Takahiko Furuya, Ryutarou Ohbuchi, Aggregating sparse binarized local features by summing for efficient 3D model retrieval, *Proc. of the Second IEEE Int'l Conf. on Multimedia Big Data* (*BigMM 2016*), Oral paper, April 20-22, 2016, Taipei, Taiwan, (2016)

Aggregating sparse binarized local features by summing for efficient 3D model retrieval

Takahiko Furuya
Graduate School of Medicine and Engineering
University of Yamanashi
Kofu, Japan
takahikof@yamanashi.ac.jp

Ryutarou Ohbuchi
Graduate School of Medicine and Engineering
University of Yamanashi
Kofu, Japan
ohbuchi@yamanashi.ac.jp

***Abstract*—An effective and widespread approach for shape-based 3D model retrieval (3DMR) is to use a feature vector per 3D model obtained by aggregating, or pooling, a set of local features extracted from the 3D model. State-of-the-art feature aggregation algorithms, such as Fisher Vector (FV) coding [7] or Super Vector (SV) coding [22], used in the approach is not spatially efficient, however. The FV or SV, for example, typically encodes a local feature into a very high-dimensional (e.g., 300k-dimensional) vector. For a database containing a large number of 3D models, the spatial cost of storing all the aggregated feature vectors for the database becomes very high. In this paper, we propose a novel, spatially efficient yet accurate feature aggregation algorithm called *Sum of Sparse Binary codes* (*SSB*) aggregation. The SSB first encodes a local feature into a highly sparse binary code. Then, a set of sparse binary codes are aggregated efficiently by simple summing into a compact feature vector. We also propose *fast SSB* (*fSSB*) aggregation, which is a computationally efficient approximation of the SSB. Experiments using a 3DMR scenario show that the proposed algorithms are significantly more efficient than the state-of-the-art feature aggregation algorithms we have compared against. At the same time, retrieval accuracies of the proposed algorithms are equal or better than the state-of-the-art aggregation algorithms.**

Keywords- feature encoding, feature aggregation, local feature, sparse coding, binary feature, shape-based 3D model retrieval

I. INTRODUCTION

Three-dimensional (3D) shape model has become an important media data type for a wide variety of application areas including mechanical design, medical diagnosis and treatment planning, architectural design, or for navigation of autonomous vehicle. Recent proliferation of 3D printers and 3D scanners (e.g., RGB-D cameras) has afforded us opportunities for capturing, editing, or generating 3D shapes in our daily life. These trends have given impetus to develop effective and efficient, and scalable methods for 3D shape analysis, recognition, or retrieval.

The most popular approach for shape-based 3D model retrieval (3DMR) (e.g., [4], [17]) is to extract a set of local features from a 3D model and then aggregating them into a single feature vector per 3D model. A feature vector per 3D model generated by such an approach often possesses invariance against articulation or global deformation of the 3D model. By using aggregated features, comparison of a pair of shapes is more efficient. In comparison, a naïve approach for comparing two sets of local features can be expensive. In *whole-based* 3DMR (e.g., [4], [28]), an entire 3D model is the query, a set of local features extracted from the 3D model is aggregated into a feature vector for the 3D model. In *part-based* 3DMR (e.g., [20]), a partial 3D shape is the query. An aggregated feature vector for the query is compared against a large number of aggregated feature vectors corresponding to multiple 3D region-of-interests (ROIs) sampled from 3D models in a database.

A number of feature aggregation algorithms have been proposed [25]. Feature aggregation generally consists of two steps, i.e., *encoding* and *pooling* of local features. A codebook, or a set of codewords, is determined, typically, by clustering the set of local features. A local feature is encoded by using codewords around the feature as well as the statistics associated with the codewords. A state-of-the-art approach, e.g., Fisher Vector (FV) coding [7] or Super Vector (SV) coding [22], performs accurate feature encoding by using statistics such as density, mean, and/or variance of local features. The set of encoded local features is pooled into a single feature to reduce temporal and spatial cost for feature comparison. In whole-based 3DMR scenario [18], SV coding especially showed superior retrieval accuracy to the other state-of-the-art approaches including, FV, Locality-constrained Linear (LL) coding [12], and Vector of Locally Aggregated Descriptors (VLAD) [9].

Despite their success, most of these feature aggregation algorithms are spatially inefficient. Their feature encoding step produces a very high-dimensional (e.g., 300k dim.), dense, and real-valued representation for a local feature. Therefore, for a database having a significant size, memory footprint for storing the set of encoded local features could become a problem. This is so even though the set of encoded features is pooled into a single feature per 3D model or per 3D ROI. Certain application, e.g., part-based 3DMR, requires the encoded local features to be stored on memory *without pooling* so that an encoded local feature can be reused for computing aggregated features of a large number of overlapping ROIs per 3D model.

