Eurographics Workshop on 3D Object Retrieval (2017)
I. Pratikakis, F. Dupont, and M. Ovsjanikov (Editors)

SHREC'17 Track
Large-Scale 3D Shape Retrieval from ShapeNet Core55

Manolis Savva[1*], Fisher Yu[2*], Hao Su[3*], Asako Kanezaki[3], Takahiko Furuya[4], Ryutarou Ohbuchi[4], Zhichao Zhou[5], Rui Yu[5], Song Bai[5]
Xiang Bai[5], Masaki Aono[6], Atsushi Tatsuma[6], S. Thermos[7], A. Axenopoulos[7], G. Th. Papadopoulos[7], P. Daras[7]
Xiao Deng[8], Zhouhui Lian[8], Bo Li[9], Henry Johan[10] and Yijuan Lu[11], Sanjeev Mk [12]

[1]Princeton University, USA
[2]UC Berkeley, USA
[2]Stanford University, USA
[3]National Institute of Advanced Industrial Science and Technology (AIST), Japan
[4]University of Yamanashi, Japan
[5]School of Electronic Information and Communications, Huazhong University of Science and Technology, China
[6]Toyohashi University of Technology, Japan
[7]Information Technologies Institute, Centre for Research and Technology Hellas, Greece
[8]Institute of Computer Science and Technology, Peking University, Beijing, China
[9]School of Computing, University of Southern Mississippi, USA
[10]Visual Computing, Fraunhofer IDM@NTU, Singapore
[11]Department of Computer Science, Texas State University, San Marcos, USA
[12]Indian Institute of Technology, Bombay
*Track organizers (contact email: shrecshapenet@gmail.com)

Abstract
With the advent of commodity 3D capturing devices and better 3D modeling tools, 3D shape content is becoming increasingly prevalent. Therefore, the need for shape retrieval algorithms to handle large-scale shape repositories is more and more important. This track provides a benchmark to evaluate large-scale 3D shape retrieval based on the ShapeNet dataset. It is a continuation of the SHREC 2016 large-scale shape retrieval challenge with a goal of measuring progress with recent developments in deep learning methods for shape retrieval. We use ShapeNet Core55, which provides more than 50 thousands models over 55 common categories in total for training and evaluating several algorithms. Eight participating teams have submitted a variety of retrieval methods which were evaluated on several standard information retrieval performance metrics. The approaches vary in terms of the 3D representation, using multi-view projections, point sets, volumetric grids, or traditional 3D shape descriptors. Overall performance on the shape retrieval task has improved significantly compared to the iteration of this competition in SHREC 2016. We release all data, results, and evaluation code for the benefit of the community and to catalyze future research into large-scale 3D shape retrieval (website: https://www.shapenet.org/shrec17).
Categories and Subject Descriptors: H.3.3 [Computer Graphics]: Information Systems- Information Search and Retrieval

1. Introduction

The rapid improvements in 3D modeling tools and the recent rise of commodity depth sensors have led to a deluge of 3D content. The increasing availability of 3D data puts a focus on the development of scalable and efficient algorithms for search, retrieval and analysis. Improved methods for 3D shape retrieval have the potential to enable many applications in virtual and augmented reality, 3D printing, and inverse graphics for computer vision.

In this track of the yearly SHREC contest we evaluate the performance of deep learning methods for 3D shape retrieval. The recent development of large scale repositories of 3D shapes such as ShapeNet [CFG*15] made available much bigger datasets to develop and evaluate new algorithms. This year we repeat the 3D shape retrieval challenge organized for SHREC 2016 around the ShapeNet dataset. Though retrieval of relevant 3D models given a query model has been studied for more than a decade, deep learning approaches have only been introduced in the last couple of years. Our goal is to evaluate the performance of cutting edge deep-learning 3D shape retrieval methods. Furthermore, we would like

DOI: 10.2312/3dor.20171050

to see how much progress has been made since last year, with more mature methods on the same dataset. To that end, we again use the ShapeNet Core55 subset of ShapeNet which consists of more than 50 thousand models in 55 common object categories.

Eight participating teams have submitted a variety of retrieval methods. All methods except one are based on deep learning models. In contrast to last year where all teams used projection-based input data, this year there were methods using various types of 3D representation: multi-view projections, point sets, volumetric occupancy grids, and traditional 3D shape features. We evaluate the methods with several standard information retrieval performance metrics. RotationNet by Kanezaki et al. is projection-based and performs the best on consistently aligned 3D models. DLAN by Furuya et al. is point set–based and performs the best on models with random orientations. For both types of data, the best performers this year have significantly outperformed the champion methods from last year. However, the overall performance still leaves space for improvement. We hope to further catalyze research into 3D model retrieval by releasing both the data and evaluation code publicly for the benefit of the community.

2. Dataset

The ShapeNet Core55 competition dataset contains a total of 51162 3D models categorized into 55 WordNet [Mil95] synsets (lexical categories belonging to a taxonomy of concepts in the English language). Before using this data for the competition, the models were deduplicated. Furthermore, each model was assigned a sub-synset (sub-category) label which indicates a more refined class for the object (e.g., a model may have a sub-synset of "fighter jet" within the synset of "airplane". There are 204 sub-synset labels across the 55 synsets and these are used to establish a gradated relevance score (beyond just matching vs not matching synset label) in one of the evaluation metrics.

The original models in the evaluation dataset were taken from the ShapeNet corpus which was collected primarily from the Trimble 3D warehouse [Tri12]. To make training and evaluation of learning based methods possible, we established standard training, validation and test splits of the dataset. The proportions chosen were 70% (35764), 10% (5133) and 20% (10265) respectively out of the total 51162 models. The ShapeNet model data was converted to OBJ format with only geometric information, and the model dimensions were normalized to a unit length cube. In addition, since ShapeNet provides consistent upright and front orientation annotation for all models in the ShapeNetCore corpus, all model data was consistently aligned. We call this the "normal" dataset. To establish a more challenging version of the data without the assumption for pre-existing consistent alignments, we generate a "perturbed" version of the model data where each model has been randomly rotated by a uniformly sampled rotation in $SO(3)$. As many deep-learning methods rely on view-based feature computation, the second dataset is a more challenging scenario. All participating methods were evaluated on both datasets.

3. Evaluation

Each split of the dataset (train, val, test) was treated independently as a query and retrieval corpus. Participants were asked to return a ranked list of retrieved models for each model in a given set, where the target models to be retrieved were all models in that set, including the query model itself. Retrieval ranked lists were required to provide at most 1000 results in descending order of similarity (or relevance) to the query model. Although a similarity score is provided for each entry in the submission, it is not used in evaluation. Each participant method was free to choose whether to return fewer relevant retrieval results to obtain better evaluation performance.

The ranked lists were evaluated against the ground truth category (synset) and subcategory (subsynset) annotations. An item in the retrieved lists is positive if it is in the same category with the target model in the retrieval. Otherwise, it is negative. For each entry in the lists, we calculate the precision and recall. The precision at an entry is defined as the percentage of positive items up to this entry. The recall at an entry is defined as the number of positive items up to this entry divided by the smaller number between the total number of objects in the category or maximally allowed retrieved list length (1000 in this competition). Combining the precision and recall at each entry, we calculate the F-score.

At each entry, we also calculate normalized discounted cumulative gain (NDCG). The NDCG metric uses a graded relevance: 3 for perfect category and subcategory match in query and retrieval, 2 for category and subcategory both being same as the category, 1 for correct category and a sibling subcategory, and 0 for no match. Subcategory is only used in NDCG. The simplistic graded relevance we defined using just categories and subcategories is a somewhat limited attempt at capturing a human notion of graded relevance between 3D models. In the near future, the track organizers plan to collect judgments of retrieved 3D model relevance from people in order to establish a more proper relevance for retrieved models. This will allow computation of a more challenging version of the NDCG evaluation metric which will measure the degree to which retrieval methods match human notions of relevance/similarity ordering of the 3D models.

