MGCN: Descriptor Learning using Multiscale GCNs

YIQUN WANG, NLPR, Institute of Automation, CAS; School of AI, University of Chinese Academy of Sciences; KAUST
JING REN, KAUST
DONG-MING YAN*, NLPR, Institute of Automation, CAS; School of AI, University of Chinese Academy of Sciences
JIANWEI GUO, NLPR, Institute of Automation, CAS; School of AI, University of Chinese Academy of Sciences
XIAOPENG ZHANG, NLPR, Institute of Automation, CAS; School of AI, University of Chinese Academy of Sciences
PETER WONKA, KAUST

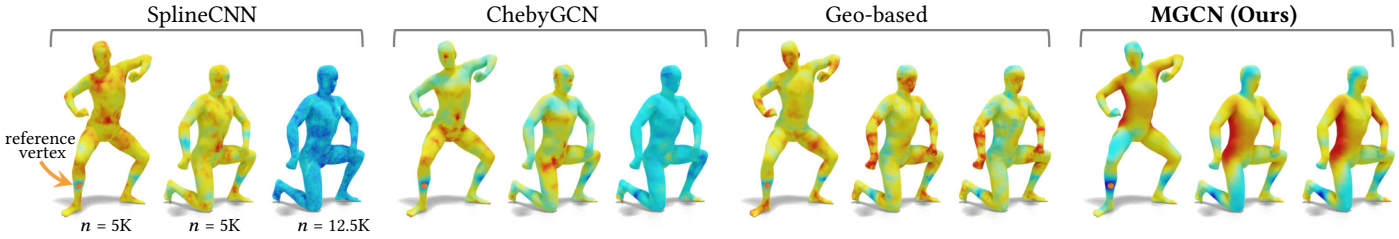

Fig. 1. Visualization of dissimilarity maps. For a vertex highlighted by the orange arrow, we take its learned desriptor and visualize the difference to other vertex descriptors on the same shape, another shape with the same 5K resolution, and the other shape in a different 12.5K resolution. We compare four different learned descriptors, from left to right: SplineCNN, ChebyGCN, Geodesic-based method [Wang et al. 2019a], and MGCN. All networks are trained on 5K and tested on 5K and 12.5K resolution. We can see that our network MGCN is most consistent between different resolutions.

We propose a novel framework for computing descriptors for characterizing points on three-dimensional surfaces. First, we present a new non-learned feature that uses graph wavelets to decompose the Dirichlet energy on a surface. We call this new feature *Wavelet Energy Decomposition Signature* (WEDS). Second, we propose a new *Multiscale Graph Convolutional Network* (MGCN) to transform a non-learned feature to a more discriminative descriptor. Our results show that the new descriptor WEDS is more discriminative than the current state-of-the-art non-learned descriptors and that the combination of WEDS and MGCN is better than the state-of-the-art learned descriptors. An important design criterion for our descriptor is the robustness to different surface discretizations including triangulations with varying numbers of vertices. Our results demonstrate that previous graph convolutional networks significantly overfit to a particular resolution or even a particular triangulation, but MGCN generalizes well to different surface discretizations. In addition, MGCN is compatible with previous descriptors and it can also be used to improve the performance of other descriptors, such as the heat kernel signature, the wave kernel signature, or the local point signature.

*Corresponding author.

Authors' addresses: Yiqun Wang, NLPR, Institute of Automation, CAS; School of AI, University of Chinese Academy of Sciences; KAUST, yiqun.wang@nlpr.ia.ac.cn; Jing Ren, KAUST, jing.ren@kaust.edu.sa; Dong-Ming Yan, NLPR, Institute of Automation, CAS; School of AI, University of Chinese Academy of Sciences, yandongming@gmail.com; Jianwei Guo, NLPR, Institute of Automation, CAS; School of AI, University of Chinese Academy of Sciences, jianwei.guo@nlpr.ia.ac.cn; Xiaopeng Zhang, NLPR, Institute of Automation, CAS; School of AI, University of Chinese Academy of Sciences, xiaopeng.zhang@ia.ac.cn; Peter Wonka, KAUST, pwonka@gmail.com.

0730-0301/2020/7-ART1 $15.00
https://doi.org/10.1145/3386569.3392443

CCS Concepts: • **Computing methodologies** → **Shape analysis**.

Additional Key Words and Phrases: Multiscale, Energy Decomposition, Wavelet Convolution, Shape Matching

ACM Reference Format:
Yiqun Wang, Jing Ren, Dong-Ming Yan, Jianwei Guo, Xiaopeng Zhang, and Peter Wonka. 2020. MGCN: Descriptor Learning using Multiscale GCNs. *ACM Trans. Graph.* 39, 4, Article 1 (July 2020), 15 pages. https://doi.org/10.1145/3386569.3392443

1 INTRODUCTION

Designing descriptors for surface points is a fundamental problem in geometry processing as descriptors are a building block for many applications, such as shape matching, registration, segmentation, and retrieval.

A good descriptor should satisfy two criteria: (1) The descriptor should be *discriminative* to map similar surface points to similar values and dissimilar surface points to dissimilar values. The definition of similarity depends on the application. In our setting, we consider the very popular requirement that descriptors should be invariant to rigid and near-isometric deformations of the surface. (2) The descriptor should be *robust* to different discretizations of the surface, *e.g.*, meshes of different resolution and triangulation. If the descriptor discriminates surface points based on the discretization, we also say it overfits or lacks generalization.

Generally, we can distiguish two types of descriptor computation: supervised and non-learned. Examples of non-learned descriptors are the *Wave Kernel Signature* (WKS) and the *Heat Kernal Signature* (HKS). While these descriptors are robust to different surface discretization, there is a lot of room for improvement making them more discriminative. This can been done successfully using neural networks to compute supervised descriptors. A very promising type of network architecture are graph convolutional networks, such as

chebyGCN [Defferrard et al. 2016], GCN [Kipf and Welling 2017], SplineCNN [Fey et al. 2018], and DGCNN [Wang et al. 2019b]. Even though many of these networks have not been applied to descriptor learning directly, their adaption to descriptor learning only requires little effort. However, the current state of the art is typically not robust to different surface discretizations and overfits. See Fig. 1 for an illustration. One main reason for this overfitting is that the convolution operation typically depends on a k-ring neighborhood of a surface point. This changes the spatial support of a convolutional filter if, for example, the resolution of the underlying surface triangulation changes. One possible approach to make descriptor learning robust to different resolutions is to resample the surface. This approach has other drawbacks, such as the additional complexity of resampling and the loss of information reducing the discrimination performance.

In this paper, we propose contributions to non-learned and supervised descriptor computation leveraging the power of wavelets. For the learning part, we introduce a novel graph convolutional network called *Multiscale Graph Convolutional Network* (MGCN). Our results will show significant improvements to descriptor performance even when tested under a variety of different surface discretizations. The key novelty of our network is a convolution operation expressed in the wavelet basis. This lets us define a multiscale convolution with filters of both local and global support.

For the non-learned descriptor computation part, we theoretically derive a novel local spectral feature called *Wavelet Energy Decomposition Signature* (WEDS) from the Dirichlet energy. Different from traditional spectral descriptors (*e.g.*, *Global Point Signature* (GPS), HKS, and WKS), we introduce additional vertex coordinate information to capture more distinctive attributes. Compared with *Local Point Signature* (LPS) [Wang et al. 2019a], our new descriptor uses wavelets to capture both local and global information, which is more discrimintive.

Our extensive experimental evaluations indicate that the WEDS descriptor outperforms recent state-of-the-art non-learned descriptors. Further, WEDS can be combined with MGCN to improve upon the currently best supervised descriptors. Besides the traditional evaluation of descriptor performance with respect to rigid, near-isometric, and non-isometric surface deformations, we also evaluate the robustness to different surface discretizations.

The main contributions of this work are as follows:

- We design a new graph convolutional network named *Multiscale Graph Convolutional Network*. While we focus on descriptor learning as main application, the robustness to resolution of our new convolution layer holds promise for many additional applications.
- We present a novel multiscale feature called *Wavelet Energy Decomposition Signature* based on energy decomposition that improves upon state-of-the-art non-learned descriptors.

2 RELATED WORK

We review related work for descriptor computation in three categories and subsequently related work for optimization-based matching methods and graph convolutional neural networks.

2.1 Descriptor Generation

Spatial domain approaches. Descriptors directly constructed in the spatial domain often rely on histograms. *Spin Images* (SI) [Johnson and Hebert 1999] and *3D Shape Context* (3DSC) [Frome et al. 2004] are generated by creating accumulators, which divide the local space into different bins and calculate the number of points that fall into each bin. *Signature of Histogram of Orientations* (SHOT) [Tombari et al. 2010] is constructed by accumulating the normal angles of the key and neighboring points in the neighborhood space. Unlike the SHOT descriptor, the *Mesh Histogram of Oriented Gradients* (MeshHOG) [Zaharescu et al. 2009] descriptor is another histogram based on the orientations of the gradients on the mesh. The *Rotational Projection Statistics* (RoPS) [Guo et al. 2013] descriptor is generated by rotationally projecting neighboring points onto 2D planes and calculating a set of statistics. Spatial domain descriptors generally have the following performance characteristics. First, they heavily rely on local information, but do not capture global information. Second, the descriptors are sensitive to the discretization of the surface. While this is desirable for some applications, an important goal of our work is to be robust to different surface discretizations.

Spectral domain approaches. Many spectral descriptors have been proposed to deal with isometric deformations. Especially popular are intrinsic descriptors based on the Laplace-Beltrami operator. Shape-DNA [Reuter et al. 2006] considers the spectrum of the Laplace-Beltrami operator as the descriptor because the spectrum is isometry-invariant and independent of spatial position. GPS [Rustamov 2007] combines the spectrum and eigenfunctions to obtain a descriptor on each vertex. HKS [Sun et al. 2010], scale-invariant HKS [Bronstein and Kokkinos 2010], and WKS [Aubry et al. 2011] were proposed based on diffusion geometry. The intrinsic properties make the descriptors invariant to isometric deformation. LPS [Wang et al. 2019a] combines coordinate information and intrinsic geometric information to get a more robust descriptor. The *Discrete Time Evolution Process* (DTEP) descriptor [Melzi et al. 2018] focuses on non-isometric deformations and achieved better results. But these methods take a lot of time to compute geodesic distances or solve optimization problems. In our results, we compare to the best performing descriptors to demonstrate an important improvement in performance. Our discussion in Section 4.5 will explain why we are able to beat the state of the arts in more detail.

Deep learning approaches. We call the descriptors reviewed in the previous two paragraphs non-learned. By contrast, supervised descriptors use supervised learning, mainly deep learning, to extract shape descriptors. Wei *et al.* [2016] generate descriptors by using a large dataset of depth maps for training. Huang *et al.* [2018] extract local descriptors by training on multiple rendered views in multiple scales. Zeng *et al.* [2017] use 3D volumetric convolutional neural networks to generate local descriptors for robustly matching RGB-D data. The method of *Compact Geometric Features* (CGF) [Khoury et al. 2017] maps high-dimensional histograms into a low-dimensional Euclidean space to generate descriptors on unstructured point clouds. Deng *et al.* [2018] found matches in unorganized point clouds by adapting the PointNet architecture. Although these methods have obtained good results through learning, they do not make full use of the structure of 3D data. Multi-view-based methods

must solve the problem of occlusion. Voxel-based methods consume a lot of resources and cannot explore the details of an object. Learning methods that work by feeding point cloud coordinates or histogram features into multi-layer perceptrons are currently not able to extract enough important information from the data.

