Yuki Ohkita, Yuya Ohishi, Takahiko Furuya, Ryutarou Ohbuchi, Non-rigid 3D Model Retrieval Using Set of Local Statistical Features, IEEE ICME 2012 Workshop on Hot Topics in 3D Multimedia (Hot3D), 2012.

Non-rigid 3D Model Retrieval Using Set of Local Statistical Features

Yuki Ohkita, Yuya Ohishi,
University of Yamanashi
Kofu, Yamanashi, Japan
{g09mk004, g11mk008}AT yamanashi.ac.jp

Takahiko Furuya
Nisca Cooporation
Minami-koma, Yamanashi, Japan
snc49925AT_gmail.com

Ryutarou Ohbuchi
University of Yamanashi
Kofu, Yamanashi, Japan
ohbuchiAT_yamanashi.ac.jp

Abstract—**Various algorithms for shape-based retrieval of non-rigid 3D models, with invariance to articulation and/or global deformation, have been developed. A majority of these algorithms assumes that 3D models have mathematically well-defined representations, e.g., closed, manifold mesh. These algorithms are thus not applicable to other types of shape models, for example, those defined as polygon soup. This paper proposes a 3D model retrieval algorithm that accepts diverse 3D shape representations and is is able to compare non-rigid 3D models. The algorithm employs a set of hundreds to thousands of 3D, statistical, local features to describe a 3D model. These features are integrated into a feature vector per 3D model by using bag-of-features approach for efficiency in comparing 3D models and for invariance against articulation and global deformation. Experimental evaluation showed that the algorithm performed well for non-rigid 3D model retrieval.**

***Index Terms*—** 3D model retrieval; 3D geometrical modeling; bag-of-words; shape descriptor, articulated model.

I. INTRODUCTION

Three-dimensional (3D) shape model has become a mainstream multi-media data type. It used for mechanical and architectural design, entertainment, medical diagnosis, and for archaeology. Accordingly, effective and efficient management methods for 3D models, especially via content based retrieval by their shape, has become quite important.

In this paper, we propose a shape-based 3D model retrieval algorithm that accepts diverse set of 3D shape representations, and is invariant to similarity transformation and articulation and/or global deformation. The algorithm employs a 3D feature, not an appearance-based feature, so that the algorithm is able to capture internal as well as external structure of a 3D model. The algorithm accepts a 3D model represented as a set of oriented point sets. Almost any surface-based 3D model representations can be converted into oriented point set by sampling its surfaces by (oriented) points. The algorithm computes hundreds to thousands of *Local Statistical Features*, or *LSF*s, from a 3D model. Each LSF is inherently invariant to similarity transformation. By integrating large number of LSFs into a feature vector per 3D model, the algorithm assumes invariance against articulation and/or global deformation. Experimental evaluation has shown that, for an algorithm that accepts polygons soup models, the algorithm has a very good, if not the best, retrieval accuracy for non-rigid (i.e., articulated) 3D models.

II. RELATED WORK

Requirements for shape-based retrieval of 3D shape models vary depending on its application. Oftentimes, invariance to similarity transformation, that is, a combination of 3DOF translation, 3DOF rotation, and 1DOF (uniform) scaling, is required. Most 3D model retrieval algorithms aim at invariance against similarity transformation.

Invariance to shape representation may also be important, as a 3D shape may be represented by using one of many mutually incompatible 3D shape representations. Recent mechanical CAD models are probably “solid” models, which defines 3D volumes embedded in 3D space. A large portion of 3D models, e.g., those used for movies and games, however, uses polygon soup and/or (a set of) open manifold mesh representations. A 3D model may also be defined by using a set of (unconnected) oriented points placed at the assumed surfaces of a model. While some previous 3D model retrieval algorithms (e.g., [8][1][3][5][4]) could accept diverse set of shape representation, others are limited to manifold or watertight mesh (e.g., [10][11]).

