Computers & Graphics 132 (2025) 104320

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

FanNet: A mesh convolution operator for learning dense maps[☆]

Güneş Sucu[*], Sinan Kalkan, Yusuf Sahillioğlu

Department of Computer Engineering, METU, Ankara, Türkiye

ARTICLE INFO

Keywords:
3D shape correspondence
Geometric deep learning
Graph neural networks
Graph convolutional operators

ABSTRACT

In this paper, we introduce a fast, simple and novel mesh convolution operator for learning dense shape correspondences. Instead of calculating weights between nodes, we explicitly aggregate node features by serializing neighboring vertices in a fan-shaped order. Thereafter, we use a fully connected layer to encode vertex features combined with the local neighborhood information. Finally, we feed the resulting features into the multi-resolution functional maps module to acquire the final maps. We demonstrate that our method works well in both supervised and unsupervised settings, and can be applied to isometric shapes with arbitrary triangulation and resolution. We evaluate the proposed method on two widely-used benchmark datasets, FAUST and SCAPE. Our results show that FanNet runs significantly faster and provides on-par or better performance than the related state-of-the-art shape correspondence methods.

1. Introduction

Geometric deep learning [1,2] has made unprecedented impact on a wide spectrum of real-world problems, such as chemistry and medicine [3–5], biology [6,7], recommender systems [8,9] and traffic forecasting [10,11]. Geometric deep learning owes this success to its ability to capture information in non-Euclidean data structures such as graphs and meshes.

The non-Euclidean nature of 3D shapes brings in many challenges for deep learning and two different approaches have been proposed over the years for 3D shape analysis: Extrinsic and intrinsic deep learning methods [2]. Extrinsic deep learning methods use vanilla convolutional neural networks (CNNs) [12,13], which treat geometric data as Euclidean. Although these first approaches provided promising results, the shape descriptors used in these methods are not suitable for shapes with non-Euclidean intrinsic structures because they use extrinsic features that are dependent on Euclidean transformations. This limitation is addressed by intrinsic methods [14–17] which modify the representations in CNN layers to extract intrinsic features of 3D shapes. It has been shown that such representations are better for capturing shape deformations [18,19]. Especially DiffusionNet [20] and its extensions [21] can efficiently learn intrinsic shape features by mimicking the heat diffusion.

In this paper, we define a fast and simple intrinsic mesh convolution operator, called FanNet, for the shape correspondence problem that has many computer graphics applications ranging from morphing to texture transfer. We generate fans around each vertex as a set of spokes (see Fig. 1), taking inspiration from geodesic fans [22,23] that have provided promising results in capturing curvature information in local surfaces. In our work, we apply FanNet to the task of dense shape correspondence between isometrically deformed shapes, i.e., shapes that differ by isometric transformations consisting of rotation, translation, and bending. The features obtained from our FanNet are fed to the multi-resolution functional map mechanism [21] to get the final mappings.

In order to show our method's versatility, we also apply FanNet to the problem of shape classification. As in the shape correspondence, we use it to extract features and then we train a FanNet-based network to minimize a Cross Entropy loss.

Main Contributions. The main contributions of our study can be listed as follows:

(1) We introduce FanNet as a novel and simple intrinsic mesh convolution operator. To the best of our knowledge, we are the first to adapt the spoke idea to geometric deep learning and shape correspondence problem.
(2) We demonstrate that FanNet provides on-par or better performance than the state-of-the-art performance for finding correct dense correspondences on two widely-used benchmark datasets, FAUST [24] and SCAPE [25].
(3) We show that FanNet provides competitive performance on other surface learning tasks such as classification.

[☆] This article was recommended for publication by Marco Attene.
This article has been certified as Replicable by the Graphics Replicability Stamp Initiative: http://www.replicabilitystamp.org.
[*] Corresponding author.
E-mail addresses: gsucu@metu.edu.tr (G. Sucu), skalkan@metu.edu.tr (S. Kalkan), ysahilli@metu.edu.tr (Y. Sahillioğlu).

https://doi.org/10.1016/j.cag.2025.104320
Received 7 February 2025; Received in revised form 17 July 2025; Accepted 21 July 2025
Available online 16 August 2025

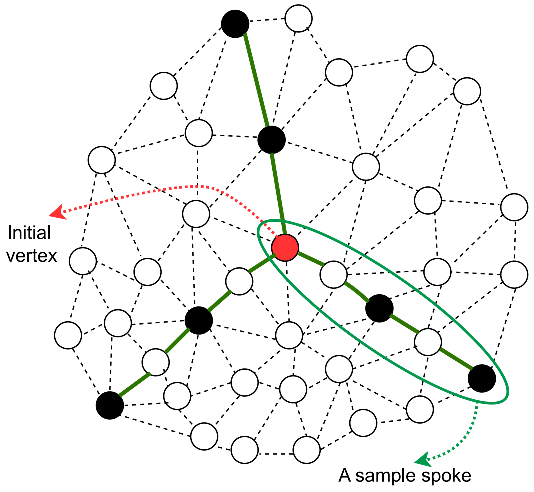


Fig. 1. FanNet uses fan-based ordering to serialize a neighborhood. The figure illustrates an example fan traversal on a triangle mesh. In order to avoid sensitivity to triangulation, it chooses the nodes corresponding to the specific radius values. Black nodes are included in the neighborhood of red node.

(4) We report that FanNet runs significantly faster than the state-of-the-art feature extractor DiffusionNet [20], regardless of the task.

2. Related work

2.1. Geometric deep learning

Geometric deep learning [1,2] is a family of methods that extend vanilla deep networks to process non-Euclidean data, such as graphs and manifolds. It has been widely used on tasks ranging from node classification [26,27], graph classification [28], graph generation [29] to physical property prediction [30] and mesh deformation prediction [31].

In vanilla convolution, a fixed-size filter matrix (kernel), W, slides over an input image to transform an input matrix to a feature map based on the values in the kernel:

$$h_v^{(l+1)} = \sigma\left(\sum_{u\in N(v)} W_u^{(l)} h_u^{(l)} + B_v^{(l)} h_v^{(l)}\right), \qquad \forall l \in \{0,\dots,L-1\} \tag{1}$$

where h_i denotes the representation (embedding) at position i and the superscript $(\cdot)^{(l)}$ denotes the layer index, $W^{(l)}$ denotes the filter for the lth layer, $B_v^{(l)}$ is the weight for position v, and $\sigma(\cdot)$ is a non-linear activation function, e.g., ReLU [32]. $N(v)$ is the neighborhood of v formed as a grid; e.g. a grid of $(v-F,..,v-1,v,v+1,\dots,v+F)\times(v-F,..,v-1,v,v+1,\dots,v+F)$ with $(F+1)\times(F+1)$ elements for 2D data, for some integer F. The initial, 0-th layer, embedding at position v is the input: $h_v^0 = x_v$; and the embedding after the last layer is denoted by $z_v = h_v^{(L-1)}$.

The regular grid neighborhood structure in Eq. (1) is not suitable for non-Euclidean data such as meshes since neighborhoods are not regular as required in Eq. (1). In a graph neural network, this can be addressed by extending Eq. (1) to allow irregular neighborhood structures present in graphs. As such, convolution with such non-Euclidean structures, i.e., "graph convolution", allows nodes to aggregate information from their neighbors using neural networks:

$$h_v^{(l+1)} = \sigma\left(W^{(l)} \sum_{u\in N(v)} \frac{h_u^{(l)}}{|N(v)|} + B_v^{(l)} h_v^{(l)}\right), \qquad \forall l \in \{0,\dots,L-1\} \tag{2}$$

where u and v denote the nodes of the graph; and $N(v)$ defines the neighborhood of v in the graph. The aggregator function in this equation is the most common message aggregator in the literature, wherein each node computes a weighted average or sum of its incoming messages [33]. Sum and max aggregators can also be used.

CNNs can be viewed as special cases of GNNs with fixed number of neighbors and ordering among neighbors. Moreover, the kernel size for a CNN is pre-defined whereas a GNN can process graphs which have nodes of different degrees. Furthermore, a CNN is not permutation equivariant in that changing the order of variables in a neighborhood will result in different values.

2.2. Shape correspondence

We can analyze shape correspondence methods in three groups (see [34] for a recent review): registration-based, similarity-based and learning-based methods.

2.2.1. Registration-based methods

In registration-based solutions, two shapes are aligned by a registration process. This can be carried out by either deforming one shape towards the other shape [35–38] or by parameterization to a common domain such as extended complex plane [39], quotient manifold [40,41] and sphere [42]. After registration, correspondence is obtained by analyzing the closeness between aligned shape components. Deformation-based approaches are generally computationally intense and sensitive to the source shape selection, whereas parameterization approaches mostly require topology restrictions. Our learning-based approach, on the other hand, is free of these shortcomings.