Some other descriptors are generated by exploiting the correlation between local points in the frequency domain or spatial domain. *Optimal Spectral Descriptors* (OSD) [Litman and Bronstein 2014] are constructed by learning parametric filters in the spectral domain. Boscaini *et al.* [2015] generalize the windowed Fourier transform to learn local shape descriptors on manifolds. Anisotropic diffusion descriptors [Boscaini et al. 2016] based on anisotropic diffusion are constructed by using a fully connected neural network to learn the kernel filters. In the spatial domain, Masci *et al.* [2015] design a geodesic convolutional network to learn shape descriptors on manifolds by extracting and regularly charting geodesic local patches and designing a patch operator for convolution. Wang *et al.* [2018] and Guo *et al.* [2020] employ a deep learning framework by projecting geodesic local patches into local geometry images to learn descriptors, respectively. Subsequently, LPS [Wang et al. 2019a] is proposed on geodesic local patches and used with deep learning to construct a more discriminative descriptor. Although these spatial domain methods can convolve local regions on manifolds and get better results, they need to extract local geodesic patches, which is very time-consuming. In addition, it is very difficult to maintain a disk topology if the local region becomes larger and the surface contains topological holes.

2.2 Optimization-based Shape Matching Methods

Optimization-based matching approaches optimize a dense map between a pair of shapes. Some of these methods are initialized by shape descriptors. We focus on the related state-of-the-art methods and refer the reader to a recent survey [Biasotti et al. 2016] for more details on shape matching. For near-isometric shapes, one of the most successful approaches is based on functional maps [Ovsjanikov et al. 2012]. For each shape, a Laplace-Beltrami basis is computed to express the function space. The functional map between two shapes is encoded as a matrix. Deep functional maps [Litany et al. 2017] combine functional maps with deep learning. In the proposed network, the initial stage learns a shape descriptor that is trained with the goal of being useful in shape matching. Halimi *et al.* [2019] extend deep functional maps so that they can be trained in an unsupervised manner. For non-isometric shapes, *Blended Intrinsic Map* (BIM) [Kim et al. 2011] is the current state of the art where a set of maps are proposed and then blended together to output a single map.

2.3 Graph Convolutional Networks

Recently, a large number of graph convolutional learning methods have emerged. This learning method has achieved high performance on irregular data. Spectral CNN [Bruna et al. 2014] is the first to perform convolution operations on the graph through a frequency domain transform. ChebyGCN [Defferrard et al. 2016] simplifies spectral CNN by designing spectral filters using a k-order polynomial parametrization. GCN [Kipf and Welling 2017] further simplifies polynomials to 1-order and is suitable for semi-supervised learning. SplineCNN [Fey et al. 2018] uses B-Spline kernels to weight the relationship between a point and its neighborhood. Although this network has strong fitting ability, the generalization is not strong as the pseudo-coordinates on the edges are not invariant to rigid transformations. *Feature-Steered Network* (FeaStNet) [Verma et al. 2018] proposes a learnable matrix to weight pseudo-coordinates in k-ring neighbors over the graph. Wang *et al.* [2019b] present a *Dynamic Graph CNN* (DGCNN) on point cloud, which can build dynamic connections by selecting a k-neighborhood in feature space. SpiralNet [Lim et al. 2018] and SpiralNet++ [Gong et al. 2019] define spiral convolution for a k-ring neighborhood on the mesh. The convolution operator of MeshCNN [Hanocka et al. 2019] is defined on the four edges of the two incident triangles of an edge. The problem with the graph convolution using a 1-neighborhood, a k-neighborhood or a k-ring neighborhood is that it is not suitable for inputs of different resolutions because the size of the receptive field changes depending on the discretization of the input. The *Graph Wavelet Neural Network* (GWNN) [Xu et al. 2019] uses the wavelet transform to formulate convolutions on a graph. The problem with this convolution method is that only a single-scale wavelet is used in the transformation. In addition, the number of filters in this algorithm is related to the number of vertices, so that convolution in multiple resolutions cannot be achieved.

Even though there are many methods of graph convolutional networks, a graph convolutional network is rarely used to learn shape descriptors. One of the problems is that graph convolutions are strongly influenced by the neighborhood relationships stemming from the surface discretization. Therefore, the result is overly sensitive to the discretization of the surface. We focus on the issues of resolution and triangulation in graph convolutional networks and present a novel network to generate an informative descriptor that is robust to the change of resolution and triangulation.

3 PROBLEM STATEMENT AND OVERVIEW

Given is a mesh $\mathcal{M}$ as discretization of an underlying smooth two-manifold surface defined as (V, E), where $V = \{v_i | i = 1, ..., N\}$ and E are the sets of vertices and edges, respectively. The vertex coordinates are defined by the function $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) : V \to \mathbb{R}^3$ Our goal is to compute a local descriptor $\mathrm{f}(v_i) \in \mathbb{R}^d$ for any given vertex v_i.

The local descriptor is generated in two stages: non-learned descriptor computation and supervised descriptor learning. At the first stage, we compute our proposed descriptor WEDS for each vertex in the wavelet domain that is robust to the change of resolution, triangulation, scale, and rotation. The second stage is about descriptor learning. Inspired by the derivation of WEDS, we propose a graph convolutional network called MGCN to generate better descriptors from WEDS. Benefiting from the expressiveness of graph wavelets, our network can be trained on one resolution and tested on other resolutions without significant reduction of performance.

4 NON-LEARNED DESCRIPTOR COMPUTATION

In this section, we first review Laplacian eigenfunctions and graph wavelets. Then, we propose a new type of shape descriptor, WEDS. Finally, we discuss properties and advantanges of WEDS.

4.1 Laplacian Eigenfunctions

Let $\mathcal{S}$ denote a continuous surface. We can find an orthonormal basis on $\mathcal{S}$, containing the k smoothest possible functions that are orthogonal to each other, by finding the first k eigenfunctions of Δ [Bronstein et al. 2017], the Laplace-Beltrami operator:

$$\Delta\phi_i = \lambda_i\phi_i, i = 0, 1, ..., k-1, \tag{1}$$

where $\{\lambda_i | i = 0, 1, ..., k-1\}$ are the smallest k eigenvalues in increasing order. To simplify the notation, we use the same variable names for discrete and continuous settings. The difference should be clear from the context.

In the discrete setting of a triangulated mesh $\mathcal{M}$ with N vertices, we can discretize the Laplace-Beltrami operator as follows:

$$\mathbf{L}\boldsymbol{\phi_i} = \lambda_i\mathbf{A}\boldsymbol{\phi_i}, i = 0, 1, ..., k-1, \tag{2}$$

where $\mathbf{L}$ is a standard cotangent Laplacian matrix with size $N \times N$, $\mathbf{A}$ is the $N \times N$ diagonal area matrix. $\boldsymbol{\phi_i}$ is a $N \times 1$ vector, the eigenvector with respect to the eigenvalue λ_i. Also, note that for the generalized eigenvalue problem, the eigenvectors $\boldsymbol{\phi_i}$ are orthogonal to each other in terms of the $\mathbf{A}$-dot product: $\left\langle \boldsymbol{\phi_i}, \boldsymbol{\phi_j} \right\rangle_{\mathbf{A}} = \boldsymbol{\phi}_i^T\mathbf{A}\boldsymbol{\phi_j}$. Any function f defined on a smooth surface can be expressed as a weighted combination of the eigenfunctions: $f = \sum_{j=0}^{\infty}\sigma_j\phi_j$, where σ_j is the coefficient corresponding to the j^{th} eigenfunction. In the discrete case, the coefficients can be calculated by

$$\sigma_j = \left\langle \mathbf{f}, \boldsymbol{\phi}_j \right\rangle_{\mathbf{A}} = \mathbf{f}^T\mathbf{A}\boldsymbol{\phi_j}, \tag{3}$$

where $\mathbf{f}$ is the corresponding discretization of a smooth function f that is defined on the vertices of a triangulated mesh.

We can see that with the help of basis functions, a function on the surface can be transformed into a set of coefficients. Graph wavelets, explained next, make use of the same idea. The main difference is that instead of eigenfunctions of the Laplace-Beltrami operator, wavelet and scaling functions are used as basis functions.

4.2 Graph Wavelets

We build on the graph wavelet framework described in paper by Hammond et al. [2011]. To make the application specific to meshes using the cotangent Laplacian rather than the uniform Laplacian we build on the notation of [Masoumi and Hamza 2017] where the inner product is defined with respect to the area matrix $\mathbf{A}$.

Wavelet function. One graph wavelet function $\boldsymbol{\psi_{t,v}}$ is defined per vertex v per time scale t. We denote the number of time scales as K (K is typically less than 100). To construct a wavelet function, a filter function g is used. Examples for g are the Mexican hat or cubic splines. Let $a(v)$ be the Voronoi area at vertex v and $\phi_j(v)$ be the element of vector $\boldsymbol{\phi}_j$ corresponding to vertex v. Then $\boldsymbol{\psi_{t,v}}$, the spectral graph wavelet localized at vertex v and scale t, is given by

$$\boldsymbol{\psi_{t,v}} = \sum_{j=0}^{N-1} a(v)\, g\left(t\lambda_j\right)\phi_j(v)\,\boldsymbol{\phi_j}. \tag{4}$$

We show examples in Fig. 2 to demonstrate the local property of wavelets.

To project a given function $\mathbf{f}$ onto the wavelet basis, we compute the inner product between the wavelet functions and the given function $\mathbf{f}$. The spectral graph wavelet coefficients are then defined as:

$$W_{\mathbf{f}}(t, v) = \left\langle \mathbf{f}, \boldsymbol{\psi}_{t,v} \right\rangle_{\mathbf{A}} = \sum_{j=0}^{N-1} a(v)\, g\left(t\lambda_j\right)\sigma_j\phi_j(v). \tag{5}$$

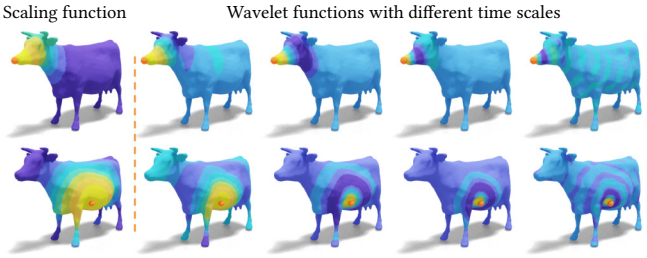

Fig. 2. Illustration of wavelet functions and scaling functions. The top row corresponds to a vertex on the mouth and the bottom row to a vertex on the stomach (shown as orange spheres). The first column shows the scaling functions ξ_v corresponding to the vertices. The second to fifth columns correspond to wavelet functions $\boldsymbol{\psi}_{t,v}$ of different time scales t = 0.0210, t= 0.0102, t= 0.0024, and t=5.59e^{-4}.

Scaling function. In addition to the graph wavelet functions, there is a single scaling function $\boldsymbol{\xi_v}$ defined per vertex v on the surface. The scaling function is defined via a filter function $h(x)$. Typically, $h(x)$ is the same type of function as $g(x)$, but using different parameters. The scaling function captures low-frequency information and is given by

$$\boldsymbol{\xi_v} = \sum_{j=0}^{N-1} a(v)\, h\left(\lambda_j\right)\phi_j(v)\,\boldsymbol{\phi_j}. \tag{6}$$

An illustration is shown in Fig. 2. Similar to the wavelet functions, we compute the inner product between a given function $\mathbf{f}$ and the scaling functions to obtain the scaling function coefficients

$$S_{\mathbf{f}}(v) = \langle \mathbf{f}, \xi_v \rangle_{\mathbf{A}} = \sum_{j=0}^{N-1} a(v)\, h\left(\lambda_j\right)\sigma_j\phi_j(v). \tag{7}$$

To use graph wavelets in our framework, the filter functions $g(x)$ and $h(x)$ cannot be chosen arbitrarily, as we require that they form a Parseval frame [Stanković and Sejdić 2019]. This is necessary so that the coefficients can be used to recover the original signal in the discrete case as follows:

$$\mathbf{f} = \sum_{m=1}^{K}\sum_{v} a(v)^{-1} W_{\mathbf{f}}(t_m, v)\,\boldsymbol{\psi_{t_m,v}} + \sum_{v} a(v)^{-1} S_{\mathbf{f}}(v)\,\boldsymbol{\xi_v}, \tag{8}$$

where t_m is the m^{th} scale of the wavelet function. We will discuss Parseval frames and the choice of filter functions in Section 4.4.

