How to train your (neural) dragon

Luiz Schirmer, Unisinos
Tiago Novello, IMPA
Vinícius da Silva, PUC-Rio
Guilherme Schardong, U Coimbra
Hélio Lopes, PUC-Rio
Luiz Velho, IMPA

***Abstract*—*Neural fields* have emerged as a promising framework for representing different types of signals. This tutorial focus on the existing literature and shares practical insights derived from hands-on experimentation with neural fields, specifically in approximating implicit functions of surfaces. Our emphasis lies in strategies leveraging differential geometry concepts to enhance training outcomes and showcase applications within this domain.**

***Index Terms*—neural fields, differential geometry**

I. Introduction

A *neural field* (NF) is a (coordinate-based) *neural network* parametrizing a *smooth* signal. Similarly to classic function approximation techniques, such as *radial base functions* (RBF), NFs provide a high-quality smooth approximation for discrete data. Moreover, NFs allow the calculation in closed form of *higher-order derivatives* through *automatic differentiation* present in modern machine learning frameworks. Schirmer et al. [1] presented a survey of NF's techniques to represent surfaces parametricaly and implicitly. In this work, we approach the implicit representation in which an NF is used to represent an implicit function of a surface S. Precisely, the surface S is approximated as the zero-level set $f^{-1}(0)$ of a NF $f : \mathbb{R}^3 \to \mathbb{R}$.

Two main implicit surface representations are *unsigned* distance functions (UDF) and *signed* distance functions (SDF). UDFs are used to represent surfaces that are not "watertight" [2]. Existing methods for converting UDFs into explicit meshes often suffer from either high computational costs or compromised accuracy. In the SDF case, the distance value can be positive or negative, representing whether a point is inside or outside the underlying compact surface. If the distance is positive (negative), it means the point is outside (inside) the surface. A distance value of zero indicates that the point lies on the surface. SDFs are solutions of the *Eikonal equation* $\|\nabla f\| = 1$ which is used to regularize the underlying implicit function. By leveraging the properties of SDFs, various algorithms can be applied, such as *sphere marching* or *level-set methods*, to render or manipulate surfaces.

This paper explores seminal strategies considering state-of-the-art shape representations using NFs. We present practical approaches to model and train NFs to fit SDFs. For this, we employ an *implicit regularization* based on the Eikonal equation to force the NF to be a SDF and a *normal alignment* constraint to regularize the orientation near the zero-level set.

We also present methods for enhancing the training performance of NFs by employing a strategy to dynamically sample data points during training. Finally, we utilize the fact that an NF is a smooth function to estimate the curvature measures of its level sets analytically (no discretization is needed).

979-8-3503-3872-0/23/$31.00 ©2023 IEEE

II. Signed distance functions and implicit representations

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

In this work, we parametrize the function $f : \mathbb{R}^3 \to \mathbb{R}$ by a coord-based neural network (NF). Thus, the zero-level set $f^{-1}(0)$ of the NF represents a surface. For such a (neural) surface to be regular we force f to be a SDF by asking it to satisfy the Eikonal equation. Solving this equation reveals that $\langle \nabla f, N \rangle = 1$ on S, indicating that ∇f aligns with the normals N of S. Examples of NFs include SIREN [3] and IGR [4].

A. Classic approaches

Radial basis functions (RBFs) [5] is a classical method which can be used to approximate the SDF of a surface S from a sample $\{p_i, f_i\}$ 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 $s(p_i) = f_i$. The *radiological function* $\phi : \mathbb{R}^+ \to \mathbb{R}$ is a real function and p_i are the centers of RBF [6]. Observe that the RBF representation depends on the data since its interpolant s depends on p_i.

Poisson surface reconstruction [7] 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: $\Delta f = \nabla \cdot V$.

B. Coordinate-based network approaches

1) Implicit Geometric Regularization: Implicit geometric regularization (IGR) [4] is a technique used to improve the quality and smoothness of NFs. The goal is to compute the parameters θ of a NF $f_\theta(x)$ by forcing it to fit the SDF of a surface S. IGR aims to refine and regularize the shape

2023 36th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI) | 979-8-3503-3872-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/SIBGRAPI59091.2023.10347177

defined by the zero-level set of an implicit function, making it more visually pleasing and mathematically well-behaved. 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). IGR drives the optimization methods to reach a plausible interpretation for the learning and favors smooth and natural zero-level sets.