In summary, we can calculate four scores at each entry in a list: precision, recall, F-score and NDCG. In Section 6, we show precision-recall plots to compare different submitted methods. Besides using average precision to evaluate the quality of each retrieved list, we also use precision, recall, F1 and NDCG at the end of the retrieval list as a summary metric. These metrics are referred to as P@N, R@N, F1@N and NDCG@N, where the N refers to the total retrieval list length chosen by the method, and is allowed to vary across queries. The scores of the lists in a category are combined by taking the average.

In order to combine the retrieval results of different categories, we use two versions of the above metrics: macro and micro averaged. The macro-averaged version is used to give an unweighted average over the entire dataset. The retrieval scores for all the models are averaged with equal weights. In the micro-averaged version, each query and retrieval results are treated equally across categories, and therefore the results are averaged without reweighting based on category size. This gives a representative performance metric average across categories.

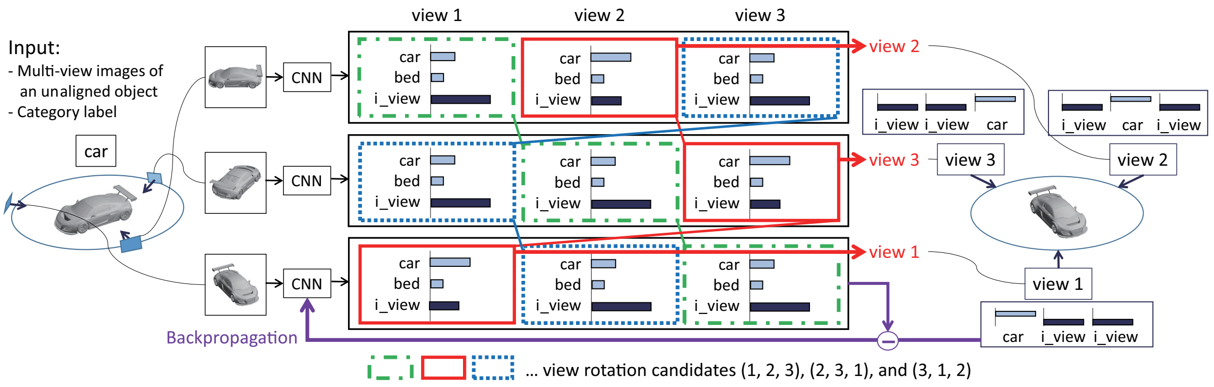
view 1
view 2
view 3
Input:
- Multi-view images of an unaligned object
- Category label
car
CNN
car
bed
i_view
view 2
view 3
view 1
Backpropagation
... view rotation candidates (1, 2, 3), (2, 3, 1), and (3, 1, 2)

Figure 1: *Overview for **RotationNet**. Illustration of the training process of RotationNet, where the number of views M is 3 and the number of categories N is 2. A training sample consists of M images of an unaligned object instance and its category label y. For each input image, our CNN (RotationNet) outputs M histograms with N + 1 bins whose norm is 1. The last bin of each histogram represents the "incorrect view" class, which serves as a weight of how likely the histogram does not correspond to each view label. According to the histogram values, we decide which image corresponds to views 1, 2, and 3. There are three candidates for view rotation: (1, 2, 3), (2, 3, 1), and (3, 1, 2). For each candidate, we calculate the score for the ground-truth category ("car" in this case) by multiplying the histograms and selecting the best choice: (2, 3, 1) in this case. Finally, we update the CNN parameters in a standard back-propagation manner with the estimated view labels.*

4. Participants

There were eight participating groups in this track. Each group submitted results on both the normal and perturbed versions of the test data, with one exception of a group that did not submit perturbed dataset results.

- **RotationNet**, by A. Kanezaki (referred to as **Kanezaki**)
- **GIFT**: Learning Deep Shape Similarity via Improved GIFT, by Z.Zhou, S.Bai, R.Yu, X.Bai (referred to as **Zhou**)
- **ReVGG**: Reduced VGG-M Network and Modified Neighbor Set Similarity, by A. Tatsuma and M. Aono (referred to as **Tatsuma**)
- **DLAN**: Deep aggregation of local 3D geometric features, by T. Furuya and R. Ohbuchi (referred to as **Furuya**)
- **MVFusionNet**: Multi-view Fusion Network, by S. Thermos, A. Axenopoulos, G. Th. Papadopoulos and P. Daras (referred to as **Thermos**)
- **CM-CNN**: View based 3D Shape Retrieval using Clock Matching and Convolutional Neural Networks, by X. Deng, Z. Lian (referred to as **Deng**)
- **ZFDR**: Hybrid Shape Descriptor ZFDR, by B. Li, H. Johan, and Y. Lu (referred to as **Li**)
- **VoxelNet**: Deep VoxelNet to extract features from voxel grids, by S. Mk (referred to as **Mk**)

5. Methods

In this section we compile the description of methods by each participating group, from the participants' perspective.

5.1. RotationNet, by A. Kanezaki

Our method is based on a Convolutional Neural Network (CNN)-based model "RotationNet" [KMN16]. Similarly to Multi-view CNN [SMKLM15], it takes multi-view images of an object as input and estimates its object category. Our method treats the pose labels as latent variables, which are optimized to self-align in an unsupervised manner during the training using an unaligned dataset. The code is available on `https://github.com/kanezaki/rotationnet`.

5.1.1. Training of RotationNet

The training process of RotationNet is illustrated in Fig. 1. We assume that multi-view images of each training object instance are observed from all the pre-defined viewpoints. Let M be the number of the pre-defined viewpoints and N denote the number of target object categories. A training sample consists of M images of an object $\{\boldsymbol{x}_i\}_{i=1}^M$ and its category label $y \in \{1,\dots,N\}$. We attach a view label variable $v_i \in \{1,\dots,M\}$ to each image $\boldsymbol{x}_i$ and set it to j when the image is observed from the j-th viewpoint, *i.e.*, $v_i \leftarrow j$. In our method, only the category label y is given during the training whereas the view labels $\{v_i\}$ are unknown, namely, $\{v_i\}$ are treated as latent variables that are optimized in the training process. We introduce an "incorrect view" label and append it to the target category labels. Letting $P_i = \{p_{j,k}^{(i)}\} \in \mathbb{R}_+^{M\times(N+1)}$ denote a matrix composed of $P(\hat{y}_i \mid \boldsymbol{x}_i, v_i)$ for all the M viewpoints and $N+1$ classes, the target value of P_i in the case that v_i is correctly estimated is defined as follows:

$$p_{j,k}^{(i)} = \begin{cases} 1 & (j = v_i \text{ and } k = y)\text{ , or} \\ & (j \neq v_i \text{ and } k = N+1) \\ 0 & (\text{otherwise}). \end{cases} \quad (1)$$

The parameters of RotationNet are iteratively updated via standard back-propagation of M softmax losses, whereas $\{v_i\}_{i=1}^M$ are determined in a manner to maximize the probability that a training sample belongs to its correct category y in each iteration.

We place virtual cameras on the $M = 20$ vertices of a dodecahedron encompassing the object. There are three different patterns of rotation from a certain view, because three edges are connected

