Pattern Recognition Letters 65 (2015) 15–21

Contents lists available at ScienceDirect

Pattern Recognition Letters

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

Neural shape codes for 3D model retrieval☆

Song Bai [a], Xiang Bai [a,*], Wenyu Liu [a], Fabio Roli [b]

[a] School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China
[b] Department of Electrical and Electronic Engineering, University of Cagliari, Cagliari 09123, Italy

ARTICLE INFO

Article history:
Received 6 March 2015
Available online 9 July 2015

Keywords:
Shape retrieval
3D model
Depth view
CNN

ABSTRACT

The paradigm of Convolutional Neural Network (CNN) has already shown its potential for many challenging applications of computer vision, such as image classification, object detection and action recognition. In this paper, the task of 3D model retrieval is addressed by exploiting such promising paradigm. However, 3D models are usually represented with a collection of orderless points, lines and surfaces in a three dimensional space, which makes it difficult to involve the operation of convolution, pooling, *etc*. Yet, we propose a practical and effective way for applying CNN to 3D model retrieval, by training the network with the depth projections of 3D model. This CNN is regarded as a generic feature extractor for depth image. With large amounts of training data, the learned feature, which is called Neural Shape Codes, can handle various deformation changes that exist in shape analysis. The reported experimental results on several 3D shape benchmark datasets show the superior performance of the proposed method.

1. Introduction

The increasing availability of RGB-D consumer sensors makes possible to handle 3D models using just a laptop, tablet or mobile terminal. Lots of large public-domain databases, such as Google 3D Warehouse, make 3D analysis a popular task, and increase the demand for efficient 3D model retrieval techniques. Given a query 3D model, the aim of retrieval is to find the most similar models in the database. The pairwise similarity is usually computed using a certain metric in the feature space, and candidate models are ranked by visual appearance with the query.

In recent years, different algorithms and features have been proposed to represent 3D models. These algorithms can be coarsely divided into two categories: model-based algorithms and view-based algorithms. For model-based algorithms, some geometric features are extracted directly at the object level. The matching between 3D models is done by analyzing the correspondences among these features. Typical model-based methods are Covariance [1], and Curve analysis [2]. View-based methods attract a lot of attention recently. They represent a 3D model with a collection of 2D views using the format of depth buffer or binary mask. Thanks to the discriminative 2D views, view-based 3D model retrieval algorithms, such as PANORAMA [3], Light Field Descriptor [4], provide good retrieval performance.

It is worth to note that most of the above algorithms work well only for some specific shape deformations (bending, articulation, rotation, *etc*.). Motivated by the excellent performance of deep learning in computer vision, we conjecture that Convolutional Neural Networks (CNNs) trained with large amounts of data is a suitable choice, and can be used effectively for 3D model retrieval. Moreover, we want to benefit from the supervised training of CNNs for solving the unsupervised 3D model retrieval task. When the traditional "man-made" descriptors [5–7] cannot handle well the large inter-class variations and the small intra-class variations, we deem that the supervised information (or prior information) exploited during the training of CNN makes the discrimination task much easier. Nevertheless, using CNN to cope with 3D model retrieval is not an easy task. 3D models are usually represented by a group of points, lines and surfaces in a three dimensional space, while the input of CNN is usually greyscale images or color images in a two dimensional space. Our solution for this issue is very simple yet effective. Instead of taking 3D models as the input of neural network, we use the projections of 3D models. The projections are rendered from various locations on the unit sphere in the format of depth buffer. It results that each 3D model is represented by a set of depth projections. All the depth projections belonging to the training set are guaranteed to have an equal size, and are used to train the Convolutional Neural Network.

For our specific scenario of 3D model retrieval, we assume that the trained CNN can be regarded as a *generic feature extractor* for depth views. The activations obtained by a depth projection of the internal layer of CNN are taken as the learned features, which we call *Neural Shape Codes (NSC)*. After computing the Neural Shape Code for each projection of a 3D model, we take a simple modification of the Hausdorff distance for multi-view matching and achieve state-of-the-art retrieval performance efficiently.

☆ This paper has been recommended for acceptance by Dr. A. Koleshnikov.

* Corresponding author. Tel.: +8613260591019; fax: 87543236.
E-mail address: xiang.bai@gmail.com, xbai@hust.edu.cn (X. Bai).

http://dx.doi.org/10.1016/j.patrec.2015.06.022
0167-8655/

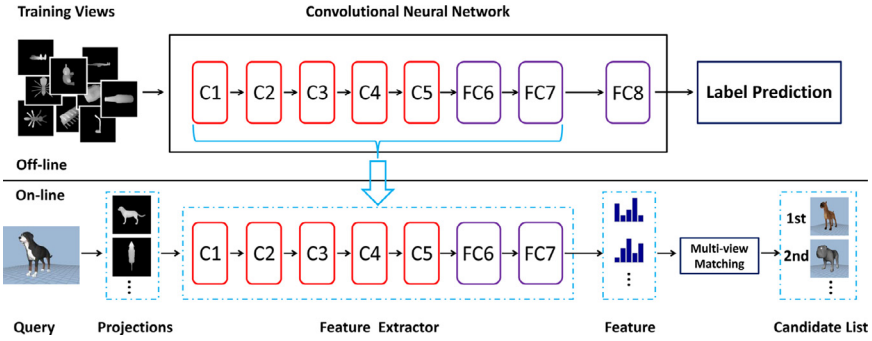

Fig. 1. The pipeline of our proposed method.

The rest of the paper is organized as follows. The details of the Neural Shape Codes are given in Section 2. The experimental analysis and comparisons are reported in Section 3. The conclusions about the contributions and limitations of our method are given in Section 4.

2. The proposed neural shape codes

In this section, we first give a brief introduction to Convolutional Neural Network (CNN), then present the procedure of projection rendering, and feature extraction using the trained CNN. The pipeline of the proposed method is illustrated in Fig. 1.

