Pacific Graphics 2015
N. J. Mitra, J. Stam, and K. Xu
(Guest Editors)

Volume 34 (2015), Number 7

Projective Feature Learning for 3D Shapes with Multi-View Depth Images

Zhige Xie[1,2] Kai Xu[†1,2] Wen Shan[3] Ligang Liu[3] Yueshan Xiong[2] Hui Huang[1]

[1]Shenzhen VisuCA Key Lab / SIAT [2]School of Computer, National University of Defense Technology
[3]School of Mathematical Sciences, University of Science and Technology of China

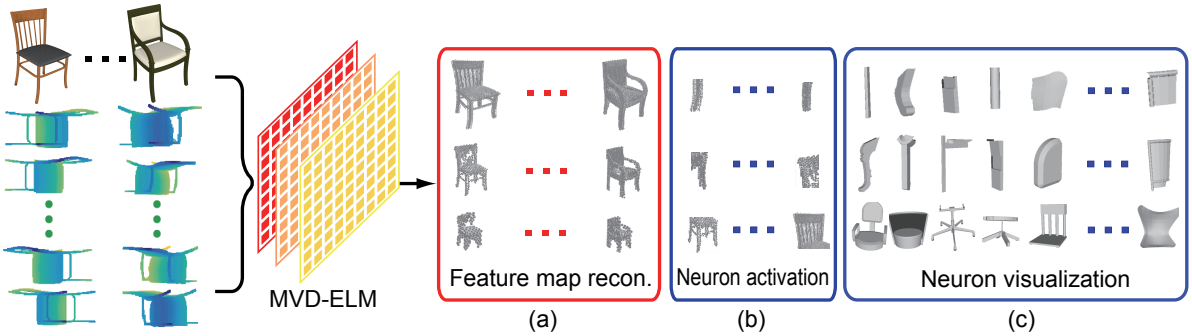

Figure 1: *We propose an projective feature learning method, called MVD-ELM, for learning 3D shape features from multi-view depth images. (a) Our network ensures that the unprojection of the feature maps in each layer together form a reconstruction of the input 3D shapes. (b) Point clouds reconstructed from the maximally activated regions of feature maps (neurons). (c) Visualization of cropped shape parts according to the maximally activated neurons.*

Abstract

Feature learning for 3D shapes is challenging due to the lack of natural paramterization for 3D surface models. We adopt the multi-view depth image representation and propose Multi-View Deep Extreme Learning Machine (MVD-ELM) to achieve fast and quality projective feature learning for 3D shapes. In contrast to existing multi-view learning approaches, our method ensures the feature maps learned for different views are mutually dependent via shared weights and in each layer, their unprojections together form a valid 3D reconstruction of the input 3D shape through using normalized convolution kernels. These lead to a more accurate 3D feature learning as shown by the encouraging results in several applications. Moreover, the 3D reconstruction property enables clear visualization of the learned features, which further demonstrates the meaningfulness of our feature learning.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Modeling—Computational Geometry and Object Modeling

1. Introduction

3D shape understanding has been a long-standing research topic in both computer vision and computer graphics. A common endeavor therein is to extract characterizing features of shape geometry and/or structure. 3D shape features, being global or local, have so far been predominantly human designed, or hand-crafted, capturing some specific aspects of 3D shapes, such as geometry [GCO06], topology [BGSF08], part-level structure [THW*14], etc.

† Corresponding author: kevin.kai.xu@gmail.com

The recent success of automatic feature learning has aroused intensive interest in machine learning and computer vision community. Instead of designing features according to human prior, these approaches strive to learn features automatically from a collection of training data. This especially benefits from the fast development of deep learning techniques [BCV12], where the learned features lead to significant performance boost in classification and recognition tasks [KSH12a, SWT14].

When adapting deep learning techniques to feature learning of 3D shapes, however, the first and foremost challenge is that 3D models, typically represented as 2D manifold, do not come with a canonical representation like 2D images. It is unclear how to utilize the existing techniques of surface parameterization or sampling to encode 3D geometry information of different 3D models into matrices with uniform dimension, which is required as input for most deep learning methods [KSH12a].

To mitigate such issue, the most natural thought is to convert 3D shapes into image representation, which has been attempted by a few recent works, demonstrating promising results. For example, Wu et al. [WSK*15] successfully apply Convolutional Deep Belief Network (CDBN) over volumetric representation of 3D shapes, or 3D images. However, due to the additional dimension of input data, the CDBN training is very time-consuming, which limits the voxelization resolution to only $48 \times 48 \times 48$ in their implementation. Zhu et al. [ZWB*14] render 3D shapes into multi-view depth images and then perform Auto-Encoder over the 2D images, achieving very good performance in 3D shape retrieval. This projective approach, however, may lose the 3D information since the projected views are treated independently. Therefore, their method has to combine hand-crafted image features to realize 3D shape classification.

Biological studies conclude that 3D shape perception in human visual cortex can be formed by depth cues from multiple views [WDC*05]. In light of this, we also adopt multi-view depth representation. To deal with the above issue and achieve a powerful feature learning, we develop a convolutional neural network called Multi-View Deep Extreme Learning Machine (MVD-ELM), which is derived from the single-layer feedfoward network ELM [HZS06]. Our method possesses the following key features:

- First, MVD-ELM maintains multiple channels, each for a projection view, which are interlinked with each other via shared weights for all convolution layers.
- Second, by introducing normalized convolution, the unprojection of feature maps from all channels can reconstruct the input shape in each hidden layer.
- Third, due to the fast training of ELM, our method trains faster than existing deep learning methods by two orders of magnitude. This allows us to use much higher resolution of depth images per view (e.g., 128×128).
- Last, our method can achieve pixel-wise feature extraction when realizing a fully convolutional [LSD15] version of MVD-ELM. Such pixel-wise features support multi-label prediction on a single 3D shape, which is especially useful for 3D shape segmentation.

Experiments demonstrate that the learned features by our method significantly outperform hand-crafted ones for 3D shape classification. For 3D shape segmentation, our learned features achieve comparable results with traditional methods employing feature selection from tens of different hand-crafted features. In addition, our method is fast and easy-to-implement, making it practical in real applications. The source code and dataset are both available online [XXS*15].

2. Related Work

Multi-view 3D shapes analysis. By treating a 3D shape as a collection of 2D projections rendered from multiple directions, multi-view projective analysis has been widely adopted for 3D shape analysis and understanding. Cyr and Kimia [CK01] utilize multi-view projections to identify 3D objects and their poses. Chen et al. [CTSO03] propose Light Field Descriptor (LFD), which is a rotation-invariant 3D shape descriptor defined with 2D contours of multi-view projections. Projective shape descriptor is also used for sketch-based 3D shape retrieval [ERB*12] which compares object projections with query hand-drawn sketches. Recently, Wang et al. [WGW*13] use a pre-labeled image database to achieve projective shape segmentation. Our 3D shape segmentation is also projective. The main difference against their work is that our method uses learned feature maps to achieve pixel label prediction, instead of directly matching a labeled exemplar images.

