Surface Networks

Ilya Kostrikov[1], Zhongshi Jiang[1], Daniele Panozzo [*1], Denis Zorin [†1], and Joan Bruna[‡1,2]

[1]Courant Institute of Mathematical Sciences, New York University
[2]Center for Data Science, New York University

Abstract

We study data-driven representations for three-dimensional triangle meshes, which are one of the prevalent objects used to represent 3D geometry. Recent works have developed models that exploit the intrinsic geometry of manifolds and graphs, namely the Graph Neural Networks (GNNs) and its spectral variants, which learn from the local metric tensor via the Laplacian operator.

Despite offering excellent sample complexity and built-in invariances, intrinsic geometry alone is invariant to isometric deformations, making it unsuitable for many applications. To overcome this limitation, we propose several upgrades to GNNs to leverage extrinsic differential geometry properties of three-dimensional surfaces, increasing its modeling power. In particular, we propose to exploit the Dirac operator, whose spectrum detects principal curvature directions — this is in stark contrast with the classical Laplace operator, which directly measures mean curvature. We coin the resulting models Surface Networks (SN).

We prove that these models define shape representations that are stable to deformation and to discretization, and we demonstrate the efficiency and versatility of SNs on two challenging tasks: temporal prediction of mesh deformations under non-linear dynamics and generative models using a variational autoencoder framework with encoders/decoders given by SNs.

1. Introduction

3D geometry analysis, manipulation and synthesis plays an important role in a variety of applications from engineering to computer animation to medical imaging. Despite the vast amount of high-quality 3D geometric data available, data-driven approaches to problems involving complex geometry have yet to become mainstream, in part due to the lack of data representation regularity which is required for traditional convolutional neural network approaches. While in computer vision problems inputs are typically sampled on regular two or three-dimensional grids, surface geometry is represented in a more complex form and, in general, cannot be converted to an image-like format by parametrizing the shape using a single planar chart. Most commonly an irregular triangle mesh is used to represent shapes, capturing its main topological and geometrical properties.

Similarly to the regular grid case (used for images or videos), we are interested in data-driven representations that strike the right balance between expressive power and sample complexity. In the case of CNNs, this is achieved by exploiting the inductive bias that most computer vision tasks are locally stable to deformations, leading to localized, multiscale, stationary features. In the case of surfaces, we face a fundamental modeling choice between *extrinsic* versus *intrinsic* representations. Extrinsic representations rely on the specific embedding of surfaces within a three-dimensional ambient space, whereas intrinsic representations only capture geometric properties specific to the surface, irrespective of its parametrization. Whereas the former offer arbitrary representation power, they are unable to easily exploit inductive priors such as stability to local deformations and invariance to global transformations.

A particularly simple and popular extrinsic method [30, 31] represents shapes as point clouds in $\mathbb{R}^3$ of variable size, and leverages recent deep learning models that operate on input sets [38, 37]. Despite its advantages in terms of ease of data acquisition (they no longer require a mesh triangulation) and good empirical performance on shape classification and segmentation tasks, one may wonder whether this simplification comes at a loss of precision as one considers more challenging prediction tasks.

In this paper, we develop an alternative pipeline that applies neural networks directly on triangle meshes, building

[*]DP was supported in part by the NSF CAREER award IIS-1652515, a gift from Adobe, and a gift from nTopology.
[†]DZ was supported in part by the NSF awards DMS-1436591 and IIS-1320635.
[‡]JB was partially supported by Samsung Electronics (Improving Deep Learning using Latent Structure) and DOA W911NF-17-1-0438. Corresponding author: bruna@cims.nyu.edu

on *geometric deep learning*. These models provide data-driven intrinsic graph and manifold representations with inductive biases analogous to CNNs on natural images. Models based on Graph Neural Networks [34] and their spectral variants [6, 9, 22] have been successfully applied to geometry processing tasks such as shape correspondence [27]. In their basic form, these models learn a deep representation over the discretized surface by combining a latent representation at a given node with a local linear combination of its neighbors' latent representations, and a point-wise nonlinearity. Different models vary in their choice of linear operator and point-wise nonlinearity, which notably includes the graph Laplacian, leading to spectral interpretations of those models.

Our contributions are three-fold. First, we extend the model to support extrinsic features. More specifically, we exploit the fact that surfaces in $\mathbb{R}^3$ admit a first-order differential operator, the *Dirac* operator, that is stable to discretization, provides a direct generalization of Laplacian-based propagation models, and is able to detect principal curvature directions [8, 16]. Next, we prove that the models resulting from either Laplace or Dirac operators are stable to deformations and to discretization, two major sources of variability in practical applications. Last, we introduce a generative model for surfaces based on the variational autoencoder framework [21, 33], that is able to exploit non-Euclidean geometric regularity.

By combining the Dirac operator with input coordinates, we obtain a fully differentiable, end-to-end feature representation that we apply to several challenging tasks. The resulting Surface Networks – using either the Dirac or the Laplacian, inherit the stability and invariance properties of these operators, thus providing data-driven representations with certified stability to deformations. We demonstrate the model efficiency on a temporal prediction task of complex dynamics, based on a physical simulation of elastic shells, which confirms that whenever geometric information (in the form of a mesh) is available, it can be leveraged to significantly outperform point-cloud based models.

Our main contributions are summarized as follows:

- We demonstrate that Surface Networks provide accurate temporal prediction of surfaces under complex non-linear dynamics, motivating the use of geometric shape information.
- We prove that Surface Networks define shape representations that are stable to deformation and to discretization.
- We introduce a generative model for 3D surfaces based on the variational autoencoder.

A reference implementation of our algorithm is available at https://github.com/jiangzhongshi/SurfaceNetworks.

