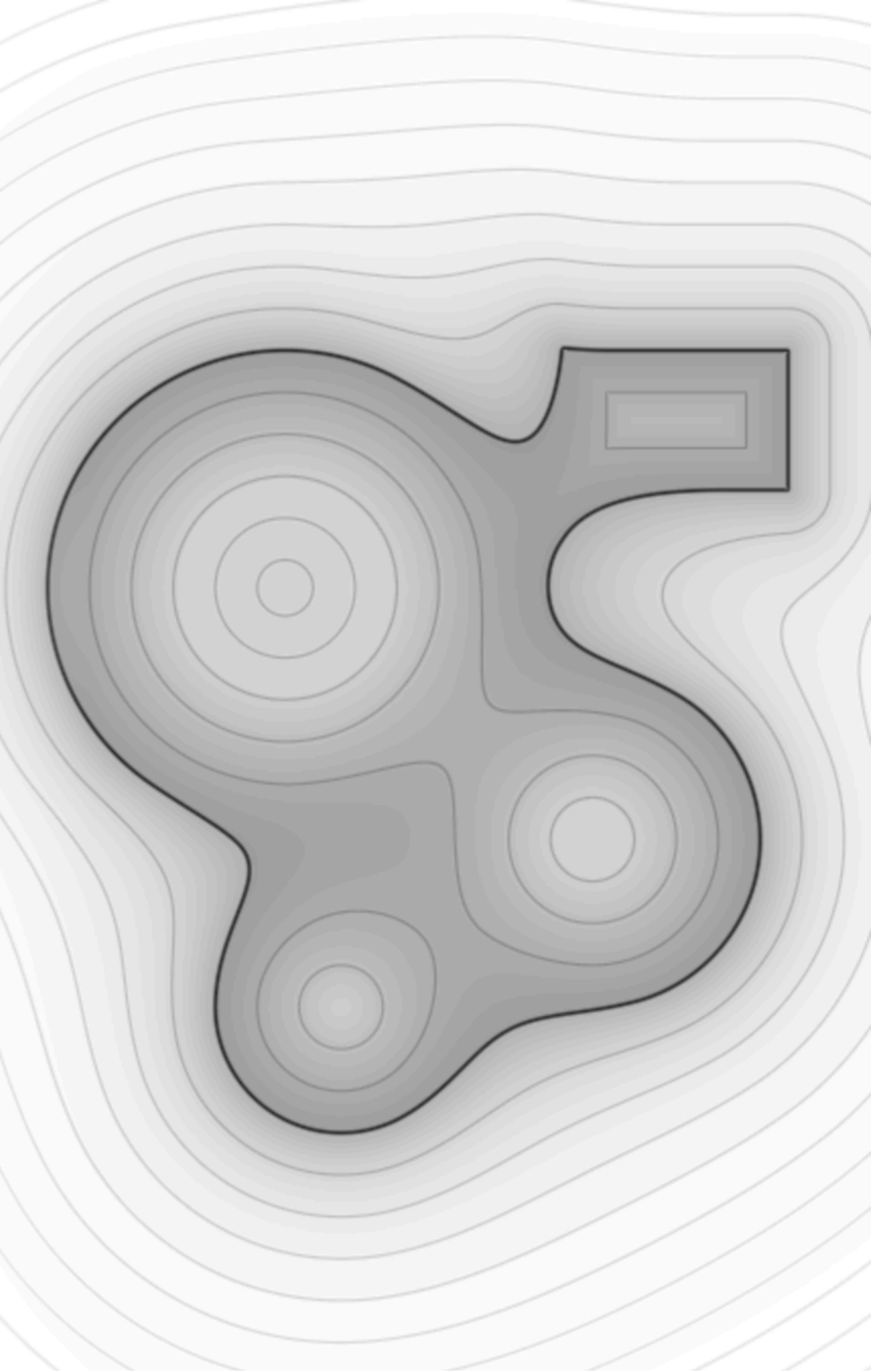
Rethinking Computer Graphics, Continuously

for images
for surfaces
for light
for motion

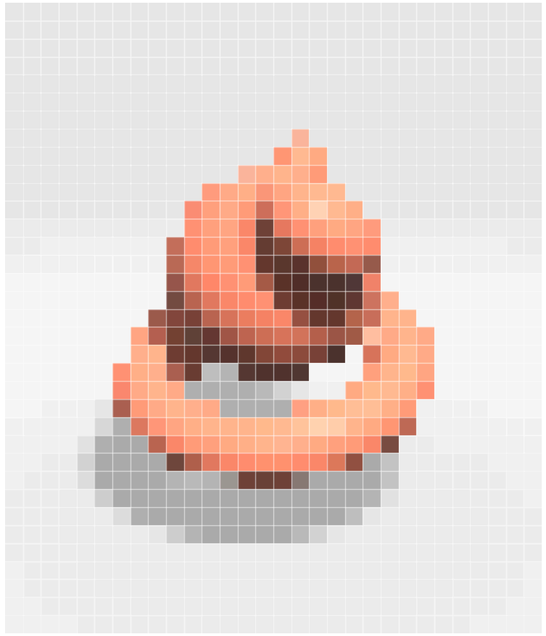


Zhecheng Wang

CSC317 Guest Lecture · University of Toronto · November 10, 2026

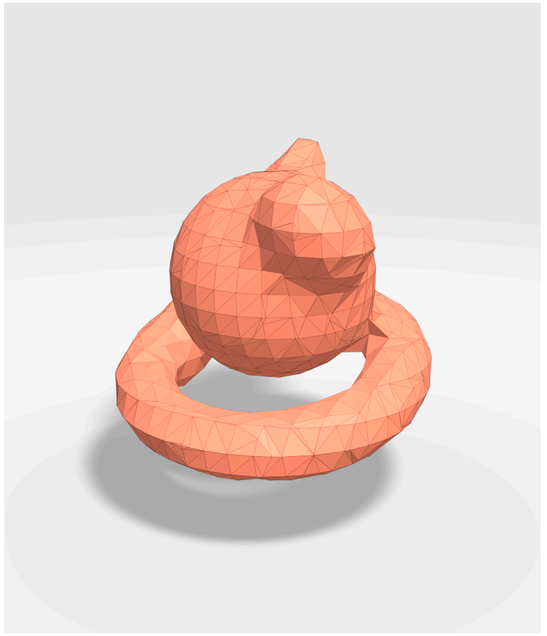
dgp dynamic graphics project

Discrete by Default



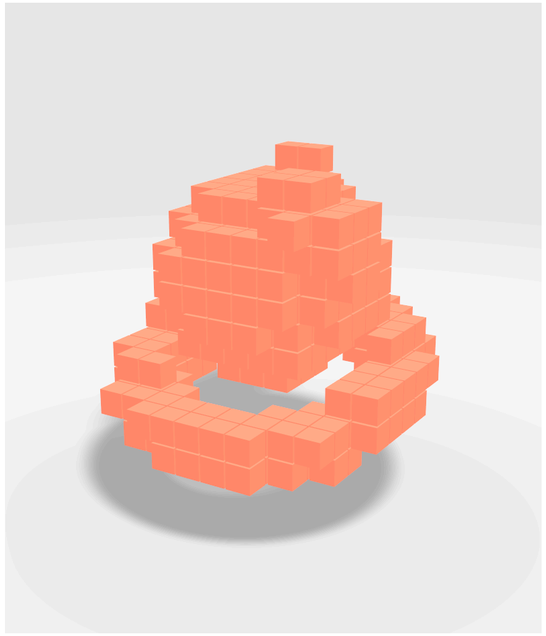
Pixels

A1 · an array of samples



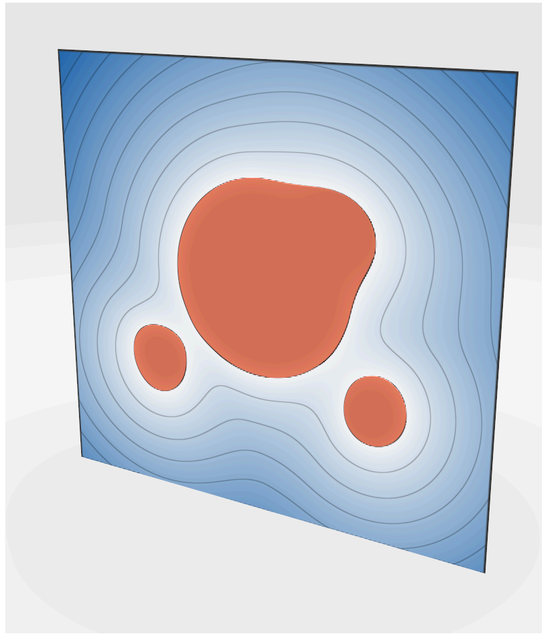
Triangles

A4 · vertices and faces



Voxels

a grid of cells



A field $f(\mathbf{x})$

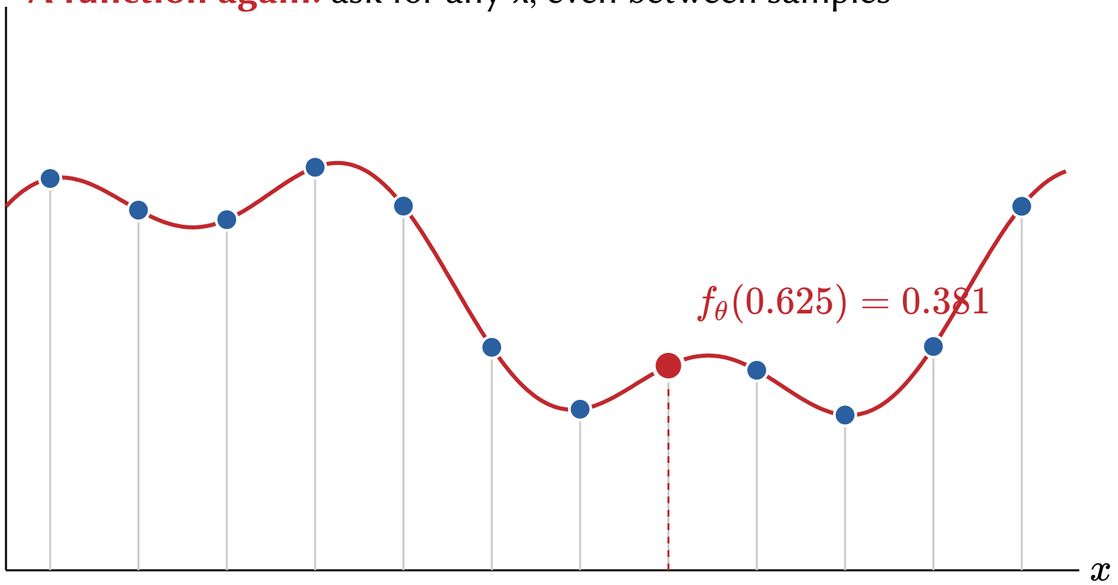
a function, queried anywhere

Images are arrays. Geometry is vertices and faces. Rendering traverses triangles. Animation moves vertices.

Properties of the scene, or choices of the implementation?

Continuous World, Discrete Computer

A function again: ask for any *x*, even between samples



I = [0.73, 0.67, 0.65, 0.75, 0.68, 0.42, 0.30, 0.38, 0.37, 0.29, 0.42, 0.68]

① World | ② Computer | ③ Function

In the world, signals are continuous

Light on a sensor, the shape of a surface, a motion over time: there is a value at every x.

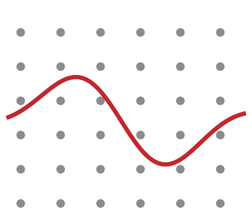
In the computer, we discretize

We keep samples: pixels, vertices, voxels, time steps. Between them, nothing is stored.

Today: store a function instead

$f_\theta(x)$ answers at any x. Computing with it is still discrete: we evaluate it at finitely many points.

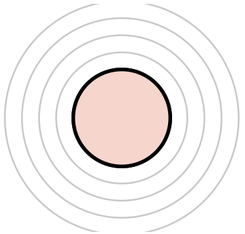
Roadmap



2D image field

$f_\theta : \mathbb{R}^2 \to \mathbb{R}^3$

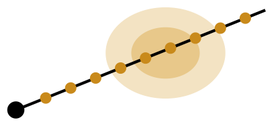
A1 · Raster Images



3D geometric field

$f_\theta : \mathbb{R}^3 \to \mathbb{R}$

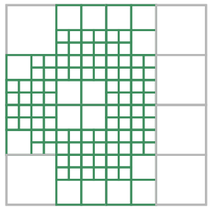
A4 · Meshes and Shaders



Differentiable rendering

$f_\theta(\mathbf{x}, \mathbf{d}) = (\sigma, \mathbf{c})$

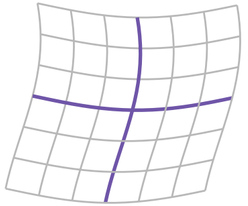
A2 · Ray Tracing



Accelerated evaluation

$\text{grids} + \text{hashing}$

A3 · BVH



Deformation fields

$\varphi(\mathbf{x}, t)$

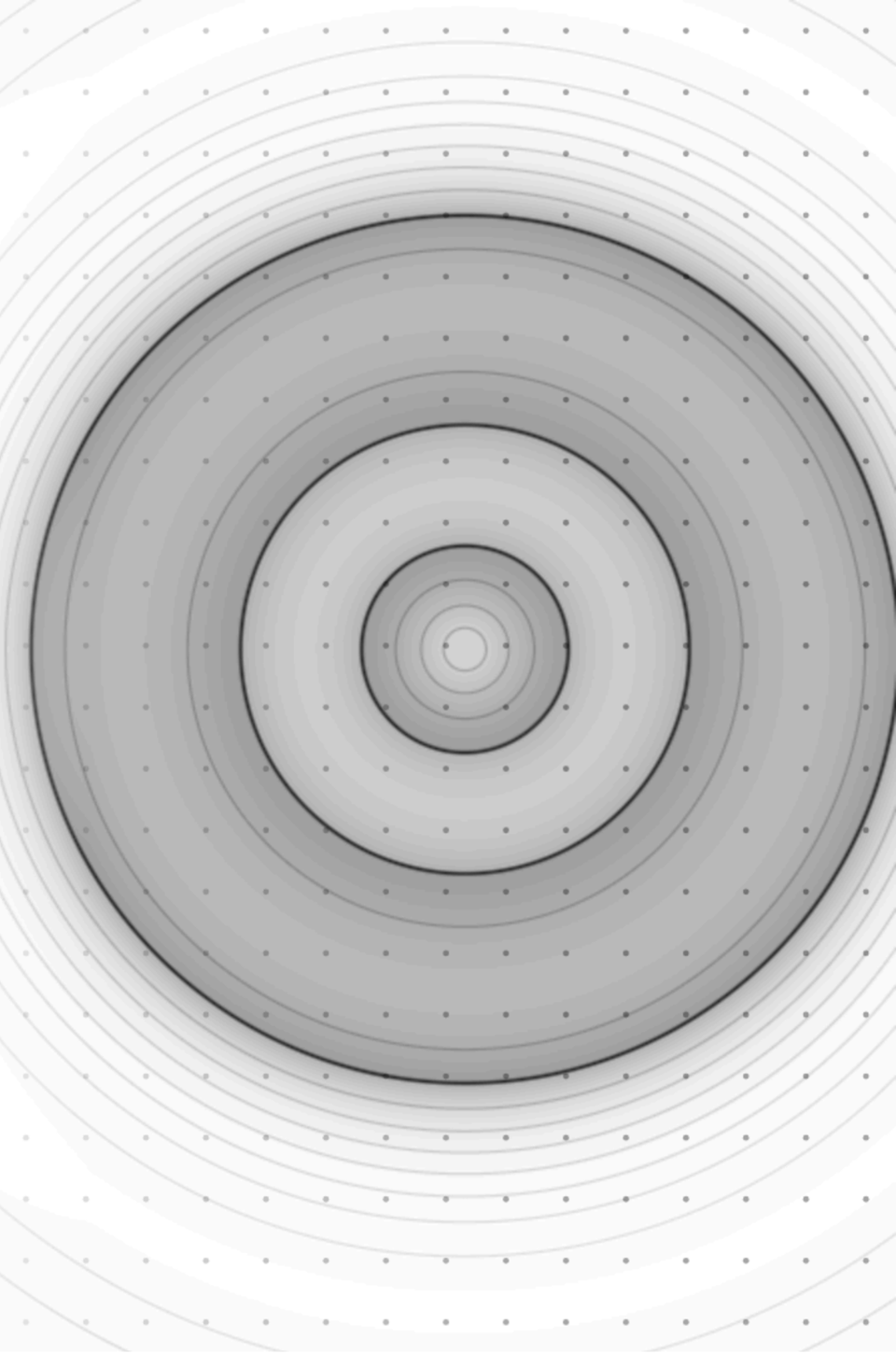
A5 · Kinematics (preview)

One argument in five steps: each step reuses the previous field and adds one idea.

