Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Image-based 3D model retrieval using manifold learning*

Pan-pan MU†[1], San-yuan ZHANG[1], Yin ZHANG[1], Xiu-zi YE[2], Xiang PAN[3]

[1]*College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China*

[2]*College of Mathematics and Information Science, Wenzhou University, Wenzhou 325003, China*

[3]*College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China*

†E-mail: mupanpan0927@zju.edu.cn

Received Dec. 1, 2016; Revision accepted May 22, 2017; Crosschecked Nov. 27, 2018

Abstract: We propose a new framework for image-based three-dimensional (3D) model retrieval. We first model the query image as a Euclidean point. Then we model all projected views of a 3D model as a symmetric positive definite (SPD) matrix, which is a point on a Riemannian manifold. Thus, the image-based 3D model retrieval is reduced to a problem of Euclid-to-Riemann metric learning. To solve this heterogeneous matching problem, we map the Euclidean space and SPD Riemannian manifold to the same high-dimensional Hilbert space, thus shrinking the great gap between them. Finally, we design an optimization algorithm to learn a metric in this Hilbert space using a kernel trick. Any new image descriptors, such as the features from deep learning, can be easily embedded in our framework. Experimental results show the advantages of our approach over the state-of-the-art methods for image-based 3D model retrieval.

Key words: Model retrieval; Euclidean space; Riemannian manifold; Hilbert space; Metric learning
https://doi.org/10.1631/FITEE.1601764 **CLC number:** TP391

1 Introduction

Distinct from traditional text retrieval, three-dimensional (3D) shapes lack semantic labels, but contain rich semantics. The text-based retrieval approach is unsuitable for 3D model retrieval. In the past decades, there has been a lot of research about instance-based 3D model retrieval. However, users must possess a 3D model before the retrieval, which is impractical. Many new studies have tried to use sketches as a search interface, on which users draw raw sketches using a simple sketchpad, and then retrieve the required 3D models. However, this method does not make full use of existing massive images and videos. In contrast, the image as a 3D model retrieval interface is very simple to use, and new algorithms in the field of computer vision can be easily applied.

In this study, we propose a new framework for image-based 3D model retrieval. First, we choose the image descriptors for images and rendered views. Then we can describe a query image as a point in the Euclidean space, and all projected views as a point set. To improve the matching efficiency and take advantage of the geometry of rendered views, we model the point set of one 3D shape as a symmetric positive definite (SPD) matrix. As known, all SPDs of 3D shapes reside on a Riemannian manifold. However, there is a great gap between the query image as a Euclidean point and SPDs on a Riemannian manifold. We can map the Euclidean space and Riemannian manifold to the same high-dimensional Hilbert space to facilitate the matching between these two heterogeneous spaces. This SPD Riemannian manifold has been well studied recently, and with kernel methods,

* Project supported by the National Key R&D Program of China (No. 2017YFB1002600), the National Natural Science Foundation of China (No. 61272304), the Natural Science Foundation of Zhejiang Province, China (Nos. LQ16F020007 and LQ17F030002), and the Natural Science Foundation of Ningbo, China (No. 2017A610108)

ORCID: Pan-pan MU, http://orcid.org/0000-0001-9224-662X

we can encode the geometry of the manifold in Riemannian-to-Hilbert space mapping.

The main contributions of this study are as follows: (1) We treat all rendered views of a shape as a whole, and model the whole as an SPD matrix that resides on a Riemannian manifold. Thus, the query image becomes a point in the Euclidean space, and all project views constitute a point set. In this way, the point-to-set matching problem is reduced to point-to-point matching. (2) We map the query points in the Euclidean space and SPD matrices on the manifold to the same high-dimensional Hilbert space, which greatly shrinks the semantic gap between the Euclidean and SPD manifold. During mapping, we encode the geometry in the SPD Riemannian manifold, thus avoiding the mismatching problems in previous studies to some extent. (3) We introduce a new image-based 3D model retrieval framework that is very different from the traditional bag-of-features framework. This framework is flexible enough in allowing the image descriptors given in previous studies to be imported without modification.

2 Related work

Our work concerns how to combine the rendered views of 3D shapes to produce a compact representation of these shapes (Mu et al., 2017). With rapid progress in computer vision in recent years, many researchers have tried to import the new techniques of image processing, such as deep learning, into 3D shape retrieval. Our work is also inspired by point-to-set matching in the computer vision field. In this section, we first review the related studies on point-to-set matching. Then we show the recent progress in multiview-based shape retrieval.

2.1 Point-to-set matching

In many metric learning methods, such as large margin nearest neighbor (LMNN), the distance between two points needs to be computed. However, in many computer vision tasks, such as video-based face recognition, pedestrian detection, and object categorization, the distance between an image (point) and an image set (point set) is required.

Many studies model the point set as a linear subspace of the Euclidean space. In Yamaguchi et al. (1998), an input face was represented by a sequence of input images, and a subspace was formed from the temporal image sequences. Then the mutual subspace method (MSM) was applied, in which the similarity was defined by the angle between the input and the reference subspaces. In Chien and Wu (2002), the novel nearest feature plane (NFP) and nearest feature space (NFS) classifiers were presented to detect the most likely identity of the query image. They achieved this by searching the nearest distance to feature planes and feature spaces spanned by prototype feature points. In Kim et al. (2007), discriminative canonical correlation (DCC), which could be considered the angles between two linear subspaces, was proposed, and the benefits of canonical correlations of linear subspaces were explained and evaluated.

There are several studies that model the point set as an affine hull. In Vincent and Bengio (2001), a K-local hyperplane distance nearest neighbor (HKNN) algorithm was proposed, in which every hyperplane was formed by the hull of K points of the K-c neighborhood of the test point. In Cevikalp and Triggs (2010), every test and training example of an individual face was a set of images. They represented images as points in a linear or affine feature space, and characterized each image set by a convex geometric region (the affine or convex hull) spanned by its feature points. In Zhu et al. (2013), an image set could be modeled as an affine hull, a reduced affine hull, a convex hull, or a reduced convex hull, which was a subspace spanned by all the available samples in the set.

In recent years, many studies have tried to model the point set as a covariance matrix. In Wang et al. (2012) and Huang et al. (2014), each image set was represented by its natural second-order statistics, i.e., covariance matrix, and the image set classification problem was formulated as classifying points lying on a Riemannian manifold spanned by SPD matrices. Vemulapalli et al. (2013) proposed a general framework for developing extrinsic classifiers for features such as linear subspace and covariance features, which follows the multiple kernel learning (MKL) approach and parameterizes the kernel as a linear combination of known base kernels.

After modeling the image set as a point on some Riemannian manifold, the point-to-set matching

problem is reduced to computation of the distance between the point in the Euclidean space and the point on the Riemannian manifold. Recently, there has been much progress in kernel-based metric learning methods (Vemulapalli et al., 2013) on the Riemannian manifold, specifically, in the mapping approach for point-to-set matching.