2.2.2. Similarity-based methods

In similarity-based solutions, the original geometry of the input shapes is not modified. Instead, some shape descriptors are computed on nodes (pointwise) or between nodes (pairwise). Correspondence is directly obtained by examining the point-wise or pairwise similarity of shape elements. Since this approach avoids any parameterization and deformation process, it is more commonly used than the registration-based approaches. A prominent example of this category is the functional map concept introduced by Ovsjanikov et al. [43]. Instead of using traditional point-to-point correspondences, they match real-valued functions of the shape surfaces. This idea has been extended by many follow-up refinement algorithms including [44–46]. Traditional point-to-point correspondence methods, on the other hand, directly work on pointwise and pairwise features over surfaces through different optimization techniques including Expectation–Maximization [47], dynamic programming [48], quadratic assignment [49], transport plans [50], genetic algorithms [51,52], and voting [53]. Due to lack of context and prior, this line of works may suffer from accuracy issues especially on challenging scenarios. Being a learning-based approach, our method can utilize such information.

2.2.3. Learning-based methods

Learning-based shape correspondence methods train a model on a shape-correspondence training set and this trained model is used on the testing set in order to predict correct correspondences.

Conventional machine learning based methods first extract hand-crafted shape descriptors requiring prior knowledge from experts and then find correspondences from those descriptors using different machine learning methods. For example, van Kaick et al. [54] find part correspondences using the semantic information of the shapes. Litman and Bronstein [55] propose a method to learn spectral descriptors for deformable shape correspondence. Kim et al. [56] propose an algorithm that learns a probabilistic model of part-based template variations. In another approach, Rodola et al. [57] use random forests to produce dense shape correspondences. Corman et al. [58], on the other hand, try to extract the optimal set of descriptors using reference pairs with known mappings.

The rise of deep learning has led to promising applications in 3D shape analysis. For example, Wei et al. [59] used a vanilla CNN

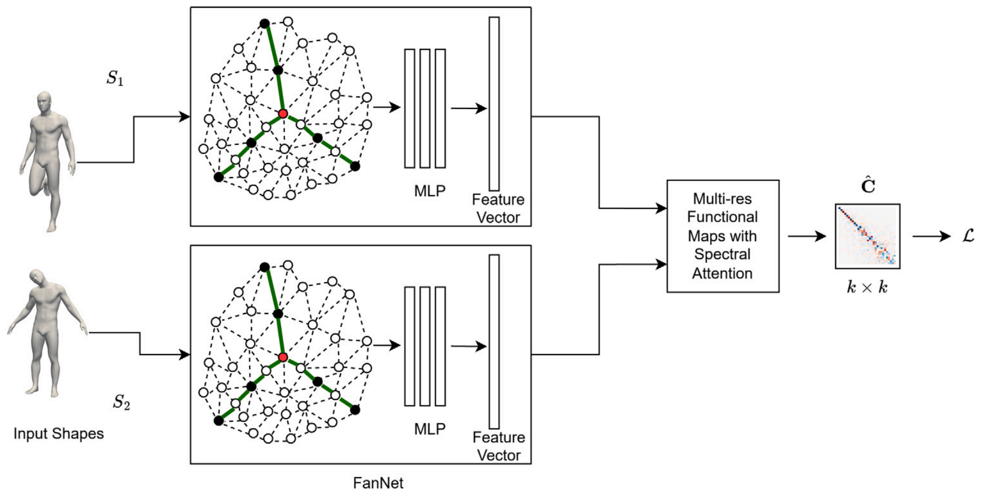


Fig. 2. Illustration of our shape-matching pipeline, which utilizes the features extracted by the FanNet architecture with multi-resolution functional maps with spectral attention framework. Our network takes as input a pair of shapes, S_1 and S_2 in a Siamese manner. It then extracts features to be passed onto the differentiable multi-resolution functional maps with spectral attention framework to get the resulting functional map $\hat{\mathbf{C}}$ of size $k \times k$ where k denotes the size of the functional map. We impose the loss $\mathcal{L}$ during training.

architecture to find dense shape correspondences. Litany et al. [60] proposed deep functional maps to predict dense shape correspondences. Halimi et al. [61] utilized an unsupervised learning scheme for dense shape correspondence. Despite their promising results, these studies, however, utilized regular grid-like connections or receptive-fields provided by conventional deep networks, and therefore, they were not suitable for capturing non-Euclidean data.

Considering the limitations of grid-like processing in conventional deep networks, methods considering the non-Euclidean nature of 3D shapes have been proposed. These so-called geometric deep learning methods try to generalize deep learning methods to non-Euclidean data structures such as graphs and meshes. For example, Masci et al. [14] introduce the geodesic CNN architecture which is an intrinsic counterpart of vanilla CNN. In another work, Boscaini et al. [15] propose anisotropic CNN, an improved version of geodesic CNN. Monti et al. [18] further generalize these concepts and propose mixture model CNN architecture to extract intrinsic features from graphs and manifolds. In another study, Verma et al. [62] present a graph convolution operator to determine the correspondences between filter weights and the nodes in a local neighborhood. Lim et al. [63] define the spiral neighborhood concept to learn the intrinsic geometric features. Gong et al. [64] also use the power of spiral serialization. But differently from [63], they create the spiral sequences just once. Li et al. [65] propose an anisotropic convolution operator for manifolds. Wiersma et al. [66] introduce a CNN architecture for surfaces which uses rotation-equivariant features. In fact, our method which will be discussed in Section 4 can be considered as a discrete mesh-walking version of the continuous logarithmic map presented in [66]. In another work, Yang et al. [67] introduce a graph convolution kernel to learn intrinsic features directly from raw meshes. Sharp et al. [20] mimic the heat diffusion and use it to extract features from the input shapes and keep the spectral resolution of functional maps constant. Li et al. [21] also use the power of diffusion. But differently from [20], they apply an attentive spectral resolution mechanism when calculating the functional maps.

3. Background

3.1. Deep multi-resolution functional maps

Our approach uses the functional maps framework, which was initially presented in [43] and has since been expanded in various works, including [68] and others (for more information, refer to [69,70]). This general approach is based on encoded maps between shapes utilizing a condensed basis representation which makes map optimization more streamlined and linear. Furthermore, this framework permits the depiction of natural restrictions, such as near-isometry or bijectivity, as linear-algebraic regularization.

A limitation of this framework is the estimation of descriptor functions, which are crucial for calculating functional maps. Previous approaches have used shape features, particularly multi-scale diffusion-based descriptors like Heat Kernel Signature (HKS) [71] and Wave Kernel Signature (WKS) [72], to address this challenge.

3.1.1. Deep functional maps

The Functional Maps approach suffers from the dependency on descriptor functions. The Deep Functional Map Pipeline [69] focuses on utilizing a feature extractor network with learnable parameters that extracts a set of descriptors to generate the Functional Map.

We are given a pair of meshes S_1 and S_2 with n_1 and n_2 vertices, respectively. The objective is to calculate a functional map between S_1 and S_2. The full pipeline can be explained with these four steps:

- For each mesh, the first k eigenfunctions of the Laplace–Beltrami operator [73] are computed. These are stored as matrices, namely $\boldsymbol{\Phi}_1 \in \mathbb{R}^{n_1 \times k}$ and $\boldsymbol{\Phi}_2 \in \mathbb{R}^{n_2 \times k}$.
- As the second step, features are extracted using a feature extractor network [20,66] and a function $\mathcal{F}_\theta$ is obtained with learnable parameters θ. For the meshes S_1 and S_2, $\mathcal{F}_\theta(S_1) \in \mathbb{R}^{n_1 \times d}$ and $\mathcal{F}_\theta(S_2) \in \mathbb{R}^{n_2 \times d}$ are calculated, where d is the number of features. Then they are projected onto the corresponding eigenfunctions $\boldsymbol{\Phi}_1$ and $\boldsymbol{\Phi}_2$ and the coefficients are stored as matrices $\mathbf{A_1}, \mathbf{A_2} \in \mathbb{R}^{k \times d}$.
- In order to find the ideal functional map $\mathbf{C} \in \mathbb{R}^{k \times k}$, the following optimization problem is solved:

$$\mathbf{C} \leftarrow \underset{\mathbf{C}'}{\arg\min} \, \|\mathbf{C}'\mathbf{A_1} - \mathbf{A_2}\|^2 + \alpha \|\mathbf{C}'\boldsymbol{\Delta}_1 - \boldsymbol{\Delta}_2\mathbf{C}'\|^2, \quad (3)$$

  where the first term supports preservation of the features, and the last term is for regularization of the map. $\boldsymbol{\Delta}_1$ and $\boldsymbol{\Delta}_2$ are the diagonal matrices of eigenvalues of Laplacian operators.
- Finally, the map $\mathbf{C}$ can be converted to a point-to-point map using nearest neighbor search between the aligned spectral embeddings $\boldsymbol{\Phi}_1\mathbf{C}^\top$ and $\boldsymbol{\Phi}_2$. Also post-refinement [45,46] can be applied during this process.

To train the feature extractor network $\mathcal{F}_\theta$, a set of training and test shape pairs are defined. The functional map between the two shapes can be obtained in a differentiable manner using the closed form solution of Eq. (3), making it suitable for training the extractor to get better descriptor functions. During training, the network seeks to reduce the Frobenius Loss defined on the Functional Map $\mathbf{C}$ estimated using extracted descriptor functions $\mathcal{F}_\theta(S_1)$ and $\mathcal{F}_\theta(S_2)$. As highlighted in [21], the Functional Map size $k \times k$ is a critical hyperparameter that can significantly impact the quality of the Functional Map.

