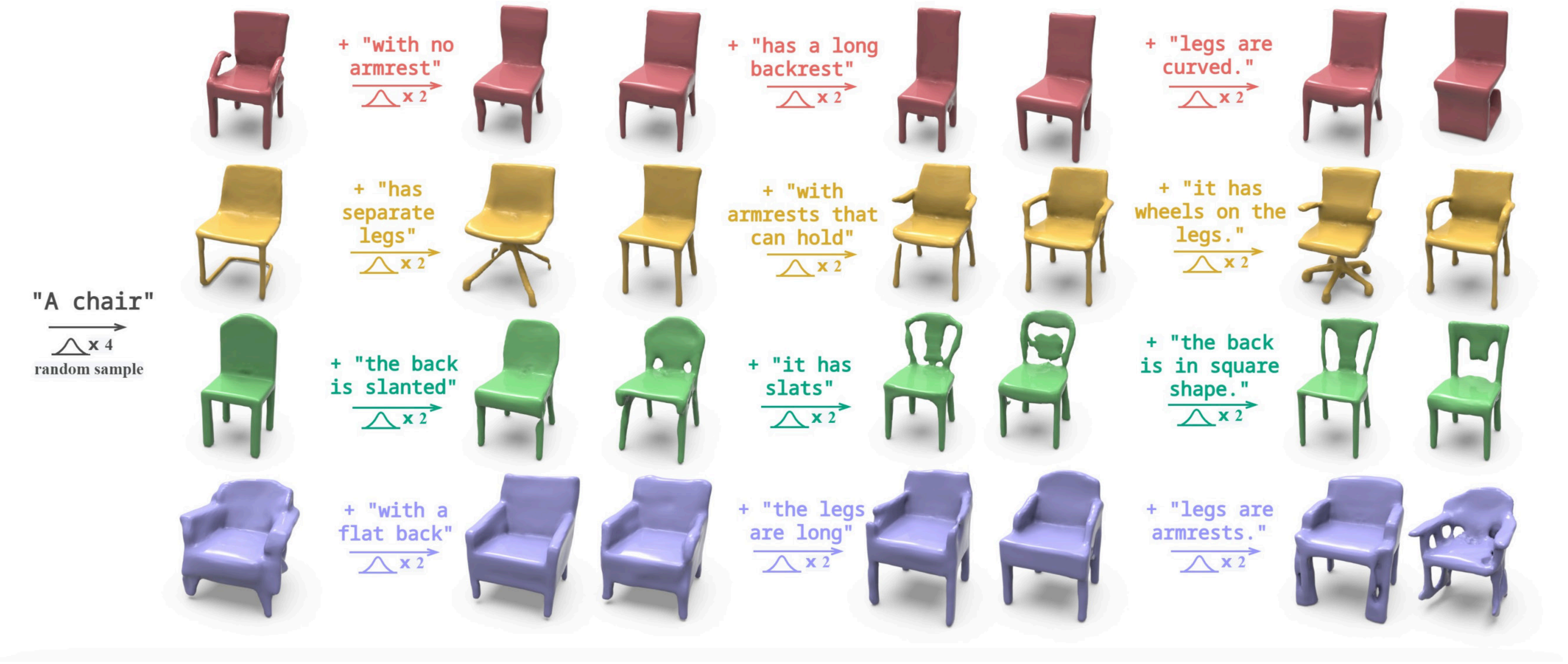
Neural Implicit Shape Representations

Deep Learning for Geometry Processing

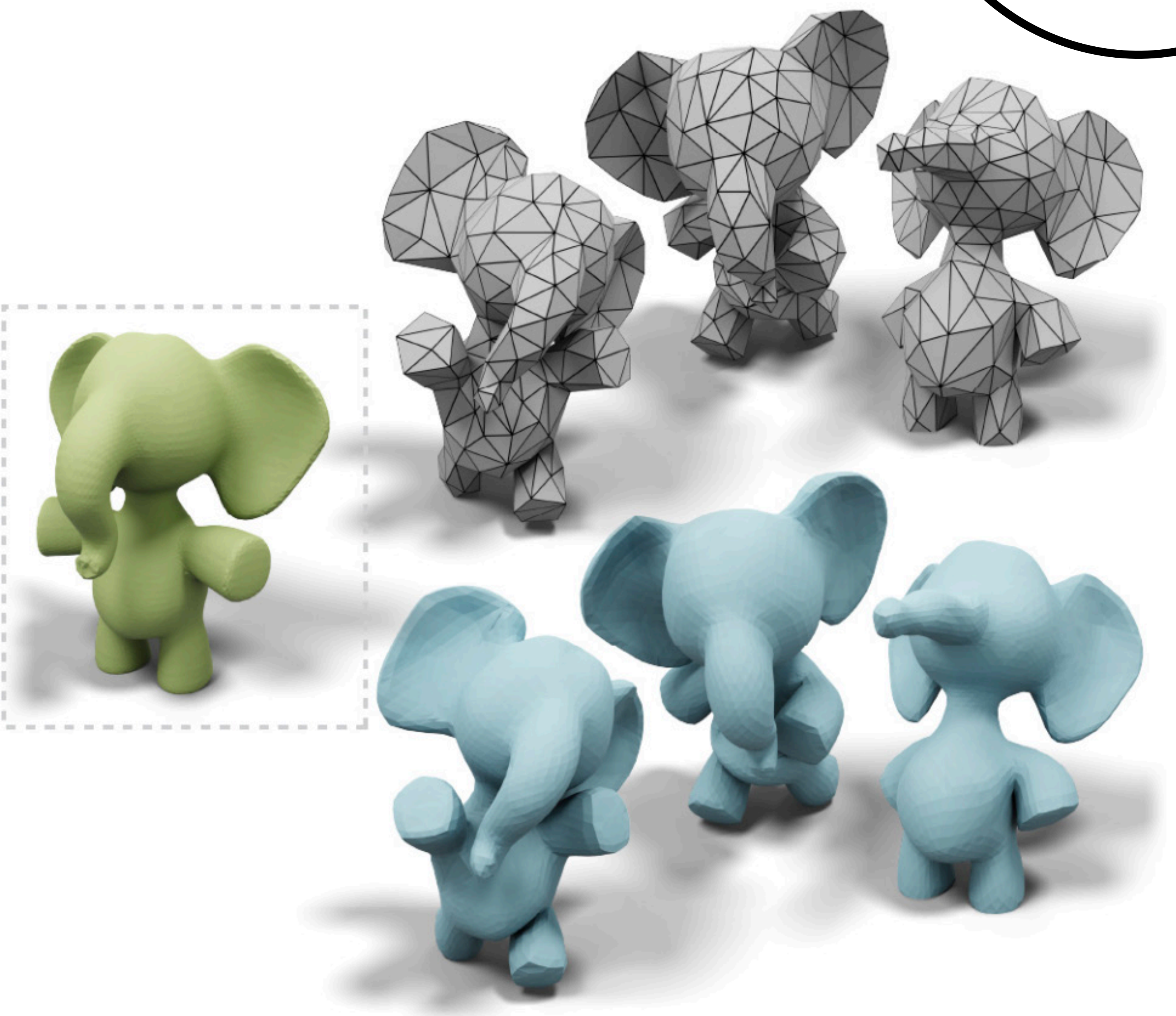
Daniel Ritchie and **Zoë Marschner**

SGP 2024 / June 22nd 2024

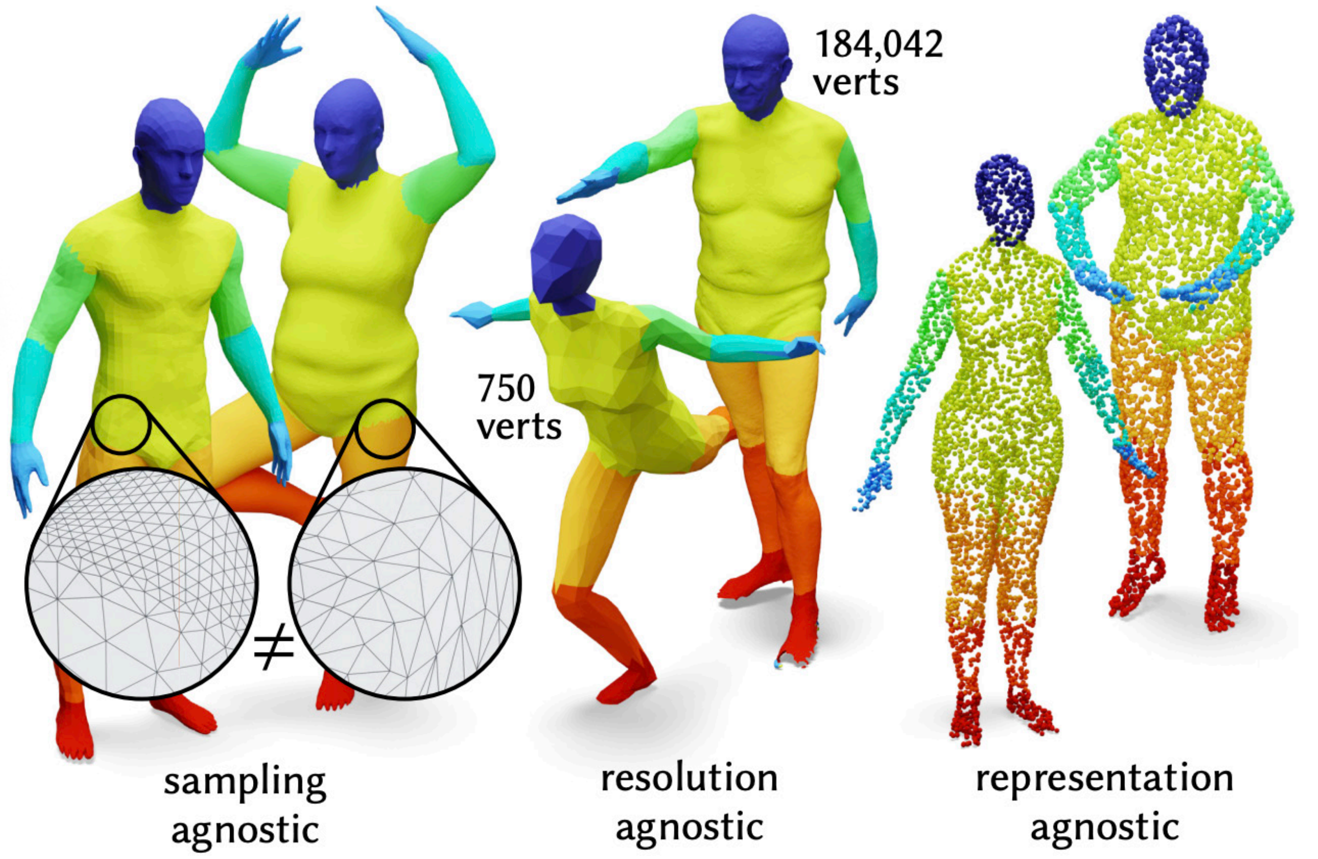
Synthesis
(e.g., ShapeCrafter [Fu et al., 2023])

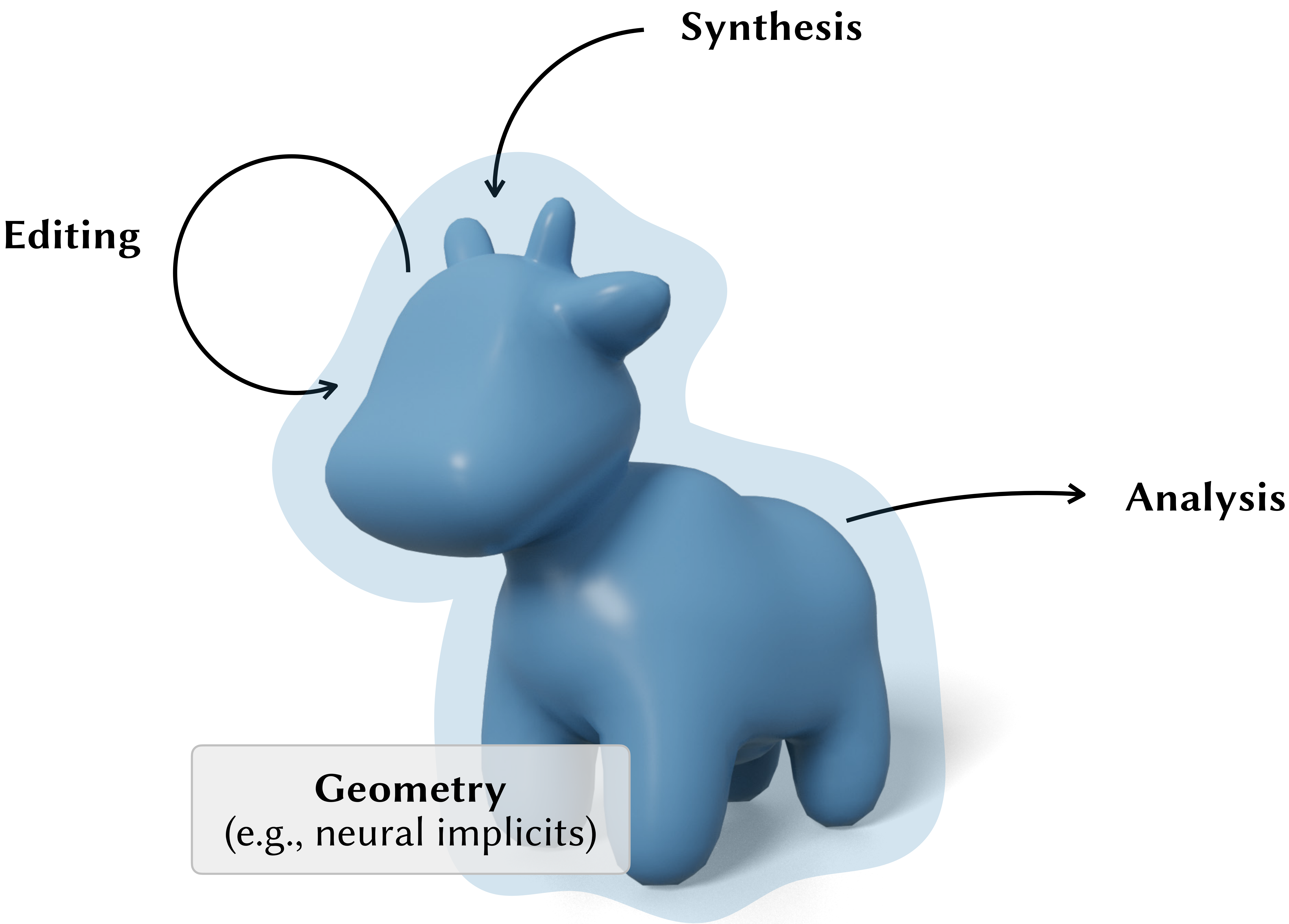


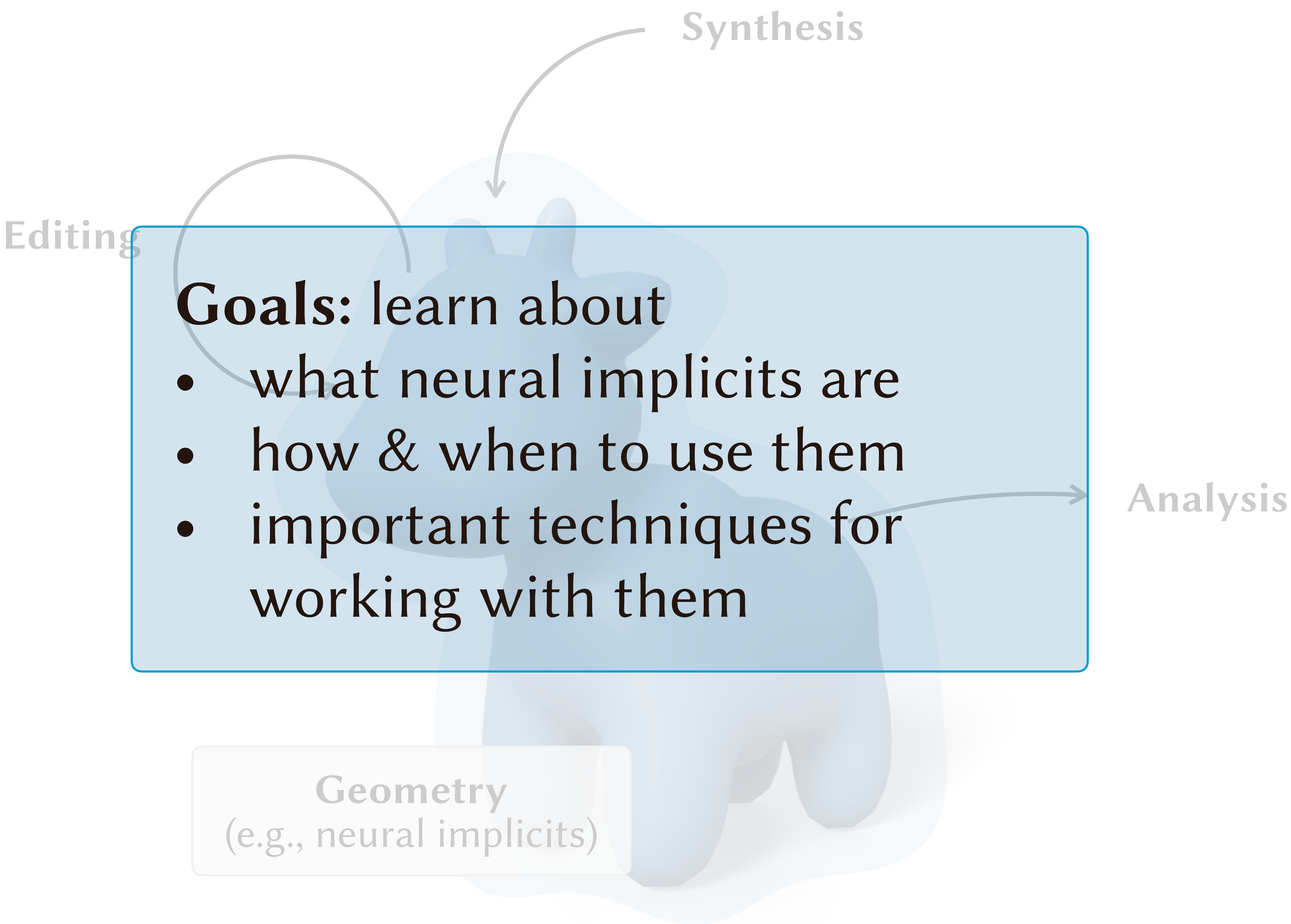
Editing
(e.g., Neural Subdivision [Liu et al., 2020])



Analysis
(e.g., DiffusionNet [Sharp et al., 2022])

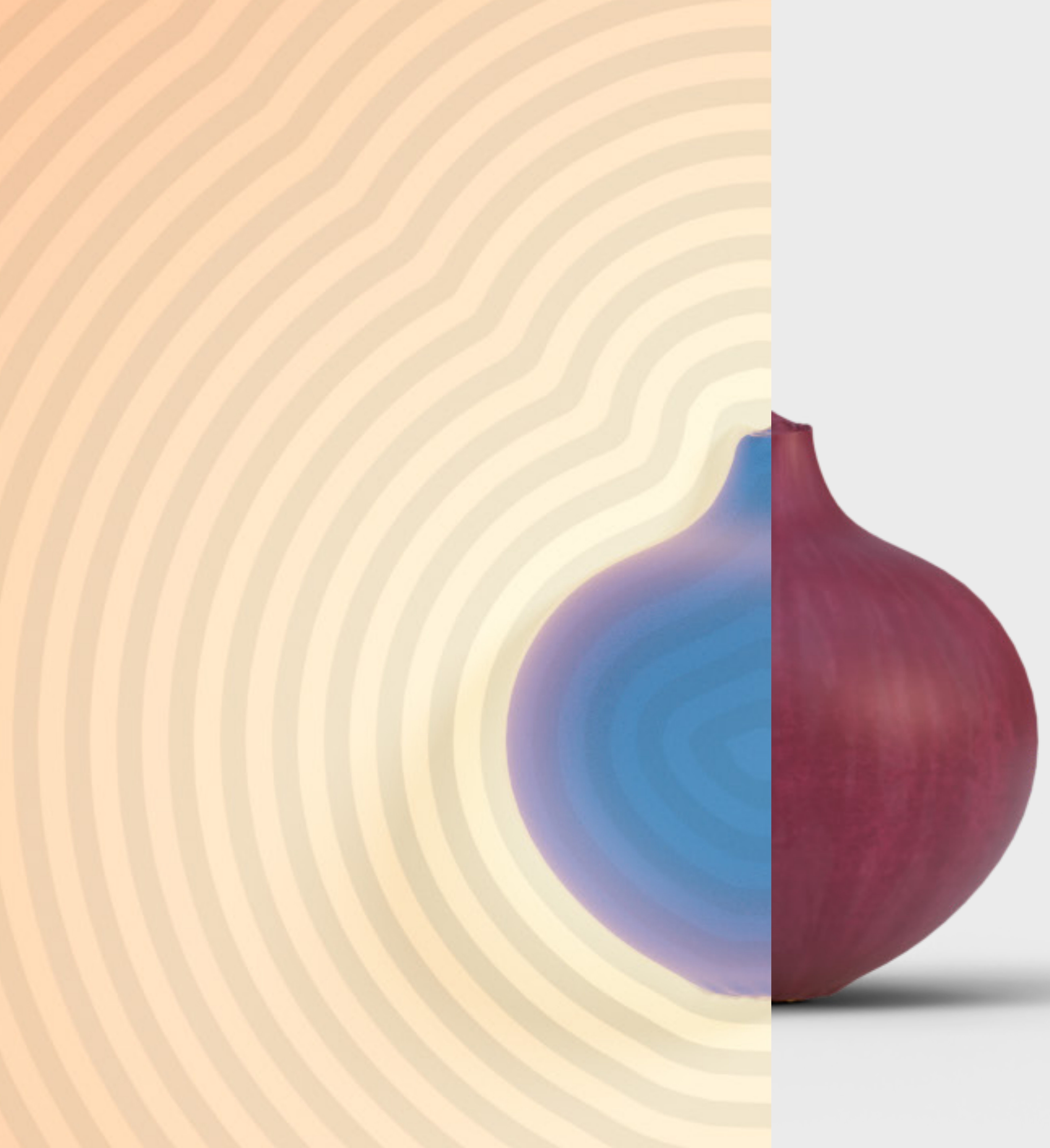



Synthesis
Editing
Analysis
Geometry
(e.g., neural implicits)



Goals: learn about

- what neural implicits are
- how & when to use them
- important techniques for working with them



OVERVIEW

IMPLICIT SURFACES

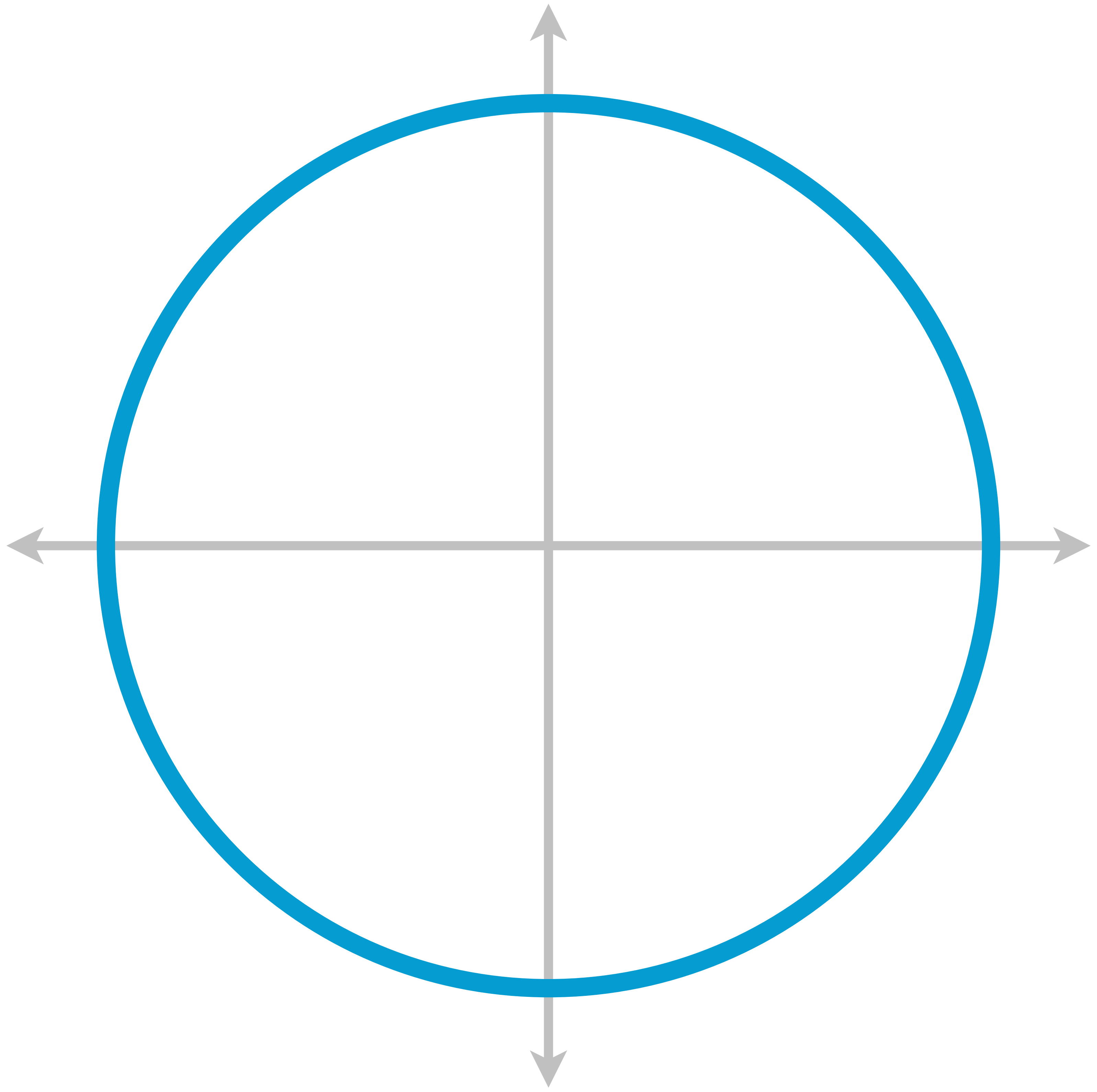
1. Background & History
2. Digital Representations

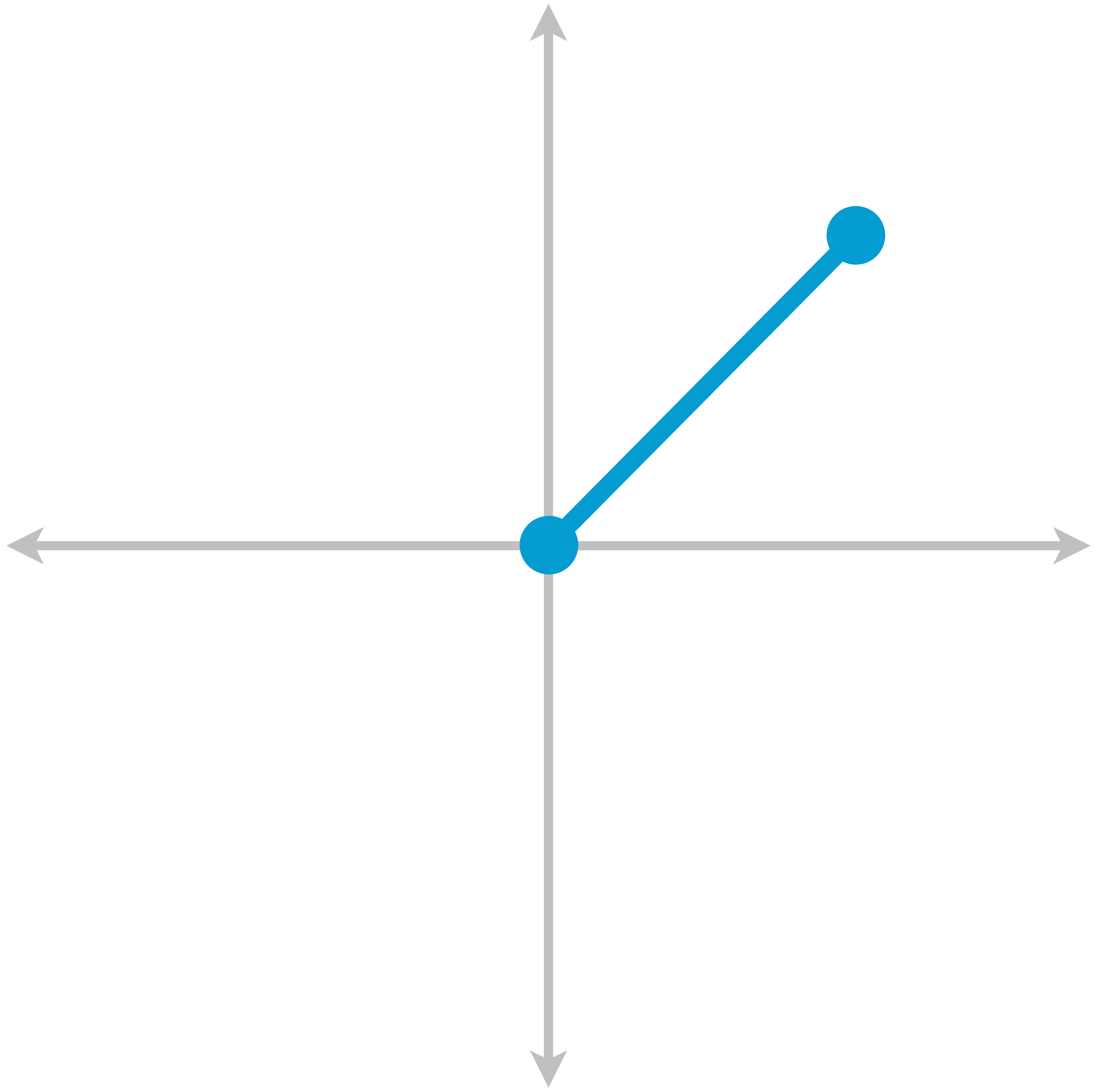
NEURAL IMPLICITS

3. The Basics
4. Why & Why Not Use Them?
5. Working out the Details
6. Geometric Operations

CONCLUSION

Implicit functions: defintion





IMPLICIT

$$x^2 + y^2 = r^2$$

EXPLICIT

$$f(t) = (1,1)t \quad t \in [0,1]$$

Implicit functions: defintion

IMPLICIT

$$f(x, y) : \mathbb{R}^2 \to \mathbb{R}$$
$$f(x, y) = x^2 + y^2 - r^2$$

Implicit functions: defintion

c-Level Sets

- surface defined by set of points where f is equal to c

$$f(x, y) = c$$

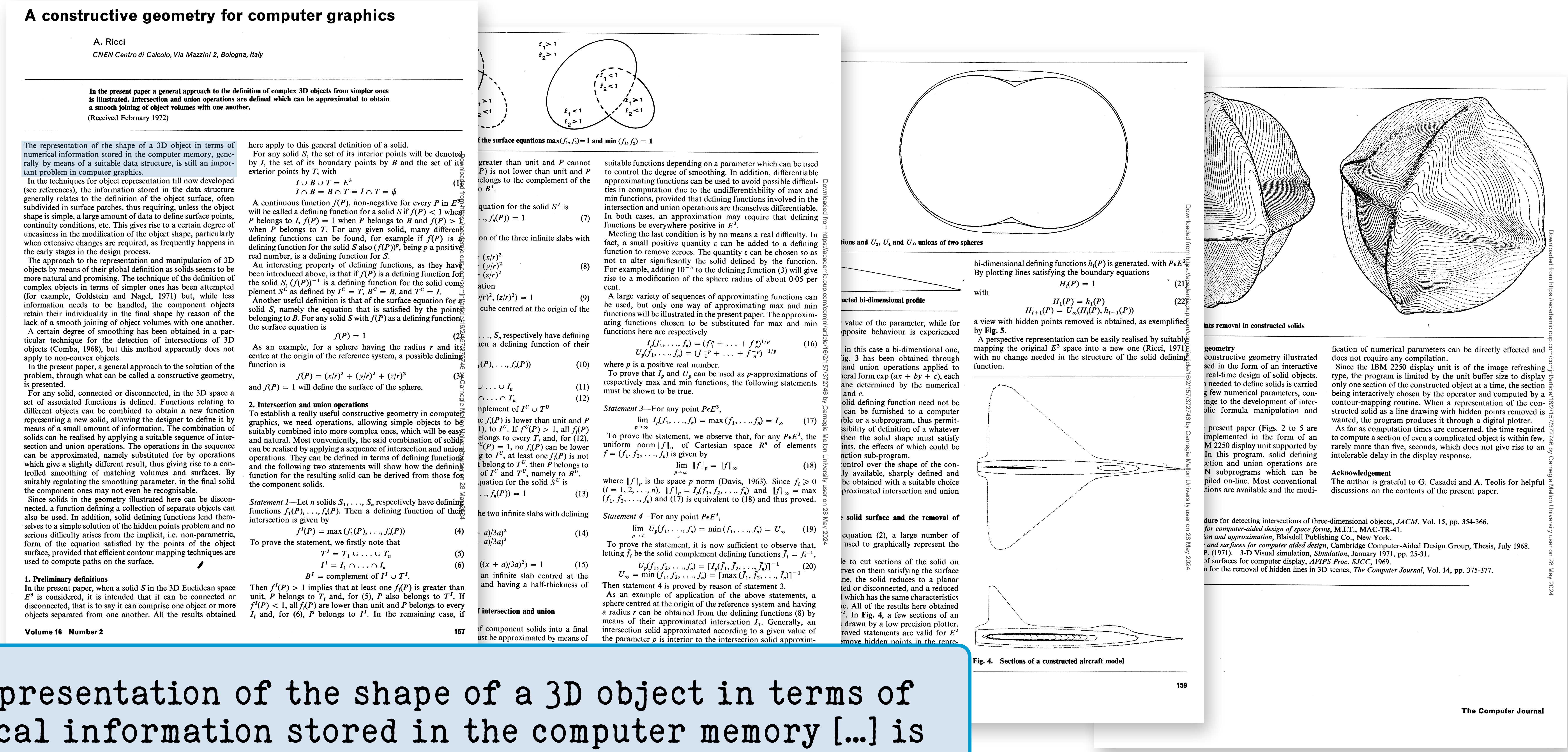
$c =$ **-1** ——— **2**

Implicit functions: defintion

c-Level Sets

- surface defined by set of points where f is equal to c
- typically, *zero-level set* is treated as the surface

Implicits in early graphics history



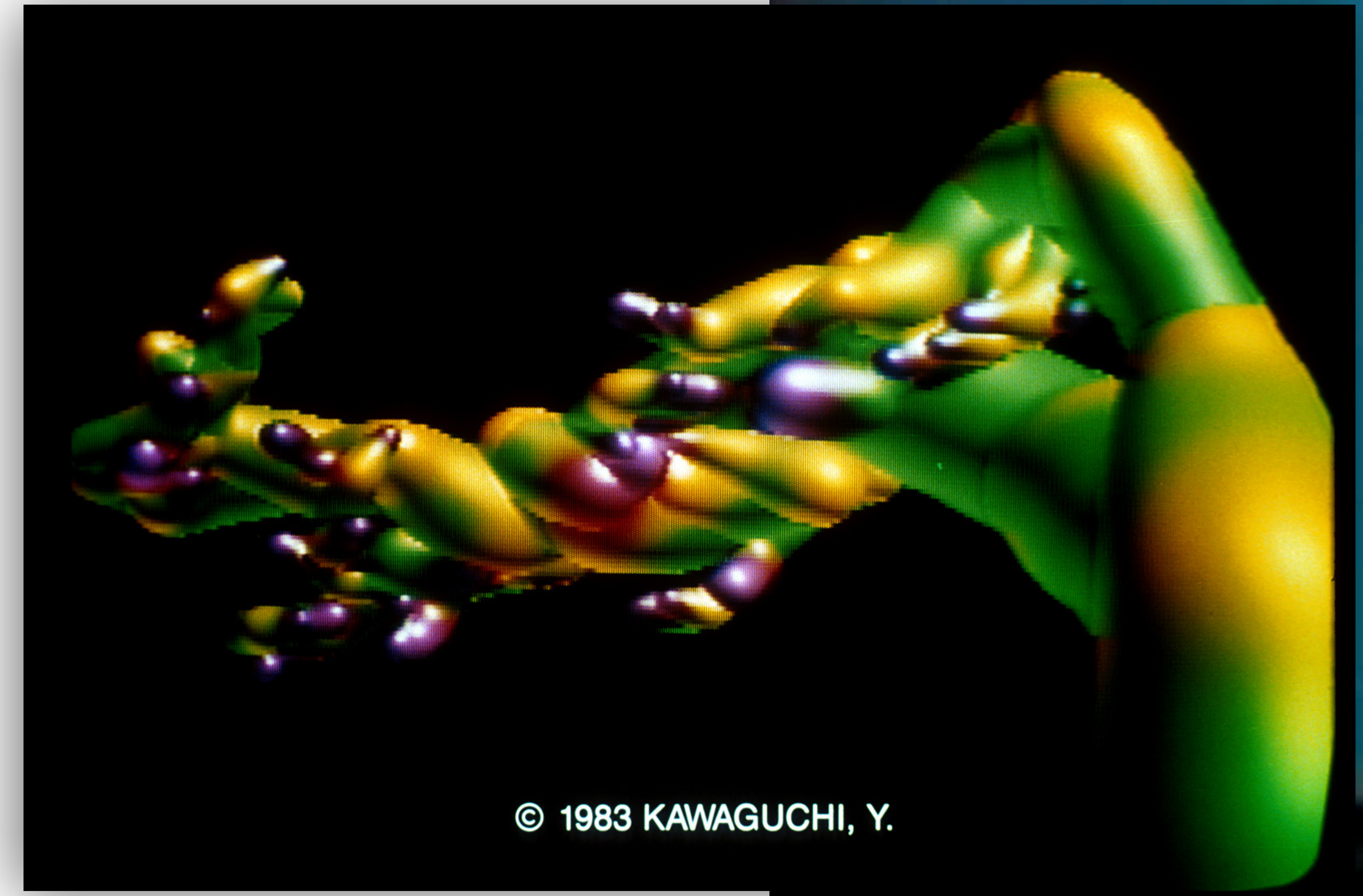
"The representation of the shape of a 3D object in terms of numerical information stored in the computer memory [...] is still an important problem in computer graphics."

Early use of implicit functions in graphics:
Ricci, "Constructive Geometry for Computer Graphics" (1973)

Implicits in early graphics history

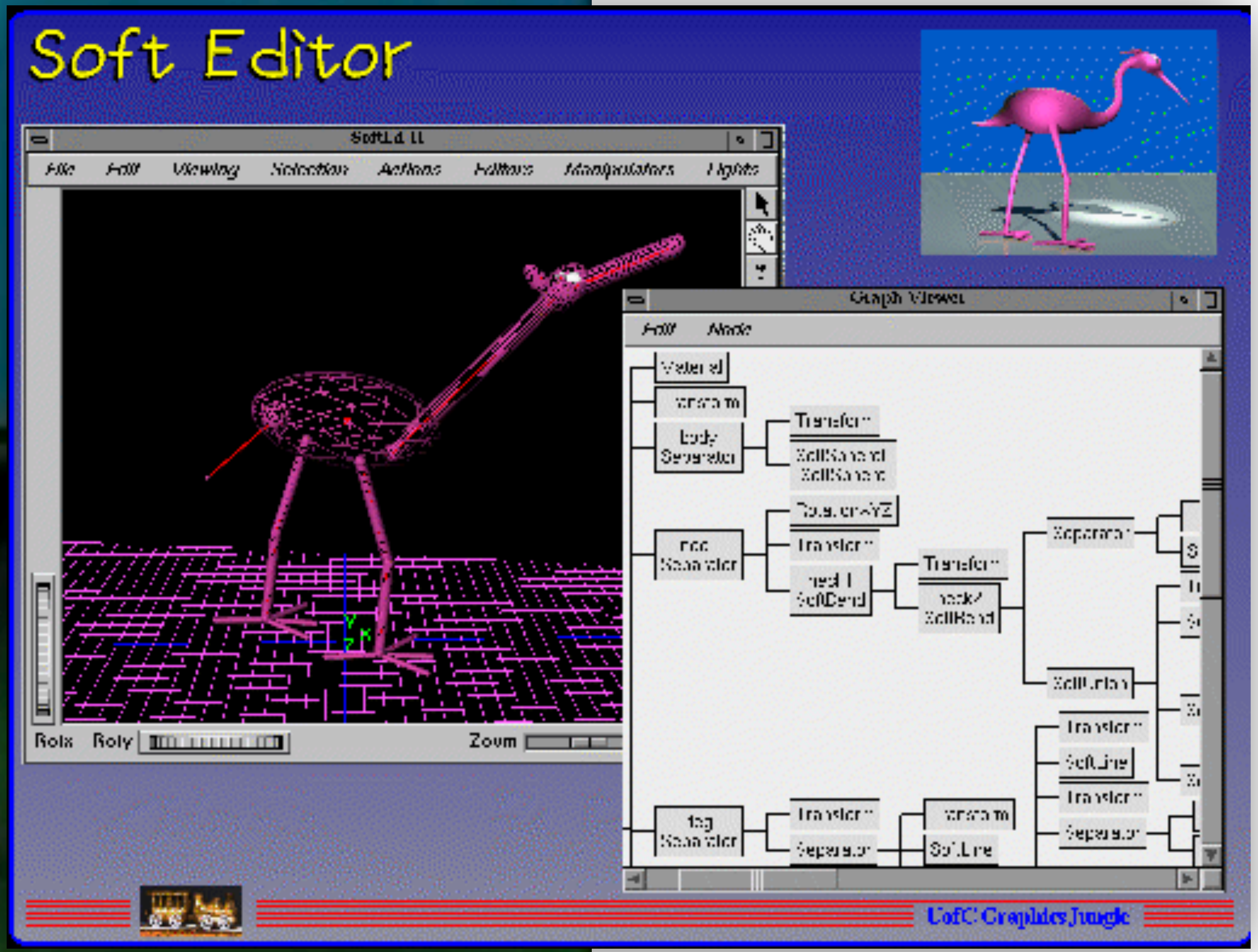
Jim Blinn's "Blobby molecules" (1982)

Implicits in early graphics history



"metaballs"

"soft objects"



For more background on implicits, see Chapter 21 of *Fundamentals of Computer Graphics*

Right image: Brian Wyvill

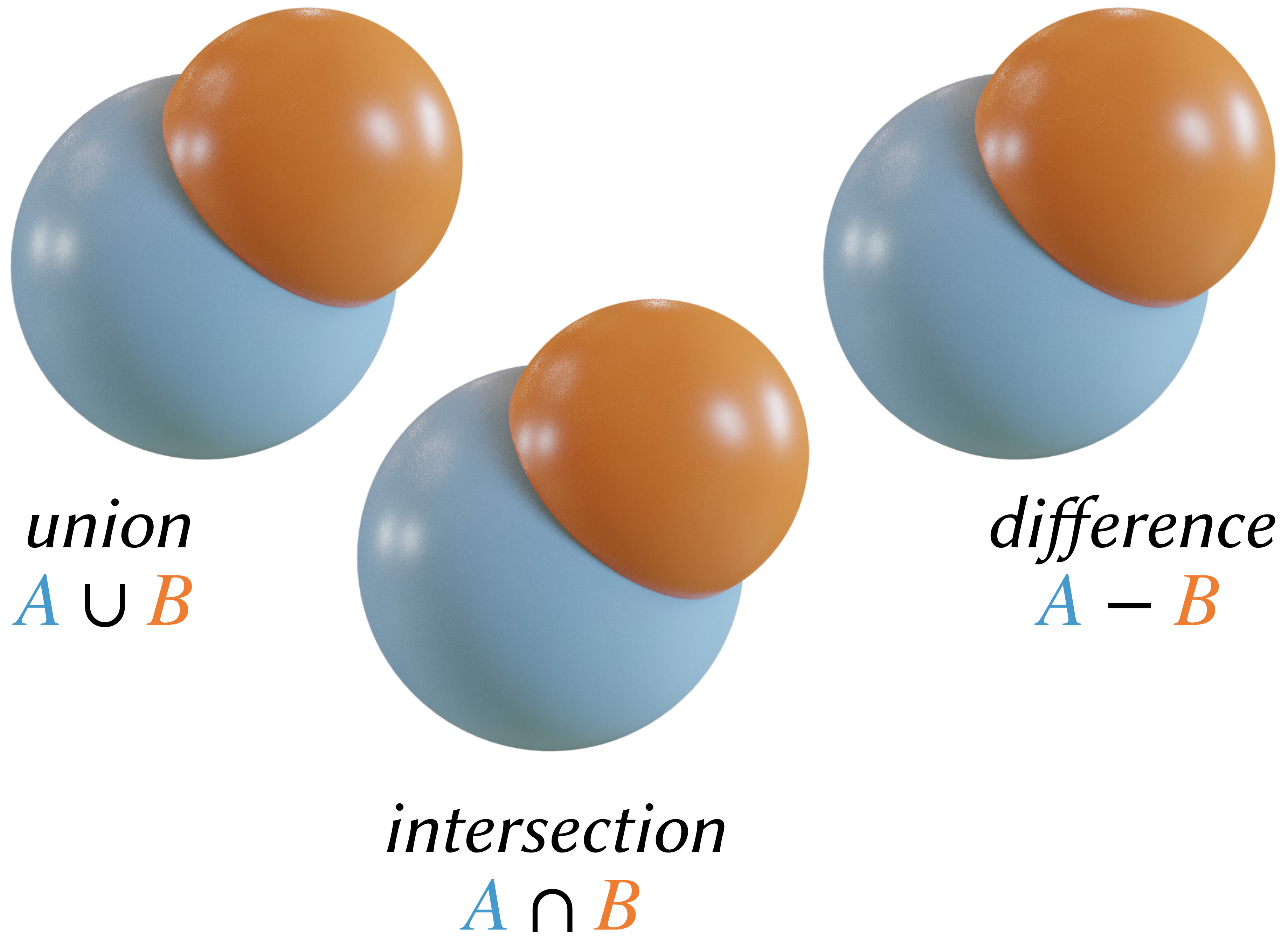
IMPLICIT SURFACES: BIG IDEAS

1. Constructive Solid Geometry
2. Taxonomy of implicits
3. Rendering
4. Surface Extraction

IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. Taxonomy of implicits
3. Rendering
4. Surface Extraction

