Eurographics Symposium on Geometry Processing 2017
J. A. Bærentzen and K. Hildebrandt
(Guest Editors)

*Volume 36 (2017), Number 5*

# GWCNN: A Metric Alignment Layer for Deep Shape Analysis

Danielle Ezuz,[1] Justin Solomon,[2] Vladimir G. Kim,[3] and Mirela Ben-Chen[1]

[1]Technion - Israel Institute of Technology
[2]MIT
[3]Adobe Research

**Abstract**

*Deep neural networks provide a promising tool for incorporating semantic information in geometry processing applications. Unlike image and video processing, however, geometry processing requires handling unstructured geometric data, and thus* data representation *becomes an important challenge in this framework. Existing approaches tackle this challenge by converting point clouds, meshes, or polygon soups into regular representations using, e.g., multi-view images, volumetric grids or planar parameterizations. In each of these cases, geometric data representation is treated as a fixed pre-process that is largely disconnected from the machine learning tool. In contrast, we propose to optimize for the geometric representation during the network learning process using a novel* metric alignment *layer. Our approach maps unstructured geometric data to a regular domain by minimizing the metric distortion of the map using the regularized Gromov–Wasserstein objective. This objective is parameterized by the metric of the target domain and is differentiable; thus, it can be easily incorporated into a deep network framework. Furthermore, the objective aims to align the metrics of the input and output domains, promoting consistent output for similar shapes. We show the effectiveness of our layer within a deep network trained for shape classification, demonstrating state-of-the-art performance for nonrigid shapes.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

## 1. Introduction

Recent advances in acquisition and modeling tools for 3D geometry, as well as the rising popularity of user-facing applications such as VR, have led to unprecedented growth in geometric data. This necessitates effective shape analysis algorithms that predict semantic attributes of shapes, essential for organizing, searching, and using geometric datasets for content creation. Deep neural networks hold significant promise for these tasks, providing a fundamental tool for learning a mapping from low-level geometric features to high-level semantic attributes. Unfortunately, these networks usually operate on regular inputs such as $n$-D grids, which are not natural representations for geometric data. Thus, most existing network architectures pre-process the data by converting unstructured point or triangle surface samples to regular representations.

One common way to do this conversion is to rasterize surfaces to volumetric grids, where each cell either stores a binary occupancy function [QSN*16] or a distance to the closest surface point [SX16]. This leads to surface representations that only capture coarse geometry. One can also project 3D models to multiple external camera planes and analyze the corresponding images [SMKLM15]; this leads to redundancy in representation and is only suitable for cases when the majority of the surface is visible from a small number of pre-defined cameras. The main limitation of the extrinsic approaches described above is that they are sensitive to articulation and deformation. A common way to circumvent this problem is to analyze the surfaces intrinsically. To this end, previous work explored flattening techniques such as geometry images [SBR16] and local geodesic polar coordinates [BMRB16]. While these pre-processing steps typically embed surfaces to a regular domain enabling application of convolutional neural networks, existing embedding procedures stay fixed across all problems and cannot be tailored specifically to the task.

Our main contribution is a *parametric* and *differentiable* mapping layer that can be optimized for a specific problem and dataset. Our key idea is to leverage an efficient algorithm that optimizes the regularized Gromov–Wasserstein (GW) objective [SPKS16] to map from unstructured data to a regular representation. Unlike most correspondence algorithms, this regularized technique is differentiable in the geometries of the mapped domains, making it amenable to gradient-based optimization techniques.

Our pipeline is visualized in Figure 1. Given an input shape and scalar geometric features, our layer maps the features to a common 2D grid. This yields a multi-channel image over the grid that we feed into standard deep architectures. As we optimize the GW

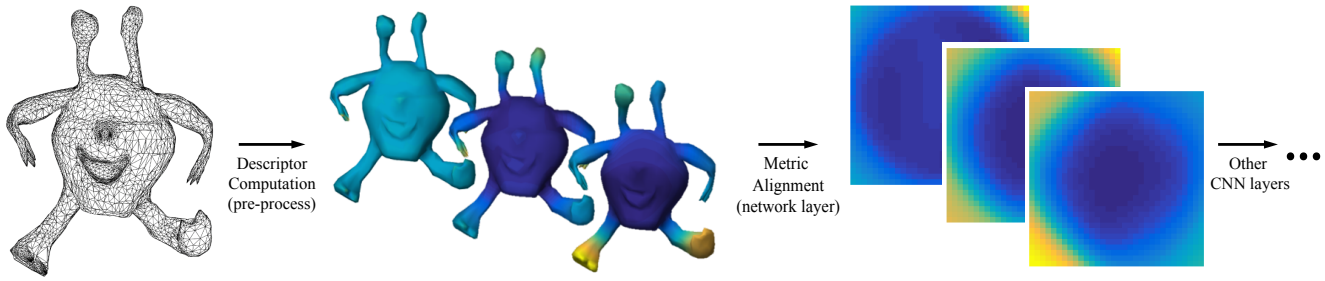


**Figure 1:** *Our pipeline. We first compute point-wise surface features that are mapped to a stack of 2D functions over a canonical domain via our novel metric alignment layer (see Fig. 5). This provides a natural input for subsequent CNN layers.*

objective [SPKS16], our input is given in the form of a *pairwise distance matrix* and thus our layer can handle polygonal meshes, point clouds or general graphs as long as pairwise distances are computable. Further, the GW map minimizes distortion in pairwise distances between the source and the target, leading to mapped features that are *consistent* under isometric deformation of the source geometry. Finally, we include the geometry of the 2D grid as a variable during the learning stage, *learning* task-optimal mappings from unstructured domains to the regular domain. We implement our layer with the Torch library [CKF11] and use it within a deep architecture for shape classification. Results demonstrate that our approach outperforms state-of-the-art methods on standard benchmarks for non-rigid shape classification and retrieval.

## 2. Related Work

A wide range of fundamental shape analysis problems such as classification [BBGO11], segmentation [KHS10], and correspondence [COC14] have been addressed with machine learning techniques (see [XKH*16] for a survey). Due to recent developments and success of deep neural networks, researchers have focused on developing appropriate shape representations suitable for deep learning. In this section, we overview relevant extrinsic and intrinsic representations as well as other work related to our approach.

### 2.1. Representations for Deep Learning

**Extrinsic.** One straightforward representation for geometry is a 3D voxel grid, where each voxel stores a binary occupancy function [MS15] or a truncated distance to the surface [SX16]. Deep CNNs have been used to analyze these grids for shape classification [WSK*15] and geometric modeling [BLRW16, WZX*16]. This representation is inefficient since typical surfaces occupy only a small fraction of the volume. Another alternative is to project surfaces to external camera planes. One can analyze rendered images [SMKLM15, KAMC17] or depth images from multiple viewpoints [WHC*16] or from panoramic views [SBZB15]. These techniques only work for shapes for which all relevant geometric details are visible from a fixed set of external cameras. Qi et al. [QSN*16] proposed view-dependent volumetric analysis with anisotropic kernels, closing the gap between multi-view and volumetric approaches.

