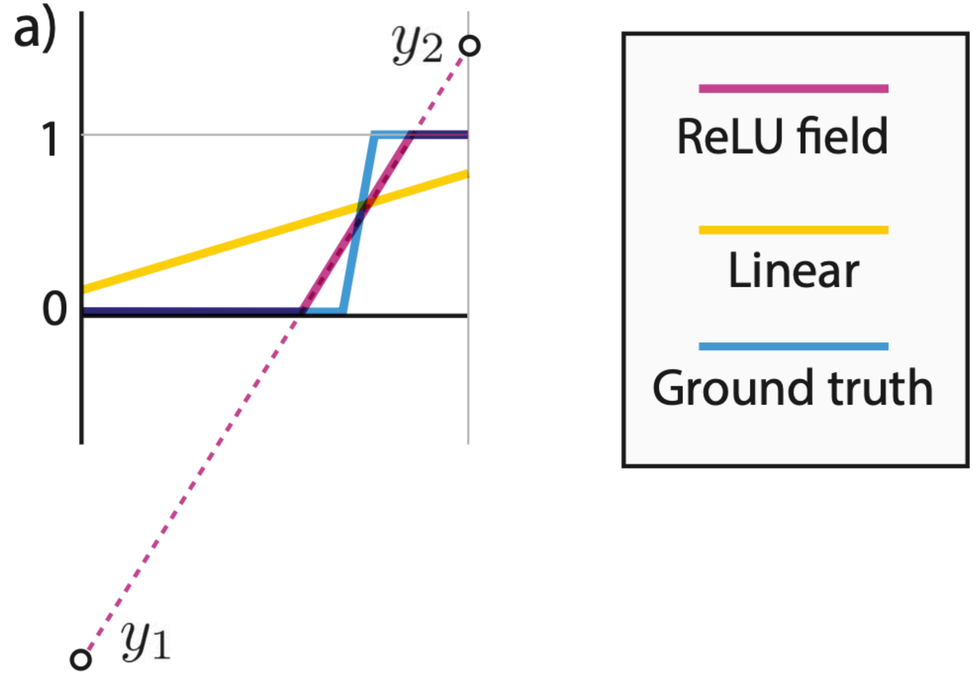
ReLU Fields: How much MLP do you need?



ReLU Fields: The Little Non-linearity That Could (Karnewar et al. 2022)

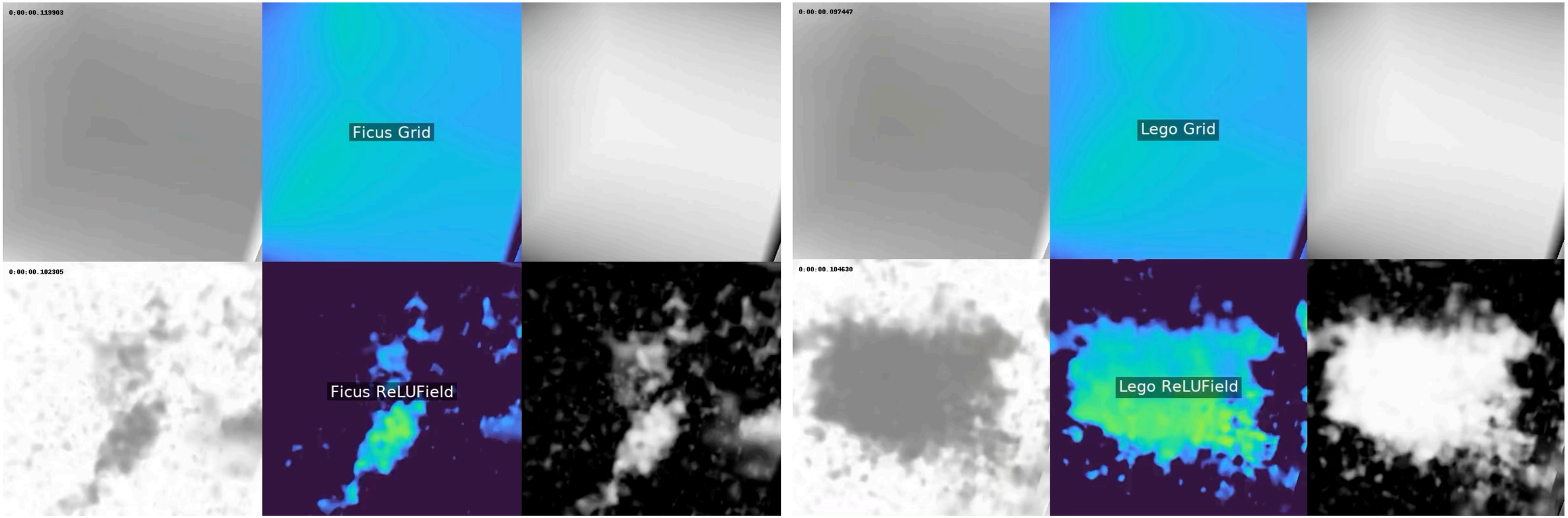
ReLU Fields: How much MLP do you need?





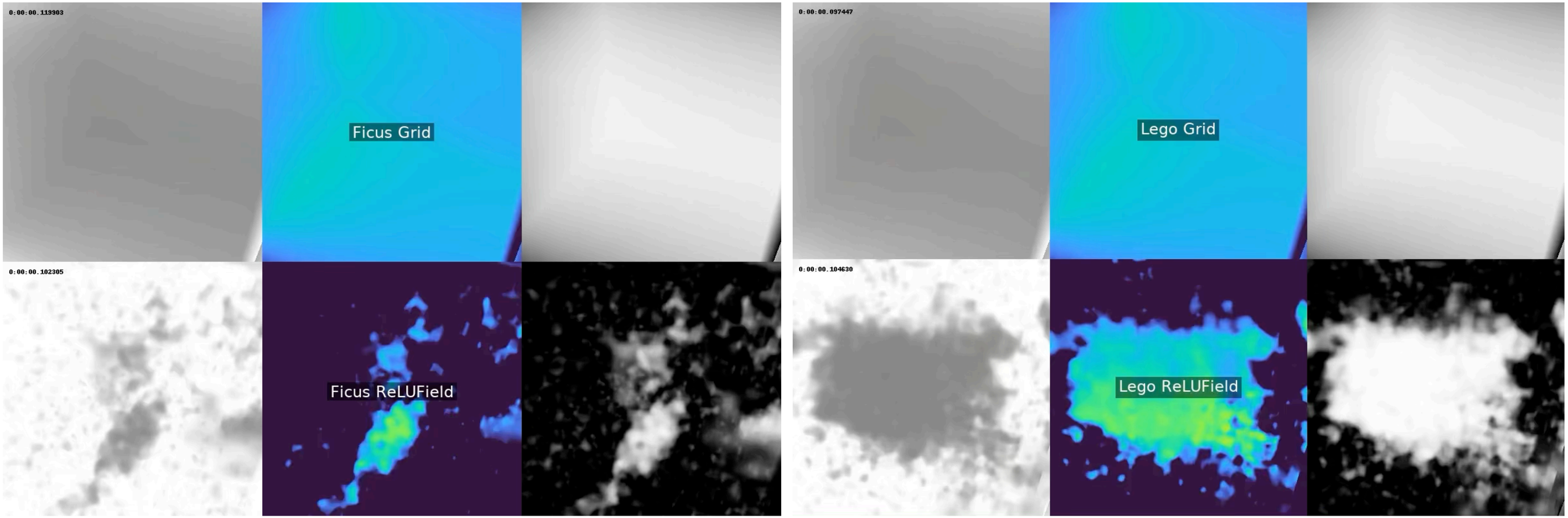
ReLU Fields: The Little Non-linearity That Could (Karnewar et al. 2022)

ReLU Fields: How much MLP do you need?



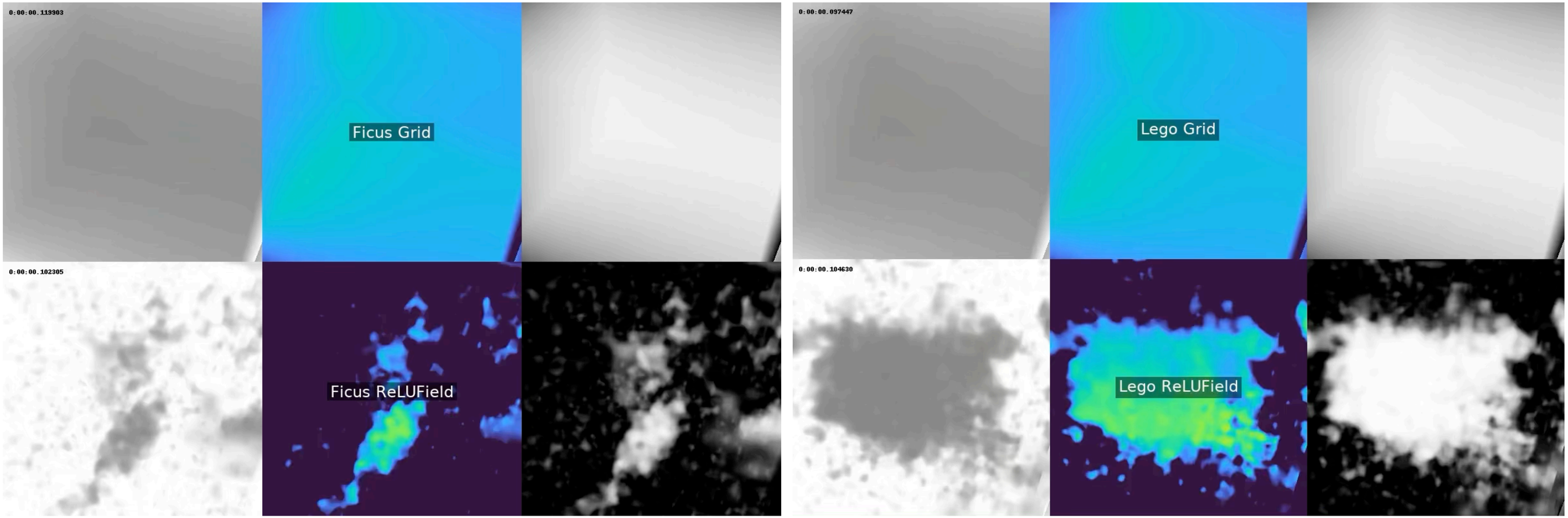
ReLU Fields: The Little Non-linearity That Could (Karnewar et al. 2022)

ReLU Fields: How much MLP do you need?



