DOI: 10.1111/cgf.15128
Eurographics Symposium on Geometry Processing 2024
R. Hu and S. Lefebvre
(Guest Editors)

COMPUTER GRAPHICS *forum*
*Volume 43 (2024), Number 5*

# 1-Lipschitz Neural Distance Fields

Guillaume Coiffier[1] and Louis Béthune[2,3]

[1]Université Catholique de Louvain, Belgium
[2]IRIT, Université Paul Sabatier, France
[3]Apple

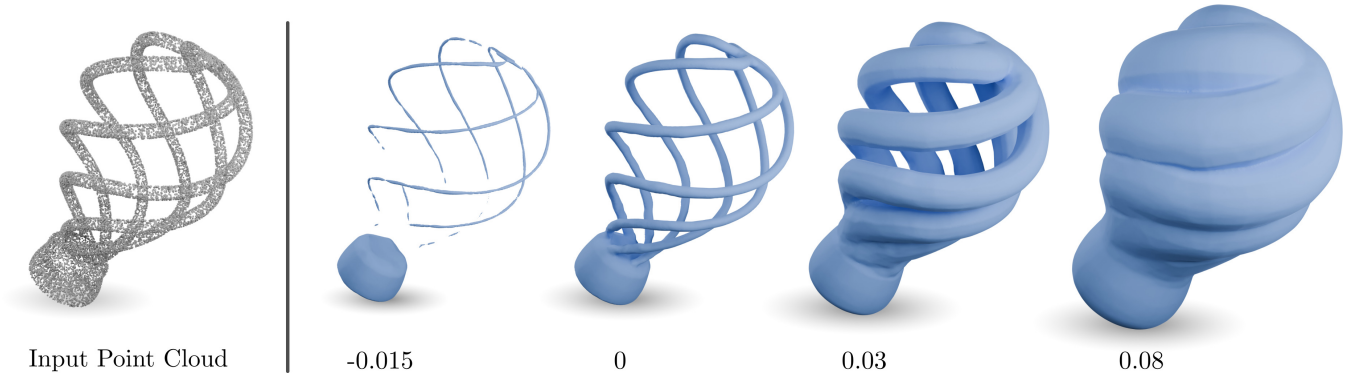



**Figure 1:** *Level sets of a neural distance field trained with our method on a lightbulb model from Thingy10k [ZJ16]. Given an input point cloud with no knowledge of the ground truth distance function, we are able to fit a neural distance field that is close to the real signed distance function while being guaranteed to be* 1*-Lipschitz.*

**Abstract**

*Neural implicit surfaces are a promising tool for geometry processing that represent a solid object as the zero level set of a neural network. Usually trained to approximate a signed distance function of the considered object, these methods exhibit great visual fidelity and quality near the surface, yet their properties tend to degrade with distance, making geometrical queries hard to perform without the help of complex range analysis techniques. Based on recent advancements in Lipschitz neural networks, we introduce a new method for approximating the signed distance function of a given object. As our neural function is made* 1*-Lipschitz by construction, it cannot overestimate the distance, which guarantees robustness even far from the surface. Moreover, the* 1*-Lipschitz constraint allows us to use a different loss function, called the* hinge-Kantorovitch-Rubinstein *loss, which pushes the gradient as close to unit-norm as possible, thus reducing computation costs in iterative queries. As this loss function only needs a rough estimate of occupancy to be optimized, this means that the true distance function need not to be known. We are therefore able to compute neural implicit representations of even bad quality geometry such as noisy point clouds or triangle soups. We demonstrate that our methods is able to approximate the distance function of any closed or open surfaces or curves in the plane or in space, while still allowing sphere tracing or closest point projections to be performed robustly.*

## 1. Introduction

Implicit surfaces [BB97] are a powerful tool for geometric modeling and computer graphics, with direct applications in constructive solid geometry, rendering or surface reconstruction. Unlike explicit representations like point clouds, surface meshes or voxel grids, which rely on a discretization of space, an implicit representation involves defining an object as the zero level set of a continuous function. In the last few years, this idea have received a lot of attention with the introduction of *neural implicit surfaces*, which encode the function as the parameters of a neural network, allowing such representations to be computed for arbitrary input shapes.

Infinitely many implicit functions can correspond to the same geometry, yet not all of them are created equal: properties of the function sometimes need to also be preserved far from the zero level set. A useful implicit representation to consider in these contexts is a *signed distance function* (SDF), which outputs the distance to

the boundary of its underlying object, counted negatively for points that are inside. A SDF has a unit-norm gradient almost everywhere, making it a 1-Lipschitz function. Having a 1-Lipschitz implicit representation is a necessary condition for applications like *ray marching* [Har95], numerical simulation [SS03, SC20] or geometrical queries like surface projection to be performed easily. When computing an approximated SDF, it is indeed crucial to never overestimate the true distance otherwise correctness of the queries cannot be guaranteed. In practice, this means that the function's Lipschitz constant should never exceed 1. However, this Lipschitz property is often overlooked by neural implicit methods, which rather focus on surface fidelity and detail preservation, making them unusable in these contexts without relying on careful range analysis [SJ22].

In this work, we propose a method to approximate the signed distance function of an object by using neural network architectures that are 1-Lipschitz *by construction* [AHD*23], thus guaranteeing correctness of geometrical queries even during training. The Lipschitz constraint of these neural architectures allow us to utilize a different loss function, called the *hinge-Kantorovitch-Rubinstein* (hKR) loss [SMG*21]. This loss has two important effects. Firstly, we prove that any minimizer over all possible Lipschitz functions is close to the SDF of the considered object, which makes our trained neural network a very good approximation of the true distance, as illustrated in Figure 1. Secondly, using the hKR loss makes us approach the problem of learning a signed distance field not from the usual *supervised* regression point of view but from a *semi-supervised* classification point of view: instead of fitting a neural network's output to precomputed distances over a dataset of points, we instead try to maximize the distance between points from inside the shape and points from outside while remaining 1-Lipschitz. This means in particular that the only information required for training is knowing in which category (inside or outside of the input shape) a point is, an information that can be robustly extracted even for point clouds or triangle soups [BDS*18]. As a consequence, we are able to approximate the SDF of an object *without access to the ground truth distance*, enabling training for a wide range of inputs including triangle soups and point clouds, even noisy, sparse or incomplete.

To summarize, our contributions are as follows:

- We apply the known method of minimizing the hKR loss on some 1-Lipschitz neural network to the problem of approximating the signed distance field of an object.
- We demonstrate that such an approach solves the usual robustness issues of similar methods, as it outputs a function that is a good approximation of the true signed distance function while being guaranteed to never overestimate it.
- As the hKR loss does not need ground truth distances but only occupancy labels, we show that we are able to compute signed or unsigned distance fields of noisy, incomplete or sparse representations of objects of any topology, including open surfaces and curves.
- We apply our method to a variety of geometry processing tasks, like surface sampling, medial axis estimation, constructive solid geometry and ray marching.

## 2. Background and Related Work

### 2.1. Signed Distance Function

In all of this work, we will denote by $\Omega$ some solid object $\mathbb{R}^n$, where $n = 2$ or 3. The *signed distance function* (SDF) of $\Omega$ is the function $S_\Omega$ defined over $\mathbb{R}^n$ as:

$$S_\Omega(x) = \left(\mathbb{1}_{\mathbb{R}^n \setminus \Omega}(x) - \mathbb{1}_\Omega(x)\right) \min_{p \in \partial\Omega} ||x - p||$$

where $\partial\Omega$ is the boundary of $\Omega$ and the distance considered in the Euclidean distance.

Signed distance functions have mainly been studied in computer graphics for the ease with which they enable certain operations like boolean composition [Ric73], smooth blending [Bli82], surface offset [FP06] or deformation [SP86] while still allowing an explicit representation, like a surface mesh, to be extracted for instance using the *marching cubes* algorithm [LC87, dLJ*15]. In essence, the SDF value at point $x$ gives two pieces of information: its sign directly tells if the query point is inside or outside the object, while its magnitude gives the radius of the largest sphere centered at $x$ that does not intersect the boundary of the object. This observation is the starting point of the *sphere tracing* algorithm [Har95] which enables direct rendering of SDFs. Additionally, the gradient of the SDF is aligned with the normal vector field of the object on its boundary and gives the direction to the closest point on the boundary. Evaluating the function and its gradient at a point therefore gives a simple strategy for projecting onto the zero level set (Figure 2, left).