Part 1 · A1 Raster Images

Images as Functions

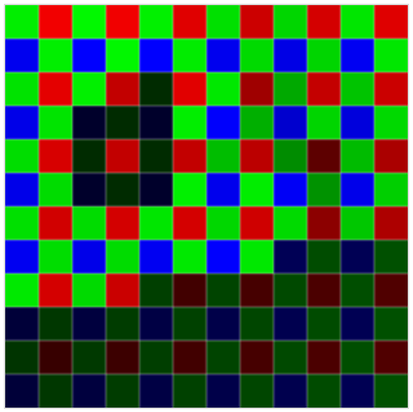
$$I[i, j] \in \mathbb{R}^3 \quad \longrightarrow \quad f_\theta : \mathbb{R}^2 \to \mathbb{R}^3$$



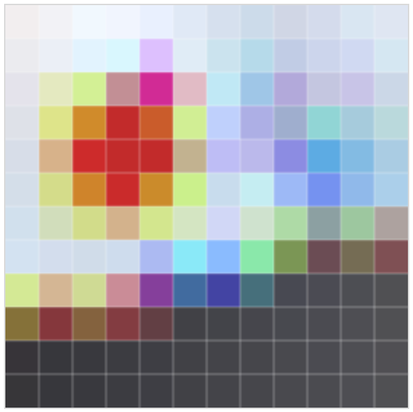
An Array, or Samples of a Function?



Light on the sensor
$I(x, y)$, defined everywhere



Bayer mosaic
one colour per photosite



Demosaiced
A1: average the neighbours

Photosites per side — 12

☑ Auto

What A1 stored

$I[i, j] \in \mathbb{R}^3$, an $H \times W \times 3$ array of numbers.

What the sensor saw

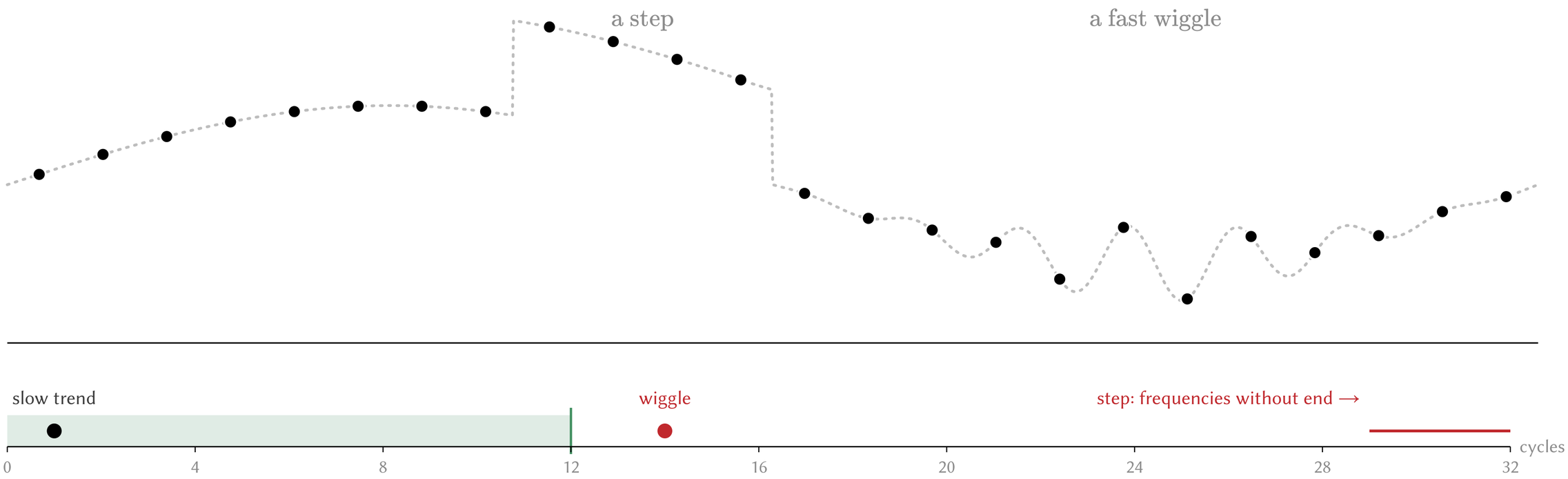
Light on a continuous plane, $I(x, y)$, measured at one colour per photosite.

What demosaicing did

Guessed the missing values between samples: averaging neighbours is a reconstruction rule.

Lecture 1 already said it: an image is a function defined for an array of pixels.

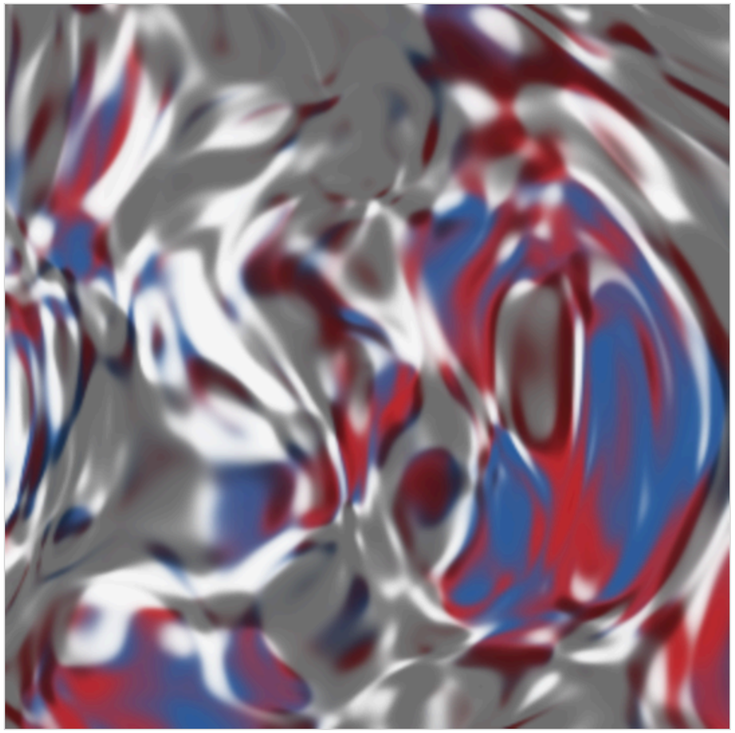
Reconstruction and the Nyquist Limit



24 samples per unit ⟶ Nyquist limit $N/2 = 12$ cycles. The wiggle (14 cycles) is **above the limit: lost, or aliased into a slower wave**. The step is never fully recoverable.

Samples 24 ☐ Nearest ☐ Linear ☐ Catmull–Rom ☐ Ideal (sinc) ↺ Reset

Coordinates In, Colour Out



hover the image: one query

zoom **×1.0**
this view: 300 × 300 = **90,000** queries

New pattern

Reset

click or scroll to zoom

$$f_\theta(x, y) = (R, G, B)$$

One coordinate in, one colour out. Hover the image: each point is one query.

40,000
queries for a 200 × 200 image

1
query per pixel, per frame

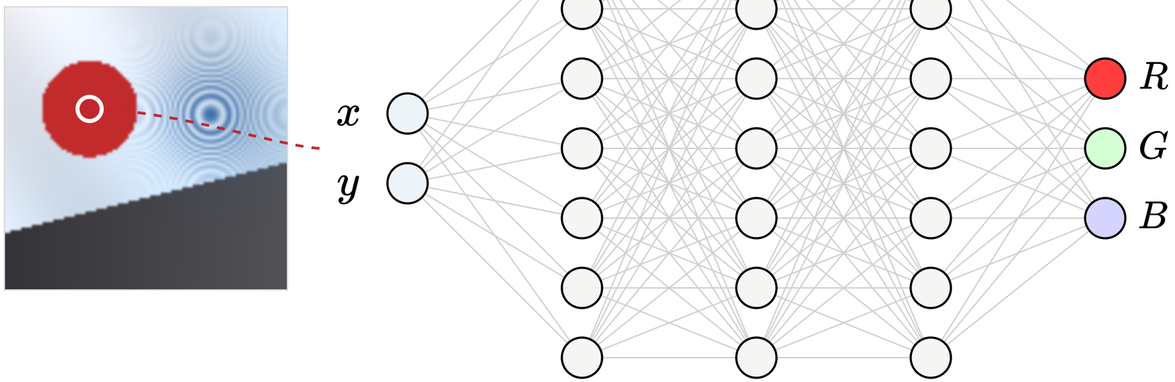
Array: $I[12,\ 40]$ — an index, integers only

Function: $f_\theta(12.5,\ 40.25)$ — a coordinate

So we can ask what lies **between** two pixels of the training image, and a good network gives a sensible estimate.

Stanley 2007. Compositional Pattern Producing Networks: A Novel Abstraction of Development. Genetic Programming and Evolvable Machines 2007.
Mordvintsev et al. 2018. Differentiable Image Parameterizations. Distill 2018.

Fitting an Image with a Network



θ : weights and biases

$$f_\theta(x, y) = (R, G, B)$$

a multilayer perceptron of the coordinate

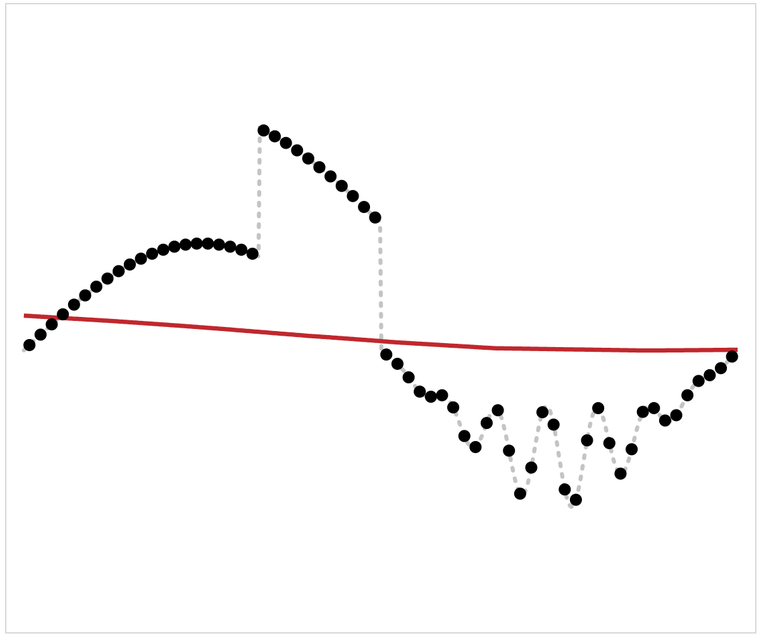
$$L(\theta) = \sum_i \left\| f_\theta(\mathbf{x}_i) - I_i \right\|^2$$

regression on the pixel samples $(\mathbf{x}_i, I_i)$

What does f_θ return between the pixels?

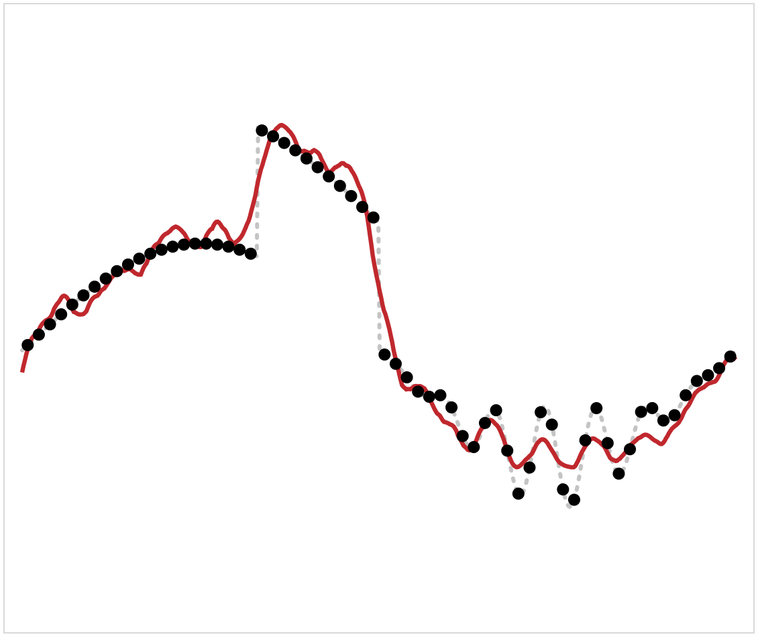
Spectral Bias

ReLU MLP $\mathrm{MLP}(x)$



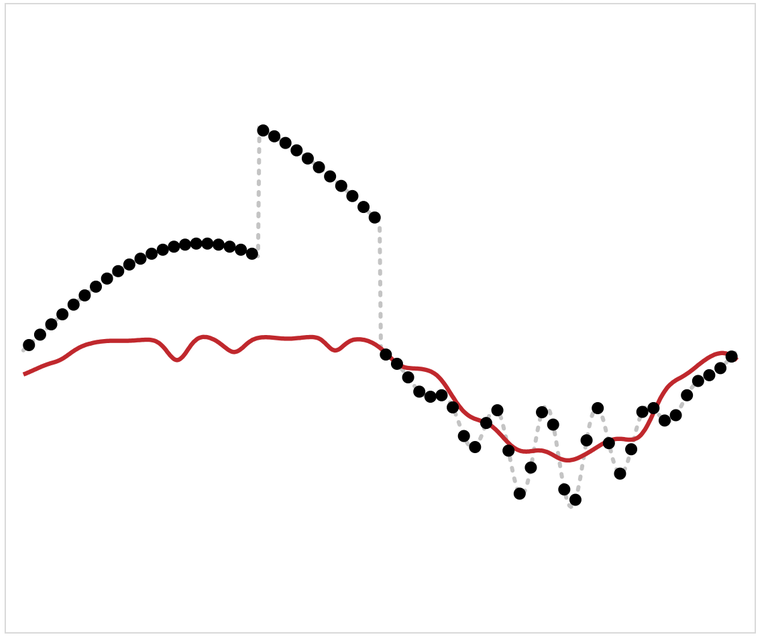
error between samples (MSE on 512 points): **0.0505**

Fourier features $\mathrm{MLP}(\gamma(x))$



error between samples (MSE on 512 points): **0.0018**

SIREN $\sin(\omega_0 W x + b)$



error between samples (MSE on 512 points): **0.0404**

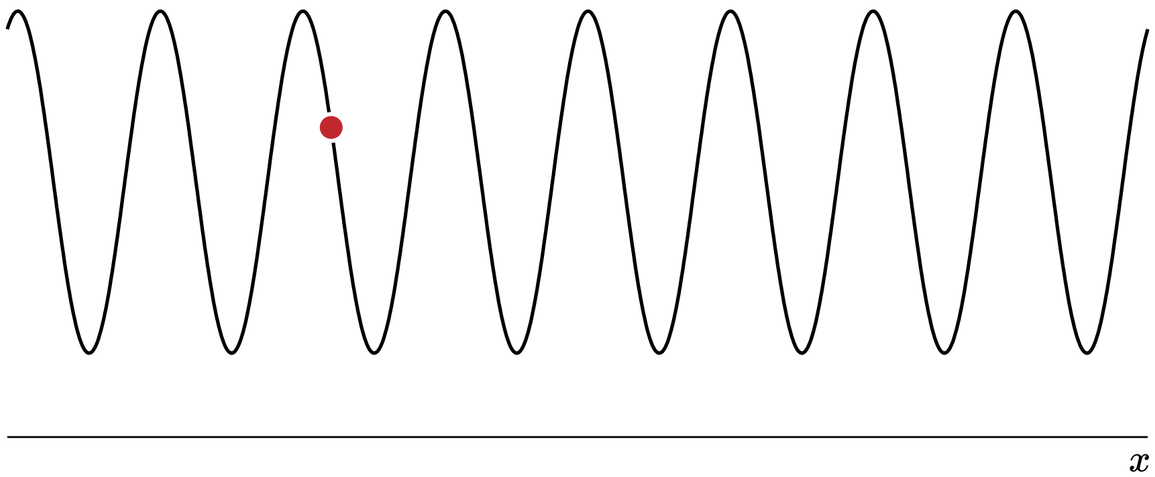
after **50** steps of gradient descent

Training step 50 ☑ Auto ☐ Fourier features: high σ ↺ Reset

Rahaman et al. 2019. On the Spectral Bias of Neural Networks. ICML 2019.
Tancik et al. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. NeurIPS 2020.
Sitzmann et al. 2020. Implicit Neural Representations with Periodic Activation Functions. NeurIPS 2020.

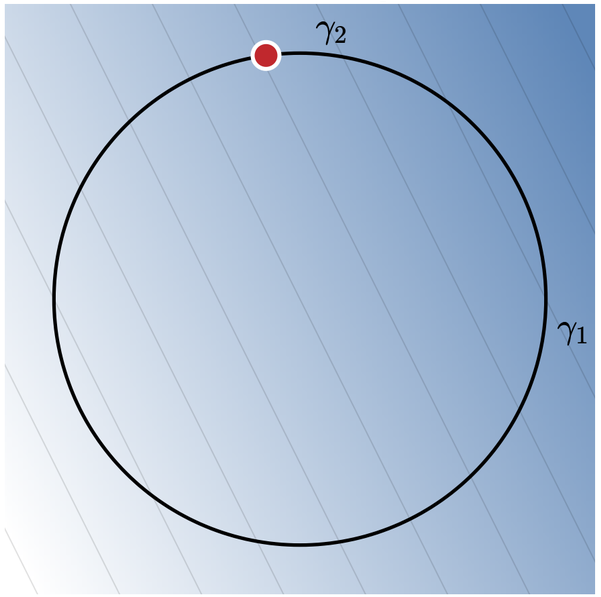
Lift, Then Fit

In x: a fast wiggle



$$\gamma(x) \longrightarrow (\cos 2\pi b x,\ \sin 2\pi b x)$$

Lifted: a tilted plane



$b = 8$: 8 wiggles in x are 8 laps around the same circle. On the circle the signal is $0.5 + 0.3\,\gamma_1 + 0.15\,\gamma_2$: one linear layer fits it.

