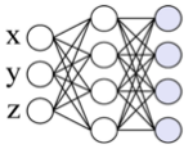
Digital Geometry Processing
Neural Fields

Andrea Tagliasacchi
Simon Fraser University



Neural Fields in Computer Vision

Full-Day Tutorial, June 20th, 2022

CVPR JUNE 19-24 2022 NEW ORLEANS LOUISIANA

neuralfields.cs.brown.edu/cvpr22


Yiheng Xie


Towaki Takikawa

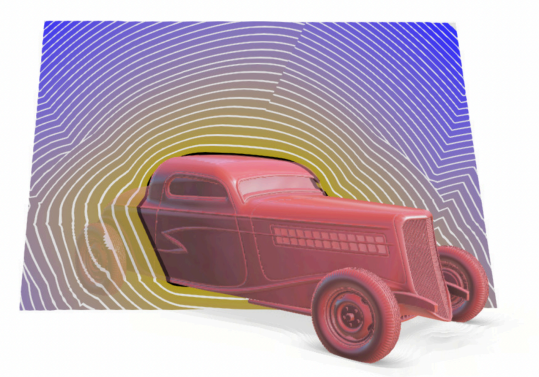

Shunsuke Saito


Or Litany

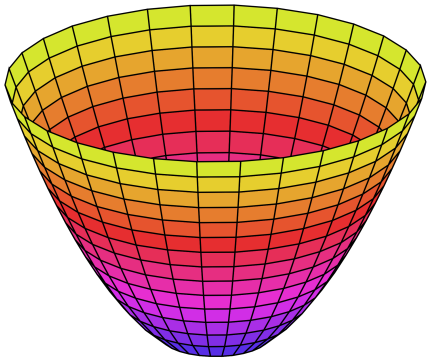

James Tompkin


Vincent Sitzmann


Srinath Sridhar

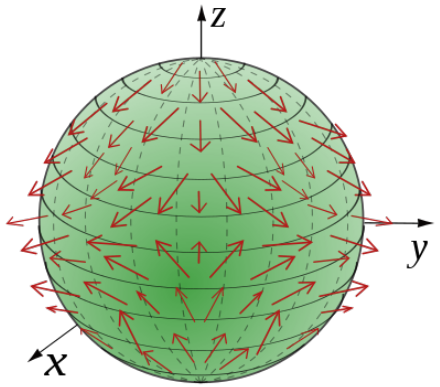


Signed Distance Fields
(map 3D → 1D)

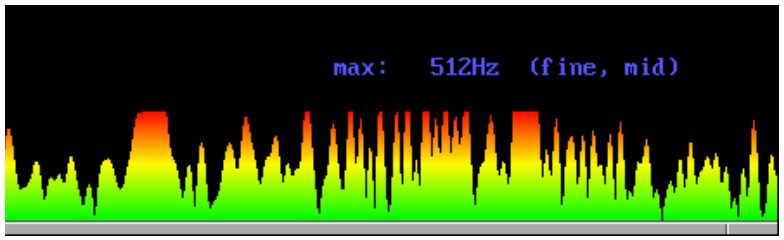


Parabola
(map 2D → 3D)

Image
(map 2D → 3D)



Vector Field
(map 3D → 3D)

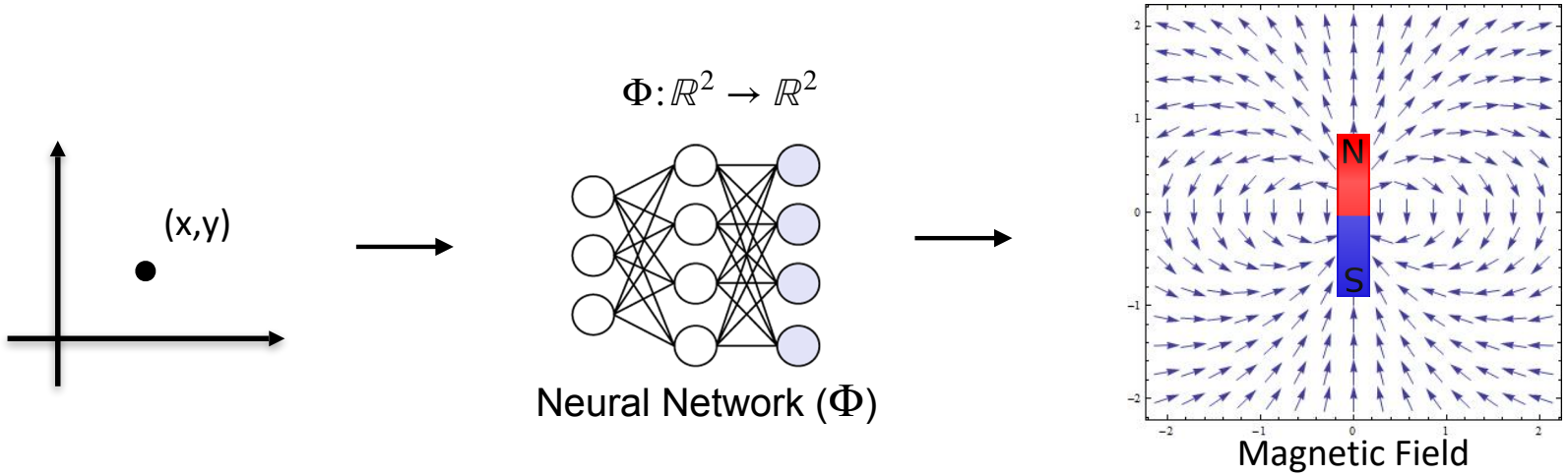


Audio
(map 1D → 1D)

Fields

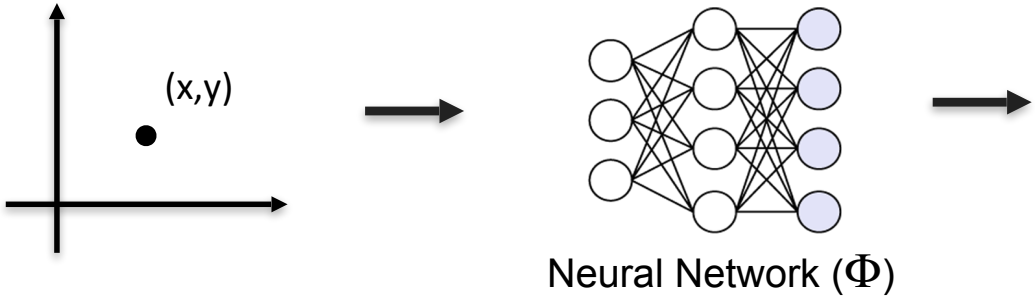
What is a Neural Field

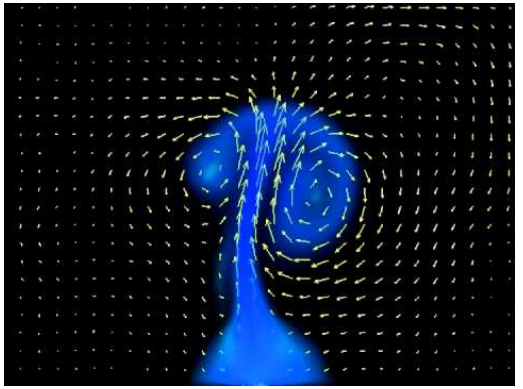
- Mapping between a coordinate and a physical quantity
  - e.g. the magnetic field induced by a magnet



What is a Neural Field

- Mapping between a coordinate and a physical quantity
  - e.g. the advection field within a fluid simulation package



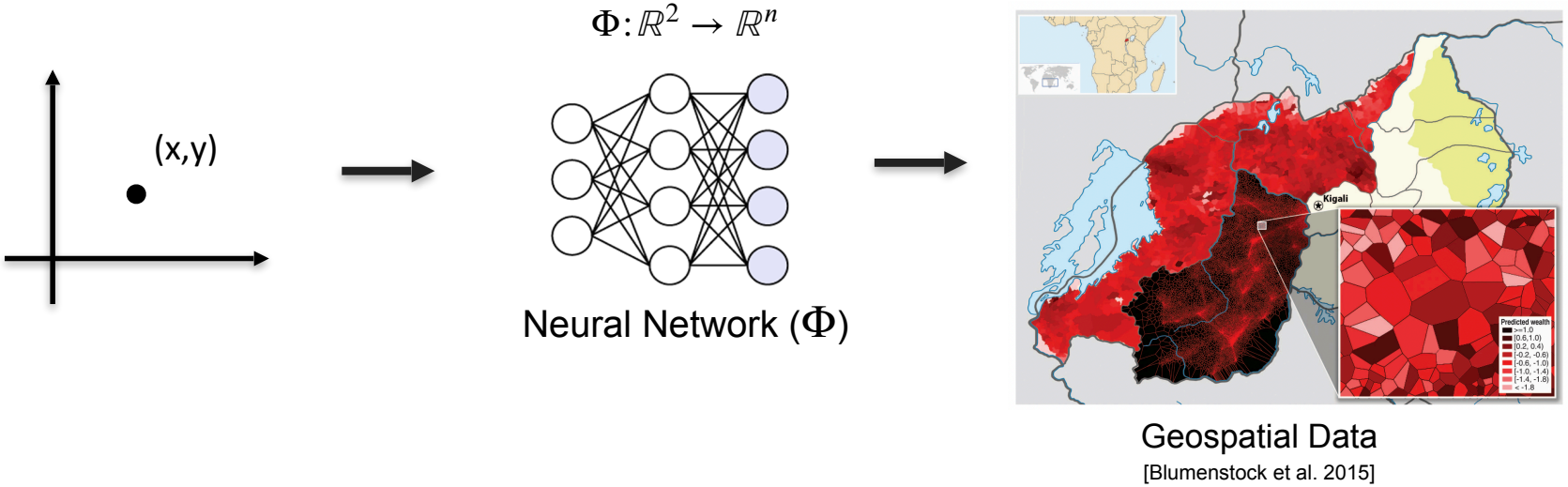


Eulerian Flow Field of a Fluid

[Koldora CC]

What is a Neural Field

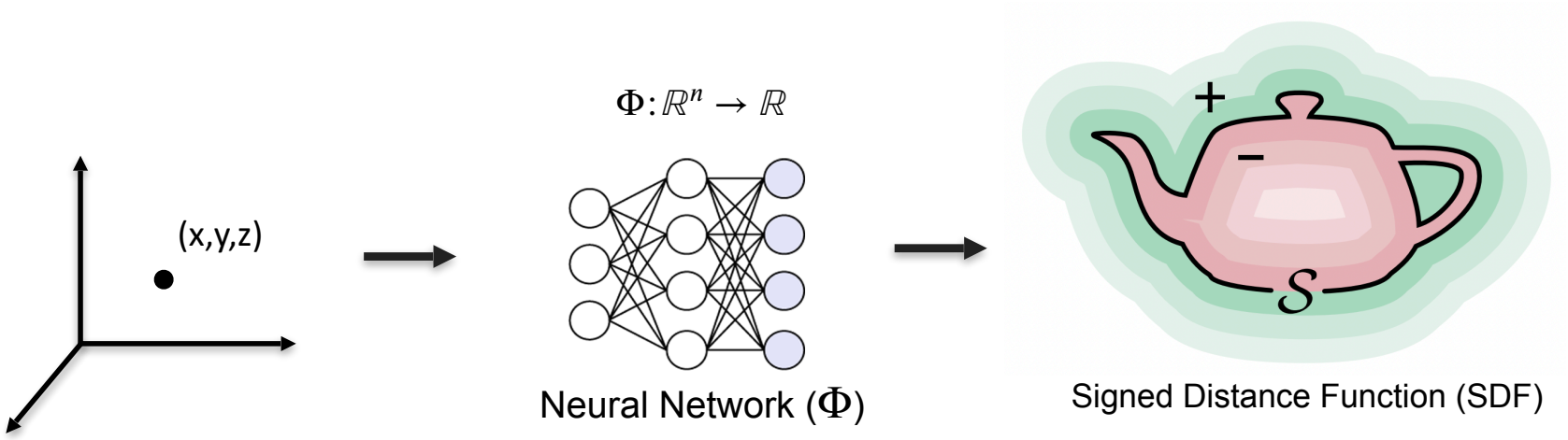
- Mapping between a coordinate and a ~~physical~~ quantity
  - e.g. the average wealth in a country



Geospatial Data
[Blumenstock et al. 2015]

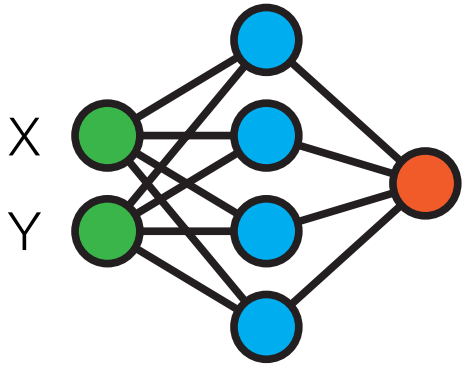
What is a Neural Field

- Mapping between a coordinate and a ~~physical~~ quantity
  - e.g. the signed distance function to the boundary of an object

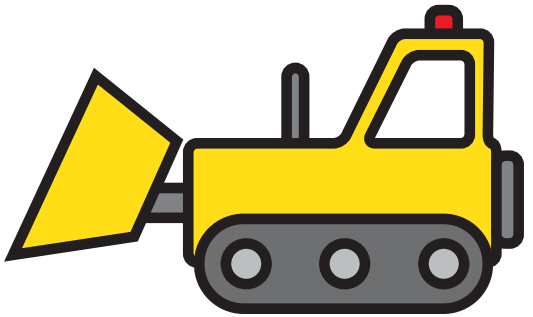


Terminology & Misnomers

Implicit Neural Representations



Coordinate-based Neural Networks

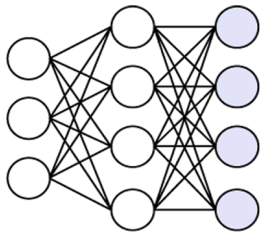


NeRFs

Definitions

- Definition 1:
  A field is a quantity defined for all spatial and / or temporal coordinates.
- Definition 2:
  A neural field is a field that is [partially] parameterized by a neural network.

MLP
Multi Layer Perceptron



coordinate input

nonlinearity

$$z^{(1)} = x$$

$$z^{(i+1)} = \sigma\left(W^{(i)} z^{(i)} + b^{(i)}\right), \quad \underbrace{i = 1, \ldots, k-1}_{\text{induction}}$$

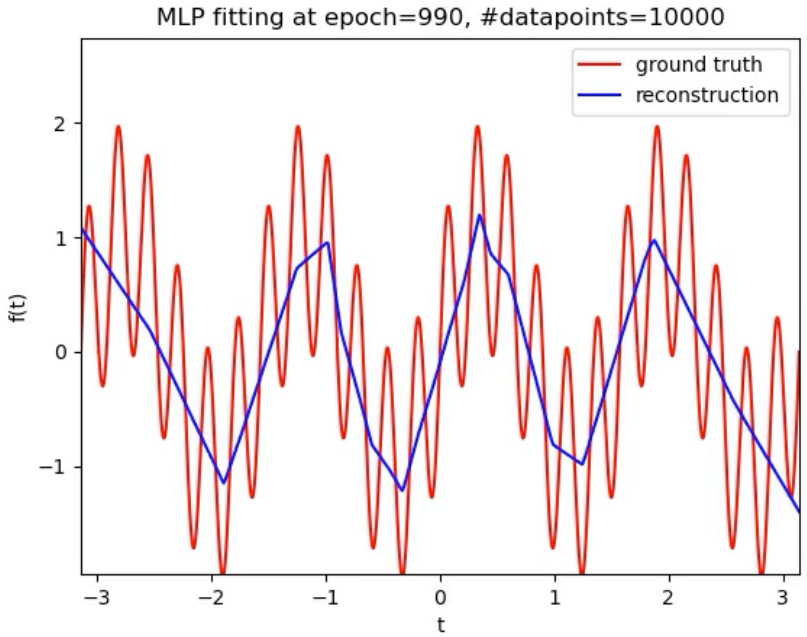
$$f(x) = W^{(k)} z^{(k)} + b^{(k)}$$

field output

Coding Exercise – Task 1

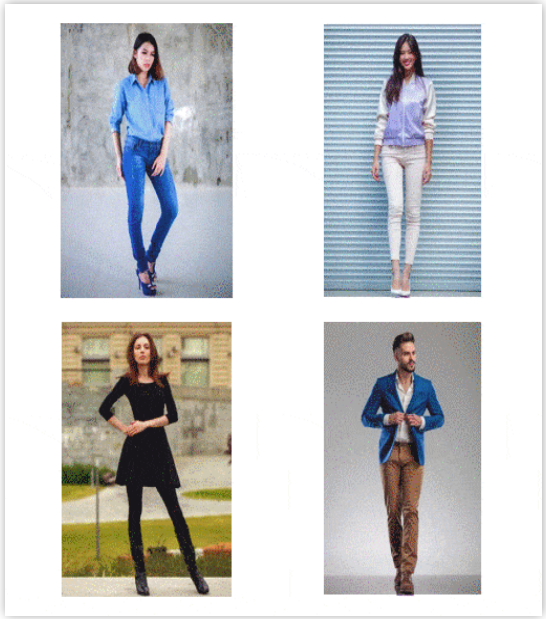
- Fit a signal by optimizing the MLP parameters
  - optimize MLP weights (W) and biases (b)
  - training data: $\{(x_k, f(x_k))\}_{k=1}^{K}$

$$
\begin{aligned}
z^{(1)} &= x \\
z^{(i+1)} &= \sigma\left(W^{(i)} z^{(i)} + b^{(i)}\right), \; i = 1, \ldots, k-1 \\
f(x) &= W^{(k)} z^{(k)} + b^{(k)}
\end{aligned}
$$



