# 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])



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









**OVERVIEW** 

## Synthesis





## Editing

Goals: learn about • what neural implicits are how & when to use them important techniques for working with them

Geometry (e.g., neural implicits) **OVERVIEW** 

## **Synthesis**

## Analysis







## **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 SURFACES** BACKGROUND & HISTORY



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





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







## **c-Level Sets**

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

f(x, y) = c

$$c = \bullet$$







## 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

#### A constructive geometry for computer graphics

A. Ricci

CNEN Centro di Calcolo, Via Mazzini 2, Bologna, Italy

In the present paper a general approach to the definition of complex 3D objects from simpler on ection and union operations are defined which can be approximated to obtain a smooth joining of object volumes with one anoth (Received February 1972)

The representation of the shape of a 3D object in terms of numerical information stored in the computer memory, generally by means of a suitable data structure, is still an impor-

tant problem in computer graphics. In the techniques for object representation till now developed (see references), the information stored in the data structure generally relates to the definition of the object surface, often subdivided in surface patches, thus requiring, unless the object shape is simple, a large amount of data to define surface points, continuity conditions, etc. This gives rise to a certain degree of ness in the modification of the object shape, particularly when extensive changes are required, as frequently happens in the early stages in the design process.

The approach to the representation and manipulation of 3D objects by means of their global definition as solids seems to be more natural and promising. The technique of the definition of complex objects in terms of simpler ones has been attempted (for example, Goldstein and Nagel, 1971) but, while less information needs to be handled, the component objects retain their individuality in the final shape by reason of the lack of a smooth joining of object volumes with one another A certain degree of smoothing has been obtained in a particular technique for the detection of intersections of 3D

objects (Comba, 1968), but this method apparently does not apply to non-convex objects. In the present paper, a general approach to the solution of the

problem, through what can be called a constructive geometry. For any solid, connected or disconnected, in the 3D space a

set of associated functions is defined. Functions relating to different objects can be combined to obtain a new function representing a new solid, allowing the designer to define it by means of a small amount of information. The combination of solids can be realised by applying a suitable sequence of intersection and union operations. The operations in the sequence can be approximated, namely substituted for by operations which give a slightly different result, thus giving rise to a controlled smoothing of matching volumes and surfaces. By suitably regulating the smoothing parameter, in the final solid the component ones may not even be recognisable

Since solids in the geometry illustrated here can be disconnected, a function defining a collection of separate objects can also be used. In addition, solid defining functions lend themselves to a simple solution of the hidden points problem and no serious difficulty arises from the implicit, i.e. non-parametric form of the equation satisfied by the points of the object surface, provided that efficient contour mapping techniques ar used to compute paths on the surface.

#### 1. Preliminary definition

 $E^3$  is considered, it is intended that it can be connected or unit, P belongs to T, and, for (5), P also belongs to  $T^I$ . I nnected, that is to say it can comprise one object or more objects separated from one another. All the results obtained  $I_i$  and, for (6), P belongs to  $I^1$ . In the remaining case, if

Volume 16 Number 2

here apply to this general definition of a solid. For any solid S, the set of its interior points will be denoted by I, the set of its boundary points by B and the set of its exterior points by T, with  $I \cup B \cup T = E^3$ 

$$I \cap B = B \cap T = I \cap T = \phi$$

A continuous function f(P), non-negative for every P in  $E^{3\frac{2}{3}}$ will be called a defining function for a solid S if f(P) < 1 when P belongs to I, f(P) = 1 when P belongs to B and f(P) > 1when P belongs to T. For any given solid, many different defining functions can be found, for example if f(P) is a defining function for the solid S also  $(f(P))^p$ , being p a positive real number, is a defining function for S.

An interesting property of defining functions, as they have been introduced above, is that if f(P) is a defining function for the solid S,  $(f(P))^{-1}$  is a defining function for the solid complement  $S^{C}$  as defined by  $I^{C} = T$ ,  $B^{C} = B$ , and  $T^{C} = I$ . Another useful definition is that of the surface equation for  $\overline{a}$ solid S, namely the equation that is satisfied by the points belonging to B. For any solid S with f(P) as a defining function the surface equation is

As an example, for a sphere having the radius r and its centre at the origin of the reference system, a possible defining

 $f(P) = (x/r)^2 + (y/r)^2 + (z/r)^2$ and f(P) = 1 will define the surface of the sphere.

2. Intersection and union operations

To establish a really useful constructive geometry in compute graphics, we need operations, allowing simple objects to be suitably combined into more complex ones, which will be easy and natural. Most conveniently, the said combination of solids can be realised by applying a sequence of intersection and unio operations. They can be defined in terms of defining function and the following two statements will show how the defining function for the resulting solid can be derived from those fo the component solids.