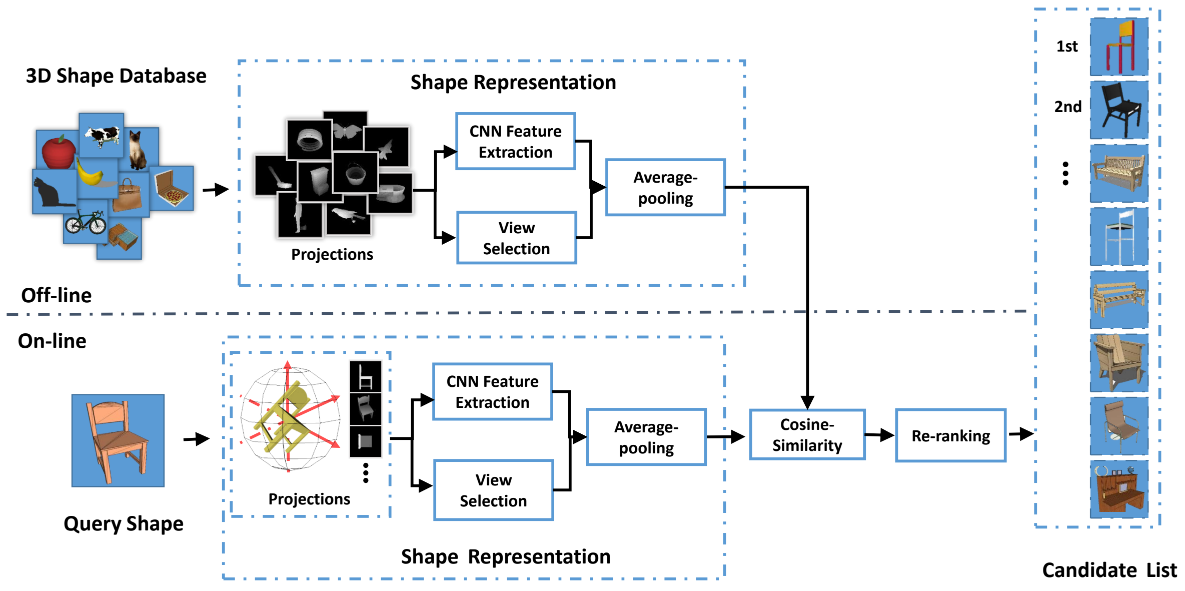

Figure 2: *Overview for **Improved GIFT**. The pipeline of the Improved GIFT method is illustrated.*

to each vertex of a dodecahedron. Therefore, the number of candidates for all the view labels $\{v_i\}_{i=1}^{M}$ is 60 ($= 3M$).

5.1.2. Retrieval

Four RotationNet models are trained respectively with (a) train+"normal" set in 55 categories, (b) train+"normal" set in 203 subcategories, (c) train+"perturbed" set in 55 categories, and (d) train+"perturbed" set in 203 subcategories. We predict category/subcategory labels of all the samples in train/val/test+"normal" set using models trained with (a) and (b), whereas we use models trained with (c) and (d) for train/val/test+"perturbed" set. For each query, we construct a retrieval set containing all the samples with the same predicted category label in the order of descending scores of the predicted category. Then we re-rank the results so that the samples with the same predicted subcategory label are ranked higher than others.

5.2. Learning Deep Shape Similarity via Improved GIFT, by Z. Zhou, R. Yu, S. Bai, X. Bai

We present a view-based shape search engine (see Figure 2) based on GIFT [BBZ*16], which is composed of four components, *i.e.*, projection rendering, view feature extraction, multi-view matching and re-ranking. We render each 3D shape with N_v ($N_v = 50$ in our experiment) depth images. Compared with the original GIFT, we propose several improvements to boost the retrieval accuracy and matching speed, detailed below.

5.2.1. View Feature Extraction

Similar to GIFT [BBZ*16], we utilize Convolution Neural Network (CNN) to extract view features, and fine-tune a pre-trained CNN model (VGG-S from [CSVZ14a]) using the rendered views. However, different from GIFT which simply assigns the projections to the label of their corresponding 3D shape, we propose a simple trick which handles the semantic clutter, *i.e.*, shapes from different categories may share very similar views at specific view points. In each batch of one iteration, we randomly select a portion of projections (30 in our experiment) from each 3D shape as the training exemplar, then exert average-pooling to aggregate those projections. The entire of network is trained using back-propagation with classification loss on the 3D shapes.

In the testing phase, we fed the projections of the testing 3D shapes to the neural network in the forward direction, and extract the activations of the 7-th fully-connected layers. The activations are then L_2 normalized, serving as the features of the projections.

5.2.2. Multi-View Matching

Given a query shape x_q and a certain shape x_p from the database $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$, we can easily obtain two feature sets $\mathcal{V}(x_q) = \{q_1, q_2, \ldots, q_{N_v}\}$ and $\mathcal{V}(x_p) = \{p_1, p_2, \ldots, p_{N_v}\}$ respectively using the trained neutral network.

5.2.3. Matching Function

In order to compute the pairwise similarity between shapes, we first get the feature of the 3D shape x_q via average-pooling as

$$F_q = \frac{1}{N_v} \sum_i q_i. \tag{2}$$

and vice versa for the shape x_p. Then, the similarity between x_q and x_p is computed directly using the cosine similarity

$$S(x_q, x_p) = F_q^{\mathrm{T}} F_p. \tag{3}$$

We draw the readers' attention that the above matching strategy is equivalent to the modified version of Hausdorff distance, which simply averaging all the pairwise similarities between two view sets, that is,

$$S(x_q, x_p) = \frac{1}{N_v^2} \sum_i \sum_j q_i^{\mathrm{T}} q_j. \tag{4}$$

Since set-to-set matching is computational expensive, the original GIFT [BBZ*16] employed inverted file to accelerate this computation. In this work, we suggest the alternative way presented in Eq. 2 and Eq. 3, which performs multi-view matching in vector space model.

5.2.4. Enhance Matching Accuracy

In general, there exist non-discriminative projections, which capture irrelevant information to represent a 3D shape. To further enhance the matching accuracy, we aim at only preserving discriminative projections while discarding those noisy projections. To this end, we define the degree of discrimination D of i-th view of x_q as

$$D_{q_i} = \exp\left(\sum_{j=1}^{C} c_{q_i}^j \cdot \log c_{q_i}^j\right), \tag{5}$$

where C denotes the number of categories in the training set ($C =$ 55 in our experiment), and $c_{q_i}^j$ denotes the classification probability of q_i to the j-th category. It suggests that we deem the view is discriminative if it has larger value of D (it means the view has a large probability belonging to only a small number of categories while unlike to be classified to the rest categories). In our experiment, we only choose top-40 views with larger discriminative scores, and discard the rest views.

5.2.5. Re-ranking

The re-ranking component is the same as GIFT [BBZ*16]. Its principle is to refine the input similarity by comparing the neighbourhood sets of two 3D shapes via the generalized Jacquard distance.

5.3. Reduced VGG-M Network and Modified Neighbor Set Similarity, by A. Tatsuma and M. Aono

Our approach consists of feature extraction from a Convolutional Neural Network (CNN) [LBD*89] with reduced number of filters for depth-buffer images and similarity calculation by an improved method of Neighbor Set Similarity (NSS) [BBW15].

We extract the feature vector of 3D model by inputting the rendered depth-buffer images to CNN. As a preprocessing, we first normalize the position and scale of 3D model. We translate the center of the 3D model to the origin and then normalize the size of the 3D model to the unit sphere. In addition, for the perturbed data set, we normalize the rotation of 3D model by using Point SVD [TA09]. Next, we rendered 38 depth-buffer images with 224×224 resolutions by setting the view point to each vertex of the unit geodesic sphere. Finally, we obtain the feature vector of 3D model by averaging the CNN output vectors, which denote the classification probability, of each depth buffer image. For the dissimilarity between two feature vectors, we employ the Euclidean distance.

Table 1: *Reduced VGG-M network configuration.*

Layer	Setup
input	224×224 grayscale image
conv1	$7 \times 7 \times 48$ stride 2 pool 2×2
conv2	$5 \times 5 \times 64$ stride 2 pool 2×2
conv3	$3 \times 3 \times 128$ stride 1
conv4	$3 \times 3 \times 128$ stride 1
conv5	$3 \times 3 \times 128$ stride 1 pool 2×2
full6	1024 dropout
full7	1024 dropout
full8	55 softmax

Our CNN configuration is summarized in the Table 1. The network can be seen as a network with reduced number of filters and units in the VGG-M network [CSVZ14b]. The activation function for all layers except the full8 layer is the REctification Linear Unit [NH10]. The pooling method for the convolutional layers is max pooling. We call the network the Reduced VGG-M network (ReVGG). We trained the ReVGG with training dataset classified to 55 classes.