### 2.2. Lipschitz Implicit Representations

Although the SDF of an object is easy to compute in closed form for simple shapes (see for instance [Quia] for a list), the same cannot be said of objects found in the wild. Representing those objects via a general implicit surface or even an approximated SDF can still achieve high visual fidelity but comes at a cost. In such contexts, there is indeed no direct strategy for closest point queries or ray intersections: one has to rely to iterative methods, where a key

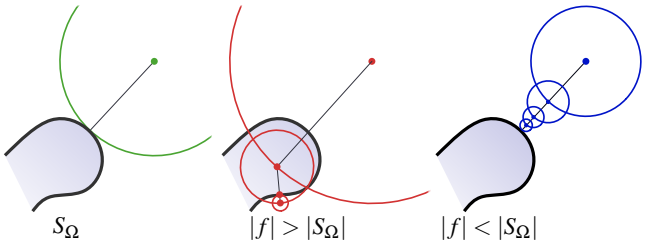



**Figure 2:** *The value of the SDF $S_\Omega$ at point $x$ is the radius of the larger sphere centered at $x$ that does not intersect $\Omega$ (left). For an approximated SDF $f$, if $f$ overestimates the distance (middle), then the query is wrong and no guarantees can be drawn. If $f$ always underestimates the distance (right), iterating the query still converges to the correct result.*

quantity to control is the Lipschitz constant $L$ of the implicit function. Recall that a function $f$ is $L$-Lipschitz if it satisfies:

$$\forall a, b \in \mathbb{R}^n, \quad ||f(b) - f(a)|| \leqslant L||b - a||.$$

If $f$ is differentiable, its Lipschitz constant $L$ is an upper bound on the norm of its gradient. Since the signed distance function has a unit-norm gradient, an implicit function with $L > 1$ may overestimate the true distance. As a consequence, the sphere centered at $x$ of radius $f(x)$ may intersect the surface and yield a false negative (Figure 2, middle). No guarantee can be drawn onto the correctness of the query in this case. On the other hand, having $L < 1$ implies that the function always underestimates the distance. Iterating the empty sphere query in this case will converge to the correct result at a cost of a greater computation time (Figure 2, right).

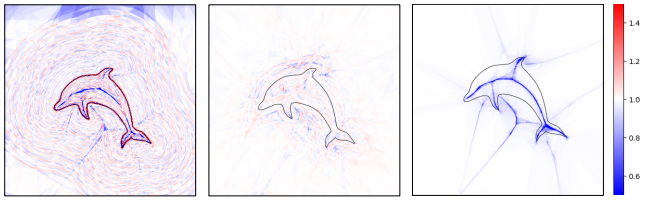
An extensive bibliography exists on algorithms for performing robust geometric queries on approximate SDFs. Previous works can be organized into two categories. Methods from the first one rely on interval arithmetic and careful range analysis to detect overshooting and false negatives [Duf92, SJ22, AZ23], thus guaranteeing robustness at the cost of more complexity. Methods from the second category instead estimate the Lipschitz constant of the implicit function and apply a local or global rescaling [KB89, GGPP20] to query $f/L$ instead. However, computing the exact Lipschitz function for neural networks is a NP-hard problem [JD20] and finding a good approximation of it remains tricky [VS18].

### 2.3. Neural Implicit Surfaces

Approximating a distance function has historically been achieved using basis functions like blobs or blended balls [Bli82, WMW86]. In the last few years, an ongoing trend has proposed to encode it into the parameters $\theta$ of a multilayer perceptron (MLP) $f_\theta$. This idea of a neural field has sparked many applications in computer graphics and learning, which are not restricted to distance fields. We refer to Xie et al. [XTS*22] for a survey.

Perhaps the most simple neural implicit representation is to represent the shape as a binary occupancy field, predicting 1 for points inside the shape and 0 otherwise [CZ19]. This can be seen as a binary classification problem and treated as such [MON*19]. While enabling total surface reconstruction, such neural fields give no information far from the surface and are therefore hard to query geometrically.

The *DeepSDF* [PFS*19] algorithm is the first proposition of a neural SDF. It is setup as a single large neural network optimized over a collection of objects, where a given object is represented via a latent vector fed as an input along the query point. This popular setup allows shape interpolation [LWJ*22], classification as well as shape segmentation [PGMK23]. In contrast, training one network per object has also been performed [DNJ21] for shape compression purposes. Learning an *unsigned* distance field has also been attempted either directly [CmP20] or using a sign-agnostic loss [AL20a], thus extending the application of neural distance fields to open surfaces and curves. All of these methods are *supervised*, meaning that they are optimized to make the network's predictions match some pre-computed signed distances. Unfortunately, the minimization of this loss alone does not guarantee that the network will correctly extrapolate the distance for points not in the dataset, especially if it is not dense enough. In particular, the resulting function may present a gradient whose norm may vary extensively and thus a large Lipschitz constant. This phenomenon is illustrated on a simple 2D dataset on Figure 3 (a).




$\max ||\nabla f|| = 13.2$ $\max ||\nabla f|| = 1.43$ $\max ||\nabla f|| = 0.998$

**(a)** Classical MLP **(b)** MLP with eikonal regularization **(c)** Lipschitz neural network

**Figure 3:** *Plot of a the gradient norm of neural distance fields on a simple 2D dolphin silhouette. While minimizing the eikonal loss stabilizes the gradient norm, only a Lipschitz network guarantees a unit bound.*

In order to bring the Lipschitz constant closer to 1, many works rely on regularization losses applied to the gradient of the network. The most widespread of such regularization is the *eikonal* loss, which computes how much the norm of the gradient differs from 1. Initially introduced to improve the stability of the *Wasserstein Generative Adversarial Networks* [GAA*17], it was then naturally adapted to neural distance fields [GYH*20, YGKL21]. An alternative to the eikonal loss is the total variation loss [CD23], which minimizes variations of the gradient norm and improves the quality of the neural implicit function far from the zero level set. Alignment losses, that penalize the difference between the gradient and some normal vector field, improving visual fidelity over the zero level set, have also been considered [AL20b, SMB*20], often alongside some eikonal term.

Yet, while these regularizing losses have a clear impact on the gradient of the considered implicit function, their minimization is performed in practice using first-order optimizers like *Adam* [KB14], which means that their global minimizer is never reached. Even though specific neural architectures have been designed to better behave during optimization, like *SIREN* [SMB*20] that uses sine activations, nothing prevents a regularized network to have a Lipschitz constant larger than 1 even after training for a long period of time, as shown in Figure 3 (b).

Few neural implicit methods tackle the case where the true signed distance to the object is not known or cannot be computed. In this harder context, Lipman [Lip21] defines the *PHASE* loss, allowing to learn an occupancy field using only a representation of the boundary of the object. The signed distance function is then retrieved using a log transformation. Finally, while their application is classification and outlier detection, the method of Béthune et al. [BNB*23] learns a SDF from an occupancy field by also minimizing the *hinge-Kantorovitch-Rubinstein* loss. Although very similar, our method is simpler as their Newton-Raphson update of the

distribution is unnecessary in our case, as well as faster since we use a more computationally efficient neural architecture.

### 2.4. Lipschitz Neural Networks

At its core, a neural network is nothing more than a function $f_\theta$ where the parameters (or weights) $\theta \in \mathbb{R}^K$ are arranged in a predetermined pattern. Specifying such an architecture defines a functional space:

$$\mathcal{F} = \{f_\theta \,|\, \theta \in \mathbb{R}^K\}$$

over which learning algorithms optimize the weights to find the function $f_\theta^*$ that minimizes some user-defined loss criterion. Knowing exactly the extent of the functional space $\mathcal{F}$ for a fixed architecture is an open problem in deep learning, but recent works have managed to define some $\mathcal{F}$ as a subset of $L$-Lipschitz functions [SMG*21].

The investigation of Lipschitz architecture in deep learning is primarily motivated by the robustness of such networks against adversarial attacks and overfitting. Early attempts focused on regularizing the weight matrices by controlling their largest singular value [YM17]. To prevent the gradient from vanishing, other efforts focused on having singular values all close to one, namely regularizing weight matrices to be orthogonal [CBG*17, TK20] and the network to be gradient preserving. Since this hindered expressiveness overall when using usual component-wise activation functions like the sigmoid or ReLU, Anil et al. [ALG19] propose a sort as a non-linearity. Lispchitz network have since been shown to have comparable results with classical neural networks on a variety of tasks [BBS*22].

More recently, other constructions of Lipschitz neural layers decreasing the computational cost of previous approaches have been proposed. Instead of relying on iterative projections of weight matrices to orthogonal ones, Prach and Lampert [PL22] introduce an almost orthogonal layer where singular values of the matrices are updated directly during training. Other Lipschitz layers have then been defined, such as the *Convex Potential Layer* [MDAA22] or the *Semi-definite Programming Lipschitz Layer* (SLL) [AHD*23]. Neural architectures used in this work are based on the latter.

## 3. Robust Learning of a Signed Distance Function

As current neural implicit representations cannot provide guarantees on geometrical queries from the implicit function nor theoretical bounds on its Lipschitz constant, we propose to directly integrate the constraint of being 1-Lipschitz directly into the neural architecture. As shown experimentally in Figure 3 (c), this will result in an implicit function that cannot overestimate the true distance by construction, even during training.

