Geometry Processing with Neural Implicit Fields

Anonymous Author(s)
Affiliation
Address
email

Abstract

Most existing geometry processing algorithms use meshes as the default shape representation. Meshes, however, are hard to optimize for topology changes and usually require remeshing when undergoing large deformation. This paper instead proposes the use of *neural implicit fields (NIFs)* for geometry processing. NIFs can compactly store complicated shapes without spatial discretization. Moreover, NIFs are infinitely differentiable, which allows them to be optimized for objectives that involve higher-order derivatives. This paper develops loss functions and architectures to perform shape filtering and deformation with NIFs. We provide experimental results that showcase the applicability of NIFs to these tasks.

1 Introduction

In many graphics applications, users may want to edit/manipulate digital shapes using just a few clicks, such as making a character bow by dragging the head downwards. Such manipulation from sparse input requires geometry processing algorithms. Most of these algorithms use polygonal meshes to represent shapes [10]. Polygonal meshes were originally created for researchers as a representation of real world shapes that they can both interpret and manipulate [13, 48, 72]. However, developing algorithms that automatically manipulate meshes is often difficult since they involve discretizing the surface. For example, changing the topology of a mesh, such as turning a sphere into a torus, will break such discretization and require additional repair procedures such as remeshing [2, 3], Given that shape editing today is increasingly performed by algorithms, it seems worthwhile to search for a shape representation that is more amenable to automatic geometry processing.

An alternative to the polygonal mesh is an implicit representation, in which the surface is represented by a level set of a field: $\{x|f(x) = c\}$ [20, 35, 59]. Since it's easy to change shape topology using implicit representations, people have applied them for geometry processing tasks such as shape merging [8, 47, 50]. These works store implicit fields using voxels or octrees, which introduce memory-intensive spatial discretization. Recent research addresses this by using continuous neural networks to represent implicit fields [17, 41, 42, 52]. These *neural implicit fields (NIF)* possess several advantages in addition to the merits inherited from implicit representation: they are compact to store [21, 41] and can produce high-quality continuous surfaces at arbitrary resolutions [41, 52]. The community has achieved compelling results using NIFs in a variety of applications [43, 60, 68, 78], but their usefulness for geometry processing tasks is still somewhat under-explored. In this paper, we ask: *Can geometry processing be done entirely using NIFs?*

Providing an affirmative answer to this question is challenging because geometry processing tasks are typically underconstrained: user input is very sparse. We therefore need to leverage priors that characterize how a natural surface behaves. Such surface priors are usually instantiated by minimizing energy functionals that describe physical properties of surfaces using operators from differential geometry. For example, one can encourage surfaces to be smooth by minimizing the curvature, which can be measured by the Laplace-Beltrami operator. However, these operators require a parametrization of the surface, which is not available for level sets of an NIF. Mesh algorithms

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

usually approximate these geometric operators using the geodesic neighbors of the surface point. This allows the algorithms to focus on intrinsic geometric properties that are independent of shape parameterization. But it is difficult to obtain geodesic neighbors in an NIF since the surface of interest is encoded implicitly by a set of points that evaluate to the same field value.

We posit that computing these geometric operators in NIFs requires a fundamentally different approach. We leverage the facts that NIFs are designed to be infinitely differentiable, and that it is possible to optimize loss functions that involve higher-order derivatives [58, 62, 69]. The differentiability of the field function allows us to approximate the local surface of the level set using the derivatives of the underlying field. Furthermore, we can evaluate intrinsic geometry properties of the level set, such as curvature, using only the field derivatives [35, 50]. This allows us to develop loss functions that describe surface priors such as elasticity or rigidity. The infinite differentiability of NIFs makes it possible to optimize these loss functions using gradient descent methods. Thus, unlike mesh-based geometry processing algorithms that approximate these objectives using the vertices of the mesh, we can directly optimize these objectives in terms of the derivatives of the implicit field.

Our formulation provides a proof of concept that geometry processing can be done entirely using NIFs. In particular, we focus on two tasks: shape filtering (e.g., smoothing and sharpening) and topology-preserving deformations. These two tasks are not only essential for downstream applications, but also bring to the fore known challenges associated with implicit representations. We first tackle sharpening and smoothing, and show how they can be done by optimizing a loss function based on geometric properties computed via NIF derivatives. To achieve shape deformation, we propose to warp the NIF using a deformation field modeled by an invertible neural network [7]. With this deformation field, we derive a training objective that models implicitly surfaces as elastic shells.

We compare our method with well-established mesh-based baselines. The results show that our method produces more natural shapes without the need to discretize the surface. We hope that our work can inspire future generations of geometry processing algorithms using NIFs.

2 Related Work

This paper builds on two bodies of work: polygon mesh geometry processing and neural implicit representations. We will focus on prior work on two geometry processing tasks: shape filtering and topology-preserving deformation. Please refer to Botsch et al. [10] for further reading.

Mesh smoothing (and sharpening). The goal of mesh smoothing is to remove high-frequency noise and produce a smooth surface. Noise removal can be achieved by Fourier analysis [70, 73] or by modeling surface motion with a diffusion equation [23]. Smoothness can be promoted by minimizing energy functionals inspired by differential operators [46, 67, 76]. Sharpening is the inverse of smoothing. In this paper, we adopt the goal of smoothing from prior work [18, 70] and design a corresponding differentiable objective for NIFs.

Implicit field smoothing. Implicit representations can be smoothed by evolving the level set according to its curvature normal [45, 50]. Many network architectures, such as ReLU MLP, bias toward smoothness when modeling an implicit field [57, 62, 69]. Regularization objectives and initialization schemes have been proposed to obtain smooth level sets from scanned surface points [5, 27]. These works mostly use smoothness as a prior when training NIFs. Our work formulates an objective that allows smoothing or sharpening an NIF as desired.

Mesh deformation. Mesh deformation is done by either deforming the shape (i.e. vertices) [9, 64] or the space [6, 33, 44, 61]. These algorithms usually draw inspiration from physics and differential geometry to develop energy functions that encourage natural deformation [65, 66, 71]. They also linearize such objectives so that they can be solved efficiently using linear solvers [9]. Recent research also applies deep learning to optimize for non-linear deformation losses [28, 31, 74, 75, 77]. Deforming a mesh, however, requires estimating the deformation objectives with spatial discretization and maintaining vertex connectivity in a way that preserves topology [38]. We deform NIFs with an invertible deformation field to circumvent these issues.

Implicit field deformation. Prior works have studied deformation of implicit fields for tasks like physics simulation [35, 47, 50, 51]. These works use voxels or octrees to represent implicit fields,

while our paper advocates for using NIFs for deformation. There is some past work on deforming NIFs to match a target shape [11, 16, 34, 49, 54] or image [39, 53, 56] without any correspondences between the two. In contrast, we perform such deformation with very few localized user-provided correspondences, but without any other information about the target shape. Our setting is very convenient for artists, but it requires the algorithm to provide a strong surface prior due to the lack of a densely specified target state. Other prior works on editing NIFs are *trained* on a dataset of shapes from a particular object category [22, 25, 26, 29, 79]. Since surface priors are specified implicitly by the dataset, it's unclear how to apply these methods to out-of-distribution shapes. In contrast, our paper requires no training dataset and instead enforces priors inspired by more general physical properties such as smoothness and elasticity. Remelli et al. [58] developed a way to differentiate iso-surface extraction, which allows deforming NIFs using objectives carried over from mesh-based algorithms. Our method avoids iso-surface extraction entirely and formulates differentiable deformation objectives directly on the implicit field.