Let $\mathcal{X} = \{p_i\}_{i \in I} \subset \mathbb{R}^3$ be a point cloud with normals $\mathcal{N} = \{N_i\}_{i \in I}$. IGR computes the parameters θ of a *multilayer perceptron network* (MLP) $f_\theta : \mathbb{R}^3 \to \mathbb{R}$ to fit the SDF of a surface. The proposed loss function has the form

$$l(\theta) = l_{\mathcal{X}}(\theta) + \lambda \mathbb{E}_p \big(\|\nabla f_\theta(p)\| - 1 \big)^2 \tag{1}$$

where $\lambda > 0$ is a parameter, $\|.\|$ is the l_2 norm, and

$$l_{\mathcal{X}}(\theta) = \frac{1}{|I|} \sum_{i \in I} \big(|f_\theta(p_i)| + \|\nabla f_\theta(p_i) - N_i\| \big) \tag{2}$$

which encourages f to vanish on $\mathcal{X}$ and ∇f to be close to the normals $\mathcal{N}$.

In Equation 1, the second term, known as the Eikonal term, ensures that the gradients ∇f have a 2-norm equals one. The implicit geometric regularization occurs due to the fact that the loss function used during training 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 as a result of the neural network structure and the optimization process [4], [8]. Figure 1 shows some level sets of MLPs trained using IGR.

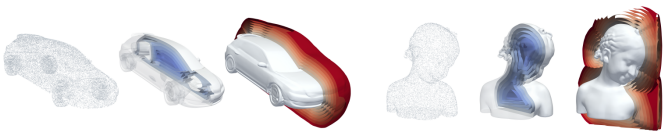

Fig. 1. Level sets of MLPs trained with IGR method [4]

2) Sinusoidal Implicit Networks: Sitzmann et al. [3] propose to leverage periodic activation functions for implicit neural representations and demonstrate that these networks are ideally suited for representing complex natural signals and their derivatives. As in IGR [4], we can use Siren to estimate an SDF to a plausible 3D surface. SIRENs can be used for high-quality reconstruction of objects and 3D scenes where it can be used in order to approximate a sampled implicit function $f : \mathbb{R}^3 \to \mathbb{R}$. SIREN has important properties that are suitable for reconstructing signals, where it has a simple architecture and uses the sine as a periodic activation [1], [3]:

$$f_\theta(p) = W_n \circ f_{n-1} \circ f_{n-2} \circ \cdots \circ f_0(p) + b_n, \tag{3}$$

$$f_i(p_i) = \sin(W_i \cdot p_i + b_i), \tag{4}$$

where the function $f_i : \mathbb{R}^{N_i} \to \mathbb{R}^{N_{i+1}}$ is the ith layer of the network. This map is obtained by applying the sine to each coordinate of the affine map given by the linear transformation $W_i : \mathbb{R}^{N_i} \to \mathbb{R}^{N_{i+1}}$ translated by $b_i \in \mathbb{R}^{N_{i+1}}$. The linear operators W_i can be represented as matrices and b_i as vectors. Therefore, the union of their coefficients corresponds to the coefficients θ of the SIREN function f_θ.

The SIREN can recover an SDF from a point cloud and surface normals by solving the Eikonal equation, a first-order PDE [3]. As we show in figure 2 given only the point cloud and surface normals of a 3D object, this approach can reproduce fine details of the object's surface. However as we will see in later sections, we can use the principal directions and curvatures of a shape to improve the reconstruction resulting in smooth and high-fidelity representation, avoiding noise regions and spurious objects.

Fig. 2. Shape representation. SIREN fit signed distance functions parameterized by implicit neural representations directly on point clouds.

3) Geometry Processing with Neural Fields: NFGP [9] 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) Physics-Informed Neural Networks: Physics-Informed Neural Networks (PINNs) [10] usually refer to modeling the solution of partial differential equations (PDEs) using artificial neural networks. PDEs play a crucial role in several scientific problems, such as the boundary conditions in fluid simulations or even quantum mechanics and reaction–diffusion systems. Physics-informed neural networks (PINNs) combine neural networks with known physical laws. They learn from data while respecting the fundamental principles of physics. PINNs

