Eurographics Workshop on 3D Object Retrieval (2010)
I. Pratikakis, M. Spagnuolo, T. Theoharis, and R. Veltkamp (Editors)

SHREC'10 Track: Large Scale Retrieval

Remco C. Veltkamp[†,1] Geert-Jan Giezeman[†,1] Hannah Bast[2] Thomas Baumbach[3] Takahiko Furuya[4]
Joachim Giesen[3] Afzal Godil[5] Zhouhui Lian[5,6] Ryutarou Ohbuchi[4] Waqar Saleem[3]

[1]Utrecht University, The Netherlands
[2]Albert-Ludwigs-Universität Freiburg, Germany
[3]Friedrich-Schiller-Universität Jena, Germany
[4]University of Yamanashi, Yamanashi-ken, Japan
[5]National Institute of Standards and Technology, Gaithersburg, USA
[6]Beihang University, Beijing, P.R. China

Abstract

This paper is a report on the 3D Shape Retrieval Constest 2010 (SHREC'10) track on large scale retrieval. This benchmark allows evaluating how wel retrieval algorithms scale up to large collections of 3D models. The task was to perform 40 queries in a dataset of 10000 shapes. We describe the methods used and discuss the results and signifiance analysis.

Categories and Subject Descriptors (according to ACM CCS): I.5.4 [Pattern Recognition]: Applications—Computer vision; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—Clustering; Experimentation; Performance

1. Introduction

In this paper we describe the 3D retrieval experiments done in the large scale retrieval track of SCHREC 2010, the 3D Shape Retrieval Contest held in conjunction with the third Eurographics Workshop on 3D Object Retrieval, see `http://www.aimatshape.net/event/SHREC`. This benchmark allows evaluating how wel retrieval algorithms behave on large collections of shapes. Section 2 describes the task that was set for the participants. The three following sections describe the methods used by the participants to solve the task. Section 6 contains a discussion of the results.

2. The task

Participants of the contest were presented a dataset of 10000 3D models in the ply format. The files contain only geometrical information, no textures or colors. 493 files contain real models, made with modelling software for real use. The other files contain random generated content. The idea behind adding the random content was to discourage manual inspection of the models. The real models were classified by human inspection in 54 classes. Similarity in shape was taken as the classification criterion. We selected 40 queries from 36 classes. We did not select queries from all classes, because there are quite a few residual categories, like *other animals* or *other electrical equipment* where the models hardly resemble each other. See `http://give-lab.cs.uu.nl/SHREC/ULS3SRB/2010/` for the Utrecht Large Scale 3D Shape Retrieval Benchmark.

3. The methods of Zhouhui Lian and Afzal Godil

The visual similarity based method has been widely considered as the most discriminative approach in the field of 3D content-based object retrieval. We ran three such kind of methods, denoted as LiCm, LiGsmd and LiVlgd, respec-

[†] Organizer of this SHREC track. For any information about the benchmark, contact Remco.Veltkamp@cs.uu.nl

