This CVPR 2021 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Neural Surface Maps

Luca Morreale[1] Noam Aigerman[2] Vladimir Kim[2] Niloy J. Mitra[1,2]

[1]University College London [2]Adobe Research

Abstract

Maps are arguably one of the most fundamental concepts used to define and operate on manifold surfaces in differentiable geometry. Accordingly, in geometry processing, maps are ubiquitous and are used in many core applications, such as paramterization, shape analysis, remeshing, and deformation. Unfortunately, most computational representations of surface maps do not lend themselves to manipulation and optimization, usually entailing hard, discrete problems. While algorithms exist to solve these problems, they are problem-specific, and a general framework for surface maps is still in need.

In this paper, we advocate considering neural networks as encoding surface maps. Since neural networks can be composed on one another and are differentiable, we show it is easy to use them to define surfaces via atlases, compose them for surface-to-surface mappings, and optimize differentiable objectives relating to them, such as any notion of distortion, in a trivial manner. In our experiments, we represent surfaces by generating a neural map that approximates a UV parameterization of a 3D model. Then, we compose this map with other neural maps which we optimize with respect to distortion measures. We show that our formulation enables trivial optimization of rather elusive mapping tasks, such as maps between a collection of surfaces.

1. Introduction

Maps are one of the most fundamental concepts in surface geometry: in differential geometry, a surface, *i.e.*, a 2-manifold, is usually (locally) defined as the image of a (non-degenerate) map

$$f : \mathbb{R}^2 \to \mathbb{R}^n.$$

Not surprisingly, maps are also used to define correspondences between different parts of surfaces in an atlas, to evaluate similarity between surface pairs, or across surface collections.

Accordingly, computing maps is central in most geometry processing tasks operating on surfaces. The ubiquitous concept of a UV map [54], mapping a surface into the plane, provides a local coordinate system on surfaces, and hence enables downstream tasks such as texturing, surface correspondence, remeshing, quad-meshing [11] to name only a few. Similarly, surface-to-surface maps [53] enable defining correspondences between surfaces, which are at the heart of shape analysis, transfer of properties, deformations, or defining morph sequences. Indeed, almost all shape processing tasks, including parameterization, surface correspondence, remeshing, and deep learning on surfaces, heavily rely on access to such surface maps.

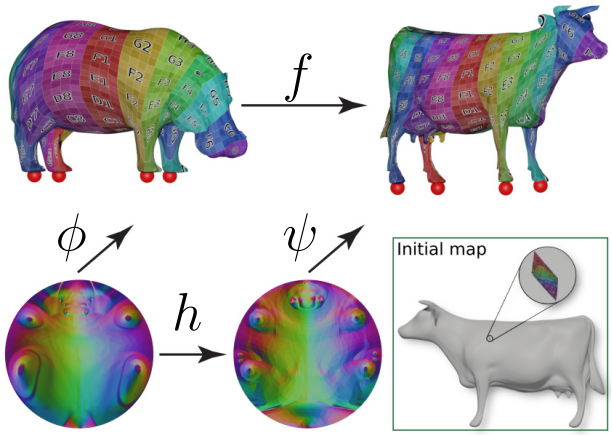


Figure 1. Two surfaces are respectively represented by two neural maps, ϕ and ψ, each mapping the unit square to 3D. A distortion-minimizing surface-to-surface map f is visualized by texture transfer. This surface map is achieved by a third neural map h between the square to itself, which yields the surface map by an implicit composition of the three neural maps. The distortion of the composed map f is trivially optimized by defining it as a loss in pytorch and optimizing with respect to h. Inset shows the initial random map of the hippo to the cow is a poor initialization, covering only a very tiny region of the cow model. Optimization results in a final map that covers the whole target surface, and reducing isometric distortion, resulting in an average Symmetric Dirichlet energy of 11 between these highly-nonisometric surfaces. As input, 4 keypoint constraints were used, each on one leg.

However, many of the tasks related to maps and their computation become extremely hard to handle when the target domain is a surface, *i.e.*, a 3D mesh ($n = 3$). This is due mostly to the fact that meshes are combinatorial representations, which in turn leads to a combinatorial representation of the surface maps, and taints the optimization task with a combinatorial nature as well. Although elegant solutions in the form of discrete differential geometry [50, 52], meshing invariant spectral analysis [42, 40], functional maps [45, 35] have been proposed to work around the combinatorial representation, the diverse choices and different data representations inhibit easy end-to-end optimization and adaptation outside the specialized geometry processing community.

As an example, consider the problem of computing a mesh-to-mesh mapping in which a continuous map from one surface to the other is computed: one needs to account for the image of each source vertex, which lands on a triangle of the other mesh, and the image of a source edge may span several triangles of the target; this leads to extensive bookkeeping, and any attempt to optimize, *e.g.*, the map's inter-surface distortion leads to combinatorial optimization of the choice of target triangle for each source vertex as in [53, 31]. An alternative is to optimize proxy maps into a common base domain [4] in the hope that the resulting surface-to-surface map will be optimized by proxy. Such an approach, however, does not yield surface maps that are even a local minimizer of the energy they set to minimize. This is particularly problematic when optimizing inter-surface maps across shape collections.

In this work, we consider *neural networks* as a parametric representation of both individual surfaces as well as inter-surface maps. Specifically, we consider networks with parameters θ that receive 2D points as input and output points either in 2D or 3D, $\phi_\theta : \mathbb{R}^2 \to \mathbb{R}^n$. While this definition is similar to, *e.g.*, AtlasNet [25], we do not aim to perform any learning task, and our network does nothing more than map 2D points with the aim of performing one task: approximate a single surface map $\phi_\theta \sim f : \mathbb{R}^2 \to \mathbb{R}^n$, so we can work with neural networks instead of with, e.g., mappings of triangular meshes.

