Convolutional Neural Networks on Surfaces via Seamless Toric Covers

H. MARON, M. GALUN, N. AIGERMAN, M. TROPE, N. DYM, Weizmann Institute of Science
V. G. KIM, E. YUMER, Adobe Research
Y. LIPMAN, Weizmann Institute of Science

The recent success of convolutional neural networks (CNNs) for image processing tasks is inspiring research efforts attempting to achieve similar success for geometric tasks. One of the main challenges in applying CNNs to surfaces is defining a natural convolution operator on surfaces.

In this paper we present a method for applying deep learning to sphere-type shapes using a global seamless parameterization to a planar flat-torus, for which the convolution operator is well defined. As a result, the standard deep learning framework can be readily applied for learning semantic, high-level properties of the shape.

An indication of our success in bridging the gap between images and surfaces is the fact that our algorithm succeeds in learning semantic information from an input of raw low-dimensional feature vectors.

We demonstrate the usefulness of our approach by presenting two applications: human body segmentation, and automatic landmark detection on anatomical surfaces. We show that our algorithm compares favorably with competing geometric deep-learning algorithms for segmentation tasks, and is able to produce meaningful correspondences on anatomical surfaces where hand-crafted features are bound to fail.

CCS Concepts: • **Computing methodologies → Neural networks**; **Shape analysis**;

Additional Key Words and Phrases: Geometric deep learning, Convolutional neural network, Shape analysis, shape segmentation

ACM Reference format:
H. Maron, M. Galun, N. Aigerman, M. Trope, N. Dym, V. G. Kim, E. Yumer, and Y. Lipman. 2017. Convolutional Neural Networks on Surfaces via Seamless Toric Covers. *ACM Trans. Graph.* 36, 4, Article 71 (July 2017), 10 pages.
DOI: http://dx.doi.org/10.1145/3072959.3073616

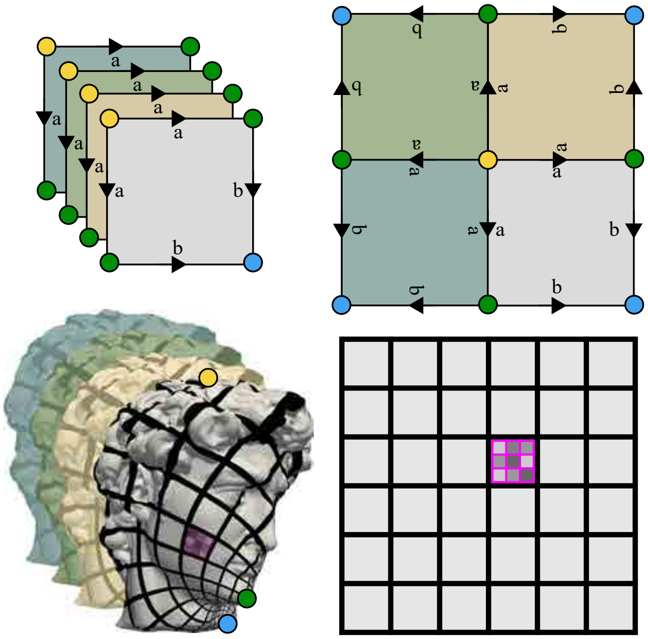

Fig. 1. Defining convolutional neural networks on sphere-like surfaces: we construct a torus (top-right) from four exact copies of the surface (top-left) and map it to the flat-torus (bottom-right) to define a local, translation invariant convolution (bottom-left). This construction is unique up to a choice of three points on the surface (colored disks).

1 INTRODUCTION

A recent research effort in the geometry processing and vision communities is to translate the incredible success of deep convolutional neural networks (CNN) to geometric settings. One particularly interesting scenario is applying CNNs for supervised learning of functions or labels over curved two dimensional sphere-like surfaces. This is a common problem in analyzing human bodies, anatomical, and medical data.

Applying CNNs to extrinsic surface representation such as volumetric grids [Qi et al. 2016b] or depth map projections on extrinsic 2D cameras [Wei et al. 2016] requires working with 3D grids, or dealing with many camera and lighting parameters, and is very sensitive to deformations (e.g., human pose changes). While it might be possible to learn a representation that is invariant to these deformations, this requires substantial amount of training data. In contrast, the goal of our paper is to provide an intrinsic representation that would enable applying CNNs directly to the surfaces.

One of the main challenges in applying CNN to curved surfaces is that there is no clear generalization of the convolution operator. In particular, two properties of the convolution operator that are considered pivotal in the success of the CNN architectures are *locality* and *translation invariance*. It is possible to parameterize a surface locally on a geodesic patch around a point [Masci et al. 2015], however, this representation lacks global context. Sinha et al. [2016] proposed geometry images to globally parameterize sphere-like surface into an image, however, although continuous, their representation is not *seamless* (the parameterization is dependent on the cuts made for its computation), their space of parameterizations, namely area-preserving maps has a very high number of degrees of freedom (*i.e.*, it requires an infinite number of point constraints to uniquely define a mapping) and therefore can represent the same shape in many different arbitrary ways in the image (see Figure 2 for an example). Lastly, the convolution on the geometry image is not translation invariant.

DOI: http://dx.doi.org/10.1145/3072959.3073616

Defining a local translation invariant convolution operator on surfaces is not trivial. The first hurdle is topological: the only surface type for which a translation invariant convolution is well defined is the torus (this will be explained in Section 4). However, clearly it is not possible to map sphere-like surfaces to a torus without introducing discontinuities, that is, mapping some neighboring points on the surface to distant ones on the torus. Information will propagate differently through discontinuities and locality of the convolution would be lost. The second difficulty is geometric: We want mappings to be consistent enough across different surfaces, so that mapped test surfaces look similar to training surfaces. This is related to the space of mappings, or in other words, the number of degrees of freedom that are needed to prescribe a unique map, up to translation of the torus. The more parameters one needs the harder it is to learn from these mappings (*e.g.*, in the worst case, target position of every source point could be a degree of freedom).

We tackle these challenges with a topological construction: instead of dealing with the original sphere-like surface we construct a *cover* of the surface that is made out of four exact copies of the surface and has the topology of the torus, see Figure 1 - top row. Furthermore, we show that this torus can be mapped conformally (preserving orthogonal directions) to the flat torus using a very efficient algorithm. This defines a local translation invariant convolution on the 4-cover, see example of a single convolution stencil in Figure 1 - bottom row. This construction is unique up to a choice of three points on the surface; the convolution depicted in Figure 1 is created by the three points (shown as colored disks) in the bottom-left inset.

This construction provides a six dimensional space of seamless convolutions on a sphere-like surface: Every choice of a triplet of points corresponds to a unique conformal map which in turn defines a convolution operator, or equivalently, a conformal flat-torus structure on the 4-cover of the surface. Since isometries are in particular conformal maps this construction is also invariant to isometric deformations of the shapes. The relatively low dimension of the convolution space allows efficient sampling of this space in the context of data augmentation. The conformality preserves the directionality of the translation directions on the flat-torus but introduces scale changes; in that sense the triplet acts as a magnifying glass - zooming into different parts of the surface.

We employ the above constructions for supervised learning over surfaces. Our goal is to learn a non-linear relation between "easy" vector valued functions over surfaces (*e.g.*, coordinate functions, normals, curvature or other commonly used geometric features) and target "hard" vector valued functions (*e.g.*, semantic segmentation or landmark labeling). The conformal map from the 4-cover to the flat-torus will be used to seamlessly transfer these functions to the flat-torus which will be used as our domain for training. To leverage existing image-based CNN architecture and optimization algorithms on the flat-torus domain, we provide three novel technical components: (i) A cyclic-padding layer replaces zero padding to achieve fully-translational invariance over the flat-torus; (ii) a projection layer on the function space of the surface to properly map functions between the original surface and its 4-cover, and (iii) an aggregation procedure to infer prediction from multiple triplets.