The above formula can be simplified as

$$\mathbf{f} = \sum_{m=0}^{K}\sum_{v} a(v)^{-1} W_{\mathbf{f}}(t_m, v)\,\boldsymbol{\psi_{t_m,v}}, \tag{9}$$

where $W_{\mathbf{f}}(t_0, v) = S_{\mathbf{f}}(v)$, and $\boldsymbol{\psi_{t_0,v}} = \boldsymbol{\xi_v}$. The proof is given in the appendix.

4.3 Wavelet Energy Decomposition Signature

To derive our new descriptor, we combine multiple ideas. First, we would like to start from the coordinate functions $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, because they completely describe the shape and are therefore very

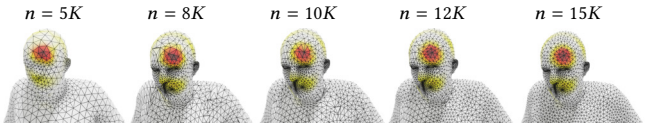

Fig. 3. We choose five resolutions to show the wavelet function on one vertex. From left to right: 5K, 8K with random filpping, 10K, 12K, 15K. The model with 5K vertices is remeshed and 8K is with random filpping. From the illustration, the wavelet functions are robust with respect to change of resolution and triangulation.

informative. Second, to make this information invariant to rigid transformations, we employ the sum of the Dirichlet energy of the three coordinate functions. This summation equals to the surface area and the computation of the Dirichlet energy is generally robust to different discretizations. Third, to aggregate local information in an area around the vertex, we employ graph wavelets at different scales described previously. This ensures that our descriptor is more discriminative than current state-of-the-art descriptors.

Given a smooth real-valued function $f : \mathcal{S} \rightarrow \mathbb{R}$ defined on the surface, the Dirichlet energy measures how smooth the function f is over the surface $\mathcal{S}$:

$$E(f) = \int_S |\nabla f(v)|^2 dv = \int_S f(v)\,\Delta f(v) dv. \tag{10}$$

In its discrete form, the Dirichlet energy is computed as $\mathbf{f}^{\mathrm{T}}\mathbf{A}\mathbf{L}\mathbf{f}$. Combined with Equations (1), (4), (5) and (9), the Dirichlet energy of the function can be expressed in the graph wavelet basis as follows:

$$E(\mathbf{f}) = \mathbf{f}^{\mathrm{T}}\mathbf{A}\mathbf{L}\mathbf{f} = \sum_{j=0}^{N-1} \lambda_j \left(\sum_{m=0}^{K} \sum_{v} \gamma_j(t_m, v) \right)^2, \tag{11}$$

where $\gamma_j(t_m, v) = W_{\mathbf{f}}(t_m, v)\, g_{t_m}(\lambda_j)\, \phi_j(v)$ and N is the number of vertices. We define $g_{t_m}(\lambda_j)$ as follows:

$$g_{t_m}(\lambda_j) = \begin{cases} h(\lambda_j), & if \quad m = 0 \\ g(t_m \lambda_j), & if \quad m > 0 \end{cases} \tag{12}$$

The Dirichlet energy of a vector-valued function $\mathbf{F} = (\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_{\mathrm{d}}) : V \rightarrow \mathbb{R}^d$ on the mesh is defined as the sum of the Dirichlet energy of the individual components:

$$E(\boldsymbol{F}) = \sum_{i=1}^{d} E(\mathbf{f}_i). \tag{13}$$

Now, we substitute the coordinate functions $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) : V \rightarrow \mathbb{R}^3$ into the Equation (13) above and express the result in the wavelet basis following Equation (11):

$$E(\mathbf{X}) = \sum_{i=1}^{d} \sum_{j=0}^{N-1} \lambda_j \sum_{m=0}^{K} \sum_{v} \gamma_{ij}(t_m, v)\omega_{ij} \tag{14}$$

$$= \sum_{m=0}^{K} \sum_{v} \sum_{j=0}^{N-1} \lambda_j \sum_{i=1}^{d} \gamma_{ij}(t_m, v)\omega_{ij}, \tag{15}$$

where $\omega_{ij} = \sum_{m=0}^{K} \sum_{v} \gamma_{ij}(t_m, v)$. The Dirichlet energy of this generalized function can be decomposed into different scale energies $\sum_{j=0}^{N-1} \lambda_j \sum_{i=1}^{d} \gamma_{ij}(t_m, v)\omega_{ij}$ on each vertex.

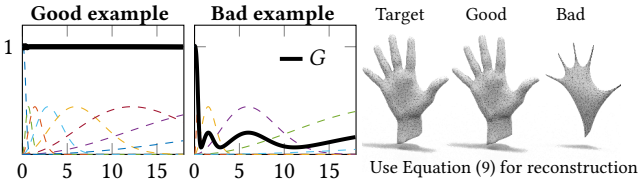

Fig. 4. We show two choices of filters (dashed lines) and the function G defined in Equation (18) (solid black line). We then use these two sets of filters to reconstruct a hand shape, *i.e.*, use Equation (9) to reconstruct the 3D coordinate functions. We can see that for the bad filter choice, where the constraint $G = 1$ is not satisfied, the reconstruction quality is very poor. On the other hand, the good filters can better reconstruct the given signal.

In addition, we ignore the first term when $j = 0$ since the first eigenvalue of a mesh $\lambda_0 = 0$. The Dirichlet energy can be decomposed into K multiscale vectors with lengths equal to the number of vertices and can be expressed as:

$$\boldsymbol{\varepsilon}_{t_m} = \left\{ \sum_{j=1}^{N-1} \lambda_j \sum_{i=1}^{d} \gamma_{ij}(t_m, v)\, \omega_{ij} \right\}, m \in [0, K]. \tag{16}$$

After energy decomposition, the amount of energy of an individual vertex depends on the number of vertices on the surface. Because the Dirichlet energy of a shape with different resolutions or discretizations is constant, the higher the resolution of a mesh is, the less energy each vertex has. To get features of similar scale at a surface point when the underlaying discretization changes, we need to collect local energy at each vertex to form our signature. Finding the geodesic neighbors of a surface point is very time-consuming. Fig. 3 shows the wavelet basis functions at the meshes of different resolutions. We find that the shape of the wavelet does not change significantly on the meshes of different resolutions, which can be used to weight different resolutions of meshes. Thanks to the natural local properties of graph wavelets, the local wavelets can be used to collect the local energy to compute our signature on different resolutions. A wavelet $\boldsymbol{\psi}_{t_s, v}$ at a particular vertex can be thought of as the associated weights of points with the particular vertex. The weights of points are significant when the points near the specific vertex when measured by geodesic distance, and the influence is small if points is far from the vertex. For one scale t_s and every vertex v, we first normalize the wavelet $\boldsymbol{\psi}_{t_s, v}$ to obtain a normalized vector $\boldsymbol{\psi}_{t_s, v^*}$ using minmax normalization. Then, we use normalized vector $\boldsymbol{\psi}_{t_s, v^*}$ to weight energy $\boldsymbol{\varepsilon}_{t_m}$. The formula of our signature on one vertex v at one scale t_s is expressed as follows:

$$WEDS_{t_s}(v) = \left\{ \sum_{x} \psi^*_{t_s, v}(x) \varepsilon_{t_m}(x) \right\}, m \in [0, K]. \tag{17}$$

To obtain scale invariance, the energy vectors $\boldsymbol{\varepsilon}_t$ are modified by multiplying its eigenvalue λ_j like LPS [Wang et al. 2019a]. Therefore, $\boldsymbol{\varepsilon}_{t_m} = \left\{ \sum_{j=1}^{N-1} \lambda_j^2 \sum_{i=1}^{d} \gamma_{ij}(t_m, v)\, \omega_{ij} \right\}$. To construct our vertex descriptor, the weighting approach with one wavelet scale is not discriminative. Therefore, we cascade the descriptors at different scales in the scale set $\boldsymbol{S}_{t_s}$: $\mathrm{WEDS}(v) = \left\{ \mathrm{WEDS}_{t_s}(v) \right\}, t_s \in \boldsymbol{S}_{t_s}$. We will mention how to select scales in Section 4.4.

Fig. 5. We show WEDS on two different shapes with different poses and resolutions. We show 8 different dimensions of our descriptors. We can see that WEDS is robust with respect to the triangulation and resolution.

4.4 Multiscale Filters

If an original signal can be recovered by a graph wavelet basis (recall Equation (8)), the filters need to satisfy the Parseval frame,

$$G(\lambda_j) = h^2(\lambda_j) + \sum_{m=1}^{K} g^2(t_m\lambda_j) \equiv 1, \tag{18}$$

and the proof is given in Appendix. Fig. 4 visualizes the importance of constraint G. We show two choices of filter functions, one that does and one that does not satisfy this constraint. We choose the Mexican hat functions as the filters of the graph wavelet, which are given by:

$$g(t_m\lambda_j) = A(t_m\lambda_j)^2 e^{\left(1-(t_m\lambda_j)^2\right)}, \quad h(\lambda_j) = B\, e^{\left(-\left(\frac{C\lambda_j}{\lambda_{\max}}\right)^3\right)},$$

where $t_m = e^{\text{linspace}\left(\log\left(\frac{D}{\lambda_{\max}}\right),\log\left(\frac{E}{\lambda_{\max}}\right),K\right)}$. Considering the balance of efficiency and accuracy, in our tests, we choose 32 wavelet filters, *i.e.*, $K = 31$. We set the tolerance to 0.01, and the five parameters can be solved: $A = 0.443, B = 1.004, C = 38.462, D = 46, E = 0.2$.

For the scale set S_{t_s}, the selection is based on the number of dimensions generated by the features on each vertex. For one scale in S_{t_s}, we can generate 32-dimensional features. So WEDS can generate features with a maximum of 1024 dimensions. But high-dimensional features are usually not needed. To represent high, medium, and low frequencies, we select at least 3 scales, and features with at least 96 dimensions are generated, after which we can obtain fewer dimensions by sampling. If more dimensions of feature need to be generated, then more scales need to be picked. We take a linear sampling approach and remove the first and last filters, which is as follows:

$$S_{t_s} = \left\{ t_m, m = \left\lfloor \text{linspace}(32, 1, \left\lceil \frac{\text{Num}}{32} \right\rceil + 2) \right\rfloor (2 : \text{end} - 1) \right\}, \tag{19}$$

where Num is the feature dimension of the output. In Fig. 5, we show our WEDS descriptors on two shapes with 8 selected dimensions.

4.5 Discussion

Many spatial and spectral descriptors have been proposed, but these descriptors cannot satisfy the expected property simultaneously, such as resolution, scale, and discrimination. Our goal is to find a new descriptor that can be discriminative and robust to different shape structure at the same time. We are inspired by the following observation: for a smooth surface $\mathcal{S}$ to any triangulated mesh $\mathcal{M}$ with any resolution. If a discrete vector $\mathbf{f}$ is sampled from a smooth function f, the discrete Dirichlet energy $E(\mathbf{f}) = \mathbf{f}^{\mathrm{T}}\mathbf{A}\mathbf{L}\mathbf{f}$ is robust to discretization. Table 1 shows two smooth functions, which is vertex coordinate and WKS.

Table 1. The discrete Dirichlet energy $E(\mathbf{f})$ of the vertex coordinate functions and the WKS functions.

	Res	Dims1	Dims2	Dims3	SUM	Area*2
$E(\mathbf{f})$ of Coord	6890	1.0153	1.4768	1.1114	3.6035	3.6035
	10K	1.0156	1.4716	1.1092	3.5964	3.5964
	15K	1.0135	1.4719	1.1061	3.5915	3.5915
	Res	Dims1	Dims2	Dims3	Dims4	SUM
$E(\mathbf{f})$ of WKS	6890	0.7645	2.9629	4.8548	4.0352	12.6174
	10K	0.7639	2.9764	4.8979	4.0372	12.6754
	15K	0.7581	2.9836	4.7605	4.0883	12.5905

It can be found that the discrete Dirichlet energy on every dimension and its sum on this two smooth functions are robust to the change of resolution. In addition, Dirichlet energy is invariant to a rigid transformation, which is very important in feature design. Therefore, we want to derive a set of descriptors from the Dirichlet energy of a given function $\mathbf{f}$. An interesting phenomenon is that the Dirichlet energy of vertex coordinates has been proved to be twice of surface area of the mesh, and the surface area is very robust to different discretization. The coordinates are also the most primitive and comprehensive information of given shape, so we choose the vertex coordinate function as input. To have other desirable properties, we just need to pick a different function $\mathbf{f}$.