Specifically, we use a map $\phi_\theta \sim f : \mathbb{R}^2 \to \mathbb{R}^3$ to directly characterize a given manifold shape (restricted to surface patches homeomorphic to a disc), and use another map $\psi_\beta \sim g : \mathbb{R}^2 \to \mathbb{R}^2$ to update the surface map by restricting movements on the underlying 2-manifold. These neural networks are, by construction, differentiable and composable with one another, hence they lend us a simple model for defining a differentiable algebra of surface maps, enabling us to compose maps with one another and optimize objectives directly over their composition, rather than propose approximations via intermediate proxy domains.

We employ this concept in two ways that build on top of one another: first, we revisit the differential-geometry definition of a surface as a map from 2D to 3D, by overfitting a neural network to a given UV parameterization computed via a standard parameterization algorithm, such as Tutte's embedding [59] or SLIM [51]. Two such maps, ϕ, ψ, are shown in Figure 1. This gives us a parametric, differentiable representation of the surface, from a canonical domain. Second, we compose the overfitted map with other maps, either to optimize the distortion of the map, or to compute distortion-minimizing maps between two or more surfaces. Figure 1 shows an example of a distortion-minimizing map f defined by composing h with ϕ, ψ.

We evaluate our method on a variety of triangular meshes with varying complexity and show their efficacy in computation of parameterizations, surface-to-surface distortion-minimizing mapping, and also for mapping across collections of shapes. We also provide comparison to baseline methods. In summary, our main contribution is introducing neural surface map as a novel representation and utilizing it towards addressing a variety of geometry processing applications. We particularly stress the modular nature of the representation that enables harnessing the power of current deep learning frameworks to solve many (classical) shape analysis tasks in a uniform framework. Code available from the project page `http://geometry.cs.ucl.ac.uk/projects/2021/neuralmaps/`.

2. Related Works

2.1. Surface maps

Mappings of surfaces (mostly, meshes) is an active research area. Usually, algorithms compute surface maps by striving for a specific type of map, such as a harmonic one [59, 48, 20, 23, 2]. In other cases the goal is to compute a map that minimizes or bounds some notion of distortion such as conformality - preservation of angles - [33, 30, 34], or isometry - preservation of local distances - [57, 4, 51, 61]. Many surface mapping algorithms focus specifically on parameterizations, see [21, 54] for a more detailed review. On the other hand, the task of computing mappings between two surfaces is a long-studied and notoriously hard problem. Our work uses the popular concept of a common base domain to which two surfaces are mapped, to define the surface map via the overlay of the two maps in the base domain, which can either be a coarse mesh [32, 53, 31, 12] or a planar domain [5, 61, 6, 3]. The resulting surface map can be optimized via direct mesh-mesh intersection at the price of yielding an optimization problem with a significant combinatorial part [53], due to the discreteness of the triangles. Otherwise, the properties of the resulting map are ignored at the hope that optimizing the maps into the common domain will be sufficient [4].

Soft notions of maps such as Functional Maps [46] enable a parametric definition of fuzzy maps which can be

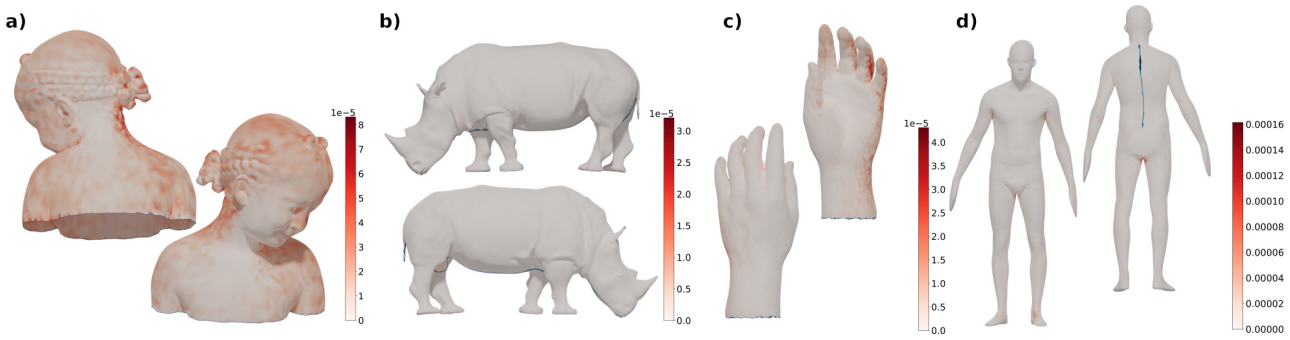


Figure 2. Parameterizations of the Bimba (a), rhino (b), Tosca hand (c) and FAUST (d) models represented via overfitted neural maps. The models are colored based on deviation from source models. Distortions, if any, introduced by the mappings are not considered at this stage.

used in a deep learning context [35, 19], however they define fuzzy correspondences and not a continuous surface map from one surface to another.

Cycle-consistency [44, 29] is a trait of maps between a collection of surfaces, that ensures that the set of surfaces are in global correspondence by ensuring that mapping a point across random shapes until reaching the source shape maps the point back to itself. We present the first method to minimize the distortion of a cycle-consistent collection of maps.

2.2. Neural surface representation

Atlas-based representations have been prevalent in geometrical deep learning, mostly focusing on generative and analysis tasks (e.g. segmentation), but not on raw representation of surface mapping.

Many works consider UV-mapping a 3D surface, rendering surface functions as a 2D image, and applying a neural network to the images [55, 41, 10, 27]. Representing surfaces as regularly-sampled image grids suffers from high distortions in the mapping, where large surface areas can be mapped to sub-pixel regions. Some works apply neural networks directly on meshes [28, 37]. Generative techniques, such as FoldingNet [63] and AtlasNet [25] propose to alleviate this issue by using a neural network to approximate the atlas map from 2D to 3D as a continuous function conditioned on a latent shape code. Many extensions have been proposed including regularizing differential properties of the mapping [9], optimizing for elementary shape of the atlas [18], and forcing surfaces to align with the level set of shape's implicit function [49].