Experiments show that our method is able to learn and generalize semantic functions better than state of the art geometric learning approaches in segmentation tasks. Furthermore, it can use only basic local data (Euclidean coordinates, curvature, normals) to achieve high success rate, demonstrating ability to learn high-level features from a low-level signal. This is the key advantage of defining a local translation invariant convolution operator. Finally, it is easy to implement and is fully compatible with current standard CNN implementations for images.

2 PREVIOUS WORK

Recent advances in convolutional neural networks (CNNs) motivated many researchers to apply these methods to geometric data. Extrinsic representations, such as 3D volumetric grid [Qi et al. 2016b; Wu et al. 2015], 2D projections [Kalogerakis et al. 2016; Su et al. 2015; Wei et al. 2016], or point coordinates [Qi et al. 2016a], have many shortcomings when analyzing non-rigid 2D manifolds: they are sensitive to articulations, they do not leverage the intrinsic structure of the geometry, and only allow very coarse representations. While these limitations can be addressed by analyzing the manifolds directly, applying CNNs to surfaces is challenging because they do not come with a planar parameterization, and thus are not directly suitable for deep learning. One possible way to address this limitation is to represent a surface as a graph of triangles and use spectral convolutions [Bruna et al. 2013; Defferrard et al. 2016; Henaff et al. 2015]. However, this representation does not take advantage of the fact that we are analyzing 2-manifold that can be parameterized in 2D domain. Another disadvantage of spectral methods (which is targeted by [Yi et al. 2016]) is their difficulty with cross-shape learning which stems from the fact that the spectral decomposition of each shape can be inconsistent. We next discuss existing techniques which preform deep learning on 2-manifolds and parameterization methods they use.

For segmentation task, Guo et al. [2015] proposed to lay out per-triangle features to a single 2D grid, and used CNN to classify each triangle. This approach cannot use contextual information on relationships between different triangles on the same surface unless this relationships are encoded in the input features.

The first paper adapting neural networks to surfaces, Masci et al. [2015], use local geodesic polar coordinates to parameterize a surface patch around a point; and map features to this patch. This parameterization requires modifying the traditional convolution filter to account for angular coordinate ambiguity, essentially ignoring patch orientation. In a follow up work, [Boscaini et al. 2016] use anisotropic heat kernels in order to generate a local description of the input function and incorporate orientation.

For classification tasks, Sinha et al. [2016] parameterize the entire surface using geometry images [Praun and Hoppe 2003] combined with spherical area-preserving parameterizations. As mentioned briefly above, geometry images are not seamless and introduce a direction jump at the cut, see Figure 2. Additionally, the convolution over the geometry image is not translation invariant since it represents a sphere-like topology. Finally, since geometry images are computed using area-preserving mappings, which have an infinite number of degrees of freedom, they can produce a wide variability of different planar realizations which will make the learning process

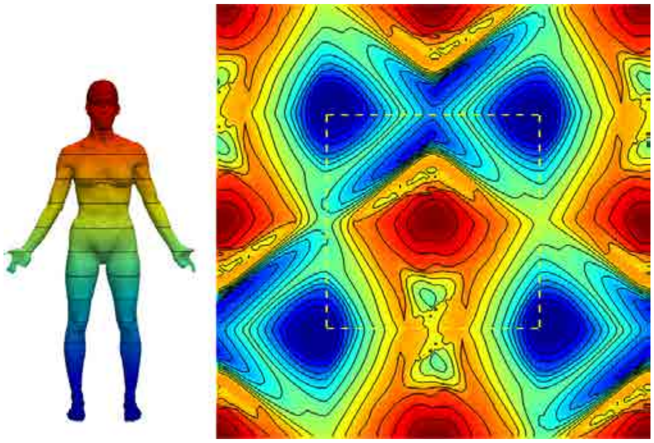

Fig. 2. Parameterization produced by the geometry image method of [Sinha et al. 2016]; the parameterization is not seamless as the isolines break at the dashed image boundary (right); although the parameterization preserves area it produces large variability in shape.

more challenging. See, for example, how one of the hands (green) and one of the legs (blue) are strongly sheared in Figure 2 (one copy of the surface is marked with dashed square; all four corners correspond to one of the legs). Lastly, their parameterization algorithm is not guaranteed to produce a bijective embedding and can cause different parts of the surface to overlap in the geometry image, *e.g.*, only one of the hands is visible in the geometry image in Figure 2.

In contrast to the methods described above, we propose a seamless parameterization technique that maps the surface to a flat torus, thus providing a well-defined convolution everywhere. Our map is uniquely defined by selection of 3 points on the surface, providing a relatively small space of possible parameterizations which makes learning easier. Our map computation routine is very effective, as we only need to solve a sparse system of linear equations per triplet of points.

3 METHOD

Convolutional neural networks (CNN) is a specific family of neural networks that leverages the spatial relationships of local regions using mostly convolutional layers. Deep CNN's with multiple consecutive convolutions followed by nonlinear functions have shown to be immensely successful in image understanding [Krizhevsky et al. 2012]. Our goal is to adapt CNN architectures to geometric surfaces.

3.1 Overview

Problem definition. Our training data consists of a set of triplets $\{(\mathcal{S}_i, x_i, y_i)\}_{i\in I}$ of sphere-like surface meshes $\mathcal{S}_i \subset \mathbb{R}^3$, "easy" d-vector valued functions over the surface $\mathcal{S}_i$, denoted $x_i \in \mathcal{F}(\mathcal{S}_i, \mathbb{R}^d)$, and ground-truth "hard" functions $y_i \in \mathcal{F}(\mathcal{S}_i, \mathcal{L})$, where $\mathcal{L} = \{1, \ldots, L\}$ is a label set. By "easy" functions we mean functions that can be computed efficiently over the surfaces, such as coordinate functions, curvature, normals or shape features; by "hard" we mean functions for which no simple algorithm exists, such as a semantic label (e.g., object part) that has to be prescribed manually.

Our goal is to find a non-linear mapping relating the "easy" and "hard" functions on surfaces. Mathematically we are looking for a function F,

$$F : \mathcal{F}(\mathcal{S}_i, \mathbb{R}^d) \to \mathcal{F}(\mathcal{S}_i, \mathbb{R}^L) \qquad (1)$$

that takes as input a d-vector valued ("easy") function over a surface $\mathcal{S}_i$ and produces a confidence L-vector valued ("hard") function over the same surface. That is, it produces a vector of confidences $F(x_i)[p] \in \mathbb{R}^L_+$ per point $p \in \mathcal{S}_i$ that correctly predicts its ground-truth label $y_i[p] \in \mathcal{L}$ (*i.e.*, the maximal value in $F(x_i)[p]$ is achieved at its $y_i[p]$-th coordinate).

CNN on the flat-torus $\mathcal{T}$. While CNN is a powerful tool for mapping "easy" to "hard" functions, existing architectures cannot run directly over $\mathcal{S}$. Therefore we propose to transfer functions to a flat torus[1], denoted $\mathcal{T}$, and train CNN over that domain. The flat torus space is favorable since we can use a traditional CNN with 2D convolutions directly to solve the problem over $\mathcal{T}$, by discretizing the flat torus space as an $m \times n$ grid (we used $m = n = 512$).

Mapping $\mathcal{S}$ to $\mathcal{T}$ is not trivial because these two domains have different topologies. We address this issue by considering a new topological construction $\mathcal{S}^4$ (Section 3.2). The domain $\mathcal{S}^4$ consists of four copies of the surface cut in the same way to a disk topology and stitched together to one (boundaryless) torus-like surface. We map $\mathcal{S}^4$ conformally to the plane, where these 4 surface copies seamlessly tile the flat-torus. Note that this mapping is not unique, and is defined by a triplet of points on $\mathcal{S}$. Each triplet provides a different image over $\mathcal{T}$ where resolution (surface area scaling) is non-uniform, and varies over $\mathcal{S}$.