Another possible requirement is invariance, against articulation and/or global deformation. A 3D model in different pose is often required to be treated as the same or similar, as in the case of snake, pliers, or human (e.g., Figure 1) in McGill Shape Benchmark [7]. Earlier 3D model retrieval algorithms (e.g., [8][1][5][3]) use global feature, so they don’t have invariance to articulation; articulated figures are recognized as different. More recent such methods as a set of local features [4] or diffusion-based distance computed on manifold mesh surface [10][11] to describe a 3D model for invariance to articulation. While [4] accepts various shape representations, it is unable to consider internal structure of 3D models as it compares 3D model

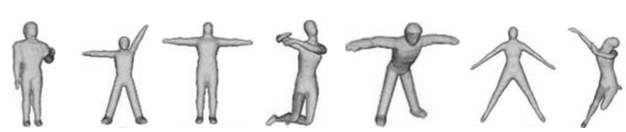

Figure 1. Human figures in McGill Shape Benchmark [7].

Yuki Ohkita, Yuya Ohishi, Takahiko Furuya, Ryutarou Ohbuchi, Non-rigid 3D Model Retrieval Using Set of Local Statistical Features, IEEE ICME 2012 Workshop on Hot Topics in 3D Multimedia (Hot3D), 2012.

based on exterior view of 3D models. Diffusion distance based algorithms (e.g., [10][11]) on the other hand, are limited to manifold meshes.

Our proposed algorithm employs a set of 3D local features to describe a 3D model. It is very tolerant of input shape representation; the feature can be computed from polygon soup or oriented point set models. By integrating the set of local features into a feature vector per 3D model, invariance to articulation is realized, and the cost of comparison between a pair of 3D models is reduced.

II. PROPOSED ALGORITHM

Proposed Local Statistical Feature (LSF) algorithm follows the steps below to compare 3D shape models. (See Figure 2.) The algorithm natively compares 3D models in oriented point set representation. If the 3D models are in surface based representation, e.g., manifold surface or polygon soup, step 1 is required to convert them into oriented point set models.

1. If a 3D model to be compare is in surface based representation it is converted into oriented point set representation consisting of m points.
2. Compute a set of LSFs from the set of m oriented points of the 3D model.
3. Integrated n LSFs into a feature vector per 3D model.
4. Distances among feature vectors of a query model and database models are computed. Top matches are returned as retrieval result.

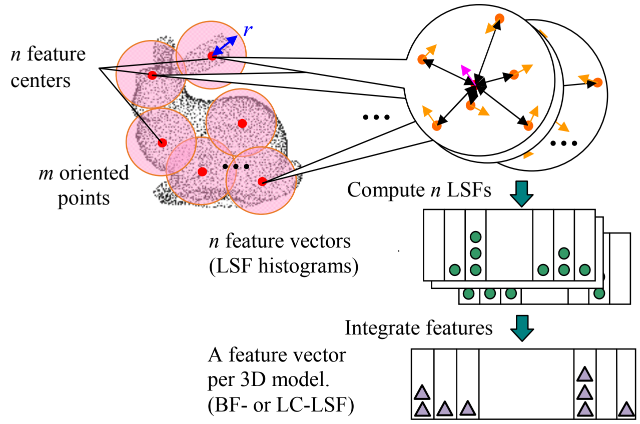

Figure 2. A large number of local features are extracted, and they are integrated into a feature vector per 3D model.

A. Converting surfaces into oriented points

As described above, if a 3D model is represented as a set of surfaces, the model needs to be converted into oriented point set model by sampling the surfaces (with normal vectors) with m uniformly spaced points. First, if the model contained non-triangular polygons, they are triangulated prior to point generation.

Osada et al [8] uses the following equation to samples the surface at position $\mathbf{P}$;

$$\mathbf{P} = \left(1-\sqrt{u_1}\right)\mathbf{t}_1 + \sqrt{r_1}\left(1-u_2\right)\mathbf{t}_2 + \sqrt{u_1}\left(u_2 \cdot \mathbf{t}_3\right). \tag{1}$$

In the equation, $\mathbf{t}_1$, $\mathbf{t}_2$, and $\mathbf{t}_3$ are vertices of the triangle, and u_1 and u_2 are pseudo-random number sequences (PRNS). We use the same equation, but instead of PRNS, we use *low-discrepancy sequence* or *quasi-random number sequence* (QRNS). The QRNS produces points that are more uniformly distributed than PRNS. Our implementation uses *Sobol's QRNS* [12].