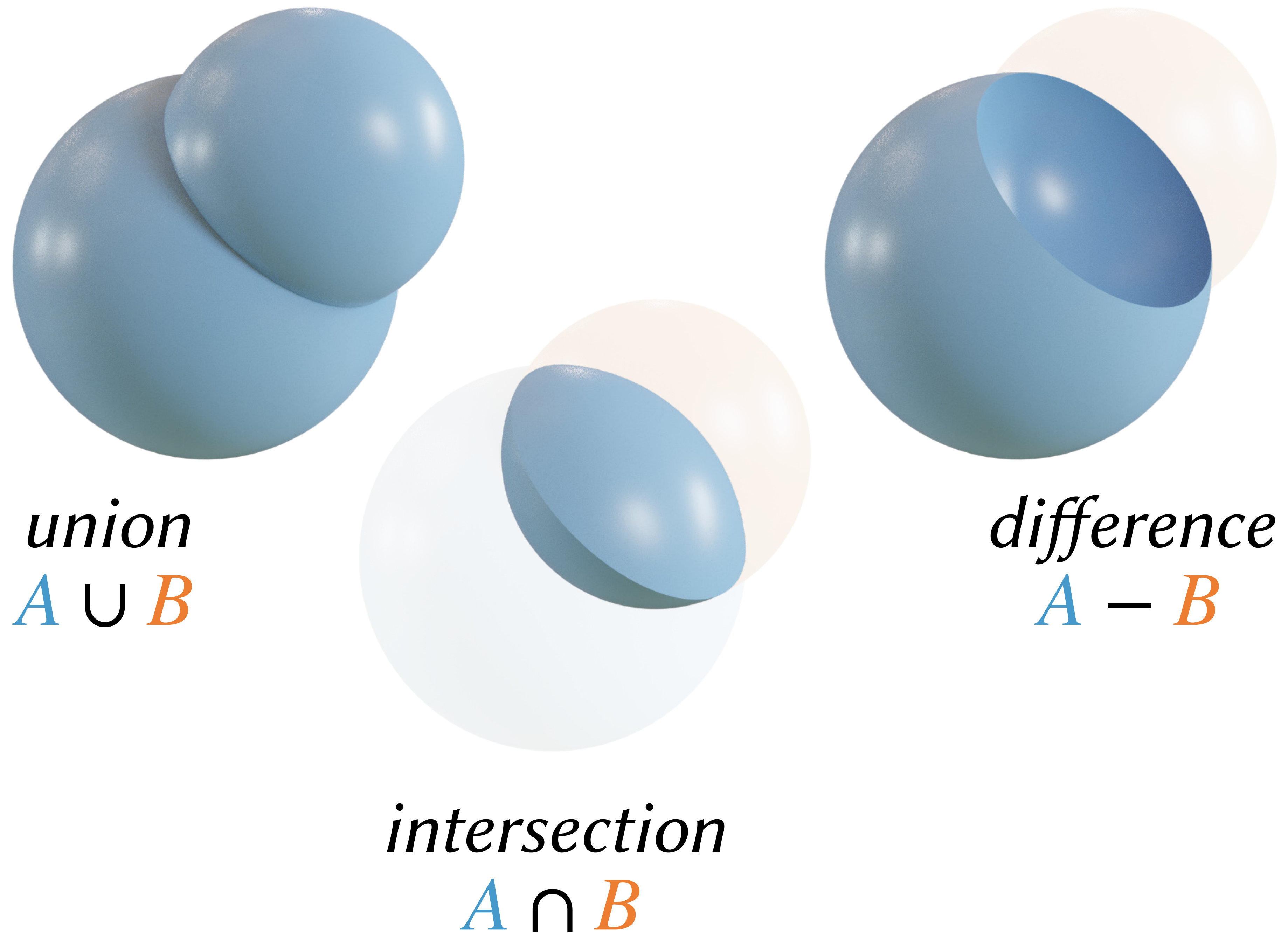
Constructive solid geometry



IMPLICIT SURFACES: BIG IDEAS

1. **CSG**
2. Taxonomy of implicits
3. Rendering
4. Surface Extraction

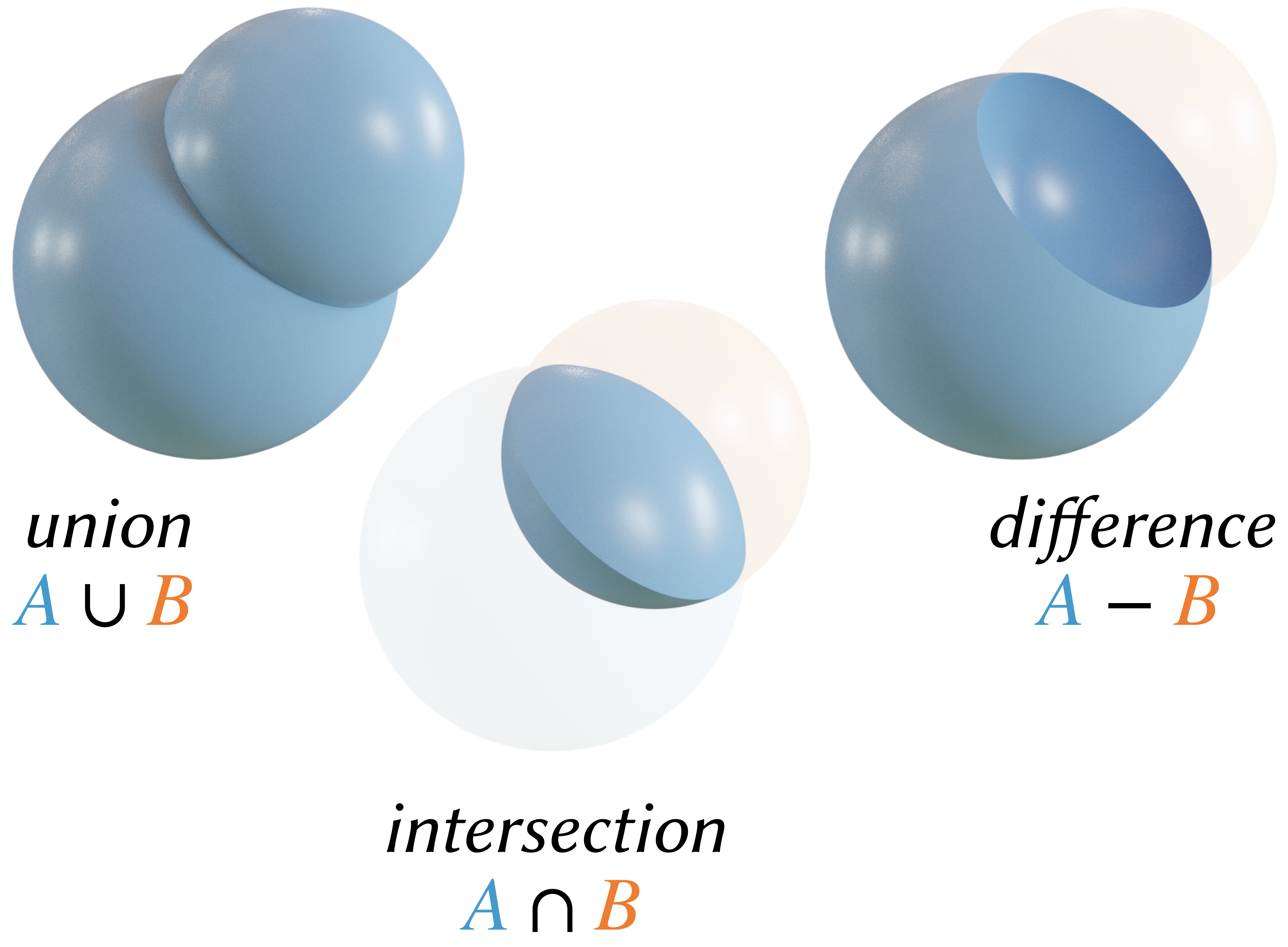
Constructive solid geometry

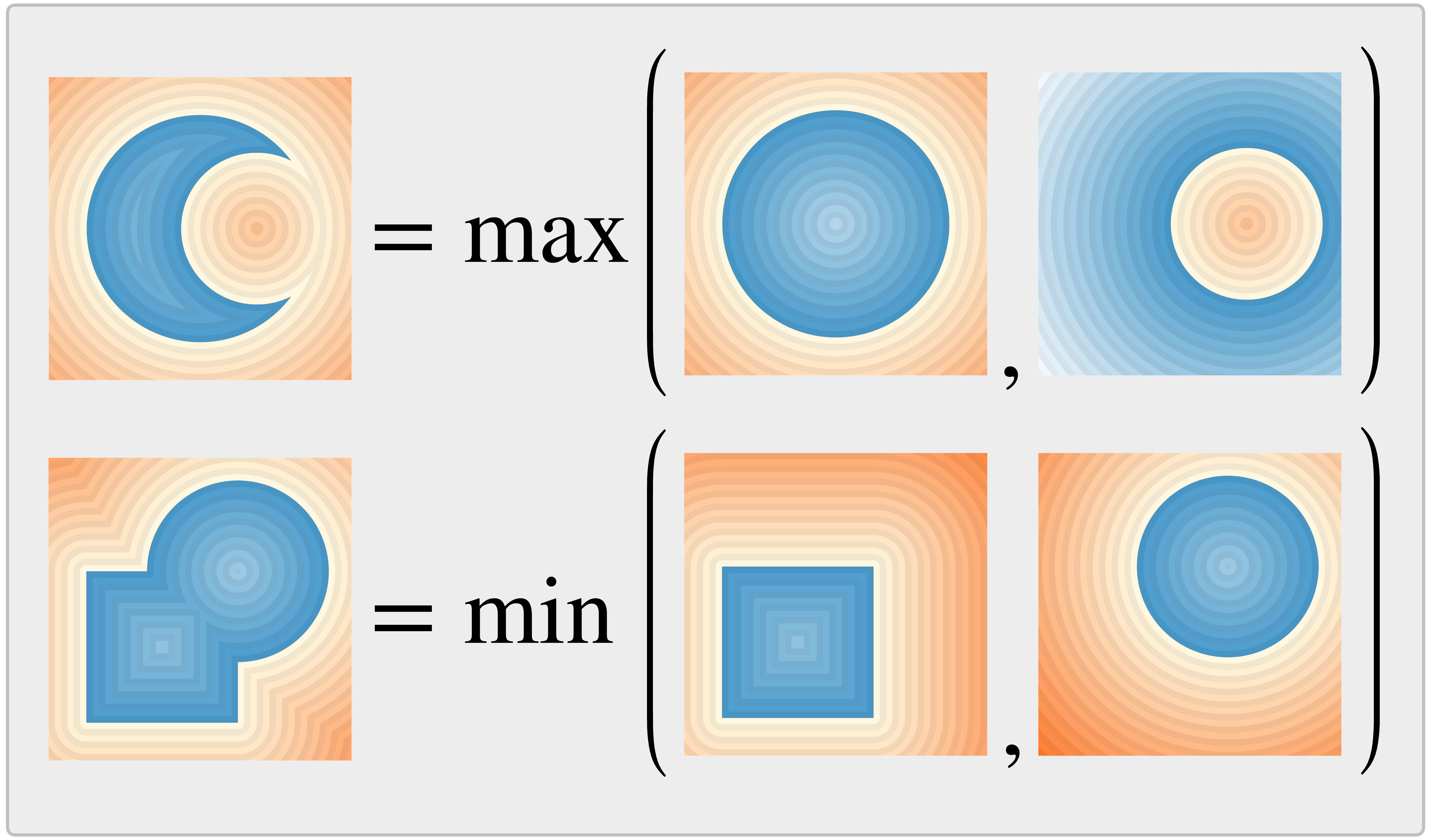


IMPLICIT SURFACES: BIG IDEAS

1. **CSG**
2. Taxonomy of implicits
3. Rendering
4. Surface Extraction

Constructive solid geometry

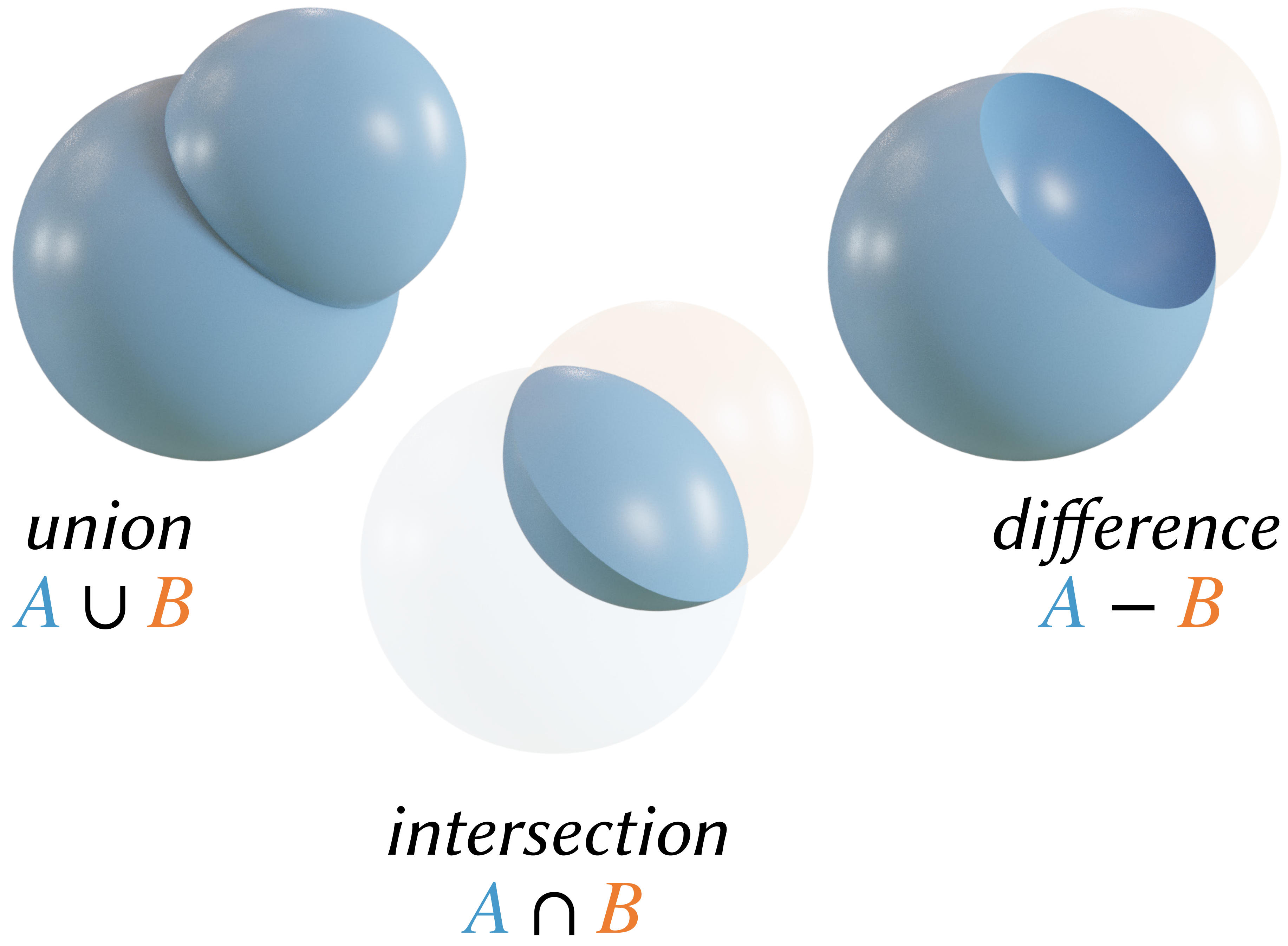




IMPLICIT SURFACES: BIG IDEAS

1. **CSG**
2. Taxonomy of implicits
3. Rendering
4. Surface Extraction

Constructive solid geometry



Modeling with implicits

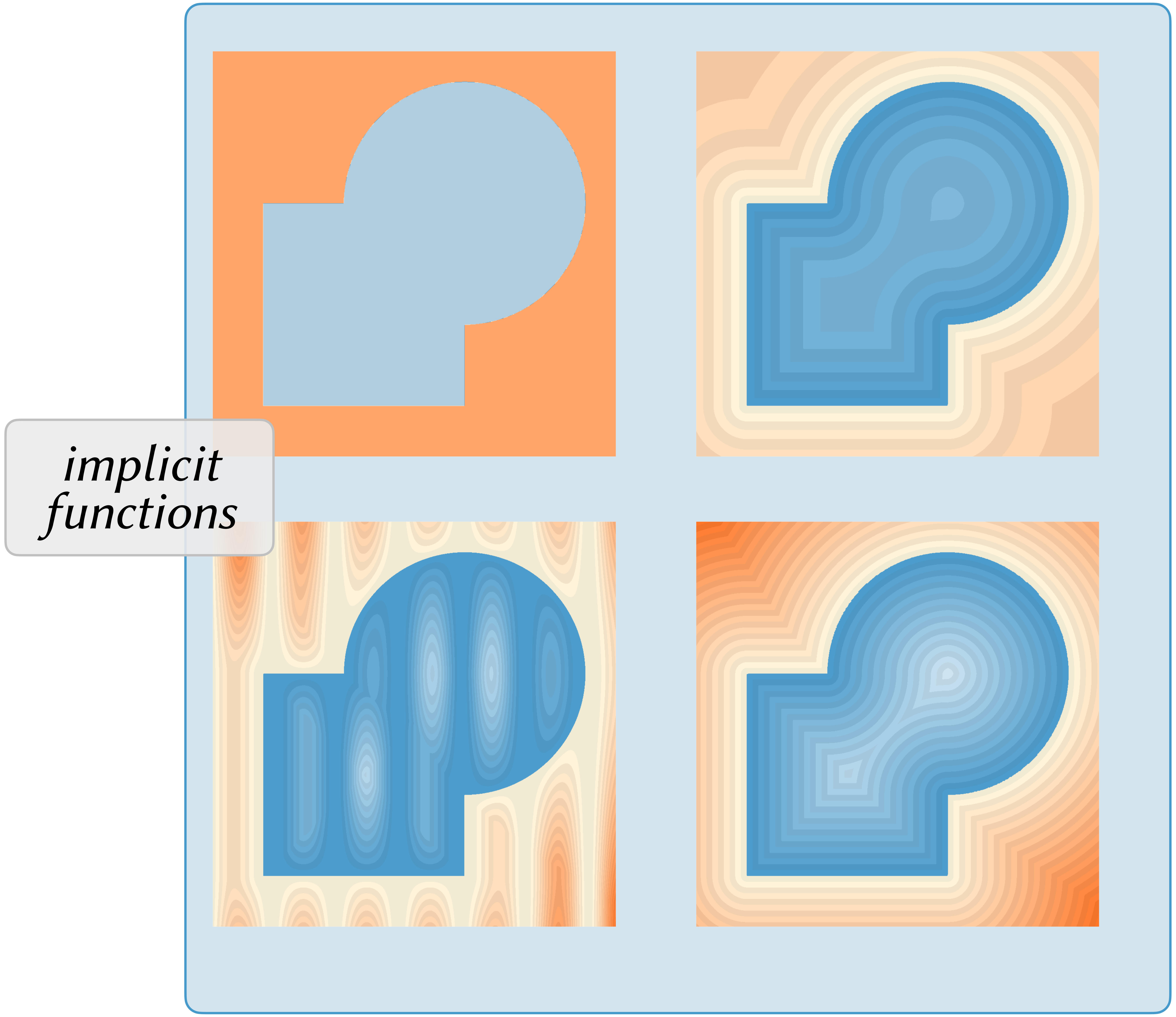
Complex shapes from simple primitives!

IMPLICIT SURFACES: BIG IDEAS

1. **CSG**
2. Taxonomy of implicits
3. Rendering
4. Surface Extraction

📖 For more amazing implicit function art, see Inigo Quilez's blog

Video: Inigo Quilez

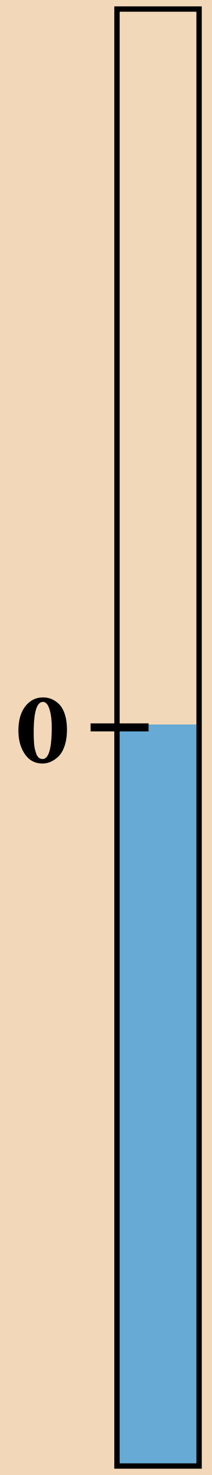


IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. **Taxonomy of implicits**
3. Rendering
4. Surface Extraction

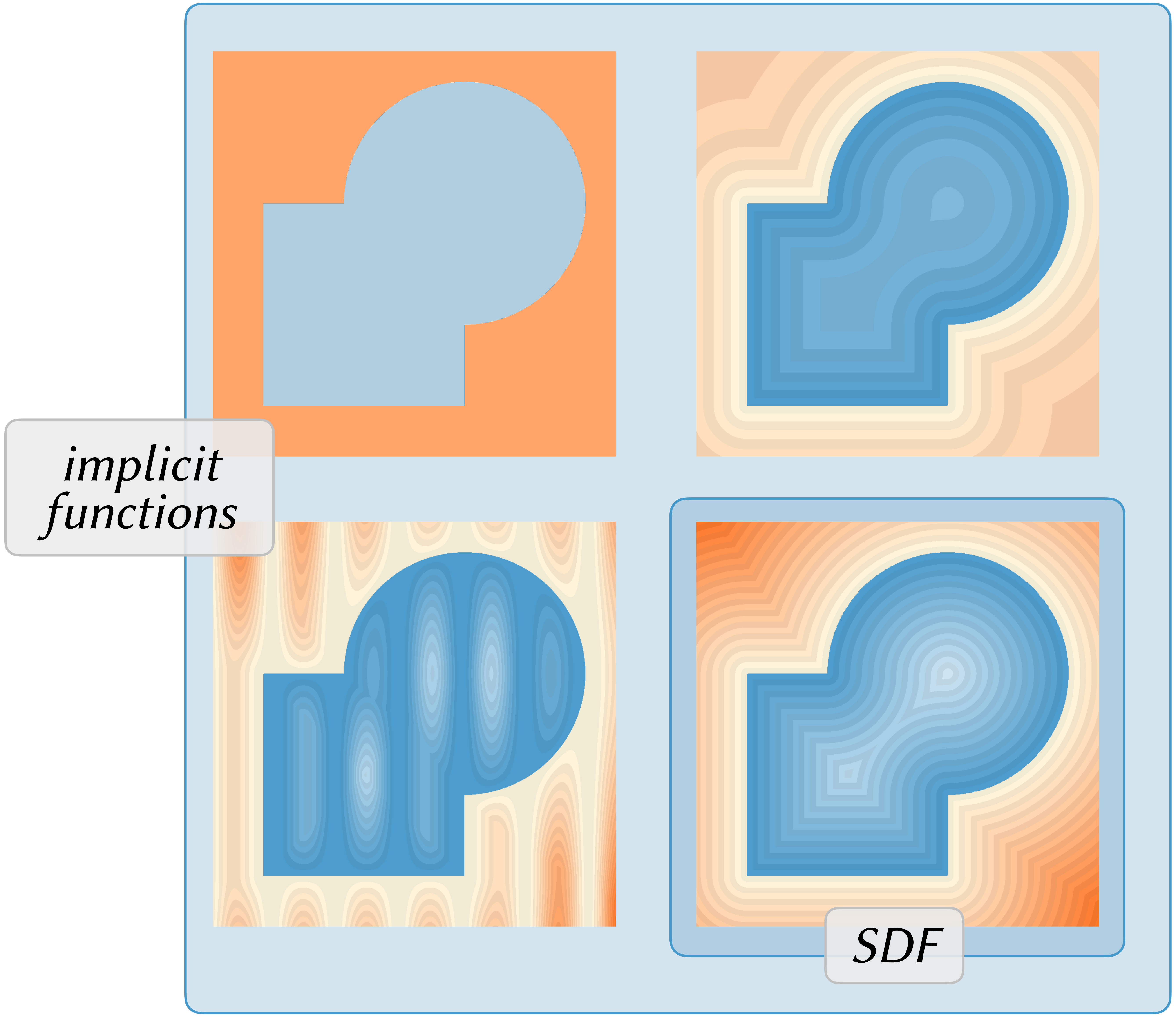
Indicator Function

$$f(\boldsymbol{x}) = \begin{cases} -1 & \boldsymbol{x} \in \text{shape} \\ 1 & \boldsymbol{x} \notin \text{shape} \end{cases}$$



IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. **Taxonomy of implicits**
3. Rendering
4. Surface Extraction



IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. **Taxonomy of implicits**
3. Rendering
4. Surface Extraction

Signed Distance Function

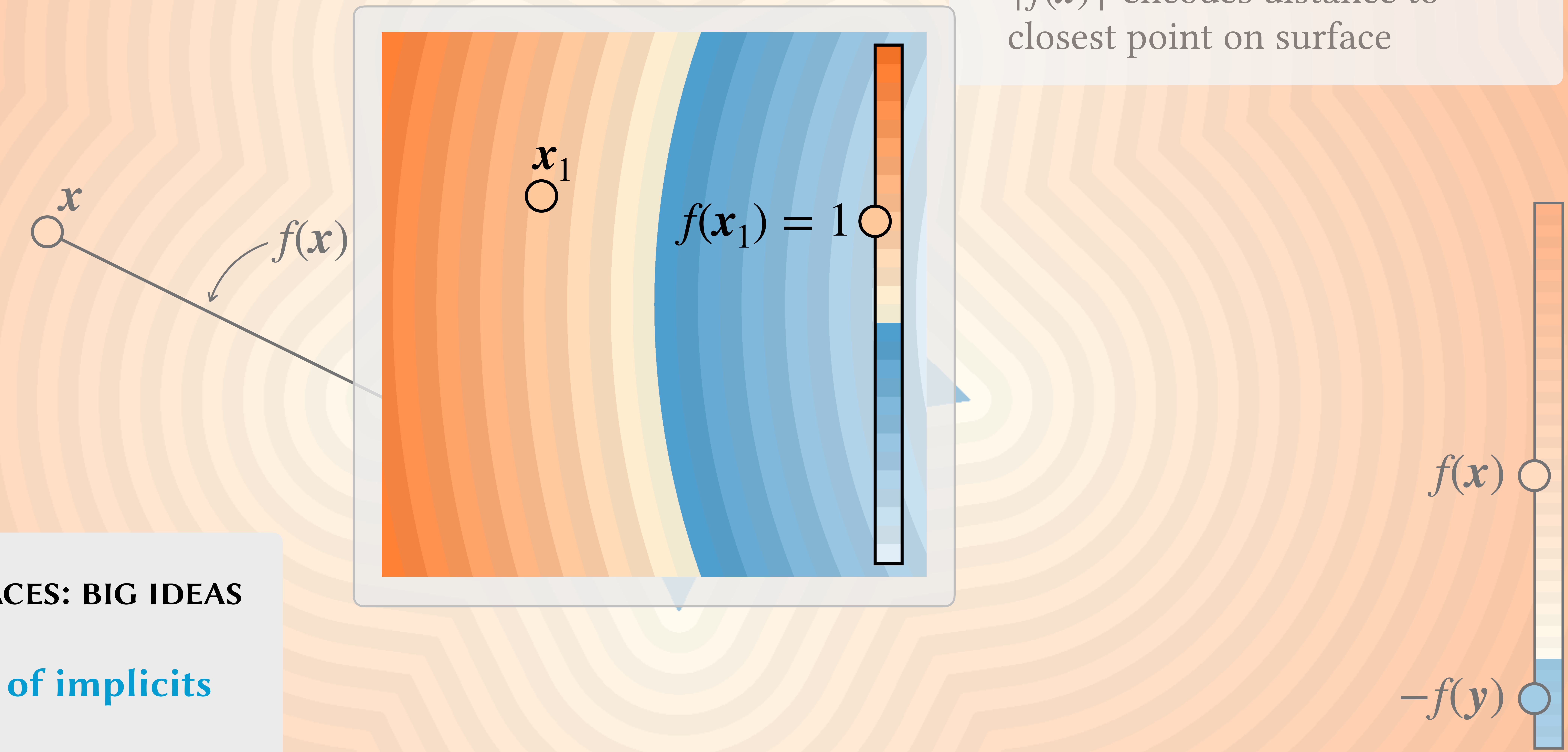
- $|f(\boldsymbol{x})|$ encodes distance to closest point on surface

IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. Taxonomy of implicits
3. Rendering
4. Surface Extraction

Signed Distance Function

- $|f(\boldsymbol{x})|$ encodes distance to closest point on surface

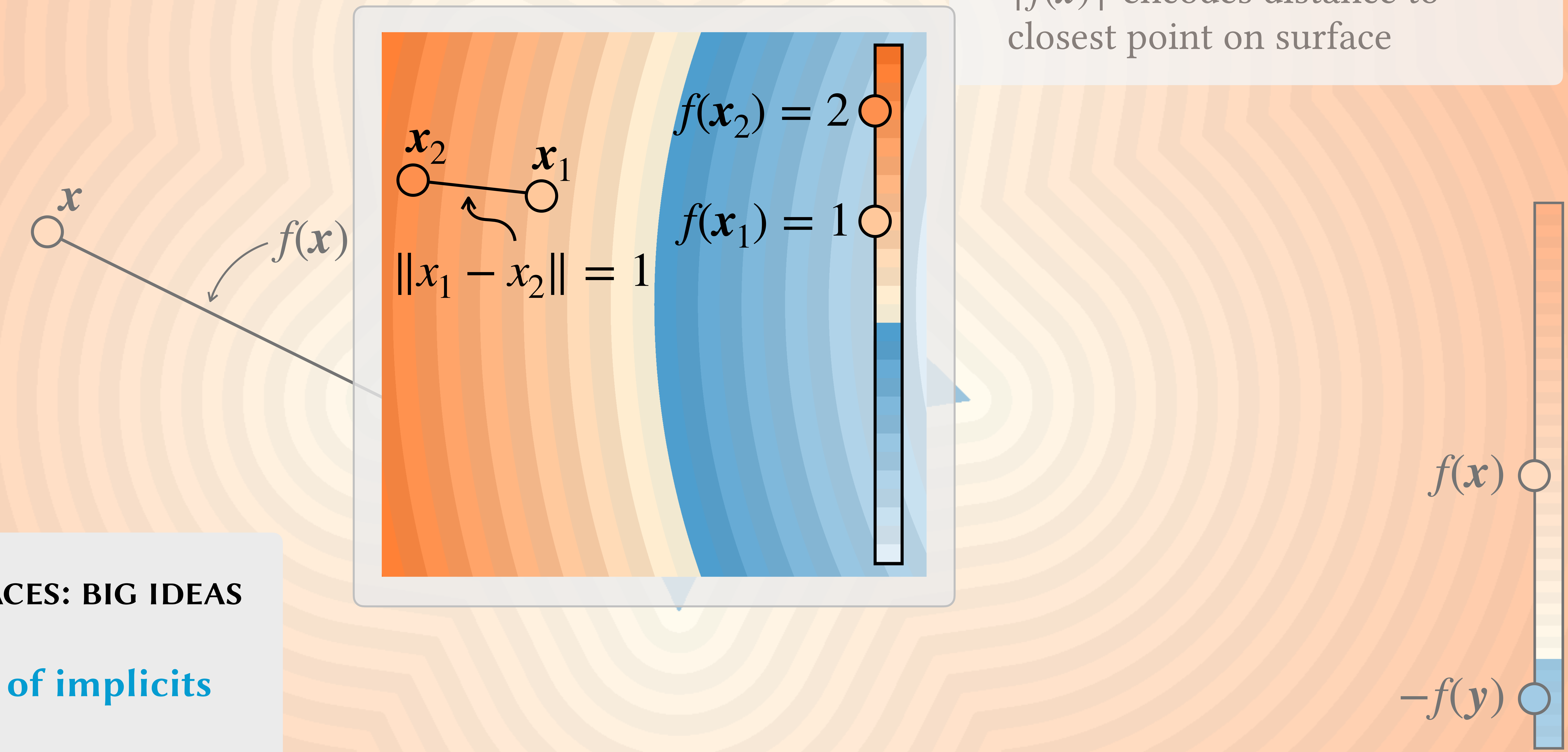


IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. **Taxonomy of implicits**
3. Rendering
4. Surface Extraction

Signed Distance Function

- $|f(\boldsymbol{x})|$ encodes distance to closest point on surface

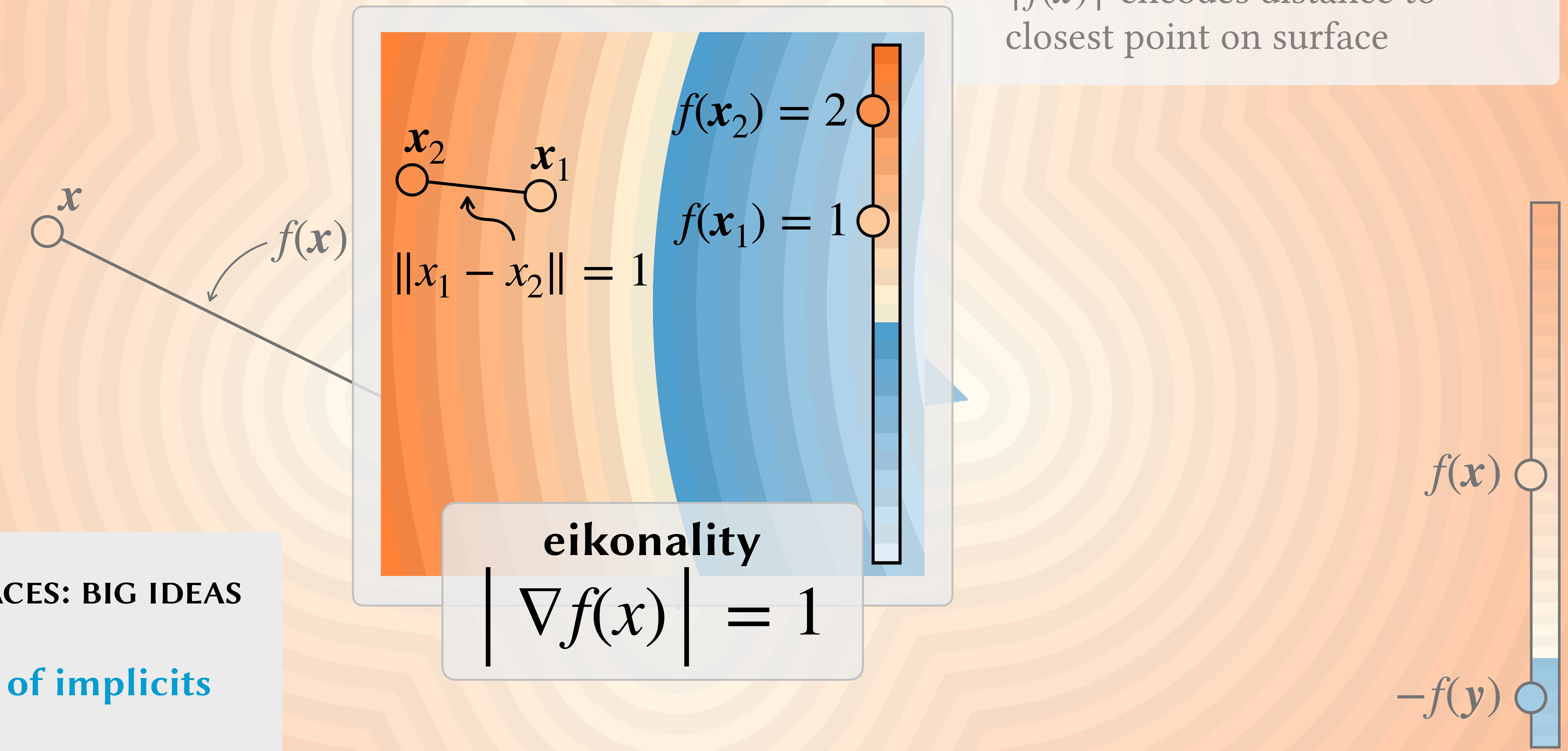


IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. **Taxonomy of implicits**
3. Rendering
4. Surface Extraction

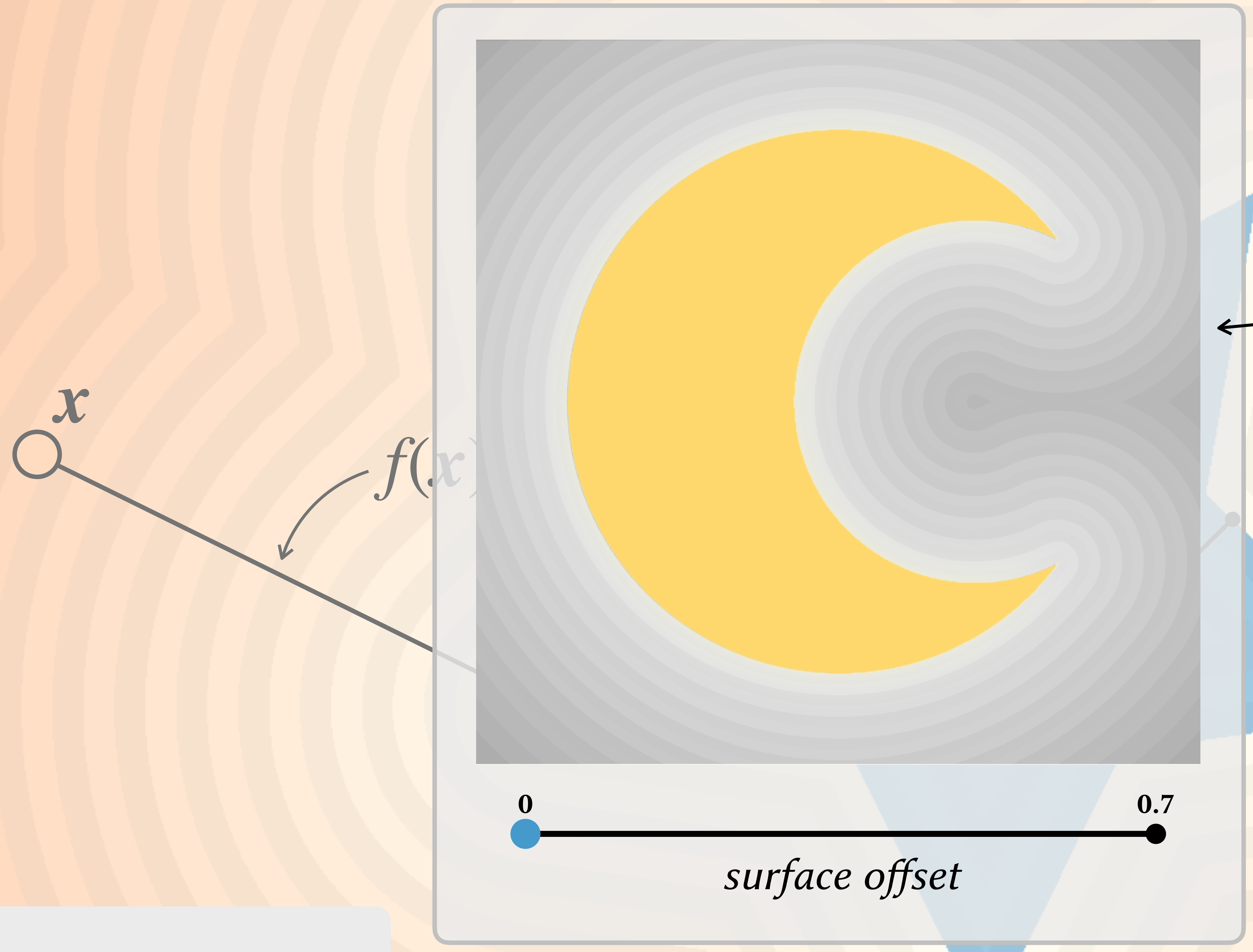
Signed Distance Function

- $|f(\boldsymbol{x})|$ encodes distance to closest point on surface



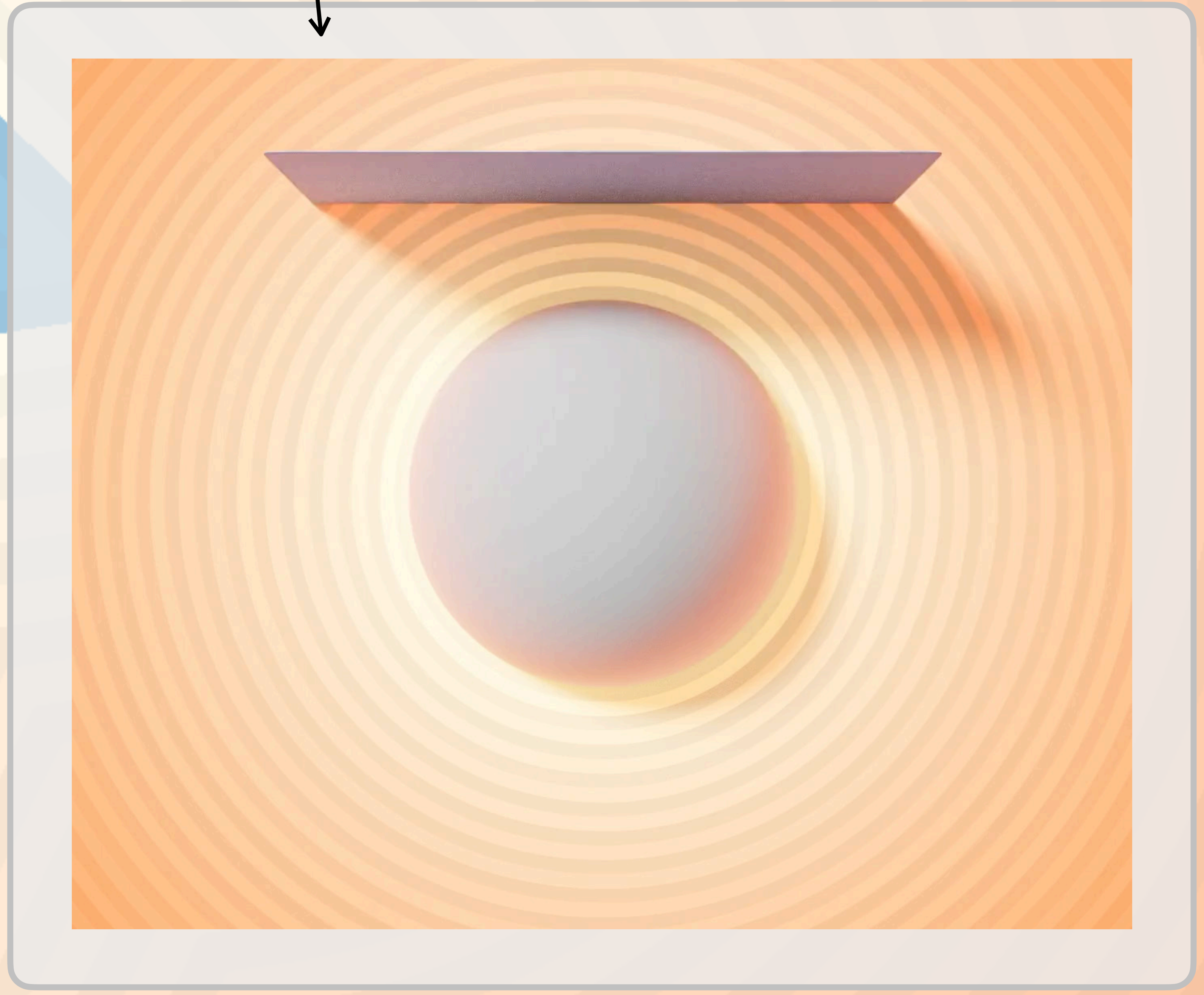
IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. **Taxonomy of implicits**
3. Rendering
4. Surface Extraction



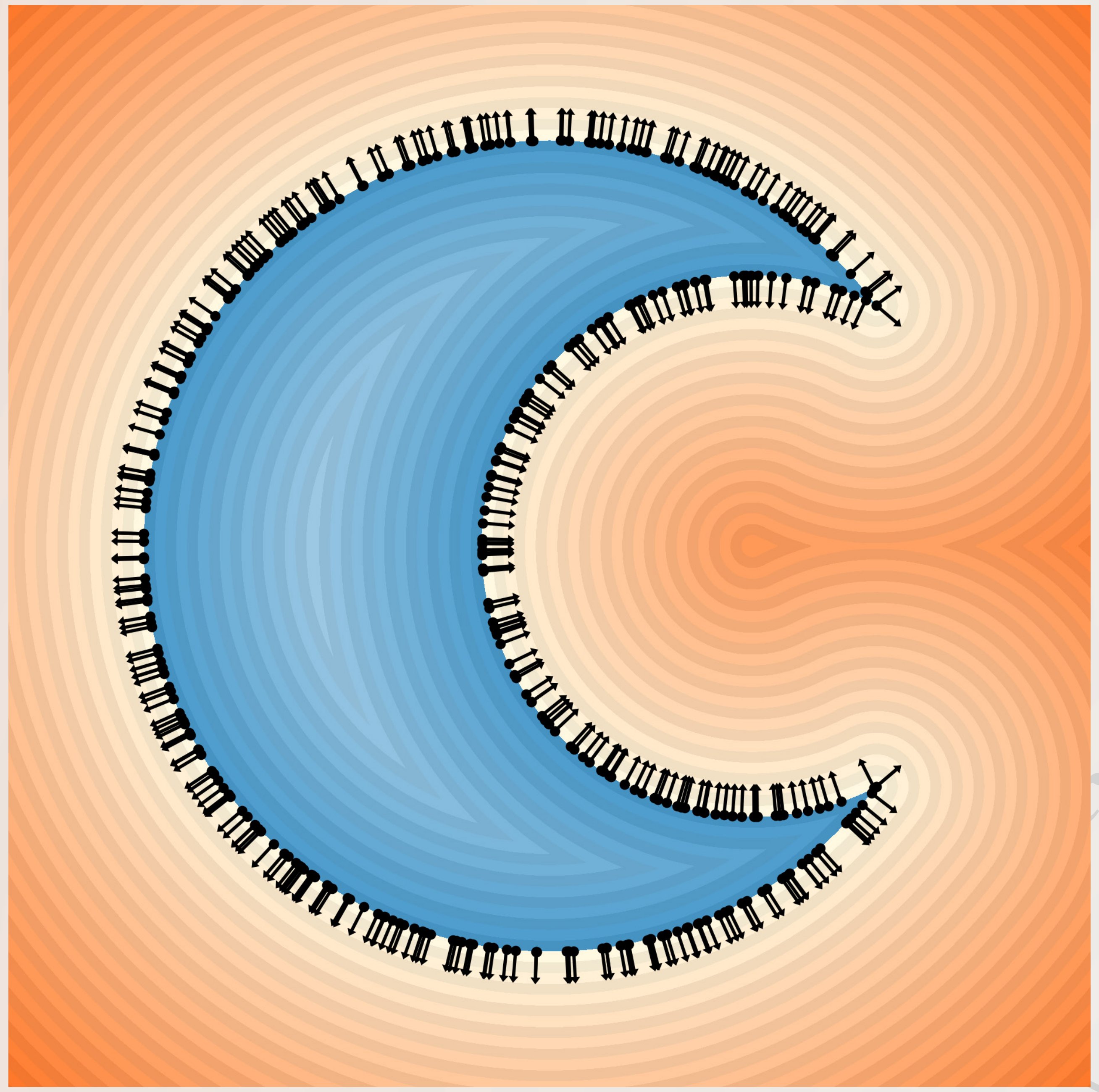
Signed Distance Function

- $|f(\boldsymbol{x})|$ encodes distance to closest point on surface
- makes many tasks *way* easier ☺
  - *e.g., offset surfaces, distance checks for simulation, etc.*

