Shape Correspondence on Riemannian Manifolds by Multiple Shape Descriptors

---

THESIS

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

MASTER OF SCIENCE (Computer Engineering)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Mengxi Wang

May 20*17*

Shape Correspondence on Riemannian Manifolds by Multiple Shape Descriptors

THESIS

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

MASTER OF SCIENCE (Computer Engineering)

at the

NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING

by

Mengxi Wang

May 20*17*

Approved:

Advisor Signature

Date

Department Chair Signature

Date

University ID: **N13884678**
Net ID: **mw3191**

VITA

Mengxi Wang

Personal Data:

- Address: 245 Ocean Parkway, Apt B4
- Brooklyn, NY
- Gender: Female
- Date of birth: 03/26/1992
- mw3191@nyu.edu

Education:

- 2015/2017, NYU Tandon School of Engineering, Master of Computer Engineering
- 2010/2014, Wuhan Institute of Technology, Automation (Electrical Engineering and Automation)

Experiments:

- 2016/2017, NYU Multimedia and Visual Computing Lab
  Research on Artificial Intelligence that is applied in 3D retrieval and correspondence
- 2014/2015, Dongfeng Honda Automobile Company
  Assisted in proposing projects
- 2012/2013, Freescale Cup" Intelligent Car Competition
  Directed software programming, as well as hardware design, manufacture and maintenance
- 2011/2012, Renesas Electronics Micom Car Rally
- 2011/2014, WIT Smart Car Team

Publication:

- Academic Paper: Balance Control of Intelligent Car's Upright Moving
  Published in Silicon Valley (ISSN: 1671-7597), Volume 12-2013, Page 155

Acknowledgement

Firstly, I would like to express my sincere gratitude to my advisor Prof. Yi Fang for the continuous support of my master study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my master study.

In addition to my advisor, I would like to thank the rest of my thesis committee: Prof. Yao Wang, and Prof. Edward Wong, for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

I thank my fellow labmates in for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the last one year.

Last but not the least, I would like to thank my family: my parents for supporting me spiritually throughout writing this thesis and my life in general.

ABSTRACT

Shape Correspondence on Riemannian Manifolds by Multiple Shape Descriptors

by

Mengxi Wang

Advisor: Prof. Yi Fang

Co-Advisor: Prof. Xiao-Kang Chen

Submitted in Partial Fulfillment of the Requirements for

the Degree of Master of Science (Computer Engineering)

May 2017

This thesis provides a novel method to perform the shape correspondence, which employs multiple shape descriptors as the input data and redesigns the intrinsic neural network architecture based on the geodesic convolution neural network. The new method concatenates multiple shape descriptors based on their characteristics to ensure the comprehensiveness of the input data. In addition, the new method redefines the network architecture, which removes unnecessary layers and introduces the dropout layer, batch normalization layer and concatenating layer to achieve dense predictions. The dropout layer prevents model from overfitting by randomly disconnecting two neurons. The batch normalization layer accelerates the training process by reducing internal covariate shift. The concatenating layer connects the outputs from three geodesic convolution layers to decrease the loss of information during the training process. This experiment shows the new method has state-of-the-art performance in shape correspondence and provides much stronger predictions than other methods in multiple tasks. Even though the new method has issues in finding correspondence for shapes from different categories, it still offers valuable references for future work.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Over the last few years, the exploration of the field of three-dimensional computer vision has become popular in conjunction with the development of related hardware devices for GPU-accelerated computing. However, compared with the booming achievements in the two-dimensional domain, the research and the development of the three-dimensional domain lag behind. In the two-dimensional domain, some achievements have been widely used in daily life, such as image recognition, image retrieval, etc. For example, *He et al.* [8] used a deep residual learning framework to achieve 3.57% error on the ImageNet test set in the ImageNet Large Scale Visual Recognition Competition in 2015. In addition, the Google image search engine team has enhanced their product, which likely accounts for its sharp increase in use. However, research on the three-dimensional domain is still relatively new. A small body of literature exists regarding point set registration, but foundational research in this domain is still growing, and the field would benefit from additional research in the area of shape correspondence.

Shape correspondence is typically used to conduct a point-wise matching between the points on two or more 3D shapes. It is an essential algorithmic component in various tasks and preconditions, such as three-dimensional reconstruction or 3D printer [31]. The correspondence methods can be divided into the traditional methods and learning-based approaches. Both methods face the same difficulties. Due to the specificity of three-dimensional data structures, the complexity and uncertainty of these data contribute to the high cost of computation as follows.

High data complexity: Without any color and texture information, 3D shapes are often represented in a great collection of points with three-dimensional coordinates. These vertexes are connected by a large amount of triangles which form the surfaces. Therefore, three-dimensional data are characterized by highly complex and abstract three-dimensional geometric data that poses great challenges in the development of an effective shape representation technique for 3D shape applications.

Shape uncertainty: Many 3D objects contain symmetric parts. For example, a 3D human model has both left and right hands and legs. These hands and legs are all symmetric parts. Because the properties of symmetric parts are quite similar, it is not easy to find a good representation with discriminative power for the symmetric regions. In light of this circumstance, the symmetry ambiguity in 3D models is one of the important factors that affect the accuracy of shape representation, particularly of local shape representation.

Computational complexity: Because shape correspondence must identify the relationship between the points of one object and the points of another object, every possibility for each pair must be considered. Although the emergence of the graphics processing unit (GPU) has significantly decreased the computing time, the cost of the computation is still high,

particularly when the number of matching points is large.

1.2 Related Work

In shape correspondence, the most significant difference between traditional methods and learning-based approaches is whether they use "knowledge" learned from the previous calculations to help calculate the correspondence to the new shape. When a new shape is provided, traditional methods recalculate the points of the new shape. However, learning-based approaches apply what is learned from previous correspondences to the new shape. In other words, learning-based approaches will use the experience gained earlier to pair the points.

Some of the most widely used traditional methods are the Minimum-Distortion Algorithm [20] [25], embedding method and functional correspondence framework. The Minimum-Distortion Algorithm establishes the correspondence between two isometric shapes. It samples high-curvature vertices and then finds an injective mapping from one vertex group to the other by minimizing the isometric distortion. Because the Minimum-Distortion Algorithm needs to calculate the distance between each two groups, the cost of computation is typically high. Therefore, the following improvements have been applied to the Minimum-Distortion Algorithm to relieve the computational cost: hierarchical subsampling [28], quadratic assignment by eigendecomposition [30], pairwise constraints [15] and game-theoretic approach [23]. These improvements are founded on the mathematical theory. Besides, some other researchers attempt to find shape correspondence by spatial transformation, such as the embedding method. This method converts the Riemannian space into low-dimensional Euclidean space, which transforms the problem from a three-dimensional

domain into a two-dimensional domain and then uses two-dimensional methods to find shape correspondence. On the other hand, soft correspondence, which is different from the point-wise correspondence, distributes a point on one shape to more than one point on another shape [24]. The functional correspondence framework is a representative method to find soft correspondence, which regards soft correspondence as the mass-transportation problem. Some new methods are proposed based on functional correspondence framework, such as the sparse modeling method [22], the spectral multidimensional scaling method [1] and the partial functional correspondence method [24].

Learning-based approaches can be divided into two categories, extrinsic deep learning and intrinsic deep learning, according to the different types of the input data. The extrinsic deep learning method converts 3D shape into Euclidean structures as the input data to train the neural network to find shape correspondence. For example, *Li et al.* [16] developed a type of convolution neural network frameworks, which is successfully applied to the 3D shape retrieval. This CNN framework [14] takes projections from different viewpoints for each 3D shape, converting the data in the three-dimensional domain into the two-dimension domain. However, Euclidean structures of objects may lose their significant part or their fine details. On the other hand, intrinsic deep learning approaches directly generalize the main components on the Riemannian manifold, which shows the excellent stability of the transformation. The first convolution neural network of intrinsic deep learning approaches is the geodesic convolution neural network, which was firstly published in [19]. Another representative architecture is the Localized spectral convolution neural network, which was firstly published in [4].

1.3 Main Method

Based on previous work, this paper introduces a new method for finding the point-wise correspondence of 3D shapes. This new method extracts the patch information and multiple shape descriptors from the surface of the shape as the input data. Additionally, the new method redefines the convolution neural network architecture based on the geodesic convolution neural network, and this method can also be used to find the shape correspondence in different tasks.

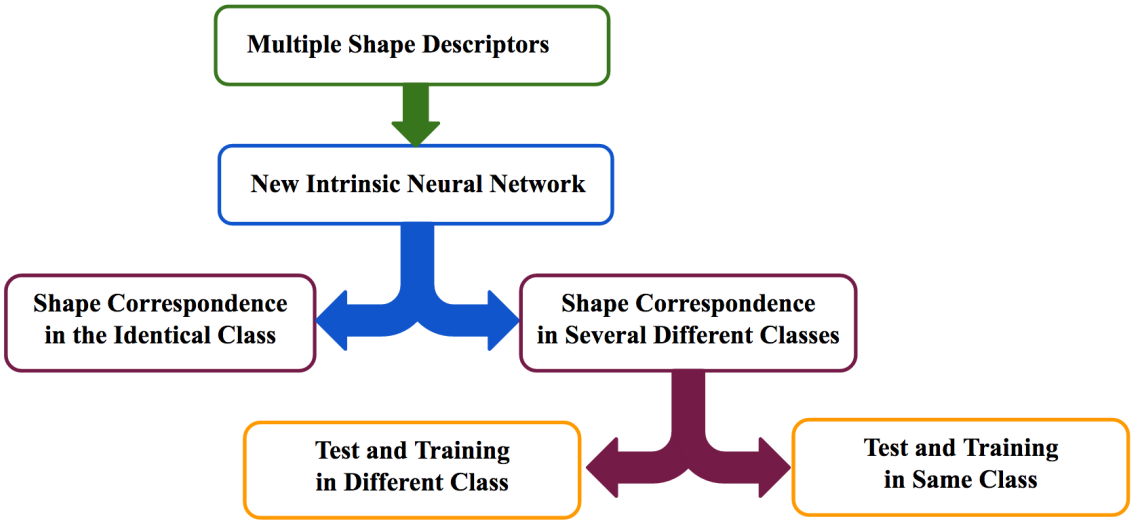
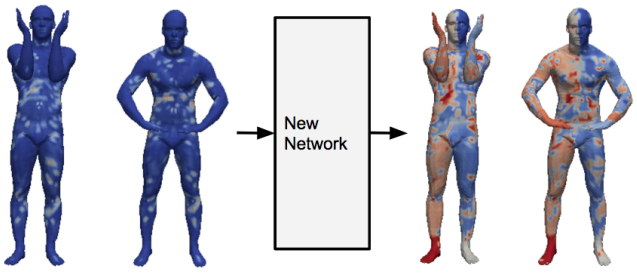


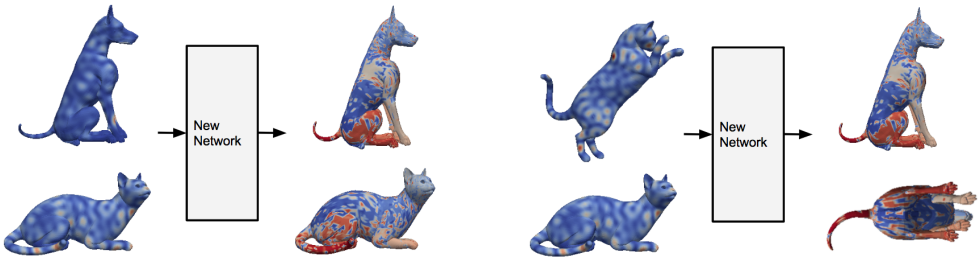
Figure 1.1: Workflow. The new method uses multiple shape descriptors as the input data to train the new intrinsic convolutional neural network, and can be used to find the shape correspondence in three tasks. The details of these three tasks are presented in Figure 1.2a 1.2b 1.2c.

In the new method, "big data" is a significant element to support the neural network with high performance. In light of this constraint, it is necessary to gather as much information as possible for each shape. According to the multi-view theory, multifaceted information can be collected by connecting multiple shape descriptors, such as the heat kernel signature (HKS), geometry vector and signature of histograms of orientations (SHOT) descriptor.

