Deep Aggregation of Local 3D Geometric Features for 3D Model Retrieval

Takahiko Furuya[1]
takahikof AT yamanashi.ac.jp
Ryutarou Ohbuchi[1]
ohbuchi AT yamanashi.ac.jp

[1] Integrated Graduate School of Medicine, Engineering, and Agricultural Sciences, University of Yamanashi, Kofu, Japan

Abstract

Aggregation of local features is a well-studied approach for image as well as 3D model retrieval (3DMR). A carefully designed local 3D geometric feature is able to describe detailed local geometry of 3D model, often with invariance to geometric transformations that include 3D rotation of local 3D regions. For efficient 3DMR, these local features are aggregated into a feature per 3D model. A recent alternative, end-to-end 3D Deep Convolutional Neural Network (3D-DCNN) [7][33], has achieved accuracy superior to the abovementioned aggregation-of-local-features approach. However, current 3D-DCNN based methods have weaknesses; they lack invariance against 3D rotation, and they often miss detailed geometrical features due to their quantization of shapes into coarse voxels in applying 3D-DCNN.

In this paper, we propose a novel deep neural network for 3DMR called *Deep Local feature Aggregation Network* (*DLAN*) that combines extraction of rotation-invariant 3D local features and their aggregation in a single deep architecture. The DLAN describes local 3D regions of a 3D model by using a set of 3D geometric features invariant to local rotation. The DLAN then aggregates the set of features into a (global) rotation-invariant and compact feature per 3D model. Experimental evaluation shows that the DLAN outperforms the existing deep learning-based 3DMR algorithms.

1 Introduction

Aggregation of Local Features (*ALF*) has been a popular approach for effective and efficient comparison of 2D images or 3D models. For 3D Model Retrieval (3DMR), local 3D regions are sampled from a 3D model and each region is described by using a handcrafted local feature. The feature is typically designed to be invariant against geometric transformations, e.g., 3D rotation of the local region. The set of local features is aggregated per 3D model by a combination of feature encoding and pooling, e.g., by Bag-of-Features [14] or Fisher Vector coding [12]. Dimension reduction of the aggregated features may further improves efficiency, and sometimes efficacy, of comparison among 3D model features.

Recently, methods based on end-to-end deep learning ([7][16][25][33]) have achieved accuracies higher than the ALF-based methods for the task of classification and retrieval of 3D models. A deep learning based method employs a Deep Convolutional Neural Network (DCNN) which applies multi-layer convolution either on a multi-view set of rendered 2D images ([16][25]) or on a 3D voxel representation of a 3D shape ([7][33]). The DCNN can extract high-level features useful for accurate comparison among 3D shapes. [16] and [25]

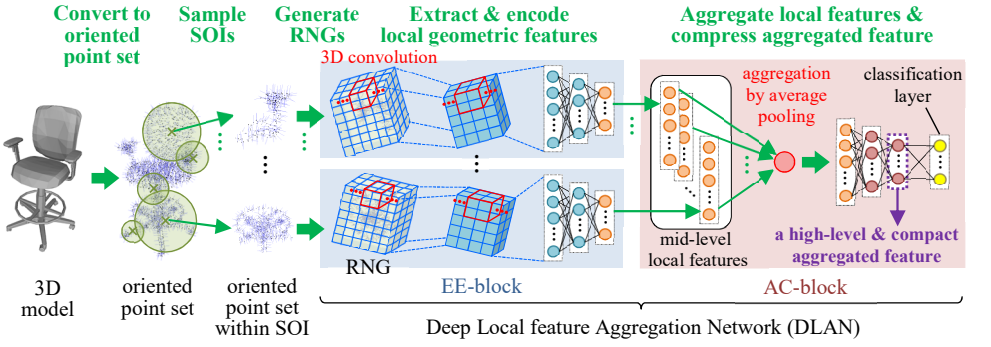


Figure 1: First half of DLAN extracts rotation-invariant and mid-level local features by applying 3D-DCNN on Rotation Normalized Grids (RNGs). The set of mid-level local features is aggregated into a feature per 3D model by average-pooling, followed by deep learning-based dimension reduction for a salient and compact feature per 3D model.

extract visual features from multi-view rendered images of 3D models by using a 2D-DCNN [3]. These methods ([16][25]) yield high retrieval accuracy by transferring the 2D-DCNN trained with numerous 2D natural images to the domain of multi-view images of 3D models. On the other hand, [7] and [33] adopt 3D-DCNN which takes a 3D voxel data as input and applies 3D convolution on the voxel data to extract high-level geometric features. A 3D-DCNN has a potential to do better than a 2D-DCNN in analyzing 3D shapes, e.g., for mesh segmentation, partial shape matching, etc., since the 3D-DCNN is able to directly capture 3D geometry of 3D model. However, retrieval accuracies of existing 3D-DCNN based 3DMR methods (i.e., [7][33]) trail those of the 2D-DCNN based methods (i.e., [16][25]).