Statement 1—Let n solids  $S_1, \ldots, S_n$  respectively have defining functions  $f_1(P), \ldots, f_n(P)$ . Then a defining function of their intersection is given by

prove the statement, we firstly note that  

$$T^{I} = T_{1} \cup \ldots \cup T_{n}$$

$$I^{I} = I_{1} \cap \ldots \cap I_{n}$$
$$B^{I} = \text{complement of } I^{I} \cup T^{I}$$

In the present paper, when a solid S in the 3D Euclidean space Then f'(P) > 1 implies that at least one  $f_i(P)$  is greater than  $f^{I}(P) < 1$ , all  $f_{i}(P)$  are lower than unit and P belongs to eve

quation for the solid  $S^{I}$  is  $f_r(P) = 1$ (7) on of the three infinite slabs with  $(z/r)^{2}$  $(r/r)^2$ ,  $(z/r)^2$ ) = 1 (9) cube centred at the origin of the

ating functions chosen to be substituted for max and min functions here are respectively  $I_p(f_1, \ldots, f_n) = (f_1^p + \ldots + f_n^p)^{1/p}$ 

where p is a positive real number. To prove that  $I_p$  and  $U_p$  can be used as *p*-approximations of respectively max and min functions, the following statements

Statement 3—For any point  $P \in E^3$ ,

 $(f_1, f_2, \ldots, f_n)$  and (17) is equivalent to (18) and thus proved

To prove the statement, it is now sufficient to observe that, letting  $\bar{f}_i$  be the solid complement defining functions  $\bar{f}_i = f_i^{-1}$ ,  $U_p(f_1, f_2, \dots, f_n) = [I_p(\bar{f}_1, \bar{f}_2, \dots, \bar{f}_n)]^{-1}$  $U_{\infty} = \min(f_1, f_2, \dots, f_n) = [\max(\bar{f}_1, \bar{f}_2, \dots, \bar{f}_n)]^{-1}$ Then statement 4 is proved by reason of statement 3. As an example of application of the above statements, a sphere centred at the origin of the reference system and having

f component solids into a final st be approximated by means of

"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)



he surface equations  $\max(f_1, f_2) = 1$  and  $\min(f_1, f_2) = 1$ greater than unit and P cannot P) is not lower than unit and Pelongs to the complement of the

(y/r)

 $\ldots \cup I_{\ldots}$ 

 $\cap \ldots \cap T_n$ 

a)/3a)

 $(-a)/(3a)^2$ 

(6)  $((x + a)/3a)^2) = 1$ 

(4)

(5)

157

functions be everywhere positive in  $E^3$ . not to alter significantly the solid defined by the function For example, adding  $10^{-5}$  to the defining function (3) will give rise to a modification of the sphere radius of about 0.05 per A large variety of sequences of approximating functions can be used, but only one way of approximating max and min functions will be illustrated in the present paper. The approxim-.,  $S_n$  respectively have defining ten a defining function of their  $U_p(f_1, \ldots, f_n) = (f_1^{-p} + \ldots + f_n^{-p})^{-1/p}$  $.., f_n(P)$ (11) must be shown to be true (12) plement of  $I^U \cup T^U$  $f_i(P)$  is lower than unit and P ), to  $I^{U}$ . If  $f^{U}(P) > 1$ , all  $f_{i}(P)$ longs to every  $T_i$  and, for (12),  ${}^{U}(P) = 1$ , no  $f_{i}(P)$  can be lowe  $f = (f_1, f_2, \ldots, f_n)$  is given by g to  $I^U$ , at least one  $f_i(P)$  is not t belong to  $T^U$ , then P belongs to of  $I^U$  and  $T^U$ , namely to  $B^U$ .  $\lim_{p \to \infty} \|f\|_p = \|f\|_{\infty}$ where  $||f||_p$  is the space p norm (Davis, 1963). Since  $f_i \ge 0$ uation for the solid  $S^U$  is  $f_n(P) = 1$ (13) he two infinite slabs with defining

Statement 4—For any point  $P \in E^3$ ,

(15) an infinite slab centred at the radius r can be obtained from the defining functions (8) by means of their approximated intersection I., Generally, an

and having a half-thickness of

f(P) = 1

#### function is







# Implicits in early graphics history



## Jim Blinn's "Blobby molecules" (1982)





# Implicits in early graphics history





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





**IMPLICIT SURFACES: BIG IDEAS** 1. Constructive Solid Geometry 2. Taxonomy of implicits 3. Rendering 4. Surface Extraction







 $\bigcirc$ 

## **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







## intersection $A \cap B$

## **IMPLICIT SURFACES: BIG IDEAS**

## **1. CSG**

- 2. Taxonomy of implicits
- 3. Rendering
- 4. Surface Extraction







## intersection $A \cap B$

## **IMPLICIT SURFACES: BIG IDEAS**

#### CSG 1.

- 2. Taxonomy of implicits
- 3. Rendering
- 4. Surface Extraction









intersection  $A \cap B$ 

## **IMPLICIT SURFACES: BIG IDEAS**

## CSG

- Taxonomy of implicits 2.
- Rendering 3.
- Surface Extraction 4.

## Modeling with implicits



*Complex shapes from simple primitives!* 

G For more amazing implicit function art, see <u>Inigo Quilez's blog</u>

*Video: Inigo Quilez* 18







# *implicit functions*

## **IMPLICIT SURFACES: BIG IDEAS**

CSG 1.