Our method takes as an input any curve or surface $\partial\Omega$ from $\mathbb{R}^n$, represented either by a point cloud with normals or a triangle soup. The first step is to define some 1-Lipschitz neural architecture, for which we use the *SLL* architecture of Araujo et al. [AHD*23] (Section 3.1). As having a Lipschitz constant strictly smaller than 1 can induce greater computation times for geometrical queries to converge, it is desired to not only be 1-Lipschitz but to have gradient as close as possible to unit norm everywhere. In our case, this is achieved by the *hinge-Kantorovitch-Rubinstein* (hKR) loss whose indirect effect is to maximize the gradient's norm (Section 3.2). Minimizing the hKR loss requires to partition a dataset of points around the object as points inside and outside. In the case of closed surfaces or curves, we use the *generalized winding number* [BDS*18] to robustly compute this partition (Section 3.3). Finally, the case of open surfaces and curves, for which we want to compute an unsigned distance function, will be discussed in Section 3.4. The different steps of our method are illustrated on Figure 4 on a corrupted point cloud of the *Botijo* model.

### 3.1. 1-Lipschitz Neural Architecture

Classically, a neural network $f_\theta$ has its parameters $\theta$ arranged in a series of layers $f^1, ..., f^l$ so that the final function is the composition of all layers in order. As the Lipschitz constant of a composition is upper bounded by the product of all Lipschitz constants, designing a 1-Lipschitz architectures boils down to defining some 1-Lipschitz layers to be chained together. To this end, Araujo et al. [AHD*23] propose the *Semi-definite Programming Lipschitz Layer* (SLL). Using a square matrix $W \in \mathbb{R}^{k\times k}$, a bias vector $b \in \mathbb{R}^k$ and an additional vector $q \in \mathbb{R}^k$ as parameters, it is defined as:

$$x \mapsto x - 2WT^{-1}\sigma(W^T x + b) \tag{1}$$

where $T$ is a diagonal matrix of size $\mathbb{R}^{k\times k}$:

$$T_{ii} = \sum_{j=1}^{k} \left|(W^T W)_{ij} \exp(q_j - q_i)\right|$$

and $\sigma(x) = \max(0, x)$ is the rectified linear unit (ReLU) function. In comparison to the classical multilayer perceptron layer $x \mapsto \sigma(W^T x + b)$, the SLL layer only adds a small amount of parameters in the form of the vector $q$ and a $\mathcal{O}(k^2)$ operations, which makes it more efficient than previous Lipschitz architectures.

The SLL function is a residual layer, meaning that its computation is added to its input. As a consequence, the layer can only be defined for matching input and output dimensions. In our case, the input of the network is a point in $\mathbb{R}^n$ with $n = 2$ or 3 and its output is a single real number. The input of the network is therefore first padded with zeros to match the size $k$ of the SLL layers. To retrieve a single number as output, the network ends with an affine layer defined as:

$$x \mapsto \frac{w^T x}{||w||_2} + b$$

where $w \in \mathbb{R}^k$ and $b \in \mathbb{R}$. Dividing by the euclidean norm of $w$ in the computation ensures that the operation is 1-Lipschitz.

### 3.2. The hinge-Kantorovitch-Rubinstein Loss Function

Given a dataset $(X, Y)$ of points with associated signed distance ground truth, a straightforward approach to learn a neural signed distance field guaranteed to always underestimate the true distance would be to minimize a fitting loss over a 1-Lipschitz architecture as defined above. While it solves the problem of robustness of neural signed distance fields, this approach fails short of our goal for

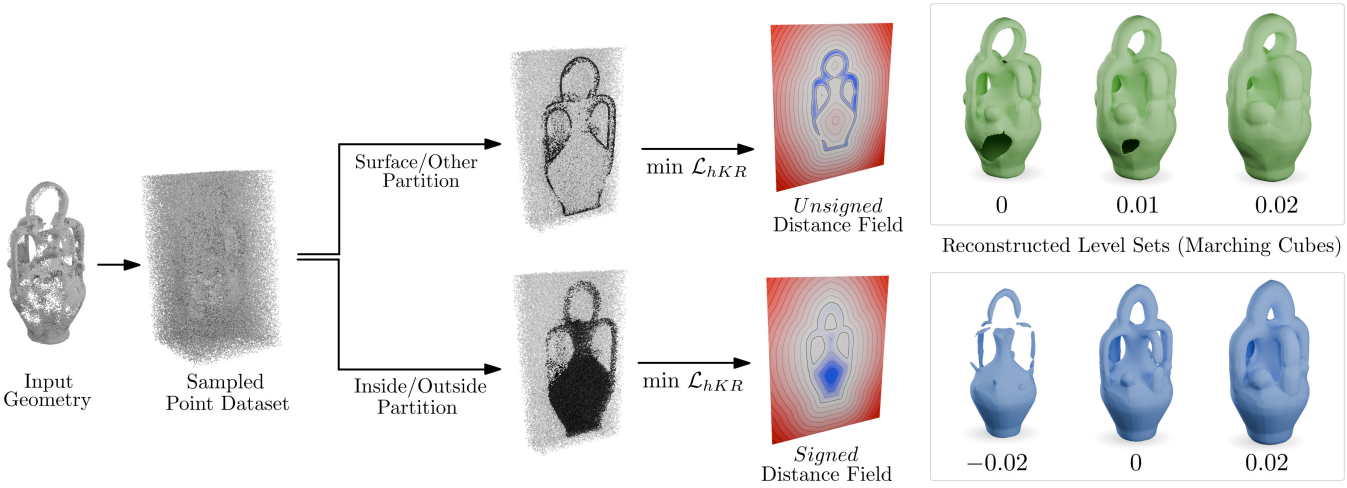



**Figure 4:** *Overview of our method on a corrupted* Botijo *dataset. Given an input geometry in the form of an oriented point cloud or a triangle soup, we uniformly sample points in a domain containing the desired geometry. Defining negative samples as points of the geometry and positive samples everywhere else yields an unsigned distance field when minimizing the hKR loss. On the other hand, partitioning samples as inside or outside the shape leads to an approximation of the signed distance function of the object.*

two reasons. Firstly, this means that the input geometry must be well known for the initial dataset to be computed with exact signed distances, making it unsuitable for only raw point clouds or triangle soups as inputs. Secondly, the function will indeed be 1-Lipschitz but can greatly underestimate the true distance. Ideally, we would like to also maximize its gradient norm so that fewer iterations are needed in geometrical queries.

These two limitations can be overcome with the *hinge-Kantorovitch-Rubinstein* (hKR) loss. Introduced by Serrurier et al. [SMG*21], the hKR is a binary classification loss that can be optimized over some $L$-Lipschitz architecture. Let $D \subset \mathbb{R}^n$ be a compact domain over which binary labels $y(x) \in \{-1,1\}$ are defined at each points. Let $\lambda > 0$ and $m > 0$ be hyperparameters. Let $\rho(x)$ be some probability distribution function over $D$, which can be thought of as an importance weight. The hKR loss of a neural function $f_\theta$ is the sum of two terms:

$$\mathcal{L}_{hKR} = \mathcal{L}_{KR} + \lambda \mathcal{L}^m_{hinge} \tag{2}$$

defined as:

$$\mathcal{L}_{KR}(f_\theta, y) = \int_D -y(x) f_\theta(x)\, \rho(x) dx \tag{3}$$

$$\mathcal{L}^m_{hinge}(f_\theta, y) = \int_D \max(0, m - y(x) f_\theta(x))\, \rho(x) dx. \tag{4}$$

The first term (Equation (3)), minimized over all possible 1-Lipschitz functions, is known in the literature as the dual formulation of the Wasserstein-1 distance [ACB17], as given by the Kantorovitch-Rubinstein duality theorem [V*09, Theorem 5.10]. While its connections to optimal transport are out of scope of this work, it can be interpreted as maximizing values of $f_\theta$ for points where $y = 1$ and minimizing them when $y = -1$. It therefore encourages the function to maximize its rate of change, and thus its Lipschitz constant. However, as observed by Serrurier et al. [SMG*21], simply optimizing $\mathcal{L}_{KR}$ leads to bad results as the zero level set of the network does not capture the boundary between positive and negative $y$. This is where the second term comes into play: the hinge loss of Equation (4) penalizes points for which the sign of $f_\theta$ and $y$ are different, that is to say points that are "misclassified" by $f_\theta$. The parameter $m > 0$, called *margin*, defines an error threshold under which this misclassification is ignored. Low values of $m$ lead to more precise results at the interface but also unstability in the optimization.