We address the mapping ambiguity by modifying network architecture, training data, and the way we interpret the network output (Section 3.3). First, we add a new cyclic-padding layer enabling translation-invariant convolutions (i.e., invariance to torus symmetry). Second, we incorporate a projection operator layer that ensures that our network's output is invariant to symmetries of $\mathcal{S}^4$ (i.e., multiple copies of the input surface). Both of these layers are differentiable and support end-to-end training. Third, we sample multiple triplets to produce multiple training images over $\mathcal{T}$, substantially augmenting our training data. And finally, as we analyze a surface at test time, we aggregate several predictions over the surface (produced with different triplets).

3.2 Transferring functions between $\mathcal{S}$ and $\mathcal{T}$

A key component of our approach is transferring functions between the surface $\mathcal{S}$ and flat torus $\mathcal{T}$. That is, given a function x_i over the surface $\mathcal{S}_i$ we want to transfer it to the flat-torus in a seamless way that preserves locality and topological (*i.e.*, neighboring) relations. We also want this transfer to be as-unique-as-possible and invariant to isometric deformations of $\mathcal{S}_i$ to avoid the need for unnecessary data augmentation. We next show that there is a unique transfer operator for a given triplet of points $\mathcal{P} = \{p_1, p_2, p_3\} \subset \mathcal{S}_i$.

Since $\mathcal{S}$ and $\mathcal{T}$ have different topologies, to create a desired seamless map between these two domains we define an intermediate surface, $\mathcal{S}^4$, a torus-type 4-cover of $\mathcal{S}$ (branched cover, similarly to [Kälberer et al. 2007]). To create $\mathcal{S}^4$ we first make four copies of the

[1] The flat-torus is the planar square $[0, 1]^2$ with its opposite sides identified.

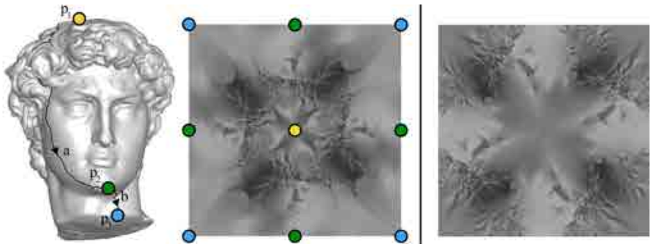

Fig. 3. Computing the flat-torus structure (middle) on a 4-cover of a sphere-type surface (left) defined by prescribing three points (colored disks). The right inset shows the flat-torus resulted from a different triplet choice.

surface and cut each one along the path

$$p_1 \xrightarrow{a} p_2 \xrightarrow{b} p_3$$

to obtain disk-like surfaces (Figure 3, left). Next, we stitch the four surfaces according to the instructions shown in Figure 1, top-right, to get a torus-like surface, $\mathcal{S}^4$. Note that this construction is indifferent to the actual cuts used (*e.g.*, a, b in Figure 3) and different cuts would produce the same surface $\mathcal{S}^4$. Lastly, we compute a map $\Phi_{\mathcal{P}} : \mathcal{S}^4 \to \mathcal{T}$ taking S^4 conformally to $\mathcal{T}$ (see Figure 3, middle). In practice, we compute this map by first mapping a single disk-like copy of $\mathcal{S}$ in $\mathcal{S}^4$ (*e.g.*, Figure 3, left) to the plane using the method in [Aigerman and Lipman 2015] (we used the $\{\pi/2, \pi, \pi/2\}$ orbifold) followed by duplicating the resulting planar mesh until we cover the representative square tile of the flat torus, namely $[0, 1]^2$. For weights we used cotan weights with negative values clamped to 10^{-2} to ensure positivity and hence bijective mapping. Per triplet $\mathcal{P}$, this approximately-conformal map can be computed very efficiently by solving a sparse system of linear equations, where the resulting map defines a 2D position for each vertex of the disk-like $\mathcal{S}$.

We use $\Phi_{\mathcal{P}}$ to transfer functions between the surface $\mathcal{S}$ and the flat-torus $\mathcal{T}$. Given a function $x \in \mathcal{F}(\mathcal{S}, \mathbb{R}^d)$ we define its *push-forward* to the flat-torus, $\text{push}_{\mathcal{P}}(x) \in \mathbb{R}^{m \times n \times d}$, by

$$\text{push}_{\mathcal{P}}(x) = x \circ \Psi \circ \Phi_{\mathcal{P}}^{-1}, \tag{2}$$

where $\Psi : \mathcal{S}^4 \to \mathcal{S}$ is the projection map taking each point in $\mathcal{S}^4$ to its corresponding point in $\mathcal{S}$. That is, given a cell (*i.e.*, pixel) on the discretized torus, we map its centroid to $\mathcal{S}^4$ via the map $\Phi_{\mathcal{P}}^{-1}$, and then to $\mathcal{S}$ via the map Ψ. We evaluate x at that point and assign the corresponding d-dimensional vector value to the cell. In practice, we use Matlab's "patch" command to generate each channel of $\text{push}_{\mathcal{P}}(x)$. Figure 4 visualizes "easy" functions x and their push-forward to $\mathcal{T}$.

An interesting alternative to the above construction of $\mathcal{S}^4$ and the mapping to the flat torus $\mathcal{S}^4 \mapsto \mathcal{T}$ would be to use a single copy of the surface, $\mathcal{S}$, and a mapping to a sphere-type flat representation (Euclidean orbifold) $\mathcal{S} \mapsto O$. This corresponds to taking one quarter of the flat torus (*e.g.*, upper left quarter of the squares in Figure 4). Although this representation is more compact it will not be topologically correct as the convolution kernels will be applied in different orientations to points on different sides of the cut.

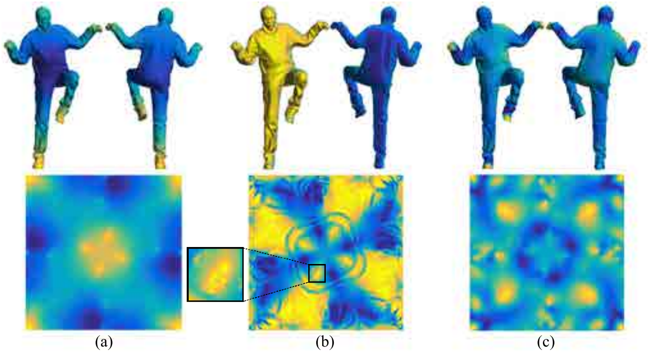

Fig. 4. Visualization of "easy" functions on the surface (top-row) and their pushed version on the flat-torus (bottom-row). We show three examples of functions we use as input to the network: (a) average geodesic distance (left), (b) the x component of the surface normal (middle), and (c) Wave Kernel Signature [Aubry et al. 2011]. The blowup shows the face area, illustrating that the input functions capture relevant information in the shape.

3.3 Neural Networks on the flat-torus $\mathcal{T}$

Now that we are able to map functions between $\mathcal{S}$ and $\mathcal{T}$ we explain how we train CNN over the flat torus. A CNN over the flat-torus is defined as a non-linear function taking d-vector valued functions over the torus to L-vector valued function over the torus. Therefore we denote:

$$f(\cdot, w) : \mathbb{R}^{m \times n \times d} \to \mathbb{R}^{m \times n \times L}, \tag{3}$$

where w denotes the network parameters.