We suspect that accuracies of the existing end-to-end 3D-DCNNs suffer for three reasons. Firstly, the previous methods lack invariance to global 3D rotation. As they do not normalize 3D rotation of a 3D model before converting it into its voxel representation, a pair of identical 3D model in different orientation won't match. Also, a pair of 3D models, range-scanned from two different viewpoints of a real-world object, would not match. Secondly, previous 3D-DCNN based methods fail to capture detailed 3D geometry of 3D models due to their coarse voxel quantization. For example, [33] and [7] use coarse grids of 30^3 and 32^3, respectively, to reduce temporal and spatial cost of 3D convolution. However, significant geometric details would be lost by converting 3D models into voxel models having such coarse grids. Thirdly, training DCNN for 3D models is more difficult than for 2D images since the number of labeled 3D models is much smaller than that of labeled 2D images.

A well-designed ALF based method often avoids the first two weaknesses listed above. A handcrafted local geometric feature can be made invariant to 3D rotation, e.g., via rotation normalization. Given a complexity, a local feature describes local geometric details better than a global feature. We thus think that an ALF-like approach still has potential. A local, low-level, rotation-invariant handcrafted feature processed by a DCNN could yield a descriptive local feature. Aggregation of a set of these features followed by another deep neural network could result in an expressive feature per 3D model for 3DMR.

In this paper, we propose a novel 3D-DCNN based 3DMR algorithm that is modeled after an ALF pipeline. The algorithm is called *Deep Local feature Aggregation Network* (*DLAN*). Given a set of local regions of a point set representation of 3D model, the first half of the DLAN (EE-block in Figure 1) extracts a "mid-level" 3D local feature per region. We employ point set representation as it is richer in feature than a coarsely quantized voxel

representation. Each local region is first normalized for 3D rotation. Then, and a low-level local feature is extracted for each cell of a 3D grid called *Rotation Normalized Grids* (*RNG*) imposed on the orientation normalized region. A 3D array of low level features of each RNG are then put through multi-layer 3D convolution to yield a mid-level local feature per region. The second half (AC-block in Figure 1) performs pooling of local features and dimension reduction of the pooled feature to generate a compact and salient feature per 3D model.

As noted above, supervised training of a 3D-DCNN is difficult as available labeled sets of 3D models are small. To alleviate this issue, we devise a two-step training approach. First, with a large number of labeled RNGs, the first half of the DLAN is trained to extract mid-level local feature. Then, the DLAN is trained end-to-end by using a relatively small number of labeled 3D models to generate high-level and compact aggregated feature per 3D model.

Experiments demonstrate that the DLAN achieves higher retrieval accuracy and better invariance against 3D rotation than the state-of-the-art DCNN-based 3DMR algorithms.

Contributions of this paper can be summarized as follows;

- Proposition of the DLAN which embodies the conventional aggregation-of-local-features approach in a deep architecture for accurate 3D model retrieval.
- Experimental evaluation of the DLAN and several state-of-the-art DCNN-based 3DMR algorithms using recently created 3D model datasets.

2 Related work

2.1 Approaches to 3D model retrieval

Aggregating local features: The ALF has been a popular approach to 3DMR [4][34]. 3DMR algorithms based on ALF can be classified into 2D view-based and 3D geometry-based approaches. The 2D view-based approach (e.g., [23][28]) uses multi-view rendered (2D) images of a 3D model for comparison. Most often, a set of local visual features (e.g., SIFT [6]), which is invariant against 2D in-plane rotation, are extracted from each image. The set of local visual features are aggregated by using feature encoding algorithms, e.g., Bag-of-Features [14], Locality-constrained Linear (LL) coding [20], Vector of Locally Aggregated Descriptors (VLAD) [15], Fisher Vector (FV) coding [12], or Super Vector (SV) coding [29]. As the aggregated feature typically has high dimensions, e.g. 300k dim. [28], it is often compressed by using a dimension reduction algorithm such as PCA for efficiency.

The 3D geometry-based approach (e.g., [27][31][32]) extracts a set of local geometric features from the 3D representation (e.g., oriented point set or voxel) of a 3D model. A carefully designed local 3D geometric feature, e.g., HKS [19], SHOT [13], and POD [27] describes detailed geometry of the 3D model with invariance against 3D rotation. The set of local geometric features is aggregated by using the feature encoding algorithms described above for efficient comparison among 3D models. Local 3D geometric features are also useful for such other purposes as mesh segmentation [9] and partial shape matching [21].

Deep learning: Recent studies (e.g., [7][16][18][25][33]) have found that end-to-end DCNN often yields better results than a conventional ALF based algorithm. [16] and [25] employ a 2D-DCNN [3] pre-trained with numerous 2D images. [7] and [33] adopt 3D-DCNN which applies successive 3D convolutions on the input voxel data to extract high-level 3D model features. However, as we discussed in the previous section, the 3D-DCNN based methods have lower retrieval accuracy than the 2D-DCNN based methods. [26] tried to learn accurate local geometric feature by using unsupervised deep learning. However, the deep framework

of [26] is not optimal in terms of feature aggregation. Also, [26] did not employ 3D convolution, which we consider important in learning salient geometric features.