The HKS analyzes a shape from the heat transformation. The geometry vector and SHOT use the distance of the surface to define the shape. Therefore, the combination of these shape descriptors can contain comprehensive information. Second, the geodesic convolution neural network contains several layers, namely, the geodesic convolution (GC) layer, the linear layer, the angular max-pooling layer, the Fourier transform magnitude (FTM) layer and the covariance layer. However, some layers, such as the covariance layer and FTM layer, are not necessary and contribute little to the final result. The covariance layer allies with the local descriptor at each point to create a global descriptor, which is used for shape retrieval. Because shape correspondence focuses on local features instead of the global approach, the covariance layer is not needed. The FTM layer helps improve the robustness of the network. However, with the growth of training data, this layer no longer makes a significant improvement in terms of the accuracy in shape correspondence. In contrast, it adds cost in terms of computation and training time. Therefore, the redefined architecture is designed to remove these layers and to introduce certain new layers, such as the batch normalization layer, the dropout layer and the concatenating layer to improve the performance. The batch normalization layer is employed to decrease the training time, and the dropout layer is used to prevent overfitting. The concatenating layer combines the output from each convolutional layer and the input layer; it then uses this output as the final descriptor that is placed into the softmax function. Ultimately, the new architecture can address various tasks. First, it can find a correspondence in the same class. The training data and test data come from the same class. Second, this architecture also achieves high accuracy in performing shape correspondence among multiple classes. Under one condition, the training data are taken from several classes, and the test data are also gathered from these categories. Under the other condition, the training data are from one or several classes, and the test data are of different types. There is no overlapping set of the training data and test data.



(a) Finding shape correspondence in the identical class.



(b) Finding the shape correspondence in the the same classes.

(c) Finding shape correspondence in different classes.

Figure 1.2: Illustration of three different tasks. Models in the left are the training data, and models in the right are the testing data.

This paper makes three main contributions. First, it uses multiple shape descriptors by connecting different traditional shape descriptors as input data to enrich the training data. Second, this paper develops a new network architecture based on the geodesic convolution neural network. This new architecture removes certain unnecessary layers or functions that contribute little to the final result and introduces new layers including the batch normalization layer, dropout layer and concatenating layer. Third, the new method has been successfully validated and achieved superior performance over state-of-the-arts methods in shape correspondence when the reference and query shapes are from the same classes.

The remainder of this thesis is organized as follows to explain the new method in details. Chapter 2 presents the related work regarding shape descriptors and the geodesic convo-

lution neural network. Chapter 3 describes multiple shape descriptors, the architecture of the new geodesic convolutional neural network and the entire experimental configuration. Chapter 4 presents experiment results on shape correspondence using the three different tasks and the corresponding analysis and conclusions. Chapter 5 summarizes the central part of this paper and discusses potential future improvements of the methods.

Chapter 2

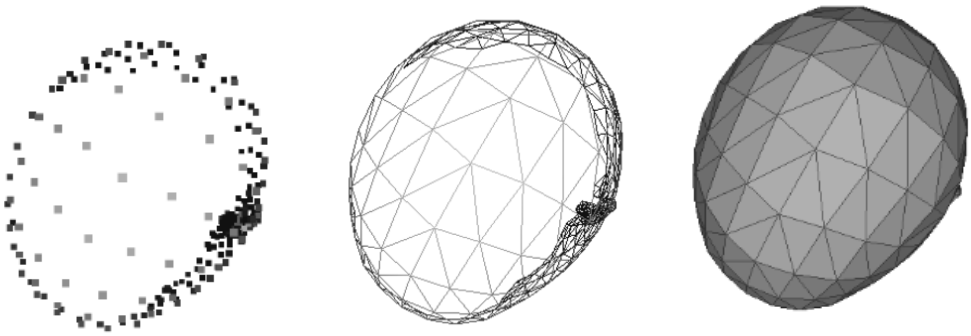
Related work

This section presents related works concerning existing research on shape correspondence. The first part discusses shape descriptors of the Riemannian manifold [10]. Based on the concept of Riemannian manifolds and Laplace-Beltrami operator [2], researchers have extended the heat diffusion [7] to Riemannian manifolds. Many shape descriptors, such as the HKS, are calculated by this mathematical regularity. All of the descriptors related to this paper are explained in terms of their physical significance. The second part introduces intrinsic neural networks and compares them with the extrinsic neural network. Then it interprets each layer's function in the geodesic convolution neural network as one representative type of intrinsic neural networks.

2.1 Shape descriptor

The surface of a 3D shape can be described by a connected smooth compact two-dimensional manifolds M, which is called Riemannian manifold. Researchers propose the Riemannian metric in order to define various geometric notions on a Riemannian manifold. A Riemannian metric is an inner product $< .,. >_x = T_xX \times T_xX$ on the tangent space and smoothly

relies on point x. T_xX denotes the tangent plane at point x, which simulates the surface as a two-dimensional Euclidean space locally around the point x. In differential geometry, the Laplace-Beltrami operator, $\nabla^2 f = \nabla \cdot \nabla f$, also referred to as the Laplace operator, can be summarized as a function which works on the surface in Riemannian manifolds. The Laplace-Beltrami operator is intrinsic because its expression entirely depends on the Riemannian metric. As a result, the Laplace-Beltrami operator has the invariance of isometric transformations in the manifold. In addition, it admits an eigendecomposition, $\Delta_X \phi_k = \lambda_k \phi_k$, with real eigenvalues, $0 = \lambda_1 \leq \lambda_2 \leq ...$ on a compact manifold. The corresponding eigenfunctions, $\phi_1, \phi_2, ...$, form an orthonormal basis on $L^2(X)$, which generalizes the Fourier basis to non-Euclidean space [18].



(a) Point (b) Edge (c) Face

Figure 2.1: Construction of Riemannian manifolds

Heat kernel signature (HKS)

A heat kernel signature (HKS) [27] relies on the heat kernel diffusion equation to compute each point's heat transfer over the model surface. HKS reflects the total quantity of heat that is left at point x through time t. For a given compact Riemannian manifold M, the heat diffusion process over M is governed by the heat equation,

(a)

(b)

Figure 2.2: HKS map. Figure 2.2a shows the HKS map at time t_1. Figure 2.2b shows the HKS map at time t_{100}.

$$\Delta_M u(x,t) = -\frac{\partial u(x,t)}{\partial t} \tag{2.1}$$

, in which Δ_M is the Laplace-Beltrami operator of M. If the manifold has a boundary, there is an another requirement that u satisfies the Dirichlet boundary condition, $u(x,t) = 0$, for all point $x \in \partial M$ and all t [4]. Therefore, the heat kernel can be expressed as

$$h_t(x,x') = \sum_{k \geq 1} e^{-t\lambda_k} \phi_k(x) \phi_k(x') \tag{2.2}$$

, in which λ_i and ϕ_i are the eigenvalue and eigenfunction of the ith Laplace-Beltrami operator, respectively. The equation 2.1 can be expressed in the spectral domain as

$$u(x,t) = \int_M u_0(x') \sum_{k \geq 1} e^{-t\lambda_k} \phi_k(x) \phi_k(x') dx' \tag{2.3}$$

, in which u_0 is the heat distribution that $t = 0$, $u_0 = u(x,0)$. $u(x,t)$ value will change along

with the time, t, to reflect the surface information.

Optimal spectral descriptors (OSD)

Optimal spectral descriptors(OSD) defines its transfer function as

$$\tau_q(\lambda) = \sum_{m=1}^{M} a_{qm}\beta_m(\lambda) \tag{2.4}$$

, where $\beta_m(\lambda) \in \{\beta_1(\lambda), ..., \beta_M(\lambda)\}$ is the basis spline, and $a_{qm}(q = 1, ..., Q, m = 1, ..., M)$ are the parametrization coefficients. Here, Q is the descriptor dimensionality. By taking equation 2.4 into equation of spectral function, the OSD can express as

$$\begin{aligned} f_q(x) &= \sum_{k\geq 1} \tau_q(\lambda_k)\phi_k^2(x) \\ &= \sum_{m=1}^{M} a_{qm} \sum_{k\geq 1} \beta_m(\lambda_k)\phi_k^2(x) \end{aligned} \tag{2.5}$$

At the same time, it can be defined as

$$g_m(x) = \sum_{k\geq 1} \beta_m(\lambda_k)\phi_k^2(x) \tag{2.6}$$

Thus, $g(x) = (g_1(x), ..., g_M(x))^T$ is a vector value function, which is referred to as the geometry vector. This geometry vector only relies on the geometry of the shape [18]. It can learn the optimal parameters, a_{qm}, by minimizing the loss function, which reduces to a Mahalanobis-type metric learning [17].

(a)

(b)

Figure 2.3: Geometry vector map. Figure 2.3a shows the geometry vector map at $m = 1$. Figure 2.3b shows the geometry vector map at $m = 100$.

Other descriptor

In addition to HKS and OSD, there are other kinds of descriptors, such as the Geodesic descriptor and the signature of histograms of orientations (SHOT) descriptor. In Riemannian manifolds, the Geodesic descriptor records the geodesic distance on the surface of 3D shapes between all points. It finds the shortest distance along the mesh edge between two points and uses the matrix to express this feature. Another expressive descriptor is the signature of histograms of orientations (SHOT) descriptor. The SHOT firstly calculates a set of local histograms and then gathers all local histograms together to form the actual descriptor. Each local histogram accumulates point counts into bins based on a function $cos\theta_i$, where θ_i is the angle between the normal, n_{v_i}, at each point at the corresponding part of the grid, and the normal,n_u, at the feature point [29]. The SHOT is rotation invariant and robust to noise as well as shape variations.

Figure 2.4: SHOT descriptor

2.2 Neural network

2.2.1 Intrinsic deep learning

Inspired by biological neural networks, the neural network is a system containing a large collection of simple neural units, which provides dynamic outputs according to external inputs. Each layer in the neural network contains a number of connected neural units. A completed neural network may contain multiple layers, and the signal path traverses from the first (input) layer to the last (output) layer. The layer between the input layer and output layer is called the hidden layer. The output y of each neural unit can be expressed as

$$y(x) = f(\sum_{i=1}^{n} w_i x_i + b). \tag{2.7}$$

, where $x_i = \{x_1, x_2, ..., x_n\}$ represents the input of each neural unit, and $w_i = \{w_1, w_2, ..., w_n\}$ represents the weight matrix, and b represents the bias. There may be a threshold function or limiting function, f, which limits the summation, $\sum_{i=1}^{n} w_i x_i + b$, before propagating to other neurons. The weights of each neural unit are adjusted by the feed-forward algorithm, which minimizes the difference between the predicted result and actual result in the training process to achieve the better performance. The neural network has achieved significant

progress in several subjects over the years, including computer vision. At the same time, the neural network has evolved into various architectures, including the convolution neural network.

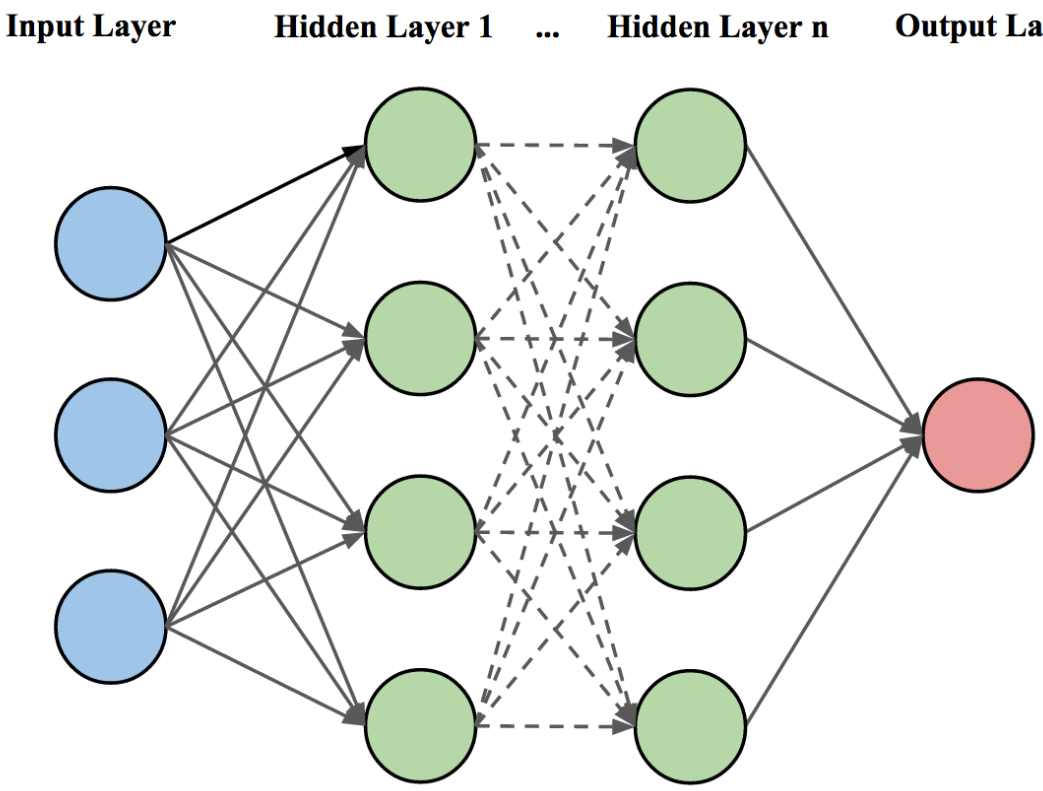


Figure 2.5: Typical neural network architecture. A typical neural network contains the input layer, output layer and certain hidden layers.

