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.

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

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

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 the zero level set of a function—for example, the Signed Distance Function (SDF) of a given surface—f_θ parameterized by a large set of network weights θ.

Direct manipulation of the network weights θ does not necessarily correspond to an intuitive change in the encoded shape. Therefore, Constructive Solid Geometry (CSG) is an especially attractive alternative for modelling with neural implicits. CSG defines geometric forms as *boolean operations* (unions, intersections and subtractions) on primitive shapes represented as implicit functions, such as SDFs. This naturally extends to editing neural implicits, by simply replacing the analytical SDFs with neural SDFs.

Unfortunately, boolean operations of SDFs are difficult to perform correctly. Common rephrasing in terms of basic comparative operations (e.g., the union of two shapes as the minimum of their SDFs) creates functions which only posses the correct zero level set. Away from the surface values are no longer necessarily true distances (see Figure 2). This makes the result ill-suited for downstream tasks, leading to potential decreases in efficiency (e.g., in sphere tracing, see Figure 4) and accuracy (e.g., in collision avoidance [Li et al. 2020], surface reconstruction [Sellán et al. 2023] or medial axis extraction [Rebain et al. 2021]). Performing even simple distance-based tasks, like surface offsetting, on the results of boolean operations performed in this way can be catastrophic (Figure 3).

In this work, we propose characterizing true SDFs in terms of a *closest point* property on an eikonal implicit function. We add this property as a simple regularizer for neural implicits that enforces true signed distance results for boolean operations, including for infinite CSG operations such as swept volumes. We show that this addition allows one to efficiently carry out downstream tasks that rely on a shape being represented by a true SDF, like sphere tracing and morphological operations. Further, we showcase the advantages of our neural approach by generalizing over a set of parametric objects and swept volume trajectories (see Figure 1).

2 RELATED WORK

Our work concerns neural implicit representations, which encode 3D objects through the iso-contour of a function. Several works have explored different types of implicit function [Chibane et al. 2020; Mescheder et al. 2019; Venkatesh et al. 2021] and their applications in, e.g., real-time rendering [Takikawa et al. 2021]. We refer the reader to [Xie et al. 2021] for a full survey of this field and instead focus our discussion on editing operations for neural

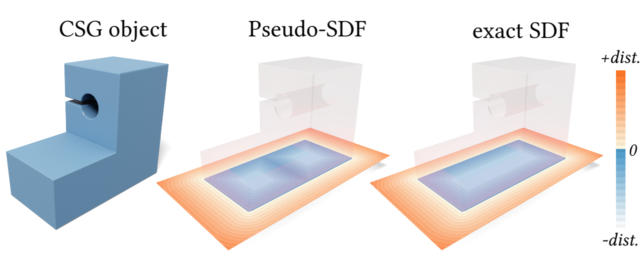


Figure 2: An implicit field built from CSG operations on SDFs (left) is often not a true distance function, but rather a "Pseudo-SDF" (middle). It satisfies the eikonal property almost everywhere, but is not a true SDF (right).

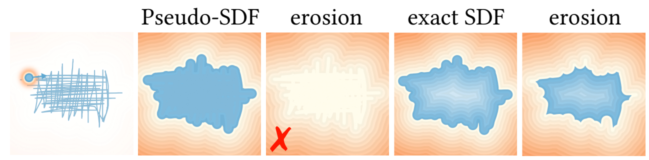


Figure 3: Sweeping through a canvas with a moving object is commonly used in modeling or digital drawing (left). One can perceive such a swept volume as the continuous union of the object as it moves along the path. Thus, as for CSG objects, swept volume also often leads to non-SDFs, on which basic geometric operations (e.g., erosion) give incorrect results (middle). A correct erosion on a true SDF is shown on the right.

Signed Distance Functions (SDFs) [Park et al. 2019]—specifically, Constructive Solid Geometry (CSG) operations [Ricci 1973]. We focus on performing CSG operations on neural SDFs, in contrast to other works that focus on using neural networks to generate CSG objects [Kania et al. 2020; Ren et al. 2021; Sharma et al. 2018; Yu et al. 2023, 2022] .

2.1 Neural Fields Editing

The increased recent attention on neural implicit representations has motivated the exploration of geometric editing operators defined on them. Perhaps the most direct way to edit a shape encoded by a neural implicit is by directly modifying network parameters [Park et al. 2019] or weights [Berzins et al. 2023; Davies et al. 2021]. However, direct weight editing often leads to undesired global changes to the output [Liu et al. 2022]. In response, a class of methods explore how to *indirectly* adjust network parameters. Their core idea, inspired by the level set method [Osher and Sethian 1988], is to formulate a target editing operation as a loss function, oftentimes a PDE. Minimizing the loss then serves as an indirect way of adjusting network weights to achieve the target edit, such as smoothing/sharpening [Mehta et al. 2022; Yang et al. 2021], physically-based deformation [Chen et al. 2022; Cuomo et al. 2022], and sculpting [Tzathas et al. 2023]. However, PDE-based approaches are quite slow as they require training the network at each time step to minimize the PDE objective. An alternative is to fix the network parameters and deform it with a vector field parameterized by another neural network (e.g., [Niemeyer et al. 2019]) to avoid re-training. Once trained, the neural vector field can directly deform any given neural implicits. By controlling the properties of the vector field, one can also achieve different editing behaviors [Zhang et al. 2022]. For a subclass of neural implicit architectures that "exposes" part of the neural features to a spatial domain, such as vertices of an octree [Takikawa et al. 2021], one can edit neural implicits by directly moving special neural features to a different location [Abou-Chakra et al. 2022; Gao et al. 2020] and compositing with different features [Liu et al. 2020].

Unfortunately, most editing operators mentioned above do not preserve the structural properties of the underlying implicit field, such as the *distance* property of neural SDFs. This results in the deformed neural "SDF" not being a distance function, preventing subsequent geometry processing operations (see Figure 3). One solution is to incorporate additional *regularization* to further control

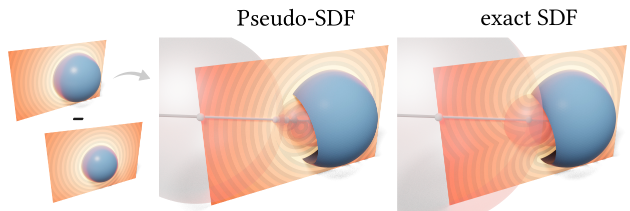


Figure 4: Using sphere tracing to render a Pseudo-SDF (middle) constructed by subtracting two SDFs (left) requires more iterations than rendering an exact SDF (right) due to incorrect exterior distance values.

