Neural Field Representations
neural fields, neural rendering, applications

CSC2529
David Lindell
University of Toronto
cs.toronto.edu/~lindell/teaching/2529
Announcements

• Poster session is next Thurs. Dec 8!
  • Remember to print your poster ahead of time, follow instructions on Quercus

• Reach out to project mentors with any questions/request feedback

• No in-person lecture next week– guest lecture over zoom
Outline

• Overview of neural fields
• Representing 3D shape
• Neural rendering with NeRF
• Case studies
  • SIREN
  • AutoInt
  • ACORN
  • BACON
Conventional Representations

• Number of samples related to global highest frequency (i.e., Nyquist)

• Can be difficult to optimize due to many parameters
  • Need hand-crafted priors for ill-posed problems

• (sparse) basis representations can help, but are not very flexible for high dimensional or multimodal signals
Neural fields

Field: a set of numbers along with mathematical operations defined on that set.

CVPR 2022 workshop: “a scalar or vector-valued quantity defined across an input domain”
• Neural field: “a field that is parameterized partly or fully by a neural network”
Neural fields

\[ \Phi(x) : \mathbb{R}^2 \rightarrow \mathbb{R} \]

Pixel coordinates → Pixel values
Neural field explosion!

Began around 2019 and accelerated since NeRF (2020)

https://neuralfields.cs.brown.edu/
Neural Field Representations

• Compact representations

• Differentiable and easy to optimize (allows learning priors)

• Multi-modal and easy to scale to high dimensions
Compact Representations

This scene is stored in about 30 MB of trained network weights (optimized from ~2 GB of image pixels)
Differentiable/Easy to Optimize

Becomes trivial to reconstruct 3D appearance and geometry from multiview imagery.
Differentiable/Easy to Optimize

Can learn priors/generative models over a space of signals

[Chan et al. ‘22]
Multi-modal

A model for text to 3D shape and appearance based on neural fields

[Poole et al. ‘22]
Outline

• Overview of neural fields
• Representing 3D shape
• Neural rendering with NeRF
• Case studies
  • SIREN
  • AutoInt
  • ACORN
  • BACON
3D Representations

• Meshes
• Point clouds
• Voxel grids
• ...

• Cumbersome to represent with a neural network!
Occupancy Fields

(a) Voxel  (b) Point  (c) Mesh  (d) Ours

$\mathbf{f}_\theta(p) = \tau$
Occupancy Fields

for $i = 1, \ldots, 5$

[Mescheder et al. ‘19]
Occupancy Fields

[Image of diagram showing the Occupancy Fields model]

Input | Ours

[Images of different objects as examples of input and output]

[Mescheder et al. ‘19]
Signed Distance Field
Signed Distance Field

(a) Single Shape DeepSDF

[Park et al. ’19]
Signed Distance Field

(a) Single Shape DeepSDF

(b) Coded Shape DeepSDF

[Park et al. '19]
Signed Distance Field

(a) Single Shape DeepSDF
(b) Coded Shape DeepSDF

(a) Noisy Input Point Cloud
(b) Shape Completion

[Park et al. ’19]
NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction

Peng Wang, Litao Tao

Deep Local Shapes: Learning Local SDF Priors for Detailed 3D Reconstruction

Rohan Chabra, Strout, etc.

We present a new approach to learning neural implicit surfaces for detailed 3D reconstruction. Our method utilizes a voxelized representation to efficiently capture fine details in complex shapes. We evaluate our approach on the DTU benchmark and show superior performance compared to previous methods.

PIFu: Pixel-Aligned Implicit Function for High-Resolution Clothed Human Digitization

Shunsuke Saito, Zeng Huang, Shigeo Morishima, Angjoo Kanazawa, Hao Li

1University of Southern California 2USC Institute for Creative Technologies 3Waseda University 4University of California, Berkeley 5Pinscreen

Fig. 1: Reconstruction of Calais scene [6] using PIFu. The function is learned from a single image and can be used for multi-view reconstruction. The results show high-resolution details of the clothing and body shapes.