In the calculation similarity, we propose an improvement method of NSS, named Modified NSS (MNSS). NSS defines the similarity between two feature vectors as the average of all the pairwise similarities between two neighbor sets. Let $\mathcal{N}_a$ and $\mathcal{N}_b$ be the sets of k-nearest neighbors of the feature vector $\mathbf{a}$ and $\mathbf{b}$. The NSS S_{NSS} between $\mathbf{a}$ and $\mathbf{b}$ is defined as follows:

$$S_{NSS}(\mathbf{a}, \mathbf{b}) = \frac{1}{k^2} \sum_{\mathbf{x} \in \mathcal{N}_a} \sum_{\mathbf{y} \in \mathcal{N}_b} s(\mathbf{x}, \mathbf{y}), \tag{6}$$

where $s(\mathbf{x}, \mathbf{y}) = \exp(-d(\mathbf{x}, \mathbf{y})^2 / 2\sigma^2)$ is the similarity, and $d(\mathbf{x}, \mathbf{y})$ is the distance between two feature vectors.

The NSS does not consider the similarity between feature vectors which are subject to similarity calculation. To capture the structure of data distribution, we think that the similarity between the subject feature vectors is also important. Thus, we define the MNSS between $\mathbf{a}$ and $\mathbf{b}$ is as follows:

$$S_{MNSS}(\mathbf{a}, \mathbf{b}) = \begin{cases} 1 & \text{if } \mathbf{a} = \mathbf{b} \\ \alpha s(\mathbf{a}, \mathbf{b}) + (1 - \alpha) S_{NSS}(\mathbf{a}, \mathbf{b})) & \text{otherwise} \end{cases}, \tag{7}$$

where $\alpha \in [0, 1)$ is the parameter representing the influence degree of neighbors to the MNSS. In addition, to emphasize neighborhood relations, MNSS updates the similarity iteratively by repeating the calculation of similarity in Eq. (7). Since the similarity value with MNSS has the range $[0, 1]$, we obtain the dissimilarity value by calclulating $1 - S_{MNSS}(\mathbf{a}, \mathbf{b})$.

In our run, we set k to 10, σ to 0.5, α to 0.9, and the number of iterations in the MNSS to 5. We output models by which the dissimilarity to the query is less than 0.8.

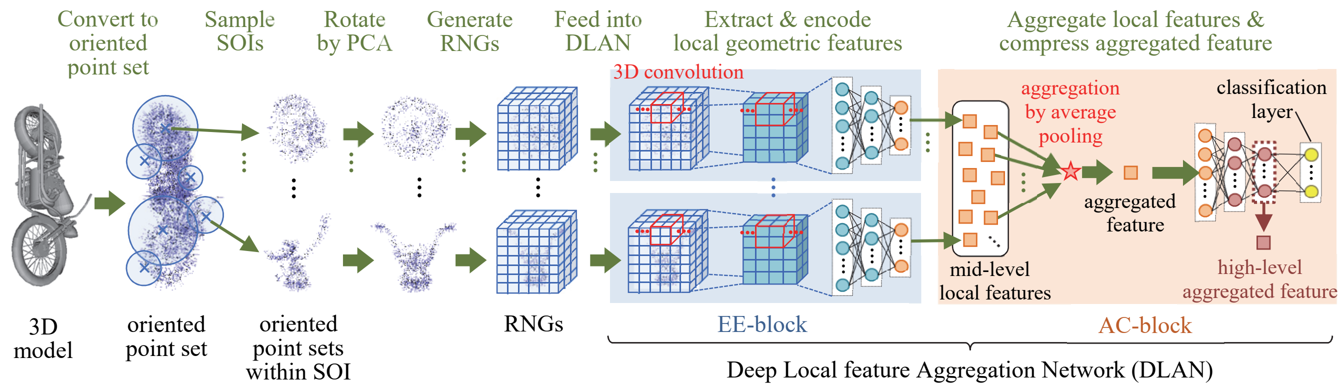
Convert to oriented point set
Sample SOIs
Rotate by PCA
Generate RNGs
Feed into DLAN
Extract & encode local geometric features
Aggregate local features & compress aggregated feature
3D convolution
aggregation by average pooling
classification layer
aggregated feature
mid-level local features
high-level aggregated feature
3D model
oriented point set
oriented point sets within SOI
RNGs
EE-block
AC-block
Deep Local feature Aggregation Network (DLAN)

Figure 3: *Overview for **DLAN**. First half of DLAN, the EE-block, extracts rotation-invariant, mid-level local features from Rotation Normalized Grids (RNGs). The AC-block aggregates the set of mid-level local features into a feature per 3D model by average-pooling. The pooling is followed by deep learning-based dimension reduction to produce a salient and compact feature per 3D model.*

5.4. Deep aggregation of local 3D geometric features, by T. Furuya and R. Ohbuchi

The algorithm aims at extracting 3D shape descriptors that are robust against geometric transformations including translation, uniform scaling, and rotation of 3D models. The algorithm is called Deep Local feature Aggregation Network (DLAN) [FO16]. DLAN takes as its input a set of low-level 3D geometric features having invariance against 3D rotation. It produces a compact, high-level descriptor per 3D model for efficient and effective matching among 3D shapes. Figure 1 describes the processing pipeline of DLAN, which consists of the following four steps.

1) Generating oriented point set: Given a polygon-based 3D model, it is first converted into a 3D oriented point set by sampling the surfaces of the 3D model by using the algorithm by Osada et al. [OFCD02]. This sampling process gives the DLAN certain invariance against shape representations of 3D model. The algorithm randomly and uniformly samples points on the surfaces of the 3D model. Each point is associated with the normal vector of the triangle on which the point is sampled. We sample 3k oriented points per 3D model. The oriented point set of the 3D model is uniformly scaled to fit a sphere having diameter 1.0.

2) Extracting rotation-invariant local features: We sample a set of Spheres-Of-Interest (SOIs) from the oriented point set of the 3D model. Position and radius of each SOI are chosen randomly. To make local features invariant against rotation of 3D shapes, each SOI is rotated to its canonical orientation, which is computed by applying Principal Component Analysis to points enclosed by the SOI. The rotation-normalized SOI is then spatially divided by using 3D grids having 7, 6, and 3 grid intervals along the first, second, and third principal axis, respectively. We refer to the 3D grid as Rotation-Normalized Grid (RNG). For each cell in the RNG, a 3D geometric feature POD [FO15] is computed to describe low-level 3D geometric feature within the cell. We extract 100 RNGs per 3D model.

3) Aggregating local features via DLAN: DLAN consists of 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 into a feature per 3D model followed by Compression of the aggregated feature. The EE-block encodes each RNG to a mid-level local feature per SOI via two 3D convolution layers followed by three fully-connected layers. The aggregation layer in the AC-block pools, by averaging, the set of encoded mid-level local features into a single feature per 3D model. The subsequent three fully-connected layers compress the aggregated feature to generate a compact and salient feature per 3D model. Detailed description of DLAN is found in [FO16].

4) Comparing aggregated features: A compact (i.e., 128-dimensional) aggregated feature of the query 3D model is efficiently compared against the aggregated features of the retrieval target 3D models in the database. Cosine similarity is used to compare a pair of aggregated feature vectors. Retrieval ranking for the query consists of 3D models whose similarity to the query is larger than 0.95.

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

A) Pre-training EE-block: To learn expressive local feature better suited for accurate 3D shape retrieval, we pre-train the EE-block by using a large number of labeled RNGs so that it could predict object categories of the RNGs. To do so, we sample RNGs having random position and scale from the labeled 3D models contained in the training dataset. The label for each training RNG is identical to the label for the 3D model from which the RNG is sampled. We collect 2M labeled RNGs from the training dataset. For the supervised training, we append the classification layer with softmax function at the output end of the EE-block. Parameters, i.e., connection weight among neurons, in the EE-block are randomly initialized and cross-entropy loss is minimized by using AdaGrad algorithm [DHS11] with initial learning rate = 0.1. We perform 50% dropout for all the layers except for the input layer and the output (i.e., classification) layer. We iterate the pre-training for 100 epochs.