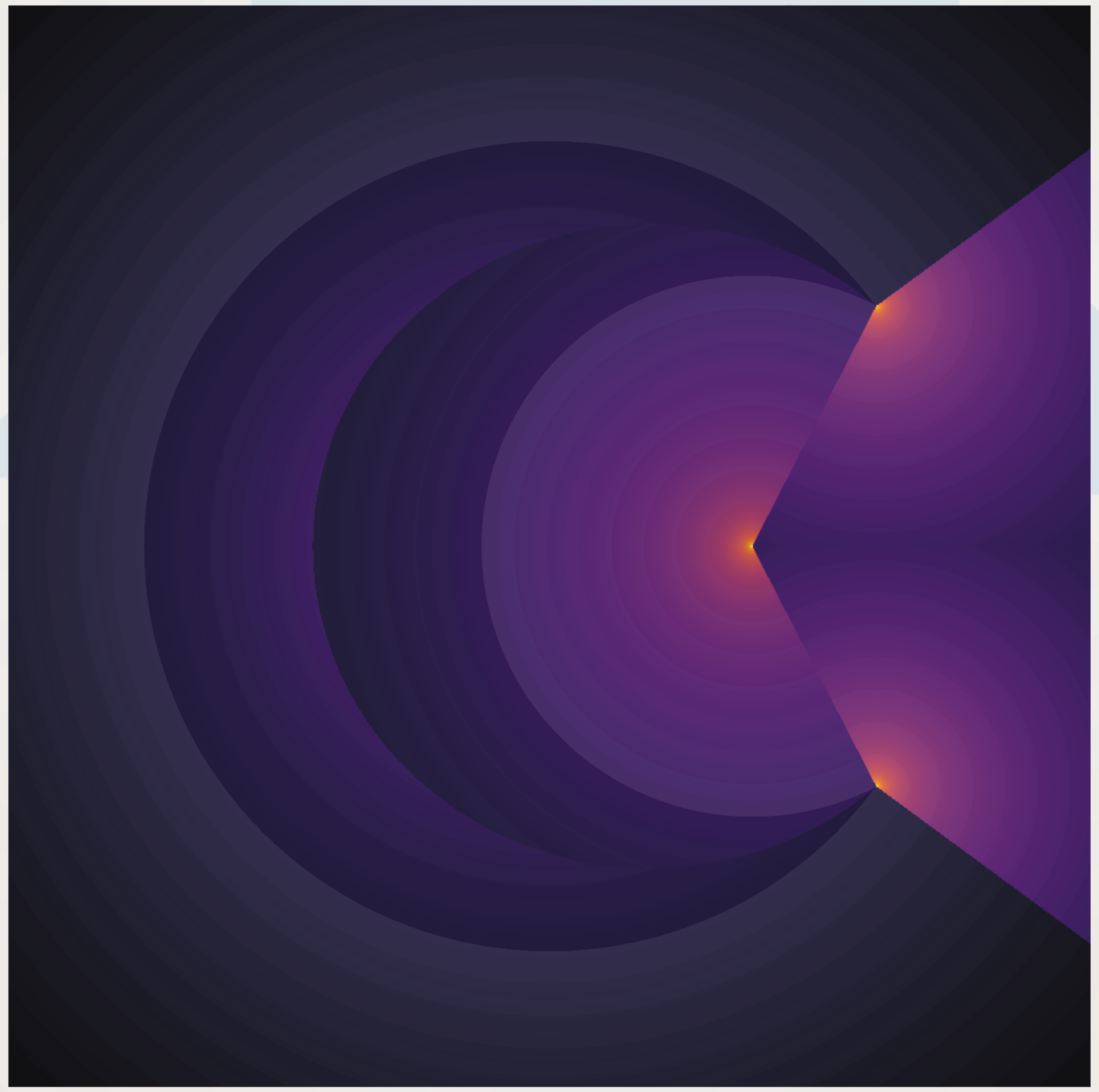


IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. **Taxonomy of implicits**
3. Rendering
4. Surface Extraction



Surface normals (∇f)



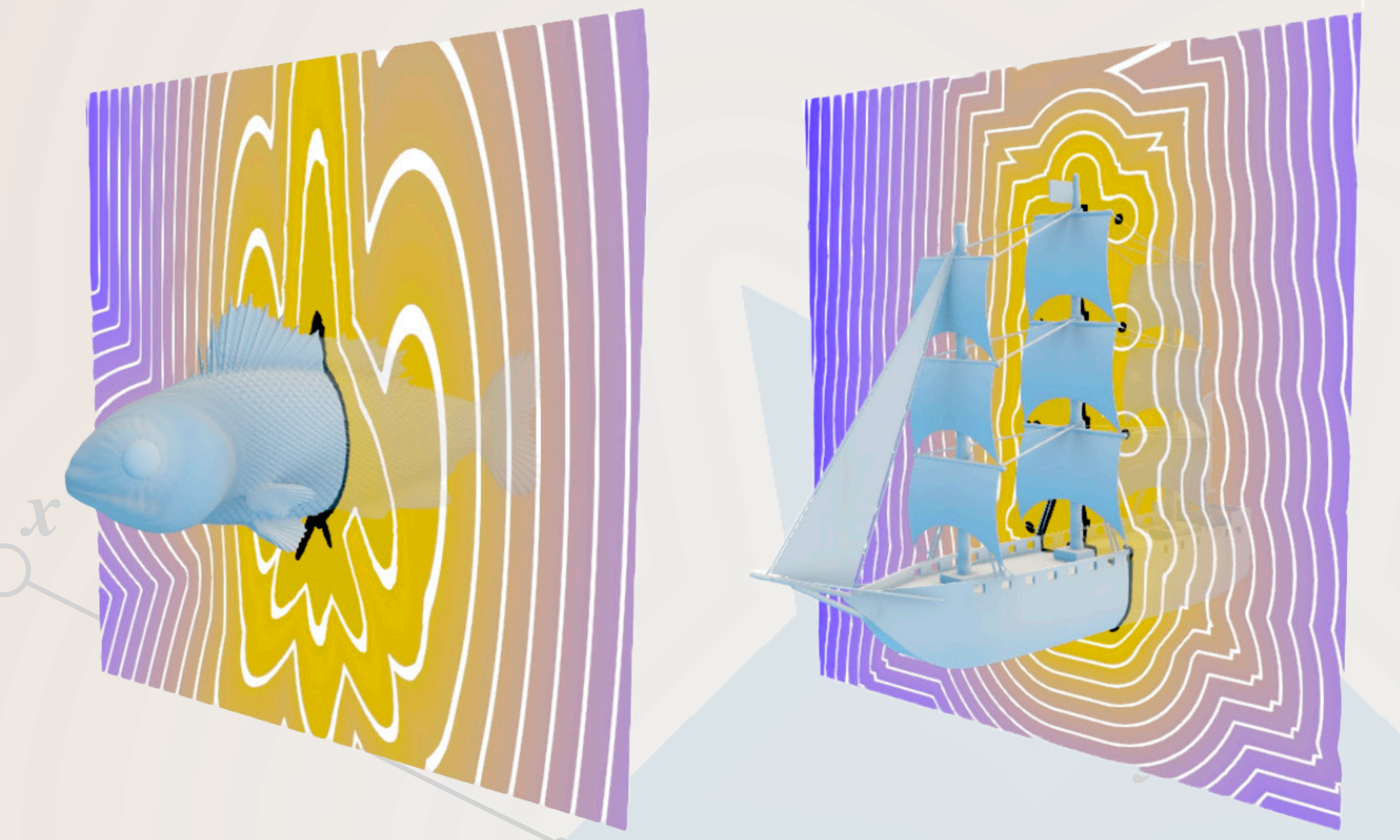
Mean curvature (Δf)

Signed Distance Function

- $|f(\boldsymbol{x})|$ encodes distance to closest point on surface
- makes many tasks *way* easier ☺
  - *e.g., offset surfaces, distance checks for simulation, etc.*
  - *e.g., geometric quantities*

IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. **Taxonomy of implicits**
3. Rendering
4. Surface Extraction



Real-world "SDFs" [Takikawa et al., 2022]

Signed Distance Function

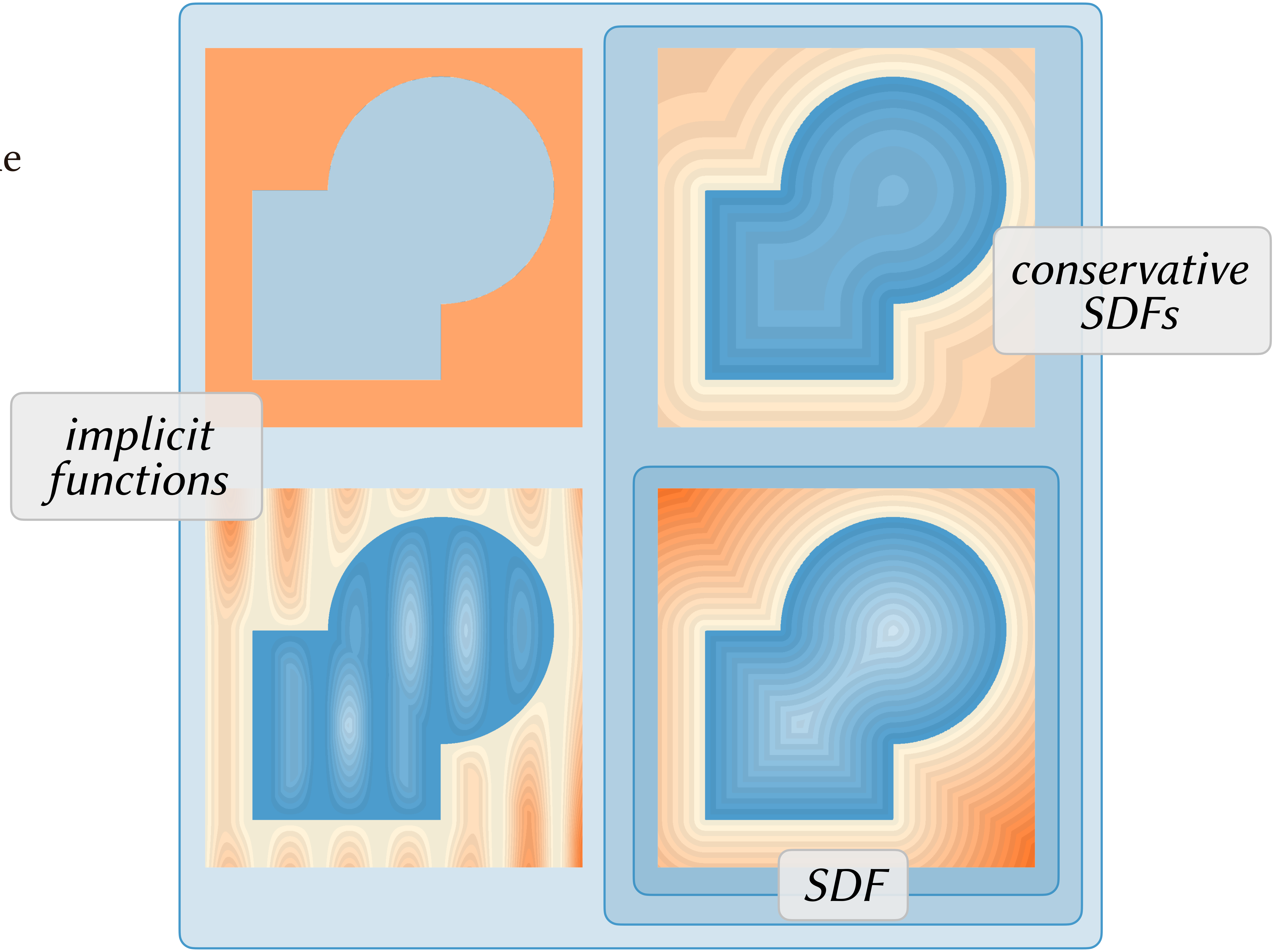
- $|f(\boldsymbol{x})|$ encodes distance to closest point on surface
- makes many tasks *way* easier ☺
- distance property is hard to maintain ☹
  - *e.g., not preserved by most editing operations*

IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. **Taxonomy of implicits**
3. Rendering
4. Surface Extraction

Conservative SDFs

- (aka approximate SDFs)
- $|f(\boldsymbol{x})|$ is less than the distance to the closest point on the surface

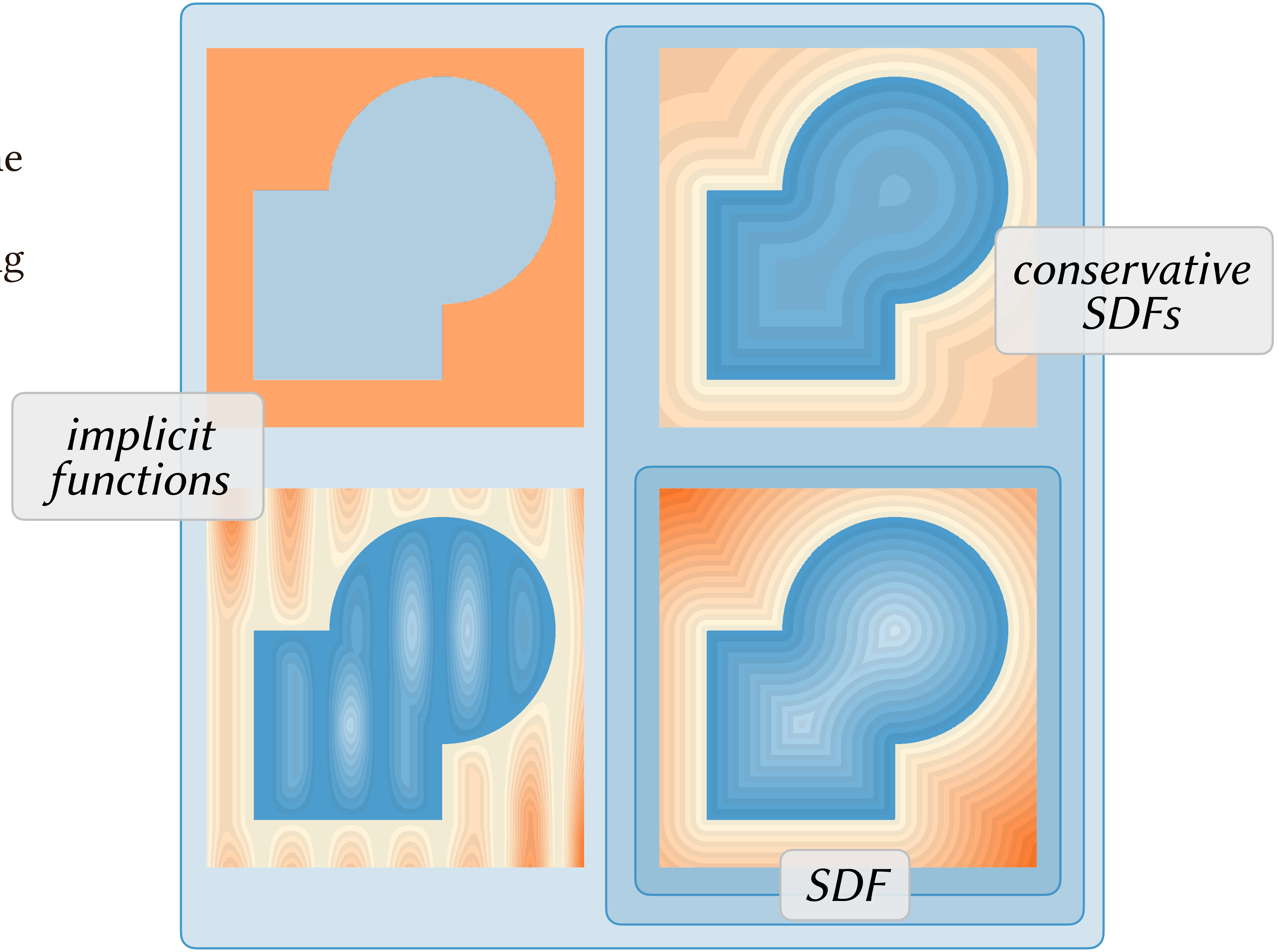


IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. **Taxonomy of implicits**
3. Rendering
4. Surface Extraction

Conservative SDFs

- (aka approximate SDFs)
- $|f(\boldsymbol{x})|$ is less than the distance to the closest point on the surface
- tradeoff between ease of maintaining & having useful properties

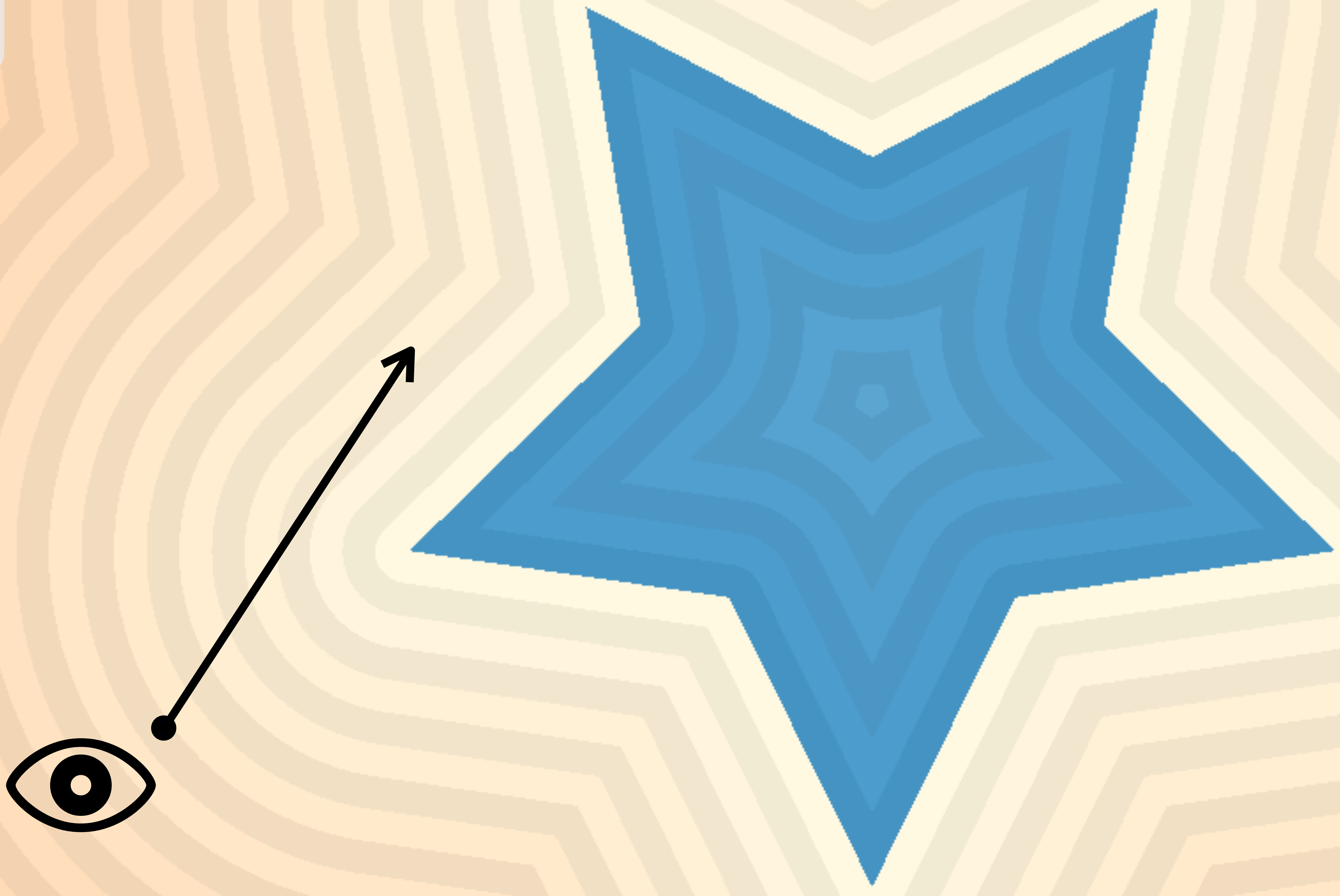


IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. **Taxonomy of implicits**
3. Rendering
4. Surface Extraction

Sphere Tracing

- introduced by [Hart, 1996]
- technique to ray trace conservative SDFs

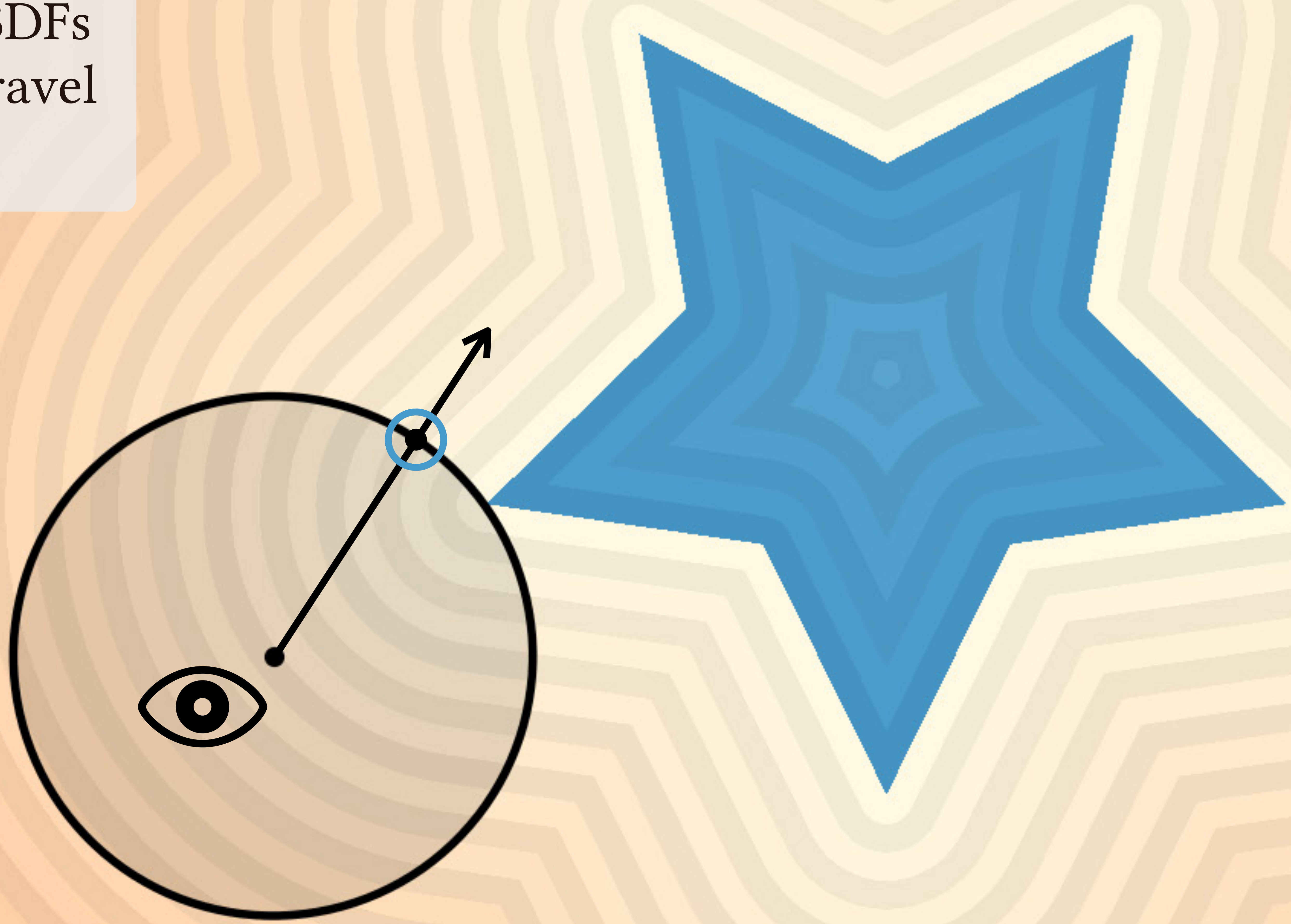


IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. Taxonomy of implicits
3. **Rendering**
4. Surface Extraction

Sphere Tracing

- introduced by [Hart, 1996]
- technique to ray trace conservative SDFs
- $|f(x)|$ is distance we can *definitely* travel without crossing the surface

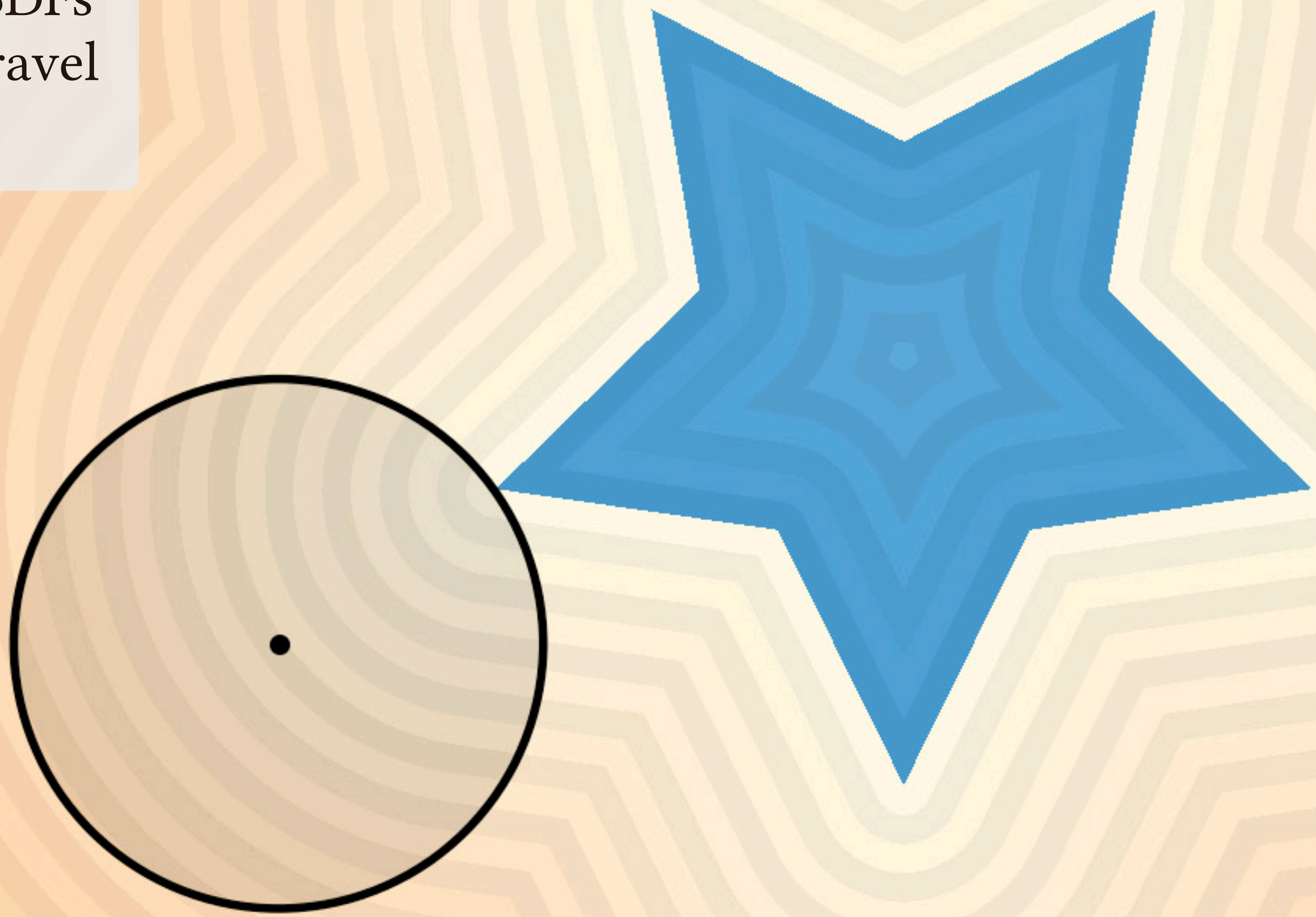


IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. Taxonomy of implicits
3. **Rendering**
4. Surface Extraction

Sphere Tracing

- introduced by [Hart, 1996]
- technique to ray trace conservative SDFs
- $|f(x)|$ is distance we can *definitely* travel without crossing the surface



IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. Taxonomy of implicits
3. **Rendering**
4. Surface Extraction

Marching cubes

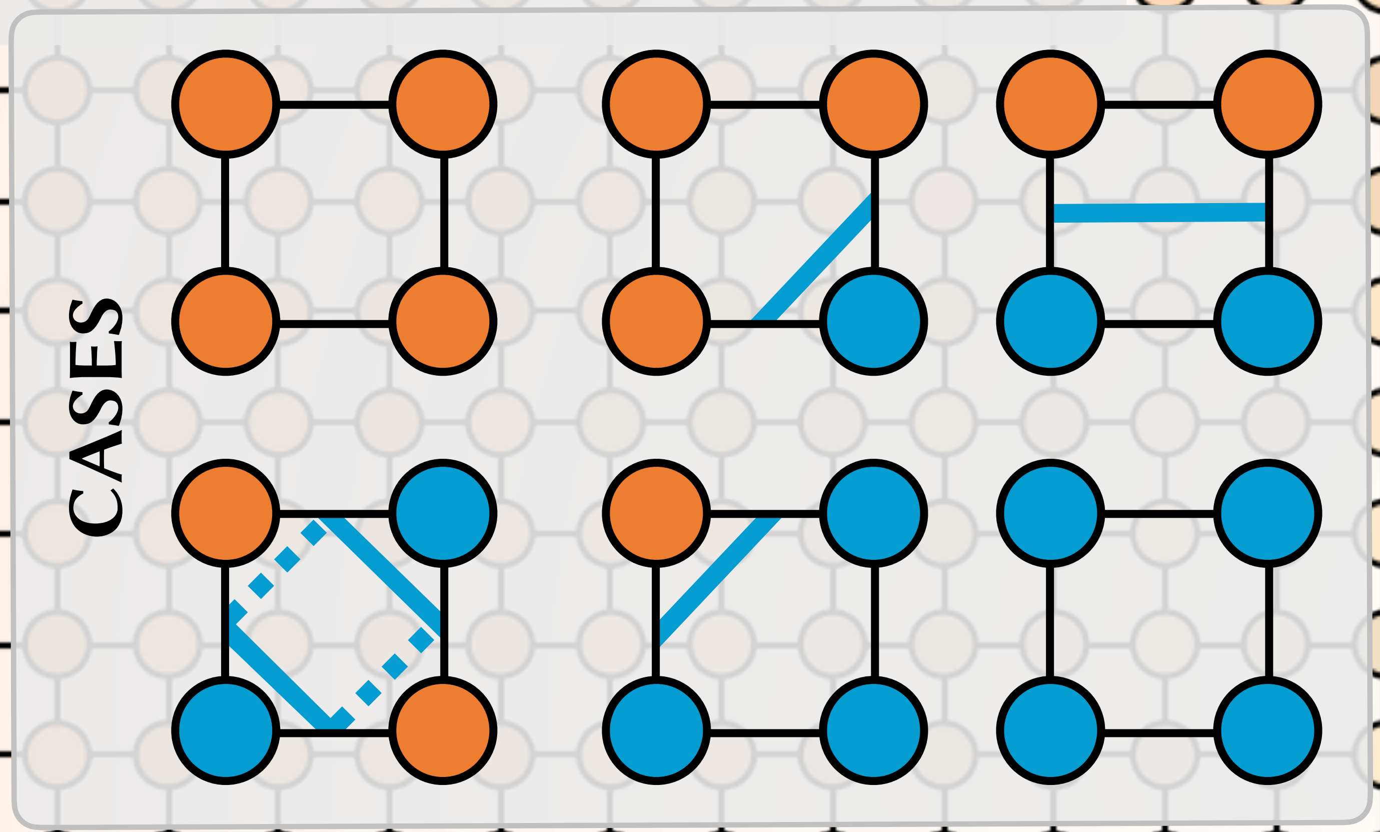
- introduced by [Lorensen & Cline, 1987]
- identify voxels with surface present, draw surface in each of these voxels

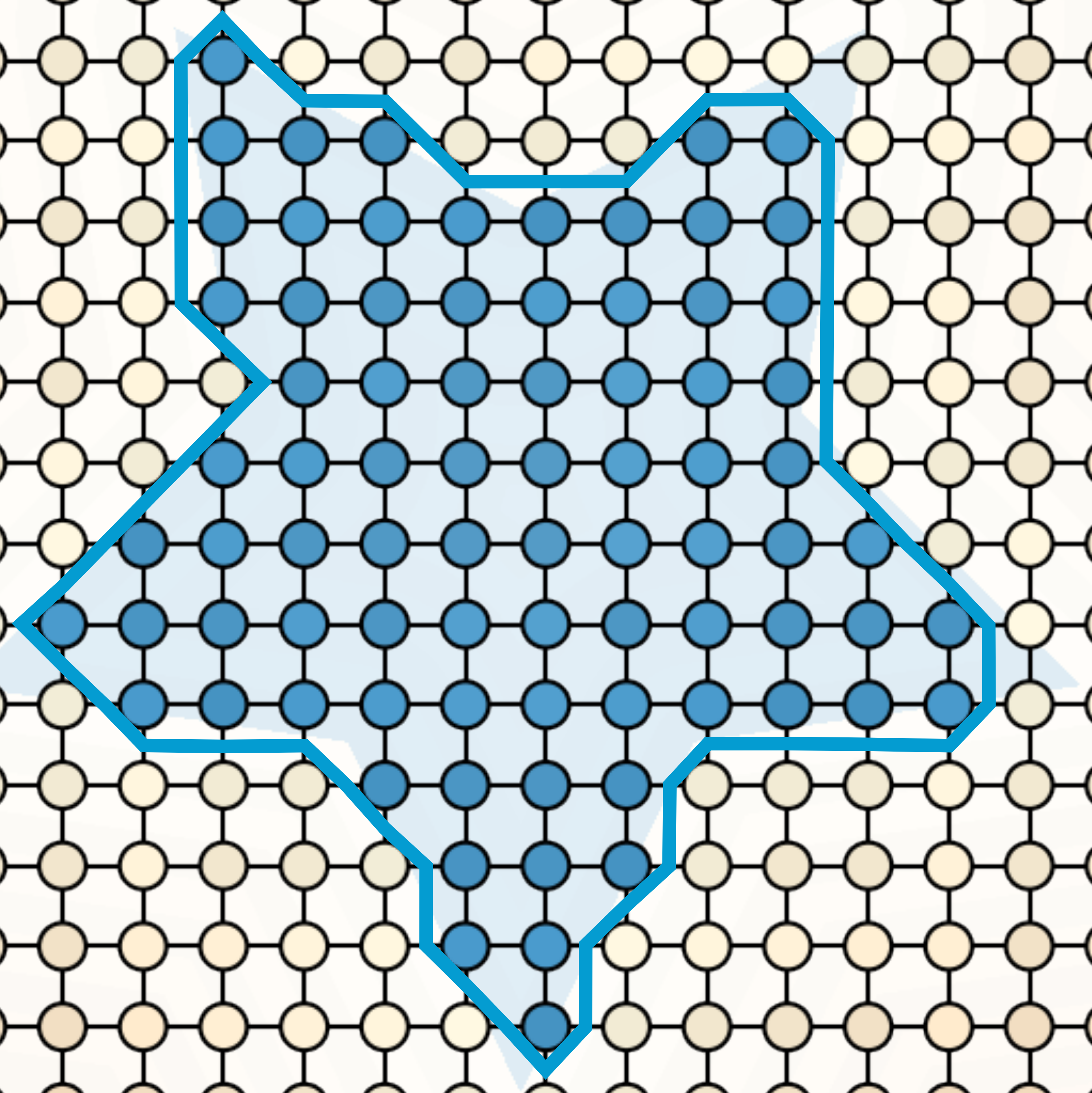
IMPLICIT SURFACES: BIG IDEAS

1. CSG
2. Taxonomy of implicits
3. Rendering
4. **Surface Extraction**

Marching cubes

- introduced by [Lorensen & Cline, 1987]
- identify voxels with surface present, draw surface in each of these voxels





IMPLICIT SURFACES: BIG IDEAS

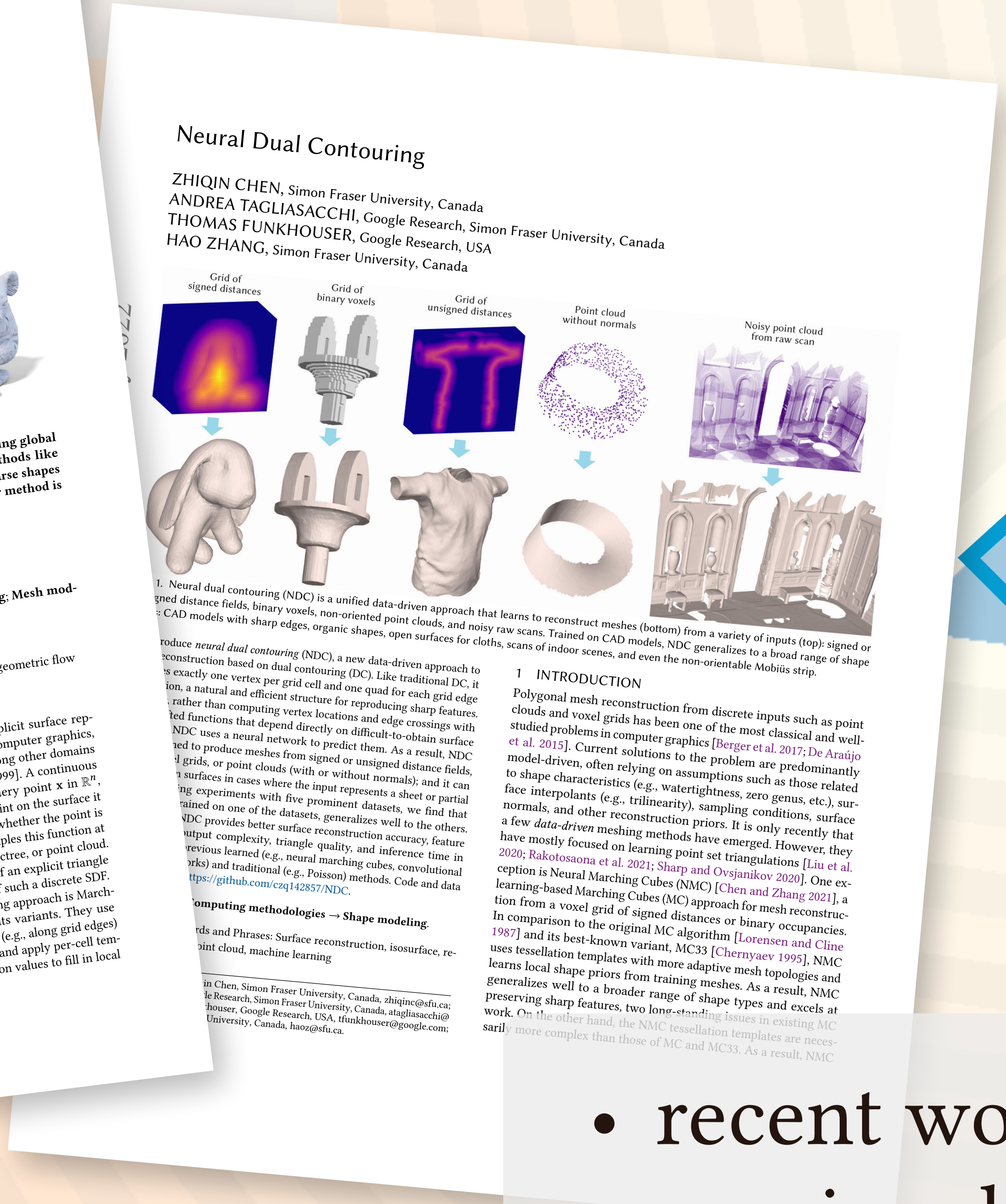
1. CSG
2. Taxonomy of implicits
3. Rendering
4. **Surface Extraction**

Marching cubes

- introduced by [Lorensen & Cline, 1987]
- identify voxels with surface present, draw surface in each of these voxels

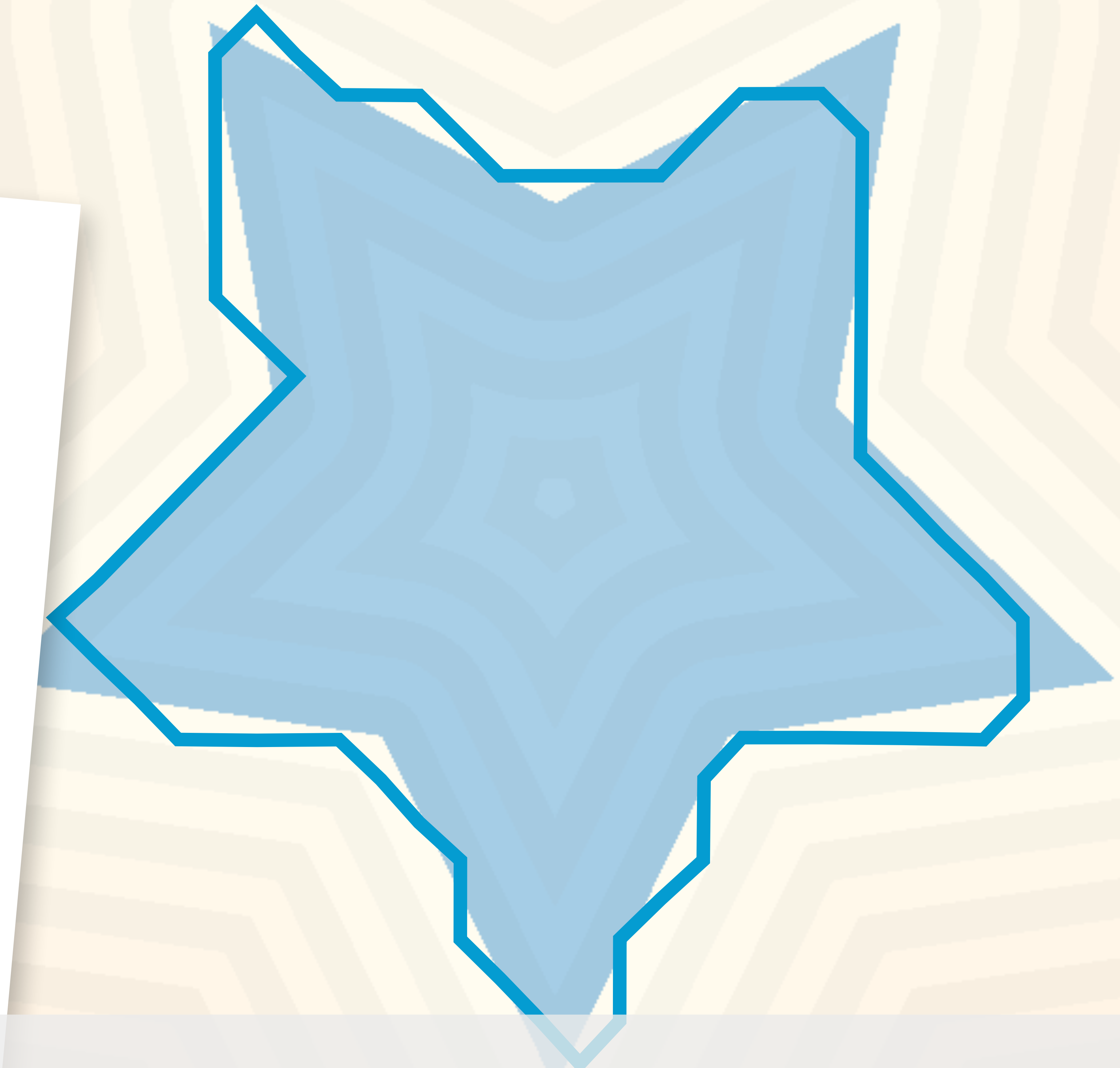
Reach For the Spheres: Tangency-Aware Surface Reconstruction of SDFs

Silvia Sellán, University of Toronto, Canada, sgsellan@cs.toronto.edu
Christopher Batty, University of Waterloo, Canada, christopher.batty@uwaterloo.ca
Oded Stein, University of Southern California, United States of America, ostein@usc.edu



Neural Dual Contouring

ZHIQIN CHEN, Simon Fraser University, Canada
ANDREA TAGLIASACCHI, Google Research, Simon Fraser University, Canada
THOMAS FUNKHOUSER, Google Research, USA
HAO ZHANG, Simon Fraser University, Canada

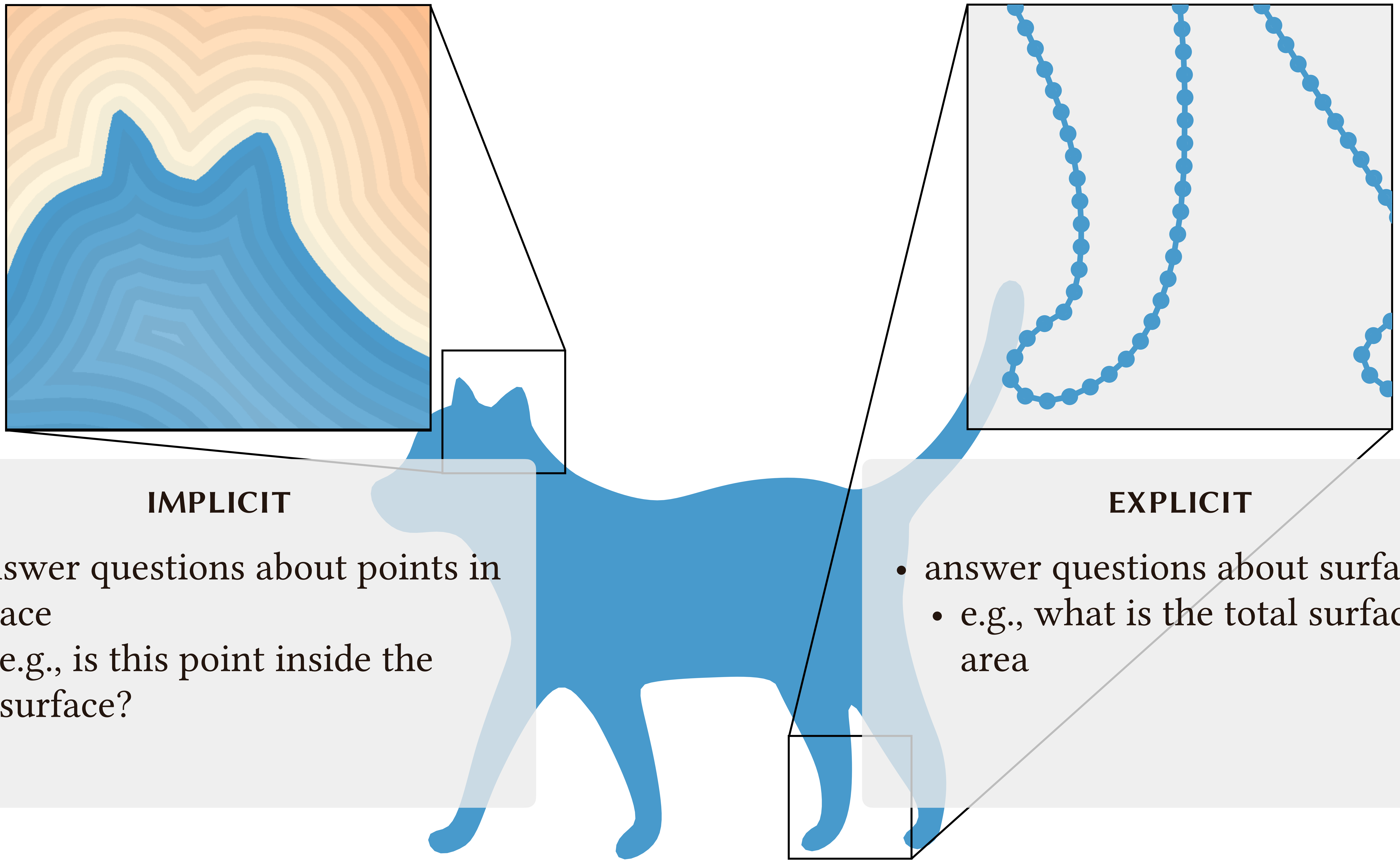


- recent work on improved surface extraction, e.g.,
  - using distance info for SDF case [Sellán et al., 2023]
  - using machine learning & data [Chen et al. 2022]

IMPLICIT SURFACES

1. CSG
2. Taxonomy of implicits
3. Rendering
4. **Surface Extraction**

Implicit vs. explicit representations



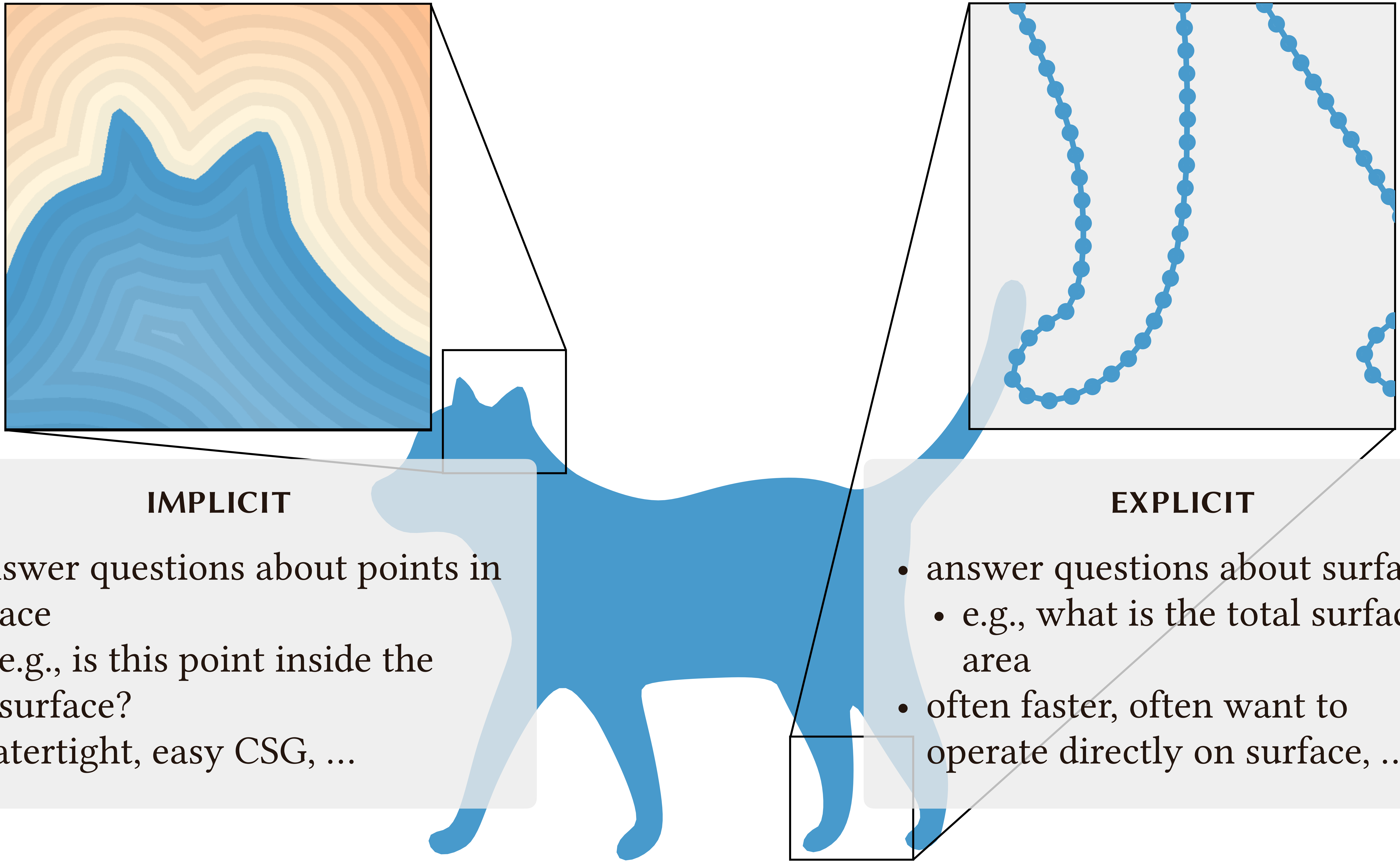
IMPLICIT

- answer questions about points in space
  - e.g., is this point inside the surface?