We first describe the appropriate architecture for f on $\mathcal{T}$ that takes into account translational symmetries of $\mathcal{T}$ and the fact it is covered by four copies of the surface (i.e., $\mathcal{S}^4$). To train f, we use multiple triplets $\mathcal{P}_k$ to push training examples on the surface $(\mathcal{S}_i, x_i, y_i)$ to the flat-torus, augmenting our training data by mapping the same surface in different ways. We use these training examples to optimize for w, parameters of the CNN. Lastly, we explain how our trained network can be used to analyze an input test shape.

Network architecture for CNN on $\mathcal{T}$. The input and output of the network $f(\cdot, w)$ is represented in the form of discrete two dimensional images, and there are many state-of-the-art network architectures that have proven highly successful for this type of data. In this work, we used the FCN32 CNN architecture of [Long et al. 2015] with two main differences: First, since we need f to be fully-translation invariant and well-defined on the flat-torus we employ a cyclic padding instead of the standard zero padding used for images. This is crucial for seamless learning (as demonstrated by experiments in Section 5). Second, since there are 4 copies of $\mathcal{S}$ in $\mathcal{S}^4$, several predictions over the flat-torus might correspond to the same point on $\mathcal{S}$. Thus, for the final output of the network $f(\cdot, w) \in \mathbb{R}^{m \times n \times L}$ to be well-defined on $\mathcal{S}$ (so that we can use push^{-1}) we incorporate a projection operator that considers values in the $m \times n$ grid that correspond to the same point in $\mathcal{S}$ and replaces them with their max. Similar to standard pooling layers, averaging corresponding values resulted in inferior results. We implemented two differentiable layers that correspond to these modifications, enabling end-to-end learning for $f(\cdot, w)$. Figure 5 shows the new layers and their incorporation in the network's architecture.

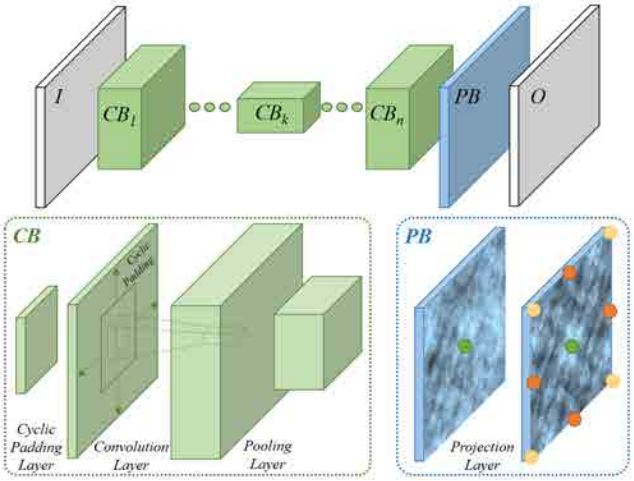

Fig. 5. Top: Segmentation network architecture where CB denotes a convolutional block, and PB denotes a projection block. Bottom: Breakdown of the convolutional and projection blocks. Our network has two new layer types: the cyclic padding layer in each convolutional block, and the final projection layer.

We note that the max-projection layer mentioned above has certain resemblance to the TI-pooling operator introduced in [Laptev et al. 2016] which pools over corresponding pixels of multiple transformed versions of the input image and aim at learning transformation invariant features. In contrast, our layer pools over corresponding points on the surface in order to get a single prediction at every point on the surface.

Data generation. To train the network, we first need to push the training data to images defined over the flat-torus $\mathcal{T}$. Given training data $\{(\mathcal{S}_i, x_i, y_i)\}_{i\in I}$, for each i we sample ρ triplets $\mathcal{P} = (p_1, p_2, p_3) \subset \mathcal{S}_i$ from $\mathcal{S}_i \times \mathcal{S}_i \times \mathcal{S}_i$. Then for each $\mathcal{P}$ we create a pair

$$(X_k, Y_k) = (\text{push}_{\mathcal{P}}(x_i), \text{push}_{\mathcal{P}}(y_i)), \tag{4}$$

where each pair corresponds to training input $X_k \in \mathbb{R}^{m\times n\times d}$ and output $Y_k \in \mathbb{R}^{m\times n\times L}$ directly compatible with $f(\cdot, w)$, and k is an index for $|I|\times\rho$ such pairs. The choice of triplets follow the rationale of well-covering the surface to allow each point to be represented with a reasonable (=not too low) scale factor at-least in one map. Hence, we sampled a small number (5-20) of uniformly spread points (including the AGD local maxima [Kim et al. 2011]) on the surface and randomly sampled triplets from this set.

Training CNN on $\mathcal{T}$. We use this data to train CNN by finding locally optimal parameters w with respect to the following loss function:

$$E(w) = \sum_k \sigma\Big(f(X_k, w), Y_k\Big), \tag{5}$$

where σ is the standard softmax loss per pixel, weighted by $1/(\delta + c)$; c is the size of the pixel's ground-truth class, and $\delta = 4000$ is a regularizer. We used Matconvnet [Vedaldi and Lenc 2015] for training using its SGD (Stochastic gradient descent) with learning rate of 0.0001 as in the original implementation of [Long et al. 2015]. We initialized the network with the parameters of FCN32, removing and/or duplicating channels to fit our data dimension.

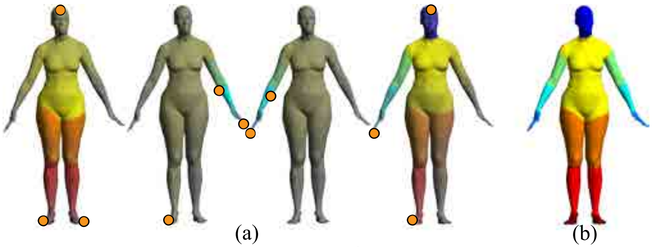

Fig. 6. Aggregating predictions from different triplets (four models on the left; triplets shown as orange disks) to produce final prediction (right). Each triplet serves as a magnifying glass for some local or global part of the surface. Note that on the third shape the third point is on the back side of the model.

Aggregating CNN output on $\mathcal{S}$. Given a new surface $\mathcal{S}$ and corresponding vector valued function x, we use the trained CNN to define the final predictor F via:

$$F(x) = \sum_{\mathcal{P}} S(\mathcal{P}) \odot \text{push}_{\mathcal{P}}^{-1}\Big(f(\text{push}_{\mathcal{P}}(x), w)\Big), \tag{6}$$

where $\mathcal{P}$ is a triplet from a set of ρ random triplets, $S(\mathcal{P})$ is a weight function chosen to compensate for the scale changes induced by the mapping $\Phi_{\mathcal{P}}$, and $\odot$ is pointwise multiplication of functions. The weight function $S(\mathcal{P})$ is taken to be the scale factor of the mapping $\Phi_{\mathcal{P}}$. It is defined at every vertex of the surface using a triangle-area weighted average of the adjacent triangles' scale. Our aggregation method is motivated by the observation that the scale factor can serve as a confidence measure for the CNN prediction at each point of the surface.

Figure 6 depicts an aggregation example where the four left models show the contribution of four different triplets $\mathcal{P}$ visualized using orange disks (gray color corresponds to points with low scale factor), and the model on the right is the final result.

4 PROPERTIES

In this section we justify the choice of the flat torus as the target domain and explain the translation invariance of the new convolution operators. Specifically, we show that the convolution operator on $\mathcal{S}^4$ is invariant to a two dimensional group of conformal translations.

Convolution on the flat torus. We start by considering the Euclidean case, namely the flat torus $\mathcal{T}$. A translation is an isometry $\tau_v : \mathcal{T} \to \mathcal{T}$ defined by $\tau_v(x) = x - v$. Translations act on functions over the torus $\tau_v : \mathcal{F}(\mathcal{T}, \mathbb{R}) \to \mathcal{F}(\mathcal{T}, \mathbb{R})$ via $\tau_v(f)(x) = f(\tau_v(x)) = f(x - v)$. *Translation invariance* of the convolution operator means it commutes with the translation operator as was just defined:

$$\tau_v(f * g) = \tau_v(f) * g$$

Conversely, under certain conditions, one can show that a linear and translation invariant operator is a convolution with some fixed function g [Davies 2007]. Therefore, linearity and translation invariance are defining properties of convolution.

