SHREC’17 Track: Point-Cloud Shape Retrieval of Non-Rigid Toys†


1University of York, United Kingdom, 2University of Science, VNU-HCM, Vietnam, 3University of Information Technology, VNU-HCM, Vietnam, 4John von Neumann Institute, VNU-HCM, Vietnam, 5University of Verona, Italy, 6Toyohashi University of Technology, Japan, 7Dept. Computer Science, University of Chile, Chile, 8Dept. Engineering, Pontificil Catholic University of Peru, Peru, 9ONERA, The French Aerospace Lab, France, 10Centre de Morphologie Mathématique, MINES Parisitech, France, 11University of Southern Mississippi, USA, 12Texas State University, USA, 13National Institute of Standards and Technology, USA

Abstract

In this paper, we present the results of the SHREC’17 Track: Point-Cloud Shape Retrieval of Non-Rigid Toys. The aim of this track is to create a fair benchmark to evaluate the performance of methods on the non-rigid point-cloud shape retrieval problem. The database used in this task contains 100 3D point-cloud models which are classified into 10 different categories. All point clouds were generated by scanning each one of the models in their final poses using a 3D scanner, i.e., all models have been articulated before scanned. The retrieval performance is evaluated using seven commonly-used statistics (PR-plot, NN, FT, ST, E-measure, DCG, mAP). In total, there are 8 groups and 31 submissions taking part of this contest. The evaluation results shown by this work suggest that researchers are in the right way towards shape descriptors which can capture the main characteristics of 3D models, however, more tests still need to be made, since this is the first time we compare non-rigid signatures for point-cloud shape retrieval.

Categories and Subject Descriptors (according to ACM CCS): H.3.3 [Computer Graphics]: Information Systems—Information Search and Retrieval

1. Introduction

With the rapid development of virtual reality (VR) and augmented reality (AR), especially in gaming, 3D data has become part of our everyday lives. Since the creation of 3D models is essential to these applications, we have been experiencing a large growth in the number of 3D models available on the Internet in the past years. The problem now has been organizing and retrieving these models from databases. Researchers from all over the world are trying to create shape descriptors in a way to organize this huge amount of models, making use of many mathematical tools to create discriminative and efficient signatures to describe 3D shapes. The importance of shape retrieval is evidenced by the 11 years of the Shape Retrieval Contest (SHREC).

There are two distinct areas which concern shape retrieval: The first, non-rigid shape retrieval, which deals with the problem of articulations of the same shape [LGT16, LGB11, LZZ15], and second, comprehensive shape retrieval [BBC10, LLL14, SYS16], which deals with any type of deformation, for example, scaling, stretching and even differences in topology. While comprehensive shape retrieval is more general, non-rigid shape retrieval is as important when it is necessary to carefully classify similar objects that are in distinct classes [PSR16].

Three-dimensional point clouds are the immediate result of scans of 3D objects. Although there are efficient methods to create meshes from point clouds, sometimes this task can be complex, particularly when point-cloud data present missing parts or noisy surfaces, for example, fur or hair. In this paper, we are interested in the non-rigid shape retrieval task, therefore we propose to create a non-rigid point-cloud shape retrieval benchmark (PRoNTo: Point-Cloud Shape Retrieval of Non-Rigid Toys), which was produced given the necessity of testing non-rigid shape signatures computed directly from unorganized point clouds, i.e., without any connectivity information. This is the first benchmark ever created to test, specifically, the performance of non-rigid point-cloud models.

This benchmark is important given the need to compare 3D non-rigid shapes based directly on a rough 3D scan of the object, which is a more difficult task than comparing signatures computed from well-formed 3D meshes. 3D scanners may introduce some sampling problems to the scanned models, given the difficulty of reaching all parts of the object by the scan head and given that some materials have specular properties and these can generate outliers.

Although some methods available in the literature use point sets to create their shape signatures, we have not seen these methods

† Website: https://www.cs.york.ac.uk/cvpr/pronto
‡ Track organizers. E-mail: pronto-group@york.ac.uk

© 2017 The Author(s)
being used directly to address the non-rigid point-cloud shape-retrieval problem since there are no specific point-cloud datasets available for this purpose. Instead, these methods normally use mesh vertices, which sometimes can be a very bad idea, unless vertices are very well distributed along the surface of the shape.

2. Dataset

Our dataset consists of 100 models that are derived from 10 different real objects. Each real object was scanned in 10 distinct poses by physically articulating them around their joints before being scanned. The different poses and each one of the objects used to create this database can be seen in Figures 1 and 2, respectively. After scanning all the poses we manually removed the supports used to scan the objects using MeshLab.