Applications: Computer Visoin

- e.g. lift 3D structures out of 2D images

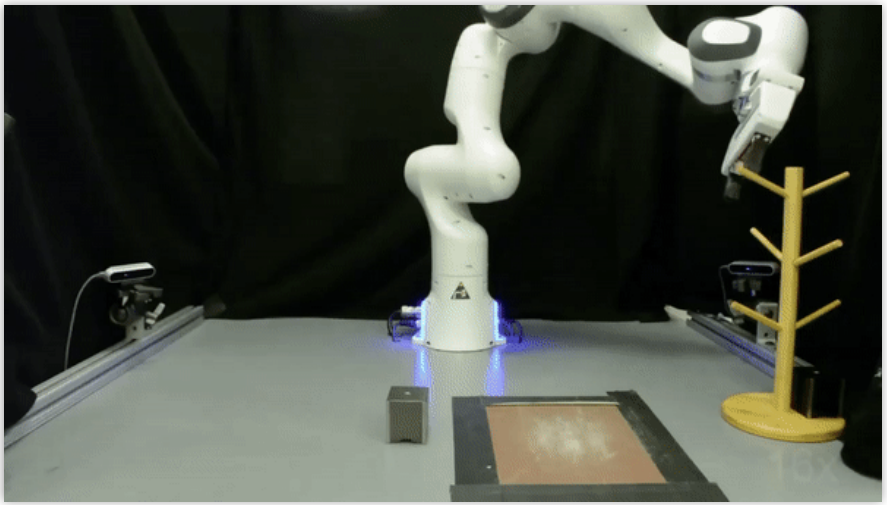


[Saito et al. 2020 PiFU]

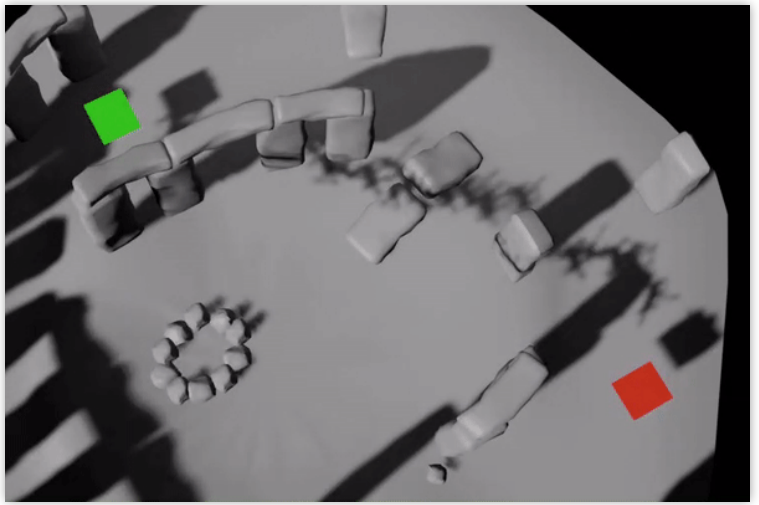
[Chan et al. 2021 EG3D]

Applications: Robotics

- e.g. few-shot learning and motion planning

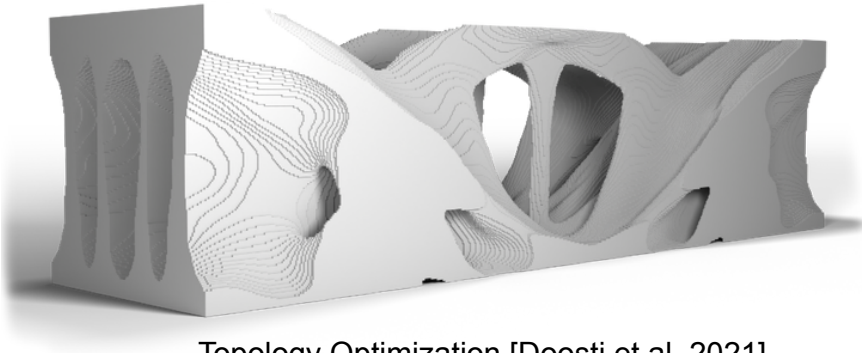


[Simeonov et al. 2021 (Neural Descriptor Fields)]

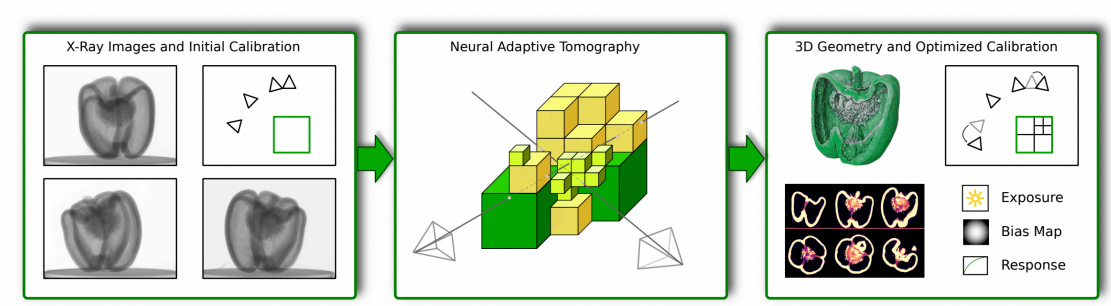


[Adamkiewicz, Chen et al. 2021]

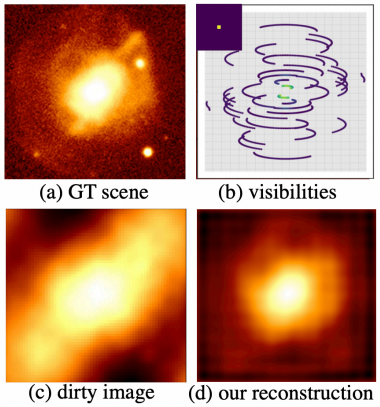
Applications: Physics



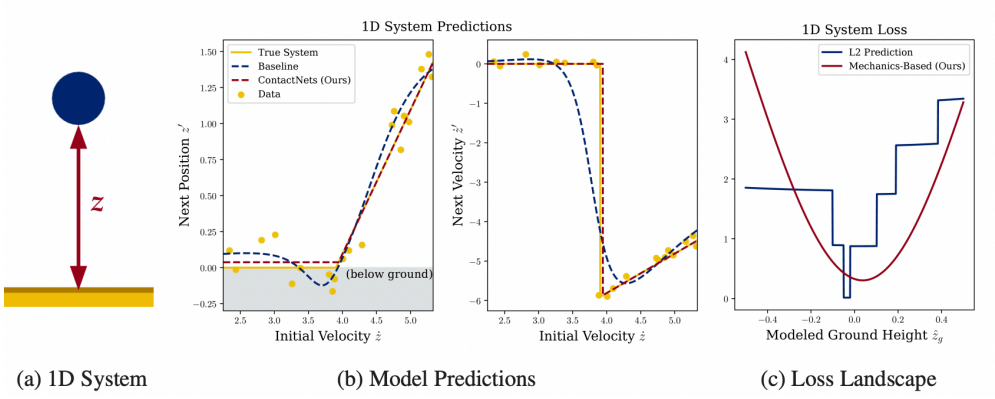
Topology Optimization [Doosti et al. 2021]



Tomographic Reconstruction [Ruckert et al. 2022]

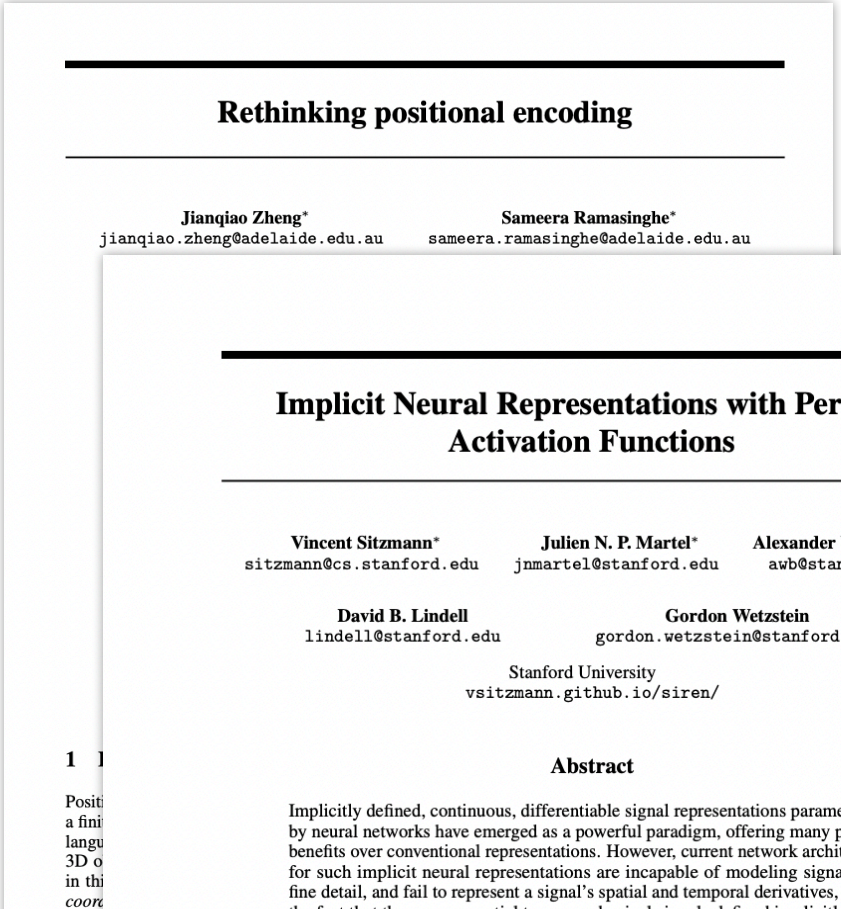


Astronomical Interferometry [Wu et al. 2021]



Contact Dynamics [Pfrommer, Halm et al. 2022]

Applications: ML / Signal Processing



Rethinking positional encoding

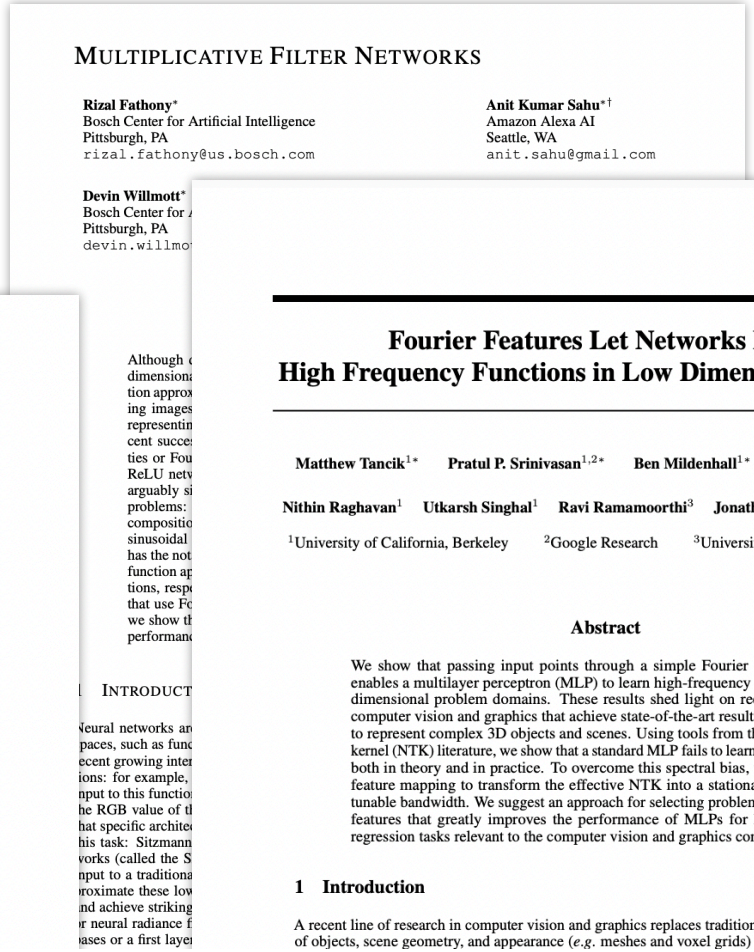
Jianqiao Zheng* jianqiao.zheng@adelaide.edu.au
Sameera Ramasinghe* sameera.ramasinghe@adelaide.edu.au

Implicit Neural Representations with Periodic Activation Functions

Vincent Sitzmann* sitzmann@cs.stanford.edu
Julien N. P. Martel* jnmartel@stanford.edu
Alexander W. Bergman awb@stanford.edu
David B. Lindell lindell@stanford.edu
Gordon Wetzstein gordon.wetzstein@stanford.edu
Stanford University
vsitzmann.github.io/siren/

Abstract

Implicitly defined, continuous, differentiable signal representations parameterized by neural networks have emerged as a powerful paradigm, offering many possible benefits over conventional representations. However, current network architectures for such implicit neural representations are incapable of modeling signals with fine detail, and fail to represent a signal's spatial and temporal derivatives, despite



MULTIPLICATIVE FILTER NETWORKS

Rizal Fathony* Bosch Center for Artificial Intelligence Pittsburgh, PA rizal.fathony@us.bosch.com
Anit Kumar Sahu*† Amazon Alexa AI Seattle, WA anit.sahu@gmail.com
Devin Willmott* Bosch Center for … Pittsburgh, PA devin.willmo…

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

Matthew Tancik[1*] **Pratul P. Srinivasan**[1,2*] **Ben Mildenhall**[1*] **Sara Fridovich-Keil**[1]
Nithin Raghavan[1] **Utkarsh Singhal**[1] **Ravi Ramamoorthi**[3] **Jonathan T. Barron**[2] **Ren Ng**[1]
[1]University of California, Berkeley [2]Google Research [3]University of California, San Diego

Abstract

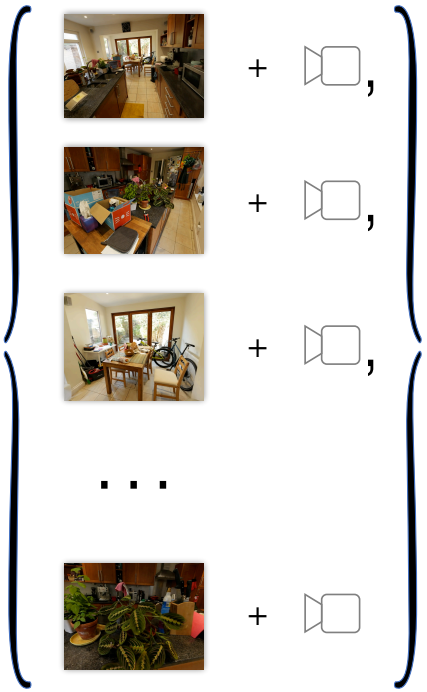
We show that passing input points through a simple Fourier feature mapping enables a multilayer perceptron (MLP) to learn high-frequency functions in low-dimensional problem domains. These results shed light on recent advances in computer vision and graphics that achieve state-of-the-art results by using MLPs to represent complex 3D objects and scenes. Using tools from the neural tangent kernel (NTK) literature, we show that a standard MLP fails to learn high frequencies both in theory and in practice. To overcome this spectral bias, we use a Fourier feature mapping to transform the effective NTK into a stationary kernel with a tunable bandwidth. We suggest an approach for selecting problem-specific Fourier features that greatly improves the performance of MLPs for low-dimensional regression tasks relevant to the computer vision and graphics communities.