EXPLICIT

- answer questions about surface
  - e.g., what is the total surface area

Implicit vs. explicit representations



IMPLICIT

- answer questions about points in space
  - e.g., is this point inside the surface?
- watertight, easy CSG, …

EXPLICIT

- answer questions about surface
  - e.g., what is the total surface area
- often faster, often want to operate directly on surface, …

Implicit vs. explicit representations

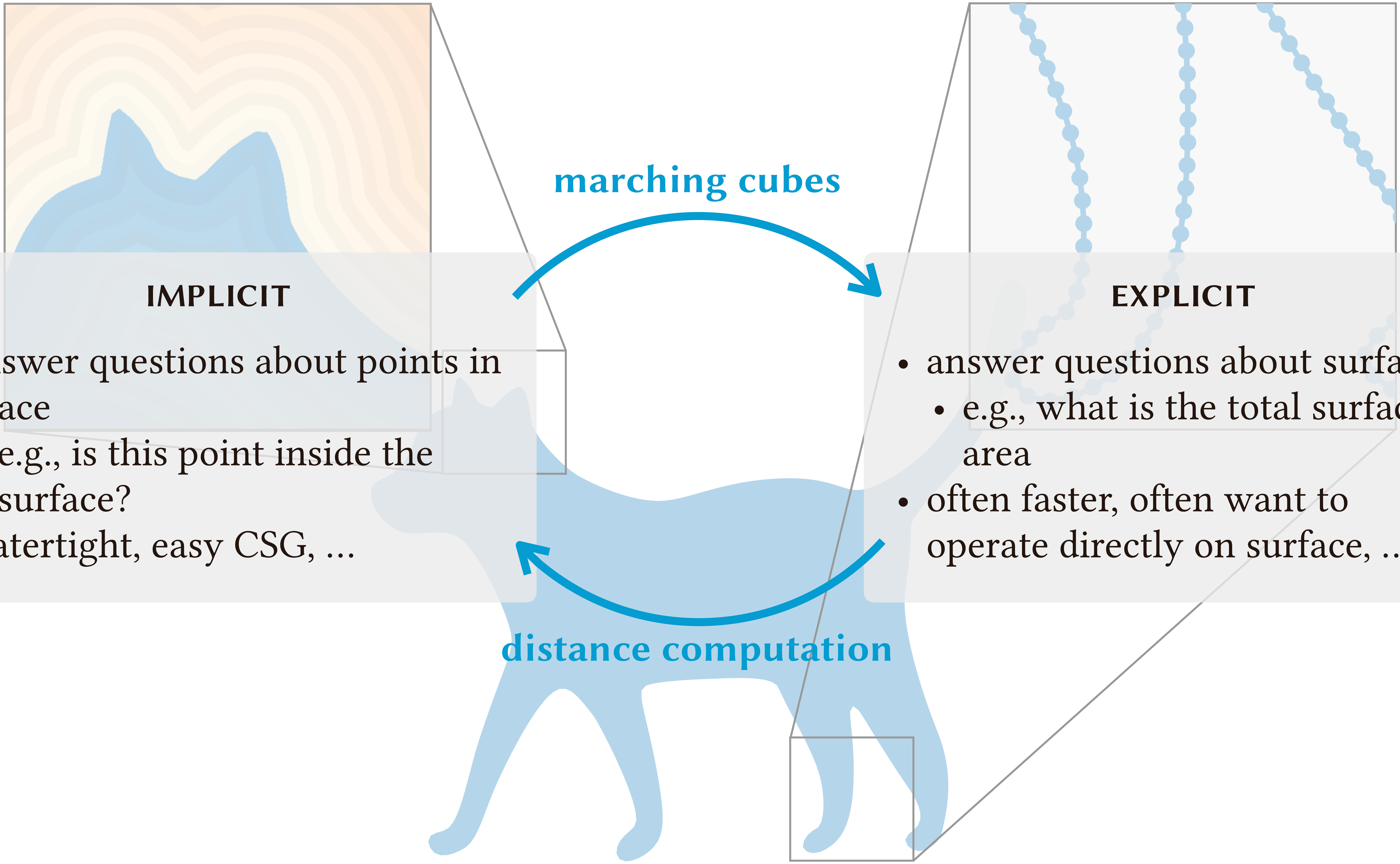
in ***implicit*** representations, topology changes are continuous (very useful for inverse tasks!)

Video: "Differentiable Signed Distance Function Rendering," Vicini et al., 2022

Implicit vs. explicit representations

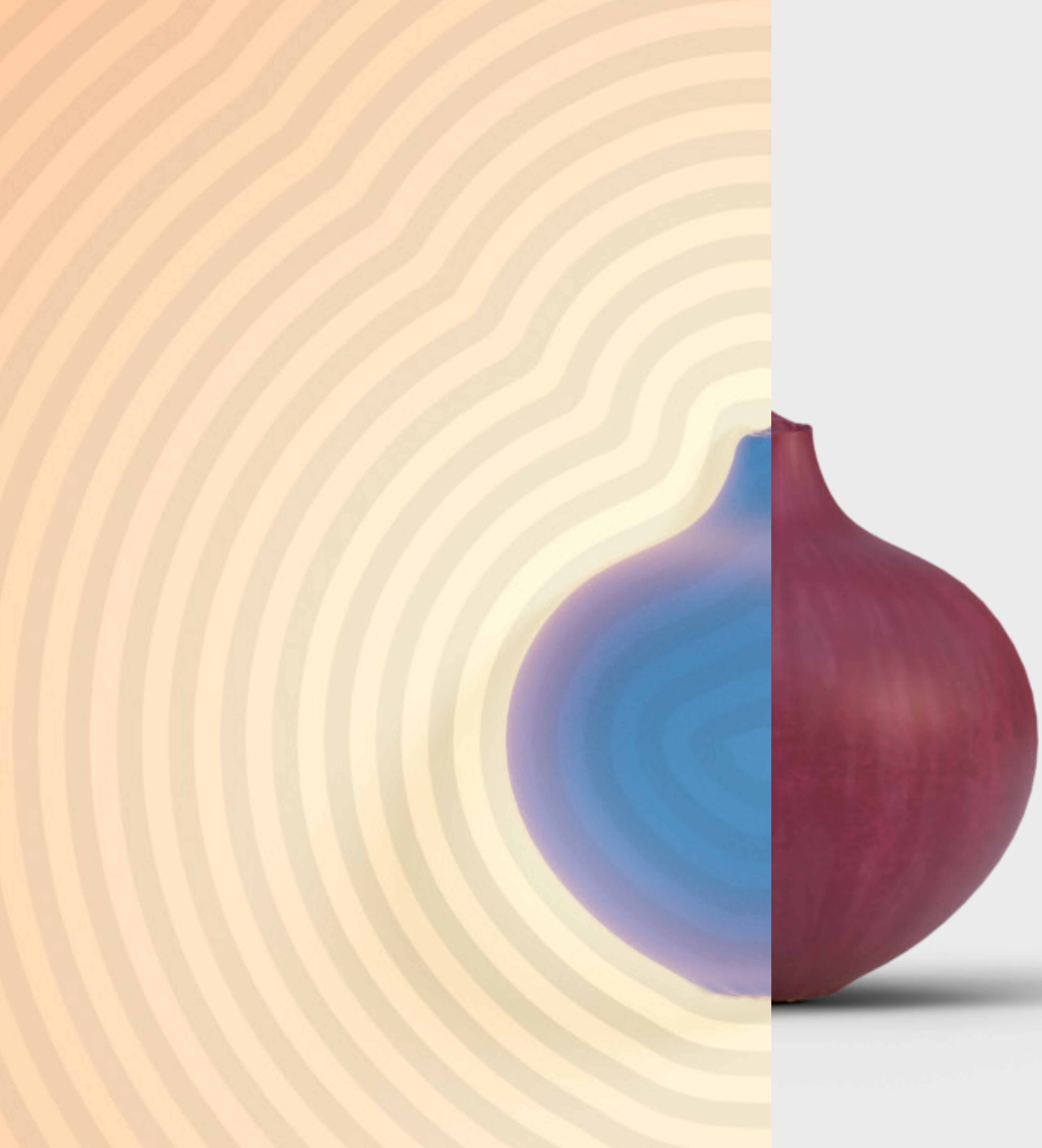
IMPLICIT

- answer questions about points in space
  - e.g., is this point inside the surface?
- watertight, easy CSG, …



EXPLICIT

- answer questions about surface
  - e.g., what is the total surface area
- often faster, often want to operate directly on surface, …



OVERVIEW

IMPLICIT SURFACES
1. Background & History
2. Digital Representations

NEURAL IMPLICITS
3. The Basics
4. Why & Why Not Use Them?
5. Working out the Details
6. Geometric Operations

CONCLUSION

Analytic representations

$$f(x, y) = x^2 + y^2 - r^2$$

Analytic representations

```
float sdStar5(in vec2 p, in float r, in float rf)
{
    const vec2 k1 = vec2(0.809016994375, -0.587785252292);
    const vec2 k2 = vec2(-k1.x,k1.y);
    p.x = abs(p.x);
    p -= 2.0*max(dot(k1,p),0.0)*k1;
    p -= 2.0*max(dot(k2,p),0.0)*k2;
    p.x = abs(p.x);
    p.y -= r;
    vec2 ba = rf*vec2(-k1.y,k1.x) - vec2(0,1);
    float h = clamp( dot(p,ba)/dot(ba,ba), 0.0, r );
    return length(p-ba*h) * sign(p.y*ba.x-p.x*ba.y);
}
```

- slow to compute complex formulas
- can’t always find analytic representation

For analytic definitions of SDFs, see “Distance Functions” page on Inigo Quillez’s blog

Code: Inigo Quillez

Grid-based representations

Grid-based representations: just a function!

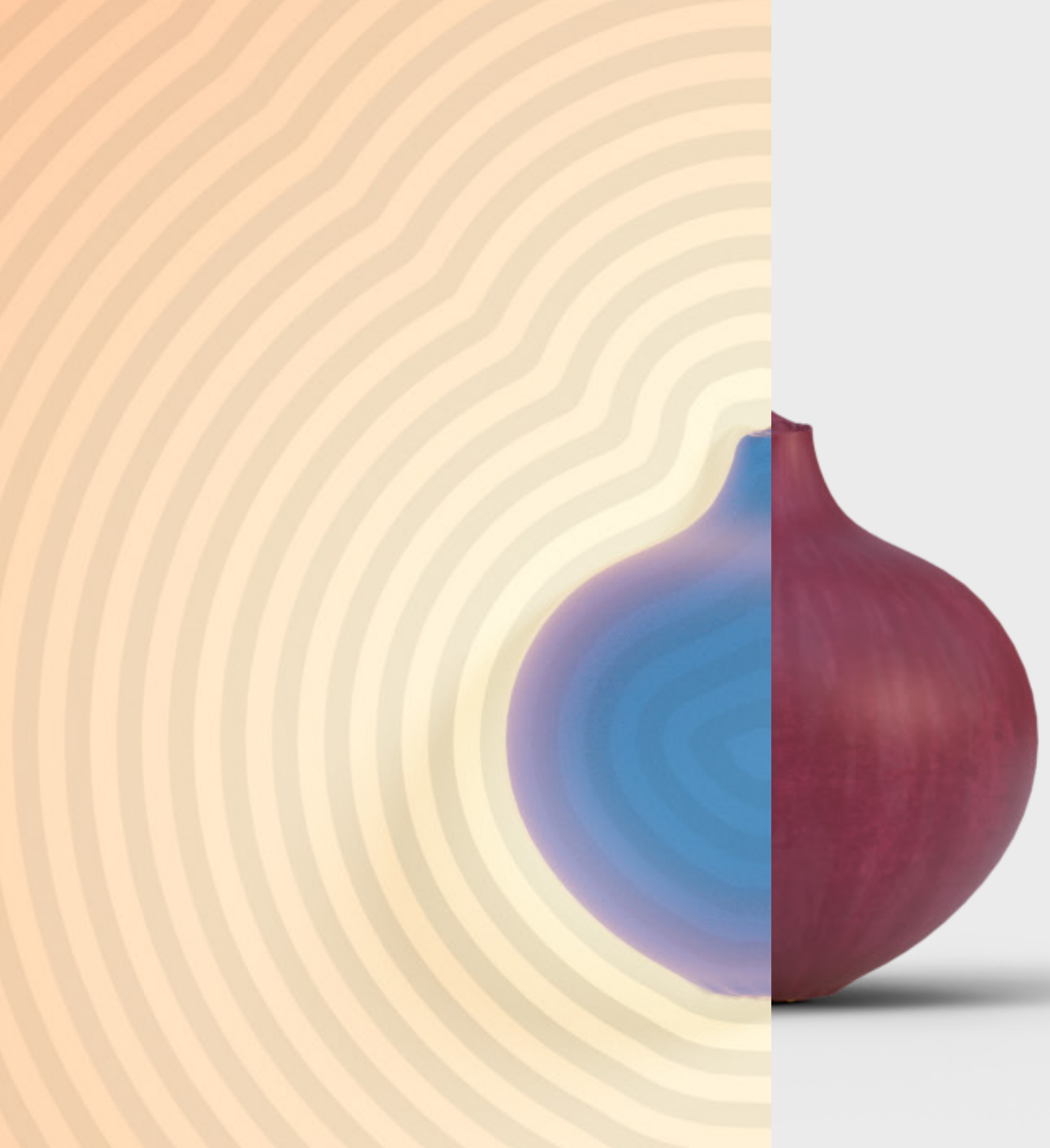
$w_{0,0}$	$w_{1,0}$	$w_{2,0}$	$w_{3,0}$
$w_{0,1}$	$w_{1,1}$	$w_{2,1}$	$w_{3,1}$
$w_{0,2}$	$w_{1,2}$	$w_{2,2}$	$w_{3,2}$
$w_{0,3}$	$w_{1,3}$	$w_{2,3}$	$w_{3,3}$

$$f(x, y) = \big(\lfloor x \rfloor = 0 \;\&\; \lfloor y \rfloor = 0\big)\, w_{0,0} + \big(\lfloor x \rfloor = 1 \;\&\; \lfloor y \rfloor = 0\big)\, w_{1,0} + \ldots$$

Generalization: *any* function space

Degree 0

$$f(x, y) = w_{n,0}x^n + w_{n-1,1}x^n y + \ldots + w_{0,1}y + w_{0,0}$$



OVERVIEW

IMPLICIT SURFACES
1. Background & History
2. Digital Representations

NEURAL IMPLICITS
3. The Basics
4. Why & Why Not Use Them?
5. Working out the Details
6. Geometric Operations

CONCLUSION

Neural networks are a function space

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

$$f_\theta(x) = A_\theta^N \varphi(A_\theta^1 \varphi(A_\theta^0 x + b_\theta^0) + b_\theta^1 + \ldots) + b_\theta^N$$

For an intro to Neural Networks, see the course "6.036 Intro to ML" on OCW

Neural networks are a function space

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

$$f_\theta(x) = A_\theta^N \varphi(A_\theta^1 \varphi(A_\theta^0 x + b_\theta^0) + b_\theta^1 + \ldots) + b_\theta^N$$

linear layer

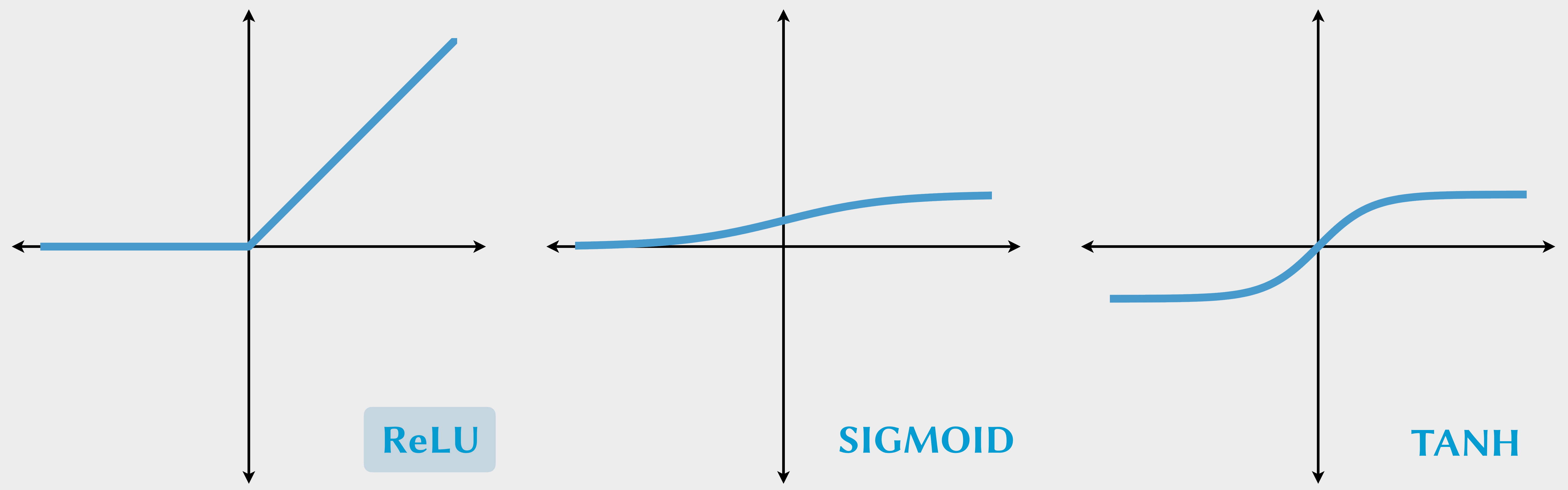
For an intro to Neural Networks, see the course "6.036 Intro to ML" on OCW

Neural networks are a function space

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

$$f_\theta(x) = A_\theta^N \varphi(A_\theta^1 \varphi(A_\theta^0 x + b_\theta^0) + b_\theta^1 + \ldots) + b_\theta^N$$

non-linearity/activation function



For an intro to Neural Networks, see the course "6.036 Intro to ML" on OCW

Neural networks are a function space

$$f_\theta : \mathbb{R}^n \rightarrow \mathbb{R}$$

MLP

hidden layers

$$f_\theta(x) = A_\theta^N \varphi(A_\theta^1 \varphi(A_\theta^0 x + b_\theta^0) + b_\theta^1 + \ldots) + b_\theta^N$$

output layer

input layer

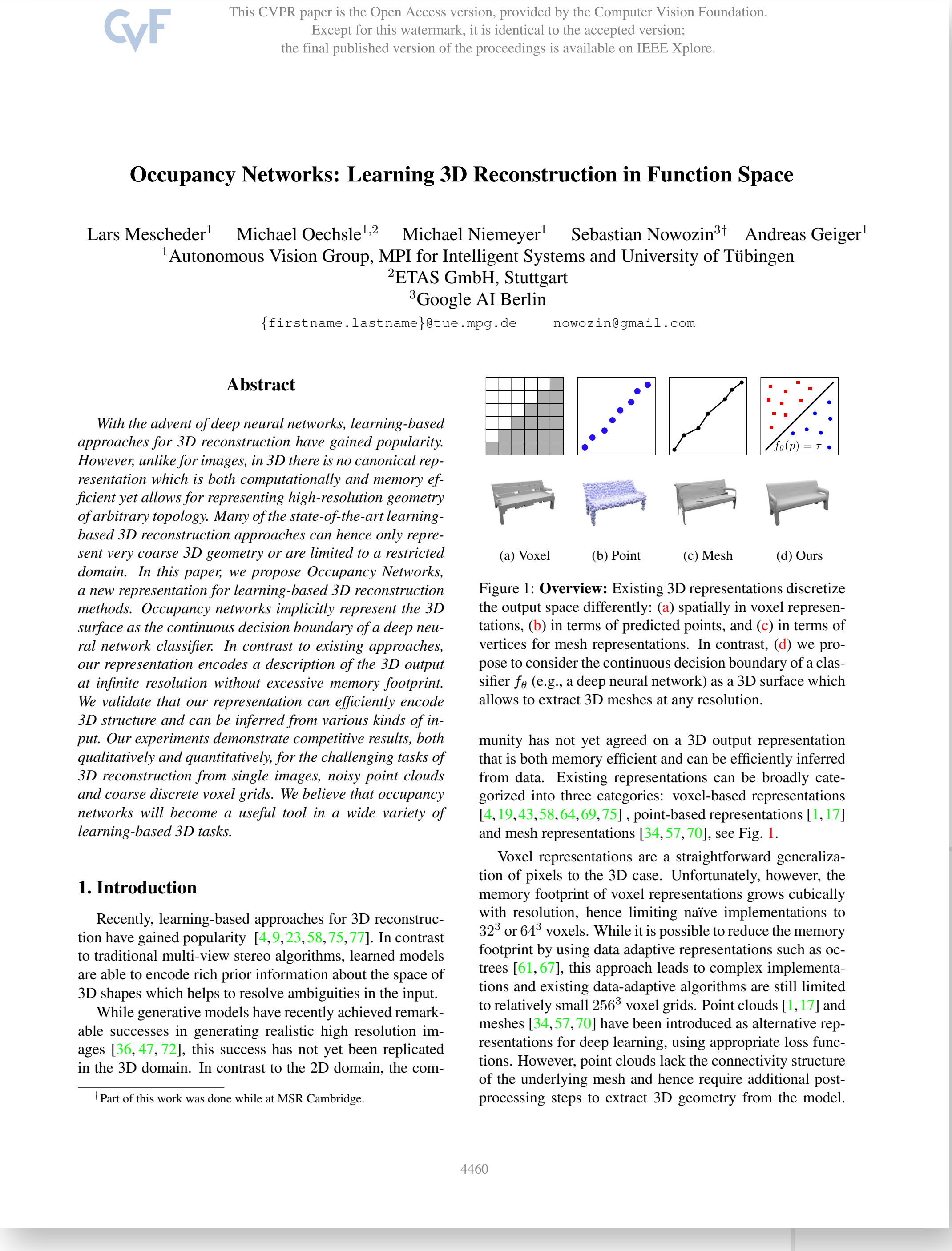
Neural networks are a function space

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

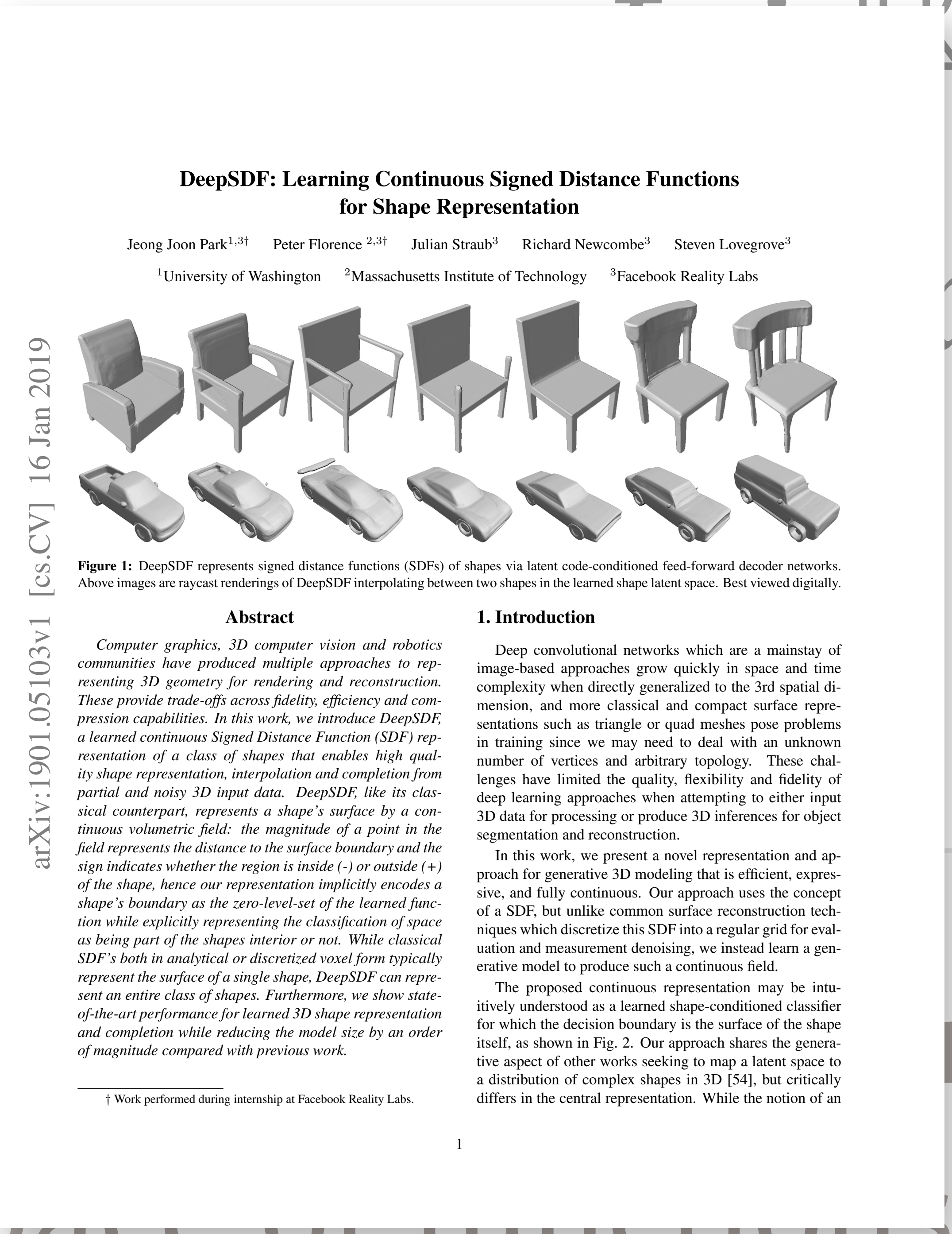
$$f_\theta(x) = A_\theta^N \varphi(A_\theta^1 \varphi(A_\theta^0 x + b_\theta^0) + b_\theta^1 + \ldots) + b_\theta^N$$

Key idea: neural networks provide a space of functions with parameters θ

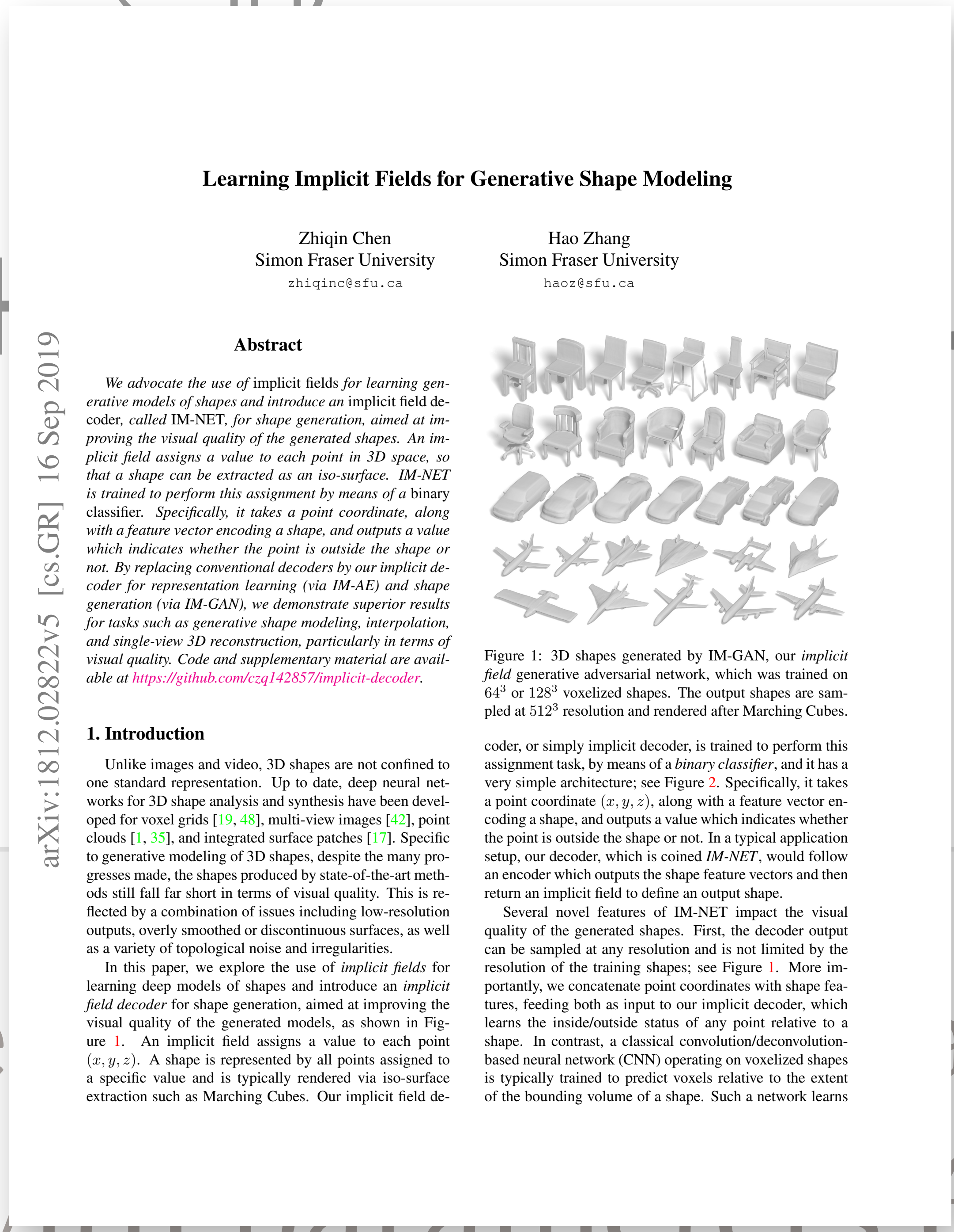
Early work on neural implicits



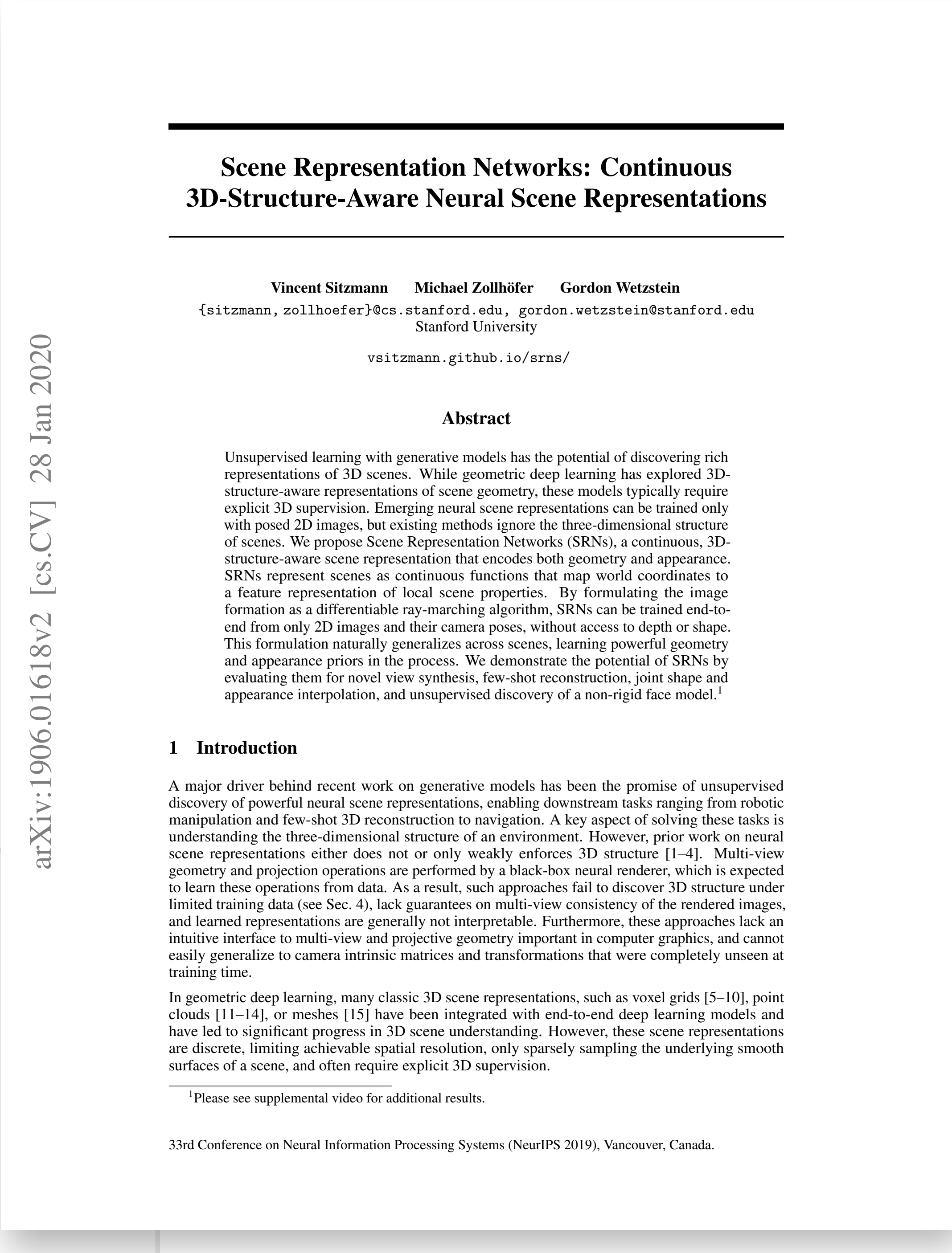
Occupancy Networks
Mescheder et al., 2019



DeepSDF
Park et al., 2019



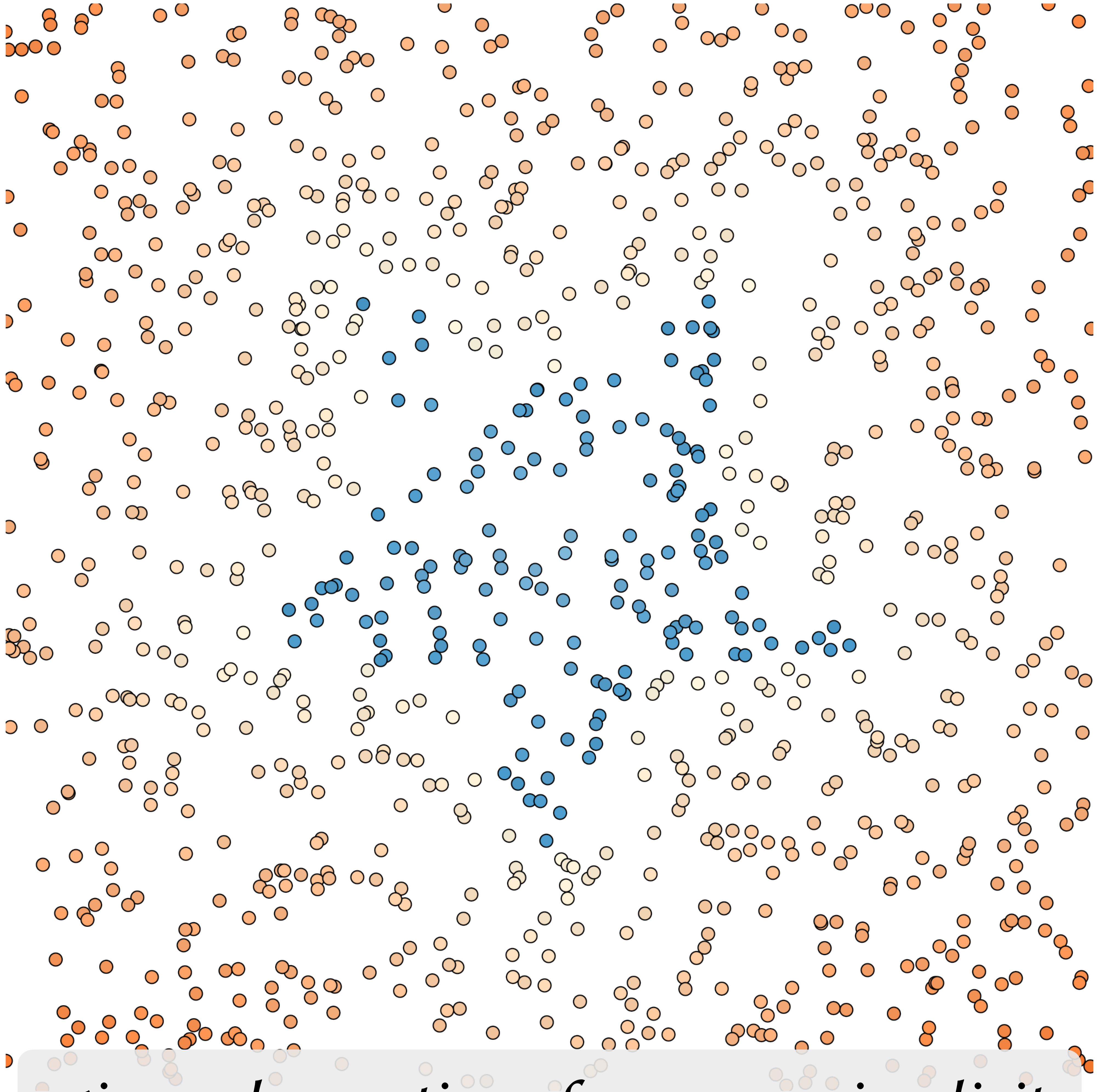
IM-NET
Chen et al., 2019



Scene Rep. Nets.
Sitzmann et al., 2019

For more info on neural implicits, see Vincent Sitzmann's course "ML for Inverse Graphics"

