Computers & Graphics (2024)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Geometric implicit neural representations

ARTICLE INFO

Article history:
Received March 19, 2024

Keywords: Neural Fields, Implicit Representations, Training Strategies

ABSTRACT

Implicit Neural Representations has emerged as a promising framework for representing different types of signals in low dimensions. This survey focuses on the existing literature and shares practical insights derived from hands-on experimentation with geometric implicit neural representations, specifically in approximating implicit functions of surfaces and modeling dynamic variations of them.

1. Introduction

An *implicit neural representation* (INR) is a *neural network* that parameterizes a signal in a low-dimensional domain. This representation differs from classical methods, as it encodes the signal implicitly in its parameters by mapping coordinates to target signal values. For example, in the case of an implicit surface, an INR f takes a 3D point p and returns the isosurface value $f(p)$. In this scenario, we aim for the INR to approximate the input data as closely as possible, similar to the classic problem of approximating signals using *radial basis functions* (RBFs). INRs provide a compact, high-quality, and smooth approximation for discrete data. Furthermore, INRs allow for the calculation of higher-order derivatives in closed form through automatic differentiation, which is present in modern machine learning frameworks.

INRs are smooth, compact networks that are fast to evaluate and have a high representational capacity. This has motivated their use in several contexts; examples include images [4, 20], face morphing [25], signed distance functions [19, 17, 27, 10, 28, 34, 26, 6], displacement fields [36], surface animation [12, 18], multiresolution signals [21, 24, 10], occupancy [13], constructive solid geometry [11], radiance fields [15, 23], textures [22], among others. These works leverage the fact that INRs are compositions of smooth maps to explore their derivatives during training.

The parameters θ of a INR f are *implicitly* defined as a solution of a non-linear equation $\mathcal{L}(\theta) = 0$. Where $\mathcal{L}$ is defined as a function that enforces f to match to the samples $\{p_i, \ell(p_i)\}$ of the ground-truth function ℓ and to fit some properties hold by ℓ. For example, to fit f to the *signed distance function* (SDF) of a surface, we add a term to the loss function to force the gradient ∇f of the network to be unitary; the Eikonal equation $\|\nabla f\| = 1$. SDFs are solutions of this equation, which is used to regularize the underlying implicit function. In this case, the output $f(p)$ distance value can be positive or negative, representing whether a point is inside or outside the underlying compact surface. A distance value of zero indicates that the point lies on the implicit surface S. The gradient $N = \nabla f$ gives the *normal* field of S and

⋆Only capitalize first word and proper nouns in the title.

its Hessian $\mathbf{H}f$, the *shape operator*, gives the curvatures of S. In this work, we present a survey on such *geometric* approaches that explore these differential objects during training and inference of INRs.

Specifically, we define a *geometric* INR as a neural network $f : \mathbb{R}^3 \to \mathbb{R}$ such that its zero-level set $f^{-1}(0) = \{p \mid f(p) = 0\}$ approximates a given surface S. To train f it is common to use a loss function $\mathcal{L}$ that incorporates a geometric regularization. Again, to enforce $f^{-1}(0) \approx S$, we force f to be the SDF of S by minimizing the Eikonal constraint $\int_\Omega (\|\nabla f\| - 1)^2 dp$, where Ω is the training domain which contains S. Another important geometric term arises from forcing the alignment of the normals N of S with the gradient ∇f, that is, $\int_S (\langle \nabla f, N\rangle - 1)^2 dS$.

This paper brings an in-depth discussion about Geometric Implicit Neural Representations. It also expands on previously explored applications and showcases a set of real-time rendering applications. To do so, we follow the *training pipeline* for geometric INRs. It begins with **input data**, an oriented point cloud $\{p_i, N_i\}_{i=1}^n$ consisting of points p_i on a surface S and their corresponding normals N_i. Next, we define a **neural network** (INR) $f : \mathbb{R}^3 \to \mathbb{R}$ with parameters θ to fit the signed distance function (SDF) of S. This fitting is achieved through optimization of a **geometric loss function** $\mathcal{L}$ using a variant of the gradient descent algorithm. However, computing the gradient $\nabla\mathcal{L}$ may be infeasible in practice due to the size n of the point cloud. Thus, we must consider mini-batches of p_i, N_i for the **sampling** step. In this step, we exploit geometric properties of the underlying surface S to improve training. Once the INR f is trained, we can leverage its SDF properties for various applications, such as geometry **inference** using *sphere tracing* or surface evolution using *level-set methods*.

We also leverage the smoothness of an INR to estimate the curvature measures of its level sets analytically, eliminating the need for discretization. We present recent frameworks that utilize this approach to enhance the training performance of INRs by dynamically sampling data points during training.

We also present examples of smooth neural networks applied to model dynamic variations of implicit surfaces governed by the **level set equation** (LSE). To achieve this, we demonstrate a framework that extends the representation of neural implicit surfaces to the space-time domain $\mathbb{R}^3 \times \mathbb{R}$[18, 12]. The network training involves two constraints: initially, a *data* term is responsible for fitting the initial condition to the corresponding time instant, typically $\mathbb{R}^3 \times \{0\}$. Subsequently, an *LSE* term guides the network to approximate the inherent geometric evolution prescribed by the LSE, **without any supervision**. Potential applications include deforming an initial surface towards general vector fields, smoothing and sharpening using the mean curvature equation, and interpolating initial conditions.

Terminology. In the visual computing community, implicit neural representations have also been referred to as neural fields, neural implicits, and coordinate-based neural networks. In this paper, we focus on the terminology "implicit neural representations" despite some references using the other terms.

The paper is organized as follows. Section 2 discusses the main aspects of implicit surface reconstruction, focusing on the application of the Eikonal equation and classical approaches. Section 3 shows a geometric framework which can be used to solve the geometric implicit neural representation problem. Section 4 presents applications considered state-of-the-art for INRs and Section 5 show a framework for modeling dynamic variations of implicit surfaces under the level set equation(LSE).The final remarks are drawn at then conclusion in Section 6.

2. Implicit surface reconstruction

Implicit representations are commonly used in computer graphics to represent 3D shapes. Unlike explicit geometric representations, which use triangle meshes, implicit geometric representations encode a surface S as the (regular) zero-level set of a function $\ell : \mathbb{R}^3 \to \mathbb{R}$. For the surface S to be regular, the zero must be a regular value of ℓ, that is, $\nabla\ell \neq 0$ on $S = \ell^{-1}(0)$. Again, SDFs is a common example of an implicit representation, where ℓ is the solution of the Eikonal equation

$$\|\nabla\ell\| = 1 \text{ subject to } \ell = 0 \text{ on } S. \tag{1}$$

In this work, we explore strategies to solve Equation 1 by parameterizing ℓ with an INR $f : \mathbb{R}^3 \to \mathbb{R}$, with parameters θ. To approximate a solution of this equation, we define a loss function $\mathcal{L}$ to enforce f to be a solution. Solving this equation reveals that $\langle \nabla \ell, N \rangle = 1$ on S, indicating that $\nabla \ell$ aligns with the normals N of S. We refer to a solution of the above problem as a *geometric INR*.