## 2. Taxonomy of implicits

- 3. Rendering
- 4. Surface Extraction







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

# **Indicator Function** $f(\mathbf{x}) = \begin{cases} -1 & \mathbf{x} \in \text{shape} \\ 1 & \mathbf{x} \notin \text{shape} \end{cases}$









# *implicit functions*

## **IMPLICIT SURFACES: BIG IDEAS**

CSG 1.

## 2. Taxonomy of implicits

- 3. Rendering
- 4. Surface Extraction







X

 $f(\mathbf{x})$ 

CSG

## 2. Taxonomy of implicits

- 3. Rendering
- 4. Surface Extraction

## **Signed Distance Function**

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

 $f(\mathbf{y})$ 





X

1. CSG

## 2. Taxonomy of implicits

- 3. Rendering
- 4. Surface Extraction

#### **IMPLICIT SURFACES →** BACKGROUND & HISTORY

## **Signed Distance Function**

 | f(x) | encodes distance to closest point on surface



 $\mathbf{X}_1$ 

f(x)



X

CSG

## 2. Taxonomy of implicits

- 3. Rendering
- 4. Surface Extraction

#### **IMPLICIT SURFACES** → BACKGROUND & HISTORY

## **Signed Distance Function**

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



 $\boldsymbol{x}_{2}$ 

 $f(\mathbf{x})$ 

 $\boldsymbol{x}_1$ 



X

 $\boldsymbol{x}_{2}$ 

 $f(\mathbf{x})$ 

CSG 1.

## 2. Taxonomy of implicits

- 3. Rendering
- 4. Surface Extraction

#### **IMPLICIT SURFACES** → BACKGROUND & HISTORY

## **Signed Distance Function**

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







X

1. CSG

## 2. Taxonomy of implicits

- 3. Rendering
- 4. Surface Extraction



- | f(x) | encodes distance to closest point on surface
- makes many tasks way easier ③
   e.g., offset surfaces, distance checks for simulation, etc.





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

## **Signed Distance Function**

- | f(x) | encodes distance to closest point on surface
- makes many tasks way easier ③
  - e.g., offset surfaces, distance checks for simulation, etc.
  - e.g., geometric quantities



Mean curvature ( $\Delta f$ )





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

#### **IMPLICIT SURFACES** → BACKGROUND & HISTORY

## **Signed Distance Function**

- $|f(\mathbf{x})|$  encodes distance to closest point on surface
- makes many tasks way easier ③
- distance property is hard to maintain ອ
  - e.g., not preserved by most editing operations



## **Conservative SDFs**

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

# *implicit functions*

## **IMPLICIT SURFACES: BIG IDEAS**

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









## **Conservative SDFs**

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

# *implicit functions*

## **IMPLICIT SURFACES: BIG IDEAS**

CSG

## 2. Taxonomy of implicits

- Rendering 3.
- 4. Surface Extraction









## **Sphere Tracing**

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

## **IMPLICIT SURFACES: BIG IDEAS**

- 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**

- 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**

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



# Marching cubes

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

SES

## **IMPLICIT SURFACES: BIG IDEAS**

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



## Marching cubes

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

# Tangency-Aware Surface Reconstruction of SDFs

Reach For the Spheres:

#### Neural Dual Contouring

ZHIQIN CHEN, Simon Fraser University, Canade ANDREA TAGLIASACCHI, Google Research, Simon Fr THOMAS FUNKHOUSER, Google Research, USA



#### **IMPLICIT SURFACE**

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

#### **IMPLICIT SURFACES** → BACKGROUND & HISTORY

• 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

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

#### **IMPLICIT SURFACES** BACKGROUND & HISTORY

#### **EXPLICIT**

answer questions about surface

• e.g., what is the total surface

area







### **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, ...







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

#### **IMPLICIT SURFACES BACKGROUND & HISTORY**

in *implicit* representations, topology changes are continuous (very useful for inverse tasks!) bout surface otal surface vant to Iteration 0 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

G For analytic definitions of SDFs, see "Distance Functions" page on Inigo Quillez's blog





# **Grid-based representations**







# **Grid-based representations: just a function!**

<i>w</i> <sub>0,0</sub>	<i>w</i> <sub>1,0</sub>	W <sub>2,0</sub>	W <sub>3,0</sub>
<i>w</i> <sub>0,1</sub>	<i>w</i> <sub>1,1</sub>	W <sub>2,1</sub>	W <sub>3,1</sub>
<i>w</i> <sub>0,2</sub>	<i>W</i> <sub>1,2</sub>	W <sub>2,2</sub>	W <sub>3,2</sub>
<i>w</i> <sub>0,3</sub>	<i>W</i> <sub>1,3</sub>	W <sub>2,3</sub>	W <sub>3,3</sub>

# $f(x, y) = (\lfloor x \rfloor = 0 \& \lfloor y \rfloor = 0) w_{0,0} + (\lfloor x \rfloor = 1 \& \lfloor y \rfloor = 0) w_{1,0} + \dots$