2.1. Preliminary: CNN for image analysis

Convolutional Neural Network (CNN) is inspired by biological mechanism and can be assumed as a variant of Multi-layer Perceptron (MLP).

CNN is designed to learn the 2D structure of an input image. Generally, CNN is a hieratical neural network with convolution layers, pooling layers, fully connected layers. These layers can be considered as a set of neurons which have learned weights and biases. The parameters are learned in a data-driven way from large scale images using back propagation, and the capacity of CNN can be controlled by varying its depth (*e.g.* adding or subtracting the layers) or breadth (*e.g.* increasing or decreasing the convolutional kernels). Feature learning and high-level recognition are done simultaneously in CNN architecture, which is the primary difference against conventional methods [6,8] that use artificial features. Although CNN typically contains millions of parameters to tune, the rapid development of GPU computing devices and some mathematical strategies (*e.g.* weight sharing) reduce the training time of CNN dramatically. Hence, CNN has exhibited great potential and achieved state-of-the-art performances in various computer vision tasks, such as retrieval, classification [9], detection, *etc*.

2.2. Projection rendering

In most cases, the input of CNN is 2D image, and this does not fit with the fact that 3D models are usually represented by a collection of points and lines in a three dimensional space. Therefore, applying CNN directly to 3D model analysis is not a straightforward task. To tackle this problem, we choose to project 3D models into 2D depth views.

Before projection rendering, we perform pose normalization for the 3D model $\mathcal{S}$ to get our proposed descriptor invariant to scale changes. Specifically, we move the center of $\mathcal{S}$ to the origin of the spherical coordinate system, and resize the maximum polar distance of the points on the surface of shape to unit length. Then we set N_v view points evenly on the unit sphere located by the azimuth angle θ_{az} and the elevation angle θ_{el}. θ_{az} is the polar angle in the x–y plane, and θ_{el} is the angle between view point and the x–y plane. At last, for each pair of θ_{az} and θ_{el}, we set a projection plane, and get the depth values of all the mesh points. The depth values are normalized to [0, 255] stored in the format of uint8. The numerical setup of the projection procedure is given in Section 3.

As a result, each 3D model $\mathcal{S}$ is represented by a view set $\mathcal{V}(\mathcal{S}) = \{v_{s_1}, v_{s_2}, \ldots, v_{s_{N_v}}\}$. All the depth projections of the training set are gathered to train CNN. The detailed setup of CNN is described in the next section.

2.3. Feature extraction with CNN

In this work, we use CNN as a *generic feature extractor* for depth views in 3D model retrieval.

Our CNN architecture takes depth images as input. For each depth image, we get an activation of the internal layer of CNN along the direction of feed-forward. The activation is the *Neural Shape Code (NSC)* of the depth image, which is used later for retrieval task.

Similar to the popular network architecture developed for image classification in [9], CNN used here is comprised of five successive convolutional layers C_1, C_2, C_3, C_4, C_5 and three fully connected layers FC_6, FC_7, FC_8, as presented in Fig. 1. After the convolutional layers, we have rectified linear (ReLU) transform, a non-saturating non-linear function that abandons the negative values in the output. It has been shown that ReLU can reduce the training time significantly. Three convolutional layers C_1, C_2, C_5 are also attached with a max pooling transform. At the top of the architecture, three fully-connected layers are computed as $\mathbf{Y_6} = \mathbf{\Theta}(\mathbf{W_6 Y_5} + \mathbf{B_6})$, $\mathbf{Y_7} = \mathbf{\Theta}(\mathbf{W_7 Y_6} + \mathbf{B_7})$, $\mathbf{Y_8} = \mathbf{\Phi}(\mathbf{W_8 Y_7} + \mathbf{B_8})$, where $\mathbf{Y_n}$ represents the output of the nth layer, and $\mathbf{W_n}$ and $\mathbf{B_n}$ are parameters of the nth layer tuned during training the network. $\mathbf{\Theta(X)}$ is the ReLU non-linear activation function that selects the non-negative values in $\mathbf{X}$, and $\mathbf{\Phi(X)}$ is the SoftMax non-linear activation function that produces the distribution over the entire class labels (352 categories in our cases). The detailed parameter setup of our architecture is listed in Table 1.

In order to avoid overfitting, a dropout strategy is applied to the first two fully-connected layers. The rate of dropout is set to 0.5 by experiments. It indicates that the outputs of half neurons, which are

Table 1
The architecture parameter setting of Neural Shape Codes (from the bottom to the top).

Layer	Setup
C_1	Convolution, kernel 11 × 11, stride 4, ReLU
C_2	Convolution, kernel 5 × 5, stride 1, ReLU
C_3	Convolution, kernel 3 × 3, stride 1, ReLU
C_4	Convolution, kernel 3 × 3, stride 1, ReLU
C_5	Convolution, kernel 3 × 3, stride 1, ReLU
FC_6	Fully-connected, output 4096, dropout
FC_7	Fully-connected, output 4096, dropout
FC_8	Fully-connected, output 352

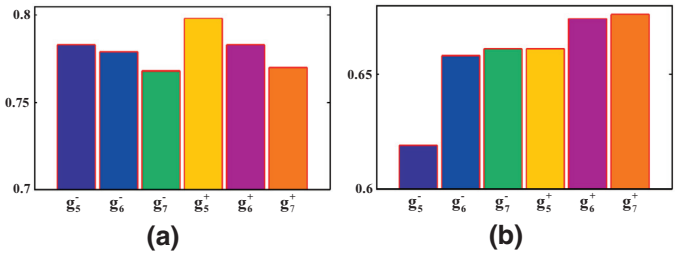

Fig. 2. The performance of Neural Shape Codes of different layers before or after ReLU transform in NN (a) and ST (b).

randomly selected, are set to zero during one training iteration, and these "dropped out" neurons contribute nothing to the forward pass and back propagation.