Before presenting examples of training pipelines for geometric INRs, we recall some classic approaches.

2.1. Classic approaches

Radial basis functions (RBFs) [3] is a classical method that can be used to approximate the SDF of a surface S from a sample $\{p_i, \ell_i\}$ of this function. The RBF is expressed as $s(p) = \sum \lambda_i \phi(\|p - p_i\|)$, where the coefficients $\lambda_i \in \mathbb{R}$ are determined by imposing $s(p_i) = \ell_i$. The *radial function* $\phi : \mathbb{R}^+ \to \mathbb{R}$ is a real function and p_i are the centers of the RBF [17]. Note that the RBF representation depends on the data since its interpolant s depends on the input points p_i.

Poisson surface reconstruction [9] is another classical method widely used in computer graphics to reconstruct a surface from an oriented point cloud $\{p_i, N_i\}$. It revolves around solving the *Poisson's equation*, using $\{p_i, N_i\}$. The objective is to reconstruct an implicit function f of the underlying surface by asking it to be zero at p_i and to have gradients at p_i equal to N_i. The pairs $\{p_i, N_i\}$ are used to define a vector field V. Then, f is computed by optimizing $\min_f \|\nabla f - V\|$ which results in a Poisson problem: $\Delta f = \nabla \cdot V$.

3. Geometric INR framework

3.1. Overview of the problem

This section presents an overall of the geometric framework used to solve the geometric INR problem of training the parameters θ of an INR $f : \mathbb{R}^3 \to \mathbb{R}$ to approximate the SDF of a desired surface S. To present this pipeline we follow the scheme illustrated in Figure 1.

The **input** is a sample of oriented points $\{p_i, N_i\}_{i=1}^n$ from the *ground truth* surface S, and the **output** is an INR f approximating the SDF of S. The framework explores the normals N to define a loss function and the curvatures of the of the surface to sample the mini-batches.

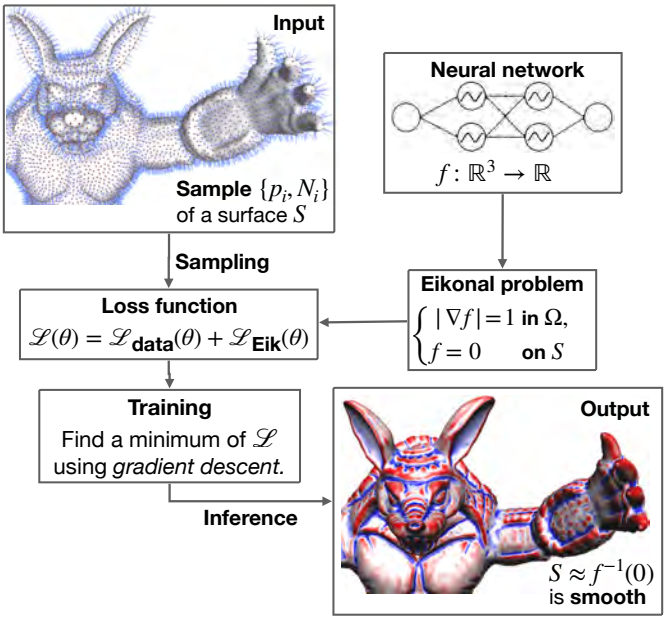

Fig. 1. geometric INR pipeline.

Next, we define a **neural network** (INR) f, with parameters θ, to approximate the SDF of S. For this, we define a **loss function** $\mathcal{L} = \mathcal{L}_{\text{Data}} + \mathcal{L}_{\text{Eik}}$ to enforce f to be a solution of Eikonal equation 1. The data term $\mathcal{L}_{\text{Data}}$ forces f to fit the input data. $\mathcal{L}_{\text{Eik}}$ forces f to be a solution of the Eikonal equation; thus, it works like a regularization, forcing f to be an SDF.

The **training** step consists of using a variant of the gradient descent algorithm to find a minimum of $\mathcal{L}$. However, in practice, computing the gradient $\nabla \mathcal{L}(\theta)$ may be unfeasible; thus, we consider **sampling** minibatches of the input data $\{p_i, N_i\}$. Once we have the INR f trained, we can infer its geometry to **render** its zero-level set $f^{-1}(0)$. The following sections present each component of this pipeline.

We now present the geometric INR training framework in detail; for this, we follow the notation in [17].

3.2. Input data

Given an (oriented) point cloud $\{p_i, N_i\}_{i=1}^n$ sampled from the ground-truth surface S, we can try to reconstruct the SDF $\ell : \mathbb{R}^3 \to \mathbb{R}$ of S. For this, points outside S may be added to the point cloud $\{p_i\}$. After estimating the SDF on the result-

ing point cloud, we obtain a set pair $\{p_i, \ell_i\}$ of points and the approximated SDF values.

3.3. Network architecture

Next, we have to define the architecture of the INR f which will be used to approximate the input data $\{p_i, \ell_i\}$. For this, we assume the INR to be parametrized by a *multilayer perceptron* (MLP) $f : \mathbb{R}^3 \to \mathbb{R}$ defined as follows.

$$f(p) = W_n \circ f_{n-1} \circ f_{n-2} \circ \cdots \circ f_0(p) + b_n \tag{2}$$

where $f_i(p_i) = \varphi(W_i p_i + b_i)$ is the ith layer, and p_i is the output of f_{i-1}, i.e. $p_i = f_{i-1} \circ \cdots \circ f_0(p)$. Here we applies the smooth activation function $\varphi : \mathbb{R} \to \mathbb{R}$ to each coordinate of the affine map, which is formed by the linear map $W_i : \mathbb{R}^{N_i} \to \mathbb{R}^{N_{i+1}}$ and the bias $b_i \in \mathbb{R}^{N_{i+1}}$. The operators W_i are represented as matrices, and b_i as vectors, combining their coefficients to form the parameters θ of the function f. In the following section we define a loss function to train θ to fit f to the input data.

The choice of the activation function φ has a great impact on the representation capacity of f. For example, using sines, that is $\varphi = \sin$, results in a powerful INR architecture [28, 17].

3.4. Loss function

We now define a loss function $\mathcal{L}$ to train the parameters θ of the INR f. For this, we consider the input data $\{p_i, N_i\}$ and the Eikonal equation 1 to define $\mathcal{L}$ as the composition of three components: $\mathcal{L}_{\text{data}}$, and $\mathcal{L}_{\text{Eik}}$.