3.1.2. *Multi-resolution functional maps*

The accuracy of existing functional map learning methods heavily depends on selecting the spectral resolution hyperparameter. If this parameter is not chosen carefully, it can result in overfitting or reduced accuracy. Multi-resolution Functional Maps with Spectral Attention Framework [21] proposes a framework to alleviate spectral resolution tuning by training the framework to adapt the spectral resolution depending on the given shape input. The framework calculates a series of n functional maps $C = \{\mathbf{C}^i\}_{i=1}^n$ at multiple resolutions where each map $\mathbf{C}^i$ is calculated using Eq. (3) and is of size $k^i \times k^i$. For $\mathbf{C}^{i+1}$, $k^{i+1} = k^i + \tau$, which means Φ_1 and Φ_2 have τ more eigenfunctions.

After this step, the framework calculates a soft attention weight $\alpha^i \in [0,1]$ for each map $\mathbf{C}^i$ to indicate its contribution to the final map. To this end, first let Φ_1^i and Φ_2^i be the matrices of the first k^i eigenfunctions on S_1 and S_2 as defined in Section 3.1.1. The spectral alignment residual r_q^i for a point $q \in S_2$ is defined as follows:

$$r_q^i = \min_{p\in S_1} \delta_{qp}^i, \qquad \delta_{qp}^i = \|(\Phi_2^i[q])^\top - \mathbf{C}^i(\Phi_1^i[p])^\top\|_2, \tag{4}$$

where the pth row of the matrix Φ_1^i and the qth row of the matrix Φ_2^i are represented by $\Phi_1^i[p]$ and $\Phi_2^i[q]$, respectively. Across C, the feature vector $\mathbf{r}_q = \left[\cdots, r_q^i/\sqrt{k^i}, r_q^{i+1}/\sqrt{k^{i+1}}, \cdots\right]^\top$ for the point q indicates the confidence of each map $\mathbf{C}^i$ at the point q. Then, these residuals $\{\mathbf{r}_q\}_{q\in S_2}$ are fed into the spectral attention network $\mathcal{A}_\Psi$ with learnable parameters Ψ which is built upon PointNet [17]. This network provides the spectral attention weights α^i where $\sum_{i=1}^n \alpha^i = 1$. After that, a differential spectral upsampling procedure [45] is applied to make all the maps of the same size. As the final step, all the maps are assembled into a final map by following linear combination:

$$\hat{\mathbf{C}} = \sum_{i=1}^n \alpha^i \mathbf{C}^i. \tag{5}$$

4. Our method: FanNet

In this section, we introduce our FanNet in detail. As illustrated in Fig. 2, our approach to dense shape correspondence has two main components: Extracting features using FanNet and calculating functional maps using AttentiveFmaps [21].

With FanNet, we introduce a novel fan-like neighborhood definition for graph neural networks. Our fan-like neighborhood definition is based on spokes, i.e., rays, that start at a center vertex and spread along paths, which are approximately equally distant with respect to each other. We take inspiration from geodesic fans [22,23], which form spokes starting from the center vertex and going along the surface. To the best of our knowledge, we are the first to adapt spokes to geometric deep learning and apply to the problem of shape correspondence.

4.1. *Problem definition*

Given a pair of meshes S_1 and S_2, we are interested in finding a one-to-one mapping $f : S_1 \to S_2$ that matches the vertices of the reference mesh S_1 with the vertices of the target mesh S_2. In our case, f is a parametric function with parameters θ, and we are interested in finding θ using geometric deep learning, namely graph convolution as defined in Section 2.1.

4.2. *Fan construction*

In our method, we serialize the neighboring vertices as the spokes of a fan (see Fig. 1). In particular, a fan consists of a set of n spokes, $s_1, \ldots, s_n$, where each spoke s_i contains varying number of vertices. Each spoke has a set of edges marching out across the surface from the starting vertex. In this way, we form a local neighborhood map around each vertex of the mesh.

Fan construction includes two steps: (1) Generating the spokes for a starting vertex. (2) Forming a fan from the spokes for each vertex. In the following, we explain these two steps in more detail — see also Algorithm 1.

4.2.1. *Spoke generation*

First we set k many radii (Fig. 4). When finding the paths, we sum the edge lengths for each spoke. When this sum reaches a radius value, the last vertex in the path is selected as the representative vertex for that radius. All spokes are of nearly equal length. This process continues until all radius values are matched (Algorithm 1). In this way, spokes will include varying number of vertices, which ensures the fan-based neighborhood to be less sensitive to mesh triangulation. That is, as shown in Fig. 4, fans can span more vertices in dense regions of the mesh and less vertices in sparser regions.

We start from the initial vertex v, select a first degree neighbor vertex u as the first vertex of the spoke ($s_i = \{u\}$) and grow the spoke s_i until it finds the representative vertex for each radius. While expanding the spoke beyond vertex u, at each step, we add a vertex that keeps the spoke as straight as possible (Fig. 3). To do this, while evaluating the neighbors of u, we first calculate difference vectors $x_j = v_j - u$, for each $v_j \in N(u)$, and choose as the next vertex for s_i, u', which is the straightest with respect to $x_0 = v - u$:

$$u' \leftarrow \operatorname*{arg\,min}_{v_j \in N(u)} \left| \arccos\left(\frac{x_j}{\|x_j\|} \cdot \frac{x_0}{\|x_0\|}\right) - \pi \right|, \tag{6}$$
$$s_i \leftarrow s_i \cup u'.$$

This process is repeated by setting $v \leftarrow u$ and $u \leftarrow u'$ until the spoke s_i contains all representative vertices for each radius.

Spoke generation can be considered as an edge-constrained instance of straightest geodesics [74].

Algorithm 1 Fan construction method as a Python-like pseudocode.

```
# Step 1: Spoke Generation -- Section 3.2.1
spokes = generate_spokes(mesh, v) # v: starting/initial vertex

# Step 2: Select/Form Fan from Generated Spokes -- Section 3.2.2
fan = [] # the fan around the starting vertex, v
radii = [R1, R2, ..., Rk] # k radii for k spheres
for spoke in spokes:
    cur_vertex = v
    for radius in radii:
        path_length = 0
        for next_vertex in spoke:
            path_length += edge_length(cur_vertex, next_vertex)
            if path_length >= radius:
                fan.append(next_vertex)
                break
            cur_vertex = next_vertex  # Continue along the path
```

4.2.2. *Forming fans from spokes*

We define a fan $F(v; n, k)$ centered at vertex v with n spokes each with k representative vertices as:

$$F(v; n, k) = \{v\} \cup \bigcup_{i=1}^n s_i(k), \tag{7}$$

where $F(\cdot;\cdot,\cdot)$ denotes the fan sequence of a node. For instance, $F(v_0; 3, 5)$ represents the fan centered at vertex v_0 with 3 spokes and each spoke having 5 representative vertices.

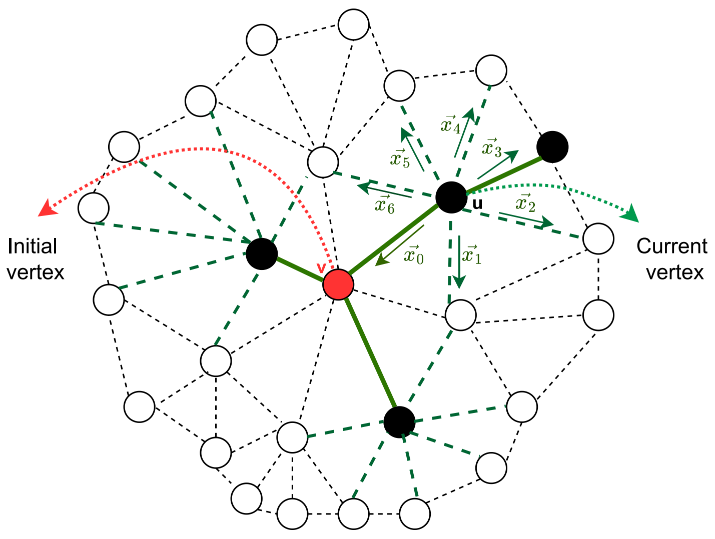


Fig. 3. Spoke generation: While growing a spoke from vertex u, we choose the vertex that keeps the spoke as straight as possible (Eq. (6)). For the example in the figure, this is the vertex along $\vec{x}_3$.

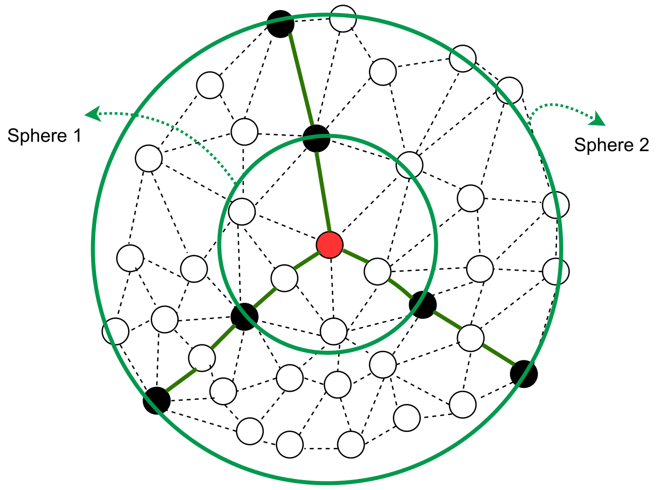


Fig. 4. Green circles represent the radius values. Black nodes are the representative vertices for each radius. In this fan sample, there are 3 spokes each of which starts with the red node and end with the second radius.