Given a total number of samples per 3D model, a number of points proportional to area of a triangle is generated for the triangle. Assume that the total area of the 3D model is S. Then, a point should be placed area $s = S/m$. For the ith triangular facet f_i having an area area(f_i), its number of points q_i is computed as follows;

$$q_i = \text{floor}\left(\text{area}\left(f_i\right)/s + \Delta q_{i-1}\right). \tag{2}$$

$$\Delta q_i = \text{area}\left(f_i\right)/s + \Delta q_{i-1} - q_i \tag{3}$$

Each point i is associated with the position $\mathbf{P}_i$ as well as the surface normal vector of the surface on which the point is placed at.

B. Computing Local Statistical Features

For each 3D model, $n \le m$ *feature points* are selected from the m oriented points, and n LSFs per 3D model are computed at these points. The algorithm selects specified number n of feature points randomly from the m sample points. Each LSF is computed using a set of sample points enclosed in a sphere of radius r centered at the feature point. The LSF radius r for a 3D model is set relative to the radius of smallest sphere enclosing the 3D model. For example, with r=0.5, the LSF sphere of influence is just large enough to enclose the 3D model.

Assume that the feature center is $\mathbf{p_1}$ and its normal vector is $\mathbf{n_1}$. Assume also that point $\mathbf{p_2}$ with an associated normal vector $\mathbf{n_2}$ is a point within the sphere of influence of radius r of the feature center $\mathbf{p_1}$. Using $\mathbf{p_1}$ and $\mathbf{p_2}$, a 4-tuple $(\alpha, \beta, \gamma, \delta)$ consisting of a distance δ and three values α, β, γ related to angles is computed. Here, point $\mathbf{p_1}$ is the LSF feature point (LSF sphere center), and $\mathbf{p_2}$ is the point other than the feature point but contained in the sphere of influence. The four values δ, α, β, and γ are computed as follows (See Figure 3);

$$\alpha = \arctan\left(\mathbf{w}\bullet\mathbf{n}_1, \mathbf{u}\bullet\mathbf{n}_2\right). \tag{4}$$

$$\beta = \mathbf{v}\bullet\mathbf{n}_2 \tag{5}$$

$$\gamma = \mathbf{u}\cdot\left(\mathbf{p}_2 - \mathbf{p}_1 / \left\|\mathbf{p}_2 - \mathbf{p}_1\right\|\right). \tag{6}$$

$$\delta = \left\|\mathbf{p}_2 - \mathbf{p}_1\right\|. \tag{7}$$

where $\mathbf{u} = \mathbf{n}_1$, $\mathbf{v} = \left(\mathbf{p}_2 - \mathbf{p}_1\right)\times\mathbf{u} / \left\|\left(\mathbf{p}_2 - \mathbf{p}_1\right)\times\mathbf{u}\right\|$, and $\mathbf{w} = \mathbf{u}\times\mathbf{v}$.

If there are k points within the sphere, a set of (k-1) tuples are computed for a feature center, and (k-1) each of values δ, α, β, and γ are collected into a 4-dimensional joint histogram. If the histogram has 5 bins for each values of the tuple, resulting LSF feature, if 'flattened' into 1D vector, is $5^4 = 625$ dimensional.

LSF is similar to *Surflet Pair Relation Histograms* (SPRH) [12] by Wahl, et al. LSF differs from SPRH in two aspects; (1) SPRH is global, while LSF is localized within a sphere of influence, and (2) LSF computes a statistics between the center ("feature point") and the other points within the sphere of influence, while SPRH computes a statistics among all the pair of points.

Parameters m, n, and r have impact on retrieval performance and computational cost. Being a statistical feature, one would like to have a large number m of sample points and a large number n of feature points. But an increase in m and/or n obviously increases computational costs. In the experiments described below, as a compromise, we will use m=2,048 and n=512.

A small r ensures locality of features, which makes the algorithm more invariant against global deformation and articulation. A smaller r also means lower computational cost. However, a small radius means a small number of sample points in the sphere. Small number of sample points results in an under-populated, less accurate histogram, especially since the histogram has relatively large number of bins (e.g., 625). A large r would produce a well-populated histogram, but with a significant increase in cost of computation. Also, reduced locality of the feature means reduced invariance to global deformation and articulation.