Objects were scanned using the Head & Face Color 3D Scanner of Cyberware. This scanner makes a 360 degrees scan around the object estimating \( x \), \( y \), and \( z \) coordinates of a vertical patch. The scanning process captures an array of digitized points and also the respective RGB colors although they are not used in this contest. The file format for the objects was chosen as the Object File Format (.off), which, in this case, contains only vertex information. We also resample models using the Poisson-Disk Sampling algorithm [CCS12] since the scan generates an arbitrary number of samples. This way, we control the sampling rate so that every model has approximately 4K points. Finally, we perform an arbitrary rotation of the model so that it is not always in the same orientation.

The point clouds acquired by our scans suffer from common scanning problems like holes and missing parts resulted from self-occlusions of the shapes, and also from noise given that some toy’s materials have specular properties. To avoid fine-tuning of parameters, we have chosen relatively similar classes to be part of this dataset, therefore it would be cumbersome to identify classes just by looking at the point clouds shapes, for example, *Teddy* and *Sheep* or *Fox* and *Dog*.

![Figure 1](image1.png)

*Figure 1: Different poses captured of the objects, showing model Monster as an example. Point clouds were coloured by \( Y \) and \( Z \) coordinates.*

3. Evaluation

The evaluation rules follow standard measures used in SHREC tracks in the past. We asked participants to submit up to 6 dis-
similarity matrices. These matrices could be the result of different algorithms or different parameter settings, at the choice of the participant. A dissimilarity matrix is the result of a shape retrieval problem which gives the difference between every model in the database. It has the size \( N \times N \), where \( N \) is the number of models of the dataset and the position \((i, j)\) in the matrix gives the difference between models \( i \) and \( j \). No class information is provided with the data, and supervised methods are not allowed in the track.

In total, seven standard quantitative evaluation measures were computed over the dissimilarity matrices submitted by the participants to test the retrieval accuracy of the algorithms: Precision-and-Recall (PR) curve, mean Average Precision (mAP), E-Measure (E), Discounted Cumulative Gain (DCG), Nearest Neighbor (NN), First-Tier (FT) and Second-Tier (ST).

4. Participants

During this contest, we had 8 groups taking part in the SHREC’17 PRoNTo contest and we received in total 31 dissimilarity-matrix submissions, as detailed below:

2. **BoW-RoPS-1, BoW-RoPS-2, BoW-RoPS-DMF-3, BoW-RoPS-DMF-4, BoW-RoPS-DMF-5** and **BoW-RoPS-DMF-6** submitted by Minh-Triet Tran, Viet-Khoi Pham, Hai-Dang Nguyen and Vinh-Tiep Nguyen. Other team members: Thuyen V. Phan, Bao Truong, Quang-Thanh Tran, Tu V. Ninh, Tu-Khiem Le, Dat-Thanh N. Tran, Ngoc-Minh Bui, Trong-Le Do, Minh N. Do and Anh-Duc Duong.
3. **POHAPT and BPHAPT** submitted by Andrea Giachetti.
4. **CDSF** submitted by Atsushi Tatsuma and Masaki Aono.
5. **SQFD(HKS), SQFD(WKS), SQFD(SIHKS) and SQFD(WKS-SIHKS)** submitted by Benjamin Bustos and Ivan Sipiran.
6. **SnapNet** submitted by Bertrand Le Saux, Nicolas Audebert and Alexandre Boulch.
7. **AlphaVol1, AlphaVol2, AlphaVol3 and AlphaVol4** submitted by Santiago Velasco-Forero.
8. **m3DSH-1, m3DSH-2, m3DSH-3, m3DSH-4, m3DSH-5 and m3DSH-6** submitted by Bo Li, Yijuan Lu and Afzal Godil.

5. Methods

In the next sections, we detail all the participant methods that have successfully competed in the PRoNTo dataset contest. Experimental settings of each method are displayed at the end of each section.

5.1. Spectral Descriptors for Point Clouds, by Frederico Limberger and Richard Wilson

The key idea of this method is to test spectral descriptors computed directly from point clouds using different formulations for the computation of the Laplace-Beltrami operator (LBO). We test three different methods for computing the LBO: the Mesh-Free Laplace operator (MFLO), the Point-Cloud Laplace (PCDLaplace) [BSW09] and the Graph Laplacian (GL).

Our framework is as follows. We first compute the eigendecomposition of the different LBO methods. Then, we compute local descriptors. We encode these local features using state-of-the-art encoding schemes (FV and SV). Furthermore, we compute the differences between shape signatures using Efficient Manifold Ranking. We now detail each part of our framework.

**Laplace-Beltrami operator**: The LBO is a linear operator defined as the divergence of the gradient, taking functions into functions over the 2D manifold \( \mathcal{M} \)

\[
\Delta_{\mathcal{M}} f = -\nabla \cdot \nabla_{\mathcal{M}} f
\]  

given that \( f \) is a twice-differentiable real-valued function. Although we compute the LBO using three different methods, all these use the same parameters that are equivalent in each approach. The eigendecomposition of the LBO results in their eigenvalues and eigenfunctions, which are commonly known as the shape spectrum and these are further used to compute a local descriptor.