2.2 Combining ALF and DCNN for 2D image retrieval

For the task of 2D image retrieval, aggregating local visual features produced by a 2D-DCNN has been shown to be effective [1][30]. [30] aggregates by VLAD local visual features extracted from mid layers of the DCNN. [1] aggregates local DCNN features by using summation. In [11], handcrafted local visual features are aggregated via FV coding, and the aggregated feature is put through a Multi-Layer Perceptron (MLP) [10] to produce discriminative features. These algorithms ([1][11][30]), however, use existing feature encoding algorithms that are not designed or optimized for DCNN feature aggregation. On the other hand, our DLAN is trained so that it generates optimized aggregated features by performing encoding and aggregation of local features in a single deep architecture.

3 Proposed algorithm

3.1 Overview of Deep Local feature Aggregation Network (DLAN)

Figure 1 shows an overview of the DLAN. The DLAN resembles a conventional ALF pipeline for 3DMR, but realizes it by using deep neural networks.

DLAN is a deep architecture consisting of 10 layers, if input RNG and output classification layers are included. DLAN can be divided into two blocks; (1) *EE-block* performs *Extraction* and *Encoding* of rotation invariant local 3D features, and (2) *AC-block* performs *Aggregation* of the local 3D features followed by *Compression* of the aggregated feature. The EE-block generates mid-level local features by using 3D convolution layers and subsequent fully-connected layers. The AC-block first aggregates the set of mid-level local features computed by the EE-block into a single feature per 3D model by a pooling layer. Then the aggregated feature is dimension reduced, or compressed, by using fully-connected layers to generate a compact and discriminative feature describing a 3D model.

DLAN achieves rotation invariance of its local feature by normalizing, by using PCA, 3D orientation of the spherical region from which a feature is extracted. After the rotation normalization, the spherical region is quantized into a 3D grid called RNG. Geometry in each cell of an RNG is described by using Point and Orientation Distribution (POD) feature [27]. Multiple POD features of the RNG is fed into a 3D-DCNN to produce a discriminative, local, rotation invariant, mid-level feature.

In general, an effective end-to-end training of a deep network requires a very large number (e.g., 1M or more) of labeled data. However, at this time, largest collection of labeled 3D models available for experiments contains only 10k models. We thus employ a two-step approach for training DLAN. We first pre-train the EE-block only by using a large number of *labeled RNGs* densely sampled from labeled 3D models. Via supervised training using RNGs having class labels, the EE-block is trained so that it yields "mid-level" local 3D features. We then train the entire DLAN, which includes both EE-block and AC-block, end-to-end by using relatively small number of labeled 3D models.

Given a query 3D model, a set of RNGs extracted from the query 3D model is fed into the trained DLAN to generate a per-3D model aggregated feature for the query. The per-3D model feature of the query is compared against those of database 3D models to generate ranked retrieval results.

3.2 Architecture of DLAN

Input layer: Input to DLAN is a set of RNGs densely sampled on a 3D model. RNG is computed as follows. If a 3D model is defined as polygonal mesh, it is first converted into an oriented point set by using the algorithm by Osada [24]. We uniformly sample 3k oriented points on surfaces of the 3D model. Then, multiple Spheres-Of-Interest (SOIs) are densely sampled in the oriented point set. In the experiments, we sample 100 SOIs per 3D model. To capture diverse local geometric features of the 3D model, position and scale for each SOI is randomly determined. Also, to achieve robustness against 3D rotation of local shapes, orientation for each SOI is normalized by using PCA on the coordinates of oriented points within the SOI. The axes of the rotated coordinate system correspond to three principal axes for the SOI. We then disambiguate the directions, or signs, for these principal axes. The sign s_1 for the first principal axis $\mathbf{e}_1$ is computed by the following equation;

$$s_1 = sign\left(\sum_{i=1}^{n_p}(\mathbf{p}_i - \mathbf{p})\cdot\mathbf{e}_1\right) \quad (1)$$

where n_p is the number of oriented points within the SOI, $\mathbf{p}_i$ is the position of i-th oriented point, and $\mathbf{p}$ is the centre of the SOI. The sign s_3 for the third principal axis $\mathbf{e}_3$ is computed in the same manner as s_1. The second principal axis $\mathbf{e}_2$ is determined as cross product of $s_1\mathbf{e}_1$ and $s_3\mathbf{e}_3$.

The bounding box of the rotated SOI is then divided into $N_1\times N_2\times N_3$ cells by using a 3D regular grid. N_1, N_2, and N_3 indicate the number of cells along the first, second, and third principal axis respectively. We place more cells along the axis having larger variance of oriented point distribution. We use $N_1\times N_2\times N_3$= 7×6×3 for the DLAN. Each cell in the 3D grid is described by using a POD feature. Specifically, within each cell, three statistics of oriented points, i.e., density of points, mean of points, and covariance of point orientation vectors, are computed. These statistics form a 10-dimensional (or 10-channel) POD feature for the cell. Previous 3D DCNN-based methods ([7][33]) use only 1 channel of information, i.e., voxel intensity. In comparison, RNG with POD feature contains richer information about 3D shape. The POD features are then normalized by using ZCA-whitening [2] so that the features have zero means and all the pair of channels are uncorrelated.