2.2 Three-dimensional model retrieval

We are concerned about how to combine the descriptors of rendered views of 3D shapes to produce a compact representation of these 3D shapes. There is extensive research on view-based 3D shape retrieval.

Princeton University has a long research history in 3D shape analysis, including 3D shape retrieval. The early publicly available benchmark (Shilane et al., 2004) greatly advanced the progress in 3D shape retrieval. Wu et al. (2015) recently released a comprehensive clean collection of 3D CAD models. Many recent studies on view-based shape retrieval have involved comparisons on this dataset.

One of the early studies on view-based shape retrieval is the 'LightField descriptor' (Chen et al., 2003). The authors first extracted Zernike moments and Fourier descriptors from the view images. Then they constructed a set of LightField descriptors to represent a 3D model, and these descriptors were robust to rotation of 3D models. However, the procedure for constructing the LightField descriptors was complex, and the image descriptors were not very expressive.

Inspired by the success of the SIFT descriptor in image processing, Ohbuchi et al. (2008) computed a set of two-dimensional (2D) multi-scale local visual features for each rendered view image. Finally, they used the bag-of-features architecture to compute the distances between 3D models.

Lian et al. (2013) also extracted SIFT descriptors from each rendered view image and used the bag-of-features architecture. However, they employed clock matching to measure the dissimilarity between two models, which was very similar to the LightField descriptor.

In the early days, researchers designed image descriptors manually, but they encountered problems in view-based shape retrieval. However, the recent success in deep learning has revived this field.

Su et al. (2015) first demonstrated the effectiveness of view-based 3D shape retrieval in the multiview convolutional neural network. When the recently developed deep learning architecture was adopted, the related 3D shapes can be recognized, even in a single rendered view, with an accuracy far higher than that achieved using conventional 3D shape descriptors. Then Su et al. (2015) designed a novel deep learning architecture in which the information from multiple views of a 3D shape is combined into a single and compact shape descriptor, offering even better recognition performance. Similarly, our work can produce a compact shape descriptor for each 3D shape.

Bai et al. (2016) proposed a real-time and scalable 3D shape search engine. They first used graphic processing unit (GPU) acceleration to render 3D shapes and extract view descriptors. Then they used two inverted files to speed up multiview matching and conduct efficient context-based reranking.

Bai et al. (2015) proposed a two-layer coding framework to conduct shape matching. The spatial relationship of each view pair was captured at the first layer; different codewords were encoded according to this relationship at the second layer.

In Tabia et al. (2014), a novel method for 3D shape analysis was proposed using the covariance matrices of the descriptors rather than the descriptors themselves. They studied covariance matrices in their native space and used geodesic distances on the manifold as a dissimilarity measure. In this study, we apply the covariance matrices of image descriptors using a very different approach. We map the Euclidean space and the Riemannian manifold to the same high-dimensional Hilbert space, to facilitate the matching between these two heterogeneous spaces.

In other words, by using metric learning from the Riemannian manifold to the Euclidean space, our work can produce a compact vector for each 3D shape. The recently proposed deep learning algorithms can also be embedded into our framework.

3 Background

In this section, we introduce some Riemannian manifolds that are frequently used in the computer vision field, and then show the fundamentals of

mapping Riemannian manifolds to the reproducing kernel Hilbert space.

3.1 Riemannian manifolds

As mentioned above, we need to model the image sets as points on a manifold. The image sets can be represented as linear subspaces, affine linear subspaces, and symmetric positive matrices, lying on a Grassmann manifold, an affine Grassmann manifold, and a Sym_d^+ manifold, respectively. These manifolds have their respective metrics, which can be used to measure the distance between two points.

The Grassmann manifold (Hamm and Lee, 2008) is composed of linear subspaces of fixed dimensionality in some Euclidean space, such as $g(n, d)$. The Grassmann manifold represents all n-dimensional linear subspaces in $\mathbb{R}^d$. As known, each n-dimensional linear subspace in $\mathbb{R}^d$ can be represented by an orthogonal matrix $\boldsymbol{U}\in\mathbb{R}^{d\times n}$. For two points $\boldsymbol{U}_i$ and $\boldsymbol{U}_j$, there is a famous projection metric to measure their distance:

$$d(\boldsymbol{U}_i,\boldsymbol{U}_j)=\sqrt{2}\left\|\boldsymbol{U}_i\boldsymbol{U}_i^{\mathrm{T}}-\boldsymbol{U}_j\boldsymbol{U}_j^{\mathrm{T}}\right\|_{\mathrm{F}}. \tag{1}$$

The affine Grassmann manifold is an extension of the Grassmann manifold; in other words, it is composed of affine subspaces. Thus, each point $\boldsymbol{A}$ on the affine Grassmann manifold can be represented by an affine span, which is defined from the $d\times n$ orthogonal matrix $\boldsymbol{U}\in\mathbb{R}^{d\times n}$ and an offset $\boldsymbol{\mu}\in\mathbb{R}^n$. Using the metric in Hamm and Lee (2009), the distance between every two points $\boldsymbol{A}_i$ and $\boldsymbol{A}_j$ is

$$\begin{aligned}d(\boldsymbol{A}_i,\boldsymbol{A}_j)=\sqrt{2}\Big(&\left\|\boldsymbol{U}_i\boldsymbol{U}_i^{\mathrm{T}}-\boldsymbol{U}_j\boldsymbol{U}_j^{\mathrm{T}}\right\|_{\mathrm{F}}\\&+\left\|(\boldsymbol{I}-\boldsymbol{U}_i\boldsymbol{U}_i^{\mathrm{T}})\boldsymbol{\mu}_i-(\boldsymbol{I}-\boldsymbol{U}_j\boldsymbol{U}_j^{\mathrm{T}})\boldsymbol{\mu}_j\right\|_{\mathrm{F}}\Big),\end{aligned} \tag{2}$$

where $\boldsymbol{I}$ is an identity matrix.

An SPD matrix of dimension $d\times d$, when endowed with an appropriate metric (Jayasumana et al., 2013), can form a Riemannian manifold Sym_d^+. For every two points $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$, when choosing a log-Euclidean metric, the distance between them is

$$d(\boldsymbol{S}_i,\boldsymbol{S}_j)=\left\|\log\boldsymbol{S}_i-\log\boldsymbol{S}_j\right\|_{\mathrm{F}}. \tag{3}$$

3.2 Mapping the Riemannian manifold to the Hilbert space

The kernel method is very effective and can be combined with many traditional methods to explore nonlinear patterns, such as support vector machine (SVM). It intrinsically maps data to a high-dimensional feature space that is linearly separable.