**Local Descriptor**: After computing the shape spectrum, we compute the Improved Wave Kernel Signature (IWKS) [LW15] which is a local spectral descriptor based on the Schrodinger equation and it is governed by the wave function \( \psi(x,t) \).

\[
i\Delta_{\mathcal{M}} \psi(x,t) = \frac{\partial \psi}{\partial t}(x,t),
\]

The IWKS is an improved version of the WKS [ASC11]. It has a different weighting filter of the shape spectrum which captures, at the same time, the major structure of the object and its fine details, therefore being more informative than the WKS.

**Encoding**: For computing the encoding of local descriptors into global signatures for shape retrieval, we use state-of-the-art encodings: Fisher Vector (FV) [PD07] and Super Vector (SV) [ZYZH10]. These methods are based on the differences between descriptors and probabilistic distribution functions, which we approximate by Gaussian Mixture models. More details about these encodings can be found in [LW15].

**Distances between signatures**: Distances between signatures are computed using Efficient Manifold Ranking (EMR) [XBC11], which accelerates the classic Manifold Ranking [ZWG04]. EMR has similar evaluation performance to MR, however, it has much lower computation times when used in large databases.

**Experimental settings**: We compute the first 100 eigenvalues and eigenfunctions of the respective LBO using 15 nearest neighbours to compute the proximity graph for all methods. In PCDLaplace, we use the following additional parameters: \( \text{hype} = \text{ddr}; \text{hs} = 2; \rho = 2 \). For the computation of the local descriptor, we use \( \text{iwksvar} = 5 \). For computing the Gaussian dictionary, we use the first 29 models of the database to create GMMs with 38 components for each signature frequency. For computing EMR we use 99 landmarks and we use \( k \)-means as the landmark selection method. The number of landmarks is usually chosen as a slightly smaller number than the number of models in total. For one model in the database, it takes approximately 15 seconds to compute the MFLO-IWKS, 8 seconds to compute the GL-IWKS and 11 seconds to compute the PCDL-IWKS. To compute the entire dissimilarity matrix it takes approximately 43 seconds with the FV and 56 seconds.
5.2. Bag-of-Words Framework for 3D Object Retrieval, by Minh-Triet Tran, Viet-Khoi Pham, Hai-Dang Nguyen and Vinh-Tiep Nguyen

We develop our framework for 3D object retrieval based on Bag-of-Words scheme for visual object retrieval [SZ03]. BoW method originated from text retrieval domain and is shown to be successfully applied on large-scale image [NNT+15] and 3D object retrieval [PSA+16]. Figure 3 illustrates the main components of our framework.

**Preprocessing:** Each point cloud is normalized into a unit cube and densified to reduce significant difference in the density between different parts.

**Feature detector:** For each model, we uniformly take random samples of $5\% \leq p_{Sampling} \leq 50\%$.

**Feature descriptor:** We describe the characteristics of the point cloud in a sphere with supporting radius $r$ surrounding a selected vertex. We propose our point-cloud-based descriptor inspired by the idea of RoPS [GSB+13] to calculate the descriptor directly from a point cloud (without reconstructing faces). We first estimate the eigenvectors of the point cloud within a supporting radius $r$ of a selected vertex, then transform the point cloud to achieve rotation invariant for the descriptor, and finally calculate the descriptor. We consider the supporting radius $r$ from 0.01 to 0.1.

**Codebook:** All features extracted from the models are used to build a codebook with size relatively equal to 10\% of the total number of features in the corpus, using Approximate K-Means.

**Quantization:** To reduce quantization error, we use soft-assignment [PCI’08] with 3 nearest neighbors.

**Distance measure metric:** instead of using a symmetric distance, we use L1, asymmetric distance measurement [ZJS13], to evaluate the dissimilarity of each pair of objects.

Our first two runs (1 and 2) are results of our BoW framework using random sampling with $p_{Sampling} = 45\%$ and codebook size of 18000. The radius of point-cloud based RoPS for run 1 and 2 are $r = 0.04$ and 0.05, respectively.

Each main component of our BoW framework is deployed on a different server. Codebook training module using Python 2.7 is deployed on Ubuntu 14.04 with 2.4 GHz Intel Xeon CPU E5-2620 v3, 64 GB RAM. It takes 30 minutes to create a codebook with 18,000 visual words from 180,000 features. 3D feature extraction and description module, written in C++, runs on Ubuntu 14.04, 2GHz Intel Xeon CPU E5-2620, 1GB RAM.

The retrieval process in Matlab R2012b with feature quantization and calculating the dissimilarity matrix is performed on Windows Server 2008 R2, 2.2GHz Intel Xeon CPU E5-2660, 12 GB RAM. The average time to calculate features of a model is 1-2 seconds and it takes on average 0.02 seconds to compare an object against all 100 objects.