Frequency b 8 Position x 0.284 Move

Fourier Features

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

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

NeurIPS · 2020

Mapping coordinates through random sinusoids lets a plain MLP fit high-frequency signals; the scale of the mapping sets the bandwidth.

$$\gamma(\mathbf{x}) = \Big[\sin(2\pi B\mathbf{x}),\ \cos(2\pi B\mathbf{x})\Big]$$

many circles at once: each row of B is one frequency, drawn from $\mathcal{N}(0, \sigma^2)$ and fixed

$$f_\theta(\mathbf{x}) = \mathrm{MLP}_\theta\big(\gamma(\mathbf{x})\big)$$

You choose the lifting: σ sets which frequencies are easy. Too small underfits; too large ripples between samples.

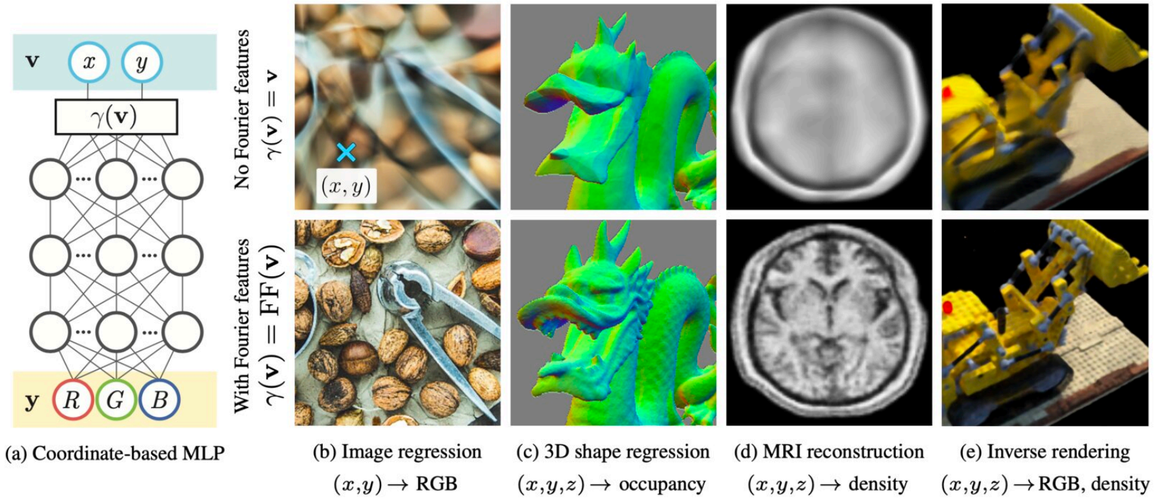


Figure: Tancik et al. 2020

Without (left) and with (right) Fourier features, during training. Tancik et al. 2020

Tancik et al. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. NeurIPS 2020.

SIREN · The Network Lifts Itself

Implicit Neural Representations with Periodic Activation Functions

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

NeurIPS · 2020

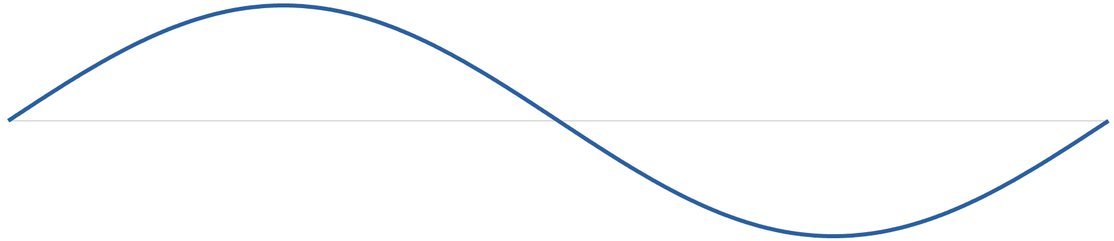
Fourier features: $x \to \underbrace{\gamma(x)}_{\text{chosen by you}} \to \text{ReLU MLP}$

SIREN: $x \to \underbrace{\sin(\omega_0(W_1 x + b_1))}_{\text{learned}} \to \sin(\cdots) \to \cdots$

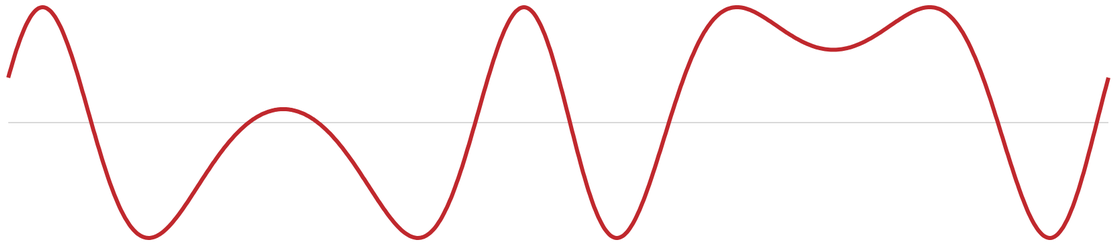
Same idea, no hand-designed lifting: every layer is a sine, its frequencies are weights, and stacking sines makes new, higher ones.

Bonus: the derivative of a sine is a sine, so gradients (normals in A4) stay smooth.

layer 1: $h_1 = \sin(2\pi\, x)$



layer 2: $h_2 = \sin(a\, h_1 + 0.4)$



$a = 6.0$: layer 2 oscillates **4** times per unit, from a layer 1 that oscillates once.

Layer-2 weight a 6.0

Sitzmann et al. 2020. Implicit Neural Representations with Periodic Activation Functions. NeurIPS 2020.

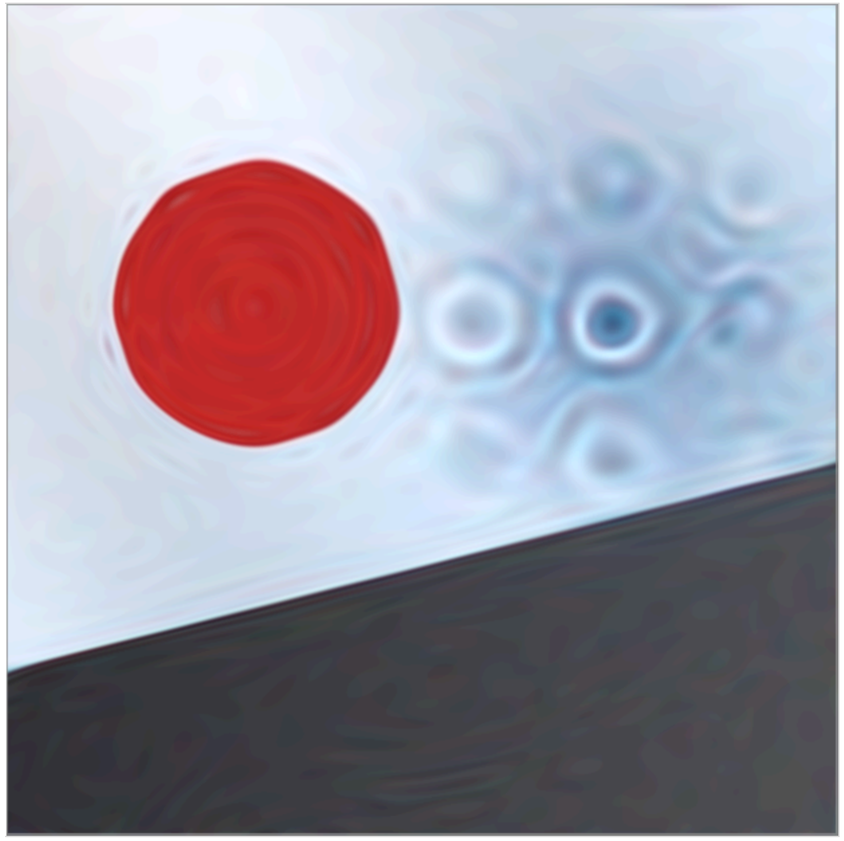
Neural Image Explorer

Truth | Pixels | Bilinear | ReLU | Fourier | SIREN



bilinear · 12,288 stored values
PSNR **25.1 dB** on a dense 256² grid

Truth | Pixels | Bilinear | ReLU | Fourier | SIREN



SIREN, $\omega_0 = 30$ · 4,995 weights · 3,000 steps
PSNR **26.6 dB** dense · 30.0 dB at the samples

Samples 32² 64²

☐ Pixel grid and samples

☐ Gradient magnitude $\|\nabla I\|$

Zoom: disc edge | Zoom: rings

↺ Reset

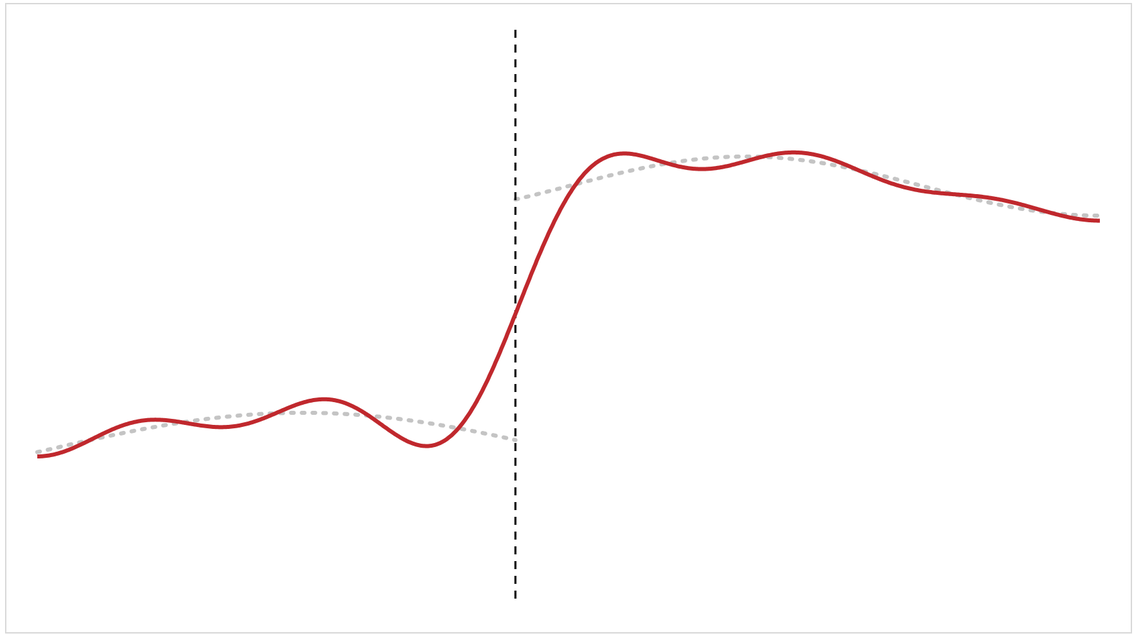
zoom **×1.0** · one sample = 10 px

Every method sees only the point samples of the image. The networks were fitted to them by gradient descent (3,000 Adam steps); each is 3 hidden layers of 48 units.

Scroll to zoom, drag to pan; both views follow.

More Frequencies Don't Fix an Edge

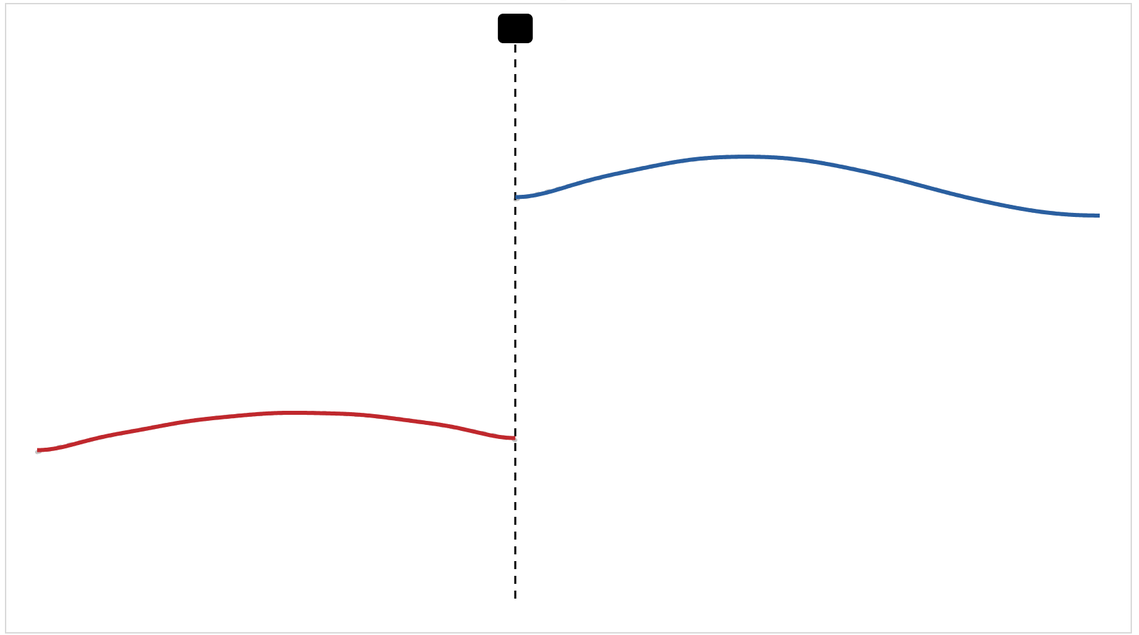
One continuous field



12 frequencies: overshoot at the edge **8.3%** of the jump.
More frequencies make the ringing narrower, not smaller.

Frequencies K 12

Two fields, a prescribed cut



12 frequencies per side: largest error **1.01%** of the jump.
The cut is given, so neither field ever sees an edge.

drag the cut Γ on the right ↺ Reset

If you know where the edge is, cut the domain there and let each side be smooth.

Discontinuity-Aware Neural Fields

Discontinuity-Aware 2D Neural Fields

Yash Belhe, Michaël Gharbi, Matthew Fisher, Iliyan Georgiev, Ravi Ramamoorthi, Tzu-Mao Li

ACM TOG (SIGGRAPH Asia) · 2023

Known discontinuity curves become edges of a curved triangle mesh; features are interpolated without crossing them and decoded by a small MLP.

$$f(\mathbf{x}) = \mathrm{MLP}\left(\sum_v w_v(\mathbf{x})\, \mathbf{z}_v \right)$$

features $\mathbf{z}_v$ on a curved triangulation whose edges follow the given discontinuity curves; interpolation never crosses them

The previous slide in 2D: explicit cuts where the edges are, a smooth neural field everywhere else. Sharp at any zoom.

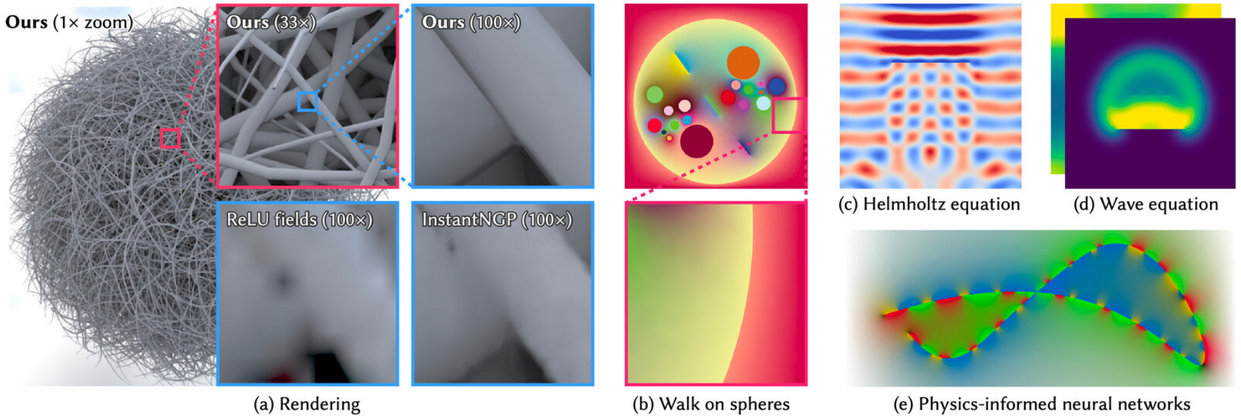


Figure: Belhe et al. 2023

Belhe et al. 2023. Discontinuity-Aware 2D Neural Fields. ACM TOG (SIGGRAPH Asia) 2023.