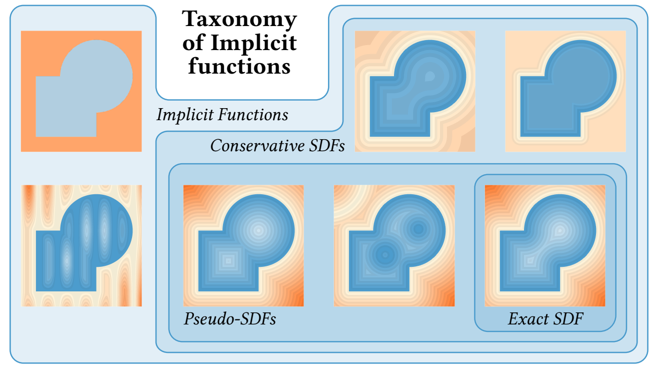


Figure 5: For a given boundary there is a uniquely defined *exact SDF* that is eikonal and satisfies the distance property, but there are many ways in which the conditions can be violated to different degrees. Pseudo-SDFs are functions that satisfy the eikonal property almost everywhere, but not the distance property (Equation 1). Conservative SDFs are not eikonal nor do they satisfy the distance property, but the function values are bounded by the actual distances. Implicit functions are generic level set functions that do not have any guarantees.

the editing operator. Many mesh-based regularizers have been proposed to encourage surface smoothness [Hertz et al. 2020; Kato et al. 2018; Liu et al. 2019; Wang et al. 2018] or isometry [Rakotosaona and Ovsjanikov 2020], but they are designed specifically for the *surface* of an object, instead of the entire implicit field. The most relevant work is the Eikonal regularization by Gropp et al. [2020] which encourages the gradient of the implicit field to have unit norm. This is an essential property all (signed) distance fields satisfy. However, in section 4, we show that the eikonal property alone is insufficient because many gradient-norm-one implicit functions are not SDFs. Our proposed regularization complements existing regularizations by encouraging the implicit field to be a true signed distance function. Our method is based on the closest point property of SDFs (see section 4), which also has proven useful in previous works [Gomes and Faugeras 2003; Ma et al. 2020]. In section 6, we demonstrate how our method can be combined with many neural editing operators to deform the implicit function while preserving the (signed) distance properties.

2.2 Constructive Solid Geometry

Constructive Solid Geometry is a modeling tool that describes a shape via boolean operations of other simpler shapes called primitives. As a general framework, CSG trees have been a part of Computer Aided Design since its infancy [Laidlaw et al. 1986; Requicha and Voelcker 1977], with more recent works proposing strategies to infer them from a given finalized shape using classic techniques [Du et al. 2018] or machine learning [Sharma et al. 2018].

Our work focuses on the atomic boolean operations that make up a CSG tree, whose efficient and robust computation has received much attention by the Computer Graphics research community. This proves a particularly difficult task for the case of explicit polygonal meshes [Bieri and Nef 1988; Fabri and Pion 2009].

In contrast, computing *implicit* booleans is simple: for example, the union of two shapes is encoded by the minimum of the two implicits representing them. Wyvill et al. [1999] [Barthe et al. 2004; Schmidt et al. 2007; Wyvill and Van Overveld 1997] exploit this fact to build full implicit modeling systems that combine booleans with other operations like blending, smoothing and deforming.

SDFs present many advantages over general implicit functions; among them is the possibility to render SDF surfaces efficiently via sphere tracing [Hart 1996]. This forms the basis for popular interactive modeling tools like *Shadertoy* [Quilez and Jeremias 2017] or Adobe Substance 3D modeler. While these typically employ SDF primitive shapes, the quality of the final representation degrades with each subsequent boolean operation (see, e.g., [Takikawa et al. 2022]). In some cases, this low quality may translate only into a less efficient rendering (Figure 4); in other applications, it could be catastrophic (see Figure 3).

A particular case of CSG operations are swept volumes, which may be phrased as an continuous union of shapes along a trajectory. Given the challenge of directly computing swept volumes of explicit meshes [Abrams and Allen 2000; Zhang et al. 2009], implicit functions like SDFs are used even in cases where both input and output are meshes [Schmidt and Wyvill 2005; Sellán et al. 2021]. Since these works rely heavily on the implicit boolean definition in terms of comparative functions, they will not produce a true SDF output. In this work, we present what we believe to be the first fully implicit swept volume algorithm that outputs a true SDF.

3 PSEUDO-SDFS

We consider the signed distance function $f_\theta : \mathbb{R}^n \to \mathbb{R}$ parameterized by a neural network with weights θ. A neural SDF f_θ has the same properties as a regular SDF. A surface Σ bounding a solid region may be implicitly encoded by the zero level set of an SDF: the set of all points $\boldsymbol{x}$ such that $f_\theta(\boldsymbol{x}) = 0$. The function outputs a positive distance from a query point $\boldsymbol{x}$ to its closest point on the surface Σ if the point $\boldsymbol{x}$ is outside of the shape, and the negative distance for interior points:

$$f_\theta(\boldsymbol{x}) = \begin{cases} d(\boldsymbol{x}, \Sigma) & \text{if } \boldsymbol{x} \text{ is outside of } \Sigma, \\ -d(\boldsymbol{x}, \Sigma) & \text{otherwise,} \end{cases} \tag{1}$$
$$d(\boldsymbol{x}, \Sigma) = \mathrm{Inf}_{\sigma\in\Sigma} \|\boldsymbol{x} - \sigma\|,$$

where $d(\boldsymbol{x}, \Sigma)$ denotes the (positive) distance from $\boldsymbol{x}$ to Σ. This implies that a true SDF must satisfy the *eikonal* property

$$\|\nabla f_\theta(\boldsymbol{x})\| = 1, \tag{2}$$

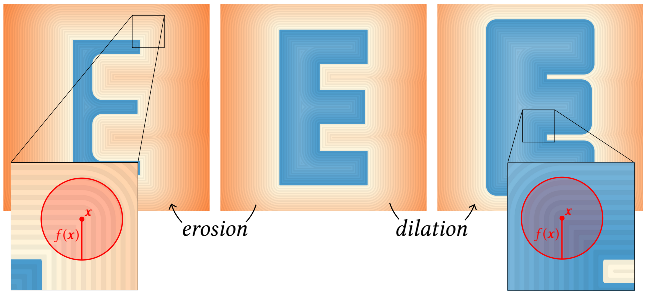


Figure 6: The dilation or erosion of an exact SDF (middle) may produce a Pseudo-SDF. Eroding a convex corner (left) or dilating a concave corner (right) results in a function that does not obey the distance property. In the blow-ups, a circle is shown centered at x with radius $f(x)$, where f is the eroded/dilated SDF. The circle does not intersect the zero level set, as it should in an exact SDF.

which states that the norm of the gradient is unit everywhere. However, the eikonal property is merely a necessary, but not sufficient, condition to guarantee that the underlying implicit function is a true SDF. As a result, solely encouraging an implicit function to be eikonal (e.g., [Gropp et al. 2020]) is insufficient to obtain a true SDF. We will first focus on discussing the family of SDF-like functions that are not SDFs, summarized in Figure 5, and then present our solution in section 4.

3.1 A Categorization of Implicit Functions

Exact SDF. When a function obeys the distance property in Equation 1, the function exactly encodes the distance and is thus a signed distance field. The unambiguity of this definition means that there is always exactly one function that satisfies this definition for any surface Σ, which we refer to as the *exact* SDF for emphasis. An exact SDF is preferable to general implicit functions in many applications, as the additional structure imposed by the distance property allows for more efficient and correct behavior in many applications, such as in Figure 3 and 4.

Conservative SDF. However, obtaining an exact SDF in practice is often difficult due to, for instance, discretization error. Thus one often enforces the *conservative* distance property; namely,

$$|f_\theta(\mathbf{x})| \leq |d(\boldsymbol{x}, \Sigma)|, \ \Sigma = f_\theta^{-1}(0). \tag{3}$$

Intuitively, this constraints the output distance of f_θ to have a smaller magnitude than the true distance d. A function that obeys this property is called a *conservative* SDF (see Figure 5). Because the conservative distance property is enforced, certain algorithms used on exact SDFs remain applicable, albeit with looser guarantees: for example, ray marching will still converge to a correct result as the steps are guaranteed to not overshoot the boundary.