While our network architectures are inspired by these prior techniques, as we also train a neural module that maps 2D points to mesh surfaces, they have very different underlying objectives. The aforementioned methods aim at training a network to reconstruct shapes, conditioned on a latent code. Our work is distinct from them, in that it focuses on considering each neural network as a unique, single surface map, and using this representation to solve classic geometry processing problems in surface mapping.

2.3. Neural shape representation

In addition to surface-based representations many alternatives have been used, such as voxels [38, 22, 13, 16], point clouds [1, 58], meshes [15], and implicit functions [47]. From these techniques only the neural implicit representations do not suffer from discretization artifacts, since they use neural modules to represent continuous functions, mapping a point in 3D to an occupancy value. Similarly to surface-based methods they aim to create a shared latent space for all shapes, and various extensions have been proposed, such as enforcing unit gradient to satisfy the Eikonal equation [24], and sign-agnostic version of this normalization [8, 7]. Littwin and Wolf in [36] introduce a meta-learning approach closely related to HyperNetworks [26] for implicit representations. A meta-model f regress weights θ_g for the implicit function g, which describes the signed distance field for a specific shape. Although the approaches introduced above are very successful, they focus on generalizing over a plethora of models rather than represent a single one. Hence, they all present artefacts and thus can only be used as a rough proxy to the actual geometry and cannot be regarded as neural maps in a strict sense.

Davies *et al.* [17] overfit neural networks to implicit fields of individual shapes, as a compact representation for their geometry. Implicit fields have also been used for multi-view reconstruction, where Sitzmann *et al.* [56] optimize a neural network that represents a single shape, with the loss that favors this representation to be consistent with observed views of the object. Following the trends from [60, 43], the proposed network learns to project the input into high-frequency features, hence learning in such domain. Although these techniques are similar to ours in that they use overfit networks to represent geometry, they focus on implicit surfaces and do not provide any mechanisms

for inter-surface mapping. In contrast, we encode explicit surfaces via neural maps, and demonstrate that these maps can be composited and used for inter-surface mapping problems.

Lastly, a work related to our data generation method, [62] suggest overfitting an atlas to a point cloud as a method for surface reconstruction. They however focus on the task of inferring the surface topology via overfitting, while we simply estimate a given map which already inholds the topological data with a neural network, and instead focus on the differentiability and composability properties of it, for geometry representation and for optimizing other maps composed with the parameterization.

3. Method

We now define neural surface maps and how to compute and optimize them.

3.1. Neural Maps

We use the term *neural surface map* to refer to any neural network considered as a function $\phi : \mathbb{R}^2 \to \mathbb{R}^n$, where the output dimension is 2 or 3. Indeed, this ensures the map's image is always a 3D surface, and, assuming the map is non-singular, also a 2-manifold.

Neural surface maps can be seen as an alternative method to represent a surface map that holds two main advantages: *differentiability* and ability to be *composed* with other neural maps. In short, this enables us to easily compose neural maps $\phi \circ \psi$, and define an objective over the composition $o(\phi \circ \psi)$ which can be differentiated and optimized via standard (deep learning) libraries and optimizers without the need to write tailor-made code to handle new objective, work with combinatorial mesh representations, or deal with the notoriously-hard map composition problem.

Furthermore, we can choose any size, architecture and activation functions for our networks, and, thanks to the universal approximation theorem [14], know there always exists a network capable of approximating a given surface function.

We obtain and manipulate neural surface maps via two processes – overfitting and optimization, which we detail next.

3.2. Overfitting Neural Surface Maps

Let $\Omega \subset \mathbb{R}^2$ be the unit circle. All our neural maps will make use of Ω as a canonical domain. Given any map $f : \Omega \to \mathbb{R}^n$, we can approximate it via a neural surface map ϕ by using black-box methods to train the neural network and *overfit* it to replicate f. Namely, we optimize the least-square deviation of ϕ from f and the surface normal

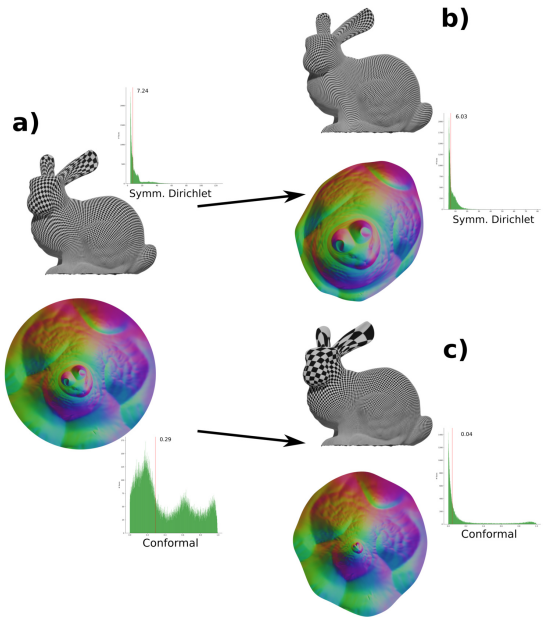


Figure 3. Free-boundary isometric (b) and conformal (c) parametrization of Stanford Bunny (a) model. Independently of the size of the model, our neural maps can parametrize the input mesh, represented as neural surface map, with very few parameters. Adding a constraint over the boundary shape is as simple as regularize the mesh boundary. Initial median Dirichlet energy $D_{iso} = 7.24$ is reduced to $D_{iso} = 6.03$; initial median conformal energy $D_{conf} = 1.29$ is reduced to $D_{conf} = 1.04$.

deviation, by minimizing the integrated error

$$\mathcal{L}_{\text{overfit}} = \int_{p\in\Omega} \|f(p) - \phi(p)\|^2 + \lambda_n \int_{p\in\Omega} \|n_{\phi p} - n_{fp}\|^2, \tag{1}$$

where $n_{\phi p}$ is the estimated normal at p, and n_{fp} is the ground truth normal. In case f is indeed a continuous map, such as a piecewise-linear map mapping triangles to triangles, we can optimize this objective by approximating the integral in Monte-Carlo fashion by summing the integrand over a random set of sample points. Namely, to use neural surface maps to represent surfaces, we first compute a ground truth map f by overfiting to a UV parameterization of the mesh into 2D, computed via any bijective parameterization algorithm of our choosing – in this paper, we show results with SLIM [51], by which we achieve an injective map of the mesh into $\Omega \subset \mathbb{R}^2$. We consider the inverse of this map, which maps Ω back into the 3D mesh $\mathcal{S}$, as our input $f : \Omega \to \mathcal{S}$, and overfit ϕ to it by minimizing Equation 1. Thus, we obtain a neural representation of the surface. More specifically, this is a *mapping* into the surface, endowed with specific UV coordinates, with point $\phi(x, y)$