One can also directly analyze features of unordered surface elements such as mesh faces [GZC15], but this approach requires powerful features that provide additional contextual information. Concurrently with our work, Qi et al. [QSMG17] proposed a novel network architecture that directly analyzes coordinates of unordered point sets by using symmetric order-independent max pooling functions.

Extrinsic techniques are sensitive to articulation and non-rigid deformation, a common problem in shape analysis addressed via intrinsic shape representations. Such representations often map the shape to a common domain in a way that is invariant to nearly-isometric deformation of the input.

**Spectral.** Several methods have been proposed for spectral analysis of functions on a graph [BZSL13, HBL15, DBV16, KW16]. While these methods are invariant to isometry, they are limited to analyzing functions on a *specific* non-Euclidean domain (e.g. a graph with fixed connectivity) and thus cannot be used to analyze different geometries. Concurrent to our approach, Yi et al. [YSGG17] propose to synchronize the spectral domains of the graphs to enable cross-shape analysis. They create canonical shape domains (described by their graph Laplacian eigenbases) as a preprocess, and use extrinsic alignments between shapes to initialize functional maps [OBCS*12] to the common domain. While potentially applicable to non-rigid shapes, they focus their analysis on rigid man-made objects and use the consistent extrinsic alignment provided in the ShapeNet dataset for the initialization.

**Local mapping.** Masci et al. [MBBV15] use geodesic polar coordinates to parameterize a surface locally around every point. Boscaini et al. [BMRB16] improve this approach with anisotropic patches. These techniques operate on relatively small geodesic patches which limits their ability to incorporate global context.

**Global mapping.** To remedy this, Sinha et al. [SBR16] proposed to use geometry images [GGH02], a global shape parameterization technique for manifold genus zero surfaces. To parameterize point clouds or polygon soups, Sinha et al. use $\alpha$-shapes [EM94] to create a manifold input, and topological processing to ensure the input is genus zero. Unfortunately, this pre-processing does not preserve the original geometric details, such as the interior structure of the model. Furthermore, the same surface can be parameterized in multiple ways depending on the placement of cuts. Thus, this method also suffers from inconsistent mappings across similar surfaces.

In contrast to existing approaches, our metric alignment layer is based on minimizing the regularized GW objective and thus explicitly optimizes for consistency in aligning surfaces to the regular domain. Moreover, we learn the geometry of the canonical domain specifically for the task at hand.

### 2.2. Shape Parameterization and Mapping.

Mapping a shape to another domain is a common task in shape analysis with applications to texture mapping [LPRM02], modeling [PSS01], correspondence [AL16], and retrieval [SK16]. Surface parameterization techniques [HLS08] typically require manifold surfaces and do not aim to embed similar shapes consistently. Spectral methods [ZvKD10] usually rely on the first few eigenfunctions of a Laplacian operator and thus only encode low-frequency attributes of shapes. Correspondence techniques map between arbitrary geometric domains [AL16, OBCS*12] while minimizing some distortion metric, thus offering a certain degree of consistency when mapping similar shapes. However, it is not immediately clear how to optimize the embedding process for these methods with respect to a back-propagated loss function of the neural network.

Typical deep learning algorithms require the gradient of the loss function with respect to all parameters of the learned network. This is problematic for shape parameterization as a customizable unit in a network, since at some level the output is a permutation. To overcome this issue, we use the metric alignment method of Solomon et al. [SPKS16] because its use of entropic regularization makes the objective differentiable in the input metric spaces. This differentiability is also leveraged by Peyré et al. [PCS16] to compute barycenters of sets of metric spaces.

## 3. Metric Alignment Layer

### 3.1. Problem Setup

The role of the metric alignment layer is to map scalar geometric features given on an input shape $\Sigma$ to features on a canonical domain $\Sigma_0$. In our design of the layer we have the following goals:

**Applicability.** The layer should be applicable to *multiple shape representations*, e.g. point clouds and triangle meshes, and its output should be usable with standard network architectures.

**Consistency.** Given geometric features that are invariant to isometries, the output of the layer should be *invariant to isometries*.

**Learn-ability.** The layer should be specified by a set of *parameters*, which can be learned and tuned within a deep learning network. Consequently, the layer should be *differentiable* with respect to these parameters.

With these goals in mind, we make a few design choices. First, since many network architectures are available for images, we define the canonical domain $\Sigma_0$ to be $n_0$ points laid out on the 2D grid, and encode the $k$ output features as a multi-channel image $f_0 : \Sigma_0 \to \mathbb{R}^k$ on this grid. Second, to allow for diverse input representations, we use the Gromov–Wasserstein (GW) generalized mapping algorithm [SPKS16]. The GW algorithm represents a map

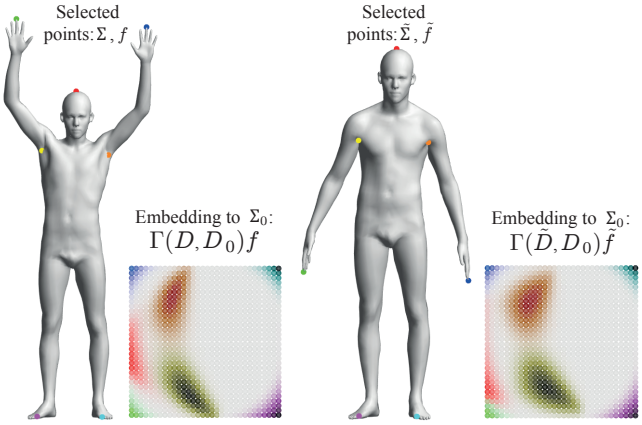


**Figure 2:** *Visualization of the GW fuzzy map for two nearly isometric shapes from FAUST [BRLB14]. (left) two shapes and corresponding color coded points, (right) distributions on the grid according to the GW map, high intensity indicates high probability for a match.*

as a "soft correspondence," namely given two geometries $\Sigma, \Sigma_0$ with $n$ and $n_0$ points respectively, the algorithm constructs a matrix $\Gamma \in \mathbb{R}^{n_0 \times n}$ such that a pair of points $(p, p_0) \in \Sigma \times \Sigma_0$ is assigned a high probability if they should be matched. We thus define the output of the layer to be $f_0 = \Gamma f$, where $f : \Sigma \to \mathbb{R}^k$ are the features defined on the input geometry. This mapping technique can be applied to point clouds, polygon soups, or any other geometric domain equipped with a metric.