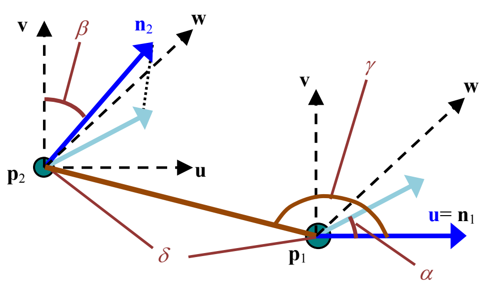

Figure 3. Compute a 4-tuple (α, β, γ, δ) from a pair of oriented points.

C. Integrating LSFs into a feature per 3D model

At this point a 3D model is described by a set of hundreds to thousands of LSF features. Comparing a pair of sets for their similarity (or distance) can be quite expensive. Our algorithm thus integrate the set (a large number of) of features per 3D model into a feature vector per 3D model by using two approaches; well known *Bag-of-Features* (BF) or *Bag-of-Words* approach, and a simpler, *Linear Combination* (*LC*) approach. In the following, a feature vectors per 3D model produced by using BF integration and LC integration is called BF-LSF and LC-LSF, respectively.

Bag-of-features integtation: BF-LSF

In the field of object recognition for 2D images, Bag-of-Features (BF) approach is one of the most popular algorithm to compare among sets, or *bags of features* [2]. The approach first extracts a set of local features from an object (e.g., 3D model) to be compared. Each one of these features is converted, by vector quantization, into a *visual word* in a given *visual codebook* having vocabulary size k. After vector quantization, the object is represented as a set of visual words. The set of visual word is integrated into a histogram with k bins by counting population of each word. This histogram is the feature describing the content. In the process of integration, location of features in the object is ignored.

The visual codebook is learned by clustering of a large set of local features extracted from the objects (e.g., images) to be compared. In our implementation, we used k-means clustering. Each visual word is a center of a cluster. Depending on the diversity and complexity of objects to be compared, optimal number of words k in the codebook, thus the dimension of feature vector per image, may vary. For image recognition and retrieval, k varies from a few hundred to hundreds of thousands.

The number of vocabulary k impacts both retrieval performance and computational cost. For a large vocabulary size k and a large number of samples to cluster, the k-means clustering for codebook learning would take significant amount of time. More importantly, cost of vector quantization, incurred for each new query, would be very significant for a large k.

Linear Combination Integration: LC-LSF

Given a set of local features, linear combination approach combines the features by simply summing, component by component, the histograms of local features into a single histogram having same dimension as the local features. Figure 4 illustrates this integration algorithm. This approach may be considered as a regressed form of the Bag-of-Features approach.

LC-LSF does not require pre-computation of visual codebook by clustering nor vector quantization. During retrieval, LC-LSC does not require vector quantization. So the cost of integration is significantly lower than BF-LSF.

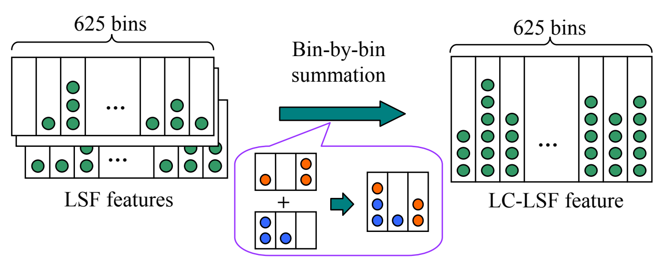

Figure 4. Linear Combination (LC) approach for integrating local statistical features.

No integration: AC-LSF

We compared proposed BF-LSF and LC-LSF with an approach that does not use integration of LSFs. That is, given n LSF features per 3D model, all pairs of LSF features are compared among two sets of LSF features. We call this approach *All pair Comparison LSF* (*AC-LSF*). This approach is computationally very expensive during a search through a database; it requires O(n^2) per pair of 3D models for comparison. For this reason, only a few experiments will be performed using AC-LSF.