In the case of the neural distance field of an object $\Omega$, we can apply the hKR loss by considering $y$ as an occupancy label, being $-1$ for points inside of $\Omega$ and $+1$ otherwise. In this context, under mild assumptions over $\rho$, minimizers of the hKR loss are good approximations of the signed distance function of $\Omega$. This is summarized as the following theorem:

**Theorem 1** *Let $D, \Omega$ be compact subsets of $\mathbb{R}^n$ such that $\Omega \subset D$. Let $y$ be binary labels defined over $D$ as:*

$$y(x) = \mathbb{1}_{D \setminus \Omega}(x) - \mathbb{1}_{\Omega}(x).$$

*Let $m > 0$ and assume that $\rho(x) = 0$ whenever $|S_\Omega(x)| \leqslant m$ and $\rho(x) > 0$ otherwise.*
*Let $f^*$ be a minimizer of $\mathcal{L}_{KR}(f, y)$ under constraint that $\mathcal{L}^m_{hinge}(f, y) = 0$, where the minimum is taken over all possible 1-Lipschitz functions. Then:*

$$\forall x \in D, \quad \begin{cases} |S_\Omega(x)| > m & \implies \quad f^*(x) = S_\Omega(x) \\ |S_\Omega(x)| \leqslant m & \implies \quad |f^*(x) - S_\Omega(x)| \leqslant 2m \end{cases}.$$

This theorem is very similar to the result of Béthune et al. [BNB*23, Theorem 1], which shows a similar result in a context where $\rho(x)$ is zero whenever $0 \leqslant S_\Omega(x) \leqslant 2m$. This translated version is relevant to their application of outlier detection; in our case however, we focus on the "symmetric" version. A fully detailed proof is available in Appendix A.

The key idea of the proof is to split the domain $D$ into two parts: the region where $\rho = 0$, which corresponds to a "shell" set of width $2m$ centered on $\partial\Omega$, and its complementary. This allows to exploit the fact that $\mathcal{L}^m_{hinge}(f^*, y) = 0$ and deduce the approximation by at most $2m$ in this region. Outside of this region, one can show that $f^*$ has the same sign as $S_\Omega$, and its Lipschitz property means that it is bounded by $|S_\Omega|$. Since $f^*$ minimizes the $\mathcal{L}_{KR}$, one can show that this bound is in fact tight, hence the equality.

Minimizing $\mathcal{L}_{hKR}$ over 1-Lipschitz functions therefore approximates the signed distance function of the considered object. The critical quantity to control here is the margin parameter $m$, which can be thought as the minimal quantity to be imposed between points of different labels. As such, smaller margins allow the minimizer to better approximate the SDF $S_\Omega$. However, in the context of 1-Lipschitz neural networks, the *fat-shattering* dimension of the resulting function class has been shown to increase as $(\frac{1}{m})^n$ [BBS*22], which means that optimizing for smaller margins requires significantly more parameters in the network and leads to less stable results in practice (see for instance Figure 6).

The constraint on $\rho$, that is the exclusion of a shell of width $2m$ centered around the interface, means that the training dataset should ideally not contain any sample at distance smaller than $m$ from $\partial\Omega$. In practice, we do not explicitly prevent this from happening. As noticed by Béthune et al. [BNB*23], this introduces an additional error of the order of $m$. Finally, as far as the other parameter $\lambda$ is concerned, the hinge loss being a hard constraint in the theorem implies that $\lambda$ should be large enough in practical optimization.

Note that the minimization of the hKR loss is *semi-supervised*: it is only necessary to know if a given point has been sampled from $D\backslash\Omega$ or from $D$ to learn a SDF and the ground truth $S_\Omega$ does not need to be known. This effectively turns the classical regression task of fitting a neural distance field into a classification task between inside and outside points. With this strategy, all we are left to do is to generate some dataset of points with associated binary $y$ labels to train a Lipschitz network, which boils down to determining if a given point $x \in \mathbb{R}^n$ belongs to $\Omega$ or not.

### 3.3. Inside/Outside Partitionning

Let us first consider the case where the input $\partial\Omega$ represents a closed curve in the plane or a closed surface. If the input geometry is clean enough, like for instance a manifold surface mesh, determining its inside from its outside is a task that can be solved in a robust way [SSS74]. When the geometry presents holes, defects, or is simply made of points, a notion of "insideness" can still be recovered by computing the *generalized winding number* [JKS13]. Intuitively, the winding number $w_\Omega$ of a surface $\partial\Omega$ at point $x$ is the sum of signed solid angles between $x$ and surface patches on $\partial\Omega$. For a closed smooth manifold, the values amounts at how many times the surface "winds around" $x$, yielding an integer value. When computed on imperfect geometries, $w_\Omega$ becomes a continuous function (see Figure 5). Through careful thresholding, it is still possible to determine points that are inside or outside the shape with high confidence.

Going back to our problem of learning a signed distance function from surface data $\partial\Omega$, we first sample points $X$ uniformly in-

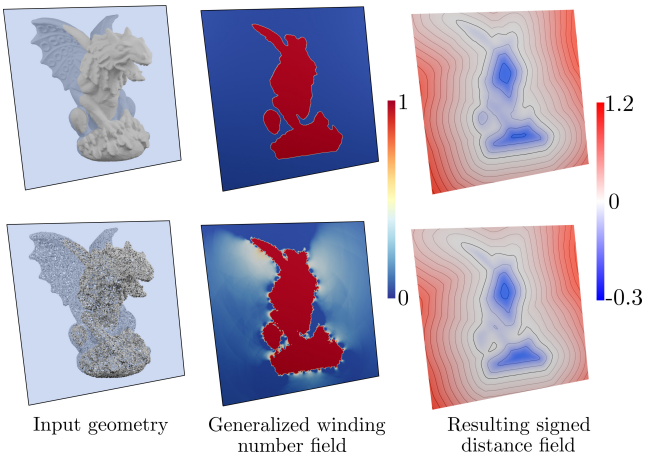



**Figure 5:** *Generalized winding number field computed for the* Gargoyle *model on the original manifold mesh (top) and a point cloud of 50K points(bottom). Thresholding this field allows to partition a dataset of points into inside and outside of the shape, from which a neural signed distance field can be optimized by minimizing the hKR loss (right column).*

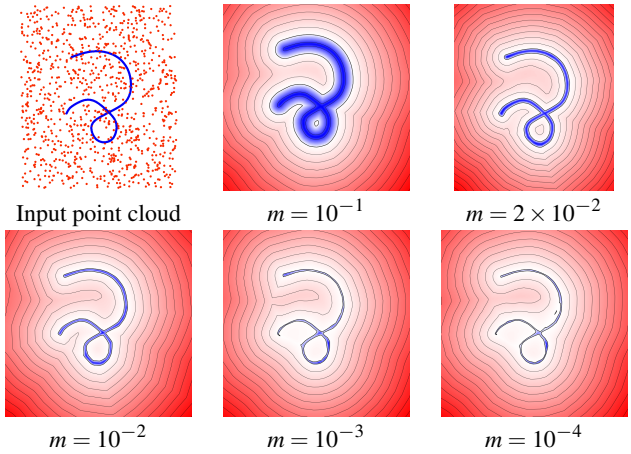



**Figure 6:** *Neural distance field of an open curve in 2D. Minimizing $\mathcal{L}_{hKR}$ on this dataset yields a distance field to the curve up to the margin parameter m. Large m create large but consistent underestimation of the true distance while smaller m lead to instabilities in training and final result. A value of m around $10^{-2}$ is a good trade-off in practice.*

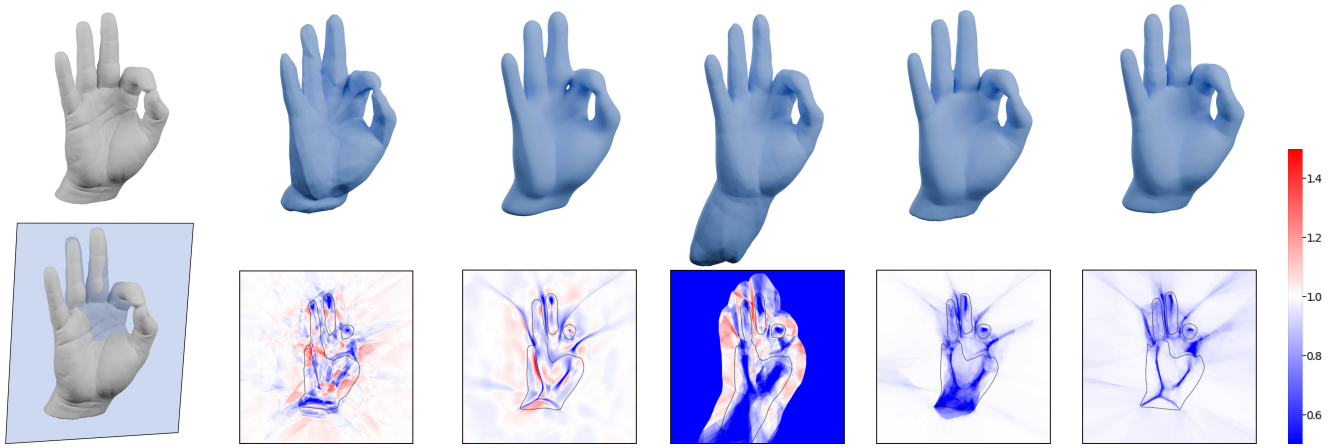



**Figure 7:** *Comparison of various neural distance field methods on a hand dataset. Top row: reconstructed zero level set using the marching cube algorithm [LC87]. Bottom row: norm of the gradient along a cut. While their reconstruction of the zero level set is comparable, previous methods all present a gradient norm that either exceeds one or vanishes far from the surface.*

side a loose bounding box $D$ of $\Omega$, and compute their generalized winding number using the method of Barill et al. [BDS*18]. Then, two distributions $X_{in}$ and $X_{out}$ are extracted from $X$ by choosing $N$ points from $X$ with winding number smaller than $\tau_i$ and greater than $\tau_o$ respectively, where $\tau_o < \tau_i$ are threshold parameters. This ensures that the network will be trained using the same amount of interior and exterior points. In practice, $N$ varies from $10^4$ to $10^6$ points, depending on the amount of details needed to be captured. Points from $X_{out}$ are assigned a label $y = 1$ and points from $X_{in}$ a label $y = -1$, before being fed in a Lipschitz neural network that minimizes the hKR loss (Equation (2)). As shown on the right of Figure 5, this process effectively allows to recover a signed distance function of the original shape.