Figure 1: Pixel-aligned Implicit function (PIFu): We present pixel-aligned implicit function (PIFu), which allows recovery of high-resolution 3D textured surfaces of clothed humans from a single input image (top row). Our approach can digitize intricate variations in clothing, such as wrinkled skirts and high-heels, including complex hairstyles. The shape and textures can be fully recovered including unseen regions such as the back of the subject. PIFu can be also extended to multi-view input images (bottom row).
Outline

• Overview of neural fields
• Representing 3D shape
• Neural rendering with NeRF
• Case studies
  • SIREN
  • AutoInt
  • ACORN
  • BACON
Neural Radiance Fields

[Mildenhall et al. ‘20]
Mildenhall et al. ‘20
Liu et al. ‘20
Schwarz et al. ‘20
...

Training Set  Learned Volume  Novel Views (from VRE)
Volume Rendering Equation (VRE)

\[
C(r) = \int_{t_0}^{t_f} \sigma(r(t)) e^{-\int_{t_0}^{t} \sigma(r(s)) \, ds} c(r(t)) \, dt
\]

- **Position, direction**
- **Emissive radiance**
- **Absorption coefficient**
- **Transmittance**
- **Color of rendered ray**
Volume Rendering Equation (VRE)

\[
C(r) = \int_{t_n}^{t_f} \sigma(r(t)) e^{-\int_{t_n}^{t} \sigma(r(s)) \, ds} c(r(t)) \, dt
\]

- Color of rendered ray
- Absorption coefficient
- Transmittance
- Emissive radiance
Discretizing the VRE

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

- color of rendered ray
- absorption coefficient
- transmittance
- emissive radiance

How much light is emitted along a single section? Assuming constant radiance...
Discretizing the VRE

\[ C(\mathbf{r}) = \int_{t_n}^{t_f} \sigma(\mathbf{r}(t)) e^{-\int_{t_n}^{t} \sigma(\mathbf{r}(s)) \, ds} c(\mathbf{r}(t)) \, dt \]

- color of rendered ray
- absorption coefficient
- transmittance
- emissive radiance

• Assume constant radiance

\[ \approx c \int_{t_n}^{t_f} \sigma(t) e^{-\int_{t_n}^{t} \sigma(s) \, ds} \, dt \]
Discretizing the VRE

\[ \approx c \int_{t_n}^{t_f} \sigma(t)e^{-\int_{t_n}^{t} \sigma(s) \, ds} \, dt \]
Discretizing the VRE

\[ \approx \mathbf{c} \int_{t_n}^{t_f} \sigma(t) e^{-\int_{t_n}^{t} \sigma(s) ds} dt \]

\[ = \mathbf{c} \int_{t_n}^{t_f} \frac{d}{dt} e^{-\int_{t_n}^{t} \sigma(s) ds} dt \]
Discretizing the VRE

\[ \approx c \int_{t_n}^{t_f} \sigma(t) e^{-\int_{t_n}^{t} \sigma(s) \, ds} \, dt \]

\[ = c \int_{t_n}^{t_f} -\frac{d}{dt} e^{-\int_{t_n}^{t} \sigma(s) \, ds} \, dt \]

\[ = c \left( 1 - e^{-\int_{t_n}^{t} \sigma(s) \, ds} \right) \]
Discretizing the VRE

\[ = c \left( 1 - e^{-\int_{t_0}^{t} \sigma(s) \, ds} \right) \]

\[ \approx \sum_{i=1}^{N} (1 - \exp(-\sigma_i \delta_i)) \exp \left( -\sum_{j=1}^{i-1} \sigma_j \delta_j \right) c_i \]

discretize and sum over sections
Discretizing the VRE

\[ = c \left( 1 - e^{- \int_{t_n}^{t} \sigma(s) \, ds} \right) \]

\[ \approx \sum_{i=1}^{N} (1 - \exp(-\sigma_i \delta_i)) \exp \left( - \sum_{j=1}^{i-1} \sigma_j \delta_j \right) c_i \]

discretize and sum over sections

transmittance
Discretizing the VRE

\[
= c \left( 1 - e^{-\int_{t_n}^{t} \sigma(s) \, ds} \right)
\]

\[
\approx \sum_{i=1}^{N} \left( 1 - \exp(-\sigma_i \delta_i) \right) \exp \left( -\sum_{j=1}^{i-1} \sigma_j \delta_j \right) c_i
\]
Discretizing the VRE

\[ C(r) = \int_{t_n}^{t_f} \sigma(r(t)) e^{\left\{-\int_{t_n}^{t} \sigma(r(s)) \, ds\right\}} c(r(t)) \, dt \]