3 Computing Surface Properties using NIF

In this section, we will discuss how to compute surface properties using NIF and establish notation for the following sections. Let $\partial\Omega$ be a surface enclosing the region Ω. A signed distance field (SDF) for surface $\partial\Omega$ is defined as $d(\mathbf{x}) = s(\mathbf{x}) \min_{\mathbf{y}\in\partial\Omega} \|\mathbf{x} - \mathbf{y}\|$. The sign function $s(\mathbf{x})$ evaluates to -1 if $\mathbf{x} \in \Omega$ and 1 otherwise. We will assume that an NIF approximate an SDF whose zero-isosurface represents the surface of interest. For an NIF f, we denote its zero-isosurface using $\mathcal{M}_f$. This section summarizes how to compute surface normals and curvatures from NIFs. This is well-understood and we summarize this here for convenience [35, 50, 55].

Surface normal. The surface normal is the vector perpendicular to the tangent plane and describes the local orientation of the surface. For an SDF, the outward oriented surface normal of its level set is the field gradient: $n_d(\mathbf{x}) = \nabla_{\mathbf{x}} d(\mathbf{x})$. It can be shown that the SDF's gradient norm is always one: $\|\nabla_{\mathbf{x}} d(\mathbf{x})\| = 1$. This property can be used to ensure that the field remains a valid SDF throughout any manipulation [4, 27]. The closest point in $\partial\Omega$ from $\mathbf{x}$ can be found by $\mathbf{x} - d(\mathbf{x})n_d(\mathbf{x})$. This property can be used to sample points from the isosurface without creating a mesh [5, 12]. Finally, since the tangent plane is perpendicular to the surface normal, we can project vectors onto the tangent plane by subtracting their projection onto the normal using the following projection matrix $\mathbf{x}$: $\mathbf{P}_d(\mathbf{x}) = \mathbf{I} - n_d(\mathbf{x})n_d(\mathbf{x})^T$. This projection matrix allows us to characterize the tangent plane, e.g., to measure how much a given deformation stretches the tangent direction [32].

Curvature. Intuitively, curvature describes how much a surface deviates from a plane. This can be captured by the total derivative of the surface normal, which is also known as the shape operator: $\mathcal{S}_D(\mathbf{x}) = \mathcal{D}n_d(\mathbf{x})$. For an SDF, the normal itself is given by the derivative of the field, thus the shape operator for an SDF is the Hessian of the field function: $H_d(\mathbf{x}) = \mathcal{D}^2 d(\mathbf{x})$. The shape operator can be used to capture many different notions of curvature, including mean curvature (half the trace of $\mathcal{S}_D(\mathbf{x})$, denoted by $\bar{\kappa}$), Gaussian curvature (the determinant of $\mathcal{S}_D(\mathbf{x})$), and principal curvatures (the eigenvalues of $\mathcal{S}_D(\mathbf{x})$). We will use these to define objectives that smooth or sharpen the surface.

4 Shape Smoothing and Sharpening

In this section, we will show that shape smoothing and sharpening can be done directly on NIFs without producing meshes. In our setup, the input shape is represented as the zero-isosurface $\mathcal{M}_F$ of NIF F. The algorithm needs to output NIF G_θ whose zero-isosurface $\mathcal{M}_{G_\theta}$ satisfies two goals. First, $\mathcal{M}_{G_\theta}$ should preserve the global structure of $\mathcal{M}_F$. Second, the surface $\mathcal{M}_{G_\theta}$ should make desired changes to curvature of shape $\mathcal{M}_F$.

To achieve these goals, we first instantiate G_θ using the same architecture and parameter values as F. Then we optimize G_θ using the following objective:

$$\mathcal{L}_f(\theta) = \int_{\mathbf{x}\in U} |G_\theta(\mathbf{x}) - F(\mathbf{x})|^2 d\mathbf{x} + k_g \left(\|\nabla_{\mathbf{x}} G_\theta(\mathbf{x})\| - 1\right)^2 d\mathbf{x} + \int_{\mathbf{x}\in V_\tau} k_c \left(\kappa_{G_\theta}(\mathbf{x}) - \beta\kappa_F(\mathbf{x})\right)^2 d\mathbf{x}.$$

The first integral encourages the network G_θ to preserve the original shape. The second term regularizes G_θ to remain a valid SDF by enforcing that the norm of the gradient is 1 (i.e. the Eikonal constraint [4, 27]). The integration is over the region U where the original NIF F is supervised over. The final term aims to smooth or sharpen the surface by increasing or decreasing the curvature κ. If we set $\beta < 1$, then the curvature of the output surface κ_{G_θ} will decrease, resulting in a smoother shape. $\beta > 1$ will lead to surfaces with higher curvature details and will thus sharpen shapes. In this paper, we will use mean curvature since it's easy to compute with NIFs: $\kappa_f(\mathbf{x}) = \operatorname{tr} \mathcal{D}(n_f(\mathbf{x}))$.

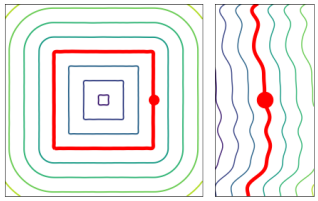

Figure 1: L: SIREN learned to fit the SDF of a square; R: Zooming in on the surface.

One challenge is that the computation of curvature can be very noisy for NIFs using periodic activations. Figure 4 shows how an ostensibly smooth isosurface learned through SIREN [62] is actually quite rough when zoomed in. The curvature evaluated on such a rough surface can be too noisy to be used for training. To alleviate this issue, we only compute the curvature mismatch in areas where the curvature of the level set is less than a certain threshold τ. Formally, we define this area as $V_\tau = \{\mathbf{x} \in U | \max(|\bar{\kappa}_{G_\theta}|, |\bar{\kappa}_F|) < \tau\}$. We use rejection sampling to sample points from V_τ when computing the loss during training.

5 Deformation

To deform a shape, the user will choose a set of deformation handles $\{\mathbf{h}_i\}_{i=1}^n$. For each handle $\mathbf{h}_i$, the user will specify a target location $\mathbf{t}_i$ describing where the handle will be dragged to. Users can set $\mathbf{t}_i = \mathbf{h}_i$ to enforce part of the surface to be unchanged. Given this input, there are two goals for the shape deformation algorithm: 1) satisfying user inputs, and 2) ensuring that the deformation resembles the natural behavior of real objects. In our setting, the input shape will be represented by the zero-isosurface of a neural network F. The algorithm must therefore output a NIF G_θ whose zero-isosurface represents the deformed shape that satisfies the aforementioned two goals.