Another advantage of using the GW algorithm is that it encourages consistency as it tries to find an embedding that preserves the metric of the source shape. Specifically, given the pairwise distance matrix $D \in \mathbb{R}^{n \times n}$ between points on the source domain $\Sigma$, and the pairwise distance matrix $D_0 \in \mathbb{R}^{n_0 \times n_0}$ on the target domain $\Sigma_0$, the matrix $\Gamma$ is constructed such that if a high probability is given to $(x, x_0)$ and $(y, y_0)$, both in $\Sigma \times \Sigma_0$, then the distances $D(x, y)$ and $D_0(x_0, y_0)$ are similar. Figure 2 shows a GW map between a human in different poses and a 2D grid. We visualize the map by selecting a few feature points on the human, assigning them consistent colors and mapping the colors to the grid with the soft correspondence produced by GW optimization. Note the similarity between the resulting representations on the grid. We use a slightly distorted target grid to avoid symmetric ambiguities in the target metric (see Section 4).

Finally, the GW algorithm is differentiable with respect to the geometry of the target domain, represented by the metric $D_0$. Hence, we can set up optimization problems over the distance matrix $D_0$ to modify the resulting map, which we demonstrate using the following toy experiment visualized in Figure 3. Given two shapes $\Sigma, \tilde{\Sigma}$ of a one-headed and a two-headed bunny, we compute a Wave Kernel Signature [ASC11] descriptor to define the source functions $f, \tilde{f}$ (left and right) respectively. We embed them to the 2D grid $\Sigma_0$ with the GW mapping which yields functions over the grid $f_0, \tilde{f}_0$ (center-top). As the surfaces are not isometric these mappings are not consistent and the resulting images are dissimilar. This can be

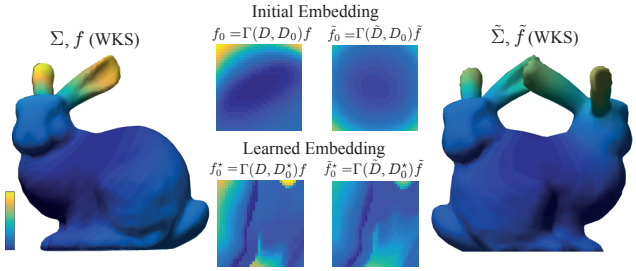


**Figure 3:** *Toy example, learning a target metric to optimize consistency. Given input descriptors $f, \tilde{f}$ on two shapes we show embeddings to canonical domain (center) before optimization (top) and after optimization (bottom).*

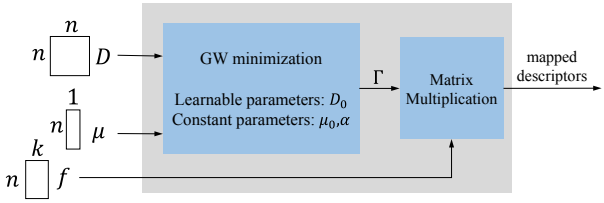


**Figure 5:** *Our metric alignment layer. The inputs are a pairwise distance matrix D, a measure μ, and k per-point descriptors. The layer maps the descriptors to a common domain by learning its metric $D_0$, and generates a mapping matrix Γ. Finally, the input descriptors are multiplied by Γ to generate a stack of 2D images to feed to the following layers.*

amended by optimizing over $D_0$ to minimize the image dissimilarity:

$$D_0^\star = \arg\min_{D_0} \|\Gamma(D, D_0)f - \Gamma(\tilde{D}, D_0)\tilde{f}\|^2.$$

We use gradient descent to find a local optimum $D_0^\star$, and the resulting mapped images $f_0^\star$ and $\tilde{f}_0^\star$ are more consistent (center-bottom). Figure 4 further demonstrates the improvement in consistency when using the optimized domain. We map a function that highlights the heads of the two-headed bunny (left) to the one-headed bunny via the original and the optimized domains. When using the original domain the two heads are mapped to the tail (center) whereas with the optimized domain the heads are mapped to the head (right).

We use a similar idea within a convolutional neural network architecture, by constructing the metric alignment layer depicted in Figure 5. The layer receives as input the distance matrices of the source shapes $D$, as well as area weights $\mu$. It then learns the target metric $D_0$ during training so that the mapped descriptors minimize the loss function when plugged into the next layers. Therefore, we effectively tune the regular embedding to produce mapped descriptors that best suit our task.

### 3.2. Implementation

**GW minimization.** The computation of the mapping matrix Γ between the input domain Σ and the regular domain $\Sigma_0$ is the main building block of our metric alignment layer. We use the method proposed by Solomon et al. [SPKS16, Algorithm 1] that requires two distance metrics $D$ and $D_0$ for two domains and nonnegative area measures $\mu, \mu_0$. The output is a matrix Γ that locally minimizes the regularized Gromov–Wasserstein distance measure:

$$GW_2^2(\mu_0, \mu, D_0, D) := \min_{\Gamma \in \bar{\mathcal{M}}(\mu_0, \mu)} \left[ \sum_{ijkl} \left(D_{0ij} - D_{kl}\right)^2 \Gamma_{ik}\Gamma_{jl}\mu_{0i}\mu_{0j}\mu_k\mu_l - \alpha H(\Gamma) \right] \quad (1)$$

where $H(\Gamma) = -\sum_{ik} \Gamma_{ik} \ln(\Gamma_{ik}) \mu_{0i}\mu_k$ is the *entropy* of Γ, and $\bar{\mathcal{M}}(\mu_0, \mu) := \left\{ \Gamma \in \mathbb{R}_+^{n_0 \times n} : \Gamma\mu = 1, \Gamma^T\mu_0 = 1 \right\}$ is the set of possible fuzzy maps, also known as *measure couplings*. Intuitively, $\Gamma_{ij}$ represents the probability that the $i^{th}$ point of $\Sigma_0$ corresponds to the $j^{th}$ point of Σ. The parameter α controls the entropy, where larger values create "fuzzier" maps. We set $\alpha = 0.005$ throughout all experiments and normalize each distance matrix by its maximal value.