The CNN is trained using mini-batch stochastic gradient descent (SGD). The batch size is set to 32, the momentum is set to 0.9, and the weight decay is set to 0.0005. The weights of each layer are initialized using Gaussian distribution with 0-mean and 0.01-variance. The neuron biases of C_1, C_3 and FC_8 are initialized to 0, and those of other layers are initialized to 1.

For each depth view v, we denote the L_2 normalized activation of C_5, FC_6, FC_7 after ReLU transform as $g_5^+(v)$, $g_6^+(v)$, $g_7^+(v)$ respectively. By contrast, the L_2 normalized activations prior to ReLU transform are denoted by $g_5^-(v)$, $g_6^-(v)$ and $g_7^-(v)$. Notice that $g_5^+(v)$, as well as $g_6^+(v)$ and $g_7^+(v)$, only contains non-negative values, due to the usage of ReLU transform.

In order to evaluate the performance of $g_5^-(v)$, $g_6^-(v)$, $g_7^-(v)$, $g_5^+(v)$, $g_6^+(v)$ and $g_7^+(v)$, we do an experiment with Princeton Shape Benchmark (PSB) [10] using the modified Hausdorff distance D_{sum_min} introduced in Section 2.4. Nearest Neighbor (NN) and Second Tier (ST) are selected to compare the difference of retrieval performance (see [10] for more details about the definition of NN and ST).

The results are shown in Fig. 2. It can be seen that the Neural Shape Codes after ReLU transform, non-negative deep learned features, usually perform better than those before ReLU transform in both NN and ST. In our specific scenario of 3D model retrieval, it is interesting that the activation from relatively high layer (*e.g.* g_7^+) generally outperforms the one from low layer (*e.g.* g_5^+) in ST, but achieves worse result in NN. For 3D model search engine, NN may not be the most important index, since NN only counts for the percentage of the closest matches that belongs to the same class as the query. However, users also pay much attention to a group of retrieved objects in the front of the candidate list, instead of a single one. We also test the activations of much lower layers (*e.g.* C_3, C_4), but get worse results. In the following experiments, we choose to use g_7^+ as our Neural Shape Code.

In Fig. 3, we give the retrieval results using g_5^+, g_6^+ and g_7^+ as the Neural Shape Codes for a given query (a desktop computer). From the candidate list, we find that the activation of the convolutional layer g_5^+ presents much different property compared with the activation of the fully-connected layer (g_6^+ or g_7^+). As the first row of Fig. 3 lists, it tends to retrieve computer monitors incorrectly using g_5^+. By contrast, when using g_6^+ or g_7^+, it is more likely to retrieve false results sharing the similar geometric appearance to the query, such as the building in the second row and the satellite in the third row. It seems that the high level Neural Shape Code is better at dealing with the geometric deformation, while the relatively low level Neural Shape Code is more robust to handle near-duplicate retrieval. Such a phenomenon inspires us to conduct feature fusion for Neural Shape Codes of different layers to further improve the performance. For efficiency, we generate a *Hybrid Neural Shape Codes* with the linear combination of g_5^+ and g_7^+ with equal weights, which makes a significant gain to a single Neural Shape Code in retrieval performance as shown in Section 3.

2.4. Multi-view matching

Let $\mathcal{Q}$ and $\mathcal{P}$ denote the query model and a certain 3D model in the database respectively, and the corresponding two feature sets are $\mathcal{F}(\mathcal{Q}) = \{q_1, q_2, \ldots, q_{N_v}\}$ and $\mathcal{F}(\mathcal{P}) = \{p_1, p_2, \ldots, p_{N_v}\}$. How to measure the matching cost between the two feature sets reliably and efficiently is a well-known key issue. It is usually an assignment problem, which can be solved by Hungarian, Dynamic Programming, Game theory [11] or some more sophisticated, probability-based techniques [12].

In our specific scenario, we utilize a simple variant of the Hausdorff distance [13] to evaluate the matching cost, which is proven to be effective for 3D model retrieval as shown in Section 3. The distance between the ith view feature q_i of the query $\mathcal{Q}$ and the feature set $\mathcal{F}(\mathcal{P})$ of the 3D model $\mathcal{P}$ can be defined in three ways as

$$d_1(q_i, \mathcal{F}(\mathcal{P})) = \min_{p_j \in \mathcal{F}(\mathcal{P})} \left\| q_i - p_j \right\|, \tag{1}$$

$$d_2(q_i, \mathcal{F}(\mathcal{P})) = \max_{p_j \in \mathcal{F}(\mathcal{P})} \left\| q_i - p_j \right\|, \tag{2}$$

$$d_3(q_i, \mathcal{F}(\mathcal{P})) = \sum_{p_j \in \mathcal{F}(\mathcal{P})} \left\| q_i - p_j \right\|. \tag{3}$$

Accordingly, we also consider three ways to define the distance between $\mathcal{F}(\mathcal{Q})$ and $\mathcal{F}(\mathcal{P})$ as

$$f_1(\mathcal{F}(\mathcal{Q}), \mathcal{F}(\mathcal{P})) = \min_{q_i \in \mathcal{F}(\mathcal{Q})} d(q_i, \mathcal{F}(\mathcal{P})), \tag{4}$$

$$f_2(\mathcal{F}(\mathcal{Q}), \mathcal{F}(\mathcal{P})) = \max_{q_i \in \mathcal{F}(\mathcal{Q})} d(q_i, \mathcal{F}(\mathcal{P})), \tag{5}$$

$$f_3(\mathcal{F}(\mathcal{Q}), \mathcal{F}(\mathcal{P})) = \sum_{q_i \in \mathcal{F}(\mathcal{Q})} d(q_i, \mathcal{F}(\mathcal{P})). \tag{6}$$

To sum up, it results in nine possible variants of Hausdorff distance when combining three point-to-set distance measures $d_i (1 \le i \le 3)$ and three set-to-set distance measures $f_i (1 \le i \le 3)$. Note that D_{max_min} is the standard Hausdorff distance that measures the greatest of all the distances from a point in one set to the closest point in the other set.