Deep learning with 3D data. Researchers have successfully built various deep models, such as Convolutional Neural Network (CNN) [LBBH98], Deep Auto-Encoder Networks [HS06], Deep Belief Nets (DBN) [HOT06] and Extreme Learning Machine (ELM) AutoEncoder [CZH13], to learn data-driven features. Such features have demonstrated superior discriminatory power for 2D images and shapes [BCV12]. It is only very recent that a few works attempt to learn 3D shape features via deep learning methods. Wu et al. [WSK*15] work with volumetric representation of 3D shapes and adopt 3D Deep Belief Nets (DBN) [HOT06], obtaining good results of shape classification on Princeton ModelNet [WSK*15]. Zhu et al. [ZWB*14] utilize Auto-Encoder to learn 3D shape feature with multi-view depth images, leading to accurate 3D shape retrieval. Su et al. [SMKLM15] propose multi-view CNN for 3D shape recognition where the multiple views features are integrated with an extra CNN.

Multi-label learning via deep learning. Deep neuron networks, such as CNN, have demonstrated promising performance in image classification [KSH12a]. Recently, there

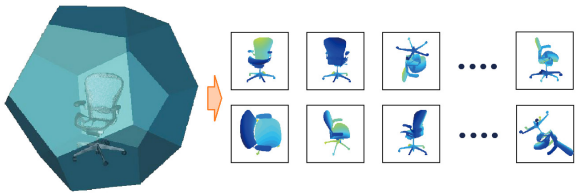

Figure 2: *For a 3D object, a series of 2.5D depth images are captured from the viewpoints which are uniformly sampled on fixed dodecahedrons.*

have been a few works on multi-label image segmentation [LSD15, CPK*14, WXH*14]. These works transform fully connected layer in CNN into convolution layers, which enables a classification net for pixel-wise label prediction. Such pixel-wise labeling can be used for labeled segmentation of the input images. We are not aware of any work on applying deep learning in multi-label 3D shape segmentation. In achieving that, we realize fully convolutional MVD-ELM to predict pixel-wise labels for multi-view depth images.

3. ELM and ELM-based feature learning

ELM was originally proposed as a single-hidden layer feedforward neural network (SLFNs) [HZS06]. Different from other neural networks with back propagation (BP) [RHW88], the hidden nodes in ELM need not to be adjusted but can be randomly generated, as long as the activation functions of the neurons are nonlinear piecewise continuous. The weights between the hidden layer and the output layer can be optimized with a closed-form solution. ELM contains two phases, feature mapping and learning.

ELM feature mapping. Given input data $\mathbf{x} \in \mathbf{R}^D$, the output function of ELM for generalized SLFNs is:

$$f(\mathbf{x}) = \sum_{i=1}^{K} \beta_i h_i(\mathbf{x}) = \mathbf{h}(\mathbf{x})\beta, \tag{1}$$

where $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), \cdots, h_K(\mathbf{x})]$ is the output vector of the hidden layer and $\beta = [\beta_1, \cdots, \beta_K]^T \in R^{K\times M}$ the vector of the output weights between the hidden layer (K nodes) and the output layer (M nodes). $\mathbf{h}$ is called ELM feature mapping which maps the input data in $\mathbf{R}^D$ to the feature space $\mathbf{R}^K$. Any nonlinear piecewise continuous functions (e.g. Sigmoid, Gaussian, etc.) can be chosen as $\mathbf{h}$ to generate feature maps. In ELM, the parameters of $\mathbf{h}$ are randomly generated based on a continuous probability distribution.

ELM learning. The second phase is supervised and is task-specific. Let us denote $\mathbf{T} \in R^{N\times M}$ as the application-dependent target matrix provided by N training data. $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1), \cdots, \mathbf{h}(\mathbf{x}_N)]^T \in R^{N\times K}$ contains N random feature maps generated in the first phase. The objective function to minimize is the weighted sum of the training error and the norm of output weights:

$$\omega\|\mathbf{H}\beta - \mathbf{T}\|_2^2 + \|\beta\|_2^2. \tag{2}$$

β can be obtained in a closed-form solution:

$$\beta = \begin{cases} (\mathbf{H}^T\mathbf{H} + \frac{1}{\omega}\mathbf{I})^{-1}\mathbf{H}^T\mathbf{T}, & L \le k, \\ \mathbf{H}^T(\mathbf{H}\mathbf{H}^T + \frac{1}{\omega}\mathbf{I})^{-1}\mathbf{T}, & L \ge k, \end{cases} \tag{3}$$

where $\mathbf{I}$ is identity matrix with respective dimension.

Local Receptive Fields (LRF) based ELM. Huang et al. [HBKV15] developed a feature learning framework called ELM-LRF. ELM-LRF relays two key characteristics of ELM. First, ELM can use different types of local receptive fields as long as they are randomly generated according to any continuous probability distribution [Hua14]. Second, each hidden node in ELM can be a combination of several hidden nodes (a subnetwork) [HC07]. These two features correspond to the convolution and pooling operations in CNN, respectively. ELM-LRF provides a deterministic solution for feature learning tasks based on local receptive fields. Same as ELM, ELM-LRF has only one hidden layer.

Multi-layer ELM-LRF. The ELM based feature learning model can be extended to a deep one with multiple hidden layers. Similar to other deep models such as CNN [KSH12b], the output feature maps of the previous layer can be used as input to the next layer. The final layer feature maps are used to train the optimized output weights. The link between different layers can be sparse combinations of feature maps (i.e., LRF) [LBBH98]. Pooling layers are inserted to extract features with increasing levels of abstraction [LBBH98, KSH12b]. Various types of pooling, such as max pooling, average pooling, etc., could be employed.

The idea of deep ELM was first proposed and realized in [CZH13] based on ELM Auto-Encoder. Such deep network extracts feature encoding in a global fashion. In contrast, our multi-layer ELM is built upon ELM-LRF which can achieve both global and local (pixel-wise) feature extractions thanks to the local connection of LRF.

4. Feature learning with MVD-ELM for 3D shape classification

Our MVD-ELM is designed to train for the task of 3D shape classification. Therefore, the features our method learns are expected characterize shape classes as much as possible.

4.1. Method overview

Projective feature learning for 3D shapes over 2D images is non-trivial. One basic requirement is that the multiple views should be interlinked so that 3D information could be retained throughout the training process in a deep network. In this section, we first describe the input representation of our

method, and then introduce the two key components contributing to achieve 3D information preservation.

Input representation. Given a set of 3D shapes, we generate for each shape a set of 2.5D depth images through projecting it along multiple view directions uniformly sampled from a sphere centered at the shape. The input to our MVD-ELM is multi-view projected depth images for all shapes. In our implementation, view points are sampled at vertices of a unit regular dodecahedron and view directions toward the dodecahedron center as shown in Figure 2. Thus, for each 3D shape, $D = 20$ depth images (with resolution of $d \times d$) are generated with OpenGL depth buffer. Background pixels are set to maximum depth. We denote the depth image at the n-th view as $x_n \in R^{d \times d}$. For each view n, we make a foreground mask $\mathbf{m}_n$ to rule out background pixels after convolution and pooling operations.