To derive a set of per-vertex descriptors from this energy, we need to distribute the energy of the function $\mathbf{f}$ to the vertices. One trivial solution is simply use the Laplacian-Beltrami basis, where we can project function $\mathbf{f}$ to the Laplacian-Beltrami basis like Equation (3). However, in this case, we only have σ_j that characterizes the global attributes of the shape. One possible way to get local features for each vertex is to cut a geodesic disk like LPS [Wang et al. 2019a], and compute the Dirichlet energy locally, but it is time-consuming and does not contain global information.

Another choice is to distribute the energy of $\mathbf{f}$ using the graph wavelet basis, which are a set of basis defined on each of the vertices. In this case, the wavelet basis can capture the local details in a spatial region around the vertex. In addition, wavelets on graphs are expressed using eigenfuctions of the Laplacian-Beltrami operator, and we can project function $\mathbf{f}$ to the wavelet basis and get the coefficients in Equation (5). After projection, it can be found that the wavelet coefficient $W_{\mathbf{f}}(t, v)$ contains the coefficient σ_j which includes global information of $\mathbf{f}$. Because the coefficients of graph wavelets can capture both global and local information, the reconstructed energy of $\mathbf{f}$ can be distributed to each vertex while maintaining the global and local information at the same time. Therefore, we derive a descriptor with high discrimination while maintaining robustness. To sum up, wavelets enable us to achieve a trade-off between local and global information that other descriptors are unable to achieve.

5 SUPERVISED DESCRIPTOR LEARNING

We propose a new graph convolutional network called MGCN. We mainly employ MGCN for descriptor learning in this paper, but it is a general architecture for graph convolutional networks. We first

describe a single layer of our network in Section 5.1, and then the complete architecture and training details in Section 5.2.

5.1 Multiscale Graph Convolution Layer

A layer of our network takes a C-dimensional vector for each of the N graph nodes as input and outputs an O-dimensional vector for each node. For simplicity of notation, we start the description by considering the case of $C = O = 1$ and extend to higher dimensions in the end. We focus on discrete descriptions for simplicity. Under this simplifying assumption, we denote the input as signal $\mathbf{x}^{\text{in}} \in \mathbb{R}^N$ and the output as $\mathbf{x}^{\text{out}} \in \mathbb{R}^N$. The goal of our layer is to convolve the signal $\mathbf{x}$ with a filter $\mathbf{y} \in \mathbb{R}^N$, *i.e.*, to compute $\mathbf{y} *_w \mathbf{x}$ where $*_w$ is the convolution operator. Convolution in the time domain is equal to the product in the frequency domain. Following previous work, we consider spectral convolutions on graphs defined as:

$$\mathbf{x}^{\text{out}} = \mathbf{y} *_w \mathbf{x}^{\text{in}} = \mathbf{\Phi}\left(\left(\mathbf{\Phi}^T\mathbf{y}\right) \odot \left(\mathbf{\Phi}^T\mathbf{x}^{\text{in}}\right)\right) = \mathbf{\Phi}\mathbf{w}_\theta\mathbf{\Phi}^T\mathbf{x}^{\text{in}}, \tag{20}$$

where $\odot$ is the element-wise product, $\mathbf{\Phi} \in \mathbb{R}^{N\times k}$ is an eigenvector matrix, and $\mathbf{w}_\theta \in \mathbb{R}^{k\times k}$ is a diagonal matrix describing the filter in the frequency domain. In general, $\mathbf{w}_\theta$ can be considered to be a function of the eigenvalue matrix $\mathbf{\Lambda}$, *i.e.*, $\mathbf{w}_\theta = f(\mathbf{\Lambda})$. To improve efficiency, ChebyNet [Defferrard et al. 2016] approximates the arbitrary function f as weighted sum of powers of $\mathbf{\Lambda}$:

$$\mathbf{w}_\theta = \sum_{m=0}^{K-1} \theta_m \text{diag}\left(\left\{\lambda_j\right\}_{j=0}^{k-1}\right)^m = \sum_{m=0}^{K-1} \theta_m \mathbf{\Lambda}^{\boldsymbol{m}}, \tag{21}$$

where $\mathbf{\Lambda}$ is a diagonal matrix of size $K \times K$, and the elements on the diagonal are eigenvalues λ_j.

In this way, the convolution can be replaced by a linear combination of m-order polynomials of the Laplacian matrix, which eliminates the need to calculate the eigenfunctions. Laplacians with order m are m-localized. For efficient computation, Chebyshev polynomials are computed recursively as follows:

$$\mathbf{x}^{\text{out}} = \mathbf{y} *_w \mathbf{x}^{\text{in}} \approx \sum_{m=0}^{K} \theta_m T_m(\mathbf{L})\mathbf{x}^{\text{in}}, \tag{22}$$

where $T_m(\boldsymbol{L}) \in \mathbb{R}^{N\times N}$ is the m-order Chebyshev polynomial.

From another point of view, the convolution of the ChebyNet can be understood as the sum of polynomials of different orders evaluated at the Laplacian. If the order is m, the receptive field of the convolution is the m-ring neighborhood of a vertex. The ChebyNet is equivalent to the weighted sum of multiple convolutions with increasing receptive field. The problem of this convolution is that it is not resolution independent, because the size of the m-ring neighborhood depends on the discretization of the surface.

Our idea is to express $\mathbf{w}_\theta$ in a wavelet filter basis, rather than a polynomial basis. Note that there is a difference between the wavelet filter basis and the wavelet basis. The wavelet filter basis is a basis in the spectral domain, but the wavelet basis exists in the spatial domain on the surface:

$$\mathbf{w}_\theta = \sum_{m=0}^{K} \theta_m \text{diag}\left(\left\{g_{t_m}(\lambda_j)\right\}_{j=0}^{k}\right) = \sum_{m=0}^{K} \theta_m g_{t_m}(\mathbf{\Lambda}). \tag{23}$$

Fig. 6. Visualization of filters learned by our network. For one input mesh and one selected vertex on the head, we show 24 manually selected and normalized filters to ensure some variability. Note that the network learns about 60K filters so this is only a small subset. Further, the filters are normalized, since the range of values between different filters differs significantly.

Now the convolution can be simplified as follows:

$$\mathbf{x}^{\text{out}} = \mathbf{y} *_w \mathbf{x}^{\text{in}} \approx \sum_{m=0}^{K} \theta_m \mathbf{\Psi}_{t_m}^T \mathbf{x}^{\text{in}}, \tag{24}$$

where each matrix $\mathbf{\Psi}_{t_m} \in \mathbb{R}^{N\times N}$ is composed of the wavelet basis with scale t_m.

In practice, there are three factors to consider. First, the magnitude of high frequency wavelet functions is very small, which leads to numerical robustness issues during learning the parameters. Second, we need to ensure that the convolution operation leads to similar results for surfaces that are discretized with different sampling densities. Therefore, to solve the first two issues, we perform L_1 normalization on the wavelet functions (*i.e.*, normalize the columns of the wavelet matrix $\mathbf{\Psi}$). Finally, we do not need to select all the filters, we only need to select a part (such as sampling about half) of the filters to reduce the calculation. Therefore, the new convolution is as follows,

$$\mathbf{x}^{\text{out}} = \mathbf{y} *_w \mathbf{x}^{\text{in}} \approx \sum_{t_s \in S_{t_s}} \theta_{t_s} \overline{\mathbf{\Psi}_{t_s}}^T \mathbf{x}^{\text{in}}, \tag{25}$$

where $\overline{\mathbf{\Psi}_{t_s}}$ is a normalized matrix with each wavelet L_1 normalized, where the sum of the normalized wavelet is 1.

For the high-dimensional case with $\mathbf{X}^{\text{in}} \in \mathbb{R}^{\text{N}\times\text{C}}$, we form our convolution module by the following formula:

$$\mathbf{Z} = \text{Norm}\left(\text{ELU}\left(\sum_{\mathbf{t_s}\in\mathbf{S_{t_s}}} \overline{\mathbf{\Psi}_{\mathbf{t_s}}}^{\text{T}} \mathbf{X}^{\text{in}} \mathbf{W}_{\mathbf{t_s}}\right)\right), \tag{26}$$

where $\mathbf{W}_{\mathbf{t_s}} \in \mathbb{R}^{C\times O}$ and O are the dimensions of the output feature, and Norm() is a minmax normalization on every dimensional feature to normalize energies of vertices. This module can be called as "MGCONV".

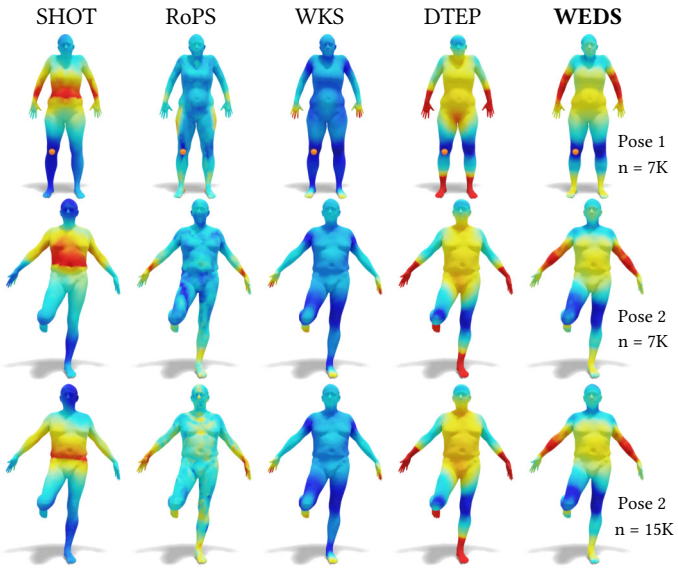

Fig. 7. For a selected vertex on the knee (shown in orange) we visualize the distance of the descriptor on the selected vertex to the descriptors of other vertices. A blue color indicates a small distance and a red color a large distance. Top row: descriptor distances on the same mesh. Second row: descriptor distances on a different mesh in the same resolution. Third row: descriptor distances on a different mesh in a different resolution. Form left to right: SHOT, RoPS, WKS, DTEP, and WEDS. We can observe that WEDS and DTEP are more discriminative than WKS and that SHOT and RoPS are not resolution independent.

5.2 Network Architecture Details

The MGCONV layer described previously can be used to construct our MGCN. Here, we describe the architecure we used for desriptor learning. To learn a shape descriptor, we build an MGCN network by stacking 6 layers of MGCONV and one fully connected layer: 5× MGCONV96(16) + MGCONV128(16) + FC256. MGCONVx(k) refers to a convolutional layer that has an x-dimensional output of feature maps, and (k) refers to scale k of our wavelet scale set. and FCx refers to a fully connected layer that outputs a vector with x-dimension, and 256 refers to the 256 dimension of the output feature. To have a fair comparison and equal number of parameters in the network, the dimension of all input features is set to 128.