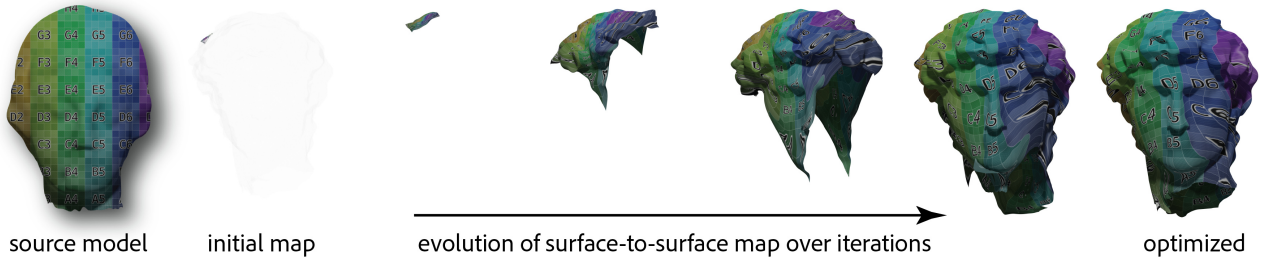


Figure 4. Evolution of surface to surface optimization between Igea and David. Final median Dirichlet energy $D_{iso} = 18.25$.

having UV coordinates x, y. Fig. 2 shows several examples of such overfitted neural maps and their faithfulness to the original geometry. Our method can faithfully represent smooth shapes as well as those having sharp edges. Note that we assume that the objects are or have been cut open to be homeomorphic to a disc.

Before progressing to discussing how can we compose maps and optimize them, we define the distortion measures we wish to optimize.

3.3. Surface Map Distortion

We wish to optimize several energies related to neural surface maps. Similarly to [9], for a neural map $\phi : \Omega \to \mathbb{R}^n$, we denote by $J_p\phi \in \mathbb{R}^{n\times 2}$ the matrix of partial derivatives at point $p \in \Omega$, called the *Jacobian* of ϕ. The Jacobian essentially quantifies the local deformation at a point. Letting $M_p = J_p^T J_p$, we subsequently can quantify the symmetric Dirichlet energy [51],

$$D_{\text{iso}} = \int_\Omega \text{trace}\left(M_p\right) + \text{trace}\left(\left(M_p + \varepsilon I\right)^{-1}\right) \qquad (2)$$

where I is the identity matrix, added with a small constant ϵ, set to 0.01, to regularize the inverse.

Likewise, we can define a measure of conformal distortion via

$$D_{\text{conf}} = \int_\Omega \left\| \frac{\text{trace}\left(M_p\right)}{\left\|M_p\right\|^2} M_p - I \right\|^2. \qquad (3)$$

We evaluate the integrals by random sampling of the function in the domain.

Next, we show how to define surface-to-surface maps via various compositions of the maps and optimize their distortion, in the pairwise and in the shape collection setting.

3.4. Geometry-preserving optimization via composition

Our basic representation of 3D geometries is, as discussed above, via an *overfitted* neural surface map $\phi : \Omega \to \mathbb{R}^3$ that approximates a given map f. We now treat ϕ as our de-facto representation of the geometry. Our goal is to optimize various properties relating to the surface map, without affecting the geometry. However, optimization of the map is not trivial since it will immediately change our 3D geometry. We propose a solution to completely avoid this issue, next.

Assume we are given a neural surface map representing some surface $\phi : \Omega \to \mathcal{S}$; we wish to optimize the distortion $D\left(\phi\right)$ of the map. It is immediate to optimize ϕ *itself* with respect to our differentiable notion of distortion, however that will cause the map to change, and thus its image, the 3D surface, will change and could, for instance, flatten to the plane. To overcome this, we suggest introducing another neural surface map $h : \Omega \to \Omega$. We can now define a new map, $\phi^h = \phi \circ h$. As long as we solely optimize h and ensure it is onto Ω, we are guaranteed that the image of ϕ^h is still the original image of ϕ, *i.e.*, respects the original surface.

We can now optimize the distortion of ϕ^h, by optimizing h and keeping ϕ fixed, thereby finding a map from Ω to $\mathcal{S}$ which is (at least a local) minimizer of the distortion measure of our choice:

$$\min_h D\left(\phi^h\right).$$

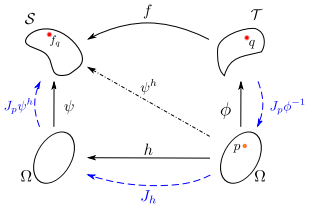
The distortion is a differentiable property of the map and hence is readily available, *e.g.*, via automatic differentiation. In fact, composition, and minimization of distortion can be achieved in a mere few lines of code in Pytorch.

We can now consider composing more than two of these maps, to enable maps into more intricate domains.

3.5. Compositing Neural Maps

Map composition via common domains. One of the many advantages of our representation's composability is to enable representing maps between a pair of surfaces, using the classic method of a common domain, as depicted in Figure 1: we posses two overfitted neural maps, $\phi, \psi : \Omega \to \mathbb{R}^3$, respectively representing two surfaces $\mathcal{S}, \mathcal{T}$, and we wish to define and optimize an inter-surface mapping between these two 3D surfaces, $f : \mathcal{T} \to \mathcal{S}$.