It can be inferred that the modified Hausdorff distance D_{sum_min} is a more proper choice for our model 3D retrieval task, as the matching

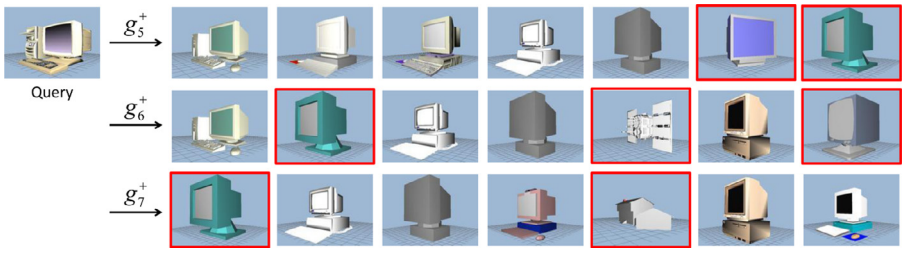

Fig. 3. Representative retrieval results in PSB dataset using g_5^+, g_6^+ and g_7^+. False positive instances are in red boxes.

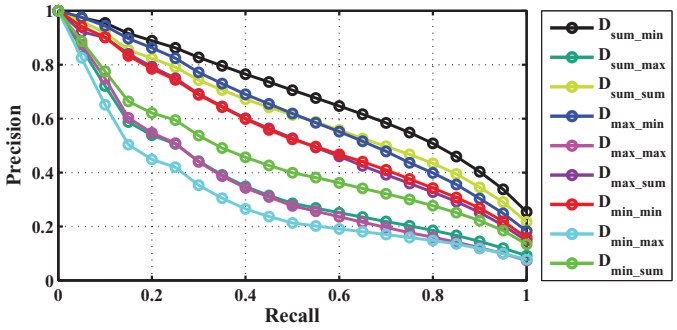

Fig. 4. The performance comparison of different variants of Hausdorff distance.

cost measured by D_{sum_min} can remain stable in the presence of some isolated views. Isolated views usually occur, since 3D model becomes unrecognized from certain view points. In the matching function defined by D_{sum_min}, the disturbance of isolated views to the distance measure is greatly eliminated, by taking the operation of *min*. Moreover, it also considers the contributions from all query views through *sum* operation. In order to prove our conjecture, we draw the P-R curves of the 9 variants using the Neural Shape Code g_7^+ in Fig. 4.

The blue curve depicts the performance of standard Hausdorff distance D_{max_min}, which achieves the second best performance among all the 9 distance measures, while D_{sum_min} provides the best result. The inferior performance of D_{max_min} compared with D_{sum_min} indicates that the standard Hausdorff distance is more likely to amplify the negative effect of isolated views in the query 3D model, while our modified Hausdorff distance can decrease it largely. In the following experiments, D_{sum_min} is selected as the matching function of Neural Shape Codes.

3. Experiments

In this section, we evaluate the proposed Neural Shape Codes on several 3D model benchmark datasets within shape retrieval framework, and experimentally assess its performance against other state-of-the-art algorithms.

Implementation details. For the procedure of projection rendering, the range of azimuth angle θ_{az} is restricted to $[0, 2 * \pi]$, and $[0, \pi]$ for the elevation angle θ_{el}. We uniformly divide the range of both θ_{az} and θ_{el} into 8 parts to obtain 8×8 view points on the unit sphere. Hence, we get $N_v = 64$ views in depth buffer. The views are resized to 200×200.

The Convolutional Neural Network is implemented with Caffe [14]. The training set we use is Toyohashi Shape Benchmark (TSB) [15], the biggest benchmark dataset for 3D model retrieval to date. It contains 100,00 3D models grouped into 352 categories, and our input to CNN is 100, 00 $\times$ 64 depth images. The object categories in TSB vary from rigid models like airships, cars, bikes to non-rigid objects like butterflies, snakes, cows. The training dataset is carefully chosen to avoid the overlap with testing datasets used below.

Table 2
Performance comparison on PSB dataset.

Methods	NN	FT	ST	DCG
LFD [4]	0.657	0.380	0.487	0.643
DESIRE [18]	0.665	0.403	0.512	0.663
tBD [19]	0.723	–	–	0.667
2D/3D hybrid [20]	0.742	0.473	0.606	–
PANORAMA [3]	0.753	0.479	0.603	–
NSC	0.774	0.557	0.687	0.774
Hybrid NSC	**0.794**	**0.572**	**0.698**	**0.785**

Datasets. To demonstrate the effectiveness and generalization capability of the proposed method, experiments are carried out with four representative 3D model benchmark datasets, including Princeton Shape Benchmark (PSB) [10], Watertight Models track of SHape REtrieval Contest 2007 dataset (WM-SHREC07) [16], National Taiwan University (NTU) dataset [4], CCCC dataset [17]. Some typical 3D models are shown in Fig. 5.

Evaluation methods. To comprehensively assess the performance, we adopt four evaluation methods, including Nearest Neighbor (NN), First Tier (FT), Second Tier (ST), Discounted Cumulative Gain (DCG). All the evaluation scores range from 0% to 100%, and a higher score indicates a better performance. We also give Precision-Recall plots to visualize the performances directly. If necessary, the readers could refer to [10] for more details about the definitions of NN, FT, ST, DCG and Precision-Recall plots.