2. Related Work

Learning end-to-end representations on irregular and non-Euclidean domains is an active and ongoing area of research. [34] introduced graph neural networks as recursive neural networks on graphs, whose stationary distributions could be trained by backpropagation. Subsequent works [23, 37] have relaxed the model by untying the recurrent layer weights and proposed several nonlinear updates through gating mechanisms. Graph neural networks are in fact natural generalizations of convolutional networks to non-Euclidean graphs. [6, 15] proposed to learn smooth spectral multipliers of the graph Laplacian, albeit with high computational cost, and [9, 22] resolved the computational bottleneck by learning polynomials of the graph Laplacian, thus avoiding the computation of eigenvectors and completing the connection with GNNs. We refer the reader to [5] for an exhaustive literature review on the topic. GNNs are finding application in many different domains. [2, 7] develop graph interaction networks that learn pairwise particle interactions and apply them to discrete particle physical dynamics. [10, 19] study molecular fingerprints using variants of the GNN architecture, and [12] further develop the model by combining it with set representations [38], showing state-of-the-art results on molecular prediction. The resulting models, so-called Message-Passing Neural Networks, also learn the diffusion operator, which can be seen as generalizations of the Dirac model on general graphs.

In the context of computer graphics, [25] developed the first CNN model on meshed surfaces using intrinsic patch representations, and further generalized in [4] and [27]. This last work allows for flexible representations via the so-called pseudo-coordinates and obtains state-of-the-art results on 3D shape correspondence, although it does not easily encode first-order differential information. These intrinsic models contrast with Euclidean models such as [42, 40], that have higher sample complexity, since they need to learn the underlying invariance of the surface embedding. Point-cloud based models are increasingly popular to model 3d objects due to their simplicity and versatility. [30, 31] use set-invariant representations from [38, 37] to solve shape segmentation and classification tasks. More recently, [24] proposes to learn surface convolutional network from a canonical representation of planar flat-torus, with excellent performance on shape segmentation and classification, although such canonical representations may introduce exponential scale changes that can introduce instabilities. Finally, [11] proposes a point-cloud generative model for 3D shapes, that incorporates invariance to point permutations, but does not encode geometrical information as our shape generative model. Learning variational deformations is an important problem for graphics applications, since it enables negligible and fixed per-frame cost [29], but it is currently limited to 2D deformations using point handles.

In constrast, our method easily generalizes to 3D and learns dynamic behaviours.

3. Surface Networks

This section presents our surface neural network model and its basic properties. We start by introducing the problem setup and notations using the Laplacian formalism (Section 3.1), and then introduce our model based on the Dirac operator (Section 3.2).

3.1. Laplacian Surface Networks

Our first goal is to define a trainable representation of discrete surfaces. Let $\mathcal{M} = \{V, E, F\}$ be a triangular mesh, where $V = (v_i \in \mathbb{R}^3)_{i \leq N}$ contains the node coordinates, $E = (e_{i,j})$ corresponds to edges, and F is the set of triangular faces. We denote as Δ the discrete Laplace-Beltrami operator (we use the popular cotangent weights formulation, see [5] for details).

This operator can be interpreted as a local, linear high-pass filter in $\mathcal{M}$ that acts on signals $x \in \mathbb{R}^{d \times |V|}$ defined on the vertices as a simple matrix multiplication $\tilde{x} = \Delta x$. By combining Δ with an *all-pass* filter and learning generic linear combinations followed by a point-wise nonlinearity, we obtain a simple generalization of localized convolutional operators in $\mathcal{M}$ that update a feature map from layer k to layer $k+1$ using trainable parameters A_k and B_k:

$$x^{k+1} = \rho\left(A_k \Delta x^k + B_k x^k\right) \ , \ A_k, B_k \in \mathbb{R}^{d_{k+1} \times d_k} \ . \quad (1)$$

By observing that the Laplacian itself can be written in terms of the graph weight similarity by diagonal renormalization, this model is a specific instance of the graph neural network [34, 5, 22] and a generalization of the spectrum-free Laplacian networks from [9]. As shown in these previous works, convolutional-like layers (1) can be combined with graph coarsening or pooling layers.

In contrast to general graphs, meshes contain a low-dimensional Euclidean embedding that contains potentially useful information in many practical tasks, despite being extrinsic and thus not invariant to the global position of the surface. A simple strategy to strike a good balance between expressivity and invariance is to include the node canonical coordinates as input channels to the network: $x^1 := V \in \mathbb{R}^{|V| \times 3}$. The mean curvature can be computed by applying the Laplace operator to the coordinates of the vertices:

$$\Delta x^1 = -2H\mathbf{n} \ , \quad (2)$$

where H is the mean curvature function and $\mathbf{n}(u)$ is the normal vector of the surface at point u. As a result, the Laplacian neural model (1) has access to mean curvature and normal information. Feeding Euclidean embedding coordinates into graph neural network models is related to the use of generalized coordinates from [27]. By cascading K layers of the form (1) we obtain a representation $\Phi_\Delta(\mathcal{M})$ that contains generic features at each node location. When the number of layers K is of the order of $\mathrm{diam}(\mathcal{M})$, the diameter of the graph determined by $\mathcal{M}$, then the network is able to propagate and aggregate information across the whole surface.

Equation (2) illustrates that a Laplacian layer is only able to extract isotropic high-frequency information, corresponding to the mean variations across all directions. Although in general graphs there is no well-defined procedure to recover anisotropic local variations, in the case of surfaces some authors ([4, 1, 27]) have considered anisotropic extensions. We describe next a particularly simple procedure to increase the expressive power of the network using a related operator from quantum mechanics: the Dirac operator, that has been previously used successfully in the context of surface deformation [8] and shape analysis [16].

3.2. Dirac Surface Networks

The Laplace-Beltrami operator Δ is a second-order differential operator, constructed as $\Delta = -\mathrm{div}\nabla$ by combining the gradient (a first-order differential operator) with its adjoint, the divergence operator. In an Euclidean space, one has access to these first-order differential operators separately, enabling oriented high-pass filters.