5.2.1. Distance Matrix Fusion

With each setting for our BoW framework, we get a different retrieval model. We propose a simple method to linearly combine $k$ distance matrices $D_1, D_2, \ldots, D_k$ with different coefficients into a new distance matrix:

$$D_{Fusion} = w_1D_1 + w_2D_2 + \ldots + w_kD_k$$

Our objective is to take advantage of different retrieval models obtained from our framework with the expectation to increase the performance of the retrieval process.

We limit the number of seeds $k$ of 2 or 3, and the value of a coefficient $w_i$ is from 0 to 1, step 0.2. Our last four runs are combinations of Run1 and Run2 with different values of $w_1$ and $w_2$: Run3: $w_1 = 0.8$ and $w_2 = 0.6$; Run4: $w_1 = 1.0$ and $w_2 = 0.8$; Run5: $w_1 = 0.6$ and $w_2 = 0.2$; and Run6: $w_1 = 1.0$ and $w_2 = 0.4$. Experimental results show that the fusion runs can even yield better performance in retrieval than the two original seeds.

5.3. Simple meshing and Histogram of Area Projection Transform, by Andrea Giachetti

The method is based on simple automatic point cloud meshing followed by the estimation of the Histogram of Area Projection Transform descriptor [GL12]. Shapes are represented through a set of $N$ voxelized maps encoding the area projected along the inner normal direction at sampled distances $R_i, i = 1..N$ in a spherical neighborhood of radius $\sigma$ around each voxel center location $x$. Values at different radii are weighted in order to have a scale-invariant behavior. Histograms of MAPT computed inside the objects are quantized in 12 bins and evaluated at 12 equally spaced radii values ranging from 3 to 39mm, with $\sigma$ always taken as half the radius. Histograms computed at the different radii considered are concatenated creating an unique descriptor. Dissimilarity matrices are generated by measuring the histogram distances with the Jeffrey divergence.

This descriptor is robust against pose variation and inaccuracy due to holes, especially if histograms are estimated inside the shape only. For this reason we applied the HAPT estimation using the code publicly available at the web site www.andreagiachetti.it on closed meshes estimated on the original point clouds with two different procedures implemented as simple Meshlab [CCC’08] scripts.

We submitted matrices corresponding to each of these procedures.
Poisson reconstruction: In the first run, we just applied Poisson reconstruction [KBH]. Points’ normals have been estimated on a 12 neighbors range, and the octree depth has been set equal to 9.

Ball Pivoting and Poisson: In the second run we first smoothed the point set using Moving Least squares, then we applied the ball pivoting method [BMR’99] to extract an open mesh. The mesh has been refined with triangle splitting and normals have been recomputed on the basis of the meshing. From mesh and normals obtained, a closed watertight mesh has been finally obtained with the Poisson reconstruction method.

Note that meshing would be not mandatory for the application of the method, as a point cloud implementation of the descriptor would be quite simple. However, we believe that Poisson meshing provides in general a better estimate of the inner part of the mesh and is effective in reconstructing missing parts in a reasonable way.

Meshlab scripts runs in less than one second per model and HAPT estimation takes 10 seconds on average. Histogram distance estimation time, performed in Matlab, is negligible in comparison.

5.4. Covariance Descriptor with Statistics of Point Features, by Atsushi Tatsuma and Masaki Aono

For non-rigid human 3D model retrieval, we previously proposed the local feature extraction method [PSR’16] that calculates the histogram, mean, and covariance of geometric point features. In this track, we further calculate the skewness and kurtosis of geometric point features to obtain more discriminative local feature. 3D point-cloud object finally is represented with the covariance of the local features consisting of the histogram, mean, covariance, skewness, and kurtosis of geometric point features. We call our approach the Covariance Descriptor with Statistics of Point Features (CDSPF).

The overview of our approach is illustrated in Figure 4. We first calculate 4D point geometric feature \( \mathbf{f} = [f_1, f_2, f_3, f_4] \) proposed in [WHHH03]. The geometric feature is computed for every pair of points \( \mathbf{p}_a \) and \( \mathbf{p}_b \) in the point’s k-neighborhood:

\[
\begin{align*}
f_1 &= \tan^{-1}(\mathbf{w} \cdot \mathbf{n}_b/|\mathbf{u} \cdot \mathbf{n}_a|), \\
f_2 &= \mathbf{v} \cdot \mathbf{n}_b, \\
f_3 &= \mathbf{u} \cdot (\mathbf{p}_b - \mathbf{p}_a/d), \\
f_4 &= d,
\end{align*}
\]

where the normal vectors of \( \mathbf{p}_a \) and \( \mathbf{p}_b \) are \( \mathbf{n}_a \) and \( \mathbf{n}_b \), \( \mathbf{u} = \mathbf{n}_a \), \( \mathbf{v} = (\mathbf{p}_b - \mathbf{p}_a) \times \mathbf{u}/|[(\mathbf{p}_b - \mathbf{p}_a) \times \mathbf{u}]| \), \( \mathbf{w} = \mathbf{u} \times \mathbf{v} \), and \( d = ||\mathbf{p}_b - \mathbf{p}_a|| \).