ReLU Fields: The Little Non-linearity That Could (Karnewar et al. 2022)

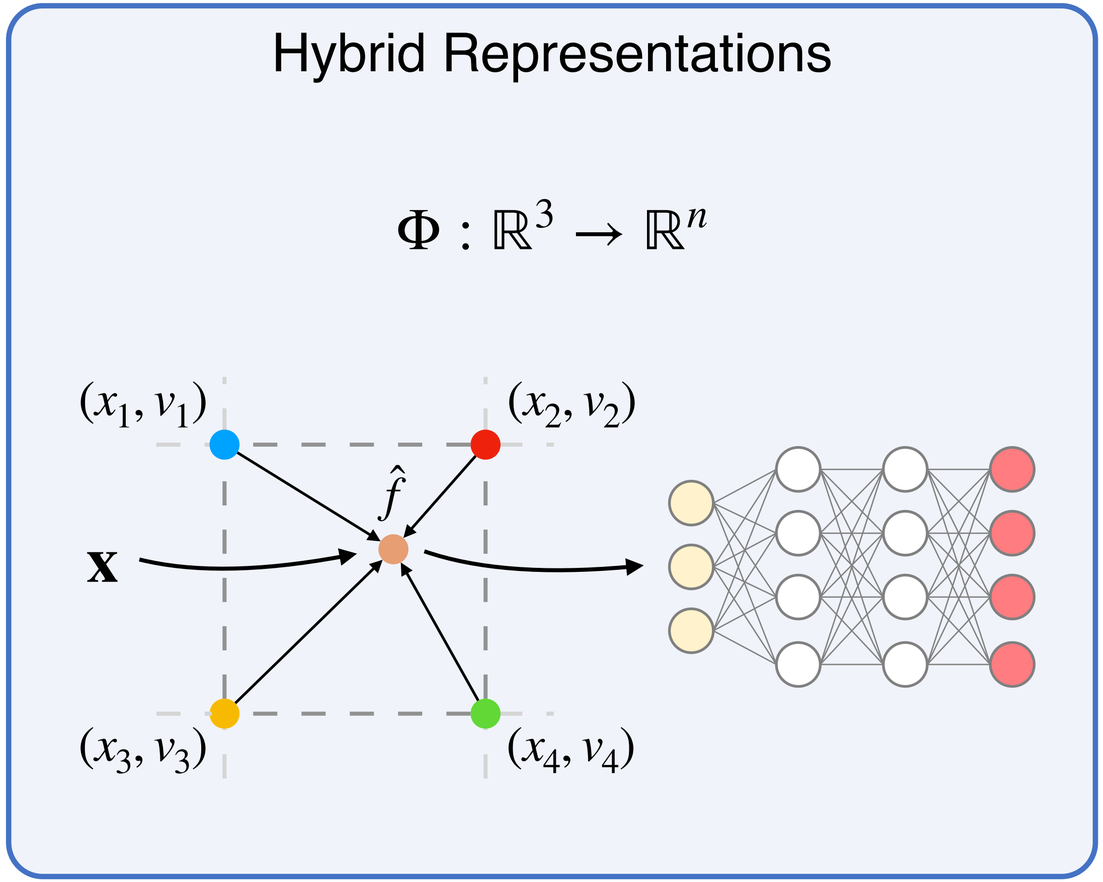
ReLU Fields: How much MLP do you need?



ReLU Fields: The Little Non-linearity That Could (Karnewar et al. 2022)

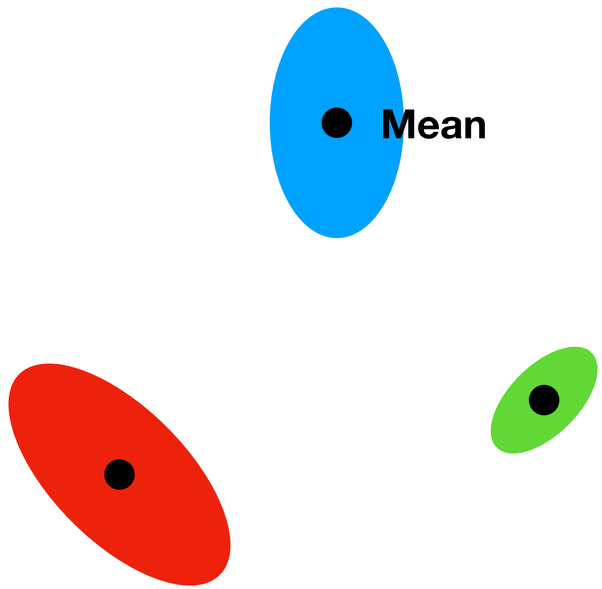
Hybrid Representations

- Very natural to trade off memory with compute.
- Neural network can locally “super resolve” discrete representation: Basically acts as interpolation kernel.
- Can be the best of both worlds: Fast and “continuous”.