Specifically, recall that $\{p_i, N_i\}$ is a sample of a compact surface S in $\mathbb{R}^3$. We aim to find a set of parameters θ such that its corresponding INR $f : \mathbb{R}^3 \to \mathbb{R}$ aproximates the SDF ℓ of S. To do this, we minimize the loss function given by Equation (3), which enforces f to be a solution of the Eikonal equation.

$$\mathcal{L}(\theta) = \underbrace{\frac{1}{n}\sum_i f(p_i)^2 + \Big(1 - \langle \nabla f(p_i), N_i \rangle\Big)}_{\mathcal{L}_{\text{data}}} + \underbrace{\int_\Omega (1 - \|\nabla f\|)^2 dp}_{\mathcal{L}_{\text{Eik}}}. \tag{3}$$

Here, $\mathcal{L}_{\text{Eik}}$ encourages f to be the SDF of a set $\mathcal{X}$ by ensuring that $\|\nabla f\| = 1$, $\mathcal{L}_{\text{data}}$ encourages $\mathcal{X}$ to contain S; i.e. $f = \ell$ on $\{p_i\}$. Additionally, it asks for the alignment between ∇f and the normal field of S.

Typically, an additional term is added to Equation (3) to penalize points outside S, forcing f to be the SDF of S (i.e., $\mathcal{X} = S$). In practice, we extended $\mathcal{L}_{\text{data}}$ to consider points outside S by using an approximation of the SDF of S. This also help us to avoid the neural network to generate spurious components in the zero-level set $f^{-1}(0)$. For this, we can incorporate off-surface points to $\mathcal{L}_{\text{data}}$. To achieve this, we consider a sample of k points $\{p_i\}_{i=1}^{n+k}$ in the training domain Ω outside the surface S. Then, we consider the following data constraint.

$$\mathcal{L}_{\text{data}} = \frac{1}{n+k}\sum_{i=1}^{n+k} (f(p_i) - \ell_i)^2 + \frac{1}{n}\sum_{i=1}^{n} \Big(1 - \langle \nabla f(p_i), N_i \rangle\Big) \tag{4}$$

Where ℓ_i is an approximation of the SDF of S at the points p_i, which are computed during training. To achieve this, we use the following approximation:

$$|\ell(p)| \approx \min_{i \le n} \|p - p_i\| \tag{5}$$

The sign of $\ell(p)$ at a specific point p is negative if p lies inside the surface S and positive otherwise. This approximation method enables the training of the neural network to work effectively with the SDF for surface reconstruction. Notice that for each vertex p_i with a normal vector N_i the sign of $\langle p - p_i, N_i \rangle$ indicates the side of the tangent plane that p belongs to [17]. Therefore, we can estimate the sign of $\ell(p)$ by adopting the dominant signs of the numbers $\langle p - p_j, N_j \rangle$, which $\{p_j\} \subset V$ is a set of vertices close to p. This set can be estimated using an Octree or KD-tree, to store the points $\{p_i\}$ [17].

3.5. Sampling

Let $\{p_i, N_i, \mathcal{S}_i\}$ be a sample from an *unknown* surface S, where $\{p_i\}$ are points on S, $\{N_i\}$ are their normals, and $\{\mathcal{S}_i\}$ are samples of the shape operator. Also, $\{p_i\}$ is a set composed of the vertices of a triangle mesh, normals, and its curvatures.

In practice, we could evaluate $\mathcal{L}$ using a dataset of points dynamically sampled during training. This dataset includes on-surface points $\{p_i\}$ and off-surface points in $\mathbb{R}^3 - S$.

For the off-surface points, we can choose uniform sampling within the domain of f_θ. Alternatively, we may bias the sampling by including points in the *tubular neighborhood* of S,

which is a region around the surface formed by segments along the normals.

The shape operator contains important geometric features of the data. Regions with points having higher absolute principal curvatures κ_1 and κ_2 encode more detailed information, while points with lower absolute curvatures represent less intricate geometry. This allows us to focus sampling efforts on regions with significant geometric variations and minimize the need to sample planar regions where curvatures are small.

We can use a smart method to select on-surface points $\{p_i\}$ for faster learning without compromising quality. We divide $\{p_i\}$ into three sets based on their features: V_1 (low), V_2 (medium), and V_3 (high). During training, we prioritize points in V_2 and V_3 as they have more geometrical features. We sample fewer points from V_1 to avoid redundancy and increase sampling from V_2 and V_3 for faster learning. In this way, we focus on essential points for faster and better results.

During training, we usually pick minibatches uniformly. But here, we can use curvature information to focus on important features. We set n as the sum of n_1, n_2, and n_3, where each n_i is a positive integer. This allows us to create three categories: low, medium, and high feature points, denoted as V_1, V_2, and V_3, respectively.

When forming minibatches, each of size $m = p_1m + p_2m + p_3m = 10000$, we can allocate p_im points to the corresponding category V_i. Instead of uniform sampling, we adjust p_1, p_2, and p_3 to prioritize V2 and V3.

In Figure 2, we can see a comparison of uniform (first line) and this adaptive approach (line 2), where we double the proportion of medium and high feature points. This depends on specific values such as n_i. In this test, n_1 is half of n, n_2 is $\frac{4n}{10}$, and n_3 is $\frac{n}{10}$, making sure V_1 contains half of all points. This innovative sampling strategy greatly enhanced convergence rates during our experiments.

3.6. Applications for INRs

Implicit neural representations has several applications for real-time rendering, visualization, and computational geometry. Usually, the neural network presented in this paper has a simple architecture and does not demand powerful hardware[27]. Also, the method can accurately represent geometric details with precision. In Figure 3, we can se the original Bunny model (left) with its reconstructed neural implicit surface (right) after 886 training epochs. The Bunny model is a triangular mesh with 106712 vertices. Notice how the original triangulation (left) appears slightly coarse, like in the ear of Bunny, but the reconstructed neural Bunny fixes this and provides a more detailed representation.