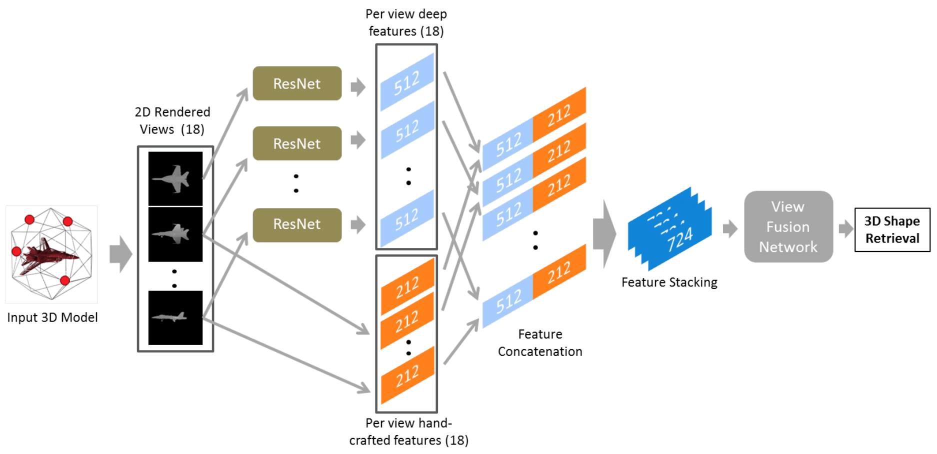

Figure 4: *Overview for* ***MVFusionNet****. Illustration of the pipeline for the Multi-view Fusion Network.*

B) 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 in the training dataset. The parameters in EE-block inherits as its initial values the result of its pre-training. The parameters in AC-block are randomly initialized. Similar to pre-training the EE-block, training of the entire DLAN is done also by minimizing cross-entropy loss by using AdaGrad algorithm with initial learning rate = 0.1 and 50% dropout during the training. We iterate the training for 70 epochs.

5.5. Multi-view Fusion Network, by S. Thermos, A. Axenopoulos, G. Th. Papadopoulos, P. Daras

The proposed framework fuses the features extracted using a deep Neural Network (NN) and a multi-view descriptor. The NN and the hand-crafted feature extraction procedure receive rendered views of 3D shapes as input. Fusion of the aforementioned features improves the 3D shape retrieval.

5.5.1. Pre-processing and feature extraction

The input 3D model may have arbitrary pose. Thus, prior to extraction of 2D views, a rotation estimation step is necessary to ensure that the sequence of 2D views is consistent for all models of the same category. In the proposed framework, the Combined Pose Estimation (CPE) method [DAL12] is applied, which intuitively merges the well-known Continuous Principal Component Analysis (CPCA) with plane symmetry and rectilinearity.

Extraction of hand-crafted features is based on the Compact Multi-View Descriptor (CMVD), which is presented in [DA09]. After rotation estimation, a set of uniformly distributed views are extracted. The viewpoints are chosen to lie at the 18 vertices of a regular 32-hedron. The 2D image types are depth images with pixel values in the range $[0, 255]$. Three rotation-invariant functionals are applied to each view to produce the descriptors: i) 2D Polar-Fourier Transform, ii) 2D Zernike Moments, and iii) 2D Krawtchouk Moments. The total number of descriptors per view is $ND = NFT + NZern + NKraw = 212$, where $NFT = 78$, $NZern = 56$, and $NKraw = 78$. For the deep features extraction, each of the 18 views of a 3D shape is passed through a ResNet [HZRS16] model with 18 layers. A 512-dimensional feature vector is extracted per view. The ResNet-18 network is pre-trained on ImageNet [RDS*15] and fine-tuned on ShapeNetCore55.

The ShapeNetCore55 benchmark consists of 55 categories, which are highly imbalanced. In the training set, the smallest category has 39 while the largest one has 5876 samples, respectively. The subcategories are also imbalanced. In order to balance the category distribution, we randomly sample a training set for each epoch according to a penalized category distribution using the same method with [SMKL15]. We use $t = 0.5$ and $t = 0.2$ as a penalty factory for category and subcategory distribution respectively.

5.5.2. Feature Fusion and Object recognition

In this section the proposed feature fusion and object recognition mechanisms are summarised. The 512-dimensioanl deep feature vector is concatenated with the corresponding 212-dimensional hand-crafted features for every considered view, as can be seen in Figure 4. All produced 724-dimensional vectors are stacked and passed through the Fusion Network. The latter consists of a 2D convolutional layer with 18 1×1 size kernels, a max pooling operator and two 4096-dimensional fully connected layers. The convolutional layer learns the inter-view spatial correlations and the pooling layer adds spatial invariance while propagating the most dominant activations.

The features extracted from ResNet's last layer and the hand-crafted ones are normalized before the concatenation.

5.5.3. Retrieval

The architecture described above is trained for both the categories and the subcategories of the dataset, resulting in two separate networks, Network1 and Network2, respectively. At test time, Network1 is used for category prediction, while Network2 is used for feature extraction, taking the last fully connected layer. Each

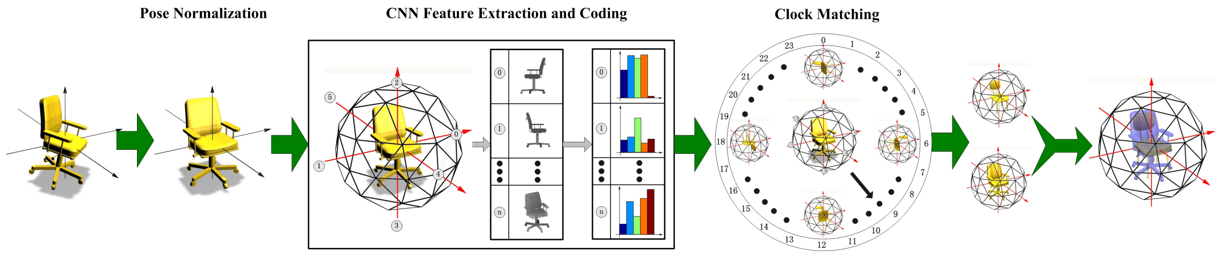

Figure 5: *Overview for **CM-VGG**. The pipeline of the method is illustrated.*

query model is passed through Network1, which predicts its category. Based on this prediction, an initial retrieval set containing all the models predicted with the same category is constructed. Then, the query and the initial retrieval set are passed through Network2, which computes the $L2$-distance between the query and the models, producing the final rank list. Models with $L2$-distance above a certain threshold are dropped (i.e. misclassified by Network1).

5.6. CM-VGG: View based 3D Shape Retrieval using Clock Matching and Convolutional Neural Networks, by X. Deng and Z. Lian

As shown in Figure 5, our method consists of the following three steps:

1) Pose Normalization: Given a 3D model which is perturbed by random rotations, a pose normalization algorithm based on Principal Component Analysis (PCA) (see [LRS10]) is implemented to get the normalized pose for the model. Note that for models that have already been perfectly aligned, this step can be skipped over.

2) Feature Extraction: After pose normalization, we capture the model's depth-buffer views on the vertices of a given unit geodesic sphere whose mass center is also located in the origin. The geodesic spheres used here are obtained by subdividing the unit regular octahedron(see Figure 5). Based on different times (0 and 2) of subdivision, the numbers of depth-buffer images captured are 6 and 66, respectively. During offline period, we train the VGG convolutional neural network with 16 layers proposed in [SZ14] on the training set whose models are classified into 55 categories by using the source code named MatConvNet [VL15]. Furthermore, we use parameters of the VGG-16 model pre-trained on the imagenet database as the initial parameters of the CNN model adopted in our method. In this way, the modified VGG model can be trained to get satisfactory performance for this task in just a few hours on a PC with a Titan X GPU. Online, we extract a 4096-dimensional feature vector from each view of the 3D model by feeding the depth-buffer image into the trained CNN and recording the output of feature layer of the network. To reduce the storage requirement, an efficient feature coding scheme proposed in [LGSX13] is also adopted. In this manner, for example, the original feature vector that contains 4096 float numbers can be coded into a feature vector with only about 1000 integers. Finally, all feature vectors of the views captured around the 3D model are combined sequentially into a single feature vector that describes the 3D shape.