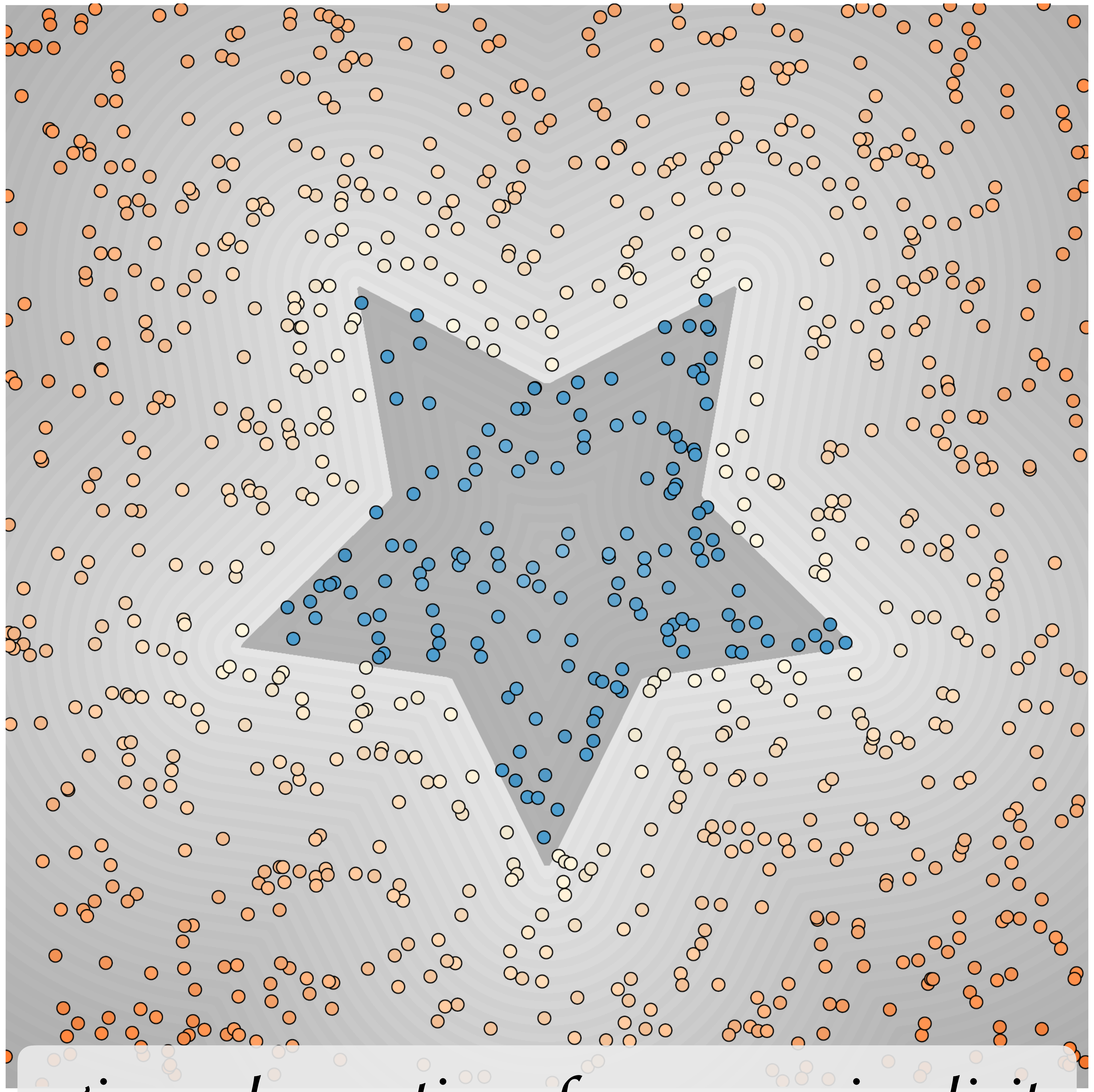
Function fitting for neural implicits



given observations from some implicit

given: $(\mathcal{X}, \mathcal{V})$ with $f(x) = v$

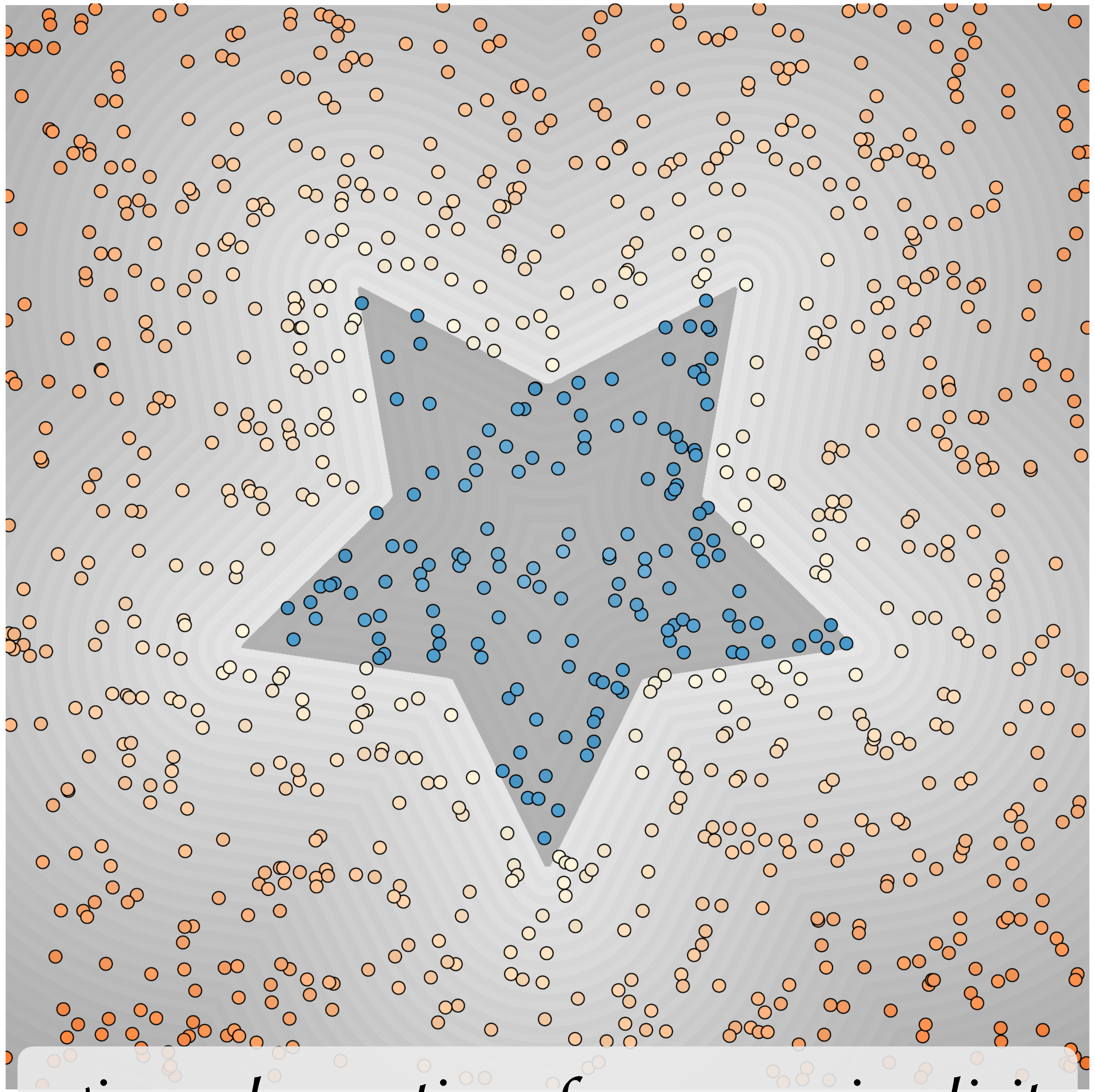
Function fitting for neural implicits



given observations from some implicit

given: $(\mathcal{X}, \mathcal{V})$ with $f(x) = v$

Function fitting for neural implicits



given observations from some implicit

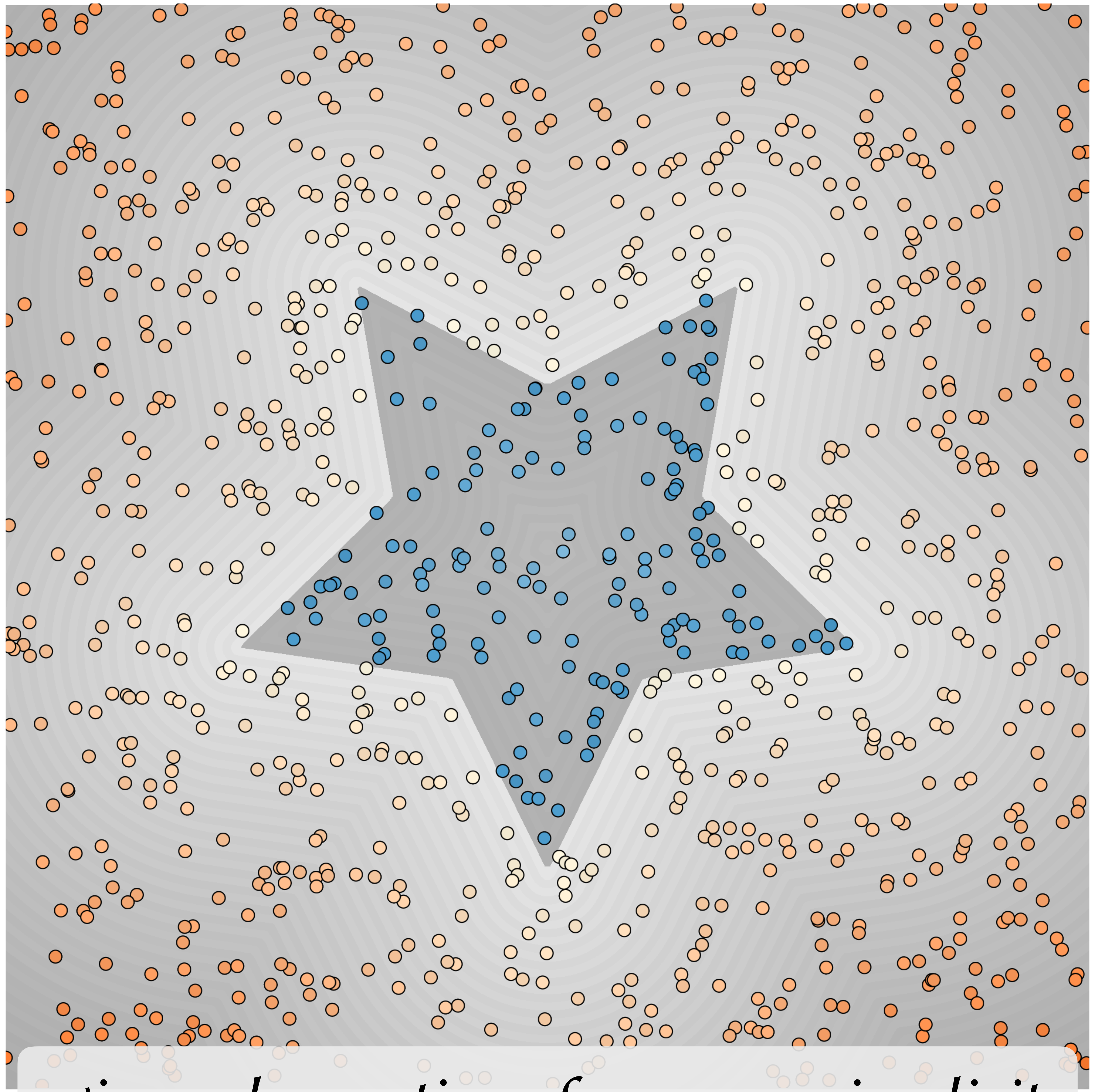
given: $(\mathcal{X}, \mathcal{V})$ with $f(x) = v$

$$\min_{\theta} \sum_{x \in \mathcal{X}} \left(f_\theta(x) - v\right)^2$$

"loss function"

find best fitting f in func. space

Function fitting for neural implicits



given observations from some implicit

given: $(\mathscr{X}, \mathscr{V})$ with $f(x) = v$

$$\min_{\theta} \sum_{x \in \mathscr{X}} \left(f_\theta(x) - v\right)^2$$

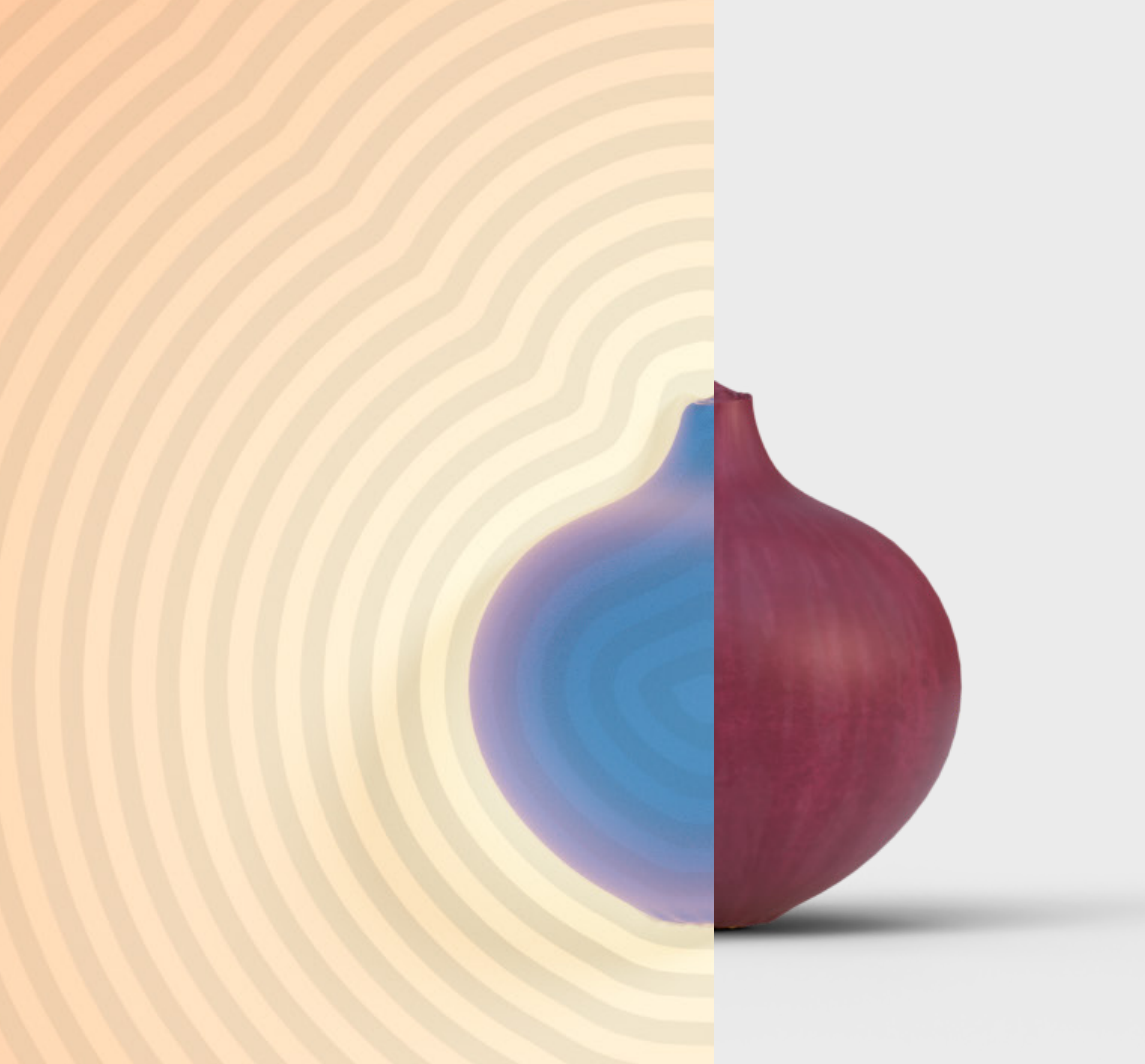
optimized using stochastic gradient descent

find best fitting f in func. space

In practice: SDF fit with pytorch

```
class SDF_NN(nn.Module):
    def __init__(self, num_layers, hidden_size):
        super().__init__()

        self.layers = nn.ModuleList([nn.Linear(2, hidden_size), nn.ReLU()])

        for _ in range(num_layers-2):
            self.layers.append(nn.Linear(hidden_size, hidden_size))
            self.layers.append(nn.ReLU())

        self.layers.append(nn.Linear(hidden_size, 1))


    def forward(self, inp):
        out = inp

        for layer in self.layers:
            out = layer(out)

        return out
```

```
def train(model, data_gen, loss_fn, steps=10000, step_size=0.001):
    optim = torch.optim.Adam(model.parameters(), lr=step_size)


    for i in tqdm(range(steps)):
        pts, gts = data_gen()

        pred = model(pts)
        loss = loss_fn(pred, gts)

        loss.backward()
        optim.step()
        optim.zero_grad()

        if i % 100 == 0:
            tqdm.write(f'Step: {i}, loss: {loss}')

    torch.save(model.state_dict(), './trained_model.pt')
```



OVERVIEW

IMPLICIT SURFACES

1. Background & History
2. Digital Representations

NEURAL IMPLICITS

3. The Basics
4. Why & Why Not Use Them?
5. Working out the Details
6. Geometric Operations

CONCLUSION

Question: is this a good function space?

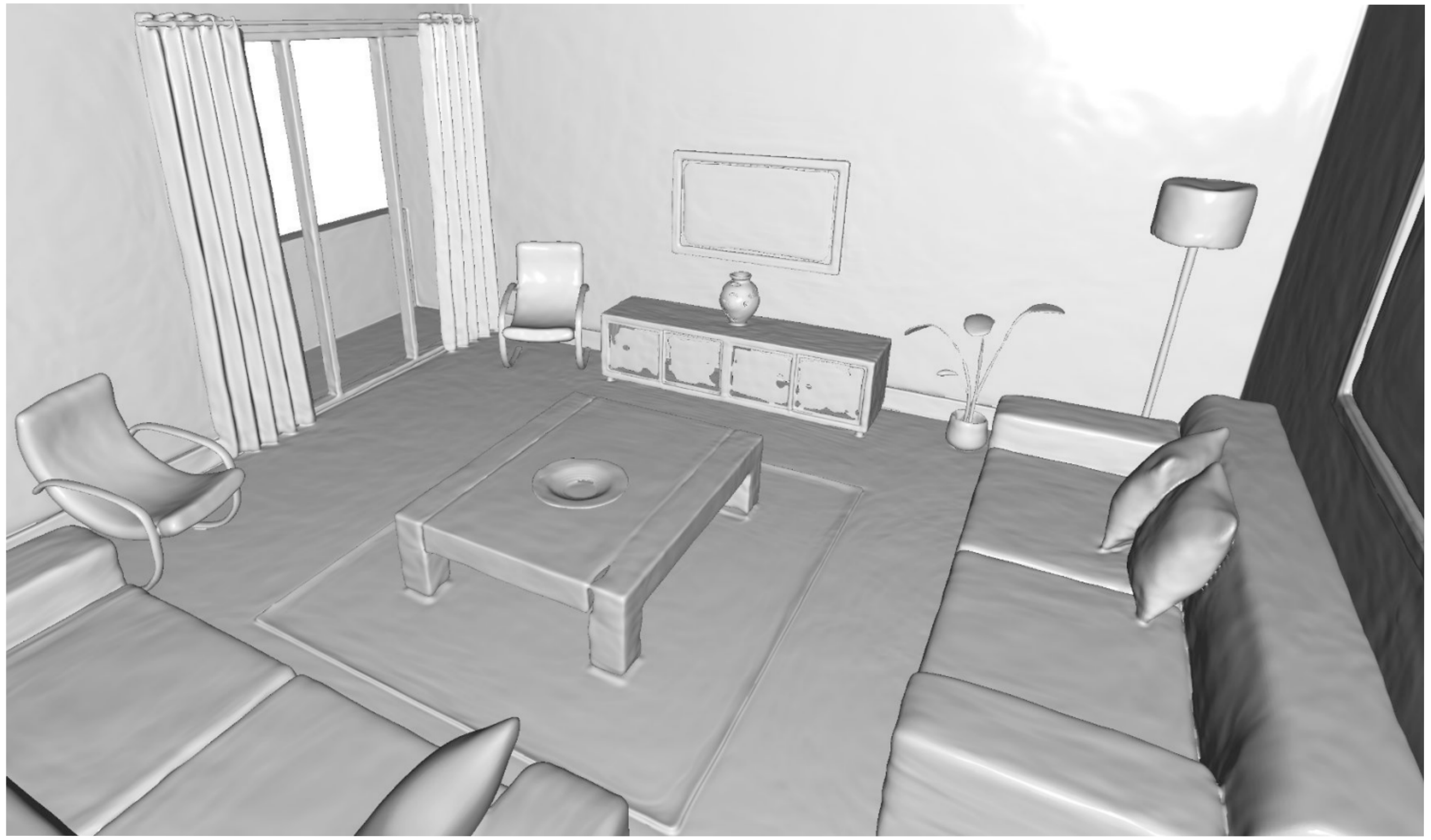
WHY?

1. **Representative ability**
2. Adaptability
3. Trivially differentiable

Image: "SIREN," Sitzmann et al., 2020

Question: is this a good function space?

Universal Approximation Theorem
[Hornik et al., 1989]
A MLP with one hidden layer of infinite width and ReLU nonlinearities can represent any continuous function



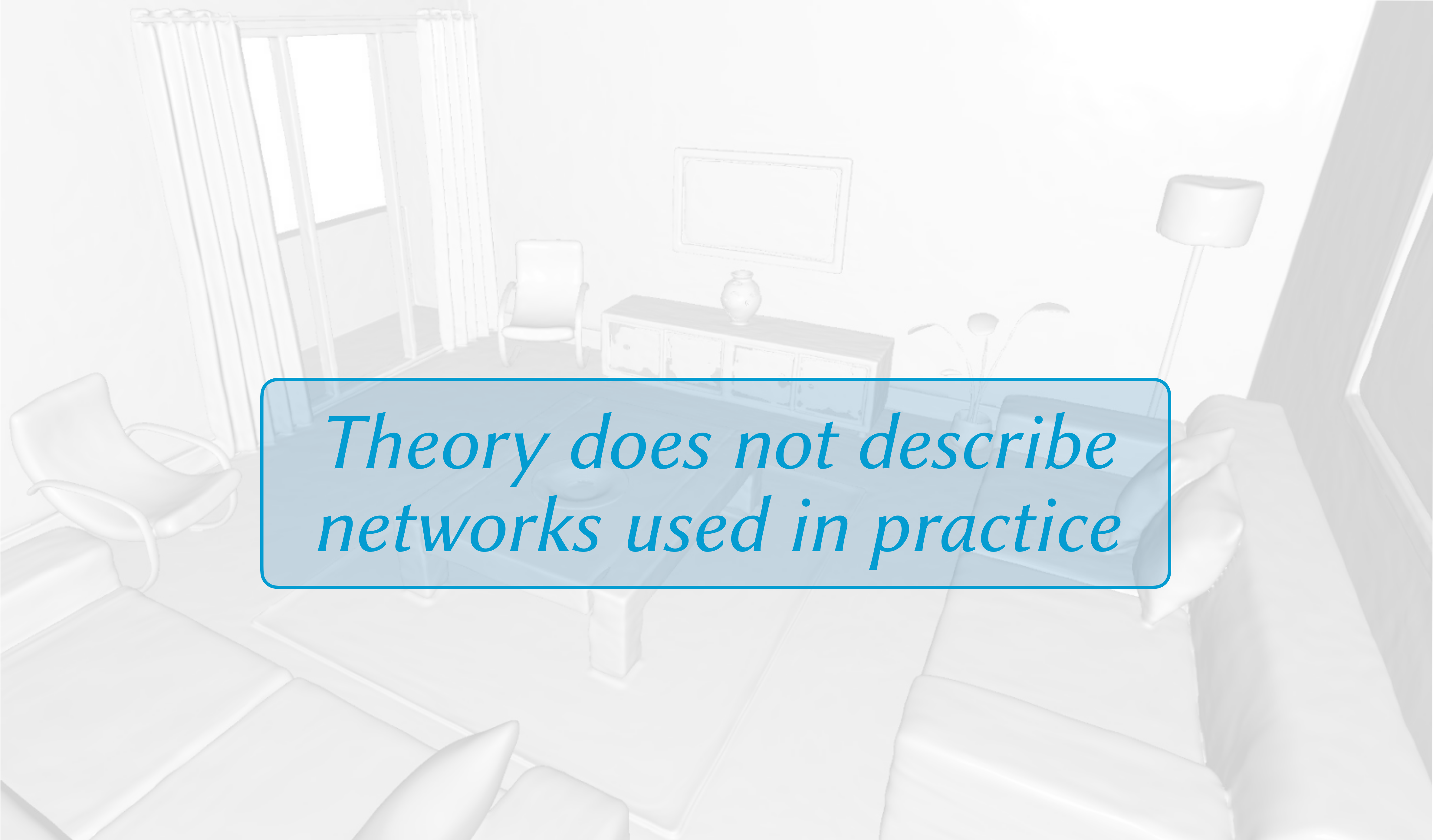
WHY?

1. **Representative ability**
2. Adaptability
3. Trivially differentiable

Image: "SIREN," Sitzmann et al., 2020

Question: is this a good function space?

Universal Approximation Theorem
[Hornik et al., 1989]
A MLP with one hidden layer of infinite width and ReLU nonlinearities can represent any continuous function

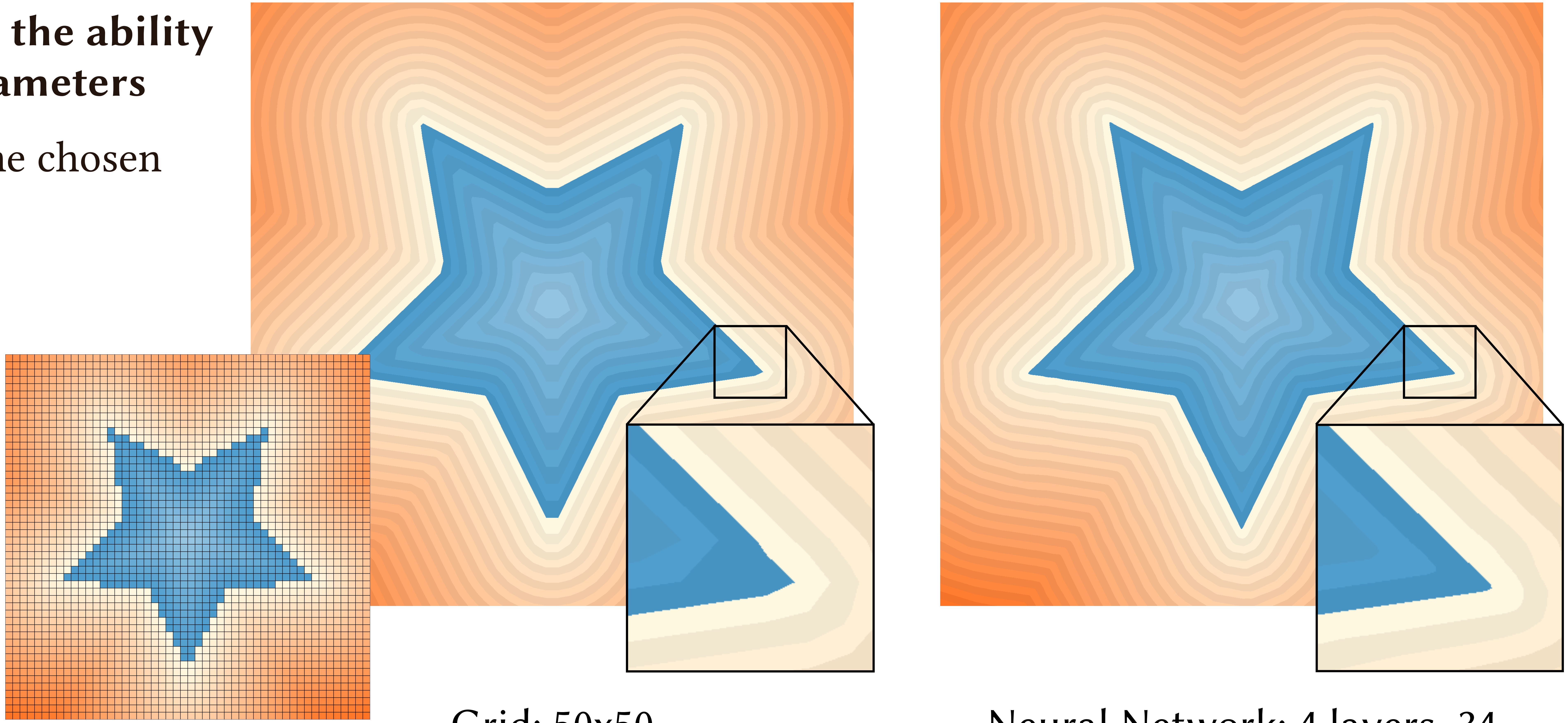


WHY?
1. **Representative ability**
2. Adaptability
3. Trivially differentiable

Image: "SIREN," Sitzmann et al., 2020

Neural implicits have the ability to adaptively use parameters

- but this depends on the chosen loss function!



Grid: 50x50

2500 parameters

Neural Network: 4 layers, 34 hidden size

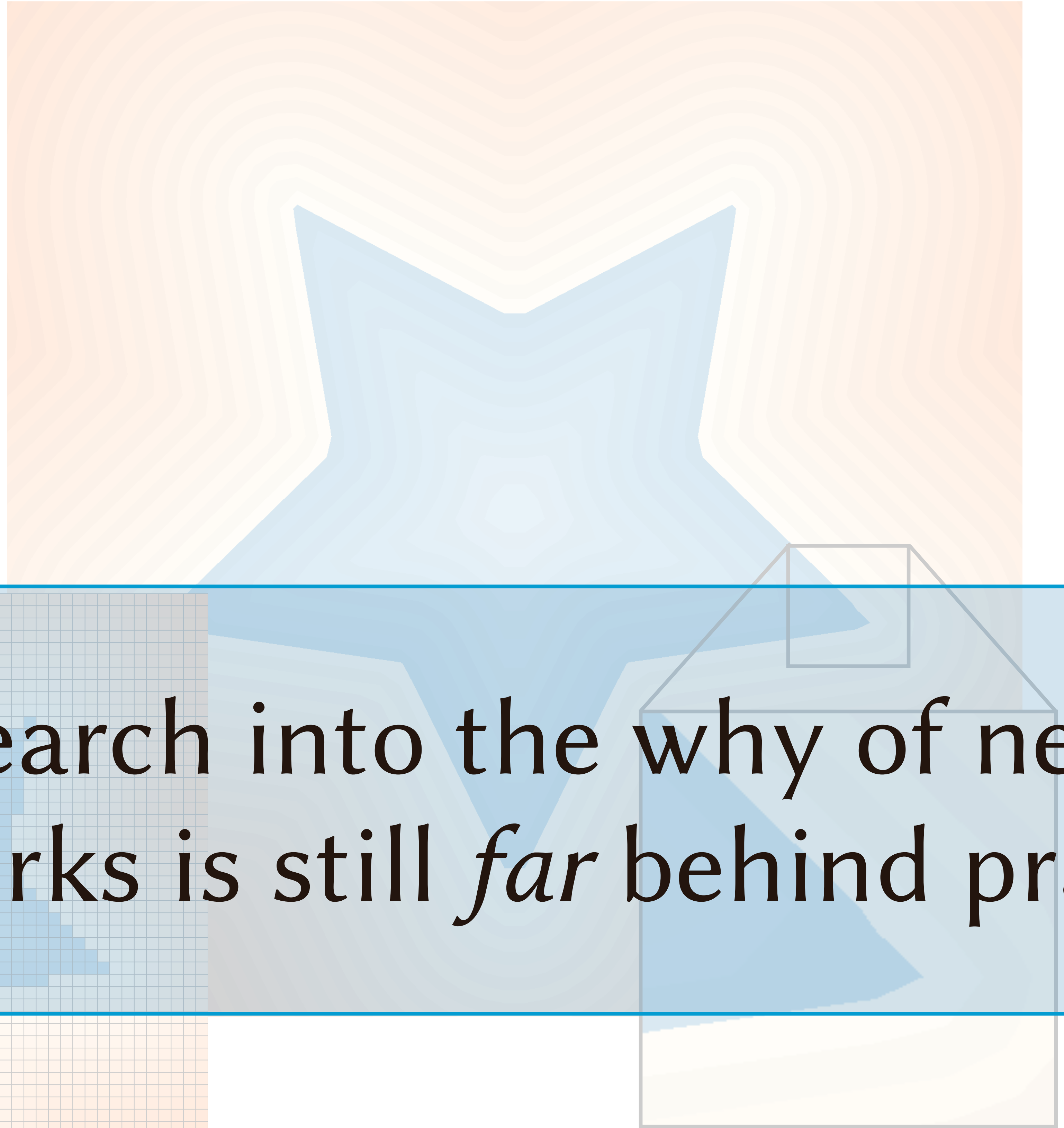
2414 parameters

WHY?

1. Representative ability
2. **Adaptability**
3. Trivially differentiable

Neural implicits have the ability to adaptively use parameters

- but this depends on the chosen loss function!



Grid: 50x50

2500 parameters

Neural Network: 4 layers, 34 hidden size

2414 parameters

Research into the why of neural networks is still *far* behind practice.

WHY?

1. Representative ability
2. **Adaptability**
3. Trivially differentiable

Neural implicits can easily be used to for many problems requiring a differentiable geometry representation

WHY?

1. Representative ability
2. Adaptability
3. ✨ **Trivially differentiable** ✨

Neural implicits can easily be used to for many problems requiring a differentiable geometry representation

Single shot reconstruction [Sitzmann et al., 2019]

WHY?

1. Representative ability
2. Adaptability
3. ✨ **Trivially differentiable** ✨

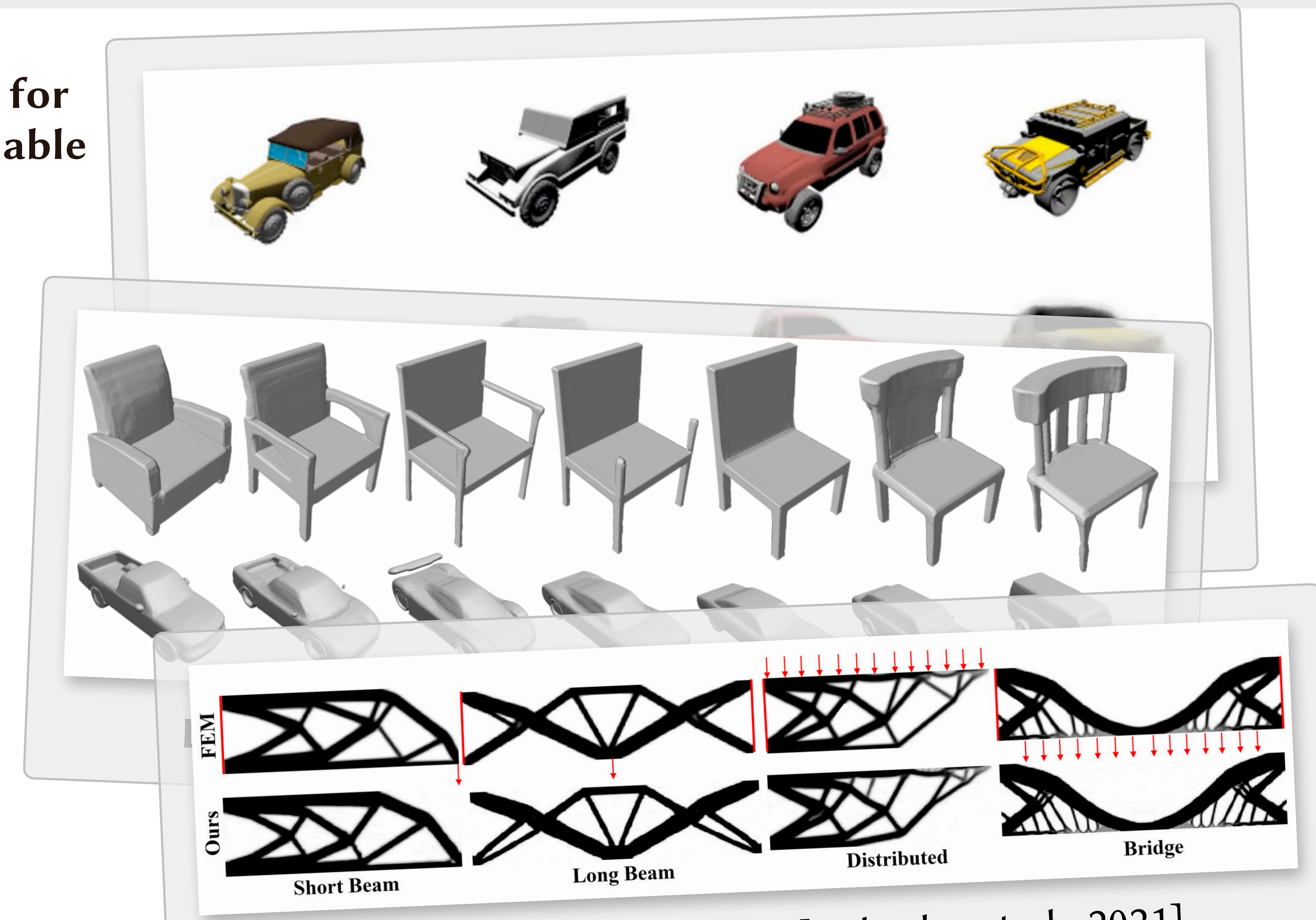
Neural implicits can easily be used to for many problems requiring a differentiable geometry representation

Latent space interpolation [Park et al., 2019]

WHY?

1. Representative ability
2. Adaptability
3. ✨ **Trivially differentiable** ✨

Neural implicits can easily be used to for many problems requiring a differentiable geometry representation



Topology Optimization [Zehnder et al., 2021]

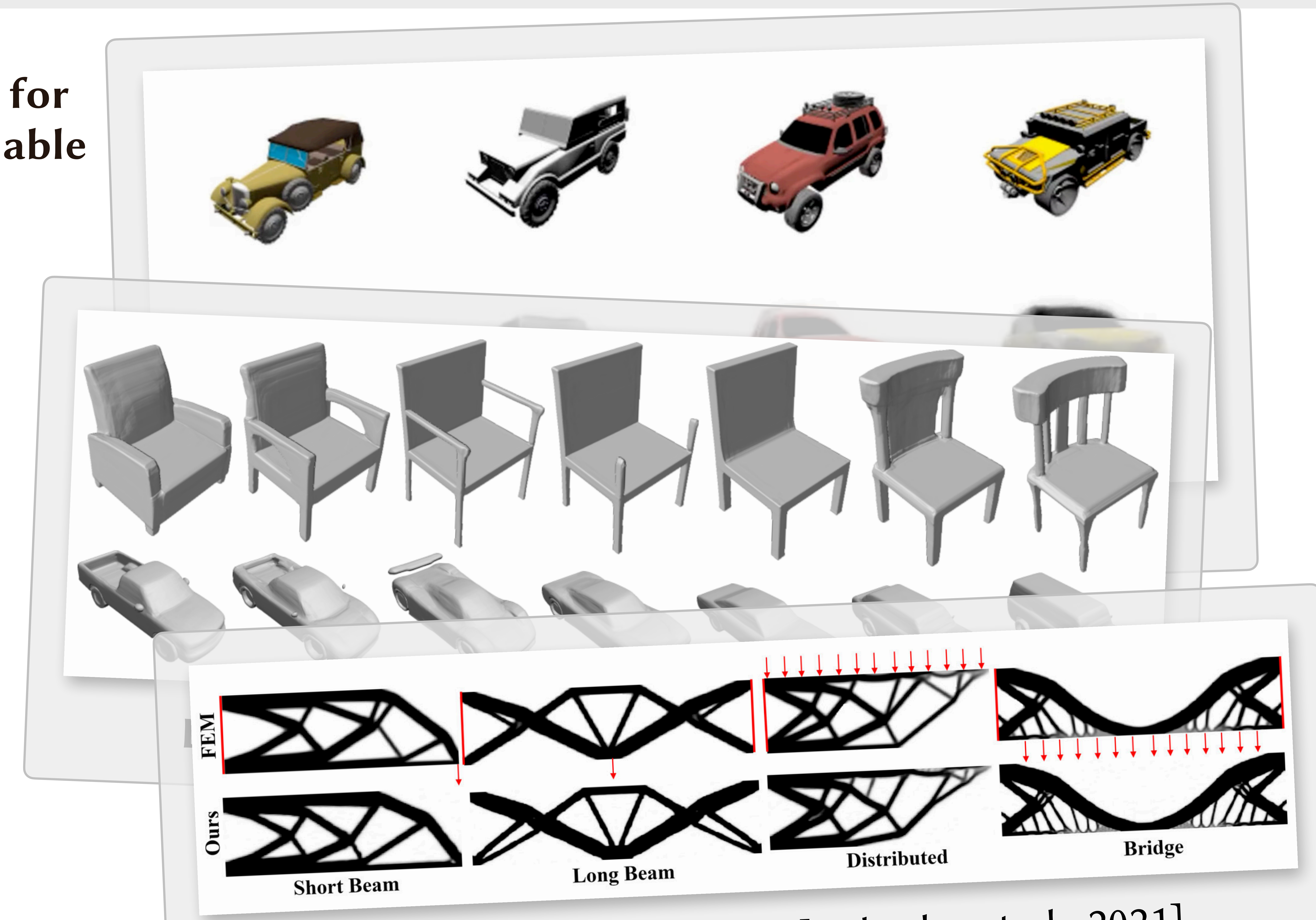
WHY?

1. Representative ability
2. Adaptability
3. ✨ **Trivially differentiable** ✨

Neural implicits can easily be used to for many problems requiring a differentiable geometry representation

In general:

- inverse tasks
- learning from data
- integrate with ML toolbox



Topology Optimization [Zehnder et al., 2021]

WHY?

1. Representative ability
2. Adaptability
3. ✨ **Trivially differentiable** ✨

Properties hold only through *soft losses*

Loss function: describes problem to be solved, e.g.,

- SDF property
- reconstruction loss
- …

$$\min_{\theta} \mathscr{L}(f_\theta)$$

WHY NOT?

1. **Imprecision**
2. Editing (etc.) is difficult
3. The wrong representation

Image: "Fourier Features," Tancik et al., 2020

Properties hold only through *soft losses*

- can be difficult to quantify amount of error
- issue for application which require guarantees!


Loss function: describes problem to be solved, e.g.,

- SDF property
- reconstruction loss
- …

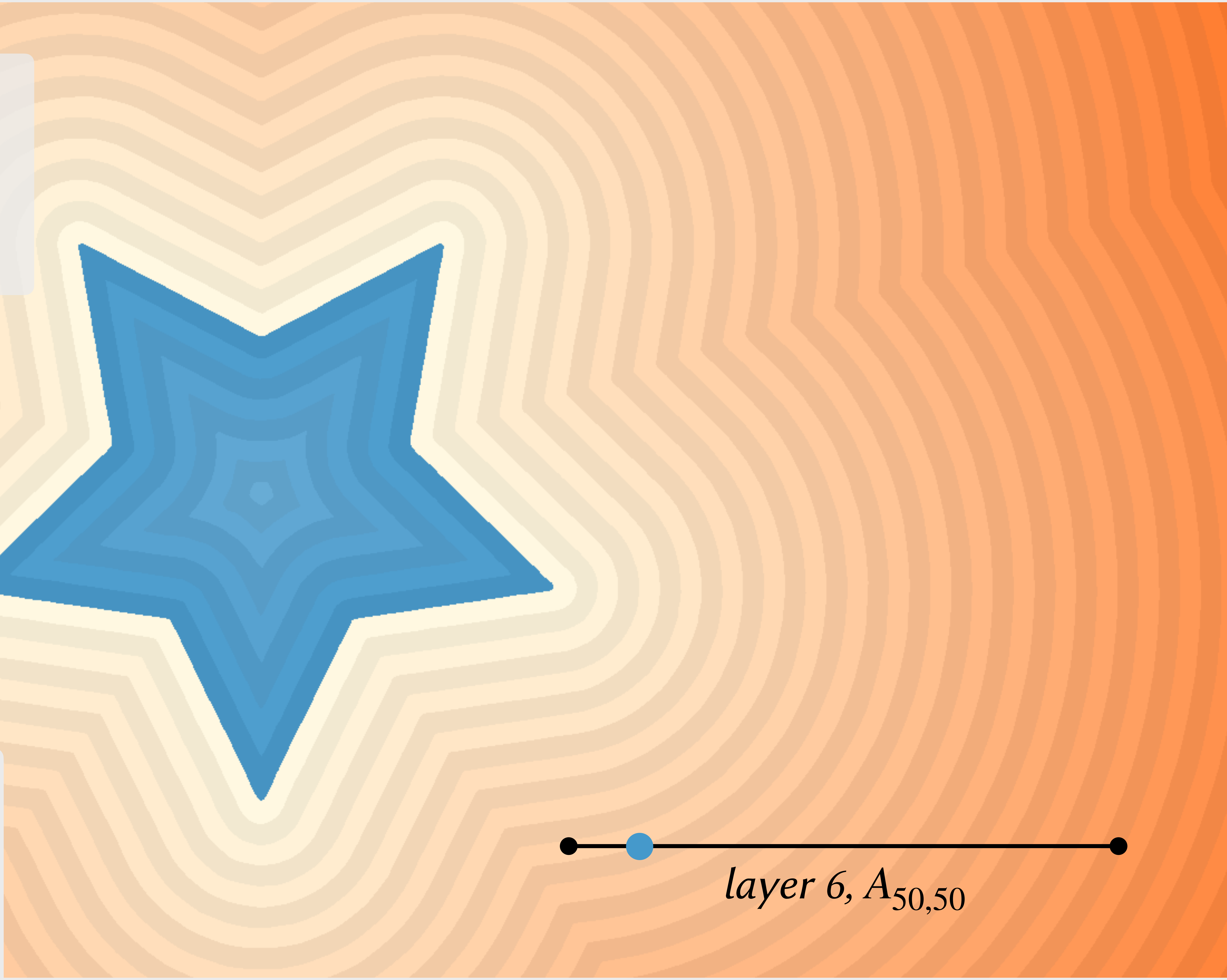
$$\min_{\theta} \mathscr{L}(f_\theta)$$

WHY NOT?

1. **Imprecision**
2. Editing (etc.) is difficult
3. The wrong representation

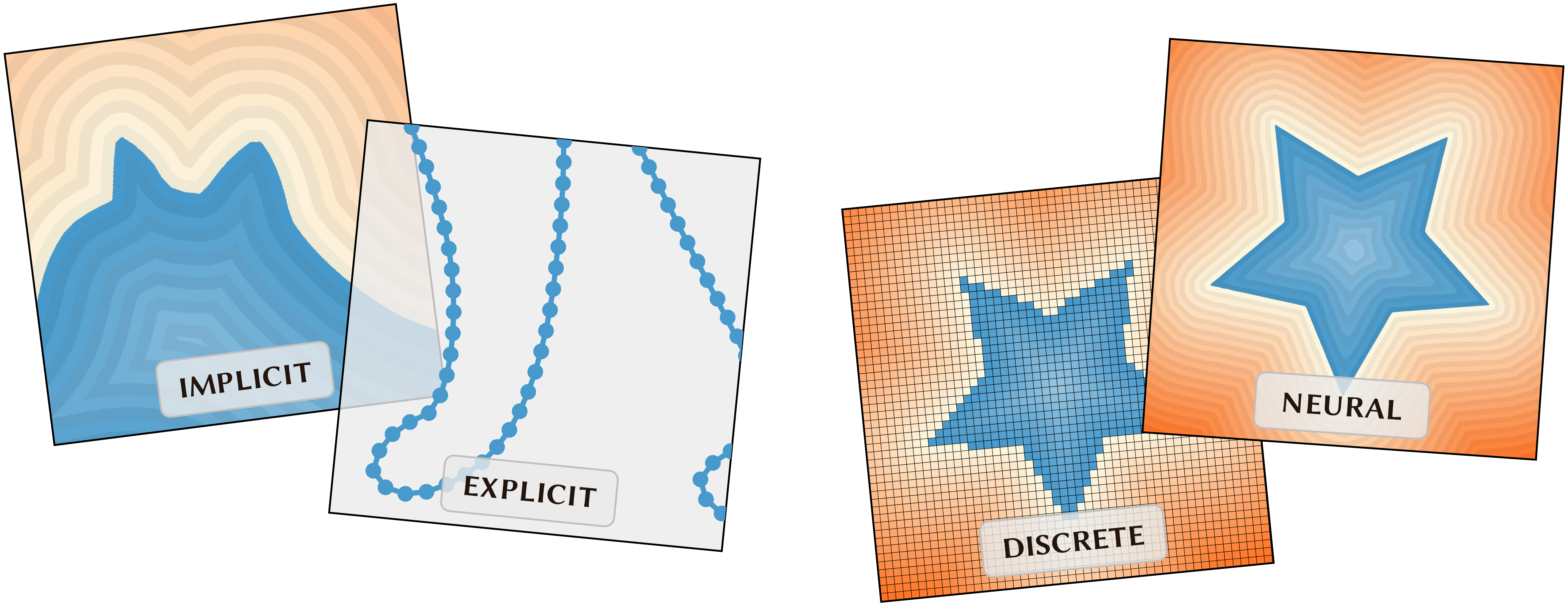
Image: "Fourier Features," Tancik et al., 2020

- geometric data stored in weights of neural network ⟶ downstream operations can be difficult
  - e.g., editing



WHY NOT?

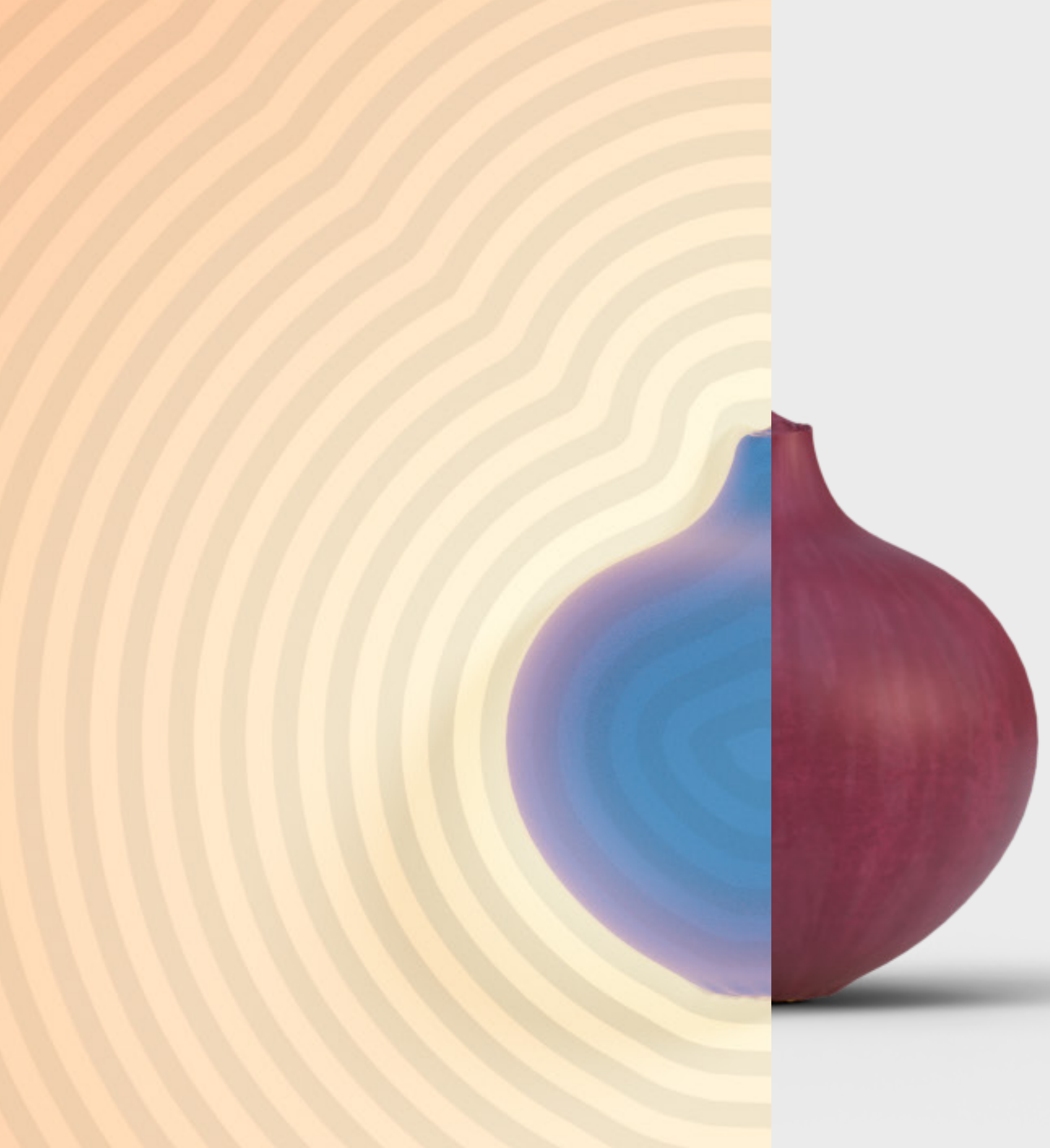
1. Imprecision
2. **Editing (etc.) is difficult**
3. The wrong representation



WHY NOT?