To enable effective feature learning, all training shapes are uniformly scaled into the unit box. Note that the shapes need not to be consistently oriented especially when the training set is large, due to the rotation-invariant nature of multi-view learning. To facilitate unprojection in neuron visualization, we record the projection matrices of all views.

Normalized convolution kernels. In training a convolutional neural network, all training images need to be normalized by subtracting their mean and then being divided by their variance [KSH12a]. However, this does not apply to our case since such nonlinear operation would destroy the depth geometry in each view. Moreover, the multi-view depth images are obtained using nonlinear viewing projection. Directly applying convolution over depth images will break their alignment after unprojection, making 3D reconstruction infeasible, as shown in the right part of Figure 3.

To retain the geometry information for all 2.5D feature maps throughout the MVD-ELM network, we opt to normalize the convolution kernels used in various convolution layers. This way, the feature maps can always preserve the 3D geometry of the input depth images. In the left part of Figure 3, we show the reconstruction of an input 3D model via unprojection of multi-view feature maps generated with normalized convolution kernels.

Multiple hidden layers. The single layer framework of ELM-LRF is unsuitable for learning 3D shape features, since 3D shapes usually possess very complex structure, which should be better characterized by a hierarchy of features with varying levels of abstraction. To achieve both global and local feature extraction in a multi-layer fashion, we introduce multi-layer ELM-LRF.

In multi-layer ELM-LRF, we adopt random convolutional for feature mapping, which is nonlinear piecewise continuous [HBKV15]. Moreover, according to [HBKV15], ELM-LRF with local randomly connected hidden nodes can be regarded as a specific type of ELM, satisfying the requirement

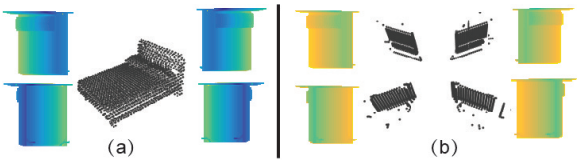

Figure 3: *The point cloud unprojected from four feature maps generated from a bed model based on different convolutional kernels. (a) Reconstruction of an input bed model by feature maps from four views when applying normalized convolution kernels. (b) Directly performing general convolution over depth images will break their alignment after unprojection, making the 3D reconstruction infeasible.*

of universal approximation and classification capabilities. Stacking such layers of non-linear hidden nodes also satisfies the requirements, making our multi-layer ELM-LRF possess learning capability.

Multiple view channels. To adapt to multi-view learning, we devise multiple view channels in our network, each learned with a multi-layer ELM-LRF. To ensure the feature maps learned for multiple views are mutually interlinked, we adopt shared weights (convolution kernels) among different views during feature mapping. There are two benefits using shared weights among different view channels. First, the depth images in different views can be consistently mapped to a shared feature space, making the unprojections of feature maps at any layer form a valid 3D reconstruction of the input shape. Second, with shared wights, the output weights of multiple view channels can be optimized in a unified closed-form solution, which greatly reduces the training time and makes our method scalable to number of views.

4.2. Formulation of MVD-ELM

We introduce the formulation of our MVD-ELM in this subsection. Figure 4 shows the architecture of MVD-ELM. The input data is $N = M \times D$ depth images (M shapes and D views per shape) represented with a matrix of $(d \times d \times N)$, i.e., an array of N depth images of $d \times d$ resolution. We now elaborate on the critical components of MVD-ELM including feature mapping, pooling, learning and testing.

Multi-view feature mapping. Suppose the MVD-ELM contains L layers of convolution and pooling. Let us take the l-th layer for example. It takes the feature map (input depth image for the first layer) of the $(l-1)$-th layer as input, and generates K_l feature maps (with resolution of $d_l \times d_l$) through convolution between input image and the randomly generated normalized convolutional kernels at this layer. Thus we obtain $N \times K_l$ different $d_l \times d_l$ feature maps in the l-th layer. The random convolutional kernels are generated similar to [HZS06], each of which is normalized so that

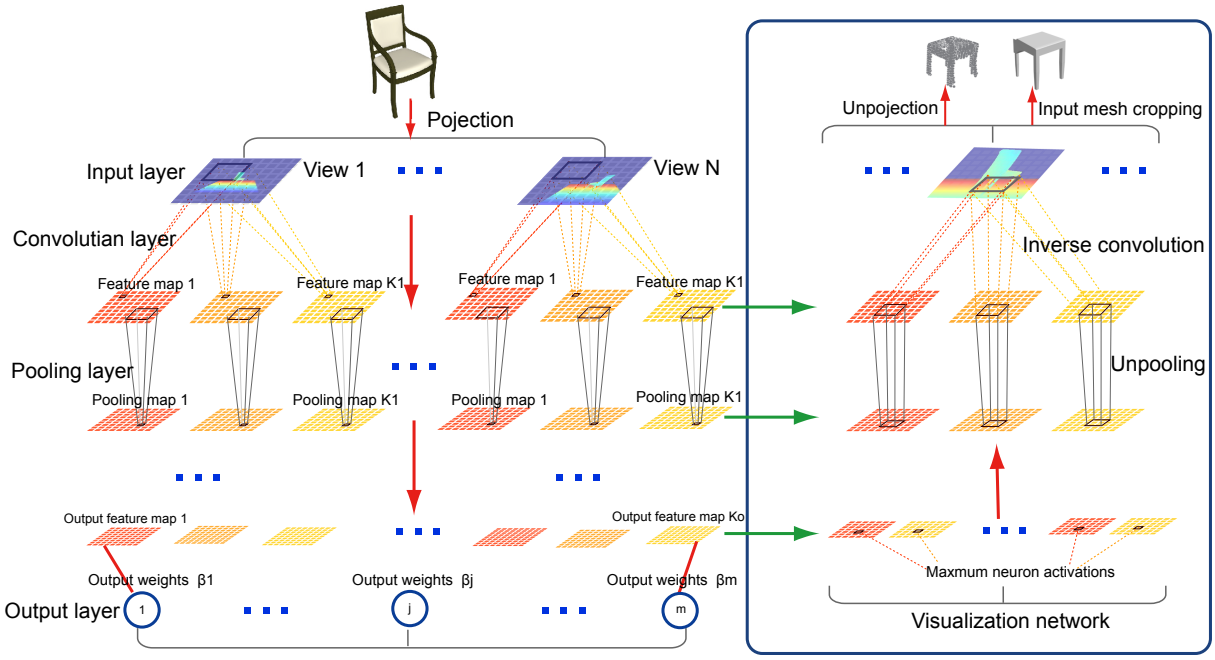

Figure 4: *The architecture of MVD-ELM learning (left part) and visualization (right part).* ***Left:*** *The projected multi-view depth images for a given input 3D shape is input to their respective view channels. In each view channel, the depth image goes through a cascading layers of convolution and pooling. Each convolution layer produces K_l feature maps with the random generated convolution kernels. When reaching to the last layer, the output feature maps are used to compute the optimized output weights.* ***Right:*** *The visualization network runs in the opposite direction of the training network, with inverse convolution and unpooling operations, until it reaches the input layer where we obtain the maximally activated regions in the depth images. These regions are then unprojected onto the input shapes and used to crop the shapes into characteristic parts.*