3.1. Experiment on PSB dataset

Princeton Shape Benchmark (PSB) [10] is a widely-used generic benchmark for 3D model retrieval and recognition task. The benchmark set of models is split into a training database and a test database, with each containing 907 3D polygonal models. Here we only use the testing dataset for evaluation, which are classified into 92 classes.

We compare the proposed deep feature with several state-of-the-art methods in Table 2, including PANORAMA [3], 2D/3D Hybrid [20], tBD [19], DESIRE [18] and Light Field descriptor (LFD) [4]. As we can see, our proposed deep feature performs best among all the listed methods for all the four evaluation metrics. NSC leads to a significant improvement of 2.1% in NN, 7.8% in FT, 8.4% in ST compared with PANORAMA, which is very encouraging since PANORAMA may be the most representative descriptor in 3D model retrieval in recent years. LFD is a classical view-based 3D descriptor, in which two artificial

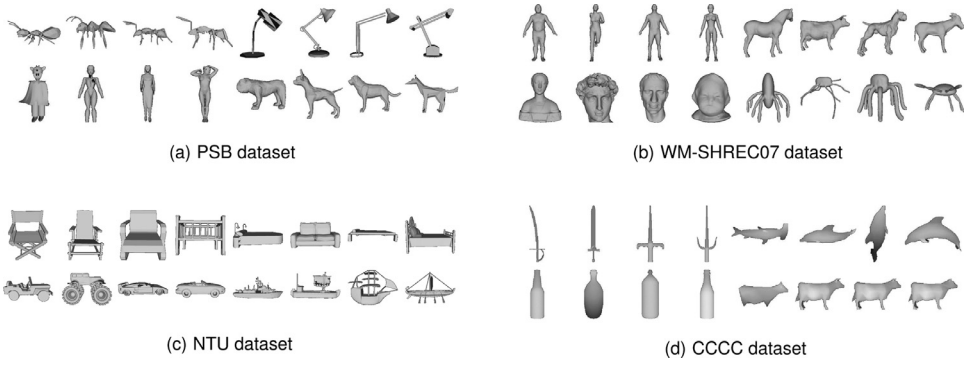

Fig. 5. Some typical 3D models from PSB dataset (a), WM-SHREC07 dataset (b), NTU dataset (c) and CCCC dataset (d).

Table 3
Performance comparison on WM-SHREC07 dataset.

Methods	NN	FT	ST	DCG
LFD [4]	0.923	0.526	0.662	–
Tabia *et al.* [2]	0.853	0.527	0.639	0.719
DESIRE [18]	0.917	0.535	0.673	–
Covariance [1]	0.930	0.623	0.737	0.864
2D/3D hybrid [20]	0.955	0.642	0.773	–
PANORAMA [3]	0.957	0.673	0.784	–
NSC	**0.962**	0.725	0.853	0.920
Hybrid NSC	0.960	**0.748**	**0.869**	**0.926**

features Zernike moments and Fourier descriptors are extracted. It is evident that NSC automatically performs markedly better than these artificial features. The excellent performance reflects the highly discriminative power of NSC for 3D model retrieval.

Moreover, the Hybrid NSC improves the performance of the proposed NSC further. The improvement brought by Hybrid NSC is 2.0%, 1.5%, 1.1% and 1.1% in NN, FT, ST, and DCG respectively. Note the cost for the performance improvement by Hybrid NSC is tiny compared with using NSC only, for we just consider the concatenation of g_5^+ and g_7^+ with equal weights, and it only increases the dimension of the final representation for views. Indeed, more complicated feature fusion methods can be used here. Compared with other hybrid methods, our hybrid of g_5^+ and g_7^+ performs better than 2D/3D Hybrid descriptor in all the four evaluation metrics, and more than 9% improvement is observed in ST.

3.2. Experiment on WM-SHREC07 dataset

Watertight Models track of SHape REtrieval Contest 2007 dataset (WM-SHREC07) [16] is a standard benchmark for the well-known 3D shape retrieval competition held each year. 400 watertight mesh models are evenly distributed into 20 classes, which exhibit sufficient and diverse variation, from pose change to shape variability in the same semantic group. Each model is used in turn as a query against the remaining part of the database.

In Table 3, the retrieval performances resulting from other state-of-the-art approaches and our proposed method are reported. Besides PANORAMA [3], 2D/3D Hybrid [20], DESIRE [18] and LFD [4], we also report the results of two recent published algorithms [1,2]. Tabia *et al.* [2] propose a curve-based method which conducts curve analysis around the detected interest points in the surface of 3D model. Covariance method [1], which uses the covariance of the features instead of the features themselves, is the latest algorithm that works well in WM-SHREC07 dataset.

From the table, we find that the NSC outperforms Covariance method by 2.8% in NN, 11.1% in FT, 12.2% in ST, 5.7% in DCG and exceeds the curve-based method by 10.9% in NN, 19.8% in FT, 21.4% in ST and 20.1% in DCG.

Despite of the fact that NSC leads to the state-of-the-art performance in this dataset, Hybrid NSC improves the retrieval performance further, as listed in the last row in Table 3.

3.3. Experiment on NTU dataset

National Taiwan University (NTU) dataset [4] comprises 1,833 3D models, 1,284 of which are unclassified and unlabeled models. Following the standard setup, only 549 models, divided into 47 categories, are used for 3D model retrieval.

The quantitative retrieval results of different algorithms are listed in Table 4. Although the proposed NSC performs slightly worse (only 0.3%) than PANORAMA [3] in NN, it outperforms PANORAMA by 2.2% in FT, 2.8% in ST and 1.4% in DCG. The Hybrid NSC achieves the best performance among all the compared methods in the four evaluation metrics.

Table 4
Performance comparison on NTU dataset.