Translations on surfaces. To define a convolution operator on a surface $\mathcal{S}$, a natural requirement is that it would be linear and translation invariant. But what is a translation $\tau : \mathcal{S} \to \mathcal{S}$? In tune with the definition of a surface, a translation on a surface should locally be a Euclidean translation. That is, a flow along a non-vanishing vector field. According to the Poincaré-Hopf Theorem [Milnor 1965] the only surfaces with non-vanishing vector fields are of Euler characteristic zero - in case of closed orientable surfaces, the torus. This implies that the only surfaces on which we can define translations in the above mentioned way are toric.

The pullback convolution. Given two toric (not necessarily flat) surfaces $\mathcal{T}_1, \mathcal{T}_2$ and a homeomorphism $\Phi : \mathcal{T}_1 \to \mathcal{T}_2$ one can define a convolution operator $*_1$ on $\mathcal{T}_1$ from a convolution operator $*_2$ defined on $\mathcal{T}_2$ via

$$f *_1 g = \left(\left(f \circ \Phi^{-1}\right) *_2 \left(g \circ \Phi^{-1}\right)\right) \circ \Phi,$$

The pullback convolution $*_1$ is linear and translation invariant w.r.t. the *pullback translations* $\Phi^{-1} \circ \tau \circ \Phi$, where τ represents translations in $\mathcal{T}_2$ for which $*_2$ is invariant to. This is proved in the appendix.

In our case $\mathcal{T}_1 = \mathcal{S}^4$, $\mathcal{T}_2 = \mathcal{T}$ the flat-torus with the Euclidean convolution and translations (modulo 1), and the mapping $\Phi : \mathcal{S}^4 \to \mathcal{T}$ is a conformal homeomorphism. As the convolution on the flat-torus is invariant to Euclidean translations τ of the type $x \mapsto x - v(\text{mod } 1)$, the pullback convolution on $\mathcal{S}^4$ is invariant to the pullback translations $\Phi^{-1} \circ \tau \circ \Phi$. Since Φ, Φ^{-1}, τ are all conformal maps, these pullback translations are conformal maps as well. To visualize an example of these translations consider Figure 1 (bottom left) and imagine that each square on the David surface moves to a well-defined "right" neighbor.

5 EVALUATION

In this section, we compare our method to alternative approaches for surface parameterization and network architecture. We compare to three other methods: The first two methods focus on the parameterization component of our algorithm, the last one on the architecture:

(i) GIM - where we used the geometry images parameterization method of [Sinha et al. 2016] followed by application of the FCN32 network [Long et al. 2015]. To the best of our knowledge this is the only parameterization method used with CNNs.

(ii) Tutte - we consider parameterization using Tutte's embedding [Tutte 1963] to the unit square followed by application of the FCN32 network [Long et al. 2015]. To generate the embedding, we use the same cut as in our method and map the boundary in a length-preserving manner (up to global scale) to the boundary of the unit square. This is a natural selection for comparison, since Tutte's embedding can be computed as-efficiently as our method by solving a sparse system of linear equations.

(iii) Seamless+FCN - where we use our parameterization technique but without the two additional layers of cyclic padding and projection. This is equivalent to considering the flat-torus as a disk with its opposite sides disconnected.

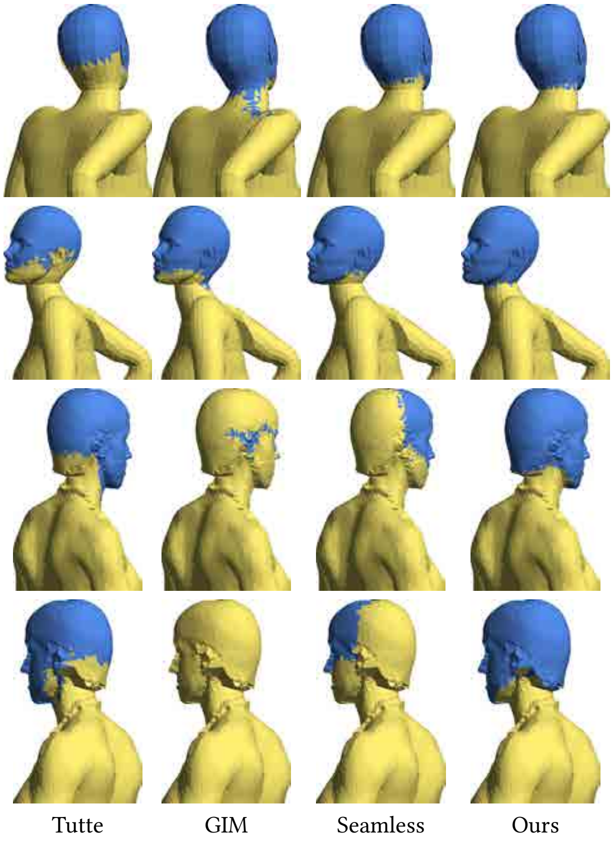

Fig. 7. Head segmentation on two test surfaces by the four algorithms in Table 1. Our algorithm produces accurate segmentation, while competing methods provide suboptimal results.

We perform the evaluation on the task of segmenting the head in human models. Our training set for this task is composed of 370 models from the SCAPE [Anguelov et al. 2005], FAUST [Bogo et al. 2014] and MIT animation datasets [Vlasic et al. 2008]. The ground truth labels are head indicator functions that are manually labeled. Our test data are the 18 sphere-like human models from the SHREC dataset [Giorgi et al. 2007]. We use *intersection-over-union* as our evaluation metric: Denoting by GT_{Head} the set of faces labeled "head" and by Alg_{Head} the faces labeled "head" by the algorithm, this metric as defined as

$$\frac{|\text{GT}_{\text{Head}} \cap \text{Alg}_{\text{Head}}|}{|\text{GT}_{\text{Head}} \cup \text{Alg}_{\text{Head}}|}$$

weighted by triangle areas. In all experiments, as input features ("easy" functions over the surfaces), we use a set of 26 basic shape features: 21 WKS features (equally spaced), 4 curvature features (max, min, arithmetic mean and geometric mean of the principal curvatures) and average geodesic distance (AGD) as input. We initialize training with the parameters obtained by the FCN32 net [Long et al. 2015]. We trained all networks with the same parameters and same number of epochs.

For all methods we considered $\rho = 10$ different parameterizations per mesh, resulting in a dataset with 3700 segmented images of

	intersection-over-union
1. Geometry images + FCN	0.625
2. Tutte + FCN	0.567
3. Seamless + FCN	0.503
4. Ours	**0.710**

Table 1. Evaluation of CNN on surfaces with different parameterization methods and network architectures.

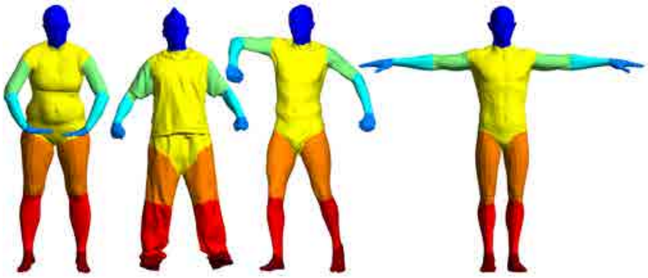

Fig. 8. Examples from the semantic segmentation training set.

size 512 × 512. For all methods excluding GIM, the different parameterizations correspond to $\rho = 10$ choices of triplets. For GIM the different parameterizations correspond to 10 uniform rotations around the polar axis of the sphere as suggested in [Sinha et al. 2016]. All networks converged after 20 epochs. For GIM we tried applying aggregation like in Eq. (6) with two choices of parameterization weighting: scale dependent (like in our method), and uniform (all predictions get the same weight). The rationale in the second version is that GIM uses area preserving parameterizations so theoretically no scaling is introduced. Both aggregation methods produced the same results.