3) Dissimilarity Calculation: An multi-view shape matching (Clock Matching) algorithm [LGSX13] is utilized to calculate the dissimilarity between two 3D objects by computing the minimum distance of 24 matching pairs.

For the contest, the number of views we capture around the 3D model is selected as 6 and the subcategory labels are not used for the training. Since the method is mainly based on *Clock Matching*, the *VGG* CNN model, and utilizes 6 views for each model, we denote it as "CM-VGG55-6DB". In fact, the performance of our approach can be further improved by increasing the number of views captured around the 3D model.

5.7. Hybrid Shape Descriptor ZFDR, by B. Li, H. Johan and Y. Lu [LJ13]

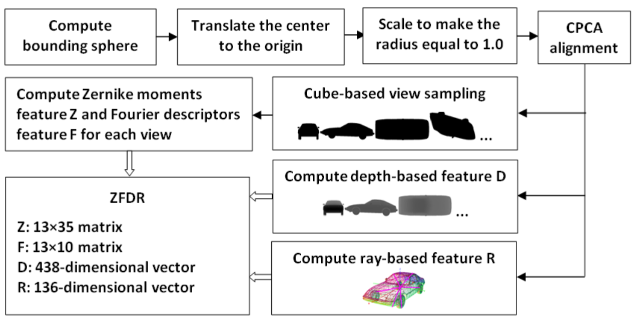
Compute bounding sphere
Translate the center to the origin
Scale to make the radius equal to 1.0
CPCA alignment
Compute Zernike moments feature Z and Fourier descriptors feature F for each view
Cube-based view sampling
...
ZFDR
Z: 13×35 matrix
F: 13×10 matrix
D: 438-dimensional vector
R: 136-dimensional vector
Compute depth-based feature D
...
Compute ray-based feature R

Figure 6: *Overview of **ZFDR** [LJ13]. Figure illustrates the feature extraction process.*

ZFDR [LJ13] is a non-learning based hybrid shape descriptor, which integrates both visual and geometric information of a 3D model: (1) Thirteen cube-based sampling views' **Z**ernike moments and **F**ourier descriptor features; (2) Six standard depth buffer views' **D**epth information features and the 3D model's **R**ay-based features.

The intention of running ZFDR on ShapeNet Core55 is to examine the performance of this typical hybrid 3D shape descriptor on the latest large-scale 3D model dataset ShapeNet Core55, thus providing a baseline in the aspect of non-learning based approaches. It will help to find out the performance gap between non-learning based approaches and learning-based ones, such as all the four deep learning-based participating methods [SYS*16] in the last track based on the same ShapeNet Core55 dataset.

The overview of ZFDR feature extraction process is illustrated in Figure 6. Continuous Principle Component Analysis (CPCA) [Vra04] is applied for 3D model normalization. Details about extraction of the four component features **Z**, **F**, **D** and **R** are described as follows.

1) Visual features: Zernike moments and Fourier descriptors (ZF). (1) Cube-based view sampling: With a consideration for both efficiency and accuracy, a set of thirteen silhouette feature views are sampled by setting cameras on a cube's four top corners, three adjacent face centers and six middle edge points. (2) Feature extraction: To represent the region-based features of each silhouette feature view, up to 10th order Zernike moments [KH90] (totally 35 moments) are computed; while to characterize its contour-based visual features, first 10 centroid distance-based Fourier descriptors [ZL01] are computed.

2) Geometric features: Depth information and Ray-based features (DR). To be more powerful in characterizing diverse types of 3D models, two additional features D and R, developed by Vranic [Vra04], are integrated together with Z and F such that we form the proposed hybrid shape descriptor ZFDR [LJ13]. D are the depth buffer-based features of six standard depth views of a 3D model, while R are the ray-based features of the 3D model which are generated by shooting a number of rays from the center of the model to its farthest surface intersection points.

Finally, a ZFDR hybrid shape descriptor distance is generated by linearly combining the four component distances d_Z, d_F, d_D, and d_R, which are measured based on Scaled-L1 [Vra04] or Canberra distance metric.

Without training, it is direct and efficient to extract our ZFDR shape descriptor and to perform retrieval. The method is implemented in C/C++ and run on a Windows laptop machine with a 2.70 GHz Intel Core i7 CPU and 16.0 GB memory. The running time information is as follows. The average time to calculate the ZFDR shape descriptor of a 3D model is 2.67 seconds (CPCA: 1.45, ZF: 1.10, D: 0.03, R: 0.06). On average, it takes 34.2 microseconds to compare a pair of 3D models, or 1.75 seconds to compare one query model with all the 51162 target 3D models. Totally, it takes 4.42 seconds to perform a query, which includes feature extraction for the query model and comparison with all the 51162 3D models in the target dataset.

5.8. Deep VoxelNet, by S. Mk

We trained a 3D convolution network on 64x64x64 binary voxel grids. The network was trained to classify the input grids into 203 categories given by the subsynset ids in the ShapeNet Core 55 dataset. An intermediate layer of this trained network is then used as a feature vector to be able to perform 3D shape retrieval.

5.8.1. Pre-processing

The only processing applied is to convert the given OBJ files into binary voxel grids. We used *binvox* [Min11] to perform this discretization.

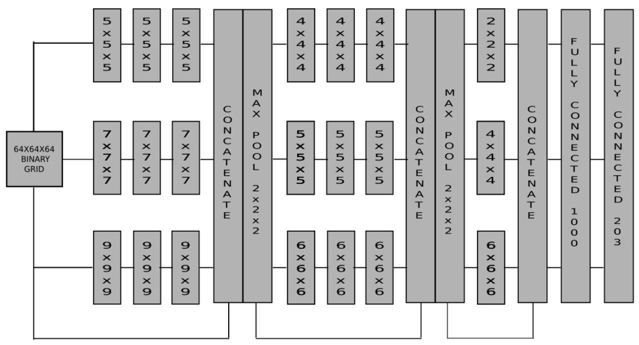

Figure 7: *Overview of **VoxelNet**. The network structure of the Deep VoxelNet is summarized.*

5.8.2. Network architecture and training

The network architecture is as illustrated in *Figure*1 . The network is 3-wide and 7-deep, and every filter depth is 1 (that is, every filter has a 1-channel output), followed by ReLU activation. After each filter, zero padding is done to maintain the output size same as input size. The voxel grid is fed through 3 branches, where each branch has 3 layers. The outputs after the 3 layers and the input are concatenated along the channel axis. We then apply Max Pooling of 2x2x2, thus halving the size. The pooled output is again sent through three 3-layer branches, this time with smaller filter sizes. The outputs at the sixth layer and the first max pool output are again concatenated, and halved by max pooling. This output is then sent to three 1-layer branches. Their outputs and the second max pool output are concatenated and then flattened into a vector. This is then fully connected to a 1000-neuron layer. The penultimate 1000-neuron layer is again fully connected to a 203-neuron layer which provides the classification outputs. We then apply softmax over the 203 outputs, to provide a distribution over the classes.

The weights in the network are initialized by the Xavier uniform initialization method and the entire network is trained from scratch using Stochastic Gradient Descent with Nesterov momentum. The network was trained for 6 iterations over the entire training set of 70000+ shapes (normal and perturbed). The classification loss used was categorical cross-entropy.

For all of the above, we used Tensorflow running on two GTX-1070 GPUs. We achieved a classification accuracy of 75.8% on the normal training set, and 74.8% on the perturbed training set. Validation accuracy obtained was 52.3% on normal, and 31% on the perturbed set.