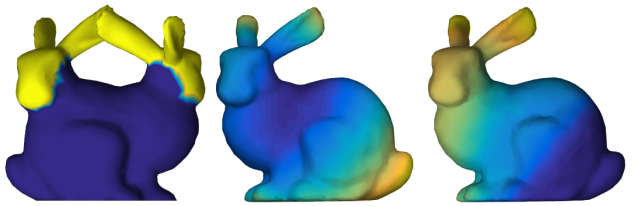


**Figure 4:** *Mapping an indicator of the heads (left) to the single-headed bunny before (center) and after (right) the optimization of the target domain. See the text for details.*

**Derivatives.** Typically, neural network parameters are optimized using stochastic gradient descent, in which the chain rule is used to differentiate the loss function with respect to the parameters in a process called backpropagation. Standard implementations of this procedure provide us with the gradient of the loss $L$ with respect to the map Γ, denoted by $\nabla_\Gamma L$; from this matrix, our goal is to compute the gradient $\nabla_{D_0} L$ of the loss $L$ with respect to the metric $D_0$, which determines Γ through Gromov–Wasserstein optimization.

Recall that [SPKS16] alternates between a closed-form exponential formula and Sinkhorn projection onto the cone of doubly stochastic matrices to compute Γ. For every iteration $i = 1, \ldots, I$ of their algorithm, we denote the map by $\Gamma^i$ and the input to Sinkhorn projection as $K^i$. Given $\nabla_{\Gamma^i} L$ from backpropagation through the later stages of our network, we obtain $\nabla_{D_0} L$ by iteratively computing $\nabla_{\Gamma^i} L$, $\nabla_{K^i} L$ in reverse order from $i = I$ to $i = 1$.

Given $\nabla_{\Gamma^i} L$, $\nabla_{K^i} L$ is computed using partial derivatives of Sinkhorn projection of $K^i$ onto the doubly stochastic cone. While [BPC16] differentiates individual iterations of the Sinkhorn algorithm for this task, for efficiency and storage reasons we choose to compute the derivative of the converged term directly using an

implicit linear system derived from the following stationarity conditions for $\Gamma^i$:

$$\begin{array}{ll} \Gamma^i = [\![v]\!] K^i [\![w]\!] & \Gamma^{i\top}\mu_0 = [\![w]\!] K^{i\top}(v \otimes \mu_0) = \mathbf{1} \\ \Gamma^i \mu = [\![v]\!] K^i (w \otimes \mu) = \mathbf{1} & \mathbf{1}^\top v = \mathbf{1}^\top w, \end{array} \tag{2}$$

where $[\![v]\!] \in \mathbb{R}^{n\times n}$ is a diagonal matrix with $v$ on the diagonal. Here, $v$ and $w$ are vectors of length $n_0$, $n$, respectively, computed during Sinkhorn projection. The first equation is the explicit computation of $\Gamma^i$, while the second and third equations define a valid fuzzy map; the last equation ensures that we have unique solution for $v, w$. All are satisfied when Sinkhorn projection converges. Differentiating these four equations with respect to each entry of $K^i$ yields a linear system whose solution gives the derivatives of $\Gamma^i$ with respect to the entries of $K^i$.

Using the chain rule and $\nabla_{\Gamma^i} L$ we iteratively compute $\nabla_{K^i} L$ and $\nabla_{\Gamma^{i-1}} L$; chaining these computations together leads to $\nabla_{D_0} L$. Full derivations and formulas are provided in the supplemental material.

**Mapped descriptors.** Once $\Gamma$ is computed, we map the $k$ pointwise input features $f \in \mathbb{R}^{n\times k}$ to the regular domain $\Sigma_0$ obtaining an image $f_0 = \Gamma [\![\mu]\!] f$. We then reshape the $k$ mapped images $f_0$ into $k$ matrices of size $\sqrt{n_0} \times \sqrt{n_0}$ (with $\sqrt{n_0} = 32$ for all experiments), to represent a multi-channel image over the 2D grid. We then feed the resulting $k$ matrices of fixed size to a neural network that may contain convolutional layers, which is expected to learn appropriate kernels for each feature independently.

**Parameters.** For a given shape $\Sigma$ we compute $n$ evenly distributed point samples, typically around 1000. To evenly sample $n$ points on a triangle mesh, we first randomly sample $10n$ triangles and barycentric coordinates for each sampled triangle, where each triangle is chosen with probability proportional to its area. Then we cluster the sampled points to $n$ clusters, and select the point closest to the centroid of each cluster. The pairwise distances $D$ are computed using Dijkstra's algorithm on a graph constructed by connecting every sampled point to its 5 nearest neighbors and iteratively connecting closest disconnected components. The measures $\mu$ are taken to be one third of the area of neighboring faces for manifold meshes, and unit for point clouds.

For intrinsic per-point features $f$ of deformable manifold meshes we use Gaussian curvature [BKP*10], conformal factor [BCG08] and the first entry of the Wave Kernel Signature [ASC11]. For extrinsic per-point features of polygon soups we use PCA-based features in the local point neighborhoods [KHS10], height, shape diameter function [SSCO08], and absolute curvature using the publicly available implementation [KCGF14]. The descriptors are normalized to have zero mean and unit variance on the training set.

Computing the metric alignment matrix and its gradients requires solving a large system of linear equations, and can become a major bottleneck in training. To remedy this, we run our metric alignment algorithm for a fixed number of iterations $I = 5$, and use $\Gamma$ from the previous epoch as the initial solution.

We implement this layer using the Torch library [CKF11] and execute the metric alignment algorithm and its differentiation on GPU using NVIDIA's cuDNN [CWV*14] and MAGMA [TDB10].

## 4. Shape Classification

Given the multi-channel 2D grid as the output of metric alignment layer we can take advantage of standard CNN layers designed for image analysis. In particular, for classification task we use a stack of convolutional layers, batch normalization, ReLU and dropout layers [SHK*14], depicted in Figure 6.

While the differentiable metric alignment layer enables us to train this network end-to-end, we found that pairwise distances of the target domain are difficult to optimize directly. Thus, we created a different mini-network dedicated to learning the common domain. Our mini-network only contains the metric alignment layer, and follows the *Siamese architecture* [BBB*93]. We construct our loss function to favor images of shapes within the same category to be as-similar-as-possible, and images of shapes from different categories to be separated by a margin. The input is a pair of shapes $\Sigma_1, \Sigma_2$ with their category labels $y_1, y_2$ represented using pairwise distance matrices $D_1, D_2$, measures $\mu_1, \mu_2$ and a set of $k$ pointwise descriptors $f, g$. The features of each shape are mapped to the common domain $\Sigma_0$ using the two copies of the metric alignment layer with shared parameters. The embedding creates images $f_0, g_0$ and we use L2 hinge loss function:

$$L(f_0, g_0) = \begin{cases} \|f_0 - g_0\|_2 & y_1 = y_2 \\ \max(0, m - \|f_0 - g_0\|_2) & y_1 \neq y_2 \end{cases} \tag{3}$$

where $m$ is the *margin*. This loss function penalizes *different* embeddings of shapes of the same category, as well as *similar* embeddings of shapes of different categories.