The number of spokes n and the values for k radii are hyper-parameters of FanNet that need to be tuned for a problem. The starting vertex v could have m neighbors (as illustrated in Fig. 4). If $m > n$, n spokes are randomly selected from m spokes at vertex v. If $m < n$, $n - m$ randomly selected spokes are repeated from m spokes to have n spokes in total. If $m = n$, all spokes are included.

4.3. Fan convolution

In FanNet, we take the vertices in a fan, serialize them in an order, and apply convolution on this serialized neighborhood:

$$h_v^{(l+1)} = \sigma\left(W^{(l)}\left(\bigoplus_{u\in F(v;n,k)} h_u^{(l)}\right) + B_v^{(l)} h_v^{(l)}\right), \qquad \forall l \in \{0,\dots,L-1\} \quad (8)$$

where $\bigoplus$ denotes the concatenation operator which is used to concatenate the embeddings for the vertices in the fan defined by $F(v;n,k)$ (Algorithm 2). $W^{(l)}$ and $B_v^{(l)}$ denote the weight and bias for the layer where $W^{(l)} \in \mathbb{R}^{d_{l+1}\times(n\cdot k\cdot d_l)}$ and $B_v^{(l)} \in \mathbb{R}^{d_{l+1}\times d_l}$.

Algorithm 2 FanNet forward-pass as a Pytorch-like pseudocode.

```
# x: input mesh, shape: num_nodes x num_features
# x type: FloatTensor
# num_features = 128 for HKS
# indices: neighboring indices for each node of x
# indices shape: num_nodes x seq_length, type: IntTensor
# seq_length: number of neighbors for each node
# in_channels = num_features

FC = nn.Linear(in_channels * seq_length, out_channels)
num_nodes = indices.size(0) # Obtain the number of vertices

# Concat vertices with their neighbors  -- Section 3.3
x = torch.index_select(x, 0, indices.view(-1))
x = x.view(num_nodes, -1)

# Obtain the features for x, shape: num_nodes x out_channels
x = FC(x)
x = nn.ELU(x) # Exponential Linear Unit as the nonlinearity
```

4.4. Overall architecture and objective

In our approach, we take meshes S_1 and S_2 and extract features $\mathcal{F}_\theta(S_1)$ and $\mathcal{F}_\theta(S_2)$ using our FanNet as the feature extractor network. Afterwards, using Eq. (3), we calculate multiple Functional Maps $\mathcal{C} = \{\mathbf{C}^i\}_{i=1}^n$ as explained in Section 3.1.2. Then, using AttentiveFmaps [21], we specify a soft attention weight $\alpha^i \in [0,1]$ for each map and merge these representations into a unified and consistent final map $\hat{\mathbf{C}}$ using Eq. (5).

Our network is trained in a supervised manner similar to AttentiveFmaps [21]. We formulate the loss on the final map output $\hat{\mathbf{C}}$, as shown in Fig. 2. The loss $\mathcal{L}$ for the final map $\hat{\mathbf{C}}$ is formulated as:

$$\mathcal{L} = L(\hat{\mathbf{C}}), \quad (9)$$

where $L(\mathbf{C})$ is the penalty for a given map $\mathbf{C}$. We define the map penalty $L(\mathbf{C}) = \|\mathbf{C} - \mathbf{C}^{gt}\|^2$, where the Frobenius norm is used. The ground-truth functional map $\mathbf{C}^{gt}$ has the same spectral resolution as $\mathbf{C}$ and can be obtained by projecting the point-to-point map onto the reduced spectral basis.

We also train our network in an unsupervised manner. In this case, we define the map penalty as $L(\mathbf{C}) = \|\mathbf{C}^\top\mathbf{C} - \mathbf{I}\|^2$ following [70].

Table 1
Vertex count ratios per degree for FAUST and SCAPE.

Degree	Datasets	
	FAUST	SCAPE
4	0.9%	0.7%
5	19.3%	20.1%
6	59.8%	58.8%
7	18.9%	19.2%
8	1.0%	1.0%
other	$< 1.0\%$	$< 1.0\%$

5. Experiments

The source code for the implementation is available at https://github.com/sucugunes/FanNet.

5.1. Experimental and implementation details

5.1.1. Datasets

We used two commonly-used benchmark datasets in our experiments.

FAUST [24]. FAUST contains triangulated meshes of 10 human subjects, with each subject in 10 different poses, and includes one-to-one correspondence between meshes of these different poses. We used this dataset in accordance with the train–test configuration of [21] for a fair comparison: The first 80 meshes were used in the training set, and the remaining 20 meshes were used in the test set.

SCAPE [25]. SCAPE contains 71 human meshes in different poses. The first 51 meshes were used in the training set, and the remaining 20 meshes were used in the test set (again in accordance with [21]). SCAPE is considered to be more challenging than FAUST since the shapes are less regular, and two shapes never share the same pose.

For both datasets, we use the more challenging remeshed versions rather than the original datasets.

5.1.2. FanNet and training details

The FAUST and SCAPE datasets have varying degrees for its vertices. Table 1 indicates that nearly 60% of vertices in both FAUST and SCAPE datasets have 6 neighbors. Therefore, taking into account the most common degree, we design FanNet with 6 spokes. FanNet performs repeated sampling if a vertex's degree is less than 6.

Our network architecture has a sequence of linear and graph convolutional layers: Lin(128) → FanConv(256) → FanConv(256) → FanConv(256) → Lin(256) → Lin(N), where the output channel size of each layer is denoted by the given numbers. N is the number of features produced by FanNet which is equal to 256. After the first linear layer and all the convolutional layers, exponential linear unit (ELU) [75] is used as non-linearity. Note that the kernel size depends on both the selected spoke length and the number of spokes per vertex.

To train on the FAUST and SCAPE datasets, we use HKS as input to the feature extractor backbone FanNet. The dimensionality of HKS is 128. We use ADAM optimizer with a learning rate which is set to 10^{-4}.

5.2. Experiment 1: Comparison with the baselines

We compare our method with existing supervised and unsupervised learning based non-rigid shape matching approaches including FMNet [60], 3D-CODED [76], HSN [66], ACSCNN [65], TransMatch [77], GeomFmaps [68], AttentiveFmaps [21], SURFM-Net [70], UnsupFMNet [61], NeuroMorph [78], DeepShells [79], WTFM [80], SSCDFM [81], URSSM [82] and SyncDiff [83]. In GeomFmaps, we use DiffusionNet as the feature extractor, same as in AttentiveFmaps.

Table 2 tabulates the shape correspondence performances of the methods in terms of mean geodesic error on unit-area shapes [39]. Best results are shown in bold and the second best results are shown as underlined. There are 12 train–test scenarios in total. In 10 of them our method gives the best result or the second best result compared to the baselines. For supervised training, in SCAPE-SCAPE and FAUST+SCAPE-SCAPE scenarios, it gives the state-of-the-art performance. For unsupervised training, in FAUST+SCAPE-SCAPE scenario, it gives the state-of-the-art performance. In FAUST-FAUST and FAUST+SCAPE-FAUST settings, it gives a competitive result compared to the state of the art. In FAUST-SCAPE and SCAPE-FAUST settings, it is still competitive but slightly worse than the best results.

5.3. Experiment 2: Ablation analysis

In this part, we first analyze the effect of the average pooling. Table 3 shows the shape correspondence performances obtained with FanNet+average pooling on different datasets. Here, after the FanNet layer an average pooling layer is added to the model. This pooling layer takes the average of neighboring nodes along with the central node and updates the value of central node with the average value found. We see that, this addition especially improves the SCAPE-FAUST scenario for FanNet.

We also analyze the compatibility of our feature extractor with different matching procedures. After the feature extraction process, this time we use the unsupervised matching procedure URSSM presented by Cao et al. [82]. Table 4 shows the comparison between FanNet and FanNet+URSSM. We see that our feature extractor works compatible with a matching procedure other than AttentiveFmaps.

5.4. Experiment 3: Qualitative analysis

In this part, we evaluate FanNet qualitatively with certain texture transfer results. Our goal here is to visually show how accurately the vertices on the reference shape are mapped to the vertices on the target shape.

As shown in Fig. 5, for the shapes in the first row, AttentiveFmaps fails for the belly and left arm but FanNet produces accurate mappings. For the shapes in the second row, AttentiveFmaps fails for the waist and belly but FanNet again produces accurate mappings.

In Fig. 6 we can see that FanNet again produces accurate mappings on different pairs of the FAUST testing set. Namely, we perform inter- and intra-matchings, where the former takes place between different people in different poses, whereas the latter is matching the different poses of the same person.