### 3.4. Distance field of shapes without interior

If the input object $\Omega$ has no interior, like the case of a curve or an open surface, we can still learn a distance function to $\Omega$ by minimizing the hKR loss without relying on the generalized winding number. We simply sample $X_{in}$ as being points on the surface or curve, either by taking directly the point cloud as input or sampling on triangles. $X_{out}$ is then a uniform distribution over the domain $D$. Up to the margin parameter $m$ in the loss, this still results in an approximation of the distance function of the considered manifold. But since level sets of the neural function can only be closed surfaces, the optimization results in an object having a "thickness". This property is directly controlled by the margin parameter $m$ in the loss, as shown in Figure 6: a larger margin leads to reconstruction of the data with a non-negligible thickness, whereas a margin too small leads to instabilities on the final result. Figure 12 further demonstrates the SDF reconstruction ability of our method in three dimensions in the context of open surfaces or curves. Note that this setup is also suitable for closed surfaces in order to learn an unsigned distance field, as it is the case in Figure 4.

## 4. Results and Applications

### 4.1. Implementation details

We implement our method in python using the *Pytorch* library for neural network training. All our experiments were performed on a Ubuntu 22 workstation using a Nvidia 4070Ti GPU. For the generalized winding number, we use the original implementation of Barill et al. [BDS*18] as provided in *libigl* [JP*18].

We use a neural network architecture of 20 SLL layers of size $k = 128$. In total, this amounts for 330K floating point parameters for a total size of approximately 1.4MB. For reference, the hand mesh used in Figure 7 is described by 150K floating point numbers for vertices and 300K integers for faces, for a total size of 2.2MB when compressed.

All inputs are normalized in a bounding box $[-\frac{1}{2};\frac{1}{2}]^n$ before any processing. We set $D = [-1,1]^n$ as our sampling domain. With the exception of Figure 6, all our experiments were performed with $m = 10^{-2}$ and $\lambda = 100$.

### 4.2. Gradient Robustness

As a first experiment, we demonstrate the robustness of our approach in comparison with neural distance field methods from the state of the art. These methods are trained on a dataset of points $(x_i, S_\Omega(x_i))$ with $S_\Omega$ being precomputed from the input mesh. They minimize a least-square fitting loss:

$$\mathcal{L}_{fit} = \sum_i [f_\theta(x_i) - S_\Omega(x_i)]^2 \tag{5}$$

to match the true distance function, as well as an eikonal loss:

$$\mathcal{L}_{eikonal} = \int_D (||\nabla f_\theta(x)|| - 1)^2 \, dx. \tag{6}$$

that regularizes their Lipschitz constant. For a fair comparison, the architecture reported by the original works are scaled to match our number of parameters.

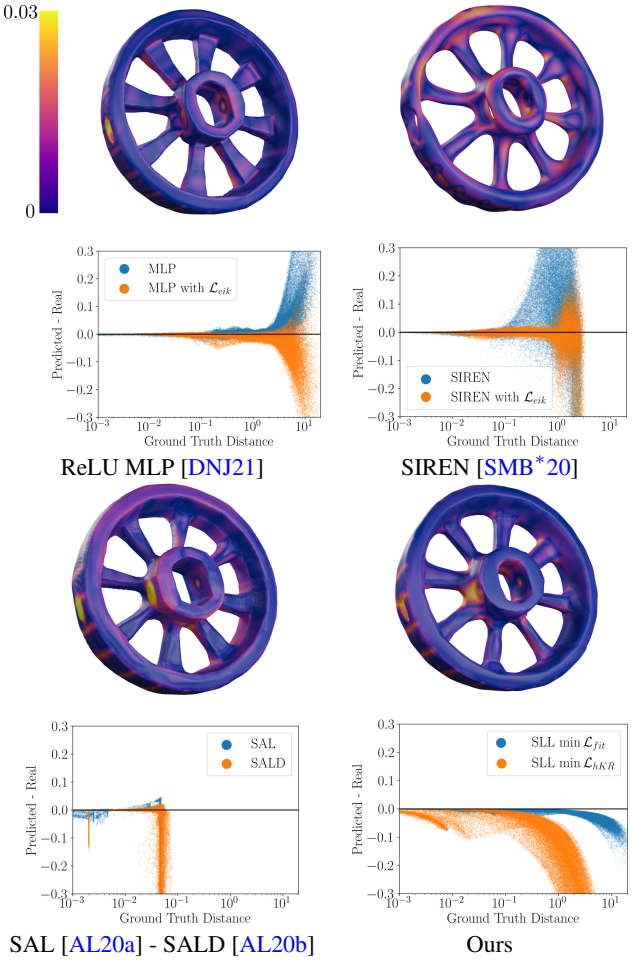



**Figure 8:** *Reconstruction experiment on a handle mesh. We plot the zero level set of each neural network with a colormap indicating the distance from the ground truth's surface. Our results around the zero level set are comparable to other methods. Moreover, we evaluate the difference $f_\theta(x) - S(x)$ for 100K points sampled at random and plot the results. As expected, this difference is always negative for our method, indicating that we always underestimate the true distance.*

On Figure 7, we reconstruct the zero level set of the neural function using marching cubes and plot its gradient norm over some planar cut. We observe that the classical multilayer perceptron (MLP) with ReLU activations and SIREN [SMB*20], which is a MLP with sine activations, are able to capture the surface and the topology of the input, yet their gradient is unstable and often exceeds unit-norm. SALD [AL20b] is a method that fits a neural implicit surface without requiring knowledge of the sign of its distance function. However, the trained network has a null gradient far from the surface. As far as our method is concerned, we observe that when $\mathcal{L}_{fit}$ (Equation (5)) is minimized over a Lipschitz architecture, the zero level set represents the input shape more accurately than with the hKR loss. This behavior is expected since the network has access

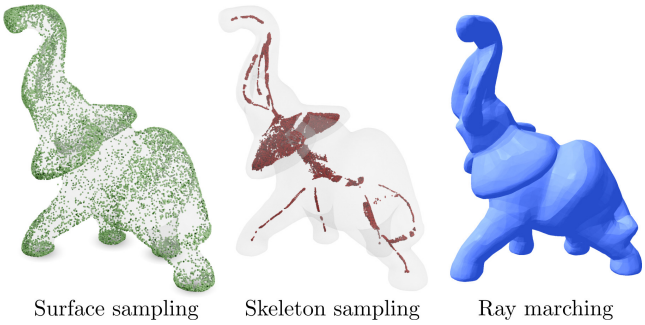



**Figure 9:** *Geometrical queries performed on the* elephant *model. Queries are all valid without relying on range analysis.*

to more information in the dataset, and also because of the slight error introduced by the margin $m$ (as discussed in Section 3.2).

### 4.3. Signed vs Unsigned Distance Field

As our method provides stable distance estimation far from the zero level set, we are also able to extract high quality isosurfaces for different values of the function. This is illustrated on Figure 1 for the signed case (using the generalized winding number) and on Figure 12 in the unsigned case, for a curve and an open surface with holes. Additional results are available in supplemental material.

While fitting an unsigned distance also works perfectly fine for a closed object, partitioning the points first with generalized winding number enables our method to benefit from its robustness to faulty and noisy input. As a result, some defects of the zero level set can be repaired, as it is the case with the corrupted *botijo* model shown in Figure 4.

### 4.4. Underestimation of the true distance field

To further justify our claim that our learned signed distance field never overestimates the true distance, we train different neural distance fields on a *handle* model (Figure 8). We then extract the zero level set of each method using marching cubes [LC87] and sample 100K points uniformly in a sphere of radius 30 centered at zero. We plot the difference between the true distance $S(x)$ to the zero level set mesh and the predicted value of the neural network, in function of $S(x)$. In this experiment, a perfect network would output a straight line, but all methods present some deviation in practice. However, we observe that only our method provide only negative values, meaning that the network's output is always smaller than the true distance.

### 4.5. Geometrical Queries

Having a robust SDF far from the surface enables efficient and robust geometrical queries on any of its level sets. In this section, we demonstrate experimentally that it is indeed possible to do on our Lipschitz network trained with the hKR loss, without requiring neither range analysis nor an estimation of the Lipschitz constant.