We initialize the pairwise distances of the 2D grid $D_0$ to the Euclidean distances between the points of a distorted 2D grid $\Sigma_0$. We distort the metric in the 2D plane to avoid perfect symmetries that lead to unnecessary ambiguity in the mapping. Since our metric alignment optimization is fairly robust to random perturbations, we introduce a consistent bias in the metric. In particular, we stretch the rows and columns of the grid using the following formula for each point of the grid with initial coordinates $(x, y)$:

$$x_{\text{distorted}} = x\left(1 + (\alpha_x - 1)\, y\right).$$

We apply a similar transformation for $y$ coordinate, and set $\alpha_x =$

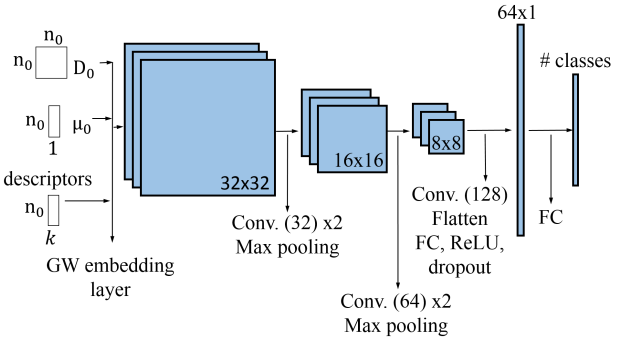


**Figure 6:** *Our convolutional neural network for classification and shape retrieval.*

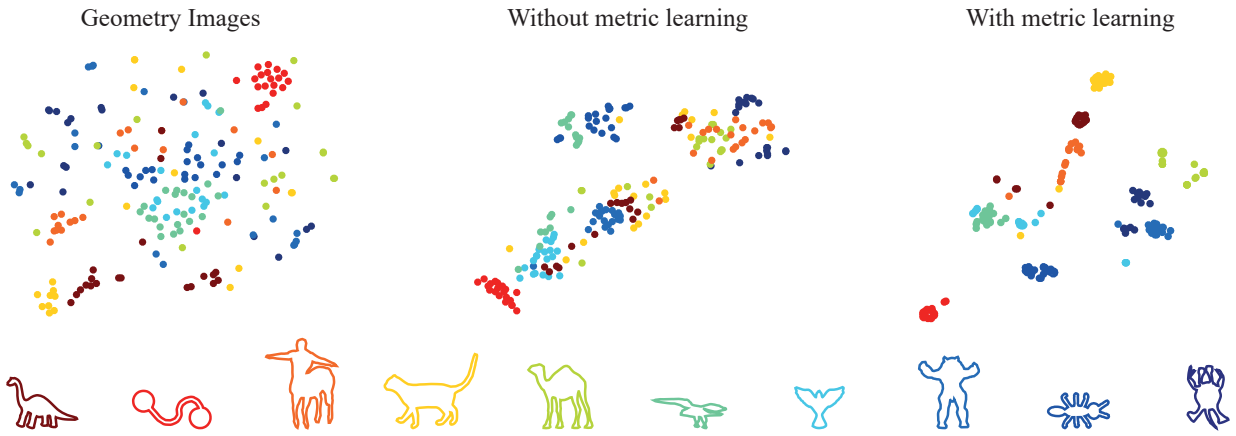


**Figure 8:** *tSNE embedding [MH08] of mapped descriptors corresponding to different mapping methods: (left) Geometry images [SBR16], (center) GW mapping to a fixed 2D grid, (right) GW mapping with a learned metric. We show the first 10 classes of SHREC'11 [LGB*11], where each point represents a shape, and shapes from the same class are in the same color. Note that after metric learning our mapped descriptors are nicely clustered, aiding the classification and retrieval tasks we optimized the embedding for.*

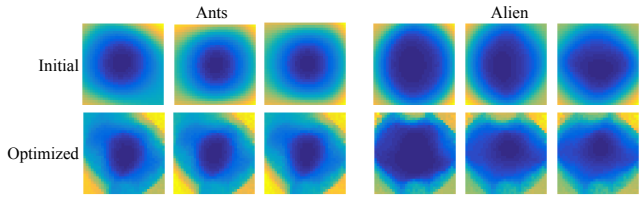


**Figure 7:** *Embedding of a single descriptor before (top) and after (bottom) metric learning, for two categories of SHREC'11, 3 meshes from each category.*

$1.1, \alpha_y = 1.2$ for all the experiments. This formula was chosen arbitrarily to generate a common domain that is similar to a grid and is not symmetric. At each epoch we compute a random permutation of shapes, and take pairs of subsequent examples for training. We use the Siamese network with the smallest training error.

Figure 7 illustrates the embedded features before (top) and after (bottom) the metric learning step. Note how the images that correspond to the meshes from the same category are similar to one another after metric learning, and dissimilar across categories. In Figure 8 we visualize the proximity between these feature images via a tSNE embedding [MH08]. Each dot corresponds to a model from the SHREC'11 dataset [LGB*11] and the color corresponds to the ground truth class label. We use three methods to map the shape features to an image: the method used by Sinha et al. [SBR16] (Geometry Images) for deep learning (left), GW mapping directly to a 2D grid (middle), and GW mapping based on the learned metric (right). Note that even without learning, the GW mapping to a 2D grid provides more consistency in the embedding, which is then enhanced after metric learning.

After we train the metric alignment layer, we keep the learned parameters $D_0$ fixed, and train the other parameters of the classification network depicted in Figure 6. We tried fine-tuning our $D_0$ parameters in the whole network and train it end-to-end, but it significantly slowed down the training step and did not yield any significant improvement in accuracy.

We next evaluate the performance of our network on classification and retrieval tasks on commonly used benchmarks.

## 5. Results

We test our method for classification and retrieval tasks on several existing benchmarks, and also demonstrate the effect of different design choices for the features and the metric.

**Classification and retrieval of deformable shapes.** We use the SHREC'11 benchmark [LGB*11] to test our method for classification of articulated shapes. The dataset contains 600 shapes containing both rigid (e.g., furniture) and non-rigid (e.g., humans, animals) shapes with significant articulations. We compute the intrinsic shape descriptors: Gaussian curvature (GC), Conformal Factor (CF) and Wave Kernel Signature (WKS). We start by learning the metric using the Siamese architecture described in Section 4 for 100 epochs. We set the margin to 50 (approximately the average distance between pairs from different classes). Then we train the classification network depicted in Figure 6. Due to the small dataset size we did not use a validation set and stopped training after a fixed number of epochs (200), similarly to Sinha et al. [SBR16]. We used $l_1$ and $l_2$ regularization with weights $10^{-4}$, $10^{-5}$ respectively, as well as weight decay $10^{-5}$.