# Generalization: any function space

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

#### **IMPLICIT SURFACES** DIGITAL REPRESNTATIONS

Degree 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

# $f_{\theta} : \mathbb{R}^n \to \mathbb{R}$ $f_{\theta}(x) = A_{\theta}^{N} \varphi(A_{\theta}^{1} \varphi(A_{\theta}^{0} x + b_{\theta}^{0}) + b_{\theta}^{1} + \dots) + b_{\theta}^{N}$

G For an intro to Neural Networks, see the course "<u>6.036 Intro to ML</u>" on OCW





 $f_{\theta} : \mathbb{R}$   $f_{\theta}(x) = A_{\theta}^{N} \varphi(A_{\theta}^{1} \varphi(A_{\theta}^{0} x + \nu_{\mu}))$ 

G For an intro to Neural Networks, see the course "<u>6.036 Intro to ML</u>" on OCW

$$\mathbb{R}^{n} \to \mathbb{R}$$

$$P(x + b_{\theta}^{0}) + b_{\theta}^{1} + \dots) + b_{\theta}^{N}$$
hear layer







Given For an intro to Neural Networks, see the course "<u>6.036 Intro to ML</u>" on OCW





 $f_{\theta} : \mathbb{R}$   $f_{\theta}(x) = A_{\theta}^{N} \varphi(A_{\theta}^{1} \varphi(A_{\theta}^{0} x + b_{\theta}^{0} x + b_{\theta}^$ 

$$\mathbb{R}^{n} \to \mathbb{R}$$
*hidden layers*

$$P(x + b_{\theta}^{0}) + b_{\theta}^{1} + \dots) + b_{\theta}^{N}$$
*put layer*





 $f_{\theta} : \mathbb{R}^n \to \mathbb{R}$  $f_{\theta}(x) = A^N_{\theta} \varphi(A^1_{\theta} \varphi(A^0_{\theta} x + b^0_{\theta}) + b^1_{\theta} + \dots) + b^N_{\theta}$ 

# Key idea: neural networks provide a space of functions with parameters $\theta$





# Early work on neural implicits

This CVPR paper is the Open Access version, provided by the Computer Vision Foundation Except for this watermark, it is identical to the accepted ve the final published version of the proceedings is available on IEEE Xplore.

#### Occupancy Networks: Learning 3D Reconstruction in Function Space

Lars Mescheder<sup>1</sup> Michael Oechsle<sup>1,2</sup> Michael Niemeyer<sup>1</sup> Sebastian Nowozin<sup>3†</sup> Andreas Geiger<sup>1</sup> <sup>1</sup>Autonomous Vision Group, MPI for Intelligent Systems and University of Tübingen <sup>2</sup>ETAS GmbH, Stuttgart

<sup>3</sup>Google AI Berlin

{firstname.lastname}@tue.mpg.de nowozin@gmail.com

#### Abstract

With the advent of deep neural networks, learning-based approaches for 3D reconstruction have gained popularity. However, unlike for images, in 3D there is no canonical representation which is both computationally and memory efficient vet allows for representing high-resolution geometry of arbitrary topology. Many of the state-of-the-art learningbased 3D reconstruction approaches can hence only repr sent very coarse 3D geometry or are limited to a restricted domain. In this paper, we propose Occupancy Networks, a new representation for learning-based 3D reconstruction methods. Occupancy networks implicitly represent the 3D surface as the continuous decision boundary of a deep neural network classifier. In contrast to existing approaches, our representation encodes a description of the 3D output at infinite resolution without excessive memory footprint We validate that our representation can efficiently encode 3D structure and can be inferred from various kinds of in put. Our experiments demonstrate competitive results, both qualitatively and quantitatively, for the challenging tasks of 3D reconstruction from single images, noisy point clouds and coarse discrete voxel grids. We believe that occupancy networks will become a useful tool in a wide variety of learning-based 3D tasks.

#### 1. Introduction

Recently, learning-based approaches for 3D reconstruct tion have gained popularity [4,9,23,58,75,77]. In contrast to traditional multi-view stereo algorithms, learned models are able to encode rich prior information about the space of 3D shapes which helps to resolve ambiguities in the input. While generative models have recently achieved remark able successes in generating realistic high resolution images [36, 47, 72], this success has not yet been replicated in the 3D domain. In contrast to the 2D domain, the com-

<sup>†</sup>Part of this work was done while at MSR Cambridge

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

Figure 1: Overview: Existing 3D representations discretize the output space differently: (a) spatially in voxel represent tations (h) in terms of predicted points and (c) in terms of vertices for mesh representations. In contrast, (d) we pronsider the continuous decision boundary of a cla sifier  $f_{\theta}$  (e.g., a deep neural network) as a 3D surface which allows to extract 3D meshes at any resolution.

munity has not yet agreed on a 3D output representation that is both memory efficient and can be efficiently inferred from data. Existing representations can be broadly cate gorized into three categories: voxel-based representations [4, 19, 43, 58, 64, 69, 75], point-based representations [1, 17] and mesh representations [34, 57, 70], see Fig. 1