From the Moore-Aronszajn theorem, we can extend the kernel method to any set. For every SPD kernel $k:(M\times M)\to\mathbb{R}$ on manifold M, there is a reproducing kernel Hilbert space (RKHS), H, whose inner product is defined by this kernel. Although there are infinitely many associated feature maps ψ: $M\to H$ that satisfy $k(x, y)=(\psi(x), \psi(y))$, we choose the one that maintains optimization constraints and geometric structure on the manifold.

4 Framework overview

The framework of our method includes two stages: offline and online (Fig. 1). In the offline stage, we render every model in the 3D model database from several directions to obtain its views. We extract features for every view. All the features of one 3D model can be constructed as one SPD. All the SPDs

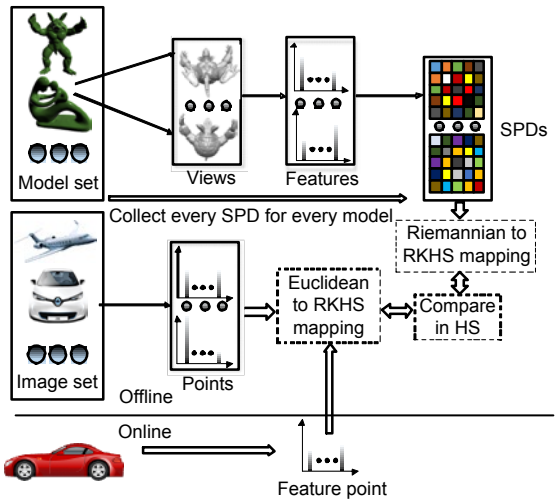

Fig. 1 Framework of our method

At the offline stage, for every 3D model, we obtain project views, extract features, and construct SPDs. For every image, we extract features as points. Using a kernel trick, we first map them to their own RKHS, then to the same Hilbert space. At the online stage, we first extract the features of the query image, and then retrieve 3D models in the above Hilbert space

lie on a Riemannian manifold. We extract features for every image in the image database and all the features lie naturally in the Euclidean space. Because all the 3D models and images are labeled, using kernel methods, we can project the Riemannian manifold and the Euclidean space into the same RKHS. In the online stage, we first extract features for every query image. Then we compare the mapped point in the RKHS with Euclidean-to-RKHS mapping. Finally, we rank the related 3D models.

5 Descriptor generation

Let $\boldsymbol{S}=[\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n]$ be the view matrix of a 3D model, where $\boldsymbol{s}_i \in \mathbb{R}^d$ denotes the i^{th} view image with a d-dimensional feature descriptor, such as the raw intensity of the image. The SPD can be represented as a $d\times d$ covariance matrix (Wang et al., 2012):

$$\boldsymbol{y}=\frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{s}_i-\overline{\boldsymbol{s}})(\boldsymbol{s}_i-\overline{\boldsymbol{s}})^{\mathrm{T}}, \tag{4}$$

where $\overline{\boldsymbol{s}}$ is the mean of all view image descriptors. In this covariance matrix, the diagonal entries represent the variance of image features, and other entries reflect the correlations of different features.

Representing the view images of a 3D model as an SPD has many advantages: (1) The form is simple enough, and thus it can be computed very quickly. (2) The covariance matrix, as a second-order statistic, places no constraints on the distribution of features, the number of features, or the types of features. (3) SPD encodes correlation information for all the features, so it is very effective in discriminating the image sets of different classes. In contrast, previous methods such as the linear subspace have discarded the correlation information. (4) In the computation of the SPD, the noise samples are filtered, and thus this method becomes very robust by accumulating information from all the samples.

6 Problem formulation

This section follows the work by Huang et al. (2014). In Fig. 1, the image-based 3D model retrieval problem is reduced to matching Euclidean points with Riemannian points. In this section, we denote Euclidean points by $\boldsymbol{X}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_m\}\subset\mathbb{R}^d$, and Riemannian points by $\boldsymbol{Y}=\{\boldsymbol{y}_1,\boldsymbol{y}_2,\cdots,\boldsymbol{y}_n\}\subset\mathcal{M}$, where $\boldsymbol{y}_i$ is as given in Eq. (4). The Euclidean points and Riemannian points are all labeled, denoted as $\{l_1^x, l_2^x, \cdots, l_m^x\}$ and $\{l_1^y, l_2^y, \cdots, l_n^y\}$, respectively.

Because there is a huge semantic gap between the Euclidean and Riemannian spaces, we need to map them to the same Hilbert space by transformations f and φ, respectively. We thus obtain two maps by training the labeled Euclidean and Riemannian points. Finally, given $\boldsymbol{x}_i$ from the Euclidean space and $\boldsymbol{y}_j$ on the Riemannian manifold, their distance $d(\boldsymbol{x}_i, \boldsymbol{y}_j)$ is reduced to an inner product in the Hilbert space:

$$d(\boldsymbol{x}_i,\boldsymbol{y}_j)=\sqrt{(f(\boldsymbol{x}_i)-\varphi(\boldsymbol{y}_j))^{\mathrm{T}}(f(\boldsymbol{x}_i)-\varphi(\boldsymbol{y}_j))}. \tag{5}$$

The huge gap between the Euclidean and Riemannian spaces cannot be resolved by simply embedding them in some space. Inspired by the RKHS of differential geometry, we first embed the Riemannian manifold in its RKHS using the kernel trick. There is much research on the Riemannian kernel (Wang et al., 2012), and by choosing the kernel wisely, we can preserve the Riemannian geometry in this RKHS. By mapping the Euclidean space and the Riemannian manifold to their own RKHS, we greatly shrink the gap, and the computation of the distance becomes very simple and fast. Finally, we further map the two RKHSs to the same Hilbert space.

In Fig. 2, the features extracted from the image set as points reside in $\mathbb{R}^d$, and the SPDs as points reside on manifold $\mathcal{M}$. The transformation φ_x: $\mathbb{R}^d\rightarrow\mathcal{H}_x$ facilitates richer representations for the original Euclidean points, and the mapping φ_y: $\mathcal{M}\rightarrow\mathcal{H}_y$ maintains the geometric metric of the Riemannian manifold in the RKHS. Additionally, the two linear maps f_x: $\mathcal{H}_x\rightarrow\mathcal{H}$ and f_y: $\mathcal{H}_y\rightarrow\mathcal{H}$ transform RKHS to the common Hilbert space. So, the two composite mappings $f=f_x\circ\mathcal{H}_x$ and $\varphi=f_y\circ\mathcal{H}_y$ are the required transformations.