- Color of rendered ray
- Absorption coefficient
- Transmittance
- Emissive radiance

\[ \approx \sum_{i=1}^{N} (1 - \exp(-\sigma_i \delta_i)) \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right) c_i \]
Neural Radiance Fields

\[(x, y, z, \theta, \phi)\]  
\[(\sigma, c)\]
Neural Radiance Fields

- Given images with known camera positions
- Sample along rays
- Optimize the absorption and radiance to minimize photometric error!
Naïve NeRF produces blurry results...
Positional Encoding

Naïve NeRF produces blurry results...  
Result with positional encoding
Positional Encoding

Toy example of image fitting

Without Pos. Enc.  With Pos. Enc
Positional Encoding

- Simple trick!
  - Instead of passing in \( v = (x, y) \) into the network we pass

\[
\begin{bmatrix}
\cos(\pi v) & \sin(\pi v) \\
\cos(2\pi v) & \sin(2\pi v) \\
\cos(4\pi v) & \sin(4\pi v) \\
\vdots & \vdots \\
\cos(2^{L-1}\pi v) & \sin(2^{L-1}\pi v)
\end{bmatrix}
\]
Positional Encoding

• Why does this work?

• Explained with theory from Neural Tangent Kernel
  • Training a network is similar to kernel regression
    (becomes closer as network layers become wider)
Positional Encoding

• Kernel Regression

weighting & kernel function computes similarity between input and training points

\[ f(x) = \sum_{i=1}^{N} w_i k(x - x_i) \]

Sum over training points

e.g., if the kernel is a Gaussian, this puts a bump at every training data point
Width of kernel is important to trade off interpolation/overfitting to data!
Width of kernel is important to trade off interpolation/overfitting to data!
NTK Visualization

- Graph 1: PSNR vs. Fourier feature scale $\sigma$, with green line for 'Train' and red line for 'Test'.
- Graph 2: Slice of 2D neural tangent kernel.
NeRF Results

Detailed geometry (depth map visualizes the location of the mean of the absorption)
NeRF Results

View dependent effects (right is fixing the ray position, but feeding different ray direction)
Outline

• Overview of neural fields
• Representing 3D shape
• Neural rendering with NeRF
• Case studies
  • SIREN
  • AutoInt
  • ACORN
  • BACON
Quantities defined by a differential equation
SIREN: Sinusoidal Representation Networks

Quantities defined by a differential equation

NN(x, y, z)
Periodicity allows SIREN to replicate activations across the input domain.

All derivatives exist, are nonzero and bounded by 1.
ReLU MLP

SIREN

Quantities defined by a differential equation
Related Work

Implicit Representations

Mescheder et al. 2018
Park et al. 2018
Chen et al. 2018
Lee et al. 1990
Lagaris et al. 1998
He et al. 2000
Gallant et al. 1988
Sopena et al. 1999
Candès et al. 1999
Mai-Duy et al. 2003
Sirignano et al. 2018
Raissi et al. 2019
Parascandolo et al. 2016
Stanley et al. 2007
Sitzmann et al. 2019
Mildenhall et al. 2020
Mildenhall et al. 2020
…

Sine Activations in NNs

NN(x, y, z)
<table>
<thead>
<tr>
<th>Input</th>
<th>Output supervised by</th>
<th>Implicit Formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbf{x} \in \mathbb{R}^2$</td>
<td>$f(\mathbf{x}) \in \mathbb{R}^3$</td>
<td>Find $\Phi$ that minimizes $\mathcal{L}$</td>
</tr>
</tbody>
</table>

Spatial coords. | RGB values |

$$\mathcal{L}_{\text{img}} = \int_{\Omega} \| \Phi(\mathbf{x}) - f(\mathbf{x}) \| \, d\mathbf{x}$$
Representing Images

<table>
<thead>
<tr>
<th>GT</th>
<th>ReLU</th>
<th>Tanh</th>
<th>ReLU P.E.</th>
<th>RBF</th>
<th>SIREN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Grad</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lapl</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Graph showing PSNR over iterations for different activation functions (ReLU, Tanh, ReLU P.E., RBF, SIREN).
Input supervised by

\[ t \in \mathbb{R} \quad f(t) \in \mathbb{R} \]