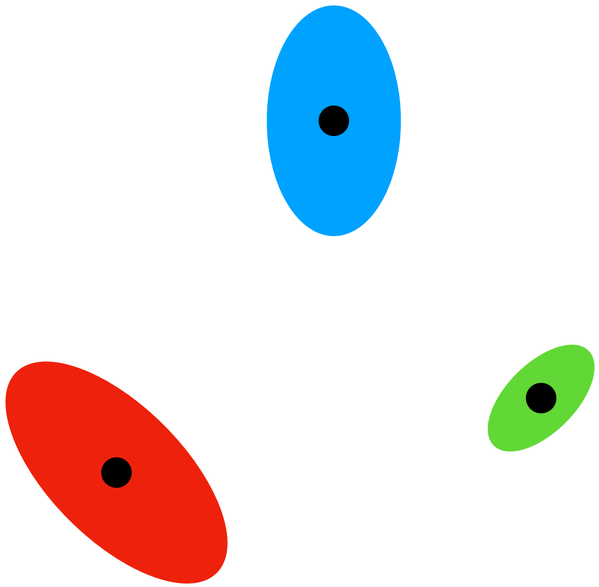
Primitive-based approaches

Primitive-based representations



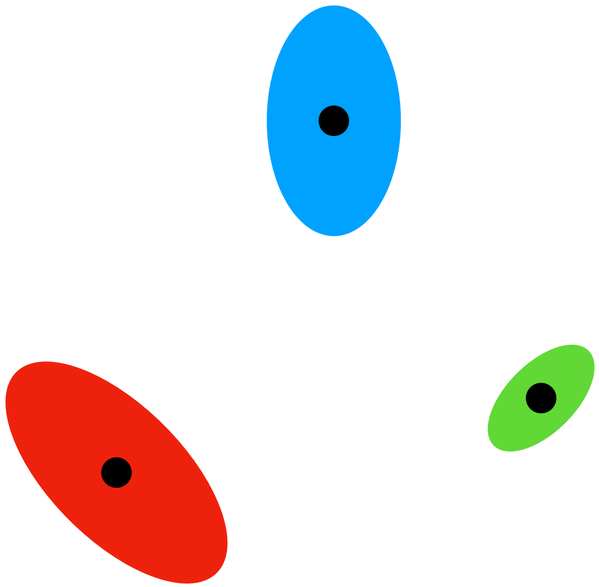
3D Gaussian Blobs floating in Space

Primitive-based representations



3D Gaussian Blobs floating in Space

Primitive-based representations

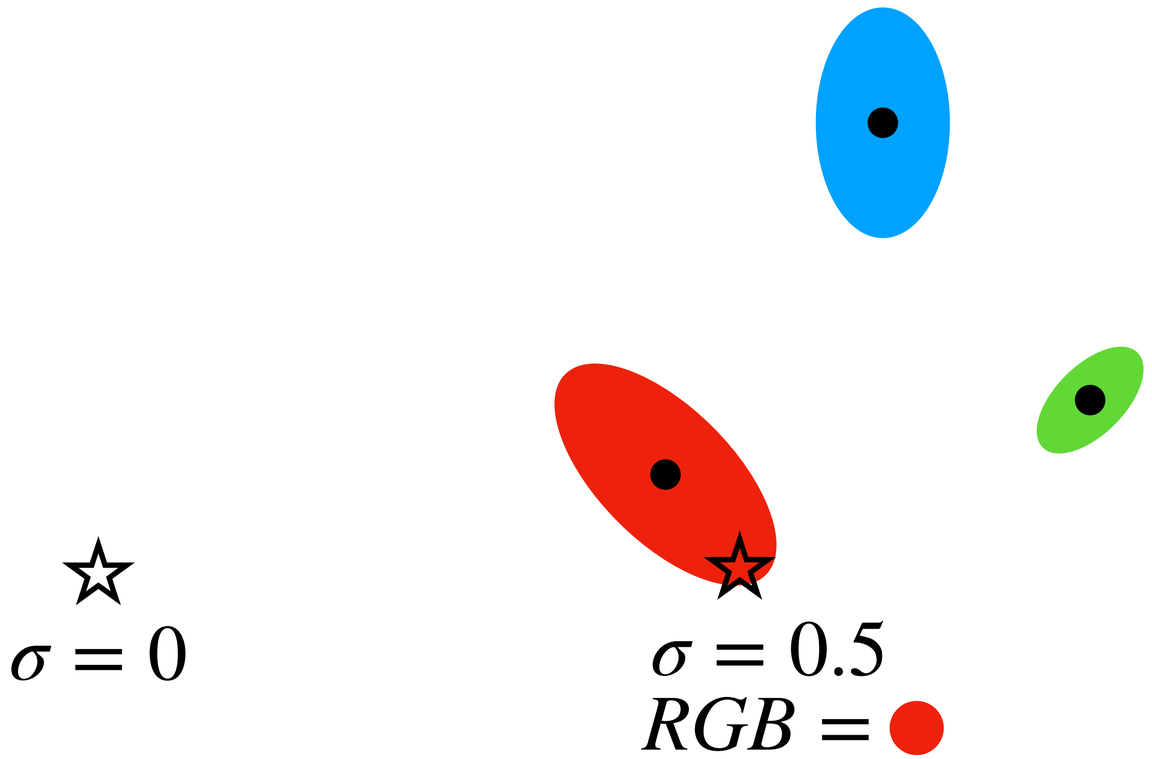


3D Gaussian Blobs floating in Space

☆

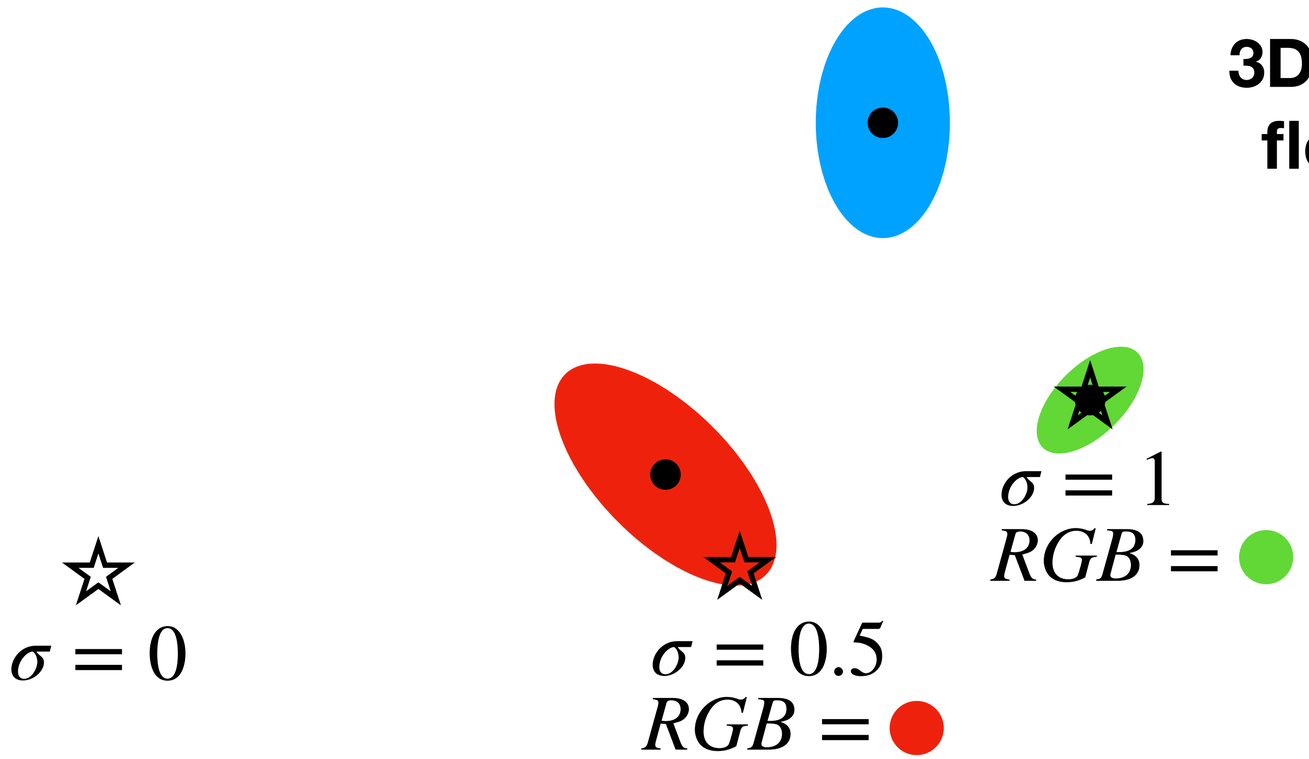
$\sigma = 0$

Primitive-based representations



3D Gaussian Blobs floating in Space

Primitive-based representations



Rendering algorithm

- Original:
- Split image into tiles
- Per-tile sorting
- Approximation with 2D flat
- https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/

- See for stochastic version: https://ubc-vision.github.io/stochasticsplats/

- New version (EVER)
- Ray tracing
- https://half-potato.gitlab.io/ever/
-

Anisotropic Volumetric 3D Gaussians



Final Rendering

3D Gaussian Visualization

Kerblk, Kopanas et al, 3D Gaussian Splatting for Real-Time Radiance Field Rendering

Anisotropic Volumetric 3D Gaussians



Final Rendering

3D Gaussian Visualization

Kerblk, Kopanas et al, 3D Gaussian Splatting for Real-Time Radiance Field Rendering

How can we represent unbounded scenes?



Image Source: Wikipedia

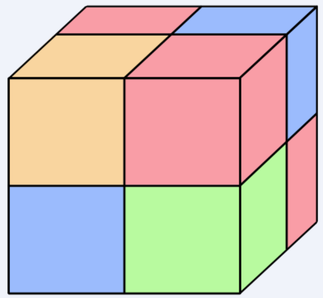
How can we represent unbounded scenes?