are useful when traditional physics-based models become computationally expensive or impractical due to complexity or limited data. The authors address two types of problems: finding solutions based on data and discovering partial differential equations using data. Depending on the characteristics of the data, it can be used to model two types of algorithms: continuous time models and discrete-time models. Continuous time models are effective in approximating spatiotemporal functions with high efficiency using limited data. On the other hand, discrete-time models enable the use of highly accurate implicit Runge-Kutta time-stepping schemes with an unlimited number of stages. While both PINNs and neural representations have the potential to solve problems related to partial differential equations, their core focuses differ. Neural implicit representations mainly target geometry and shape reconstruction in 3D contexts, whereas PINNs are designed for modeling physical processes. Nevertheless, there are instances where these concepts can merge. For instance, neural implicit representations might be incorporated into a PINN framework to depict intricate shapes that influence a physical phenomenon [11]. Such integration would unite the geometric strengths of neural implicit methods with the physics-driven essence of PINNs, enhancing precision in simulations and predictions within specific applications and it remains an open problem of research.

C. Exploring Differential Geometry in Neural Implicits

Exploring Differential Geometry in Neural Implicits, as known as I3D [6], 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.

III. Neural field framework

Several techniques exist for creating neural implicit representations. In this section, we delve into a training framework primarily influenced by the I3D approach [12]. In the upcoming sections, we will present two approaches for shaping the loss function of a neural network, drawing from principles of differential geometry, as well as an "biased" sampling technique aimed at enhancing convergence rates in our experiments.

A. Input data

Given an (oriented) point cloud $\{p_i, N_i\}$ sampled from a surface S, we can try to reconstruct the SDF of S. For this, points outside S may be added to the point cloud $\{p_i\}$. After estimating the SDF on the resulting point cloud we obtain a set pairs $\{p_i, f_i\}$ of points and the approximated SDF values.

B. Network architecture

Sines were first proposed as non-linearities for MLPs by [13]. Their main advantage compared to other functions is that their derivatives are a phase-shifted version of the original function. However, they were not promptly adopted due to instabilities and proneness to converge to local minima during training. Interest in their usage surfaced with the work of [14], where the authors performed a thorough study of sines as an activation function and correctly inferred that the main issue is in the initialization of the network's weights, although they still failed to find a proper solution for the instability issue. Sitzmann et al. [3] proposed a weight initialization technique that drastically mitigated this instability, leading to the proper implementation of sines as activation layers for MLPs.

We assume the neural function $f_\theta : \mathbb{R}^3 \to \mathbb{R}$ to be MLP

$$f_\theta(p) = W_n \circ f_{n-1} \circ f_{n-2} \circ \cdots \circ f_0(p) + b_n \qquad (5)$$

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_θ [6].

C. Loss function

In this context, we investigate a loss functional $\mathcal{L}$, which is composed of three components: $\mathcal{L}_{\text{Eikonal}}$, $\mathcal{L}_{\text{Dirichlet}}$, and $\mathcal{L}_{\text{Neumann}}$. These components are employed to train neural implicit functions effectively [3], [12].

Consider a compact surface S in $\mathbb{R}^3$ with its Signed Distance Function (SDF) denoted by $f : \mathbb{R}^3 \to \mathbb{R}$. We aim to find an *unknown* neural implicit function $f_\theta : \mathbb{R}^3 \to \mathbb{R}$ by training the parameters θ. To do this, we minimize the loss function given by Equation (6), which enforces f_θ to be a solution of the 3D Eikonal equation.

$$\mathcal{L}(\theta) = \underbrace{\int_{\mathbb{R}^3} \left|1 - \|\nabla f_\theta\|\right| dp}_{\mathcal{L}_{\text{Eikonal}}} + \underbrace{\int_S |f_\theta| dS}_{\mathcal{L}_{\text{Dirichlet}}} + \underbrace{\int_S 1 - \left\langle \frac{\nabla f_\theta}{\|\nabla f_\theta\|}, N \right\rangle dS}_{\mathcal{L}_{\text{Neumann}}}. \qquad (6)$$

Here, $\mathcal{L}_{\text{Eikonal}}$ encourages f_θ to be the SDF of a set $\mathcal{X}$ by ensuring that $\|\nabla f_\theta\| = 1$, $\mathcal{L}_{\text{Dirichlet}}$ encourages $\mathcal{X}$ to contain S, and $\mathcal{L}_{\text{Neumann}}$ asks for the alignment between ∇f_θ and the normal field of S.