5.5. Experiment 4: Vertex degree analysis

In this part, we evaluate FanNet in terms of correct correspondences per vertex degree. Our goal here is to measure the effect of degree variance on shape correspondence performance. In Table 5 and Fig. 7 we can see that for zero geodesic error threshold, our method gives better results for degrees higher than 4. For the remaining geodesic threshold values, FanNet gives similar performances for vertices of different degrees. For example, FanNet finds correct correspondences for vertices having 6 neighbors (most common degree) on target shapes with an accuracy of 83.5% for the threshold value of 0.02. Threshold values can be interpreted geometrically by noticing that the maximum geodesic distance over the mesh, i.e., the diameter, is 1.

5.6. Experiment 5: Geodesic error analysis

In this part, we compare methods using geodesic error. The purpose of this analysis is to identify the vertices that the method incorrectly classifies and to measure the degree of inaccuracy in terms of geodesic distance. In other words, if a vertex on the reference shape is mapped to a wrong vertex on the target shape, we try to understand whether the wrong mapping is within tolerable distances. Fig. 8 shows that our method gives the best performance when the margin of error is 0. As the margin of error increases, error curve of FanNet still remains on the top. We use Princeton benchmark protocol [39] to analyze geodesic errors.

Table 2
Mean geodesic error (×100) on FAUST and SCAPE.

	Train on →	FAUST		SCAPE		FAUST+SCAPE	
	Test on →	FAUST	SCAPE	FAUST	SCAPE	FAUST	SCAPE
supervised	FMNet [60]	11.0	30.0	33.0	30.0	–	–
	3D-CODED [76]	2.5	31.0	33.0	31.0	–	–
	HSN [66]	3.3	25.4	16.7	3.5	–	–
	ACSCNN [65]	2.7	8.4	6.0	3.2	–	–
	TransMatch [77]	2.7	33.6	18.6	18.3	2.7	18.6
	TransMatch + Refine	1.7	30.4	15.5	12.0	1.6	11.7
	GeomFmaps [68]	2.6	3.6	2.9	2.9	2.6	2.9
	AttentiveFmaps [21]	**1.4**	**2.2**	**1.7**	1.8	**1.3**	1.8
	FanNet (**Ours**)	**1.4**	2.9	2.7	**1.6**	1.4	**1.6**
unsupervised	SURFMNet [70]	15.0	32.0	32.0	12.0	33.0	29.0
	UnsupFMNet [61]	10.0	29.0	22.0	16.0	11.0	13.0
	NeuroMorph [78]	8.5	28.5	18.2	29.9	9.1	27.3
	DeepShells [79]	1.7	5.4	2.7	2.5	**1.6**	2.4
	GeomFmaps [68]	3.5	4.8	4.0	4.3	3.5	4.4
	AttentiveFmaps [21]	1.9	2.6	2.2	2.2	1.9	2.3
	WTFM [80]	2.5	2.7	2.8	2.5	–	–
	SSCDFM [81]	1.7	2.6	2.0	2.2	–	–
	URSSM [82]	1.6	**2.2**	**1.6**	1.9	**1.6**	2.1
	SyncDiff [83]	**1.5**	–	–	**1.8**	–	–
	FanNet (**Ours**)	1.7	3.4	3.2	**1.8**	**1.6**	**1.9**

Table 3
Ablation analysis for FanNet with average pooling. Mean geodesic error (×100) on FAUST and SCAPE.

	Train on →	FAUST		SCAPE		FAUST+SCAPE	
	Test on →	FAUST	SCAPE	FAUST	SCAPE	FAUST	SCAPE
sup	FanNet	1.4	**2.9**	2.7	1.6	1.4	1.6
	FanNet + pooling	**1.3**	3.3	**2.4**	**1.5**	**1.3**	**1.5**
unsup	FanNet	**1.7**	**3.4**	3.2	1.8	**1.6**	1.9
	FanNet + pooling	1.8	3.7	**2.9**	1.8	1.7	1.9

Table 4
Ablation analysis for FanNet with different matching procedures. Compatibility analysis.

	Train on →	FAUST	SCAPE	FAUST+SCAPE	
	Test on →	FAUST	SCAPE	FAUST	SCAPE
unsup	FanNet	1.7	1.8	**1.6**	1.9
	FanNet + URSSM	**1.6**	**1.7**	**1.6**	**1.8**

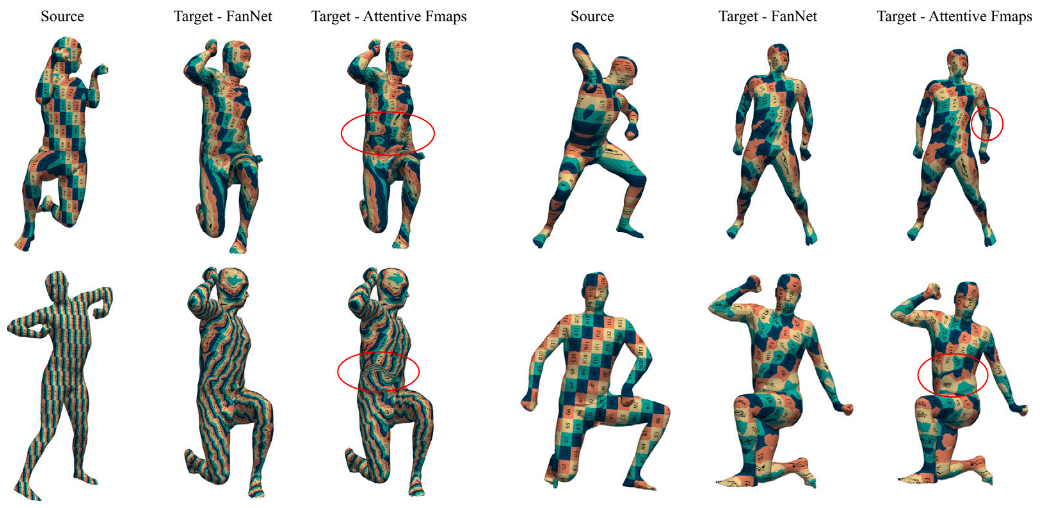


Fig. 5. Examples of **texture transfer** for FanNet and Attentive Fmaps. Train–test setting: unsupervised training on FAUST+SCAPE and testing on SCAPE. In the regions surrounded by red circles, FanNet produces more accurate mappings compared to AttentiveFmaps.

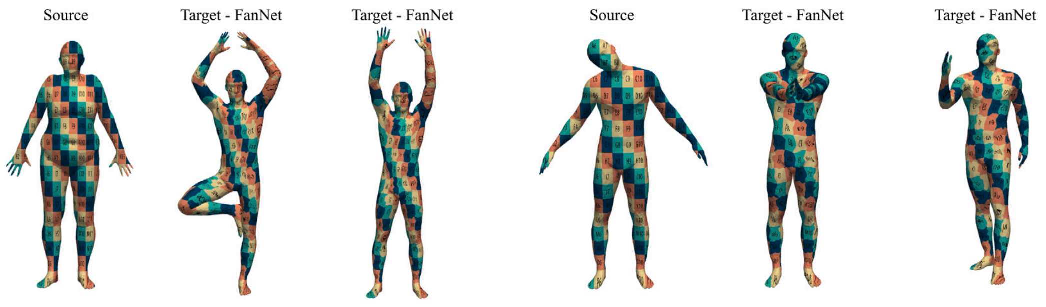


Fig. 6. Examples of **texture transfer** for FanNet. Train–test setting: supervised training on FAUST and testing on FAUST. FanNet can handle both inter (left) and intra (right) matchings.

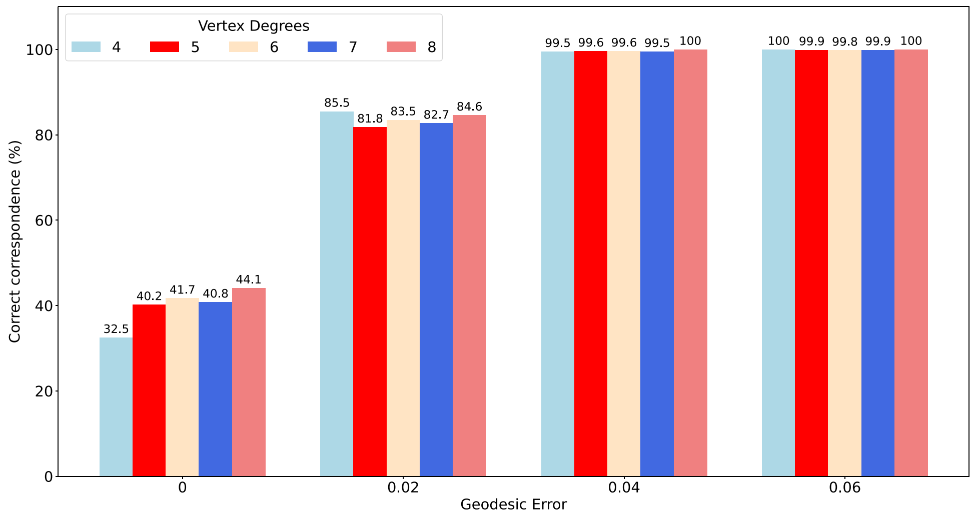


Fig. 7. Correct correspondence (%) vs Geodesic Error. Each color represents a different vertex degree.