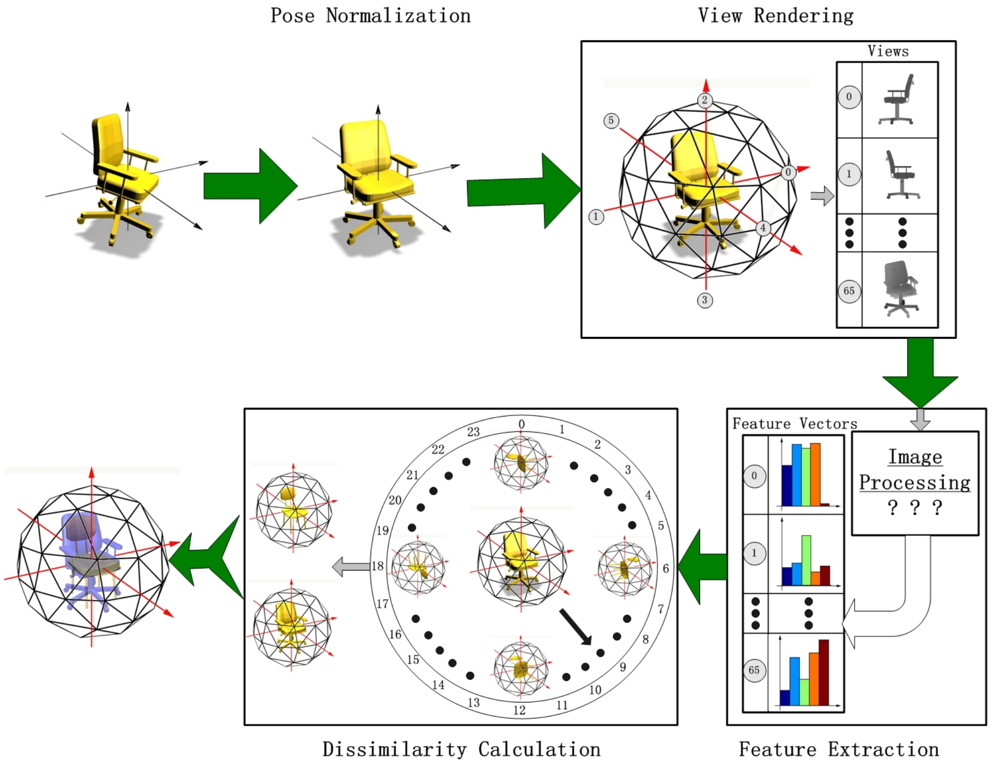

Figure 1: *The visual similarity based framework.*

tively, on the Large Scale Benchmark of SHREC'10. Basically, these three algorithms are similar due to the fact that, they all utilize a particular visual similarity based framework, and the only difference between them is how to describe the depth-buffer views captured around a 3D object. More specifically, CM-BOF uses a local feature based shape descriptor to represent a view as a histogram, and GSMD describes the view by a global feature vector, while VLGD utilizes a linear combination of above mentioned 2D shape descriptors.

3.1. A Visual Similarity based Framework

As demonstrated in Figure 1, our visual similarity based 3D shape retrieval framework is implemented subsequently in four steps:

1. ***Pose Normalization:*** Given a 3D object, we first translate the center of its mass to the origin of canonical coordinate frame and then scale the maximum polar distance of the points on the surface to one. Rotation invariance is achieved by applying the PCA technique to find the principal axes and align them to the canonical coordinate frame. Note that, we only employ the information of eigenvectors to fix the positions of three principal axes, namely, the direction of each axis is still undecided and the x-axis, y-axis, z-axis of the canonical coordinate system can be located in all three axes. That means 24 different orientations are still plausible for the normalized 3D object, or rather, 24 matching operations should be carried out when comparing two models. It should also be pointed out that, the exact values of the surface moments used in our PCA-based pose normalization are calculated via the explicit formulae introduced by [ST01].
2. ***View Rendering:*** After pose normalization, 66 depth-buffer views with size 256×256 are captured on the vertices of a given unit geodesic sphere whose mass center is also located in the origin, such that a 3D model can be represented by a set of images. We render the views base on OpenGL.
3. ***Feature Extraction:*** For each view, a specific image processing technique is applied to represent the view as a compact feature vector. Based on the different 2D shape descriptors used (see corresponding subsections), our three algorithms are classified as the following three categories: local feature based (i.e. CM-BOF), global feature based (i.e. GSMD), and composite (i.e. VLGD) methods. The average dimension of this 3D shape descriptor is about 3000.
4. ***Dissimilarity Calculation:*** The last step of our framework is the dissimilarity calculation for two shape de-

scriptors. The basic idea is that, after we get the principal axes of an object, instead of completely solving the problem of fixing the exact positions and directions of these three axes to the canonical coordinate frame, all possible poses are taken into account during the shape matching stage.
The dissimilarity between the query model q and the source model s is defined as,