Methods	NN	FT	ST	DCG
LFD [4]	0.700	0.390	0.501	–
DESIRE [18]	0.719	0.427	0.554	–
2D/3D hybrid [20]	0.762	0.466	0.591	–
PANORAMA [3]	0.797	0.490	0.610	0.755
NSC	0.794	0.512	0.638	0.769
Hybrid NSC	**0.828**	**0.520**	**0.646**	**0.777**

Table 5
Performance comparison on CCCC dataset.

Methods	NN	FT	ST	DCG
LFD [4]	0.798	0.502	0.631	–
DESIRE [18]	0.828	0.556	0.700	–
2D/3D hybrid [20]	0.874	0.602	0.758	–
PANORAMA [3]	0.879	0.663	0.812	–
NSC	0.887	0.686	0.835	0.868
Hybrid NSC	**0.896**	**0.696**	**0.840**	**0.874**

Table 6
DCG comparison on SHREC 2010 non-rigid dataset.

Methods	HKS	SIHKS
BoF [22]	0.806	0.919
SSBoF [22]	0.804	0.916
ISPM [23]	0.859	0.931
NSC	0.930	
Hybrid NSC	**0.945**	

3.4. Experiment on CCCC dataset

CCCC [17] is made of 472 3D models, subdivided into 55 categories. The number of models per category ranges from 2 to 56.

In Table 5, retrieval results using several state-of-the-art methods are provided. In correspondence with the previous results in other 3D datasets, NSC and Hybrid NSC perform better that all other approaches, which validates again the effectiveness and generalization of our propose descriptor.

3.5. Visualization of Precision-Recall

Besides the quantitative results listed above, P-R plots are also given to visualize the difference of various algorithms in performance directly. From Fig. 6, it can be found easily that our proposed features, Neural Shape Codes and Hybrid Neural Shape Codes outperform the other compared methods significantly.

3.6. Robustness to non-rigid deformation

Neural Shape Code is partly robust to non-rigid deformation, due to the inborn nature of CNN. Different from many previous man-made features that utilize intrinsic information, Neural Shape Codes uses the neutral network to automatically capture the repetitive structures from large amounts of training data. In order to confirm this, an extra experiment on SHREC 2010 non-rigid dataset [21] is conducted, and the performance comparison is listed in Table 6. As it suggests, the DCG value of NSC and Hybrid NSC are 0.930 and 0.945 respectively, while the best performance of Shape Google [22] is 0.919, which confirms the robustness of Neural Shape Codes to non-rigid deformation.

3.7. Parameter discussion

Some primary parameters of NSC are the view number N_v and the view size. In this section, we discuss the influence of these parameters on the final retrieval accuracy (DCG) in Fig. 7 with PSB dataset.

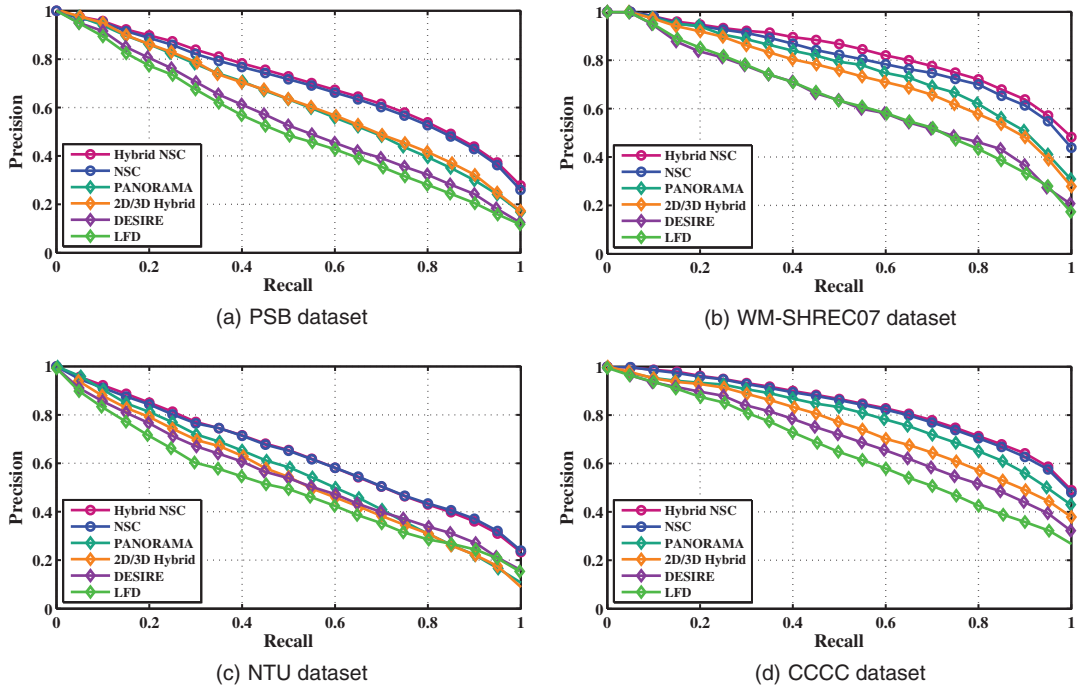

Fig. 6. The Precision-Recall plots of Neural Shape Codes and other compared algorithms in various 3D model datasets.

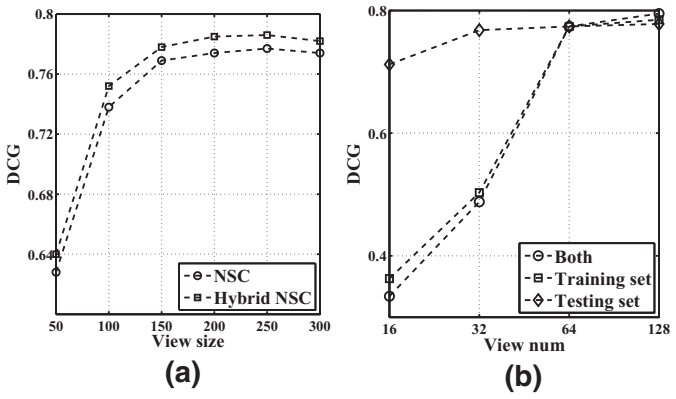

Fig. 7. The influence of view size (a) and view number (b) on ST in PSB dataset.