Part 1 · Takeaway

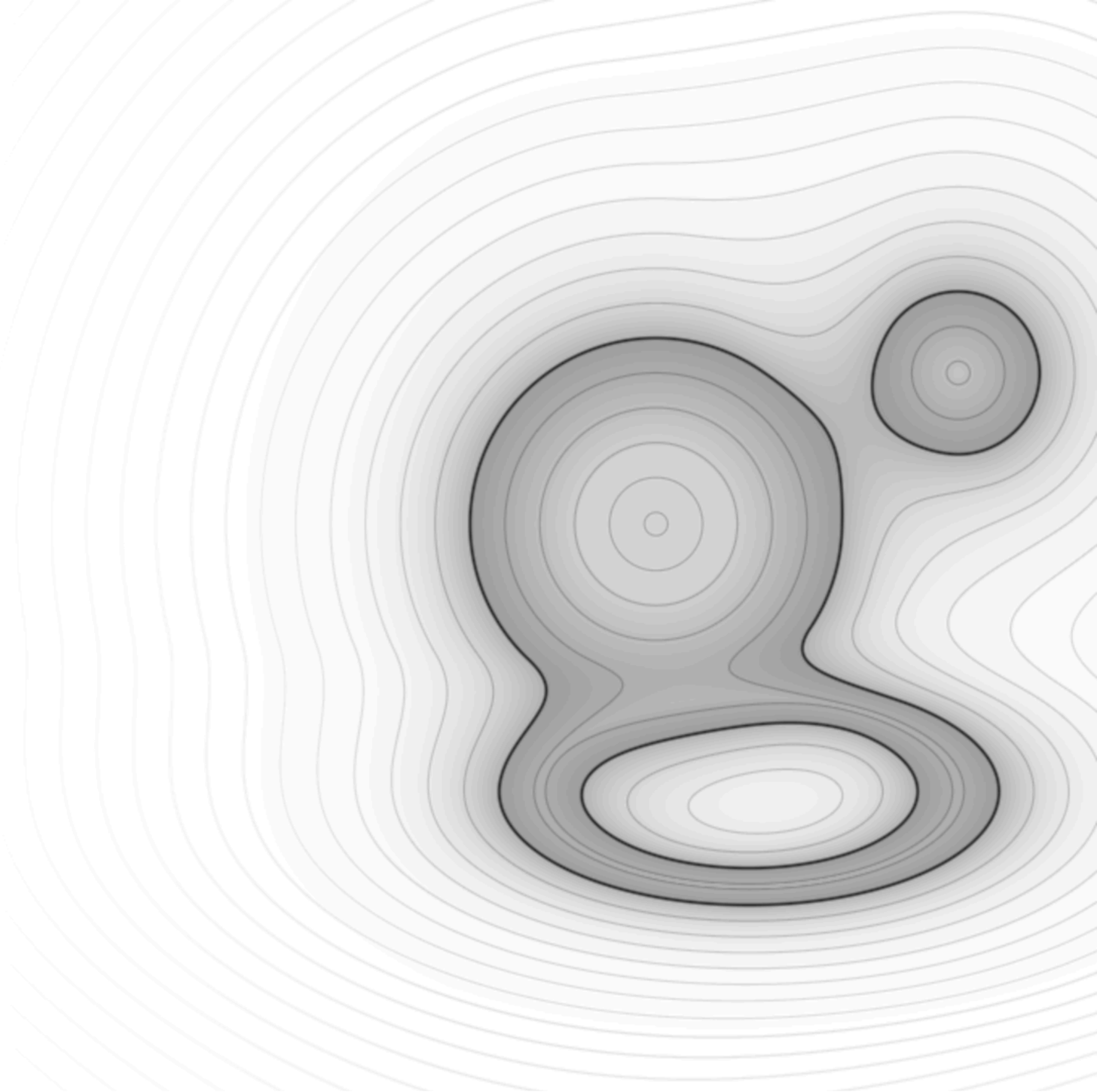
A neural representation is a function we can query anywhere, not a guarantee of infinite detail or correct discontinuities.

Detail between samples comes from a prior: an interpolation kernel, an encoding, an architecture.

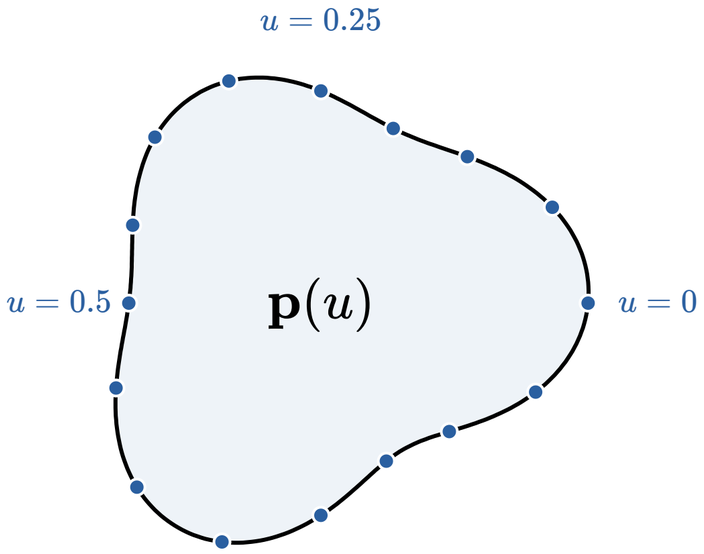
Part 2 · A4 Meshes and Shaders

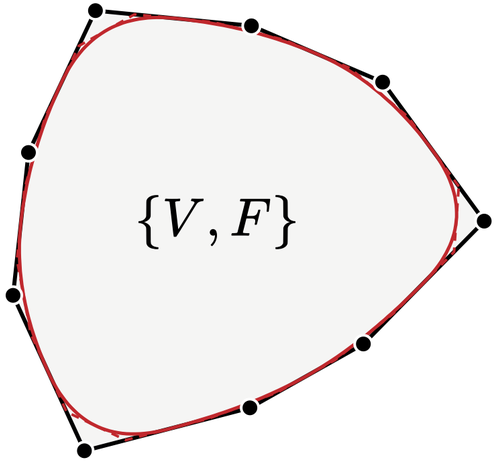
Surfaces as Functions

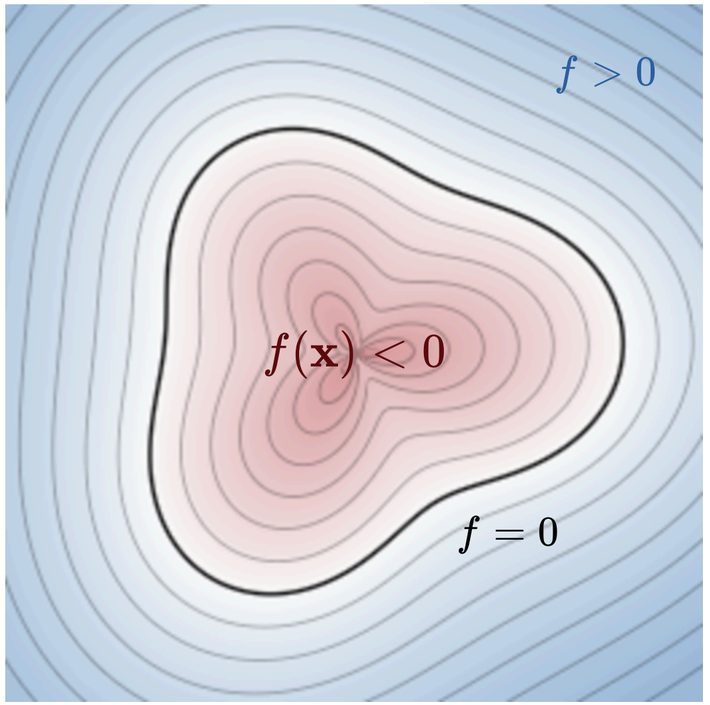
$$f_\theta : \mathbb{R}^3 \to \mathbb{R}, \qquad S = \{\mathbf{x} \mid f_\theta(\mathbf{x}) = 0\}$$



What Is a Surface?







☑ one subdivision step ☑ limit curve

Parametric

$\mathbf{p}(u, v)$: generate points.
Easy to sample, hard to test inside/outside.

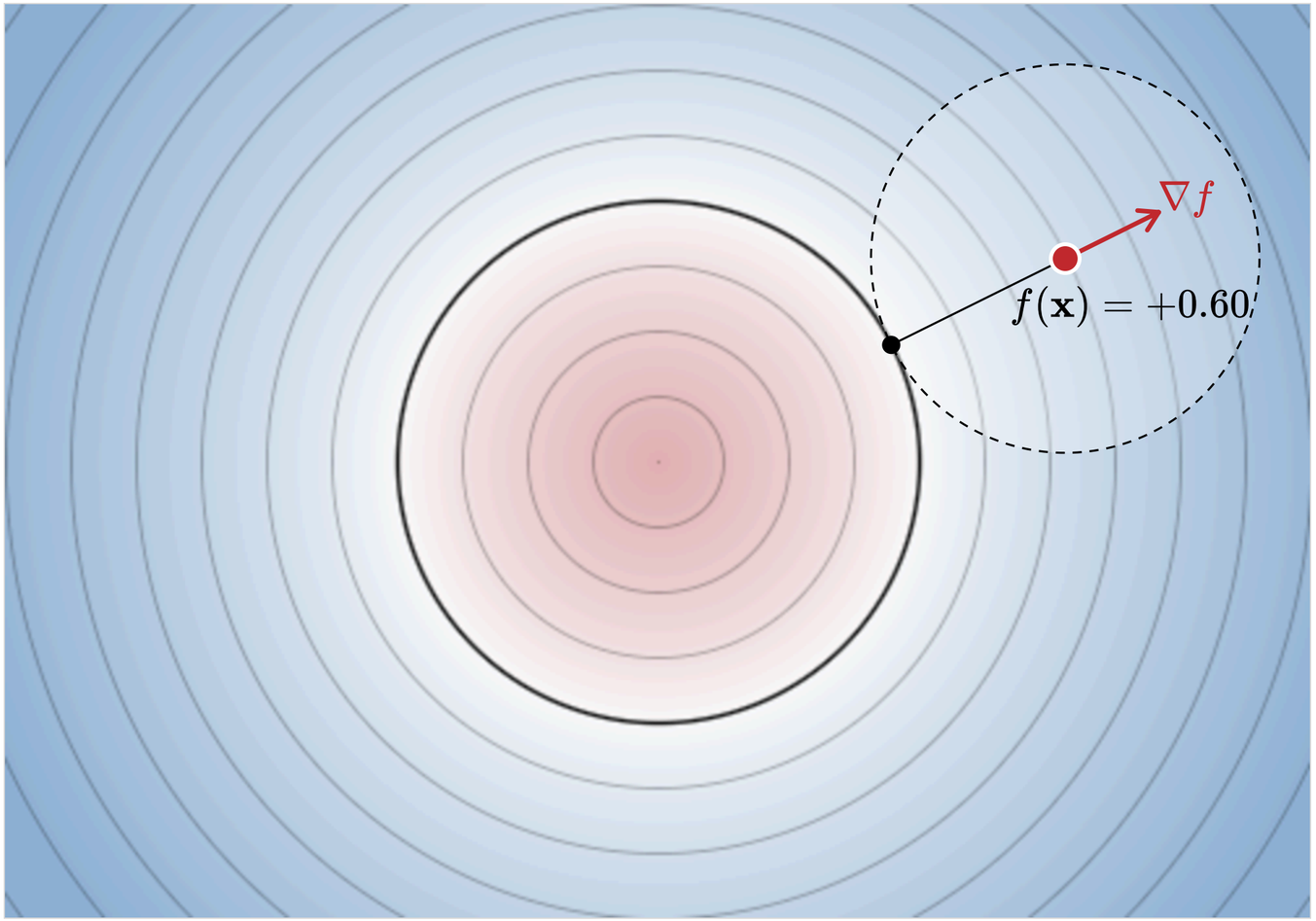
Mesh

Piecewise-linear $\mathbf{p}$ over triangles; continuous, kinked at edges. Catmull–Clark refines toward a smooth limit surface.

Implicit

$S = \{\mathbf{x} \mid f(\mathbf{x}) = 0\}$: test points.
Easy inside/outside, harder to enumerate.

Signed Distance



Shape | Circle | Two circles | Box

outside: f > 0

Reset

$$f(\mathbf{x}) = \pm \min_{\mathbf{y} \in S} \|\mathbf{x} - \mathbf{y}\|$$

negative inside, positive outside

$$\text{circle: } f(\mathbf{x}) = \|\mathbf{x} - \mathbf{c}\| - r$$

$$\mathbf{n}(\mathbf{x}) = \frac{\nabla f(\mathbf{x})}{\|\nabla f(\mathbf{x})\|}$$

and for a true distance, $\|\nabla f\| = 1$ (eikonal)

Drag the point. The value says how far; the sign says which side; the gradient points away from the surface.

DeepSDF

DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation

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

CVPR · 2019

One network, conditioned on a latent code per shape, represents the signed distance fields of a whole category.

$$f_\theta(\mathbf{x}, \mathbf{z}) \approx \mathrm{SDF}_{\mathbf{z}}(\mathbf{x})$$

one network for a category; one latent code $\mathbf{z}$ per shape

Auto-decoder

Codes are optimized jointly with θ; no encoder.

Interpolation

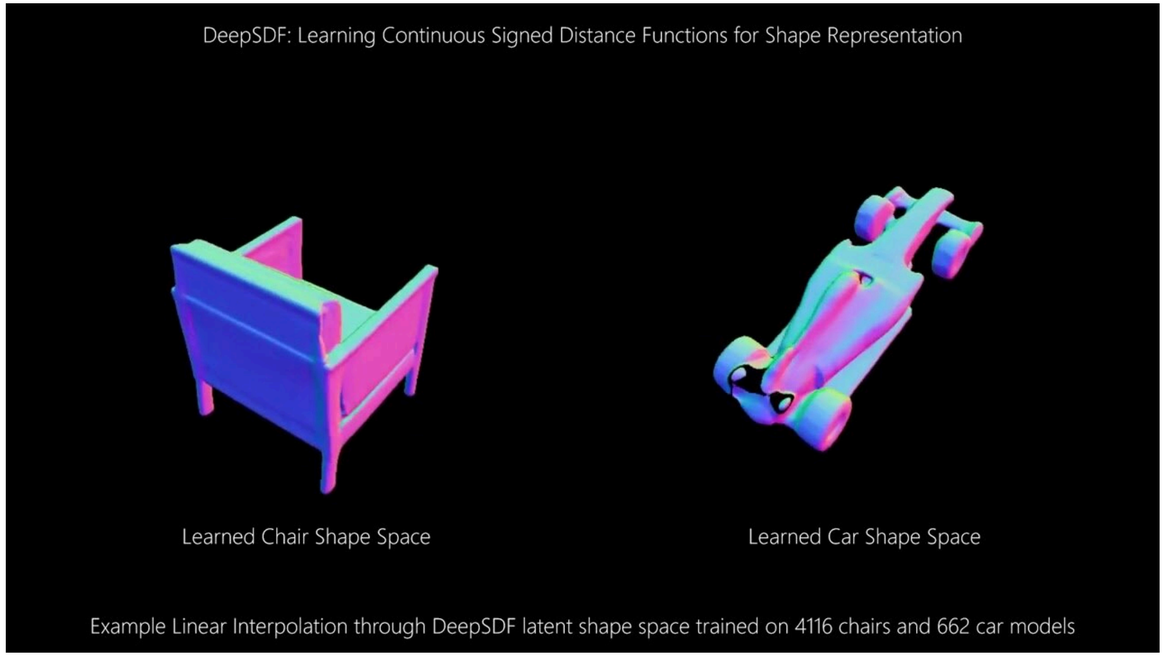
Blend codes: shapes in between.

Completion

Fix θ; fit $\mathbf{z}$ to a partial scan; query everywhere.

Instance vs. category

$f_\theta(\mathbf{x})$ stores one shape; $f_\theta(\mathbf{x}, \mathbf{z})$ stores a space of shapes.



Latent-space interpolation, chairs and cars. Park et al. 2019 (video frame)

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

Occupancy Networks

Occupancy Networks: Learning 3D Reconstruction in Function Space

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

CVPR · 2019

Reconstruct a shape as the decision boundary of a classifier of 3D points: inside or outside.

$$o_\theta(\mathbf{x}) \in [0, 1]$$

occupancy: probability of inside
surface at $o_\theta = 0.5$

$$f_\theta(\mathbf{x}) \in \mathbb{R}$$

signed distance
surface at $f_\theta = 0$

Occupancy needs only inside/outside labels. Distance also says how far, which gives safe ray steps, normals and a well-scaled gradient.



Reconstruction from a single image. Mescheder et al. 2019

Mescheder et al. 2019. Occupancy Networks: Learning 3D Reconstruction in Function Space. CVPR 2019.