$$Dis_{q,s} = \min_{0\le i\le 23} \sum_{k=0}^{65} D\left(FV_q(p_0^{'}(k)), FV_s(p_i^{'}(k))\right),$$

where $FV_m = \{FV_m(k)|0 \le k \le 65\}$ denotes the shape descriptor of 3D object m, $FV_m(k)$ stands for the feature vector of view k, the permutations $p_i^{'} = \{p_i^{'}(k)|0 \le k \le 65\}$, $0 \le i \le 23$ indicate the arrangements of views for all (24) possible poses of a normalized model, and $D(\cdot,\cdot)$ measures the dissimilarity between two views. For more details about this multi-view shape matching scheme, we refer the readers to our previous papers [LRS] [LGS10].

3.2. A Local feature based Method: CM-BOF

In our CM-BOF algorithm, each view is described as a word histogram obtained by the vector quantization of the view's salient local features, and the distance between two histograms H_1, H_2 with N_w bins is evaluated by the formula,

$$D(H_1,H_2) = 1 - \frac{\sum_{j=0}^{N_w-1}\min(H_1(j),H_2(j))}{\max(\sum_{j=0}^{N_w-1}H_1(j),\sum_{j=0}^{N_w-1}H_2(j))}.$$

Note that, the 2D salient local feature is calculated using the *VLFeat* matlab source code developed by Vedaldi and Fulkerson [VF]. On average, this 3D shape descriptor contains about 3000 integers. This method is denoted LiCm in the results section.

3.3. A Global feature based Method: GSMD

In our GSMD algorithm, each view is represented as a global feature vector with 47 elements including 35 Zernike moments, 10 Fourier coefficients, eccentricity and compactness, and the dissimilarity between two feature vectors is measured by their L_1 difference.

Note that, the global feature vector is calculated using the C++ source code provided by [CTSO03], and the vector is normalized to its unit L_1 norm. The dimension of this 3D shape descriptor is 3102. This method is denoted LiGsmd in the results section.

3.4. A Composite Method: VLGD

Our VLGD algorithm is a composite method based on a linear combination of CM-BOF and GSMD. More specifically, in this method, a view is expressed by a feature vector consisting of two kinds of shape descriptors, which are used in CM-BOF and GSMD, with pre-specified weights. We experimentally select the weights as $W_{local} = 7.0$ and $W_{global} = 1.0$ for local and global features, respectively, by maximizing the retrieval accuracy on PSB train set with base classification. This method is denoted LiVlgd in the results section.

4. The method of Ryutarou Ohbuchi and Takahiko Furuya

Our algorithm (denoted ObFu in the results section) compares 3D models based on their appearances, that is, by using range images of the 3D models rendered from multiple viewpoints. The algorithm is designed so that it could handle (1) a diverse range of shape representations, including polygon soup, point set, or B-rep solid, and (2) models having articulation or deformation. Almost all 3D model retrieval algorithms deal with geometrical transformation invariance, typically up to similarity transformation. However, invariance to articulation has been dealt with by only a small subset of algorithms, and only for a limited subset of shape representations (e.g., manifold mesh).

Appearance based comparison gives the algorithm its ability to handle diverse shape representation. Invariance to articulation and/or global deformation is achieved through the use of a set of multi-scale, local, visual features integrated into a feature vector per 3D model by using Bag-of-Features (BoF) approach. By comparing an integrated feature per 3D model, instead of a set of thousands of local features per 3D model, cost of comparing a pair of 3D models has become manageable. We call the algorithm Bag-of-Features Dense-SIFT with Extremely randomized clustering tree (BF-DSIFT-E). We present a short description of the algorithm in the following. Please refer to the paper by Furuya et al [TF09] for details.