We evaluate our classification network by measuring the percentage of correct classifications of shapes in the test set. We also use the output of a penultimate fully connected layer as a global shape descriptor, and evaluate the quality of retrieval using Mean Average Precision (mAP). We use the metrics and the protocol established in Sinha et al. [SBR16] and similarly run on two data splits: 10 training samples from each category (and 10 test shapes), and 16 training samples from each category (4 test shapes).

| | Classification | | | | Retrieval | | | |
|---|---|---|---|---|---|---|---|---|
| Data \ method | SG | SN | GI | Ours | SG | SN | GI | Ours |
| SHREC, 10 | 62.6 | 52.7 | 88.6 | **90.3** | 0.65 | 0.1 | 0.65 | **0.87** |
| SHREC, 16 | 70.8 | 48.4 | **96.6** | **96.6** | 0.74 | 0.13 | 0.72 | **0.96** |

**Table 1:** *Performance on the SHREC'11 classification benchmark, percentage of correct classification / mAP. We compare with Shape Google [BBGO11] (SG), 3D ShapeNets [WSK*15] (SN) and Geometry Images [SBR16] (GI).*

We compare our results with three other methods: Shape Google [BBGO11] (SG), which uses a bag of features representation, 3D ShapeNets [WSK*15] (SN), which uses a volumetric representation, and Geometry Images [SBR16] (GI), which uses a deep network trained over a flattened shape. Note that GI use substantially larger images in their representation ($32 \times 32$ pixels). We present the accuracy and retrieval results in Table 1. Our method produces more accurate classifications than state-of-the-art tools using only 10 training examples, and performs comparably to geometry images with 16 training examples. Our retrieval mAP score is consistently higher for all experiments.

**Visualization of learned features.** After mapping the shapes to the common domain each shape is represented by a multi channel image. The output of the layers of the classification network is therefore challenging to interpret directly. To visualize properties of the classification network we map the output of each convolution layer back to the shape using the GW map, generating a shape descriptor with 320 values for each vertex. Interestingly, the features of the first layer highlight relatively low resolution properties, while the features of the last layer capture finer details, as has been shown for natural images classification. This is visualized in Figure 9 using distances in *feature space*. We pick a point $p$ on a bird model from SHREC, and plot the $l_2$ distance between the features of the first convolution layer of $p$ and all the other points on the shape (top), and similarly for the last convolution layer (bottom) . We also show the distance between the features of the same point $p$ and all the points on two other shapes, one is another shape from the same class and the second is from a different class of a different kind of bird. The features of the first convolution layer are symmetric and similar for points with similar functionality, even for birds from different classes. However, the features of the last convolution layer can distinguish between the two wings of the same bird, and are different for birds of different classes.

**Effect of learning the metric of the common domain.** We next investigate how important is it to learn the metric of the common domain in comparison to simpler alternatives. We run experiments with 10 training examples on the SHREC'11 dataset. First, we use the initial metric of the grid, computed using distorted Euclidean distances between grid cells (Section 4). Another option is to use the undistorted metric of the grid, which will yield some symmetric ambiguities. To resolve these ambiguities, we can augment the data at training time and feed all the elements of the symmetry group to the classification CNN (i.e., rotations and reflections of the 2D square). Both options are presented in Table 2 and yield inferior results to our method.

**Effect of input features.** In Table 3 we compare our method when using Euclidean distances for computing the distance matrices for the input shapes, and with different geometric features. Note that since SHREC'11 has many models with severe articulations the use of an extrinsic (Euclidean) distance metric significantly decreases the quality of the results. Our retrieval results are higher than the other methods (Table 1) even with very simple features.

| Metric | Classification | Retrieval |
|---|---|---|
| Ours (metric learning) | **90.3** | **0.87** |
| Distorted grid (no learning) | 88.66 | 0.85 |
| Augmented grid (no learning) | 86.66 | 0.87 |

**Table 2:** *Results of our method with and without metric learning on SHREC'11 with 10 training samples per class.*

| Features | Classification | Retrieval |
|---|---|---|
| GC,CF,WKS, Geodesic | **90.3** | **0.87** |
| GC,CF,WKS, Euclidean | 84.6 | 0.81 |
| GC, CF, Geodesic | 89.3 | 0.87 |
| GC, Geodesic | 89 | 0.84 |

**Table 3:** *Comparison of our method with different geometric features on SHREC'11 with 10 training samples per class. The features are Gaussian curvature (GC), Conformal factor (CF), Wave Kernel Signature (WKS), Geodesic distances (Geodesic), Euclidean distances (Euclidean).*

**Classification and retrieval of rigid shapes.** We also evaluate our method on the ModelNet40 and ModelNet10 benchmarks [WSK*15] that contain mostly rigid shapes. For these datasets we use as features the distance histogram, PCA in the local neighborhood of a point and the height, computed using the publicly available code [KCGF14]. Due to the size of this dataset we do the initial metric optimization with the Siamese network only for 10 epochs. We set the margin to 120 (approximately the average distance between pairs from different classes). We do 5-fold cross validation during training and use $l_2$ regularization with the weight $10^{-3}$, as well as weight decay $10^{-4}$. The results are presented in Table 4. Our method provides competitive retrieval results with respect to GI, but classification suffers from lack of resolution (we use $32 \times 32$ images to represent a shape, whereas GI uses images of size $64 \times 64$). Our method under-performs on ModelNet40 when compared to multi-view representation methods, e.g. MVCNN [SMKLM15] (classification and retrieval rates of 90.1 and 0.79, respectively), or [QSN*16] (classification rate 91.4). Therefore, it appears that some of our design choices lead to non-optimal performance on rigid shapes, however as our main focus is non-rigid shapes, we leave further investigation of this direction for future work.

| | Classification | | | Retrieval | | |
|---|---|---|---|---|---|---|
| Data \ method | SN | GI | Ours | SN | GI | Ours |
| ModelNet10 | 83.5 | 88.4 | 85.8 | 0.68 | 0.75 | 0.74 |
| ModelNet40 | 77 | 83.9 | 74.6 | 0.49 | 0.51 | 0.59 |