One way to deform NIFs is to warp the coordinate space of the input network using a deformation field predicted by a neural network D_θ [22, 39, 53]. Under this framework, the output network can be defined as $G_\theta(\mathbf{x}) = F(D_\theta(\mathbf{x}))$. Deformation amounts to solving the following constrained optimization problem:

$$\min_\theta \ \mathcal{L}_{dfm}(G_\theta, F, D_\theta) \quad \text{s.t. } D_\theta(\mathbf{t}_i) = \mathbf{h}_i, 1 \leq i \leq n, \tag{1}$$

where the objective $\mathcal{L}_{dfm}(G_\theta, F, D_\theta)$ measures how natural the deformation D_θ is and the constraints ensure that the deformation matches user input. Naively adapting the training objectives from mesh-based algorithms [65, 66] to NIF is challenging because they require dense correspondences between two surfaces and efficient access to geodesic neighbors.

In this section, we will show how to address these challenges. To obtain good correspondence between the original and deformed shapes, we predict the deformation field using an invertible neural network (Sec 5.1). We then develop loss functions to encourage natural deformations by modeling the shape's resistance to stretching and bending during deformation (Sec 5.2). Finally, we will show how to compute and optimize these objectives (Sec 5.3 and Sec 5.4).

5.1 Invertible Deformation Field

We parameterize D_θ using an invertible neural network. With this formulation, we create one-to-one correspondences between points on the deformed surface $\mathbf{x}$ and points on the input shape $D_\theta(\mathbf{x})$. The invertible deformation network is composed of a series of invertible residual blocks.

Invertible residual block. Prior works [7, 15] have shown that a sufficient condition for the residual block $f(x) = x + g(x)$ to be invertible is that the Lipschitz constant of function g is less than 1. The architecture of the network $g(x)$ is usually composed of spectral normalized linear layers and Lipschitz continuous non-linearities such as ELU [19]. The inverse of such residual block can be computed by finding the fixed point of function $y \mapsto y - g(x)$ [7]. Intuitively, deforming using one invertible residual block amounts to moving point x with deformation vector $g(x)$.

Lipschitz continuous positional encoding. Applying such architecture directly without modification fails to produce deformations with many different local rotations. Recent research suggests that periodic functions are essential for coordinate MLPs to predict complex signals [43, 62, 69]. Periodic functions are usually used as positional encoding or activations of the form $\sin(ax+b)$, where a controls the frequency of the activation. To use such periodic function as part of the invertible residual block without restricting the frequency, we normalize the output of the periodic function $\sin(ax+b)|a|^{-1}$, bringing its Lipschitz constant below 1. Formally, the positional encoding with normalized Lipschitz constant applied to one dimension is

$$\gamma_i(\mathbf{x}) = \frac{1}{\sqrt{2L+1}}\left(x_i,\ \frac{\cos(2^0\pi\mathbf{x}_i)}{2^0\pi},\ \frac{\sin(2^0\pi\mathbf{x}_i)}{2^0\pi},\ \ldots,\ \frac{\cos(2^L\pi\mathbf{x}_i)}{2^L\pi},\ \frac{\sin(2^L\pi\mathbf{x}_i)}{2^L\pi}\right). \tag{2}$$

We will apply such encoding to each of the dimensions of the input coordinate for the body of the invertible residual block: $R_\theta(\mathbf{x}) = \mathbf{x} + g_\theta\left(\frac{1}{\sqrt{d}}[\gamma_1(\mathbf{x}),\ \ldots,\ \gamma_d(\mathbf{x})]\right)$.

5.2 Implicit Thin Shell Loss

One way to ensure a natural-looking deformation is to assume that the surface behaves like an elastic thin shell [1, 65]. Classically, this is achieved by minimizing the thin shell energy. This is a combination of two terms: the stretching energy, which measures how the surface (and in particular the tangent planes) has been stretched, and the bending energy, which measures the degree to which the curvatures along different directions have changed. When the surface has a natural parameterization, this shell energy is written as $\int_\Omega k_s \left\|\mathbf{I}'-\mathbf{I}\right\|_F^2 + k_b\left\|\mathbf{II}'-\mathbf{II}\right\|_F^2 dudv$, where $\mathbf{I}$ and $\mathbf{II}$ are the first and second fundamental forms ($\mathbb{R}^{2\times 2}$ matrices), and $\|\cdot\|_F$ is the Frobenius norm. The first fundamental form describes lengths and areas, so the difference $\left\|\mathbf{I}'-\mathbf{I}\right\|_F^2$ measures stretching. The second fundamental form describes curvatures, so the difference $\left\|\mathbf{II}'-\mathbf{II}\right\|_F^2$ measures bending. k_s and k_b control the resistance to stretching and bending, respectively. Since surface parameterizations are not available for NIFs, we develop two new loss functions for measuring stretching and bending.

5.2.1 Stretching Loss

Stretching is characterized by the change of dot product in tangent space. The change of dot product affects norms of tangent vectors, as a result it affects the angles and area between them. Assume the surface is parameterized by $\mathbf{x}(u,v)$, then the dot product is captured by the first fundamental form, a $\mathbb{R}^{2\times 2}$ matrix defined as $\mathbf{I}(\mathbf{x}) = \mathbf{B}(\mathbf{x})^T\mathbf{B}(\mathbf{x})$ with $\mathbf{B}(\mathbf{x}) = [\mathbf{x}_u\ \mathbf{x}_v]$. The tangent vectors at point $\mathbf{x}$ can be written as $\mathbf{B}(\mathbf{x})[a\ b]^T$. The dot product between two tangent vectors $\mathbf{t}_1 = \mathbf{B}(\mathbf{x})[a_1\ b_1]^T$ and $\mathbf{t}_2 = \mathbf{B}(\mathbf{x})[a_2\ b_2]^T$ at $\mathbf{x}$ can be defined using the first fundamental form:

$$\mathbf{t}_1^T\mathbf{t}_2 = [a_1\ b_1]\mathbf{B}(\mathbf{x})^T\mathbf{B}(\mathbf{x})[a_2\ b_2]^T = [a_1\ b_1]\mathbf{I}(\mathbf{x})[a_2\ b_2]^T. \tag{3}$$

The change of tangent dot product can thus be measured by the difference between two fundamental forms $\mathbf{t}_1^T\mathbf{t}_2 - \mathbf{t}_1'^T\mathbf{t}_2' = [a_1\ b_1]\left(\mathbf{I}(\mathbf{x}) - \mathbf{I}(\mathbf{x})'\right)[a_2\ b_2]^T$. If matrix norm $\|\mathbf{I}(\mathbf{x}) - \mathbf{I}(\mathbf{x})'\|$ is 0, then there will be no stretching to all tangent vectors.