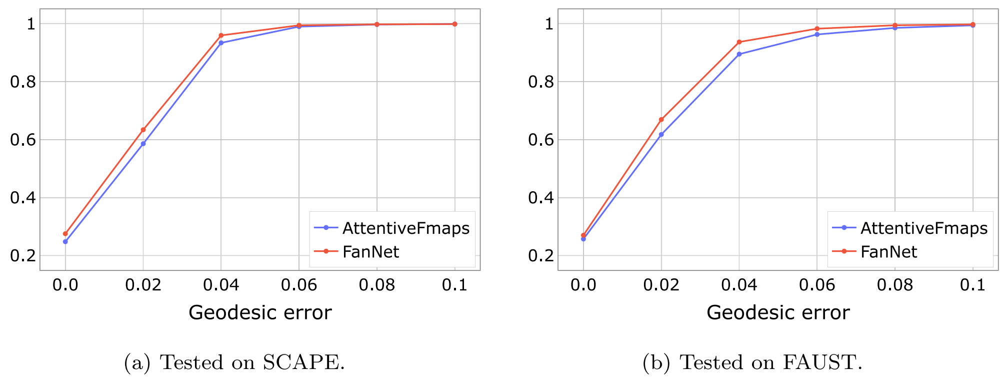


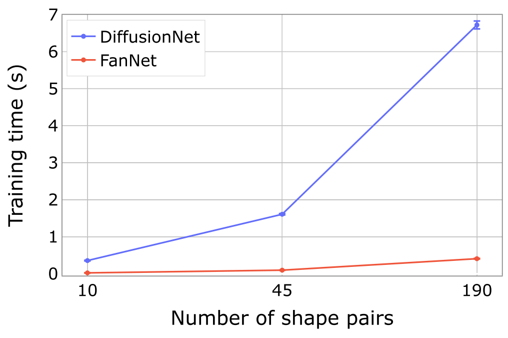
Fig. 8. A comparison of the methods in terms of geodesic error. (a) Train–test setting: supervised training on SCAPE and testing on SCAPE. (b) Train–test setting: unsupervised training on FAUST+SCAPE and testing on FAUST.

Table 5
Correct correspondence ratios per vertex degree for each geodesic error threshold. Train–test setting: supervised training on FAUST and testing on FAUST.

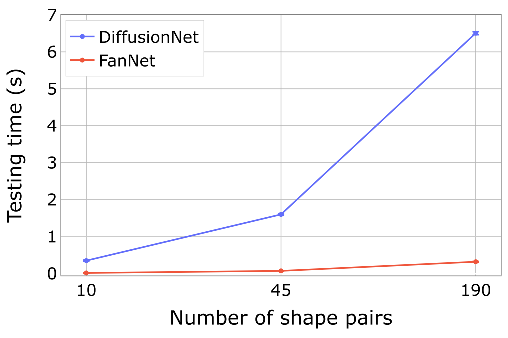
Vertex degree	Geodesic error threshold			
	0	0.02	0.04	0.06
4	32.5%	85.5%	99.5%	100.0%
5	40.2%	81.8%	99.6%	99.9%
6	41.7%	83.5%	99.6%	99.8%
7	40.8%	82.7%	99.5%	99.9%
8	44.1%	84.6%	100.0%	100.0%

5.7. Experiment 6: Performance on classification task

In this experiment, we use FanNet to classify meshes in the commonly-used SHREC-11 dataset [84], which has 30 classes with each class having 20 shapes. Aligned with the compared methods, we use 10 samples per class during training, repeat the experiment 10 times with random training splits and report the average performance over these 10 runs. We train a 256-width FanNet for hks features to minimize Cross-Entropy Loss. The classification results in Table 6 show that, when applied on the original meshes or the simplified versions, FanNet provides on-par performance with the state-of-the-art DiffusionNet.



(a) Training Time.



(b) Testing Time.

Fig. 9. Average running time per run for (a) training and (b) testing on SCAPE, calculated over 10, 45, and 190 shape pairs respectively over 5 runs.

Table 6
SHREC11 classification. Results marked by † are trained and tested on simplified meshes.

Method	Accuracy
GWCNN [85]	90.3%
MeshCNN† [86]	91.0%
HSN† [66]	96.1%
MeshWalker† [87]	97.1%
PD-MeshNet† [88]	99.1%
HodgeNet† [89]	94.7%
FC† [90]	99.2%
DiffusionNet [20] - xyz†	99.4%
DiffusionNet [20] - xyz	99.0%
DiffusionNet [20] - hks†	99.5%
DiffusionNet [20] - hks	**99.7%**
FanNet (**Ours**) - hks†	99.2%
FanNet (**Ours**) - hks	99.5%

Table 7
Average running time per run for training and testing on SCAPE, calculated over 10, 45, and 190 shape pairs respectively over 5 runs.

#Shape pairs	Training time (s)	Testing time (s)
DiffusionNet [20]		
10	0.357 ± 0.010	0.352 ± 0.014
45	1.609 ± 0.019	1.603 ± 0.015
190	6.724 ± 0.107	6.509 ± 0.041
FanNet (Ours)		
10	$\mathbf{0.025 \pm 0.004}$	$\mathbf{0.019 \pm 0.001}$
45	$\mathbf{0.098 \pm 0.004}$	$\mathbf{0.075 \pm 0.002}$
190	$\mathbf{0.408 \pm 0.009}$	$\mathbf{0.322 \pm 0.009}$

5.8. *Experiment 7: Running time analysis*

For these experiments, we use a machine equipped with an Nvidia Quadro RTX 4000 GPU to evaluate the running time of our method and the competitor methods. Note that precomputation is not included in the timing cost when evaluating the methods.

Shape Correspondence. In this experiment, we evaluate the running time of our method and DiffusionNet, the closest competitor method [20]. The analysis is performed over the SCAPE dataset both in training and testing (inference) mode. We report runtime measurements (listed in Table 7 and plotted in Fig. 9 to highlight the gap) for the feature extractor networks only (FanNet vs. DiffusionNet), across varying batch sizes (i.e., different numbers of shape pairs per run). The running-time results reflect the runtime behavior of the two models, and demonstrate that our method runs notably faster (×15.7 on training and ×20.1 on testing) than DiffusionNet during training and testing.

Shape Classification. In this experiment, we evaluate the running time of our method and DiffusionNet, the closest competitor method [20]. The analysis is performed over the simplified meshes of SHREC11 dataset both in training and testing (inference) mode. The running-time results in Table 8 suggest that our method runs much faster (×5.8 on training and ×6.8 on testing) than DiffusionNet during training and testing.

Table 8
Average running time per run for training and testing on SHREC11 (calculated over 5 runs each having 300 iterations).

	Training time (s)	Testing time (s)
DiffusionNet [20]	9.883 ± 0.072	2.808 ± 0.007
FanNet (Ours)	$\mathbf{1.693 \pm 0.027}$	$\mathbf{0.412 \pm 0.011}$

As we can see, running time difference is significant regardless of the task (correspondence or classification). FanNet's lightweight construction (no spectral convolutions, no diffusion) allows it to run significantly faster than DiffusionNet which relies on solving PDE-inspired diffusion layers.

6. Conclusion

We have presented a novel intrinsic mesh convolution operator, named FanNet, for learning dense maps. FanNet is a very simple, fast and easy to implement mesh convolution operator. FanNet defines a local neighborhood around each vertex as a Fan and uses this neighborhood to capture local geometric structure of a mesh. As summarized in Algorithm 2, FanNet takes the previously calculated indices in a fan-shaped order and use them to aggregate node features. Subsequently, it encodes vertex features combined within local neighborhood with the help of a fully connected layer.

We have shown that FanNet is a versatile method which works well in shape correspondence tasks with different matching procedures under both supervised and unsupervised settings. Furthermore, it can be applied to other mesh problems such as mesh classification.

We have demonstrated that thanks to its lightweight design FanNet runs notably faster than the prior art.

We have extensively evaluated our approach on two commonly-used shape correspondence benchmarks, FAUST and SCAPE. Our results have shown that FanNet provides better or on par performance compared to prior work.

The success of FanNet stems from the power of Fans spanning a large region of the surface when constructing the local neighborhood. This allows including meaningful information about the local surface structure around a vertex. Also FanNet computes spokes dynamically based on local surface characteristics. This enables Fans to span larger regions around low curvature surfaces.

6.1. Limitations and future work

While FanNet demonstrates strong performance in near-isometric shape correspondence, its effectiveness decreases in highly non-isometric scenarios. This is partly due to its reliance on intrinsic neighborhoods, which are less stable under large deformations. Additionally, FanNet's fan construction can be sensitive to mesh triangulation, since it depends on local connectivity. Despite this, FanNet remains efficient, simple to implement, and competitive with strong baselines. In future work, our work can be extended in different ways. For example, FanNet can be easily applied to the problem of matching partial shapes. Moreover, the neighborhood calculation can be extended to be more dynamic with respect to local surface structure and triangulation. For example, deformable convolution can be considered when constructing the local neighborhood around a vertex. This way Fans would be even less dependent on the triangulation. Moreover, geodesic fans can be considered during fan construction. This may allow Fans spread along the surface more smoothly. Another promising direction is to explore resolution-invariant designs that decouple network capacity from fan size, using attention-based aggregation or adaptive pooling to handle variable-length input sequences. Another avenue is to incorporate learned downsampling or hierarchical architectures to balance detail sensitivity with computational efficiency. On a related note for efficiency, our spectral method can also benefit from spectrum-preserving simplification techniques [91] to accelerate Laplacian-based functional maps computations by significantly reducing the complexity of the mesh representations while maintaining accuracy.