Next, we collect the point features in a 16-bin histogram \( h \). The index of histogram bin \( h \) is defined by the following formula:

\[
h = \sum_{i=1}^{4} 2^{i-1} s(t, f_i),
\]

where \( s(t, f) \) is a threshold function defined as 0 if \( f < t \) and 1 otherwise. The threshold value used for \( f_1, f_2 \) and \( f_3 \) is 0, while the threshold value for \( f_4 \) is the average value of \( f_4 \) in the \( k \)-neighborhood.

Furthermore, we calculate the mean, covariance matrix, skewness, and kurtosis of the point features. Let \( f_1, f_2, \ldots, f_N \) be the point features of size \( N \). The mean \( \mu \), covariance matrix \( C \), skewness \( s \), and kurtosis \( k \) are calculated as follows [Mar70]:

\[
\begin{align*}
\mu &= \frac{1}{N} \sum_{i=1}^{N} f_i, \\
C &= \frac{1}{N} \sum_{i=1}^{N} (f_i - \mu)(f_i - \mu)\top, \\
s &= \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} \{(f_i - \mu)\top C^{-1}(f_j - \mu)\}^3, \\
k &= \frac{1}{N} \sum_{i=1}^{N} \{(f_i - \mu)\top C^{-1}(f_i - \mu)\}^2.
\end{align*}
\]

Since the covariance matrix \( C \) lies on the Riemannian manifold of symmetric positive semi-definite matrices, we map the covariance matrix onto a point in the Euclidean space by using Pencnec et al.’s method [PFA06].

We finally obtain the local feature by concatenating the histogram, mean, covariance, skewness, and kurtosis of the point features. The local feature is normalized with the signed square root and \( \ell_2 \) normalization [JC12]. To compare 3D point-cloud objects, we integrated the set of local features into a feature vector with the covariance descriptor approach [TPM06].

Since 3D point-cloud objects in the dataset do not have normal vector information, we used the Point Cloud Library [RC11] for estimating normal vector of each point. Moreover, we set the size of the neighborhood \( k \) to 30. We employ the Euclidean distance for the dissimilarity between two feature vectors.

The method was implemented in C++. Experiments were carried out under Debian Linux 8.7 on a CPU 3.4GHz Intel Core i7-6800K and 128GB DDR4 memory. The average time to calculate the shape descriptor for a 3D model is about 1.46 seconds and it takes approximately 10 seconds to compute the dissimilarity matrix.

5.5. Signature Quadratic Form Distance on Spectral Descriptors, by Ivan Sipiran and Benjamin Bustos

Our method combines the flexibility of the Signature Quadratic Form Distance (SQFD) [BUS09] with the robustness of intrinsic spectral descriptors. On the one hand, the SQFD distance has proven to be effective in multimedia domains where objects are represented as a collection of local descriptors [SLBS16]. On the other hand, intrinsic descriptors are useful to keep robustness to non-rigid transformations. Our proposal consists of representing...
the input 3D point cloud as a set of local descriptors which will be compared through the use of the SQFD distance.

Let $P$ be a 3D point cloud. The first step of our method is to compute a set of local descriptors on $P$. The spectral descriptors depends on the computation of the Laplace-Beltrami operator on the point cloud. So a pre-processing step is needed to guarantee a proper computation of this operator. The pre-processing is performed as follows

- **Normal computation.** We compute a normal for each point in the point cloud. For a given point, we get the 20 nearest neighbors and compute the less dominant direction of the neighborhood.
- **Poisson reconstruction.** We reconstruct the surface for the point cloud using the screened Poisson reconstruction method [KH13]. We set the octree depth to eight and the depth for the Laplacian solver to six. The output is a Manifold triangle mesh that preserves the structure of the original point cloud.

Let $M$ be the obtained mesh. We compute a local descriptor for each vertex in the mesh. We denote the set of local descriptors of the mesh $M$ as $FM$. The challenge now is how to compare two objects through their collections of local descriptors. The approach to use the SQFD distance establishes that we need to compute a more compact representation called signature. Let suppose the existence of a local clustering on $FM$ that groups similar local descriptors such that the number of clusters is $n$ and $FM = C_1 \cup C_2 \cup \ldots \cup C_n$. The signature is defined as $SM = \{ (c_i^M, w_i^M), i = 1, \ldots , n \}$, where $c_i^M = \frac{\sum_{v \in C_i} d(v)}{|C_i|}$ and $w_i^M = \frac{|C_i|}{|FM|}$. Each element in the signature contains the average descriptor in the cluster ($c_i^M$) and a weight ($w_i^M$) to quantify how representative is the cluster in the collection of local descriptors.