In our setting, we first assume the surface $\mathcal{M}_{G_\theta}$ can be parameterized by $\mathbf{x}(u,v)$ and surface $\mathcal{M}_F$ is parameterized by $\mathbf{y}(u,v) = D_\theta(\mathbf{x}(u,v))$. By chain rule we have $\mathbf{y}_u = \mathbf{J}_D(\mathbf{x})\mathbf{x}_u$ and $\mathbf{y}_v = \mathbf{J}_D(\mathbf{x})\mathbf{x}_v$. Then the difference between the first fundamental forms of these two surfaces is $\mathbf{I}_{G_\theta}(\mathbf{x}) - \mathbf{I}_F(\mathbf{y}) = \mathbf{B}(\mathbf{x})^T\left(\mathbf{I} - \mathbf{J}_{D_\theta}(\mathbf{x})^T\mathbf{J}_{D_\theta}(\mathbf{x})\right)\mathbf{B}(\mathbf{x})$. However, since the shape is represented with NIFs, we don't have access to parameterization $\mathbf{x}(u,v)$ to compute $\mathbf{B}(\mathbf{x})$. To circumvent this, we parameterize the tangent vector using the projection matrix $\mathbf{P}_f = \mathbf{I} - n_f n_f^T$. A tangent vector $\mathbf{t}_i$ at point $\mathbf{x} \in \mathcal{M}_{G_\theta}$ can be expressed as $\mathbf{P}_{G_\theta}(\mathbf{x})\mathbf{v}_i$ for some $\mathbf{v}_i \in \mathbb{R}^3$. This tangent vector will be transformed by the deformation field D_θ into $\mathbf{t}_i' = \mathbf{J}_{D_\theta}(\mathbf{x})\mathbf{t}_i = \mathbf{J}_{D_\theta}(\mathbf{x})\mathbf{P}_{G_\theta}(\mathbf{x})\mathbf{v}_i$. With this, we can compute the change of dot product as follows:

$$\mathbf{t}_1^T\mathbf{t}_2 - \mathbf{t}_1'^T\mathbf{t}_2' = \mathbf{v}_1^T\mathbf{P}_{G_\theta}(\mathbf{x})^T\left(\mathbf{I} - \mathbf{J}_{D_\theta}(\mathbf{x})^T\mathbf{J}_{D_\theta}(\mathbf{x})\right)\mathbf{P}_{G_\theta}(\mathbf{x})\mathbf{v}_2. \tag{4}$$

To minimize this change of dot product, we will minimize the matrix norm of $\mathbf{P}_{G_\theta}^T\left(\mathbf{I} - \mathbf{J}_{D_\theta}^T\mathbf{J}_{D_\theta}\right)\mathbf{P}_{G_\theta}$. Let $\mathcal{T} = \mathbf{J}_{D_\theta}\mathbf{P}_{G_\theta}$, then the stretch loss can be defined as

$$\mathcal{L}_s(G_\theta) = \int_{\mathbf{x}\in\mathcal{M}_{G_\theta}} \left\|\mathbf{P}_{G_\theta}(\mathbf{x})^T\mathbf{P}_{G_\theta}(\mathbf{x}) - \mathcal{T}(\mathbf{x})^T\mathcal{T}(\mathbf{x})\right\|_F^2 d\mathbf{x}. \tag{5}$$

5.2.2 Bending Loss

Bending can be characterized by the change of surface curvature (e.g. making the surface more or less curved). The shell energy characterizes the curvature using the second fundamental form matrix: $\mathbf{II}(\mathbf{x}) = \begin{bmatrix} \mathbf{x}_{uu} \cdot n(\mathbf{x}) & \mathbf{x}_{uv} \cdot n(\mathbf{x}) \\ \mathbf{x}_{uv} \cdot n(\mathbf{x}) & \mathbf{x}_{vv} \cdot n(\mathbf{x}) \end{bmatrix}$. $\mathbf{II}(\mathbf{x})$ describes the change of tangent dot product along the surface normal direction [14, 55]. Specifically, let $\mathbf{t}_1 = \mathbf{B}(\mathbf{x})[a_1\ b_1]^T$ and $\mathbf{t}_2 = \mathbf{B}(\mathbf{x})[a_2\ b_2]^T$ be two tangent vectors at $\mathbf{x}$. Consider the tangent dot product of the surface family $\mathbf{x}(u,v) + tn(u,v)$. Then the change of dot product along the surface normal direction will be $\frac{d}{dt}\mathbf{t}_1^T\mathbf{t}_2|_{t=0} = [a_1\ b_1]\mathbf{II}[a_2\ b_2]^T$. Intuitively, the larger this value is, the faster the surface deviates from the tangent plane when going along the surface normal direction, which means the surface has larger curvature. As a result, the difference of the second fundamental forms captures the change of curvature since $\frac{d}{dt}\left(\mathbf{t}_1^T\mathbf{t}_2 - \mathbf{t}_1'^T\mathbf{t}_2'\right)|_{t=0} = [a\ b]\left(\mathbf{II}(\mathbf{x}) - \mathbf{II}'(\mathbf{x})\right)[a\ b]^T$. If the norm of $\mathbf{II} - \mathbf{II}'$ is 0 at point $\mathbf{x}$, then the two surfaces will have the same curvature at that point.

Again, computing the second fundamental form directly assumes a surface parameterization $\mathbf{x}(u,v)$, which is not accessible when the surface is represented by a NIF. Since it's easy to compute the shape operator using NIFs, we will use the shape operator to measure the amount of bending. The second fundamental form is related to the shape operator by $\mathbf{II}(\mathbf{x}) = \mathbf{B}(\mathbf{x})^T\mathcal{S}(\mathbf{x})\mathbf{B}(\mathbf{x})$. We first reparameterize the tangent vectors using the projection matrix $\mathbf{t}_i = \mathbf{P}\mathbf{v}_i$ for some $\mathbf{v}_i \in \mathbb{R}^3$. Then the change of tangent vector along the surface normal direction after the reparameterization is $\frac{d}{dt}\mathbf{t}_1^T\mathbf{t}_2 = \mathbf{v}_1^T\mathbf{P}^T\mathcal{S}\mathbf{P}\mathbf{v}_2$. The tangent vector transformed after D_θ will be $\mathbf{t}_i' = \mathbf{J}_{D_\theta}\mathbf{P}\mathbf{v}_i = \mathcal{T}\mathbf{v}_i$. As a result, the change of tangent dot product after deformation is $\frac{d}{dt}\mathbf{t}_1'^T\mathbf{t}_2' = \mathbf{v}_1^T\mathcal{T}^T\mathcal{S}'\mathcal{T}\mathbf{v}_2$, where $\mathcal{S}'$ is the shape operator of the deformed surface. Putting these together, we can again compute the difference between the change of tangent dot products:

$$\frac{d}{dt}\left(\mathbf{t}_1^T\mathbf{t}_2 - \mathbf{t}_1'^T\mathbf{t}_2'\right) = \mathbf{v}_1^T\left(\mathcal{P}^T\mathcal{S}\mathcal{P} - \mathcal{T}^T\mathcal{S}'\mathcal{T}\right)\mathbf{v}_2. \tag{6}$$

With this, the bending loss can be set up to minimize the matrix norm:

$$\mathcal{L}_b(G_\theta) = \int_{\mathbf{x}\in\mathcal{M}_{G_\theta}} \left\|\mathcal{P}_{G_\theta}(\mathbf{x})^T\mathcal{S}_{G_\theta}(\mathbf{x})\mathcal{P}_{G_\theta}(\mathbf{x}) - \mathcal{T}(\mathbf{x})^T\mathcal{S}_F(\mathbf{y})\mathcal{T}(\mathbf{x})\right\|_F^2 d\mathbf{x}. \tag{7}$$