Voxel representations are a straightforward generalization of pixels to the 3D case. Unfortunately, however, the memory footprint of voxel representations grows cubically with resolution, hence limiting naïve implementations to 32<sup>3</sup> or 64<sup>3</sup> voxels. While it is possible to reduce the memory footprint by using data adaptive representations such as oc trees [61, 67], this approach leads to complex implementations and existing data-adaptive algorithms are still limited to relatively small  $256^3$  voxel grids. Point clouds  $\left[1,17\right]$  and meshes [34, 57, 70] have been introduced as alternative representations for deep learning, using appropriate loss functions. However, point clouds lack the connectivity structure of the underlying mesh and hence require additional postprocessing steps to extract 3D geometry from the model

**DeepSDF: Learning Continuous Signed Distance Functions** for Shape Representation

Jeong Joon Park<sup>1,3†</sup> Peter Florence <sup>2,3†</sup> Julian Straub<sup>3</sup> Richard Newcombe<sup>3</sup> Steven Lovegrove<sup>3</sup> <sup>2</sup>Massachusetts Institute of Technology <sup>3</sup>Facebook Reality Labs

Figure 1: DeepSDF represents signed distance functions (SDFs) of shanes via latent code-conditioned feed-forward decoder natural mages are raycast renderings of DeepSDF interpolating between two shapes in the learned shape latent space. Best viewed digitally

#### Abstract

Computer graphics, 3D computer vision and robotics communities have produced multiple approaches to representing 3D geometry for rendering and reconstruction. These provide trade-offs across fidelity, efficiency and compression capabilities. In this work, we introduce DeepSDF, a learned continuous Signed Distance Function (SDF) representation of a class of shapes that enables high qual ity shape representation, interpolation and completion from partial and noisy 3D input data. DeepSDF, like its classical counterpart, represents a shape's surface by a continuous volumetric field: the magnitude of a point in the field represents the distance to the surface boundary and the sign indicates whether the region is inside (-) or outside (+) of the shape, hence our representation implicitly encodes a shape's boundary as the zero-level-set of the learned function while explicitly representing the classification of space as being part of the shapes interior or not. While classical SDF's both in analytical or discretized voxel form typically present the surface of a single shape, DeepSDF can repre sent an entire class of shapes. Furthermore, we show stateof-the-art performance for learned 3D shape representation and completion while reducing the model size by an order of magnitude compared with previous work.

+ Work performed during internship at Facebook Reality Lab

#### 1. Introduction

Deep convolutional networks which are a mainstay of image-based approaches grow quickly in space and time complexity when directly generalized to the 3rd spatial dimension, and more classical and compact surface representations such as triangle or quad meshes pose problems in training since we may need to deal with an unknown number of vertices and arbitrary topology. These chalenges have limited the quality, flexibility and fidelity of deep learning approaches when attempting to either input 3D data for processing or produce 3D inferences for object nentation and reconstruction

In this work, we present a novel representation and ap-proach for generative 3D modeling that is efficient, expressive, and fully continuous. Our approach uses the concept of a SDF, but unlike common surface reconstruction techniques which discretize this SDF into a regular grid for evaluation and measurement denoising, we instead learn a generative model to produce such a continuous field.

The proposed continuous representation may be intutively understood as a learned shape-conditioned classifier for which the decision boundary is the surface of the shape itself, as shown in Fig. 2. Our approach shares the generative aspect of other works seeking to map a latent space to a distribution of complex shapes in 3D [54], but critically differs in the central representation. While the notion of ar

**Occupancy Networks** Mescheder et al., 2019

**DeepSDF** Park et al., 2019

D For more info on neural implicits, see Vincent Sitzmann's course "<u>ML for Inverse Graphics</u>"



Learning Implicit Fields for Generative Shape Modeling

Zhiqin Chen Simon Fraser University zhiqinc@sfu.ca

#### Abstract

We advocate the use of implicit fields for learning gen erative models of shapes and introduce an implicit field de-coder, called IM-NET, for shape generation, aimed at improving the visual quality of the generated shapes. An implicit field assigns a value to each point in 3D space, so that a shape can be extracted as an iso-surface. IM-NET is trained to perform this assignment by means of a binary classifier. Specifically, it takes a point coordinate, along with a feature vector encoding a shape, and outputs a value which indicates whether the point is outside the shape or not. By replacing conventional decoders by our implicit decoder for representation learning (via IM-AE) and shape generation (via IM-GAN), we demonstrate superior result. for tasks such as generative shape modeling, interpolation and single-view 3D reconstruction, particularly in terms of visual quality. Code and supplementary material are avail able at https://s a142857/implicit-decode