We use two phases of training. In the first phase, a classification network is used to train the MGCN. One fully connected layer FCd is added after the last MGCONV layer, and the cross-entropy loss is used to classify each point. For the second phase, we propose to use the HardNet loss [Mishchuk et al. 2017]. Its advantage is that it can directly calculate the loss of N pairs by computing N triplet distances. We apply this loss to the task of learning shape descriptors. The HardNet loss is used to directly train MGCN to reduce the distance between positive examples and increase the distance between negative examples. During training, we use a batch size of 1 because the shapes in the dataset can have a different number of vertices. In our tests, we first use training with cross-entropy loss for 200 epochs and then train with HardNet loss for 100 epochs. Compared with training the network only using cross-entropy loss with 300 epoch, this configuration can further reduce average geodesic error about 0.01. Training using only the HardNet loss is slower, so the first phase can be considered an acceleration strategy to have a good initialization. For training with the cross-entropy loss, we use the ADAM optimizer ($\beta_1 = 0.9, \beta_2 = 0.999, \varepsilon = 10^{-8}$) with a learning rate of 10^{-3} and a weight decay of 10^{-4}. For training with the HardNet loss, we use the ADAM optimizer with a learning rate of 5×10^{-4} and a weight decay of 5×10^{-5}.

To give some intuition of the learned filters, we visualize a small subset of them in Fig. 6 for a training on the FAUST dataset.

5.3 Discussion

Spectral CNN exploits the fact that the convolution in the time domain is equal to the product in the frequency domain, but this network leads to descriptors that are too smooth and do not contain enough local information. ChebyGCN uses k-order Chebyshev polynomials to expand filters in the frequency domain (Note that in this section k does not refer to the size of the Laplacian basis). GCN further simplifies ChebyGCN and uses a 1-ring neighborhood to approximate filters to reduce the amount of calculations. By contrast, SplineCNN and DGCNN belong to the class of networks using spatial convolution. The core questions of the spatial convolution method are how to find neighbors and how to aggregate the information from the neighbors. Comparing convolution in the frequency domain to the convolution in the spatial domain, it can be seen that the locality is important to improve performance. But this locality also brings some problems. For example, GCN and SplineCNN use 1-ring neighborhood information. ChebyGCN uses k-order Laplacian polynomials, so it uses k-ring neighborhood information. DGCNN uses k-nearest neighbors in feature space to define the support of a convolution filter. However, such convolution operations do not generalize well to meshes of different resolution, because the size of a k- or 1-ring neighborhood (the receptive field of the filters) changes with the resolution of a mesh. An alternative method is to calculate a geodesic disk centered at each vertex and locally resample the surface [Wang et al. 2019a]. However, computing dense geodesic disk is time-consuming and resampling the surface introduces additional errors.

An alternative approach is the graph wavelet neural network [Xu et al. 2019]. The formula for convolution is $\mathbf{y} *_{\mathbf{g}} \mathbf{x} = \mathbf{\Psi} \mathbf{g}_\theta \mathbf{\Psi}^{-1} \mathbf{x}^{\text{in}}$. However, because the filter $\mathbf{g}_\theta$ has parameters equal to the number of vertices, the network only works for meshes with the same number of vertices. Second, the calculation of matrix inversion is very slow for large graphs. In summary, there is currently no reliable network to handle multi-resolution datasets.

In our MGCONV layer, we use many advantages of the nature of wavelets, but make important changes to the existing graph wavelet neural network. First, the spectral filters can be expressed in the wavelet filter basis. In the spatial domain, convolution operations are replaced by multiplying with a matrix composed of the wavelet basis. Because the wavelet basis functions are robust to the change of resolution and triangulation, our convolution also has the ability to cope with different resolutions. Second, due to the multiscale nature of the wavelet basis, we have the ability to simultaneously capture local and global information. Most importantly, we can capture local information without the explicit computation of local

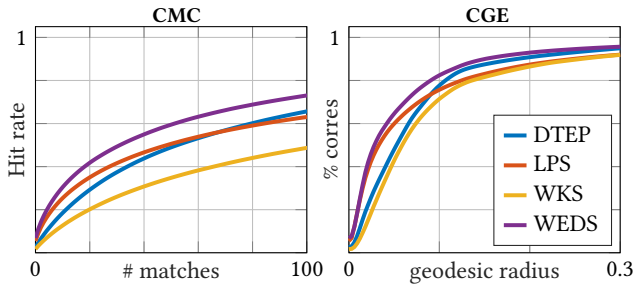

Fig. 8. The symmetric CMC and CGE metrics of non-learned descriptors on the FAUST dataset. We use different *line colors* to indicate different networks. Note that our WEDS descriptor is the most discrimintive especially according to the CMC curves.

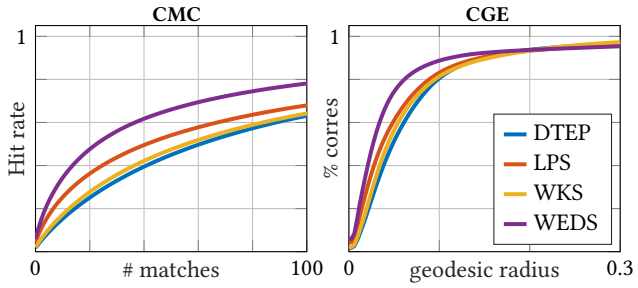

Fig. 9. The symmetric CMC and CGE metrics of non-learned descriptors on the SCAPE dataset. We use different *line colors* to indicate different networks. Note that our WEDS descriptor is the most discrimintive especially according to the CMC curves.

neighborhoods using geodesic distances as used in [Wang et al. 2019a].

6 EXPERIMENTAL RESULTS

We present multiple qualitative results. After describing the used evaluation metrics, we compare WEDS with other non-learned descriptors, compare MGCN with other network architectures for descriptor learning, and evaluate different parameter settings. In our experiments, the results are obtained using an Intel Core i7-7700 processor with 4.2 GHz and 16 GB RAM. Offline training is run on an NVIDIA GeForce GTX RTX (24 GB memory) GPU.

6.1 Evaluation Metrics

We use three metrics to evaluate descriptors: average geodesic error, cumulative geodesic error, and cumulative match characteristic. We report results for two types of ground truth matches: the closest vertex on the direct map and the closest vertex on the symmetric map. Specifically,

(1) **Average geodesic error** is a scalar to compute the average per-vertex error. The direct error evaluates the geodesic error between a predicted vertex and its ground-truth correspondence. The symmetry-aware error is the minimum of the direct error and the error between predicted vertex and the symmetric ground truth. The predicted vertex is obtained by computing the nearest-neighbor using the L_2 distance in feature space. For simplicity, the errors reported in all tables are scaled by 10^{-3}.

(2) **Cumulative geodesic error (CGE)** measures matching quality by plotting the percentage of nearest-neighbor correspondences that are at most r-geodesically distant from the ground-truth correspondence. According to the type of ground truth used, CGE is also divided into direct CGE and symmetry-aware CGE.

(3) **Cumulative match characteristic (CMC)** evaluates the percentage of vertices that find a correct match among the k-nearest neighbors in the descriptor space. According to the type of ground truth used, CMC is also divided into direct CMC and symmetry-aware CMC.

For the network evaluation, we use the descriptor evaluation if the network is used for learning descriptors.

Table 2. Average (direct/symmetry-aware) geodesic error computed on 15×14 shape pairs with different descriptors. WEDS improves about 10% compared to the best competitor.

Descriptors	Dataset	
	FAUST(6890)	SCAPE(12.5K)
SI	352 / 153	380 / 251
SHOT	381 / 262	271 / 196
RoPS	346 / 252	267 / 187
HKS	511 / 396	507 / 409
WKS	335 / 118	260 / 72
LPS	325 / 97	227 / 71
DTEP	312 / 89	267 / 76
WEDS	**287 / 69**	**225 / 66**

Table 3. Average direct geodesic error computed on 8×7 non-isometric shape pairs with different descriptors.

Descriptors	SI	SHOT	RoPS	HKS
Error	482	456	413	475
Descriptors	WKS	LPS	DTEP	**WEDS**
Error	558	395	328	**314**

6.2 Non-learned Descriptor Evaluation

As competitors, we select three spatial domain descriptors (SI [Johnson and Hebert 1999], SHOT [Tombari et al. 2010], RoPS [Guo et al. 2013]) and four spectral domain descriptors (HKS [Sun et al. 2010], WKS [Aubry et al. 2011], LPS [Wang et al. 2019a], and DTEP [Melzi et al. 2018]).

6.2.1 Evaluation on near-isometric shapes. We first analyze the discriminative power of descriptors. To verify the effectiveness of the proposed local descriptor, we choose FAUST [Bogo et al. 2014] and SCAPE [Anguelov et al. 2005] as our test datasets. We also use an extended version of FAUST [Wang et al. 2019a], which contains meshes with different resolution, triangulation, scale, and rotation.

We conduct an extensive evaluation for different non-learned descriptors on FAUST and SCAPE. Between different humans, the pairs are isometric or near-isometric. To reduce variability for fair test, we selected 15 models on two datasets randomly and test every two pairs of fifteen models. The total number of tested pairs is 15×14. Table 2 shows the average geodesic error A/B on 15×14 pairs. A is direct error and B is symmetry-aware error.

From Table 2, we find that RoPS is the better among the spatial domain descriptors, but the geodesic error is still large. The results show that spatial domain descriptors cannot handle non-rigid matching well. In addition to HKS being too smooth, frequency domain descriptors have better performance. Among the frequency domain descriptors, WEDS has the best discrimination among state-of-the-art descriptors two datasets and has about 10% performance

Table 4. Average (direct/symmetry-aware) geodesic error computed on 15×14 shape pairs of FAUST with different descriptors. WEDS + MGCN significantly improves upon the best previous work LPS + Geodesic-based.

Descriptors	Network	#Resolution		
		6890 - 6890	6890 - 10K	6890 - 15K
Histogram	CGF	424 / 298	433 / 300	460 / 332
-	OSD	398 / 182	457 / 254	430 / 221
-	SplineCNN	276 / 161	488 / 378	524 / 438
WEDS	ChebyGCN	**6 / 1**	527 / 387	551 / 377
WKS	Geo-based	204 / 35	221 / 52	252 / 69
LPS	Geo-based	164 / 22	203 / 43	223 / 55
WEDS	Geo-based	147 / 24	207 / 34	239 / 39
WKS	MGCN	19 / 13	88 / 34	124 / 69
LPS	MGCN	18 / 12	44 / 24	84 / 39
WEDS	MGCN	8 / 7	**26 / 18**	**66 / 34**

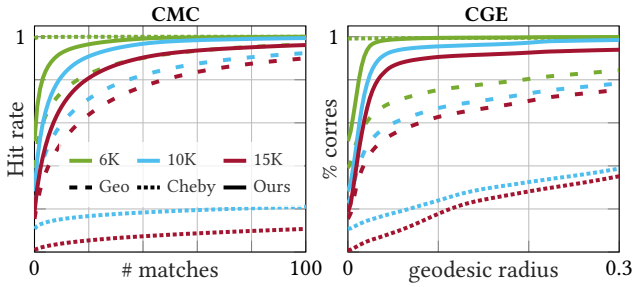

Fig. 10. The direct CMC and CGE metrics of learned descriptors on the FAUST dataset. We use different *line types* to indicate different networks, where the Geo-based network [Wang et al. 2019a] is in dashed lines, the ChebyCNN is in dotted lines, and our MGCN is in solid lines. We use different *colors* to indicate different resolutions on the target shapes. Note that other networks do not generalize to different resolution as well as our network.

improvement. A visualization of the distance of the descriptor of a selected vertex to other vertices' descriptors is shown in Fig. 7. More details in the form of curves are provided in Fig. 8 and Fig. 9. Here we see that the improvement of the CMC metric is even more significant than the average geodesic error. Other tests are given in additional materials.

6.2.2 Evaluation on non-isometric shapes. Non-isometric shapes are more challenging to match than near-isometric shapes. We choose SMAL [Zuffi et al. 2017] as our test dataset to compare different non-learned descriptors. SMAL is a small dataset with four-legged shapes such as lions, horses, cows, and hippos, where all base shapes share the same triangulation connectivity. A shape pair is considered non-isometric if two shapes are from different categories. We introduce the remeshed SMAL 5K dataset where all shapes are remeshed independently to test the effectiveness of different descriptors.