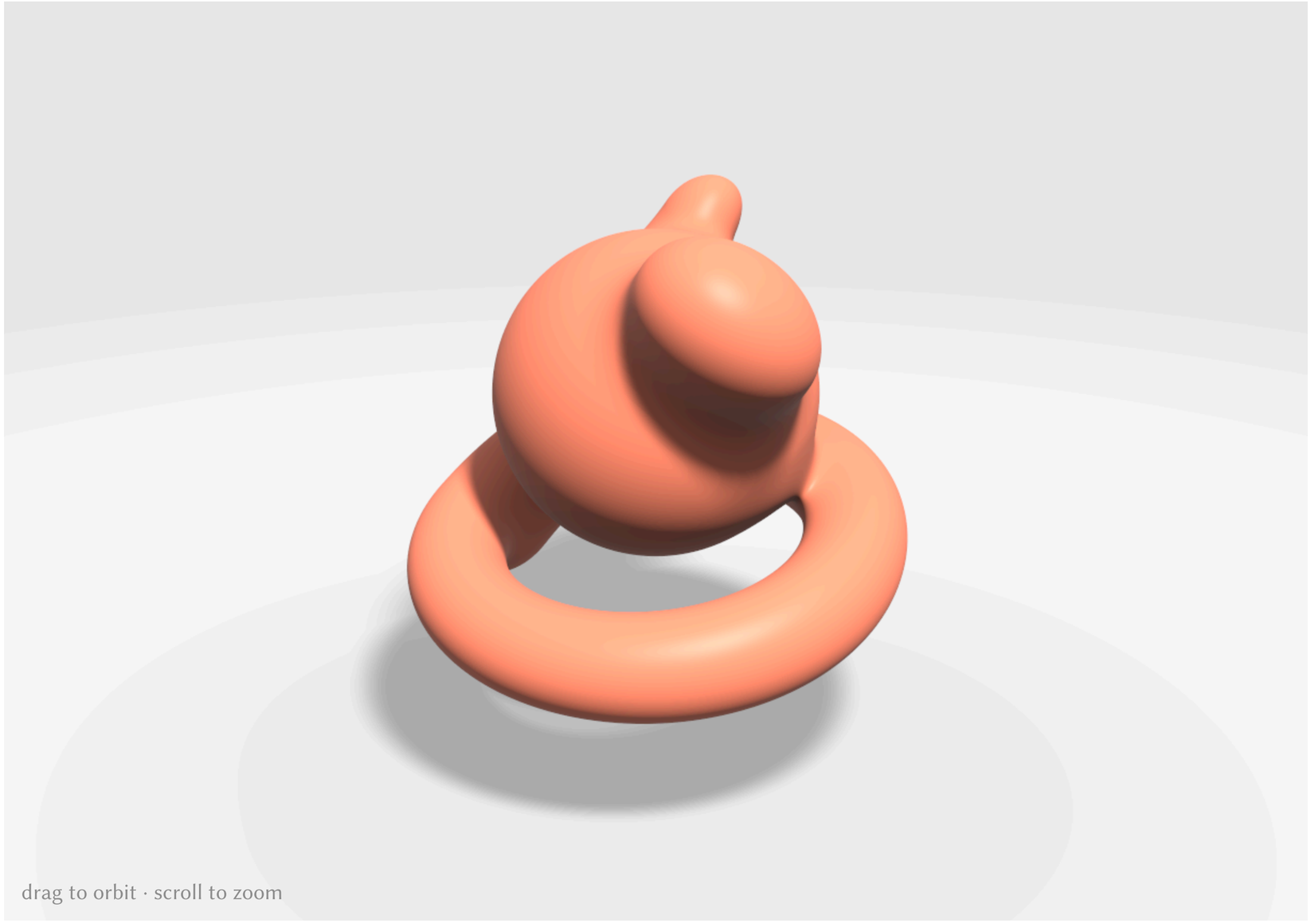
Question

You must ray-trace the surface.
Would you rather have occupancy or distance?

Distance. At any point, $|f(\mathbf{x})|$ is the radius of an empty ball:
you can step that far along the ray and never miss the surface.
Occupancy only tells you that you have not crossed yet.

Implicit Surface Explorer

A4 · Meshes and Shaders



Shape: Sphere | Torus | **Blend** | Lattice

View: **Direct** | Normals | Triangulated

Iso-value c — 0.00

☐ Slice — z = 0.00

↺ Reset

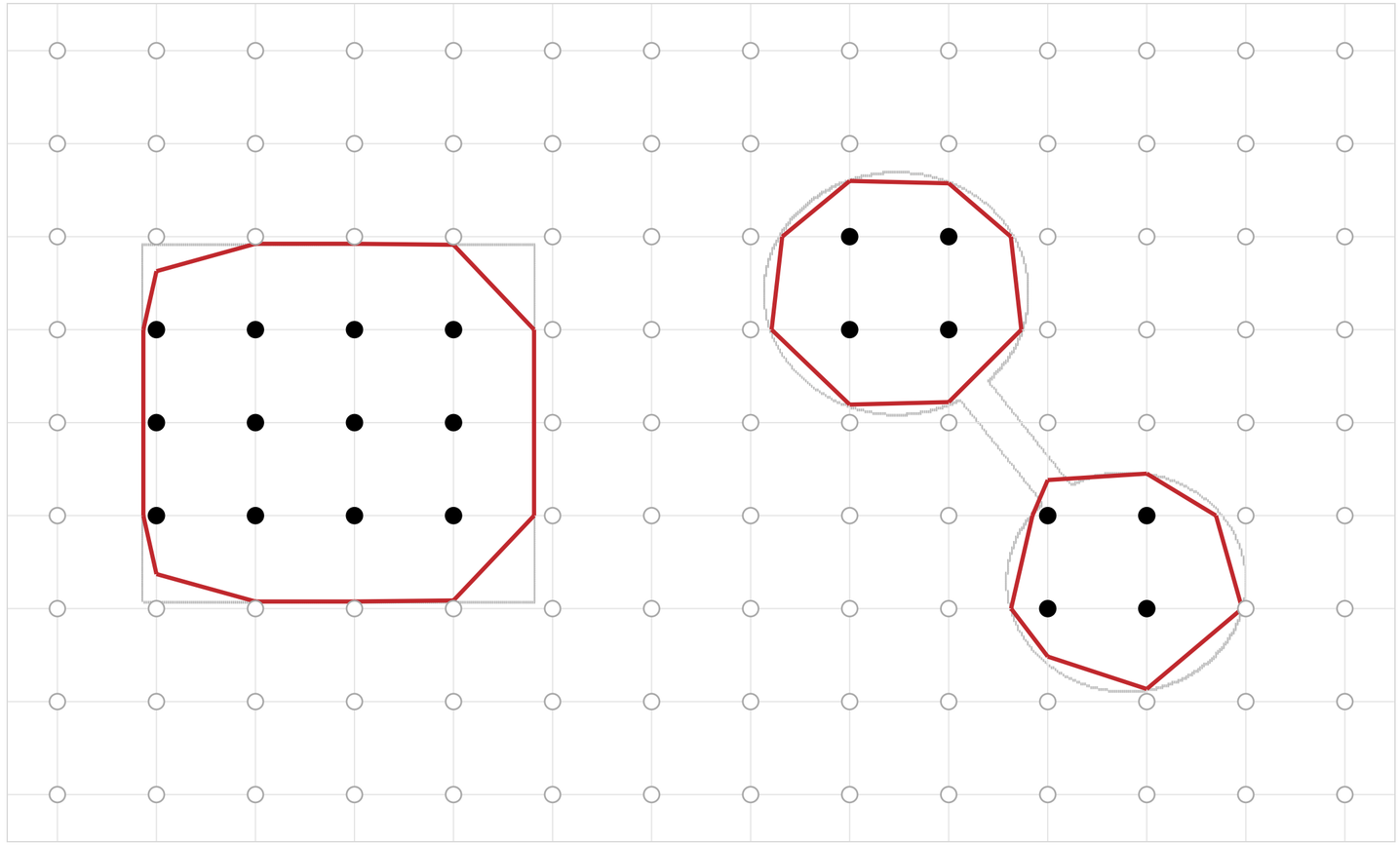
sphere tracing $f(\mathbf{x}) = 0.00$ at every pixel

inside 0 outside

Slice colour is f; lines every 0.1; black: the surface $f = c$.

Analytic SDFs, rendered live.

From Field to Mesh



Grid 14 per row

true zero set

14 × 9 samples of $f \longrightarrow$ **30** line segments

Reset

Marching squares / cubes

Sample f on a grid. In each cell, the signs at the corners choose a case; edge crossings are found by linear interpolation of f.

2D: 16 cases. 3D: 256 cases, 15 up to symmetry.

What is lost

Features smaller than a cell; sharp corners (they are cut off); exact topology where the field is ambiguous.

Lorensen & Cline 1987

Lorensen & Cline 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. SIGGRAPH 1987.

Three Ways to Store a Surface

	Triangle mesh	**Dense voxel SDF**	**Neural SDF**
Memory	$O(\text{faces})$	$O(n^3)$ for resolution n	fixed number of weights
Point query $f(\mathbf{x})$	needs a BVH (A3)	one trilinear lookup	one network evaluation
Render	rasterize, fast hardware	ray march	ray march, many evaluations
Topology change	remesh	free	free
Sharp features	exact, by construction	rounded below a voxel	smoothed by the network
Gradients	piecewise constant normals	finite differences	analytic, by autodiff
Editing	move vertices	write voxels	retrain, or condition on a code

No column wins every row.

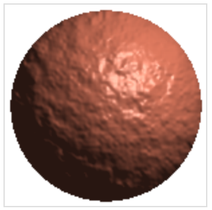
Appearance Is a Function Too



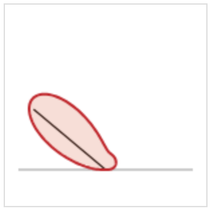
Noise
$f(\mathbf{x})$



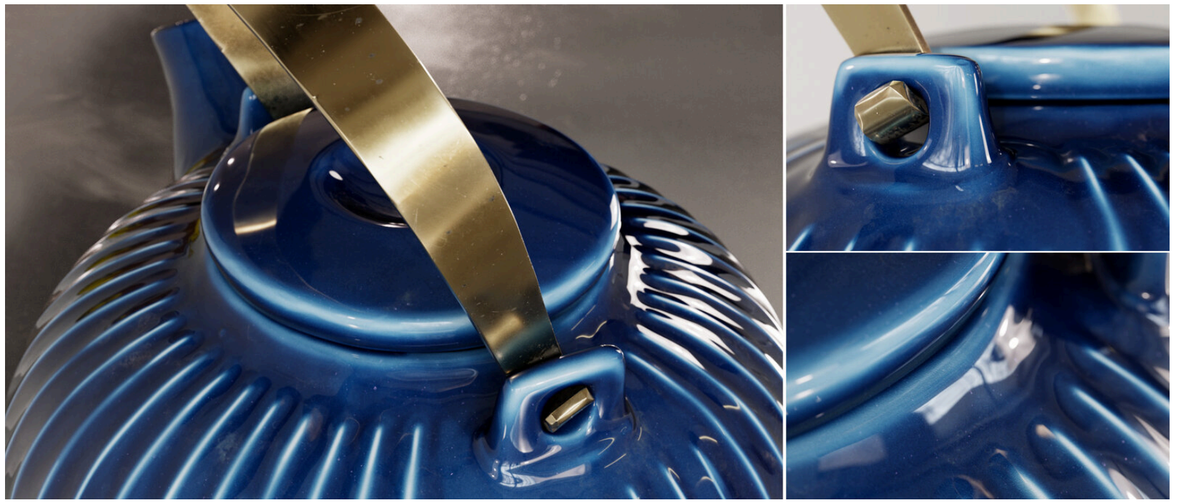
Texture
$f(u, v)$



Bump
∇h



BRDF
$\rho(\omega_i, \omega_o)$



Neural material rendered in real time. Zeltner et al. 2024

Real-Time Neural Appearance Models

Tizian Zeltner, Fabrice Rousselle, Andrea Weidlich, Petrik Clarberg, Jan Novák, Benedikt Bitterli, Alex Evans, Tomáš Davidovič, Simon Kallweit, Aaron Lefohn

ACM TOG (presented at SIGGRAPH 2024) · 2024

Layered production materials are compressed into latent textures and tiny MLP decoders that run inside a real-time path tracer.

$$f_\theta\big(\mathbf{z}(u, v),\ \omega_i,\ \omega_o\big) \approx \rho(\omega_i, \omega_o) \cos\theta_i$$

a latent texture $\mathbf{z}$ and a tiny decoder replace a layered material graph

Perlin 1985. An Image Synthesizer. SIGGRAPH 1985.
Zeltner et al. 2024. Real-Time Neural Appearance Models. ACM TOG (presented at SIGGRAPH 2024.

Part 2 · Takeaway

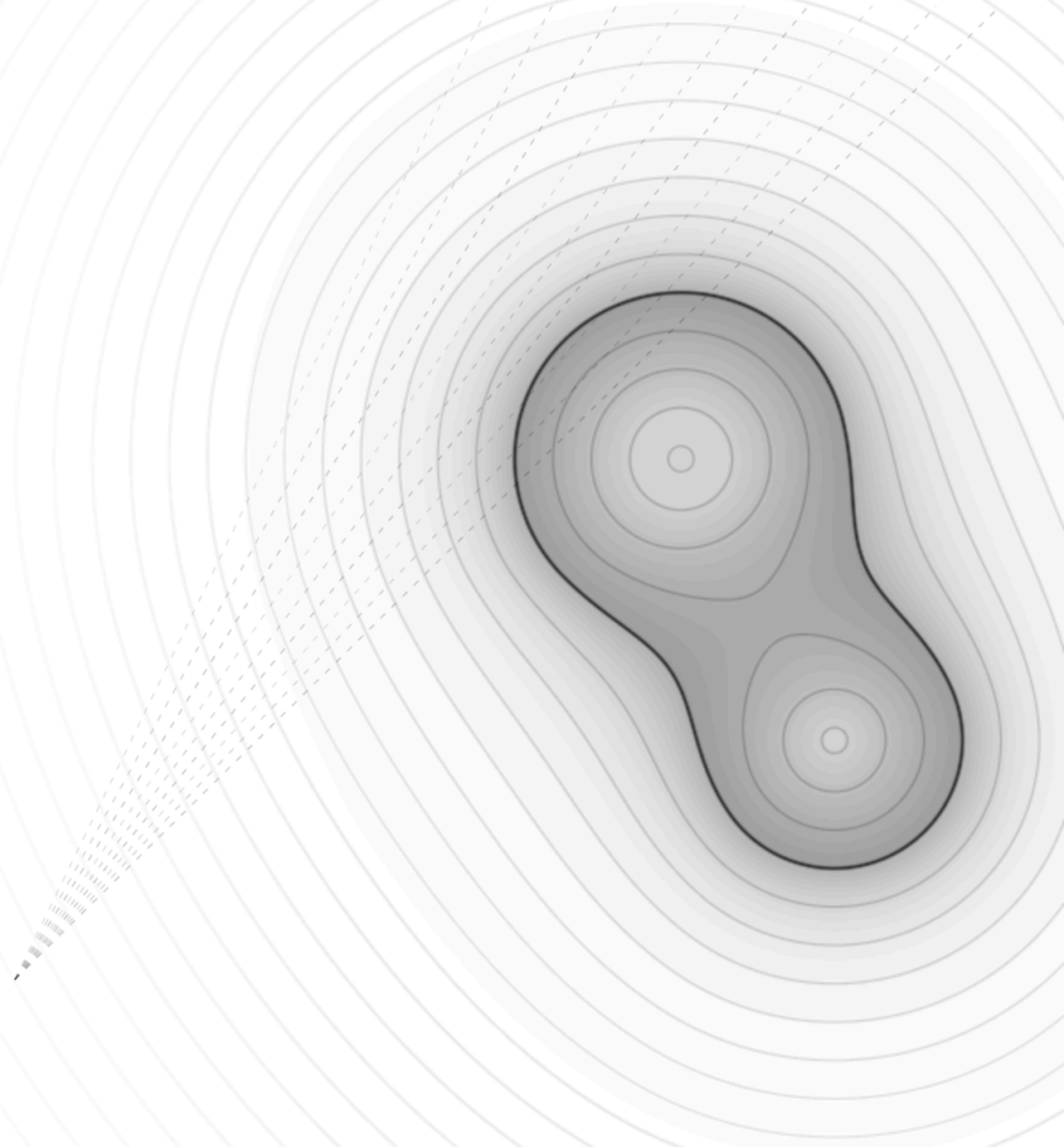
Geometry need not be defined by vertices and faces, but rendering and editing continuous geometry still require algorithms and structure.

To see a field, we must find where a ray meets it. That is A2.