D. Computing Distance between 3D Models

Distances between integrated feature vectors $\mathbf{x}_p$ and $\mathbf{x}_q$ produced by LC-LSF or BF-LSF are computed by using one of the following two distance "metrics", $L1$-norm D_{L1} and (symmetric version of) *Kullback-Leibler Divergence* (*KLD*) D_{KLD}.

$$D_{L1}\left(\mathbf{x}_p,\mathbf{x}_q\right)=\sum_{k=1}^{m}\left|x_{p,k}-x_{q,k}\right| \tag{8}$$

$$D_{KLD}\left(\mathbf{x}_p,\mathbf{x}_q\right)=\sum_{k=1}^{m}\left(x_{p,k}-x_{q,k}\right)\log\left(\frac{x_{p,k}}{x_{q,k}}\right) \tag{9}$$

In case of AC-LSF, a set of features, instead of a single feature vector, is given per 3D model. The distance between two models whose sets of features are X_p and X_q, respectively, is computed by using the following equation;

$$D_{AC}\left(X_p,X_q\right)=\sum_{i=1}^{n}\min_{1<j<n}\left\{d\left(\mathbf{x}_{p,i},\mathbf{x}_{q,j}\right)\right\} \tag{10}$$

In the equation, $d\left(\mathbf{x}_{p,i},\mathbf{x}_{q,j}\right)$ is a distance among a pair of LSF features computed by using either $L1$-norm or *KLD*.

If the distance among a pair of LSF features $d\left(\mathbf{x}_{p,i},\mathbf{x}_{q,j}\right)$ in (13) is computed by using $L1$-norm, the distance among 3D models using AC-LSF is called $D_{AC,L1}\left(X_p,X_q\right)$. If $d\left(\mathbf{x}_{p,i},\mathbf{x}_{q,j}\right)$ is computed by using KLD, it is called $D_{AC,KLD}\left(X_p,X_q\right)$.

IV. EXPERIMENTS AND RESULTS

We have conducted experiments to evaluate the proposed algorithms LC-LSF and BF-LSF for shape-based 3D model retrieval. They are compared against AC-LSF as well as SPRH [13] (combined with the surface to point-set model conversion described in section 2,1), Light Field Descriptor (LFD) [1], Spherical Harmonic Descriptor (SHD) [5], and Bag-of-Features Dense SIFT (BF-DSIFT) [4].

Retrieval experiments are performed by using two benchmark databases. McGill Shape Benchmark (MSB) [7] is a set of highly articulated (non-rigid), watertight, high polygon-count yet less geometrically varied/detailed models. Princeton Shape Benchmark (PSB) [9] contains a set of polygon soup models having diverse shape variations. Figure 5 shows examples of models from the databases. MSB consists of 255 models in 10 classes, and includes such articulated shape classes as "humans", "octopuses", "snakes", "pliers", and "spiders". PSB contains two subsets of 907 models each, "training" set and "test" set. Both training set and test set contains about 90 classes. Number of classes for the two sets differs by a few, and while most of the classes overlap, there are classes that are unique to one or the other set. We used PSB test set partitioned into 92 classes for evaluation. Each model from the test set is presented as a query to retrieve models in the test set.

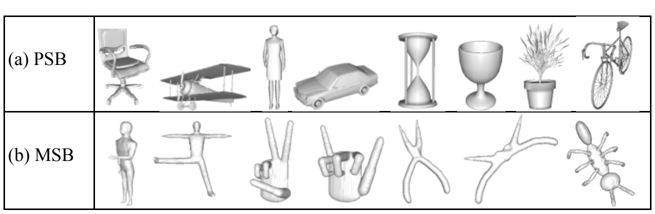

Figure 5. Examples 3D models from Princeton Shape Benchmark (PSB) (a) [12] and McGill Shape Benchmark (MSB) (b) [7].

As the numerical performance index, we use R-Precision, which is a ratio, in percentile, of the models retrieved from the desired class C_k (i.e., the same class as the query) in the top R retrievals, in which R is the size of the class $\left|C_k\right|$

For the experiments below, we used the number of surface sample points m=2,048 and number of feature points n=512. Visual codebook for BF-LSF is learned by k-means clustering from 500k LSF features randomly drawn from all the LSF features extracted from all the 3D models in a PSB test set. Training samples are limited to 500k to contain time necessary for k-means clustering.

A. LSF Radius of Influence and Retrieval Accuracy

In this experiment, impact of LSF radius of influence r on retrieval accuracy is evaluated. Figure 6 plots the results of the experiment.

For MSB, performance peaks exists; the peak for BF-LSF exists at r=0.3 and the peak for LC-LSF exists at r=0.1~0.2. For PSB, on the other hand, no visible peak exists; both for BF-LSF and LC-LSF, the larger the LSF radius, the higher the retrieval accuracy. These tendencies can be explained as follows. The non-rigid, highly articulated models of MSB database prefer bag-of-feature integration of local (smaller radius) features. PSB database that consists mostly of rigid 3D models, however, prefers global feature.