their weights sum to 1, and are shared among all view channels in the same layer. The normalized random convolution kernels for a given layer l is denoted as

$$\mathbf{W}_l = [\mathbf{w}_{l,k}]_{k=1}^{K_l} \subset R^{c_l \times c_l \times K_l}, \quad l = 1, \cdots, L \tag{4}$$

which contains K_l normalized convolutional kernels $\mathbf{w}_{l,k}$ of size $c_l \times c_l$. Specifically, the k-th normalized, random convolutional kernel is generated as following:

$$\begin{aligned} \mathbf{w}_{l,k}(i,j) &= rand(0,1), \quad i,j = 1, \cdots, c_l \\ \mathbf{w}_{l,k}(i,j) &= \mathbf{w}_{l,k}(i,j) / \textstyle\sum_{i,j}(\mathbf{w}_{l,k}(i,j)), \end{aligned} \tag{5}$$

where $rand(0,1)$ generates a random number in $[0,1]$.

Convolution operation is performed over the feature map of the previous layer (Figure 4(left)). If the current layer is the input one, the operation is applied directly on the input depth image. The convolution kernels of layer l are *shared* across all views. Specifically, the k-th feature map at layer l for any given view n can be computed as:

$$\mathbf{F}_{l,k,n} = (\mathbf{F}_{l-1,k,n} * \mathbf{w}_{l,k}) \otimes \mathbf{m}_{l,n}, \quad n = 1, \cdots, D \tag{6}$$

where $*$ is convolution operation and $\otimes$ is element-wise multiplication which applies the foreground mask $\mathbf{m}_{l,n}$ to remove the background. For the input layer, $\mathbf{F}_{1,k,n} = x_n, k = 1, \cdots, K_1$. We next down-sample the feature maps using pooling operation to generate input for the next layer.

Multi-view pooling. We choose average pooling since it introduces relatively small distortion to the depth geometry than other methods such as max pooling. For the l-th layer, we take the pooling size as s_l which leads to pooling maps of size $d_l/s_l \times d_l/s_l$. The k-th pooling map is obtained by applying average pooling operation over the k-th feature map, for view n and layer l:

$$\begin{aligned} \mathbf{P}_{l,k,n}(p,q) = \frac{1}{s_l^2} \textstyle\sum_{i=(p-1)*s_l+1}^{p*s_l} \sum_{j=(p-1)*s_l+1}^{p*s_l} \mathbf{F}_{l,k,n}(i,j). \\ p,q = 1, \cdots, s_l \end{aligned} \tag{7}$$

$\mathbf{P}_{l,k,n}$ is then used as the input feature map of the next layer: $\mathbf{F}_{l+1,k,n} = \mathbf{P}_{l,k,n}$. Note that the foreground mask should also be pooled, in the same way as feature map pooling, to generate the mask for the next layer.

Output weights learning. The output feature maps in the last layer are then used in the MVD-ELM learning phase to train the output weights. The last layer is in full connection with the output layer, as shown in Figure 4(left). We simply concatenate the feature maps of all M training shapes into M row vectors together, obtaining the fully connected layer matrix $\mathbf{H} \subset R^{M \times Z}$, where $Z = (D \cdot K_L \cdot (d_L/s_L)^2)$. K_L, d_L, and s_L are the number of feature maps per view, feature map size, and pooling size, respectively, all for the last layer. Then the output weights β can be computed as:

$$\beta = \begin{cases} (\mathbf{H}^T\mathbf{H} + \frac{1}{\omega}\mathbf{I})^{-1}\mathbf{H}^T\mathbf{T}, & M \geq Z \\ \mathbf{H}^T(\mathbf{H}\mathbf{H}^T + \frac{1}{\omega}\mathbf{I})^{-1}\mathbf{T}, & M \leq Z \end{cases} \quad (8)$$

where ω is the weight defined in Equation (2).

Testing. During the testing process, the multi-view depth images of a test 3D shape go through the whole network so that the features maps $\mathbf{H}$ of the testing data can be obtained. The output labels are those with the maximum probabilities according to $\mathbf{H}\beta^T$.

5. Neuron visualization

Understanding the performance of MVD-ELM requires interpreting the neuron activation in multiple layers. Since the convolutional kernels used in our networks are randomly generated, to better visualize the feature learned by our MVD-ELM, we weight the K_o output feature maps using the optimized output weights β (Equation (8)), leading to K_o activation maps. These activations are then mapped back from the output layer to the input 3D mesh through inverse operations of convolution and pooling, similar to [ZF13]. Figure 4(right) shows the visualization networks in accordance to the MVD-ELM training network.

Formally, given a 3D mesh, let us denote $\mathbf{H}_o$ as the matrix of its output feature maps and β, the learned features can be visualized through the following operations:

$$invConv(unpooling(\cdots invConv(unpooling(\beta\mathbf{H}_o)))) \quad (9)$$

where *invConv* indicates inverse convolution and *unpooling* inverse average pooling operation. Given an input 2D image x and a convolution kernel w, the convolution operation is $y = x * w$. The inverse convolution is then defined as $x' = y * flip(w)$, where $flip$ means left-right flipping over the input convolution kernel along its second dimension. *unpooling* is achieved by placing the average value back into each pixel within the pool region. In Figure 5, we show the activations mapped on the input meshes with color coding.

To visualize the learned discriminative patches of the 3D models, we crop the input 3D models based on the activation maps. Among all the weighted output feature maps $\beta\mathbf{H}_o$, we select the top m with the highest neuron activations (pixel values of feature maps). To visualize the neuron activation of a selected feature map, we keep only the neighboring pixels around the maximum activation pixel and zero out all other pixels, resulting in a masked activation map. Such masked activation map is then passed to the visualization network to reconstruct the activation of the lower layers via performing *unpooling* and *inverseConv* operations, until the input layer is reached. Finally, we unproject the regions of the activation maps into 3D space based on the corresponding projection matrix and used the unprojected area to crop the most strongly activated region on the input 3D model.

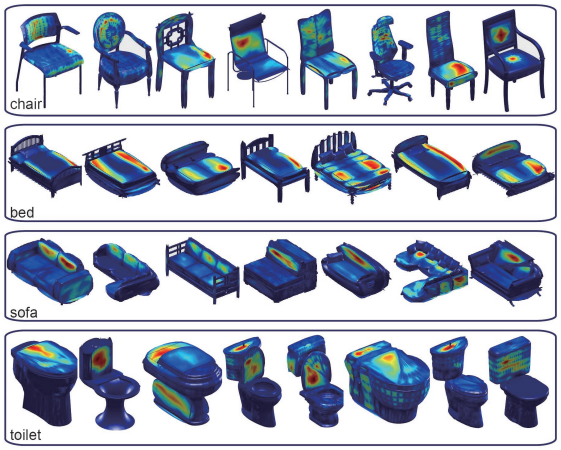

Figure 5: *Color-coded visualization of neuron activations on the input mesh.*

As shown in Figure 6, the learned MVD-ELM features are visualized as point clouds in 3D, as well as cropped parts of input 3D models corresponding to maximally activated regions. Since our MVD-ELM networks are trained discriminatively with shape classification task, these features essentially imply which regions of the input models are discriminative in characterizing shape classes.