5.3 Sampling

Computing $\mathcal{L}_s$ and $\mathcal{L}_b$ during training requires approximating a surface integral of the form $\int_{\mathbf{x}\in\mathcal{M}_{G_\theta}} \mathcal{L}(\mathbf{x})d\mathbf{x}$ using Monte Carlo integration, which requires an efficient way to sample points uniformly from surface $\mathcal{M}_{G_\theta}$. While it's feasible to sample uniformly from a SDF, this is challenging to do with G_θ since there is no guarantee that G_θ remains a valid SDF during the course of training. Thanks to the invertibility of D_θ, we can apply change of variable $\mathbf{x} = D_\theta^{-1}(\mathbf{y})$ to the integration, where $\mathbf{y}$ are points on the input surface $\mathcal{M}_F$:

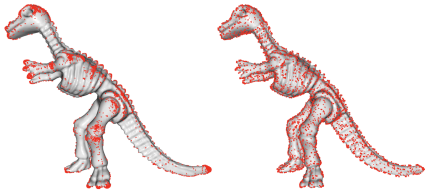

Figure 2: Sampling points from Dino. L: No rejection sampling. R: Ours.

$$\int_{\mathbf{x}\in\mathcal{M}_{G_\theta}} \mathcal{L}(\mathbf{x})d\mathbf{x} = \int_{\mathbf{y}\in\mathcal{M}_F} \mathcal{L}(\mathbf{x})\left|\det\left(\mathbf{J}_{D_\theta}(\mathbf{x})\mathbf{P}_{G_\theta}(\mathbf{x}) + n_F(\mathbf{y})n_{G_\theta}(\mathbf{x})^T\right)\right|^{-1} d\mathbf{y}. \tag{8}$$

$\mathbf{J}_{D_\theta}\mathbf{P}_{G_\theta} \in \mathbb{R}^{3\times3}$ maps tangent vector from $\mathcal{M}_{G_\theta}$ to $\mathcal{M}_F$. We apply the extension trick from Iglesias et al. [32] to create a transformation matrix whose change of volume equals to the surface area. This is achieve by adding the surface normal component $n_F n_G^T$ and keep it unchanged.

The right hand side of Equation 8 can be evaluated by sampling points uniformly from $\mathcal{M}_F$. One way to sample points from $\mathcal{M}_F$ is to run Langevin dynamics using the gradient of the field F [5, 12]:

$$\mathbf{x}_{t+1} = \tilde{\mathbf{x}}_t - F(\tilde{\mathbf{x}}_t)n_F(\tilde{\mathbf{x}}_t), \quad \tilde{\mathbf{x}}_t \sim \mathcal{N}(\mathbf{x}_t, \sigma\mathbf{I}), \tag{9}$$

with $\mathbf{x}_0$ initialized by sampling from a prior distribution. The mixing time of the procedure, however, is very long without appropriate coarse-to-fine annealing [63]. To alleviate this issue, we instantiate the sampling procedure with $\mathbf{x}_0$ sampled from a coarse shape. This can be done by first sampling uniformly from the bounded space and rejecting points that are too far away from the isosurface: $\mathbf{x}_0 \sim \{\mathbf{x} \in U(-1,1) | F(\mathbf{x}) < \tau_s\}$. We found that τ_s can be set to a relatively large number (e.g. about 0.1). This prevents high rejection rates without sacrificing sampling quality.

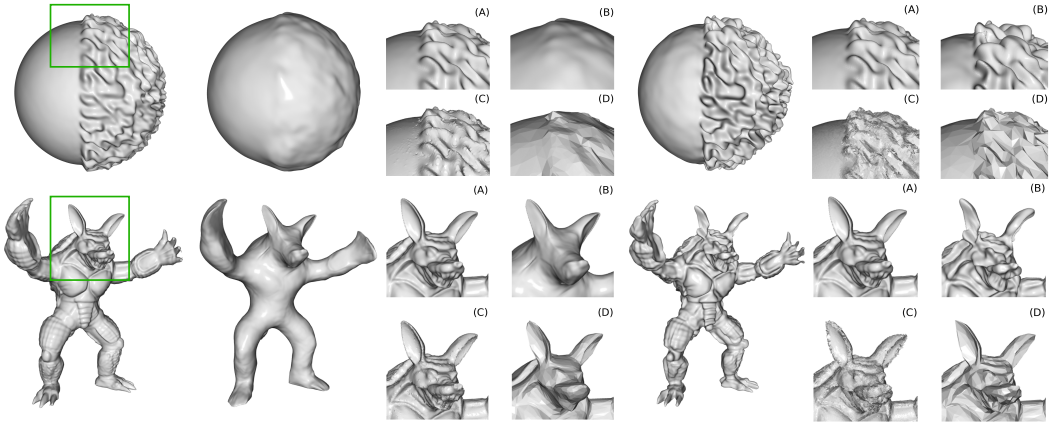

Figure 3: Smoothing and sharpening results. (A) zoomed-in input; (B) our method; (C) baseline without remeshing; (D) baseline with remeshing.

5.4 Optimization

Putting the losses together, we deform an NIF by solving the constrained optimization problem

$$\arg\min_{\theta} \mathcal{L}_{dfm}(G_\theta, F, D_\theta) = k_s \mathcal{L}_s(G_\theta) + k_b \mathcal{L}_b(G_\theta), \quad \text{s.t. } \mathcal{C}_i(D_\theta) = 0, \tag{10}$$

where $\mathcal{C}_i(D_\theta)$ is the constraint for the ith handle and k_s, k_b are hyperparameters that determine the material properties. We specify the constraints as $\mathcal{C}_i(D_\theta) = \max(\tau_c, \|D_\theta(\mathbf{h}_i) - \mathbf{t}_i\|)$, where $\tau_c > 0$ is the threshold within which we consider the constraint to be satisfied. We use the Augmented Lagrangian method [30] to minimize $\mathcal{L}_{dfm}$ while satisfying constraints $\mathcal{C}(D_\theta, \mathbf{h}_i, \mathbf{t}_i)$ for all i. Specifically, at each iteration we update θ to minimize the following objective for a single SGD step: $\mathcal{L}_{dfm}(G_\theta, F, D_\theta) + \sum_i \lambda_i \mathcal{C}_i(D_\theta) + \frac{\mu}{2} \sum_i \mathcal{C}_i(D_\theta)^2$. Then we update the Lagrange multiplier λ_i by $\lambda_i \leftarrow \lambda_i + \mu \mathcal{C}_i(D_{\theta_i})$. Hyperparameters are provided in the supplement.

6 Results

In this section, we will demonstrate the results of our methods to smooth, sharpen, and deform shapes. We will start by demonstrating the results on 3D shapes, then analyze the design choices of our architecture and training objectives on 2D examples. Code will be released upon acceptance.