#### 1. Introduction

 $\frown$ 

Unlike images and video, 3D shapes are not confined to one standard representation. Up to date, deep neural networks for 3D shape analysis and synthesis have been develped for voxel grids [19, 48], multi-view images [42], poin clouds [1, 35], and integrated surface patches [17]. Specific to generative modeling of 3D shapes, despite the many proresses made, the shapes produced by state-of-the-art methods still fall far short in terms of visual quality. This is reflected by a combination of issues including low-resolution outputs, overly smoothed or discontinuous surfaces, as well

as a variety of topological noise and irregularities. In this paper, we explore the use of *implicit fields* for learning deep models of shapes and introduce an *implicit* field decoder for shape generation, aimed at improving the visual quality of the generated models, as shown in Fig-ure 1. An implicit field assigns a value to each point (x, y, z). A shape is represented by all points assigned to a specific value and is typically rendered via iso-surface extraction such as Marching Cubes. Our implicit field de-



Hao Zhang

Simon Fraser University

haoz@sfu.ca

Figure 1: 3D shapes generated by IM-GAN, our implicit field generative adversarial network, which was trained on 643 or 1283 voxelized shapes. The output shapes are sampled at 512<sup>3</sup> resolution and rendered after Marching Cubes

coder, or simply implicit decoder, is trained to perform this assignment task, by means of a binary classifier, and it has a very simple architecture; see Figure 2. Specifically, it takes a point coordinate (x, y, z), along with a feature vector ending a shape, and outputs a value which indicates whether the point is outside the shape or not. In a typical application setup, our decoder, which is coined IM-NET, would follow an encoder which outputs the shape feature vectors and then return an implicit field to define an output shape.

Several novel features of IM-NET impact the visual quality of the generated shapes. First, the decoder output can be sampled at any resolution and is not limited by the resolution of the training shapes; see Figure 1. More importantly, we concatenate point coordinates with shape feaures, feeding both as input to our implicit decoder, which learns the inside/outside status of any point relative to a shape. In contrast, a classical convolution/deconvolution based neural network (CNN) operating on voxelized shapes is typically trained to predict voxels relative to the extent of the bounding volume of a shape. Such a network learns

2 C

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

Vincent Sitzmann Michael Zollhöfer Gordon Wetzstein {sitzmann, zollhoefer}@cs.stanford.edu, gordon.wetzstein@stanford.edu Stanford Univ

vsitzmann.github.io/srns/

#### Abstract

Unsupervised learning with generative models has the potential of discovering rich representations of 3D scenes. While geometric deep learning has explored 3Dstructure-aware representations of scene geometry, these models typically require explicit 3D supervision. Emerging neural scene representations can be trained only with posed 2D images, but existing methods ignore the three-dimensional structure of scenes. We propose Scene Representation Networks (SRNs), a continuous, 3D-structure-aware scene representation that encodes both geometry and appearance. SRNs represent scenes as continuous functions that map world coordinates to a feature representation of local scene properties. By formulating the image formation as a differentiable ray-marching algorithm, SRNs can be trained end-toend from only 2D images and their camera poses, without access to depth or shape. This formulation naturally generalizes across scenes, learning powerful geometry and appearance priors in the process. We demonstrate the potential of SRNs by evaluating them for novel view synthesis, few-shot reconstruction, joint shape and appearance interpolation, and unsupervised discovery of a non-rigid face model.<sup>1</sup>

#### 1 Introduction

A major driver behind recent work on generative models has been the promise of unsupervise discovery of powerful neural scene representations, enabling downstream tasks ranging from robotic manipulation and few-shot 3D reconstruction to navigation. A key aspect of solving these tasks is understanding the three-dimensional structure of an environment. However, prior work on neural scene representations either does not or only weakly enforces 3D structure [1–4]. Multi-view geometry and projection operations are performed by a black-box neural renderer, which is expected to learn these operations from data. As a result, such approaches fail to discover 3D structure under limited training data (see Sec. 4), lack guarantees on multi-view consistency of the rendered image and learned representations are generally not interpretable. Furthermore, these approaches lack an intuitive interface to multi-view and projective geometry important in computer graphics, and cannot easily generalize to camera intrinsic matrices and transformations that were completely unseen at training time

In geometric deep learning, many classic 3D scene representations, such as voxel grids [5–10], point clouds [11–14], or meshes [15] have been integrated with end-to-end deep learning models and have led to significant progress in 3D scene understanding. However, these scene representations are discrete, limiting achievable spatial resolution, only sparsely sampling the underlying smooth surfaces of a scene, and often require explicit 3D supervision.

<sup>1</sup>Please see supplemental video for additional result

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

# **IM-NET** Chen et al., 2019

### Scene Rep. Nets. Sitzmann et al., 2019









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







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







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

 $\min_{\theta} \sum_{x \in \mathcal{X}} (f_{\theta}(x) - v)^2$ "loss function"