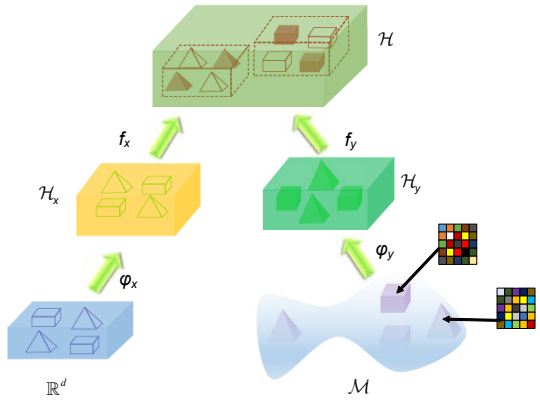

Fig. 2 Mapping hierarchy

$\mathbb{R}^d$ is the Euclidean space, and $\mathcal{M}$ is the Riemannian manifold. $\mathcal{H}_x$ and $\mathcal{H}_y$ are RKHSs, and $\mathcal{H}$ is the final Hilbert space. φ_x/φ_y and f_x/f_y are nonlinear and linear transformations, respectively. The wired shapes originate from the Euclidean space and the solid shapes originate from the Riemannian manifold. Different shapes represent different classes

6.1 Objective function

For the manifold learning problem, we formulate an optimization algorithm to resolve the two transformations. The objective function is as follows:

$$\min_{f,\varphi} D(f,\varphi)+\lambda_1 G(f,\varphi)+\lambda_2 T(f,\varphi), \tag{6}$$

where $D(f, \varphi)$ is a distance constraint, which characterizes the similarities and dissimilarities among the mapped points from the same or different classes. $G(f, \varphi)$ is a geometric constraint, which means that the mapped points should constrain the geometric metric from the original space. $T(f, \varphi)$ is a transformation constraint, which means that a different axis of the mapped points should be unified.

1. Distance constraint

The mapped points from the same class should be close to each other, while points from different classes should be away from each other. So, we define this item as

$$D(f,\varphi)=\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n}Z(i,j)\left\|f(\boldsymbol{x}_i)-\varphi(\boldsymbol{y}_j)\right\|^2, \tag{7}$$

where

$$Z(i,j)=\begin{cases}1, & l_i^x=l_j^y,\\ -1, & l_i^x\neq l_j^y,\end{cases}$$

and l_i^x and l_j^y denote the labels of Euclidean points and Riemannian points, respectively. To facilitate computing, we use the sum of squared distances, which is also convex.

2. Geometric constraint

When mapping Euclidean points and manifold points to the same high-dimensional Hilbert space, we need the two transformations (f and φ) to preserve the geometric metric of the original data separately, which means $G(f, \varphi)=G_x(f)+G_y(\varphi)$. $G_x(f)$ and $G_y(\varphi)$ are formulated as

$$G_x(f)=\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}Z_x(i,j)\left\|f(\boldsymbol{x}_i)-f(\boldsymbol{x}_j)\right\|^2, \tag{8}$$

$$G_y(\varphi)=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}Z_y(i,j)\left\|\varphi(\boldsymbol{y}_i)-\varphi(\boldsymbol{y}_j)\right\|^2, \tag{9}$$

where

$$Z_x(i,j)=\begin{cases}d_{ij}, & \text{if } l_i^x=l_j^x \text{ and } k_1(i,j),\\ -d_{ij}, & \text{if } l_i^x\neq l_j^x \text{ and } k_2(i,j),\\ 0, & \text{otherwise},\end{cases}$$

$$Z_y(i,j)=\begin{cases}d_{ij}, & \text{if } l_i^y=l_j^y \text{ and } k_1(i,j),\\ -d_{ij}, & \text{if } l_i^y\neq l_j^y \text{ and } k_2(i,j),\\ 0, & \text{otherwise},\end{cases}$$

$d_{ij}=\exp\left(-\left\|\boldsymbol{p}_i-\boldsymbol{p}_j\right\|^2/\sigma^2\right)$, and $\boldsymbol{p}$ should be replaced by $\boldsymbol{x}$ or $\boldsymbol{y}$. These are the so-called Gaussian kernels, which preserve the geometry of the original spaces. Note that $k_1(i,j)$ and $k_2(i,j)$ represent that i is in the k_1 and k_2 neighborhood of point j, respectively. The neighboring points of the same class should also be near each other after mapping.

3. Transformation constraint

For stability of comparison in the final Hilbert space, the mapped points should be uniform in every dimension, which means

$$T(f,\varphi)=\frac{1}{2}\left(\left\|f(\boldsymbol{X})\right\|^2+\left\|\varphi(\boldsymbol{Y})\right\|^2\right), \tag{10}$$

because the unit covariance should maintain a small value.

6.2 Problem solving

To solve this optimization problem, we use an iterative strategy. First, we transform this problem into its matrix-vector form. The outer mappings of composite functions $f=f_x\circ\varphi_x$ and $\varphi=f_y\circ\varphi_y$ are linear, which means $f_x(\boldsymbol{x}_i)=\boldsymbol{V}_x^{\mathrm{T}}\varphi_x(\boldsymbol{x}_i)$ and $f_y(\boldsymbol{y}_j)=\boldsymbol{V}_y^{\mathrm{T}}\varphi_y(\boldsymbol{y}_j)$. The matrices $\boldsymbol{V}_x$ and $\boldsymbol{V}_y$ can be described as a linear combination of the coordinate basis, which means $\boldsymbol{V}_x^{\mathrm{T}}=\Sigma_j\boldsymbol{W}_x^{\mathrm{T}}\varphi_x(\boldsymbol{x}_j)$ and $\boldsymbol{V}_y^{\mathrm{T}}=\Sigma_i\boldsymbol{W}_y^{\mathrm{T}}\varphi_y(\boldsymbol{y}_i)$. Finally, $f_x(\boldsymbol{x}_i)=\Sigma_j\boldsymbol{W}_x^{\mathrm{T}}\varphi_x(\boldsymbol{x}_j)\varphi_x(\boldsymbol{x}_i)=\boldsymbol{W}_x^{\mathrm{T}}\boldsymbol{K}_{x,j}$, where $\boldsymbol{K}_{x,j}(\boldsymbol{x}_i,\boldsymbol{x}_j)=\langle\varphi_x(\boldsymbol{x}_i),\varphi_x(\boldsymbol{x}_j)\rangle$. We rewrite Eqs. (7)−(9) as follows:

$$\begin{aligned}D(\boldsymbol{W}_x,\boldsymbol{W}_y)=\frac{1}{2}(&\boldsymbol{W}_x^{\mathrm{T}}\boldsymbol{K}_x\boldsymbol{B}_x'\boldsymbol{K}_x^{\mathrm{T}}\boldsymbol{W}_x\\&+\boldsymbol{W}_y^{\mathrm{T}}\boldsymbol{K}_y\boldsymbol{B}_y'\boldsymbol{K}_y^{\mathrm{T}}\boldsymbol{W}_y-2\boldsymbol{W}_x^{\mathrm{T}}\boldsymbol{K}_x\boldsymbol{Z}\boldsymbol{K}_y\boldsymbol{W}_y^{\mathrm{T}}),\end{aligned}\tag{11}$$