Data preparation. For smoothing and sharpening, we follow prior works [18, 70] to use Armadillo [36] and a sphere with one half of it corrupted by Gaussian noise. The shapes for deformation are taken from Alexa et al. [1]. To create NIFs from these meshes, we follow the procedure of Park et al. [52] to compute ground-truth SDF for locations sampled within $[-1, 1]^3$. We then fit a SIREN [62] to the ground-truth SDF to generate our initial NIF.

Baseline. Our main baselines use mesh-based geometry processing algorithms to edit NIFs. Specifically, we first use marching cubes [40, 41] to extract a mesh from the input NIF. We then run a mesh-based geometry processing algorithm to process the mesh. To produce an NIF as output, we then rerun our data preparation pipeline to turn the processed mesh back into an NIF.

6.1 Smoothing and Sharpening

We will first show that we can smooth or sharpen the shape represented in an NIF without meshing using the algorithm described in Section 4. In our setting, the smoothing algorithms aim at producing a shape that resembles the input shape overall but with a smoother surface. We compare our algorithm to two baselines. The first baseline directly applies the shape preserving smoothing algorithm developed by Taubin [70] to the mesh extracted from the NIF. The second baseline first simplifies the extracted mesh by quadratic decimation [24] and then applies the filtering algorithms. The results are shown in Figure 3. The first baseline fails to smooth or sharpen the surface appropriately. It only modifies the surface with high frequency noise. While the second baseline is able to filter the

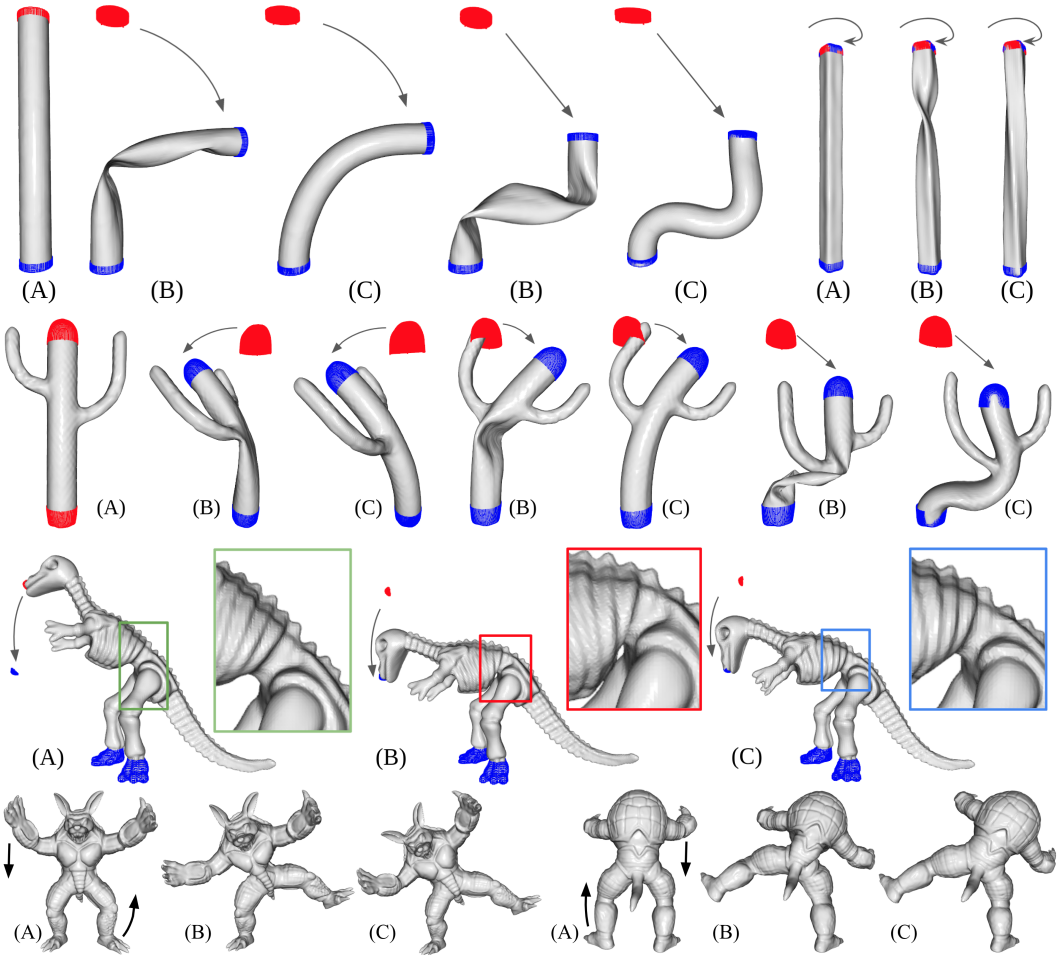

Figure 4: Deformation results. (A) Input shape. (B) Baseline. (C) Ours. Red points are user specified handles $\mathbf{h}_i$. Blue points denote user specified target points $\mathbf{t}_i$. First row: single rotation or translation on simple shapes. Second and third rows: single rotation or translation on more complex objects. Fourth row: multiple edits on complex objects.

surface correctly, it introduces discretization artifacts due to the decimation process. This suggests it is challenging to filter the NIF surface with mesh processing algorithms since these algorithms are sensitive to the quality of surface discretization. Our algorithm, on the other hand, can produce good filtering results with no surface discretization.

6.2 Deformation

We follow Sorkine-Hornung and Alexa [65] to create a set of basic shape deformation operations to evaluate our algorithm. In this setting, the user will first specify a set of handles used for manipulation (shown in Figure 4 in blue). In general, the user can apply three basic types of operation on a handle: 1) make it static (i.e. no deformation), 2) translate it, or 3) rotate it around a center. Usually, the first operation is used in combination with the latter two to produce useful deformation. In this section, we first show how our algorithm deforms simple objects (e.g. a cylinder and a rectangle bar) when the user rotates or translates one end while fixing the other. We then extend this set of operations to shapes with more detail (e.g. Cactus, Armadillo, and Dino). Finally, we will test our algorithms when the user specifies both rotation and translations on handles.

The results are shown in Figure 4. Our algorithm is able to produce deformation results that are natural while satisfying the user intention. The baseline we compared to applies as-rigid-as-possible deformation [65, 80] on the mesh extracted from the input NIF. Applying ARAP directly on the

extracted mesh creates undesirable volume distortion. This artifact is partially due to the surface discretization, which doesn't agree with the assumptions made by the mesh algorithm [37]. Our algorithm doesn't suffer from such issues as we optimize the thin shell objective directly without discretizing the surface.

6.3 Analysis

In this section, we will evaluate the key design decisions for our deformation algorithm. We fist conduct an ablation study to remove key ideas to examine their contributions. We then provide analysis on loss functions and sampling schemes to show their effectiveness.

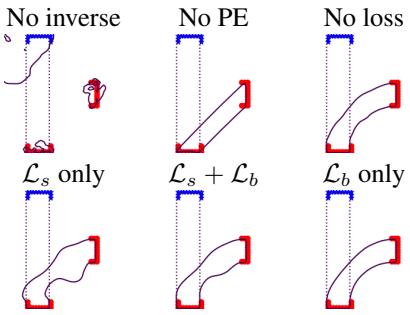

Figure 5: Blue points are handles; Red points are targets.