Typically, an additional term is added to Equation (6) to penalize points outside S, forcing f_θ to be the SDF of S (i.e.,

$\mathcal{X} = S$). In practice, we extended $\mathcal{L}_{\text{Dirichlet}}$ to consider points outside S by using an approximation of the SDF of S.

We need to define the signed distance constraints. We need to sample points $\{p_i\}_{i=1}^n$ being the vertices of a triangulation T of S. W can replace $\mathcal{L}_{\text{Dirichlet}}$ by the following equation [12]:

$$\widetilde{\mathcal{L}}_{\text{Dirichlet}}(\theta) = \frac{1}{n}\sum_{i=1}^{n} |f_\theta(p_i)|. \tag{7}$$

Equation 7 forces $f_\theta = f$ on $\{p_i\}$.

Also, to avoid the neural network to generate spurious components we can enhance $\widetilde{\mathcal{L}}_{\text{Dirichlet}}$ by incorporating off-surface points. To achieve this, we consider the point cloud $p_i i = 1^{n+k}$, consisting of the n vertices of T along with a sample of k points in $\mathbb{R}^3$ outside the surface S. We extend the constraint as follows:

$$\widetilde{\mathcal{L}}_{\text{Dirichlet}}(\theta) = \frac{1}{n+k}\sum_{i=1}^{n+k} |f_\theta(p_i) - f(p_i)| \tag{8}$$

The proper signed distance function, is defined by, during the training of the network f_θ, an approximation of the Signed Distance Function (SDF) f for surface S. To achieve this, we utilize a point-sampled surface, consisting of n points $\{p_i\}$ and their corresponding normals $\{N_i\}$. The absolute value of f is approximated as follows:

$$|f(p)| \approx \min_{i \le n} \|p - p_i\| \tag{9}$$

The sign of $f(p)$ 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 [12]. Therefore, we can estimate the sign of $f(p)$ 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 $\{p_i\}$ [12] By employing the eikonal approach and incorporating model curvatures, we can explore an implicit regularization strategy instead of relying on a simple loss function.

The on-surface constraint $\int 1 - \langle \nabla f_\theta, N\rangle \, dS$ ensures that the gradient of f_θ aligns with the normals of the surface S.

To extend this constraint further, we seek to match the shape operators of $f_\theta^{-1}(0)$ and S. This involves aligning their eigenvectors and matching their eigenvalues, leading to the following expression:

$$\int_S \sum_{i=1,2,3} \Big(1 - \langle (e_i)_\theta, e_i\rangle^2 + |(\kappa_i)_\theta - \kappa_i|\Big) dS, \tag{10}$$

where $(e_i)_\theta$ and $(\kappa_i)_\theta$ represent the eigenvectors and eigenvalues of the shape operator of $f_\theta^{-1}(0)$, and e_i and κ_i refer to the eigenvectors and eigenvalues of the shape operator of the surface S. By incorporating this constraint, we enhance the regularization and ensure alignment between the two shape operators, contributing to the overall performance of the method.

We use the square of the dot product because the principal directions do not consider vector orientation. As the normal is one of the shape operator eigenvectors associated with the zero eigenvalue, Equation (10) simplifies to:

$$\int_S 1 - \left\langle \frac{\nabla f_\theta}{\|\nabla f_\theta\|}, N \right\rangle dS + \int_S \sum_{i=1,2} \Big(1 - \langle (e_i)_\theta, e_i\rangle^2 + |(\kappa_i)_\theta - \kappa_i|\Big) dS \tag{11}$$

The first integral in Equation (11) coincides with $\mathcal{L}_{Neumann}$. In the second integral, the term $1 - \langle (e_i)_\theta, e_i\rangle^2$ enforces the alignment between the principal directions, and $|(\kappa_i)_\theta - \kappa_i|$ demands the matching of the principal curvatures. Requesting alignment between $(e_1)_\theta$ and e_1 automatically ensures alignment between $(e_2)_\theta$ and e_2 since the principal directions are orthogonal.

D. 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 $m = p_1 m + p_2 m + p_3 m = 10000$, we can allocate $p_i m$ 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 3, 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.