$$\begin{aligned}G_x(\boldsymbol{W}_x)&=\boldsymbol{W}_x^{\mathrm{T}}\boldsymbol{K}_x\boldsymbol{B}_x\boldsymbol{K}_x^{\mathrm{T}}\boldsymbol{W}_x-\boldsymbol{W}_x^{\mathrm{T}}\boldsymbol{K}_x\boldsymbol{Z}_x\boldsymbol{K}_x^{\mathrm{T}}\boldsymbol{W}_x\\&=\boldsymbol{W}_x^{\mathrm{T}}\boldsymbol{K}_x\boldsymbol{L}_x\boldsymbol{K}_x^{\mathrm{T}}\boldsymbol{W}_x,\end{aligned}\tag{12}$$

$$\begin{aligned}G_y(\boldsymbol{W}_y)&=\boldsymbol{W}_y^{\mathrm{T}}\boldsymbol{K}_y\boldsymbol{B}_y\boldsymbol{K}_y^{\mathrm{T}}\boldsymbol{W}_y-\boldsymbol{W}_y^{\mathrm{T}}\boldsymbol{K}_y\boldsymbol{Z}_y\boldsymbol{K}_y^{\mathrm{T}}\boldsymbol{W}_y\\&=\boldsymbol{W}_y^{\mathrm{T}}\boldsymbol{K}_y\boldsymbol{L}_y\boldsymbol{K}_y^{\mathrm{T}}\boldsymbol{W}_y,\end{aligned}\tag{13}$$

where $\boldsymbol{B}_x'$, $\boldsymbol{B}_y'$, $\boldsymbol{B}_x$, and $\boldsymbol{B}_y$ are diagonal matrices with $B_x'(i,i)=\sum_{j=1}^{n}Z(i,j)$, $B_y'(j,j)=\sum_{i=1}^{m}Z(i,j)$, $B_x(i,i)=\sum_{j=1}^{m}Z_x(i,j)$, and $B_y(i,i)=\sum_{j=1}^{n}Z_y(i,j)$.

1. Initialization

We define the within- and between-class templates for $\boldsymbol{Z}$, $\boldsymbol{Z}_x$, and $\boldsymbol{Z}_y$. Without loss of generality, the within- and between-class templates for $\boldsymbol{Z}$ can be represented as

$$\begin{cases}Z^{\mathrm{w}}(i,j)=\begin{cases}1, & l_i^x=l_j^y,\\0, & l_i^x\neq l_j^y,\end{cases}\\Z^{\mathrm{b}}(i,j)=\begin{cases}0, & l_i^x=l_j^y,\\1, & l_i^x\neq l_j^y.\end{cases}\end{cases}\tag{14}$$

By replacing $\boldsymbol{Z}$ with $\boldsymbol{Z}^{\mathrm{w}}$ and $\boldsymbol{Z}^{\mathrm{b}}$ in Eq. (11), we obtain the within- and between-class templates for $D(\boldsymbol{W}_x,\boldsymbol{W}_y)$, i.e., $D^{\mathrm{w}}(\boldsymbol{W}_x,\boldsymbol{W}_y)$ and $D^{\mathrm{b}}(\boldsymbol{W}_x,\boldsymbol{W}_y)$.

Similarly, by replacing $\boldsymbol{Z}$ with $\boldsymbol{Z}^{\mathrm{w}}$ and $\boldsymbol{Z}^{\mathrm{b}}$ in Eqs. (12) and (13), respectively, we obtain the within- and between-class templates for G_x and G_y, i.e., G_x^{w}, G_x^{b}, G_y^{w}, and G_y^{b}. Now, we can use the optimization algorithm to initialize $\boldsymbol{W}_x$ and $\boldsymbol{W}_y$:

$$\begin{aligned}&\max_{\boldsymbol{W}_x,\boldsymbol{W}_y}D^{\mathrm{b}}(\boldsymbol{W}_x,\boldsymbol{W}_y)+\lambda_1G^{\mathrm{b}}(\boldsymbol{W}_x,\boldsymbol{W}_y)\\&\text{s.t. }D^{\mathrm{w}}(\boldsymbol{W}_x,\boldsymbol{W}_y)+\lambda_1G^{\mathrm{w}}(\boldsymbol{W}_x,\boldsymbol{W}_y)=1,\end{aligned}\tag{15}$$

where $G^{\mathrm{b}}=G_x^{\mathrm{b}}+G_y^{\mathrm{b}}$ and $G^{\mathrm{w}}=G_x^{\mathrm{w}}+G_y^{\mathrm{w}}$. This optimization algorithm maximizes the sum of between-class templates, and at the same time minimizes the sum of within-class templates.

2. Fixing $\boldsymbol{W}_y$ to update $\boldsymbol{W}_x$

We denote objective function (6) by $Q(\boldsymbol{W}_x,\boldsymbol{W}_y)$. Taking the partial derivatives of $Q(\boldsymbol{W}_x,\boldsymbol{W}_y)$ with respect to $\boldsymbol{W}_x$, we set the result to zero to obtain the following equation:

$$\begin{aligned}\frac{\partial Q(\boldsymbol{W}_x,\boldsymbol{W}_y)}{\partial\boldsymbol{W}_x}&=\boldsymbol{K}_x\boldsymbol{B}_x'\boldsymbol{K}_x^{\mathrm{T}}\boldsymbol{W}_x-\boldsymbol{K}_x\boldsymbol{Z}\boldsymbol{K}_y^{\mathrm{T}}\boldsymbol{W}_y\\&+2\lambda_1\boldsymbol{K}_x\boldsymbol{L}_x\boldsymbol{K}_x^{\mathrm{T}}\boldsymbol{W}_x+2\lambda_2\boldsymbol{K}_x\boldsymbol{K}_x^{\mathrm{T}}\boldsymbol{W}_x=\boldsymbol{0}.\end{aligned}\tag{16}$$

Simplifying the above equation, we have

$$\boldsymbol{W}_x=\left[\boldsymbol{K}_x(\boldsymbol{B}_x'+2\lambda_1\boldsymbol{L}_x+2\lambda_2\boldsymbol{I})\boldsymbol{K}_x^{\mathrm{T}}\right]^{-1}\boldsymbol{K}_x\boldsymbol{Z}\boldsymbol{K}_y^{\mathrm{T}}\boldsymbol{W}_y.\tag{17}$$

3. Fixing $\boldsymbol{W}_x$ to update $\boldsymbol{W}_y$

Likewise, we can obtain the analytical solution of $\boldsymbol{W}_y$ as follows:

$$\boldsymbol{W}_y=\left[\boldsymbol{K}_y(\boldsymbol{B}_y'+2\lambda_1\boldsymbol{L}_y+2\lambda_2\boldsymbol{I})\boldsymbol{K}_y^{\mathrm{T}}\right]^{-1}\boldsymbol{K}_y\boldsymbol{Z}\boldsymbol{K}_x^{\mathrm{T}}\boldsymbol{W}_x.\tag{18}$$

After initialization of $\boldsymbol{W}_x$ and $\boldsymbol{W}_y$, we can alternatively update $\boldsymbol{W}_x$ and $\boldsymbol{W}_y$ by Eqs. (17) and (18). After tens of iterations, they will converge to a stable solution.