Figure 2 shows the processing steps of the BF-DSIFT-E algorithm. After normalizing the 3D model for its position and scale, a set of range images of the model is generated by using multiple virtual cameras looking inward at the model sitting at the coordinate origin. From each depth image, our algorithm densely and randomly samples a few hundreds of local, multi-scale image feature using Scale Invariant Feature Transform (SIFT) algorithm by David Lowe ([Low04]). The original SIFT algorithm first detects salient points in the image, and then compute local, multi-scale, visual features at these points. However, we disable the salient-point detector of the SIFT for dense and random placement of sample points. Each SIFT feature encodes, in its 128D vector, position, orientation, and scale of gray-scale gradient change about the sample point.

Typically, a 3D model is rendered into 42 depth images, each one of which then is sampled at 300 or so locations. Thus, a 3D model is described by a set of 13k SIFT features. Naïve comparison of a pair of such sets would take a

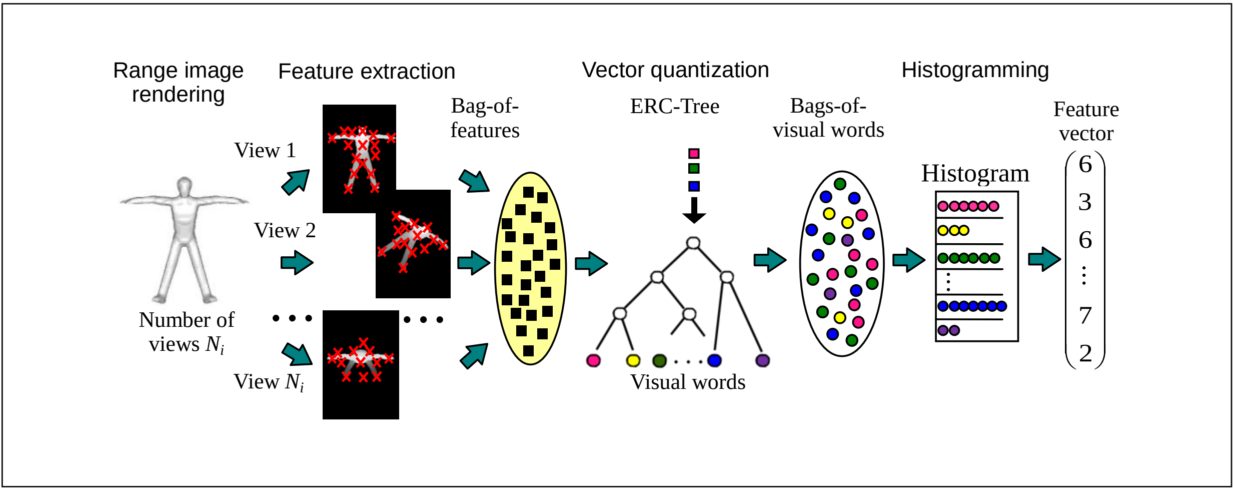

Figure 2: *Flow of the Bag-ofFeature Dense-SIFT with ERC-Tree algorithm*

long time. To reduce the cost of model-to-model dissimilarity computation, the set of 13k visual features is integrated into a feature vector per model by using the BoF ([CDF*04], [SZ03]) approach. The BoF approach vector quantizes, or encodes, each SIFT feature into a representative vector, or "visual word", using a previously learned codebook. These visual words are accumulated into a histogram, which then becomes the feature vector. The optimal dimension of the histogram (i.e., feature vector per 3D model) depends on the shape and diversity of the 3D models in a database. We experimentally choose dimension of about 30k for this track.

To extract this many features quickly, we use a fast GPU-based implementation of the SIFT algorithm called SiftGPU by Wu ([Wu]). We also adopted Extremely Randomized Clustering Tree, or ERC-Tree, by Guerts, et al ([PG06]), for both feature set clustering during codebook learning and for vector quantization of the SIFT features during retrieval [TF09]. The ERC-Tree clustering is much faster for codebook learning than the k-means clustering. The ERC-Tree is again much faster than the linear search of nearest neighbor in the vector quantization step.