EE-block: The input RNGs are fed into the EE-block to extract mid-level local features from the RNGs. The EE-block has two 3D convolution layers followed by three fully-connected layers. Each RNG having size 7×6×3 is convolved twice by using 3D convolutional filters with size 2×2×2 to produce a convolved RNG, whose size is 5×4×1 after convolution. The first convolution layer has 256 filters while the second convolution layer has 512 filters. The convolved RNG is then vectorized for input to the first fully-connected layer. Each of the three fully-connected layers has 3,072, 2,048, and 512 neurons respectively. To non-linearly transform local features at each layer, we use Rectified Linear Units (ReLU) [3] both in the 3D convolution layers and the fully-connected layers.

AC-block: The set of encoded local features, which is the output from the EE-block, is aggregated to a compact and discriminative feature per 3D model in the AC-block. The AC-block consists of one aggregation layer and subsequent two fully-connected layers. The aggregation layer pools the set of encoded local features to a single vector per 3D model. As a pooling function, average pooling or max pooling are generally used. We employ average pooling for DLAN since it yields better retrieval accuracy than max pooling as we will show in the experiments. The number of dimensions for the aggregated feature, 512, is the same as the number of neurons in the fully-connected layer in front of the aggregation layer. The aggregation layer differs from the spatial pooling layer [3] used in the 2D-DCNN in that the

aggregation layer pools all the local features of the 3D model ignoring their positions sampled at the 3D model.

The aggregated feature is further transformed by using the two fully-connected layers that follow the aggregation layer to produce a more compact and discriminative 3D model feature. These two fully-connected layers contain 256 and 128 neurons with ReLU, respectively. We use unit activations of the last fully-connected layer as a feature for the 3D model. Therefore, each 3D model is represented as a 128-dimensional vector. At the retrieval stage, 3D model features are compared by using Cosine similarity to generate ranking results.

Output layer: The DLAN has a classification layer with softmax function at the output end of the network. The classification layer has the same number of neurons with the number of semantic categories used during training of the network. Note that we do not use unit activation of the output layer as a feature for the 3D model since the DLAN usually outputs a vector in which only one element is 1 while the other elements are 0. We think such vectors are inappropriate for computing gradual similarities among 3D models.

3.3 Training of DLAN

To effectively train the DLAN, we employ two-step training of the network. That is, EE-block of the DLAN is first pre-trained by using a large set of labeled RNGs. Then, the entire network is trained by using a relatively small number of labeled 3D models.

Pre-training EE-block: To learn expressive local feature representation better suited for accurate 3DMR, EE-block should be trained by using a large number of labeled data. To do so, we use a large set of RNGs sampled from labeled 3D models contained in a training dataset. Specifically, we sample RNGs having random position and scale from the labeled 3D models by using the method described in Section 3.2. The label for each training RNG is identical to the label for the 3D model from which the training RNG is sampled. We sample 300k labeled RNGs from the training dataset.

By using the set of labeled RNGs, the EE-block is trained so that it could predict labels for RNGs. For the supervised training, we append the classification layer with softmax function at the output end of the EE-block. The classification layer has the same number of neurons as the number of semantic categories for training 3D models. We use cross-entropy loss function for pre-training the EE-block. Parameters, i.e., connection weight among neurons, are randomly initialized by using a standard normal distribution whose mean is 0 and deviation is 1. The parameters are tuned by using Stochastic Gradient Descent (SGD) with mini-batch. Each mini-batch contains 30 labeled RNGs. To adaptively assign learning rate for each parameter, we employ AdaGrad algorithm [17] with initial learning rate = 0.1 for training. To avoid over-fitting, we also perform 50% dropout [3] for all the layers except for input and output layers. We iterate the training for 200 epochs.

Training whole network: After supervised pre-training of the EE-block, we train the entire network including both the EE-block and the AC-block by using labeled 3D models. The parameters in EE-block inherits as its initial values the result of pre-training. The initial values for the parameters in AC-block are sampled from the standard normal distribution. Similar to pre-training the EE-block, training of the entire DLAN also minimizes cross-entropy loss by using SGD with mini-batch. Each mini-batch contains 30 labeled 3D models. That is, each mini-batch comprises of 30×100=3k RNGs since we sample 100 RNGs per 3D model. We use AdaGrad algorithm with initial learning rate = 0.1 and 50% dropout during the training. We iterate the training for 200 epochs.