Pseudo-SDF. A Pseudo-SDF is a piecewise differentiable implicit function that satisfies the eikonal property wherever the gradient is well defined, but does not satisfy the distance property. If a Pseudo-SDF is continuous, it must be conservative since the eikonal property bounds the maximum rate at which the function values can change. The fact that any Pseudo-SDF will satisfy the eikonal property implies that the widely-used eikonal regularization [Gropp et al. 2020] will not necessarily convert a function into an exact SDF, motivating the development of our method.

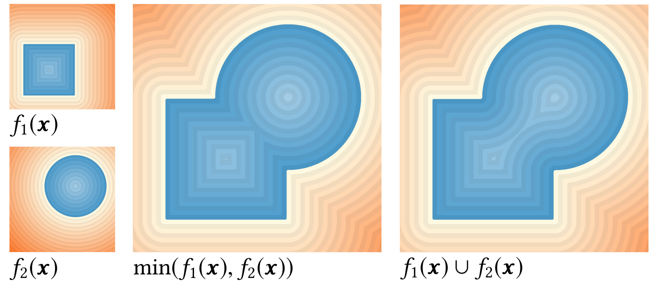


Figure 7: CSG operations, like the union between these two shapes (left) performed naïvely with min **and** max **operations produce Pseudo-SDFs (middle), differing from the expected exact SDF of the unioned shape (right).**

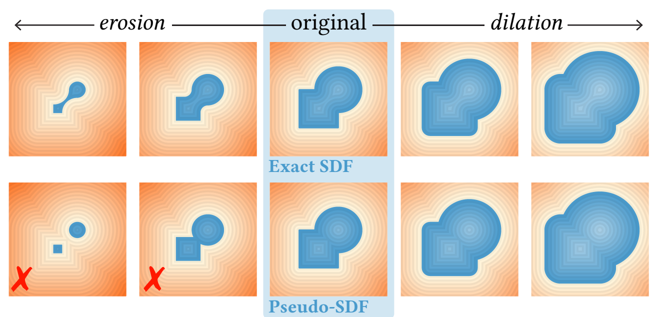


Figure 8: Dilation and erosion operations, commonly used for smoothing or extracting bounding shapes, can fail on Pseudo-SDFs. Compared to the true SDF of this union between two shapes (top), the Pseudo-SDF produced by the min union is not a true SDF in the interior, and therefore produces incorrect level sets when eroded (marked with red Xs). For large erosions, even the topology of the surface in the eroded Pseudo-SDF is incorrect.

These SDF-like functions, especially Pseudo-SDFs, arise naturally when performing geometric editing on exact SDFs. Even basic distance-based offsetting will result in Pseudo-SDFs in regions with non-zero curvature (see Figure 6). A more representative class is CSG operations, where Pseudo-SDFs are ubiquitous. CSG operations, including union $\cup$, intersection $\cap$, and difference $-$, on implicit functions f_1, f_2 are often rephrased with *max* and *min* operations

$$\begin{aligned} f_1 \cup f_2 &= \min(f_1, f_2) \\ f_1 \cap f_2 &= \max(f_1, f_2) \\ f_1 - f_2 &= f_1 \cap -f_2 = \max(f_1, -f_2), \end{aligned} \tag{4}$$

but performing CSG operations with these formulae does not produce an exact SDF (see Figure 7). This phenomenon also generalizes to swept volumes, which can be perceived as performing an infinite amount of CSG operations (see Figure 17).

In the context of neural "SDFs," the function obtained from a network may also be far from an exact SDF. This often happens in applications, such as in shape reconstruction and interpolation, where the ground truth supervision is not presented (see Figure 9). A popular (insufficient) remedy is to add the eikonal regularization

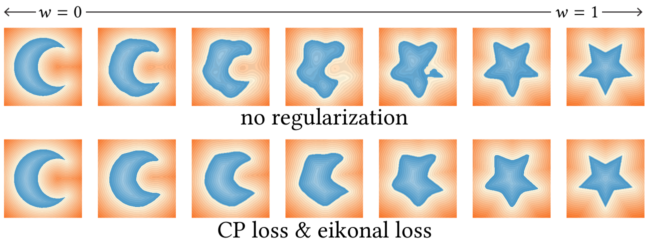


Figure 9: A parametric neural implicit $f_\theta(x, w)$ is trained to fit a moon for $f_\theta(x, 0)$ and a star for $f_\theta(x, 1)$. For $0 < w < 1$, the vanilla neural net produces non-SDFs results (top). Our method is able to encourage the interpolation to be exact SDFs (bottom).

[Gropp et al. 2020] to encourage $\|\nabla f_\theta(\boldsymbol{x})\| = 1$. However, as we discussed above, the eikonal property is insufficient to guarantee the output being an exact SDF. Instead, one often obtains a Pseudo-SDF as a outcome, which still causes difficulties in downstream tasks.

Our method, based on sufficient conditions for the exact SDF, is designed to remedy generic (neural) non-SDF functions and Pseudo-SDFs that are ubiquitous in geometric editing so that one can obtain an exact SDF and easily perform downstream tasks.

4 CLOSEST POINT ENERGY

In this section we introduce the *closest point loss* to quantify how far an implicit function is from a true SDF. Our method serves as a regularization during neural network training to encourage a neural implicit to be an exact SDF. Unlike the commonly used eikonal loss, our closest point loss is able to detect the difference between Pseudo-SDFs and exact SDFs.

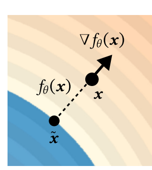
Given a query point $\boldsymbol{x}$, the closest point $\tilde{\boldsymbol{x}}$ on the surface Σ is defined as

$$\tilde{\boldsymbol{x}} := \arg\min_{\sigma \in \Sigma} \|\boldsymbol{x} - \sigma\|, \tag{5}$$

If Σ is represented by an exact SDF f_θ one can, by definition, obtain the closest point $\tilde{\boldsymbol{x}} \in \Sigma$ for a given query $\boldsymbol{x}$ by multiplying the negated gradient $-\nabla f_\theta$ at $\boldsymbol{x}$ with the value of f_θ and adding the vector to $\boldsymbol{x}$ (see inset):

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} - f_\theta(\boldsymbol{x}) \nabla_{\boldsymbol{x}} f_\theta(\boldsymbol{x}). \tag{6}$$

We call this the *closest point function.* Intuitively, $-\nabla f_\theta$ is the unit vector (due to the eikonal property) pointing towards the direction of maximal decrease in distance. f_θ gives the distance to the the surface, indicating how far we need to travel along $-\nabla f_\theta$. We define our closest point energy for a set of points $\boldsymbol{x} \in \mathcal{X}$ as



$$E_{\text{CP}} = \frac{1}{|\mathcal{X}|} \sum_{\boldsymbol{x} \in \mathcal{X}} f_\theta(\boldsymbol{x} - f_\theta(\boldsymbol{x}) \nabla_{\boldsymbol{x}} f_\theta(\boldsymbol{x}))^2. \tag{7}$$

This energy measures how far from the zero level set a point is mapped by the closest point function by computing the norm of the SDF at the mapped point. In Figure 10, we visualize the computation of the energy and show how our E_{CP} identifies the difference between Pseudo-SDFs and exact SDFs.