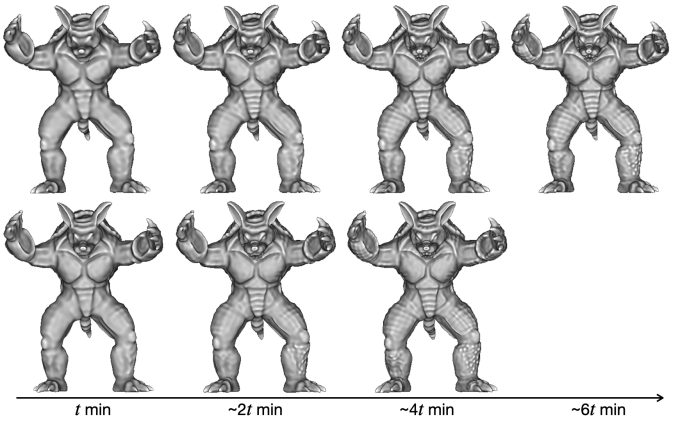

Fig. 3. Neural implicit surfaces approximating the Armadillo model. The columns indicate the zero-level sets of the neural implicit functions after 29, 52, 76, and 100 epochs of training. Line 1 shows the results using minibatches sampled uniformly in V. Line 2 presents the results using the adapted sampling of minibatches with 10% / 70% / 20% of points with low/medium/high features.

IV. Applications

Neural implicit representation 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. Also, the method can accurately represent geometric details with precision. In Figure 4, 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 Bunny's ear, but the reconstructed neural Bunny fixes this and provides a more detailed representation. We adopt the same architecture as SIREN [3] and running it on a computer with an AMD Ryzen 7 5700G processor, 16GiB of memory, and an NVIDIA GeForce RTX 2060 with 6GiB of memory, the model took 1.32 seconds to run the inference.

A. Sphere tracing

One application for neural implicit representations is to use neural networks in real-time rendering applications such as Sphere-tracing. To do so, the I3D [12] method provides a robust SDF approximation even compared with RBF. Figure 5 gives a visual evaluation presenting a sphere tracing of the zero-level of this method. Since it has a good SDF approximation, the algorithm is able to ray-cast the surface with precision avoiding spurious components.

B. 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 [6].

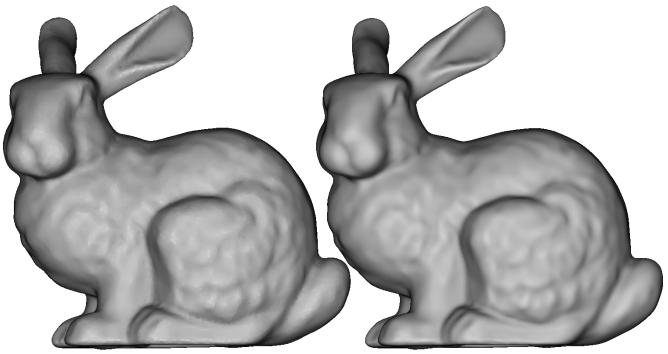

Fig. 4. The Bunny represented by the original triangular mesh and a implicit representation.

Fig. 5. 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.

The study of discrete variations of triangle mesh normals is an important topic in discrete differential geometry and it is very helpful in neural fields applications, especially when representing 3D shapes. These variations are represented by a discrete shape operator. 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. [6], [15]–[17] 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 6, 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 7 we show the principal curvatures and directions calculated for the Dragon model.

V. Conclusion

In this paper, we introduced 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.

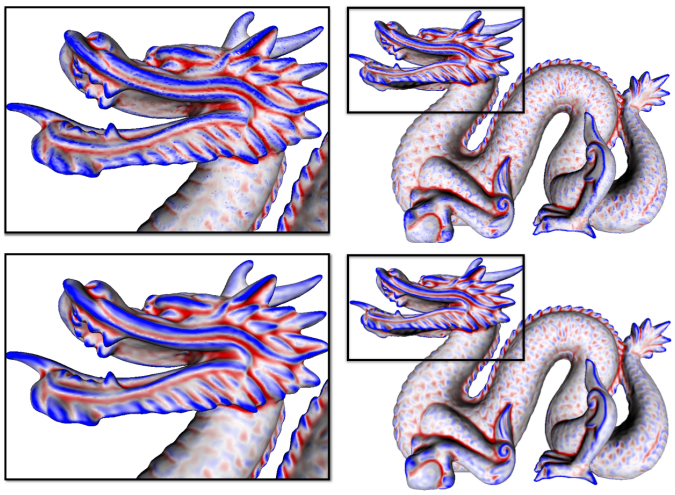

Fig. 6. Gaussian and mean curvatures of the Dragon neural surface. The surface points were computed using sphere tracing and its analytic curvatures using `PyTorch` framework.