The convolution neural network [13] is a type of feed-forward neural networks that is primarily applied to computer vision. The convolutional layer is the core building block of a convolution neural network. It contains a set of learnable kernels (or filters), and each kernel is an array of weights in a neural unit. These kernels have a small receptive field but extend through the full depth of the input volume. Each kernel is convolved across the width and height of the input volume, which computes the dot product between all kernels and the inputs. Therefore, the output y of each neural unit can be expressed as

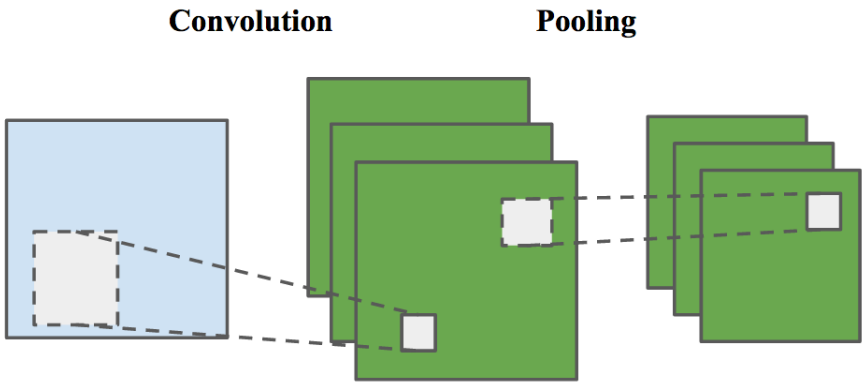
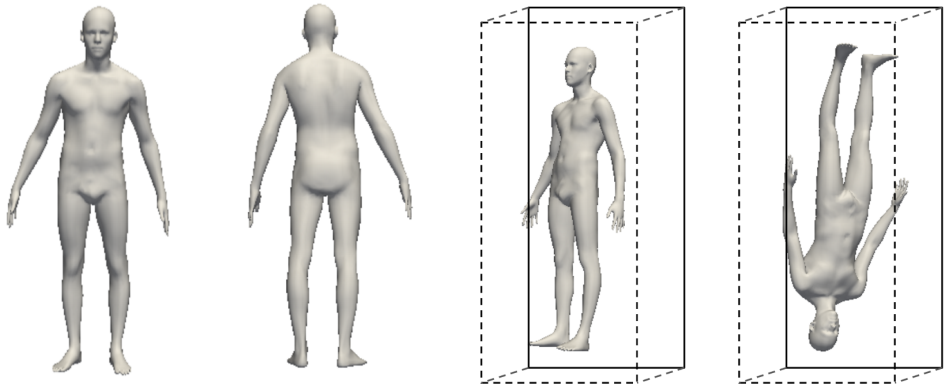


Figure 2.6: Convolution layer and pooling layer

$$y_{ij}^{k}(x) = tanh((W_{ij}^{k} * x_{ij}) + b_k) \tag{2.8}$$

, where $w_i = \{w_1, w_2, ..., w_n\}$ represents the kernel, and $x_i = \{x_1, x_2, ..., x_n\}$ represent the input data, and b represents the bias. Because each kernel is connected to a small receptive field and convolved across the width and height of the input volume, this connection is local in space but always extends along the depth of the input volume, which ensures the local connectivity. In addition, each neuron unit is replicated in convolution layers, which shares the same parameterization (weight vector and bias). In other words, the weights in the convolution layers are set to be equal. In light of this circumstance, this pattern dramatically reduces the number of parameterization of neurons, thus, reducing the computation complexity and training time.

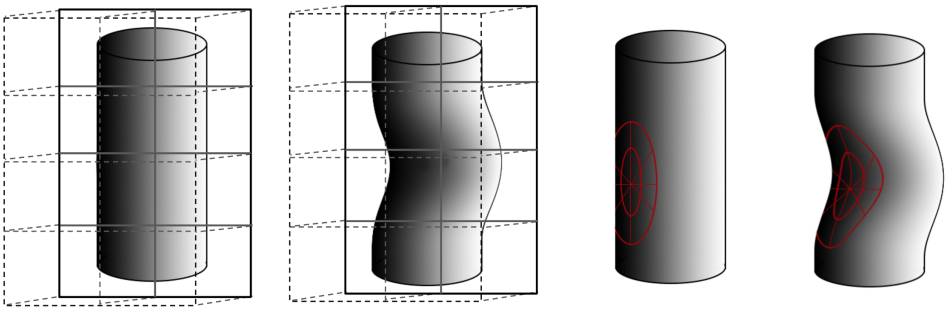
The intrinsic deep learning method applies the neural network to the input data that is intrinsic, such as the geometric data. Compared to the extrinsic deep learning method, the intrinsic deep learning method directly uses the main gradients which are generalized by convolution or other methods on non-Euclidean domains. The extrinsic deep learning method converts 3D shapes into Euclidean structures, such as depth images, as the input



(a) Projection (b) Volume data

Figure 2.7: Non-intrinsic data. Projections obtained from the front and back viewpoint are different. The volume data of the same shape can be different.

data to train the neural network. However, Euclidean structures of objects may lose their significant parts or their fine details, or even break their topological structure during the conversion process. On the other hand, Euclidean structures are not intrinsic and vary as the result of pose or deformation of the object [5]. These drawbacks of Euclidean structures limit the development of the extrinsic deep learning method. However, the intrinsic deep learning method uses the intrinsic data as the input data to train the neural network. The intrinsic data directly generalize the main ingredients from the non-Euclidean domain and is invariant to the deformation. In light of this circumstance, the intrinsic deep learning method does not have the drawbacks of the extrinsic deep learning method. Because 3D shapes are highly complex in shape correspondence, the intrinsic deep learning method uses the convolution neural network as the computational model to reduce the computation complexity and training time. The first intrinsic convolution neural network architecture is the geodesic convolution neural network, which was presented in [18]. It is an extension of the convolution neural network to Riemannian manifolds based on the local system of geodesic polar coordinates and shows the excellent performance in shape correspondence.



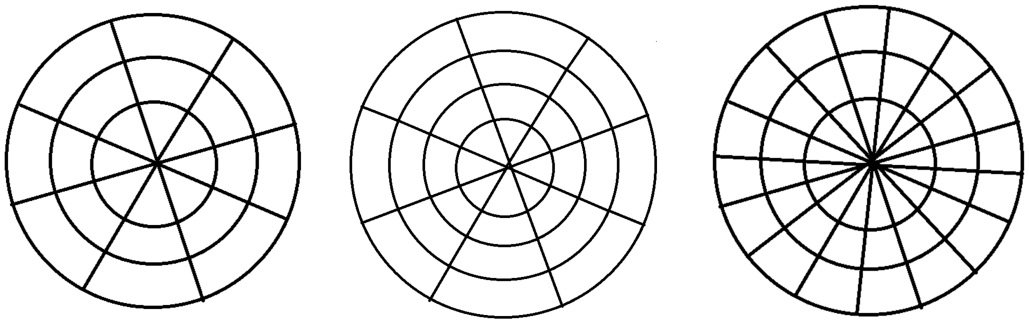
(a) Non-intrinsic data (b) Intrinsic data

Figure 2.8: Difference of the non-intrinsic data and intrinsic data. (a) shows the non-intrinsic data. When the cylinder warps, the volume data are not invariant to the deformation. (b) shows the intrinsic data. When the cylinder warps, the patch on the surface is invariant.

2.2.2 Geodesic convolution

The core building block of the geodesic convolution neural network is the geodesic convolutional layer, which applies geodesic convolution on Riemannian manifolds. Geodesic convolution follows the "correlation with template" idea by constructing a local system of geodesic polar coordinates constructed at point x to extract patches on Riemannian manifolds [18]. In other words, geodesic convolution constructs geodesic polar coordinates to define the circular area as the "patch" around each point on the surface of 3D shapes and the center of each circular area is point x. The radial coordinate is built as ρ-level sets $x' : d_X(x,x') = \rho$ of the shortest path (geodesic) distance function for $\rho \in [0,\rho_0]$, where ρ_0 is the radius of the geodesic disc and is empirically set to be 1%. The angular coordinate is constructed as a set of rays that radiate from the point x in the direction θ and these rays are perpendicular to the geodesic distance level sets [18].

The bijective map, $\Omega(x) : B_{\rho_0}(x) \rightarrow [0,\rho_0) \times [0,2\phi)$, extracts from Riemannian manifolds into the local geodesic polar coordinates (ρ,θ) at point x. The patch operator, $(D(x)f)(\rho,\theta) = (f \circ \Omega^{-1}(x))(\rho,\theta)$, interpolates f in the local coordinates, and $D(x)f$ can be regarded as a "patch" on Riemannian manifolds. The patch operator is firstly introduced



(a) (b) (c)

Figure 2.9: Patch

in [12], which can be expressed as

$$(D(x)f)(\rho,\theta) = \int_X v_{\rho,\theta}(x,x')f(x')dx' \tag{2.9}$$

$$v_{\rho,\theta}(x,x') = \frac{v_\rho(x,x')v_\theta(x,x')}{\int_M v_\rho(x,x')v_\theta(x,x')dx'} \tag{2.10}$$

, where $v_\rho(x,x')$ is the radial interpolation weight, and $v_\theta(x,x')$is the angular weight. Due to the definition of the patch operator, the geodesic convolution can be defined as

$$(f*a)(x) = \sum_{\theta,r} a(\theta+\Delta\theta,r)(D(x)f)(r,\theta) \tag{2.11}$$

, where $a(\theta,\rho)$ is a filter applied on the patch. This filter can be rotated by arbitrary angle because of the angular coordinate ambiguity $\triangle\theta$ [18].

Because the local system of geodesic polar coordinates is discrete, the patch operator is also discrete. The local geodesic polar coordinates can be constructed on Riemannian manifolds by dividing a 1-ring of the point x into N_θ equal-angular bins by multiple emanative lines. These emanative lines propagate into adjacent triangles by unfolding the respective triangles used in [11]. The angular bins are created by these emanative lines and radial bins

are built as level sets of the geodesic distance function, which are computed by fast marching [11]. An $N_\theta N_\rho N \times N$ matrix can be used to represent the discrete patch operator.

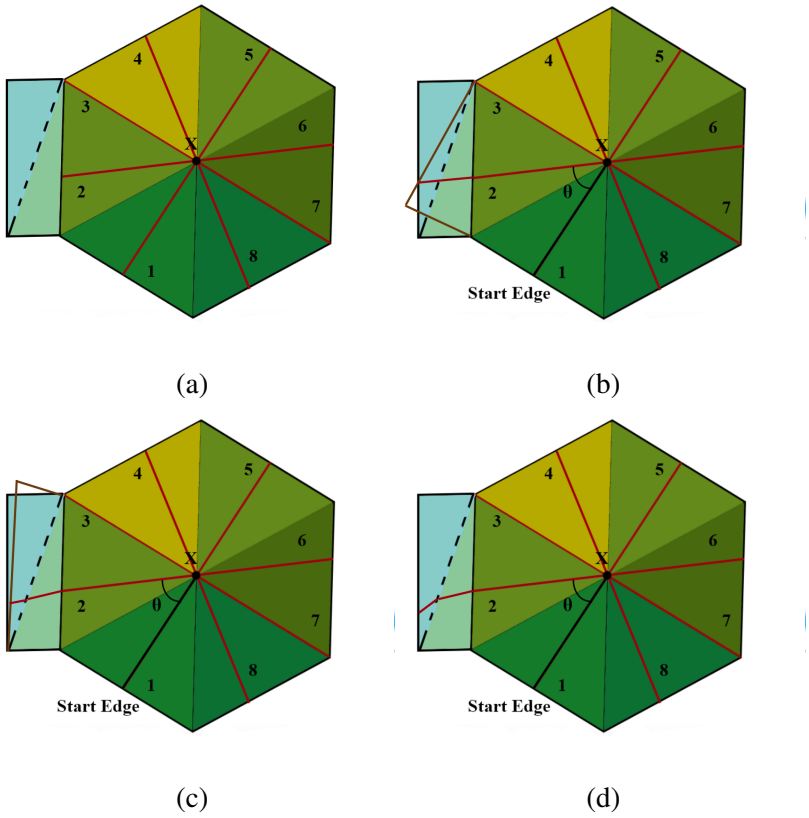


Figure 2.10: Steps to attach the patch on manifolds. (a) partitions a ring into N_θ equiangular bins with radial lines. (b) propagates along one radial line into conterminal triangular mesh by unfolding procedure resembling method. (c) repeats the same step to another adjacent triangular mesh. (d) shows the final result.