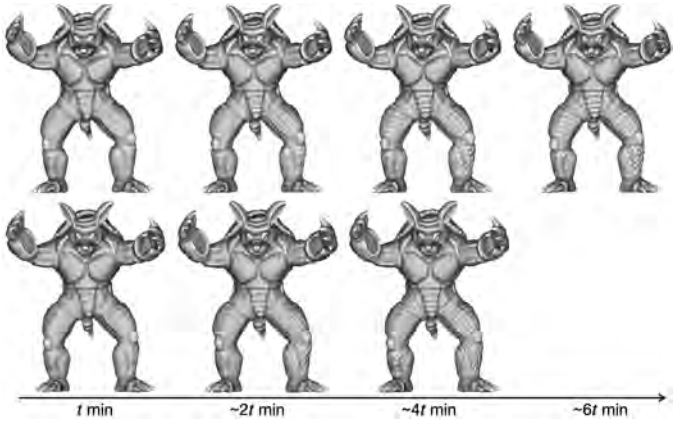

Fig. 2. Neural implicit surfaces approximating the Armadillo model. The columns indicate the zero-level sets of the neural implicit functions after 29, 52, 76, and 100 epochs of training. Line 1 shows the results using minibatches sampled uniformly in V. Line 2 presents the results using the adapted sampling of minibatches with 10% / 70% / 20% of points with low/medium/high features.

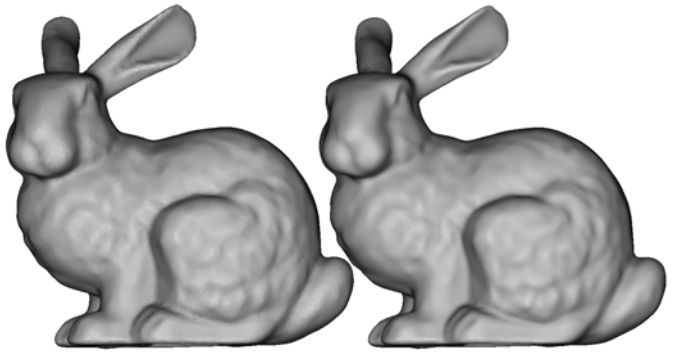

Fig. 3. The Bunny represented by the original triangular mesh and a implicit representation.

3.6.1. Sphere tracing

One application for neural implicit representations is to use neural networks in real-time rendering using algorithms such as Sphere Tracing[6]. To do so, the method provides a robust SDF approximation even compared with RBF. Figure 4 gives a visual evaluation presenting a sphere tracing of the zero-level of

this method. Since it has a good SDF approximation, the algorithm can ray-cast the surface with precision avoiding spurious components.

Fig. 4. Sphere tracing of neural surfaces representing the Armadillo and Bunny models. Both networks to represent the objects have the same architecture and were trained on the same data during 500 **epochs.**

3.6.2. Curvature estimation

To train neural implicit functions, we can use a loss function that approximates an SDF that allows the use of high-order derivatives, such as the alignment between the principal directions of curvature, to learn more geometric details [17].

The study of discrete variations of triangle mesh normals is an important topic in discrete differential geometry and it is very helpful in neural implicit applications, especially when representing 3D shapes. A discrete shape operator represents these variations. Principal directions and curvatures can be defined on edges, with one curvature being zero along the edge direction, and the other measured across the edge using the dihedral angle between adjacent faces. The shape operator at the vertices is estimated by averaging the shape operators of the neighboring edges. The approach of Cohen et al. [17, 14, 5, 31] can be considered in this context.

We train f_θ to approximate the SDF of T. Using the network, we map properties of its level sets to T and estimate curvature measures. In Figure 5, we trained a neural implicit function for the Dragon model, and the calculated curvature using f_θ is smoother and respects the original mesh's curvature distribution. Also, in Figure 6 we show the principal curvatures and directions calculated for the Dragon model.

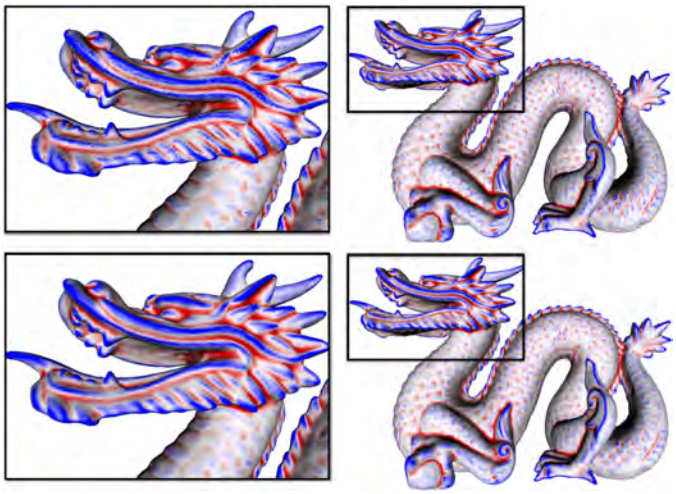

Fig. 5. Gaussian and mean curvatures of the Dragon neural surface. The surface points were computed using sphere tracing and its analytic curvatures using `PyTorch` **framework.**

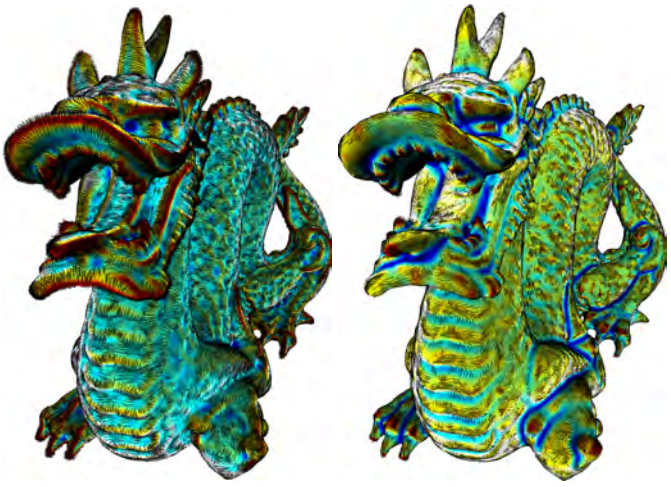

Fig. 6. Principal curvatures and directions overlaid on the Dragon.

4. Signed distance functions and implicit representations

4.1. INR approaches

4.1.1. Implicit Geometric Regularization

Several techniques exist for creating neural implicit representations [19]. For example, IGR [7] is one of the first methods to propose to use the Eikonal equation as regularization. Since they needed to use the gradient ∇f in the loss function, they considered f to be neural network with the softplus activation function. Implicit Differentiable Renderer (IDR) [35] used IGR as basis to propose a neural network that learns geometry, camera parameters, and a neural renderer.

Implicit geometric regularization (IGR) [7] is a geometric INR technique to compute the parameters θ of an INR f by

forcing it to fit the SDF of a surface S. In other words, it provides geometric pipeline to solve the Eikonal problem 1. IGR aims to refine and regularize the shape defined by the zero-level set of an implicit function using additional regularization terms. The authors observe that a relatively simple loss function, similar to the loss function in SIREN, encourages the neural network to vanish on the input point cloud and to have a unit norm gradient (Eikonal constraint).