7 Experiments

In Fig. 2, our framework consists of two branches: one branch maps the descriptors of 3D shapes to a Euclidean space; the other branch maps the descriptors of images to the same Euclidean space. Thus, after proper model sets and image sets are

chosen, with fine training, our framework can be used in the following two scenarios: (1) Through the first branch, we can retrieve similar shapes using a shape as the query; (2) Through the two branches, we can retrieve similar shapes using a sketch image as the query.

To create a multiview representation, we render each 3D shape from viewpoints spaced equally over the upper viewing hemisphere, and obtain 12 rendered depth views for each 3D shape. This virtual camera configuration is reasonable, because we often take photos in this way.

7.1 Using a shape as the query

In this scenario, we choose the Princeton ModelNet dataset, as most state-of-the-art shape retrieval methods make comparisons on this dataset.

There are 127915 3D CAD models from 662 categories in the Princeton ModelNet dataset. The authors of ModelNet have released the ModelNet40 dataset on their website, which contains 12311 shapes from 40 common categories and all the shapes are well annotated. In addition, all the shapes are upright oriented along a consistent axis. For our experiments, we use the same training and testing split of ModelNet40 as in Wu et al. (2015), which means that we use 80% for training and 20% for testing. We construct the image set by sampling rendered views from the above model set.

We compare the performance of our method with those of Spherical Harmonic (Kazhdan et al., 2003), LightField descriptor (Chen et al., 2003), PANORAMA (Papadakis et al., 2010), 3D ShapeNet (Wu et al., 2015), multi-view convolutional neural network (MVCNN) (Su et al., 2015), and GIFT (Bai et al., 2016).

To demonstrate the flexibility of our method, we adopt three types of image descriptors for the SPDs of 3D shapes. The first is the raw gray intensities of the 12 rendered depth images. The second is very delicate: (1) We train the deep learning architecture of VGG-M (Chatfield et al., 2014) on ImageNet, which contains 1000 image categories; (2) We use all the rendered depth views of 3D shapes to fine-tune the trained convolutional neural network (CNN); (3) We use the vector of 4096 dimensions in the first full-connect layer (FC6) as the image descriptor (FC6+machine learning (ML)). The third deep descriptor is constructed as follows: (1) We adopt the deep learning architecture of DeepFace (Wen et al., 2016), which is pre-trained on CASIA-Webface; (2) We fine-tune this model with all rendered depth views of 3D shapes; (3) We use the vector of 128 dimensions in the last but one full-connect layer as the image descriptor (DF+ML). The performances of the three produced 3D shape descriptors corresponding to the three image descriptors are summarized in Table 1.

Table 1 Retrieval results on the ModelNet40 dataset

Method	Training configuration			Test configuration	mAP
	Pre-training	Fine-tuning	#Views	#Views	
SPH	–	–	–	–	33.3%
LFD	–	–	–	–	40.9%
3D ShapeNet	ModelNet40	ModelNet40	–	–	49.2%
FV	–	ModelNet40	12	1	37.5%
FV, 12×	–	ModelNet40	12	12	43.9%
CNN	ImageNet1K	–	–	1	44.1%
CNN, f.t.	ImageNet1K	ModelNet40	12	1	61.7%
CNN, 12×	ImageNet1K	–	–	12	49.6%
CNN, f.t., 12×	ImageNet1K	ModelNet40	12	12	62.8%
MVCNN, f.t.+metric, 12×	ImageNet1K	ModelNet40	12	12	80.2%
GIFT	ImageNet1K	ModelNet40	64	64	81.9%
RAW+ML, 12×	–	–	12	12	57.8%
FC6+ML, f.t., 12×	ImageNet1K	ModelNet40	12	12	77.1%
DF+ML, f.t., 12×	CASIA-Webface	ModelNet40	12	12	87.5%

f.t.=fine-tune; ML=machine learning; metric=low-rank Mahalanobis metric learning (Su et al., 2015). mAP: mean average precision

In Table 1, the performance of the first type of shape descriptor is comparable only to the fine-tuned VGG-M. In contrast, the second type of image descriptor is comparable to MVCNN and GIFT, which are the two best 3D shape retrieval algorithms currently. By replacing the raw image descriptor with deep image features, the performance of our framework improves by nearly 20%. The results of this experiment are in line with our intuition. It proves that our framework is flexible and effective enough. This also shows one advantage of our framework: for every new type of image descriptor, our framework can produce the corresponding type of shape descriptor for 3D shapes. These descriptor types are compact representations of 3D shapes, and can speed up the computation of the distance between 3D shapes.

To further compare the performance of our framework with those of other algorithms, we offer the precision-recall curve. In Fig. 3, the performance of our algorithm is comparable to those of the state-of-the-art methods, because we adopt a very good image feature and our framework is very flexible. However, this is not the upper bound of our algorithm. By embedding better image descriptors, our framework can achieve even better performance.

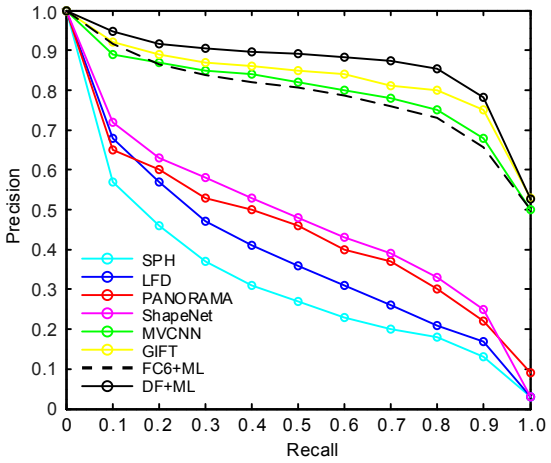

Fig. 3 Precision-recall curves on the ModelNet40 dataset
Our method is comparable to the state-of-the-art methods. References to color refer to the online version of this figure

7.2 Using a sketch as the query

The number of available 3D models on the Internet is growing rapidly and new techniques for acquiring 3D shapes are emerging. The users of the 3D engine often do not have a 3D query shape before using the engine. Thus, most online repositories (e.g., 3D Warehouse, TurboSquid, and Shapeways) provide only text-based search engines or hierarchical catalogs for 3D shapes. Text-based methods are deficient in describing 3D shapes, because 3D shapes contain rich semantic information. In contrast, 2D sketches also contain rich semantic information and they can be drawn easily by ordinary users. In this subsection, we investigate the performance of our method using sketches as queries.