find best fitting f in func. space







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

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

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
```









### **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

#### **NEURAL IMPLICITS** → WHY & WHY NOT





## **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?

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









## **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?

- **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

## WHY?

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

#### **NEURAL IMPLICITS** → WHY & WHY NOT





**2500 parameters** 

Neural Network: 4 layers, 34 hidden size 2414 parameters





## Neural implicits have the ability to adaptively use parameters

• but this depends on the chosen loss function!

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

Grid: 50x50

WHY?

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

**2500 parameters** 

Neural Network: 4 layers, 34 hidden size **2414 parameters** 







## WHY?

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

#### **NEURAL IMPLICITS** → WHY & WHY NOT





## WHY?

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

#### **NEURAL IMPLICITS** → WHY & WHY NOT



# Single shot reconstruction [Sitzmann et al., 2019]







# WHY?

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

## WHY?

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

#### **NEURAL IMPLICITS** → WHY & WHY NOT



**Topology Optimization** [Zehnder et al., 2021]



## In general:

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

# WHY?

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



**Topology Optimization** [Zehnder et al., 2021]



# Properties hold only through soft losses

## WHY NOT?

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

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!



### WHY NOT?

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

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



 $\theta$ 





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

# WHY NOT?

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

#### **NEURAL IMPLICITS** → WHY & WHY NOT








## WHY NOT?

- Imprecision 1.
- 2. Editing (etc.) is difficult
- 3. <sup>‡</sup> The wrong representation <sup>‡</sup>



**Pick the right representation for your problem!** 





73



### **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 are hard for standard techniques to capture!

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

### **NEURAL IMPLICITS** WORKING OUT THE DETAILS



## 2020 Jun 18.CV] CS arXiv:2006.10739v1

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

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



### **SIREN** [Sitzmann et al., 2020]





### **Fourier Features** [Tancik et al., 2020]

### **NEURAL IMPLICITS** WORKING OUT THE DETAILS

# augment input points with "Fourier features"

tivation Functions

Julien N. P. Martel\*

Abstract

## **SIREN** [Sitzmann et al., 2020]





**Fourier Features** [Tancik et al., 2020]

### **NEURAL IMPLICITS** WORKING OUT THE DETAILS

**SIREN** [Sitzmann et al., 2020]





## Hybrid discrete/continuous fields

### **Convolutional Occupanc**

Songyou Peng<sup>1,2</sup> Michael Niemeyer<sup>2,3</sup> Marc Pollefeys<sup>1,5</sup> Andreas

<sup>1</sup>ETH Zurich <sup>2</sup>Max Planck Institute for Intel <sup>3</sup>University of Tübingen <sup>4</sup>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 metl connected network architecture which does not a 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.

### Introduction

3D reconstruction is a fundamental problem in co applications. An ideal representation of 3D geome properties: a) encode complex geometries and art large scenes, c) encapsulate local and global inform 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 networl

Recently, several works [3, 26, 27, 31] have introc

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



**Convolutional Occupancy** Network [Peng et al., 2020]





 $<sup>^{\</sup>star}$  This work was done prior to joining Amazon.

## 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 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 parallelism by implementing the whole system using fully-fused CUDA kernew input encoding that permits the use of a smaller network without sac-rificing quality, thus significantly reducing the number of floating point nels with a focus on minimizing wasted bandwidth and compute operations. We achieve a combined speedup of several orders of magnitude, enabling and memory access operations: a small neural network is augmented by a training of high-quality neural graphics primitives in a matter of seconds, multiresolution hash table of trainable feature vectors whose values are opand rendering in tens of milliseconds at a resolution of  $1920 \times 1080.$ timized through stochastic gradient descent. The multiresolution structure CCS Concepts: • Computing methodologies  $\rightarrow$  Massively parallel algoallows the network to disambiguate hash collisions, making for a simple rithms; Vector / streaming algorithms; Neural networks. 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. Additional Key Words and Phrases: Image Synthesis, Neural Networks, Encodings, Hashing, GPUs, Parallel Computation, Function Approximation. ACM Reference Format: © 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. 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 ACM Trans. Graph., Vol. 41, No. 4, Article 102. Publication date: July 2022. Instant NGP [Müller et al., 2022]

### **NEURAL IMPLICITS** WORKING OUT THE DETAILS







## Training neural implicits: sampling points

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

Thomas Davies<sup>1</sup>, Derek Nowrouzezahrai<sup>2</sup> & Alec Jacobson<sup>1</sup> <sup>1</sup>Department of Computer Science, University of Toronto <sup>2</sup>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 latentencoded 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 reconstruc tion 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

a latent-encoded neural implicit.

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 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} := \left\{ \vec{x} \in \mathbb{R}^3 | f_\theta(\vec{x}; \vec{z}) = 0 \right\},\$ 