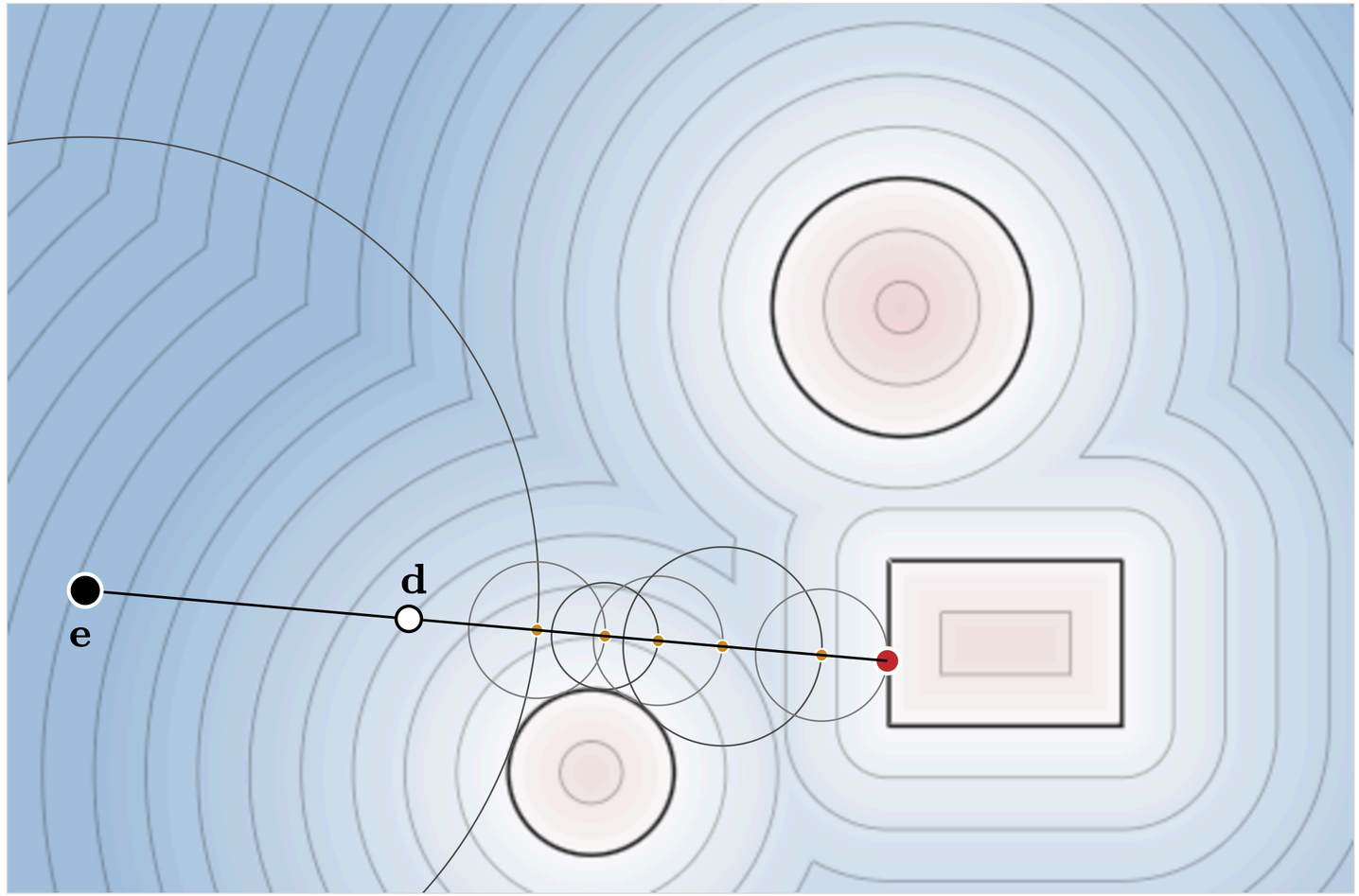
Part 3 · A2 Ray Tracing

Rendering a Field

$$f_\theta(\mathbf{x}, \mathbf{d}) = (\sigma, \mathbf{c})$$



Rays Meet Functions



hit at **t = 3.111** after **6** steps

drag the eye e and the direction handle d

Reset

$$\mathbf{r}(t) = \mathbf{e} + t\,\mathbf{d}$$

the A2 viewing ray

A2: solve $\mathbf{r}(t) \in$ sphere, plane, triangle

Now: solve $f(\mathbf{r}(t)) = 0$

$$t_{k+1} = t_k + f\big(\mathbf{r}(t_k)\big)$$

sphere tracing: step by the distance; never overshoot a true SDF

Hart 1996. Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces. The Visual Computer 1996.

Scene Representation Networks

Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations

Vincent Sitzmann, Michael Zollhöfer, Gordon Wetzstein

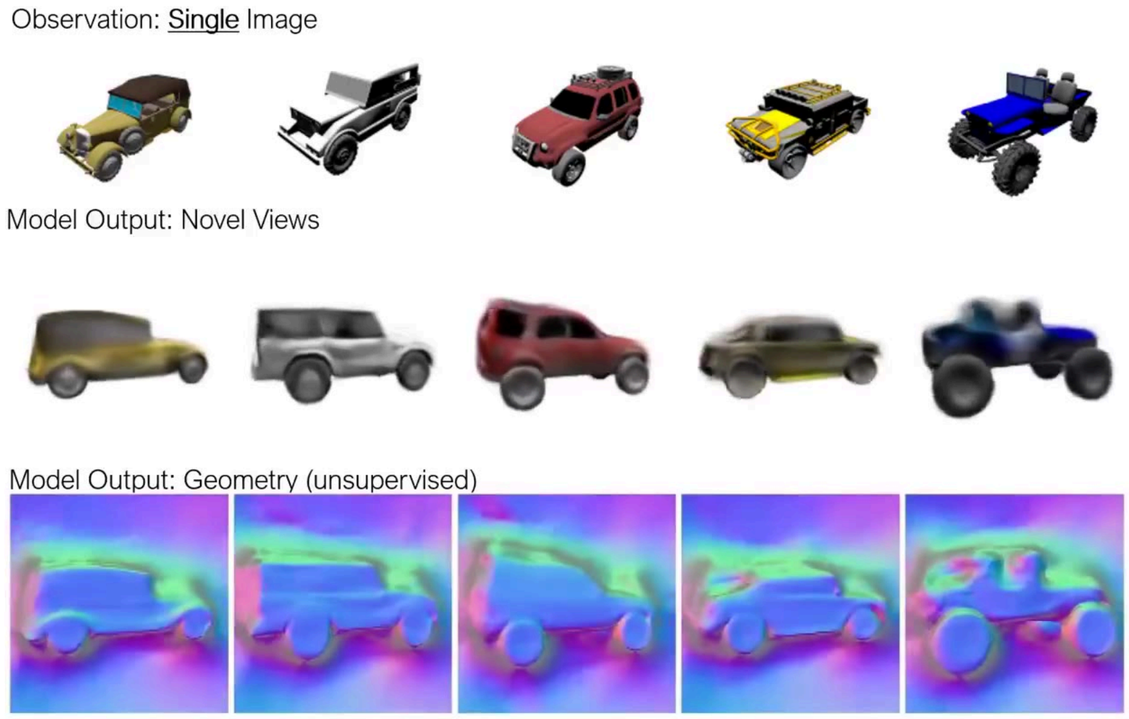
NeurIPS · 2019

A scene is a function from 3D points to features; a learned ray marcher renders it, so it can be trained from posed images alone.

$$\Phi_\theta : \mathbb{R}^3 \to \mathbb{R}^n$$

a feature at every point; a learned (LSTM) ray marcher finds the surface; a pixel generator decodes colour

The bridge: a 3D field, rendered differentiably, supervised only by 2D images.



Novel views and unsupervised geometry from a single image. Sitzmann et al. 2019

Sitzmann et al. 2019. Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations. NeurIPS 2019.

NeRF · Neural Radiance Fields

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng

ECCV · 2020

A scene as density and view-dependent emitted colour over 5D, fitted to photographs through classical volume rendering.

$$f_\theta(\mathbf{x}, \mathbf{d}) = (\sigma, \mathbf{c})$$

density $\sigma(\mathbf{x})$ from position only; emitted colour $\mathbf{c}(\mathbf{x}, \mathbf{d})$ also from direction

Not a material and not a light: $\mathbf{c}$ is the radiance leaving $\mathbf{x}$ toward $-\mathbf{d}$, with the scene's lighting baked in. No Blinn–Phong, no shadow rays.

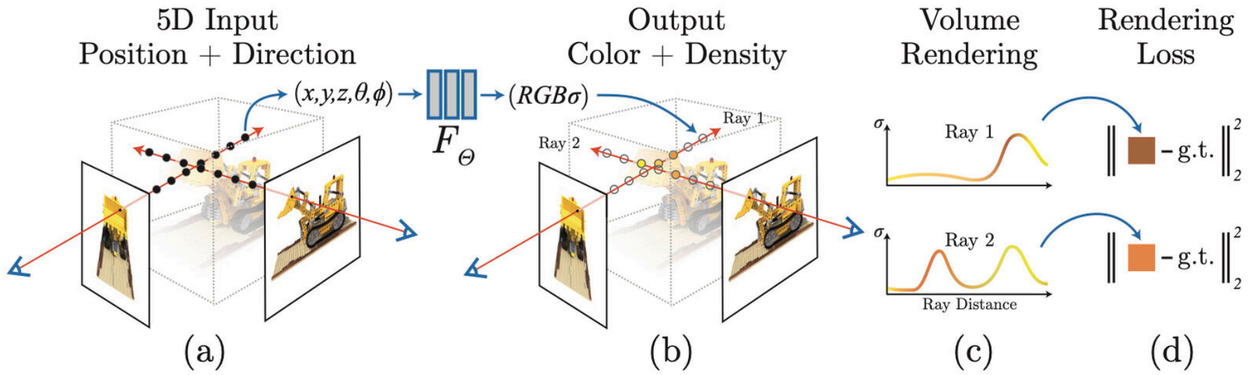


Figure: Mildenhall et al. 2020



Novel views. Mildenhall et al. 2020

Mildenhall et al. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. ECCV 2020.

The Volume Rendering Integral

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

$$T(t) = \exp\Big(-\int_{t_n}^{t} \sigma\big(\mathbf{r}(s)\big)\,ds\Big)$$

transmittance: the chance light gets from t to the eye

$\sigma\,dt$

chance of stopping in $[t, t+dt]$

$\mathbf{c}$

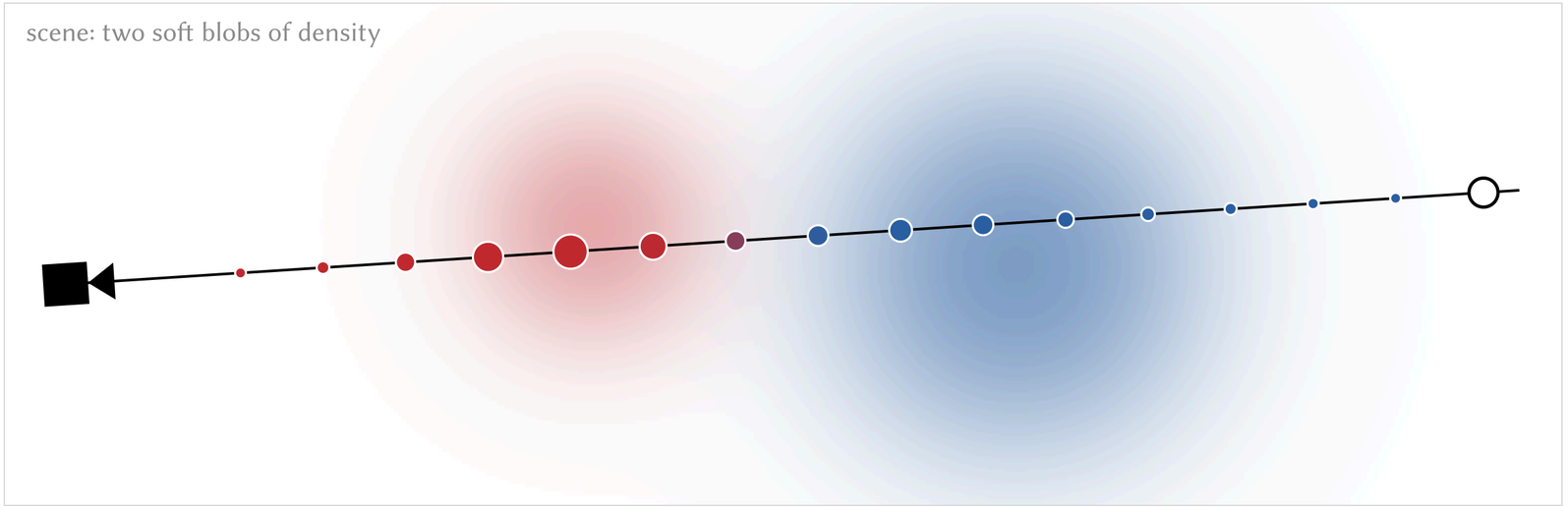
what is emitted there

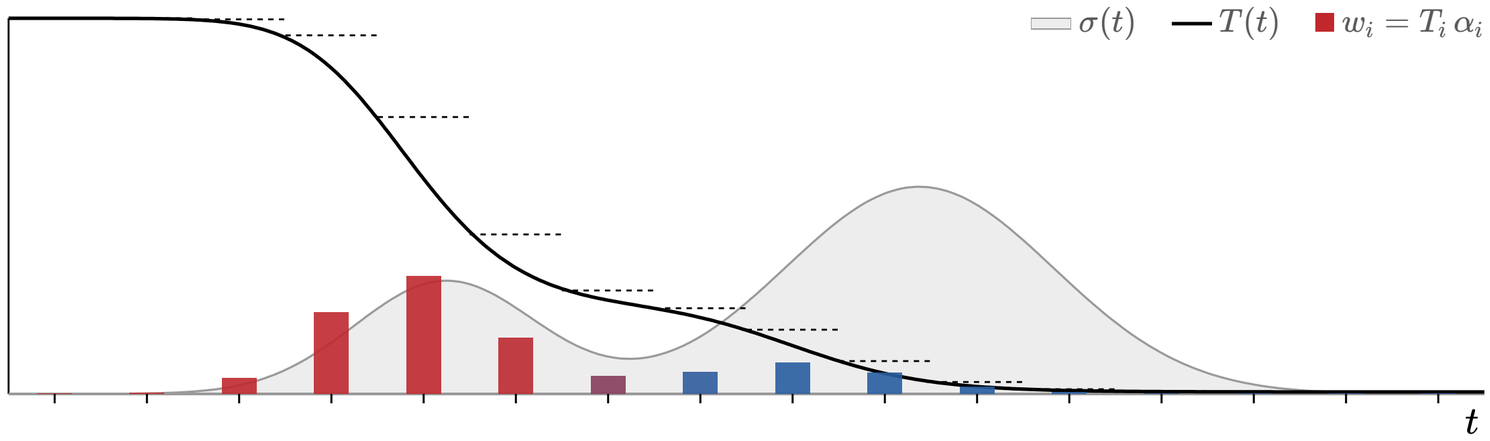
$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i\,\alpha_i\,\mathbf{c}_i, \quad \alpha_i = 1 - e^{-\sigma_i \delta_i}, \quad T_i = \prod_{j<i}(1-\alpha_j)$$

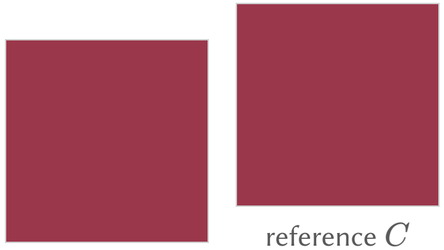
the continuous integral, evaluated at N samples with spacing δ_i: this is what the computer does

Kajiya & Von Herzen 1984. Ray Tracing Volume Densities. SIGGRAPH 1984.
Max 1995. Optical Models for Direct Volume Rendering. IEEE TVCG 1995.
Mildenhall et al. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. ECCV 2020.

Volume Rendering Explorer







front-to-back accumulation, sample by sample



opacity $\sum_i w_i$ = **0.995** · background seen T_{N+1} = **0.005**

error $\|\hat{C} - C\|$ = **0.000**

Samples N 16

☐ Jitter (stratified) Resample

Front density 1.4

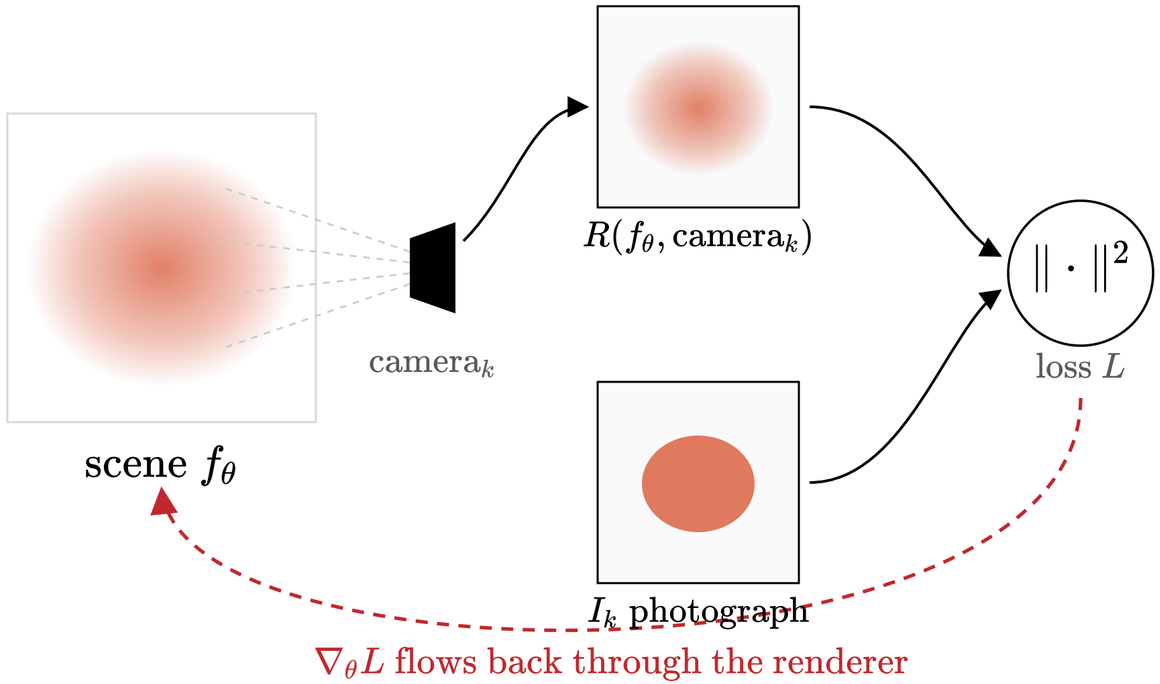
Back density 2.6

Front colour back

Reset

drag the camera or the aim point

Rendering, Run Backwards



$$\min_\theta \sum_k \left\| R(f_\theta, \text{camera}_k) - I_k \right\|^2$$

fit the scene so that its renders match the photographs

Every operation in $\hat{C}$ is differentiable in σ_i and $\mathbf{c}_i$, so the image error flows back through the renderer into θ.