For non-isometric shapes, we select 8 non-isometric models and test direct error on 8×7 pairs. Table 3 shows the results for different descriptors. From Table 3, we find that the descriptors directly constructed in the spatial domain such as SI, SHOT, and RoPS have average performance compared to other methods on non-isometric shapes. The intrinsic descriptors such as WKS and HKS have the worst performance. DTEP and our WEDS are better than the other descriptors, but we believe that there is a lot of room for improvement.

Table 5. Average (direct/symmetry-aware) geodesic error computed on 10×9 shape pairs of SCAPE with different descriptors. MGCN outperforms its best competitors ChebyGCN and Geo-based by a large margin.

Descriptors	Network	#Resolution	
		SCAPE 5K	SCAPE 5K-12.5K(Test)
Histogram	CGF	374 / 264	428 / 323
-	OSD	259 / 94	835 / 742
-	SplineCNN	297 / 180	503 / 353
WEDS	ChebyGCN	68 / 29	458 / 332
WKS	Geo-based	185 / 72	192 / 80
LPS	Geo-based	175 / 68	182 / 74
WEDS	Geo-based	163 / 65	179 / 71
WKS	MGCN	67 / 46	98 / 64
LPS	MGCN	54 / 26	82 / 44
WEDS	MGCN	**48 / 17**	**73 / 39**

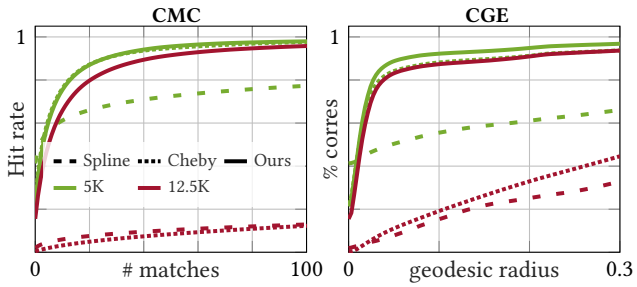

Fig. 11. The direct CMC and CGE metrics of learned descriptors on the SCAPE dataset. We use different *line types* to indicate different networks and different *colors* to indicate different resolutions on the target shapes. Note that other networks do not generalize to different resolution as well as our network.

6.3 Graph Network Evaluation

To test the effectiveness of the network, we first test the shape descriptors that have been further improved by the network. The network structure has been described in Section 5.2.

6.3.1 Learning descriptor on near-isometric shapes. In this task, we focus on learning robust descriptors on different resolutions. We use two datasets, FAUST and SCAPE, for evaluation that have been remeshed with different algorithms. FAUST has been remeshed by local remeshing operations [Wang et al. 2019c] that maintain the positions of the original vertices and SCAPE has been remeshed to about 5K vertices per shape using the LRVD remeshing method [Yan et al. 2014]. The vertex position and triangulation are totally different. We have a ground-truth correspondence between the remeshed vertex and exact 5K points sampled from the original dataset. There are not many shape descriptor learning approaches considering different resolutions. One of papers used geodesic-based networks embedded LPS [Wang et al. 2019a] to achieve the effect of training at one resolution and testing at another resolution without too much decline. In our study, three non-learned descriptors (WKS, LPS, WEDS) are selected. We set feature dimension to 128 for fair comparison. For network comparison, we choose the most competitive methods (CGF32 [Khoury et al. 2017], OSD [Litman and Bronstein 2014], SplineCNN(1-neiberhood) [Fey et al. 2018] and geodesic-based networks [Wang et al. 2019a]). We also build a network stacked by ChebyGCN (k-neiberhood) [Defferrard et al. 2016]

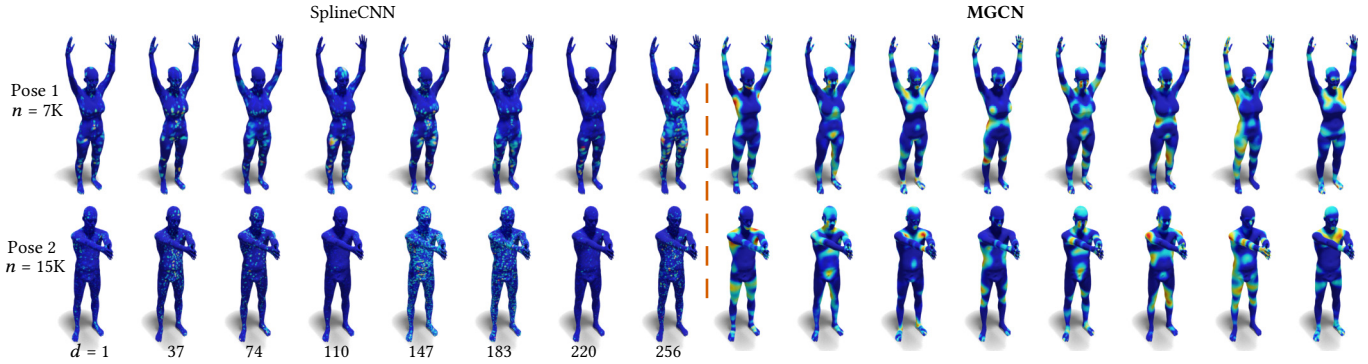

Fig. 12. We train SplineCNN and our MGCN on the WEDS descriptors as visualized in Fig. 5 and show the learned descriptors on the same shape and the same dimension. We can see that, though the learned descriptors from SplineCNN has reasonable accuracy in shape matching, the learned descriptors are not smooth and do not encode any semantic information. As a comparison, the learned descriptors from our network are much more coherent between shapes with different poses and resolutions. Also, our learned descriptors are more smooth.

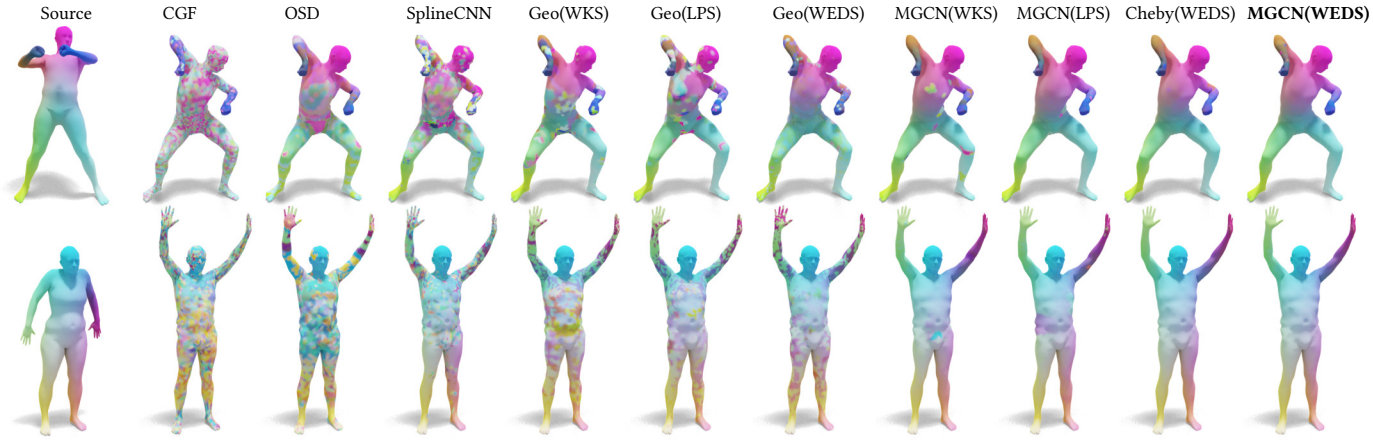

Fig. 13. Here we show an example pair from SCAPE (top row) and FAUST (bottom row) , where the source and the target shape has the same resolution and we compare the maps obtained from different methods visualized by color transfer. The descriptor input of a network is shown inside brackets.

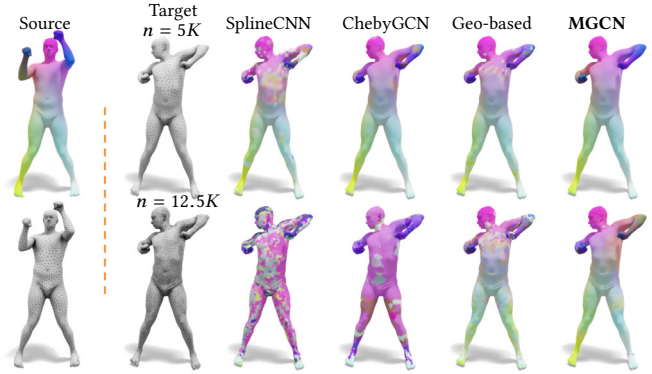

Fig. 14. Here we compare the performance of different methods with respect to different resolutions. The source shape has 5K vertices and the target shape has 5K vertices in the top row and 12K vertices in the bottom row. All the networks take WEDS as input. Note that the matches change considerably from the top row to the bottom row for all networks except our MGCN. This indicates that our network can greatly improves upon the generalization performance of the current state of the art.

layers. The structure is followed by Section 5.2, we replace MGCONV layers with ChebyGCN layers. The rest of the structure such as input and output dimension is the same as MGCN.

Experimental results on FAUST. In this setting, all the combinations for learning descriptors are generated by learning only on FAUST 6890 vertices and testing on FAUST 6890 and other vertices. We use the nearest neighbor of the feature space to detect the matching discrimination of descriptors between different resolutions. Table 4 shows the average geodesic errors on different settings. It can be seen that the ChebyGCN network significantly overfits and the performance drops by a factor of about 100 if the resolution changes. SplineCNN was designed to directly learn a mapping between points of different shapes, so that we had to make modifications to use SplineCNN for descriptor learning: we take the output of the second to last linear mapping layer as the descriptor to learn. In general, we observe that the performance of SplineCNN, CGF, and OSD is significantly worse than our MGCN. Based on the results we consider the geodesic-based network to be our main competitor. Similar to our network, the geodesic-based network is fairly robust to different surface discretizations, however, overall

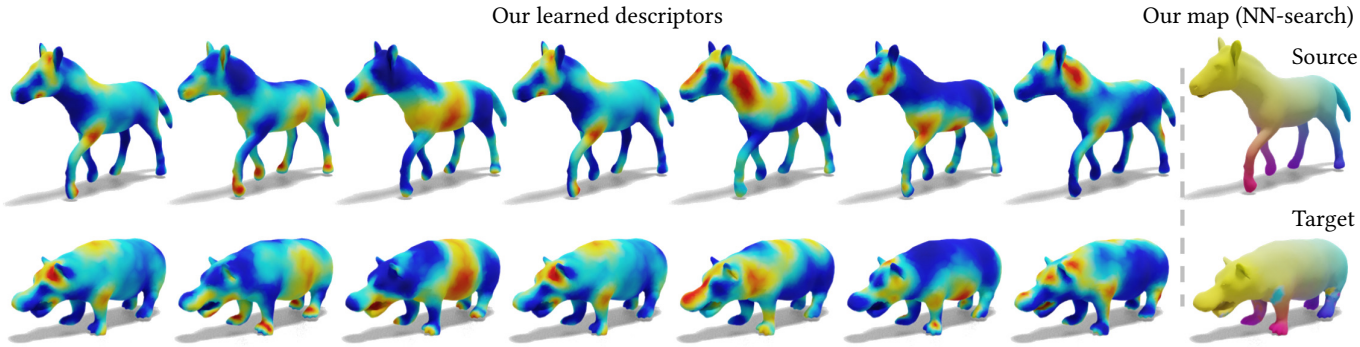

Fig. 15. Here we show an example pair from the SMAL dataset, where the source and the target shape have large deformation. The learned descriptors are shown on the left. We also show the map visualized by color transfer on the right.

Table 6. Comparison with deep functional maps (FMnet). Here we report the average (direct/symmetry-aware) geodesic error computed on 15×14 shape pairs from FAUST and 10×9 shape pairs from SCAPE.

Methods	Dataset	
	FAUST(6890)	SCAPE(12.5K)
FMNet(desc) + NN	169 / 58	205 / 93
FMNet	17 / 11	131 / 72
FMNet + ZoomOut	15 / 10	50 / 36
MGCN + NN	**8 / 7**	**46 / 16**
MGCN + NN + ZoomOut	**8 / 6**	**22 / 14**

Table 7. Average direct geodesic error computed on non-isometric animal shapes from SMAL dataset. Our method outperforms both BIM and SplineCNN, the current state-of-the-art non-learning and learning method, respectively.