For convenience, we embed $\mathbb{R}^3$ to the imaginary quaternion space $\mathrm{Im}(\mathbb{H})$ (see Appendix A in the Suppl. Material for details). The Dirac operator is then defined as a matrix $D \in \mathbb{H}^{|F| \times |V|}$ that maps (quaternion) signals on the nodes to signals on the faces. In coordinates,

$$D_{f,j} = \frac{-1}{2|\mathcal{A}_f|} e_j \ , \ f \in F, j \in V \ ,$$

where e_j is the opposing edge vector of node j in the face f, and $\mathcal{A}_f$ is the area (see Appendix A) using counter-clockwise orientations on all faces.

To apply the Dirac operator defined in quaternions to signals in vertices and faces defined in real numbers, we write the feature vectors as quaternions by splitting them into chunks of 4 real numbers representing the real and imaginary parts of a quaternion; see Appendix A. Thus, we always work with feature vectors with dimensionalities that are multiples of 4. The Dirac operator provides first-order differential information and is sensitive to local orientations. Moreover, one can verify [8] that

$$\mathrm{Re}\ D^* D = \Delta \ ,$$

where D^* is the adjoint operator of D in the quaternion space (see Appendix A). The adjoint matrix can be computed as $D^* = M_V^{-1} D^H M_F$ where D^H is a conjugate transpose of D and M_V, M_F are diagonal mass matrices with one third of areas of triangles incident to a vertex and face areas respectively.

The Dirac operator can be used to define a new neural surface representation that alternates layers with signals defined over nodes with layers defined over faces. Given a d-dimensional feature representation over the nodes $x^k \in \mathbb{R}^{d\times|V|}$, and the faces of the mesh, $y^k \in \mathbb{R}^{d\times|F|}$, we define a d'-dimensional mapping to a face representation as

$$y^{k+1} = \rho\left(C_k D x^k + E_k y^k\right), C_k, E_k \in \mathbb{R}^{d_{k+1}\times d_k}, \quad (3)$$

where C_k, E_k are trainable parameters. Similarly, we define the adjoint layer that maps back to a $\tilde{d}$-dimensional signal over nodes as

$$x^{k+1} = \rho\left(A_k D^* y^{k+1} + B_k x^k\right), A_k, B_k \in \mathbb{R}^{d_{k+1}\times d_k}, \quad (4)$$

where A_k, B_k are trainable parameters. A surface neural network layer is thus determined by parameters $\{A, B, C, E\}$ using equations (3) and (4) to define $x^{k+1} \in \mathbb{R}^{d_{k+1}\times|V|}$. We denote by $\Phi_D(\mathcal{M})$ the mesh representation resulting from applying K such layers (that we assume fixed for the purpose of exposition).

The Dirac-based surface network is related to edge feature transforms proposed on general graphs in [12], although these edge measurements cannot be associated with derivatives due to lack of proper orientation. In general graphs, there is no notion of square root of Δ that recovers oriented first-order derivatives.

4. Stability of Surface Networks

Here we describe how Surface Networks are geometrically stable, because surface deformations become additive noise under the model. Given a continuous surface $S \subset \mathbb{R}^3$ or a discrete mesh $\mathcal{M}$, and a smooth deformation field $\tau : \mathbb{R}^3 \to \mathbb{R}^3$, we are particularly interested in two forms of stability:

- Given a discrete mesh $\mathcal{M}$ and a certain non-rigid deformation τ acting on $\mathcal{M}$, we want to certify that $\|\Phi(\mathcal{M}) - \Phi(\tau(\mathcal{M}))\|$ is small if $\|\nabla\tau(\nabla\tau)^* - \mathbf{I}\|$ is small, i.e when the deformation is nearly rigid; see Theorem 4.1.

- Given two discretizations $\mathcal{M}_1$ and $\mathcal{M}_2$ of the same underlying surface S, we would like to control $\|\Phi(\mathcal{M}_1) - \Phi(\mathcal{M}_2)\|$ in terms of the resolution of the meshes; see Theorem 4.2.

These stability properties are important in applications, since most tasks we are interested in are stable to deformation and to discretization. We shall see that the first property is a simple consequence of the fact that the mesh Laplacian and Dirac operators are themselves stable to deformations. The second property will require us to specify under which conditions the discrete mesh Laplacian $\Delta_{\mathcal{M}}$ converges to the Laplace-Beltrami operator Δ_S on S. Unless it is clear from the context, in the following Δ will denote the discrete Laplacian.

Theorem 4.1 *Let $\mathcal{M}$ be a N-node mesh and $x, x' \in \mathbb{R}^{|V|\times d}$ be input signals defined on the nodes. Assume the nonlinearity $\rho(\cdot)$ is non-expansive ($|\rho(z) - \rho(z')| \leq |z - z'|$). Then*

(a) $\|\Phi_\Delta(\mathcal{M}; x) - \Phi_\Delta(\mathcal{M}; x')\| \leq \alpha_\Delta \|x - x'\|$, *where α_Δ depends only on the trained weights and the mesh.*

(b) $\|\Phi_D(\mathcal{M}; x) - \Phi_D(\mathcal{M}; x')\| \leq \alpha_D \|x - x'\|$, *where α_D depends only on the trained weights and the mesh.*

(c) Let $|\nabla\tau|_\infty := \sup_u \|\nabla\tau(u)(\nabla\tau(u))^* - \mathbf{1}\|$, *where $\nabla\tau(u)$ is the Jacobian matrix of $u \mapsto \tau(u)$. Then* $\|\Phi_\Delta(\mathcal{M}; x) - \Phi_\Delta(\tau(\mathcal{M}); x)\| \leq \beta_\Delta |\nabla\tau|_\infty \|x\|$, *where β_Δ is independent of τ and x.*