**Table 4:** *Performance on ModelNet. We compare with 3D ShapeNets [WSK*15] (SN) and Geometry Images [SBR16] (GI).*

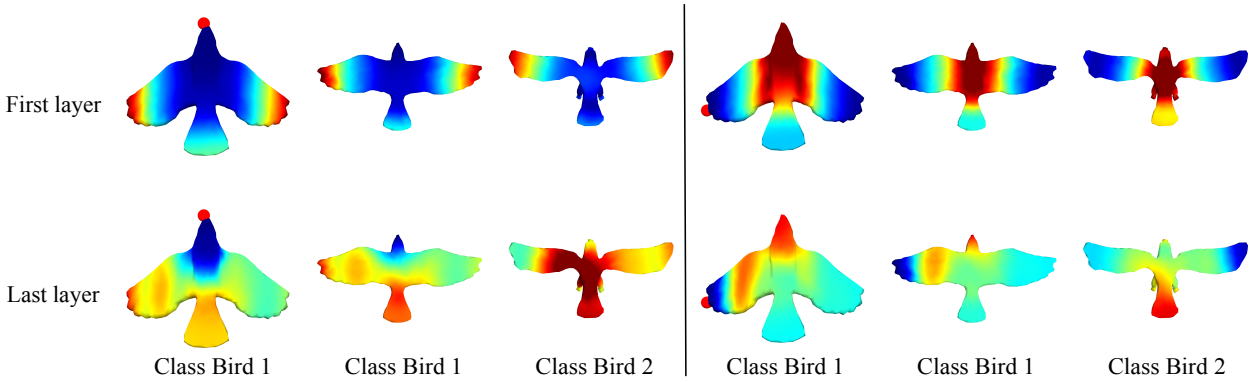


**Figure 9:** *Features learned at the first convolution layer (top row) vs. last convolution layer (bottom row) of the classification network for SHREC'11. We visualize the distance between a selected point (tip of the nose or tip of the wing) and other points on the same or different shapes. Note that features from the last convolution layer provide more details and differentiate between birds from different classes.*

**Timing.** The new metric alignment layer takes about 1s to compute an *initial* matrix $\Gamma$ (50 internal GW iterations) for shapes with approximately 1000 points. After computing the initial maps we only use 5 internal GW iterations and computing $\Gamma$ takes about 0.2s. Computing the gradient with respect to $D_0$ takes about 1s. Our approach took 12h to train the Siamese network for SHREC'11 classification (100 epochs), and 6m to train the classification network. Training the Siamese network for ModelNet40 took 25h and 20m to train the classification network.

**Limitations.** While learning the optimal embedding is beneficial, the number of parameters grows quadratically with the number of discrete elements in the target domain; this affects training complexity. While the GW computation is robust to uniformly distributed noise in the input pairwise distances, topological noise might drastically change pairwise geodesic distances and undermine the quality of near-isometric mappings. Finally, the user's choice of point-wise shape descriptors affects the applicability of the method, and thus requires some prior knowledge about the analyzed dataset. We found our method and design choices to be more suitable for nonrigid shape analysis, as is evident from poor performance of our method on the ModelNet40 dataset.

## 6. Conclusion and Future Work

The main contribution of this work is a novel metric alignment layer that is based on a differentiable and parametric algorithm that embeds non-structured input to a structured domain suitable for deep learning architectures. Our layer works with a range of geometric representations such as point clouds and polygon soups and can capture intrinsic as well as extrinsic geometric structure as long as it is possible to compute a metric over the domain. Unlike any of the existing techniques our embedding layer can be trained specifically for a particular dataset and loss function.

**Future work.** We believe that there are many directions to explore using metric alignment. Since convolutions need to be redefined as the geometry of the regular domain changes, modeling the regular domain as a general graph and applying a convolution operator on that graph [BZSL13] should further improve our results. Incorporating automatic feature learning can also potentially improve results and reduce user intervention. While end-to-end learning did not provide a significant performance boost and was omitted from the current training, the above changes coupled with computational speedups may enable learning metric alignment jointly with the subsequent network layers. Training deep networks for other geometry analysis tasks, such as keypoint detection and segmentation, is also an interesting future direction. More generally, metric alignment may help in the analysis of images with deformable structures, as long as a meaningful non-Euclidean metric over the image grid exits (e.g., [BS09]).

**Acknowledgments.** This project started as an internship project at Adobe Research. We thank Fisher Yu for his assistance with the classification network design. J. Solomon acknowledges funding from an MIT Skoltech Seed Fund grant ("Boundary Element Methods for Shape Analysis") and from the MIT Research Support Committee ("Structured Optimization for Geometric Problems"). M. Ben-Chen acknowledges funding from ISF grant 699/12 and a gift from Adobe.

## References

[AL16] AIGERMAN N., LIPMAN Y.: Hyperbolic Orbifold Tutte Embeddings. *ACM SIGGRAPH Asia* (2016). 3

[ASC11] AUBRY M., SCHLICKEWEI U., CREMERS D.: The wave kernel signature: A quantum mechanical approach to shape analysis. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on* (2011), IEEE, pp. 1626–1633. 3, 5

[BBB*93] BROMLEY J., BENTZ J. W., BOTTOU L., GUYON I., LECUN Y., MOORE C., SÄCKINGER E., SHAH R.: Signature verification using a "siamese" time delay neural network. *IJPRAI 7*, 4 (1993), 669–688. 5

[BBGO11] BRONSTEIN A. M., BRONSTEIN M. M., GUIBAS L. J., OVSJANIKOV M.: Shape Google: Geometric words and expressions for invariant shape retrieval. *TOG 30*, 1 (Feb. 2011). 2, 7

[BCG08] BEN-CHEN M., GOTSMAN C.: Characterizing shape using conformal factors. In *3DOR* (2008), pp. 1–8. 5

[BKP*10] BOTSCH M., KOBBELT L., PAULY M., ALLIEZ P., LEVY B.: *Polygon Mesh Processing*. AK Peters Series. Taylor & Francis, 2010. 5

[BLRW16] BROCK A., LIM T., RITCHIE J., WESTON N.: Generative

and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236* (2016). 2

[BMRB16] BOSCAINI D., MASCI J., RODOLÀ E., BRONSTEIN M. M.: Learning shape correspondence with anisotropic convolutional neural networks. In *NIPS* (2016). 1, 2

[BPC16] BONNEEL N., PEYRÉ G., CUTURI M.: Wasserstein barycentric coordinates: Histogram regression using optimal transport. *TOG 35*, 4 (2016). 4

[BRLB14] BOGO F., ROMERO J., LOPER M., BLACK M. J.: FAUST: Dataset and evaluation for 3D mesh registration. In *Proc. CVPR, IEEE* (Piscataway, NJ, USA, June 2014), IEEE. 3

[BS09] BAI X., SAPIRO G.: Geodesic matting: A framework for fast interactive image and video segmentation and matting. *IJCV 82*, 2 (2009), 113–132. 8

[BZSL13] BRUNA J., ZAREMBA W., SZLAM A., LECUN Y.: Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203* (2013). 2, 8

[CKF11] COLLOBERT R., KAVUKCUOGLU K., FARABET C.: Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop* (2011), no. EPFL-CONF-192376. 2, 5