Implicit Formulation

Find \( \Phi \) that minimizes \( \mathcal{L} \)

\[ \mathcal{L}_{\text{audio}} = \int_{\Omega} \| \Phi(t) - f(t) \| \, dt \]
Representing Audio – Voice

Ground Truth

ReLU MLP

ReLU w/ positional encoding

SIREN
Representing Audio – Music

Ground Truth

ReLU MLP

ReLU w/ positional encoding

SIREN
<table>
<thead>
<tr>
<th>Input</th>
<th>Output supervised by</th>
<th>Implicit Formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(x, t) \in \mathbb{R}^3$</td>
<td>$f(x, t) \in \mathbb{R}^3$</td>
<td>Find $\Phi$ that minimizes $\mathcal{L}$</td>
</tr>
</tbody>
</table>

- **space-time coord.**
- **RGB value**

$$\mathcal{L}_{video} = \int_\Omega \| \Phi(x, t) - f(x, t) \| \, dx \, dt$$
Representing Video

Ground Truth

ReLU MLP

SIREN
Representing Video

Ground Truth  ReLU MLP  SIREN
Input

\[ \mathbf{x} \in \mathbb{R}^2 \]

spatial coord.

\[ f(\mathbf{x}) \in \mathbb{R}^2 \]

gray level

Output supervised by

Implicit Formulation

Find \( \Phi \) that minimizes \( \mathcal{L} \)

\[ \mathcal{L}_{\text{Poisson}} = \int_{\Omega} \| \nabla \Phi(\mathbf{x}) - \nabla f(\mathbf{x}) \| \, d\mathbf{x} \]
Poisson’s Equation

Image

Gradient

Laplacian

supervision
Input supervised by Output

$x \in \mathbb{R}^3$ $f(x) \in \mathbb{R}$ 
spatial coord. signed distance

Implicit Formulation
Find $\Phi$ that minimizes $\mathcal{L}$

$$\mathcal{L}_{\text{Eikonal}} = \int_{\Omega_0} |\Phi(x)| + (1 - \langle \nabla \Phi(x), \nabla f(x) \rangle) \, dx$$
$$+ \int_{\Omega} \| |\nabla \Phi(x)| - 1 \| \, dx$$
3D Shapes - solving the Eikonal equation

ReLU

SIREN

5 layers, 256 hidden units
Input

Output supervised by

Implicit Formulation
Find $\Phi$ that minimizes $\mathcal{L}$

\[ x \in \mathbb{R}^2 \quad f(x) \in \mathbb{C} \quad \mathcal{L}_{\text{Helmholtz}} = \int_{\Omega} \| (\nabla + m(x)\omega^2)\Phi(x) - f(x) \| \, dx \]

spatial coord.
complex wave field
Solving the Helmholtz Equation

Perfectly-Matched Boundary Layers
Solving the Helmholtz Equation

Ground Truth  ReLU  Tanh  SIREN

Real

Imaginary
Input.Supervised by.

\[(x, t) \in \mathbb{R}^3\]

spatial coord.

\[f(x, t) \in \mathbb{R}\]

real wave field

\[\mathcal{L}_{\text{wave}} = \int_{\Omega} \left\| \frac{\partial^2 \Phi}{\partial t^2} - c^2 \Delta \Phi \right\| \, dx \, dt\]
Solving the Wave Equation
SIREN in generative models

Uses SIREN-based backbone for generative 3D model synthesis!

Pi-GAN [Chan et al. ’21]
GRAM [Deng et al. ‘22]
SIREN in physics solvers

Uses SIREN for efficient physics solvers (NVIDIA SimNet)

[Hennigh et al. ‘20]

NVIDIA SimNet™: An AI-accelerated Multi-Physics Simulation Framework

Preprint: December 16, 2020

Oliver Hennigh ohennigh@nvidia.com
Akshay Subramaniam ashsubramaniam@nvidia.com
Sushrutha Natarajan snatarajan@nvidia.com
Karteek Tampali ktampali@nvidia.com
Mohammad Amin Nabian mnabian@nvidia.com
Max Rietmann mrietmann@nvidia.com
Juse del Aguila Ferradis jaguilanvidia.com
Wynnin Byeon wbyeon@nvidia.com
Zhiwei Fang zf Fang@nvidia.com
Sanjay Chowdhury schowdhury@nvidia.com