CRediT authorship contribution statement

Güneş Sucu: Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis. **Sinan Kalkan:** Writing – review & editing, Supervision, Methodology, Investigation. **Yusuf Sahillioğlu:** Writing – review & editing, Supervision, Project administration, Methodology.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yusuf Sahillioglu reports financial support provided by the Scientific and Technological Research Council of Turkey.

Acknowledgments

This work was supported by TÜBİTAK (The Scientific and Technological Research Council of Turkey) under project EEEAG-119E572. Some of the experiments were run in the computational environment provided by METU CENG Data Mining Research group. We also gratefully acknowledge the computational resources kindly provided by METU-ROMER, Center for Robotics and Artificial Intelligence.

Data availability

Data will be made available on request.

References

[1] Bronstein MM, Bruna J, Cohen T, Veličković P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. 2021, arXiv preprint arXiv:2104.13478.
[2] Cao W, Yan Z, He Z, He Z. A comprehensive survey on geometric deep learning. IEEE Access 2020;8:35929–49.
[3] Stokes JM, Yang K, Swanson K, Jin W, Cubillos-Ruiz A, Donghia NM, et al. A deep learning approach to antibiotic discovery. Cell 2020;180(4):688–702.
[4] Anderson B, Hy TS, Kondor R. Cormorant: Covariant molecular neural networks. Adv Neural Inf Process Syst 2019;32.
[5] Zitnik M, Agrawal M, Leskovec J. Modeling polypharmacy side effects with graph convolutional networks. Bioinformatics 2018;34(13):i457–66.
[6] Senior AW, Evans R, Jumper J, Kirkpatrick J, Sifre L, Green T, et al. Improved protein structure prediction using potentials from deep learning. Nature 2020;577(7792):706–10.
[7] Gainza P, Sverrisson F, Monti F, Rodola E, Boscaini D, Bronstein M, et al. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. Nature Methods 2020;17(2):184–92.
[8] Ying R, He R, Chen K, Eksombatchai P, Hamilton WL, Leskovec J. Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 2018, p. 974–83.
[9] Hao J, Zhao T, Li J, Dong XL, Faloutsos C, Sun Y, et al. P-companion: A principled framework for diversified complementary product recommendation. In: Proceedings of the 29th ACM international conference on information & knowledge management. 2020, p. 2517–24.
[10] Derrow-Pinion A, She J, Wong D, Lange O, Hester T, Perez L, et al. Eta prediction with graph neural networks in google maps. In: Proceedings of the 30th ACM international conference on information & knowledge management. 2021, p. 3767–76.
[11] Fang X, Huang J, Wang F, Zeng L, Liang H, Wang H. Constgat: Contextual spatial-temporal graph attention network for travel time estimation at baidu maps. In: Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining. 2020, p. 2697–705.
[12] Wu Z, Song S, Khosla A, Yu F, Zhang L, Tang X, et al. 3D shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2015, p. 1912–20.
[13] Su H, Maji S, Kalogerakis E, Learned-Miller E. Multi-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE international conference on computer vision. 2015, p. 945–53.
[14] Masci J, Boscaini D, Bronstein M, Vandergheynst P. Geodesic convolutional neural networks on riemannian manifolds. In: Proceedings of the IEEE international conference on computer vision workshops. 2015, p. 37–45.
[15] Boscaini D, Masci J, Rodolà E, Bronstein M. Learning shape correspondence with anisotropic convolutional neural networks. Adv Neural Inf Process Syst 2016;29.
[16] Boscaini D, Masci J, Melzi S, Bronstein MM, Castellani U, Vandergheynst P. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. Comput Graph Forum 2015;34(5):13–23.
[17] Qi CR, Su H, Mo K, Guibas LJ. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 652–60.
[18] Monti F, Boscaini D, Masci J, Rodola E, Svoboda J, Bronstein MM. Geometric deep learning on graphs and manifolds using mixture model cnns. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 5115–24.
[19] Fey M, Lenssen JE, Weichert F, Müller H. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 869–77.
[20] Sharp N, Attaiki S, Crane K, Ovsjanikov M. Diffusionnet: Discretization agnostic learning on surfaces. ACM Trans Graph 2022;41(3):1–16.
[21] Li L, Donati N, Ovsjanikov M. Learning multi-resolution functional maps with spectral attention for robust shape matching. Adv Neural Inf Process Syst 2022;35:29336–49.
[22] Zelinka S, Garland M. Similarity-based surface modelling using geodesic fans. In: Proceedings of the 2004 eurographics/ACM SIGGRAPH symposium on geometry processing. 2004, p. 204–13.
[23] Gatzke T, Grimm C, Garland M, Zelinka S. Curvature maps for local shape comparison. In: International conference on shape modeling and applications 2005. IEEE; 2005, p. 244–53.
[24] Bogo F, Romero J, Loper M, Black MJ. FAUST: Dataset and evaluation for 3D mesh registration. In: Proceedings IEEE conf. on computer vision and pattern recognition. 2014, p. 3794–801.
[25] Anguelov D, Srinivasan P, Koller D, Thrun S, Rodgers J, Davis J. Scape: shape completion and animation of people. In: ACM SIGGRAPH. 2005.
[26] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. 2016, arXiv preprint arXiv:1609.02907.
[27] Atwood J, Towsley D. Diffusion-convolutional neural networks. Adv Neural Inf Process Syst 2016;29.
[28] Lee JB, Rossi R, Kong X. Graph classification using structural attention. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 2018, p. 1666–74.