Graphics renders a scene into an image. Inverse graphics recovers a scene from images. The renderer is the bridge.

NeuS · Surfaces from Volume Rendering

NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, Wenping Wang

NeurIPS · 2021

Volume-render a signed distance field through an unbiased, occlusion-aware weight, so the surface is recovered from images alone.

$f_\theta(\mathbf{x})$: an SDF, as in DeepSDF

$$\Phi_s(f) = (1 + e^{-sf})^{-1}$$

$$\alpha_i = \max\left(\frac{\Phi_s(f_i) - \Phi_s(f_{i+1})}{\Phi_s(f_i)},\ 0\right)$$

opacity rises exactly where the SDF crosses zero; s is learned and sharpens over training

Density fields give fuzzy geometry; an SDF gives a surface. NeuS renders the SDF with volume rendering so that photographs alone can fit it.



Multi-view reconstruction without masks. Wang et al. 2021

Wang et al. 2021. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. NeurIPS 2021.
Park et al. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. CVPR 2019.

3D Gaussian Splatting · A Counterpoint

3D Gaussian Splatting for Real-Time Radiance Field Rendering

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis

ACM TOG (SIGGRAPH) · 2023

Millions of explicit anisotropic Gaussians, optimized through a differentiable tile rasterizer, render radiance fields in real time.

$$G_k(\mathbf{x}) = o_k \, e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \Sigma_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)}$$

each primitive: position, covariance, opacity, view-dependent colour (spherical harmonics)

$$C = \sum_k T_k \, \alpha_k \, \mathbf{c}_k$$

the same compositing sum, over depth-sorted splats instead of ray samples

Continuous primitives, no network. Rasterized in screen tiles, optimized by gradient descent, real time.



Real-time novel-view synthesis. Kerbl et al. 2023

Kerbl et al. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. ACM TOG (SIGGRAPH) 2023.

Part 3 · Takeaway

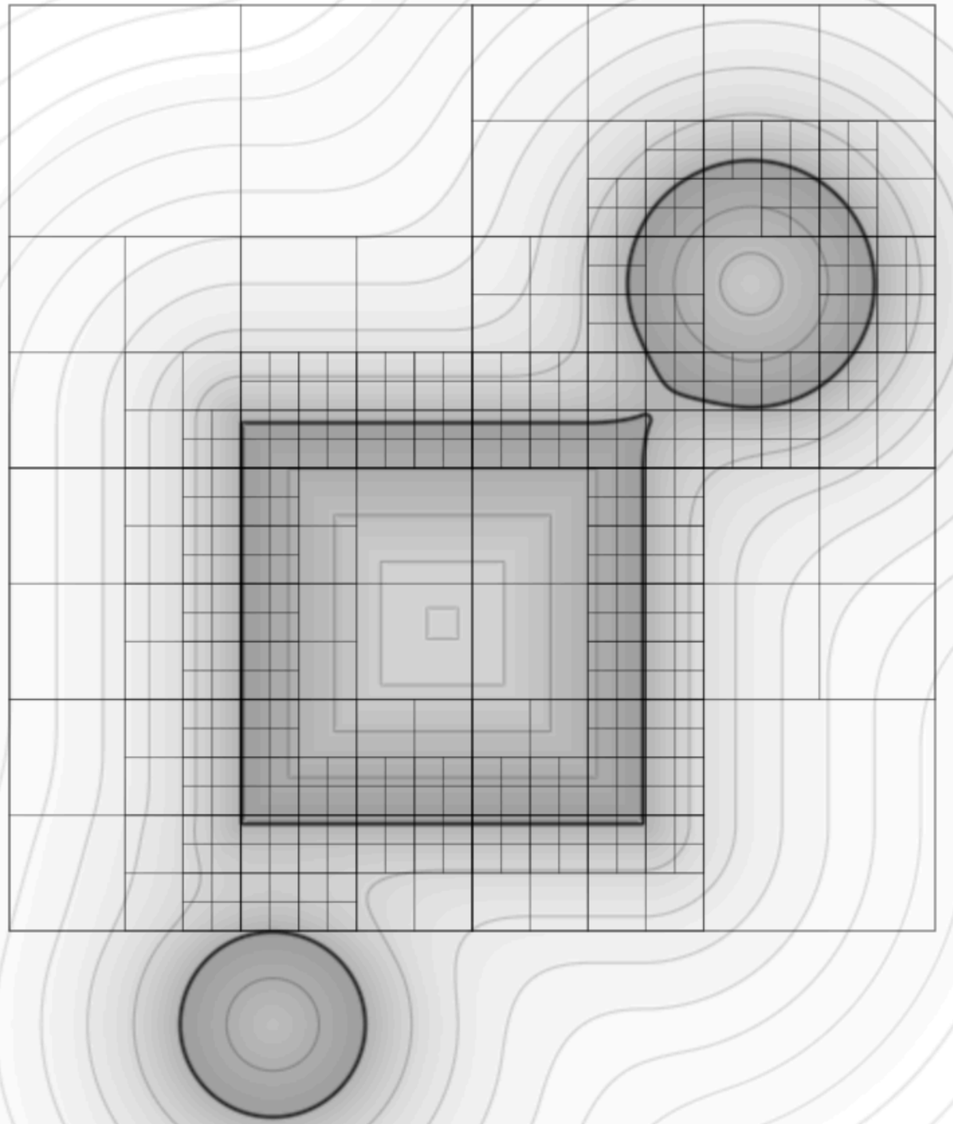
Neural rendering connects continuous scene functions with discrete image observations through differentiable image formation.

But every pixel cost hundreds of network evaluations. That is A3's problem.

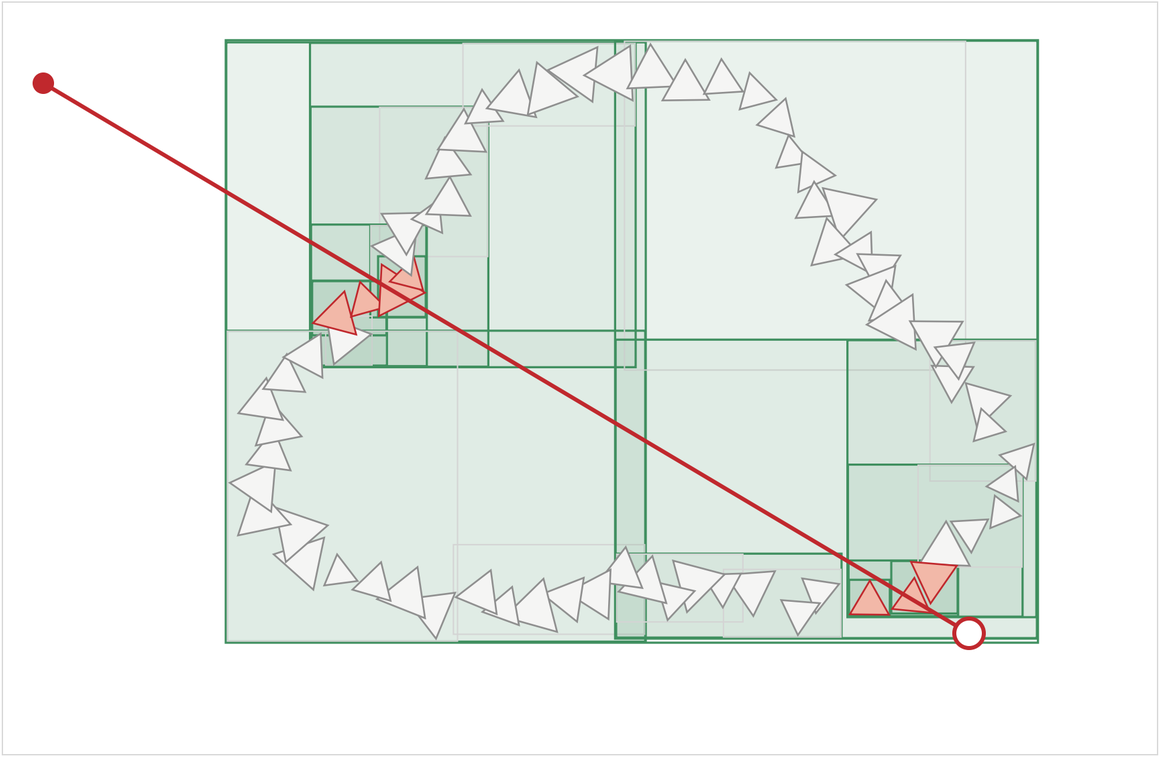
Part 4 · A3 Bounding Volume Hierarchy

Accelerating a Field

If geometry is now a neural network, have we eliminated the need for spatial data structures?



What a BVH Buys, and What a Field Costs



29 box tests + **7** triangle tests, instead of **70** triangle tests

Reset

A3: skip what the ray cannot hit

A miss on a box prunes its whole subtree: $O(n)$ triangle tests become $O(\log n)$ box tests.

A field: every query is a network

NeRF: 64 coarse + 128 fine samples per ray, each an MLP with ~0.6 M weights. Most samples land in empty space.

192

network queries per ray

~$2 \cdot 10^8$

queries per 1 MP image

Mildenhall et al. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. ECCV 2020.

Neural Geometric Level of Detail

Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes

Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, Sanja Fidler

CVPR · 2021

Features stored on a sparse octree, interpolated and summed across levels, feed a tiny MLP; the octree also tells the ray where to look.

$$\hat{f}_L(\mathbf{x}) = \mathrm{MLP}_L\left(\left[\mathbf{x},\ \sum_{\ell=1}^{L} \mathrm{trilerp}\left(\mathbf{z}^{(\ell)}, \mathbf{x}\right)\right]\right)$$

features live on the corners of a sparse octree's voxels; summed over levels $1..L$

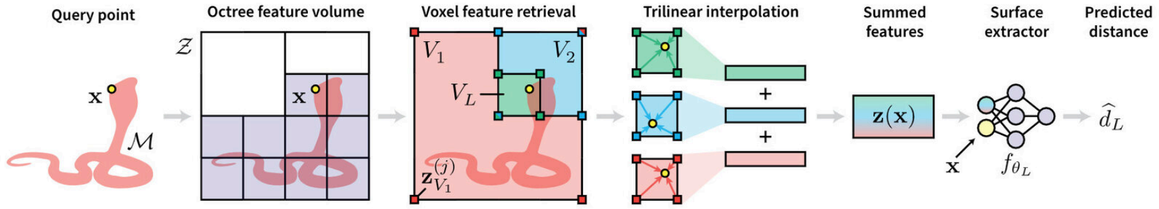


Figure: Takikawa et al. 2021



Real-time rendering of neural SDFs. Takikawa et al. 2021

Locality

A query reads only the 8 corners per level around it.

Level of detail

Stop at level L: a coarser, cheaper surface.

Empty space

Rays traverse the octree (like a BVH) and sphere-trace only inside occupied voxels.

Tiny decoder

The MLP is small because features carry the detail.

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

Instant-NGP · Multiresolution Hash Encoding

Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

Thomas Müller, Alex Evans, Christoph Schied, Alexander Keller

ACM TOG (SIGGRAPH) · 2022

Trainable feature grids at many resolutions, stored in fixed-size hash tables, let a tiny MLP fit images, SDFs and NeRFs in seconds.

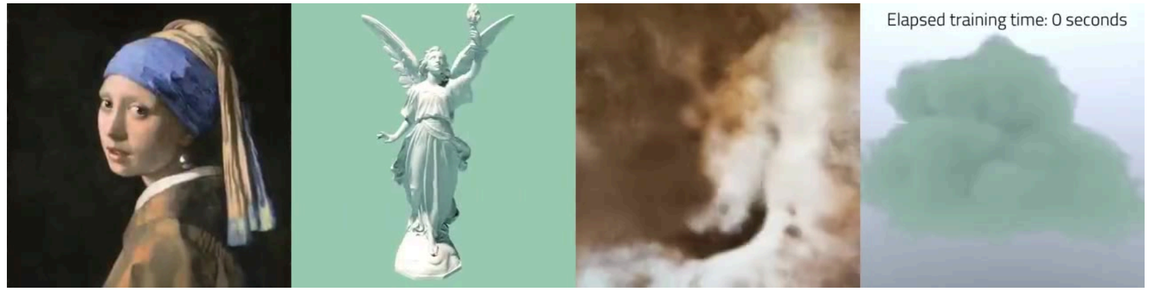
$$h(\mathbf{x}) = \left(\bigoplus_{i=1}^{d} x_i \, \pi_i \right) \bmod T$$

integer grid vertex $\mathbf{x}$ → slot in a table of T trainable features; $\pi_1 = 1$, $\pi_2 = 2654435761$, $\pi_3 = 805459861$

$$f(\mathbf{x}) = m_\phi\big(\,\mathrm{interp}_1(\mathbf{x}),\ \ldots,\ \mathrm{interp}_L(\mathbf{x})\,\big)$$

$L = 16$ levels, $F = 2$ features each, $T = 2^{14}$–2^{24}; decoder: 2 hidden layers of 64

Coarse levels: one-to-one. Fine levels: more vertices than slots, so collisions are allowed, and gradients from the visible surface dominate.



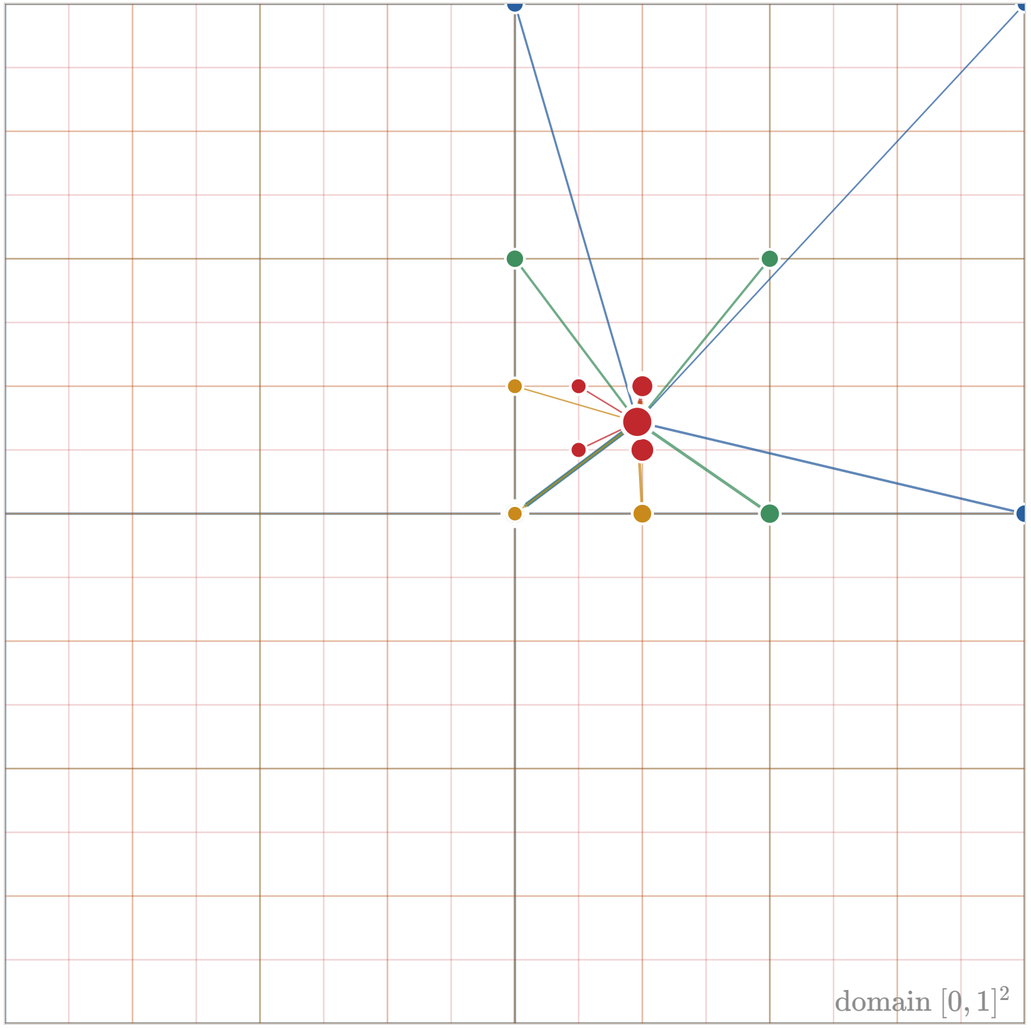
Image, SDF, NeRF and volume, one encoding. Müller et al. 2022



A neural SDF with the hash encoding. Müller et al. 2022

Müller et al. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. ACM TOG (SIGGRAPH) 2022.