2.2.3 Network Components

The geodesic convolution neural network contains several layers that are applied subsequently, including the linear layer, the dropout layer, the geodesic convolution (GC) layer, the angular max-pooling layer, the batch normalization layer and the softmax layer. The linear layer typically follows the input layer or precedes the softmax layer to adjust dimensions of the input data. The GC layer substitutes the convolution layer used in the classical

convolution neural network. The dropout layer is a fixed layer that typically precedes the GC layer. The angular max-pooling and batch normalization layers are two other fixed layers that follow the GC layer. The softmax layer applies the softmax function to the input data of this layer and exports final probability distributions.

The linear layer adjusts dimensions of the input and output data by means of a linear combination. The output of linear layer can be defined as

$$f^{out}(x) = f_2(\sum_{n=1}^{N} w_n f_n^{in}(x)) \tag{2.12}$$

, where f_2 is the activation function, and f^{in} are the input data of each neuron. Optionally, there is a non-linear function, which follows the output of the linear layer, such as $ReLU$, $\xi(t) = max(0,t)$. Therefore, the linear layer has the same function with the fully connected layer used in the convolution neural network after the introduction of a non-linear function.

The GC layer applies geodesic convolution on Riemannian manifolds, which is the homogeneous function in the convolution neural network. Because the angular coordinate is ambiguous, the GC layer calculates the geodesic convolution result for all N_θ rotations of the filters. The output of the GC layer can be expressed as

$$f_{\Delta\theta}^{out}(x) = \sum_{n=1}^{N} (f_n^{in} * a_{\Delta\theta,n})(x) \tag{2.13}$$

, where $a_{\Delta\theta,n}(\theta,r) = a_n(\theta+\Delta\theta,r)$ is the weight of the nth filter, and $\Delta\theta = 0, \frac{2\pi}{N_\theta}, ..., \frac{2\pi(N_\theta-1)}{N_\theta}$.

The dropout layer is a fixed layer that precedes the GC layer and it applies regularization technique to prevent the neural network from overfitting [26]. The dropout layer randomly

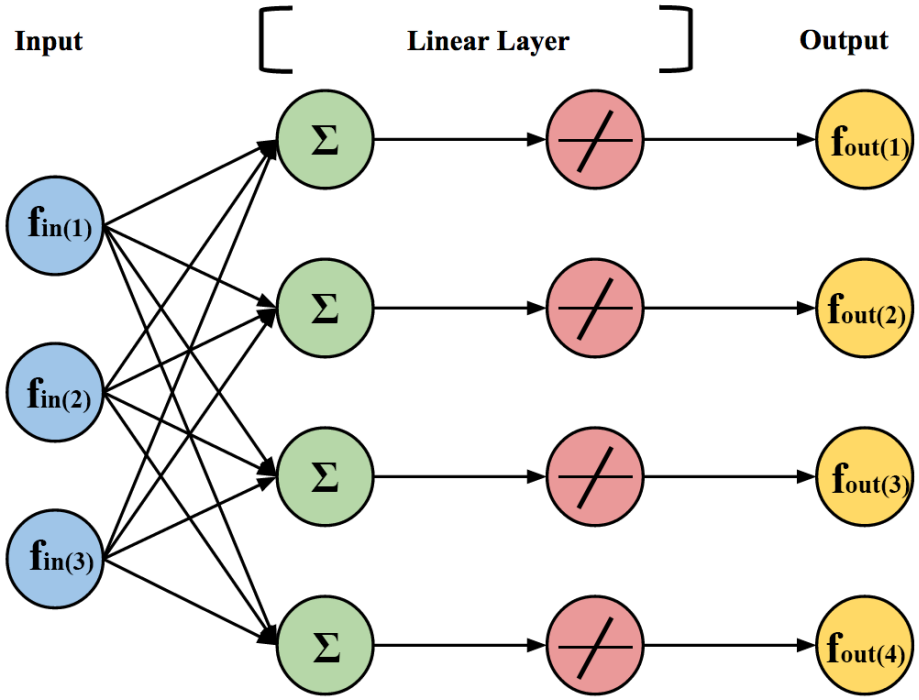


Figure 2.11: Linear layer. Blue circles represent the input, green circles represent the linear combination, red circles represent the activation function, and yellow circles represent the output.

disconnects connections between neurons in the previous layer and neurons in the following layer. Empirically, the hyper-parameter, $\sigma \in (0,1)$, is set to 0.5, which determines the dropout ratio. A binary mask b_n is generated based on the dimension of the input data during the training phase. Each binary mask contains 0 with probability σ_n and 1 with probability $1-\sigma_n$. Therefore, the dropout layer can be defined as

$$f_n^{out}(x) = b_n f_n^{in}(x). \tag{2.14}$$

, where b_n is the binary mask, and $f_n^{in}(x)$ are the input data of the dropout layer.

The angular max-pooling layer and the batch normalization layer are two fixed layers that follow the GC layer. The angular max-pooling layer is a fixed layer that directly follows the GC, which takes the maximum value from the output of all N_θ in the GC layer. The

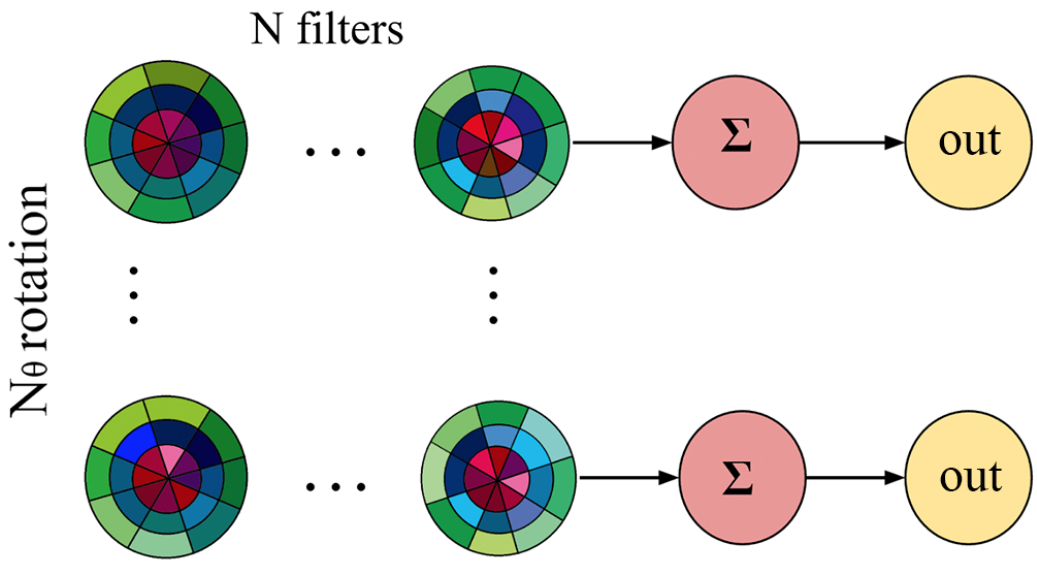


Figure 2.12: GC layer

angular max-pooling layer can be defined as

$$f_{\Delta\theta}^{out}(x) = \max_{\Delta\theta} f_{\Delta\theta}^{in}(x) \tag{2.15}$$

, where $f_{\Delta\theta}^{in}$ is the output of the GC layer. The batch normalization layer is a fixed layer that directly follows the angular max-pooling layer. Because 3D shapes are complex, the cost of computation is typically high. Therefore, the geodesic convolution neural network needs a long time to compute geodesic convolution in each GC layer, resulting in the long training process. However, the introduction of the batch normalization layer can reduce the cost of computation, which also reduces the training time. The batch normalization layer is firstly presented in the paper [9], which describes a new way to accelerate deep network training by reducing internal covariate shift. Due to the change of the parameter in the previous layer, the distribution of the input data in each GC layer would change too, and this change is referred to as internal covariate shift. Therefore, the change of the input

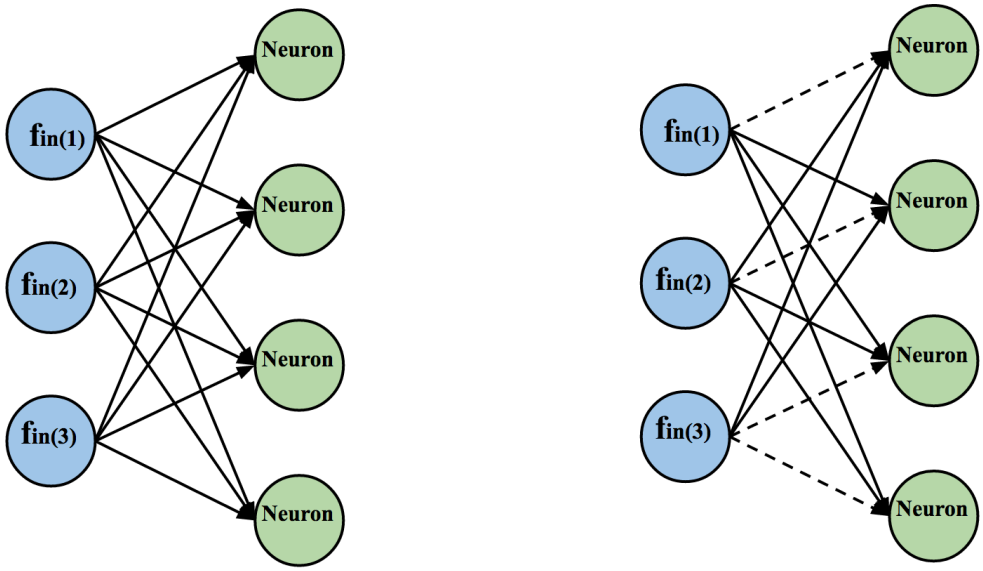


Figure 2.13: Dropout layer

data in each layer leads to lower learning rates and meticulous parameter initialization, resulting in the difficult training process with saturating nonlinearities [9]. Therefore, the batch normalization layer addresses the internal covariate shift by normalizing the input data, which can be defined as

$$f^{out}(x) = \frac{f^{in}(x) - \mu}{\sqrt{\sigma^2 + \varepsilon}}\gamma + \beta \tag{2.16}$$

, where μ is the mean of the input data, and σ^2 is the variance of the input data estimated on the training set by exponential moving average, and ε is a small positive constant to prevent numerical errors [5]. The experiment result proves that the batch normalization layer dramatically reduces the training time.

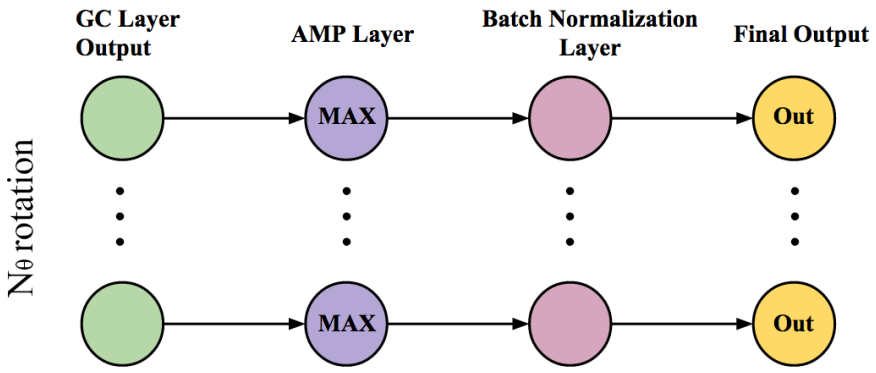


Figure 2.14: Angular max-pooling layer and batch normalization layer

The softmax layer applies the softmax function to predict the probability of correspondence between each pair of points. This layer is usually treated as the final layer and exports the vector of the probability distribution. The softmax function, which also is referred to as the normalized exponential function, is a generalization of the logistic function that rescales an arbitrary vector into the same dimensional vector with value in the range $(0, 1)$.