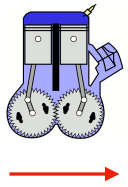
1 Introduction

A recent line of research in computer vision and graphics replaces traditional discrete representations of objects, scene geometry, and appearance (*e.g.* meshes and voxel grids) with continuous functions

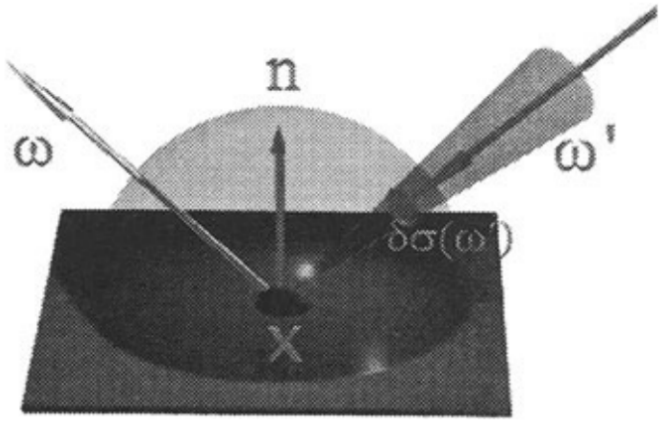
Applications: Computer Graphics

- Render a scene from a novel viewpoint (i.e. Novel View Synthesis)



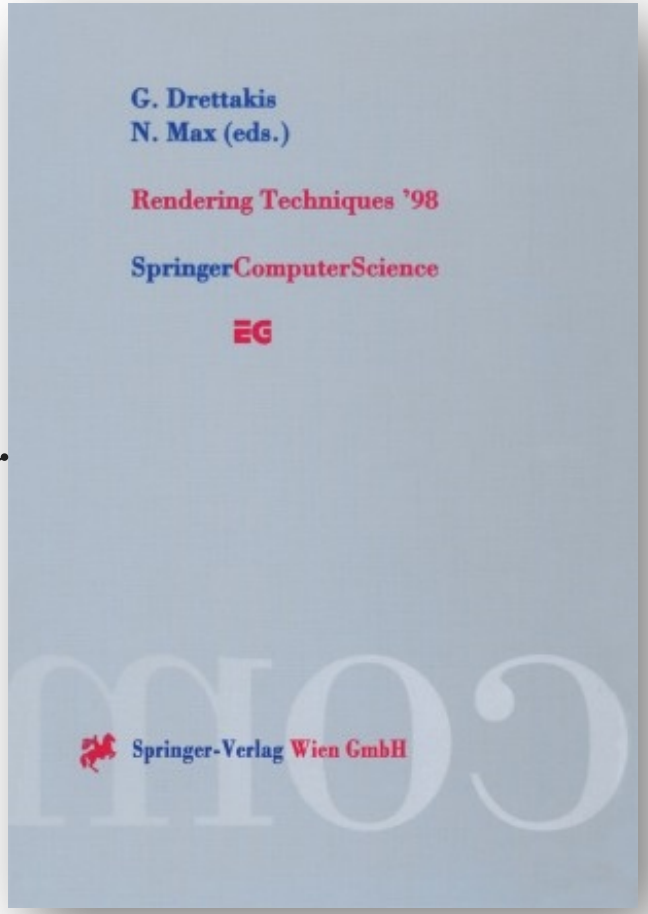


Cambrian Explosion

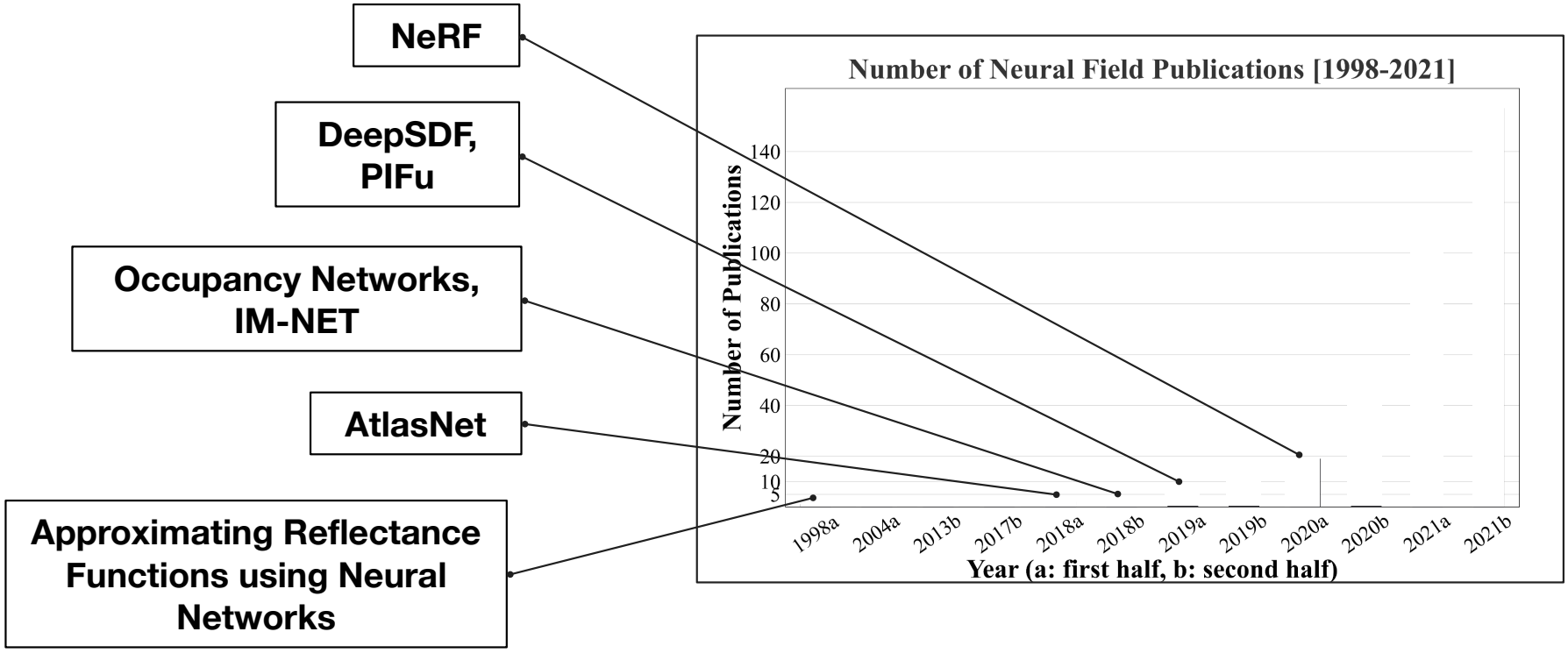


[Gargan and Neelamkavil 1998]

Approximating Reflectance Functions using Neural Networks



Cambrian Explosion

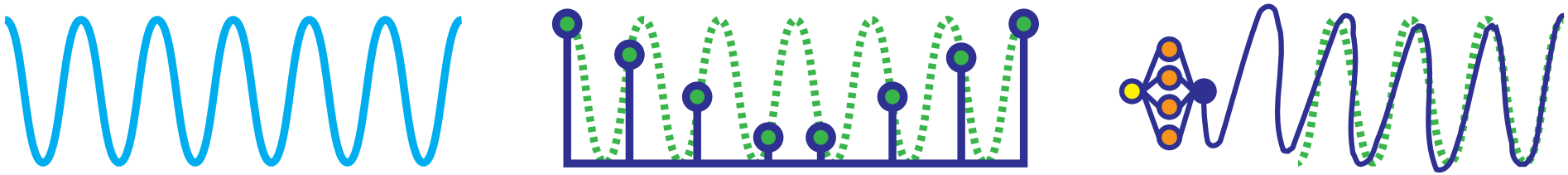


Outline

- Signals
- Forward map
- Architectures

Signals – Representations

Many different types of signals:



Continuous

Discrete

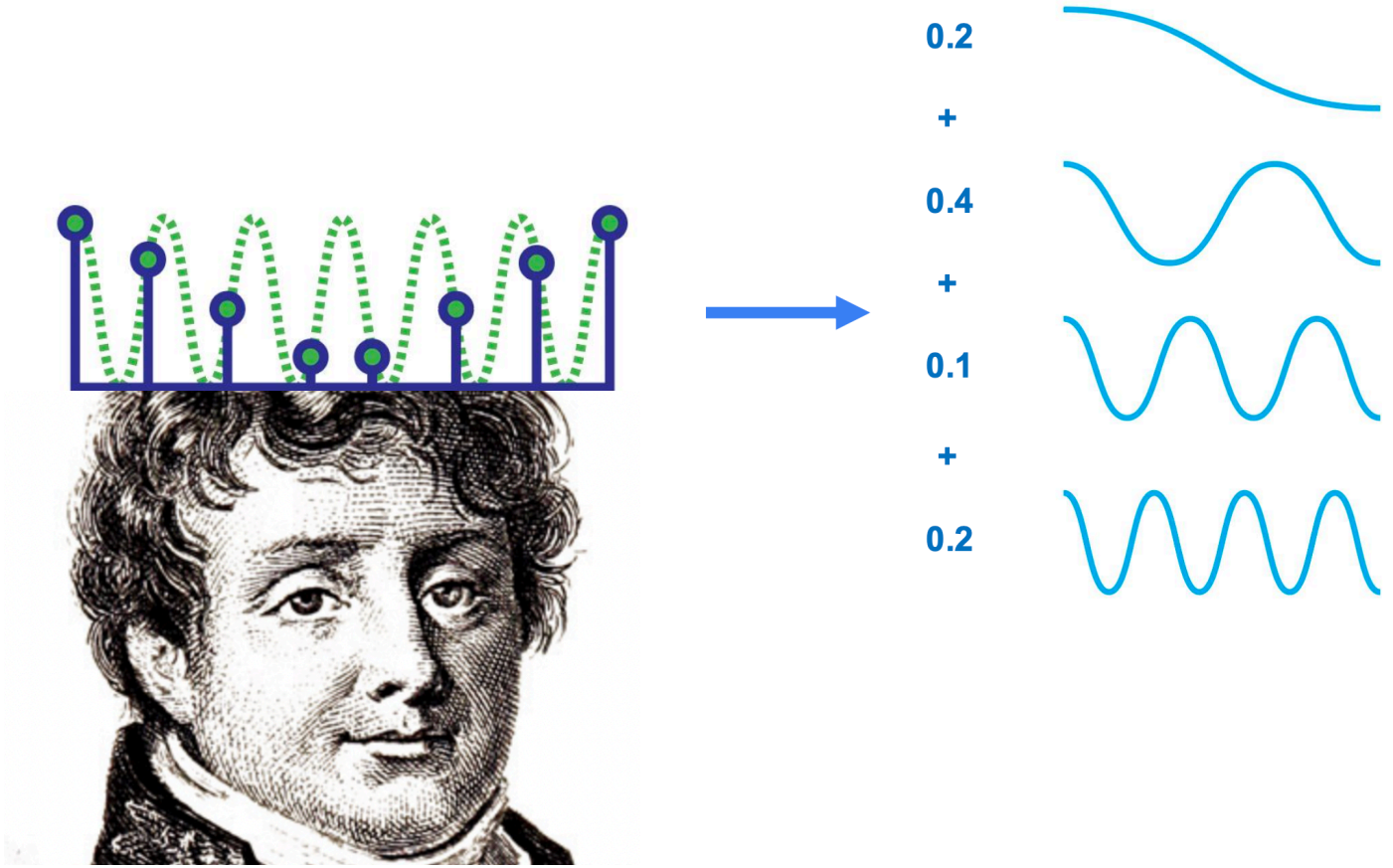
Neural

Signals – Discrete

$$f_x \in \mathbb{R}^m$$

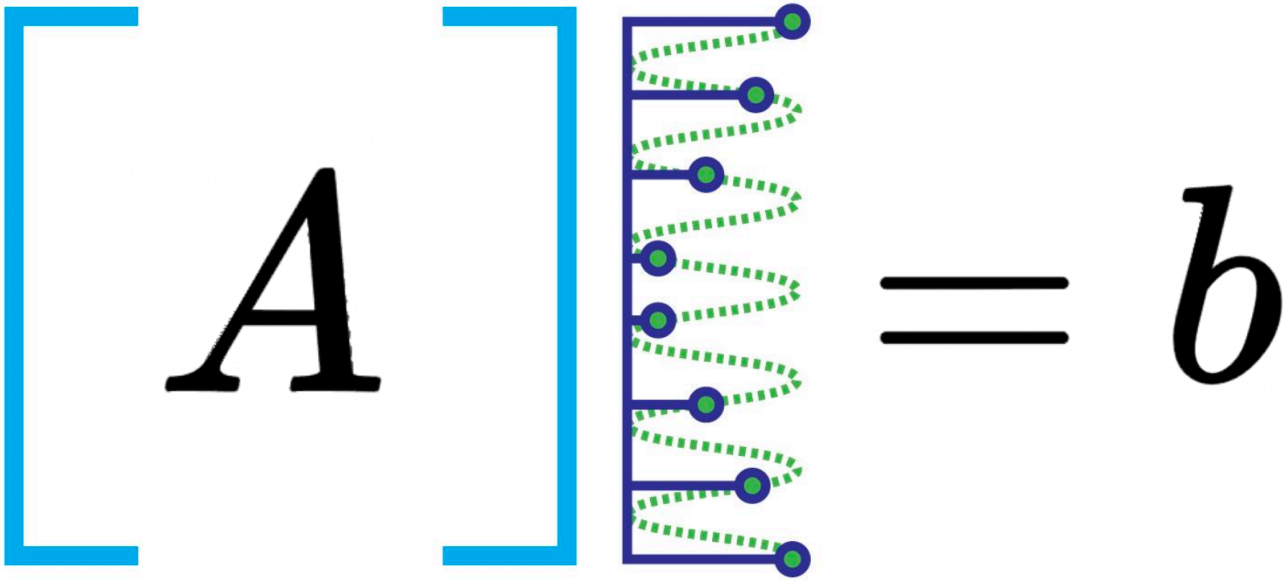
Signals – Fourier Transform

- Ties together discrete / continuous representations



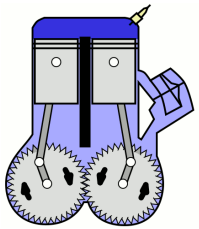
Signals – Linear Algebra

- Convenient manipulation frameworks based on (sparse) linear algebra



Signals – Properties

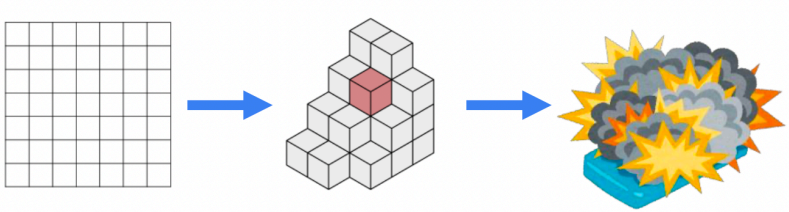
1. Compactness



2. Regularization

$$\text{argmin}_x \|y - F(x)\| + \lambda P(x).$$

3. Domain Agnostic



Signals – Compactness

- How much does it cost to represent a signal?
  - compression leverages coherence of data in space (“intra-frame” codecs)

[Chen et al.]

$$1024 \times 1024 \times 3 = 3 \text{ megabytes}$$

Signals – Compactness

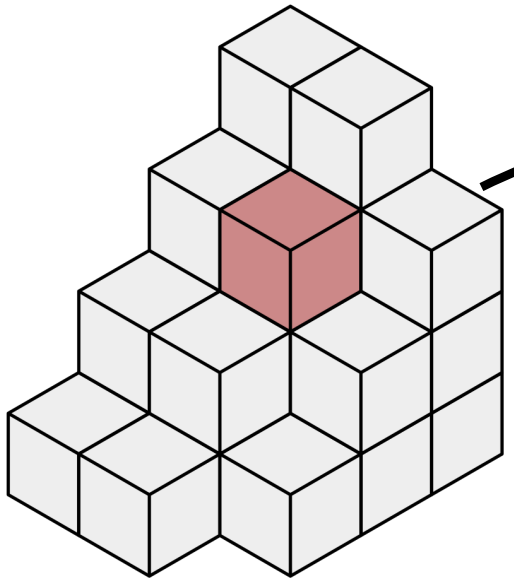
- How much does it cost to represent a signal? (2h movie @ 30fps)
  - compression also leverages coherence of frames in time (“inter-frame” codecs)

[Chen et al.]

$$1024 \times 1024 \times 3 \times 7200 \text{ seconds} \times 30 \frac{\text{frames}}{\text{seconds}} = 0.68 \text{ terabytes}$$

Signals – Compactness

- Signals in 3D are generally sparse…
  - usually 2D manifolds embedded in 3D space
  - a dense 3D grid? curse of dimensionality

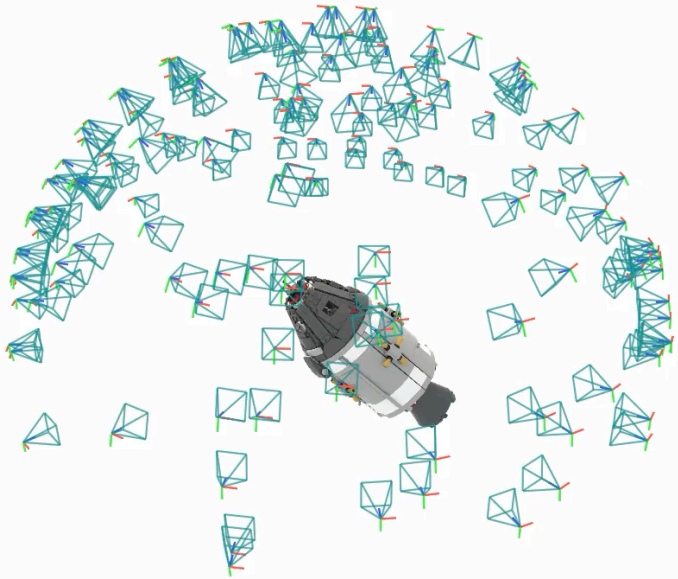


Signed distance, color, density, etc…

Very expensive in storage!