(d) Denote by $\widetilde{|\nabla\tau|}_\infty := \sup_u \|\nabla\tau(u) - \mathbf{1}\|$. *Then* $\|\Phi_D(\mathcal{M}; x) - \Phi_D(\tau(\mathcal{M}); x)\| \leq \beta_D \widetilde{|\nabla\tau|}_\infty \|x\|$, *where β_D is independent of τ and x.*

Properties (a) and (b) are not specific to surface representations, and are a simple consequence of the non-expansive property of our chosen nonlinearities. The constant α is controlled by the product of ℓ_2 norms of the network weights at each layer and the norm of the discrete Laplacian operator. Properties (c) and (d) are based on the fact that the Laplacian and Dirac operators are themselves stable to deformations, a property that depends on two key aspects: first, the Laplacian/Dirac is localized in space, and next, that it is a high-pass filter and therefore only depends on relative changes in position.

One caveat of Theorem 4.1 is that the constants appearing in the bounds depend upon a bandwidth parameter given by the reciprocal of triangle areas, which increases as the size of the mesh increases. This corresponds to the fact that the spectral radius of $\Delta_{\mathcal{M}}$ diverges as the mesh size N increases.

In order to overcome this problematic asymptotic behavior, it is necessary to exploit the smoothness of the signals incoming to the surface network. This can be measured with Sobolev norms defined using the spectrum of the Laplacian operator. Given a mesh $\mathcal{M}$ of N nodes approximating an underlying surface S, and its associated cotangent Laplacian $\Delta_{\mathcal{M}}$, consider the spectral decomposition of $\Delta_{\mathcal{M}}$ (a symmetric, positive definite operator):

$$\Delta_{\mathcal{M}} = \sum_{k\leq N} \lambda_k e_k e_k^T \ , \ e_k \in \mathbb{R}^N \ , \ 0 \leq \lambda_1 \leq \lambda_2 \cdots \leq \lambda_N \ .$$

Under normal uniform convergence [1] [39], the spectrum of $\Delta_{\mathcal{M}}$ converges to the spectrum of the Laplace-Beltrami operator Δ_S of S. If S is bounded, it is known from the Weyl

[1] which controls how the normals of the mesh align with the surface normals; see [39].

law [41] that there exists $\gamma > 0$ such that $k^{-\gamma(S)} \lesssim \lambda_k^{-1}$, so the eigenvalues λ_k do not grow too fast. The smoothness of a signal $x \in \mathbb{R}^{|V| \times d}$ defined in $\mathcal{M}$ is captured by how fast its spectral decomposition $\hat{x}(k) = e_k^T x \in \mathbb{R}^d$ decays [36]. We define $\|x\|_{\mathcal{H}}^2 := \sum_k \lambda(k)^2 \|\hat{x}(k)\|^2$ is Sobolev norm, and $\beta(x, S) > 1$ as the largest rate such that its spectral decomposition coefficients satisfy

$$\|\hat{x}(k)\| \lesssim k^{-\beta} , \ (k \to \infty) . \tag{5}$$

If $x \in \mathbb{R}^{|V| \times d}$ is the input to the Laplace Surface Network of R layers, we denote by $(\beta_0, \beta_1, \ldots, \beta_{R-1})$ the smoothness rates of the feature maps $x^{(r)}$ defined at each layer $r \leq R$.

Theorem 4.2 *Consider a surface S and a finite-mesh approximation $\mathcal{M}_N$ of N points, and Φ_Δ a Laplace Surface Network with parameters $\{(A_r, B_r)\}_{r \leq R}$. Denote by $d(S, \mathcal{M}_N)$ the uniform normal distance, and let x_1, x_2 be piece-wise polyhedral approximations of $\bar{x}(t)$, $t \in S$ in $\mathcal{M}_N$, with $\|\bar{x}\|_{\mathcal{H}(S)} < \infty$. Assume $\|\bar{x}^{(r)}\|_{\mathcal{H}(S)} < \infty$ for $r \leq R$.*

(a) If x_1, x_2 are two functions such that the R feature maps $x_l^{(r)}$ have rates $(\beta_0, \beta_1, \ldots, \beta_{R-1})$, then

$$\|\Phi_\Delta(x_1; \mathcal{M}_N) - \Phi_\Delta(x_2; \mathcal{M}_N)\|^2 \leq C(\beta) \|x_1 - x_2\|^{h(\beta)} , \tag{6}$$

with $h(\beta) = \prod_{r=1}^{R} \frac{\beta_r - 1}{\beta_r - 1/2}$, and where $C(\beta)$ does not depend upon N.

(b) If τ is a smooth deformation field, then $\|\Phi_\Delta(x; \mathcal{M}_N) - \Phi_\Delta(x; \tau(\mathcal{M}_N))\| \leq C |\nabla \tau|_\infty^{h(\beta)}$, where C does not depend upon N.

(c) Let $\mathcal{M}$ and $\mathcal{M}'$ be N-point discretizations of S, If $\max(d(\mathcal{M}, S), d(\mathcal{M}', S)) \leq \epsilon$, then $\|\Phi_\Delta(\mathcal{M}; x) - \Phi_\Delta(\mathcal{M}', x')\| \leq C \epsilon^{h(\beta)}$, where C is independent of N.