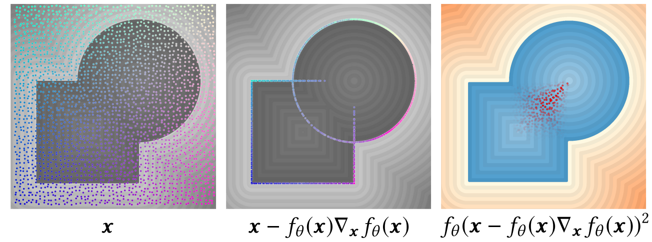


Figure 10: The closest point loss detects the regions where a Pseudo-SDF is not a distance function. Schematically, the loss is computed for a set of points (left) by applying the closest point formula Equation 6 (middle), yielding a loss based on the value of the mapped points in f_θ (right), shown here with higher opacity denoting higher values of the closest point loss.

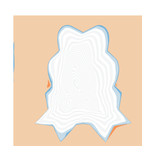
4.1 Regularizations for Exact SDFs

Our closest point energy in Equation 7 is the missing piece needed to encourage a neural implicit function to be an exact Distance Function (DF), either an SDF or unsigned distance function. In practice, it is always necessary for the problem formulation to include some specification of the sign of different regions, to break the symmetry inherent from the fact that the orientation of a surface is not determined by its zero level set alone. In our problems, the editing energies break this symmetry, giving us exact *signed* distance fields. If an *eikonal* function f_θ has zero level set Σ and at every point $\boldsymbol{x} \in \mathbb{R}^n$ the point $\tilde{\boldsymbol{x}}$ computed from Equation 6 is in Σ, then f_θ is an exact DF. This can be proven by contradiction: if f is not an DF, then at some $\boldsymbol{x}$ it does not measure the distance to its closest point on Σ; in other words, $\tilde{\boldsymbol{x}}$ is not the closest point to $\boldsymbol{x}$ on Σ. If that is the case, then there exists some other point $\boldsymbol{y} \in \Sigma$ such that $|f(\boldsymbol{x})| = \|\boldsymbol{x} - \tilde{\boldsymbol{x}}\| > \|\boldsymbol{x} - \boldsymbol{y}\|$. But by eikonality, $|f(\boldsymbol{x}) - f(\boldsymbol{y})| < \|\boldsymbol{x} - \boldsymbol{y}\|$, meaning that $|f(\boldsymbol{x}) - f(\boldsymbol{y})| < |f(\boldsymbol{x})|$ and thus necessarily $|f(\boldsymbol{y})| > 0$, which is impossible since $\boldsymbol{y} \in \Sigma$.

The above observation proves that the *eikonal* property (Equation 2) and the *closest point* property (Equation 6) combined are sufficient to characterize exact SDFs. This forms the foundation of our proposed regularization for neural implicits. When training a neural network f_θ to find the optimal parameters θ, we propose augmenting the original loss function $\mathcal{L}$ with the closest point loss E_{CP} and with the eikonal regularization E_{eik} [Gropp et al. 2020] as:

$$\theta^\star = \arg\min_\theta \mathcal{L}(\theta) + \lambda_c E_{\text{CP}}(\theta, \mathcal{X}) + \lambda_e E_{\text{eik}}(\theta, \mathcal{X}). \tag{8}$$

Our regularizations are sufficient conditions to encourage fitting an exact SDF. In the inset, we visualize a neural swept volume with E_{CP} but without E_{eik} (see Figure 19 for its setup). This ablation demonstrates the importance of having both regularization terms: without E_{eik}, training can get stuck in local minima like this one, where all the values are essentially zero. In section 5, we will discuss in detail how we augment our regularizations for each neural field editing operation.



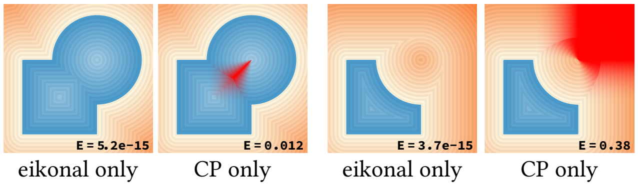


Figure 11: Given two Pseudo-SDFs computed from union and subtraction, the eikonal energy E_{eik} returns zero loss up to machine precision, providing zero gradient to repair the Pseudo-SDFs. In contrast, the closest point loss E_{CP} returns non-zero loss in regions that do not satisfy the distance property (shown in red, with higher opacity corresponding to higher loss values).

5 LOSS FUNCTIONS FOR NEURAL SDF EDITING

To encourage CSG style editing operations to output exact neural SDFs, we train a network f_θ using a combination of *editing loss functions*, which capture the specific editing operations and are described in subsection 5.1 for CSG operations and subsection 5.2 for swept volumes, and the regularization terms described in Equation 8.

5.1 Constructive Solid Geometry

For this operation, we aim to learn a network representing the SDF of a sequence of CSG operations $\text{CSG}(\boldsymbol{x}) = ((s_0 \oplus_0 s_1) \oplus_1 s_2 \ldots)(\boldsymbol{x})$ where each $\oplus_i$ is a CSG operation (union, intersection, and subtraction) and s_i are the input SDFs. The s_i can be represented in any queryable form—such as a primitive SDF defined through math operations, an SDF on a voxel grid, or even a neural SDF. For the given sequence of CSG operations, we can compute the Pseudo-SDF of the result using min and max functions as described in Equation 4. We denote the output of this series of min and max operations $\text{CSG}_\approx$. Because this sequence of operations results in a Pseudo-SDF, the conservative distance property holds. That is,

$$|\text{CSG}(\{s_i\}, \{\oplus_i\})| \geq |\text{CSG}_\approx|. \tag{9}$$

We therefore want to constrain our network f_θ to obey the inequality

$$|f_\theta(\boldsymbol{x})| \geq |\text{CSG}_\approx(\boldsymbol{x})| \quad \forall\, \boldsymbol{x} \in \mathbb{R}^n. \tag{10}$$

This condition can be equivalently written using the Heaviside step function as

$$H(-\text{sgn}(\text{CSG}_\approx(\boldsymbol{x}))(f_\theta(\boldsymbol{x}) - \text{CSG}_\approx(\boldsymbol{x}))) = 0 \quad \forall\, \boldsymbol{x} \in \mathbb{R}^n. \tag{11}$$

While this function correctly formulates CSG operations with a loss function, it is not practical as the step function has no meaningful gradients. To overcome this, we relax the step function to a narrow sigmoid parameterized by a width parameter α, leading to the final loss function which, computed at a set of points $\mathcal{X}$, is

$$E_{\text{CSG}}(f_\theta, \mathcal{X}; \alpha) = \frac{1}{|\mathcal{X}|} \sum_{\boldsymbol{x} \in \mathcal{X}} \sigma\left(-\alpha\, \text{sgn}(\text{CSG}_\approx(\boldsymbol{x}))(f_\theta(\boldsymbol{x}) - \text{CSG}_\approx(\boldsymbol{x})\right)$$

where σ denotes the sigmoid function. This loss E_{CSG} ensures the correct zero level set is learned. When combined with our regularization losses $E_{\text{CP}}, E_{\text{eik}}$ (see Equation 8), this will encourage the CSG result be an exact SDF.