[29] Chen B, Sun L, Han X. Sequence-to-action: End-to-end semantic graph generation for semantic parsing. 2018, arXiv preprint arXiv:1809.00773.
[30] Coley CW, Barzilay R, Green WH, Jaakkola TS, Jensen KF. Convolutional embedding of attributed molecular graphs for physical property prediction. J Chem Inf Model 2017;57(8):1757–72.
[31] Kostrikov I, Jiang Z, Panozzo D, Zorin D, Bruna J. Surface networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 2540–8.
[32] Agarap AF. Deep learning using rectified linear units (relu). 2018, arXiv preprint arXiv:1803.08375.
[33] Boureau Y-L, Le Roux N, Bach F, Ponce J, LeCun Y. Ask the locals: multi-way local pooling for image recognition. In: 2011 international conference on computer vision. IEEE; 2011, p. 2651–8.
[34] Sahillioğlu Y. Recent advances in shape correspondence. Vis Comput 2020;36(8):1705–21.
[35] Sahillioğlu Y, Kavan L. Skuller: a volumetric shape registration algorithm for modeling skull deformities. Med Image Anal 2015;23(1):15–27.
[36] Cosmo L, Panine M, Rampini A, Ovsjanikov M, Bronstein MM, Rodolà E. Isospectralization, or how to hear shape, style, and correspondence. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 7529–38.
[37] Ezuz D, Heeren B, Azencot O, Rumpf M, Ben-Chen M. Elastic correspondence between triangle meshes. Comput Graph Forum 2019;38(2):121–34.
[38] Eisenberger M, Lahner Z, Cremers D. Smooth shells: Multi-scale shape registration with functional maps. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 12265–74.
[39] Kim VG, Lipman Y, Funkhouser T. Blended intrinsic maps. ACM Trans Graph 2011;30(4):1–12.
[40] Aigerman N, Lipman Y. Orbifold tutte embeddings. ACM Trans Graph 2015;34(6):1–190.
[41] Aigerman N, Lipman Y. Hyperbolic orbifold tutte embeddings. ACM Trans Graph 2016;35(6):1–217.
[42] Lee SC, Kazhdan M. Dense point-to-point correspondences between genus-zero shapes. Comput Graph Forum 2019;38(5):27–37.
[43] Ovsjanikov M, Ben-Chen M, Solomon J, Butscher A, Guibas L. Functional maps: a flexible representation of maps between shapes. ACM Trans Graph (ToG) 2012;31(4):1–11.
[44] Ren J, Poulenard A, Wonka P, Ovsjanikov M. Continuous and orientation-preserving correspondences via functional maps. ACM Trans Graph (ToG) 2018;37(6):1–16.
[45] Melzi S, Ren J, Rodola E, Sharma A, Wonka P, Ovsjanikov M. Zoomout: Spectral upsampling for efficient shape correspondence. ACM Trans Graph 2019;38(6):1–14.
[46] Pai G, Ren J, Melzi S, Wonka P, Ovsjanikov M. Fast sinkhorn filters: Using matrix scaling for non-rigid shape correspondence with functional maps. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021, p. 384–93.
[47] Sahillioğlu Y, Yemez Y. Minimum-distortion isometric shape correspondence using EM algorithm. IEEE Trans Pattern Anal Mach Intell 2012;34(11):2203–15.
[48] Sahillioğlu Y, Yemez Y. Multiple shape correspondence by dynamic programming. Comput Graph Forum 2014;33(7):121–30.
[49] Solomon J, Peyré G, Kim VG, Sra S. Entropic metric alignment for correspondence problems. ACM Trans Graph (ToG) 2016;35(4):1–13.
[50] Mandad M, Cohen-Steiner D, Kobbelt L, Alliez P, Desbrun M. Variance-minimizing transport plans for inter-surface mapping. ACM Trans Graph (ToG) 2017;36(4):1–14.
[51] Edelstein M, Ezuz D, Ben-Chen M. ENIGMA: Evolutionary Non-Isometric Geometry MAtching. ACM Trans Graph (ToG) 2020;39(4).
[52] Sahillioğlu Y. A genetic isometric shape correspondence algorithm with adaptive sampling. ACM Trans Graph 2018;37(5):1–14.
[53] Sahillioğlu Y, Horsman D. Augmented paths and reodesics for topologically-stable matching. ACM Trans Graph 2022;42(2):1–15.
[54] Van Kaick O, Tagliasacchi A, Sidi O, Zhang H, Cohen-Or D, Wolf L, et al. Prior knowledge for part correspondence. Comput Graph Forum 2011;30(2):553–62.
[55] Litman R, Bronstein AM. Learning spectral descriptors for deformable shape correspondence. IEEE Trans Pattern Anal Mach Intell 2013;36(1):171–80.
[56] Kim VG, Li W, Mitra NJ, Chaudhuri S, DiVerdi S, Funkhouser T. Learning part-based templates from large collections of 3D shapes. ACM Trans Graph 2013;32(4):1–12.
[57] Rodola E, Rota Bulo S, Windheuser T, Vestner M, Cremers D. Dense non-rigid shape correspondence using random forests. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2014, p. 4177–84.
[58] Corman E, Ovsjanikov M, Chambolle A. Supervised descriptor learning for non-rigid shape matching. In: European conference on computer vision. Springer; 2014, p. 283–98.
[59] Wei L, Huang Q, Ceylan D, Vouga E, Li H. Dense human body correspondences using convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, p. 1544–53.
[60] Litany O, Remez T, Rodola E, Bronstein A, Bronstein M. Deep functional maps: Structured prediction for dense shape correspondence. In: Proceedings of the IEEE international conference on computer vision. 2017, p. 5659–67.
[61] Halimi O, Litany O, Rodola E, Bronstein AM, Kimmel R. Unsupervised learning of dense shape correspondence. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019, p. 4370–9.
[62] Verma N, Boyer E, Verbeek J. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 2598–606.
[63] Lim I, Dielen A, Campen M, Kobbelt L. A simple approach to intrinsic correspondence learning on unstructured 3d meshes. In: Proceedings of the European conference on computer vision workshops. 2018.
[64] Gong S, Chen L, Bronstein M, Zafeiriou S. Spiralnet++: A fast and highly efficient mesh convolution operator. In: Proceedings of the IEEE international conference on computer vision workshops. 2019.
[65] Li Q, Liu S, Hu L, Liu X. Shape correspondence using anisotropic Chebyshev spectral CNNs. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 14658–67.
[66] Wiersma R, Eisemann E, Hildebrandt K. CNNs on surfaces using rotation-equivariant features. ACM Trans Graph (ToG) 2020;39(4):1–92.
[67] Yang Z, Litany O, Birdal T, Sridhar S, Guibas L. Continuous geodesic convolutions for learning on 3d shapes. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. 2021, p. 134–44.
[68] Donati N, Sharma A, Ovsjanikov M. Deep geometric functional maps: Robust feature learning for shape correspondence. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, p. 8592–601.
[69] Ovsjanikov M, Corman E, Bronstein M, Rodolà E, Ben-Chen M, Guibas L, et al. Computing and processing correspondences with functional maps. In: SIGGRAPH ASIA. 2016, p. 1–60.
[70] Roufosse J-M, Sharma A, Ovsjanikov M. Unsupervised deep learning for structured shape matching. In: Proceedings of the IEEE/CVF international conference on computer vision. 2019, p. 1617–27.
[71] Sun J, Ovsjanikov M, Guibas L. A concise and provably informative multi-scale signature based on heat diffusion. Comput Graph Forum 2009;28(5):1383–92.
[72] Aubry M, Schlickewei U, Cremers D. The wave kernel signature: A quantum mechanical approach to shape analysis. In: 2011 IEEE international conference on computer vision workshops. IEEE; 2011, p. 1626–33.
[73] Vallet B, Lévy B. Spectral geometry processing with manifold harmonics. Comput Graph Forum 2008;27(2):251–60.
[74] Polthier K, Schmies M. Straightest geodesics on polyhedral surfaces. In: ACM SIGGRAPH 2006 courses. 2006, p. 30–8.
[75] Clevert D-A, Unterthiner T, Hochreiter S. Fast and accurate deep network learning by exponential linear units (elus). 2015, arXiv preprint arXiv:1511.07289.
[76] Groueix T, Fisher M, Kim VG, Russell BC, Aubry M. 3D-coded: 3d correspondences by deep deformation. In: Proceedings of the European conference on computer vision. 2018, p. 230–46.
[77] Trappolini G, Cosmo L, Moschella L, Marin R, Melzi S, Rodolà E. Shape registration in the time of transformers. Adv Neural Inf Process Syst 2021;34:5731–44.
[78] Eisenberger M, Novotny D, Kerchenbaum G, Labatut P, Neverova N, Cremers D, et al. Neuromorph: Unsupervised shape interpolation and correspondence in one go. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021, p. 7473–83.
[79] Eisenberger M, Toker A, Leal-Taixé L, Cremers D. Deep shells: Unsupervised shape correspondence with optimal transport. Adv Neural Inf Process Syst 2020;33:10491–502.
[80] Liu S, Xu H, Yan D-M, Hu L, Liu X, Li Q. Wtfm layer: An effective map extractor for unsupervised shape correspondence. Comput Graph Forum 2022;41(7):51–61.
[81] Sun M, Mao S, Jiang P, Ovsjanikov M, Huang R. Spatially and spectrally consistent deep functional maps. In: Proceedings of the IEEE/CVF international conference on computer vision. 2023, p. 14497–507.
[82] Cao D, Roetzer P, Bernard F. Unsupervised learning of robust spectral shape matching. ACM Trans Graph 2023;42(4):1–15.
[83] Cao D, Lähner Z, Bernard F. Synchronous diffusion for unsupervised smooth non-rigid 3D shape matching. In: European conference on computer vision. Springer; 2024, p. 262–81.
[84] Lian Z, Godil A, Bustos B, Daoudi M, Hermans J, Kawamura S, et al. Shape retrieval on non-rigid 3D watertight meshes. In: Eurographics workshop on 3d object retrieval. Citeseer; 2011.
[85] Ezuz D, Solomon J, Kim VG, Ben-Chen M. GWCNN: A metric alignment layer for deep shape analysis. In: Computer graphics forum, vol. 36, no. 5. Wiley Online Library; 2017, p. 49–57.
[86] Hanocka R, Hertz A, Fish N, Giryes R, Fleishman S, Cohen-Or D. MeshCNN: a network with an edge. ACM Trans Graph 2019;38(4):1–12.
[87] Lahav A, Tal A. Meshwalker: Deep mesh understanding by random walks. ACM Trans Graph 2020;39(6):1–13.

[88] Milano F, Loquercio A, Rosinol A, Scaramuzza D, Carlone L. Primal-dual mesh convolutional neural networks. Adv Neural Inf Process Syst 2020;33:952–63.
[89] Smirnov D, Solomon J. HodgeNet: Learning spectral geometry on triangle meshes. ACM Trans Graph 2021;40(4):1–11.
[90] Mitchel TW, Kim VG, Kazhdan M. Field convolutions for surface cnns. In: Proceedings of the IEEE/CVF international conference on computer vision. 2021, p. 10001–11.
[91] Yazgan M, Sahillioğlu Y. A partition based method for spectrum-preserving mesh simplification. IEEE Trans Vis Comput Graphics 2023;30(10):6839–50.

Güneş Sucu is a research and teaching assistant in the Department of Computer Engineering at Middle East Technical University, where he received his B.S., M.Sc., and Ph.D. degrees. His research interests include computer graphics, graph representation learning, and geometric deep learning.

Sinan Kalkan received his M.Sc. degree in Computer Eng. from Middle East Technical University (METU), Turkey in 2003, and his Ph.D. degree in Informatics from the Uni. of Göttingen, Germany in 2008. After working as a postdoctoral researcher at the Uni. of Göttingen and at METU, he joined METU in 2010 as a faculty member and since then, has been working on problems within Computer Vision and Machine Learning.

Yusuf Sahillioğlu is a professor in the Department of Computer Engineering at Middle East Technical University. He is also an associate editor of The Visual Computer journal. His research interests include digital geometry processing and computer graphics. He has a Ph.D. in computer science from Koç University, Turkey. Contact him at or visit http://www.ceng.metu.edu.tr/~ys.