This result ensures that if we use as generator of the SN an operator that is consistent as the mesh resolution increases, the resulting surface representation is also consistent. Although our present result only concerns the Laplacian, the Dirac operator also has a well-defined continuous counterpart [8] that generalizes the gradient operator in quaternion space. Also, our current bounds depend explicitly upon the smoothness of feature maps across different layers, which may be controlled in terms of the original signal if one considers nonlinearities that demodulate the signal, such as $\rho(x) = |x|$ or $\rho(x) = \mathrm{ReLU}(x)$. These extensions are left for future work. Finally, a specific setup that we use in experiments is to use as input signal the canonical coordinates of the mesh $\mathcal{M}$. In that case, an immediate application of the previous theorem yields

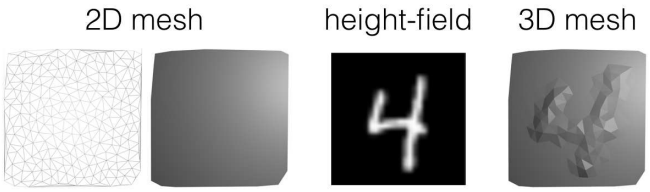

Figure 1. Height-Field Representation of surfaces. A 3D mesh $\mathcal{M} \subset \mathbb{R}^3$ (right) is expressed in terms of a "sampling" 2D irregular mesh $\tilde{\mathcal{M}} \subset \mathbb{R}^2$ (left) and a depth scalar field $f : \tilde{\mathcal{M}} \to \mathbb{R}$ over $\tilde{\mathcal{M}}$ (center).

Corollary 4.3 *Denote $\Phi(\mathcal{M}) := \Phi_{\mathcal{M}}(V)$, where V are the node coordinates of $\mathcal{M}$. Then, if $A_1 = 0$,*

$$\|\Phi(\mathcal{M}) - \Phi(\tau(\mathcal{M}))\| \leq \kappa \max(|\nabla \tau|_\infty, \|\nabla^2 \tau\|)^{h(\beta)} . \tag{7}$$

5. Generative Surface Models

State-of-the-art generative models for images, such as generative adversarial networks [32], pixel autoregressive networks [28], or variational autoencoders [21], exploit the locality and stationarity of natural images in their probabilistic models, in the sense that the model satisfies $p_\theta(x) \approx p_\theta(x_\tau)$ by construction, where x_τ is a small deformation of a given input x. This property is obtained via encoders and decoders with a deep convolutional structure. We intend to exploit similar geometric stability priors with SNs, owing to their stability properties described in Section 4. A mesh generative model contains two distinct sources of randomness: on the one hand, the randomness associated with the underlying continuous surface, which corresponds to shape variability; on the other hand, the randomness of the discretization of the surface. Whereas the former contains the essential semantic information, the latter is not informative, and to some extent independent of the shape identity. We focus initially on meshes that can be represented as a depth map over an (irregular) 2D mesh, referred as *height-field* meshes in the literature. That is, a mesh $\mathcal{M} = (V, E, F)$ is expressed as $(\tilde{\mathcal{M}}, f(\tilde{\mathcal{M}}))$, where $\tilde{\mathcal{M}} = (\tilde{V}, \tilde{E}, \tilde{F})$ is now a 2D mesh and $f : \tilde{V} \to \mathbb{R}$ is a *depth*-map encoding the original node locations V, as shown in Figure 1.

In this work, we consider the variational autoencoder framework [21, 33]. It considers a mixture model of the form $p(\mathcal{M}) = \int p_\theta(\mathcal{M} \mid h) p_0(h) dh$, where $h \in \mathbb{R}^{|S|}$ is a vector of latent variables. We train this model by optimizing the variational lower bound of the data log-likelihood:

$$\min_{\theta, \psi} \frac{1}{L} \sum_{l \leq L} -\mathbb{E}_{h \sim q_\psi(h \mid \mathcal{M}_l)} \log p_\theta(\mathcal{M}_l \mid h) + D_{KL}(q_\psi(h \mid \mathcal{M}_l) \,||\, p_0(h)) . \tag{8}$$

We thus need to specify a conditional generative model $p_\theta(\mathcal{M} \mid h)$, a prior distribution $p_0(h)$ and a variational approximation to the posterior $q_\psi(h \mid \mathcal{M})$, where θ and ψ denote respectively generative and variational trainable parameters. Based on the height-field representation,

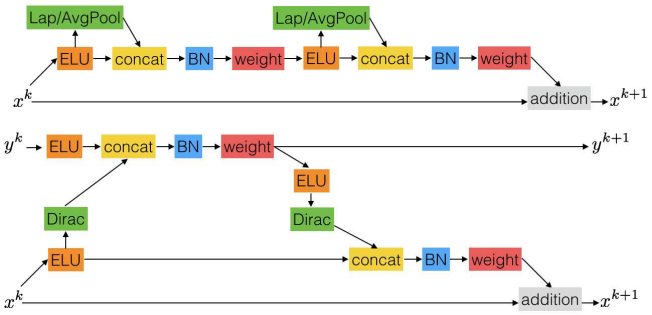

Figure 2. A single ResNet-v2 block used for Laplace, Average Pooling (top) and Dirac models (bottom). The green boxes correspond to the linear operators replacing convolutions in regular domains. We consider Exponential Linear Units (ELU) activations (orange), Batch Normalization (blue) and '1×1' convolutions (red) containing the trainable parameters; see Eqs (1, 3 and 4). We slightly abuse language and denote by x^{k+1} the output of this 2-layer block.

we choose for simplicity a separable model of the form $p_\theta(\mathcal{M} \mid h) = p_\theta(f \mid h, \tilde{\mathcal{M}}) \cdot p(\tilde{\mathcal{M}})$, where $\tilde{\mathcal{M}} \sim p(\tilde{\mathcal{M}})$ is a homogeneous Poisson point process, and $f \sim p_\theta(f \mid h, \tilde{\mathcal{M}})$ is a normal distribution with mean and isotropic covariance parameters given by a SN:

$$p_\theta(f \mid h, \tilde{\mathcal{M}}) = \mathcal{N}(\mu(h, \tilde{\mathcal{M}}), \sigma^2(h, \tilde{\mathcal{M}})\mathbf{1}) \, ,$$

with $[\mu(h, \tilde{\mathcal{M}}), \sigma^2(h, \tilde{\mathcal{M}})] = \Phi_D(\tilde{M}; h)$. The generation step thus proceeds as follows. We first sample a 2D mesh $\tilde{\mathcal{M}}$ independent of the latent variable h, and then sample a depth field over $\tilde{\mathcal{M}}$ conditioned on h from the output of a decoder network $\Phi_D(\tilde{M}; h)$. Finally, the variational family q_ψ is also a Normal distribution whose parameters are obtained from an encoder Surface Neural Network whose last layer is a global pooling that removes the spatial localization: $q_\psi(h \mid \mathcal{M}) = \mathcal{N}(\bar{\mu}, \bar{\sigma}^2\mathbf{1})$, with $[\bar{\mu}, \bar{\sigma}] = \bar{\Phi}_D(\mathcal{M})$.