Architecture. We conduct ablation studies on two architecture choices and show the results in Figure 5. If we replace the invertible architecture with SIREN as done in Deng et al. [22], the deformation field will tend to break topology (*No inverse*). Removing the positional encoding will fail to create a complex field (*No PE*). Even if we only optimize for the constraint, our architecture can create a reasonable guess (*No loss*). This shows that our network architecture prioritizes natural deformation that preserves topology.

Losses. We compare models optimized for only the stretching loss $\mathcal{L}_s$ and bending loss $\mathcal{L}_b$ in Figure 5. Optimizing only for $\mathcal{L}_s$ produces a shape that best preserves surface area (or the perimeter in the 2D case), which explains the wiggling. Optimizing only for $\mathcal{L}_b$ produces a shape that tries to prevent wiggling of the surface, which changes the curvature of the surface. The user can choose appropriate k_s and k_b to produce deformation with resistance to both bending and stretching.

Sampling. Since computing our losses requires the ability to uniformly sample from the surface, we quantitatively evaluate the effectiveness of the sampling scheme described in Section 5.3. Specifically, we compare our sampling scheme with directly applying Langevin dynamics [12] (*Base* in Table 1). To do that, we compute the Chamfer Distance (CD) and Earth Mover Distance (EMD) between points sampled by the valuated algorithms and points sampled uniformly from the extracted mesh. Since larger surfaces can lead to larger CD and EMD values, we normalize the metrics by the values obtained by comparing two sets of uniform samples from the mesh. The results are shown in Table 1. Initializing $\mathbf{x}_0$ by rejection sampling drastically improve both CD and EMD compared to naively applying Langevin dynamics, at a minor cost in sampling time.

Table 1: Our sampling achieves better CD and EMD.

	Dino		Armadillo	
Metrics	Base	Ours	Base	Ours
CDr	1.54	**1.04**	1.36	**1.02**
EMDr	3.38	**1.15**	3.30	**1.08**
Time	**0.15**	0.21	**0.12**	0.18

7 Conclusion

Our work provides a proof of concept that geometry processing can be done entirely with neural implicit fields (NIFs) without discretizing the surface. We develop network architectures and derive training objectives for filtering and deforming shapes represented by NIFs. We demonstrate the advantages of using NIFs for geometry processing. We hope that our work can inspire a new generation of geometry processing algorithms using NIFs.

Limitations. A major limitation of our method is the slow optimization speed that hinders interactive editing. It's also difficult to ensure the output field remains an SDF, which may create obstacles to performing different editing operations on a shape consecutively. We believe this can be addressed in future work.

Societal impact. Our work can lead to more efficient geometry processing algorithms. Such algorithms can empower artists to generate creative content. Potential negative impact includes misuse of geometry processing algorithms to create offensive content.

References

[1] M. Alexa, D. Cohen-Or, and D. Levin. As-rigid-as-possible shape interpolation. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 2000. 5, 7

[2] Pierre Alliez, Mark Meyer, and Mathieu Desbrun. Interactive geometry remeshing. *ACM Transactions on Graphics (TOG)*, 21(3):347–354, 2002. 1

[3] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. Anisotropic polygonal remeshing. In *ACM SIGGRAPH 2003 Papers*, pages 485–493. 2003. 1

[4] Matan Atzmon and Y. Lipman. Sal: Sign agnostic learning of shapes from raw data. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2562–2571, 2020. 3, 4

[5] Matan Atzmon, Niv Haim, Lior Yariv, Ofer Israelov, Haggai Maron, and Y. Lipman. Controlling neural level sets. In *NeurIPS*, 2019. 2, 3, 6

[6] D. Bechmann. Space deformation models survey. *Comput. Graph.*, 18:571–586, 1994. 2

[7] Jens Behrmann, D. Duvenaud, and J. Jacobsen. Invertible residual networks. In *ICML*, 2019. 2, 4

[8] J. Bloomenthal and B. Wyvill. Introduction to implicit surfaces. 1997. 1

[9] Mario Botsch and O. Sorkine-Hornung. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14:213–230, 2008. 2

[10] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Bruno Lévy. *Polygon mesh processing*. CRC press, 2010. 1, 2

[11] Aljavz Bovzivc, Pablo Rodríguez Palafox, Michael Zollhofer, Justus Thies, Angela Dai, and Matthias Nießner. Neural deformation graphs for globally-consistent non-rigid reconstruction. *ArXiv*, abs/2012.01451, 2020. 3

[12] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 3, 6, 9

[13] Edwin Catmull. A system for computer generated movies. In *Proceedings of the ACM annual conference-Volume 1*, pages 422–431, 1972. 1

[14] Hunter Chase. Fundamental forms of surfaces and the gauss-bonnet theorem. 2012. 6

[15] Ricky T. Q. Chen, Jens Behrmann, D. Duvenaud, and J. Jacobsen. Residual flows for invertible generative modeling. *ArXiv*, abs/1906.02735, 2019. 4

[16] Xu Chen, Yufeng Zheng, M. Black, Otmar Hilliges, and Andreas Geiger. Snarf: Differentiable forward skinning for animating non-rigid neural implicit shapes. *ArXiv*, abs/2104.03953, 2021. 3

[17] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019. 1

[18] Ming Chuang and M. Kazhdan. Interactive and anisotropic geometry processing using the screened poisson equation. *ACM SIGGRAPH 2011 papers*, 2011. 2, 7

[19] Djork-Arné Clevert, Thomas Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv: Learning*, 2016. 4

[20] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996. 1

[21] T. Davies, Derek Nowrouzezahrai, and A. Jacobson. On the effectiveness of weight-encoded neural implicit 3d shapes. 2020. 1

[22] Yu Deng, J. Yang, and Xin Tong. Deformed implicit field: Modeling 3d shapes with learned dense correspondence. 2021. 3, 4, 9

[23] Mathieu Desbrun, M. Meyer, P. Schröder, and A. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99*, 1999. 2

[24] Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997. 7

[25] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7154–7164, 2019. 3

[26] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4857–4866, 2020. 3

[27] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Y. Lipman. Implicit geometric regularization for learning shapes. In *ICML*, 2020. 2, 3, 4

[28] Kunal Gupta and Manmohan Chandraker. Neural mesh flow: 3d manifold mesh generationvia diffeomorphic flows. *ArXiv*, abs/2007.10973, 2020. 2

[29] Zekun Hao, Hadar Averbuch-Elor, Noah Snavely, and Serge J. Belongie. Dualsdf: Semantic shape manipulation using a two-level representation. pages 7628–7638, 2020. 3

[30] M. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 4:303–320, 1969. 7

[31] Jingwei Huang, C. Jiang, Baiqiang Leng, Bin Wang, and L. Guibas. Meshode: A robust and scalable framework for mesh deformation. *ArXiv*, abs/2005.11617, 2020. 2

[32] José A. Iglesias, M. Rumpf, and O. Scherzer. Shape-aware matching of implicit surfaces based on thin shell energies. *Foundations of Computational Mathematics (New York, N.y.)*, 18:891 – 927, 2018. 3, 6