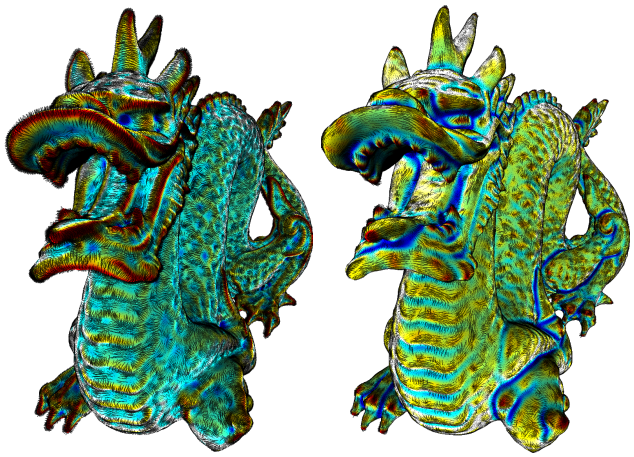

Fig. 7. Principal curvatures and directions overlaid on the Dragon.

Our exploration included notable contributions from papers such as Siren [3] and IGR [4] within this domain. Our focus was on issues related to noisy and spurious data in neural implicit representations.

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 data's discrete curvatures could enhance training by targeting points with more comprehensive geometric data in minibatch sampling.

In summary, this paper aimed to uncover strategies, such as sampling approaches, to accelerate training convergence. 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] L. Schirmer, G. Schardong, V. da Silva, H. Lopes, T. Novello, D. Yukimura, T. Magalhaes, H. Paz, and L. Velho, "Neural networks for implicit representations of 3d scenes," in *2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2021, pp. 17–24.

[2] B. Guillard, F. Stella, and P. Fua, "Meshudf: Fast and differentiable meshing of unsigned distance field networks," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part III*. Springer, 2022, pp. 576–592.

[3] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[4] A. Gropp, L. Yariv, N. Haim, M. Atzmon, and Y. Lipman, "Implicit geometric regularization for learning shapes," *arXiv preprint arXiv:2002.10099*, 2020.

[5] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. 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*, 2001, pp. 67–76.

[6] T. Novello, G. Schardong, L. Schirmer, V. da Silva, H. Lopes, and L. Velho, "Exploring differential geometry in neural implicits," *Computers & Graphics*, vol. 108, pp. 49–60, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849322001649

[7] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, 2006.

[8] B. Neyshabur, "Implicit regularization in deep learning," *arXiv preprint arXiv:1709.01953*, 2017.

[9] T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler, "Neural geometric level of detail: Real-time rendering with implicit 3d shapes," *arXiv preprint arXiv:2101.10994*, 2021.

[10] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[11] Z. Fang and J. Zhan, "A physics-informed neural network framework for pdes on 3d surfaces: Time independent problems," *IEEE Access*, vol. 8, pp. 26 328–26 335, 2019.

[12] T. Novello, G. Schardong, L. Schirmer, V. da Silva, H. Lopes, and L. Velho, "Exploring differential geometry in neural implicits," *Computers & Graphics*, vol. 108, pp. 49–60, 2022.

[13] A. Lapedes and R. Farber, "Nonlinear signal processing using neural networks: Prediction and system modelling," *IEEE international conference on neural networks*, 6 1987. [Online]. Available: https://www.osti.gov/biblio/5470451

[14] G. Parascandolo, H. Huttunen, and T. Virtanen, "Taming the waves: sine as activation function in deep neural networks," 2017. [Online]. Available: https://openreview.net/forum?id=Sks3zF9eg

[15] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, "Discrete differential-geometry operators for triangulated 2-manifolds," in *Visualization and mathematics III*. Springer, 2003, pp. 35–57.

[16] D. Cohen-Steiner and J.-M. Morvan, "Restricted delaunay triangulations and normal cycle," in *Proceedings of the nineteenth annual symposium on Computational geometry*, 2003, pp. 312–321.

[17] G. Taubin, "Estimating the tensor of curvature of a surface from a polyhedral approximation," in *Proceedings of IEEE International Conference on Computer Vision*. IEEE, 1995, pp. 902–907.