To address the above problem, we define as before a map $h : \Omega \to \Omega$ and the composition $\psi^h = \psi \circ h$. At first glance, it would seem that in this case, to map a point from $\mathcal{T}$ to $\mathcal{S}$, we will need to consider the map $\psi^h \circ \phi^{-1}$, which includes an inverse of the *entire* map, that is of course not readily tractable.



However, we can define the map f via the following simple definition: for any point $p \in \Omega$, f is implicitly defined as the map satisfying $f \circ \phi \triangleq \psi^h$, or in simple words: for any point $p \in \Omega$, f matches the image of p under ϕ^h with the image of p, mapped through h and then through ψ (refer to Figure 1 for an illustration). This definition is known as the *common domain* definition of a map and has been used in many works [32, 53, 31, 12, 5, 61, 6, 3]. It is easy to verify that this definition is identical to the one using the inverse, as long as the inverse exists, and can still provide a bijective map between the surfaces even in cases where it does not exist (cf., [61, 4]).

Computing distortion in the common domain. Even though f itself is not tangible for optimization, as it is implicitly defined by h, luckily the only differential quantity we need from f to compute the distortion, is the Jacobian of f, denoted $J_q f$ at point $q = \phi(p)$. Using basic differential calculus arithmetic, $J_q f$ can be derived to be exactly

$$J_q f = J_p \psi^h \left(J_p \phi\right)^{-1}, \tag{4}$$

which is composed of the Jacobian of ψ and the inverted Jacobian of ϕ at point p, both readily available. Hence to optimize the distortion of f, we can take (4), and plug it as the Jacobian used to define M in one of the distortion measures (2),(3), which we denote as $D(f)$.

Optimizing h for bijectivity. In order for h to indeed be a well-define surface map, it needs to map exactly bijectively (*i.e.*, 1-to-1 and onto) to the source domain of ψ, which is Ω. To ensure that, we only need to ensure that h has a positive-determinant Jacobian everywhere, and maps to the target boundary injectively. We optimize h to map the boundary onto itself, via the energy

$$B(h) = \int_{p \in \partial\Omega} \sigma\left(h(p)\right), \tag{5}$$

where σ is the squared signed distance function to the boundary of Ω. Note that the boundary map is free to slide along the boundary of Ω during optimization, enabling the boundary map to change. This is true for all points on the boundary, except those mapped to the four corners which are fixed to place and are essentially keypoint constraints between the two models.

Further, we also optimize h to encourage its Jacobian's determinant to be positive, via

$$G = \lambda_{\text{inv}} \int \max\left(-sign\left(|J_h|\right) exp\left(-|J_h|\right), 0\right). \tag{6}$$

Keypoint constraints. Lastly, in many cases, a sparse set of corresponding key points on the two surfaces are given, and it is required that the surface map f maps those points to one another. Given keypoints on $\mathcal{S}$, we can, in a preprocess before optimization, find their preimages in Ω, to get a set of points P s.t. $\phi(P_i)$ maps to the ith keypoint. We likewise can find the preimages of the keypoints from $\mathcal{T}$ and their preimages Q under ψ. If these key points are required to be mapped to one another between the two surfaces by f, we can achieve that by requiring $h(P_i) = Q_i$, which guarantees the induced f maps the points correctly. We optimize for that equality by reducing its least-squares error:

$$C(h) = \lambda_C \sum_i \left\| h(P_i) - Q_i \right\|_2^2. \tag{7}$$

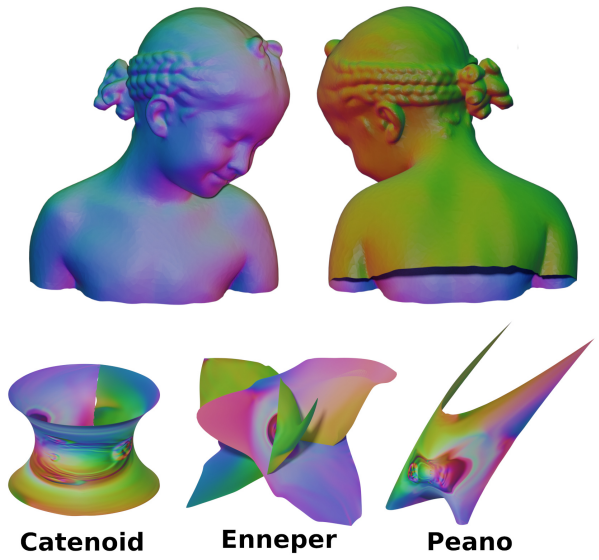
To facilitate the optimization, we apply a rotation, R, to the input of h. R is pre-computed from the landmarks.

Optimization for surface-to-surface maps. To compute the surface map, we optimize the distortion of f with respect to h, while ensuring h respects the mapping constraints

$$\min_h D(f) + C(h) + B(h) + G(h). \tag{8}$$

This yields a map h that maps onto the domain square, and represents a distortion-minimizing surface map f that maps the given sets of corresponding keypoints correctly, as shown for instance in Figure 1.

Cycle-consistent surface mapping. We also extend our method to discover inter-surface mapping among a collection of k surfaces $\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_k$ represented respectively via neural maps $\phi_1, \phi_2, ..., \phi_k$, we can define a *cycle consistent* [44, 29] set of surface maps by considering k additional neural maps, $h_i : \Omega \to \Omega$, define the composition $\phi_i^h = \phi_i \circ h_i$, and then define the surface-to-surface maps $F_{i \to j} : \mathcal{S}_i \to \mathcal{S}_j$ via $F_{i \to j} \circ \phi_i^h \triangleq \phi_j^h$. This naturally allows extracting a set of mutually consistent maps while additionally optimizing for (all pairs) surface-to-surface maps, see Figure 6. Note that achieving similar qualities via classic methods is significantly challenging, and to the best of our knowledge, while previous methods could compute cycle consistency, none could optimize for true surface-surface distortion minimization over the entire collection.