Signals – Compactness

- Application: reconstructing 3D from multiview images



Signal of Interest: 3D Geometry/Appearance

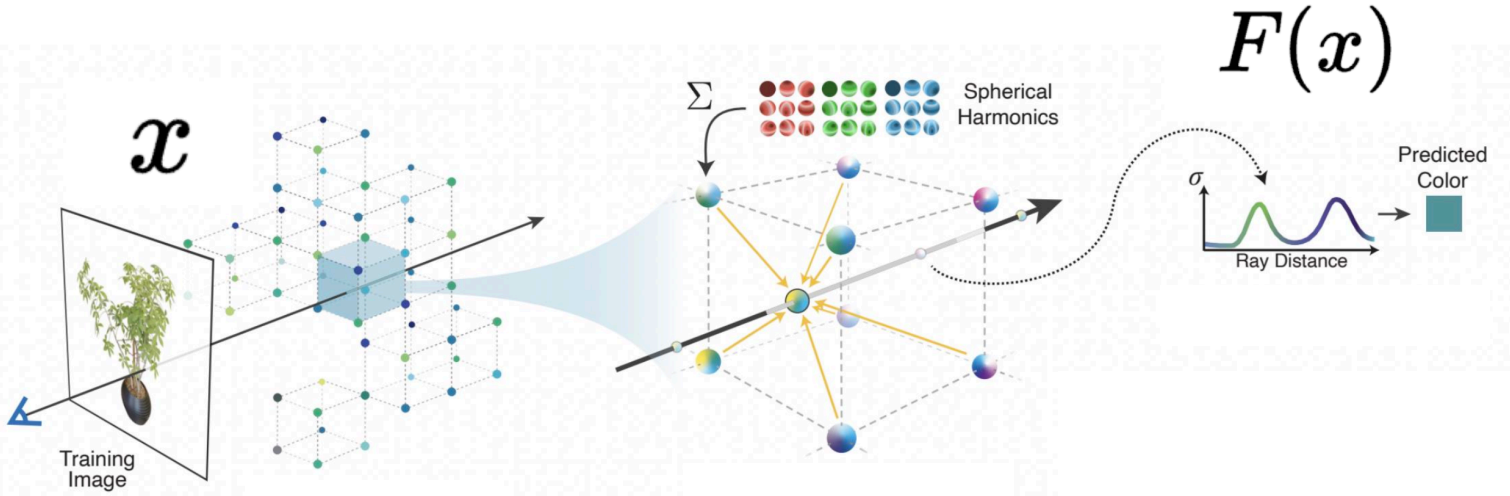
Signal Available: Multiview Images

Inverse Problem: $\text{argmin}_x \|y - F(x)\|$

Meshry et al. RTMV

Signals – Compactness

- Application: reconstructing 3D from multiview images
  - fast to train, but still expensive in memory!



Sparse Grid

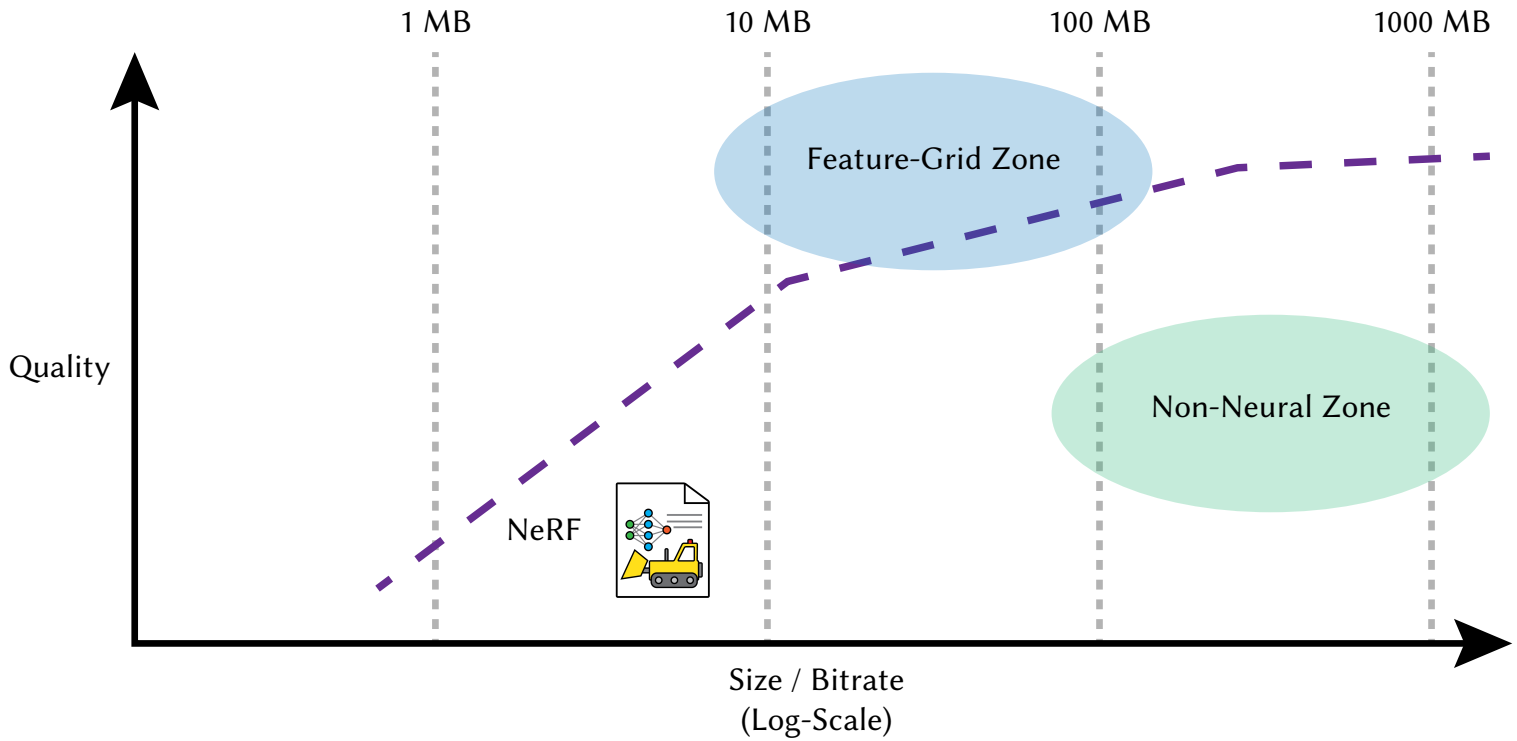
Interpolate (Appearance + Density)

Forward Map (Differentiable Render)

Plenoxels (Yu et al.) CVPR'22, DirectVoxGo (Sun et al.) CVPR'22

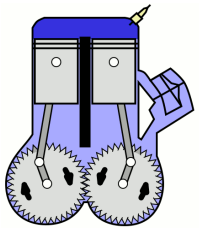
Signals – Compactness

- Application: reconstructing 3D from multiview images
  - objective: get a better rate-distortion curve (closer to top-left of diagram)



Signals – Properties

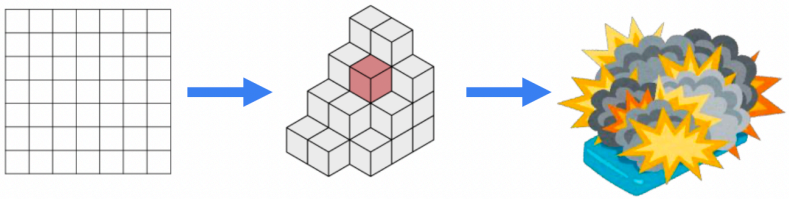
1. Compactness



2. Regularization

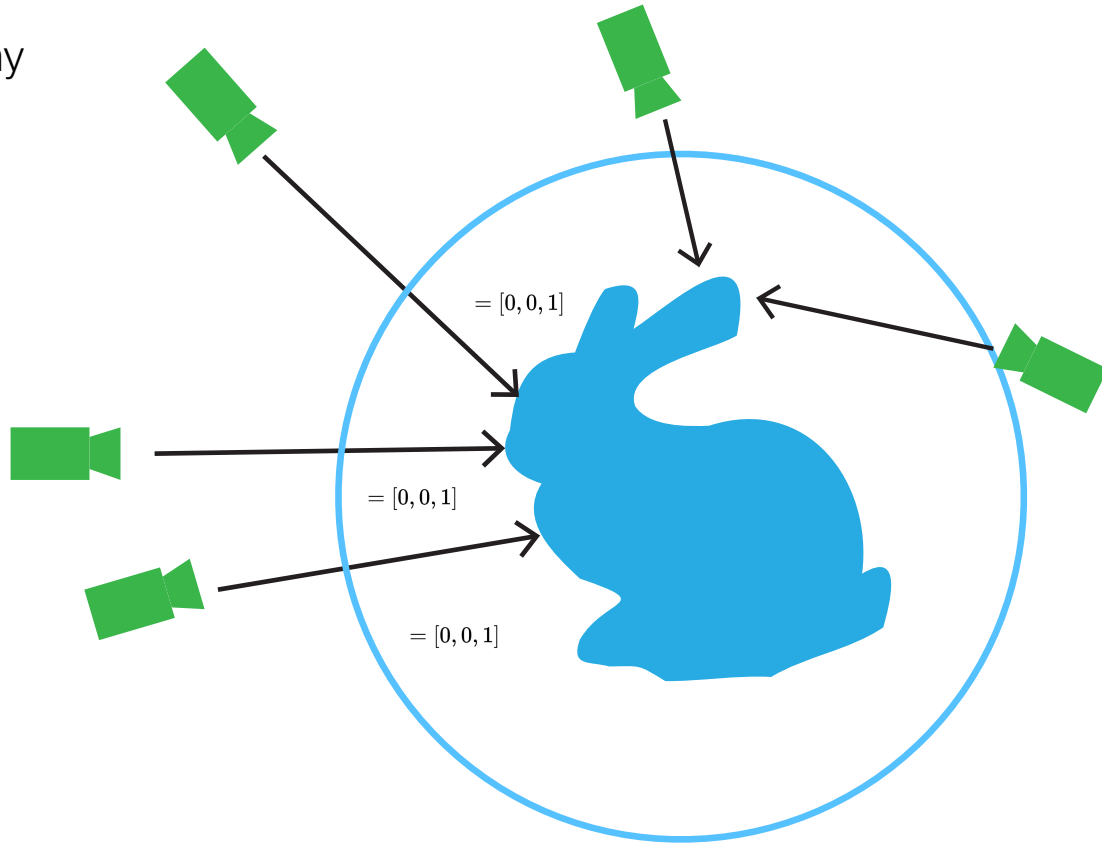
$$\text{argmin}_x \|y - F(x)\| + \lambda P(x).$$

3. Domain Agnostic



Signals – Regularization

- Many problems are under-constrained
  - simple 2D example: a solid blue 2D bunny
- How fix?
  - add more observations? maybe…
  - regularize the problem? yes!



Signals – Regularization

- How to deal with underconstrained problems?
  - Bayesian optimization: Maximum a Posteriori (MAP)

controls balance
data vs. prior

$$\operatorname{argmin}_x \|y - F(x)\| + \lambda P(x).$$

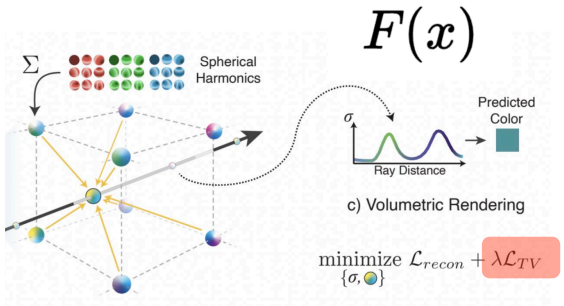
find model
parameters

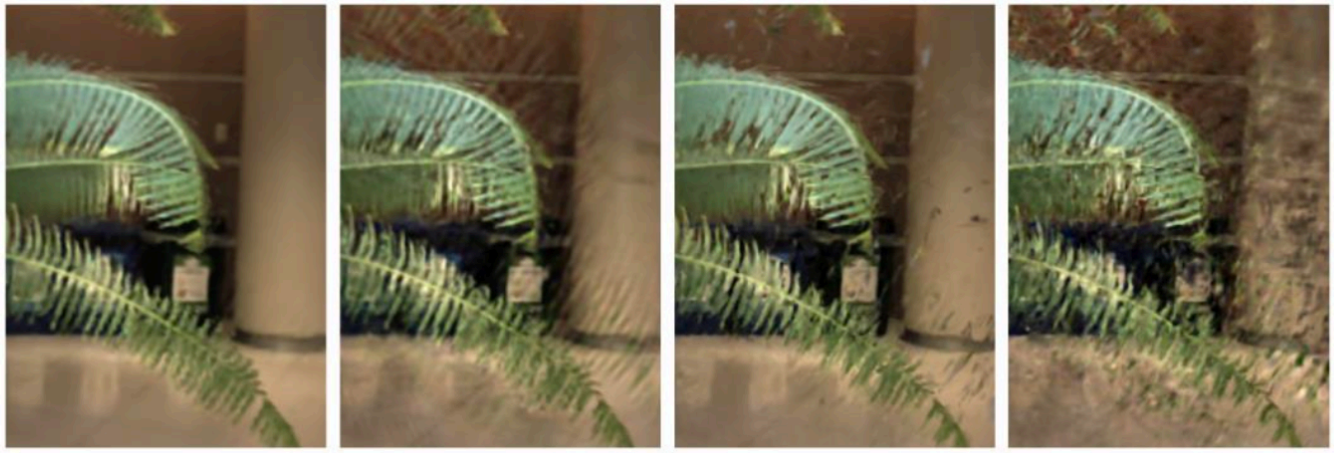
that reproduce
my data

so that the model
parameters are plausible

Signals – Regularization

- How to deal with underconstrained problems?
  - Bayesian optimization: Maximum a Posteriori (MAP)
  - Example: Total Variation (TV) in Plenoxels



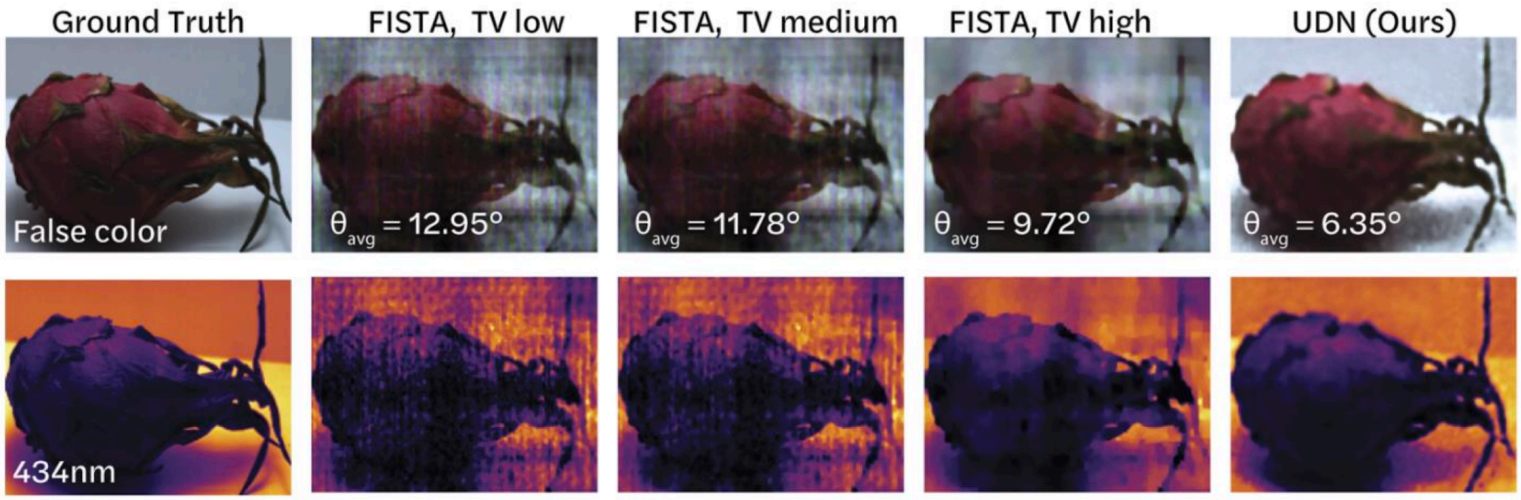


Plenoxels (Yu et al.) CVPR'22

Signals – Regularization

- How to deal with underconstrained problems?
  - Bayesian optimization: Maximum a Posteriori (MAP)
  - Example: Total Variation (TV) in Compressive Sensing

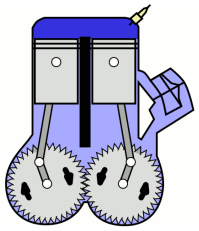
(neural field) does not require regularization!



Untrained Networks for Compressive Lensless Photography (Monakhova et al.)