5.2 Swept Volumes

The swept volume problem involves computing the region of space covered by an object as it moves along a path. We consider a general formulation of this problem: the geometry of a shape moving in time is input as an SDF spacetime function $s(\boldsymbol{x}, t)$; to solve the swept volume problem we seek to compute the infinite union of the SDFs $s(\boldsymbol{x}, t_i)$ for all $t_i \in [0, 1]$. This problem is harder than the CSG problem as there is no feasible way to explicitly construct an approximate SDF representing the swept volume for the general case, so an overall loss derived simply from bounding the approximate SDF as obtained in the previous subsection is not feasible. Instead, we rely on the following observation: if there exists a spacetime point $(\boldsymbol{x}, t)$ satisfying $s(\boldsymbol{x}, t) < 0$, then the value at $\boldsymbol{x}$ in the exact swept volume, $\text{SV}(\boldsymbol{x})$, must satisfy

$$\text{SV}(\boldsymbol{x}) \leq s(\boldsymbol{x}, t). \tag{12}$$

This bound allows us to write a loss analogous to E_{CSG} for the swept volume operation:

$$E_{\text{SV},-}(f_\theta, \mathcal{X}; \alpha) = \frac{1}{|\mathcal{X}|} \sum_{\boldsymbol{x} \in \mathcal{X}} [s(\boldsymbol{x}, t) \leq 0]\, \sigma\left(\alpha(f_\theta(\boldsymbol{x}) - s(\boldsymbol{x}, t))\right), \tag{13}$$

where the term $s(\boldsymbol{x}, t) \leq 0$ is a scalar mask. This loss encodes the structure we know about the interior of the SDF of the swept volume. However, this loss alone cannot solve the swept volume problem as one can construct many different exact SDFs that satisfy $E_{\text{SV},-}(f_\theta, \mathcal{X}; \alpha) = 0$ for all points. Thus, we add additional loss term which captures the idea that every point not required to be negative by $E_{\text{SV},-}$ should be positive:

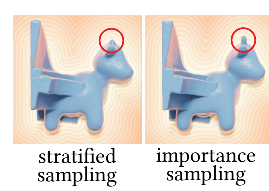
$$E_{\text{SV},+}(f_\theta, \mathcal{X}; \alpha) = \frac{1}{|\mathcal{X}|} \sum_{\boldsymbol{x} \in \mathcal{X}} \sigma\left(-\alpha f_\theta(\boldsymbol{x})\right). \tag{14}$$

The final editing loss for swept volumes is constructed from the sum of these two losses (see subsection 6.2).

5.3 Implementation Details

We represent our implicit function with a multi-layer perceptron with ReLU non-linearities. For all tests, we use 7 hidden layers with a hidden size of 128, initialized with the method by He et al. [2015]. The network is trained using Adam [Kingma and Ba 2015], with step size 10^{-4}. At each epoch of training, the loss functions are evaluated on a set of points, which are generated during training. To reduce the time spent generating sample points, we reuse the same point set for $N \approx 20$ epochs.

5.3.1 Sampling. As our loss is defined uniformly across a set of sample points $\mathcal{X}$ (see Equation 7), the way we sample points $\mathcal{X}$ during training creates a bias towards different regions. Because the zero level set of f_θ—the surface represented by the SDF—has much more visual importance than the rest of the function, we choose to sample points in a way that weighs the values of losses near the zero level set more. The inset shows the importance of this in achieving a visually sharp zero level set: in the union computed with no importance sampling, Spot's horns—a



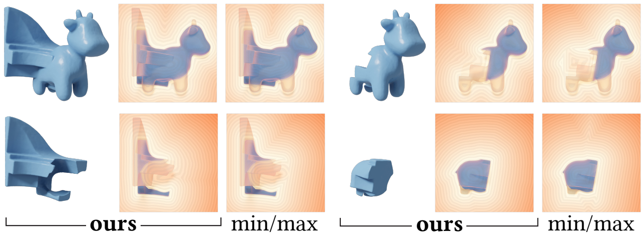


Figure 12: Computing CSG operations—union (top left), differences (top right, bottom left), and intersection (bottom right)—with the min/max operators leads to Pseudo-SDFs (third columns). Adding our regularization encourages the neural SDF to be an exact SDF (second columns).

sharp feature—do not show up in the zero level set. To implement this sampling strategy, we generate two point sets, $\mathcal{X} = \mathcal{X}_{\text{imp}} + \mathcal{X}_{\text{amb}}$. The points $\boldsymbol{x} \in \mathcal{X}_{\text{imp}}$ are importance sampled based on their nearness to the zero level sets of the shape SDFs defining the operation. We implement this with rejection sampling: points $\boldsymbol{x}$ are sampled using stratified sampling in the entire domain ($\mathbb{R}^n$ for CSG and $\mathbb{R}^{n+1}$ for swept volume) and accepted when they satisfy the criteria

$$\text{rand}(0, 1) \geq G(s(\boldsymbol{x}), \sigma), \tag{15}$$

where $G(\mu, \sigma)$ is a Gaussian with mean μ and standard deviation σ, and $\text{rand}(0, 1)$ is a random number from the uniform distribution between 0 and 1. This procedure produces points near the zero level sets of the original functions, a super set of the zero level set of the edited shape.

The second set of points, $\mathcal{X}_{\text{amb}}$ is simply constructed through stratified sampling on the entire domain. It is necessary to ensure the function is an SDF everywhere in the trained region. This sampling procedure has two parameters, the σ of the Gaussian used for importance sampling and the fraction of points sampled by importance sampling.

5.3.2 Parametric Neural SDFs. A particular advantage of neural SDFs is their ability to represent families of shapes in their latent space. This is especially applicable to our application, as it allows for the computation of entire families of CSG operations and swept volumes by training a single network—such as a network parametric over the path of the swept volume or over parameters of shapes involved in CSG operations. Such a network requires a single upfront training cost, after which specific instances of the problem can be solved through a simple network evaluation. To implement this, we increase the dimension of the network: in n dimensional space, a network representing an SDF with m parameters will be a function $f_\theta : \mathbb{R}^{n+m} \to \mathbb{R}$. With this change, we can apply the same loss functions as before. Note that the gradient $\nabla_{\boldsymbol{x}} f_\theta$ term in our losses is the spatial gradient, excluding the additional parametric dimensions.

6 RESULTS

We evaluate how our method can be used to train neural implicits of exact SDFs under some geometric editing operations, specifically CSG operations and swept volumes. Our model is implemented in PyTorch [Paszke et al. 2019] and trained on a NVIDIA GeForce RTX 3090 GPU.

6.1 Constructive Solid Geometry Operations

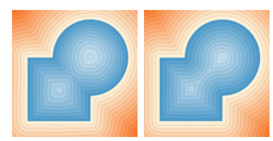
We use our regularizations from Equation 8 to compute CSG operations without obtaining a Pseudo-SDF, as the naïve method does. We define the loss function as

$$E = \lambda_1 E_{\text{CSG}} + \lambda_2 E_{\text{CP}} + \lambda_3 E_{\text{eik}}. \tag{16}$$

We use the same hyper parameters for each of the tests: the weightings of the losses are chosen to be $(\lambda_1, \lambda_2, \lambda_3) = (15, 1, 1)$ and the sharpness of the sigmoid in E_{CSG} is $\alpha = 300$.

In Figure 12, we show how one can use our method to learn SDFs of CSG operations on 3D objects. Unlike the Pseudo-SDFs created by computing CSG operations with min and max, we can correctly compute dilations and erosions of our result (see Figure 18). In addition to computing SDFs of single CSG operations, we can apply our method to CSG operations over parametric shapes to obtain a parametric neural SDF (see subsubsection 5.3.2) that outputs exact SDFs for a family of shapes. In Figure 16, we train a neural network to represent the entire family of pin shapes. Once trained (≈ 10 hours on a single NVIDIA GeForce RTX 3090 GPU), one can easily obtain the exact SDF of any instance within the family via a single network evaluation.