NVIDIA developer@nvidia.com/simnet

Abstract

We present SimNet, an AI-driven multi-physics simulation framework, to accelerate simulations across a wide range of disciplines in science and engineering. Compared to traditional numerical solvers, SimNet addresses a wide range of use cases - coupled forward simulations without any training data, inverse and data assimilation problems. SimNet offers fast turnaround time by enabling parameterized system representation that solves for multiple configurations simultaneously, as opposed to the traditional solvers that solve for one configuration at a time. SimNet is integrated with parameterized constructive solid geometry as well as STL modules to generate point clouds. Furthermore, it is customizable with APIs that enable users to extend to new geometry, physics and simulation requirements.

Table 3: Total compute time needed for different solvers for the NVSwitch heat sink design optimization

<table>
<thead>
<tr>
<th>Solver</th>
<th>OpenFOAM</th>
<th>Commercial Solver</th>
<th>SimNet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute Time (x 1000 hrs.)</td>
<td>405935</td>
<td>137494</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 11: Streamlines colored with pressure and temperature profile in the fluid for optimal NVSwitch geometry.
Outline

• Overview of neural fields
• Representing 3D shape
• Neural rendering with NeRF
• Case studies
  • SIREN
  • ACORN
  • AutoInt
  • BACON
Evaluating a volume of 1K x 1K x 1K voxels requires 1 billion forward passes!

Challenges Towards Large Scale Neural Representations
Challenges

- Neural network capacity
- Training time takes hours or days already!
- Inference time is prohibitive

Evaluating a volume of 1K x 1K x 1K voxels requires 1 billion forward passes!
<table>
<thead>
<tr>
<th>Method</th>
<th>Comp. Efficiency</th>
<th>Mem. Efficiency</th>
<th>Online multiscale</th>
<th>Pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Global implicit</td>
<td>x</td>
<td>✓</td>
<td></td>
<td>x</td>
</tr>
<tr>
<td>Local implicit</td>
<td>x ✓</td>
<td>x ✓</td>
<td></td>
<td>x ✓</td>
</tr>
<tr>
<td>Hybrid Implicit-Explicit (ours)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Explicit methods

**Input:** features

**Output:** signal
<table>
<thead>
<tr>
<th>Method</th>
<th>Comp. Efficiency</th>
<th>Mem. Efficiency</th>
<th>Online multiscale</th>
<th>Pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Global implicit</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Local implicit</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Hybrid Implicit-Explicit (ours)</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

Global implicit methods

**Input:** coordinates  
**Output:** signal

- DeepSDF
- SIREN
- Fourier Features
<table>
<thead>
<tr>
<th>Method</th>
<th>Comp. Efficiency</th>
<th>Mem. Efficiency</th>
<th>Online multiscale</th>
<th>Pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Global implicit</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Local implicit</td>
<td>✗ ✔</td>
<td>✗ ✔</td>
<td>✗</td>
<td>✔ ✗</td>
</tr>
<tr>
<td>Hybrid Implicit-Explicit (ours)</td>
<td>✔ ✔</td>
<td>✔ ✔</td>
<td>✔ ✔</td>
<td>✔ ✔</td>
</tr>
</tbody>
</table>

**Local implicit methods**

**Input:** features & coordinates  
**Output:** signal

- Local Implicit Image Functions
- Deep Local Shapes
- Convolutional Occupancy Nets
- Neural Geometric Level of Detail
- Neural Sparse Voxel Fields
<table>
<thead>
<tr>
<th></th>
<th>Comp. Efficiency</th>
<th>Mem. Efficiency</th>
<th>Online multiscale</th>
<th>Pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explicit</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Global implicit</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Local implicit</td>
<td>× ✓</td>
<td>× ✓</td>
<td>×</td>
<td>✓ ×</td>
</tr>
<tr>
<td>Hybrid Implicit-Explicit (ours)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**ACORN: an hybrid implicit-explicit architecture**

- **Global coordinate encoder**
  - Input: global coordinates
  - Output: features

- **Local coordinate decoder**
  - Input: local coordinates & features
  - Output: signal
Scaling Up

Ground Truth (64 MP)

Closeup

Optimized ACORN

8192 px

PSNR (dB)

training time (hours)

ACORN

SIREN
The partition is optimized online, using an Integer Linear Program.