Note that the local clustering is a key ingredient of the computation of the signatures. Here we briefly give some details about the clustering. We use an adaptive clustering method that searches configurations. Here, we describe the parameters used in each run

- **SQFD(WKS).** We use the normalized Wave Kernel Signature [ASC11] as local descriptor. The parameters for local clustering are $\alpha = 0.2$, $\beta = 0.4$, $N_m = 30$.
- **SQFD(HKS).** We use the normalized Heat Kernel Signature [SOG09] as local descriptor. The parameters for local clustering are $\alpha = 0.1$, $\beta = 0.2$, $N_m = 20$.
- **SQFD(SIHKS).** We use the Scale-invariant Heat Kernel Signature [BK10] as local descriptor. The parameters for local clustering are $\alpha = 0.1$, $\beta = 0.2$, $N_m = 20$.
- **SQFD(WKS-SIHKS).** We use a distance function as a combination of distances. For every pair of objects, we compute the sum of the distances obtained with SQFD(WKS) and SQFD(SIHKS).
- **SQFD(HKS-WKS-SIHKS).** We use the combination of three distances. We use the weighted sum of distances SQFD(HKS), SQFD(WKS) and SQFD(SIHKS). The weights are 0.15, 0.15 and 0.7, respectively.

We implemented our method in Matlab under Windows 10 on a PC CPU i7 3.6 GHz, 12GB RAM. The average time to compute the shape signature for a 3D model is about 5 seconds and it takes approximately 0.3 seconds to compare a pair of signatures.

### 5.6. SnapNet for Dissimilarity Computation, by Alexandre Boulch, Bertrand Le Saux and Nicolas Audebert

The objective of this approach is to learn a classifier, in an unsupervised way, that will produce similar outputs for the same shapes with different poses. As we do not know the ground truth, i.e. the model used to generate the pose, we will train the classifier as if each pose was a different class. It is a 100-class problem.

**Training dataset:** The training dataset is generated by taking snapshots around the 3D model [Gra14]. In order to create visually consistent snapshots, we mesh the point cloud using [MRB09]. The snapshots are 3-channel images. The first channel encodes the distance to the camera (i.e. depth map), the second is the normal orientation to the camera direction and the third channel is an estimation of the local noise in the point cloud (ratio of eigenvalues of the neighborhood covariance matrix). An example of such a snapshot is presented in Figure 5 (left).

**CNN training:** The CNN we train is a VGG16 [SZ14] with a last fully connected layer with 100 outputs. We initialize the weights with the model trained on the ILSVRC12 contest. We then finetune the network using a step learning rate policy.

**Distance computation:** The classifier is then applied to images and produces images classification vectors $V_{im}$. For each model we compute a prediction vector $VM$ based on the images:

$$V_M = \frac{\sum_{im \in M} V_{im}}{||\sum_{im \in M} V_{im}||_2}$$

Note that to compute the transformation, we need to choose the value of parameter $\alpha$ and the ground distance for descriptors. In all our experiments, we use $\alpha = 0.9$ and $L^2$ as ground distance. More details about the computation of signatures and the SQFD distance can be found in [SLBS16].

**Experimental Settings:** We provide five runs using different configurations. Here, we describe the parameters used in each run

- **SQFD(WKS).** We use the normalized Wave Kernel Signature [ASC11] as local descriptor. The parameters for local clustering are $\lambda = 0.2$, $\beta = 0.4$, $N_m = 30$.
- **SQFD(HKS).** We use the normalized Heat Kernel Signature [SOG09] as local descriptor. The parameters for local clustering are $\lambda = 0.1$, $\beta = 0.2$, $N_m = 20$.
- **SQFD(SIHKS).** We use the Scale-invariant Heat Kernel Signature [BK10] as local descriptor. The parameters for local clustering are $\lambda = 0.1$, $\beta = 0.2$, $N_m = 20$.
- **SQFD(WKS-SIHKS).** We use a distance function as a combination of distances. For every pair of objects, we compute the sum of the distances obtained with SQFD(WKS) and SQFD(SIHKS).
- **SQFD(HKS-WKS-SIHKS).** We use the combination of three distances. We use the weighted sum of distances SQFD(HKS), SQFD(WKS) and SQFD(SIHKS). The weights are 0.15, 0.15 and 0.7, respectively.

We implemented our method in Matlab under Windows 10 on a PC CPU i7 3.6 GHz, 12GB RAM. The average time to compute the shape signature for a 3D model is about 5 seconds and it takes approximately 0.3 seconds to compare a pair of signatures.
The distance matrix $X$ contains the pairwise $\ell_2$ distances between the $V_M$. Each line is then normalized using a soft max:

$$X_{i,j} = \frac{\exp(X_{ij})}{\sum\exp(X_{ij})}$$ (16)

Matrix $X$ is not symmetrical. We finally define the symmetrical distance matrix as $D$ such that $D = X^T X$. The values of $D$ are clipped according to the 5$^{th}$ and 95$^{th}$ percentiles and then re-scaled in $[0, 1]$. The resulting matrix is presented on Figure 5 (right).