6. Experiments

For experimental evaluation, we compare models built using ResNet-v2 blocks [14], where convolutions are replaced with the appropriate operators (see Fig. 2): *(i)* a point cloud based model from [37] that aggregates global information by averaging features in the intermediate layers and distributing them to all nodes; *(ii)* a Laplacian Surface network with input canonical coordinates; *(iii)* a Dirac Surface Network model. We report experiments on generative models using an unstructured variant of MNIST digits (Section 6.1), and on temporal prediction under non-rigid deformation models (Section 6.2).

6.1. MeshMNIST

For this task, we construct a MeshMNIST database with only *height-field* meshes (Sec. 5). First, we sample points on a 2D plane ($[0, 27] \times [0, 27]$) with Poisson disk sampling with $r = 1.0$, which roughly generates 500 points, and apply a Delaunay triangulation to these points. We then overlay the triangulation with the original MNIST images and assign to each point a z coordinate bilinearly interpolating the grey-scale value. Thus, the procedure allows us to define a sampling process over 3D height-field meshes.

We used VAE models with decoders and encoders built using 10 ResNet-v2 blocks with 128 features. The encoder converts a mesh into a latent vector by averaging output of the last ResNet-v2 block and applying linear transformations to obtain mean and variance, while the decoder takes a latent vector and a 2D mesh as input (corresponding to a specific 3D mesh) and predicts offsets for the corresponding locations. We keep variance of the decoder as a trainable parameter that does not depend on input data. We trained the model for 75 epochs using Adam optimizer [20] with learning rate 10^{-3}, weight decay 10^{-5} and batch size 32. Figures 3,4 illustrate samples from the model. The geometric encoder is able to leverage the local translation invariance of the data despite the irregular sampling, whereas the geometric decoder automatically adapts to the specific sampled grid, as opposed to set-based generative models.

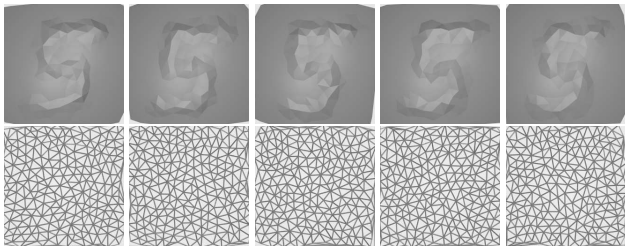

Figure 3. Samples generated for the same latent variable and different triangulations. The learned representation is independent of discretization/triangulation (Poisson disk sampling with p=1.5).

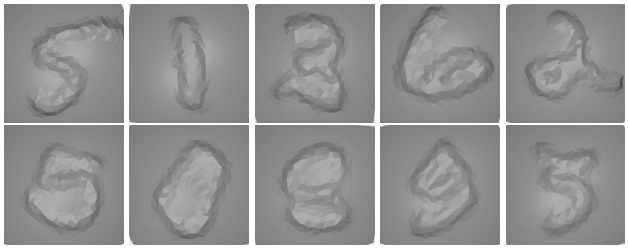

Figure 4. Meshes from the dataset (first five). And meshes generated by our model (last five).

6.2. Spatio-Temporal Predictions

One specific task we consider is temporal predictions of non-linear dynamics. Given a sequence of frames $X = X^1, X^2, \ldots, X^n$, the task is to predict the following frames $Y = Y^1 = X^{n+1}, Y^2, \ldots, Y^m = X^{n+m}$. As in [26], we use a simple non-recurrent model that takes a concatenation of input frames X and predicts a concatenation of frames Y. We condition on $n = 2$ frames and predict the next

Model	Receptive field	Number of parameters	Smooth L1-loss (mean per sequence (std))
MLP	1	519672	64.56 (0.62)
PointCloud	-	1018872	23.64 (0.21)
Laplace	16	1018872	17.34 (0.52)
Dirac	8	1018872	**16.84 (0.16)**