4 Experiments and results

4.1 Experimental setup

We evaluate retrieval accuracy of the DLAN by using three benchmark datasets recently created by Wu et *al.* [33] or Savva et *al.* [22] for retrieval or classification of 3D models. The first and second datasets [33] are called ModelNet40 and ModelNet10. Figure 2 shows examples of 3D models contained in the datasets. The ModelNet40 is divided into the training set and the test set. The training set of the ModelNet40 contains 9,843 3D models classified into 40 semantic categories such as airplane, plant, sofa, etc. while the test set has 2,468 3D models belong to the same categories as the training set. The ModelNet10 contains 10 semantic categories of indoor objects, e.g., chair, table, bed, etc. The training set and the test set of ModelNet10 have 3,991 and 908 3D models respectively.

The third dataset is SHREC'16 Large-Scale 3D Shape Retrieval from ShapeNet Core55 (SH16LS) [22]. The SH16LS contains 51,300 3D models over 55 categories similar to those in ModelNet40. The SH16LS consists of three subsets, i.e., training set, validation set, and test set. Each subset contains 35,765, 5,159, and 10,266 3D models, respectively. SH16LS has two evaluation protocols, which we call "original orientation (OO)" and "perturbed orientation (PO)". For "OO", 3 DOF orientations of the 3D models are consistently aligned. And for "PO", orientations of the 3D models are randomized. These protocols are favorable for evaluating our DLAN in terms of invariance against 3D rotation of 3D models.

For all the three datasets, labeled 3D models in the training set are used for training the DLAN and 3D models in the test set are used for evaluation of retrieval accuracy. We use Mean Average Precision (MAP) [%] and Recall-Precision curve as accuracy indices.

For evaluation on ModelNet40 and ModelNet10, we compare the DLAN against nine existing algorithms. Three of the nine algorithms, i.e., LFD [8], SV-DSIFT [28], and DM-POD [27], adopt unsupervised, handcrafted feature. SV-DSIFT and DM-POD are state-of-the-art ALF-based algorithms. Remaining six algorithms use supervised feature derived from a deep network. MVCNN [16], DeepPano [5], and GIFT [25] describe 2D views of 3D model by using the 2D-DCNN called AlexNet [3]. On the other hand, 3D ShapeNets [33] extract a global 3D geometric feature from a voxel representation of a 3D model by using a 3D-DCNN. SV-DSIFT+MLP and DM-POD+MLP are similar to [11], which combines the existing handcrafted feature with classical MLP. The MLP takes the handcrafted feature as input and extracts a discriminative feature for the 3D model from the last fully-connected layer. The MLP consists of three layers, each of which has 2,048, 1,024, and 128 neurons.

We use our own implementation of DLAN for the experiments, which utilizes a GPU via *CUDA* library. The code of DLAN is run on a PC having an *Intel Core i7-5930K* CPU, a *GeForce GTX Titan X* GPU, and 32GB DRAM. Pre-training of EE-block of DLAN took 2 to 3 days while training of entire DLAN took about 10 hours.

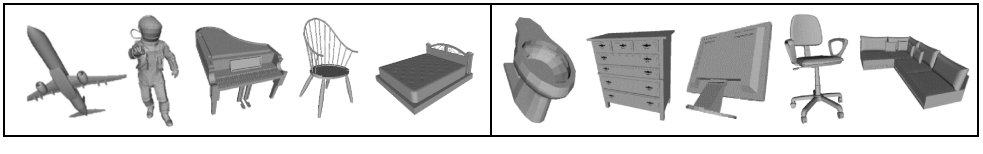


Figure 2: Examples of 3D models contained in ModelNet40 (left) and ModelNet10 (right).

4.2 Experimental results

Effectiveness of two-step DLAN training: Table 1 shows the effect of two-step training of DLAN. In Table 1, "Yes" or "No" indicate that supervised training is performed or not. For example, "Case 1" performs no training, i.e., DLAN with completely random parameters is used to generate 3D model features. "Case 2" means the whole DLAN is trained from scratch without pre-training of EE-block. "AG", "FC1", and "FC2" indicate MAP for the features extracted from the aggregation layer (512 dim.), the first fully-connected layer (256 dim.), and the second fully-connected layer (128 dim.) in AC-block, respectively.

Comparison between the results for "Case 2" and "Case 4" verifies the effectiveness of supervised pre-training of the EE-block. "Case 4" significantly outperforms "Case 2" with large margin, i.e., about 10% in MAP, both for ModelNet40 and ModelNet10. By using numerous (i.e., 300k) labeled RNGs, EE-block is trained so that it can yield mid-level local features suited for 3DMR. Pre-training EE-block prior to training entire DLAN seem to find better network parameters that leads to more accurate aggregated features for 3D models.

Figure 3 plots cross-entropy loss throughout the DLAN training on ModelNet40. For "Case 2", the loss slowly decreases and the training does not seem to converge even after 200 training epochs. On the other hand, for "Case 4", the loss falls more quickly than "Case 2". At 200 epochs, the loss for "Case 4" is lower than that for "Case 2", which indicates that supervised pre-training of the EE-block helps the DLAN find better solution. We also noticed that "Case 4" slightly over-fits to the training set as loss for the test set gradually increases after 100 epochs. Over-fitting could be alleviated by introducing regularization techniques such as weight decay or early-stopping, which are not employed in our DLAN.