Given an exact SDF $S$ and a point $x$ in space, the closest

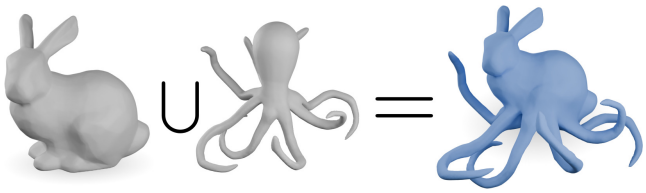


**Figure 10:** *Constructive Solid Geometry. The union of two SDFs can be computed using the* min *operator. Although not necessarily a true SDF, the resulting function remains* 1*-Lipschitz.*

point of $x$ on the zero level set of $S$ can be directly computed as $x - \nabla S(x)S(x)$. This expression falls short of the level set in the case where $||\nabla f|| < 1$ but can be iteratively evaluated until a convergence criterion is met. We illustrate this process on Figure 9 (left), where points on a sphere of radius 2 are projected onto the surface of the *elephant* dataset. As points tend to accumulate in salient regions of the model, a more uniform sampling can be recovered with the Iso-points method [YWÖS21].

Another application of neural distance fields is to sample the medial axis of an object. As the medial axis is defined as the set of points where the closest point on the surface is not unique, it corresponds to discontinuities of the gradient of the SDF. In our case, the Lipschitz network is fully differentiable and approximates the discontinuity by dropping the gradient's norm, as illustrated in Figure 3 (right). This gives us a sample-and-reject strategy to construct the model's skeleton by evaluating the norm of the gradient. A mesh of the medial axis can then be extracted by the method of Clemot and Digne [CD23], whose skeleton sampling results we reproduce (Figure 9, middle) without any gradient regularization loss.

Finally, *ray marching* queries the implicit function along a ray to determine the maximum step size that guarantees non-intersection. Without our 1-Lipschitz network, we are able to take a step size of 1 and recover a correct image (Figure 9, right).

### 4.6. Constructive Solid Geometry

A key feature of implicit geometries is the ease with which we can apply boolean operations on their function to represent intersections, unions or differences of shapes [Ric73], thus building more complex shapes for simple ones. We illustrate this property on our neural distance field on Figure 10: given two SDFs $f_1$ and $f_2$, we can compute the surface of their union as the 0-level set of $\min(f_1, f_2)$. Similarly, their intersection coincides with the 0-level set of $\max(f_1, f_2)$. These operations preserve the Lipschitz constant so the guarantee of being 1-Lipschitz still holds for the composition. However these point-wise minimum and maximum are not necessarily SDFs themselves [Quib, MSLJ23].

### 4.7. Robustness to noise and sparsity of input

Finally, approaching the neural distance field problem with the hKR loss means that we are still able to recover a good approximation of the SDF of an object in contexts where the ground truth distance cannot be computed or is not available. We illustrate this on the *fertility* dataset in Figure 11 on four different settings. The first one is generated with *Blensor* [GKUP11] and emulates a noisy Lidar acquisition of 7K points. The second one consists of 5K points where 50 points and their closest 40 neighbors have been removed from the point cloud, thus creating holes in the shape. The third one pushes sparsity to its maximum with only 500 points in total. Finally, the fourth one adds a Gaussian noise of standard variation 0.03. In all four settings, the winding number still allows us to sample 10K inside and outside points. Minimizing the hKR loss then yields the correct topology on the zero level set, albeit some expected geometrical error. This also means that our method can be used as a noise reduction or an outlier elimination technique, a context where the hKR loss has already been used with success [BNB*23].

## 5. Discussion and conclusion

Neural distance fields are a promising technique for representing arbitrary geometry without relying on a discretization of space. While previous methods have demonstrated outstanding results in surface reconstruction and visual fidelity, we demonstrated that these neural functions could also be made 1-Lipschitz and support geometrical queries without the need of range analysis. Having this 1-Lipschitz constraint also allowed to learn a distance field without needing any distance information from the input shape, thus enabling neural fields applications for a broader set of geometry representations, even including bad quality point clouds or triangle soups.

Yet, this built-in robustness did not come without drawbacks. The hKR loss can only be successfully minimized up to a margin $m > 0$, which prevents a perfectly accurate surface reconstruction. Smaller margin allows for more marginally more fine-grained details to be captured, but the resulting functions are usually biased towards low frequencies. Some recent neural methods solve this problem by using positional encoding at the start of the network [Lip21], like the *Fourier features* [TSM*20]. A network using such an encoding could still be made 1-Lipschitz: as the Lipschitz constant $K$ of the embedding can be known in closed form (it depends on the frequencies of the harmonic functions), it suffices to build a $1/K$-Lipschitz network by scaling the output. Yet, as the embedding does not preserve the gradient norm, so will the hKR-trained final network, which could lead to significant underestimation of the true distance. Improving reconstruction quality while being almost gradient preserving thus remains a challenge for future works.

Overall, for a fixed number of parameters, there seems to be a clear trade-off between quality of the zero level set and quality of the gradient far from it, an observation that has already been made in classification contexts regarding 1-Lipschitz networks [BBS*22]. For a network of bounded depth, the zero level set is made of affine pieces whose maximum number grows as a polynomial of the width [Tel16, Yar18, BHLM19, PKH23], which would require a similar increase in numbers of parameters.

Finally, as Lipschitz networks are computationally heavier than their classical counterpart, future works could also focus on improving speed of convergence of the method, for example by reducing the total number of points needed in the dataset. This could

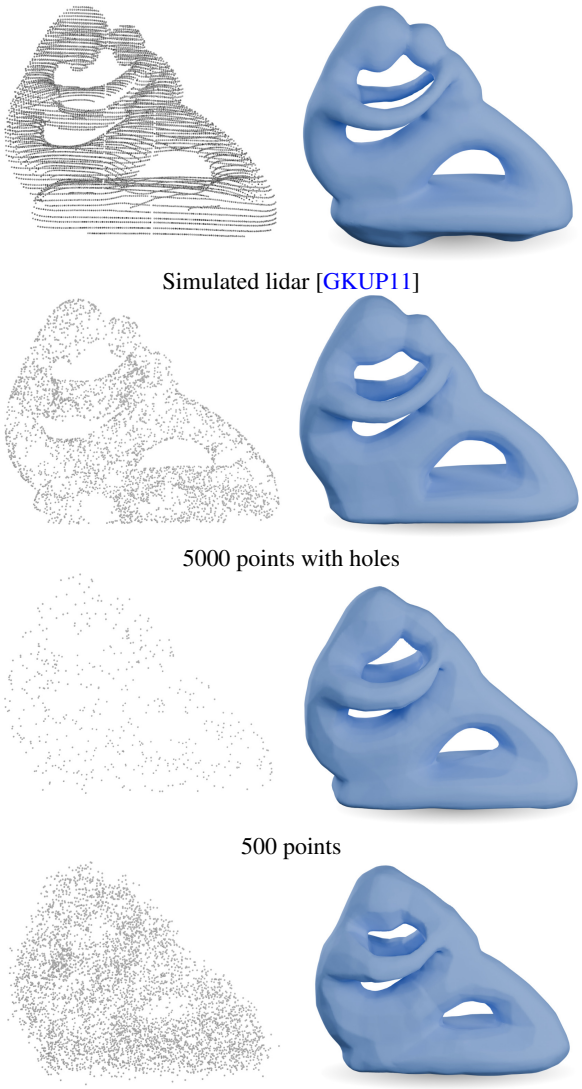



**Figure 11:** *Neural SDF reconstruction of the* fertility *model in contexts where the ground truth is not available. Our method is able to recover the correct topology of the input surface even in very hard settings.*

be achieved by importance sampling before training, or by eliminating unnecessary points during training.

**Appendix A:** Proof of Theorem 1

Let $f^*$ be a 1-Lipschitz minimizer of $\mathcal{L}_{KR}$, under the constraint that $\mathcal{L}_{hinge}(f^*, y) = 0$. $f^*$ is shown to exist by Serrurier et al. [SMG*21, Theorem 1]. Define:

$$\partial\Omega^m = \{x \in D, |S_\Omega(x)| < m\}$$

as the "shell" of width $2m$ centered around $\partial\Omega$. Given the assumption on $\rho$, it exactly corresponds to the points that are not involved in the computation of the hinge loss. Then, the hinge loss being 0 for $f^*$ implies that:

$$m - y f^*(x) < 0 \text{ for all } x \in D \backslash \partial\Omega^m.$$

In other words, for any point with distance larger than $m$ from the boundary $\partial\Omega$, $f^*$ has the same sign as $y$ and $|f^*(x)| \geqslant m$. Note that it would not be possible to have zero hinge loss if $\rho$ was not zero in $\partial\Omega^m$.