Catenoid $\mathbf{x}(u,v) := (\cosh(v/c)\cos u, \cosh(v/c)\sin u, v)$

Enneper $\mathbf{x}(u,v) := (u(1-u^2/3+v^2)/3, v(1-v^2/3+u^2)/3, (u^2-v^2)/3)$

Peano $\mathbf{x}(u,v) := (u, v, (2u^2-v)(v-u^2))$

Figure 5. Surface maps between a neural surface mapping representing the Bimba model, into several surfaces represented directly via analytic functions. Colors are based on the normals of Bimba model. Please refer tot he supplemental for further visualizations.

4. Experiments

We use our neural-mapping representation in the context of various mapping problems, such as surface parameterization, inter-surface mapping, and mapping a collection of shapes. See the supplementary material for more examples.

Neural Mapping. For all surfaces shown in this paper, we render the reconstructions obtained with our neural mapping representation. Note how our overfitting procedure is able to capture even very detailed features of the original shape with a high fidelity. Figure 2 illustrates the difference between our reconstruction and the input mesh (highlighted in red). There are minor discrepancies between the models in regions like hairs of the bust and paws of the rhino. We observe that our reconstructions tend to be slightly smoother than the original shapes due to the use of softplus.

Surface Parameterization. The main advantage of neural mapping is not in representing the surfaces, but in representing the mapping. We now take the map $\phi : \Omega \to \mathbb{R}^3$ from Figure 2, and introduce another map $h : \Omega \to \mathbb{R}^2$, where we don't constrain its output domain. Similarly to the discussion in Subsection 3.5, we can define the map f from the 3D model implicitly via as $f(\phi(p)) = h(p)$ for all $p \in \Omega$. We then minimize the isometry distortion of f (Eq. 2), using the method to extract the Jacobian discussed in Subsection 3.5. Note that this objective is different from the one that was used to produce ϕ, hence we undo the original parameterization's distortion by compositing the neural map with a newly optimized map in Figure 3. See the supplementary for more results.

In contrast to UV parameterizations of meshes, the complexity of our optimization for this composition is completely independent of the resolution of the geometry.

Surface-to-surface Maps. We can obtain a surface-to-surface map by compositing neural maps with a map between two atlases, as discussed in Subsection 3.5. In Figure 4, we show the evolution of the map during optimization. Note how despite significant geometric differences between surfaces, the result is a bijective, low-distortion mapping. Please see more such maps in the supplementary.

Composition with Analytical Maps. Our method can optimize an inter-surface map f from ϕ, ψ just as well when ψ is not a neural map, but rather an analytical mapping defining some surface. Indeed, only h itself is required to be neural in our formulation of surface-to-surface maps. In Figure 5, we show mappings of Bimba into three such analytical surfaces. In this case, we optimize the conformal distortion (3) of f. Please refer to the supplementary for further visualizations.

Cycle-consistent Mapping for Collections of Surfaces. Finally, we show that thanks to the compose-ability of neural surface maps, our method can be efficiently applied to cycle-consistent mapping problem for a collection of shapes. Furthermore, since we use a common domain, the maps are guaranteed to be cycle-consistent, as in [44, 29]. We minimize the isometric distortion of the surface-to-surface maps between all pairs of surfaces in a collection of three models, following the method discussed in Subsection 3.5. Figure 6 illustrates that we were able to obtain cycle-consistent low distortion maps between all shapes in the collection. We used one keypoint on the nose and shoulders of each model to ensure correct alignment. See the supplementary for more collection-maps.

Baseline comparison. To validate neural surface maps, we offer visual comparisons with the classic inter-surface method [53] and Mandad *et al.* [40]. Schreiner *et al.* fails to produce smooth maps while matching landmarks: respectively for the bust and animal shown in Fig. 7, [53] presents 8.58% and 8.54% triangles flips with a median $D_{iso} = 4.90$ and $D_{iso} = 7.00$. Similarly, Mandad *et al.* achieve a median $D_{iso} = 7146$, with 49.91% of flips, and $D_{iso} = 10669$ with 49.86% of flips, see Fig. 8. Note, [40] introduces dis-

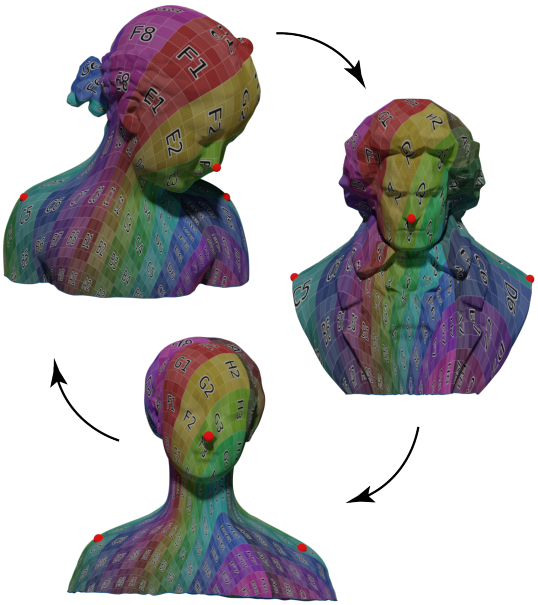
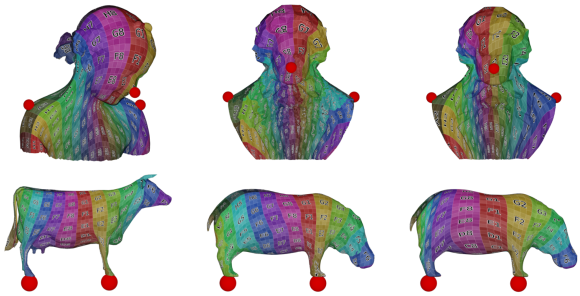


Figure 6. Collection mapping. We map from one model to the other through a neural map. We then minimize the distortion between each different model. Cycle consistency is ensured by construction.