The plot also shows that, for the MSB, BF-LSF significantly outperforms LC-LSF. However, for the PSB, while BF-LSF consistently outperforms LC-LSF, the discrepancy in performance is much smaller.

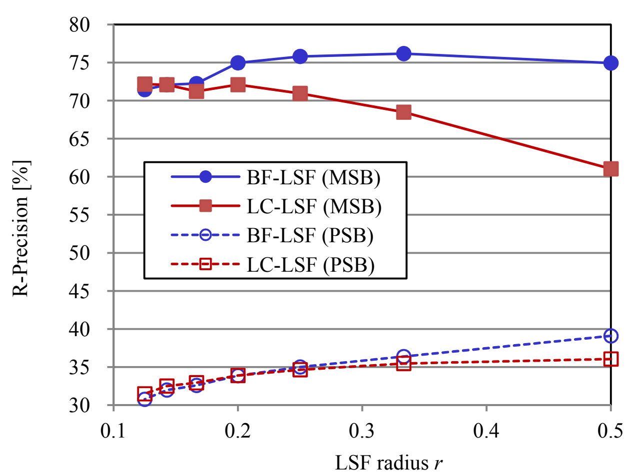

Figure 6. LSF radius of influence r and retrieval accuracy. (L1-norm for distance among integrated features.)

Note that the peak in retrieval accuracy may be affected by other factors than *r*, most notably number of samples *m* and number of LSFs *n* per 3D model. There is interaction between parameters *m*, *n*, and *r*. For an LSF with good locality (i.e., small *r*) to have a well populated histogram, a very large *m* is needed. Also, a large *n* is necessary for BF integration to work well. Thus, ideally, we'd like to have experimented with *m*>>2,048 and/or *n*>>512. However, due mostly to computational cost concern, we settled for *m*=2,048 and *n*=512 for the experiments in this paper.

B. Vocabulary size and retrieval performance

In this set of experiments, we explored the effect of vocabulary size on retrieval accuracy. The result is plotted in Figure 7 and Figure 8.

Recall that the *k*-means has a random component (initialization) so that the quality of vocabulary and thus retrieval accuracy may vary depending on runs. Ideally, for a given vocabulary size, an average of multiple runs should be plotted. In this experiment, however, retrieval accuracy of an individual run is plotted, resulting in a plot with random variation in R-precision. Nonetheless, the plots for both MSB and LSB suggest that a vocabulary size of more than a few thousand words is necessary to achieve a plateau in retrieval accuracy.

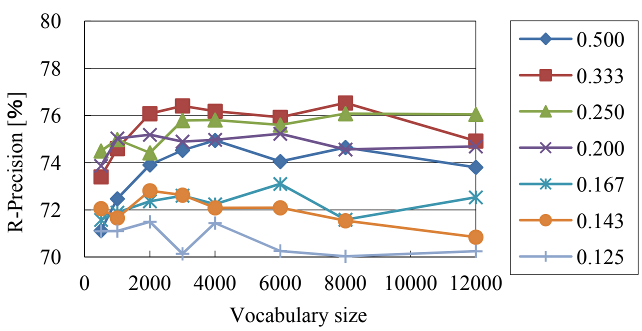

Figure 7. Vocabulary size and retrieval accuracy (MSB).

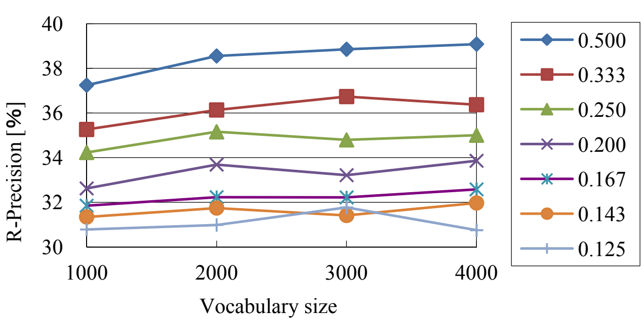

Figure 8. Vocabulary size and retrieval accuracy. (PSB)

C. Computational cost

Table 1 and Table 2 show computational costs of a naive implementation of the algorithm. The timings for feature extraction and integration are for all database models. The timings are for processing all the database models and all the queries of the benchmarks.