5.8.3. Retrieval

For every voxel grid, we use the 1000-dimensional vector output by the penultimate layer of the network as the corresponding feature vector. Retrieval was performed by building a KD-Tree on these feature vectors. For every shape, we query the nearest neighbors in this 1000-dimensional feature space and output the retrieved results.

		micro					macro				
Dataset	Method	P@N	R@N	F1@N	mAP	NDCG@N	P@N	R@N	F1@N	mAP	NDCG@N
Normal	Kanezaki	0.810	0.801	**0.798**	**0.772**	**0.865**	0.602	0.639	**0.590**	**0.583**	0.656
	Zhou	0.786	0.773	0.767	0.722	0.827	0.592	0.654	0.581	0.575	**0.657**
	Tatsuma	0.765	**0.803**	0.772	0.749	0.828	0.518	0.601	0.519	0.496	0.559
	Furuya*	**0.818**	0.689	0.712	0.663	0.762	**0.618**	0.533	0.505	0.477	0.563
	Thermos	0.743	0.677	0.692	0.622	0.732	0.523	0.494	0.484	0.418	0.502
	Deng	0.418	0.717	0.479	0.540	0.654	0.122	**0.667**	0.166	0.339	0.404
	Li	0.535	0.256	0.282	0.199	0.330	0.219	0.409	0.197	0.255	0.377
	Mk†	0.793	0.211	0.253	0.192	0.277	0.598	0.283	0.258	0.232	0.337
	SHREC16-Su	0.770	0.770	0.764	0.735	0.815	0.571	0.625	0.575	0.566	0.640
	SHREC16-Bai	0.706	0.695	0.689	0.640	0.765	0.444	0.531	0.454	0.447	0.548
Perturbed	Furuya*	**0.814**	0.683	0.706	0.656	0.754	**0.607**	0.539	**0.503**	**0.476**	**0.560**
	Tatsuma	0.705	**0.769**	**0.719**	**0.696**	**0.783**	0.424	0.563	0.434	0.418	0.479
	Zhou	0.660	0.650	0.643	0.567	0.701	0.443	0.508	0.437	0.406	0.513
	Kanezaki	0.655	0.652	0.636	0.606	0.702	0.372	0.393	0.333	0.327	0.407
	Deng	0.412	0.706	0.472	0.524	0.642	0.120	**0.659**	0.164	0.329	0.395
	Li	0.496	0.234	0.258	0.172	0.303	0.199	0.373	0.179	0.215	0.336
	Mk†	0.690	0.012	0.020	0.009	0.043	0.546	0.052	0.052	0.047	0.109
	SHREC16-Bai	0.678	0.667	0.661	0.607	0.735	0.414	0.496	0.423	0.412	0.518
	SHREC16-Su	0.632	0.613	0.612	0.535	0.653	0.405	0.484	0.416	0.367	0.459

Table 2: *Summary table of evaluation metrics for all participating teams and methods. Methods are ranked by the average of the micro and macro mAP. * indicates the method is based on point set representation and †indicates the method is based on volumetric representation.*

6. Results

The summary results for all participating methods are given in Table 2. The corresponding precision-recall plots are given in Figure 8. We can draw several conclusions from the summary evaluation metrics.

First of all, it is exciting to observe that the top submissions this year outperform the best methods in SHREC16 significantly. Similar to last year's competition, most of the submissions are based on deep learning approaches. Shape retrieval is benefiting from the recent developments in deep learning methods for computer vision.

However, unlike last year when all approaches were projection-based, the approaches this year are based on various input formats, including multiple projected views, point sets, 3D voxel grids and traditional 3D shape descriptors. The point set–based approach of **Furuya** has the best overall performance for the randomly oriented 3D model dataset. The projection-based approach of **Kanezaki** performs the best on the normal, consistently aligned 3D models.

The overall retrieval performance on the normal test dataset as measured by micro-averaged mAP and NDCG values is fairly high (in the 70-80% range for mAP and in the high 80% to 90% range for NDCG). The mAP measure is highly correlated with the AUC giving a sense of the area under the precision-recall curve for each method's chosen retrieval list length. Similarly, NDCG evaluates the overall relevance ranking of the retrieved model list against an ideal ranking of the same retrieved list. All methods perform well when judged on the chosen retrieval list lengths. However, the methods have very different trade offs of performance when we consider precision and recall. For example, **Mk** have high P@N values but quite low R@N values, whereas **Deng** have high R@N values but lower P@N values. Other methods are more balanced in precision and recall.

When using macro-averaging instead, we no longer adjust for class sizes, thus giving proportionally more weight to synsets with fewer models. The same relative patterns between methods as observed in micro-averaging hold with macro-averaging as well, but the overall performance for all methods is significantly lower, as expected (F1@N scores drop from a high in the 70% range to a high in the 50% range).

When contrasting the normal dataset with the perturbed dataset we see that, as expected, there is a drop in performance for all methods, though this drop is much less for **Furuya** which comes up as the best performer for un-aligned 3D model data. This is an interesting strengh of the approach which is based on the local features learned on an oriented point set of the 3D mesh.

Overall, deep learning models again achieve high performance in this competition compared to the more traditional approach by **Li**. Projection-based inputs are used by most submissions, because it is easy to make use of additional image data to help learn features for 3D shapes. Because of this, the models are based on image classification models and they are not built from scratch for the 3D problems. However, in the case of unknown rotations, the point set–based approach of **Furuya** which uses local point features outperforms the view–based approaches. These results indicate that an even more targeted exploration of the impact of different 3D shape representations for deeply learned methods would be interesting. Though projection-based models have dominated in computer vision, there is much space for improvement using volumetric, or point set–based methods.

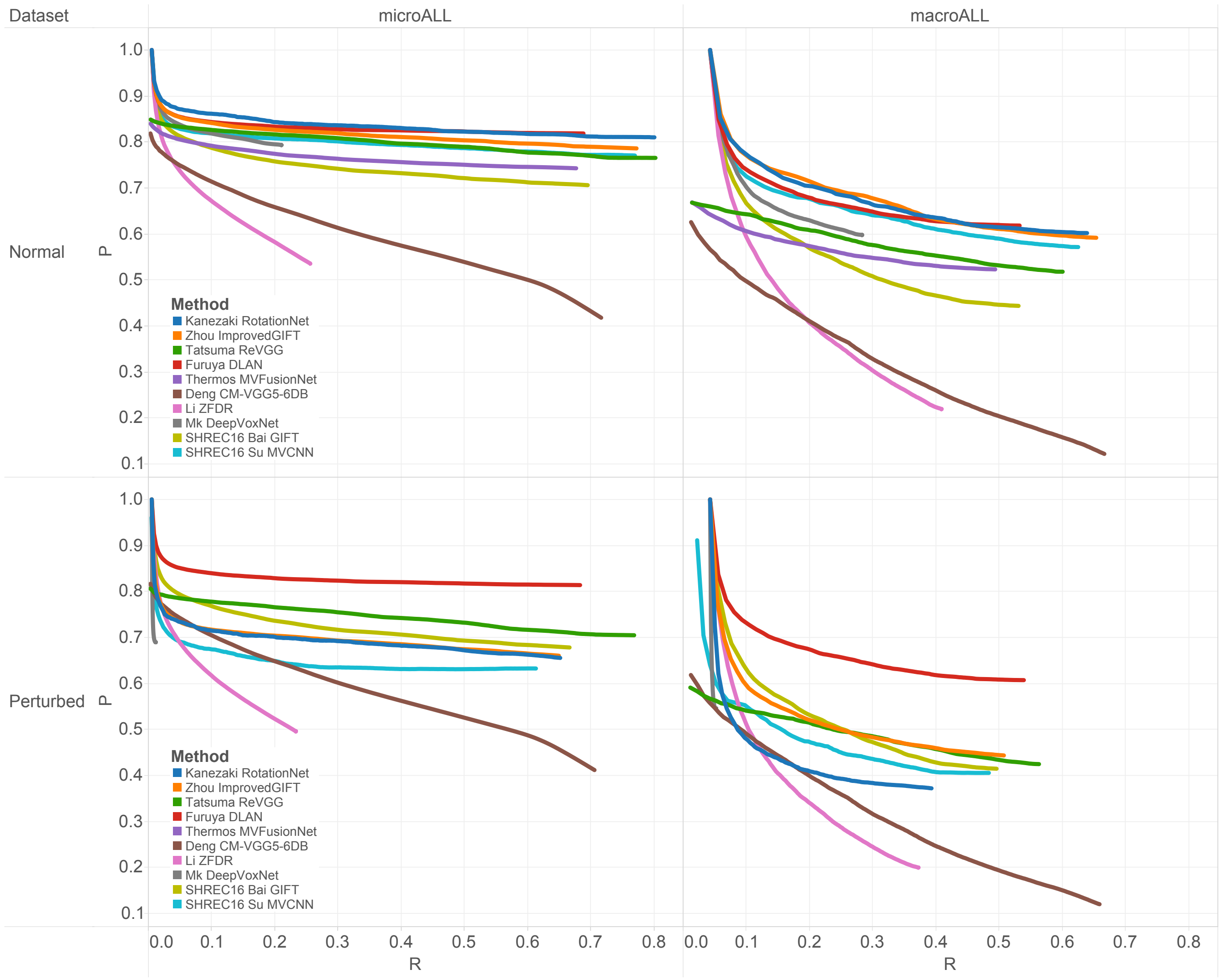

Figure 8: *Precision-recall plots for all participating teams and methods, on the normal and perturbed datasets.*

7. Conclusion

In this paper, we compared several methods for 3D model retrieval on a large-scale dataset of 3D models. Most methods use some form of neural network architecture, and outperform a more traditional non-learning based approach based on 3D shape descriptors. Though the performance on a consistently aligned version of the dataset is fairly high, there is still much space for improvement, particularly on the more challenging case of 3D model data with unknown orientations.