Shape Pairs	BIM	SplineCNN + NN	MGCN + NN
Average (8×7 pairs)	59	225	**45**
Cow, Wolf	37	219	**18**
Tiger, Dog	39	227	**38**
Fox, MaleLion	104	221	**51**
Horse, Hippo	**76**	244	80

our results are much stronger. For example, the direct error is 3 to 18 times lower when our network is used accross different resolutions. Fig. 10 reports the curves for the CMC and CGE metric of different descriptors. Therefore, MGCN utilizes GCN's powerful fitting capabilities and guarantees robustness at different resolutions simultaneously. In addition, using different descriptors as input will also affect the network. The results show that the performance of the setting of WEDS and MGCN is the best.

Experimental results on SCAPE. The same network may behave differently on different datasets, we need another dataset to test the network. In this setting, all the combinations for learning descriptors are generated by learning only on SCAPE 5K vertices and testing on SCAPE 5K and 12.5K vertices. We then perform feature matching between different resolutions. SCAPE 5K and 12.5K have a completely different shape structure, which is more difficult. Table 5 shows average geodesic errors on different settings. Compared with FAUST dataset, OSD has better results on SCAPE, but it seems that overfitting is more severe at resolutions. The descriptors of CGF still perform poorly on the mesh. Same as FAUST, SplineCNN and ChebyGCN overfit at one resolution. Geodesic-based method seems to be more stable with the change of resolution on FAUST, but the discrimination of descriptor is difficult to improve further. The setting of WEDS and MGCN still generates the most discriminative descriptor, while ensuring robustness to the change of resolution. Fig. 11 shows the curves for the CMC and CGE metric of different descriptors. Compared with SplineCNN and ChebyGCN, MGCN outputs better performing descriptors that are more robustness at different resolutions. This is also illustrated by Fig. 12, where we visualize the learned descriptors of SplineCNN and MGCN.

Fig. 13 shows a qualitative example of a pair of FAUST shapes and SCAPE shapes, where we use the learned descriptors to find correspondences and compare the quality of the obtained maps between different competitors. Another comparison of the performance of different methods with respect to different resolutions is given in Fig. 14. In both figures, we observe that our learned descriptor generated by MGCN leads to the best maps.

Comparison with deep functional maps. Deep functional maps (FMNet) [Litany et al. 2017] are a state-of-the-art shape matching framework. While the final output of the proposed framework is dense correspondences between a pair of shapes, the first part of the proposed network can be used to learn shape descriptors. Since this competing descriptor learning method is resolution dependent, we only compare on a dataset where all meshes have exactly the same number of vertices. We selected the original FAUST and SCAPE datasets for this comparison (see Table 6). Specifically, we compare to three settings of FMNet: (1) we apply nearest-neighbor search to the *learned descriptors* from the FMNet to obtain the pointwise correspondence (called "FMNet(desc) + NN" in Table 6). (2) the direct output of the FMNet (called "FMNet"). (3) we apply ZoomOut [Melzi et al. 2019], a recent state-of-the-art refinement technique to further refine the output of FMNet. Table 6 shows that our method outperforms FMNet in all three different settings. In future work, it will be interesting to compare to an extension of the deep functional maps pipeline that was developed concurrently to our work [Donati et al. 2020].

6.3.2 Learning descriptors on non-isometric shapes. We also test our learning framework on a small animal dataset SMAL to see how our pipeline can be generalized to shape pairs with larger deformation. As introduced previously, the SMAL dataset has only 50 shapes from different animal categories with different poses. Since this

Table 8. Average (direct/symmetry-aware) geodesic error computed on 6×5 shape pairs of FAUST. We vary the number of eigenfunctions (#basis) and set the parameters #scales and #samples to 96.

Descriptors (#96)	#Basis			
	50	100	200	300
HKS	500 / 382	**500 / 381**	501 / 382	501 / 382
WKS	341 / 152	**325 / 112**	358 / 139	339 / 118
DTEP	374 / 208	302 / **95**	**295** / 100	366 / 146
WEDS	352 / 110	266 / 83	263 / **68**	**256** / 77

Table 9. Average (direct/symmetry-aware) geodesic error computed on 6×5 shape pairs of FAUST. We vary the number of eigenfunctions (#basis) and set the parameters #scales and #samples to 128.

Descriptors (#128)	#Basis			
	50	100	200	300
HKS	503 / 383	**502 / 383**	501 / 382	502 / 384
WKS	334 / 134	**322 / 112**	336 / 126	355 / 115
DTEP	397 / 226	**307 / 96**	310 / 100	377 / 158
WEDS	364 / 126	262 / 85	257 / **65**	**248** / 69

number is too small for efficient network training, we use the as-rigid-as-possible [Sorkine and Alexa 2007] deformation method to efficiently extend the dataset to 116 shapes with more poses. We divide the 5K remeshed dataset into 100 shapes for training, 8 shapes for validation, and 8 shapes for testing. The test data set contains shape pairs from 8 different categories to ensure that we evaluate only on non-isometric pairs. We compute the geodesic error for 8×7 non-isometric pairs. We consider BIM [Kim et al. 2011] and SplineCNN [Fey et al. 2018] as a baseline. BIM is the current state of the art for non-isometric shape matching and can handle large deformations.

As before, we use nearest-neighbor matching to evaluate the performance of two learned descriptors. We use WEDS as input for MGCN training. For BIM, we use the complete pipeline as BIM does not compute descriptors. Table 7 reports the average direct error of 56 non-isometric SMAL shape pairs and we also show the performance of different methods on four example shape pairs with large distortion. It can be seen that MGCN outperforms BIM. Compared with SplineCNN, the performance of our descriptors is significantly better. Fig. 15 shows an example pair of a horse and a hippo. While the map computed by nearest-neighbor search has reasonable quality overall, there are also some outliers. An interesting direction of future work will be to combine our learned descriptors with a suitable optimization method to refine the initial matches. A drawback of our comparison is that we only used a small dataset. For future work, it would be important to collect larger datasets containing correspondences of non-isometric shape pairs.

6.4 Parameter Settings

Different parameters affect the performance of different descriptors. To compare fairly, we need to choose the best parameters for each descriptor. We focus on the spectral descriptors such as HKS, WKS, DTEP. For other descriptors, because of the variety of parameters, we use the parameters recommended by the authors. We analyze the following three parameters of these spectral descriptors: the number of eigenfunctions (#basis), the number of feature scales (#scales), and the number of dimensions we sample (#samples). The number of feature scales in WEDS is Num, and $\lceil \frac{\text{Num}}{32} \rceil$ denotes the number of wavelet scales we choose to collect the wavelet energy. For other spectral descriptors, the number of feature scales describes how often the time is sampled in the diffusion process. In WKS, the number of scales also encodes diffusion variance. Finally, all feature scales can be chosen to be used in a descriptor. Alternatively, feature scales can be subsampled uniformly. This subsampling is encoded by the parameter #samples. Therefore, the number of samples has to be smaller or equal to the number of feature scales. We test on 6 models resulting in 6×5 shape pairs to evaluate the parameters in the following experiments.

Table 10. Average (direct/symmetry-aware) geodesic error computed on 6×5 shape pairs of FAUST. We fix the number of eigenfunctions and vary the parameters #scales and #samples jointly.

#Scales	HKS	WKS	DTEP	WEDS*
16	508 / 390	469 / 290	305 / 105	290 / 100
32	499 / 379	539 / 371	311 / 93	269 / 85
64	**495 / 37**	350 / 142	305 / 91	257 / 78
96	500 / 381	**325 / 112**	**302** / 95	256 / 77
128	502 / 383	332 / 112	307 / 96	248 / 69
192	506 / 387	335 / 117	304 / **89**	247 / 68
256	510 / 390	337 / 120	305 / 89	245 / 65
512	-	-	-	244 / 62
768	-	-	-	245 / 62
1024	-	-	-	**244 / 59**

The number of eigenfunctions. We evaluate four different parameter settings for the number of eigenfunctions (#basis): 50, 100, 200, and 300. We do this for two settings of the parameter feature scales: 96 and 128. The number of samples is equal to the number of feature scales. Table 8 and Table 9 show the average geodesic error for these tests. HKS does equally well for all tested parameter settings. WKS works best with 100 eigenfunctions and the performance descreases with more eigenfunctions. DTEP works well with 100 and 200 eigenfunctions and the performance decreases with 300. WEDS seems to do well with 200 to 300 eigenfunctions. For the subsequent tests we pick HKS:100, WKS:100, DEP: 100, WEDS: 300 as the number of eigenfunctions. Generally, it can be seen that WEDS has better performance with more eigenfunctions, while other frequency-domain descriptors do not perform better with more eigenfunctions. One possible explanation is that when constructing the WEDS descriptor, the vertex information needs to be reconstructed by the basis functions, so more basis functions may lead to higher reconstruction accuracy.

The number of feature scales. We fix the number of eigenfunctions as described before and vary the number of feature scales. We use all feature scales so that the number of samples (#samples) is equal to the number of feature scales. Table 10 shows the average geodesic error for different scales. Based on this test, we choose HKS:64, WKS:96, DEP: 96, WEDS: 1024 for the number of scales. Our descriptor behaves differently from other descriptors: the higher the number of feature scales, the better the performance. We believe this is due to the fact that a larger number of feature scales leads to a better accuracy of the signal reconstruction and in turn to a better performance of our descriptor. This behaviour is consistent

Table 11. Average (direct/symmetry-aware) geodesic error computed on 6×5 shape pairs of FAUST. We fix the number of eigenfunctions and the number of feature scales and vary the parameter #samples.

Descriptors	#Samples			
	16	32	64	96
HKS	**486 / 367**	489 / 369	495 / 375	-
WKS	**323** / 111	325 / **109**	323 / 111	325 / 112
DTEP	309 / 98	305 / 96	304 / 95	**302 / 95**
WEDS	295 / 96	305 / 103	268 / 69	286 / 71
	128	256	512	1024
	250 / 66	253 / 59	250 / 62	**244 / 59**

with the previous test where we also observed that our descriptor performs better with a higher number of eigenfunctions.

The number of samples. For a fixed number of eigenfunctions and fixed number of feature scales, we vary the parameter #samples by uniformly subsampling the number of feature scales (#scales). we show the average geodesic error in the Table 11 and pick HKS:16(64), WKS:16(96), DEP: 96(96), WEDS: 128(1024). The first number is the parameter #samples and the number in brackets describes the parameter #scales. Even though, WEDS has the best performance with 1024 samples, we believe that 128 samples are a better trade-off between memory consumption and descriptor performance. Selecting 1024 samples would also greatly impact the runtime and neural network complexity.

Runtime. We also compared the computation time of the four spectral descriptors. The increase in performance is often accompanied by a decrease in computing efficiency. We choose the recommended parameters of these four spectral methods. And the computation cost is shown in Table 12. It can be seen that LPS and DTEP have improved performance compared to traditional spectral descriptors, but it takes a lot of time because of the computation of geodesic disks and optimization. WEDS can reduce time consumption while achieving the best performance by energy decomposition.

6.5 Limitations and Future Work

There are still important challenges left for future work. First, we suspect that our descriptor learning solution still overfits the training data too much, since there is not enough variability in current shape matching datasets. The most beneficial and practical future work would therefore be to collect larger training datasets with more variability. Second, we only use isotropic convolutional kernels in our networks. The anisotropic kernels would be an interesting direction for future work. Finally, our current implementation and evaluation is limited to meshes. It would be interesting to extend our work to point clouds and triangle soups in future work.