Table 1. Evaluation of different models on the temporal task

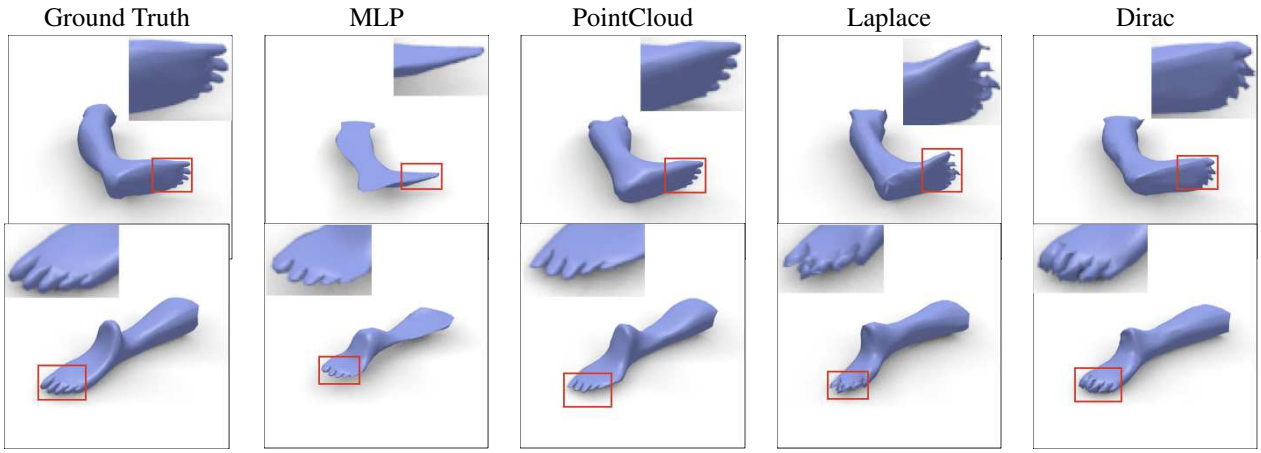

Figure 5. Qualitative comparison of different models. We plot 30th predicted frames correspondingly for two sequences in the test set. Boxes indicate distinctive features. For larger crops, see Figure 6

$m = 40$ frames. In order to generate data, we first extracted 10k patches from the MPI-Faust dataset[3], by selecting a random point and growing a topological sphere of radius 15 edges (i.e. the 15-ring of the point). For each patch, we generate a sequence of 50 frames by randomly rotating it and letting it fall to the ground. We consider the mesh a thin elastic shell, and we simulate it using the As-Rigid-As-Possible technique [35], with additional gravitational forces [17]. Libigl [18] has been used for the mesh processing tasks. Sequences with patches from the first 80 subjects were used in training, while the 20 last subjects were used for testing. The dataset and the code are available on request. We restrict our experiments to temporal prediction tasks that are deterministic when conditioned on several initial frames. Thus, we can train models by minimizing smooth-L1 loss [13] between target frames and output of our models.

We used models with 15 ResNet-v2 blocks with 128 output features each. In order to cover larger context for Dirac and Laplace based models, we alternate these blocks with Average Pooling blocks. We predict offsets to the last conditioned frame and use the corresponding Laplace and Dirac operators. Thus, the models take 6-dimensional inputs and produce 120-dimensional outputs. We trained all models using the Adam optimizer [20] with learning rate 10^{-3}, weight decay 10^{-5}, and batch size 32. After 60k steps we decreased the learning rate by a factor of 2 every 10k steps. The models were trained for 110k steps in overall.

Table 1 reports quantitative prediction performance of different models, and Figure 5 displays samples from the prediction models at specific frames. The set-to-set model [38, 37], corresponding to a point-cloud representation used also in [30], already performs reasonably well on the task, even if the visual difference is noticeable. Nevertheless, the gap between this model and Laplace-/Dirac-based models is significant, both visually and quantitatively. Dirac-based model outperforms Laplace-based model despite the smaller receptive field. Videos comparing the performance of different models are available in the additional material.

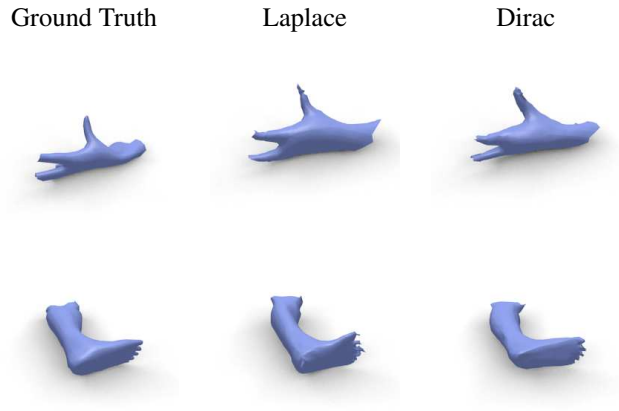

Figure 6. Dirac-based model visually outperforms Laplace-based models in the regions of high mean curvature.

Figure 6 illustrates the effect of replacing Laplace by Dirac in the formulation of the SN. Laplacian-based models, since they propagate information using an isotropic operator, have more difficulties at resolving corners and pointy

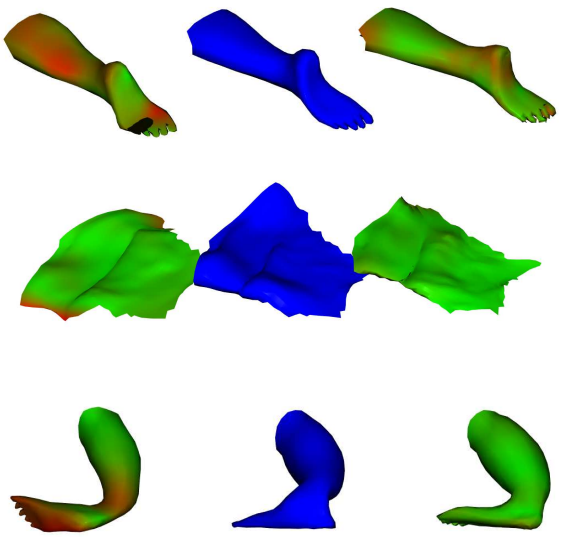

Figure 7. From left to right: PointCloud (set2set), ground truth and Dirac based model. Color corresponds to mean squared error between ground truth and prediction: green - smaller error, red - larger error.