Blocks are split and merged based on a feedback of the training error.
Coordinate encoder $\Phi$

$\mathbf{x}_g^i$

$S$
Coordinate encoder $\Phi$

Feature grid $\Gamma$

$N_1 \times N_2 \times N_3 \times C$

$N_1$

$N_2$

$N_3$

$C$
Feature grid $\Gamma$

Interpolated feature vector

Linear interpolation at $x^i_l$
Interpolated feature vector

Feature decoder $\Psi$

Reconstructed signal

e.g. occupancy at $X_l^i$
ACORN

Coordinate Encoder

Feature grid

Interpolated feature vector

Feature decoder
Learned decomposition

ACORN

Block decomposition

Predicted

Loss

Ground truth

W_b

updated online

optimized online

Signal to represent
Coarser grid for low level of details

Finer grid for high level of details
1. should \( \square \) split, stay as is, or merge?

\[
(I_b^\perp, I_b^\equiv, I_b^\uparrow) \in \{0, 1\}^3
\]

\[
I_b^\perp + I_b^\equiv + \frac{1}{8}(I_b^\uparrow + \sum_{b' \in S(b)} I_{b'}^\uparrow) = 1
\]
2. error to split, stay as is, merge

\[ \mathbf{w}_b = (w_b^\downarrow, w_b^\equiv, w_b^\uparrow)^\top \]

with

\[ \mathbf{I}_b = (I_b^\downarrow, I_b^\equiv, I_b^\uparrow)^\top \]

What is the best partition?

minimize \[ \sum_b \mathbf{w}_b^\top \mathbf{I}_b \]
3. Limit the total number of blocks in the partition

Assumption: the network has a given capacity, thus it can only fit a given number of blocks

\[ \sum_b \frac{1}{8} I_b^+ + I_b^- + 8 I_b^+ \leq \text{Max. Num. Blocks} \]
Learned decomposition

ACORN

Block decomposition

Integer Linear Program

Predicted

Loss

Ground truth

Optimized online

Updated online

Signal to represent
Learned decomposition

Block decomposition
Integer Linear Program

ACORN

Signal to represent

minimize \( \sum_b w_b^T I_b \), with \( I_b = (I_b^+, I_b^−, I_b^0)^T \) and \( w_b = (w_b^+, w_b^−, w_b^0)^T \)

subject to:
- a maximum number of blocks
  \( \sum_b \frac{1}{8} I_b^+ + I_b^− + 8 I_b^0 \leq \text{Max. Num. Blocks} \)
- each block can either be split or merged, and the partition remains valid
  \( I_b^+ + I_b^− + \frac{1}{8} (I_b^+ + \sum_{b' \in S(b)} I_{b'}^+) = 1 \)
Results
3D Shapes

Lucy Statue: Stanford 3D Scanning Repository
Adaptive Block Decomposition

Block Decomposition Training Progression

Final ACORN Reconstruction
Training Time

Thai Statue Training

Engine Training
Recent followup: Instant NGP

Real-time Gigapixel fitting!  [Müller et al. '22]
Recent followup: Instant NGP

...and NeRF fitting!

[Müller et al. ‘22]
Outline

• Overview of neural fields
• Representing 3D shape
• Neural rendering with NeRF
• Case studies
  • SIREN
  • ACORN
  • AutoInt
  • BACON
\[ \Phi(x) = \int \Psi(x) \, dx \]
Mildenhall et al. ‘20
Liu et al. ‘20
Schwarz et al. ‘20
...

Training Set → Learned Volume → Novel Views (from VRE)
DIFFERENTIATION

START

TRY APPLYING
CHAIN POWER RULE QUOTIENT PRODUCT ETC

DONE?