After encoding into visual words, frequencies of words are accumulated into a histogram, which becomes a feature vector for the 3D model. Optimal dimension of the feature vector depends on visual complexity of the image from which the SIFT features are extracted, that are, the 3D models in the database. We experimentally found the dimensionality of about 30k for the feature used in this benchmark.

Distance computation among feature vector used a symmetric version of the Kullback-Leibler Divergence (KLD), which is sometimes called relative entropy. The KLD performs well for comparing two histograms.

The KLD for a 30k dimensional vector is rather expensive to compute, especially sing the KLD must be computed 10k times for a database having 10k entries. To accelerate distance computation, the algorithm reduces quantization levels of the histogram entry to a small integer (e.g., 512) via clipping, so that the distance computation loop fits easily in a CPU cache. The algorithm then employs table lookup into a small array to compute costly ln() function quickly [RO08].

5. The methods of Waqar Saleem, Thomas Baumbach, Hannah Bast, and Joachim Giesen

In a nutshell our approach to large scale shape retrieval can be characterized as *localize* (geometric features) and *quantize* (the localized features). Our approach is based on the observation that many geometric feature based shape descriptors—also global ones—can be localized in the following sense: first the shape is sampled, i.e., the boundary that represents it, and then the features are computed at each sample point. The localized features can be treated like words in a document—turning shape retrieval into a text retrieval problem. But localization alone is not enough to provide an efficient, robust and relevant 3D shape search engine. While localization essentially allows for efficient top-k search by enabling efficient indexing of the local features their quantization is needed for robustness (recall), accuracy (precision) and also efficiency. With almost no quantization a shape query might not provide any hit due to small shape variations, noise or even just numerical inaccuracies, whereas quantizing too aggressively can render many hits irrelevant and makes it harder to rank them properly. The two extremes—almost no hits when not quantizing and almost every shape in the collection being a hit—illustrate the virtue of quantization, namely trading-off accuracy and robustness. Note that sampling already introduces some sort of quantization, but this is not enough to ensure robustness (a high recall value). To achieve the latter also the computed words need be quantized.

Our *generic* approach follows the following pipeline to build an index:

1. Sample the shape which is given by a boundary description.
2. Compute features localized at the sample points.
3. Quantize the localized features, providing a (feature) word frequency vector representation.
4. Build a weighted inverted index on the (feature) words.

This pipeline is generic as it allows many different geometric features to be used in its second step. For our experiments we used a localized version of the D2 shape distribution which we will describe in detail. The D2 shape distribution is a global feature in the sense that it considers the relation of the sample point at which the feature is localized to other sample points that can be located all over the shape. In that sense we are dealing with a localized global feature that is not invariant under articulated motions of the shape. We are also considering an example of a localized local feature, namely geodesic D2 distributions that should perform better for articulated motion invariant shape search. For geodesic D2 distributions we consider the distances of sample points on the surface in contrast to the Euclidean distance in ambient space that is used for the D2 distributions. The rest of the pipeline is standard, we use a standard weighting scheme in our index, namely "term-frequency-inverse document frequency" (tf-idf).

A query shape is processed exactly the same way as the shapes in the collection, i.e., a weighted (feature) word frequency vector is computed for the query shape. Shapes s in the collection a ranked with respect to query shape q by the normalized inner product of the weighted word frequency vectors for s the and q.

5.1. Examples: localized D2 shape descriptors

5.1.0.1. Localization We assume that we are given a boundary representation of a 3D shape that is normalized to fit into the unit cube. Our two localized shape descriptors are for each point on the boundary the distribution of distances to the other points on the boundary. Globally the shape is described by the functions that map each point on the boundary to its associated distribution of distances. Our two shape functions differ only in the distance measure used to define them. The first distance measure (denoted SalEucl in the results section) is the Euclidean distance of boundary points, and the second measure (denoted SalGeod) is the geodesic distance between points on the boundary surface, i.e., the length of a shortest path connecting the two points on the boundary. Note that especially the first function is very similar to the D2 shape function studied by Osada et al. [OFCD02].