Chapter 3

Approach

3.1 Multiple shape descriptors

The training data are enriched by connecting different shape descriptors. All of the shape descriptors are of the same length and are concatenated to construct the final matrix of shape descriptors. This matrix contains all of the information that is extracted from the surface of the shape. There are two reasons to join these descriptors. One reason is to present more details for a given shape, and the other reason is to ensure the better performance of the neural network. For each shape, a single type of shape descriptor contains only limited information about the surface because the process of extracting shape descriptor filters data by a series of mathematical functions. These functions characterize the most valuable information on a shape and discard other parts; however, certain useful parts that do not satisfy the criterion of significance are eliminated as well. For example, the HKS focuses only on the heat distribution on the surface and ignores the orientation of each part of the body. If a shape is a symmetrical structure, the HKS is symmetrical as well. However, the HKS is associated with the geometrical information, and it is difficult to distinguish the right hand from the left side for a strictly symmetric human shape. Therefore, other shape

descriptors are required to compensate for this flaw, such as the SHOT. The SHOT descriptor focuses on the local geometrical features but neglects the global features, which is complementary to the HKS. In light of this circumstance, it is necessary to connect a variety of shape descriptors, such as the HKS, SHOT and the geometry vector, as the input data.

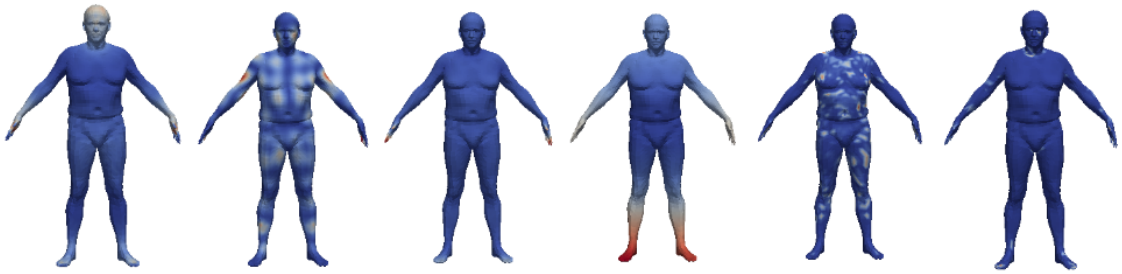


Figure 3.1: Multiple shape descriptors. Every two shapes is a group. The first group represents the geometry vector, the second group represents the HKS, and the third group represents the SHOT descriptor. The dimension of the first item in each group is 1 and the dimension of the second item is 100.

In addition, multiple shape descriptors improve the performance of the neural network. First, a comprehensive network framework requires ”big data” to support it. Otherwise, the network can easily become overfitted during the training process without sufficient data, particularly for a multistory structure. Second, multiple shape descriptors can ensure that a neural network has integrated input data, which means that the input data include all of the possible situations. Because the input data represent the key points of shapes that are refined on the surface, these data might not be the information that the network requires. In other words, the quality of a network structure is determined by the data provided. If the input data do not contain the information that a real neural network is required to know, then this incompatibility can lead an unwanted training process. When the training set does not provide the corresponding information, poor accuracy can arise in the test results. This situation is worse for certain delicate tasks, such as shape correspondence. When the training set is similar but not entirely consistent with the data that the neural network requires,

then the result can also result in distortion of the refined information. However, the use of multiple shape descriptors provides the comprehensive and concentrated data for the training process. This approach ensures that the training process is controllably oriented.

When multiple shape descriptors are applied to the new method, parameter assignment, including the selection of shape descriptors and the dimension of each shape descriptor, is an important element that must be considered. Due to the introduction of the neural network, the training processing selects the required data and removes unnecessary information, which leads to the convergence of deterministic data even if multiple shape descriptors have overlapping data. The training process selects adequate data and removes redundant terms. Even though the process could also filter the data that is not intimately related, the existence of sufficient data can make up for this deficiency. In addition, an analogous problem exists regarding the dimensional proportion of each shape descriptor. In general, each type of input data has the same dimensions as that of previous works. However, this tendency is not a universal rule. Because of the selection function used in the training process, there is a small to minimal effect on the final result when the dimension of each shape descriptor is different. In this regard, the proper method must ensure coverage of the input data. Ideally, it is better to concatenate shape descriptors as much as possible, thus ensuring the comprehensiveness of the underlying information on each shape. However, the associated computational cost limits the practicality of this goal. In reality, the memory cannot accommodate such large consumption. Additionally, this approach also leads to an enormous amount of training time. Even though the existence of a GPU could speed up the calculation, the training process is still time-consuming. There must exist overlapping areas in the input data from multiple shape descriptors, which would consume power during the calculation. In this regard, the approach is to use pre-selected elements when choosing appropriate shape descriptors. Figure 3.1 shows three descriptors that were used in our experiments. The first group is the geometry vector. At time $t = 1$, the vector exhibits an

abstract gradual change to distinguish the main parts, such as the head, body, and hands. After time $t = 100$, it is subdivided into several smaller parts, thus making the global viewpoint clear. However, this change is still symmetric. Thus, it is difficult to distinguish the right hand from the left side. The second group is associated with the HKS. When $t = 1$, the complete model is shown in almost in the same color, which represents similar values in this parts. However, there are significant other colors on the top of the head, the hands, and the feet. When $t = 100$, the feature map also becomes clearer. The leg can be distinguished from the feet. The SHOT descriptor is asymmetric. These three shape descriptors exhibit differences and any one descriptor can complement another if one is insufficient.

3.2 New Intrinsic Neural Network

This part introduces the new intrinsic neural network architecture and justifies its use. The new architecture removes the FTM layer and adds the batch normalization layer and dropout layer. Additionally, it incorporates the concatenating layer into the model. Through the recombination of these layers, this paper proposes the new structure shown in Figure 3.2. An input layer is followed by three combined GC layers. The output of these four layers feeds into the concatenating layer and then goes to the softmax layer. The final result is the prediction of the shape correspondence.

The FTM layer can apply multiple phase ambiguity by adjusting the absolute value to improve the performance. Although this step does not contribute substantially to the accuracy of the final result, it greatly increases the training time due to requiring a significant amount of computation to apply the Fourier transform. Because of the balance between the growth of the training time and the accuracy, the new architecture removes the FTM layer. Addi-

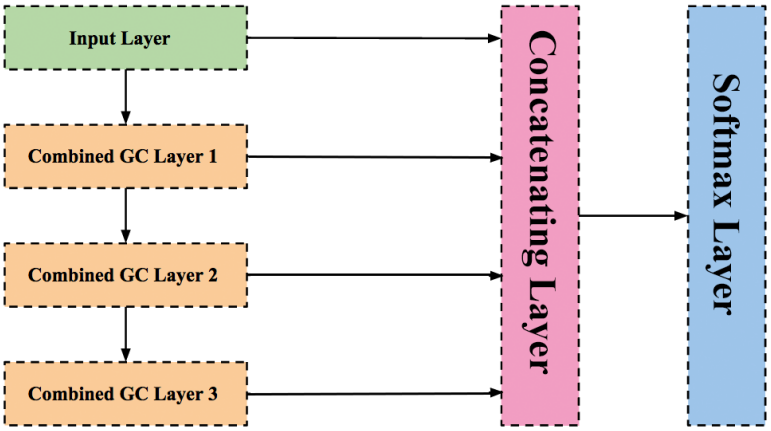


Figure 3.2: Our complete intrinsic neural network architecture.

tionally, the batch normalization layer can decrease the training time by reducing the internal covariate shift. This approach makes normalization a part of the model architecture and performs the normalization for each training mini-batch, providing excellent performance in terms of decreasing the time cost in the training process based on prior and current experience. The dropout layer is a function that randomly sets the weight of each convolutional layer to zero to prevent overfitting and ensure a fast and efficient training process. The linear layer is used primarily to calculate the appropriate parameters in the optimal spectral descriptor (OSD). It also can adjust the dimension of the input data. However, if no linear layer exists in the architecture, this framework still has the same accuracy as a model with a linear layer, according to experimental results. In light of this finding, the new design removes this layer to reduce the computation cost.

The concatenating layer connects the output of each previous layer to compensate for the missing information. It performs a merging function to link multiple inputs along the specified axis, which is defined as follows:

$$f_n^{out}(x) = \sum_{n=1}^{N} a_n f_n^{in}(x) \tag{3.1}$$

, where a_n are the weights of each layer. This formula manually sets the weight to 1 based on past experiences. Shape correspondence is a delicate task that requires a dense prediction. By connecting the output of each GC layer, this dense prediction can be obtained from the collection of preliminary results. The training data is selected and extracted through the training process. However, this aggregation process does not guarantee that all of the useful information can be retained. This reason is that the feature map becomes abstract but focuses on the edge information during the training process in the traditional convolution neural network of the two-dimensional domain. This feature map shows the initial but indicated predictions, which is an advantage for certain tasks, such as shape retrieval or shape segmentation. Shape retrieval uses the global features to distinguish one instance from other instances from different classes. Therefore, such retrieval focuses primarily on what can be expressed in the complete model instead of the local part of the body. Shape segmentation focuses on the main features of each part. Because the new framework treats shape correspondence as a classification problem, shape correspondence is similar to the problem that classifies each point into the correct group. Because shape correspondence performs classification based on the vertices, it has additional classes that must be divided. Therefore, a high demand exists for the details information. One possibility is to connect the output from each convolution layer and the input layer to make up for the losses in the training process. The experimental results indicate that the accuracy is dramatically improved. The output of this layer can be referred to as the final feature map and can be input into the softmax layer to predict the possibility.

The lose function adjusts the parameters by minimizing the loss of each training epoch to achieve a sufficiently trained model. The new model can be expressed as a non-linear

hierarchical parametric function F, where $F = \{f(x_1), f(x_2), ...f(x_N)\}$ is a $P \times N$ matrix of input shape descriptors at all the points of the mesh [18]. In addition, W represents the parameters of all of the layers. For a query shape X and reference shape Y, $y_1, ..., y_N$ indicates the vertex of shape Y, and let y_{j_i} indicate the vertex that is responding to x_i for $i = 1, ..., N$. For each vertex on X, the new model produces an N-dimensional vector to show the probability of Y. Therefore, our loss function is designed as follows:

$$L(W) = -\sum_{i=1}^{|T|} e_{j_i} log f(i) \tag{3.2}$$

, where $T = \{f(x_i), j_i\}$ is a training set, which includes all known point correspondence. Through minimizing the loss function, the optimal weights can be determined to complete the training process.

The new geodesic convolutional neural network combines the above-mentioned improvements to build the final architecture, which converts shape correspondence into shape classification. For each vertex of a shape, a label is given to distinguish this vertex from the other points. The corresponding vertex in the other shape has the same label, which places them in the same group. During the training process, the new model would find the common characteristics of the vertex that has the same label, and this process is analogous to a classification problem. However, in the regular corresponding problem, practitioners generally prefer to develop a Siamese network. This type of network minimizes the similarity between positive samples and maximizes the dissimilarity between negative samples. For example, there are two shapes X and Y, and these shapes contain 100 vertices. Positive samples are any two corresponding points, and negative samples are any two points that do not have any a relationship. In a Siamese network, the goal is to make the output of the corresponding points as similar as possible. At the same time, the network also needs to make

the mismatching points as dissimilar as possible. However, the number of negative samples is much higher than the number of positive samples. For one point in shape X, there is only one positive sample and 99 negative samples. This imbalance can lead unexpected results during training. Additionally, this approach relies on a large number of calculations, which could increase training time dramatically. In light of these considerations, the new model can accelerate the training process by solving the problem as a classification problem rather than the usual problem of shape correspondence. In addition, the model can also extend into another application, such as shape segmentation, because shape segmentation is also a type of classification problem. However, the model requires more abstract descriptors to describe each part of the shape relative to subtle correspondence. The geodesic convolution process propagates the information at one point to the adjacent points. Through the convolution function, information on one patch is gradually delivered to a nearby patch. Consequently, the new model can extract a global shape descriptor by combining the local descriptors, which can be used to do perform shape retrieval.