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 S encoded in  $\vec{z}$ . We refer to the representation in Eq. (1) as



Park et al. (2019) propose to optimize the weights  $\theta$  so each shape  $S_i \in \mathcal{D}$  in a dataset or shape distribution  $\mathcal{D}$  is encoded into a corresponding latent vector  $\vec{z_i}$ . If successfully trained, the weights  $\theta$  of their DEEPSDF implicit function  $f_{\theta}$  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  $S_i$ . We can draw a simple but powerful conclusion: in this setting, one can completely forgo the latent vector

high density



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



## Training neural implicits: sampling points

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

Thomas Davies<sup>1</sup>, Derek Nowrouzezahrai<sup>2</sup> & Alec Jacobson





stratified sampling

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



ght-Encoded vies et al., 2021] 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

# offset surfaces

distance queries

### **Signed Distance Function**

- | f(x) | encodes distance to closest point on surface
- many geometric queries easy



 $f(\mathbf{y})$ 



## How do we make a neural implicit an SDF?





## How do we make a neural implicit an SDF?

 $\mathscr{L}(x) = \mathscr{L}_0$ 

add regularization term to loss function

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





## **Eikonal loss**

**Implicit Geometric Regularization for Learning Shapes** 

Amos Gropp<sup>1</sup> Lior Yariv<sup>1</sup> Niv Haim<sup>1</sup> Matan Atzmon<sup>1</sup> Yaron Lipman<sup>1</sup>

### 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.

### 1. Introduction

2020

Jul

6

[cs.LG]

v:2002.10099v2

arXi

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.,

### $\mathcal{M} = \left\{ oldsymbol{x} \in \mathbb{R}^3 \mid f(oldsymbol{x}; heta) = 0 ight\},$ (1)

we call such representations *implicit neural representations*. point data  $\mathcal{X}$  and normals  $\mathcal{N}$ . Compared to the more traditional way of representing surfaces via implicit functions defined on volumetric grids (Wu

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

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



Eikonal Loss Figu our meth neural n the opti

16; Charter J., 2016; Dai et al., 2017; Stutz & 18), nurral implicit representations have the ben-ting the digrees of freedom of the network (i.e., s) directive bes sharp rather than to the fixed ion of the antiperturb Rece. So far, most previ-using implicit neural representations computed supervision; that is, by comparing f to a known et al., Geige ; efit of re paran e discre f with 3 (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  $\mathcal{X} = \{x_i\}_{i \in I} \subset \mathbb{R}^3$ , with or without normal data,  $\mathcal{N} = \{n_i\}_{i \in I} \subset \mathbb{R}^3$ , our goal is to compute  $\theta$  such that  $f(x; \theta)$  is approximately the signed where  $f : \mathbb{R}^3 \times \mathbb{R}^m \to \mathbb{R}$  is a multilayer perceptron (MLP); distance function to a plausible surface  $\mathcal{M}$  defined by the

> 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



## [Gropp et. al, 2020]







## Eikonal loss: walkthrough





## Eikonal loss: walkthrough

















## = max





















### **NEURAL IMPLICITS** GEOMETRIC OPERATIONS

# **PSEUDO-SDF** *eikonality distance property*





## **Eikonal loss**







## **Eikonal loss**

Eikonal Loss avg. value =  $7.1 \times 10^{-5}$ 





## **Closest point loss**



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

### This work is licensed 4.0 License

SA Conference Papers '23, December 12–15, 2023, Sydney, NSW, Australia © 2023 Copyright held by the owner/author(s). ACM ISBN 979-8-4007-0315-7/23/12. https://doi.org/10.1145/3610548.3618170

arise, such as CSG and swept volumes, and produce a true (neural) SDF for the result of these operations.

### CCS CONCEPTS

• Computing methodologies  $\rightarrow$  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 SIG-GRAPH 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]



### **NEURAL IMPLICITS GEOMETRIC OPERATIONS**

# Swept Volumes



Eikonal Loss avg. value = 7.1×10<sup>-5</sup>



















# Closest Point Map $x \mapsto \mathbf{x} - f(\mathbf{x}) \nabla_{\mathbf{x}} f(\mathbf{x})$

















## Closest Point Map $x \mapsto \mathbf{x} - f(\mathbf{x}) \nabla_{\mathbf{x}} f(\mathbf{x})$









## **NEURAL IMPLICITS**


# **Closest Point Loss** $E_{\rm CP} = f(\mathbf{x} - f(\mathbf{x}) \nabla_{\mathbf{x}} f(\mathbf{x}))^2$

# **NEURAL IMPLICITS** → GEOMETRIC OPERATIONS







# **Closest Point Loss**

### **NEURAL IMPLICITS** → GEOMETRIC OPERATIONS

# Eikonal Loss

**Closest Point Loss (ours)** avg. value = 0.003





110

# Neural signed distance fields... (+) enable many geometric operations (-) are difficult to maintain



**Closest Point Loss (ours)** avg. value = 0.003



111



# **NEURAL IMPLICITS** → GEOMETRIC OPERATIONS

# ed distance f any geometric



NeRF NeuS [Wang et al., 2021]03



# **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 net-works, enabling their use in training objectives and beyond. We also show a preliminary application to inverse rendering.

CCS Concepts: • Computing methodologies → Shape analysis; Shape **representations**; • **Mathematics of computing**  $\rightarrow$  *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

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



Fig. 1. Our method enables geometric queries on neural implicit surfaces, hout 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.

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).

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

# **NEURAL IMPLICITS GEOMETRIC OPERATIONS**





# **Alternative to SDFs for GP**

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



# **NEURAL IMPLICITS GEOMETRIC OPERATIONS**

**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

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





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





- 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 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**









Zoë Marschner zoem@cmu.edu



Daniel Ritchie daniel\_ritchie@brown.edu