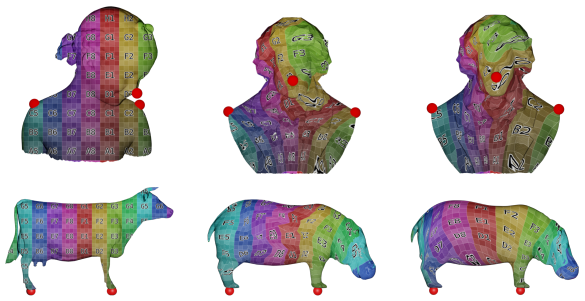
(a) source **(b)** [53] **(c)** ours

Figure 7. Comparison with inter-surface mapping [53].

continuities in the map, resulting in large distortion and misalignment. On the other hand, our method offer a continuous, properly aligned, map. Numerically, our map for busts exhibit $D_{iso} = 7.00$ with no triangle flips, $D_{iso} = 8.56$ and 0.03% flips for animal.

4.1. Implementation Details

In all our experiments, we use a neural network consisting of ten-layer residual fully-connected network, with 256 hidden units per layer, with a Softplus activation function. We use $\lambda_n = 0.01$, $\lambda_B = 10^6$, $\lambda_{inv} = 10^2$, $\lambda_C = 10^3$ in all experiments. We sample the initial mesh uniformly with 500k points. Since our goal is to fully-optimize the networks, they are trained until the gradient's norm drops below a threshold of 0.1. In all cases, we optimize the network with and RMSProp, and initialize the optimization



(a) source **(b)** [40] **(c)** ours

Figure 8. Comparison with state of art shape correspondence [40].

procedure with a learning rate of 10^{-4} and momentum 0.9, the step size is modulated with [39]. Similarly, maps used for surface mapping are four-layer fully-connected network of 128 hidden units, with Softplus. In general, overfitted networks converge in 3-7h based on the complexity of the model, while, surface-map and collection-map optimization take around 3h to reach a stable configuration.

5. Conclusions and Future Works

We introduced neural surface maps as a core representation for surfaces that is easily differentiable and composeable. Using the common domain approach, we can easily use these traits to optimize for different properties. Overfit to individual meshes allows encoding shapes as network weights, and subsequently optimize maps while keeping the surface approximation quality fixed. We demonstrated the universality of neural maps addressing a wide range of challenging classical tasks including parameterization, surface-to-surface distortion minimization, and extracting maps across a collection of shapes.

Our work has several limitations. For one, we only discussed representing disk-topology surfaces. Other topologies can be approached with cuts. Second, we relied on the assumption of h being bijective and mapping the keypoints correctly; in theory, we cannot guarantee that this requirement is upheld, however, in our experiments, it is rare for this condition to be violated.

We see many immediate uses to the differentiability and composability of our representation, such as applying differential geometry operators to the models as well as solving PDEs on them. Resorting to neural network generalization capabilities can bring large high-resolution dataset within our reach, exposing neural surface maps to applications like segmentation and classification.

Acknowledgements

LM thanks Manish Mandad for helping comparing with [40]. LM was partially supported by the UCL Centre for AI and the UCL Adobe PhD program.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas J Guibas. Learning representations and generative models for 3d point clouds. *ICML*, 2018. 3

[2] Noam Aigerman and Yaron Lipman. Orbifold Tutte embeddings. *ACM Transactions on Graphics*, 34(6):1–12, Nov. 2015. 2

[3] Noam Aigerman and Yaron Lipman. Hyperbolic orbifold tutte embeddings. *ACM Transactions on Graphics*, 35(6):1–14, Nov. 2016. 2, 6

[4] Noam Aigerman, Roi Poranne, and Yaron Lipman. Lifted bijections for low distortion surface mappings. *ACM Transactions on Graphics*, 33(4):1–12, July 2014. 2, 6

[5] Noam Aigerman, Roi Poranne, and Yaron Lipman. Lifted bijections for low distortion surface mappings. *ACM Trans. Graph.*, 33(4):69:1–69:12, July 2014. 2, 6

[6] Noam Aigerman, Roi Poranne, and Yaron Lipman. Seamless surface mappings. *ACM Transactions on Graphics (TOG)*, 34(4):72, 2015. 2, 6

[7] Matan Atzmon and Yaron Lipman. SAL: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020. 3

[8] Matan Atzmon and Yaron Lipman. SAL++: Sign agnostic learning with derivatives. *arXiv preprint arXiv:2006.05400*, 2020. 3

[9] Jan Bednarik, Shaifali Parashar, Erhan Gundogdu, Mathieu Salzmann, and Pascal Fua. Shape reconstruction by learning differentiable surface representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4716–4725, 2020. 3, 5

[10] Heli Ben-Hamu, Haggai Maron, Itay Kezurer, Gal Avineri, and Yaron Lipman. Multi-chart generative surface modeling. *SIGGRAPH Asia*, 2018. 3

[11] David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini, and Denis Zorin. Quad-mesh generation and processing: A survey. *Comput. Graph. Forum*, 32(6):51–76, 2013. 1

[12] Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. Markerless garment capture. *ACM Trans. Graph.*, 27(3):99:1–99:9, Aug. 2008. 2, 6

[13] André Brock, Theodore Lim, James M. Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *CoRR*, abs/1608.04236, 2016. 3

[14] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989. 4

[15] Angela Dai and Matthias Nießner. Scan2mesh: From unstructured range scans to 3d meshes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019. 3

[16] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017. 3

[17] Thomas Davies, Derek Nowrouzezahrai, and Alec Jacobson. Overfit neural networks as a compact shape representation, 2020. 3

[18] Theo Deprelle, Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Learning elementary structures for 3d shape generation and matching. *arXiv preprint arXiv:1908.04725*, 2019. 3

[19] Nicolas Donati, Abhishek Sharma, and Maks Ovsjanikov. Deep geometric functional maps: Robust feature learning for shape correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8592–8601, 2020. 3

[20] Michael Floater. One-to-one piecewise linear mappings over triangulations. *Mathematics of Computation*, 72(242):685–696, 2003. 2

[21] Michael S Floater and Kai Hormann. Surface parameterization: a tutorial and survey. *Advances in multiresolution for geometric modelling*, pages 157–186, 2005. 2