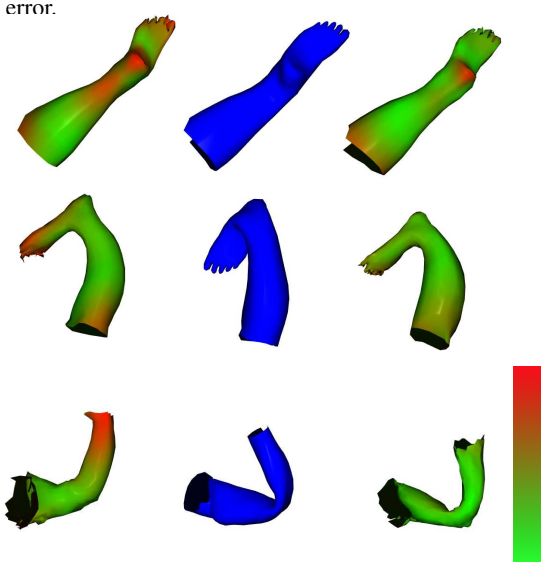

Figure 8. From left to right: Laplace, ground truth and Dirac based model. Color corresponds to mean squared error between ground truth and prediction: green - smaller error, red - larger error.

structures than the Dirac operator, that is sensitive to principal curvature directions. However, the capacity of Laplace models to exploit the extrinsic information only via the input coordinates is remarkable and more computationally efficient than the Dirac counterpart. Figures 7 and 8 overlay the prediction error and compare Laplace against Dirac and PointCloud against Dirac respectively. They confirm first that SNs outperform the point-cloud based model, which often produce excessive flattening and large deformations, and next that first-order Dirac operators help resolve areas with high directional curvature. We refer to the supplementary material for additional qualitative results.

7. Conclusions

We have introduced Surface Networks, a deep neural network that is designed to naturally exploit the non-Euclidean geometry of surfaces. We have shown how a first-order differential operator (the Dirac operator) can detect and adapt to geometric features beyond the local mean curvature, the limit of what Laplacian-based methods can exploit. This distinction is important in practice, since areas with high directional curvature are perceptually important, as shown in the experiments. That said, the Dirac operator comes at increased computational cost due to the quaternion calculus, and it would be interesting to instead *learn* the operator, akin to recent Message-Passing NNs [12] and explore whether Dirac is recovered.

Whenever the data contains good-quality meshes, our experiments demonstrate that using intrinsic geometry offers vastly superior performance to point-cloud based models. While there are not many such datasets currently available, we expect them to become common in the next years, as scanning and reconstruction technology advances and 3D sensors are integrated in consumer devices. SNs provide efficient inference, with predictable runtime, which makes them appealing across many areas of computer graphics, where a fixed, per-frame cost is required to ensure a stable framerate, especially in VR applications. Our future plans include applying Surface Networks precisely to having automated, data-driven mesh processing, and generalizing the generative model to arbitrary meshes, which will require an appropriate multi-resolution pipeline.

References

[1] M. Andreux, E. Rodolà, M. Aubry, and D. Cremers. Anisotropic Laplace-Beltrami operators for shape analysis. In *Proc. NORDIA*, 2014. 3

[2] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, pages 4502–4510, 2016. 2

[3] F. Bogo, J. Romero, M. Loper, and M. J. Black. Faust: Dataset and evaluation for 3d mesh registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3794–3801, 2014. 7

[4] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 3189–3197, 2016. 2, 3

[5] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *arXiv preprint arXiv:1611.08097*, 2016. 2, 3

[6] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *Proc. ICLR*, 2013. 2

[7] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. *ICLR*, 2016. 2

[8] K. Crane, U. Pinkall, and P. Schröder. Spin transformations of discrete surfaces. In *ACM Transactions on Graphics (TOG)*. ACM, 2011. 2, 3, 5

[9] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3837–3845, 2016. 2, 3

[10] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Neural Information Processing Systems*, 2015. 2

[11] H. Fan, H. Su, and L. Guibas. A point set generation network for 3d object reconstruction from a single image. *arXiv preprint arXiv:1612.00603*, 2016. 2

[12] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017. 2, 4, 8

[13] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015. 7

[14] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016. 6

[15] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv:1506.05163*, 2015. 2

[16] K. C. Hsueh-Ti Derek Liu, Alec Jacobson. A dirac operator for extrinsic shape analysis. *Computer Graphics Forum*, 2017. 2, 3

[17] A. Jacobson. *Algorithms and Interfaces for Real-Time Deformation of 2D and 3D Shapes*. PhD thesis, ETH, Zürich, 2013. 7

[18] A. Jacobson, D. Panozzo, et al. libigl: A simple C++ geometry processing library, 2016. http://libigl.github.io/libigl/. 7

[19] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, and P. Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 2016. 2

[20] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representation*, 2015. 6, 7

[21] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2, 5

[22] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 2, 3

[23] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015. 2

[24] H. Maron, M. Galun, N. Aigerman, M. Trope, N. Dym, E. Yumer, V. Kim, and Y. Lipman. Convolutional neural networks on surfaces via seamless toric covers. In *SIGGRAPH*, 2017. 2

[25] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015. 2

[26] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015. 6

[27] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. *arXiv preprint arXiv:1611.08402*, 2016. 2, 3

[28] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016. 5

[29] R. Poranne and Y. Lipman. Simple approximations of planar deformation operators. Technical report, ETHZ, 2015. 2

[30] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *arXiv preprint arXiv:1612.00593*, 2016. 1, 2, 7

[31] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 1, 2

[32] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 5

[33] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015. 2, 5

[34] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. 2, 3

[35] O. Sorkine and M. Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, 2007. 7

[36] D. A. Spielman. Spectral graph theory and its applications. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 29–38. IEEE, 2007. 5

[37] S. Sukhbaatar, R. Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252, 2016. 1, 2, 6, 7

[38] O. Vinyals, S. Bengio, and M. Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015. 1, 2, 7

[39] M. Wardetzky. Convergence of the cotangent formula: An overview. In *Discrete Differential Geometry*, pages 275–286. 2008. 4

[40] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li. Dense human body correspondences using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1544–1553, 2016. 2

[41] H. Weyl. Über die asymptotische verteilung der eigenwerte. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1911:110–117, 1911. 5

[42] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. 2