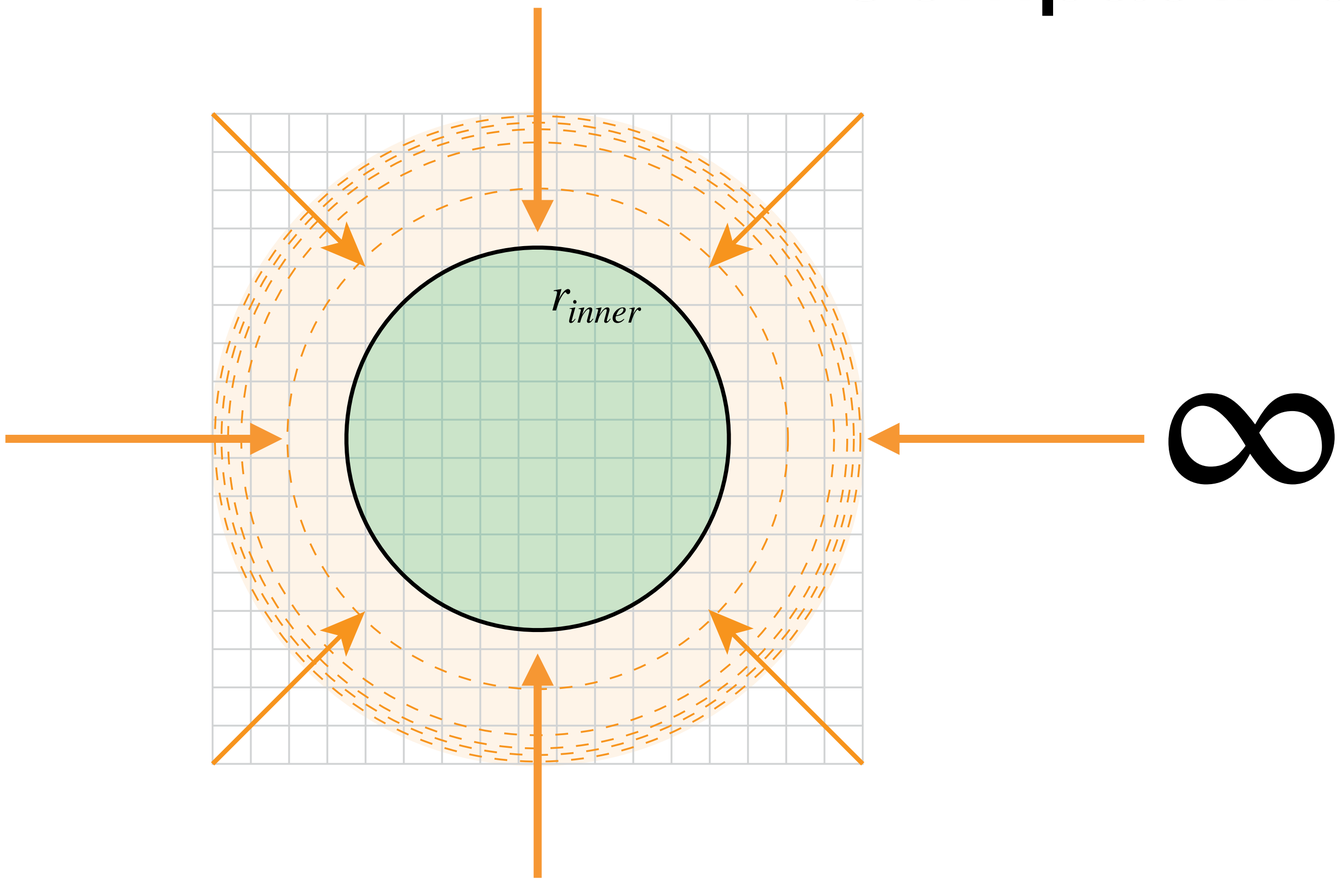
Image Source: Wikipedia

Scene Representation

$$\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}^n$$



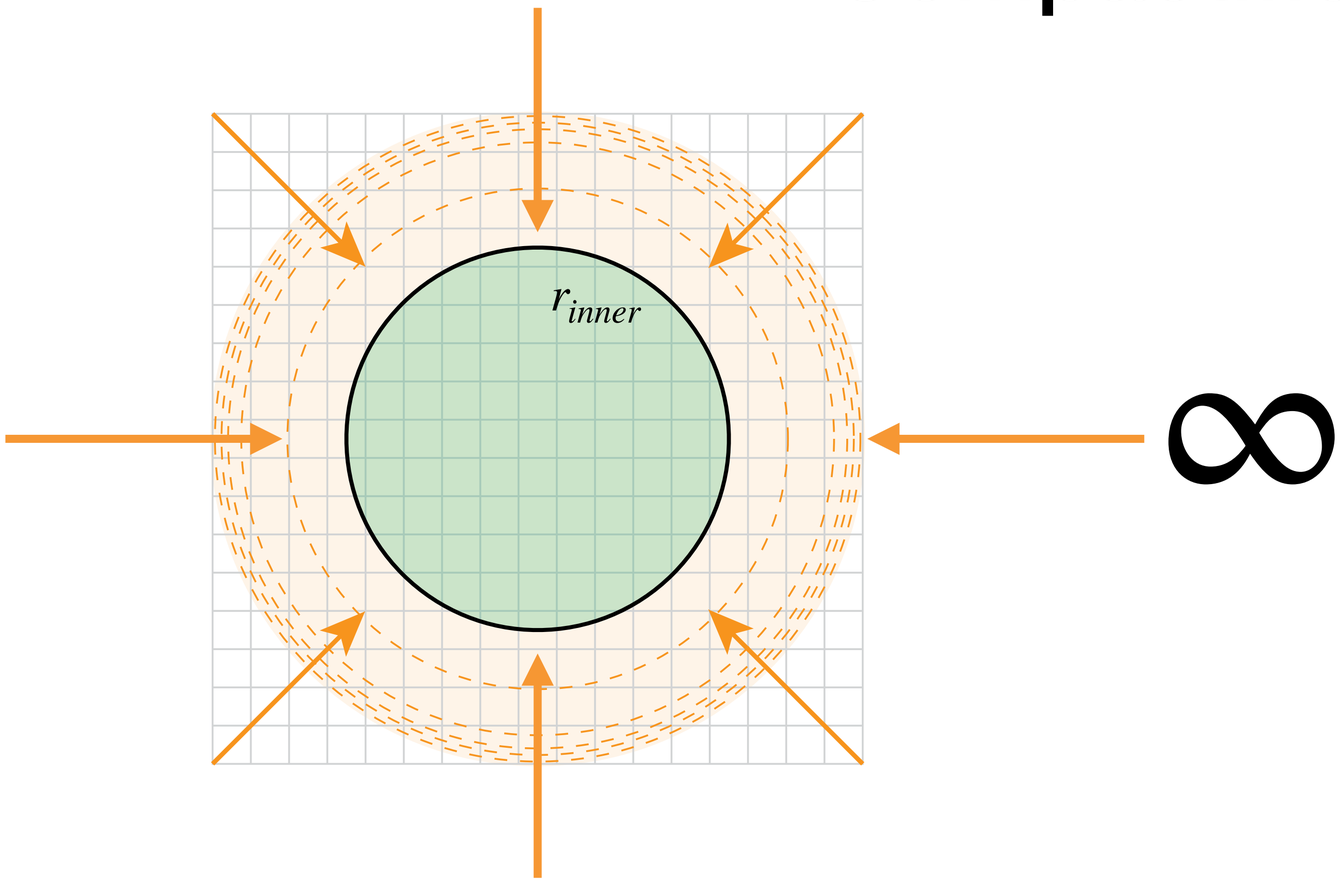
Parameterizing Unbounded Scenes: Compactification



Source: “Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields”, Barron et al., CVPR 2022

Come up with nonlinear mapping that “squashes” space, such that infinity is mapped to some **finite** value. Aligns with intuition “further away, less resolution”.

Parameterizing Unbounded Scenes: Compactification



Source: “Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields”, Barron et al., CVPR 2022

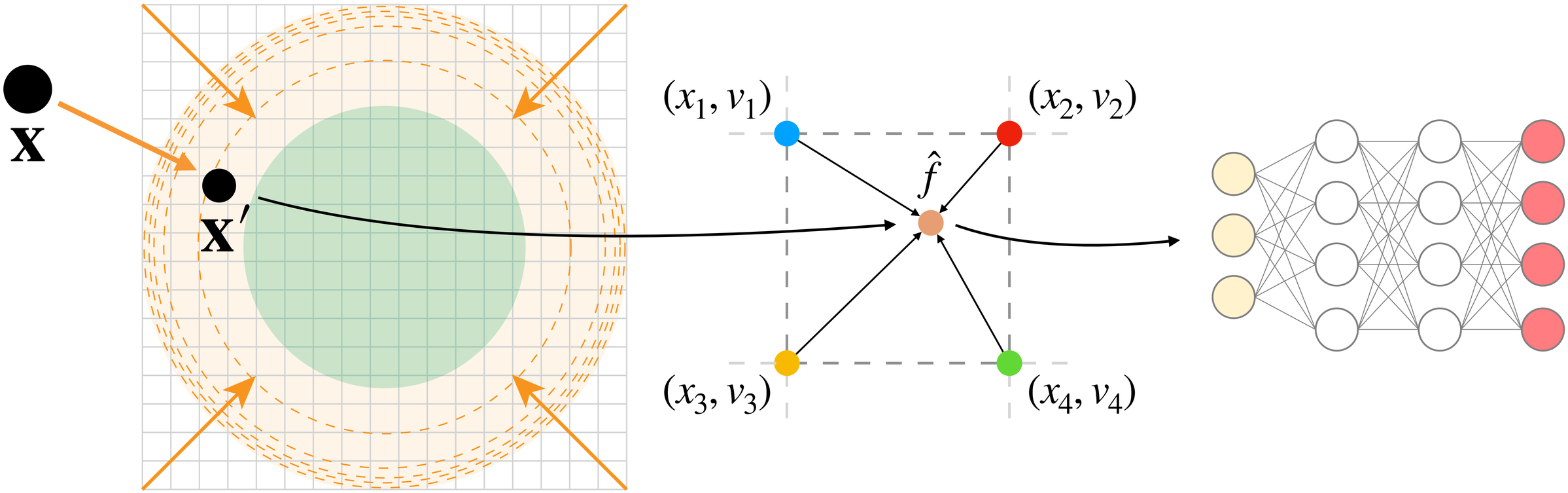
$$\mathbf{x}' = \frac{(1+k) - k/||\mathbf{u}||}{\mathbf{u}/||\mathbf{u}||} r_{inner}$$

$$\mathbf{u} = \frac{\mathbf{x}}{r_{inner}}$$

Example Compactification function from “Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields”, Barron et al., CVPR 2022

Come up with nonlinear mapping that “squashes” space, such that infinity is mapped to some **finite** value. Aligns with intuition “further away, less resolution”.

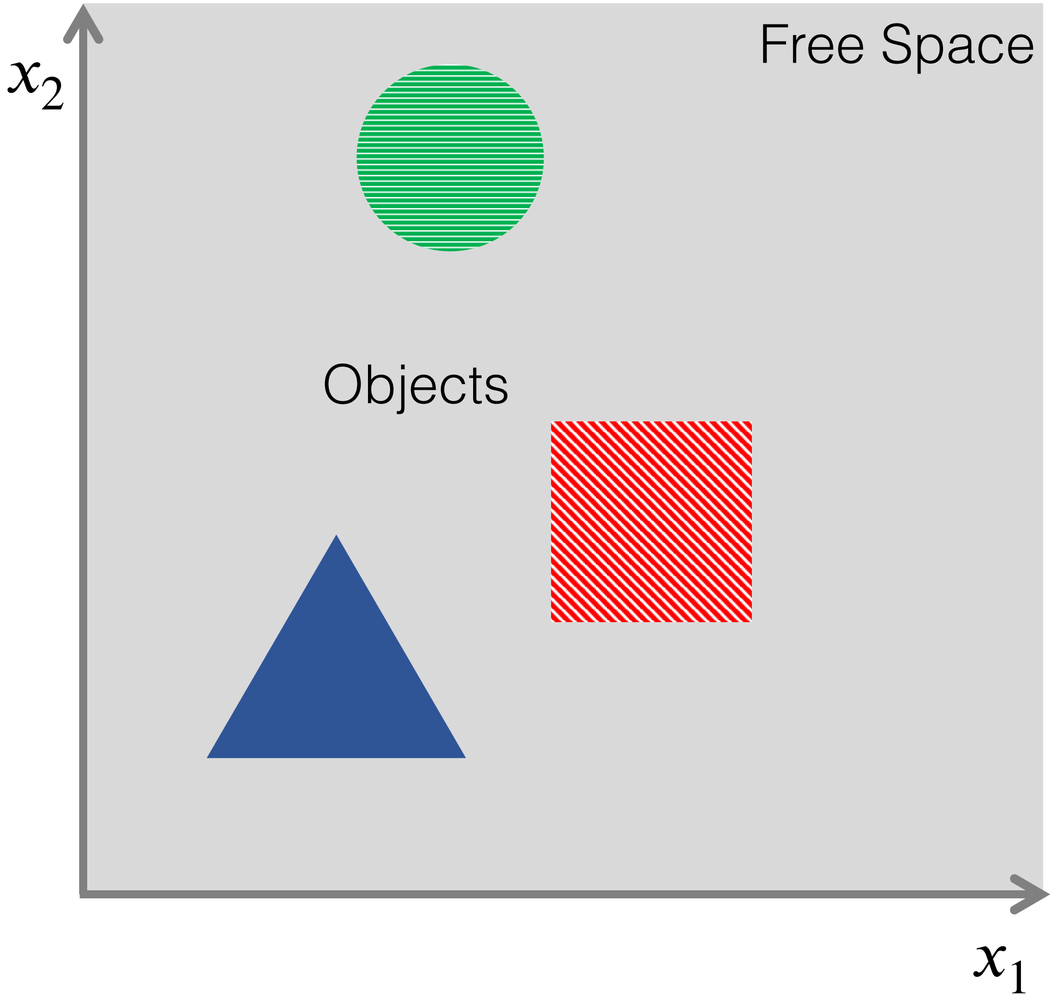
Parameterizing Unbounded Scenes: Compactification