	Pre-training EE-block	Training whole DLAN	MAP [%] for ModelNet40			MAP [%] for ModelNet10		
			AG	FC1	FC2	AG	FC1	FC2
Case 1	No	No	33.8	31.6	28.1	45.4	43.8	40.2
Case 2	No	Yes	73.2	74.5	74.2	81.0	81.9	81.5
Case 3	Yes	No	74.1	73.5	70.2	87.6	86.8	85.5
Case 4	**Yes**	**Yes**	**83.9**	**85.0**	**85.0**	**90.3**	**91.0**	**90.6**

Table 1: Effectiveness of two-step training of DLAN.

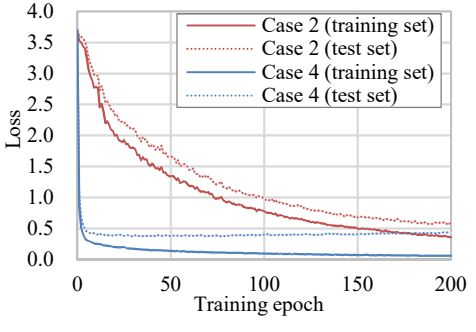


Figure 3: Cross-entropy loss during training of whole DLAN.

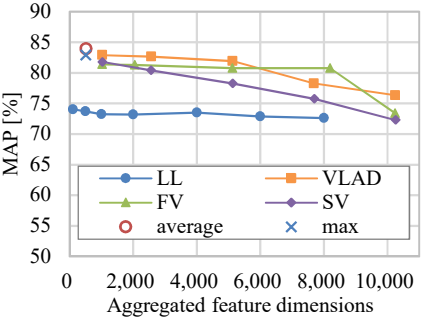


Figure 4: Comparison of feature aggregation algorithms.

Comparison of feature aggregation methods: In this section, we demonstrate that simple average pooling at the aggregation layer of the DLAN produces discriminative aggregated features. In Figure 4, "average" shows accuracy for DLAN trained with average pooling at the aggregation layer while "max" indicates accuracy for DLAN trained with max pooling.

We use 512-dimensional features extracted from the aggregation layer. We can observe that the average pooling (MAP=83.9%) slightly outperforms the max pooling (MAP=82.9%).

We also evaluate accuracies for aggregated features generated by using state-of-the-art unsupervised feature encoding algorithms. To do so, we extract unpooled local features at the last fully-connected layer of the EE-block and aggregate them by using either LL, VLAD, FV, or SV with different codebook size. Figure 4 plots accuracies against the number of dimensions for aggregated features. Unsupervised feature encoding yields inferior MAP scores to average pooling and max pooling. Since parameters of DLAN is optimized by using average (or max) pooling at the aggregation layer, these simple pooling functions produce more accurate aggregated feature than the existing feature encoding algorithms.

Effectiveness of 3D convolution and 3D rotation normalization: We show that both 3D convolution and 3D rotation normalization of local regions are essential for DLAN to achieve high accuracy. We used the ModelNet40 and 3D models in the dataset are rotated randomly to evaluate the effectiveness of rotation normalization. In Table 2, "Case A" shows accuracy of the proposed DLAN which employs both 3D convolution and 3D rotation normalization. "Case B" shows accuracy for DLAN trained without the first two 3D convolution layers of the EE-block. "Case C" indicates accuracy for DLAN trained without rotation normalization, i.e., local regions having their original orientation are fed into the first 3D convolution layer of DLAN. For "Case D", we employed a different approach to 3D rotation invariance by using data augmentation. That is, each local 3D region is randomly rotated and the DLAN is trained by using these local regions having diverse 3D orientations.

In Table 2, comparing "Case A" (MAP=85.0%) and "Case B" (MAP=81.8%) verifies efficacy of 3D convolution. Also, comparison among "Case A", "Case C" (MAP=51.2%) and "Case D" (MAP=66.8%) shows effectiveness of 3D rotation normalization.

	3D convolution	3D rotation of local region	MAP [%]
Case A	Yes	Normalized	85.0
Case B	No	Normalized	81.8
Case C	Yes	Unchanged	51.2
Case D	Yes	Randomized	66.8

Table 2: Effectiveness of 3D convolution and 3D rotation normalization.

algorithms	feat. [s]	sim. [s]	total [s]
SV-DSIFT [28]	1.01	0.2125	1.22
DM-POD [27]	0.10	0.1654	0.27
MVCNN [16]	1.63	0.0016	1.63
3D ShapeNets [33]	0.11	0.0004	0.11
DLAN (proposed)	0.22	0.0001	0.22

Table 3: Computation time per query on ModelNet40 test set.