5.1.0.2. Quantization Obviously our localized descriptors cannot be used or even computed in practice. To discretize them we sample 5000 points from the shape uniformly at random (with respect to surface area). For each sample point we compute the distance to another 100 sample points (sampled uniformly at random from the 4999 remaining sample points). The distributions of these distances is our discretized version of the localized shape descriptor. Globally the shape is now represented by the distributions at all the sample points.

Computing the Euclidean distances of sample points is straightforward, but computing the geodesic distance is not possible without knowing the boundary surface. Hence we only approximate the geodesic distances by building a weighted graph on the sample points and approximating the geodesic distance of two sample points by the shortest path connecting the sample points in the graph. The graph connects each sample point to its k nearest neighbors by an edge. We weigh the edges of the graph by the Euclidean distance of its endpoints. The rationale being that for "small values" the Euclidean and the geodesic distance almost coincide. All the geodesic distances can now by approximated by computing all pairs shortest paths, e.g., by using Dijkstra's algorithm.

6. Discussion of results

For all methods we received for every query a ranking of all models in the dataset. We computed five quality measures for the methods. The results are listed in figure 3.

Furthermore we computed the 11-point interpolated average precision for the methods. We plotted the results in figure 4.

Most of the measures seem to indicate that the order of the methods from most to least succesful is LiVlgd, LiCm, ObFu, LiGsmd, SalEucl, SalGeod.

6.1. Analysis of significance

We want to estimate whether the methods perform significantly different. We took the average precision as the basis for this procedure, as it is a good overall measure for the performance of a method. For every query, we rank the methods by means of their average precision, using the Friedman procedure. We then apply the Tukey-Kramer criterion –also known as honestly significant difference criterion– to decide whether the differences are significant.

In figure 5 we depict the average means and the confidence interval. We take a confidence level of 0.05. In the table below we list which defferences are significany (S) according to this procedure.

Measure	LiCm	LiGsmd	LiVlgd	ObFu	SalEucl	SalGeod
Mean Average Precision	0.60	0.52	0.63	0.55	0.37	0.26
Average First Tier	0.54	0.46	0.55	0.49	0.36	0.23
Average Second Tier	0.70	0.63	0.72	0.61	0.42	0.30
Mean Reciprocal Rank	0.77	0.69	0.78	0.77	0.62	0.51
Recognition Rate	0.68	0.60	0.70	0.68	0.52	0.45