[COC14] CORMAN E., OVSJANIKOV M., CHAMBOLLE A.: Supervised descriptor learning for non-rigid shape matching. *Proc. ECCV Workshops, NORDIA* (2014). 2

[CWV*14] CHETLUR S., WOOLLEY C., VANDERMERSCH P., COHEN J., TRAN J., CATANZARO B., SHELHAMER E.: cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759* (2014). 5

[DBV16] DEFFERRARD M., BRESSON X., VANDERGHEYNST P.: Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS* (2016), pp. 3837–3845. 2

[EM94] EDELSBRUNNER H., MÃIJCKE E. P.: Three-dimensional alpha shapes. *TOG* (1994). 2

[GGH02] GU X., GORTLER S., HOPPE. H.: Geometry images. In *SIGGRAPH* (2002). 2

[GZC15] GUO K., ZOU D., CHEN X.: 3d mesh labeling via deep convolutional neural networks. *TOG 35*, 1 (2015). 2

[HBL15] HENAFF M., BRUNA J., LECUN Y.: Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* (2015). 2

[HLS08] HORMANN K., LÉVY B., SHEFFER A.: Mesh Parameterization: Theory and Practice. *SIGGRAPH Asia, Course Notes* (2008). 3

[KAMC17] KALOGERAKIS E., AVERKIOU M., MAJI S., CHAUDHURI S.: 3d shape segmentation with projective convolutional networks. *Proc. CVPR, IEEE* (2017). 2

[KCGF14] KIM V. G., CHAUDHURI S., GUIBAS L., FUNKHOUSER T.: Shape2Pose: Human-Centric Shape Analysis. *TOG 33*, 4 (2014). 5, 7

[KHS10] KALOGERAKIS E., HERTZMANN A., SINGH K.: Learning 3D Mesh Segmentation and Labeling. *TOG 29*, 3 (2010). 2, 5

[KW16] KIPF T. N., WELLING M.: Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016). 2

[LGB*11] LIAN Z., GODIL A., BUSTOS B., DAOUDI M., HERMANS J., KAWAMURA S., KURITA Y., LAVOUÉ G., NGUYEN H., OHBUCHI R., ET AL.: SHREC'11 track: shape retrieval on non-rigid 3d watertight meshes. In *3DOR* (2011), pp. 79–88. 6

[LPRM02] LEVY B., PETITJEAN S., RAY N., MAILLOT J.: Least squares conformal maps. *SIGGRAPH* (2002). 3

[MBBV15] MASCI J., BOSCAINI D., BRONSTEIN M., VANDERGHEYNST P.: Geodesic convolutional neural networks on Riemannian manifolds. In *Proc. ICCV Workshops* (2015), pp. 37–45. 2

[MH08] MAATEN L. V. D., HINTON G.: Visualizing data using t-SNE. *Journal of Machine Learning Research 9*, Nov (2008), 2579–2605. 6

[MS15] MATURANA D., SCHERER S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS* (2015), IEEE, pp. 922–928. 2

[OBCS*12] OVSJANIKOV M., BEN-CHEN M., SOLOMON J., BUTSCHER A., GUIBAS L.: Functional maps: a flexible representation of maps between shapes. *TOG 31*, 4 (2012), 30. 2, 3

[PCS16] PEYRÉ G., CUTURI M., SOLOMON J.: Gromov–Wasserstein averaging of kernel and distance matrices. In *ICML* (2016). 3

[PSS01] PRAUN E., SWELDENS W., SCHRODER P.: Consistent Mesh Parameterizations. *SIGGRAPH* (2001). 3

[QSMG17] QI C. R., SU H., MO K., GUIBAS L. J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. CVPR, IEEE* (2017). 2

[QSN*16] QI C. R., SU H., NIESSNER M., DAI A., YAN M., GUIBAS L.: Volumetric and multi-view cnns for object classification on 3d data. In *Proc. CVPR, IEEE* (2016). 1, 2, 7

[SBR16] SINHA A., BAI J., RAMANI K.: Deep learning 3d shape surfaces using geometry images. In *ECCV* (2016), Springer, pp. 223–240. 1, 2, 6, 7

[SBZB15] SHI B., BAI S., ZHOU Z., BAI X.: Deeppano: Deep panoramic representation for 3-d shape recognition. *IEEE Signal Processing Letters 22*, 12 (2015), 2339–2343. 2

[SHK*14] SRIVASTAVA N., HINTON G. E., KRIZHEVSKY A., SUTSKEVER I., SALAKHUTDINOV R.: Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research 15*, 1 (2014), 1929–1958. 5

[SK16] SAHILLIOĞLU Y., KAVAN L.: Detail-preserving mesh unfolding for non-rigid shape retrieval. *TOG 35*, 2 (2016). 3

[SMKLM15] SU H., MAJI S., KALOGERAKIS E., LEARNED-MILLER E.: Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV* (2015), pp. 945–953. 1, 2, 7

[SPKS16] SOLOMON J., PEYRÉ G., KIM V. G., SRA S.: Entropic metric alignment for correspondence problems. *TOG 35*, 4 (July 2016). 1, 2, 3, 4

[SSCO08] SHAPIRA L., SHAMIR A., COHEN-OR D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. *Vis. Comput. 24*, 4 (2008), 249–259. 5

[SX16] SONG S., XIAO J.: Deep Sliding Shapes for amodal 3D object detection in RGB-D images. In *Proc. CVPR, IEEE* (2016). 1, 2

[TDB10] TOMOV S., DONGARRA J., BABOULIN M.: Towards dense linear algebra for hybrid GPU accelerated manycore systems. *Parallel Computing 36*, 5-6 (June 2010), 232–240. 5

[WHC*16] WEI L., HUANG Q., CEYLAN D., VOUGA E., LI H.: Dense human body correspondences using convolutional networks. In *Proc. CVPR, IEEE* (June 2016). 2

[WSK*15] WU Z., SONG S., KHOSLA A., YU F., ZHANG L., TANG X., XIAO J.: 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1912–1920. 2, 7

[WZX*16] WU J., ZHANG C., XUE T., FREEMAN B., TENENBAUM J.: Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NIPS* (2016), pp. 82–90. 2

[XKH*16] XU K., KIM V. G., HUANG Q., MITRA N. J., KALOGERAKIS E.: Data-driven shape analysis and processing. *SIGGRAPH Asia Course notes* (2016). 2

[YSGG17] YI L., SU H., GUO X., GUIBAS L.: Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. *Proc. CVPR, IEEE* (2017). 2

[ZvKD10] ZHANG H., VAN KAICK O., DYER R.: Spectral mesh processing. *Computer Graphics Forum 29*, 6 (2010), 1865–1894. 3