In the inset, we perform an ablation study to investigate the influence of the closest point loss. Removing E_{CP} from Equation 16 without changing the initialization causes the network to get stuck in a local minimum which is eikonal but has the incorrect zero level set (not pictured). When initialized with the Pseudo-SDF, optimizing with E_{eik} only remains at this minimum. This shows the importance of incorporating both E_{CP} and E_{eik} as regularizers to encourage convergence to the exact SDF.



6.2 Swept Volumes

For swept volumes problems, the loss function is

$$E = \lambda_1 E_{\text{SV},-} + \lambda_2 E_{\text{SV},+} + \lambda_3 E_{\text{CP}} + \lambda_4 E_{\text{eik}}. \tag{17}$$

As before, we use the same set of hyper-parameters for all problems described: $(\lambda_1, \lambda_2, \lambda_3, \lambda_4) = (10, 0.5, 0.1, 0.1)$, and for the sharpness of the sigmoid functions we choose $\alpha_1 = \alpha_2 = 300$.

In Figure 19, we evaluate our method on computing a swept volume along a single cubic path. In Figure 13, we test our swept volume method on SDFs computed from brush profiles included in Photoshop. Because of the complex profile, these shapes lead to especially incorrect interior SDFs in the stamped result, while our network learns an exact SDF of the resulting sweep. We can also take advantage of neural implicit's ability to learn swept volumes over a family of paths. In Figure 14, we take a *data-driven* approach, collecting a set of handwriting data from [Marti and Bunke 2002] as our training data, represented as cubic Bézier curves. This ensures our network spends its capacity on the distribution of handwriting, instead of infinite arbitrary paths. Our method enables a trained network to output an exact SDF of a "hello" hand-written by another writer who is not included in the training data. In addition to cubic Bézier curves, our method is applicable to swept volume of different kinds of path, such as those including rotations (see Figure 15). We can train networks parametric over properties of the path, allowing

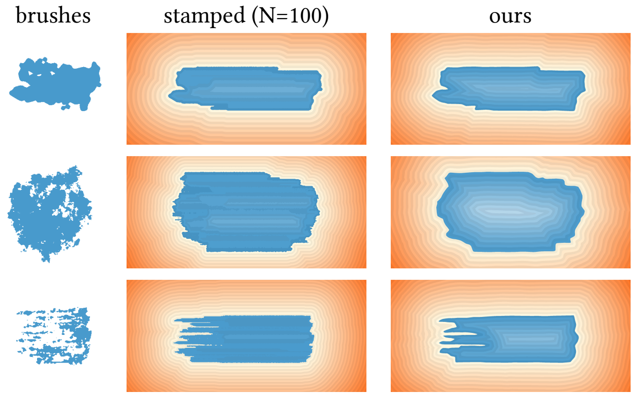


Figure 13: Given a set of brush profiles included in Photoshop (left), our method is able to learn exact SDFs of swept volumes (right), as opposed to the incorrect SDF resulting from stamping (middle).

Figure 14: We train a neural swept volume network with a circle sweep shape on a set of handwriting paths obtained from [Marti and Bunke 2002], visualized on the left. On the right, we demonstrate how our method is able to produce and generalize an exact SDF of the "hello" that is written by a writer not included in the training data.

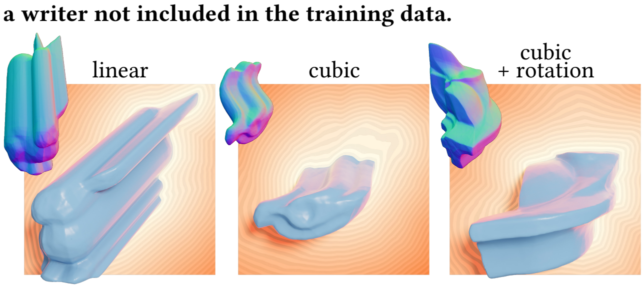


Figure 15: Our method generalizes across different types of swept paths, such as linear and cubic Bézier paths with or without rotations. Our method learns an exact SDF in 3D space, here visualized on a 2D slice.

us to sweep a rotating space shuttle (Figure 20) or construct families of paths for drawing in virtual reality (Figure 21). Figure 1 brings together CSG operations and swept volumes by directly computing the unions of shapes sampled from a neural implicit trained to represent sweeps parametric over cubic paths.

7 LIMITATIONS & FUTURE WORK

We propose regularizing neural implicits with the closest point and eikonal losses to encourage the result to be an exact SDF (see Equation 8). Combined with geometric editing losses (see section 5), our approach is able to correctly produce exact SDFs for CSG objects and swept volumes. Our current approach, however, requires hours of re-training to repair a Pseudo-SDF; exploring fast fine-tuning (e.g., low-rank updates [Hu et al. 2022]) of a pre-trained model could boost the efficiency of the repairing process.

An exciting future direction is to evaluate our regulraization on neural SDF applications beyond computation of CSG operations, such as test-time reconstruction [Duggal et al. 2022] or generation [Chen and Zhang 2019].

ACKNOWLEDGMENTS

The second author is supported in part by an NSERC Vanier Doctoral scholarship. The MIT Geometric Data Processing group acknowledges the generous support of Army Research Office grants W911NF2010168 and W911NF2110293, of Air Force Office of Scientific Research award FA9550-19-1-031, of National Science Foundation grants IIS-1838071 and CHS-1955697, from the CSAIL Systems that Learn program, from the MIT–IBM Watson AI Laboratory, from the Toyota–CSAIL Joint Research Center, from a gift from Adobe Systems, and from a Google Research Scholar award.

We thank the MIT.nano Immersion Lab and Steve Marschner for providing VR equipment and Annie, Heidi, and Steve Marschner for contributing VR drawings.

REFERENCES

Jad Abou-Chakra, Feras Dayoub, and Niko Sünderhauf. 2022. ParticleNeRF: A Particle-Based Encoding for Online Neural Radiance Fields in Dynamic Scenes. *CoRR* abs/2211.04041 (2022). https://doi.org/10.48550/arXiv.2211.04041 arXiv:2211.04041

Steven Abrams and Peter K Allen. 2000. Computing swept volumes. *The Journal of Visualization and Computer Animation* 11, 2 (2000), 69–82.

Loïc Barthe, Brian Wyvill, and Erwin De Groot. 2004. Controllable Binary CSG Operators for Soft Objects. *International Journal of Shape Modeling* 10, 02 (2004), 135–154.

Arturs Berzins, Moritz Ibing, and Leif Kobbelt. 2023. Neural Implicit Shape Editing using Boundary Sensitivity. *CoRR* abs/2304.12951 (2023). https://doi.org/10.48550/arXiv.2304.12951 arXiv:2304.12951

Hanspeter Bieri and Walter Nef. 1988. Elementary set operations with d-dimensional polyhedra. In *Computational Geometry and its Applications: CG'88, International Workshop on Computational Geometry Würzburg, FRG, March 24–25, 1988 Proceedings*. Springer, 97–112.

Honglin Chen, Rundi Wu, Eitan Grinspun, Changxi Zheng, and Peter Yichen Chen. 2022. Implicit Neural Spatial Representations for Time-dependent PDEs. *CoRR* abs/2210.00124 (2022). https://doi.org/10.48550/arXiv.2210.00124 arXiv:2210.00124

Zhiqin Chen and Hao Zhang. 2019. Learning Implicit Fields for Generative Shape Modeling. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

Julian Chibane, Aymen Mir, and Gerard Pons-Moll. 2020. Neural Unsigned Distance Fields for Implicit Function Learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.).