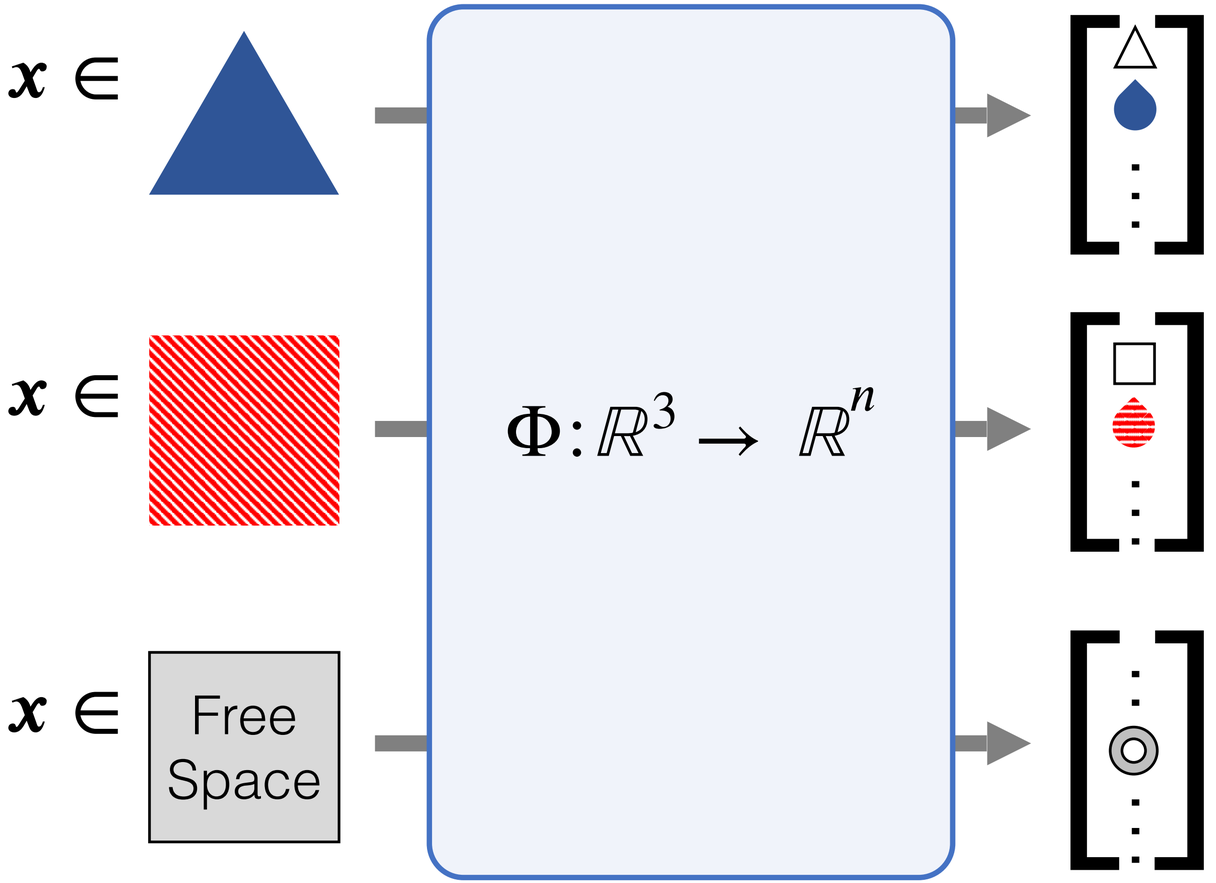


Source: “Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields”, Barron et al., CVPR 2022

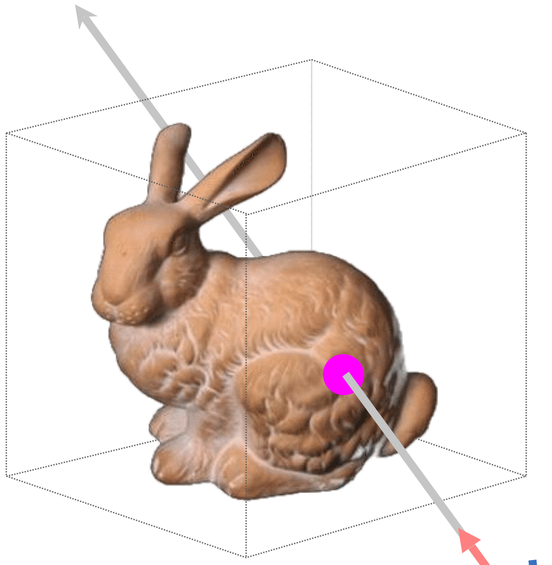
Light field approaches

Do we have to parameterize 3D Scene as 3D Function?