[22] Rohit Girdhar, David F. Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. *CoRR*, abs/1603.08637, 2016. 3

[23] Steven Gortler, Craig Gotsman, and Dylan Thurston. Discrete one-forms on meshes and applications to 3d mesh parameterization. *Computer Aided Geometric Design*, 2006. 2

[24] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. *arXiv preprint arXiv:2002.10099*, 2020. 3

[25] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018. 2, 3

[26] David Ha, Andrew Dai, and Quoc V Le. HyperNetworks. *arXiv preprint arXiv:1609.09106*, 2016. 3

[27] Niv Haim, Nimrod Segol, Heli Ben-Hamu, Haggai Maron, and Yaron Lipman. Surface networks via general covers. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 632–641, 2019. 3

[28] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. MeshCNN: A network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. 3

[29] Qi-Xing Huang and Leonidas Guibas. Consistent shape maps via semidefinite programming. In *Computer Graphics Forum*, volume 32, pages 177–186. Wiley Online Library, 2013. 3, 6, 7

[30] Michael Kazhdan, Jake Solomon, and Mirela Ben-Chen. Can mean-curvature flow be modified to be non-singular? In *Computer Graphics Forum*, volume 31, pages 1745–1754. Wiley Online Library, 2012. 2

[31] Vladislav Kraevoy and Alla Sheffer. Cross-parameterization and compatible remeshing of 3d models. *ACM Trans. Graph.*, 23(3):861–869, Aug. 2004. 2, 6

[32] Aaron W. F. Lee, David Dobkin, Wim Sweldens, and Peter Schröder. Multiresolution mesh morphing. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '99, pages 343–350, New

York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. 2, 6

[33] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM transactions on graphics (TOG)*, 21(3):362–371, 2002. 2

[34] Yaron Lipman. Bounded distortion mapping spaces for triangular meshes. *ACM Trans. Graph.*, 31(4):108:1–108:13, July 2012. 2

[35] Or Litany, Tal Remez, Emanuele Rodola, Alex Bronstein, and Michael Bronstein. Deep functional maps: Structured prediction for dense shape correspondence. In *Proceedings of the IEEE international conference on computer vision*, pages 5659–5667, 2017. 2, 3

[36] Gidi Littwin and Lior Wolf. Deep meta functionals for shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1824–1833, 2019. 3

[37] Hsueh-Ti Derek Liu, Vladimir G Kim, Siddhartha Chaudhuri, Noam Aigerman, and Alec Jacobson. Neural subdivision. *arXiv preprint arXiv:2005.01819*, 2020. 3

[38] Jerry Liu, Fisher Yu, and Thomas Funkhouser. Interactive 3d modeling with a generative adversarial network. *International Conference on 3D Vision (3DV)*, 2017. 3

[39] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 8

[40] Manish Mandad, David Cohen-Steiner, Leif Kobbelt, Pierre Alliez, and Mathieu Desbrun. Variance-minimizing transport plans for inter-surface mapping. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017. 2, 7, 8

[41] Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G. Kim, and Yaron Lipman. Convolutional neural networks on surfaces via seamless toric covers. *SIGGRAPH*, 2017. 3

[42] Simone Melzi, Jing Ren, Emanuele Rodolà, Abhishek Sharma, Peter Wonka, and Maks Ovsjanikov. ZoomOut: Spectral upsampling for efficient shape correspondence. 2

[43] 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. 3

[44] Andy Nguyen, Mirela Ben-Chen, Katarzyna Welnicka, Yinyu Ye, and Leonidas Guibas. An optimization approach to improving collections of shape maps. *Computer Graphics Forum*, 30(5):1481–1491, 2011. 3, 6, 7

[45] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: A flexible representation of maps between shapes. 31(4):30:1–30:11. 2

[46] Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (TOG)*, 31(4):1–11, 2012. 2

[47] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proc. CVPR*, pages 165–174, 2019. 3

[48] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2:15–36, 1993. 2

[49] Omid Poursaeed, Matthew Fisher, Noam Aigerman, and Vladimir G. Kim. Coupling explicit and implicit surface representations for generative 3d modeling. *ECCV*, 2020. 3

[50] Emil Praun, Wim Sweldens, and Peter Schröder. Consistent mesh parameterizations. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 179–184. ACM, 2001. 2

[51] Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. Scalable locally injective mappings. *ACM Transactions on Graphics (TOG)*, 36(4):1, 2017. 2, 4, 5

[52] Patrick Schmidt, Marcel Campen, Janis Born, and Leif Kobbelt. Inter-surface maps via constant-curvature metrics. *ACM Transactions on Graphics (TOG)*, 39(4):119–1, 2020. 2

[53] John Schreiner, Arul Asirvatham, Emil Praun, and Hugues Hoppe. Inter-surface mapping. *ACM Trans. Graph.*, 23(3):870–877, Aug. 2004. 1, 2, 6, 7, 8

[54] Alla Sheffer, Emil Praun, and Kenneth Rose. Mesh parameterization methods and their applications. *Foundations and Trends® in Computer Graphics and Vision*, 2(2):105–171, 2006. 1, 2

[55] Ayan Sinha, Jing Bai, and Karthik Ramani. Deep learning 3d shape surfaces using geometry images. In *European Conference on Computer Vision*, pages 223–240. Springer, 2016. 3

[56] Vincent Sitzmann, Julien NP Martel, Alexander W Bergman, David B Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *arXiv preprint arXiv:2006.09661*, 2020. 3

[57] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116, 2007. 2

[58] Hao Su, Haoqiang Fan, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image. *CVPR*, 2017. 3

[59] William Thomas Tutte. How to draw a graph. *Proceedings of the London Mathematical Society*, 3(1):743–767, 1963. 2

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 3

[61] Ofir Weber and Denis Zorin. Locally injective parametrization with arbitrary fixed boundaries. *ACM Transactions on Graphics (TOG)*, 33(4):75, 2014. 2, 6

[62] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, and Daniele Panozzo. Deep geometric prior for surface reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10130–10139, 2019. 4

[63] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. FoldingNet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018. 3