Let $\{p_i, N\}_{i=1}^{n} \subset \mathbb{R}^3$ be an oriented point cloud sampled from a smooth compact surface S. IGR uses a pipeline similar to the one described in Section 3 to train the parameters θ of a geometric INR $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ by enforcing it to be a solution of Equation 1. Therefore, it forces $f = 0$ in $\{p_i\}$ and $\nabla f(p_i) = N_i$. Again, the implicit geometric regularization occurs because the loss function used during training (Equation 3) encourages certain geometric properties to be preserved by using the Eikonal term. These properties are not explicitly encoded in the training data or loss function but emerge due to the neural network structure and the optimization process [16, 7]. Figure 7 shows some level sets of MLPs trained using IGR.

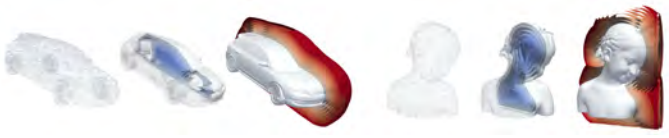

Fig. 7. Level sets of MLPs trained with IGR method [7]

4.1.2. Sinusoidal Implicit Networks

Sitzmann et al. [28] propose to leverage periodic activation functions for implicit neural representations and demonstrate that these networks can represent signals in high detail. Specifically, they parameterize the INR by a sinusoidal MLP (Equation 2) using the sine as activation function ($\varphi = \sin$) and propose an initialization scheme that guaranties instability and good convergence during training. As in IGR[7], we can use Sinusoidal INR (SIREN) to estimate the SDFs of surfaces with high-quality from oriented point samples.

This approach can reproduce fine details of a 3D surface, as show in Figure 8, however, exploring the surface curvatures, we can improve the reconstruction to achieve a smooth and high-fidelity representation.

Fig. 8. Shape representation. SIREN fit signed distance functions parameterized by implicit neural representations directly on point clouds.

4.1.3. Band-limited Coordinate Networks (BACON)

Lindell et al. [10] introduce BACON, an architecture aimed at addressing the limitations associated with traditional coordinate-based networks, especially their challenges in spectral analysis and behavior prediction at unsupervised points, and their confinement to single-scale signal representation. BACON proposes to create an analytical Fourier spectrum, which facilitates controlled behavior at unsupervised points and allows for the signal's spectral characteristics-driven design. It also supports multiscale signal representation without requiring supervision at each scale. This approach has been applied to the neural representation of images, radiance fields, and 3D scenes through signed distance functions, illustrating its capability to represent signals across scales effectively.

In addressing the limitations of BACON, it is noted that its cut-off of the Fourier spectrum introduces artifacts, particularly the ringing effect observed in images and noise on surfaces. This limitation stems from the inherent constraints of band-limiting, which, while facilitating certain advantages in signal representation, can also result in undesirable visual artifacts that affect the quality of the output in applications involving high-frequency detail.

4.1.4. Geometry Processing with Neural Fields

NFGP [30] suggests utilizing neural fields for geometry processing, as they offer distinct advantages. The authors propose approximating a local surface of a level set by utilizing the derivatives of the underlying field. By solely relying on the field derivatives, it is possible to use intrinsic geometric properties of the level set, such as curvatures. This enables the construction of loss functions that capture surface priors like elasticity or rigidity. This is made possible by exploiting the inherent infinite differentiability of neural fields which facilitates the optimization of loss functions involving higher-order derivatives through gradient descent methods. Consequently, unlike mesh-based geometry processing algorithms that rely on surface discretizations to approximate these objectives, this strategy can directly optimize the derivatives of the field.

4.1.5. Differential Geometry in Neural Implicits

In this subsection, we will present works that use concepts of differential geometry combined with the properties described in their loss functions. Exploring Differential Geometry in Neural Implicits (i3D) [17], introduces a new implicit neural representation model, which includes a framework that takes a sample of points from a surface S, along with their corresponding normals and curvatures, as input (the ground truth). Then, a neural network generates the SDF approximation as its output. Also, the authors proposes a loss function that enables the incorporation of tools from continuous differential geometry during the training of the neural implicit function. During the network's training, the method leverages the discrete differential geometry of the point-sampled surface to selectively sample significant regions. This approach ensures a robust and efficient training process while preserving essential geometrical details. The method uses the closed-form derivatives of the neural implicit function to estimate differential measures, such as normals and curvatures, for the underlying point-sampled surface. This estimation is feasible because the point-sampled surface lies in the vicinity of the network's zero-level set. This feature allows for the accurate calculation of differential measures using the neural implicit function.

ORCa [32] uses the concepts established by i3D. The authors discuss the potential of using reflections on glossy objects as a means to capture valuable and hidden information about the surrounding environment. The proposed approach converts glossy objects with unknown geometry into radiance-field cameras to capture images from the perspective of the object. The key insight involves transforming the object surface into a virtual sensor, capturing cast reflections as a 2D projection of the 5D environment radiance field surrounding the object. Recovering the environment radiance fields enables depth and radiance estimation from the object to its surroundings, facilitates beyond field-of-view novel-view synthesis, and allows imaging around occluders caused by nearby objects in the scene. In particular, they use the differential geometry techniques developed by Novello *et al.* [17] to estimate the curvature for neural implicit surfaces.

MARF [29] introduces the Medial Atom Ray Fields, a neural object representation enabling differentiable surface rendering with a single network evaluation per camera ray. MARFs address challenges like multi-view consistency and surface discontinuities by using a medial shape representation, offering cost-effective geometrically grounded surface normals and analytical curvature computation. They map camera rays to multiple medial intersection candidates and demonstrate applicability in sub-surface scattering, part segmentation, and representing articulated shapes. With the ability to learn shape priors, MARFs hold promise for tasks like shape retrieval and completion.

NAISR [8] shows that some of the existing neural implicit shape representations fail to represent shapes while capturing individual dependencies on covariates precisely. To address this, they propose the 3D Neural Additive Model for Interpretable Shape Representation (NAISR), which deforms a shape atlas based on disentangled covariates. It captures shape population trends and allows for patient-specific predictions through shape transfer. NAISR combines deep implicit shape representations with atlas deformation according to covariates, offering interpretability. Also they use the same concepts for

the loss function as the ones defined by Sitzmann *et al.* [28] and Novello *et al.* [17].

Zhang *et al.* [37] introduce implicit neural representations to topology optimization, creating a framework named TOINR. In TOINR, a Topology Description Function, formed by a Neural Network, determines the topology of structural materials. They use sine activations with a MLP architecture, with predefined spatial points as inputs and implicit representations of topological boundaries as outputs. The neural network parameters evolve iteratively, updating structural topologies based on response analysis and optimization functions. TOINR ensures computational differentiability, employing automatic differentiation for sensitivity analysis.