Signals – Properties

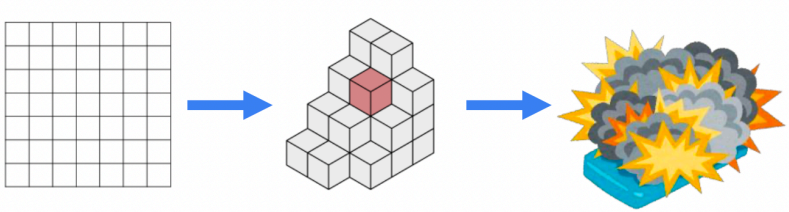
1. Compactness



2. Regularization

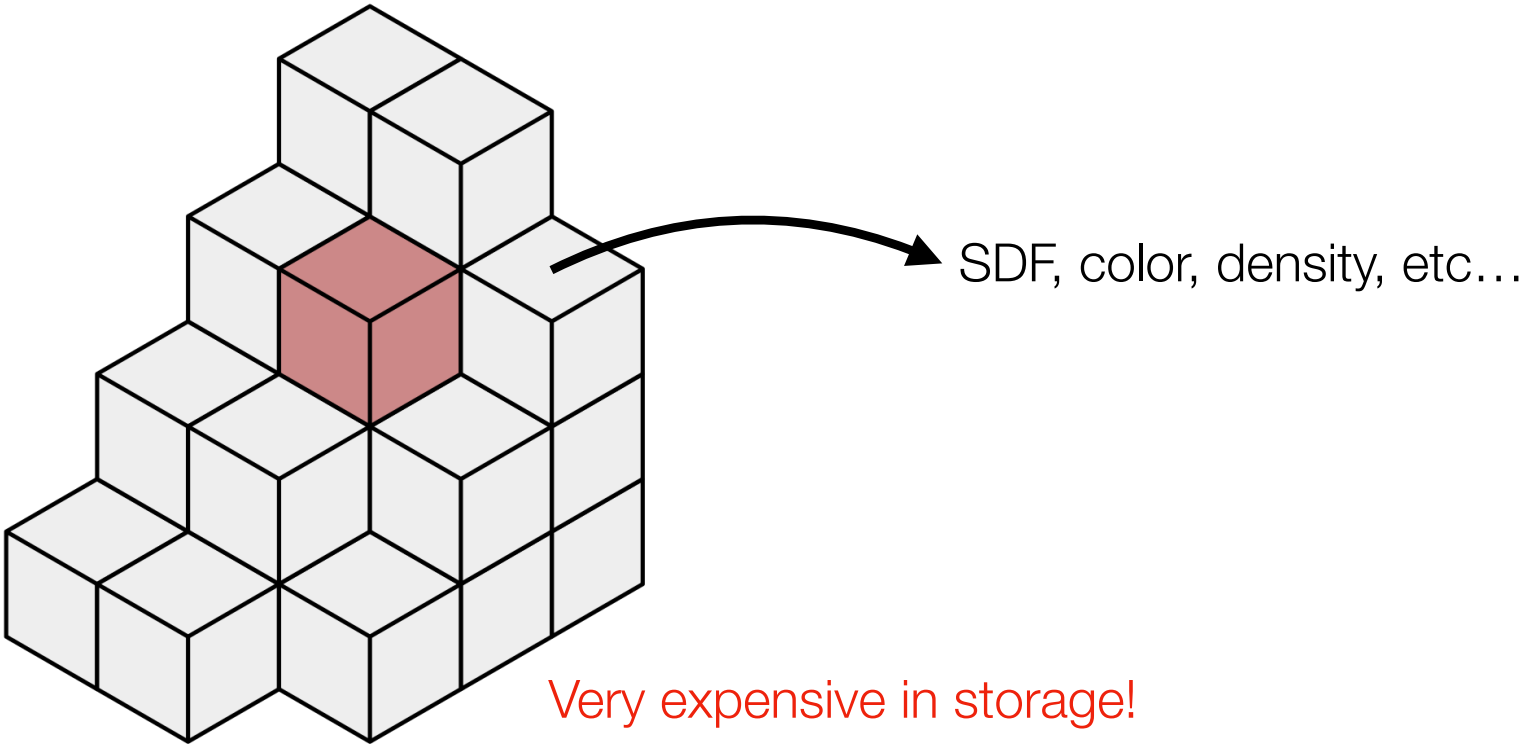
$$\text{argmin}_x \|y - F(x)\| + \lambda P(x).$$

3. Domain Agnostic



Signals – Agnosticism

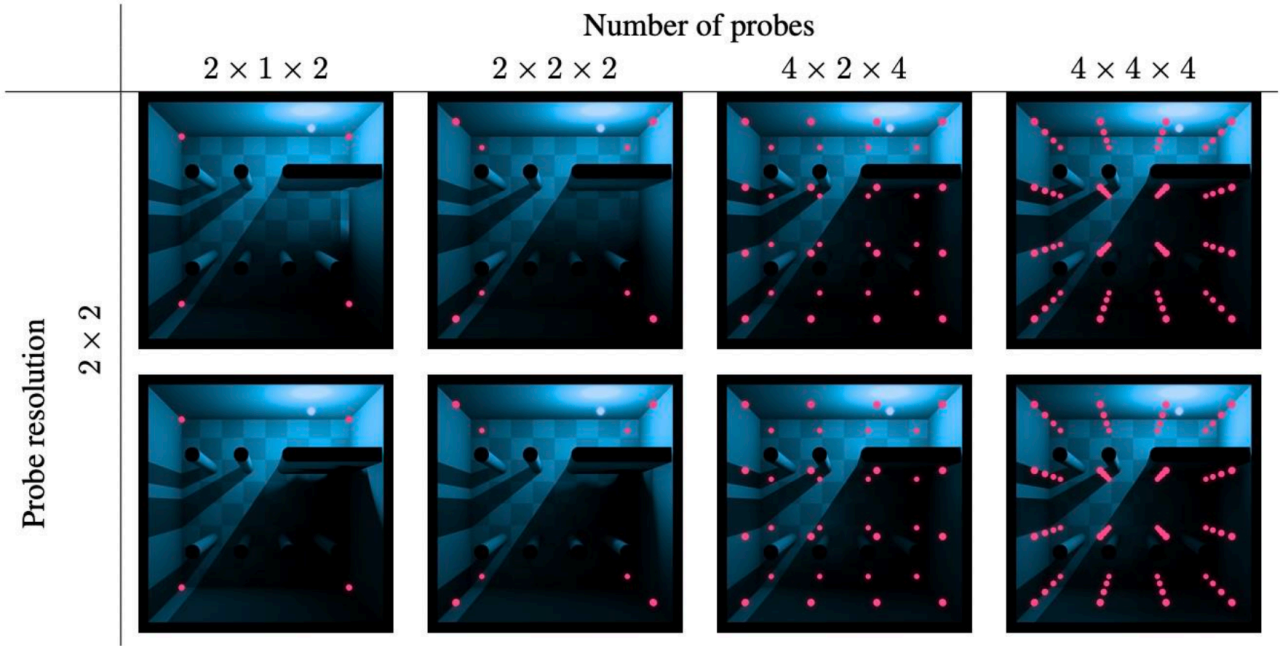
- Represent 3D signals for computer graphics



Very expensive in storage!

Signals – Agnosticism

- Good practice: don’t over-tailor representation to the problem at hand
  - e.g. 3D grids of 2D basis functions

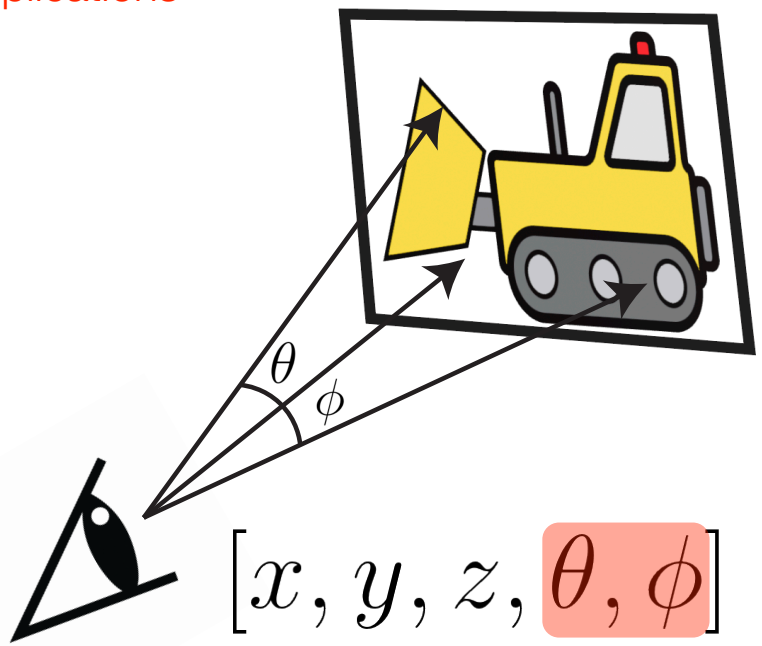


Majercik et al. “Dynamic diffuse global illumination with ray-traced irradiance fields” JCGT’19

Signals – Agnosticism

- Good practice: don't over-tailor representation to the problem at hand
  - e.g. how to capture view-dependent effects? (e.g. NeRF)
  - …neural networks are glorified vector-matrix multiplications

$$h = \begin{bmatrix} a & b & \mathbf{l} \\ c & d & \mathbf{m} \\ e & f & \mathbf{n} \\ j & k & \mathbf{o} \end{bmatrix} \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$



Coding Exercise – Task 3

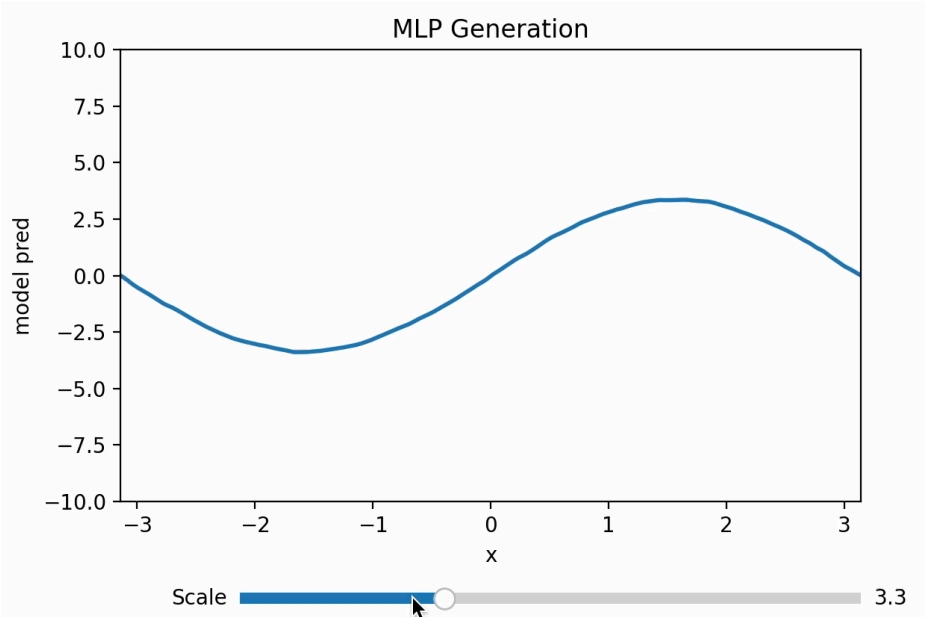
- Fit a signal by optimizing the MLP parameters
  - same network/objective as before (but fit to multiple signals vs. just one)
  - add a conditioning signal by concatenation

$$z^{(1)} = \gamma(x), s$$

$$z^{(1)} = x$$

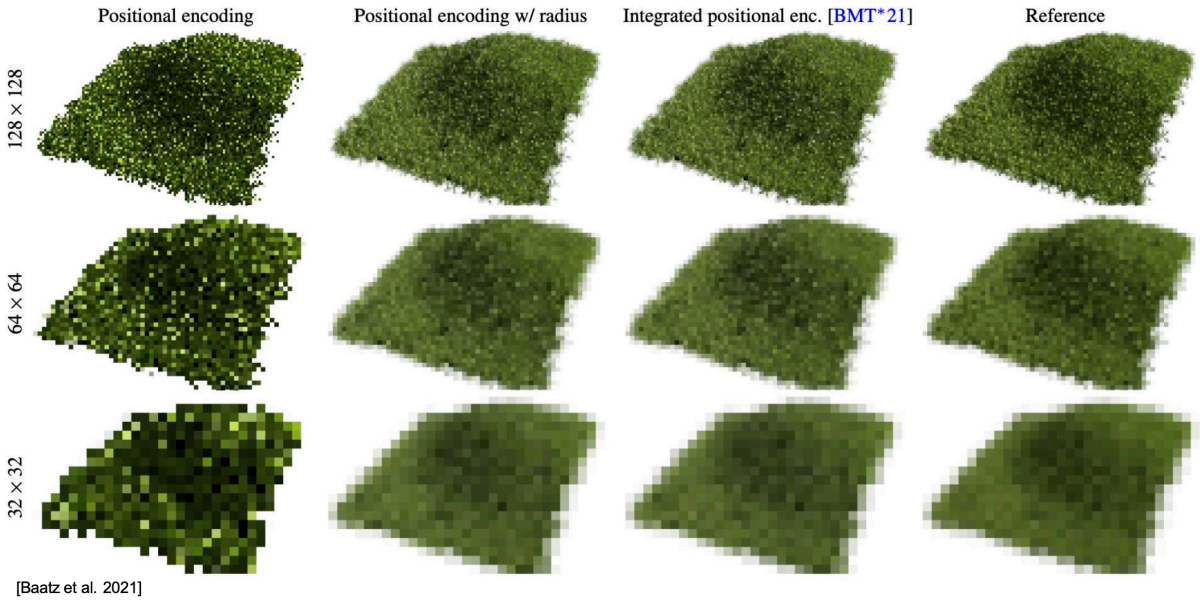
$$z^{(i+1)} = \sigma\left(W^{(i)} z^{(i)} + b^{(i)}\right), \; i = 1, \ldots, k-1$$

$$f(x) = W^{(k)} z^{(k)} + b^{(k)}$$



Signals – Agnosticism

- Good practice: don’t over-tailor representation to the problem at hand
  - e.g. same representation can be adapted to texture synthesis



Baatz et al. “NeRF-Tex: Neural Reflectance Field Textures” EGSR’21

Signals – Neural Fields

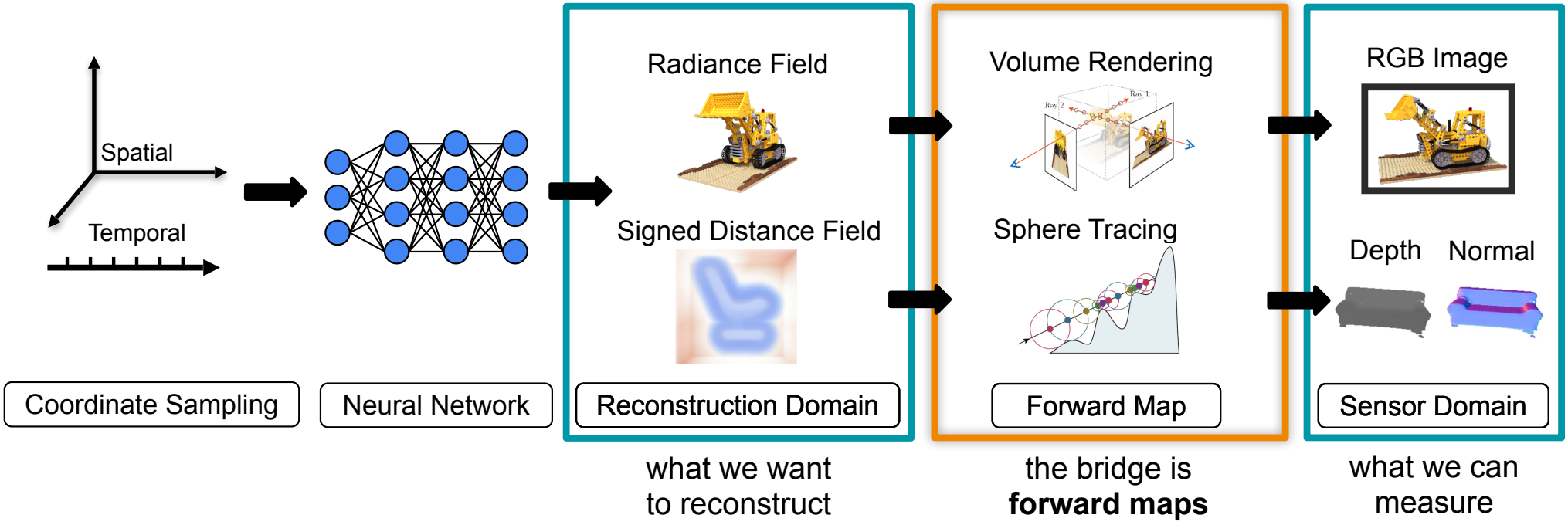
- Problems
  - Editability / manipulability
  - Computational complexity
  - Spectral bias