Table 1 summarizes the results. Our method achieves superior results with respect to all alternatives. The results indicate that the layers added to the network to account for the flat-torus topology indeed play an important role.

Figure 7 shows results of the head segmentation task of the four algorithms on two different models from the test set. The first two rows show the first model from two viewing directions, and the third and fourth rows show the second model. For both models our segmentation was satisfactory while the segmentation of the remaining methods was suboptimal. The small variability in the head shape in our parametric space, as well invariance to cuts that could pass through the head, allow our method to learn the head function to a greater accuracy.

In an additional experiment we replaced the weighted aggregation method described in (6) with maximal aggregation method and found that the performance of all methods degraded, with our algorithm still providing superior results to the alternative methods.

6 APPLICATIONS

In this section we demonstrate the usefulness of our method for two applications: semantic segmentation and automatic landmark detection on anatomical surfaces.

method	# feat	features used	accuracy
Ours	10	normals, Euclidean, curv.	81.6%
Guo	10	normals, Euclidean, curv.	43.6%
Ours	26	WKS, AGD, curv.	**88.0**%
Guo	26	WKS, AGD, curv.	76.0%
Guo	600	HKS, WKS, AGD, curv.	87.8%

Table 2. Semantic segmentation results.

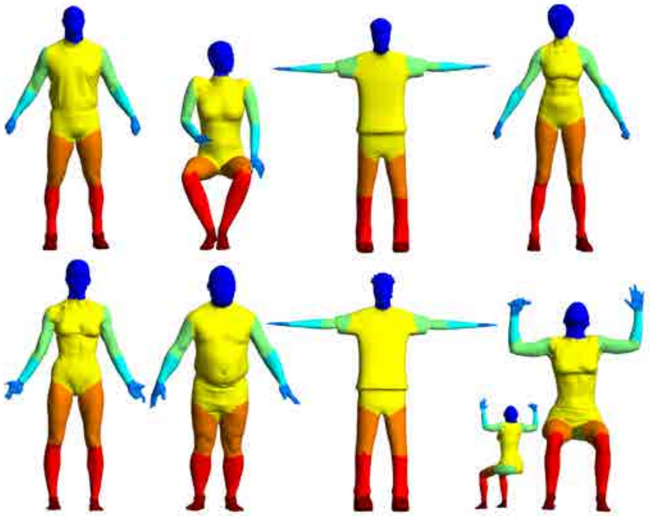

Fig. 9. Results of our method for human body segmentation. Bottom right: Failure case where part of the thigh was incorrectly identified.

6.1 Semantic segmentation of surfaces

We applied our algorithm to the task of semantic segmentation of human models. As training data we used 370 models from SCAPE, FAUST, MIT (excluding two classes which are not suitable for full-body segmentation), and Adobe Fuse [fus 2016]. All models are manually segmented into eight labels according to the labels in [Kalogerakis et al. 2010]. Our test set is again the 18 models from the SHREC07 dataset in human category (all sphere-like models). Note that, in contrast with previous works, the training set does not include any models from the SHREC07 dataset which we use solely for testing. Figure 8 shows examples from the training set.

We generated data from 300 triplets per model in the training set which resulted in approximately 110K segmented images. The networks converged after 30-50 epoches. We compared our method to the state-of-the-art CNN method for mesh segmentation [Guo et al. 2015]. Both methods were trained on the same training data. Gou et al. also use convolutional nets, but on per-triangle features reshaped to a square grid. These convolutions do not enable aggregating information from nearby regions on the surface, so to get global context Guo et al. need to leverage global features. The training set for Guo et al. was created by randomly sampling $\sim 360k$ triangles from the set of training models. We applied their method with different sets of training features (ranging from 10 to 600) and as shown in Table 2 their performance drops considerably as the number of training features decreases; accuracy is measured as ratio of correctly labeled triangles, weighted by triangle areas. In contrast, our method can learn features by leveraging convolution defined

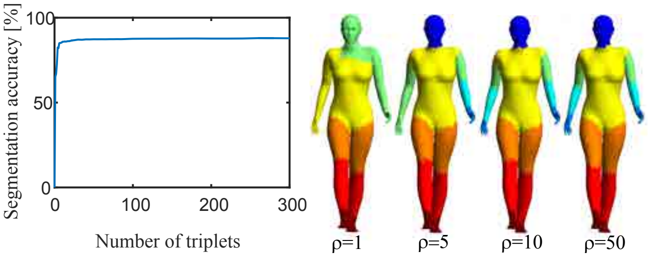

Fig. 10. Average result of human body segmentation as a function of the number of triplets used in the prediction (using 26 features on the SHREC07 dataset).

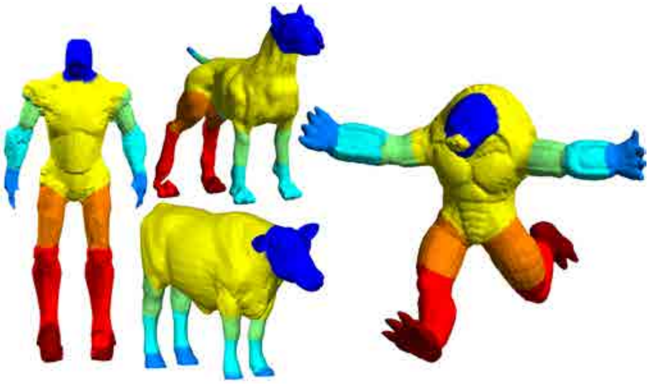

Fig. 11. Results of our algorithm trained solely on human body semantic segmentation applied to surfaces from other classes. Note that the method still produces semantic plausible results, which demonstrates the strong generalization power of the method.

over the surface, demonstrating consistently better performance with the same number of features. In fact, our method, using only 26 features, modestly outperforms the method of Guo et al. with 600 features, see Table 2 (for evaluation we used $\rho = 260/480$ triplets for 26/10 features, respectively). Figure 9 shows several results obtained with our algorithm.

Figure 10 shows the segmentation accuracy for the above 26 feature experiment as a function of the number of triplets ρ used for aggregation. Using 5 triplets already provides meaningful results, while 10 triplets are almost identical to the final result with $\sim$ 300 triplets.

Figure 11 shows successful application of the human segmentation method to shapes from other classes: robot [Yob 2016], armadillo and four-legged [Giorgi et al. 2007]. Note that although the training set contains only standard human segmentation data (see Figure 8), the network still produces plausible segmentations on this very different set of models. This demonstrates the robustness and generalization power of our method.

The robot example also demonstrates a possible way to bypass the sphere topology restriction of our method: In this case the input surface contains multiple connected components and non-manifold edges. We approximated the input with a genus-0 surface using a simple reconstruction algorithm [Zhao et al. 2000] and applied our method to it. The result (shown on the approximated surface) is plausible.