Feature extraction and integration are done once per 3D model. To compute an LSF, the naive implementation searched for points in a sphere of LSF influence by brute-force linear search. Similarly, to vector quantize an LSF feature, the naive implementation of BF-LSF searched for the nearest visual word, i.e., cluster center, by using linear search. Both of these could have been made much more efficient, e.g., by using *kd*-tree and other algorithms for fast nearest neighbor search.

In an experiment to search through a database of 907 models, distance computation among a pair of 3D models is performed 907^2=822,649 times. While AC-LSF performs admirably in terms of accuracy, its computational cost is prohibitive. We aborted the AC-LSF experiment for the PSB database after a long time. Overall, while BF-LSF costs more than LC-LSF for the feature integration, gain in retrieval accuracy would justify the cost.

Table 1. Execution timing for MSB benchmark.

MSB	Extract feature	Integrate feature	Compute distance	R-precision [%]
AC-LSF	21min	n/a	>48hrs	73.9
LC-LSF	21min	10min	6s	72.1
BF-LSF	21min	80min	20s	76.5

Table 2. Execution timing for PSB benchmark.

PSB	Extract feature	Integrate feature	Compute distance	R-precision [%]
AC-LSF	75min	aborted	aborted	-
LC-LSF	75min	31min	52s	37.7
BF-LSF	75min	95min	186s	40.9

D. Performance comparison with other algorithms

We compared the retrieval accuracy of proposed LC-LSF and BF-LSF algorithms with four other algorithms, SPRH [13], SHD [5], LFD [1], and BF-DSIFT [4] algorithms. We used our own original implementations of SPRH, LSF, and BF-DSIFT. Executables for SHD and LFD are downloaded from respective author's web sites.

Figure 9 compares these methods for their retrieval accuracy. The first three, SPRH, SHD, and LFD, are all global features, while the latter three, BF-DSIFT, LC-LSF, and BF-LSF employ sets of local features. As expected, the latter three using sets of local features performed better for articulated models of MSB. BF-DSIFT and BF-LSF tied for the 1st place with R-Precision=76%. For rigid yet complex and diverse models of PSB, view-based local feature BF-DSIFT and view-based global feature LFD did better than the others including BF-LSF.

E. Comparison using SHREC 2011 Non-rigid 3D

We also compared the performance of BF-LSF with those of participants in the *SHape REtrieval Contest 2011 Non-rigid 3D Watertight Meshes* track. Please refer to [6] for details of the results. Among 9 entrants of the track, BF-LSF placed 3rd in *Nearest Neighbor* (*NN*) and 6th in the *Discounted Cumulative Gains* (*DCG*) index. Note that an

Yuki Ohkita, Yuya Ohishi, Takahiko Furuya, Ryutarou Ohbuchi, Non-rigid 3D Model Retrieval Using Set of Local Statistical Features, IEEE ICME 2012 Workshop on Hot Topics in 3D Multimedia (Hot3D), 2012.

entry by Sipiran, et al that employed HKS [11], performed worst among the 9 entrants.

While not at the top in terms of retrieval accuracy among the entrants, the proposed algorithm does have an advantage other entrants did not have. As the name of the track suggests, its benchmark database consists of watertight meshes representing 3D solids. All but one of the algorithms in the track requires 3D models to be represented as watertight meshes. (Please note that this is not an issue for the track, since it is a track for "Non-rigid *watertight meshes*".) Consequently, those other algorithms can't handle PSB that contains polygon soup models. Our BF-LSF, on the other hand, does not require the model to be watertight; it can handle a diverse set of 3D model representations, e.g., those found in PSB.