The method was implemented in Python and C++, using the deep learning framework Pytorch. We ran the experiments on Linux, CPU Intel Xeon(R) E5-1620 3.50GHz. The training part was operated on a NVidia Titan X Mawell GPU and the test part (predictions) on a NVidia GTX 1070. Generating the snapshots took around 10 seconds per model. The training took around 8 hours. The prediction vectors were generated in 2 seconds per model and the dissimilarity matrix is computed in less than 10s.

### 5.7. Alpha-shapes volume curve descriptor, by Santiago Velasco-Forero

Given $S$ be the set of finite set of points in $\mathbb{R}^3$, we have computed a set of three-dimensional alpha-shapes of radius $r$, proposed by Edelsbrunner [EM94], and denoted by $\alpha_r(S)$. Rather than finding an optimal fixed value, we focus on a range of values for the scale parameter $r$, and our descriptor computes the volume of each $\alpha_r(S)$. Thus, the similarity of two shapes is then computed by the distance of their alpha-shapes volume curve in Euclidean norm. An example of different $\alpha_r(S)$ by varying $r$ is illustrated in Figure 6.

The values of parameter ($r$) used in the different submission are:

- **AlphaVol 1**: $r \in [0.02, 0.045, 0.07]$  
- **AlphaVol 2**: $r \in [0.02, 0.045, 0.07, 0.095]$  
- **AlphaVol 3**: $r \in [0.02, 0.045, 0.07, 0.095, 0.12]$  
- **AlphaVol 4**: $r \in [0.02, 0.045, 0.07, 0.095, 0.12, 0.145]$  

The 100 non-rigid point cloud toy models contain only 3D points to represent ten different poses for each of the ten toys. We can first reconstruct a 3D surface for each 3D point cloud such as to extract our previously developed 3D surface-based non-rigid shape descriptors. However, considering retrieval efficiency, the raw point cloud data is directly used for 3D shape descriptor extraction for shape comparison. For simplicity, we chose 3D Shape Histogram (3DSh) [AKKS99]. The original 3DSh descriptor uniformly partitions the surrounding space of a 3D shape into a set of shells, sectors or spider web bins and counts the percentage of the surface sampling points falling in each bin to form a histogram as the 3DSh descriptor. Rather than like the original 3DSh descriptor which divides the space uniformly, to increase its descriptiveness we developed a modified variation of 3DSh descriptor, that is m3DSh, by dividing the 3D space occupied by a 3D shape in a non-uniform way.

Figure 7 illustrates the overview of the feature extraction process: Principal Component Analysis (PCA) [Jo02]-based 3D model normalization, and extraction of a modified 3D Shape Histogram descriptor m3DSh. The details of our algorithm are described as follows.

1) **PCA-based 3D shape normalization**: PCA-based 3D shape normalization: We utilize PCA [Jo02] for 3D model normalization (scaling, translation and rotation). After this normalization, each 3D point cloud is scaled to be enclosed in the same bounding sphere with a radius of 1. centered at the origin, and rotated to have as close-as-possible consistent orientations for different poses of the same toy object. These are important for the following m3DSh descriptor extraction.
2) Modified 3D Shape Histogram descriptor m3DSH extraction: The original 3DSH descriptor only has two degrees of freedom (DOF), which are numbers of sectors and number of shells. As we know, a 3D space has three DOFs according to its spherical coordinate representation (ρ, φ, θ). The reason is that 3DSH uniformly divides φ and θ into the same number of bins, which forms a certain number of sectors. Here, in order to improve its flexibility and descriptiveness, we individually divide φ and θ into a number of vertical bins (V) and a number of horizontal bins (H), since the two dimensions do not have the same importance. We denote the number of radius bins for ρ as R. In the experiments, we tested two different combinations of V, H and R: V=5, H=8, R=6; and V=12, H=12, R=6.

3) Quadratic form shape descriptor distance computation and ranking: Similar as [AKKS99], we adopt the quadratic form distance to measure the distance between the extracted histogram features of the 3D models. It has a parameter σ to control the similarity degree of the resulting distance to Euclidean distance. In our experiments, we tested three σ [AKKS99] values: σ=1, σ=5, σ=10. Finally, we rank 3D models according to the computed shape descriptor distances in an ascending order.

We implemented our method in Java and carried out experiments under Windows 7 on a personal laptop with a 2.70 GHz Intel Core i7 CPU, 16GB memory. The average time to calculate the shape descriptor for a 3D model is about 0.03 seconds and it takes approximately 0.14 seconds to compute the dissimilarity matrix.