[33] A. Jacobson, I. Baran, Jovan Popovic, and O. Sorkine-Hornung. Bounded biharmonic weights for real-time deformation. *Communications of the ACM*, 57:99 – 106, 2014. 2

[34] Timothy Jeruzalski, Boyang Deng, Mohammad Norouzi, J. P. Lewis, G. Hinton, and Andrea Tagliasacchi. Nasa: Neural articulated shape approximation. In *ECCV*, 2020. 3

[35] M. W. Jones, J. A. Bærentzen, and M. Srámek. 3d distance fields: a survey of techniques and applications. *IEEE Transactions on Visualization and Computer Graphics*, 12:581–599, 2006. 1, 2, 3

[36] Venkat Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996. 7

[37] Zohar Levi and C. Gotsman. Smooth rotation enhanced as-rigid-as-possible mesh animation. *IEEE Transactions on Visualization and Computer Graphics*, 21:264–277, 2015. 9

[38] Y. Lipman, O. Sorkine-Hornung, D. Cohen-Or, D. Levin, Christian Rössl, and H. Seidel. Differential coordinates for interactive mesh editing. *Proceedings Shape Modeling Applications, 2004.*, pages 181–190, 2004. 2

[39] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan C. Russell. Editing conditional radiance fields. *ArXiv*, abs/2105.06466, 2021. 3, 4

[40] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87*, 1987. 7

[41] Lars M. Mescheder, Michael Oechsle, M. Niemeyer, S. Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4455–4465, 2019. 1, 7

[42] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Deep level sets: Implicit surface representations for 3d shape inference. *arXiv preprint arXiv:1901.06802*, 2019. 1

[43] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, J. T. Barron, R. Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 5

[44] Tim Milliron, R. Jensen, Ronen Barzel, and A. Finkelstein. A framework for geometric warps and deformations. *ACM Trans. Graph.*, 21:20–51, 2002. 2

[45] I. Mitchell. The flexible, extensible and efficient toolbox of level set methods. *Journal of Scientific Computing*, 35:300–329, 2008. 2

[46] Henry P. Moreton and C. Séquin. Functional optimization for fair surface design. *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, 1992. 2

[47] K. Museth, D. Breen, R. Whitaker, and A. Barr. Level set surface editing operators. *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002. 1, 2

[48] Martin E Newell. The utilization of procedure models in digital image synthesis. Technical report, UTAH UNIV SALT LAKE CITY SCHOOL OF COMPUTING, 1975. 1

[49] M. Niemeyer, Lars M. Mescheder, Michael Oechsle, and Andreas Geiger. Occupancy flow: 4d reconstruction by learning particle dynamics. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5378–5388, 2019. 3

[50] S. Osher and Ronald Fedkiw. Level set methods and dynamic implicit surfaces. In *Applied mathematical sciences*, 2003. 1, 2, 3

[51] S. Osher and N. Paragios. Geometric level set methods in imaging, vision, and graphics. In *Springer New York*, 2003. 2

[52] Jeong Joon Park, Peter R. Florence, J. Straub, Richard A. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174, 2019. 1, 7

[53] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020. 3, 4

[54] Sida Peng, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Animatable neural radiance fields for human body modeling. *ArXiv*, abs/2105.02872, 2021. 3

[55] H. Piaggio. Differential geometry of curves and surfaces. *Nature*, 169:560–560, 1952. 3, 6

[56] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and F. Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *ArXiv*, abs/2011.13961, 2020. 3

[57] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019. 2

[58] Edoardo Remelli, Artem Lukoianov, Stephan R. Richter, Benoit Guillard, Timur M. Bagautdinov, Pierre Baqué, and P. Fua. Meshsdf: Differentiable iso-surface extraction. *NeurIPS*, 2020. 2, 3

[59] A. Rosenfeld and J. Pfaltz. Sequential operations in digital picture processing. *J. ACM*, 13: 471–494, 1966. 1

[60] S. Saito, Zeng Huang, R. Natsume, S. Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2304–2314, 2019. 1

[61] T. Sederberg and S. Parry. Free-form deformation of solid geometric models. *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, 1986. 2

[62] V. Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *ArXiv*, abs/2006.09661, 2020. 2, 4, 5, 7

[63] Yang Song and S. Ermon. Generative modeling by estimating gradients of the data distribution. *ArXiv*, abs/1907.05600, 2019. 6

[64] Olga Sorkine. Differential representations for mesh processing. In *Computer Graphics Forum*, volume 25, pages 789–807. Wiley Online Library, 2006. 2

[65] O. Sorkine-Hornung and M. Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry Processing*, 2007. 2, 4, 5, 8

[66] O. Sorkine-Hornung, D. Cohen-Or, Y. Lipman, M. Alexa, Christian Rössl, and H. Seidel. Laplacian surface editing. In *SGP '04*, 2004. 2, 4

[67] O. Stein, E. Grinspun, M. Wardetzky, and A. Jacobson. Natural boundary conditions for smoothing in geometry processing. *ACM Transactions on Graphics (TOG)*, 37:1 – 13, 2018. 2

[68] Towaki Takikawa, Joey Litalien, K. Yin, K. Kreis, Charles T. Loop, Derek Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. *ArXiv*, abs/2101.10994, 2021. 1

[69] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, R. Ramamoorthi, J. T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *ArXiv*, abs/2006.10739, 2020. 2, 5

[70] G. Taubin. A signal processing approach to fair surface design. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995. 2, 7

[71] Demetri Terzopoulos, John C. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *SIGGRAPH '87*, 1987. 2

[72] Jason Torchinsky. The first real object ever 3D scanned and rendered was a VW beetle. *Jalopnik*, 2013. https://jalopnik.com/the-first-real-object-ever-3d-scanned-and-rendered-was-494241353. 1

[73] B. Vallet and B. Lévy. Spectral geometry processing with manifold harmonics. *Computer Graphics Forum*, 27, 2008. 2

[74] Weiyue Wang, D. Ceylan, R. Mech, and U. Neumann. 3dn: 3d deformation network. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1038–1046, 2019. 2

[75] Yifan Wang, Noam Aigerman, Vladimir G. Kim, S. Chaudhuri, and O. Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 72–80, 2020. 2

[76] William Welch and A. Witkin. Variational surface modeling. *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, 1992. 2

[77] Zhan Xu, Yang Zhou, E. Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. *ArXiv*, abs/2005.00559, 2020. 2

[78] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020. 1

[79] Zerong Zheng, Tao Yu, Qionghai Dai, and Y. Liu. Deep implicit templates for 3d shape representation. *ArXiv*, abs/2011.14565, 2020. 3

[80] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 8

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec 7, and supplementary materials will provide more details.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See sec 7 and supplementary for more details
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A] Our paper doesn't provide new theoretical results though we might build on some theoretical results that's not commonly known in the community.
   (b) Did you include complete proofs of all theoretical results? [N/A] No theorem needed to be proven, but we include important derivations in the supplementary.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplementary for detailed hyperparameters.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See supplementary for these details.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] The error bars/ std are reported in the supplementary for all tables in the main paper.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please see the supplementary materials.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See the data preparation section of Sec 6
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]