In summary, the complete architecture of the new model can be described as follows. Each GC layer is connected to a dropout layer. At the same time, an angular max-pooling layer and a batch normalization layer follow the output of the GC layer. Because these layers are always combined, these four layers can be referred to as the combined GC layer. In the new model, the combined GC layer is used to replace the continuous four layers, which are the GC layer, dropout layer, angular max-pooling layer and batch normalization layer. Figure 3.2 shows the completed neural network architecture. Three combined GC layers follow the input layer, and the outputs of these three layers feed into the concatenating layer. Finally, the output of the concatenating layer is processed by the softmax function to obtain the final probability distribution vector. The main point is the appearance of the concatenating layer. The reason is that in the training process, the feature map of each convolution layer becomes abstract, which implies a loss of information. By adhering to a

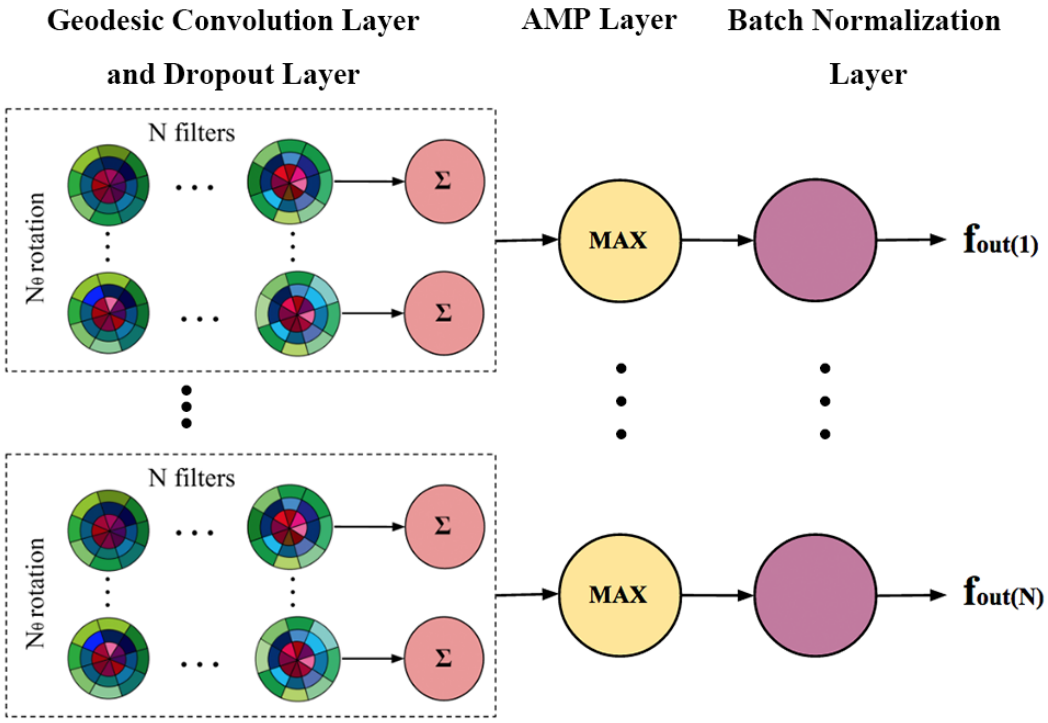


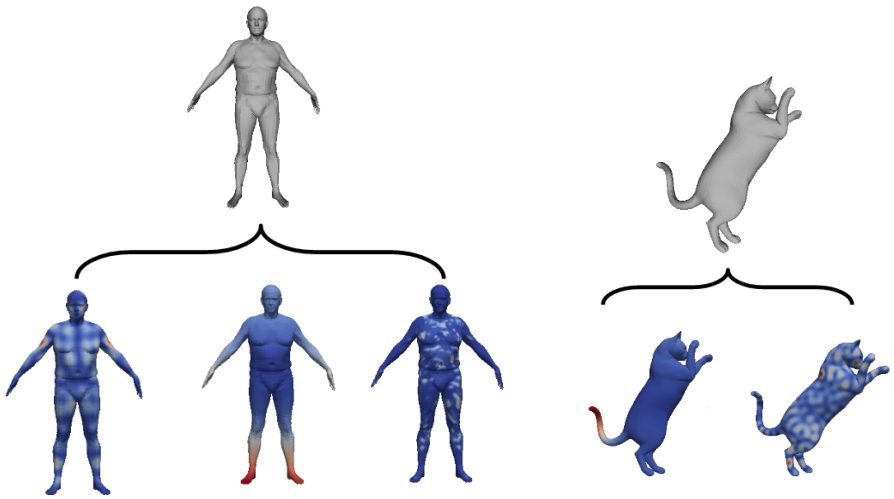
Figure 3.3: Combined GC layer

coarse output, a dense feature map can be obtained, which leads to a dense prediction.

3.3 Experiment

Two datasets are used in the experiment in shape correspondence. One is the TOSCA dataset [21, 6], which contains high-resolution three-dimensional non-rigid shapes. The TOSCA dataset contains a total of eighty objects, including eleven cats, nine dogs, three wolves, eight horses, six centaurs, four gorillas, twelve female figures, and two different male figures, one of which containing seven poses, and the other containing twenty poses. Typically, the number of vertices is approximately 50,000. In the same class, the models have the same triangulation and an equivalent number of vertices numbered in a compatible way. The other dataset is the FAUST [2] [3] dataset, which contains 300 real, high-

resolution human scans. These human models have dense ground-truth correspondences, and each human model has 6890 vertices. A previous study [18] resampled human models in TOSCA to ensure that those human models have the same number of vertices with the shapes in FAUST. Additionally, the human shapes in TOSCA and FAUST are scaled to a unit geodesic diameter. After this conversion, the human class has 100 shapes, including male and female figures.



(a) Same class (b) Multiple classes

Figure 3.4: Illustration of shape descriptors

3.3.1 Find correspondence in the single class

This task is to find shape correspondence in the single class. The training and test sets are both taken from the human class, which has 100 shapes with 6890 points. These shapes are randomly separated into two sets. One is the training set, which contains 80 shapes. The other is the test set, which contains 20 shapes. The corresponding points are assigned to the same label and these labels are automatically distributed to a series of numbers, from 1 to 6890. Before the training, the input data are assembled by concatenating the geometry vector (150- dimensional vectors), HKS (101-dimensional vectors) and SHOT descriptors

(544-dimensional vectors), and the total dimension is 795. In Figure 3.4a, the first row shows the original shape. The second row shows the geometry vector, HKS, and the SHOT descriptor in the left, middle, and right position, respectively. In addition, the patch operator is computed based on the geometry feature of the shapes. For the preferences, the learning rate is 0.001. The training epoch is 100, and the test epoch is 50.

This task includes different experiments to display the various features of the new method. The first experiment compares the results of the new method, blended maps, functional maps, random forest, and geodesic convolution neural network with three GC layers (GCNN3), which shows that this new method can improve the performance and obtain superior results for the single class. The second experiment decomposes the new method into three techniques to show each one's impact. The first technique uses multiple shape descriptors to train the geodesic convolution neural network with three GC layers. The second technique trains the new intrinsic neural network with the geometry vector (150-dimensional vectors) as input data. The final technique uses the geometry vector (150-dimensional vectors) to train the geodesic convolution neural network with three GC layers. The third experiment displays the influence of changes in the network structure on the result and training time. It decreases the number of combined GC layers and adds the linear layer based on the complete model. The fourth experiment shows the impact of the parameter setting on the result and training time. In the parameter setting, the test epoch indicates the number of tests or iterations and the training epoch indicates the number of training times in each test phase.

3.3.2 Find correspondence in multiple classes

Query shapes from the same classes: This task is to find shape correspondence in multiple classes. The training and test sets are both taken from dog, cat and horse classes

because the shapes in these three classes have similar outlines. In the TOSCA dataset, the dog, cat, horse and wolf classes have similar outlines and each shape in theses classes has 27894, 25290, 19248 and 4344 vertices respectively. We manually adjust the point label to achieve the shape correspondence between different classes. For example, the top point on the head of a cat has the same label as the top point on the head of a dog. We equally distribute the labels to the point over the entire surface of each shape to ensure that each part has approximately the same number of points. However, this process is manual, which can cause errors of correspondence. In this task, there are 6890 labels, but the wolf class contains only 4344 vertices in each shape. Given that the wolf class contains only three shapes, we remove this class from the experimental dataset. In addition, we remove the shapes that have empty disks because these disks cause errors during the training process. Finally, there is a total of 23 shapes, which are randomly separated into the training and test sets. The training set contains 19 shapes and the test set contains 4 shapes. Each shape has 6890 points, which are assigned to 6890 labels. Before the training, the input data are assembled by connecting the geometry vector (150-dimensional vectors) and HKS (101-dimensional vectors), and the total dimension is 251. In addition, the patch operator is computed based on the geometry feature of the shapes. For the parameter setting, the learning rate is set to 0.001 and the training and test epochs are set to 100.

Query shapes from the different classes: This task is also to find shape correspondence in multiple classes. It uses the same experimental dataset with the previous task but takes the different measurements during the separation of the training and test sets. The training set contains all of the shapes from the cat and dog classes, which have 18 shapes; the testing set includes all of the shapes from the horse class, which has 5 shapes. The preparation phase follows the same steps in the previous task and this task have the same parameter setting with the previous one. In Figure 3.4b, the first row shows the original shape. The second row shows the HKS and the geometry vector in the left and right side respectively.

Chapter 4

Experiment Result

This chapter presents the final experiment results and discussions for three different tasks. The first part discusses the shape correspondence in the single class. The second part shows the results of shape correspondence in multiple classes.

4.1 Find correspondence in the single class

This part shows the results of shape correspondences in the single class. Because the human class has the largest number of models, including males and females in the different poses, this class is selected to be the experimental data. In this class, some of the shapes are from the FAUST dataset, and the remaining shapes are from TOSCA dataset. These shapes are randomly separated into the training and test sets, which have 80 shapes and 20 shapes respectively. The first experiment compares the results of the new method and previous works to demonstrate the superior performance of this new method. The second experiment shows the benefits of improvements from multiple shape descriptors and the new intrinsic neural network. The third and fourth experiments show the influence of changes

in the network structure and parameter setting on the final result and training time.

4.1.1 Method comparison

This experiment compares the results of the new method, the blended maps, functional maps, random forest and geodesic convolution neural network of three layers (GCNN3). The training and test epochs are set to 50 and 100 respectively and the learning rate is set to 0.001. The new method concatenates the HKS (101-dimensional vectors), geometry vector(544-dimensional vectors) and SHOT descriptor (150-dimensional vectors) to train the redesigned intrinsic neural network. The combined GC layer is a GC layer with a dropout layer, an angular max-pooling layer and a batch normalization layer, in sequence. The input data go through the input layer, and three combined GC layers in order. The outputs of these four layers are placed into the concatenating layer and softmax layer to obtain the final probability distribution. The result of shape correspondence is as follows: blended maps, 6.78%; functional map, 31.23%; random forest, 18.67%; GCNN3, 60.27%; and our method, 73.61%. In Table 4.1, our method clearly has a higher accuracy than the other methods. This finding proves that our architecture has the sufficient ability to accomplish shape correspondence. Relative to GCNN3, the new structure improves the result by 13.34%. Figure 4.1 shows the complete process and the neural network architecture. Figure 4.2a, 4.2b, 4.2c and 4.2d illustrate three shape descriptors (the geometry vector, SHOT descriptor, HKS) and the final output result. The geometry vector can indicate the coarse correspondence of the whole body, but it cannot clearly distinguish each part, such as the left and right hand, shank, and thigh. In contrast, the SHOT descriptor can accommodate the detailed sections. For example, the model of the SHOT descriptor has the asymmetric distribution of spots with a distinct color depth in the body. In addition, the stripe on the left leg is distinct from the stripe on the right leg. Relative to the geometry vector and SHOT

descriptor, the HKS is better at determining main parts, such as hands and feet, which helps refine the classification. The model of final output result shows that the new method can distinguish symmetric parts and make good correspondence of shapes.