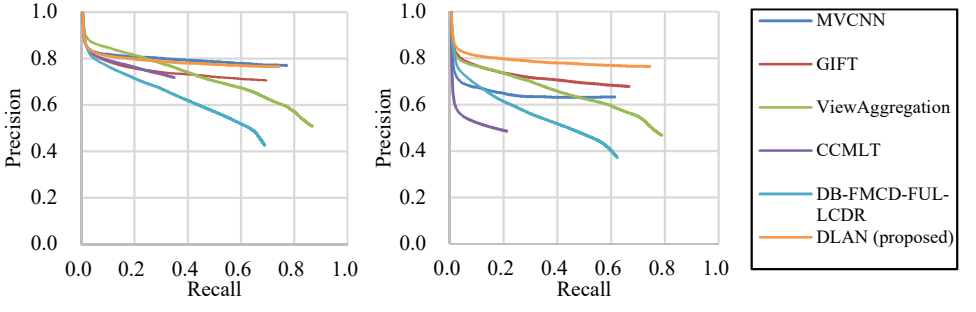
Comparison with other 3DMR algorithms: Table 3 compares computation times for querying the database. In Table 3, "feat." shows time for extracting a feature from a query 3D model and "sim." is time for computing similarities among the feature of the query and features of the database 3D models (2,468 models for ModelNet40). The DLAN, although not the fastest, quickly processes a query in about 0.2 seconds.

Table 4 compares retrieval accuracy of DLAN with those of existing algorithms. For the ModelNet40, the proposed DLAN yields the highest MAP score among the ten algorithms, including both 2D view-based and 3D geometric-based algorithms. MAP for the DLAN (85.0%) is 3% higher than MAP for the GIFT (81.9%), which is, to our knowledge, the best performing algorithm for ModelNet40. On the other hand, for the ModelNet10, SV-DISFT+MLP, GIFT, and DLAN show almost the same retrieval accuracies (MAP=90~91%). Note that, however, the DLAN performs the best among 3D geometric-based algorithms we have compared in Table 4. Compared to 2D view-based algorithms, 3D geometric-based algorithms have larger potential for 3D shape analysis, e.g., for segmentation, partial shape matching, or shape completion.

algorithms	feature type	supervised training	ModelNet40	ModelNet10
LFD [8]	2D view	No	40.9	49.8
SV-DSIFT [28]	2D view	No	58.1	68.0
SV-DSIFT+MLP	2D view	Yes	**81.0**	**90.2**
MVCNN [16]	2D view	Yes	79.5	
DeepPano [5]	2D view	Yes	76.8	84.2
GIFT [25]	2D view	Yes	**81.9**	**91.1**
DM-POD [27]	3D geometric	No	51.6	61.2
DM-POD+MLP	3D geometric	Yes	74.8	85.4
3D ShapeNets [33]	3D geometric	Yes	49.2	68.3
DLAN (proposed)	3D geometric	Yes	**85.0**	**90.6**

Table 4: Comparison of retrieval accuracies (MAP [%]).

We also evaluated 3D rotation invariance of DLAN by using SH16LS dataset. Figure 5 compares accuracy of DLAN against those of five 2D-DCNN based algorithms. The "OO" evaluation protocol (Figure 5a) uses original SH16LS database as-is, in which most of their models are rotationally aligned. In this case, DLAN performs as well as MVCNN. On the other hand, for the "PO" protocol (Figure 5b), where each 3D model is randomly rotated, DLAN performs better than the five competitors. We can also observe that DLAN has invariance against 3D rotation of 3D models since DLAN yields almost the same Recall-Precision curves in "OO" and "PO" evaluation protocols.



(a) Original orientation (OO). (b) Perturbed orientation (PO).

Figure 5: Recall-Precision curves on SH16LS test set.

5 Conclusion and future work

In this paper, we proposed a novel deep neural network architecture called *Deep Local feature Aggregation Network* (*DLAN*) for 3D model retrieval. The DLAN first extracts a set of local 3D features at multiple positions and scales from a 3D model. To obtain rotation invariant and mid-level, as opposed to low-level, local features, DLAN applies 3D-DCNN on a rotation normalized 3D region. These mid-level local features are globally aggregated and then refined into a discriminative and compact feature per 3D model. Experimental evaluation demonstrated that the DLAN yields higher retrieval accuracy than the existing deep learning-based 3D model retrieval algorithms.

As a future work, we will evaluate the effectiveness of the local 3D feature derived from the DLAN on such applications as 3D mesh segmentation and partial 3D shape matching.

Acknowledgement: This research is supported by *JSPS Grant-in-Aid for Young Scientists (B)* #16K16055 and *JSPS Grants-in-Aid for Scientific Research (C)* #26330133.

References

[1] A. Babenko and V. Lempitsky. Aggregating Local Deep Features for Image Retrieval. *Proc. ICCV 2015*, 1269–1277, 2015.

[2] A. Bell, and T.J. Sejnowski. Edges are the `Independent Components' of Natural Scenes. *Proc. NIPS 1996*, 1996.

[3] A. Krizhevsky, I. Sutskever, and G.E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Proc. NIPS 2012*, 1097–1105, 2012.

[4] B. Li et al. A comparison of 3D shape retrieval methods based on a large-scale benchmark supporting multimodal queries. *Computer Vision and Image Understanding*, **131**, 1–27, 2015.