6. Fully Convolutional MVD-ELM for 3D shape segmentation

Labeled segmentation is important for 3D shape understanding [CGF09]. Significant progresses has been made recently with the help of machine learning techniques [KHS10], through casting it as a multi-class face labeling problem. However, existing learning-based methods have so far been based on hand-crafted 3D features. Inspired by the work of [LSD15], we extend our networks into a fully convolutional version (referred to as FC-MVD-ELM) to achieve face labeling. Our method is the first that utilizes learned features from deep neural networks for 3D mesh segmentation.

Approach. In [LSD15], the last two fully connected layers in CNN are modified to convolutional layers, leading to a fully convolutional classification net outputting pixel-wise label prediction. Similarly, we devise a fully convolutional MVD-ELM for the task of depth image segmentation. This

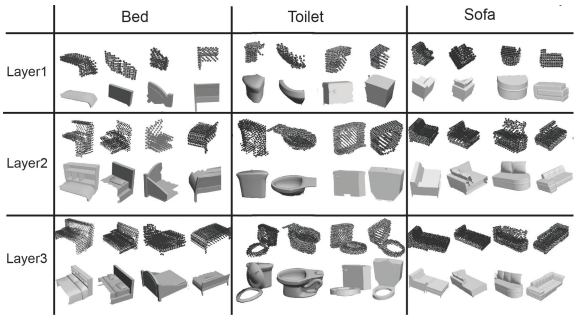

Figure 6: *Visualization of neurons of the first three layers of MVD-ELM, for three categories from ModelNet10, i.e., bed, toilet and sofa. For each category in various layers, we show two visualization results in two rows. The upper row shows point clouds generated from neurons activations in MVD-ELM, while the lower row visualizes cropped parts of input models corresponding to those maximally activated regions.*

network is composed of two convolution layers but without any pooling layer. After predicting pixel-wise labels for the depth images of all views, we can obtain the mesh segmentation based on the mapping between 3D shape and the projected depth images. This shares similar spirit with the projective 3D shape segmentation proposed in [WGW*13]. The depth image segmentation process, similar to 2D image segmentation, proceeds in three steps described below.

Multi-label training. Since the mappings between pixels in depth images and triangles of 3D mesh have been recorded during the depth image generation process, the semantic tags on the triangles can be transferred onto the corresponding pixels, thus generating labeled, training depth images. Let S be the number of training shapes, T the total number of tags, and F the number of feature maps. We mask each feature map with each tag as foreground, ruling out the rest region covered by other tags, leading to $T \times F$ masked depth images as training data. The masked depth images are then fed into MVD-ELM with the corresponding masking tag as target label. This learns the output weights which transform the last-layer feature maps of a given pixel into the probabilities of labels for it. For a given pixel, the corresponding pixels in the last-layer feature maps form its learned features. Now we use these features to achieve segmentation.

Multi-label testing. Given a test 3D shape, we first extract its multi-view testing depth images. For each pixel of a testing image, we utilize the pixels of the output feature maps (having the same resolution as input depth images since there is no pooling layer) of all masked training depth images as learned feature of the pixel, leading to a $S \times T \times F$ dimensional feature vector. Such features are treated as the output layer feature maps of the test depth image, and the final labeling probability can be computed by multiplying

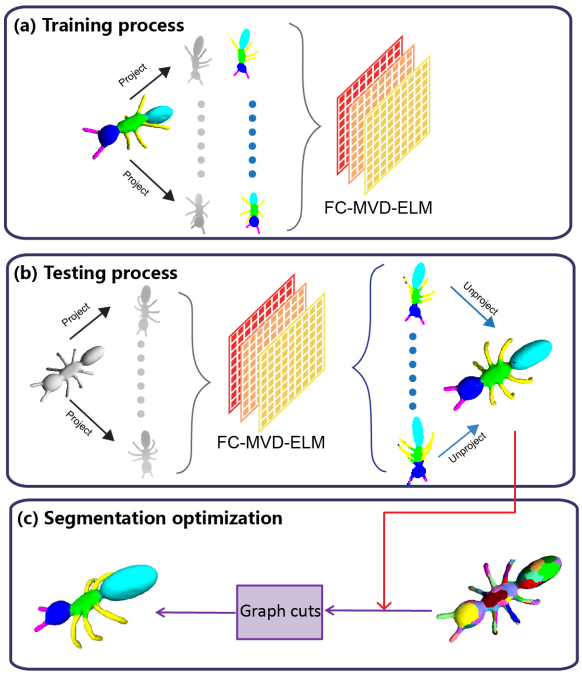

Figure 7: *3D mesh segmentation using FC-MVD-ELM. (a)After generating multi-view depth images with labels from training meshes, FC-MVD-ELM is trained using them as input. (b) FC-MVD-ELM will predict labels on multi-view depth images of testing mesh. These labels are unprojected back into original mesh, forming initial segmentation result. (c) At last stage, graph cuts optimization is utilized to obtain final smooth segmentation results.*

it with the learned output weights, as in Equation 8. This results in T probabilities for each pixel of the test depth image. The final pixel-wise label simply takes the tag with the maximum probability. This step is illustrated in Figure 7 (b).

Label back-projection and final segmentation. After the multi-view depth images have been labeled, we unproject these labels back onto the corresponding triangles of the test 3D shape and then compute the final segmentation using graph-cut. Specifically, the average of the probabilities predicted by multiple views serves as the data term of the graph-cut formulation. To produce smooth segmentation boundaries, we employ the geometric smoothness term of [XXLX14]. The final segmentation integrates the segmentations estimated from multiple views; see Figure 7 (c).

7. Experiments and evaluations

The 3D shape features learned by the proposed MVD-ELM can be used in a wide range of applications. In this section we evaluate the performance of our MVD-ELM feature learning and explore its applications. We have implemented

MVD-ELM using MATLAB 2014b running on a computer with Intel(R) Core i5 3.2 GHz CPU and 32GB RAM. Other procedures, such as multi-view projection and unprojection, are implemented with C++ and OpenGL.

7.1. 3D shape classification

Parameter setting. We adopt a four-layer MVD-ELM network ($L = 4$) in our implementation. The number of views is set to $D = 20$ and the regularization weight $\omega = 0.01$. Table 1 summerizes the parameters used in our implementation.