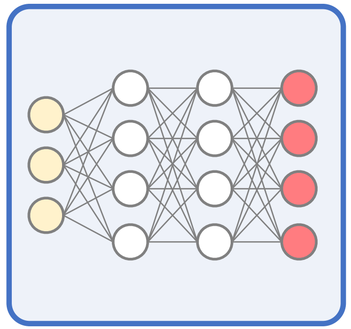
3D-structured Representation



$\mathbf{r}$

Camera Ray

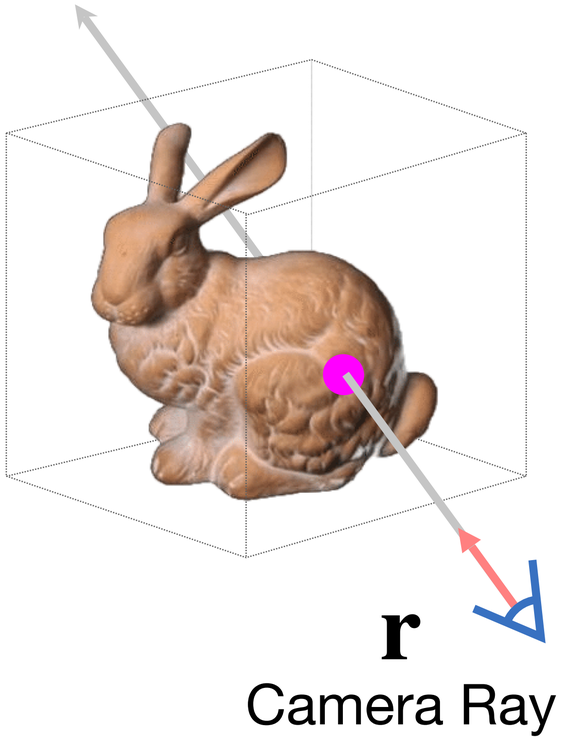




$\Phi^{3D} : \mathbb{R}^3 \rightarrow \mathbb{R}^n$

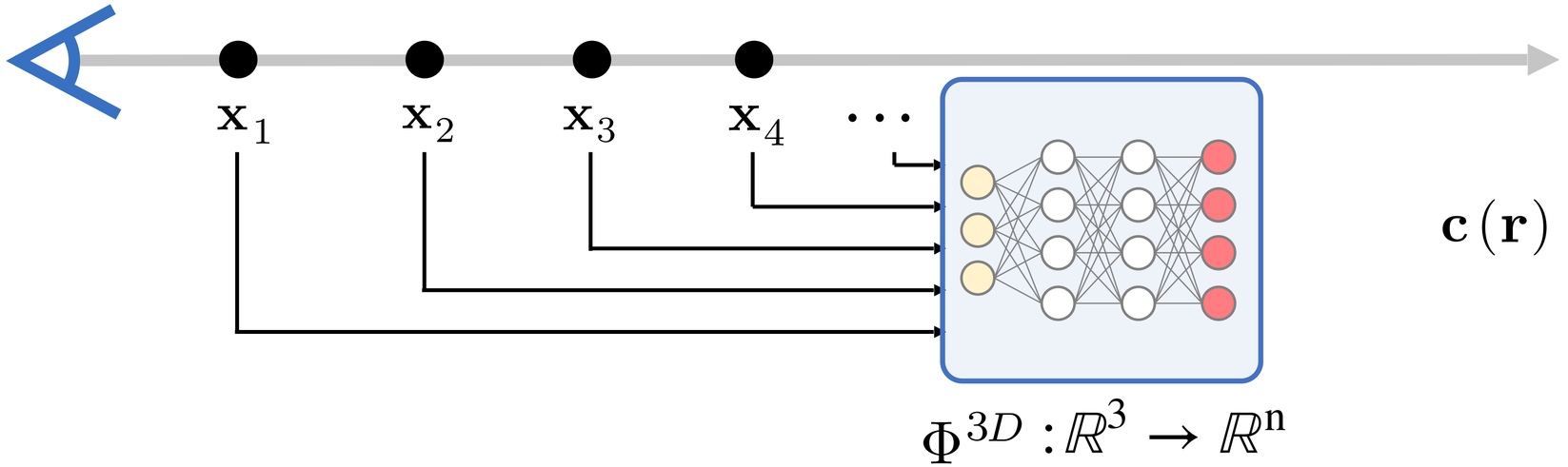
$\mathbf{c}(\mathbf{r}) =$ 

3D-structured Representation

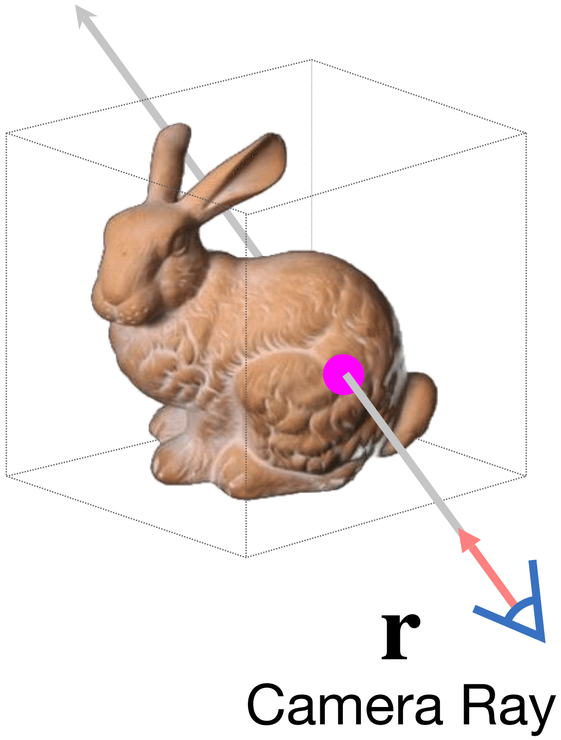


$\mathbf{r}$

Camera Ray

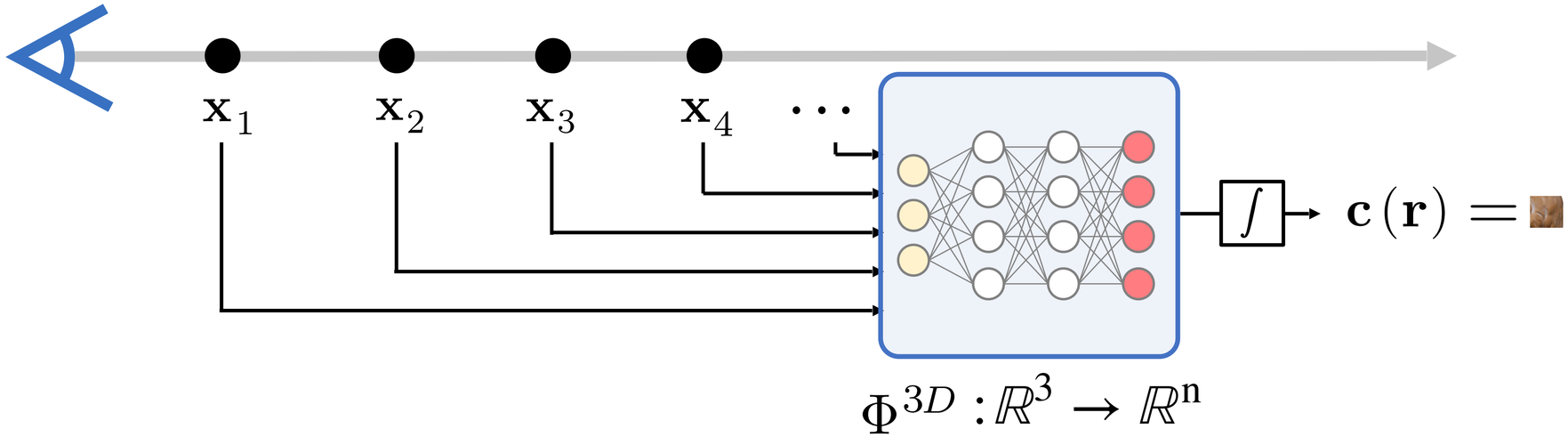


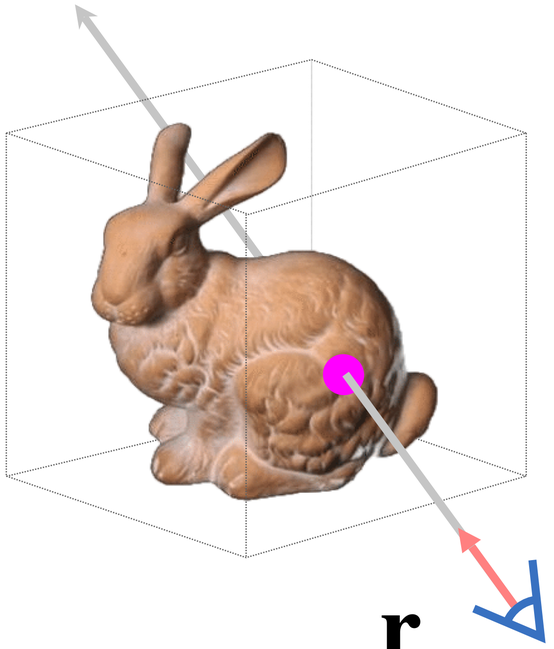
3D-structured Representation



$\mathbf{r}$

Camera Ray





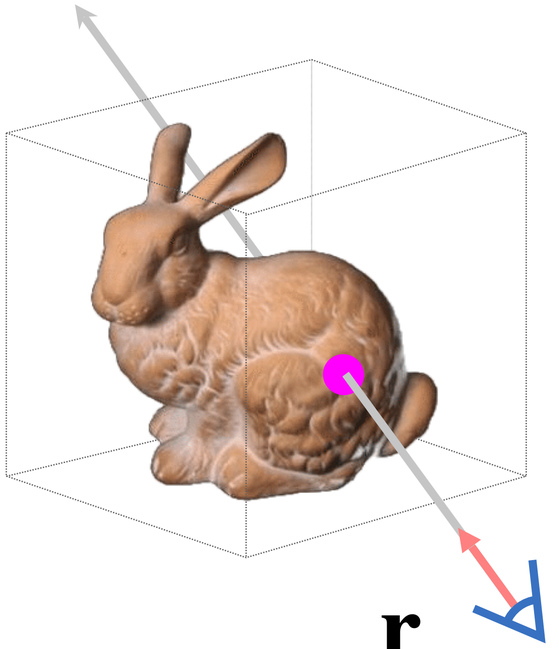
$\mathbf{r}$

Camera Ray

Light Field

$\mathbf{c}(\mathbf{r}) =$

[Adelson et al. 1991, Levoy et al. 1996, Gortler et al. 1996]



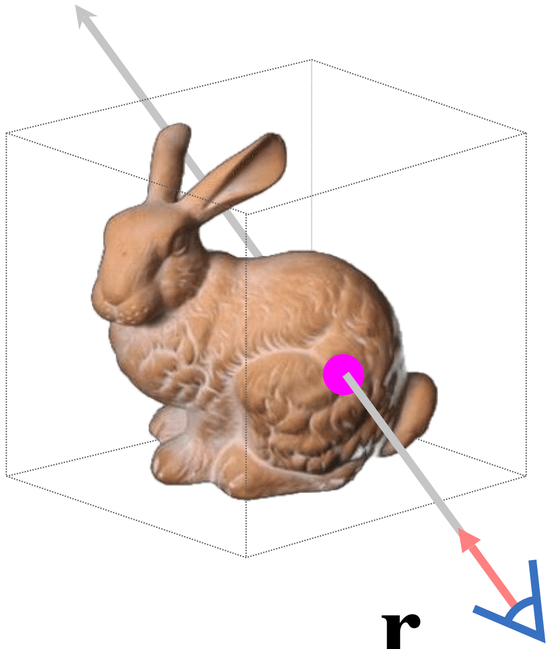
$\mathbf{r}$

Camera Ray

Light Field Networks

$\mathbf{c}(\mathbf{r}) =$

Light Field Networks



$\mathbf{r}$

Camera Ray

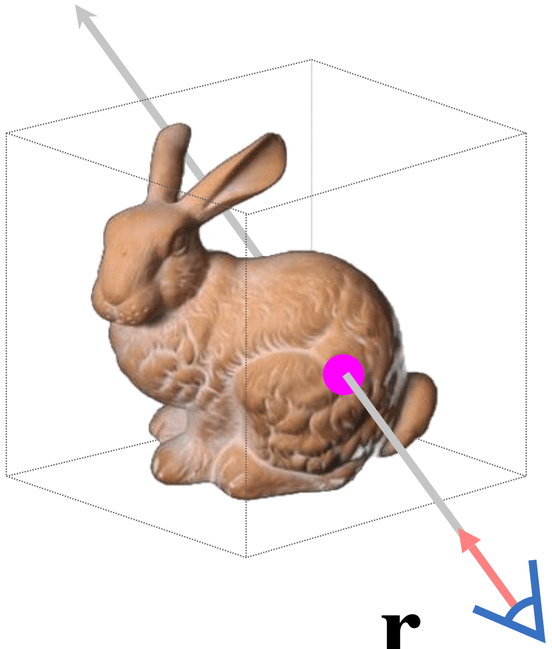
$\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_4$ $\cdots$