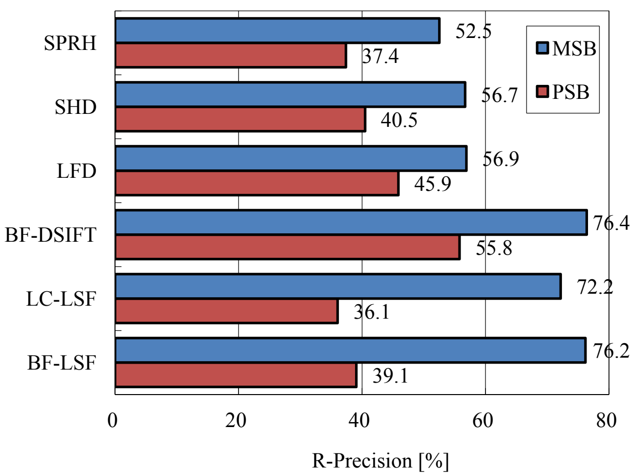

Figure 9. Retrieval accuracies of various algorithms.

V. SUMMARY AND CONCLUSION

We proposed a 3D model retrieval algorithm that tolerates articulation and global deformation, in addition to being able to handle a diverse set of shape representations including polygon soup and oriented point set. A set of 3D local geometrical features is integrated into a feature vector per 3D model to achieve articulation invariance and efficient comparison among 3D models.

Experimental evaluation using two 3D model retrieval benchmarks showed that the algorithm is quite effective for database consisting of highly articulated, yet simpler shapes, that is, McGill Shape Benchmark [7]. For rigid, highly diverse set of shape models in Princeton Shape Benchmark [10], retrieval accuracy of the proposed algorithm is modest compared to the other state of the art algorithms.

As a future work, we'd like to evaluate the method using a benchmark that contains meaningful internal structure. In such a benchmark, BF-LSF, which captures 3D geometrical feature, might have an advantage over algorithms that are based on external appearance of 3D models. We also need to explore the impacts of various parameters, such as the number of sample points m and the radius r of influence of LSF. We'd also like to experiment with distance metric learning for improved retrieval accuracy.

ACKNOWLEDGMENT

This research has been funded in part by the Ministry of Education, Culture, Sports, Sciences, and Technology of Japan (No. 18300068).

REFERENCES

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

[2] G. Csurka, C.R. Dance, L. Fan, J. Willamowski, C. Bray, Visual Categorization with Bags of Keypoints, *Proc. ECCV '04 workshop on Statistical Learning in Computer Vision*, 59-74, (2004).

[3] P. Daras, A. Axenopoulos, A Compact Multi-view Descriptor for 3D Object Retrieval, *Proc. Seventh International Workshop on Content Based Multimedia Indexing*, 115-119, (2009).

[4] T. Furuya, R. Ohbuchi, Dense Sampling and Fast Encoding for 3D Model Retrieval Using Bag-of-Visual Features, *Proc. ACM Intern'l Conf. on Image and Video Retrieval* 2009 (*CIVR* 2009), July 8-10, (2009).

[5] M. Kazhdan, T. Funkhouser, S. Rusinkiewicz, Rotation Invariant Spherical Harmonics Representation of 3D Shape Descriptors, *Proc. Symp. of Geometry Processing* 2003, 167-175 (2003)

[6] Z. Lian, et al., Shape Retrieval on Non-rigid 3D Watertight Meshes, *Proc. Eurographics/ACM SIGGRAPH Symp. on 3D Object Retrieval* 2011, (2011).

[7] McGill 3D Shape Benchmark, <http://www.cim.mcgill.ca/~shape/benchMark/>

[8] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, Shape Distributions, *ACM Trans. on Graphics*, **21**(4), pp. 807-832, October 2002.

[9] Princeton Shape Benchmark, <http://shape.cs.princeton.edu/benchmark/>

[10] M. Reuter, F. E. Wolter, M. Shenton, M. Niethammer, Laplace-Beltrami eigenvalues and topological features of eigenfunctions for statistical shape analysis, *Computer-Aided Design*, **41**(10), 739–755, (2009).

[11] J. Sun, M. Ovsjanikov, L. Guibas, A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion, *Proc. Eurographics Symp. On Geometry Processing* 2009 (2009).

[12] W.H. Press et al. *Numerical Recipes in C –The art of Scientific Computing*, Cambridge Universe Press, Cambridge, UK, 1992, pp.309-315.

[13] E. Wahl, U. Hillenbrand, G. Hirzinger, Surflet-Pair-Relation Histograms: A Statistical 3D-Shape Representation for Rapid Classification, Proc. *of the Fourth Intern'l Conf. on 3-D Digital Imaging and Modeling* (*3DIM 2003*), 474-481, Banff, Canada, (2003).