Param.	Layer 1	Layer 2	Layer 3	Layer 4
FM	64×64	32×32	16×16	8×8
K_l	2	2	2	2
c_l	4	4	4	4
s_l	2	2	2	2

Table 1: *Feature map size (FM), number of convolution kernel (K_l), normalized convolution kernel size (c_l) and pooling size (s_l) of various layers used by our implementation. The resolution of input images is* 128×128.

Dataset. To verify the representation capability of the shape features learned by MVD-ELM, we run our algorithm on a large collection of 3D shapes with relatively large intra-class variation, Princeton ModelNet, a recently available online shape dataset [WSK*15] containing 127,915 CAD models in 662 categories. We run our algorithm on its two subsets: ModelNet10 and ModelNet40 (Table 2). Models in ModelNet10 are all adjusted manually with consistent orientation while those in ModelNet40 are not oriented.

Dataset	Oriented	#Models	#Training	#Testing
ModelNet10	Yes	4807	3899	908
ModelNet40	No	10695	8567	2128

Table 2: *Statistics of the two subsets of Princeton ModelNet.*

	Hand-crafted		Learned features	
	SPH	LFD	CDBN	MVD-ELM
ModelNet10	79.97%	79.87%	86.5%	88.99%
	–	–	2 days	674 sec.
ModelNet40	68.23%	75.47%	77.32%	81.39%
	–	–	> 2 days	306.4 sec.

Table 3: *Comparing with SVM classifier trained over hand-crafted features and with CDBN with volumetric representation. Note that the CDBN is trained on GPU. The ModelNet40 experiment was executed on a workstation with 128GB RAM (single-thread).*

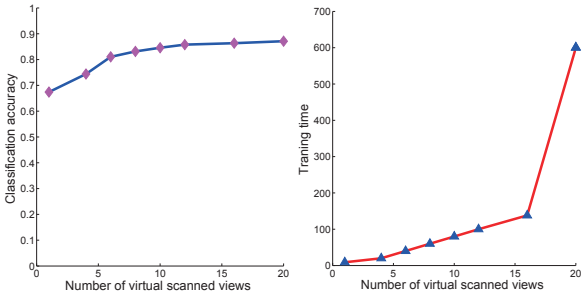

Figure 8: *Performance of MVD-ELM on ModelNet10 with varying view counts. The dramatic increase of training time around* 18 ~20 *views is due to the limitation of main memory.*

Comparison with alternative classification methods. For ModelNet10, our MVD-ELM takes 674 seconds for training, obtaining 88.99% classification accuracy. We compare our method against the CDBN with volumetric 3D representation proposed by Wu et al. [WSK*15]. Their method takes about two days to train the networks and achieves 83.54% classification accuracy. We also compare our learned features against two hand-crafted shape descriptors, i.e., Light Field Descriptor (LFD) [CTSO03] and Spherical Harmonics (SPH) [KFR03]. For LFD and SPH, we use linear SVM with one-vs-all strategy to train classifiers and evaluate the classification accuracy on the testing set. From Table 3, MVD-ELM achieves better accuracy than both descriptors.

To better demonstrate the performance of our learned features, we train MVD-ELM with lower resolution (48×48) of depth images, which is comparable to the voxelization resolution of CDBN [WSK*15]. Meanwhile, we train CNN [LBBH98] on the same training data. Furthermore, we also train linear SVM classifiers using the data-driven features learned by our MVD-ELM, i.e., random feature maps weighted by learned output weights. Table 4 shows the classification accuracy and training time of different methods. The results verify that our features present good representation capability, even with relatively low resolution 2D projections. Moreover, our method outperforms alternative approaches, including both deep and shallow models.

Dataset	MVD-ELM	CDBN	CNN	SVM
ModelNet10	87.89%	86.5%	82.2%	84.2%
	58 sec.	2 days	38 min.	163 sec.

Table 4: *Classification results of different methods at comparable input resolutions. SVM is trained on the learned features by our MVD-ELM. CDBN is trained on GPU. The training time of SVM does not include the time spent on feature learning of our method.*

Comparison with single layer ELM-LRF. Stacking multiple layers of ELM-LRF enables our method to learn a hi-

erarchy of features with varying levels of abstraction, which is expected to be more powerful in characterizing complex shapes than single layer ELM-LRF. To verify this, we compare multi-layer ELM-LRF with single layer one on the MNIST dataset [LBBH98], which contains 60K training and 10K testing handwriting images. We use 40 convolution kernels for both methods. For multi-layer ELM-LRF, we choose a two-layer ELM-LRF version. As shown in Table 5, two-layer ELM-LRF achieves better classification performance than the single layer one.

Dataset	single-layer ELM-LRF	multi-layer ELM-LRF
MNIST	98.26% 385.42 sec.	98.89% 414.59 sec.

Table 5: *Comparison between multi-layer and single-layer ELM-LRF over the MNIST dataset.*

We also compare single-layer, two-layer and four-layer versions of MVD-ELM over ModelNet10. For all these versions of MVD-ELM, the number of normalized convolution kernels takes 8 while the size of feature maps in the output layer is set to 8×8. As demonstrated in Table 6, multi-layer MVD-ELM achieves better classification performance on ModelNet10. Because of multiple layer feature mapping, multi-layer ELM-LRF usually takes more time to train.

Dataset	single layer	two-layer	four-layer
ModelNet10	87.22% 181 sec.	87.89% 283 sec.	88.99% 674 sec.

Table 6: *Comparison of single-layer, two-layer and four-layer versions of MVD-ELM on the ModelNet10 dataset.*

Number of projection views. For multi-view learning method, the output accuracy is expected to improve as the number of views increases. To demonstrate the effect of view count, we run our MVD-ELM on ModelNet10 with different numbers of projection views D. As shown in Figure 8, the classification accuracy of MVD-ELM grows as D increases.

Rotation invariance of MVD-ELM. To test the rotation invariance of MVD-ELM, we run it over ModelNet40 where the models are not oriented. Similar to [WSK*15], we rotate each model per 30 degrees to generate more model instances in arbitrary poses. The training set and testing set are arranged to avoid overlapping models, even for same model with different poses. This experiment is executed on a workstation with 128GB RAM (single-thread). As shown in Table 3 (bottom row), our method takes 306.4 seconds for training and obtains 81.39% classification accuracy, while CDBN [WSK*15] achieves 77.32%, LFD [CTSO03] 75.47% and SPH [KFR03] 63.59%.

	Ours	Training	Ours(75%)	SB19
Ant	93.2%	197.6 sec.	91.7 %	91.5 %
Bird	86.1%	284.2 sec.	83.8 %	92.5%
Fish	88.4%	209.1 sec.	87.1%	96.7%
Octopus	92.3%	199.5 sec.	90.6%	98.4%
Teddy	91.0%	212.8 sec.	89.9%	98.1 %
Glasses	90.4%	169.1 sec.	89.3%	97.4%

Table 7: *Segmentation results and time consumption of proposed FC-MVD-ELM on some typical categories of 3D meshes from PSB [CGF09]. Our segmentation results is comparable to those by [KHS10] which utilizes tens of types of hand-crafted features. Our method is purely based on our learned features and is also much faster in training time.*

7.2. 3D shape segmentation

Dataset. To evaluate the performance of the proposed FC-MVD-ELM in labeled segmentation of 3D shapes, we test it on six categories (20 models per category) from the Princeton Segmentation Benchmark (PSB) [CGF09]. The ground truth labeled segmentations used for training and evaluation are provided by [KHS10].