Berzins et al. [2] show that compared to classic geometry representations, neural representations lack intuitive control over shape. To address this, they utilize boundary sensitivity to interpret how parameter changes affect the shape boundary, enabling the study of achievable deformations. This allows for geometric editing by finding parameter updates that approximate prescribed deformations. The method allows local deformation according to priors like semantics or rigidity.

NeuS [33] proposes a volume rendering method to render images from an implicit SDF and minimize the difference between the rendered images and the input images, resulting in a neural network representing the SDF. Yifan et al. [36] used displacement fields to represent detail related to a simplified base SDF to render a detailed surface.

5. Dynamic INRs

As presented in previous sections, coordinate-based neural networks serve as efficient geometry representations, similar to parametric level sets where the zero-level set defines the surface of interest. In this section, we explore a framework to apply deformations or animations to these implicit surfaces. This Level set theory is a powerful mathematical framework and provides a versatile approach for representing and evolving complex shapes over time. In the next subsections, we define the relationship between neural homotopies, partial differential equations (PDE) and Dynamic INRs, the problem modeling, and three application examples.

5.1. Neural homotopies and PDEs

To train neural networks to model animations of an implicit surface S, the level sets $g^{-1}(c)$, with $c \in g(\mathbb{R}^3)$, of the neural implicit function g could be used to animate $S \approx g^{-1}(0)$. However, this does not allow intersections between surfaces at different time steps, because the level sets of g are disjoint. To avoid this, it is possible to extend the domain of the neural implicit function adding parameters to control animations. $\mathbb{R}^3$ has been considered as the domain of smooth neural implicit functions [28, 7, 18]. However, we can augment $\mathbb{R}^3$ by a product space $\mathbb{R}^3 \times \mathbb{R}^k$, where $p \in \mathbb{R}^3$ denotes a point in the space and $c \in \mathbb{R}^k$ denotes a vector that controls k direction of possible animations (rigid motion, smoothing, morphing, ...) of the initial implicit surface.

Specifically, let $f : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ be a smooth neural function representing an *one-parameter* family of neural implicit functions $f_t : \mathbb{R}^3 \to \mathbb{R}$ defined by $f_t(p) = f(p, t)$. Topologically, the family f_t represents *homotopies* between any two functions f_{t_0}, $f_{t_1} : \mathbb{R}^3 \to \mathbb{R}$, with $t_0, t_1 \in \mathbb{R}$. Thus, we say that f is a *neural homotopy function*. The underlying family of neural implicit surfaces $S_t = f_t^{-1}(0)$ is a *neural animation* of the initial surface $S_0 = S$. Restricted to the interval $[t_0, t_1]$, the surfaces S_t provides a *neural morphing* between the neural implicit surfaces S_{t_0} and S_{t_1}. Observe that these surfaces can contain singularities as their topologies may change over time t.

For a neural implicit function, we could consider the neural homotopy $f : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ be a solution of a general *partial differential equation* (PDE) problem

$$\mathcal{F}\left(\nabla^n f, \ldots, \nabla^1 f, f, p, t\right) = 0$$

where $\nabla^k f$ denotes the derivative of order k of f. *Initial-boundary conditions* could also be used.

The time t allows continuous navigation in the neural animation $S_t = f_t^{-1}(0)$. Each instant t also represents a *transformation* $S_0 \to S_t$ of the initial implicit surface S_0.

5.2. Modeling level Sets

By representing a surface evolution with a neural network $f_\theta : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$, given an LSE with initial conditions, it is possible to define a a machine learning framework, consisting of a loss functional, sampling strategies, and a network initialization, to train f_θ to approximate a solution to the LSE problem [18].

Let $g_i : \mathbb{R}^3 \to \mathbb{R}$ be the SDFs of n surfaces S_i. We train f_θ by forcing it to approximate a solution a *neural* LSE:

$$\begin{cases} \mathcal{F} := \dfrac{\partial f_\theta}{\partial t} + \langle \nabla f_\theta, V \rangle = 0 & \text{in } \mathbb{R}^3 \times (a, b), \\ f_\theta = g_i & \text{on } \mathbb{R}^3 \times \{t_i\}. \end{cases} \tag{6}$$

We employed the notation $\mathcal{F}$ to represent the LSE for brevity. The untrained network f_θ must encode the movement governed by the vector field V. The interval (a, b) can be used to control the resulting neural animation S_t of S.

5.3. Using Vector fields

Moving a surface S towards a vector field $V : \mathbb{R}^3 \to \mathbb{R}^3$ yields a simple Level Set Equation (LSE) where g represents the Signed Distance Function (SDF) of S, and V can be predefined and tailored for various applications.

Novello et al. [18] trained a neural network $f_\theta : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}$ to implicitly encode the evolution of S by V using the resulting LSE described by Equation 7:

$$\begin{cases} \dfrac{\partial f_\theta}{\partial t} - v \|\nabla f_\theta\| = 0 & \text{in } \mathbb{R}^3 \times (a, b), \\ f_\theta = g & \text{on } \mathbb{R}^3 \times \{0\}. \end{cases} \tag{7}$$

where v denotes the size of the normal component of V, and the minus sign in Equation 7 is utilized to accommodate the inverse of the resulting flow to compose with g.

Let us present a deformation of the Spot, where g is the SDF of the original mesh. Let $V = V_1 - V_2$ as the sum of a *source* V_1 and a *sink* $-V_2$, with $V_i(p) = e^{-\frac{|p-p_i|^2}{0.18}}(p - p_i)$. The points p_1, and p_2 are the centers of Spot's body and head. We can use V to derive a loss function to train f_θ and parameterize f_θ with one hidden layer $f_i : \mathbb{R}^{128} \to \mathbb{R}^{128}$ and train it for 46000 epochs. This approach deformed the Spot's head while it increased the body size, as we can see in Figure 9.

Fig. 9. Evolving the zero-level sets of a network according to a vector field with a source and a sink. The SDF of the Spot is the initial condition at $t = 0$ (middle). The sink/source is inside the head/body of the Spot [18].

5.4. Interpolation between surfaces

The SDFs (g_i) of two surfaces (S_i) can be interpolated using an LSE approach, where a vector field V is constructed to interpolate between the surfaces, given by $V(p, t) = -(g_2(p) - f(p, t))\frac{\nabla f(t,p)}{\|\nabla f(t,p)\|}$, with initial condition $f(p, 0) = g_1(p)$. This evolution enforces each c-level set of g_1 to align with g_2.