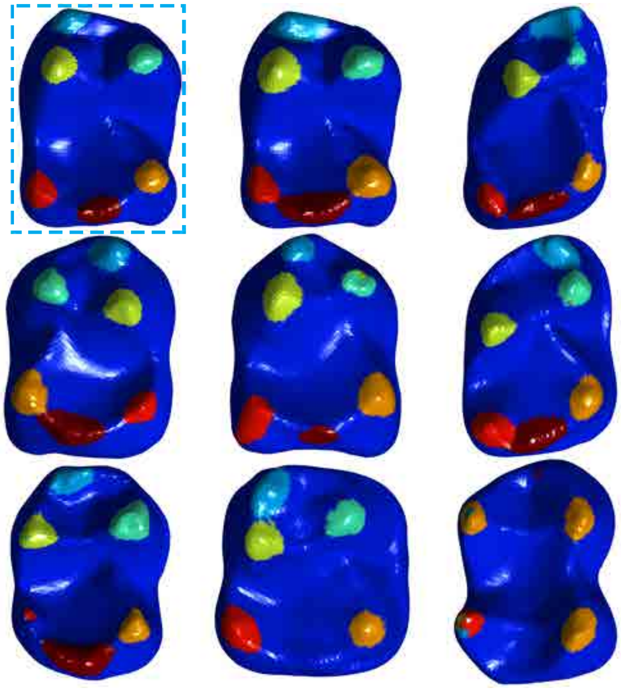

Fig. 12. Landmark detection in anatomical surfaces. Results on the test set are shown; in dashed rectangle we show ground-truth for the tooth in the middle of the first row.

6.2 Landmark detection on anatomical surfaces

Our algorithm can be applied to automatic landmark detection on 3D shapes in general and on biological data in particular. Here we show results of an experiment we conducted on models of animal teeth from the [Boyer et al. 2011] dataset. On each tooth 6 biologically significant landmarks where manually marked by experts. Our task here is to detect these landmarks on an unseen tooth. For this purpose we took 81 teeth from this dataset, converted them to sphere topology using [Ju 2004] and marked each landmark area using a geodesic disk of constant radius. We trained on a random subset of 73 teeth and tested on 8 models. For each tooth we generated 125 triplets, resulting in a training set of approximately 9K images. For the input "easy" function we only used curvature and (the logarithm of) the conformal scale factor. The network converged after 50 epoches.

Figure 12 shows the results we obtained on all eight test models. In a dashed rectangle we show the ground-truth labeling for the tooth in the middle of the first row. Although we used only a five-dimensional vector of basic features our method was successful in identifying most of the landmarks despite the local as-well as global variability in the data.

This dataset contains both right-side as-well as left-side teeth. The landmarks are therefore reflected when comparing right- and left-side teeth. Remarkably, our algorithm is able to correctly label landmarks on both right- and left-side teeth according to their biological meaning and is not "fooled" by their orientation (see the first tooth in the second row).

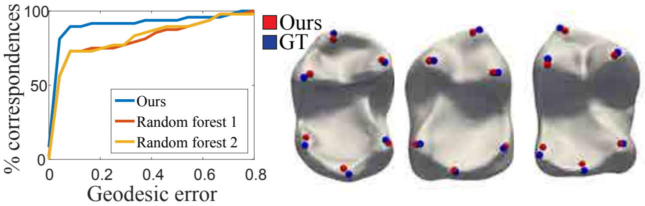

Fig. 13. Quantitative evaluation of the landmarks extracted by our method.

In the bottom row of Figure 12, one of the landmark areas in the first tooth from the left is small, but is still detected. In the middle tooth 5/6 landmarks were detected successfully and one is missing (possibly due to the lack of the ridge on which the missing landmark is usually marked). In the last tooth on the right only one landmark was detected successfully. The failure in this case may be related to the fact that both the genus of the animal and the specific peak structure are not represented in the training set [Maier 1984]. This experiment further demonstrates the ability of our method to learn high-level semantic data from low-level information on the surface.

Using the output of our algorithm, we extract the point landmarks by computing the geodesic centroid of each label. Figure 13 shows a quantitative evaluation of the keypoints we extracted using the above method compared to ground truth. We plot the fraction of the predicted points (y-axis) that are within a certain geodesic error threshold of their true position (x-axis). We compared our algorithm to a baseline random forest classifier [Breiman 2001] (using maltab's implementation) which was recently shown to be a successful classifier for shape analysis problems [Rodolà et al. 2014]. The input per-face feature vectors consisted of the 600 WKS, HKS, curvature and AGD as before, with additional (logarithm of) the conformal scale factor generate with 125 triplet (generated as above). We tried two versions - the first with 50 trees and 73K sampled faces and the second with 100 trees and 292K sampled faces. Our algorithm was able to extract more accurate landmarks despite the large number and expressive power of the features fed to the baseline.

In Figure 14 we show a smooth bijective map between a pair of teeth obtained by interpolating the 6 landmarks identified by our method. The map was obtained using the method of [Aigerman and Lipman 2016]. This provides a fully automatic pipeline for producing semantically correct mappings between biological surfaces.

6.3 Timing

We present average running times. Computing the parameterization on a mesh with 12.5K vertices takes 0.862 seconds for a given triplet of cones. Computing the scale factor of the parametrization (used for aggregation) takes 0.534 seconds. The training can process about one image with 5-26 channels per second for a single GPU in Nvidia K80. We used three such dual GPUs which made the training 6 times faster. For a dataset containing 110k images (as in the full segmentation experiment) a single epoch takes about 5 hours. Feed-forward calculation in the network (using a single GPU of Nvidia K80) takes 0.35 seconds on average for a single image with 5-26 channels. Full prediction for a single triplet (feed-forward and pull-back of functions to the surface) takes about 2.94 seconds, and this process can be parallelized for multiple triplets. Consequently, in case of sequential runs, a prediction on a single model with 1/10/50 triples takes 3/30/150 seconds. Using 50 triples, it takes 45 minutes to calculate predictions on the human class of SHREC07 and 20 minutes on the teeth dataset. These experiments were done on an Intel Xeon E5 CPU with 64GB of RAM.

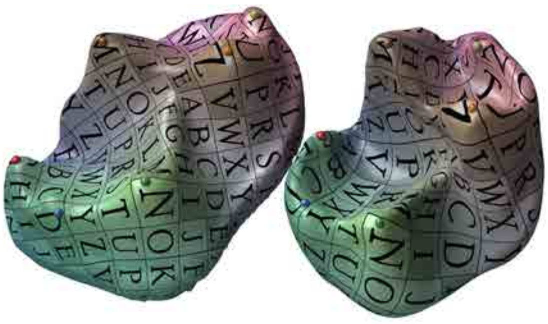

Fig. 14. A visualization of a map between teeth obtained by interpolating the correspondence between the 6 landmarks found automatically by our method.

7 CONCLUSION

We presented a methodology and an algorithm for applying deep convolutional neural networks to geometric surfaces. The algorithm is based on seamless, conformal mapping of surfaces to the flat-torus on which convolution is well defined. Standard CNN architecture can then be used with minor modifications to perform supervised learning on the flat-torus. We demonstrated the usefulness of our approach for semantic segmentation and automatic landmark detection on anatomical surfaces, and showed it compares favorably to competing methods.

A limitation of our technique is that it assumes the input shape is a mesh with a sphere-like topology. An interesting direction for future work is extending our method to meshes with arbitrary topologies. This problem is especially interesting since in certain cases shapes from the same semantic class may have different genus. Another limitation is that currently aggregation is done as a separate post-process step and not as a part of the CNN optimization. An interesting future work in this regard is to incorporate the aggregation in the learning stage and produce end-to-end learning framework.

8 ACKNOWLEDGEMENTS

This research was supported in part by the European Research Council (ERC Starting Grant "Surf-Comp", Grant No. 307754), I-CORE program of the Israel PBC and ISF (Grant No. 4/11). The authors would like to thank Ayan Sinha and Karthik Ramani for sharing their code, Shahar Kovalsky and Julia Winchester for providing biological insights, and the anonymous reviewers for their useful comments and suggestions.

REFERENCES

2016. Adobe Fuse 3D Characters. https://www.mixamo.com. (2016). Accessed: 2016-10-15.

2016. Yobi3d - free 3d model search engine. https://www.yobi3d.com. (2016). Accessed: 2016-10-15.

Noam Aigerman and Yaron Lipman. 2015. Orbifold tutte embeddings. *ACM Trans. Graph* 34, 6 (2015), 190.