[5] B. Shi, S. Bai, Z. Zhou, and X. Bai. DeepPano: Deep Panoramic Representation for 3-D Shape Recognition. *IEEE Signal Processing Letters*, **22**(12), 2339–2343, 2015.

[6] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints, *IJCV*, **60**(2), 91–110, 2004.

[7] D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. *Proc. IROS 2015*, 922–928, 2015.

[8] D.Y. Chen, X.P. Tian, Y.T. Shen, and M. Ouhyoung. On Visual Similarity Based 3D Model Retrieval. *Computer Graphics Forum*, **22**(3), 223–232, 2003.

[9] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3D Mesh Segmentation and Labeling. *ACM TOG*, **29**(4), Article No.102, 2010

[10] F. Rosenblatt. Principles of neurodynamics: perceptrons and the theory of brain mechanisms. *DTIC Document*, 1961.

[11] F. Perronnin and D. Larlus. Fisher vectors meet Neural Networks: A hybrid classification architecture. *Proc. CVPR 2015*, 3743–3752, 2015.

[12] F. Perronnin, J. Sánchez, T. Mensink, Improving the fisher kernel for large-scale image classification, *Proc. ECCV 2010*, Part IV, 143–156, 2010.

[13] F. Tombari, S. Salti, and L. D. Stefano. Unique signatures of histograms for local surface description, *Proc. ECCV 2010*, 356–369, 2010.

[14] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual Categorization with Bags of Keypoints. *Proc. ECCV 2004 workshop on Statistical Learning in Computer Vision*, 59–74, 2004.

[15] H. Jégou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation, *Proc. CVPR 2010*, 3304–3311, 2010.

[16] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-View Convolutional Neural Networks for 3D Shape Recognition. *Proc. ICCV 2015*, 945–953, 2015.

[17] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, **12**, 2121–2159, 2011.

[18] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on Riemannian manifolds. *Proc. International Workshop on 3D Representation and Recognition (3dRR) 2015*, 2015.

[19] J. Sun, M. Ovsjanikov, and L. Guibas. A Concise and Provably Informative Multi-Scale Signature-Based on Heat Diffusion. *Computer Graphics Forum*, **28**(5), 1383–1392, 2009.

[20] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained Linear Coding for Image Classification. *Proc. CVPR 2010*, 3360–3367, 2010.

[21] L. Shapira, S. Shalom, A. Shamir, D. Cohen-Or, and H. Zhang. Contextual part analogies in 3D objects. *IJCV*, **89**(2), 309–326, 2010.

[22] M. Savva et al. SHREC'16 Track: Large-Scale 3D Shape Retrieval from ShapeNet Core55. *Proc. Eurographics Workshop on 3D Object Retrieval* (*3DOR*) *2016*, 2016.

[23] R. Ohbuchi, K. Osada, T. Furuya, and T. Banno. Salient local visual features for shape-based 3D model retrieval. *Proc. SMI 2008*, 93–102, 2008.

[24] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape Distributions. *ACM Transactions on Graphics*, **21**(4), 807–832, 2002.

[25] S. Bai, X. Bai, Z. Zhou, Z. Zhang, and L. J. Latecki. GIFT: A Real-time and Scalable 3D Shape Search Engine, *Proc. CVPR 2016*, 5023–5032, 2016.

[26] S. Bu, P. Han, Z. Liu, J. Han, and H. Lin. Local deep feature learning framework for 3D shape. *Computers and Graphics*, **46**, 117–129, 2015.

[27] T. Furuya, and R. Ohbuchi. Diffusion-on-Manifold Aggregation of Local Features for Shape-based 3D Model Retrieval. *Proc. ICMR 2015*, 171–178, 2015.

[28] T. Furuya and R. Ohbuchi. Fusing Multiple Features for Shape-based 3D Model Retrieval. *Proc. BMVC 2014*, 2014.

[29] X. Zhou, K. Yu, T. Zhang, and T.S. Huang. Image Classification using Super-Vector Coding of Local Image Descriptors, *Proc. ECCV 2010*, 141–154, 2010.

[30] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-Scale Orderless Pooling of Deep Convolutional Activation Features. *Proc. ECCV 2014*, 392–407, 2014.

[31] Y. Liu, H. Zha, and H. Qin. Shape Topics: A Compact Representation and New Algorithms for 3D Partial Shape Retrieval. *Proc. CVPR 2006*, 2025–2032, 2006.

[32] Y. Ohkita, Y. Ohishi, T. Furuya, and R. Ohbuchi.Non-rigid 3D Model Retrieval Using Set of Local Statistical Features, *Proc. ICME 2012 Workshop on Hot Topics in 3D Multimedia*, 593–598, 2012.

[33] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang and J. Xiao. 3D ShapeNets: A Deep Representation for Volumetric Shape Modeling. *Proc. CVPR 2015*, 1912–1920, 2015.

[34] Z. Lian et al. SHREC'15 Track: Non-rigid 3D Shape Retrieval. *Proc. EG 3DOR 2015*, 107–120, 2015.