7 CONCLUSIONS

We proposed a novel framework for computing two types of shape descriptors. First, the non-learned descriptor WEDS is computed using graph wavelets to decompose the Dirichlet energy on a surface. Second, WEDS can be refined by our proposed MGCN to yield a learned descriptor. Our results show that the new descriptor WEDS is more discriminative than the current state-of-the-art non-learned descriptors and that the combination of WEDS and MGCN is better than the state-of-the-art learned descriptors. An important attribute of descriptors is the robustness to different surface discretizations.

Table 12. The computation time of different descriptors with respect to different resolutions.

Time(s)	#Resolution					
	5K	6890	8K	10K	12K	15K
HKS	0.21	0.23	0.27	031	0.36	0.41
WKS	0.27	0.31	0.33	0.36	0.41	0.47
LPS	185	464	655	1090	1482	2174
DTEP	1124	1467	1659	2025	2235	2639
WEDS	16.6	24.7	42.2	65.7	82.5	144.3

Our results demonstrate that MGCN generalizes significantly better to different surface discretizations than previous work.

In this paper, we proposed a descriptor learning framework including a new descriptor and a graph neural network. We first verified that WEDS is robust to resolution, rigid transformations, and is also a discriminative descriptor. Then MGCN was proposed to improve the discrimination of non-learned descriptors. Most importantly, MGCN can maintain robustness to the change of resolution while improving discrimination. Our framework was demonstrated by comparing to several recent state-of-the-art descriptors and neural networks on near-isometric shapes and non-isometric shapes. Our framework not only improved the performance of the descriptor, but also maintained the robustness to resolution while improving the discrimination of the descriptor.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their comments. This work was supported by the National Key R&D Program of China (2018YFB2100602 and 2019YFB2204104), the National Natural Science Foundation of China (61620106003, 61772523, 61802406 and 61972459), the Beijing Natural Science Foundation (L182059), the CCF-Tencent Open Research Fund, Shenzhen Basic Research Program (JCYJ20180507182222355), the Alibaba Group through Alibaba Innovative Research Program, and the KAUST OSR Award No. CRG-2017-3426.

REFERENCES

Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. 2005. SCAPE: shape completion and animation of people. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 24, 3 (2005), 408–416.

Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. 2011. The wave kernel signature: A quantum mechanical approach to shape analysis. In *IEEE International Conference on Computer Vision Workshops*. 1626–1633.

Silvia Biasotti, Andrea Cerri, Alex Bronstein, and Michael Bronstein. 2016. Recent trends, applications, and perspectives in 3D shape similarity assessment. *Computer Graphics Forum* 35, 6 (2016), 87–119.

Federica Bogo, Javier Romero, Matthew Loper, and Michael J Black. 2014. FAUST: Dataset and evaluation for 3D mesh registration. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 3794–3801.

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. *Computer Graphics Forum* 34, 5 (2015), 13–23.

Davide Boscaini, Jonathan Masci, Emanuele Rodolà, Michael M Bronstein, and Daniel Cremers. 2016. Anisotropic diffusion descriptors. *Computer Graphics Forum* 35, 2 (2016), 431–441.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.

Michael M Bronstein and Iasonas Kokkinos. 2010. Scale-invariant heat kernel signatures for non-rigid shape recognition. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 1704–1711.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *International Conference on Learning*

Representations.
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*. 3844–3852.
Haowen Deng, Tolga Birdal, and Slobodan Ilic. 2018. Ppfnet: Global context aware local features for robust 3D point matching. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 195–205.
Nicolas Donati, Abhishek Sharma, and Maks Ovsjanikov. 2020. Deep Geometric Functional Maps: Robust Feature Learning for Shape Correspondence. *arXiv preprint arXiv:2003.14286* (2020).
Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. 2018. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 869–877.
Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. 2004. Recognizing objects in range data using regional point descriptors. In *European Conference on Computer Vision (ECCV)*. 224–237.
Shunwang Gong, Lei Chen, Michael Bronstein, and Stefanos Zafeiriou. 2019. SpiralNet++: A Fast and Highly Efficient Mesh Convolution Operator. In *IEEE International Conference on Computer Vision Workshops*. 4141–4148.
Jianwei Guo, Hanyu Wang, Zhanglin Cheng, Xiaopeng Zhang, and Dong-Ming Yan. 2020. Learning local shape descriptors for computing non-rigid dense correspondence. *Computational Visual Media* 6, 1 (2020), 95–112.
Yulan Guo, Ferdous Sohel, Mohammed Bennamoun, Min Lu, and Jianwei Wan. 2013. Rotational projection statistics for 3D local surface description and object recognition. *Int. Journal of Computer Vision* 105, 1 (2013), 63–86.
Oshri Halimi, Or Litany, Emanuele Rodola, Alex M Bronstein, and Ron Kimmel. 2019. Unsupervised learning of dense shape correspondence. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 4370–4379.
David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis* 30, 2 (2011), 129–150.
Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. 2019. MeshCNN: a network with an edge. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 38, 4 (2019), 90:1–90:12.
Haibin Huang, Evangelos Kalogerakis, Siddhartha Chaudhuri, Duygu Ceylan, Vladimir G Kim, and Ersin Yumer. 2018. Learning Local Shape Descriptors from Part Correspondences with Multiview Convolutional Networks. *ACM Transactions on Graphics* 37, 1 (2018), 6:1–6:14.
A. E. Johnson and M. Hebert. 1999. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 21, 5 (1999), 433–449.
Marc Khoury, Qian-Yi Zhou, and Vladlen Koltun. 2017. Learning compact geometric features. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 153–161.
Vladimir G Kim, Yaron Lipman, and Thomas Funkhouser. 2011. Blended intrinsic maps. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 30, 4 (2011), 79:1–79:12.
Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR Posters*.
Isaak Lim, Alexander Dielen, Marcel Campen, and Leif Kobbelt. 2018. A Simple Approach to Intrinsic Correspondence Learning on Unstructured 3D Meshes. In *European Conference on Computer Vision Workshops* . 349–362.
Or Litany, Tal Remez, Emanuele Rodolà, Alexander M Bronstein, and Michael M Bronstein. 2017. Deep Functional Maps: Structured Prediction for Dense Shape Correspondence.. In *IEEE International Conference on Computer Vision (ICCV)*. 5660–5668.
Roee Litman and Alexander M Bronstein. 2014. Learning spectral descriptors for deformable shape correspondence. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 36, 1 (2014), 171–180.
Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. 2015. Geodesic convolutional neural networks on riemannian manifolds. In *IEEE International Conference on Computer Vision Workshops*. 37–45.
Majid Masoumi and A Ben Hamza. 2017. Spectral shape classification: A deep learning approach. *Journal of Visual Communication and Image Representation* 43 (2017), 198–211.
Simone Melzi, Maks Ovsjanikov, Giorgio Roffo, Marco Cristani, and Umberto Castellani. 2018. Discrete time evolution process descriptor for shape analysis and matching. *ACM Transactions on Graphics* 37, 1 (2018), 4:1–4:18.
Simone Melzi, Jing Ren, Emanuele Rodolà, Abhishek Sharma, Peter Wonka, and Maks Ovsjanikov. 2019. Zoomout: Spectral upsampling for efficient shape correspondence. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)* 38, 6 (2019), 155:1–155:14.
Anastasiia Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. 2017. Working hard to know your neighbor's margins: Local descriptor learning loss. In *Advances in Neural Information Processing Systems*. 4826–4837.
Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. 2012. Functional maps: a flexible representation of maps between shapes. *ACM Trans. on Graphics (Proc. SIGGRAPH)* 31, 4 (2012), 30:1–30:11.
Martin Reuter, Franz-Erich Wolter, and Niklas Peinecke. 2006. Laplace–Beltrami spectra as 'Shape-DNA' of surfaces and solids. *Computer-Aided Design* 38, 4 (2006), 342–366.
Raif M Rustamov. 2007. Laplace-Beltrami eigenfunctions for deformation invariant shape representation. In *Proc. of Symp. of Geometry Processing*. 225–233.
Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Proc. of Symp. of Geometry Processing*. 109–116.
Ljubiša Stanković and Ervin Sejdić. 2019. *Vertex-frequency analysis of graph signals*. Springer.
Jian Sun, Maks Ovsjanikov, and Leonidas J. Guibas. 2010. A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. *Computer Graphics Forum* 28, 5 (2010), 1383–1392.
Federico Tombari, Samuele Salti, and Luigi Di Stefano. 2010. Unique signatures of histograms for local surface description. In *European Conference on Computer Vision (ECCV)*. 356–369.
Nitika Verma, Edmond Boyer, and Jakob Verbeek. 2018. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 2598–2606.
Hanyu Wang, Jianwei Guo, Dong-Ming Yan, Weize Quan, and Xiaopeng Zhang. 2018. Learning 3D Keypoint Descriptors for Non-rigid Shape Matching. In *European Conference on Computer Vision (ECCV)*. 3–20.
Yiqun Wang, Jianwei Guo, Dong-Ming Yan, Kai Wang, and Xiaopeng Zhang. 2019a. A Robust Local Spectral Descriptor for Matching Non-Rigid Shapes With Incompatible Shape Structures. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 6231–6240.
Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. 2019b. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics* 38, 5 (2019), 146:1–146:12.
Yiqun Wang, Dong-Ming Yan, Xiaohan Liu, Chengcheng Tang, Jianwei Guo, Xiaopeng Zhang, and Peter Wonka. 2019c. Isotropic surface remeshing without large and small angles. *IEEE Trans. on Vis. and Comp. Graphics* 25, 7 (2019), 2430–2442.
Lingyu Wei, Qixing Huang, Duygu Ceylan, Etienne Vouga, and Hao Li. 2016. Dense human body correspondences using convolutional networks. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 1544–1553.
Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. 2019. Graph Wavelet Neural Network. In *ICLR (Poster)*.
Dong-Ming Yan, Guanbo Bao, Xiaopeng Zhang, and Peter Wonka. 2014. Low-resolution remeshing using the localized restricted Voronoi diagram. *IEEE Trans. on Vis. and Comp. Graphics* 20, 10 (2014), 1418–1427.
Andrei Zaharescu, Edmond Boyer, Kiran Varanasi, and Radu Horaud. 2009. Surface feature detection and description with applications to mesh matching. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 373–380.
Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 2017. 3DMatch: Learning local geometric descriptors from rgb-d reconstructions. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 199–208.
Silvia Zuffi, Angjoo Kanazawa, David Jacobs, and Michael J. Black. 2017. 3D Menagerie: Modeling the 3D Shape and Pose of Animals. In *IEEE Computer Vision and Pattern Recognition (CVPR)*. 6365–6373.

A APPENDIX

In this appendix, we give a proof for the reconstruction capabilities of our wavelet filter basis.

$$\sum_{m=0}^{K}\sum_{v} a(v)^{-1} W_{\mathbf{f}}(t_m, v)\, \boldsymbol{\psi}_{\boldsymbol{t_m},\boldsymbol{v}} \tag{27}$$

$$= \sum_{m=0}^{K}\sum_{v} a(v)^{-1} \sum_{j=0}^{N-1} a(v)\, g_{t_m}(\lambda_j)\, \sigma_j \phi_j(v) \sum_{j=0}^{N-1} a(v)\, g_{t_m}(\lambda_j)\, \phi_j(v)\, \boldsymbol{\phi}_j \tag{28}$$

$$= \sum_{m=0}^{K}\sum_{i=0}^{N-1} g_{t_m}(\lambda_i)\, \sigma_i \sum_{j=0}^{N-1} g_{t_m}(\lambda_j)\, \boldsymbol{\phi}_j \sum_{v} \phi_i(v)\, a(v)\, \phi_j(v) \tag{29}$$

$$= \sum_{m=0}^{K}\sum_{j=0}^{N-1} g_{t_m}^2(\lambda_j)\, \sigma_j \boldsymbol{\phi}_j \quad (i=j) \tag{30}$$

$$= \sum_{j=0}^{N-1} \sigma_j \boldsymbol{\phi}_j \quad (\textbf{If} \quad \sum_{m=0}^{K} g_{t_m}^2(\lambda_j) = 1 \quad \textbf{Parseval} \quad \textbf{frame}) \tag{31}$$

$$= \mathbf{f} \tag{32}$$