Outline

- Signals
- Forward map
- Architectures

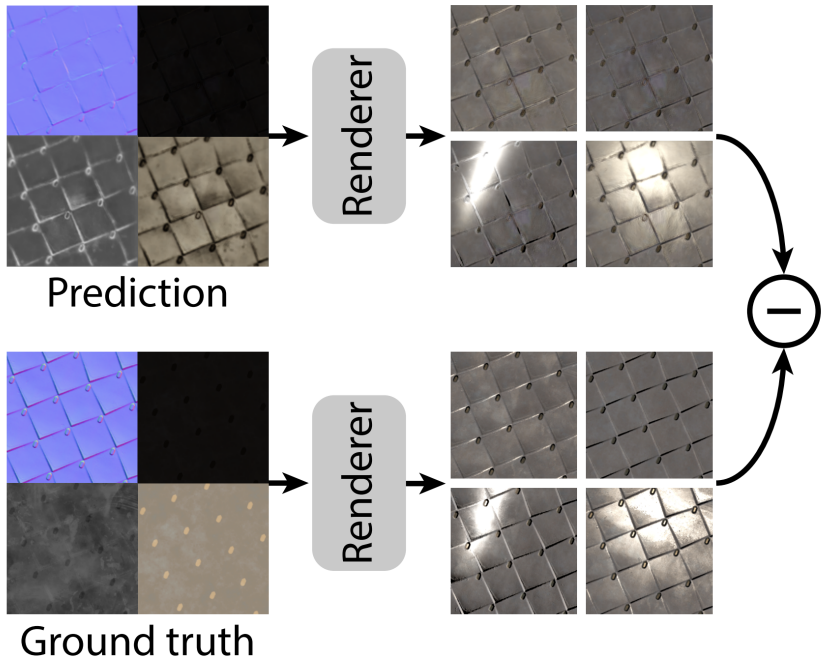
Foward Maps

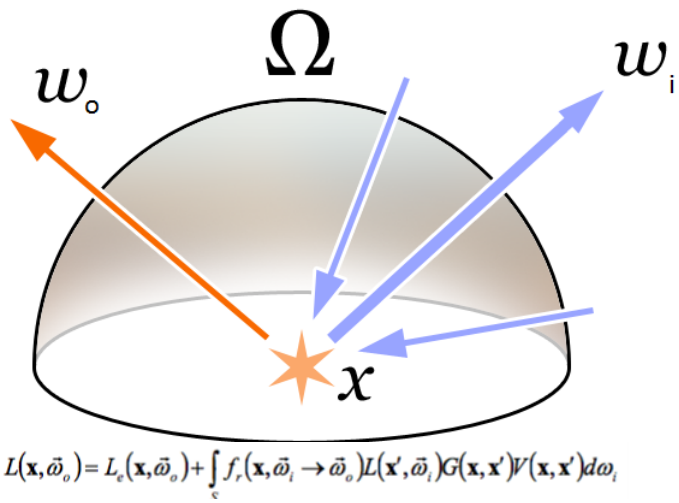
- Connect our neural representation to out measurement domain



Forward Map

- Graphics: foward map is the rendering equation (BRDF modeling)

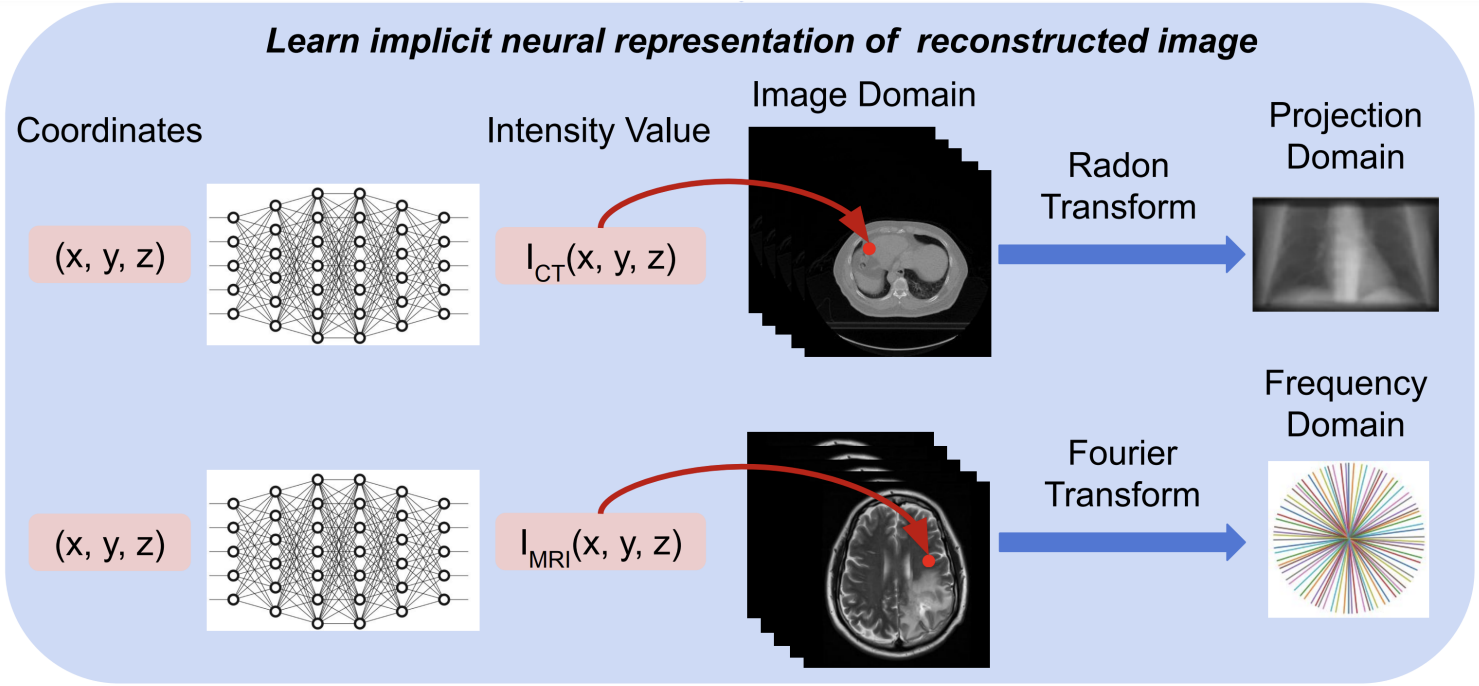




[Oechsle et al. 2018fr]

Forward Maps

- Medical imaging: forward map is Radon / Fourier transforms

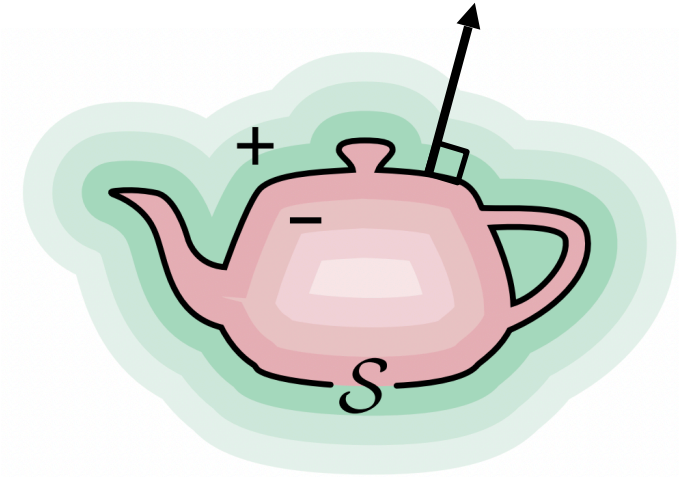


[Shen et al. 2020 (NeRP)]

Forward Maps

- Geometry: forward map is differential properties of signal
  - e.g. Eikonal loss to make the field an SDF

$$\mathcal{L} = \left\| \sum_{i=1}^{n} \left(\frac{\partial u}{\partial x_i} \right)^2 - 1 \right\|$$

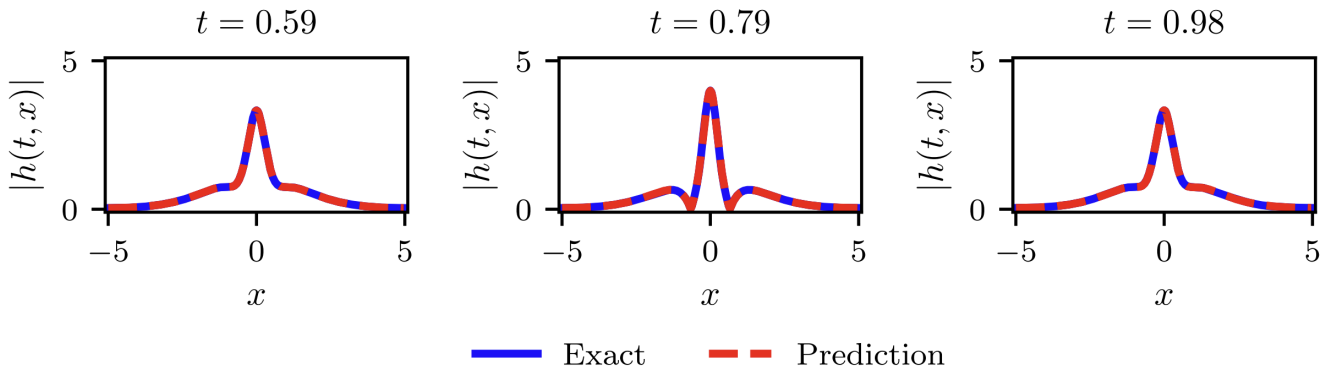


Gropp et al. “Implicit Geometric Regularization for Learning Shapes” ICML 2020

Forward Map

- Physics: a differential equation
  - e.g. Schrödinger's Equation

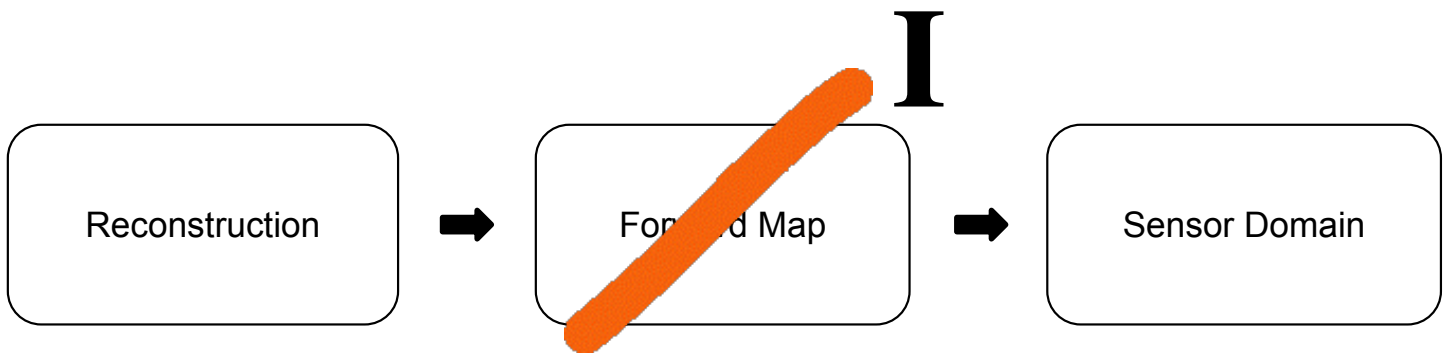
$$-\frac{\hbar^2}{2m}\nabla^2\psi + V\psi = E\psi$$



[Raissi et al. 2019 (PINN)]

Forward Map

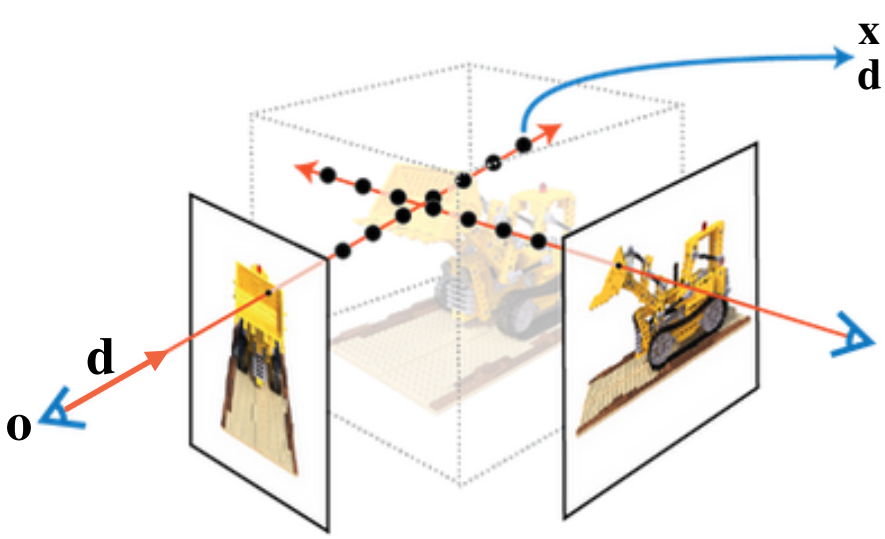
- Image compression: the identity mapping



Fathony et al. “Multiplicative Filter Networks” ICLR’2021

Forward Map

- Volume rendering: optimize the neural field parameters $\boldsymbol{\theta}$
  - pick random rays/pixels from training images
  - minimize the L2 hotometric reconstruction loss

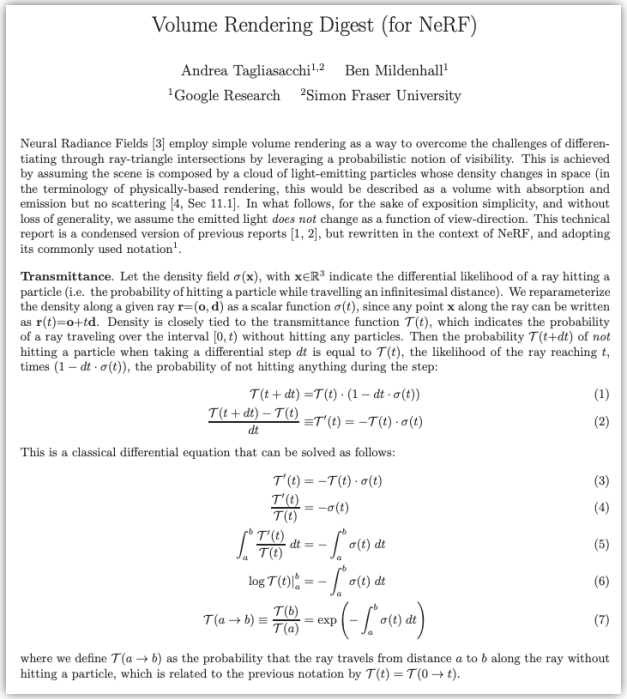


$$\mathcal{L}_{\text{rgb}}(\boldsymbol{\theta}) = \sum_i \mathbb{E}_{\mathbf{r}\sim\mathbf{C}_i}\left[\mathcal{L}^{\mathbf{r},i}_{\text{rgb}}(\boldsymbol{\theta})\right]$$

$$\mathcal{L}^{\mathbf{r},i}_{\text{rgb}}(\boldsymbol{\theta}) = ||\mathbf{C}(\mathbf{r};\boldsymbol{\theta}) - \mathbf{C}_i(\mathbf{r})||_2^2$$

Forward Map

- Volume rendering: pixel color is a convex combination of spatial radiance



Volume Rendering Digest (for NeRF)

Andrea Tagliasacchi[1,2] Ben Mildenhall[1]

[1]Google Research [2]Simon Fraser University

Neural Radiance Fields [3] employ simple volume rendering as a way to overcome the challenges of differentiating through ray-triangle intersections by leveraging a probabilistic notion of visibility. This is achieved by assuming the scene is composed by a cloud of light-emitting particles whose density changes in space (in the terminology of physically-based rendering, this would be described as a volume with absorption and emission but no scattering [4, Sec 11.1]. In what follows, for the sake of exposition simplicity, and without loss of generality, we assume the emitted light *does not* change as a function of view-direction. This technical report is a condensed version of previous reports [1, 2], but rewritten in the context of NeRF, and adopting its commonly used notation[1].