1. Imprecision
2. Editing (etc.) is difficult
3. ✨ **The wrong representation** ✨

Pick the right representation for your problem!



OVERVIEW

IMPLICIT SURFACES

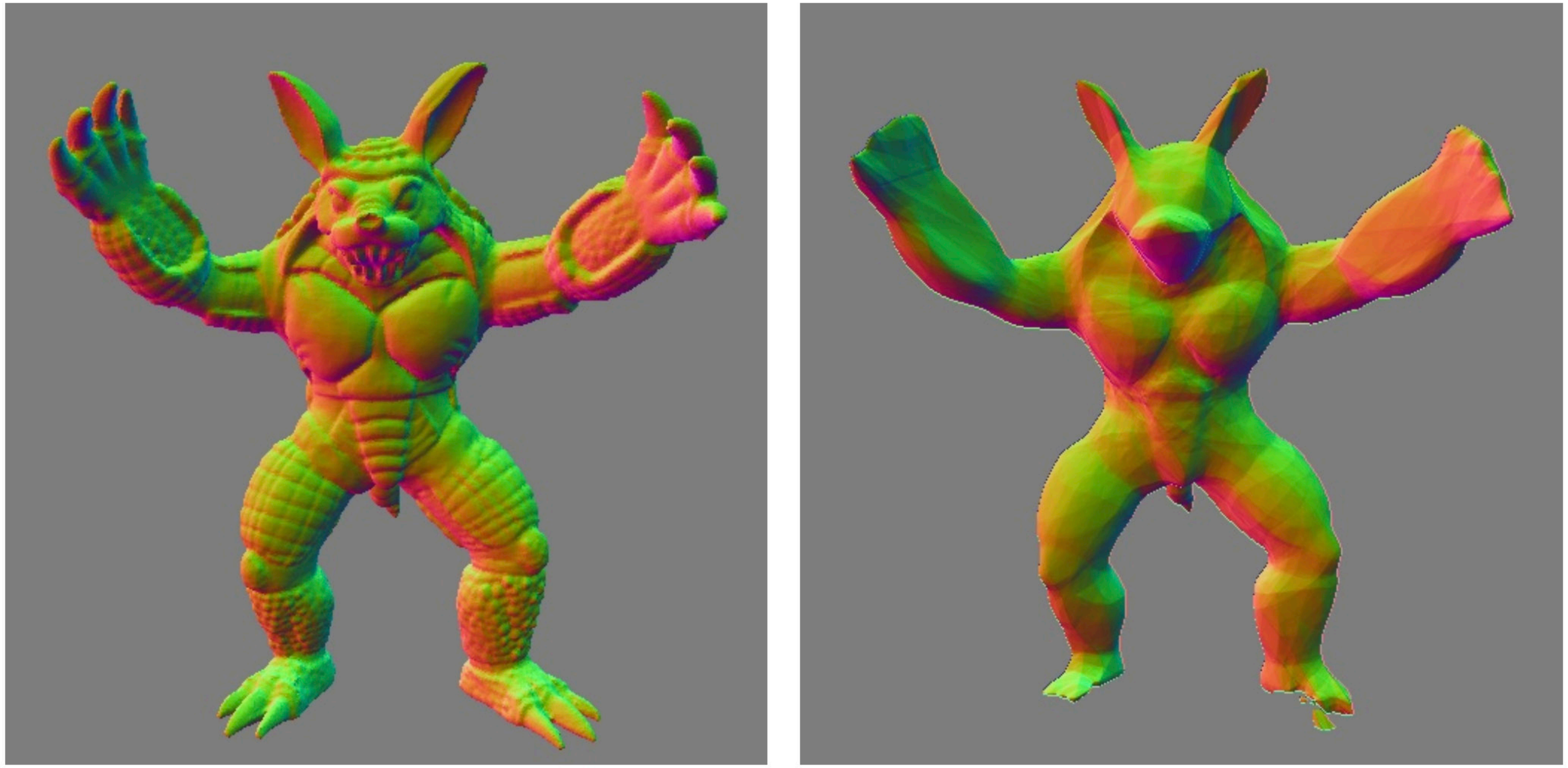
1. Background & History
2. Digital Representations

NEURAL IMPLICITS

3. The Basics
4. Why & Why Not Use Them?
5. Working out the Details
6. Geometric Operations

CONCLUSION

Sharp features in neural implicits



Sharp features are hard for standard techniques to capture!

Sharp features in neural implicits

arXiv:2006.10739v1 [cs.CV] 18 Jun 2020

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

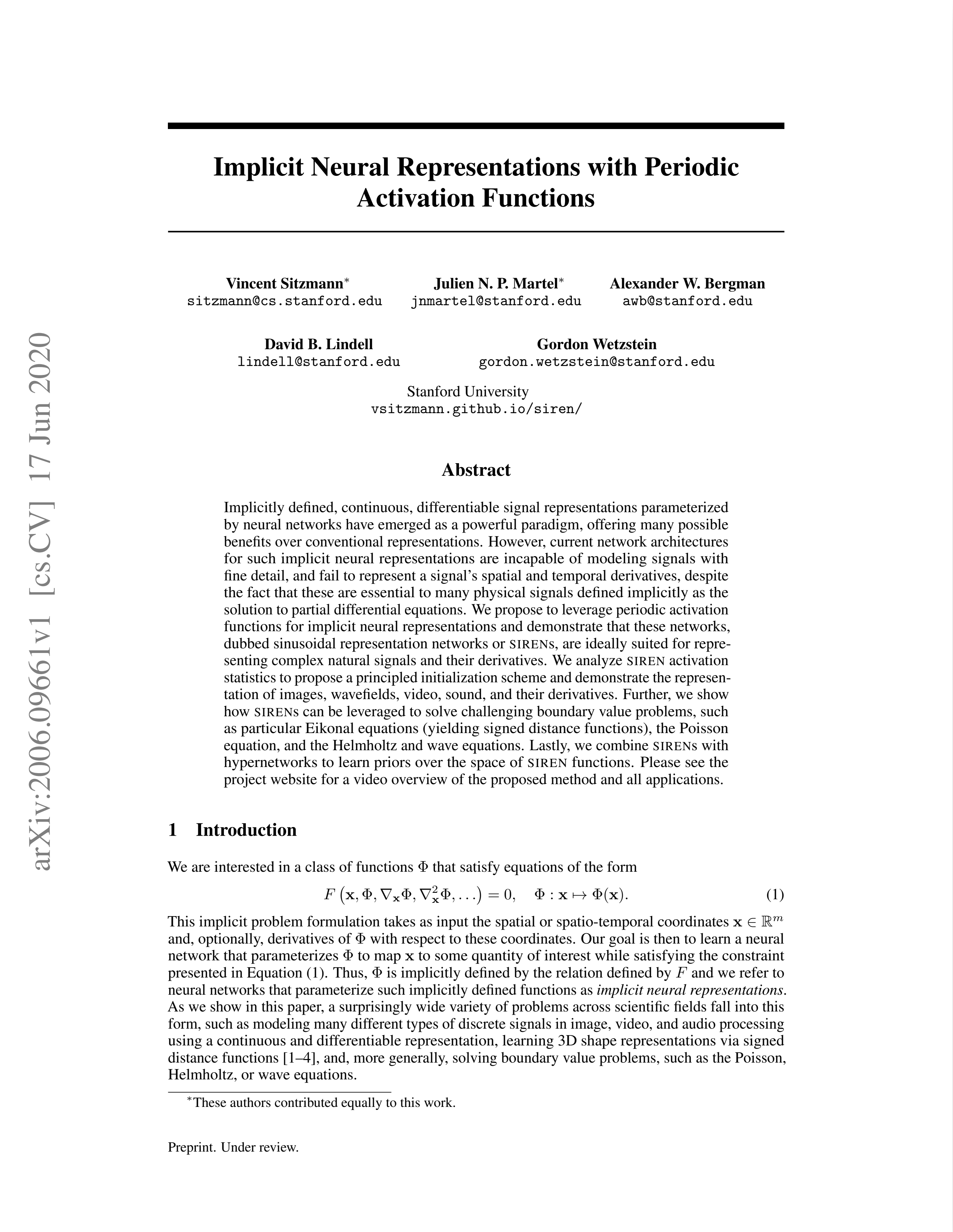
A recent line of research in computer vision and graphics replaces traditional discrete representations of objects, scene geometry, and appearance (*e.g.* meshes and voxel grids) with continuous functions parameterized by deep fully-connected networks (also called multilayer perceptrons or MLPs). These MLPs, which we will call "coordinate-based" MLPs, take low-dimensional coordinates as inputs (typically points in $\mathbb{R}^3$) and are trained to output a representation of shape, density, and/or color at each input location (see Figure 1). This strategy is compelling since coordinate-based MLPs are amenable to gradient-based optimization and machine learning, and can be orders of magnitude more compact than grid-sampled representations. Coordinate-based MLPs have been used to represent images [28, 38] (referred to as "compositional pattern producing networks"), volume density [27], occupancy [24], and signed distance [32], and have achieved state-of-the-art results across a variety of tasks such as shape representation [9, 10, 12, 13, 17, 26, 32], texture synthesis [15, 31], shape inference from images [22, 23], and novel view synthesis [27, 29, 35, 37].

We leverage recent progress in modeling the behavior of deep networks using kernel regression with a neural tangent kernel (NTK) [16] to theoretically and experimentally show that standard MLPs are poorly suited for these low-dimensional coordinate-based vision and graphics tasks. In particular, MLPs have difficulty learning high frequency functions, a phenomenon referred to in the literature as "spectral bias" [3, 33]. NTK theory suggests that this is because standard coordinate-based MLPs correspond to kernels with a rapid frequency falloff, which effectively prevents them from being able to represent the high-frequency content present in natural images and scenes.

A few recent works [27, 44] have experimentally found that a heuristic sinusoidal mapping of input coordinates (called a "positional encoding") allows MLPs to represent higher frequency content.

* Authors contributed equally to this work. Preprint. Under review.

Fourier Features [Tancik et al., 2020]



arXiv:2006.09661v1 [cs.CV] 17 Jun 2020

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 the fact that these are essential to many physical signals defined implicitly as the solution to partial differential equations. We propose to leverage periodic activation functions for implicit neural representations and demonstrate that these networks, dubbed sinusoidal representation networks or SIRENs, are ideally suited for representing complex natural signals and their derivatives. We analyze SIREN activation statistics to propose a principled initialization scheme and demonstrate the representation of images, wavefields, video, sound, and their derivatives. Further, we show how SIRENs can be leveraged to solve challenging boundary value problems, such as particular Eikonal equations (yielding signed distance functions), the Poisson equation, and the Helmholtz and wave equations. Lastly, we combine SIRENs with hypernetworks to learn priors over the space of SIREN functions. Please see the project website for a video overview of the proposed method and all applications.

1 Introduction

We are interested in a class of functions Φ that satisfy equations of the form

$$F\left(\mathbf{x}, \Phi, \nabla_{\mathbf{x}}\Phi, \nabla^2_{\mathbf{x}}\Phi, \ldots\right) = 0, \quad \Phi : \mathbf{x} \mapsto \Phi(\mathbf{x}). \tag{1}$$

This implicit problem formulation takes as input the spatial or spatio-temporal coordinates $\mathbf{x} \in \mathbb{R}^m$ and, optionally, derivatives of Φ with respect to these coordinates. Our goal is then to learn a neural network that parameterizes Φ to map $\mathbf{x}$ to some quantity of interest while satisfying the constraint presented in Equation (1). Thus, Φ is implicitly defined by the relation defined by F and we refer to neural networks that parameterize such implicitly defined functions as *implicit neural representations*. As we show in this paper, a surprisingly wide variety of problems across scientific fields fall into this form, such as modeling many different types of discrete signals in image, video, and audio processing using a continuous and differentiable representation, learning 3D shape representations via signed distance functions [1–4], and, more generally, solving boundary value problems, such as the Poisson, Helmholtz, or wave equations.

*These authors contributed equally to this work.

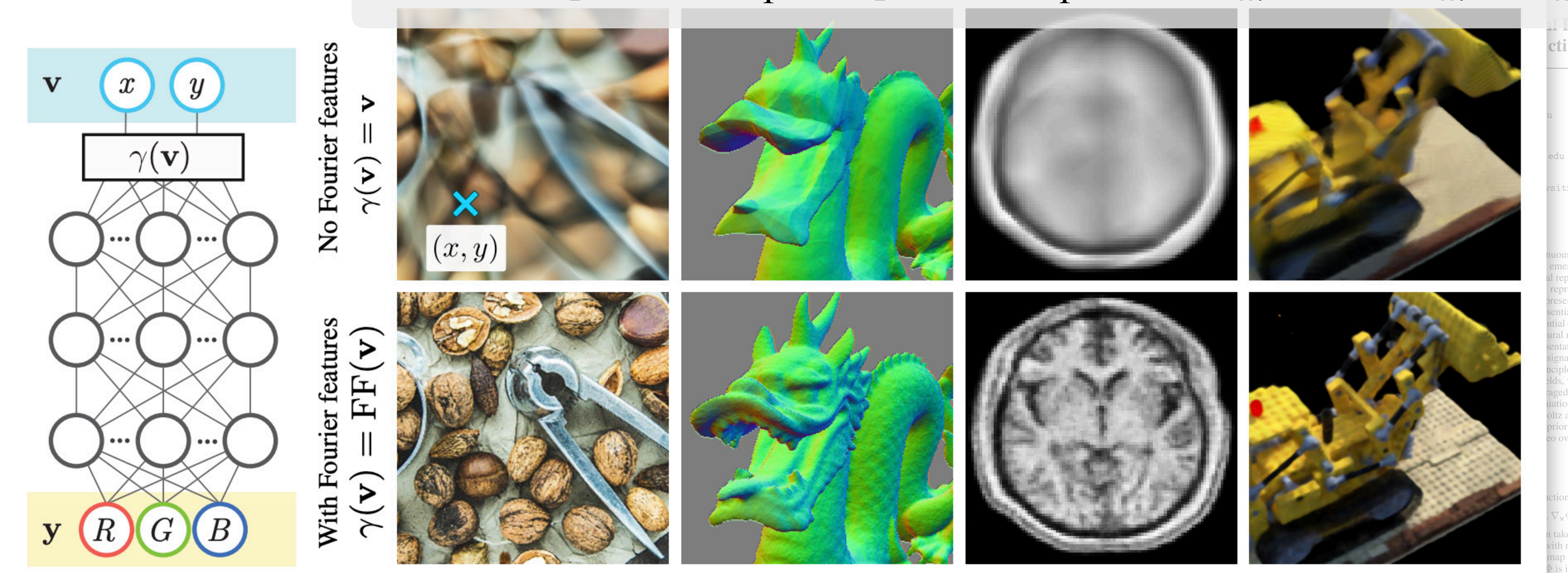
Preprint. Under review.

SIREN [Sitzmann et al., 2020]

Sharp features in neural implicits

augment input points with "Fourier features"

$$\gamma(\mathbf{v}) = [a_1 \cos(2\pi \mathbf{b}_1^T \mathbf{v}), a_1 \sin(2\pi \mathbf{b}_1^T \mathbf{v}), \ldots, a_m \cos(2\pi \mathbf{b}_m^T \mathbf{v}), a_m \sin(2\pi \mathbf{b}_m^T \mathbf{v})]$$

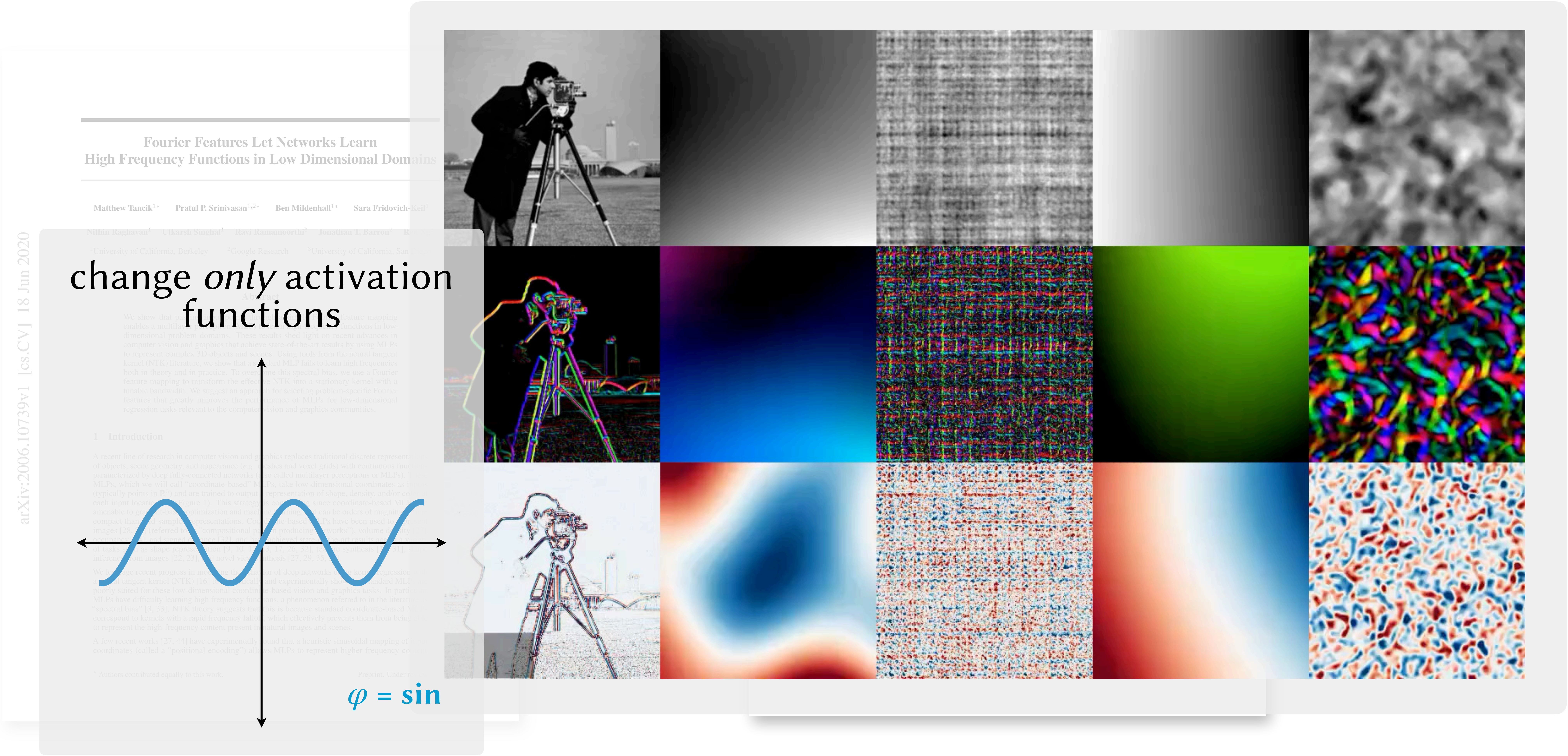


(a) Coordinate-based MLP
(b) Image regression $(x,y) \to$ RGB
(c) 3D shape regression $(x,y,z) \to$ occupancy
(d) MRI reconstruction $(x,y,z) \to$ density
(e) Inverse rendering $(x,y,z) \to$ RGB, density

Fourier Features [Tancik et al., 2020]

SIREN [Sitzmann et al., 2020]

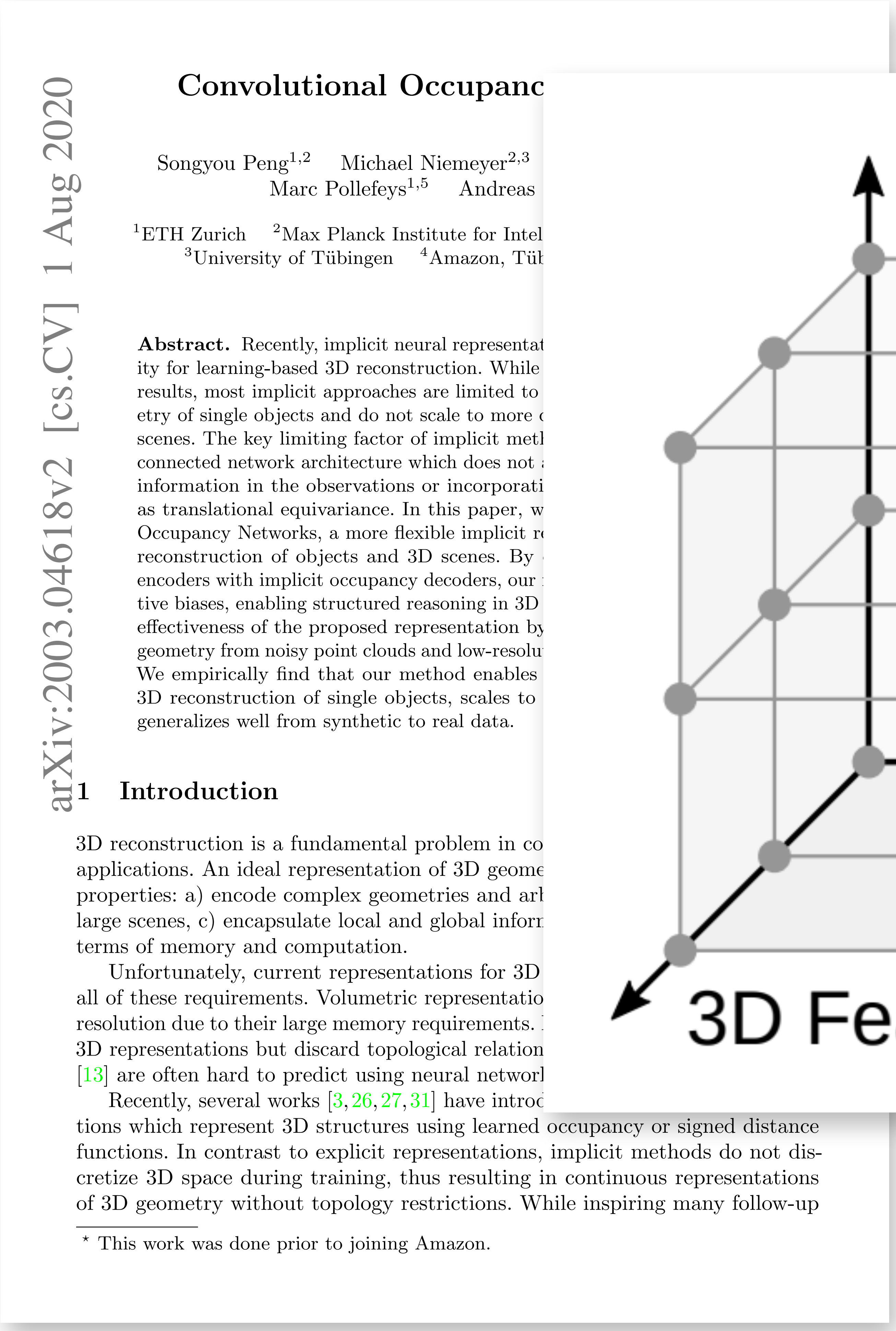
Sharp features in neural implicits



Fourier Features [Tancik et al., 2020]

SIREN [Sitzmann et al., 2020]

Hybrid discrete/continuous fields



arXiv:2003.04618v2 [cs.CV] 1 Aug 2020

Convolutional Occupanc

Songyou Peng[1,2] Michael Niemeyer[2,3]
Marc Pollefeys[1,5] Andreas

[1]ETH Zurich [2]Max Planck Institute for Intel
[3]University of Tübingen [4]Amazon, Tüb

Abstract. Recently, implicit neural representat ity for learning-based 3D reconstruction. While results, most implicit approaches are limited to etry of single objects and do not scale to more scenes. The key limiting factor of implicit meth connected network architecture which does not information in the observations or incorporati as translational equivariance. In this paper, w Occupancy Networks, a more flexible implicit re reconstruction of objects and 3D scenes. By encoders with implicit occupancy decoders, our tive biases, enabling structured reasoning in 3D effectiveness of the proposed representation by geometry from noisy point clouds and low-resolu We empirically find that our method enables 3D reconstruction of single objects, scales to generalizes well from synthetic to real data.

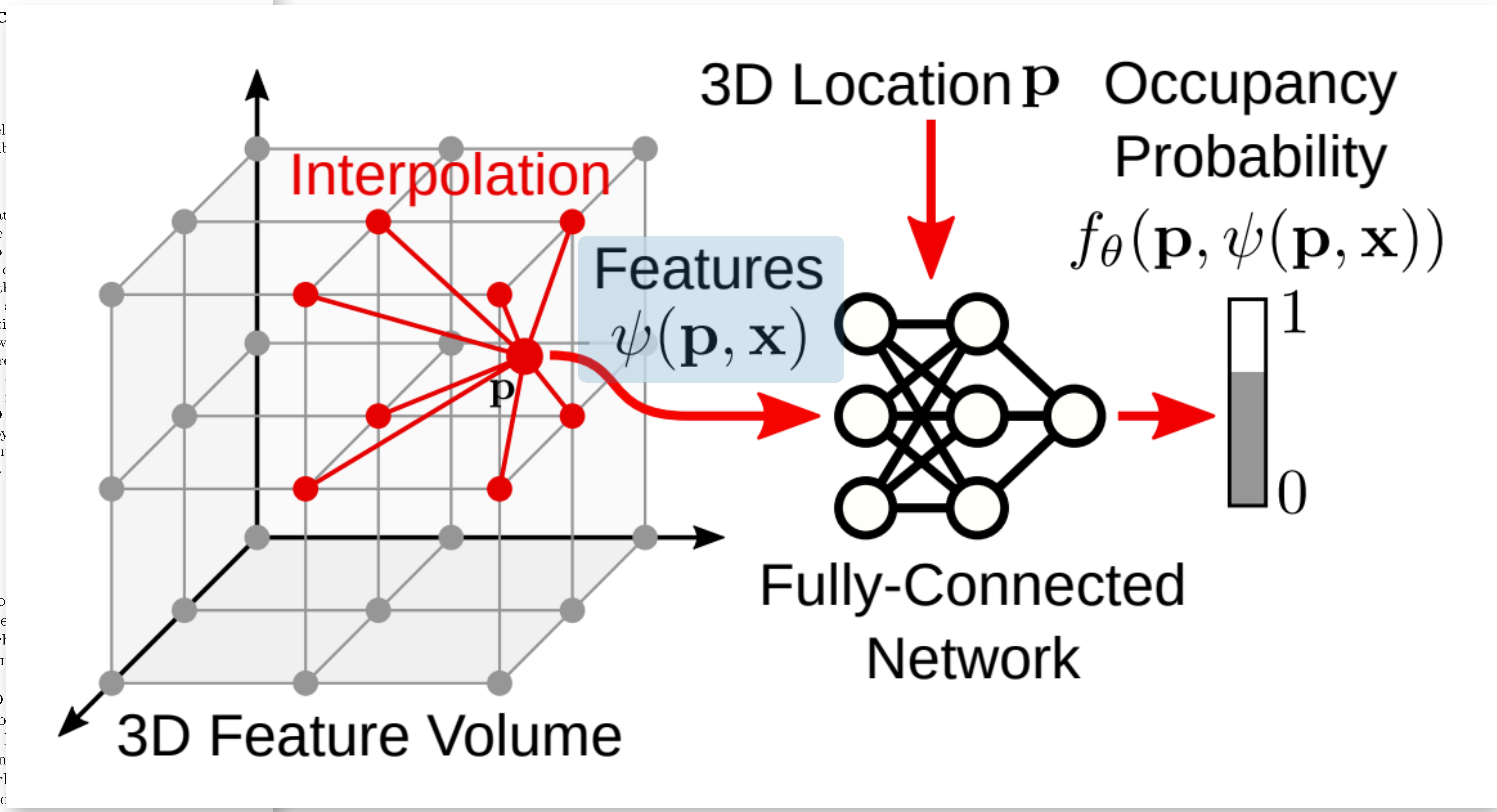
1 Introduction

3D reconstruction is a fundamental problem in co applications. An ideal representation of 3D geome properties: a) encode complex geometries and arb large scenes, c) encapsulate local and global infor terms of memory and computation.

Unfortunately, current representations for 3D all of these requirements. Volumetric representatio resolution due to their large memory requirements. 3D representations but discard topological relation [13] are often hard to predict using neural networ

Recently, several works [3,26,27,31] have introd tions which represent 3D structures using learned occupancy or signed distance functions. In contrast to explicit representations, implicit methods do not discretize 3D space during training, thus resulting in continuous representations of 3D geometry without topology restrictions. While inspiring many follow-up

* This work was done prior to joining Amazon.



Convolutional Occupancy Network [Peng et al., 2020]

Hybrid discrete/continuous fields

Instant Neural Graphics Primitives with a Multiresolution Hash Encoding

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

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

Trained for 1 second | 15 seconds | 1 second | 15 seconds | 60 seconds | reference

Gigapixel image
SDF
NRC
NeRF

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

Neural graphics primitives, parameterized by fully connected neural networks, can be costly to train and evaluate. We reduce this cost with a versatile new input encoding that permits the use of a smaller network without sacrificing quality, thus significantly reducing the number of floating point and memory access operations: a small neural network is augmented by a multiresolution hash table of trainable feature vectors whose values are optimized through stochastic gradient descent. The multiresolution structure allows the network to disambiguate hash collisions, making for a simple architecture that is trivial to parallelize on modern GPUs. We leverage this parallelism by implementing the whole system using fully-fused CUDA kernels with a focus on minimizing wasted bandwidth and compute operations. We achieve a combined speedup of several orders of magnitude, enabling training of high-quality neural graphics primitives in a matter of seconds, and rendering in tens of milliseconds at a resolution of 1920×1080.

CCS Concepts: • **Computing methodologies** → *Massively parallel algorithms*; *Vector / streaming algorithms*; **Neural networks**.

Additional Key Words and Phrases: Image Synthesis, Neural Networks, Encodings, Hashing, GPUs, Parallel Computation, Function Approximation.

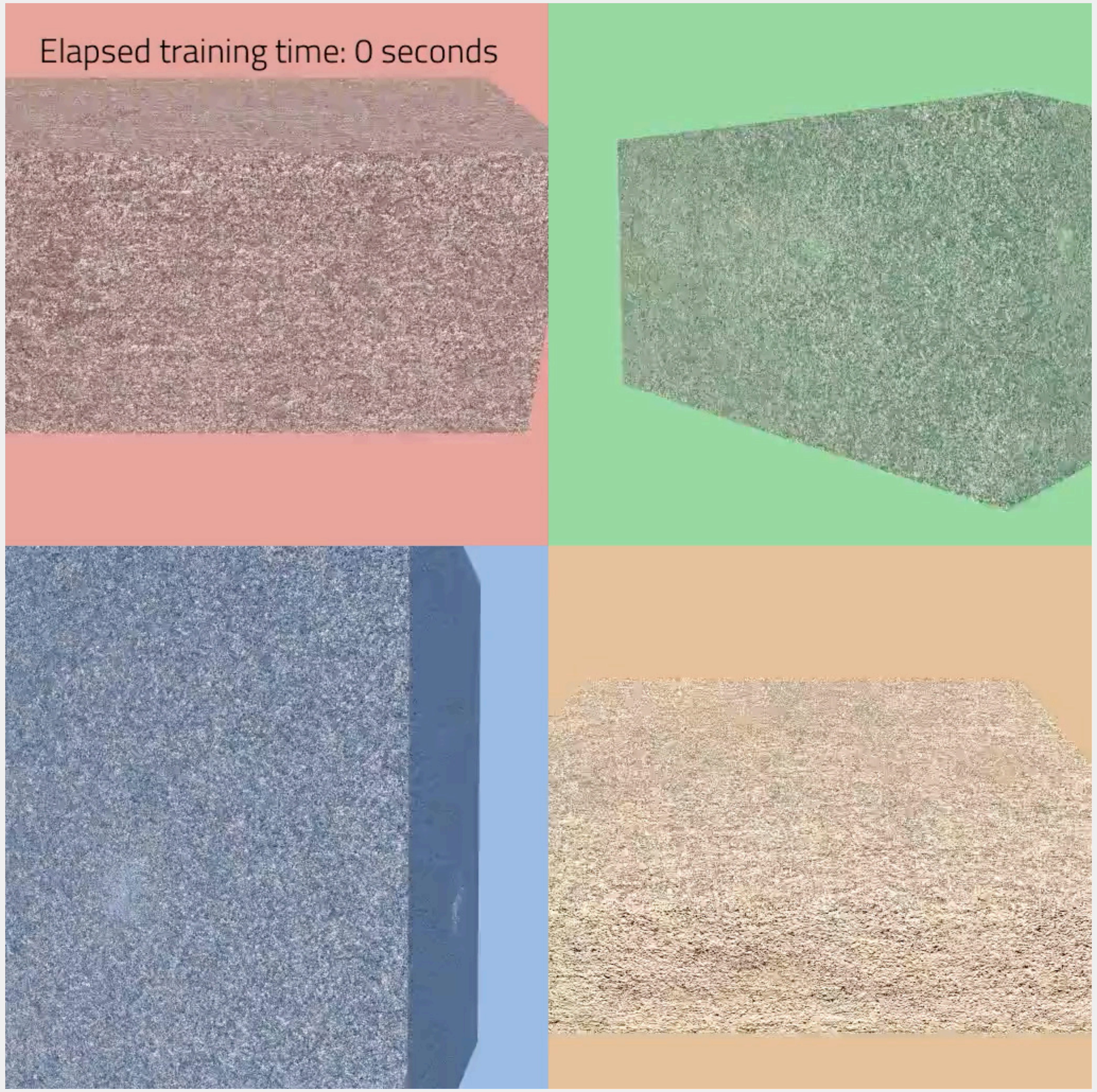
ACM Reference Format:
Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.* 41, 4, Article 102 (July 2022), 15 pages. https://doi.org/10.1145/3528223.3530127

Authors' addresses: Thomas Müller, NVIDIA, Zürich, Switzerland, tmueller@nvidia.com; Alex Evans, NVIDIA, London, United Kingdom, alexe@nvidia.com; Christoph Schied, NVIDIA, Seattle, USA, cschied@nvidia.com; Alexander Keller, NVIDIA, Berlin, Germany, akeller@nvidia.com.

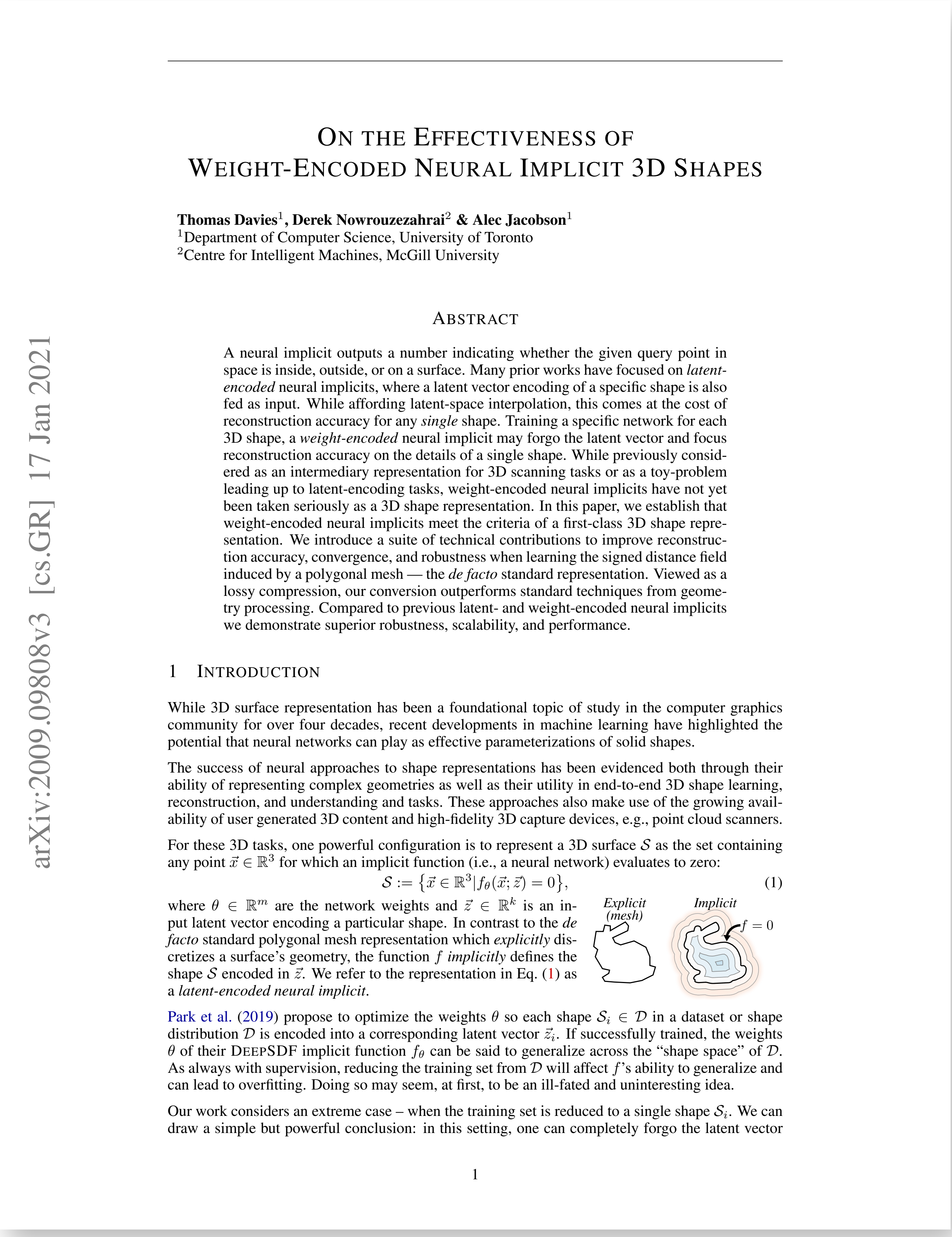
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *ACM Transactions on Graphics*, https://doi.org/10.1145/3528223.3530127.

ACM Trans. Graph., Vol. 41, No. 4, Article 102. Publication date: July 2022.

Instant NGP [Müller et al., 2022]



Training neural implicits: sampling points


arXiv:2009.09808v3 [cs.GR] 17 Jan 2021

ON THE EFFECTIVENESS OF
WEIGHT-ENCODED NEURAL IMPLICIT 3D SHAPES

Thomas Davies[1], Derek Nowrouzezahrai[2] & Alec Jacobson[1]
[1]Department of Computer Science, University of Toronto
[2]Centre for Intelligent Machines, McGill University

ABSTRACT

A neural implicit outputs a number indicating whether the given query point in space is inside, outside, or on a surface. Many prior works have focused on *latent-encoded* neural implicits, where a latent vector encoding of a specific shape is also fed as input. While affording latent-space interpolation, this comes at the cost of reconstruction accuracy for any *single* shape. Training a specific network for each 3D shape, a *weight-encoded* neural implicit may forgo the latent vector and focus reconstruction accuracy on the details of a single shape. While previously considered as an intermediary representation for 3D scanning tasks or as a toy-problem leading up to latent-encoding tasks, weight-encoded neural implicits have not yet been taken seriously as a 3D shape representation. In this paper, we establish that weight-encoded neural implicits meet the criteria of a first-class 3D shape representation. We introduce a suite of technical contributions to improve reconstruction accuracy, convergence, and robustness when learning the signed distance field induced by a polygonal mesh — the *de facto* standard representation. Viewed as a lossy compression, our conversion outperforms standard techniques from geometry processing. Compared to previous latent- and weight-encoded neural implicits we demonstrate superior robustness, scalability, and performance.

1 INTRODUCTION

While 3D surface representation has been a foundational topic of study in the computer graphics community for over four decades, recent developments in machine learning have highlighted the potential that neural networks can play as effective parameterizations of solid shapes.

The success of neural approaches to shape representations has been evidenced both through their ability of representing complex geometries as well as their utility in end-to-end 3D shape learning, reconstruction, and understanding and tasks. These approaches also make use of the growing availability of user generated 3D content and high-fidelity 3D capture devices, e.g., point cloud scanners.

For these 3D tasks, one powerful configuration is to represent a 3D surface $\mathcal{S}$ as the set containing any point $\vec{x} \in \mathbb{R}^3$ for which an implicit function (i.e., a neural network) evaluates to zero:

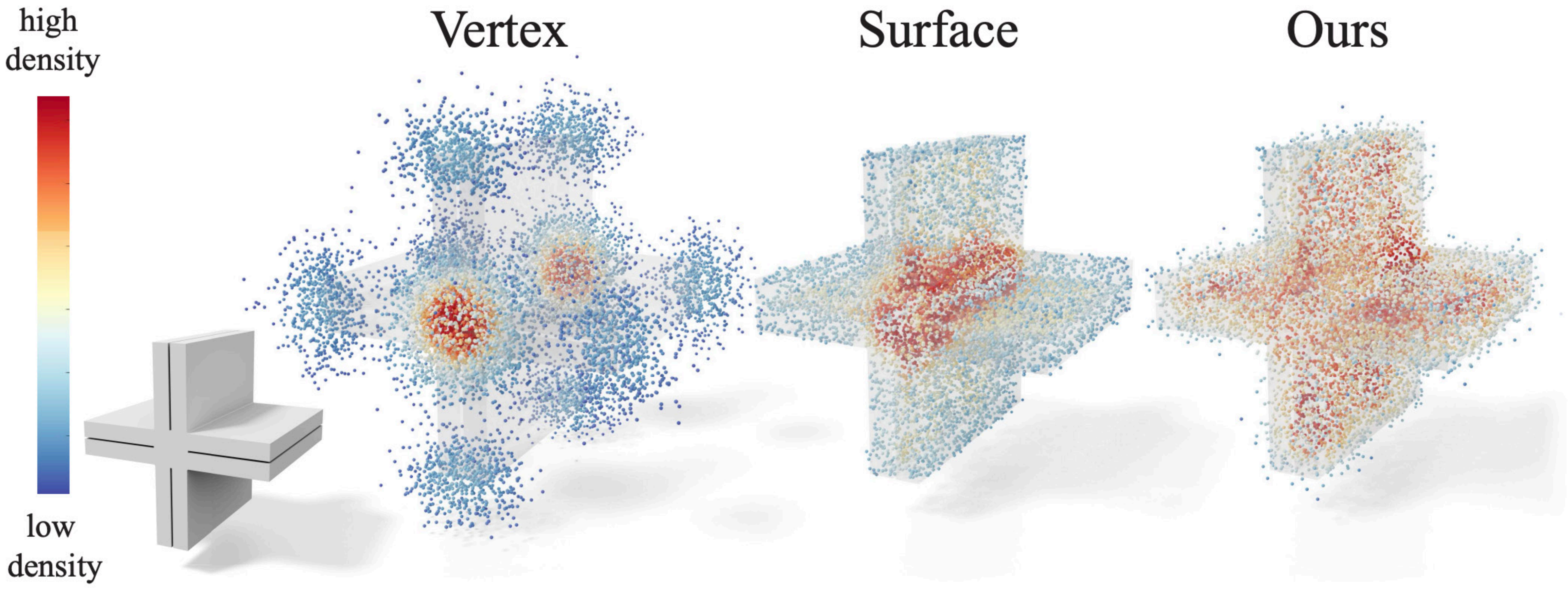
$$\mathcal{S} := \{\vec{x} \in \mathbb{R}^3 | f_\theta(\vec{x}; \vec{z}) = 0\}, \quad (1)$$

where $\theta \in \mathbb{R}^m$ are the network weights and $\vec{z} \in \mathbb{R}^k$ is an input latent vector encoding a particular shape. In contrast to the *de facto* standard polygonal mesh representation which *explicitly* discretizes a surface's geometry, the function f *implicitly* defines the shape $\mathcal{S}$ encoded in $\vec{z}$. We refer to the representation in Eq. (1) as a *latent-encoded neural implicit*.

Park et al. (2019) propose to optimize the weights θ so each shape $\mathcal{S}_i \in \mathcal{D}$ in a dataset or shape distribution $\mathcal{D}$ is encoded into a corresponding latent vector $\vec{z}_i$. If successfully trained, the weights θ of their DEEPSDF implicit function f_θ can be said to generalize across the "shape space" of $\mathcal{D}$. As always with supervision, reducing the training set from $\mathcal{D}$ will affect f's ability to generalize and can lead to overfitting. Doing so may seem, at first, to be an ill-fated and uninteresting idea.