We adopt the sketch-based 3D shape retrieval benchmark SHREC'13 provided by Li et al. (2014). The benchmark is reorganized from the datasets PSB and SBSR (Eitz et al., 2012), containing 1258 target models from 90 categories and 80 sketch images for each category. For the 80 sketches in each category, 50 are used for training and the rest for testing.

Because sketch images are abstract and contain rich semantic information, we need an abstract line-rendering method to shrink the semantic gap between sketches and rendered view images. We employ the line-rendering method of Fig. 4d to render all target models in this experiment (Fig. 4).

As in the last experiment, we adopt the two deep image descriptors to represent each rendered view and sketch. We pass all the rendered depth images and sketches to the fine-tuned VGG-M (or DeepFace)

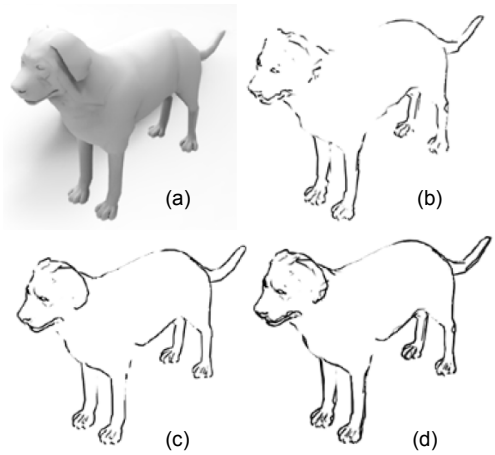

Fig. 4 Line-rendering of 3D models: (a) a dog model; (b) apparent ridges; (c) contours and suggestive contours; (d) contours and suggestive contours combined with apparent ridges
We employ (d) in our experiment

and obtain the image descriptors of 4096 (or 128) dimensions. Then we feed the model set and image set into our framework, and obtain the compact vectors for 3D shapes and query sketches. We rank 3D shapes by simply computing the Euclidean distance between these vectors.

In this experiment, we compare our method with the sketch-based 3D shape retrieval methods (Sousa and Fonseca, 2010; Saavedra et al., 2012; Furuya and Ohbuchi, 2013; Li et al., 2014; Wang et al., 2015). Table 2 shows the results of a comprehensive comparison among these methods, in terms of nearest neighbor (NN), first tier (FT), second tier (ST), E-measure (E), discounted cumulative gain (DCG), and mean average precision (mAP). Sketch-based shape retrieval is still a very difficult problem. In the future, we need to pre-train the CNNs on large datasets of sketches, which can greatly improve the performance of our method.

Table 2 Comparison on the SHREC'13 dataset

Method	NN	FT	ST	E	DCG	mAP
Saavedra et al. (2012)	0.110	0.069	0.107	0.061	0.307	0.086
Sousa and Fonseca (2010)	0.017	0.016	0.031	0.018	0.240	0.026
Li et al. (2014)	0.164	0.097	0.149	0.085	0.348	0.116
Furuya and Ohbuchi (2013)	0.279	0.203	0.296	0.166	0.458	0.250
Wang et al. (2015)	0.405	0.403	0.548	0.287	0.607	0.469
FC6+ML	0.357	0.332	0.484	0.241	0.556	0.402
DF+ML	0.426	0.401	0.553	0.310	0.625	0.471

NN: nearest neighbor; FT: first tier; ST: second tier; E: E-measure; DCG: discounted cumulative gain; mAP: mean average precision

7.3 Parameter analysis

There are two parameters λ_1 and λ_2 in objective function (6), and we tune them by a simple grid search. Specifically, we first randomly divide the training set of each dataset into two equal parts, one for training and the other for validation. Then we conduct a grid search over [0.001, 0.01, 0.1, 1, 10, 100, 1000] for both parameters. In this experiment, we set λ_1=0.01 and λ_2=0.1.

All the kernel widths σ's are specified from the mean of distances, and the neighborhood numbers are set as k_1=1 and k_2=20.

7.4 Execution time

In Section 7.1, we offer the time cost analysis of our methods and MVCNN in Table 3. The offline operations include rendering all the 3D models in the training dataset, training the CNN (MVCNN, VGG-M, or DeepFace), extracting image descriptors from all rendered view images, and training the manifold learning algorithm. Note that MVCNN does not need extraction of image descriptors or training of the manifold learning algorithm. The most time-consuming part is training the CNN, which varies significantly with different methods. Training of the manifold learning algorithm (≈0.3 h) is the second most time-consuming operation.

For every query, the online operations include rendering the query model, extracting deep image descriptors for every rendered view of the query model, constructing the SPD, mapping the SPD using the manifold learning algorithm, and comparison with the vectors in the dataset. In practice, there are a lot of techniques to speed up this process.

The above analysis also applies to the online operation in Section 7.2. The approximate time cost is given in Table 3.

Table 3 Time cost analysis on the Model40 dataset

Method	Offline (h)	Online (s)
MVCNN	≈10.8	≈0.01
RAW+ML	≈0.31	≈0.11
FC6+ML	≈9.10	≈0.21
DF+ML	≈13.5	≈0.13

The configuration of our computational platform is as follows: CPU Intel Core i7-2600, 16-GB memory, and GPU GTX TITAN X 12 GB. The code is implemented using C++ and MATLAB.

8 Conclusions

In this paper, we have proposed a point-to-set matching method for image-based 3D model retrieval, which shrinks the great semantic gap between the Euclidean space and the Riemannian manifold. In the mapping of the Euclidean space and the Riemannian manifold to the same high-dimensional Hilbert space, we have preserved the constraints of distance, geometry, and transformation. In this way, our method

ensures the similarities and dissimilarities between the mapped points of the same class and different classes, and accomplishes high-precision retrieval. In addition, by modeling image sets as SPDs, point-to-set matching has been greatly accelerated.

Our framework is so flexible that it can benefit from the recent progress in deep learning. By feeding the deep representations of each rendered view into our framework, we can output the corresponding compact representations for the 3D shape.

Despite the above achievements, we realize that set-to-set matching could further improve the stability and accuracy of our method. In the future, we will extend our framework to multi-image-based shape matching.