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. 2022. Scientific Machine Learning Through Physics-Informed Neural Networks: Where we are and What's Next. *J. Sci. Comput.* 92, 3 (2022), 88. https://doi.org/10.1007/s10915-022-01939-z

Thomas Davies, Derek Nowrouzezahrai, and Alec Jacobson. 2021. On the Effectiveness of Weight-Encoded Neural Implicit 3D Shapes. arXiv:2009.09808 [cs.GR]

Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. 2018. Inversecsg: Automatic conversion of 3d models to csg trees. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–16.

Shivam Duggal, Zihao Wang, Wei-Chiu Ma, Sivabalan Manivasagam, Justin Liang, Shenlong Wang, and Raquel Urtasun. 2022. Mending neural implicit modeling for 3d vehicle reconstruction in the wild. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 1900–1909.

Andreas Fabri and Sylvain Pion. 2009. CGAL: The computational geometry algorithms library. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*. 538–539.

Jun Gao, Wenzheng Chen, Tommy Xiang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2020. Learning Deformable Tetrahedral Meshes for 3D Reconstruction. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/7137debd45ae4d0ab9aa953017286b20-Abstract.html

José Gomes and Olivier Faugeras. 2003. The vector distance functions. *International Journal of Computer Vision* 52 (2003), 161–187.

Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. 2020. Implicit Geometric Regularization for Learning Shapes. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 3789–3799.

John C Hart. 1996. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer* 12, 10 (1996), 527–545.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*. IEEE Computer Society, 1026–1034. https://doi.org/10.1109/ICCV.2015.123

Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. 2020. Deep geometric texture synthesis. *ACM Trans. Graph.* 39, 4 (2020), 108.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net. https://openreview.net/forum?id=nZeVKeeFYf9

Kacper Kania, Maciej Zieba, and Tomasz Kajdanowicz. 2020. UCSG-NET-unsupervised discovering of constructive solid geometry tree. *Advances in Neural Information Processing Systems* 33 (2020), 8776–8786.

Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. 2018. Neural 3D Mesh Renderer. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 3907–3916.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

David H Laidlaw, W Benjamin Trumbore, and John F Hughes. 1986. Constructive solid geometry for polyhedral objects. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 161–170.

Minchen Li, Zachary Ferguson, Teseo Schneider, Timothy R Langlois, Denis Zorin, Daniele Panozzo, Chenfanfu Jiang, and Danny M Kaufman. 2020. Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. *ACM Trans. Graph.* 39, 4 (2020), 49.

Hsueh-Ti Derek Liu, Francis Williams, Alec Jacobson, Sanja Fidler, and Or Litany. 2022. Learning Smooth Neural Functions via Lipschitz Regularization. In *SIGGRAPH '22: Special Interest Group on Computer Graphics and Interactive Techniques Conference, Vancouver, BC, Canada, August 7 - 11, 2022*, Munkhtsetseg Nandigjav, Niloy J. Mitra, and Aaron Hertzmann (Eds.). ACM, 31:1–31:13. https://doi.org/10.1145/3528233.3530713

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020. Neural Sparse Voxel Fields. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/b4b758962f17808746e9bb832a6fa4b8-Abstract.html

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7708–7717.

Baorui Ma, Zhizhong Han, Yu-Shen Liu, and Matthias Zwicker. 2020. Neural-pull: Learning signed distance functions from point clouds by learning to pull space onto surfaces. *arXiv preprint arXiv:2011.13495* (2020).

U-V Marti and Horst Bunke. 2002. The IAM-database: an English sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition* 5 (2002), 39–46.

Ishit Mehta, Manmohan Chandraker, and Ravi Ramamoorthi. 2022. A Level Set Theory for Neural Implicit Evolution Under Explicit Flows. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 13662)*, Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer, 711–729. https://doi.org/10.1007/978-3-031-20086-1_41

Lars M. Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. 2019. Occupancy Networks: Learning 3D Reconstruction in Function Space. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 4460–4470.

Michael Niemeyer, Lars M. Mescheder, Michael Oechsle, and Andreas Geiger. 2019. Occupancy Flow: 4D Reconstruction by Learning Particle Dynamics. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 5378–5388. https://doi.org/10.1109/ICCV.2019.00548

Stanley Osher and James A Sethian. 1988. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of computational physics* 79, 1 (1988), 12–49.

Jeong Joon Park, Peter Florence, Julian Straub, Richard A. Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 165–174.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 8024–8035. https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

Iñigo Quilez and Pol Jeremias. 2017. Shadertoy. *Retrieved March* 27 (2017), 2017.

Marie-Julie Rakotosaona and Maks Ovsjanikov. 2020. Intrinsic Point Cloud Interpolation via Dual Latent Space Navigation. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12347)*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer, 655–672.

Daniel Rebain, Ke Li, Vincent Sitzmann, Soroosh Yazdani, Kwang Moo Yi, and Andrea Tagliasacchi. 2021. Deep medial fields. *arXiv preprint arXiv:2106.03804* (2021).

Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, Haiyong Jiang, Zhongang Cai, Junzhe Zhang, Liang Pan, Mingyuan Zhang, Haiyu Zhao, and Shuai Yi. 2021. CSG-Stump: A Learning Friendly CSG-Like Representation for Interpretable Shape Parsing. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 12458–12467.

Aristides AG Requicha and Herbert B Voelcker. 1977. Constructive solid geometry. (1977).

A. Ricci. 1973. A constructive geometry for computer graphics. *Comput. J.* 16, 2 (01 1973), 157–160.

Ryan Schmidt and Brian Wyvill. 2005. Implicit sweep surfaces. *Department of Computer Science. University of Calgary* (2005), 133.

Ryan Schmidt, Brian Wyvill, Mario Costa Sousa, and Joaquim A Jorge. 2007. Shapeshop: Sketch-based solid modeling with blobtrees. In *ACM SIGGRAPH 2007 courses*. 43–es.

Silvia Sellán, Noam Aigerman, and Alec Jacobson. 2021. Swept volumes via spacetime numerical continuation. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–11.

Silvia Sellán, Christopher Batty, and Oded Stein. 2023. Reach For the Spheres: Tangency-Aware Surface Reconstruction of SDFs. *arXiv preprint arXiv:2308.09813* (2023).

Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. 2018. Csgnet: Neural shape parser for constructive solid geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5515–5523.

Towaki Takikawa, Andrew Glassner, and Morgan McGuire. 2022. A Dataset and Explorer for 3D Signed Distance Functions. *Journal of Computer Graphics Techniques (JCGT)* 11, 2 (27 April 2022), 1–29. http://jcgt.org/published/0011/02/01/

Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. 2021. Neural Geometric Level of Detail: Real-Time Rendering With Implicit 3D Shapes. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 11358–11367.

Petros Tzathas, Petros Maragos, and Anastasios Roussos. 2023. 3D Neural Sculpting (3DNS): Editing Neural Signed Distance Functions. In *IEEE/CVF Winter Conference on Applications of Computer Vision, WACV 2023, Waikoloa, HI, USA, January 2-7, 2023*. IEEE, 4510–4519. https://doi.org/10.1109/WACV56688.2023.00450

Rahul Venkatesh, Tejan Karmali, Sarthak Sharma, Aurobrata Ghosh, R. Venkatesh Babu, László A. Jeni, and Maneesh Singh. 2021. Deep Implicit Surface Point Prediction Networks. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 12633–12642.

Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. 2018. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI (Lecture Notes in Computer Science, Vol. 11215)*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (Eds.). Springer, 55–71.