Method	Blended Maps	Functional Map	Random Forest	GCNN3	Our Method
Precision	6.78 %	31.23 %	18.67 %	60.27 %	73.61 %

Table 4.1: Experimental results of the method comparison in the identical class

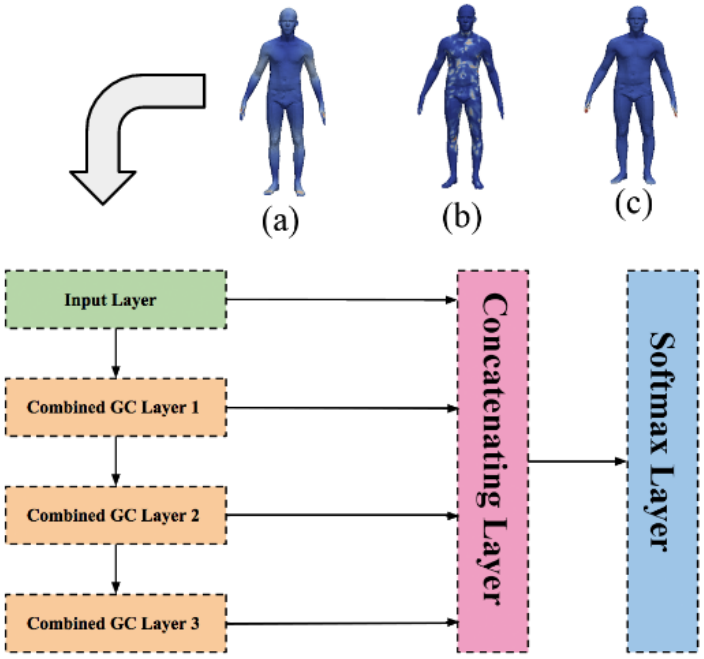
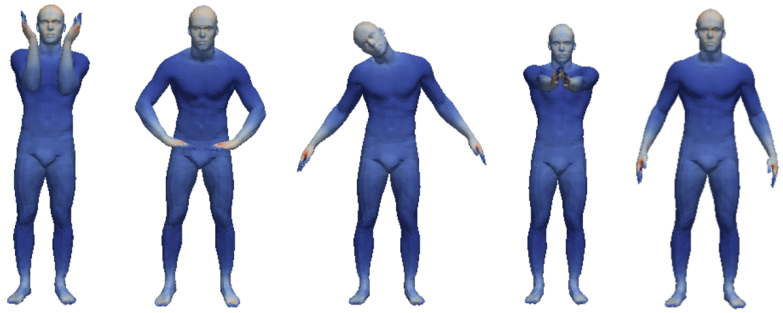
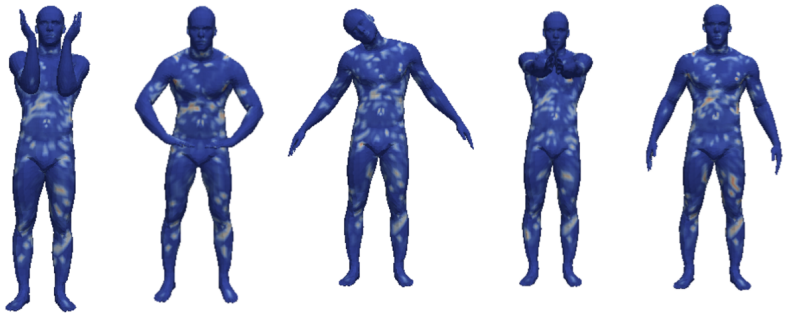


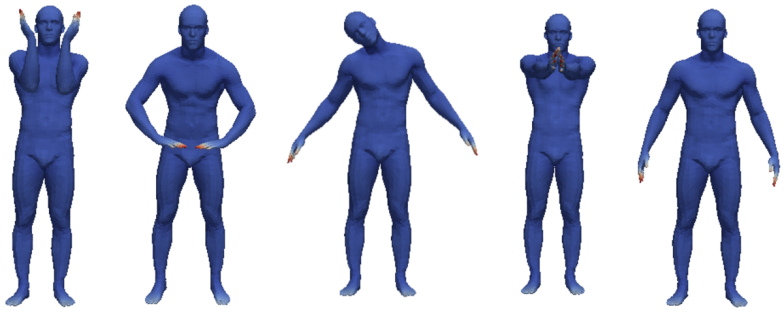
Figure 4.1: Illustration of the complete neural network architecture, (a) represents the geometry vector; (b) represents the SHOT descriptor. (C) represents the HKS. (a)(b)(c) are the input data and traverse the input layer, three combined GC layers, the concatenating layer and softmax layer.



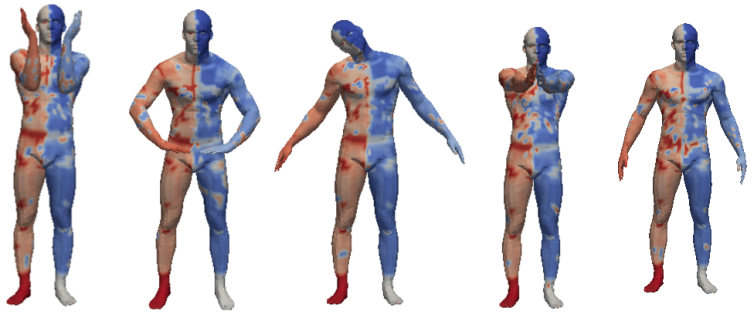
(a) Geometry vector



(b) SHOT descriptor



(c) HKS



(d) Results of the complete model

Figure 4.2: Illustration of shape descriptors and final result

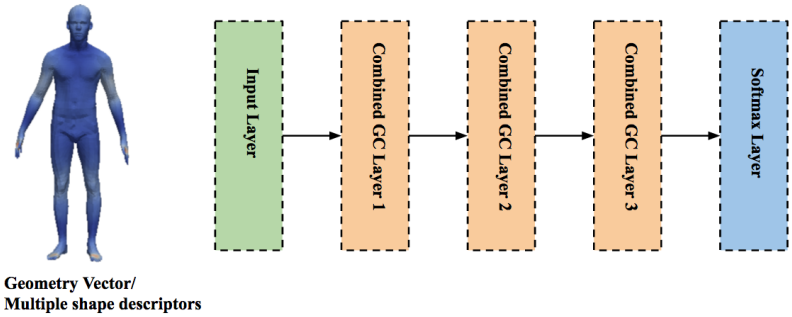


Figure 4.3: Illustration of the architecture without the concatenating layer

4.1.2 Technique comparison

This experiment decomposes the new method into different parts to show the effect of each improvement. For the complete method, we use multiple shape descriptors to train the new intrinsic network architecture. For the method without multiple descriptors, we use only the geometry vector (150-dimensional vectors) as the input data to train the new network. For no concatenating layer, we use multiple shape descriptors as input data but remove the concatenating layer from the complete model. Therefore, the input data goes through the input layer and three combined GC layers and is then directly processed by the softmax layer. For the method without a concatenating layer and multiple descriptors, we not only remove the concatenating layer but also change the input data, which is only the geometry vector (150-dimensional vectors). The results show the impact of each change on the shape correspondence as follows: the complete method, 73.61%; the method without multiple descriptors, 69.56%; the method with no concatenating layer, 67.48%; and the multiple descriptors with no concatenating layer, 60.89%. The method of multiple shape descriptors provides a 6.59% improvement, and the method of concatenating layer provides a 8.67% improvement. Relative to the method of concatenating layer, the accuracy of the method of multiple shape descriptors is decreased by 2.08%, which shows that the concatenating layer can preserve more information than multiple shape descriptors. The reason could be that

the information loss in the training process is more extensive than the loss in extracting the shape descriptors. The information becomes concentrate through the filter of each layer, but these filters also remove some useful data at the same time.

Method	Complete Model	No Multiple Descriptors	No Concatenating Layer	No Concatenating Layer & Multiple Descriptors
Precision	73.61 %	69.56 %	67.48 %	60.89 %

Table 4.2: Experimental results of the improvement comparison

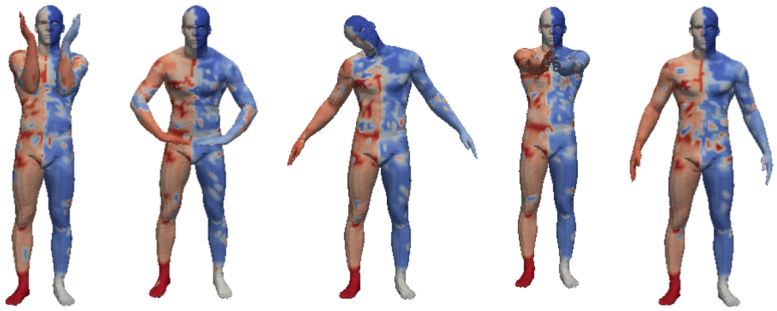


Figure 4.4: Results of the method without multiple shape descriptors

4.1.3 Architecture and parameter comparison

This experiment shows that the different layers have different effects on the accuracy of the shape correspondence. The method with 2 combined GC layers removes the third GC layer and the method with 1 combined GC layer removes the second and third layer based on the complete model. The method with the input linear layer adds the linear layer between the input and the first combined GC layer. This linear layer is referred to as the input linear layer to distinguish it from the linear layers between the combined GC layer and the concatenating layer, which is referred to as the adjusted linear layer. Other parameters (e.g., the testing epoch, training epoch, and learning rate) are set to be the same with the previous experiment. The accuracy of the shape correspondences is as follows: the complete

method, 73.61%; the method of 2 combined GC layers, 56.22%; the method of 1 combined GC layer, 40.09%; the method with the input linear layer, 73.59%; the method with the adjusted linear layer, 71.66%. Relative to the complete model, the accuracy of the method with 2 combined GC layers is decreased by 17.39 and the accuracy of the method with 1 combined GC layers is decreased by 33.52, which shows the significance of the GC layer. Additionally, the accuracy of the method with the input linear layer decreases by 0.02% and the accuracy of the method with the adjusted linear layer decreases by 1.95%. These differences can be ignored because of the fluctuation in the training process. Therefore, the linear layer does not have a serious impact on the prediction.

Method	Complete Model	2 Combined GC Layers	1 Combined GC Layer	Input Linear Layer	Adjusted Linear Layer
Precision	73.61 %	56.22 %	40.09 %	73.59 %	71.66 %

Table 4.3: Experimental results of the architecture comparison

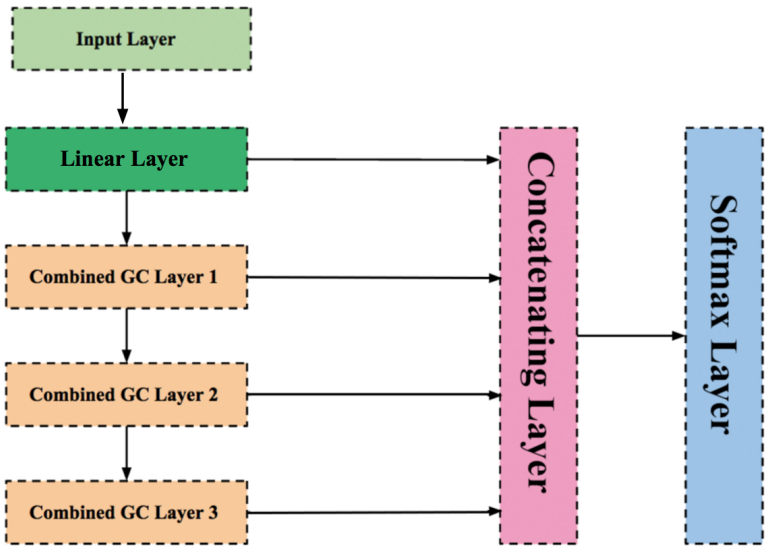


Figure 4.5: Illustration of the network with the input linear layer

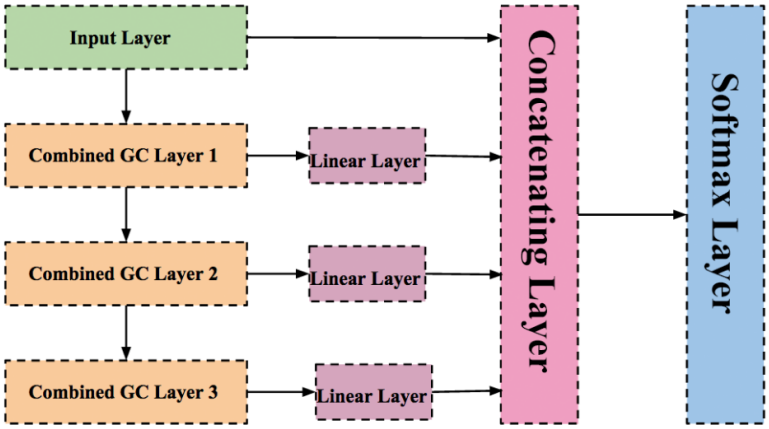


Figure 4.6: Illustration of the network with the adjusted linear layer

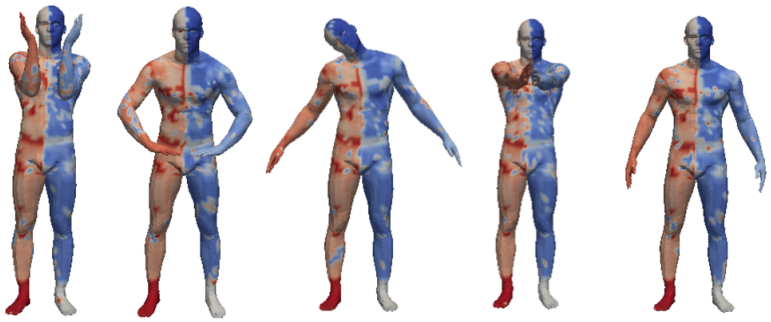
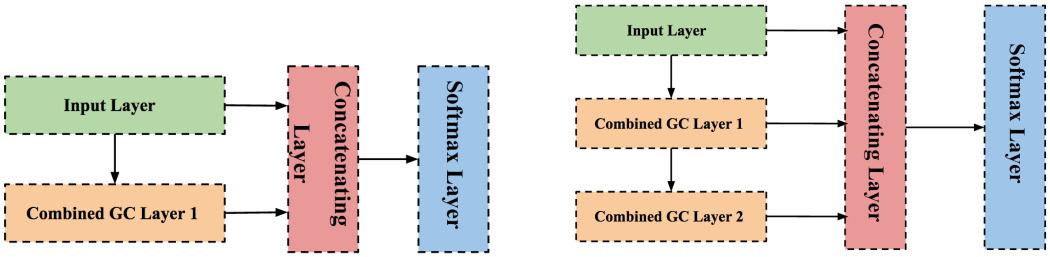