Transmittance. Let the density field $\sigma(\mathbf{x})$, with $\mathbf{x}\in\mathbb{R}^3$ indicate the differential likelihood of a ray hitting a particle (i.e. the probability of hitting a particle while travelling an infinitesimal distance). We reparameterize the density along a given ray $\mathbf{r}=(\mathbf{o},\mathbf{d})$ as a scalar function $\sigma(t)$, since any point $\mathbf{x}$ along the ray can be written as $\mathbf{r}(t)=\mathbf{o}+t\mathbf{d}$. Density is closely tied to the transmittance function $\mathcal{T}(t)$, which indicates the probability of a ray traveling over the interval $[0,t)$ without hitting any particles. Then the probability $\mathcal{T}(t+dt)$ of *not* hitting a particle when taking a differential step dt is equal to $\mathcal{T}(t)$, the likelihood of the ray reaching t, times $(1-dt\cdot\sigma(t))$, the probability of not hitting anything during the step:

$$\mathcal{T}(t+dt) = \mathcal{T}(t)\cdot(1-dt\cdot\sigma(t)) \quad (1)$$

$$\frac{\mathcal{T}(t+dt)-\mathcal{T}(t)}{dt} \equiv \mathcal{T}'(t) = -\mathcal{T}(t)\cdot\sigma(t) \quad (2)$$

This is a classical differential equation that can be solved as follows:

$$\mathcal{T}'(t) = -\mathcal{T}(t)\cdot\sigma(t) \quad (3)$$

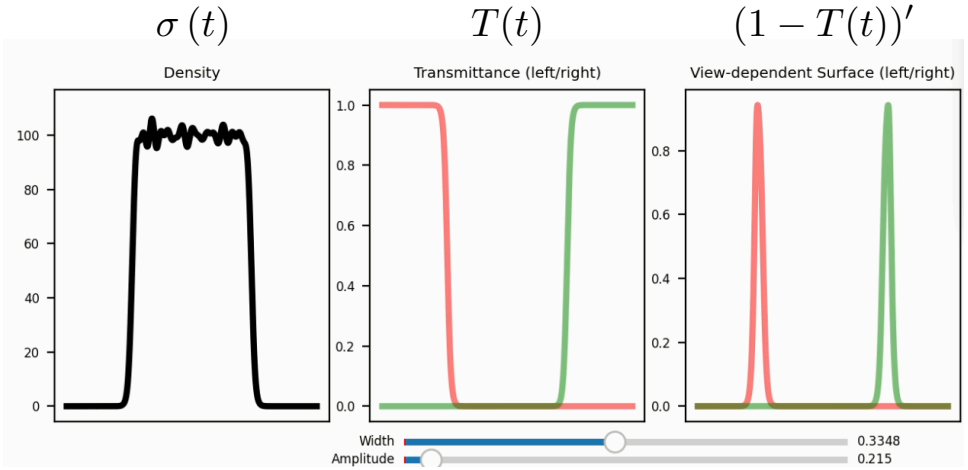
$$\frac{\mathcal{T}'(t)}{\mathcal{T}(t)} = -\sigma(t) \quad (4)$$

$$\int_a^b \frac{\mathcal{T}'(t)}{\mathcal{T}(t)}\,dt = -\int_a^b \sigma(t)\,dt \quad (5)$$

$$\log\mathcal{T}(t)\Big|_a^b = -\int_a^b \sigma(t)\,dt \quad (6)$$

$$\mathcal{T}(a\to b) \equiv \frac{\mathcal{T}(b)}{\mathcal{T}(a)} = \exp\left(-\int_a^b \sigma(t)\,dt\right) \quad (7)$$

where we define $\mathcal{T}(a\to b)$ as the probability that the ray travels from distance a to b along the ray without hitting a particle, which is related to the previous notation by $\mathcal{T}(t)=\mathcal{T}(0\to t)$.



$$\mathbf{C}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\cdot\sigma(\mathbf{r}(t))\cdot\mathbf{c}(\mathbf{r}(t),\mathbf{d})\,dt$$

$$= \int_{t_n}^{t_f} \underbrace{(1-T(t))'}_{\text{geometry}}\cdot\underbrace{\mathbf{c}(\mathbf{r}(t),\mathbf{d})}_{\text{appearance}}\,dt$$

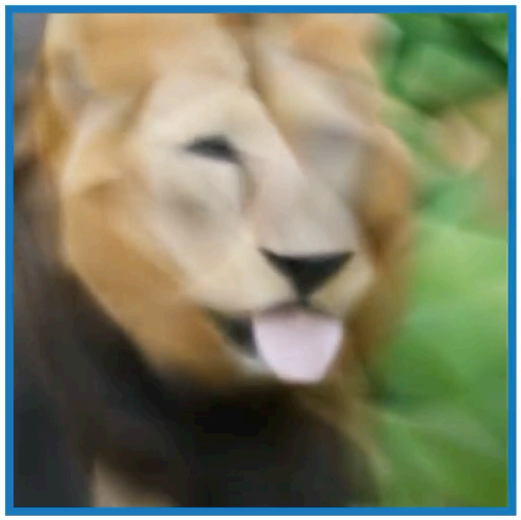
Tagliasacchi and Mildenhall "Volume Rendering Digest (for NeRF)" arXiv'2022

Outline

- Signals
- Forward map
- Architectures

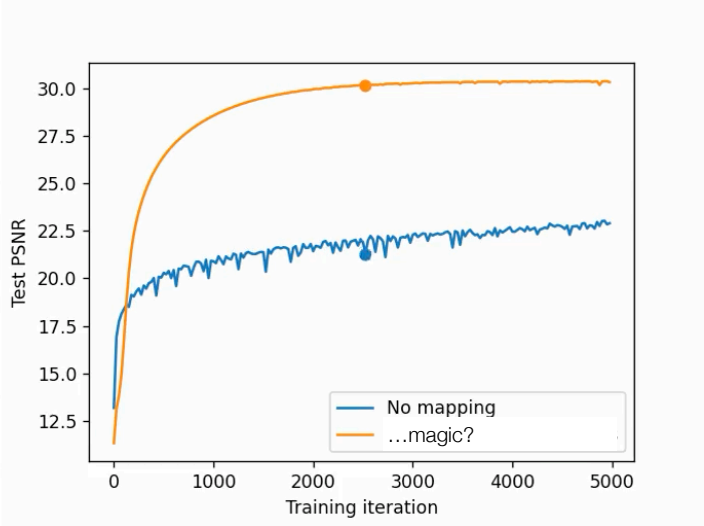
Architecture: input encodings

- Spectral bias: neural networks are biased to fit lower frequency signals
  - difficult to model high-frequency details



ReLU MLP

…magic?



Rahman et al. “On the spectral bias of neural networks” ICML’19

Architecture: input encodings

- The signals we are interested in are high frequency
  - Are then MLP really the right way to represent them?

Baatz et al. “NeRF-Tex: Neural Reflectance Field Textures” EGSR’21

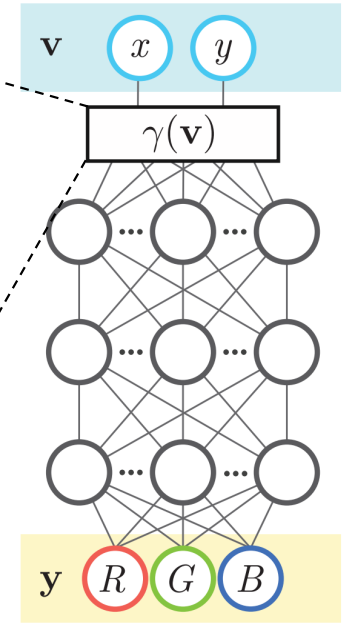
Architecture: input encodings

- Positional encodings
  - expand the dimensionality of the input (e.g. 2 → 2M dimensions)
  - via a collection of sinusoidal basis functions

$$\gamma(\mathbf{x}) = [\gamma_1(\mathbf{x}), \gamma_2(\mathbf{x}), ..., \gamma_m(\mathbf{x})]$$

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

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



Vaswani et al. “Attention is all you need” NeurIPS’17

Coding Exercise – Task 2

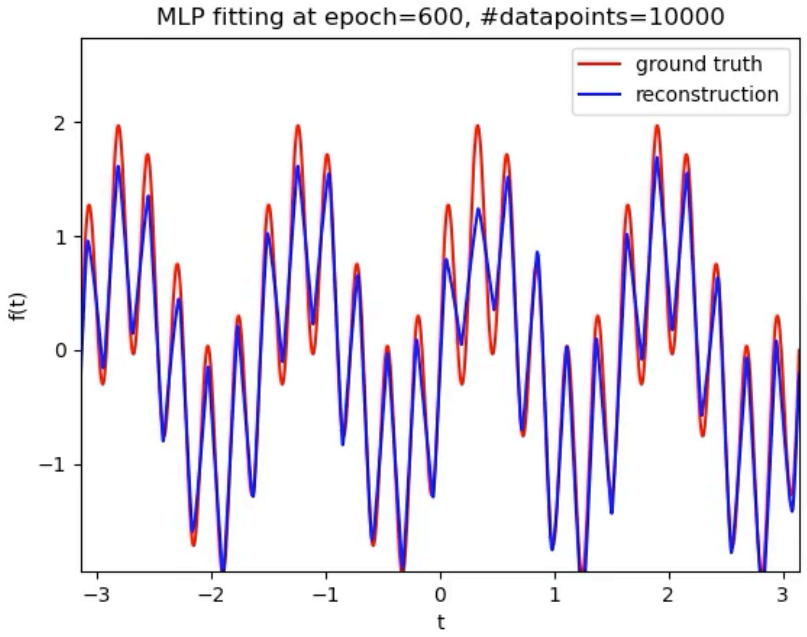
- Fit a signal by optimizing the MLP parameters
  - same network/objective as before
  - add positional encoding at input layer

$$z^{(1)} = \gamma(x)$$

~~$z^{(1)} = x$~~

$$z^{(i+1)} = \sigma\left(W^{(i)} z^{(i)} + b^{(i)}\right), \; i = 1, \ldots, k-1$$

$$f(x) = W^{(k)} z^{(k)} + b^{(k)}$$



Architecture: input encodings

- Input encodings: many new variants emerging

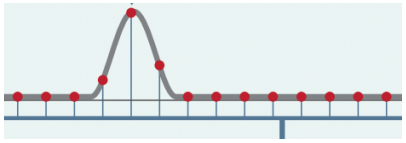
$$\boldsymbol{\Phi}(\boldsymbol{x}) = [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \ldots, \phi_D(\boldsymbol{x})]^T,$$

$$\left[e^{-\frac{(\boldsymbol{x}\cdot\boldsymbol{\alpha}-t_i)^2}{2\sigma_{\boldsymbol{x}}^2}}\right]^b$$

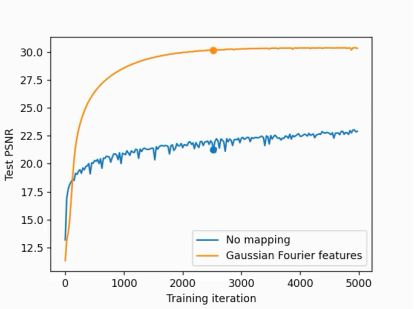
Super Gaussian [Ramasinghe et al. 2021]

$$\gamma(\mathbf{v}) = [\cos(2\pi\mathbf{B}\mathbf{v}), \sin(2\pi\mathbf{B}\mathbf{v})]^{\mathrm{T}}$$

Random Fourier [Tancik et al. 2020]

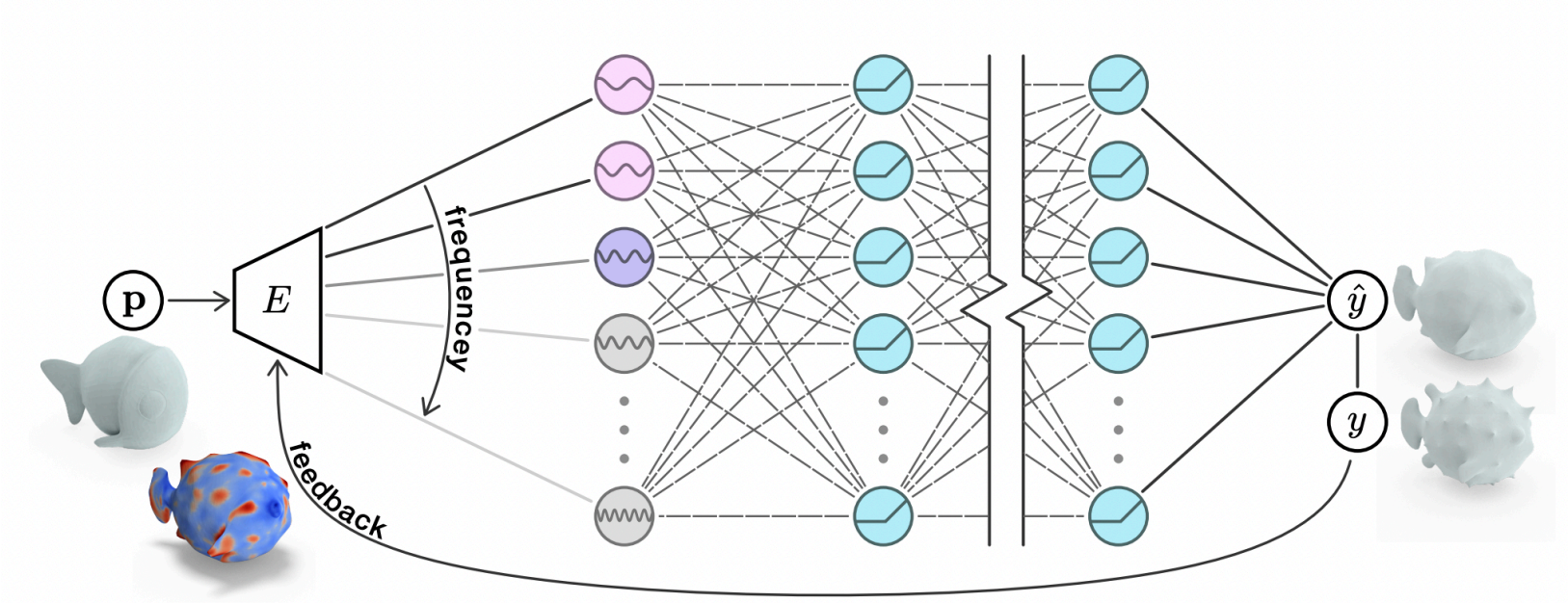


One-blob [Müller et al. 2020]



Architecture: input encodings

- Progressive Positional Encodings (PPE)
  - often a good idea to optimize multi-scale / coarse-to-fine



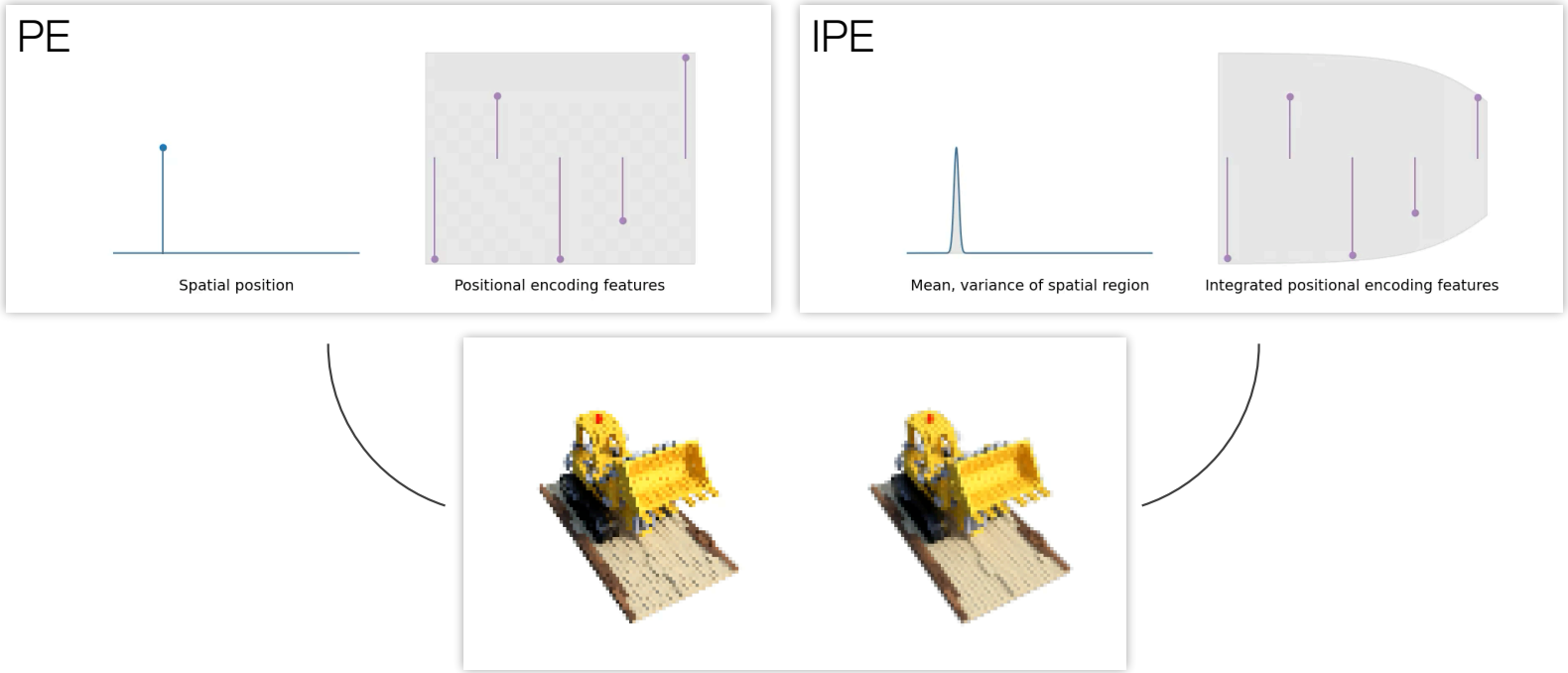
Hertz et al. “SAPE: Spatially-Adaptive Progressive Encoding for Neural Optimization” NeurIPS’21

Architecture: input encodings

- Integrated Positional Encodings (IPE)