Let us first focus on what happens inside $\partial\Omega^m$. Since $f^*$ is 1-Lipschitz, we have $|f^*(x)| \leqslant m$ for any $x \in \partial\Omega^m$. Indeed, if it were not the case, let $x_0 \in \partial\Omega^m$ such that $f^*(x_0) > m$. Let:

$$z_0 \in \underset{S_\Omega(z)=-m}{\operatorname{argmin}} ||x_0 - z||$$

be a projection of $x_0$ onto the $-m$ isosurface of $S_\Omega$. We have $f^*(z_0) \leqslant -m$ by continuity of $f^*$. By definition of $\partial\Omega^m$ and the Lipschitz property of $f^*$, we have:

$$|f^*(x_0) - f^*(z_0)| \leqslant ||x_0 - z_0|| \leqslant 2m$$

but also $f^*(x_0) > m$ and $f^*(z_0) \leqslant m$ which means that:

$$|f^*(x_0) - f^*(z_0)| = f^*(x_0) - f^*(z_0) > 2m$$

which is a contradiction. The case $f(x_0) < -m$ can be treated similarly by projecting onto the $+m$ isosurface of $S_\Omega$ and deriving the same inequalities.

In summary, the following inequalities hold:

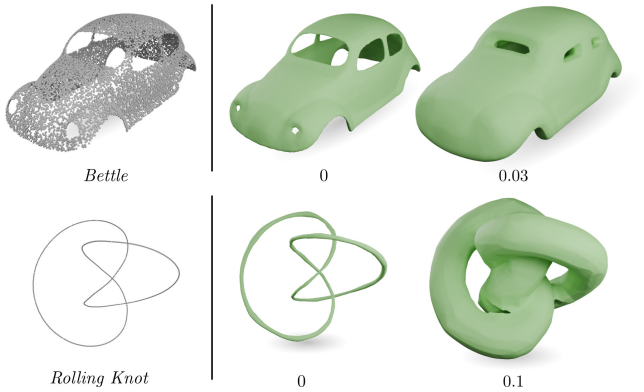



**Figure 12:** *Learning of an unsigned distance field in the case of an open surface with holes (top) and a curve in 3D (bottom) represented as point clouds. We display level-set extracted via the marching cube algorithm for two values of the distance function. The neural field correctly captures the geometry of the object.*

$$\begin{array}{ll} |S_\Omega(x)| > m \implies & |f^*(x)| > m \\ |S_\Omega(x)| \leqslant m \implies & |f^*(x)| \leqslant m \end{array}.$$

In particular, this means that inside $\partial\Omega^m$, the error made by $f^*$ is at most $2m$.

Now, consider the exterior of the shell. By the Lipschitz property of $f^*$, it is clear that $yf^*(x) \leqslant yS_\Omega(x)$ for all $x$. Suppose that this inequality is not tight for some $x_0 \in D\backslash\partial\Omega^m$. By continuity, this means that there exists a ball $B_\varepsilon$ included in $D\backslash\partial\Omega^m$ around $x_0$ for which the inequality is also strict. Computing the KR loss over the domain yields:

$$\int_D -\rho y f(x)dx > \int_D -\rho y S_\Omega(x).$$

However, since $S_\Omega$ is a 1-Lipschitz function and $\mathcal{L}^m_{hinge}(S_\Omega, y) = 0$, this inequality is in contradiction with the fact that $f^*$ is a minimizer of $\mathcal{L}_{KR}$. Combined with the fact that $f^*$ has the same sign as $y$ (and therefore as $S_\Omega$), we can conclude that:

$$\forall x, |S_\Omega(x)| > m \implies f^*(x) = S_\Omega(x).$$

□

## References


[ACB17] ARJOVSKY M., CHINTALA S., BOTTOU L.: Wasserstein GAN, Dec. 2017. arXiv:1701.07875. 5

[AHD*23] ARAUJO A., HAVENS A., DELATTRE B., ALLAUZEN A., HU B.: A Unified Algebraic Perspective on Lipschitz Neural Networks, Oct. 2023. arXiv:2303.03169. 2, 4

[AL20a] ATZMON M., LIPMAN Y.: SAL: Sign Agnostic Learning of Shapes From Raw Data. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Seattle, WA, USA, June 2020), IEEE, pp. 2562–2571. doi:10.1109/CVPR42600.2020.00264. 3, 8

[AL20b] ATZMON M., LIPMAN Y.: SALD: Sign Agnostic Learning with Derivatives. In *International Conference on Learning Representations* (Oct. 2020). 3, 7, 8

[ALG19] ANIL C., LUCAS J., GROSSE R.: Sorting out Lipschitz function approximation, June 2019. arXiv:1811.05381. 4

[AZ23] AYDINLILAR M., ZANNI C.: Forward inclusion functions for ray-tracing implicit surfaces. *Computers & Graphics 114* (Aug. 2023), 190–200. doi:10.1016/j.cag.2023.05.026. 3

[BB97] BLOOMENTHAL J., BAJAJ C.: *Introduction to Implicit Surfaces*. Morgan Kaufmann, Aug. 1997. 1

[BBS*22] BÉTHUNE L., BOISSIN T., SERRURIER M., MAMALET F., FRIEDRICH C., GONZÁLEZ-SANZ A.: Pay attention to your loss: Understanding misconceptions about 1-Lipschitz neural networks, Oct. 2022. arXiv:2104.05097, doi:10.48550/arXiv.2104.05097. 4, 6, 9

[BDS*18] BARILL G., DICKSON N. G., SCHMIDT R., LEVIN D. I. W., JACOBSON A.: Fast winding numbers for soups and clouds. *ACM Transactions on Graphics 37*, 4 (Aug. 2018), 1–12. doi:10.1145/3197517.3201337. 2, 4, 7

[BHLM19] BARTLETT P. L., HARVEY N., LIAW C., MEHRABIAN A.: Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research 20*, 63 (2019), 1–17. 9

[Bli82] BLINN J. F.: A Generalization of Algebraic Surface Drawing. *ACM Transactions on Graphics 1*, 3 (July 1982), 235–256. doi:10.1145/357306.357310. 2, 3

[BNB*23] BETHUNE L., NOVELLO P., BOISSIN T., COIFFIER G., SERRURIER M., VINCENOT Q., TROYA-GALVIS A.: Robust One-Class Classification with Signed Distance Function using 1-Lipschitz Neural Networks, Jan. 2023. arXiv:2303.01978. 3, 5, 6, 9

[CBG*17] CISSE M., BOJANOWSKI P., GRAVE E., DAUPHIN Y., USUNIER N.: Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (Sydney, NSW, Australia, Aug. 2017), ICML'17, JMLR.org, pp. 854–863. 4

[CD23] CLÉMOT M., DIGNE J.: Neural skeleton: Implicit neural representation away from the surface. *Computers & Graphics 114* (Aug. 2023), 368–378. doi:10.1016/j.cag.2023.06.012. 3, 9

[CmP20] CHIBANE J., MIR M. A., PONS-MOLL G.: Neural Unsigned Distance Fields for Implicit Function Learning. In *Advances in Neural Information Processing Systems* (2020), vol. 33, Curran Associates, Inc., pp. 21638–21652. 3

[CZ19] CHEN Z., ZHANG H.: Learning Implicit Fields for Generative Shape Modeling, Sept. 2019. arXiv:1812.02822. 3

[dLJ*15] DE ARAÚJO B. R., LOPES D. S., JEPP P., JORGE J. A., WYVILL B.: A Survey on Implicit Surface Polygonization. *ACM Computing Surveys 47*, 4 (May 2015), 60:1–60:39. doi:10.1145/2732197. 2

[DNJ21] DAVIES T., NOWROUZEZAHRAI D., JACOBSON A.: On the Effectiveness of Weight-Encoded Neural Implicit 3D Shapes, Jan. 2021. arXiv:2009.09808, doi:10.48550/arXiv.2009.09808. 3, 7, 8

[Duf92] DUFF T.: Interval arithmetic recursive subdivision for implicit functions and constructive solid geometry. *ACM SIGGRAPH Computer Graphics 26*, 2 (July 1992), 131–138. doi:10.1145/142920.134027. 3

[FP06] FRISKEN S. F., PERRY R. N.: Designing with distance fields. In *ACM SIGGRAPH 2006 Courses* (New York, NY, USA, July 2006), SIGGRAPH '06, Association for Computing Machinery, pp. 60–66. doi:10.1145/1185657.1185675. 2

[GAA*17] GULRAJANI I., AHMED F., ARJOVSKY M., DUMOULIN V., COURVILLE A.: Improved Training of Wasserstein GANs, Dec. 2017. arXiv:1704.00028, doi:10.48550/arXiv.1704.00028. 3

[GGPP20] GALIN E., GUÉRIN E., PARIS A., PEYTAVIE A.: Segment Tracing Using Local Lipschitz Bounds. *Computer Graphics Forum 39*, 2 (2020), 545–554. doi:10.1111/cgf.13951. 3

[GKUP11] GSCHWANDTNER M., KWITT R., UHL A., PREE W.: Blensor: Blender sensor simulation toolbox. In *Advances in Visual Computing: 7th International Symposium, ISVC 2011, Las Vegas, NV, USA, September 26-28, 2011. Proceedings, Part II 7* (2011), Springer, pp. 199–208. 9, 10