The resulting LSE is expressed by substituting the vector interpolation into the PDE, resulting in $\frac{\partial f}{\partial t} - \|\nabla f\|(g_2 - f) = 0$ in $\mathbb{R}^3 \times \mathbb{R}$, with $f = g_i$ on $\mathbb{R}^3 \times \{t_i\}$. A solution f locally inflates S_1 if inside S_2 and deflates it if outside, ensuring S_1 fits into S_2. A loss function $\mathcal{L}_{\text{LSE}} + \mathcal{L}_{\text{data}}$ is defined to fit a solution of the PDE using $\mathcal{F} := \frac{\partial f}{\partial t} - \|\nabla f\|(g_2 - f)$.

Suppose g_i are the SDFs of the Bob and Spot (Figure 10, left-right) and that f_θ has 1 hidden layer $f_i : \mathbb{R}^{128} \to \mathbb{R}^{128}$. It is possible to train f_θ using $\mathcal{L}_{\text{LSE}} + \mathcal{L}_{\text{data}}$. Figure 10 shows the reconstructions of the level sets.

Fig. 10. Interpolation between Bob and Spot.

5.5. Using the mean curvature equation

The *mean curvature equation* can be used to evolve the level sets with velocity given by the negative of their mean curvature, resulting in a smoothing along the time [1, 18].

Let $V(p, t) = -\kappa(p, t)N(p, t)$ be the *mean curvature vector*, where N is the normal field of the level sets and $\kappa = \operatorname{div} N$ is the *mean curvature*; div is the *divergence* operator. We can

model the mean curvature equation as:

$$\begin{cases} \dfrac{\partial f}{\partial t} - \alpha \|\nabla f\| \kappa_\theta = 0 & \text{in } \mathbb{R}^3 \times (a, b), \\ f = g & \text{on } \mathbb{R}^3 \times \{t = 0\}. \end{cases} \tag{8}$$

Intuitively, the zero-level set moves toward the mean curvature vector $-\kappa N$, contracting regions with positive curvature and expanding regions with negative curvature. Thus, such procedure *smooths* (*sharpens*) the surface if $t > 0$ ($t < 0$) and α controls the level set evolution.

Figure 11 presents three reconstructions of the zero-level sets of f_θ at times $t = 0, 0.1, 0.2$ using a neural network with 2 hidden layers $f_i : \mathbb{R}^{256} \rightarrow \mathbb{R}^{256}$.In this example, the Armadillo surface was reconstructed at $t = 0$ and, as time progressed, it became smoother. Regions with positive mean curvature, such as the fingers, contracted.

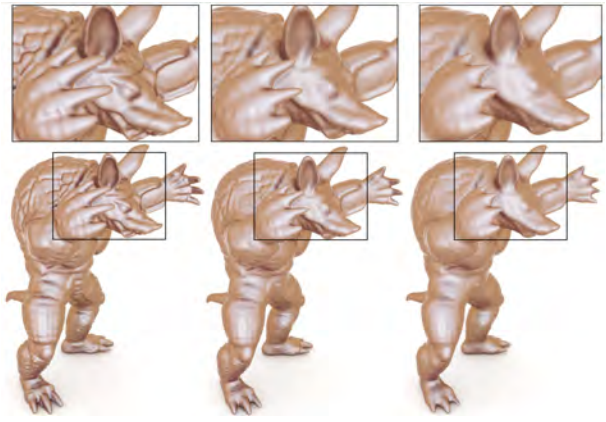

Fig. 11. Armadillo smoothing using the mean curvature equation[18].

For the sharpening, the zero-level sets where reconstructed at $t = 0, -0.1, -0.2$. In Figure Figure 12, regions with positive curvature have expanded, resulting in an enhancement of the geometrical features of the surface.

5.6. Neural Implicit evolution techniques

By using some concepts derived from i3D, Novello et al. [18] proposed a new strategy that investigates the use of smooth neural networks for modeling dynamic variations of implicit surfaces under the level set equation (LSE). For this, it extends the representation of neural implicit surfaces to the space-time $\mathbb{R}^3 \times \mathbb{R}$, which opens up mechanisms for continuous geometric transformations. Examples include evolving an initial surface towards general vector fields, smoothing and sharpening

Fig. 12. Using the mean curvature equation to enhance the geometrical details of the Armadillo surface[18].

using the mean curvature equation, and interpolating initial conditions.

The training for dynamic implicit neural representations involves two constraints. A data term that fits the initial condition to a corresponding time-step, typically $\mathbb{R}^3 \times \{0\}$. Then, an LSE term guides the network to approximate the geometric evolution without supervision. Also, initializing the network with previously trained conditions accelerates convergence compared to the standard approach. This method is one of the first neural approaches in geometry processing that does not consider numerical approximations of the LSE solution during sampling and does not discretize the LSE in the loss function. In this sense, NFGP [34] and NIE [12] are another two recent neural approaches exploring the concepts of dynamic implicit surfaces. Unlike Novello *et al.*[18], both operate in discrete time steps. They represent the evolution as a sequence of networks, with each network corresponding to a time step. Specifically, NFGP and NIE use a network g_ϕ to fit an initial function g, and then update ϕ at each time step, resulting in a sequence of networks $g_{\phi_i} : \mathbb{R}^3 \rightarrow \mathbb{R}$. While this approach is similar to Runge-Kutta methods, where the solution is fitted to a grid, NFGP and NIE use neural networks instead. Consequently, they need to retrain the networks to evaluate at intermediate times. Also, the NIE[12] method is one of the unique approaches that evolve g_ϕ using the mean curvature equation. However, it faces challenges as it relies on discretizing the partial derivative $\frac{\partial f}{\partial t}$. To update ϕ, NIE employs a finite difference scheme to approxi-

mate the discrete solutions. In computing the mean curvature κ, NIE extracts level sets using marching cubes and utilizes the cotangent Laplacian, which poses issues as it relies on approximating level sets by meshes using marching cubes. NFGP evolves a network $g_\theta : \mathbb{R}^3 \to \mathbb{R}$ such that the level set of the resulting network g_ϕ can smooth or sharpen $g_\theta^{-1}(0)$. The training optimizes $(\kappa_\phi - \beta\kappa_\theta)^2$, the difference between the mean curvatures of the level sets of g_ϕ and g_θ. Then, using $\beta < 1$ or $\beta > 1$, it would force a smoothing or sharpening of the initial surface. NFGP trains a network g_ϕ for each β, thus this strategy does not represent a continuous evolution over time and does not use the mean curvature equation model.

6. Conclusion

In this paper, we presented a survey of techniques for training a neural network framework that capitalizes on both the differentiability of neural networks and the discrete geometry of point-sampled surfaces, resulting in the creation of neural surfaces. Our exploration included notable contributions from papers such as SIREN [28] and IGR [7] within this domain.