Compared with last year's iteration of this competition where all teams leveraged view-based methods, this year we observe a higher diversity in the input 3D representations, including point set–based and volumetric methods. The point set method of Furuya and Ohbuchi performs the best in the unaligned 3D model test scenario. In contrast the view-based method of Kanezaki performs the best on the consistently aligned 3D model data. An interesting direction for future work is to consider integrating the discriminative power of view-based approaches and the robustness to arbitrary orientation exhibited by approaches reasoning more locally with geometry.

Acknowledgments: Atsushi Tatsuma and Masaki Aono were supported by Kayamori Foundation of Informational Science Advancement, Toukai Foundation for Technology, and JSPS KAKENHI Grant Numbers 26280038, 15K15992. Bo Li is partially supported by the University of Southern Mississippi Faculty Startup Funds Award. The research done by Henry Johan is supported by the National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative.

References

[BBW15] Bai X., Bai S., Wang X.: Beyond diffusion process: Neighbor set similarity for fast re-ranking. *Information Sciences 325* (2015), 342–354. doi:10.1016/j.ins.2015.07.022. 5

[BBZ*16] Bai S., Bai X., Zhou Z., Zhang Z., Jan Latecki L.: Gift: A real-time and scalable 3D shape search engine. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 5023–5032. 4, 5

[CFG*15] Chang A. X., Funkhouser T., Guibas L., Hanrahan P., Huang Q., Li Z., Savarese S., Savva M., Song S., Su H., Xiao J., Yi L., Yu F.: *ShapeNet: An Information-Rich 3D Model Repository*. Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 1

[CSVZ14a] Chatfield K., Simonyan K., Vedaldi A., Zisserman A.: Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531* (2014). 4

[CSVZ14b] Chatfield K., Simonyan K., Vedaldi A., Zisserman A.: Return of the devil in the details: Delving deep into convolutional nets. In *In Proc. of the 25th British Machine Vision Conference* (2014), BMVC'14, pp. 1–12. 5

[DA09] Daras P., Axenopoulos A.: A compact multi-view descriptor for 3D object retrievall. *IEEE 7th International Workshop on Content-Based Multimedia Indexing* (June 2009). 7

[DAL12] Daras P., Axenopoulos A., Litos G.: Investigating the effects of multiple factors towards more accurate 3D object retrieval. *IEEE Transactions on Multimedia 14*, 2 (Apr. 2012), 374–388. 7

[DHS11] Duchi J., Hazan E., Singer Y.: Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research 12*, Jul (2011), 2121–2159. 6

[FO15] Furuya T., Ohbuchi R.: Diffusion-on-manifold aggregation of local features for shape-based 3D model retrieval. In *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval* (2015), ACM, pp. 171–178. 6

[FO16] Furuya T., Ohbuchi R.: Deep aggregation of local 3D geometric features for 3D model retrieval. In *BMVC 2016* (2016). 6

[HZRS16] He K., Zhang X., Ren S., Sun J.: Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016). 7

[KH90] Khotanzad A., Hong Y.: Invariant image recognition by Zernike moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12*, 5 (1990), 489–497. 9

[KMN16] Kanezaki A., Matsushita Y., Nishida Y.: Rotationnet: Joint learning of object classification and viewpoint estimation using unaligned 3D object dataset. *arXiv preprint arXiv:1603.06208* (2016). 3

[LBD*89] LeCun Y., Boser B., Denker J. S., Henderson D., Howard R. E., Hubbard W., Jackel L. D.: Backpropagation applied to handwritten zip code recognition. *Neural Computation 1*, 4 (Dec. 1989), 541–551. doi:10.1162/neco.1989.1.4.541. 5

[LGSX13] Lian Z., Godil A., Sun X., Xiao J.: Cm-bof: visual similarity-based 3D shape retrieval using clock matching and bag-of-features. *Machine Vision and Applications 24*, 8 (2013), 1685–1704. 8

[LJ13] Li B., Johan H.: 3D model retrieval using hybrid features and class information. *Multimedia Tools Appl. 62*, 3 (2013), 821–846. 8, 9

[LRS10] Lian Z., Rosin P. L., Sun X.: Rectilinearity of 3D meshes. *International Journal of Computer Vision (IJCV) 89*, 2-3 (2010), 130–151. 8

[Mil95] Miller G.: WordNet: a lexical database for English. *CACM* (1995). 2

[Min11] Min P.: Binvox 3D mesh voxelizer. http://www.google.com/search?q=binvox, 2011. 9

[NH10] Nair V., Hinton G. E.: Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning* (2010), ICML'10, pp. 807–814. 5

[OFCD02] Osada R., Funkhouser T., Chazelle B., Dobkin D.: Shape distributions. *ACM Transactions on Graphics (TOG) 21*, 4 (2002), 807–832. 6

[RDS*15] Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., Huang Z., Karpathy A., Khosla A., Bernstein M., Berg A. C., Fei-Fei L.: Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV) 115*, 3 (2015), 211–252. 7

[SMKL15] Su H., Maji S., Kalogerakis E., Learned-Miller E. G.: Multi-view convolutional neural networks for 3D shape recognition. 7

[SMKLM15] Su H., Maji S., Kalogerakis E., Learned-Miller E. G.: Multi-view convolutional neural networks for 3D shape recognition. In *Proceedings of International Conference on Computer Vision (ICCV)* (2015), pp. 945–953. 3

[SYS*16] Savva M., Yu F., Su H., Aono M., Chen B., Cohen-Or D., Deng W., Su H., Bai S., Bai X., Fish N., Han J., Kalogerakis E., Learned-Miller E. G., Li Y., Liao M., Maji S., Tatsuma A., Wang Y., Zhang N., Zhou Z.: Large-Scale 3D Shape Retrieval from ShapeNet Core55. In *Eurographics Workshop on 3D Object Retrieval* (2016), Ferreira A., Giachetti A., Giorgi D., (Eds.), The Eurographics Association. doi:10.2312/3dor.20161092. 8

[SZ14] Simonyan K., Zisserman A.: Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2014). 8

[TA09] Tatsuma A., Aono M.: Multi-Fourier spectra descriptor and augmentation with spectral clustering for 3D shape retrieval. *The Visual Computer 25*, 8 (2009), 785–804. doi:10.1007/s00371-008-0304-2. 5

[Tri12] Trimble: Trimble 3D warehouse. https://3dwarehouse.sketchup.com/, 2012. Accessed: 2015-12-13. 2

[VL15] Vedaldi A., Lenc K.: Matconvnet: Convolutional neural networks for matlab. In *ACM MM* (2015). 8

[Vra04] Vranic D.: *3D Model Retrieval*. PhD thesis, University of Leipzig, 2004. 9

[ZL01] Zhang D., Luo G.: A comparative study on shape retrieval using Fourier Descriptors with different shape signatures. In *Proc. of International Conference on Intelligent Multimedia and Distance Education (ICIMADE01)* (2001), pp. 1–9. 9