Figure 4.11: Result of the method with the adjusted linear layer

Additionally, the parameter setting has the impact on the final result. The test epoch determines how many times the model uses the testing set to test the model and the training epoch determines the number of times between the two test phases. In theory, the large training and test epochs can enhance the accuracy of the final result, but this training process also can be time-consuming. When the training and test epochs are 50, the precision is 69.5%, and the required time is approximately 6 hours. When the test epoch is 50 and the training epoch is 100, the precision is 72.66%, and the time is approximately 10 hours. When the training epoch is 50 and the test epoch is 100, the precision is 71.45%, and the time is approximately 11 hours. When the training and test epochs are 100, the precision is 73.14%, and the time is approximately 20 hours. The uncertainty of the neural network



(a) 1 combined GC layer model (b) 2 combined GC layer model

Figure 4.7: Illustration of the network with 1 or 2 combined GC layers

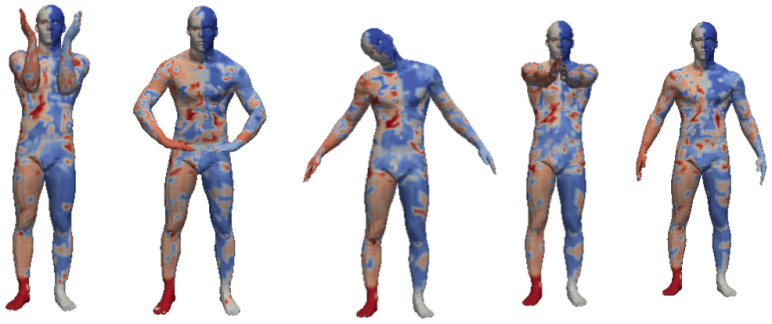


Figure 4.8: Results of the network with 1 combined GC layer

could cause a fluctuation in the result and the condition of the hardware could affect the training time. In Table 4.4, the precision and time increase with a larger test epoch and training epoch. However, the increment of the test epoch does not lead to the same consequence of the rise in the training epoch. In addition, the rise in the time is not linear with the increment of test or training epoch. According to the trade-off between time and precision, the test epoch is set to 50 and the training epoch is set to 100.

Method	Test Epoch=50 Train Epoch=50	Test Epoch=50 Train Epoch=100	Test Epoch=100 Train Epoch=50	Test Epoch=100 Train Epoch=100
Precision	69.50 %	73.61 %	71.45 %	73.14 %
Time	6+ hours	10+ hours	11+ hours	20+ hours

Table 4.4: Experimental results of the parameter comparison

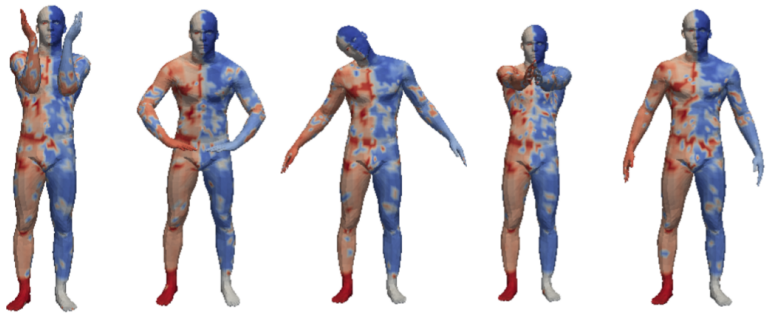


Figure 4.9: Results of the network with 2 combined layers

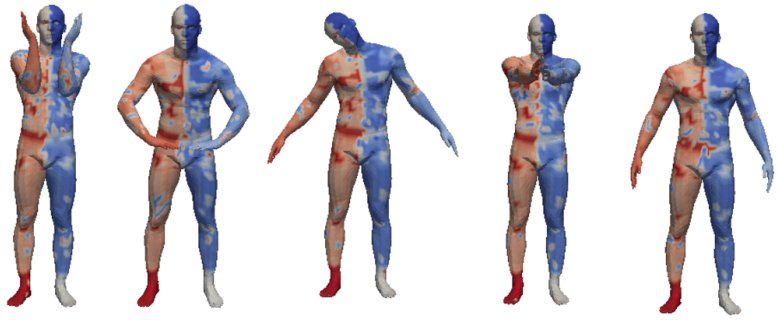


Figure 4.10: Result of the method with the input linear layer

4.2 Find correspondence in multiple classes

This section shows the experiments that find shape correspondences in multiple classes. For the first task, the training and test set are both taken from the cat, dog and horse classes. For the second task, all of the shapes in the cat and dog class are used as the training data and all of the shapes in the horse class are used as the test data. The learning rate is set to 0.001, and the training and testing epochs are set to 100.

In this task, the experiment dataset includes 23 shapes from the cat, dog, and horse classes. The training set has 19 shapes, and the test set has 4 shapes, including 2 cat shapes, 1 dog shape and 1 horse shape. The accuracy of the shape correspondence is as follows: blended maps, 6.15%; functional map, 30.22%; random forest, 17.29%; GCNN3, 45.23%, and our method, 60.82%. In the experiment, we find that the result could change significantly even

with the same training and test sets. In addition, different query shapes also significantly affect the final accuracy. In light of this finding, we randomly selected shapes from the experiment dataset to build five test sets and ensure that each test set has 2 cat shapes, 1 dog shape and 1 horse shape. In addition, we repeated experiment three times for these five test sets and averaged the accuracy to obtain the final result. The result shows that our method has better performance in shape correspondences among multiple classes.

Method	Blended Maps	Functional Map	Random Forest	GCNN3	Our Method
Precision	6.15 %	30.22 %	17.29 %	45.23 %	60.82 %

Table 4.5: Experimental results of the method comparison in the multiple classes

Figure 4.12: Geometry vector. The right shape of the cat/dog shows the geometry vector when the dimension is 1, the left shape shows the geometry vector when the dimension is 100.

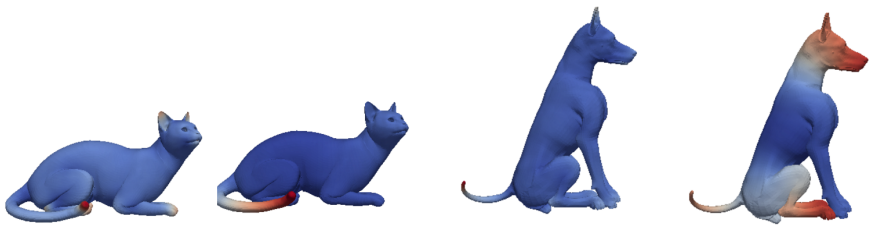


Figure 4.13: HKS. The right shape of cat/dog shows the HKS when the dimension is 1, the left shape shows the HKS when the dimension is 100.

Figure 4.14: Results of our method in the cat class

Figure 4.15: Results of our method in the dog class

In the second part, the categories of training shapes are different from the categories of test shapes. We choose one class as our test set and the remaining classes as the training set. In the experiment, the horse class is used as the test set, which contains 5 shapes. The cat and dog classes are used as the training set, which contains 18 shapes. However, the accuracy of the shape correspondence is not satisfiable. The reason could be the inadequate training. The model is not comprehensive in performing shape correspondence across different classes. In addition, even though we equally distribute the labels to each point of the shape, this manual process can cause errors of correspondence. Therefore, all of these reasons could lead to inaccuracy in the final result.

Chapter 5

Conclusion and future work

This paper provides a novel method that offers outstanding performance relative to previous works in shape correspondence. This proposed method employs multiple shape descriptors as the input data, which concatenates different shape descriptors to increase the input data richness. It also redesigns the intrinsic neural network architecture based on the geodesic convolution neural network. This redefined intrinsic neural network removes the linear layers, covariance layer and FTM layer. In addition, it introduces the dropout layer, batch normalization layer and concatenating layer. The dropout layer randomly disconnects two neurons to prevent overfitting. The batch normalization layer reduces internal covariate shift to accelerate the training process. The concatenating layer decreases the loss of information by connecting the outputs from geodesic convolution layers. The experiments indicate that the proposed method provides much stronger predictions than other methods.

However, there are still many areas to be improved in the future. Even though the proposed method shows outstanding performance, we need to make a strong assumption that the training and test sets belong to the same classes. Experiments show that the proposed method works comparatively worse when the categories of query shapes are different from

the categories of reference shapes. One possible solution is to add other traditional shape descriptors into input data to enrich the training data. Another solution is to increase the complexity of the network architecture by adding more geodesic convolution layers. Given appropriate dropout ratio, in practice, we do not detect obvious overfitting problem when increasing model complexity. Moreover, the proposed method requires 20 hours for 100 steps training using one K80 GPU. This training efficiency can be improved in the future. For example, we can adjust the parameter in the computing patch to reduce the number of rings and rays, which might be one possible solution to reduce the computation cost and training time.

Bibliography

[1] Yonathan Aflalo and Ron Kimmel. Spectral multidimensional scaling. *Proceedings of the National Academy of Sciences*, 110(45):18052–18057, 2013.

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

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

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

[5] Davide Boscaini, Jonathan Masci, Emanuele Rodolà, and Michael M Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. *arXiv preprint arXiv:1605.06437*, 2016.

[6] Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. *Numerical geometry of non-rigid shapes*. Springer Science & Business Media, 2008.

[7] John Crank. *The mathematics of diffusion*. Oxford university press, 1979.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[10] Jürgen Jost. *Riemannian geometry and geometric analysis*. Springer Science & Business Media, 2008.

[11] Ron Kimmel and James A Sethian. Computing geodesic paths on manifolds. *Proceedings of the National Academy of Sciences*, 95(15):8431–8435, 1998.

[12] Iasonas Kokkinos, Michael M Bronstein, Roee Litman, and Alex M Bronstein. Intrinsic shape context descriptors for deformable shapes. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 159–166. IEEE, 2012.

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

[14] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[15] Marius Leordeanu and Martial Hebert. A spectral technique for correspondence problems using pairwise constraints. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1482–1489. IEEE, 2005.

[16] Yangyan Li, Hao Su, Charles Ruizhongtai Qi, Noa Fish, Daniel Cohen-Or, and Leonidas J Guibas. Joint embeddings of shapes and images via cnn image purification. *ACM Trans. Graph*, 5, 2015.

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

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

[19] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Shapenet: Convolutional neural networks on non-euclidean manifolds. Technical report, 2015.

[20] Joel Max. Quantizing for minimum distortion. *IRE Transactions on Information Theory*, 6(1):7–12, 1960.

[21] Facundo Mémoli and Guillermo Sapiro. A theoretical and computational framework for isometry invariant recognition of point cloud data. *Foundations of Computational Mathematics*, 5(3):313–347, 2005.

[22] Jonathan Pokrass, Alexander M Bronstein, Michael M Bronstein, Pablo Sprechmann, and Guillermo Sapiro. Sparse modeling of intrinsic correspondences. In *Computer Graphics Forum*, volume 32, pages 459–468. Wiley Online Library, 2013.

[23] Emanuele Rodola, Alex M Bronstein, Andrea Albarelli, Filippo Bergamasco, and Andrea Torsello. A game-theoretic approach to deformable shape matching. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 182–189. IEEE, 2012.

[24] Emanuele Rodolà, Luca Cosmo, Michael M Bronstein, Andrea Torsello, and Daniel Cremers. Partial functional correspondence. In *Computer Graphics Forum*. Wiley Online Library, 2016.

[25] Yusuf Sahillioğlu and Yücel Yemez. Minimum-distortion isometric shape correspondence using em algorithm. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2203–2215, 2012.

[26] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[27] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, volume 28, pages 1383–1392. Wiley Online Library, 2009.

[28] Art Tevs, Alexander Berner, Michael Wand, Ivo Ihrke, and H-P Seidel. Intrinsic shape matching by planned landmark sampling. In *Computer Graphics Forum*, volume 30, pages 543–552. Wiley Online Library, 2011.

[29] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *European conference on computer vision*, pages 356–369. Springer, 2010.

[30] Shinji Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE transactions on pattern analysis and machine intelligence*, 10(5):695–703, 1988.

[31] Oliver Van Kaick, Hao Zhang, Ghassan Hamarneh, and Daniel Cohen-Or. A survey on shape correspondence. In *Computer Graphics Forum*, volume 30, pages 1681–1707. Wiley Online Library, 2011.