NO

YES

DONE!
Numerical Integration Techniques

Riemann Sums

Quadrature

Monte Carlo
Automatic integration
Integral Network
AutoInt steps

1. Specify integral network: $\Phi_\theta(x)$
2. Instantiate grad network: $\Psi_\theta^i(x)$
3. Train grad network:
   $\mathcal{L}(\Psi_\theta^i(x), f(x))$
4. Reassemble into the integral network: $\theta^*$
AutoInt steps

1. Specify integral network
   \[ \Phi_{\theta}(x) \]

2. Instantiate grad network
   \[ \Psi^i_{\theta}(x) \]

3. Train grad network
   \[ \mathcal{L}(\Psi^i_{\theta}(x), f(x)) \]

4. Reassemble into the integral network

\[ \theta^* \]
$f(x)$

Grad Network

$\Psi(x)$
\[ \Phi(x) = \int_0^x \Psi(x) \, dx \]

\[ \int_a^b \Psi(x) \, dx = \Phi(b) - \Phi(a) \]
Training loss

\[ \int_{a}^{b} \Psi(x) \, dx \]

\[ \approx \frac{b - a}{N} \sum_{i}^{N} \Psi(x_i) \]

Monte Carlo approximation
Integrating 2D Signals

ReLU

Softplus

SIREN (sine)

Swish

subsample x4
Integrating 2D Signals

ReLU
Softplus
SIREN (sine)
Swish

subsample x4

subsample x8
Integrating 2D Signals

ReLU

Softplus

SIREN (sine)

Swish

subsample x4

subsample x16
Volume Rendering Equation (VRE)

\[ C(r) = \int_{t_n}^{t_f} \sigma(r(t)) e \left\{ - \int_{t_n}^{t} \sigma(r(s)) \, ds \right\} c(r(t)) \, dt \]

- color of rendered ray
- absorption coefficient
- transmittance
- emissive radiance
Volume Rendering Equation (VRE)

\[ C(r) = \int_{t_n}^{t_f} \sigma(r(t)) e^{-\int_{t_n}^{t} \sigma(r(s)) \, ds} c(r(t)) \, dt \]
Approximation of the VRE

\[ \tilde{C}(r) = \sum_{i=1}^{N} \tilde{c}_i(t) \tilde{T}_i(t) \tilde{\sigma}_i(t) \delta_i \]

- color of rendered ray
- emissive radiance
- transmittance
- absorption coefficient

with \( \delta_i = t_i - t_{i-1} \)
Approximation of the VRE

\[ \bar{T}_i = \exp \left( - \sum_{j=1}^{i-1} \bar{\sigma}_j \delta_j \right) \]

average transmittance

\[ \tilde{C}(r) = \sum_{i=1}^{N} \bar{c}_i(t) \bar{T}_i(t) \bar{\sigma}_i(t) \delta_i \]

\[ \bar{\sigma}_i = \delta_i^{-1} \int_{t_{i-1}}^{t_i} \sigma(t) \, dt \]

average absorption value

\[ \bar{c}_i = \delta_i^{-1} \int_{t_{i-1}}^{t_i} c(t) \, dt \]

average radiance value
NeRF
[Mildenhall '20]
30 s/frame

Neural Volumes
[Lombardi '19]
0.3 s/frame

AutoInt
8 Sections
2.6 s/frame

AutoInt
32 Sections
9.3 s/frame
Open questions

1. Expressiveness of grad network architectures deriving from standard MLPs

...network is a tree that contains both \( \mathbf{NL} \) and \( \mathbf{NL}' \)

2. Other pairs of (grad, integral) networks...

...that might be easier to train and more expressive

3. Training the integral network directly...

...for instance via the bounds:

\[ \Phi(x) \quad \Psi^i(x) \]

seems to work when \( a \) and \( b \) are sampled densely
Outline

• Overview of neural fields
• Representing 3D shape
• Neural rendering with NeRF
• Case studies
  • SIREN
  • ACORN
  • AutoInt
  • BACON
coordinates
\( \in \mathbb{R}^2, \mathbb{R}^3, \ldots \)
Coordinate Network

coordinates $\in \mathbb{R}^2, \mathbb{R}^3, \ldots$

pixel values

signed distance functions

neural radiance fields
coordinates $\in \mathbb{R}^2, \mathbb{R}^3, \ldots$ → coordinate network → array → voxel grid
coordinate network \rightarrow \text{supervised coordinates} \quad \text{unsupervised coordinates} \quad \text{downsampled coordinates (aliased)}