Results and timings. In fully convolutional CNN [LSD15], the authors combined high layer label predictions with low layer label predictions to generate high accuracy multi-label prediction for every pixel in testing images. Since the training data for our case is small (19 depth images per view for leave-one-out training), we do not adopt multi-layer prediction integration as in [LSD15]. Therefore, we adopt a two-layer FC-MVD-ELM without pooling layer to directly predict pixel-wise labels in full resolution of depth images. See Figure 9 for visual results of our segmentation method.

Due to the fast training speed of ELM, our training over 19 meshes takes only several minutes. After graph-cut optimization on segmentation boundaries, we obtain around 90% labeling accuracy on several categories of PSB segmentation benchmark, as shown in Table 7. To verify the robustness of our method, we also report the testing accuracy when the training set takes 75% of the whole set, corresponding to the column of "Ours(75%)" in Table 7. Our method performs comparably with [KHS10], but, more importantly, with much less training time and no hand-crafted feature, demonstrating both practical and theoretical merits.

7.3. Segmentation and recognition of single-view depth images

Recently, deep learning methods have been used for the task of object detection from 2.5D depth images [GGAM14], where a large CNN pre-trained on RGB images can be adapted to learn features for depth images. Our MVD-ELM can also be applied for object detection on RGBD data. To do so, we deploy a single-view MVD-ELM network

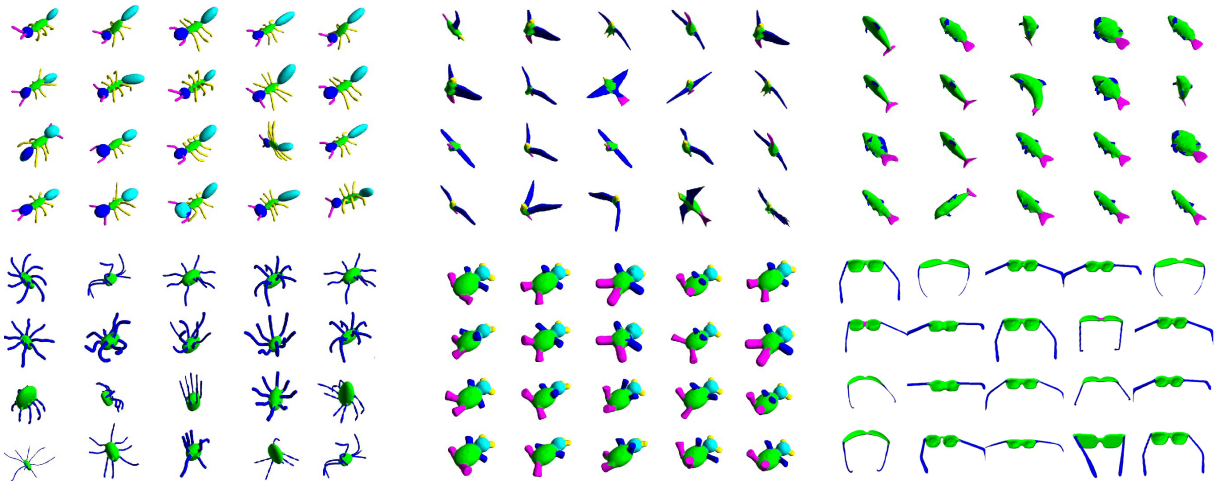

Figure 9: *Segmentation results by our FC-MVD-ELM on six categories from PSB [CGF09].*

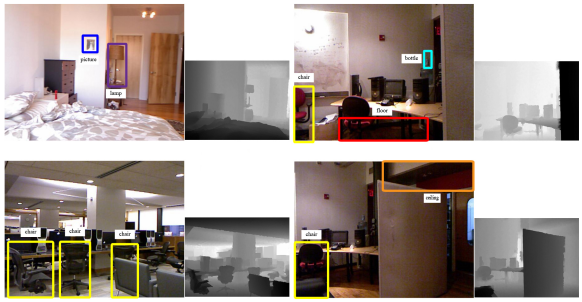

Figure 10: *Results of object detection from single-view depth images using single-view MVD-ELM. We only use the depth information in the NYU2 dataset [SHKF12].*

($D = 1$) and run it over the NYU2 dataset [SHKF12] containing 8965 depth images for training and 14878 for testing. MVD-ELM takes only 30 seconds for training and achieves 17.33% recognition accuracy. In comparison, the method in [GGAM14], which is the state-of-the-art on this task, achieves 20.1% recognition accuracy (only using depth information) but takes 7.5 hours for training their CNN network. Figure 10 shows some examples of object recognition from single 2.5D depth images.

8. Conclusions

We adopt the multi-view depth image representation and propose a multi-view deep neural network based on ELM-LRF to achieve fast and quality projective feature learning for 3D shapes. In contrast to existing 3D shape feature learning methods, our method ensures the feature learning for different views are mutually dependent via shared weights. Moreover, the unprojections of feature maps in each layer together form a valid 3D reconstruction. This leads to a more accurate 3D feature learning as shown by the visualization and applications. In addition, our training is faster than existing deep learning methods by about two orders of magnitude, making it much more practical in real applications.

Limitations. Our method has a few limitations. First, there are many parameters in the algorithm need to be tuned. This is the case for all deep learning methods. However, the fast training of MVD-ELM makes its parameter tuning easier. Second, since the internal geometry and structure of 3D shapes may be not fully visible due to occlusion in view-based projection, our feature learning may not be able to capture all the characterizing features of a 3D shape. Last, our method requires many views of depth images as input, which is difficult to capture in real scenarios. However, with fast development of 3D scanning techniques, even real-time acquisition devices such as Microsoft Kinect, 3D data can be obtained more easily.

Future work. Firstly, it is worth to develop more applications based on our deep feature learning model. Some possible applications include 3D object recognition in cluttered indoor scenes, 3D scene understanding for robot navigation and 3D object analysis for robot manipulation, etc. In robotics setting, it is also interesting to develop online feature learning where the training data may come from robot actions. Secondly, it is interesting to develop new methods for learning comprehensive 3D features covering full geometry and structure of input shapes through, for example, generating a hierarchy of depth images from cameras placed at different distances. Finally, we are interested in utilizing our deep projective learning model to connect 2D images with 3D shapes, thus opening more research opportunities towards 2D-3D data fusion.

Acknowledgement

We thank the anonymous reviewers for valuable comments. This work was supported in part by NSFC (61202333, 61222206, 11426236, 61379103), National 973 Program (2015CB352501), Guangdong Sci. and Tech. Program (2014B050502009, 2014TX01X033), Shenzhen VisuCA Key Lab (CXB201104220029A) and One Hundred Talent Project of Chinese Academy of Sciences.