Our work considers an extreme case – when the training set is reduced to a single shape $\mathcal{S}_i$. We can draw a simple but powerful conclusion: in this setting, one can completely forgo the latent vector

1

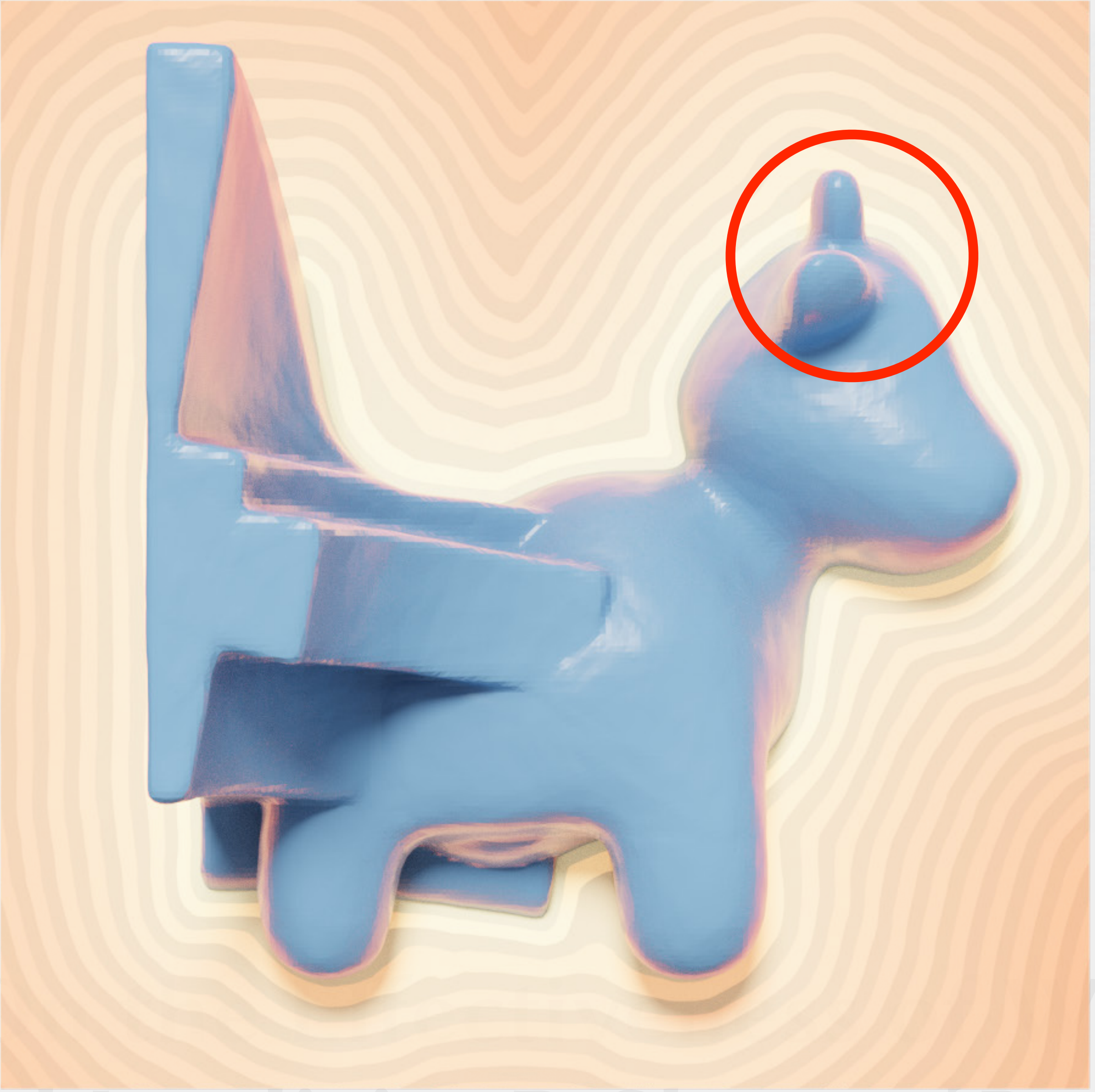


from **On the Effectiveness of Weight-Encoded Neural Implicit 3D Shapes** [Davies et al., 2021]

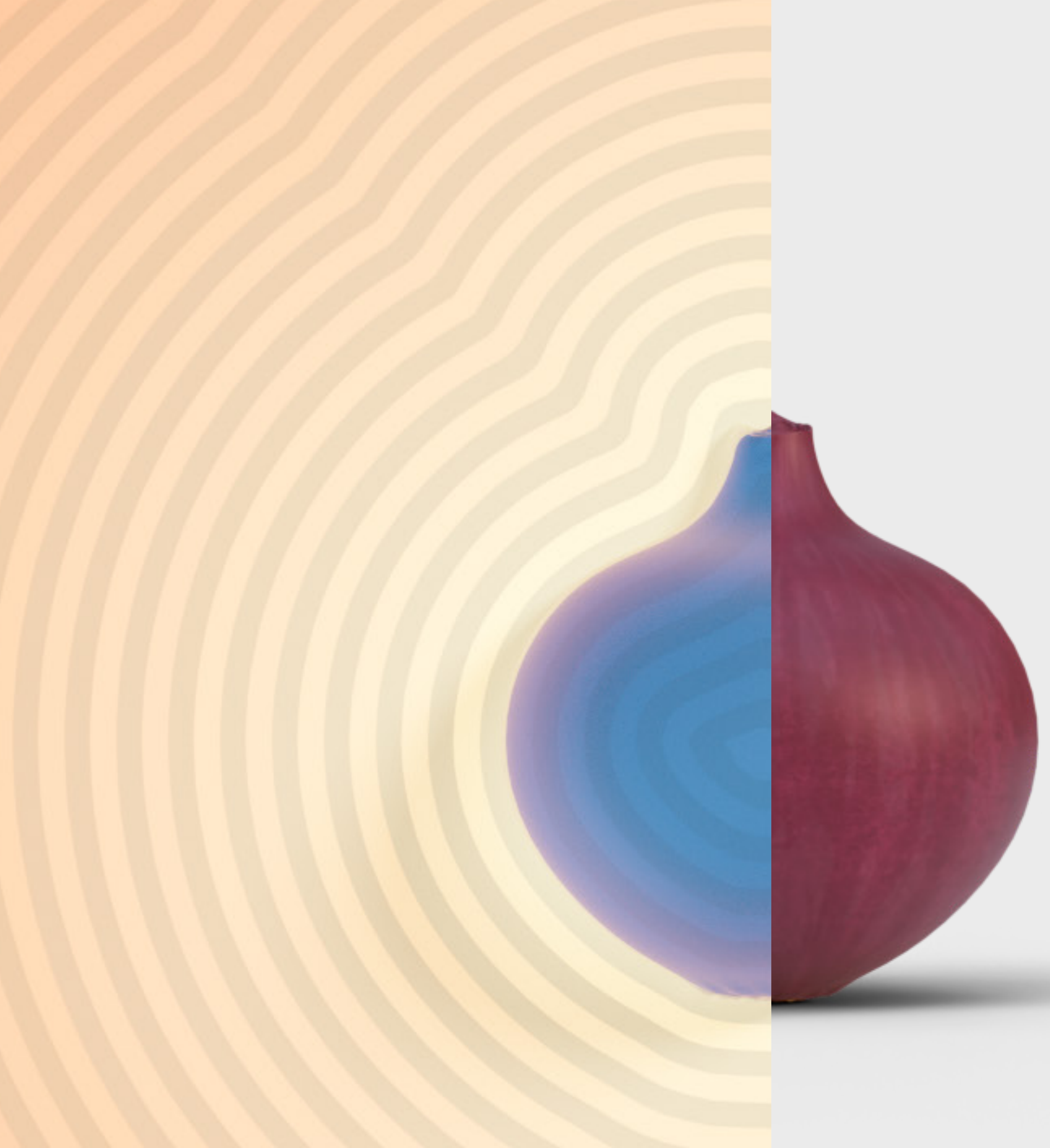
Training neural implicits: sampling points

care more about reconstruction quality near surface $\implies$ bias loss, achieved via importance sampling near surface

stratified sampling



importance sampling



OVERVIEW

IMPLICIT SURFACES

1. Background & History
2. Digital Representations

NEURAL IMPLICITS

3. The Basics
4. Why & Why Not Use Them?
5. Working out the Details
6. Geometric Operations

CONCLUSION

Signed Distance Function

- $|f(\boldsymbol{x})|$ encodes distance to closest point on surface
- many geometric queries easy

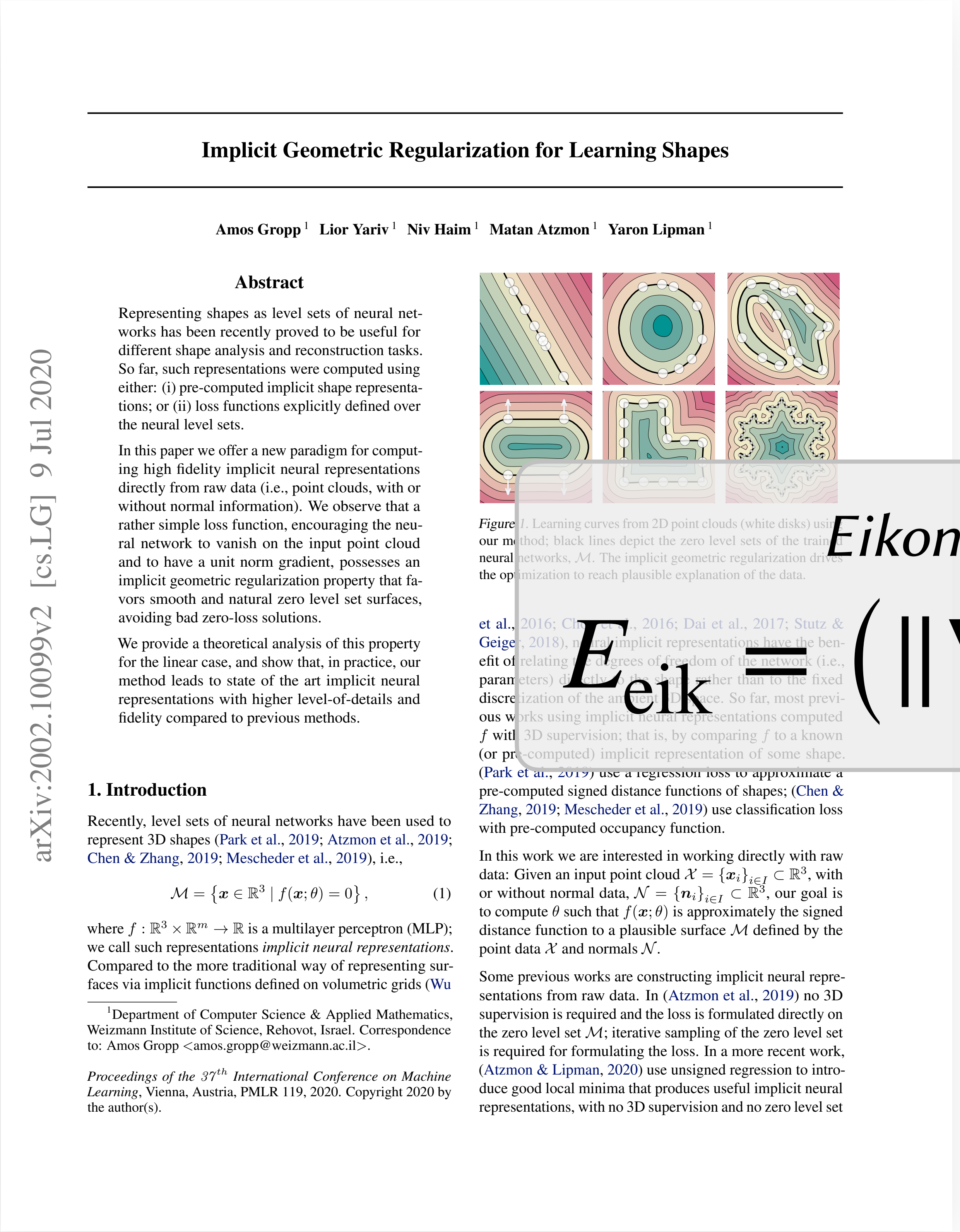
How do we make a neural implicit an SDF?

How do we make a neural implicit an SDF?

$$\mathscr{L}(x) = \mathscr{L}_{\text{org}}(x) + \mathscr{L}_{\text{SDF}}(x)$$

add regularization term to loss function

Eikonal loss



arXiv:2002.10099v2 [cs.LG] 9 Jul 2020

Implicit Geometric Regularization for Learning Shapes

Amos Gropp [1] Lior Yariv [1] Niv Haim [1] Matan Atzmon [1] Yaron Lipman [1]

Abstract

Representing shapes as level sets of neural networks has been recently proved to be useful for different shape analysis and reconstruction tasks. So far, such representations were computed using either: (i) pre-computed implicit shape representations; or (ii) loss functions explicitly defined over the neural level sets.

In this paper we offer a new paradigm for computing high fidelity implicit neural representations directly from raw data (i.e., point clouds, with or without normal information). We observe that a rather simple loss function, encouraging the neural network to vanish on the input point cloud and to have a unit norm gradient, possesses an implicit geometric regularization property that favors smooth and natural zero level set surfaces, avoiding bad zero-loss solutions.

We provide a theoretical analysis of this property for the linear case, and show that, in practice, our method leads to state of the art implicit neural representations with higher level-of-details and fidelity compared to previous methods.

Figure 1. Learning curves from 2D point clouds (white disks) using our method; black lines depict the zero level sets of the trained neural networks, $\mathcal{M}$. The implicit geometric regularization drives the optimization to reach plausible explanation of the data.

1. Introduction

Recently, level sets of neural networks have been used to represent 3D shapes (Park et al., 2019; Atzmon et al., 2019; Chen & Zhang, 2019; Mescheder et al., 2019), i.e.,

$$\mathcal{M} = \{ \boldsymbol{x} \in \mathbb{R}^3 \mid f(\boldsymbol{x}; \theta) = 0 \}, \quad (1)$$

where $f : \mathbb{R}^3 \times \mathbb{R}^m \to \mathbb{R}$ is a multilayer perceptron (MLP); we call such representations *implicit neural representations*. Compared to the more traditional way of representing surfaces via implicit functions defined on volumetric grids (Wu et al., 2016; Choy et al., 2016; Dai et al., 2017; Stutz & Geiger, 2018), neural implicit representations have the benefit of relating the degrees of freedom of the network (i.e., parameters) directly to the shape rather than to the fixed discretization of the ambient 3D space. So far, most previous works using implicit neural representations computed f with 3D supervision; that is, by comparing f to a known (or pre-computed) implicit representation of some shape. (Park et al., 2019) use a regression loss to approximate a pre-computed signed distance functions of shapes; (Chen & Zhang, 2019; Mescheder et al., 2019) use classification loss with pre-computed occupancy function.

In this work we are interested in working directly with raw data: Given an input point cloud $\mathcal{X} = \{\boldsymbol{x}_i\}_{i \in I} \subset \mathbb{R}^3$, with or without normal data, $\mathcal{N} = \{\boldsymbol{n}_i\}_{i \in I} \subset \mathbb{R}^3$, our goal is to compute θ such that $f(\boldsymbol{x}; \theta)$ is approximately the signed distance function to a plausible surface $\mathcal{M}$ defined by the point data $\mathcal{X}$ and normals $\mathcal{N}$.

Some previous works are constructing implicit neural representations from raw data. In (Atzmon et al., 2019) no 3D supervision is required and the loss is formulated directly on the zero level set $\mathcal{M}$; iterative sampling of the zero level set is required for formulating the loss. In a more recent work, (Atzmon & Lipman, 2020) use unsigned regression to introduce good local minima that produces useful implicit neural representations, with no 3D supervision and no zero level set

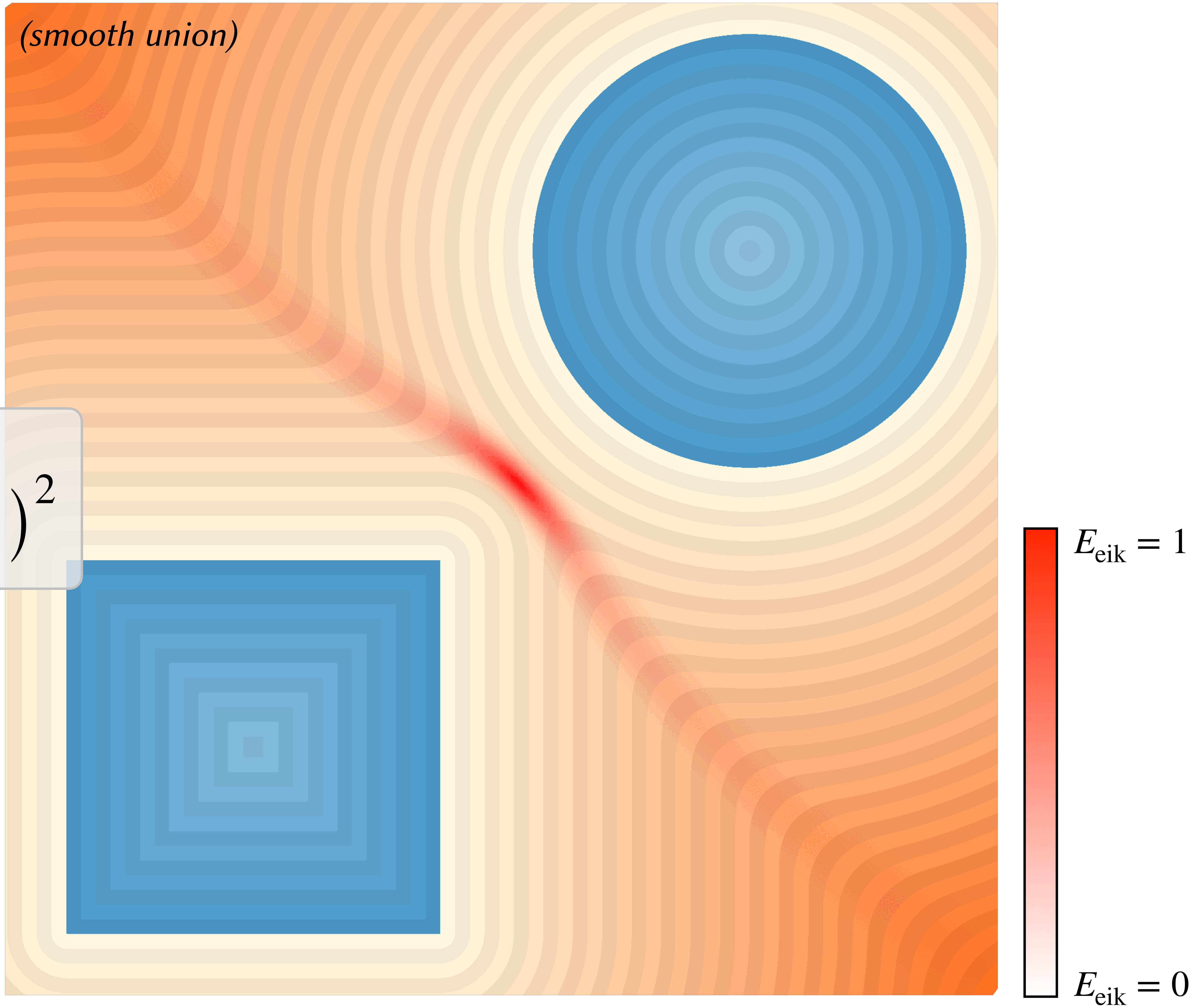
[1] Department of Computer Science & Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel. Correspondence to: Amos Gropp <amos.gropp@weizmann.ac.il>.

Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

Eikonal Loss

$$E_{\text{eik}} = \left(\|\nabla f_\theta\| - 1 \right)^2$$

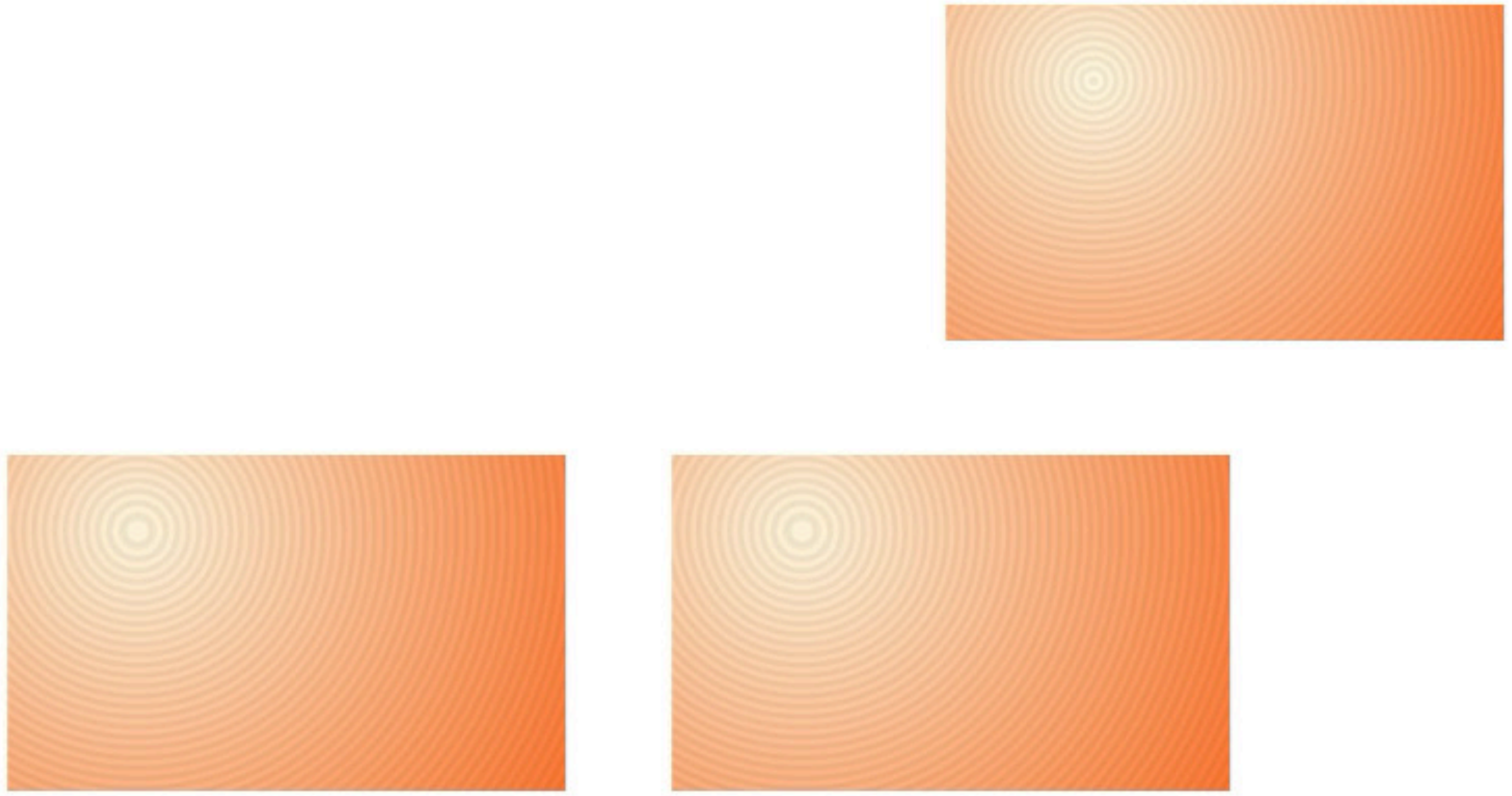
[Gropp et. al, 2020]

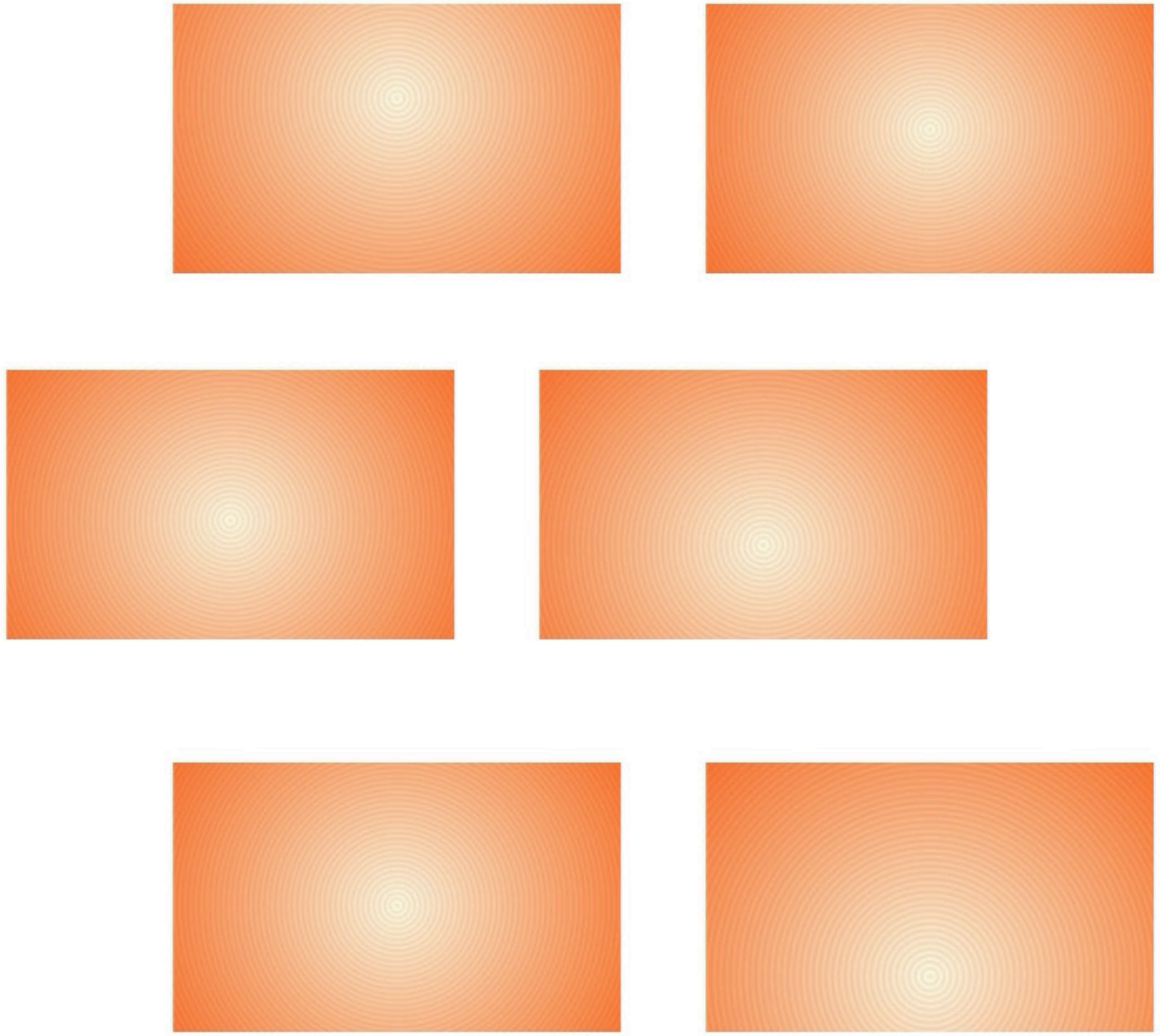


Eikonal loss: walkthrough

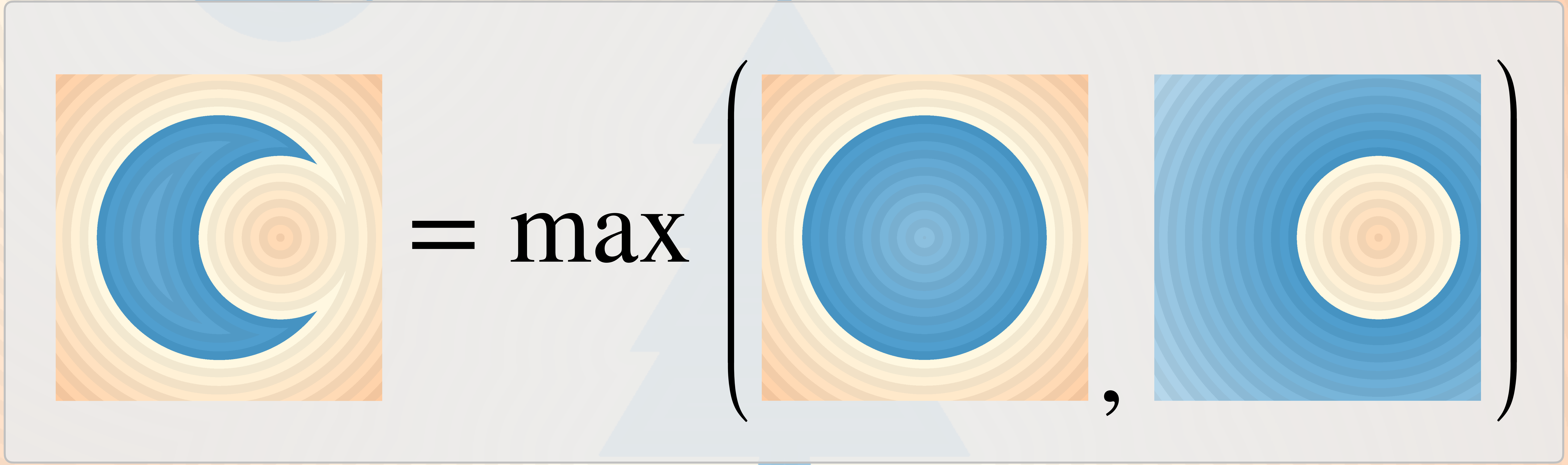
Eikonal loss: walkthrough

Eikonal but not SDF





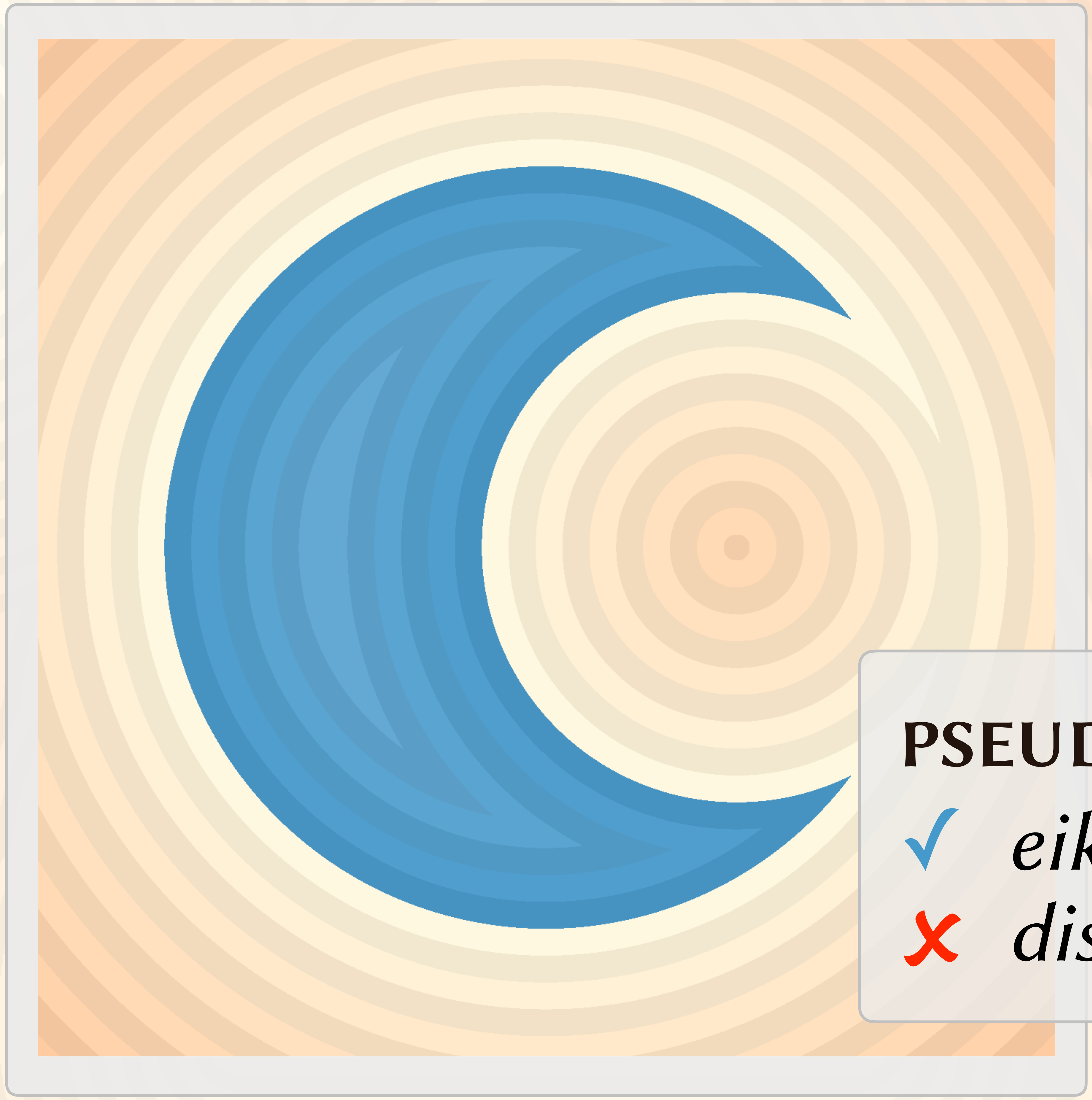
Eikonal but not SDF



Eikonal but not SDF

Eikonal but not SDF

Eikonal but not SDF



PSEUDO-SDF

✓ *eikonality*

✗ *distance property*

Eikonal loss

Eikonal loss

Closest point loss

Constructive Solid Geometry on Neural Signed Distance Fields

Zoë Marschner
zoem@cmu.edu
Massachusetts Institute of Technology
Carnegie Mellon University
United States

Silvia Sellán
sgsellan@cs.toronto.edu
University of Toronto
Canada

Hsueh-Ti Derek Liu
hsuehtil@gmail.com
Roblox Research
University of Toronto
Canada

Alec Jacobson
jacobson@cs.toronto.edu
University of Toronto
Adobe Research
Canada

Figure 1: Our method allows for the computation of exact neural SDFs of CSG operations. Here, we train one network to learn the swept volume of a stellated dodecahedron shape, parametric over the control points of the cubic Bézier path it is swept along. Specific swept volumes within this parameter space are then unioned together and with cylinders, resulting in a neural implicit which thanks to our regularization term forms an exact SDF of the word "SDF."

ABSTRACT

Signed Distance Fields (SDFs) parameterized by neural networks have recently gained popularity as a fundamental geometric representation. However, editing the shape encoded by a neural SDF remains an open challenge. A tempting approach is to leverage common geometric operators (e.g., boolean operations), but such edits often lead to incorrect non-SDF outputs (which we call Pseudo-SDFs), preventing them from being used for downstream tasks. In this paper, we characterize the space of Pseudo-SDFs, which are eikonal yet not true distance functions, and derive the *closest point loss*, a novel regularizer that encourages the output to be an exact SDF. We demonstrate the applicability of our regularization to many operations in which traditional methods cause a Pseudo-SDF to arise, such as CSG and swept volumes, and produce a true (neural) SDF for the result of these operations.

SA Conference Papers '23, December 12–15, 2023, Sydney, NSW, Australia

ACM ISBN 979-8-4007-0315-7/23/12.
https://doi.org/10.1145/3610548.3618170

CCS CONCEPTS

• **Computing methodologies → Shape modeling**; **Shape representations**.

KEYWORDS

signed distance field, neural implicit, CSG, swept volumes

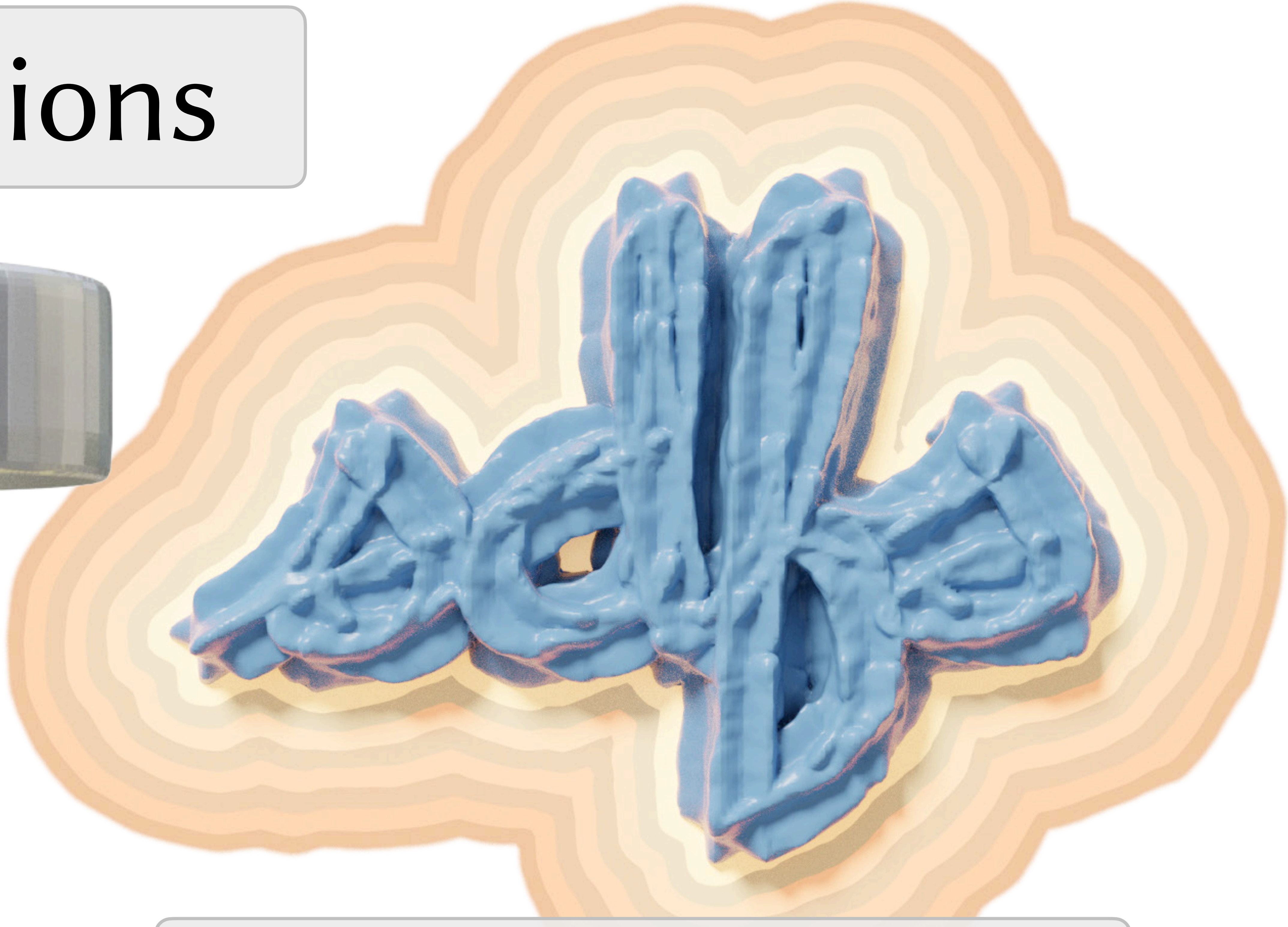
ACM Reference Format:
Zoë Marschner, Silvia Sellán, Hsueh-Ti Derek Liu, and Alec Jacobson. 2023. Constructive Solid Geometry on Neural Signed Distance Fields. In *SIGGRAPH Asia 2023 Conference Papers (SA Conference Papers '23), December 12–15, 2023, Sydney, NSW, Australia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3610548.3618170

1 INTRODUCTION

Neural implicit functions have gained attention as a fundamental representation of 3D objects due to their state-of-the-art performance in tasks like compression and reconstruction, as well as their generative power. They describe the boundary of a solid shape as

CSG on Neural SDFs
[Marschner et al., 2023]

CSG Operations



Swept Volumes


0.015
0.010
0.005
0.000
Eikonal Loss
avg. value = 7.1×10^{-5}

Closest point loss: walkthrough

Closest point loss: walkthrough

Closest point loss: walkthrough

Closest point loss: walkthrough

Closest point loss: walkthrough

Closest point loss: walkthrough

Closest point loss: walkthrough

Closest point loss: walkthrough

Closest Point Map

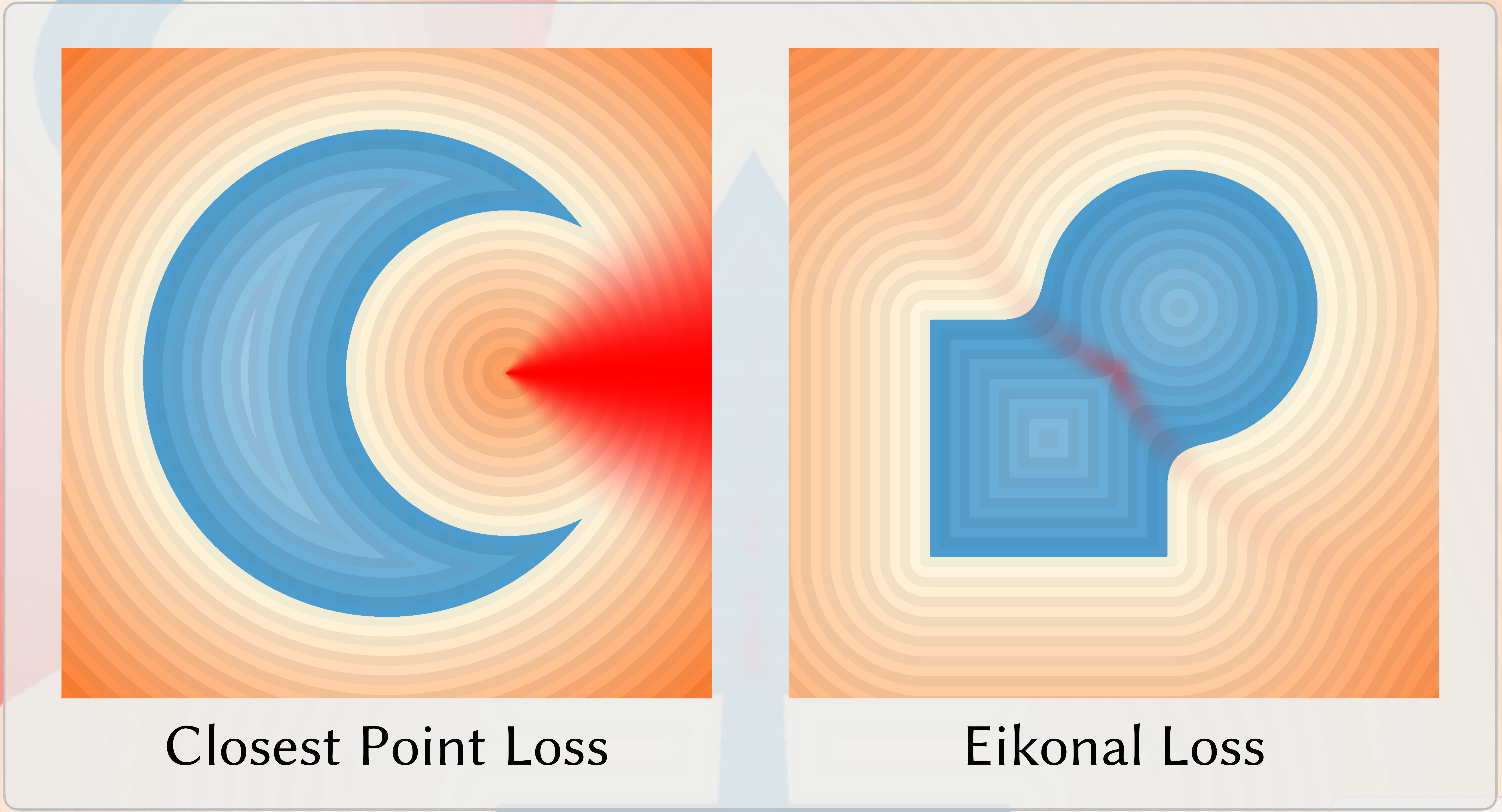
$$x \mapsto \boldsymbol{x} - f(\boldsymbol{x})\,\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$$

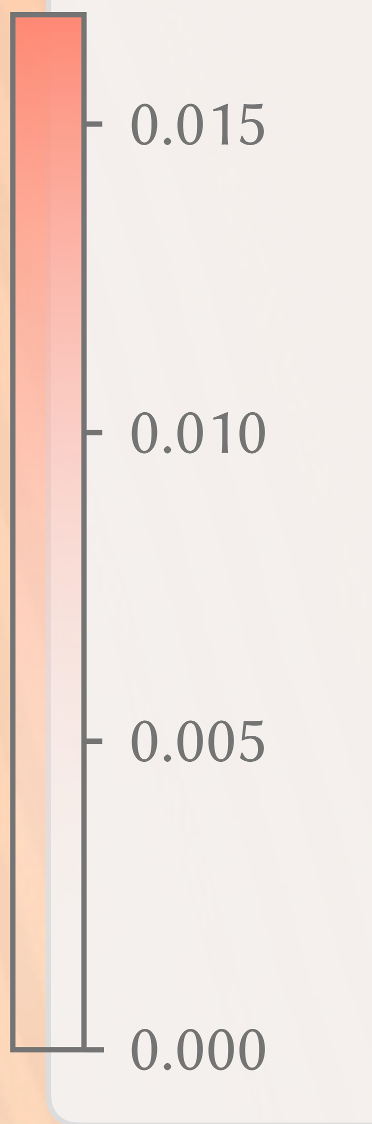
Closest point loss: walkthrough

Closest point loss: walkthrough

Closest Point Loss

$$E_{\mathrm{CP}} = f\left(\boldsymbol{x} - f(\boldsymbol{x})\,\nabla_{\boldsymbol{x}} f(\boldsymbol{x})\right)^2$$



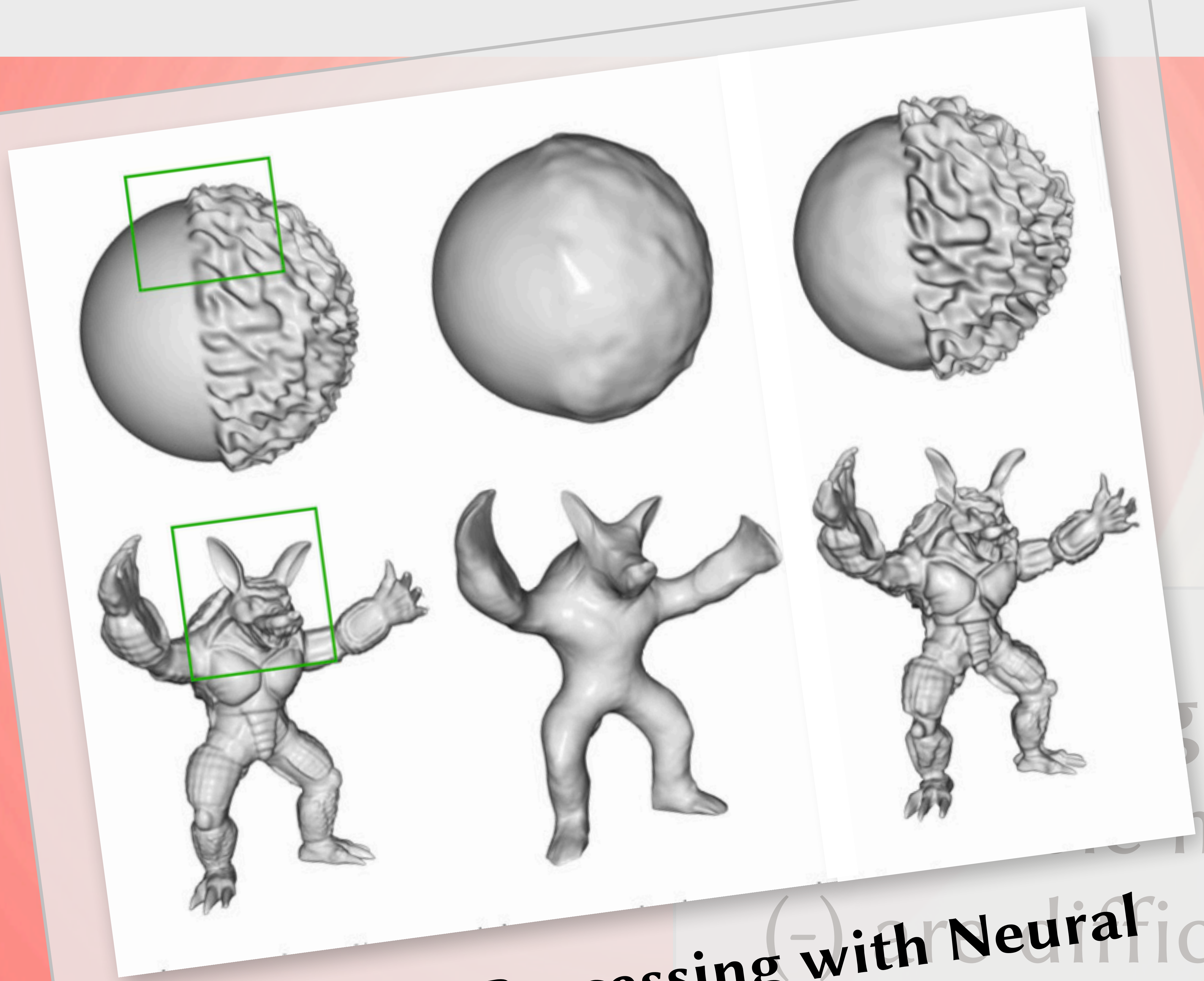


Closest Point Loss (ours)
avg. value = 0.003

Neural signed distance fields...