coordinates \in \mathbb{R}^2, \mathbb{R}^3, \ldots
Band-Limited Coordinate Network (BACON)

coordinates

\( \mathbb{R}^2, \mathbb{R}^3, \ldots \)

supervised coordinates

anti-aliased downsampling

network spectrum
Band-Limited Coordinate Network (BACON)

coordinates $\in \mathbb{R}^2, \mathbb{R}^3, \ldots$

SDF Supervision

supervised scale

anti-aliased downsampling

network spectra
Band-Limited Coordinate Network (BACON)

coordinates \( \in \mathbb{R}^2, \mathbb{R}^3, \ldots \)

supervised scale

anti-aliased downsampling

multiview image supervision (neural radiance field)

network spectra
Sinusoidal Rep. Networks (SIREN) [Sitzmann et al. ‘20]

• single scale

Fourier Features [Tancik et al. ‘20]

• single scale
Sinusoidal Rep. Networks (SIREN) [Sitzmann et al. '20]

- single scale

Fourier Features [Tancik et al. '20]

- single scale

Mip-NeRF [Barron et al. '21]

- multiscale
- multiscale supervision
Neural Geometric LOD [Takikawa et al. ‘21]

Adaptive Coordinate Networks [Martel et al. ‘21]

PlenOctrees [Yu et al. ‘21]

• single scale

• multiresolution outputs not bandlimited

• black box behavior
Neural Geometric LOD (proposed) [Takikawa et al. '21]
Adaptive Coordinate Networks [Martel et al. '21]
PlenOctrees [Yu et al. '21]

- black box behavior
- multiresolution outputs not bandlimited
Neural Geometric LOD [Takikawa et al. '21]

Adaptive Coordinate Networks [Martel et al. '21]

PlenOctrees [Yu et al. '21]

Sinusoidal Rep. Networks (SIREN) [Sitzmann et al. '20]

Fourier Features [Tancik et al. '20]

Mip-NeRF [Barron et al. '21]

BACON (proposed)

- multiscale
- single-scale supervision

Multiplicative Filter Networks [Fathony et al. '21]

- black box behavior
- multiresolution outputs not bandlimited
Sinusoidal Rep. Networks (SIREN) [Sitzmann et al. '20]

Fourier Features [Tancik et al. '20]

Mip-NeRF [Barron et al. '21]

BACON (proposed)

• multiscale
• multiscale supervision

• multiscale
• single-scale supervision

Neural Geometric LOD [Takikawa et al. ‘21]

Adaptive Coordinate Networks [Martel et al. ‘21]

PlenOctrees [Yu et al. ‘21]

Multiplicative Filter Networks [Fathony et al. ‘21]

• analytical Fourier spectra
• adjustable bandwidth
• initialization for deep networks

• black box behavior
• multiresolution outputs not bandlimited
Target Signal

Target Spectrum (magnitude)

supervised points
BACON (low-pass)

Target Signal

Target Spectrum (magnitude)

model output

BACON cutoff frequency

Target Signal

SIREN

Target Spectrum (magnitude)

spurious frequencies

Fourier Features

Target Spectrum (magnitude)
Architecture
\[ \sin(\omega_0 x + \phi_0) \]
# Parameterized Sines

\[ N_{\text{sinc}}^{(N_L)} = \sum_{i=0}^{N_L-1} 2^i d_{l_i}^{i+1} \]
Results
Fourier Spectrum
4x Downsampling

Low-pass Reference

SIREN

Fourier Features

$F$

Mip-NeRF PE

BACON
4x Upsampling

High-res Reference

SIREN

Fourier Features

Mip-NeRF PE

BACON
Ground Truth
[Mildenhall et al. ‘20]

NeRF
[Barron et al. ‘21]

Mip-NeRF
[Barron et al. ‘21]

BACON
Neural Fields

• Exciting and rapidly evolving research area!
• Many hard problems being solved, but still more work to be done
  • Robust generalization
  • Compositionality
  • Compact, efficient, & scalable 3D reconstruction
• How to integrate with computational imaging problems?
Guest lecture (on Zoom!)

Mark Sheinin (CMU)

Next time...

Imaging, Fast and Slow: Computational Imaging for Sensing High-speed Phenomena

(two-time CVPR Best Paper Award winner!)