References

[BCV12] BENGIO Y., COURVILLE A. C., VINCENT P.: Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR abs/1206.5538* (2012). 2

[BGSF08] BIASOTTI S., GIORGI D., SPAGNUOLO M., FALCIDIENO B.: Reeb graphs for shape analysis and applications. *Theoretical Computer Science 392*, 1-3 (2008), 5–22. 1

[CGF09] CHEN X., GOLOVINSKIY A., FUNKHOUSER T.: A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28*, 3 (Aug. 2009). 6, 9, 10

[CK01] CYR C. M., KIMIA B. B.: 3d object recognition using shape similiarity-based aspect graph. In *ICCV* (2001), pp. 254–261. 2

[CPK*14] CHEN L., PAPANDREOU G., KOKKINOS I., MURPHY K., YUILLE A. L.: Semantic image segmentation with deep convolutional nets and fully connected CRFs. *CoRR abs/1412.7062* (2014). 3

[CTSO03] CHEN D.-Y., TIAN X.-P., SHEN Y.-T., OUHYOUNG M.: On visual similarity based 3d model retrieval. *Computer Graphics Forum 22*, 3 (2003), 223–232. 2, 8, 9

[CZH13] CHAMARA L. L., ZHOU H., HUANG G.-B.: Representational learning with ELMs for big data. *Intelligent Systems, IEEE 28*, 6 (Nov 2013), 31–34. 2, 3

[ERB*12] EITZ M., RICHTER R., BOUBEKEUR T., HILDEBRAND K., ALEXA M.: Sketch-based shape retrieval. *ACM Trans. Graph. (SIGGRAPH) 31*, 4 (2012), 31Í–40. 2

[GCO06] GAL R., COHEN-OR D.: Salient geometric features for partial shape matching and similarity. *ACM Trans. Graph. 25*, 1 (2006), 130–150. 1

[GGAM14] GUPTA S., GIRSHICK R. B., ARBELAEZ P., MALIK J.: Learning rich features from RGB-D images for object detection and segmentation. In *ECCV* (2014). 9, 10

[HBKV15] HUANG G.-B., BAI Z., KASUN L. L. C., VONG C. M.: Local receptive fields based extreme learning machine. *IEEE Computational Intelligence Magazine 10*, 2 (2015), 18–29. 3, 4

[HC07] HUANG G.-B., CHEN L.: Convex incremental extreme learning machine. *Neurocomputing 70*, 16-18 (2007), 3056–3062. 3

[HOT06] HINTON G. E., OSINDERO S., TEH Y.-W.: A fast learning algorithm for deep belief nets. *Neural Computation 18*, 7 (2006), 1527–1554. 2

[HS06] HINTON G. E., SALAKHUTDINOV R. R.: Reducing the dimensionality of data with neural networks. *Science 313*, 5786 (2006), 504–507. 2

[Hua14] HUANG G.-B.: An insight into extreme learning machines: Random neurons, random features and kernels. *Cognitive Computation 6*, 3 (2014), 376–390. 3

[HZS06] HUANG G.-B., ZHU Q.-Y., SIEW C.-K.: Extreme learning machine: theory and applications. *Neurocomputing 70*, 1 (2006), 489–501. 2, 3, 4

[KFR03] KAZHDAN M., FUNKHOUSER T., RUSINKIEWICZ S.: Rotation invariant spherical harmonic representation of 3d shape descriptors. In *SGP* (2003), pp. 156–164. 8, 9

[KHS10] KALOGERAKIS E., HERTZMANN A., SINGH K.: Learning 3d mesh segmentation and labeling. *ACM Transactions on Graphics (TOG) 29*, 4 (2010), 102. 6, 9

[KSH12a] KRIZHEVSKY A., SUTSKEVER I., HINTON G. E.: Imagenet classification with deep convolutional neural networks. In *NIPS* (2012). 2, 4

[KSH12b] KRIZHEVSKY A., SUTSKEVER I., HINTON G. E.: ImageNet classification with deep convolutional neural networks. In *NIPS* (2012), pp. 1097–1105. 3

[LBBH98] LECUN Y., BOTTOU L., BENGIO Y., HAFFNER P.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE 86*, 11 (1998), 2278–2324. 2, 3, 8, 9

[LSD15] LONG J., SHELHAMER E., DARRELL T.: Fully convolutional networks for semantic segmentation. In *CVPR* (2015). 2, 3, 6, 9

[RHW88] RUMELHART D. E., HINTON G. E., WILLIAMS R. J.: Learning representations by back-propagating errors. *Cognitive modeling 5* (1988). 3

[SHKF12] SILBERMAN N., HOIEM D., KOHLI P., FERGUS R.: Indoor segmentation and support inference from RGBD images. In *ECCV* (2012), pp. 746–760. 10

[SMKLM15] SU H., MAJI S., KALOGERAKIS E., LEARNED-MILLER E.: Multi-view convolutional neural networks for 3d shape recognition. ArXiv e-prints 1505.00880v1, 2015. 2

[SWT14] SUN Y., WANG X., TANG X.: Deep learning face representation from predicting 10,000 classes. In *CVPR* (2014), pp. 1891–1898. 2

[THW*14] TEVS A., HUANG Q., WAND M., SEIDEL H.-P., GUIBAS L.: Relating shapes via geometric symmetries and regularities. *ACM Transactions on Graphics 33*, 4 (2014). 1

[WDC*05] WELCHMAN A. E., DEUBELIUS A., CONRAD V., BÜLTHOFF H. H., KOURTZI Z.: 3d shape perception from combined depth cues in human visual cortex. *Nature Neuroscience 8*, 6 (2005), 820–827. 2

[WGW*13] WANG Y., GONG M., WANG T., NAD HAO ZHANG D. C.-O., CHEN B.: Projective analysis for 3d shape segmentation. *ACM Trans. on Graphics 32*, 6 (2013), 192: 1–11. 2, 7

[WSK*15] WU Z., SONG S., KHOSLA A., TANG X., XIAO J.: 3d shapenets: A deep representation for volumetric shapes. In *CVPR* (2015). 2, 8, 9

[WXH*14] WEI Y., XIA W., HUANG J., NI B., DONG J., ZHAO Y., YAN S.: CNN: single-label to multi-label. *CoRR abs/1406.5726* (2014). 3

[XXLX14] XIE Z., XU K., LIU L., XIONG Y.: 3d shape segmentation and labeling via extreme learning machine. *Computer Graphics Forum 33*, 5 (2014), 85–95. 7

[XXS*15] XU K., XIE Z., SHAN W., LIU L., XIONG Y., HUANG H.: Project page. `www.kevinkaixu.net\project\mvd-elm.html`, 2015. 2

[ZF13] ZEILER M. D., FERGUS R.: Visualizing and understanding convolutional neural networks. *arXiv preprint arXiv:1311.2901* (2013). 6

[ZWB*14] ZHU Z., WANG X., BAI S., YAO C., BAI X.: Deep learning representation using autoencoder for 3d shape retrieval. *CoRR abs/1409.7164* (2014). 2