The integration of concepts from differential geometry holds promise for modeling applications requiring curvature terms in the loss function. Additionally, we demonstrated that a sampling strategy based on discrete curvatures of data could enhance training by targeting points with more comprehensive geometric data in minibatch sampling. We also present strategies that explore level sets of implicit neural function sets in Dynamic INRs.

In summary, this paper aimed to uncover strategies, such as sampling approaches, to accelerate training convergence, and application for Dynamic INRs. We harnessed the potential of using approximate SDF values during training to mitigate artifacts and shed light on future directions in this field through the application of discrete geometry concepts.

References

[1] Giovanni Bellettini. *Lecture notes on mean curvature flow: barriers and singular perturbations*, volume 12. Springer, 2014.

[2] Arturs Berzins, Moritz Ibing, and Leif Kobbelt. Neural implicit shape editing using boundary sensitivity. *arXiv preprint arXiv:2304.12951*, 2023.

[3] Jonathan C Carr, Richard K Beatson, Jon B Cherrie, Tim J Mitchell, W Richard Fright, Bruce C McCallum, and Tim R Evans. Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 67–76, 2001.

[4] Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8628–8638, 2021.

[5] David Cohen-Steiner and Jean-Marie Morvan. Restricted delaunay triangulations and normal cycle. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 312–321, 2003.

[6] Vinícius da Silva, Tiago Novello, Guilherme Schardong, Luiz Schirmer, Hélio Lopes, and Luiz Velho. Mip-plicits: Level of detail factorization of neural implicits sphere tracing. *arXiv preprint arXiv:2201.09147*, 2, 2022.

[7] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020.

[8] Yining Jiao, Carlton Zdanski, Julia Kimbell, Andrew Prince, Cameron Worden, Samuel Kirse, Christopher Rutter, Benjamin Shields, William Dunn, Jisan Mahmud, et al. Naisr: A 3d neural additive model for interpretable shape representation. *arXiv preprint arXiv:2303.09234*, 2023.

[9] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[10] David B Lindell, Dave Van Veen, Jeong Joon Park, and Gordon Wetzstein. Bacon: Band-limited coordinate networks for multiscale scene representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16252–16262, 2022.

[11] Zoë Marschner, Silvia Sellán, Hsueh-Ti Derek Liu, and Alec Jacobson. Constructive solid geometry on neural signed distance fields. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–12, 2023.

[12] Ishit Mehta, Manmohan Chandraker, and Ravi Ramamoorthi. A level set theory for neural implicit evolution under explicit flows. In *European Conference on Computer Vision*, pages 711–729. Springer, 2022.

[13] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019.

[14] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*, pages 35–57. Springer, 2003.

[15] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

[16] Behnam Neyshabur. Implicit regularization in deep learning. *arXiv preprint arXiv:1709.01953*, 2017.

[17] Tiago Novello, Guilherme Schardong, Luiz Schirmer, Vinicius da Silva, Helio Lopes, and Luiz Velho. Exploring differential geometry in neural implicits. *Computers & Graphics*, 108:49–60, 2022.

[18] Tiago Novello, Vinicius da Silva, Guilherme Schardong, Luiz Schirmer, Helio Lopes, and Luiz Velho. Neural implicit surface evolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14279–14289, 2023.

[19] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[20] Hallison Paz, Tiago Novello, Vinicius Silva, Luiz Schirmer, Guilherme Schardong, Fabio Chagas, Helio Lopes, and Luiz Velho. Multiresolution neural networks for imaging. In *Proceedings of 35th Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2022. to appear.

[21] Hallison Paz, Daniel Perazzo, Tiago Novello, Guilherme Schardong, Luiz Schirmer, Vinicius da Silva, Daniel Yukimura, Fabio Chagas, Helio Lopes, and Luiz Velho. Mr-net: Multiresolution sinusoidal neural networks. *Computers & Graphics*, 2023.

[22] Hallison Paz, Tiago Novello, and Luiz Velho. Implicit neural representation of tileable material textures. *arXiv preprint arXiv:2402.02208*, 2024.

[23] Daniel Perazzo, João Paulo Lima, Luiz Velho, and Veronica Teichrieb. Directvoxgo++: Grid-based fast object reconstruction using radiance fields.

Computers & Graphics, 114:96–104, 2023.

[24] Vishwanath Saragadam, Jasper Tan, Guha Balakrishnan, Richard G Baraniuk, and Ashok Veeraraghavan. Miner: Multiscale implicit neural representation. In *European Conference on Computer Vision*, pages 318–333. Springer, 2022.

[25] Guilherme Schardong, Tiago Novello, Hallison Paz, Iurii Medvedev, Vinícius da Silva, Luiz Velho, and Nuno Gonçalves. Neural implicit morphing of face images, 2024.

[26] Luiz Schirmer, Guilherme Schardong, Vinícius da Silva, Hélio Lopes, Tiago Novello, Daniel Yukimura, Thales Magalhaes, Hallison Paz, and Luiz Velho. Neural networks for implicit representations of 3d scenes. In *2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 17–24. IEEE, 2021.

[27] Luiz Schirmer, Tiago Novello, Guilherme Schardong, Vinícius da Silva, Hélio Lopes, and Luiz Velho. How to train your (neural) dragon. In *Conference on Graphics, Patterns and Images*, 2023. URL `http://urlib.net/ibi/8JMKD3MGPEW34M/49T46US`.

[28] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[29] Peder Bergebakken Sundt and Theoharis Theoharis. Marf: The medial atom ray field object representation. *Computers & Graphics*, 115:122–136, 2023.

[30] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. *arXiv preprint arXiv:2101.10994*, 2021.

[31] Gabriel Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *Proceedings of IEEE International Conference on Computer Vision*, pages 902–907. IEEE, 1995.

[32] Kushagra Tiwary, Akshat Dave, Nikhil Behari, Tzofi Klinghoffer, Ashok Veeraraghavan, and Ramesh Raskar. Orca: Glossy objects as radiance-field cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20773–20782, 2023.

[33] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021.

[34] Guandao Yang, Serge Belongie, Bharath Hariharan, and Vladlen Koltun. Geometry processing with neural fields. *Advances in Neural Information Processing Systems*, 34:22483–22497, 2021.

[35] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Ronen Basri, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *arXiv preprint arXiv:2003.09852*, 2020.

[36] Wang Yifan, Lukas Rahmann, and Olga Sorkine-Hornung. Geometry-consistent neural shape representation with implicit displacement fields. *arXiv preprint arXiv:2106.05187*, 2021.

[37] Zeyu Zhang, Wen Yao, Yu Li, Weien Zhou, and Xiaoqian Chen. Topology optimization via implicit neural representations. *Computer Methods in Applied Mechanics and Engineering*, 411:116052, 2023.