Noam Aigerman and Yaron Lipman. 2016. Hyperbolic orbifold tutte embeddings. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 217.

Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. 2005. SCAPE: shape completion and animation of people. In *ACM Transactions on Graphics (TOG)*, Vol. 24. ACM, 408–416.

Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. 2011. The wave kernel signature: A quantum mechanical approach to shape analysis.. In *ICCV Workshops*. IEEE, 1626–1633. http://dblp.uni-trier.de/db/conf/iccvw/iccvw2011.html#AubrySC11

Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. 2014. FAUST: Dataset and evaluation for 3D mesh registration. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Piscataway, NJ, USA.

Davide Boscaini, Jonathan Masci, Simone Melzi, Michael M Bronstein, Umberto Castellani, and Pierre Vandergheynst. 2015. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 13–23.

Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael M. Bronstein. 2016. Learning shape correspondence with anisotropic convolutional neural networks. In *NIPS*.

Doug M Boyer, Yaron Lipman, Elizabeth St Clair, Jesus Puente, Biren A Patel, Thomas Funkhouser, Jukka Jernvall, and Ingrid Daubechies. 2011. Algorithms to automatically quantify the geometric similarity of anatomical surfaces. *Proceedings of the National Academy of Sciences* 108, 45 (2011), 18221–18226.

Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013).

Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2014. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062* (2014).

E Brian Davies. 2007. *Linear operators and their spectra*. Vol. 106. Cambridge University Press.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*. 3837–3845.

Daniela Giorgi, Silvia Biasotti, and Laura Paraboschi. 2007. Shape retrieval contest 2007: Watertight models track. *SHREC competition* 8 (2007).

Xianfeng Gu, Steven Gortler, and Hugues Hoppe. 2002. Geometry Images. In *SIGGRAPH*.

Kan Guo, Dongqing Zou, and Xiaowu Chen. 2015. 3D Mesh Labeling via Deep Convolutional Neural Networks. *ACM Trans. Graph.* 35, 1 (2015).

Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* (2015).

Tao Ju. 2004. Robust repair of polygonal models. *ACM Transactions on Graphics (TOG)* 23, 3 (2004), 888–895.

Felix Kälberer, Matthias Nieser, and Konrad Polthier. 2007. QuadCover-Surface Parameterization using Branched Coverings. In *Computer Graphics Forum*, Vol. 26. Wiley Online Library, 375–384.

Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 2016. 3D Shape Segmentation with Projective Convolutional Networks. *arXiv preprint arXiv:1612.02808* (2016).

Evangelos Kalogerakis, Aaron Hertzmann, and Karan Singh. 2010. Learning 3D mesh segmentation and labeling. *ACM Transactions on Graphics (TOG)* 29, 4 (2010), 102.

Vladimir G Kim, Yaron Lipman, and Thomas Funkhouser. 2011. Blended intrinsic maps. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 79.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*. 1097–1105.

Dmitry Laptev, Nikolay Savinov, Joachim M Buhmann, and Marc Pollefeys. 2016. TI-POOLING: transformation-invariant pooling for feature learning in Convolutional Neural Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 289–297.

Roee Litman and Alexander M Bronstein. 2014. Learning spectral descriptors for deformable shape correspondence. *IEEE transactions on pattern analysis and machine intelligence* 36, 1 (2014), 171–180.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3431–3440.

Wolfgang Maier. 1984. Tooth morphology and dietary specialization. In *Food acquisition and processing in primates*. Springer, 303–330.

Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. 2015. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 37–45.

J Milnor. 1965. Topology from a differentiable viewpoint(University of Virginia Press, Charlottesville, VA). (1965).

Emil Praun and Hugues Hoppe. 2003. Spherical parametrization and remeshing. In *ACM Transactions on Graphics (TOG)*, Vol. 22. ACM, 340–349.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2016a. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *arXiv preprint arXiv:1612.00593* (2016).

Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas Guibas. 2016b. Volumetric and Multi-View CNNs for Object Classification on 3D Data. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*.

Emanuele Rodolà, Samuel Rota Bulo, Thomas Windheuser, Matthias Vestner, and Daniel Cremers. 2014. Dense non-rigid shape correspondence using random forests. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4177–4184.

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

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. 2015. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision*. 945–953.

Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. 2009. A Concise and Provably Informative Multi-scale Signature Based on Heat Diffusion. In *Proceedings of the Symposium on Geometry Processing (SGP '09)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 1383–1392. http://dl.acm.org/citation.cfm?id=1735603.1735621

Federico Tombari, Samuele Salti, and Luigi Di Stefano. 2010. Unique signatures of histograms for local surface description. In *European Conference on Computer Vision*. Springer, 356–369.

W. T. Tutte. 1963. How to draw a graph. *Proc. London Math. Soc.* 13, 3 (1963), 743–768.

Andrea Vedaldi and Karel Lenc. 2015. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 689–692.

Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. 2008. Articulated mesh animation from multi-view silhouettes. In *ACM Transactions on Graphics (TOG)*, Vol. 27. ACM, 97.

Lingyu Wei, Qixing Huang, Duygu Ceylan, Etienne Vouga, and Hao Li. 2016. Dense Human Body Correspondences Using Convolutional Networks. In *Proc. CVPR*.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1912–1920.

Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. 2016. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Advances in Neural Information Processing Systems*. 1696–1704.

Li Yi, Hao Su, Xingwen Guo, and Leonidas Guibas. 2016. SyncSpecCNN: Synchronized Spectral CNN for 3D Shape Segmentation. *arXiv preprint arXiv:1612.00606* (2016).

Hong-Kai Zhao, Stanley Osher, Barry Merriman, and Myungjoo Kang. 2000. Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method. *Computer Vision and Image Understanding* 80, 3 (2000), 295–314.

9 APPENDIX

We prove that given a bijection $\Phi : \mathcal{T}_1 \to \mathcal{T}_2$ and assuming $\tau(f *_2 g) = \tau(f) *_2 g$ we have that $\overline{\tau}(f *_1 g) = \overline{\tau}(f) *_1 g$ where $\overline{\tau} = \Phi^{-1} \circ \tau \circ \Phi$. As before, we use the notation $\tau(f) = f \circ \tau$. First,

$$\begin{aligned}
\overline{\tau}(f *_1 g) &= \left[(f \circ \Phi^{-1}) *_2 (g \circ \Phi^{-1})\right] \circ \Phi \circ \left[\Phi^{-1} \circ \tau \circ \Phi\right] \\
&= \left[(f \circ \Phi^{-1}) *_2 (g \circ \Phi^{-1})\right] \circ \tau \circ \Phi \\
&= \tau \left[(f \circ \Phi^{-1}) *_2 (g \circ \Phi^{-1})\right] \circ \Phi
\end{aligned}$$

On the other hand we have

$$\begin{aligned}
\overline{\tau}(f) *_1 g &= \left[(\overline{\tau}(f) \circ \Phi^{-1}) *_2 (g \circ \Phi^{-1})\right] \circ \Phi \\
&= \left[(f \circ \Phi^{-1} \circ \tau \circ \Phi \circ \Phi^{-1}) *_2 (g \circ \Phi^{-1})\right] \circ \Phi \\
&= \left[(f \circ \Phi^{-1} \circ \tau) *_2 (g \circ \Phi^{-1})\right] \circ \Phi \\
&= \left[\tau(f \circ \Phi^{-1}) *_2 (g \circ \Phi^{-1})\right] \circ \Phi \\
&= \tau \left[(f \circ \Phi^{-1}) *_2 (g \circ \Phi^{-1})\right] \circ \Phi,
\end{aligned}$$

where in the last equality we used the invariance of $*_2$ to τ. We proved invariance of $*_1$ to $\overline{\tau}$.