$\Phi^{3D} : R^3 \rightarrow R^n$

$\int$

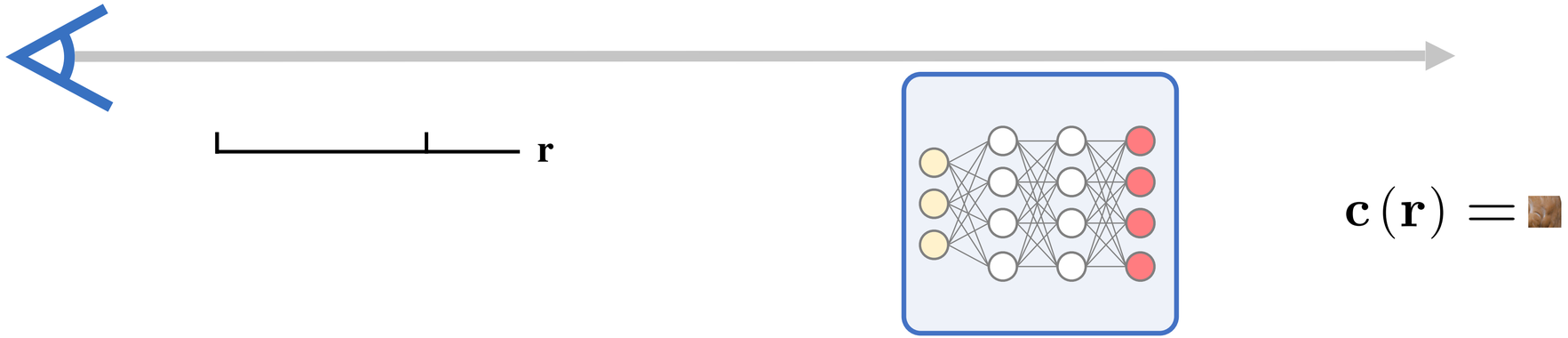
$\mathbf{c}(\mathbf{r}) =$

Light Field Networks



$\mathbf{r}$

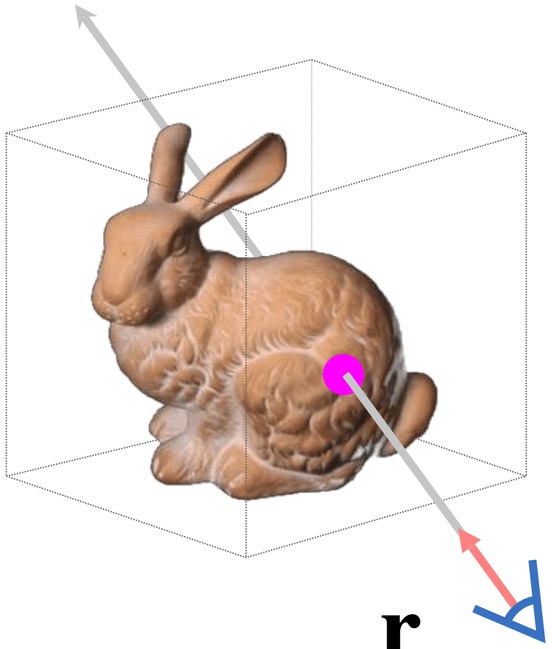
Camera Ray



$\mathbf{c}(\mathbf{r}) =$

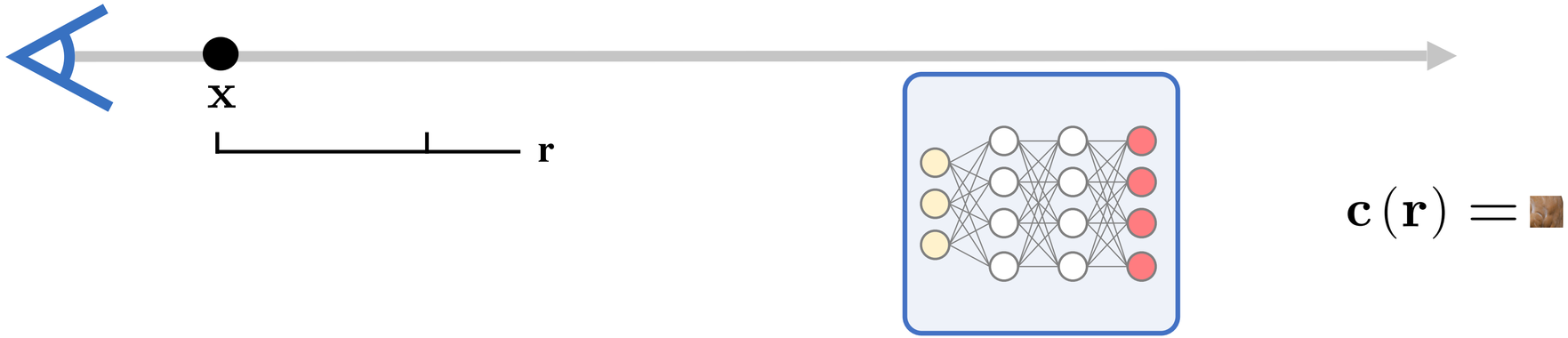
An Alternative Scene Representation

Light Field Networks



$\mathbf{r}$

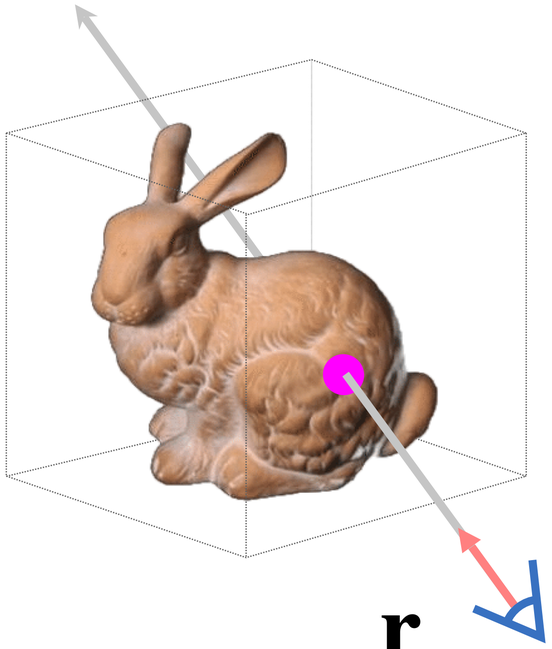
Camera Ray



$\mathbf{c}(\mathbf{r}) =$

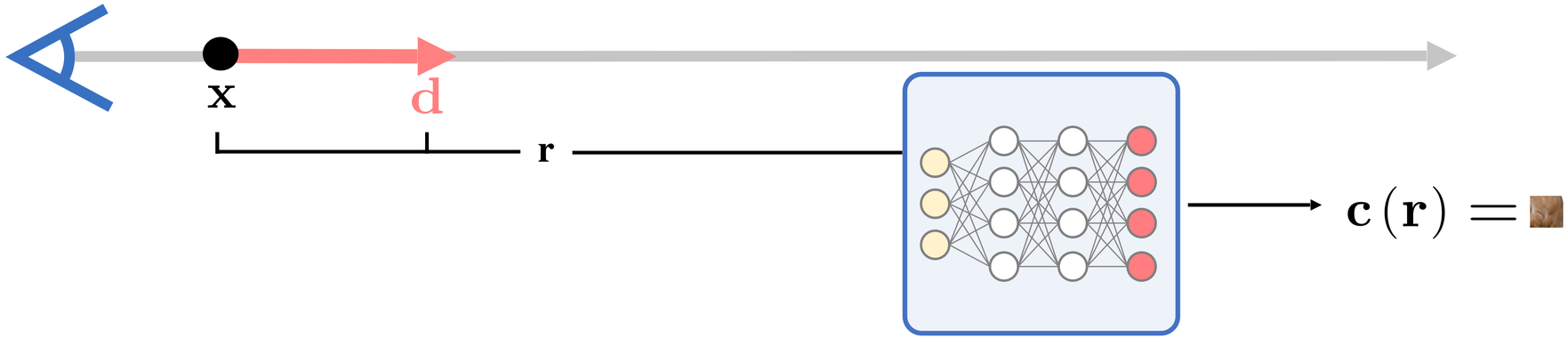
An Alternative Scene Representation

Light Field Networks



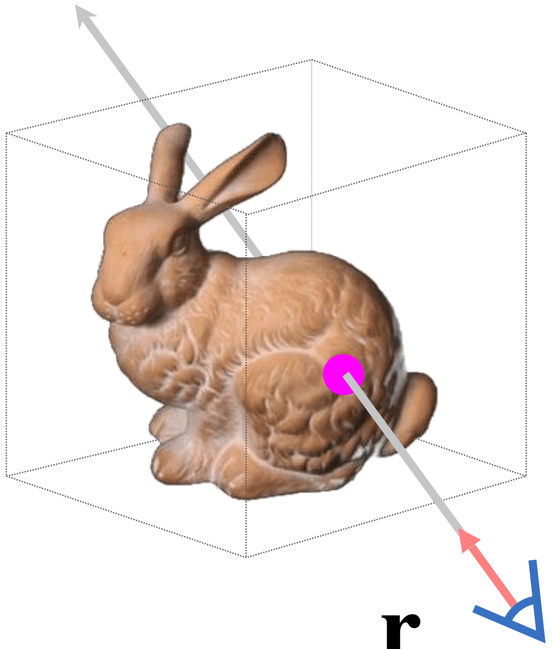
r

Camera Ray



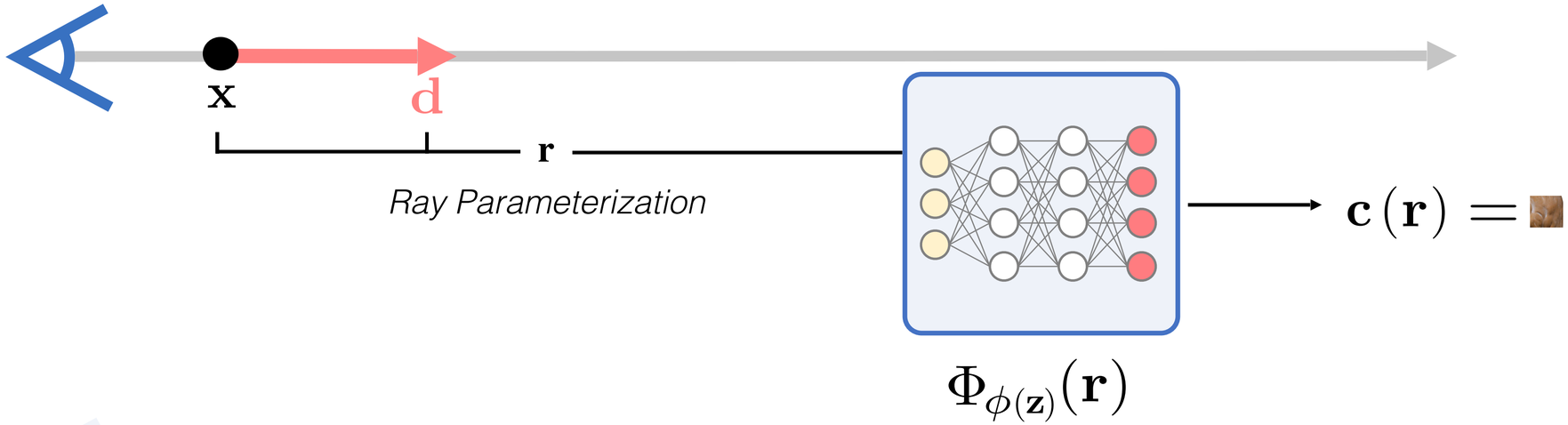
An Alternative Scene Representation

Light Field Networks



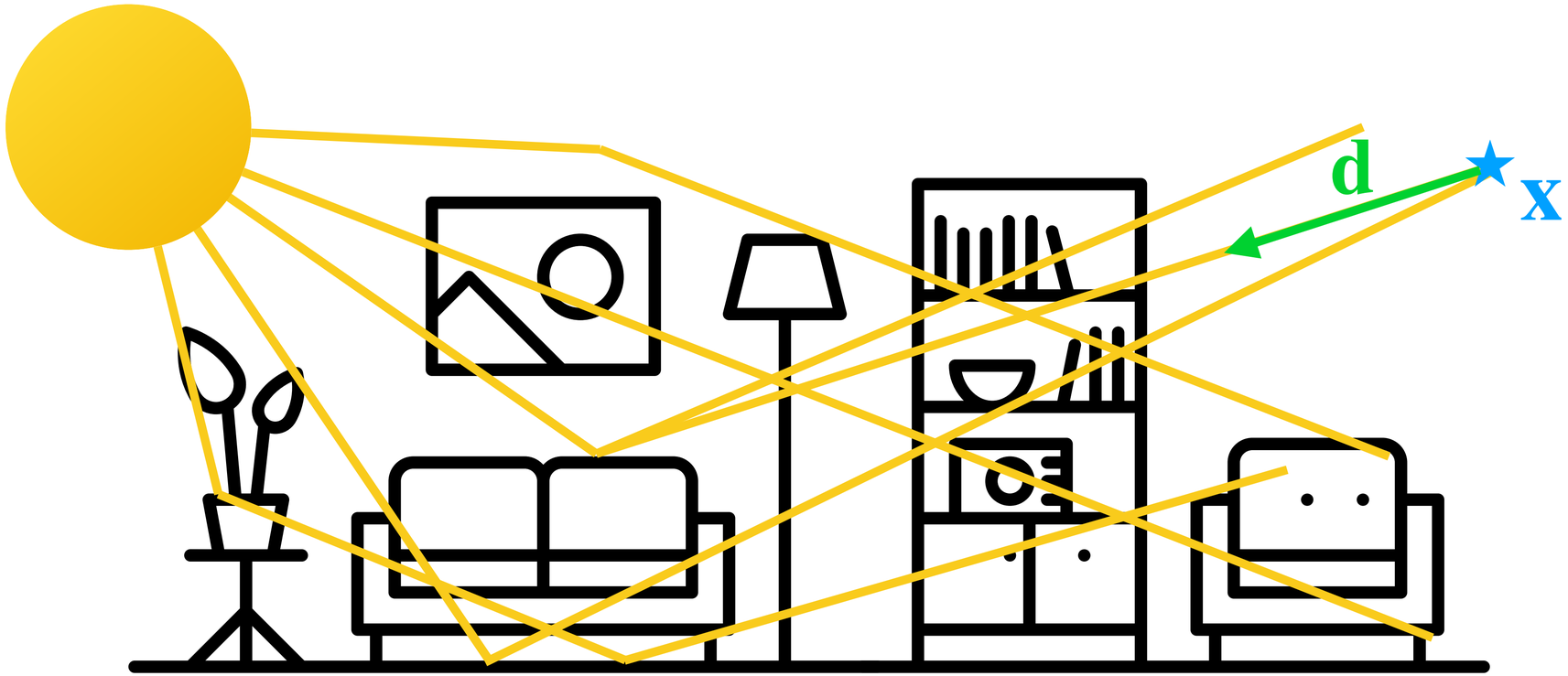
$\mathbf{r}$

Camera Ray



An Alternative Scene Representation