References

Bai S, Bai X, Zhou Z, et al., 2016. GIFT: a real-time and scalable 3D shape search engine. 16th IEEE Conf on Computer Vision and Pattern Recognition, p.5023-5032. https://doi.org/10.1109/CVPR.2016.543

Bai X, Bai S, Zhu Z, et al., 2015. 3D shape matching via two layer coding. *IEEE Trans Patt Anal Mach Intell*, 37(12): 2361-2373. https://doi.org/10.1109/TPAMI.2015.2424863

Cevikalp H, Triggs B, 2010. Face recognition based on image sets. IEEE Society Conf on Computer Vision and Pattern Recognition, p.2567-2573. https://doi.org/10.1109/CVPR.2010.5539965

Chatfield K, Simonyan K, Vedaldi A, et al., 2014. Return of the devil in the details: delving deep into convolutional nets. p.1-11. https://arxiv.org/abs/1405.3531

Chen DY, Tian XP, Shen YT, et al., 2003. On visual similarity based 3D model retrieval. *Comput Graph Forum*, 22(3): 223-232. https://doi.org/10.1111/1467-8659.00669

Chien JT, Wu CC, 2002. Discriminant waveletfaces and nearest feature classifiers for face recognition. *IEEE Trans Patt Anal Mach Intell*, 24(12):1644-1649. https://doi.org/10.1109/TPAMI.2002.1114855

Eitz M, Richter R, Boubekeur T, et al., 2012. Sketch-based shape retrieval. *ACM Trans Graph*, 31(4):31-40. https://doi.org/10.1145/2185520.2185527

Furuya T, Ohbuchi R, 2013. Ranking on cross-domain manifold for sketch-based 3D model retrieval. Int Conf on Cyberworlds, p.274-281. https://doi.org/10.1109/CW.2013.60

Hamm J, Lee DD, 2008. Grassmann discriminant analysis: a unifying view on subspace-based learning. Proc 25th Int Conf on Machine Learning, p.376-383. https://doi.org/10.1145/1390156.1390204

Hamm J, Lee DD, 2009. Extended Grassmann kernels for subspace-based learning. Advances in Neural Information Processing Systems, p.601-608.

Huang Z, Wang R, Shan S, et al., 2014. Learning Euclidean-to-Riemannian metric for point-to-set classification. IEEE Conf on Computer Vision and Pattern Recognition, p.1677-1684. https://doi.org/10.1109/CVPR.2014.217

Jayasumana S, Hartley R, Salzmann M, et al., 2013. Kernel methods on the Riemannian manifold of symmetric positive definite matrices. IEEE Conf on Computer Vision and Pattern Recognition, p.73-80. https://doi.org/10.1109/CVPR.2013.17

Kazhdan M, Funkhouser T, Rusinkiewicz S, 2003. Rotation invariant spherical harmonic representation of 3D shape descriptors. Proc Eurographics/ACM SIGGRAPH Symp on Geometry Processing, p.156-164.

Kim T, Kittler J, Cipolla R, 2007. Discriminative learning and recognition of image set classes using canonical correlations. *IEEE Trans Patt Anal Mach Intell*, 29(6): 1005-1018. https://doi.org/10.1109/TPAMI.2007.1037

Li B, Lu Y, Godil A, et al., 2014. A comparison of methods for sketch-based 3D shape retrieval. *Comput Vis Image Underst*, 119:57-80. https://doi.org/10.1016/j.cviu.2013.11.008

Lian Z, Godil A, Sun X, et al., 2013. CM-BOF: visual similarity-based 3D shape retrieval using clock matching and bag-of-features. *Mach Vis Appl*, 24(8):1685-1704. https://doi.org/10.1007/s00138-013-0501-5

Mu P, Zhang S, Ye X, 2017. A metric learning method for image-based 3D shape retrieval. Proc Int Conf on Data Mining, Communications and Information Technology, Article 17. https://doi.org/10.1145/3089871.3089876

Ohbuchi R, Osada K, Furuya T, et al., 2008. Salient local visual features for shape-based 3D model retrieval. IEEE Int Conf on Shape Modeling and Applications, p.93-102. https://doi.org/10.1109/SMI.2008.4547955

Papadakis P, Pratikakis I, Theoharis T, et al., 2010. Panorama: a 3D shape descriptor based on panoramic views for unsupervised 3D object retrieval. *Int J Comput Vis*, 89(2-3):177-192. https://doi.org/10.1007/s11263-009-0281-6

Saavedra JM, Bustos B, Schreck T, et al., 2012. Sketch-based 3D model retrieval using keyshapes for global and local representation. Proc 5th Eurographics Conf on 3D Object Retrieval, p.47-50. https://doi.org/10.2312/3DOR/3DOR12/047-050

Shilane P, Min P, Kazhdan M, et al., 2004. The Princeton Shape Benchmark. Proc Shape Modeling Applications, p.167-178. http://doi.org/10.1109/SMI.2004.1314504

Sousa P, Fonseca MJ, 2010. Sketch-based retrieval of drawings using spatial proximity. *J Vis Lang Comput*, 21(2):69-80. https://doi.org/10.1016/j.jvlc.2009.12.001

Su H, Maji S, Kalogerakis E, et al., 2015. Multi-view convolutional neural networks for 3D shape recognition. IEEE Int Conf on Computer Vision, p.945-953. https://doi.org/10.1109/ICCV.2015.114

Tabia H, Laga H, Picard D, et al., 2014. Covariance descriptors for 3D shape matching and retrieval. IEEE Conf on

Computer Vision and Pattern Recognition, p.4185-4192. https://doi.org/10.1109/CVPR.2014.533

Vemulapalli R, Pillai JK, Chellappa R, 2013. Kernel learning for extrinsic classification of manifold features. IEEE Conf on Computer Vision and Pattern Recognition, p.1782-1789. https://doi.org/10.1109/CVPR.2013.233

Vincent P, Bengio Y, 2001. K-local hyperplane and convex distance nearest neighbor algorithms. Proc 14th Int Conf on Neural Information Processing Systems: Natural and Synthetic, p.985-992.

Wang F, Kang L, Li Y, 2015. Sketch-based 3D shape retrieval using convolutional neural networks. IEEE Conf on Computer Vision and Pattern Recognition, p.1875-1883. https://doi.org/10.1109/CVPR.2015.7298797

Wang R, Guo H, Davis LS, et al., 2012. Covariance discriminative learning: a natural and efficient approach to image set classification. IEEE Conf on Computer Vision and Pattern Recognition, p.2496-2503. https://doi.org/10.1109/CVPR.2012.6247965

Wen Y, Zhang K, Li Z, et al., 2016. A discriminative feature learning approach for deep face recognition. European Conf on Computer Vision, p.499-515. https://doi.org/10.1007/978-3-319-46478-7_31

Wu Z, Song S, Khosla A, et al., 2015. 3D shapenets: a deep representation for volumetric shapes. IEEE Conf on Computer Vision and Pattern Recognition, p.1912-1920. https://doi.org/10.1109/CVPR.2015.7298801

Yamaguchi O, Fukui K, Maeda K, 1998. Face recognition using temporal image sequence. Proc 3rd IEEE Int Conf on Automatic Face and Gesture Recognition, p.318-323. https://doi.org/10.1109/AFGR.1998.670968

Zhu P, Zhang L, Zuo W, et al., 2013. From point to set: extend the learning of distance metrics. IEEE Int Conf on Computer Vision, p.2664-2671. https://doi.org/10.1109/ICCV.2013.331