Figure 3: *Retrieval measures for all methods*

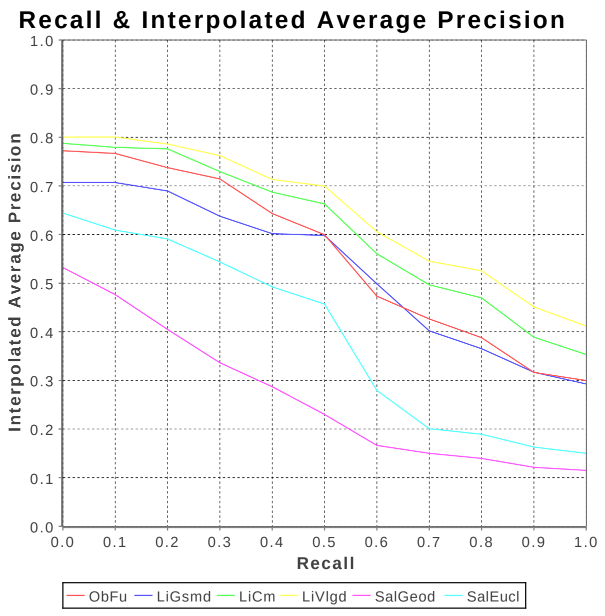

Figure 4: *Recall and precision for all methods.*

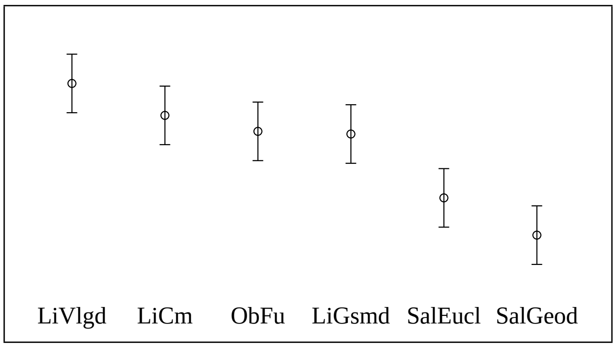

Figure 5: *Friedman Tukey-Kramer confidence intervals*

	1	2	3	4	5	6
1 LiVlgd		-	-	-	S	S
2 LiCm	-		-	-	S	S
3 LiGsmd	-	-		-	S	S
4 ObFu	-	-	-		S	S
5 SalEucl	S	S	S	S		-
6 SalGeod	S	S	S	S	-	

We see that we can only conclude that the first four methods are better than the last two. In order to see whether the differences we see between the other methods are significant, an even larger experiment should be run.

References

[CDF*04] CSURKA G., DANCE C., FAN L., WILLAMOWSKI J., BRAY C.: Visual categorization with bags of keypoints. In *Proc. ECCV '04 workshop on Statistical Learning in Computer Vision* (2004), pp. 59–74. 3

[CTSO03] CHEN D.-Y., TIAN X.-P., SHEN Y.-T., OUHYOUNG M.: On visual similarity based 3D model retrieval. In *Proc. Eurographics 2003* (2003), vol. 22, pp. 223–232. 3

[LGS10] LIAN Z., GODIL A., SUN X.: Visual similarity based 3D shape retrieval using bag-of-features. *Proc. Shape Modeling International (to appear)* (2010). 3

[Low04] LOWE D.: Distinctive image features from scale-invariant keypoints. *IJCV 60(2)* (2004). 3

[LRS] LIAN Z., ROSIN P. L., SUN X.: Rectilinearity of 3D meshes. *International Journal of Computer Vision (in press)*. 3

[OFCD02] OSADA R., FUNKHOUSER T. A., CHAZELLE B., DOBKIN D. P.: Shape distributions. *ACM Trans. Graph. 21*, 4 (2002), 807–832. 5

[PG06] P. GUERTS D. ERNST L. W.: Extremely randomized trees. *Machine Learning 36(1)* (2006), 3–42. 4

[RO08] R. OHBUCHI T. F.: Accelerating bag-of-features sift algorithm for 3d model retrieval. In *Proc. SAMT 2008 workshop on Semantic 3D Media* (2008), pp. 23–30. 4

[ST01] SHEYNIN S. A., TUZIKOV A. V.: Explicit formulae for polyhedra moments. *Pattern Recognition Letters 22* (2001), 1103–1109. 2

[SZ03] SIVIC J., ZISSERMAN A.: Video google: A text retrieval approach to object matching in videos. In *Proc. ICCV 2003* (2003), vol. Vol. 2, pp. 1470–1477. 4

[TF09] T. FURUYA R. O.: Dense sampling and fast encoding for 3d model retrieval using bag-of-visual features. In *Proc. ACM CIVR 2009* (2009). 3, 4

[VF] VEDALDI A., FULKERSON B.: VLFeat: An open and portable library of computer vision algorithms. *http://www.vlfeat.org/*. 3

[Wu] WU C.: Siftgpu: A gpu implementation of david lowe's scale invariant feature transform (sift). http://cs.unc.edu/ ccwu/siftgpu/. 4