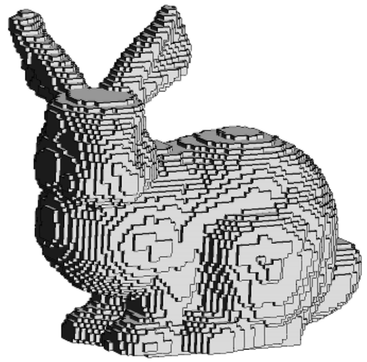
Just because Scene is 3D, don't have to store information in 3D data structure…

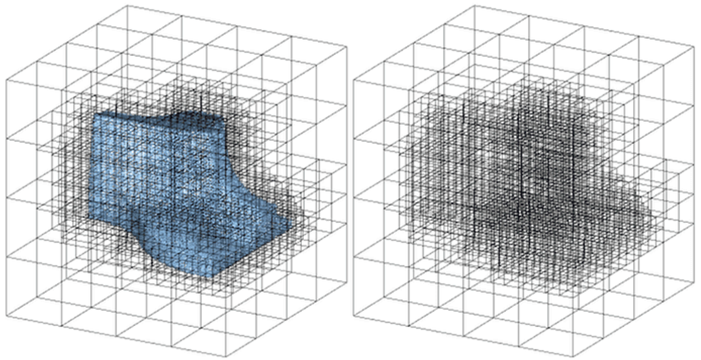


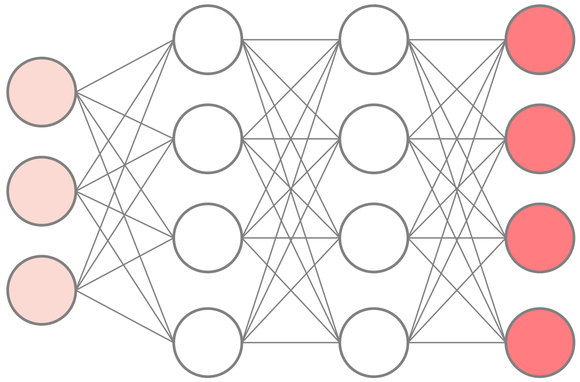
$$LF : \mathbb{R}^3 \times \mathbb{S}^2 \rightarrow \mathbb{R}^3, \quad LF(\mathbf{x}, \mathbf{d}) = \mathbf{c}$$

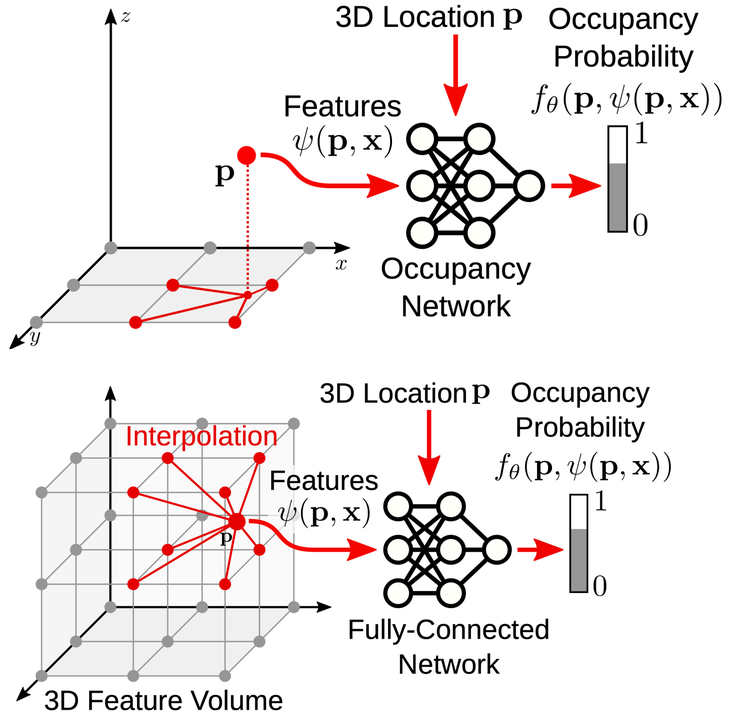
Light Field is 5-dimensional! What parameterizations are possible…?

Summary









- No “Artificial Intelligence”, “understanding” or “Machine Learning” when fitting a single representation to a single 3D thing, even if there are neural networks in them.
- It becomes *a lot* more interesting when considering *inference,* i.e., we want representation to be output of an encoder! More on that later in “Prior-based reconstruction”.