As can be drawn from Fig. 7(a), the retrieval performance improves with the view size increasing, and it gets saturated at the size of 250. Since larger view size leads to heavier computational cost, the default view size is set to 200 considering that such a setup merely performs slightly inferior to the best performance. The influence of N_v of NSC is illustrated in Fig. 7(b). Here we consider three situations: (1) varying N_v of training set, while keep N_v of testing set unchanged; (2) N_v is fixed for training set, while varying N_v of testing set; (3) N_v of training set and testing set are changed at the same time. The three situations are denoted as "Training set", "Testing set" and "Both" respectively in the figure, and the three curves intersect in the default setup, *i.e.* N_v is 64. It can be drawn easily that the view number of training set affects the retrieval performance more largely than that of testing set. Generally it leads to more benefit when using more views, and thus demonstrates the potential of NSC further.

3.8. Complexity analysis

The Hausdorff distance is used for multi-view matching, which may be the most efficient approach for measuring a set-to-set distance. However, the complexity of Neural Shape Codes using the Hausdorff distance is $O(N^2 \times N_v^2)$ (N denotes the database size), which is higher than $O(N^2)$ if our learned features for 3D models lie in the vector space. We implement the multi-view matching procedure on a desktop machine with an Intel(R) Core(TM) i5 CPU (3.40 GHz) and 16 GB memory, and give the comparison of pairwise matching time in Table 7 on PSB dataset. Here, one more algorithm is considered for the matching of NSC, *i.e.* Hungarian. In this situation, the time complexity of NSC is increased to $O\left(N^2 \times (N_v^2 + N_v^3)\right)$.

Table 7
The comparison of pairwise matching time on PSB dataset.

Methods	The matching time
LFD [4]	1.30 ms
PANORAMA [3]	0.23 ms
2D/3D hybrid [20]	0.17 ms
NSC	0.18 ms
NSC (Hungarian)	0.72 s

As the table shows, the matching time of NSC is still comparable with other representative algorithms. NSC using Hungarian algorithm for multi-view matching takes 0.72 *s*, and the entire time for finishing the retrieval test on PSB dataset requires totally seven days. As for retrieval accuracy, the performance of NSC, 0.775 in DCG, is only comparable with modified Hausdorff distance. It indicates that Hausdorff distance tends to be a better choice than Hungarian in terms of efficiency.

4. Conclusions

In this paper, we addressed the task of 3D model retrieval using Convolutional Neural Network (CNN). We use the depth projections of 3D model as the inputs of the neural network, instead of the 3D model itself. The activation of the internal layer of CNN is regarded as the learned feature for 2D depth view. The matching between two 3D models is achieved by efficiently computing the modified Hausdorff distance between two corresponding view sets in the feature space.

4.1. Why to use the NSC?

Indeed, many "man-made" features [24–37] have been proposed or introduced for 3D model retrieval, recognition and matching in the past decade, but they fail to achieve satisfactory performance. By contrast, our proposed Neural Shape Code is a non-artificial feature, which is learned using CNN. With large amounts of training data, the supervised learning procedure gets Neural Shape Codes more capable of handling the large intra-class variance and small inter-class variance. The discriminative power and generalization capability of our proposed Neural Shape Codes have been demonstrated on four popular 3D shape benchmark datasets.

If more 3D objects are used for training CNN, it can be expected that Neural Shape Codes can provide better performance further. Moreover, the neural network structure is very flexible. More neurons and more layers can be added to enhance the capacity of network, which meets the requirements for big data. In one word, we think that Neural Shape Code is an excellent 3D model descriptor with great potential.

4.2. Limitations

As stated in Section 3.8, the time complexity of NSC is a bit high regarding real-time retrieval. Hence how to aggregate the Neural Shape Codes efficiently without decreasing the retrieval performance is still an open issue.

Acknowledgments

This work was primarily supported by National Natural Science Foundation of China (NSFC) (no. 61222308), and in part by NSFC (no. 61173120), Program for New Century Excellent Talents in University (no. NCET-12-0217), Fun-damental Research Funds for the Central Universities (no. HUST 2013TS115).