Brian Wyvill, Andrew Guy, and Eric Galin. 1999. Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. In *Computer Graphics Forum*, Vol. 18. Wiley Online Library, 149–158.

Brian Wyvill and Kees Van Overveld. 1997. Warping as a modelling tool for csg/implicit models. In *Proceedings of 1997 International Conference on Shape Modeling and*

Applications. IEEE, 205–213.

Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. 2021. Neural Fields in Visual Computing and Beyond. *CoRR* abs/2111.11426 (2021). arXiv:2111.11426

Guandao Yang, Serge J. Belongie, Bharath Hariharan, and Vladlen Koltun. 2021. Geometry Processing with Neural Fields. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 22483–22497.

Fenggen Yu, Qimin Chen, Maham Tanveer, Ali Mahdavi Amiri, and Hao Zhang. 2023. DualCSG: Learning Dual CSG Trees for General and Compact CAD Modeling. https://arxiv.org/abs/2301.11497

Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. 2022. CAPRI-Net: learning compact CAD shapes with adaptive primitive assembly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11768–11778.

Paul Zhang, Dmitriy Smirnov, and Justin Solomon. 2022. Wassersplines for Neural Vector Field-Controlled Animation. *Comput. Graph. Forum* 41, 8 (2022), 31–41. https://doi.org/10.1111/cgf.14621

Xinyu Zhang, Young J Kim, and Dinesh Manocha. 2009. Reliable sweeps. In *2009 SIAM/ACM joint conference on geometric and physical modeling*. 373–378.

Figure 16: Given a family of pin shapes controlled by (h_1, h_2, r_2, h_3) (left), we use our method to train a parametric neural implicit to represent the exact SDFs of all CSG operations in this family (middle). One the right, we compare the implicit function of a particular instance within this pin family computed naïvely with min and max operations (top right) and as a neural SDF with our regularization applied during training (bottom right).

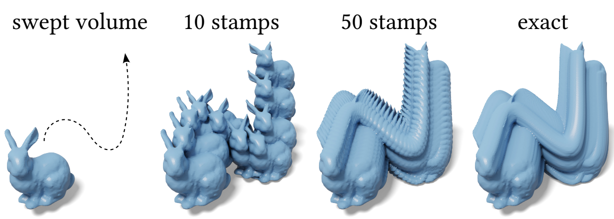


Figure 17: A swept volume can be perceived as a union of an infinite amount of shape "stamps" along a path.

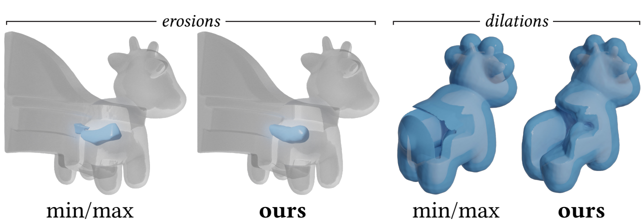


Figure 18: With Pseudo-SDFs computed from CSG operations based on min/max (first, third), one does not obtain the correct erosion and dilation behavior. Our method, which does not produce the same Pseudo-SDF artifacts, is able to obtain correct results.

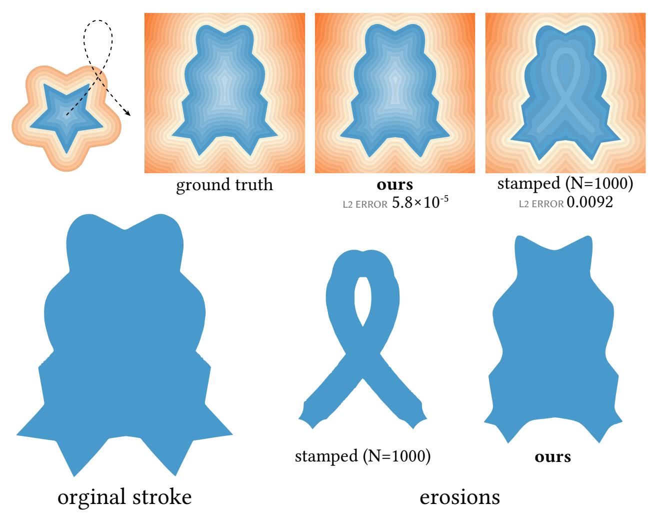


Figure 19: Computing the swept volume of a star shape along a cubic curve (top first) via the naïve union of many stamps leads to non-SDF function (top fourth). With our method we compute an SDF of the sweep (top third) that is much closer to the ground truth (top second), which is reflected quantitatively by the lower L2 error over the shown domain for our result. Having a correct sweep is important for later tasks, for instance taking the the erosion of the stroke (bottom).

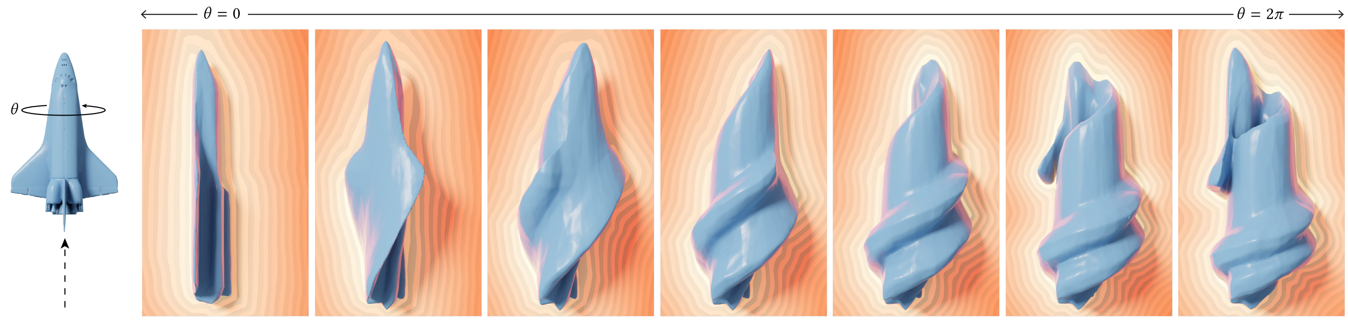


Figure 20: Our method is used to create the swept volume of a space shuttle flying along a fixed linear path. The rotation of the space shuttle around its vertical axis through the path is learned as a one dimensional latent space, sampled here at many points. Model courtesy of NASA.

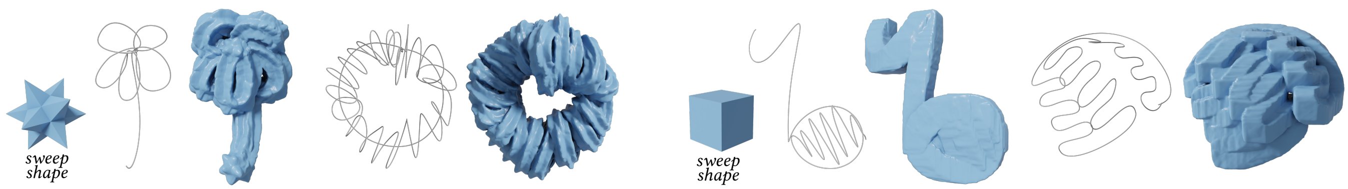


Figure 21: We train a single network to learn the swept volume of a shape along any cubic Bézier path. We show networks for two different sweep shapes here, evaluated on cubic Bézier curves fit to data recorded from sketching in VR. We must train this model only once, and are then able to compute swept volumes simply through a network evaluation.