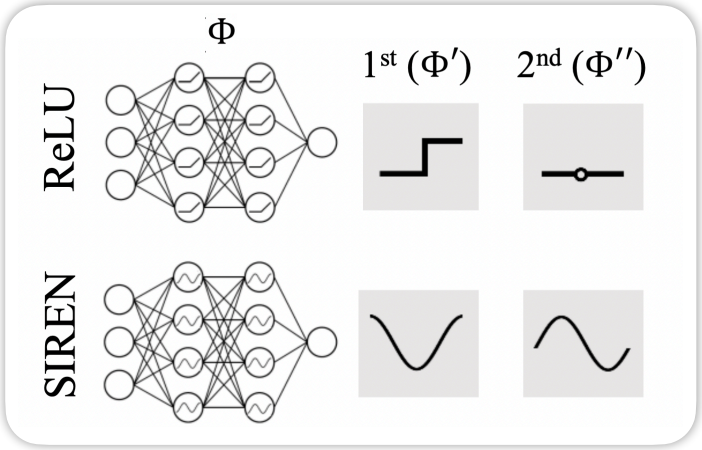
$$\gamma_\omega(x) = \sin(\omega x), \quad \gamma(x) = \left(\gamma_{2^\ell}(x)\right)_{\ell=0}^{L-1}$$

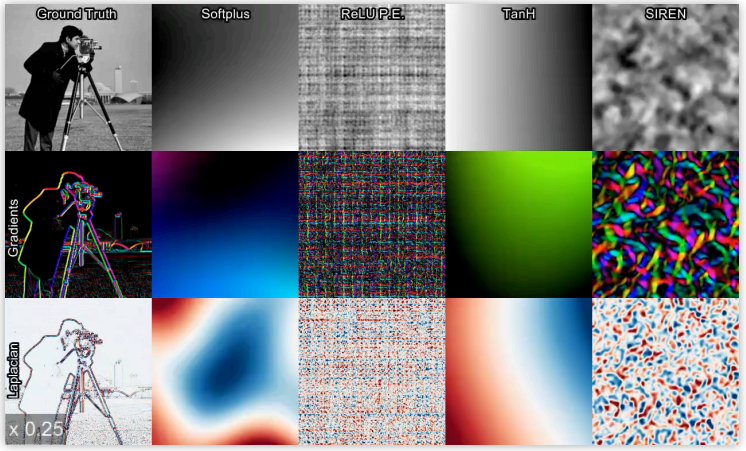
$$E_{x\sim\mathcal{N}(\mu,\sigma^2)}\left[\gamma_\omega(x)\right] = \sin(\omega\mu)\exp\left(-(\omega\sigma)^2/2\right)$$



Architectures: activation functions

- Sinusoids at the input layer are “good”…
  - then why not try to apply them at every layer?
  - outcome: not only field is continuous, but also all of its derivatives





$$\arg\min_{\boldsymbol{\theta}} \|\nabla\Phi(\mathbf{x};\boldsymbol{\theta}) - \mathbf{n}(\mathbf{x})\|_2^2$$

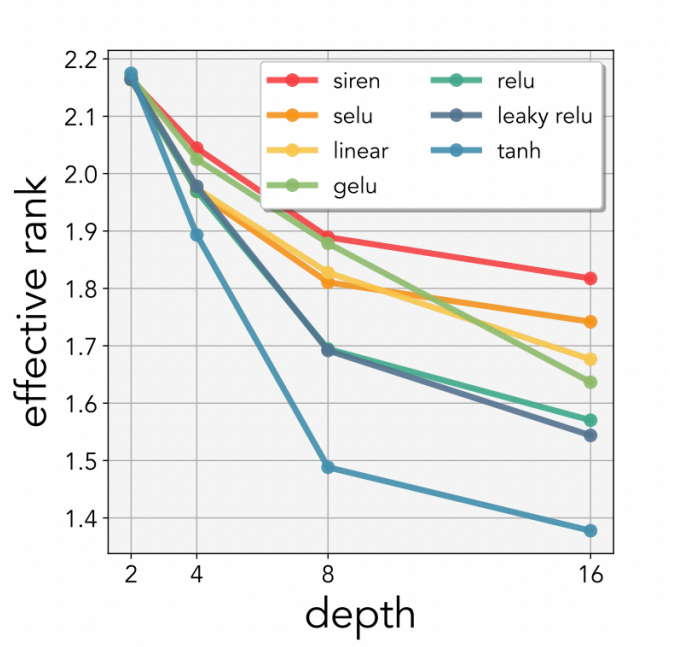
Sitzmann et al. “Implicit Neural Representations with Periodic Activation Functions” NeurIPS 2021 – https://www.vincentsitzmann.com/siren

Architectures: activation functions

- Smooth functions are “good” (sinusoids are smooth)
  - … what other smooth functions can we try?

Gaussian	$e^{\frac{-0.5x^2}{a^2}}$
Quadratic	$\frac{1}{1+(ax)^2}$
Multi Quadratic	$\frac{1}{\sqrt{1+(ax)^2}}$
Laplacian	$e^{(\frac{-\lvert x\rvert}{a})}$
Super-Gaussian	$[e^{\frac{-0.5x^2}{a^2}}]^b$
ExpSin	$e^{-\sin(ax)}$

[Ramasinghe et al. 2021]



[Huh et al. 2021]

Architectures: activation functions

- Smooth functions are “good” (sinusoids are smooth)
  - Can we modify the network architecture to decompose the signal?

MLP

$$z^{(1)} = x$$

$$z^{(i+1)} = \sigma\left(W^{(i)} z^{(i)} + b^{(i)}\right), \; i = 1, \ldots, k-1$$

$$f(x) = W^{(k)} z^{(k)} + b^{(k)}$$

non-linearity

MFN

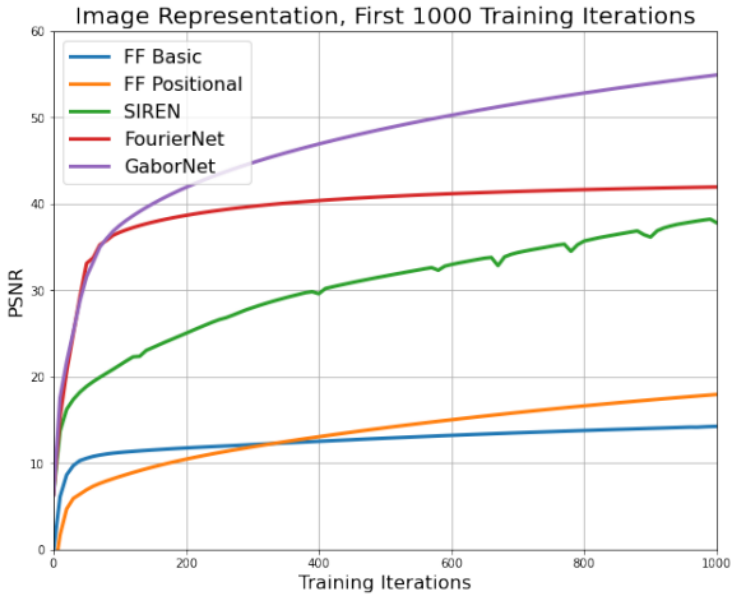
$$g(x; \theta^{(i)}) = \sin(\omega^{(i)} x + \phi^{(i)})$$

FourierNet

$$z^{(1)} = g\left(x; \theta^{(1)}\right)$$

$$z^{(i+1)} = \left(W^{(i)} z^{(i)} + b^{(i)}\right) \circ g\left(x; \theta^{(i+1)}\right), \; i = 1, \ldots, k-1$$

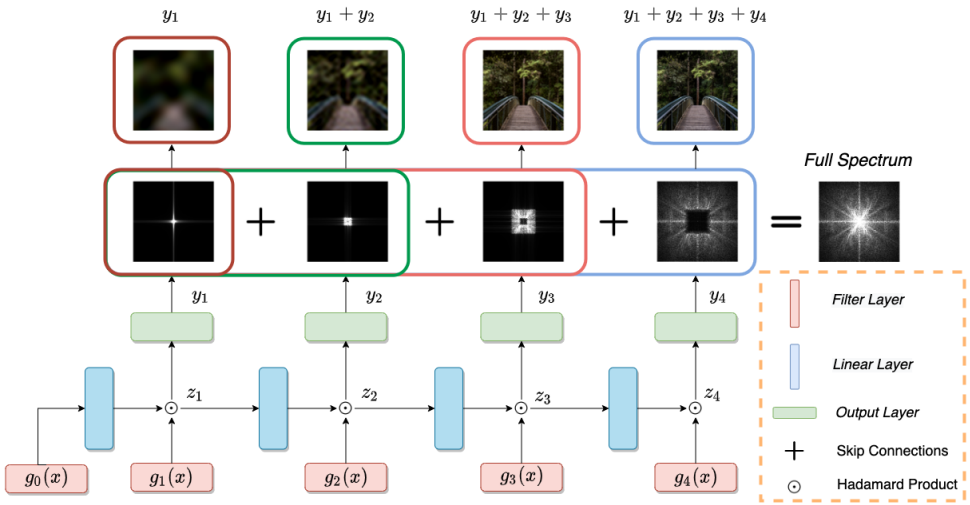
$$f(x) = W^{(k)} z^{(k)} + b^{(k)}$$



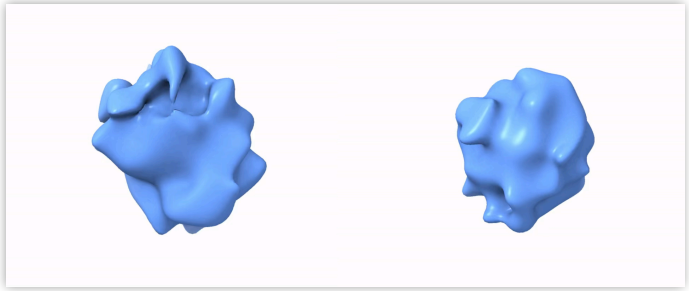
Fathony et al. “Multiplicative Filter Networks” ICLR’2021

Architectures: activation functions

- Smooth functions are “good” (sinusoids are smooth)
  - Modify the network architecture to decompose the signal into frequency bands?



training data

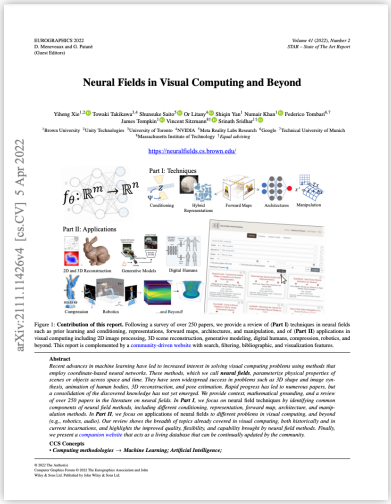


reconstruction

Shekarforoush et al. “Residual Multiplicative Filter Networks for Multiscale Reconstruction” NeurIPS’2022

Summary

- Resources
  - Xie et al. “Neural Fields in Visual Computing and Beyond”, Eurographics 2022
  - Tagliasacchi et al. “Volume Rendering Digest (for NeRF)” arXiv’2022
  - Tewari et al. “Advances in Neural Rendering” Eurographics 2022



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

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

Neural Fields in Visual Computing and Beyond

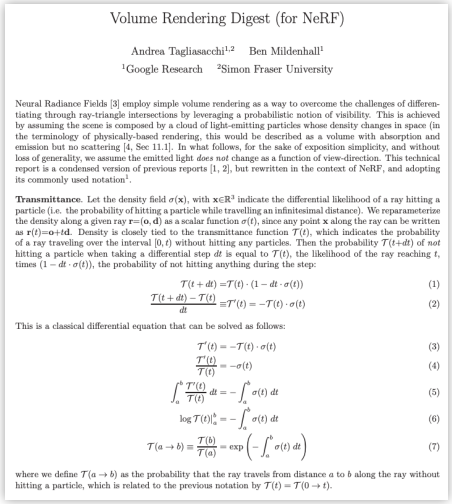
Yiheng Xie[1,2] Towaki Takikawa[1,3] Shunsuke Saito[4] Or Litany[5] Shiqin Yan[1] Numair Khan[6] Federico Tombari[6,7]
James Tompkin[8] Vincent Sitzmann[9] Srinath Sridhar[1,8]
[1]Brown University [2]Unity Technologies [3]University of Toronto [4]NVIDIA [5]Meta Reality Labs Research [6]Google [7]Technical University of Munich
[8]Massachusetts Institute of Technology [9]Equal advising

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

Abstract

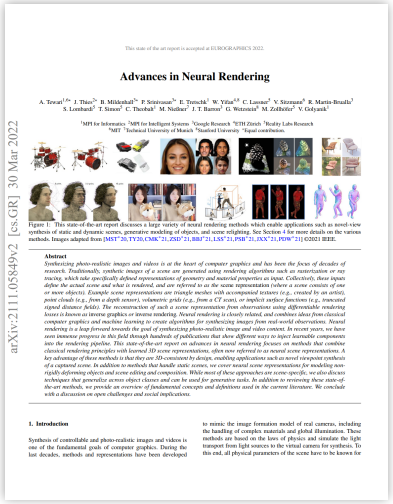
*Recent advances in machine learning have led to increased interest in solving visual computing problems using methods that employ coordinate-based neural networks. These methods, which we call **neural fields**, parameterize physical properties of scenes or objects across space and time. They have seen widespread success in problems such as 3D shape and image synthesis, animation of human bodies, 3D reconstruction, and pose estimation. Rapid progress has led to numerous papers, but a consolidation of the discovered knowledge has not yet emerged. We provide context, mathematical grounding, and a review of over 250 papers in the literature on neural fields. In **Part I**, we focus on neural field techniques by identifying common components of neural field methods, including different conditioning, representation, forward map, architecture, and manipulation methods. In **Part II**, we focus on applications of neural fields to different problems in visual computing, and beyond (e.g., robotics, audio). Our review shows the breadth of topics already covered in visual computing, both historically and in current incarnations, and highlights the improved quality, flexibility, and capability brought by neural field methods. Finally, we present a companion website that acts as a living database that can be continually updated by the community.*

CCS Concepts
• ***Computing methodologies*** → ***Machine Learning; Artificial Intelligence;***



Volume Rendering Digest (for NeRF)

Andrea Tagliasacchi[1,2] Ben Mildenhall[1]
[1]Google Research [2]Simon Fraser University



Advances in Neural Rendering

A. Tewari[1,25] J. Thies[2,25] B. Mildenhall[3,25] P. Srinivasan[3,25] E. Tretschk[1] W. Yifan[4,5] C. Lassner[6] V. Sitzmann[8] R. Martin-Brualla[7]
S. Lombardi[6] T. Simon[6] C. Theobalt[1] M. Nießner[2] J. T. Barron[3] G. Wetzstein[8] M. Zollhöfer[6] V. Golyanik[1]
[1]MPI for Informatics [2]MPI for Intelligent Systems [3]Google Research [4]ETH Zürich [5]Reality Labs Research
[6]MIT [7]Technical University of Munich [8]Stanford University [25]Equal contribution.

Figure 1: This state-of-the-art report discusses a large variety of neural rendering methods which enable applications such as novel-view synthesis of static and dynamic scenes, generative modeling of objects, and scene relighting. See Section 4 for more details on the various methods. Images adapted from [MST*20, FYT20, CMK*21, ZSD*21, BBJ*21, LSS*21, PSH*21, DXX*21, PDW*21] ©2021 IEEE.

Abstract

Synthesizing photo-realistic images and videos is at the heart of computer graphics and has been the focus of decades of research. Traditionally, synthetic images of a scene are generated using rendering algorithms such as rasterization or ray tracing, which take specifically defined representations of geometry and material properties as input. Collectively, these inputs define the actual scene and what is rendered, and are referred to as the scene representation (where a scene consists of one or more objects). Example scene representations are triangle meshes with accompanied textures (e.g., created by an artist), point clouds (e.g., from a depth sensor), volumetric grids (e.g., from a CT scan), or implicit surface functions (e.g., truncated signed distance fields). The reconstruction of such a scene representation from observations using differentiable rendering losses is known as inverse graphics or inverse rendering. Neural rendering is closely related, and combines ideas from classical computer graphics and machine learning to create algorithms for synthesizing images from real-world observations. Neural rendering is a leap forward towards the goal of synthesizing photo-realistic image and video content. In recent years, we have seen immense progress in this field through hundreds of publications that show different ways to inject learnable components into the rendering pipeline. This state-of-the-art report on advances in neural rendering focuses on methods that combine classical rendering principles with learned 3D scene representations, often now referred to as neural scene representations. A key advantage of these methods is that they are 3D-consistent by design, enabling applications such as novel viewpoint synthesis of a captured scene. In addition to methods that handle static scenes, we cover neural scene representations for modeling non-rigidly deforming objects and scene editing and composition. While most of these approaches are scene-specific, we also discuss techniques that generalize across object classes and can be used for generative tasks. In addition to reviewing these state-of-the-art methods, we provide an overview of fundamental concepts and definitions used in the current literature. We conclude with a discussion on open challenges and social implications.

1. Introduction

Synthesis of controllable and photo-realistic images and videos is one of the fundamental goals of computer graphics. During the last decades, methods and representations have been developed to mimic the image formation model of real cameras, including the handling of complex materials and global illumination. These methods are based on the laws of physics and simulate the light transport from light sources to the virtual camera for synthesis. To this end, all physical parameters of the scene have to be known for

Coding Exercises

- Task 1: fit a signal with an MLP
- Task 2: add positional encoding (remove spectral bias)
- Task 3: make the network conditional (learn families of functions)

https://classroom.github.com/a/duduR83_