[GYH*20] GROPP A., YARIV L., HAIM N., ATZMON M., LIPMAN Y.: Implicit Geometric Regularization for Learning Shapes, July 2020. arXiv:2002.10099. 3

[Har95] HART J.: Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces. *The Visual Computer 12* (June 1995). doi:10.1007/s003710050084. 2

[JD20] JORDAN M., DIMAKIS A. G.: Exactly Computing the Local Lipschitz Constant of ReLU Networks. In *Advances in Neural Information Processing Systems* (2020), vol. 33, Curran Associates, Inc., pp. 7344–7353. 3

[JKS13] JACOBSON A., KAVAN L., SORKINE-HORNUNG O.: Robust inside-outside segmentation using generalized winding numbers. *ACM Transactions on Graphics 32*, 4 (July 2013), 33:1–33:12. doi:10.1145/2461912.2461916. 6

[JP*18] JACOBSON A., PANOZZO D., ET AL.: libigl: A simple C++ geometry processing library, 2018. https://libigl.github.io/. 7

[KB89] KALRA D., BARR A. H.: Guaranteed ray intersections with implicit surfaces. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, July 1989), SIGGRAPH '89, Association for Computing Machinery, pp. 297–306. doi:10.1145/74333.74364. 3

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 3

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics 21*, 4 (Aug. 1987), 163–169. doi:10.1145/37402.37422. 2, 7, 8

[Lip21] LIPMAN Y.: Phase Transitions, Distance Functions, and Implicit Neural Representations. In *Proceedings of the 38th International Conference on Machine Learning* (July 2021), PMLR, pp. 6702–6712. 3, 9

[LWJ*22] LIU H.-T. D., WILLIAMS F., JACOBSON A., FIDLER S., LITANY O.: Learning Smooth Neural Functions via Lipschitz Regularization. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings* (Vancouver BC Canada, Aug. 2022), ACM, pp. 1–13. doi:10.1145/3528233.3530713. 3

[MDAA22] MEUNIER L., DELATTRE B., ARAUJO A., ALLAUZEN A.: A Dynamical System Perspective for Lipschitz Neural Networks, Feb. 2022. arXiv:2110.12690. 4

[MON*19] MESCHEDER L., OECHSLE M., NIEMEYER M., NOWOZIN S., GEIGER A.: Occupancy Networks: Learning 3D Reconstruction in Function Space, Apr. 2019. arXiv:1812.03828. 3

[MSLJ23] MARSCHNER Z., SELLÁN S., LIU H.-T. D., JACOBSON A.: Constructive Solid Geometry on Neural Signed Distance Fields. In *SIGGRAPH Asia 2023 Conference Papers* (New York, NY, USA, Dec. 2023), SA '23, Association for Computing Machinery, pp. 1–12. doi:10.1145/3610548.3618170. 9

[PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation, Jan. 2019. arXiv:1901.05103, doi:10.48550/arXiv.1901.05103. 3

[PGMK23] PETROV D., GADELHA M., MECH R., KALOGERAKIS E.: ANISE: Assembly-based Neural Implicit Surface rEconstruction. *IEEE Transactions on Visualization and Computer Graphics* (2023), 1–11. arXiv:2205.13682, doi:10.1109/TVCG.2023.3265306. 3

[PKH23] PIWEK P., KLUKOWSKI A., HU T.: Exact count of boundary pieces of relu classifiers: Towards the proper complexity measure for classification. In *Uncertainty in Artificial Intelligence* (2023), PMLR, pp. 1673–1683. 9

[PL22] PRACH B., LAMPERT C. H.: Almost-Orthogonal Layers for Efficient General-Purpose Lipschitz Networks. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXI* (Berlin, Heidelberg, Oct. 2022), Springer-Verlag, pp. 350–365. doi:10.1007/978-3-031-19803-8_21. 4

[Quia] QUILEZ I.: 3d sdf functions. https://iquilezles.org/articles/distfunctions/. Accessed: 2024-05-29. 2

[Quib] QUILEZ I.: Interior sdfs. https://iquilezles.org/articles/interiordistance/. Accessed: 2024-05-29. 9

[Ric73] RICCI A.: A constructive geometry for computer graphics. *The Computer Journal 16*, 2 (Jan. 1973), 157–160. doi:10.1093/comjnl/16.2.157. 2, 9

[SC20] SAWHNEY R., CRANE K.: Monte Carlo geometry processing: A grid-free approach to PDE-based methods on volumetric domains. *ACM Transactions on Graphics 39*, 4 (Aug. 2020), 123:123:1–123:123:18. doi:10.1145/3386569.3392374. 2

[SJ22] SHARP N., JACOBSON A.: Spelunking the deep: Guaranteed queries on general neural implicit surfaces via range analysis. *ACM Transactions on Graphics 41*, 4 (July 2022), 1–16. doi:10.1145/3528223.3530155. 2, 3

[SMB*20] SITZMANN V., MARTEL J., BERGMAN A., LINDELL D., WETZSTEIN G.: Implicit Neural Representations with Periodic Activation Functions. In *Advances in Neural Information Processing Systems* (2020), vol. 33, Curran Associates, Inc., pp. 7462–7473. 3, 7, 8

[SMG*21] SERRURIER M., MAMALET F., GONZALEZ-SANZ A., BOISSIN T., LOUBES J.-M., DEL BARRIO E.: Achieving robustness in classification using optimal transport with hinge regularization. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Nashville, TN, USA, June 2021), IEEE, pp. 505–514. doi:10.1109/CVPR46437.2021.00057. 2, 4, 5, 10

[SP86] SEDERBERG T. W., PARRY S. R.: Free-form deformation of solid geometric models. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, Aug. 1986), SIGGRAPH '86, Association for Computing Machinery, pp. 151–160. doi:10.1145/15922.15903. 2

[SS03] SETHIAN J. A., SMEREKA P.: Level Set Methods for Fluid Interfaces. *Annual Review of Fluid Mechanics 35*, 1 (2003), 341–372. doi:10.1146/annurev.fluid.35.101101.161105. 2

[SSS74] SUTHERLAND I. E., SPROULL R. F., SCHUMACKER R. A.: A Characterization of Ten Hidden-Surface Algorithms. *ACM Computing Surveys 6*, 1 (Mar. 1974), 1–55. doi:10.1145/356625.356626. 6

[Tel16] TELGARSKY M.: Benefits of depth in neural networks. In *Conference on learning theory* (2016), PMLR, pp. 1517–1539. 9

[TK20] TROCKMAN A., KOLTER J. Z.: Orthogonalizing Convolutional Layers with the Cayley Transform. In *International Conference on Learning Representations* (Oct. 2020). 4

[TSM*20] TANCIK M., SRINIVASAN P., MILDENHALL B., FRIDOVICH-KEIL S., RAGHAVAN N., SINGHAL U., RAMAMOORTHI R., BARRON J., NG R.: Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *Advances in Neural Information Processing Systems* (2020), vol. 33, Curran Associates, Inc., pp. 7537–7547. 9

[V*09] VILLANI C., ET AL.: *Optimal transport: old and new*, vol. 338. Springer, 2009. 5

[VS18] VIRMAUX A., SCAMAN K.: Lipschitz regularity of deep neural networks: Analysis and efficient estimation. In *Advances in Neural Information Processing Systems* (2018), vol. 31, Curran Associates, Inc. 3

[WMW86] WYVILL G., MCPHEETERS C., WYVILL B.: Data structure forsoft objects. *The Visual Computer 2*, 4 (Aug. 1986), 227–234. doi:10.1007/BF01900346. 3

[XTS*22] XIE Y., TAKIKAWA T., SAITO S., LITANY O., YAN S., KHAN N., TOMBARI F., TOMPKIN J., SITZMANN V., SRIDHAR S.: Neural Fields in Visual Computing and Beyond. *Computer Graphics Forum 41*, 2 (2022), 641–676. doi:10.1111/cgf.14505. 3

[Yar18] YAROTSKY D.: Optimal approximation of continuous functions by very deep relu networks. In *Conference on learning theory* (2018), PMLR, pp. 639–649. 9

[YGKL21] YARIV L., GU J., KASTEN Y., LIPMAN Y.: Volume Rendering of Neural Implicit Surfaces. In *Advances in Neural Information Processing Systems* (2021), vol. 34, Curran Associates, Inc., pp. 4805–4815. 3

[YM17] YOSHIDA Y., MIYATO T.: Spectral Norm Regularization for Improving the Generalizability of Deep Learning, May 2017. arXiv:1705.10941. 4

[YWÖS21] YIFAN W., WU S., ÖZTIRELI C., SORKINE-HORNUNG O.: Iso-Points: Optimizing Neural Implicit Surfaces with Hybrid Representations. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2021), pp. 374–383. doi:10.1109/CVPR46437.2021.00044. 9

[ZJ16] ZHOU Q., JACOBSON A.: Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797* (2016). 1