Hash Grid Explorer



- **level 0** 2×2 cells · 9 vertices · dense, one slot per vertex
- **level 1** 4×4 cells · 25 vertices · dense, one slot per vertex
- **level 2** 8×8 cells · 81 vertices · hashed into 32 slots
- **level 3** 16×16 cells · 289 vertices · hashed into 32 slots

Each query reads **16** slots out of **128**; bilinear weights per level sum to 1.

$\ell = 0$: weights (0.14, 0.04, 0.62, 0.20)
$\ell = 1$: weights (0.19, 0.17, 0.33, 0.31)
$\ell = 2$: weights (0.03, 0.69, 0.01, 0.27)
$\ell = 3$: weights (0.04, 0.40, 0.04, 0.52)

$$\left[\text{interp}_0, \ldots, \text{interp}_3\right] \rightarrow m_\phi \rightarrow f(\mathbf{x})$$

Table size T | 8 | 16 | **32** | 64 | Levels L — 4

☐ Hash every vertex (show sharing) ↺ Reset

$h(\mathbf{x}) = (x \cdot 1 \oplus y \cdot 2654435761) \bmod T$; outlined slot: two corners of this query share it. Drag the red point.

Three Architectures

	Pure coordinate MLP SIREN, NeRF	**Hierarchical spatial field** NGLOD	**Multiresolution hashed field** Instant-NGP
Memory	small, fixed	grows with occupied voxels	fixed by $T \cdot L \cdot F$
Cost per query	whole network	8 lookups per level + tiny MLP	8 lookups per level + tiny MLP
Locality	every weight affects every point	local features	local features (shared by collisions)
Training speed	hours	minutes	seconds to minutes
Interpolation / prior	smooth, global	local, guided by the octree	local; weak away from data
Structure	implicit in θ	explicit octree, must be built	explicit grids, implicit hashing

Part 4 · Takeaway

Continuous representation and discrete acceleration are not opposites.
Effective neural graphics frequently combines both.

A BVH, an octree, a hash table: discretization chosen for speed, not imposed by the scene.

Motion as a Map

$$\varphi(\mathbf{x}, t) : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^3$$

where does the material point $\mathbf{x}$ of the rest shape go at time t?

Canonical space

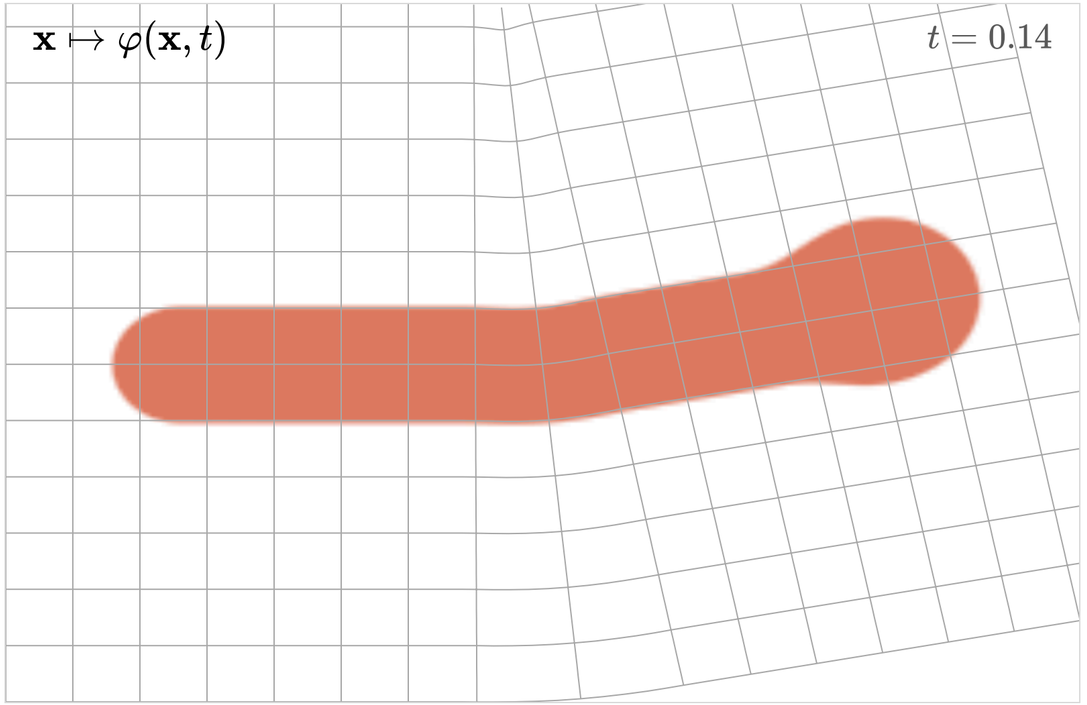
One rest shape, stored once.

Deformed space

What the camera sees at time t.

Next week, A5

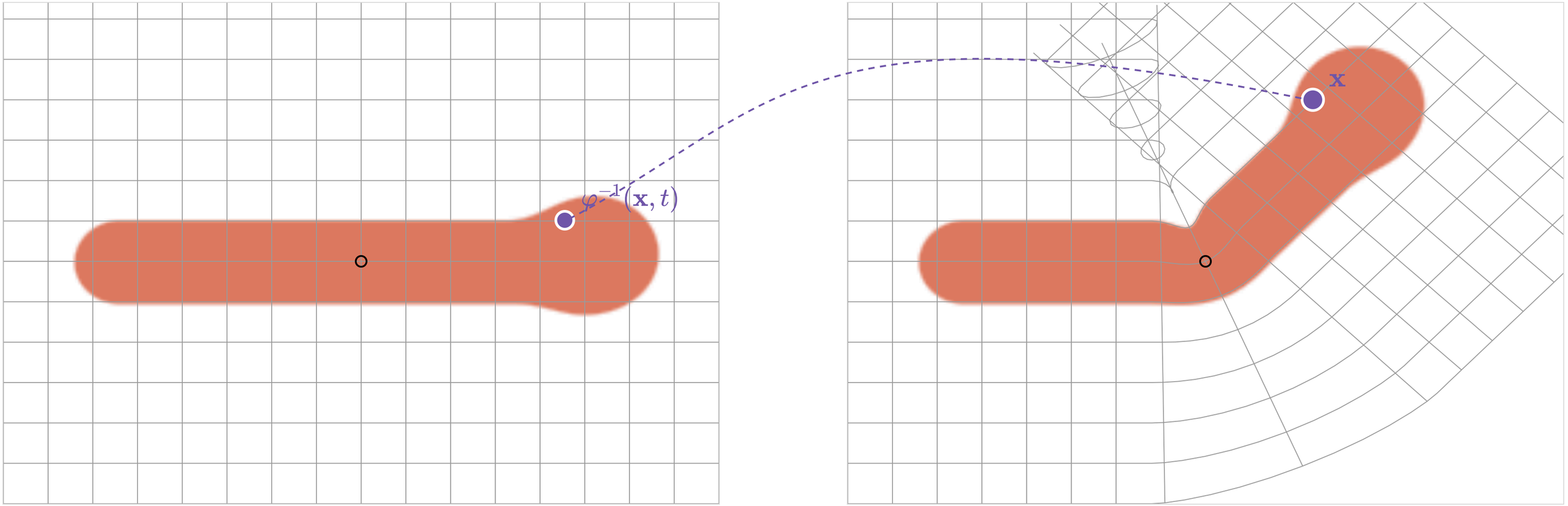
Linear blend skinning is such a map, built from bones: $\varphi(\mathbf{x}) = \sum_j w_j(\mathbf{x})\, T_j\, \mathbf{x}$.



Canonical and Observed Space

Canonical space

Observed space at time t



Time t 0.60 ▸ Play ☑ Grid

φ^{-1} maps the observed point (0.60, 1.00) to canonical (1.14, 0.25); its density there is **inside**.

↺ Reset

To render the deformed field, map each observed point back: $\sigma_t(\mathbf{x}) = \sigma\left(\varphi^{-1}(\mathbf{x}, t)\right)$

Nerfies · Deformable Neural Radiance Fields

Nerfies: Deformable Neural Radiance Fields

Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, Ricardo Martin-Brualla

ICCV · 2021

A per-frame deformation field warps each observation into one canonical NeRF, with elastic regularization and coarse-to-fine optimization.

$$\mathbf{x}' = T(\mathbf{x}, \boldsymbol{\omega}_i), \qquad (\sigma, \mathbf{c}) = F_\theta(\mathbf{x}', \mathbf{d}, \boldsymbol{\psi}_i)$$

a per-frame code $\boldsymbol{\omega}_i$ drives an MLP warp (an SE(3) field) from observation to canonical space

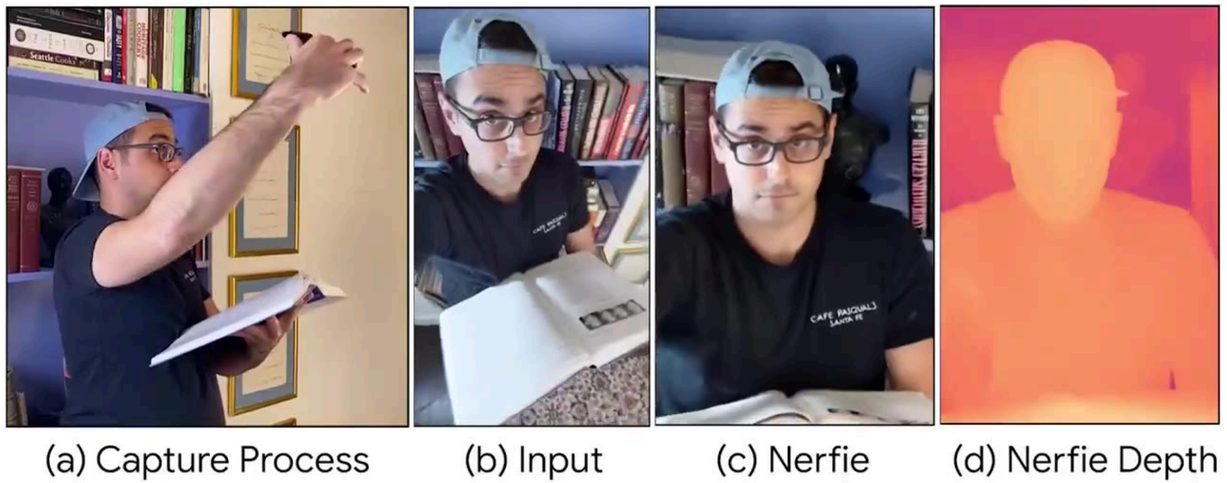
Elastic regularizer

Penalize the warp's Jacobian for deviating from a rotation.

Coarse to fine

Anneal the warp's positional-encoding frequencies: big motions first.

Correspondence is never observed; it is whatever makes all frames agree on one canonical scene.



Casually captured selfie, free-viewpoint. Park et al. 2021

Park et al. 2021. Nerfies: Deformable Neural Radiance Fields. ICCV 2021.

The Pipeline, Revisited

CSC317	As a function	Seen in
Pixels A1	continuous image functions	CPPN, Fourier features, SIREN, DANF
Meshes A4	implicit surfaces	DeepSDF, Occupancy Networks
Surface shading A4	learned appearance functions	Real-time neural appearance models
Ray casting A2	volumetric scene integration	SRN, NeRF, NeuS, Gaussian splatting
BVHs A3	neural-field acceleration hierarchies	NGLOD, Instant-NGP
Animated vertices A5	deformation fields	Nerfies

What Is Actually Gained?

Resolution independence

Query anywhere; detail is still bounded by data and capacity.

Differentiability

Gradients in space and in parameters: normals, inverse rendering, optimization.

Compactness and priors

A few weights, or a latent code, for a shape or a category.

Evaluation cost

Every query is computation; acceleration brings structure back.

Discontinuities

Edges, creases and occlusion boundaries resist smooth functions.

Control

No vertices to grab; geometry and semantics are entangled in θ.

Continuity is a modeling choice. Discretization is a computational tool. Neither should be taken for granted.

Thank you

Questions?

Zhecheng Wang

zhecheng@cs.toronto.edu · Dynamic Graphics Project

dgp.toronto.edu/~zhecheng

slides, demos and references

References 1 / 2

Stanley 2007. Compositional Pattern Producing Networks: A Novel Abstraction of Development. Kenneth O. Stanley. Genetic Programming and Evolvable Machines 8(2), 2007. link

Mordvintsev et al. 2018. Differentiable Image Parameterizations. Alexander Mordvintsev, Nicola Pezzotti, Ludwig Schubert, Chris Olah. Distill, 2018. link

Ulyanov et al. 2018. Deep Image Prior. Dmitry Ulyanov, Andrea Vedaldi, Victor Lempitsky. CVPR, 2018. link

Rahaman et al. 2019. On the Spectral Bias of Neural Networks. Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, Aaron Courville. ICML, 2019. link

Tancik et al. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, Ren Ng. NeurIPS, 2020. link

Sitzmann et al. 2020. Implicit Neural Representations with Periodic Activation Functions. Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, Gordon Wetzstein. NeurIPS, 2020. link

Belhe et al. 2023. Discontinuity-Aware 2D Neural Fields. Yash Belhe, Michaël Gharbi, Matthew Fisher, Iliyan Georgiev, Ravi Ramamoorthi, Tzu-Mao Li. ACM TOG (SIGGRAPH Asia), 2023. link

Catmull & Clark 1978. Recursively Generated B-spline Surfaces on Arbitrary Topological Meshes. Edwin Catmull, James Clark. Computer-Aided Design 10(6), 1978. link

Blinn 1982. A Generalization of Algebraic Surface Drawing. James F. Blinn. ACM TOG 1(3), 1982. link

Lorensen & Cline 1987. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. William E. Lorensen, Harvey E. Cline. SIGGRAPH, 1987. link

Perlin 1985. An Image Synthesizer. Ken Perlin. SIGGRAPH, 1985. link

Park et al. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, Steven Lovegrove. CVPR, 2019. link

Mescheder et al. 2019. Occupancy Networks: Learning 3D Reconstruction in Function Space. Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, Andreas Geiger. CVPR, 2019. link

Paper figures and videos are reproduced from the authors' project pages for teaching, with attribution on each slide. All other figures and demos were made for this lecture. Course material: github.com/panuelosj/computer-graphics-csc317-F26.

References 2 / 2

Zeltner et al. 2024. Real-Time Neural Appearance Models. Tizian Zeltner, Fabrice Rousselle, Andrea Weidlich, Petrik Clarberg, Jan Novák, Benedikt Bitterli, Alex Evans, Tomáš Davidovič, Simon Kallweit, Aaron Lefohn. ACM TOG (presented at SIGGRAPH 2024), 2024. link

Hart 1996. Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces. John C. Hart. The Visual Computer 12(10), 1996. link

Kajiya & Von Herzen 1984. Ray Tracing Volume Densities. James T. Kajiya, Brian P. Von Herzen. SIGGRAPH, 1984. link

Kajiya 1986. The Rendering Equation. James T. Kajiya. SIGGRAPH, 1986. link

Max 1995. Optical Models for Direct Volume Rendering. Nelson Max. IEEE TVCG 1(2), 1995. link

Sitzmann et al. 2019. Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations. Vincent Sitzmann, Michael Zollhöfer, Gordon Wetzstein. NeurIPS, 2019. link

Mildenhall et al. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng. ECCV, 2020. link

Wang et al. 2021. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, Wenping Wang. NeurIPS, 2021. link

Kerbl et al. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis. ACM TOG (SIGGRAPH), 2023. link

Takikawa et al. 2021. Neural Geometric Level of Detail: Real-time Rendering with Implicit 3D Shapes. Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, Sanja Fidler. CVPR, 2021. link

Müller et al. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. Thomas Müller, Alex Evans, Christoph Schied, Alexander Keller. ACM TOG (SIGGRAPH), 2022. link

Park et al. 2021. Nerfies: Deformable Neural Radiance Fields. Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, Ricardo Martin-Brualla. ICCV, 2021. link

Xie et al. 2022. Neural Fields in Visual Computing and Beyond. Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, Srinath Sridhar. Computer Graphics Forum (Eurographics STAR), 2022. link

Paper figures and videos are reproduced from the authors' project pages for teaching, with attribution on each slide. All other figures and demos were made for this lecture. Course material: github.com/panuelosj/computer-graphics-csc317-F26.