(+) enable many geometric operations

(-) are difficult to maintain

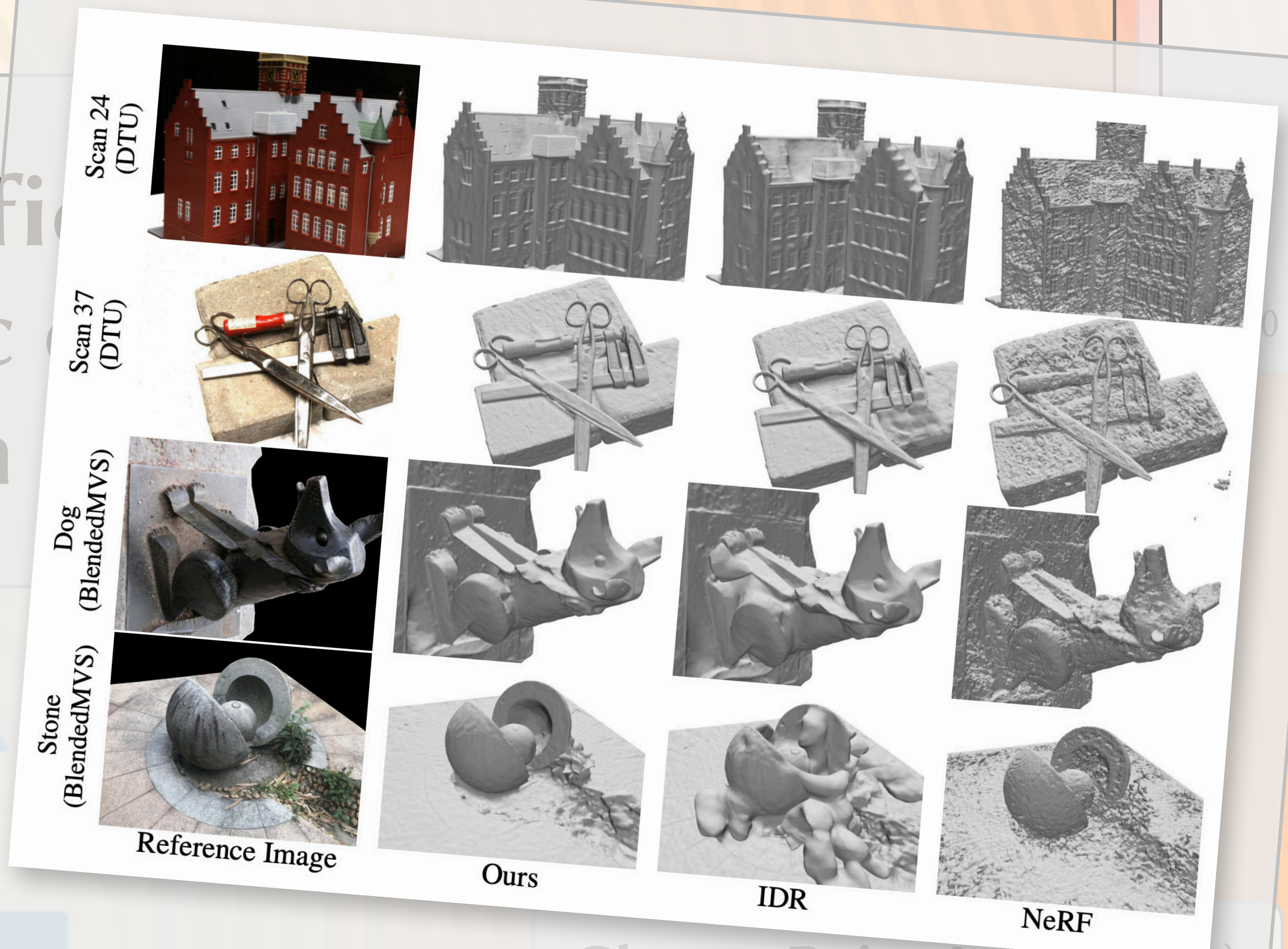


Geometry Processing with Neural Fields [Yang et al., 2021]

gned distance fi

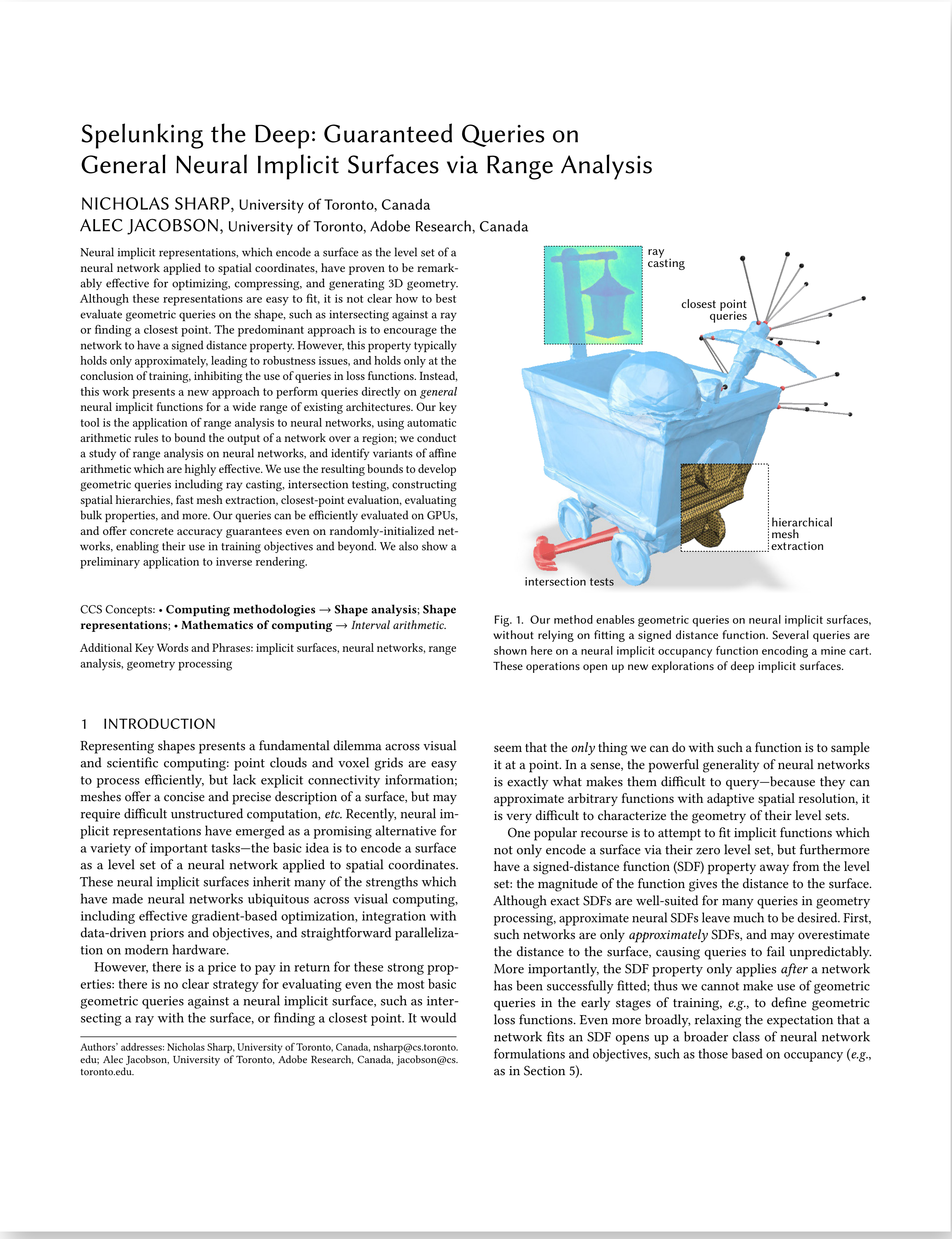
many geometric

(-) are difficult to maintain



NeuS [Wang et al., 2021]

Alternative to SDFs for GP



Spelunking the Deep: Guaranteed Queries on General Neural Implicit Surfaces via Range Analysis

NICHOLAS SHARP, University of Toronto, Canada
ALEC JACOBSON, University of Toronto, Adobe Research, Canada

Neural implicit representations, which encode a surface as the level set of a neural network applied to spatial coordinates, have proven to be remarkably effective for optimizing, compressing, and generating 3D geometry. Although these representations are easy to fit, it is not clear how to best evaluate geometric queries on the shape, such as intersecting against a ray or finding a closest point. The predominant approach is to encourage the network to have a signed distance property. However, this property typically holds only approximately, leading to robustness issues, and holds only at the conclusion of training, inhibiting the use of queries in loss functions. Instead, this work presents a new approach to perform queries directly on *general* neural implicit functions for a wide range of existing architectures. Our key tool is the application of range analysis to neural networks, using automatic arithmetic rules to bound the output of a network over a region; we conduct a study of range analysis on neural networks, and identify variants of affine arithmetic which are highly effective. We use the resulting bounds to develop geometric queries including ray casting, intersection testing, constructing spatial hierarchies, fast mesh extraction, closest-point evaluation, evaluating bulk properties, and more. Our queries can be efficiently evaluated on GPUs, and offer concrete accuracy guarantees even on randomly-initialized networks, enabling their use in training objectives and beyond. We also show a preliminary application to inverse rendering.

Fig. 1. Our method enables geometric queries on neural implicit surfaces, without relying on fitting a signed distance function. Several queries are shown here on a neural implicit occupancy function encoding a mine cart. These operations open up new explorations of deep implicit surfaces.

CCS Concepts: • **Computing methodologies** → **Shape analysis**; **Shape representations**; • **Mathematics of computing** → *Interval arithmetic.*

Additional Key Words and Phrases: implicit surfaces, neural networks, range analysis, geometry processing

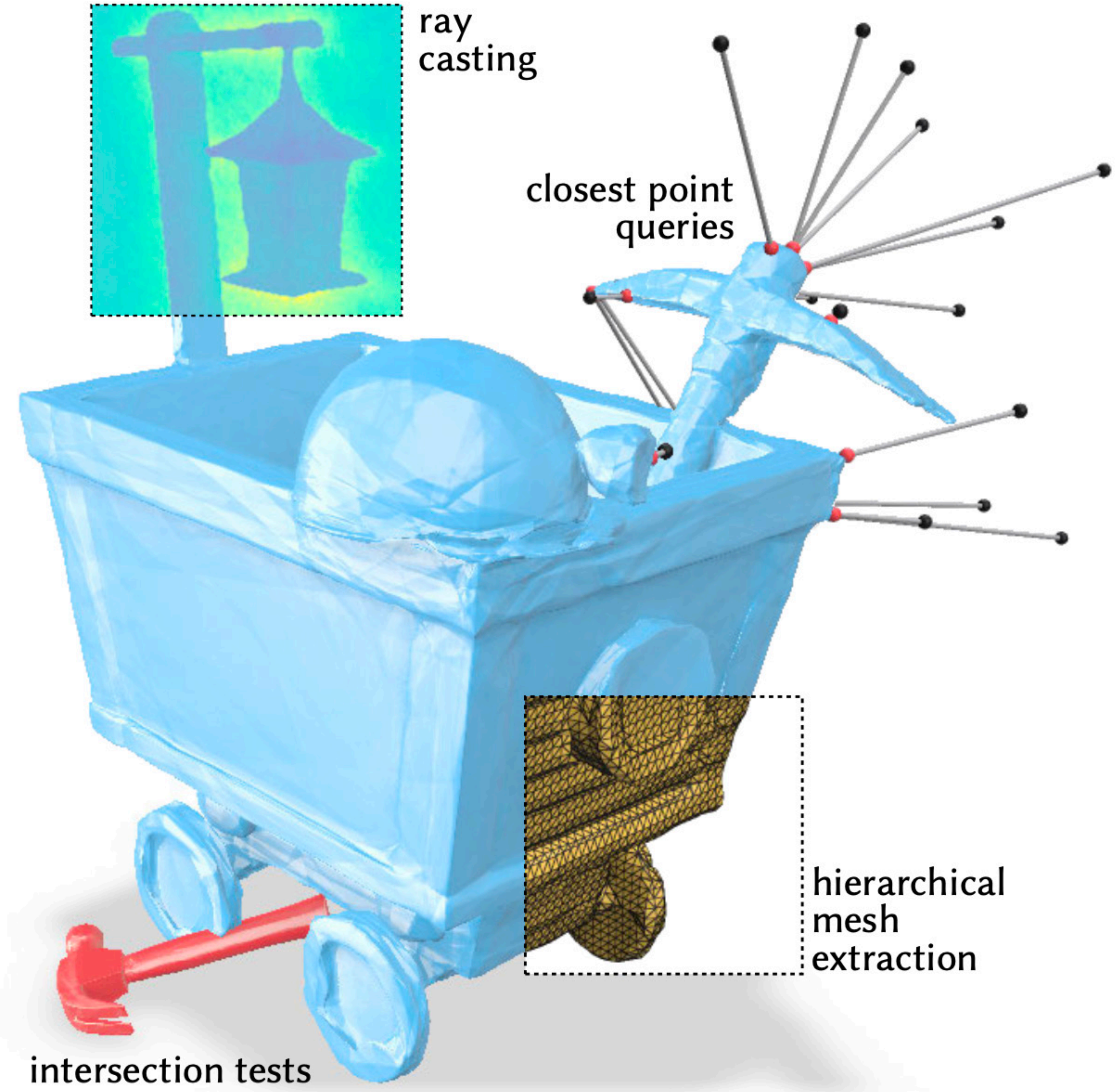
1 INTRODUCTION

Representing shapes presents a fundamental dilemma across visual and scientific computing: point clouds and voxel grids are easy to process efficiently, but lack explicit connectivity information; meshes offer a concise and precise description of a surface, but may require difficult unstructured computation, *etc.* Recently, neural implicit representations have emerged as a promising alternative for a variety of important tasks—the basic idea is to encode a surface as a level set of a neural network applied to spatial coordinates. These neural implicit surfaces inherit many of the strengths which have made neural networks ubiquitous across visual computing, including effective gradient-based optimization, integration with data-driven priors and objectives, and straightforward parallelization on modern hardware.

However, there is a price to pay in return for these strong properties: there is no clear strategy for evaluating even the most basic geometric queries against a neural implicit surface, such as intersecting a ray with the surface, or finding a closest point. It would seem that the *only* thing we can do with such a function is to sample it at a point. In a sense, the powerful generality of neural networks is exactly what makes them difficult to query—because they can approximate arbitrary functions with adaptive spatial resolution, it is very difficult to characterize the geometry of their level sets.

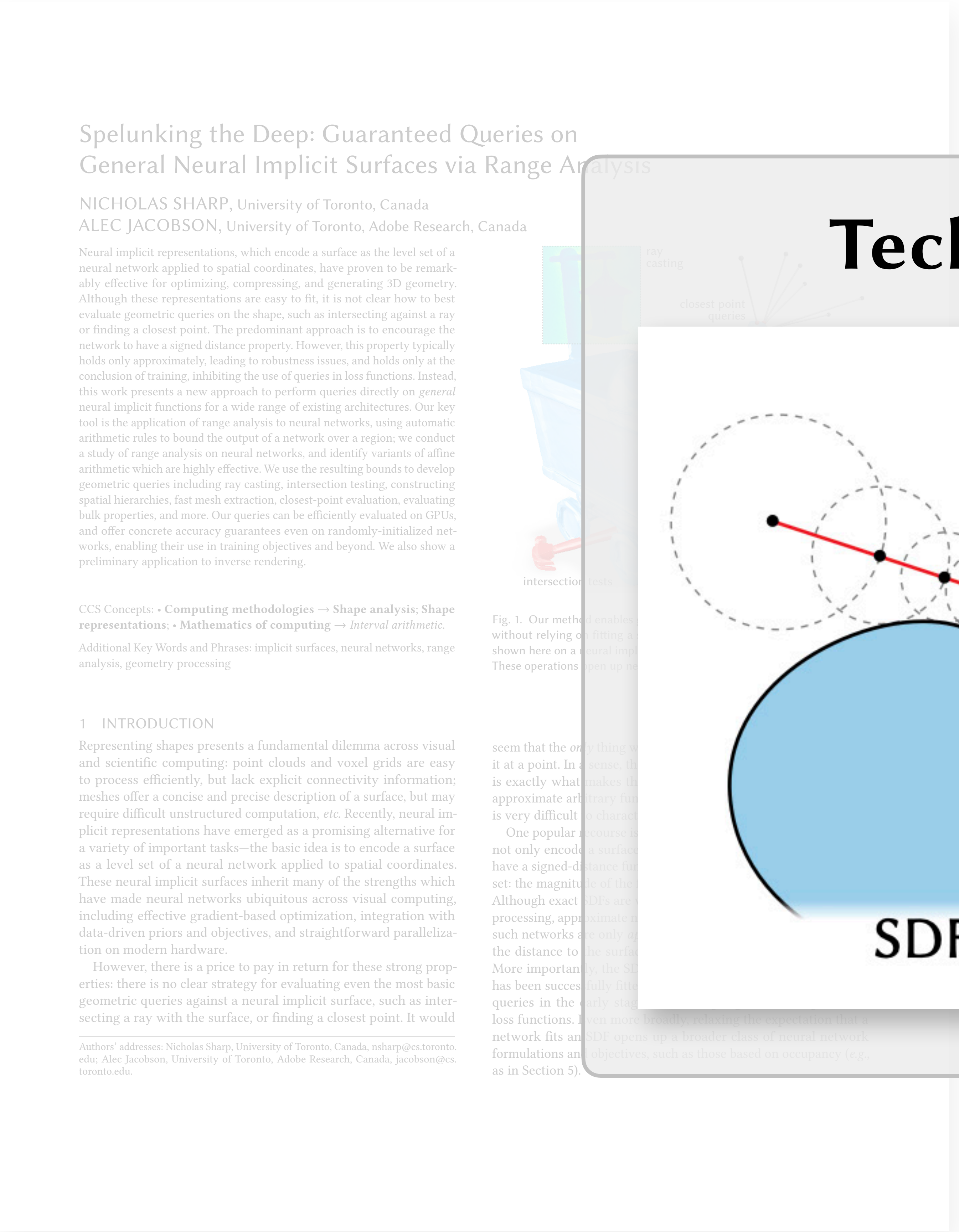
One popular recourse is to attempt to fit implicit functions which not only encode a surface via their zero level set, but furthermore have a signed-distance function (SDF) property away from the level set: the magnitude of the function gives the distance to the surface. Although exact SDFs are well-suited for many queries in geometry processing, approximate neural SDFs leave much to be desired. First, such networks are only *approximately* SDFs, and may overestimate the distance to the surface, causing queries to fail unpredictably. More importantly, the SDF property only applies *after* a network has been successfully fitted; thus we cannot make use of geometric queries in the early stages of training, *e.g.*, to define geometric loss functions. Even more broadly, relaxing the expectation that a network fits an SDF opens up a broader class of neural network formulations and objectives, such as those based on occupancy (*e.g.*, as in Section 5).

Authors' addresses: Nicholas Sharp, University of Toronto, Canada, nsharp@cs.toronto.edu; Alec Jacobson, University of Toronto, Adobe Research, Canada, jacobson@cs.toronto.edu.

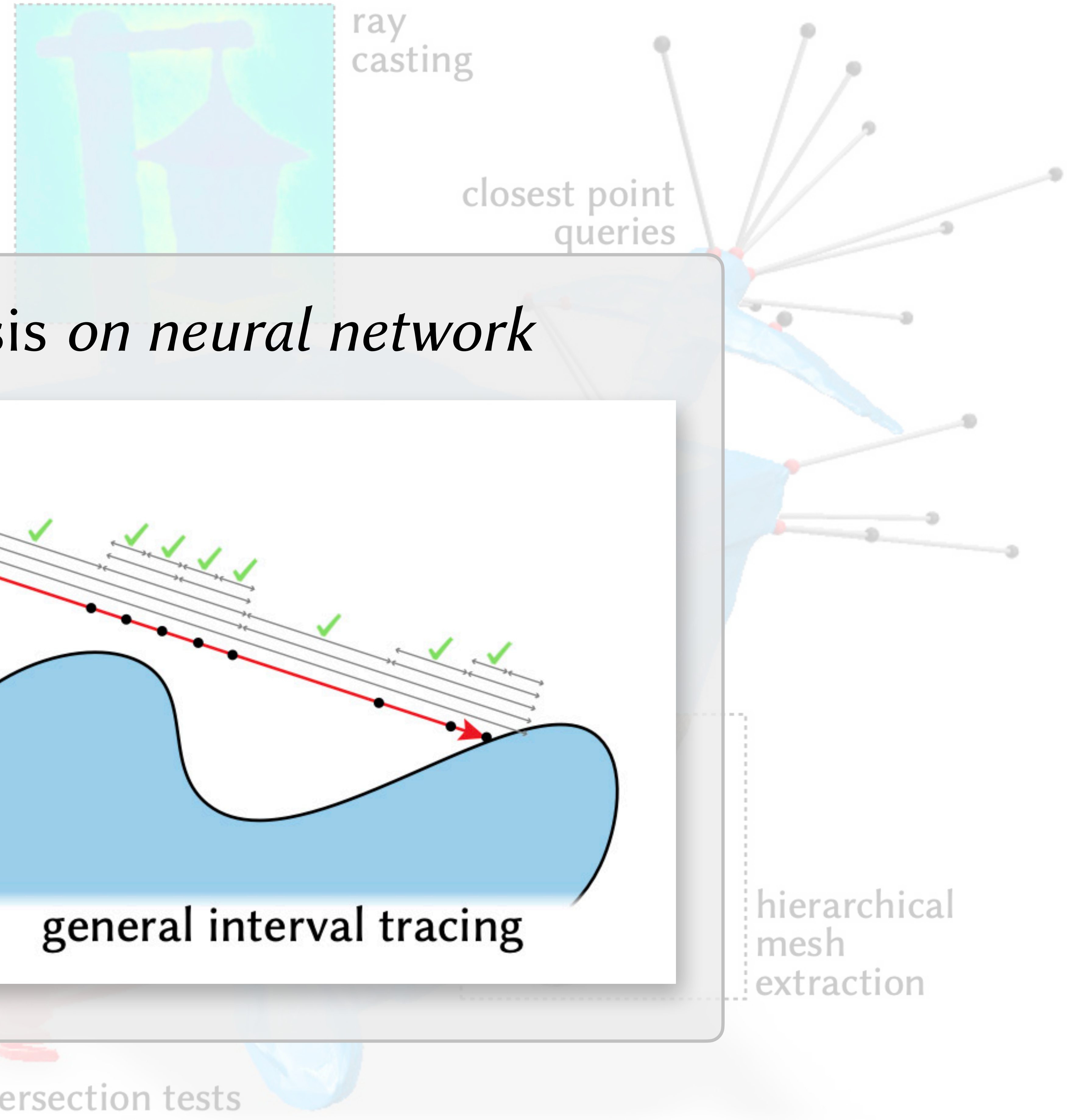


Guaranteed Queries on General Neural Implicit Surfaces via Range Analysis [Sharp et al., 2022]

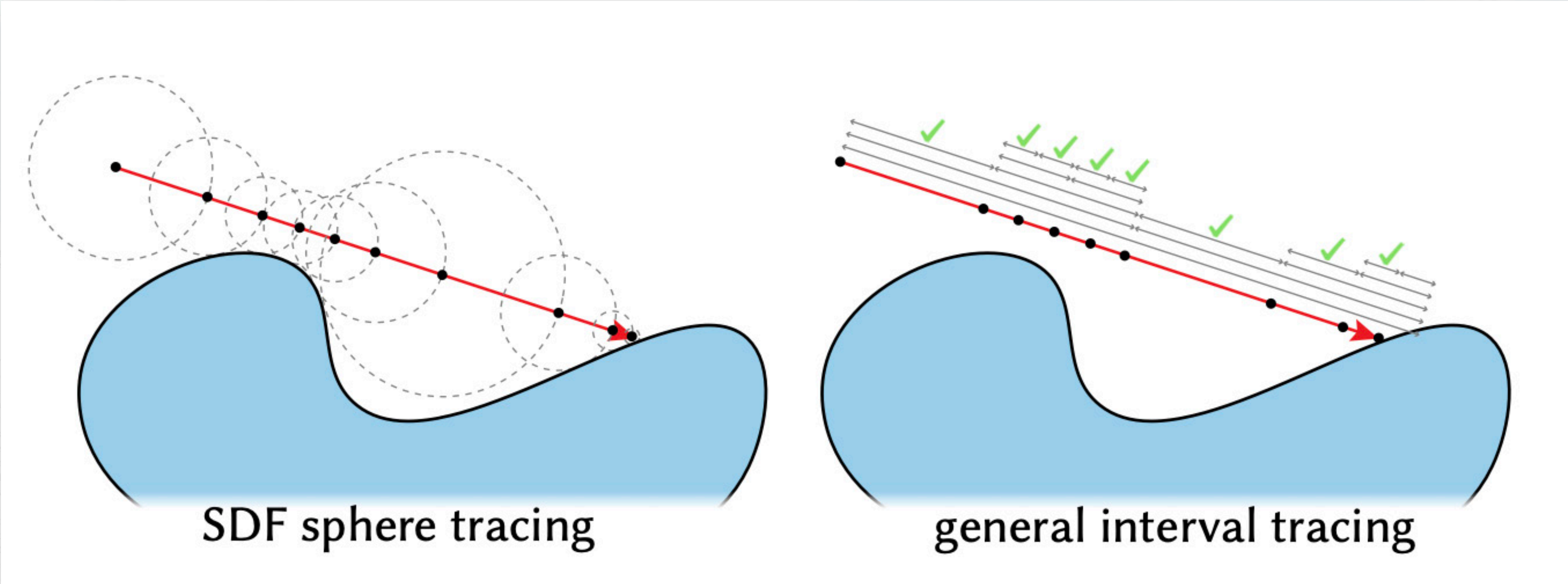
Alternative to SDFs for GP


Spelunking the Deep: Guaranteed Queries on General Neural Implicit Surfaces via Range Analysis

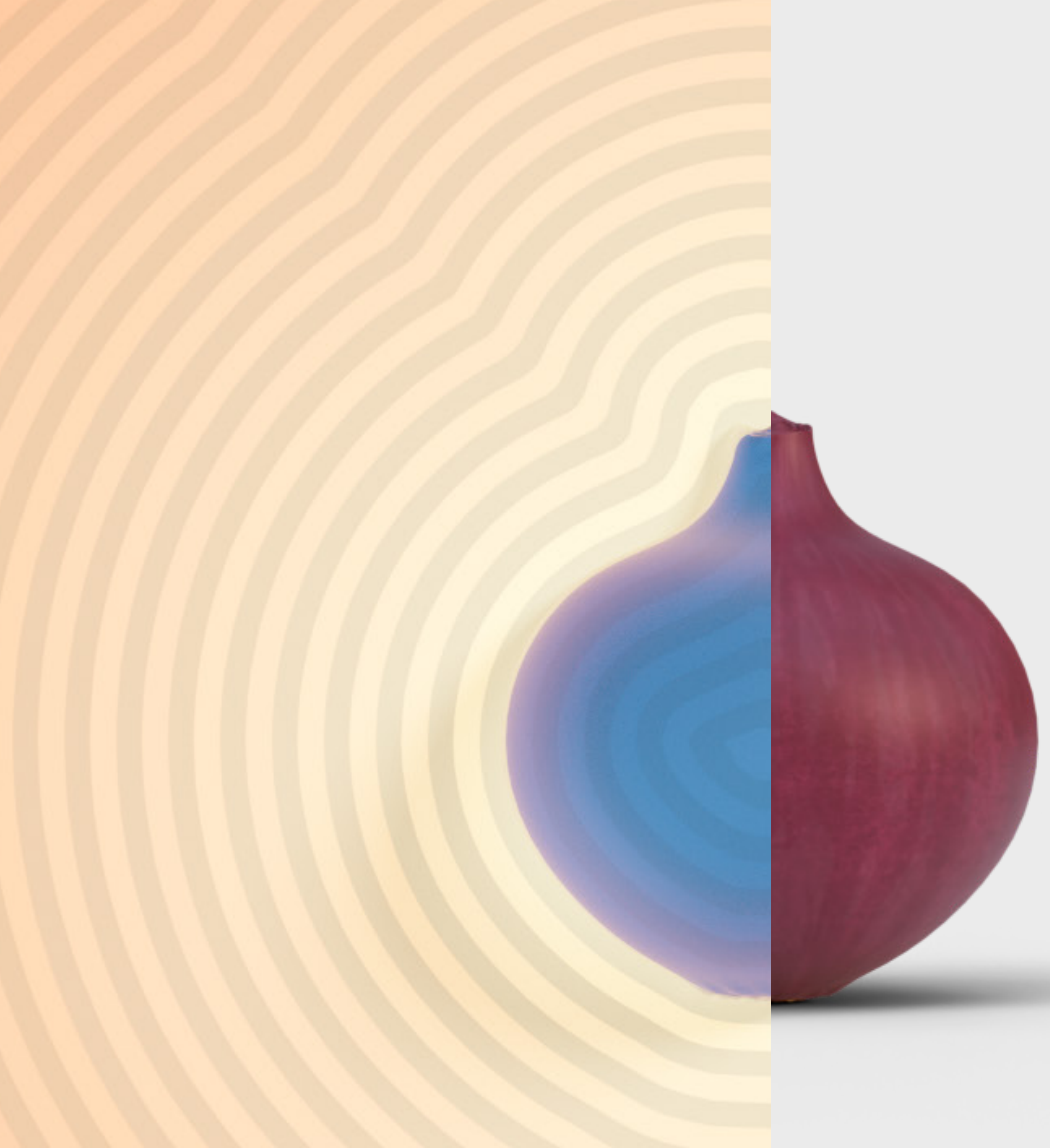
NICHOLAS SHARP, University of Toronto, Canada
ALEC JACOBSON, University of Toronto, Adobe Research, Canada



Technique: interval analysis *on neural network*



Guaranteed Queries on General Neural Implicit Surfaces via Range Analysis [Sharp et al., 2022]



OVERVIEW

IMPLICIT SURFACES

1. Background & History
2. Digital Representations

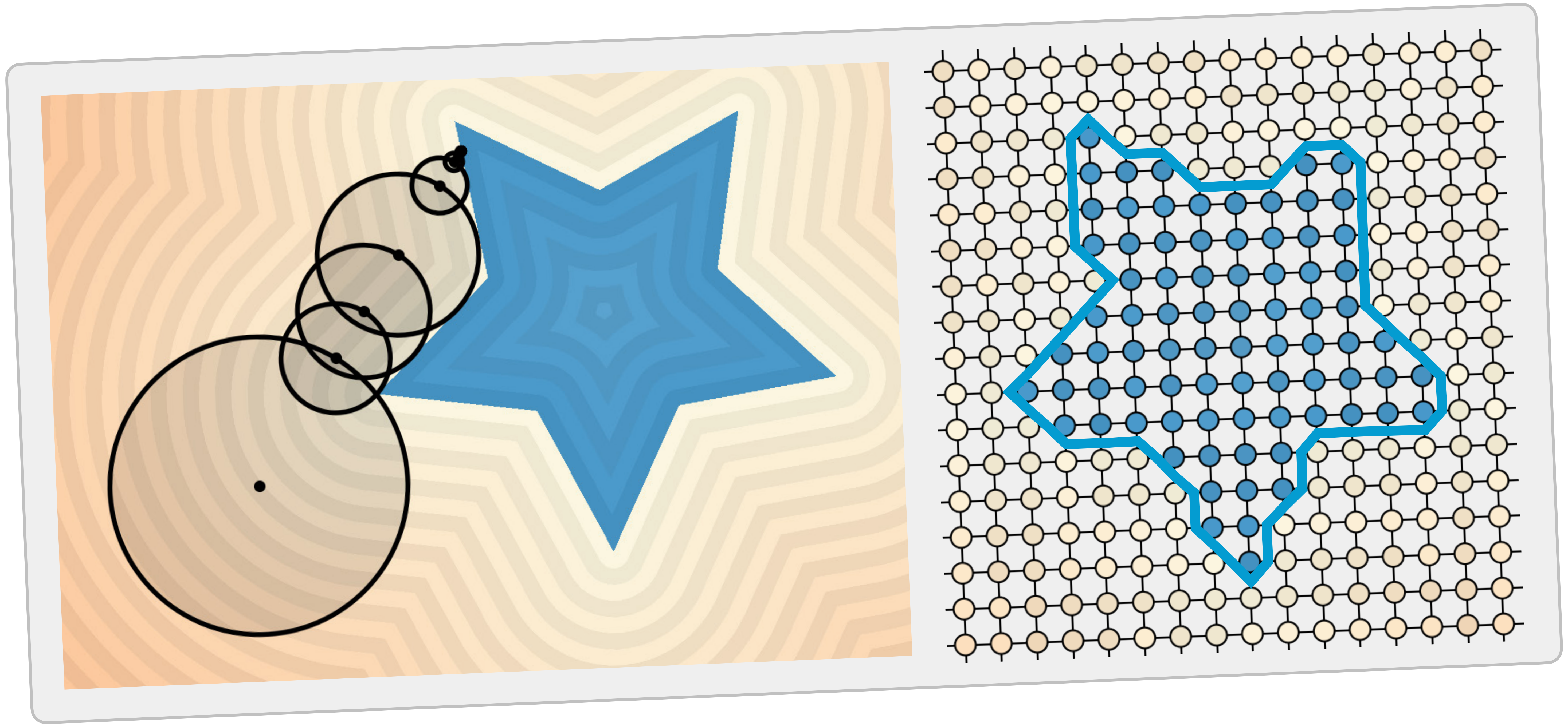
NEURAL IMPLICITS

3. The Basics
4. Why & Why Not Use Them?
5. Working out the Details
6. Geometric Operations

CONCLUSION

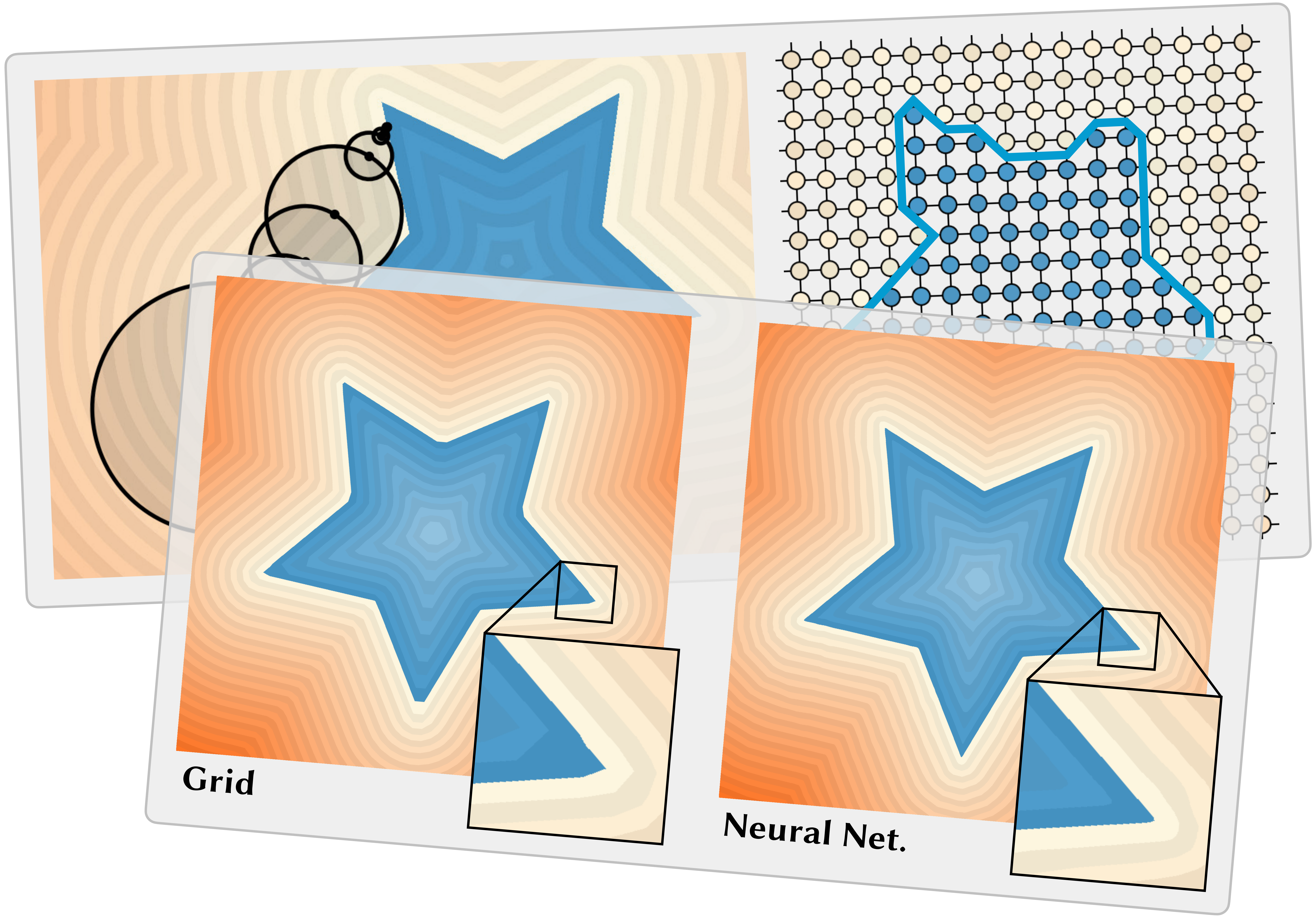
We saw...

- how to work with a classic representation (implicit functions)



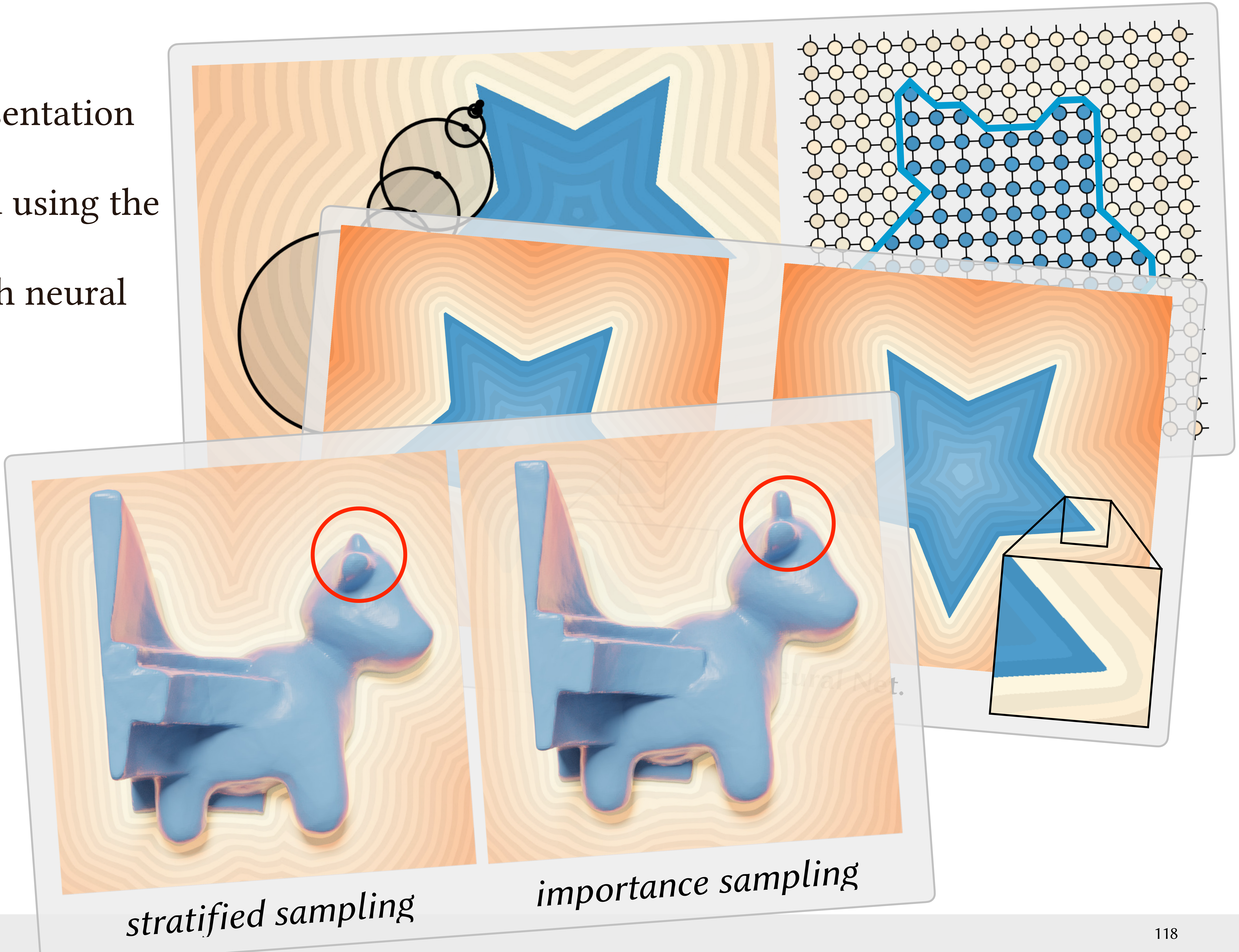
We saw...

- how to work with a classic representation (implicit functions)
- implicit functions can be encoded using the neural network function space



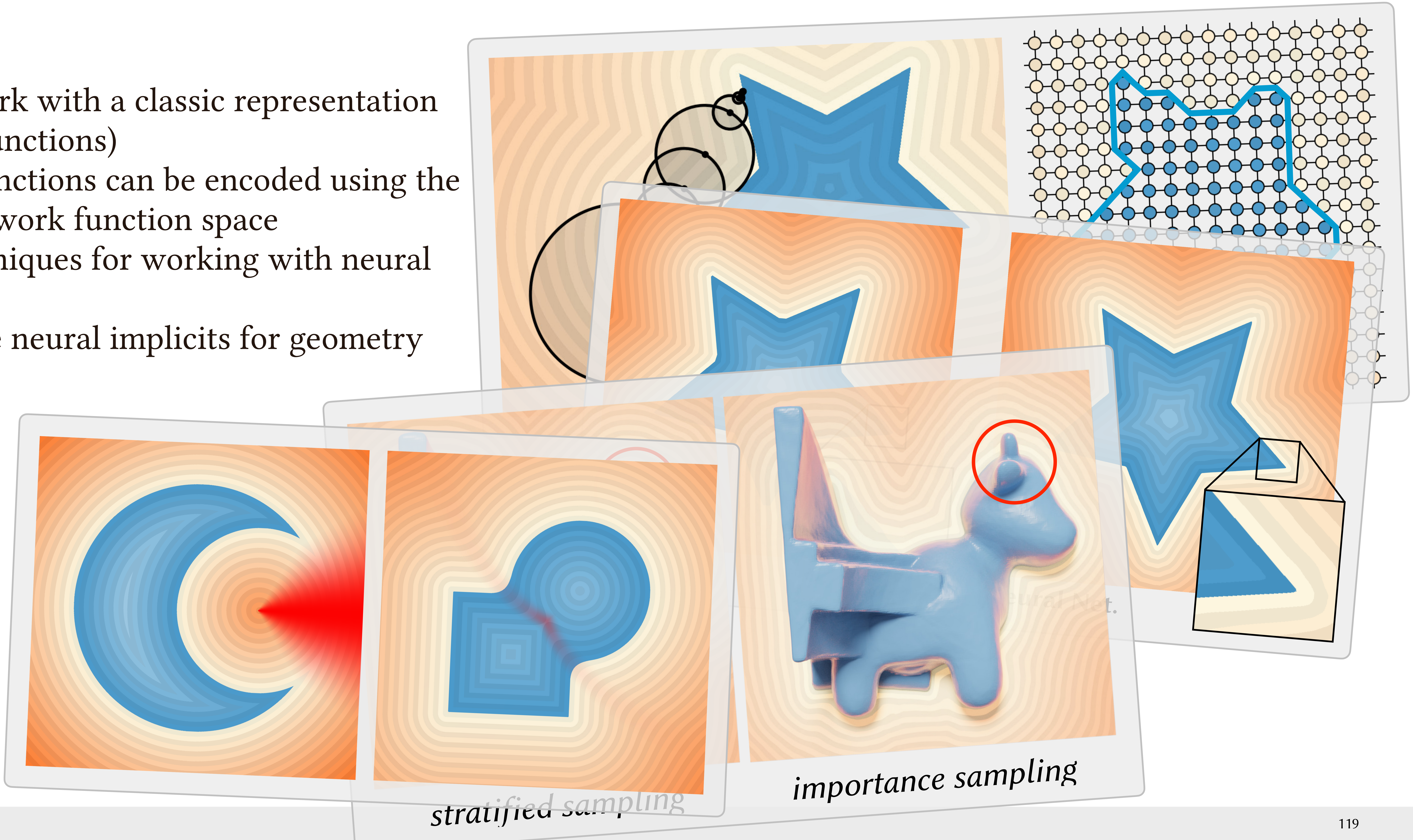
We saw...

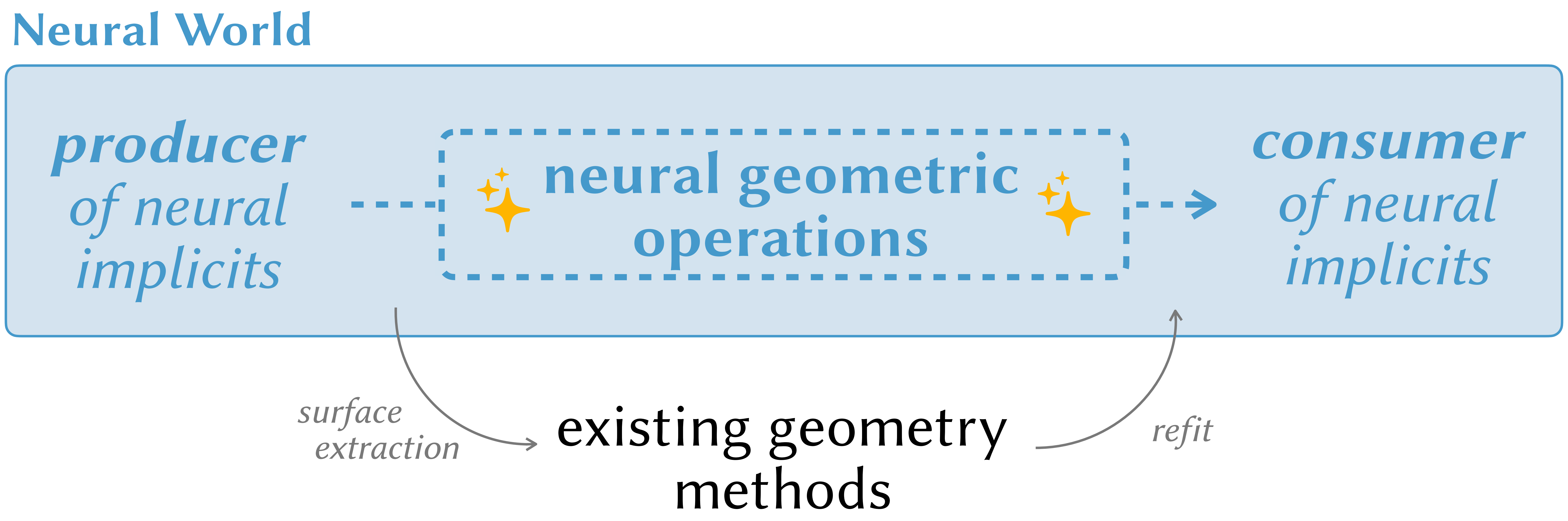
- how to work with a classic representation (implicit functions)
- implicit functions can be encoded using the neural network function space
- some techniques for working with neural implicits



We saw…

- how to work with a classic representation (implicit functions)
- implicit functions can be encoded using the neural network function space
- some techniques for working with neural implicits
- how to use neural implicits for geometry processing




Neural World
producer
of neural
implicits
neural geometric
operations
consumer
of neural
implicits
surface
extraction
existing geometry
methods
refit

Zoë Marschner zoem@cmu.edu

Daniel Ritchie daniel_ritchie@brown.edu