References

[1] H. Tabia, H. Laga, D. Picard, P.-H. Gosselin, Covariance descriptors for 3d shape matching and retrieval, in: Proceedings of the CVPR, 2014, pp. 4185–4192.
[2] H. Tabia, M. Daoudi, J.-P. Vandeborre, O. Colot, A new 3d-matching method of nonrigid and partially similar models using curve analysis, Trans. Pattern Anal. Mach. Intell. 33 (4) (2011) 852–858.
[3] P. Papadakis, I. Pratikakis, T. Theoharis, S.J. Perantonis, Panorama: a 3d shape descriptor based on panoramic views for unsupervised 3d object retrieval, Int. J. Comput. Vis. 89 (2-3) (2010) 177–192.
[4] D.Y. Chen, X.P. Tian, Y.T. Shen, M. Ouhyoung, On visual similarity based 3d model retrieval, Comput. Gr. Forum 22 (3) (2003) 223–232.
[5] H. Ling, D.W. Jacobs, Shape classification using the inner-distance, Trans. Pattern Anal. Mach. Intell. 29 (2) (2007) 286–299.
[6] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[7] J. Wang, X. Bai, X. You, W. Liu, L.J. Latecki, Shape matching and classification using height functions, Pattern Recognition Lett. 33 (2) (2012) 134–143.
[8] S. Bai, X. Wang, C. Yao, X. Bai, Multiple stage residual model for accurate image classification, in: Proceedings of the ACCV, 2013, pp. 430–445.
[9] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proceedings of the NIPS, 2012, pp. 1106–1114.
[10] P. Shilane, P. Min, M.M. Kazhdan, T.A. Funkhouser, The princeton shape benchmark, in: Proceedings of the SMI, 2004, pp. 167–178.
[11] A. Albarelli, E. Rodola, F. Bergamasco, A. Torsello, A non-cooperative game for 3d object recognition in cluttered scenes, in: Proceedings of the International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission, 2011, pp. 252–259.
[12] E. Rodolà, T. Harada, Y. Kuniyoshi, D. Cremers, Efficient shape matching using vector extrapolation, in: Proceedings of the BMVC, 1, 2013.
[13] D.P. Huttenlocher, G.A. Klanderman, W.J. Rucklidge, Comparing images using the hausdorff distance, Trans. Pattern Anal. Mach. Intell. 15 (9) (1993) 850–863.
[14] Y. Jia, Caffe: An open source convolutional architecture for fast feature embedding, 2013, (http://caffe.berkeleyvision.org/).
[15] A. Tatsuma, H. Koyanagi, M. Aono, A large-scale shape benchmark for 3d object retrieval: Toyohashi shape benchmark, 2012, pp. 1–10.
[16] D. Giorgi, S. Biasotti, L. Paraboschi, Shape retrieval contest 2007: Watertight models track, SHREC competition (2007).
[17] D.V. Vranić, 3d model retrieval, University of Leipzig, Germany, PhD thesis (2004).
[18] D.V. Vranic, Desire: a composite 3d-shape descriptor, in: Proceedings of the ICME, 2005, pp. 962–965.
[19] M. Liu, B.C. Vemuri, S.-i. Amari, F. Nielsen, Shape retrieval using hierarchical total bregman soft clustering, Trans. Pattern Anal. Mach. Intell. 34 (12) (2012) 2407–2419.
[20] P. Papadakis, I. Pratikakis, T. Theoharis, G. Passalis, S.J. Perantonis, 3d object retrieval using an efficient and compact hybrid shape descriptor, in: Proceedings of the 3DOR, 2008, pp. 9–16.
[21] Z. Lian, A. Godil, T. Fabry, T. Furuya, J. Hermans, R. Ohbuchi, C. Shu, D. Smeets, P. Suetens, D. Vandermeulen, S. Wuhrer, Shrec'10 track: Non-rigid 3d shape retrieval, in: Proceedings of the 3DOR, 2010, pp. 101–108.
[22] A.M. Bronstein, M.M. Bronstein, L.J. Guibas, M. Ovsjanikov, Shape google: geometric words and expressions for invariant shape retrieval, Trans. Gr. 30 (1) (2011) 1.
[23] C. Li, A.B. Hamza, Intrinsic spatial pyramid matching for deformable 3d shape retrieval, Int. J Multimed. Inf. Retr. 2 (4) (2013) 261–271.
[24] H. Tabia, D. Picard, H. Laga, P.H. Gosselin, Compact vectors of locally aggregated tensors for 3d shape retrieval, in: Proceedings of the 3DOR, 2013, pp. 17–24.
[25] G. Lavoué, Combination of bag-of-words descriptors for robust partial shape retrieval, Vis. Comput. 28 (9) (2012) 931–942.
[26] A. Agathos, I. Pratikakis, P. Papadakis, S.J. Perantonis, P.N. Azariadis, N.S. Sapidis, Retrieval of 3d articulated objects using a graph-based representation, in: Proceedings of the 3DOR, 2009, pp. 29–36.
[27] S. Jayanti, Y. Kalyanaraman, N. Iyer, K. Ramani, Developing an engineering shape benchmark for cad models, Comput. Aided Des. 38 (9) (2006) 939–953.
[28] R. Toldo, U. Castellani, A. Fusiello, Visual vocabulary signature for 3d object retrieval and partial matching, in: Proceedings of the 3DOR, 2009, pp. 21–28.
[29] Z. Lian, A. Godil, X. Sun, H. Zhang, Non-rigid 3d shape retrieval using multidimensional scaling and bag-of-features, in: Proceedings of the ICIP, 2010, pp. 3181–3184.
[30] S.A. Eslami, N. Heess, C.K. Williams, J. Winn, The shape Boltzmann machine: a strong model of object shape, Int. J. Comput. Vis. 107 (2) (2014) 155–176.
[31] Q. Ke, Y. Li, Is rotation a nuisance in shape recognition? in: Proceedings of the CVPR, 2014, pp. 4146–4153.
[32] H. Van Nguyen, F. Porikli, Support vector shape: A classifier-based shape representation, Trans. Pattern Anal. Mach. Intell. 35 (4) (2013) 970–982.
[33] S. Belongie, J. Malik, J. Puzicha, Shape matching and object recognition using shape contexts, Trans. Pattern Anal. Mach. Intell. 24 (4) (2002) 509–522.
[34] A. Torsello, E. Rodola, A. Albarelli, Sampling relevant points for surface registration, in: Proceedings of the International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission, 2011, pp. 290–295.
[35] E. Rodola, S.R. Bulò, D. Cremers, Robust region detection via consensus segmentation of deformable shapes, Comput. Gr. Forum 33 (5) (2014) 97–106.
[36] X. Bai, C. Rao, X. Wang, Shape vocabulary: A robust and efficient shape representation for shape matching, Trans. Image Process. 23 (2014) 3935–3949.
[37] X. Bai, S. Bai, Z. Zhu, L.J. Latecki, 3d shape matching via two layer coding, Trans. Pattern Anal. Mach. Intell. (2015).