6. Results
In this section, we compare the results of all participant’s runs. In total, we had 8 groups participating and we received 31 dissimilarity matrices. The retrieval scores computed from these matrices represent the overall retrieval performance of each method, i.e., how well they perform on retrieving all models from the same class when querying every model in the database. The quantitative statistics used to measure the performance of methods are: NN, FT, ST, E-measure, DCG, mAP and the Precision-and-Recall plot. For the meaning of each measure, we refer the reader to [SMKF04].

Table 1 shows the method performances of all 31 runs. It is worth pointing out that some methods perform quite well on this database. By analysing particularly DCG, which is a very good and stable measure for evaluating shape retrieval methods [LZC*15], we can see that three methods have DCG greater than 0.900 (BoW-RoPS-DMF-3, BPHAPT and MFLO-FV-IWKS). Surprisingly, Tran’s methods have DCG values greater than 0.990. The method clearly sees that three methods have DCG greater than 0.900 (BoW-RoPS-DMF-3, BPHAPT and MFLO-FV-IWKS). Surprisingly, Tran’s method is in the first place and uses local features. Clearly, our first guess was that local features would be more popular to represent non-rigid shapes, as evidenced by [LZC*15]. Our guess was based on the fact that ideally local features should be more similar than global features because same-class shapes were captured originally from the same 3D object, and locally they should be more similar than globally. For example, while a shape can be in a totally different pose, locally only joint regions are deformed. However, we also need to consider local noise in the formula, which does not affect global methods in the same level.

Tran’s method is in the first place and uses local features. Clearly, in the second place is Giachetti’s method, which is based on global descriptors, one from a global descriptor (m3DSH-3) and other from a local descriptor SQFD(WKS). Interestingly, two methods use quadratic form distance to compute dissimilarities between descriptors, one from a global descriptor (m3DSH-3) and other from a local descriptor SQFD(WKS).

Even though no training set was available in this track, Bouch’s method uses a Convolutional Neural Network by employing an unsupervised learning architecture where every model is considered belonging to a different class. On the other hand, more methods also adopt unsupervised learning algorithms to create dictionaries using the Bag of Words encoding paradigm (BoW-RoPS-DMF-3 and MFLO-FV-IWKS) being these ranked first and third on this contest, respectively, and showing that the BoW model is a good way of representing local features. Furthermore, two other methods use histogram encoding (vector quantization) to create a unique descriptor for each point cloud (BPHAPT and CDSPF).
We also observed a couple of new other ideas applied to PRoNTo dataset. For instance, Velasco uses alpha-shapes to represent point clouds; by varying the alpha-shape radius he compares models given their alpha-volume shape. Limberger’s method uses a new formulation to compute the Laplace-Beltrami operator of point clouds, which leads to better results than the standard Graph Laplacian. Tatsuma computes additional statistics of point features in addition to the geometric feature proposed by [WHH03]. Two groups use matrix-fusion methods with different weights to improve the performance of their methods (Tran and Sipiran), however, these methods did not show a substantial improvement from the performance of the original descriptors.

After analysing retrieval statistics of the methods, we concluded that the easiest pose to retrieve was definitely Default, followed by Hands Front and Walking. The most difficult pose to retrieve was Tilted followed by Straight, which are the most different from the others in respect to topology. Regarding classes, the easiest class to retrieve by the participant methods was Tiger, followed by Bear, while the most difficult was Sheep, followed by Rat and Teddy.

For more information about this track, please refer to the official website [LW17] where the database, the corresponding evaluation code and classification file are available for academic use.

7. Conclusion

In this paper, we have created a non-rigid point cloud dataset which is derived from real toy objects. In the beginning, we discussed the importance of this data to future researchers. Then, we explained the dataset characteristics and we showed how the evaluation was carried out. Afterwards, we introduced each one of the 8 groups and their methods which competed on this track. In the end, we presented quantitative measures of the 31 runs submitted by the participants and analysed their results.

The interest in non-rigid shape retrieval is overwhelming and evident by the previous SHREC tracks. This track was not different. It has attracted a large number of participants (8 groups and 31 runs) given that it is the first time that a non-rigid point-cloud dataset is used in the SHREC contest. We believe that the organization of this track is just a beginning and it will encourage other researchers to further investigate this important research topic.

Several research directions in point-cloud shape retrieval can be pursued from this work and are listed as follows: (1) Create a larger dataset which contains more types of objects (not only human shaped toys) to better evaluate shape signatures. (2) Create more discriminative local or global signatures for 3D point clouds. (3) Employ state-of-the-art Deep Learning techniques which do not depend on large training datasets.

ACKNOWLEDGMENTS

Frederico A. Limberger was supported by CAPES Brazil (Grant No.: 11892-13-7). Atsushi Tatsuma and Masaki Aono were supported by Kayamori Foundation of Informational Science Advancement, Toukai Foundation for Technology, and JSPS KAKENHI (Grant No.: 26280038 and 15K15992).

References