In this paper, to achieve efficient and accurate feature aggregation, we propose a pair of novel feature aggregation algorithms that employ sparse binary encoding of local features. One of the proposed algorithms is called *Sum of Sparse Binary*

codes (*SSB*) aggregation. The SSB encodes a local feature into a sparse binary code in which a small number (e.g., 3) of bits are '1' while the remaining bits are '0'. The sparse binary code is very compact to store since only the positions of '1' bits need to be recorded. Feature encoding in SSB is done by combining *k*-Sparse Autoencoder (*k*SA) [3] with Angular Quantization-based Binary Codes (AQBC) algorithm [23]. A local feature, which is typically dense, real-valued vector, is first sparsely encoded via *k*SA and is then binarized by using AQBC. The *k*SA is expected to perform accurate feature encoding for two reasons. Firstly, *k*SA could *jointly optimize* the codebook learning and feature encoding. In comparison, most of the previous feature aggregation algorithms, e.g., SV and FV, employ a "greedy" approach. That is, they first learn a codebook by clustering the set of training local features. Then, in a disconnected step, they encode local features by using the learned codebook. Secondly, *k*-sparse constraint of *k*SA could enhance saliency of encoded local features. Effectiveness of *k*-sparseness has been demonstrated in several state-of-the-art feature aggregation algorithms, e.g., LL or Localized Soft-assignment (LS) coding [13]. The AQBC efficiently converts a sparsely encoded, real-valued local feature into a binary code having low quantization error. Then, simply by summing, a set of sparse binary codes are efficiently pooled into a feature per 3D model.

In addition to SSB, we also propose *fast SSB* (*fSSB*) aggregation, a computationally efficient approximation of SSB. Computational bottleneck of the SSB is its feature encoding step using *k*SA. We employ a set of "landmark" local features and their tree-structured index to accelerate the feature encoding.

We evaluate the proposed algorithms by using a whole-based 3DMR scenario. Experimental results using multiple benchmark databases and a variety of local features show superior performance of the proposed algorithms in terms of memory efficiency, time efficiency, and retrieval accuracy.

Contributions of this paper can be summarized as follows;

- Proposition of SSB aggregation and its approximation, fSSB aggregation, for more efficient yet accurate aggregation of local features.
- Empirical evaluation of the proposed feature aggregation algorithms using a whole-based 3DMR scenario.

II. Related Work

A. Aggregation of Local Features

Algorithms for aggregating local features may be classified into two groups, that are, sparse coding (SC)-based approach and higher-order statistics (HS)-based approach. The SC-based approach includes Bag-of-Features (BF) [8], ScSPM [11], LL coding, LS coding, and Diffusion-on-Manifold [19]. They sparsely encode a local feature by using weighted linear sum of neighboring codewords. Sparse representation of local features is compact since it can be described by indices of codewords and their coefficients. However, the existing SC-based aggregation algorithms are not necessarily optimal in terms of accuracy since their codebook learning and feature encoding are done separately. The HS-based approach includes FV, SV, and VLAD. They could perform accurate feature encoding by exploiting high order statistics, e.g., density, mean, and/or variances, computed from local features and codewords associated with them. Since a HS-based approach often produces very high dimensional encoded features, the HS-based feature aggregation would not be suitable for spatially efficient 3DMR.

The proposed SSB aggregation falls into the SC-based approach. The SSB differs from the other SC-based algorithms in that the SSB jointly optimizes codebook and feature encoding by using *k*SA for higher accuracy. The SSB also binarizes the sparsely encoded local features to further compress the features.

B. Efficient 3DMR using Binary Local Features

Several recent studies employed binary local features for efficient retrieval or matching of 3D shapes. Matsuda et al. [21] proposed lightweight binary local features for 3D voxels, i.e., 3DBRIEF and 3DORB. They can be extracted quickly from a 3D voxel. Also, these local features are compact since each local feature is represented as a binary code whose length is hundreds of bits. Prakhya et al. [16] proposed a fast-to-compute binary local feature called B-SHOT for fast and memory-efficient keypoint matching on 3D point clouds. However, these binary local features discard a large amount of information about local 3D geometry during feature extraction. Therefore, aggregated features computed from the set of binary local features tend to yield insufficient retrieval accuracy. Also, feature aggregation algorithms intended for binary local features, e.g., bag of binary words [6] or Fisher Vector of Bernoulli Mixture Model [27], yield high-dimensional and dense aggregated features as with the BF or FV for real-valued local features. In contrast, the SSB aggregation can be applied to a variety of rich real-valued local features to produce compact yet salient aggregated features.

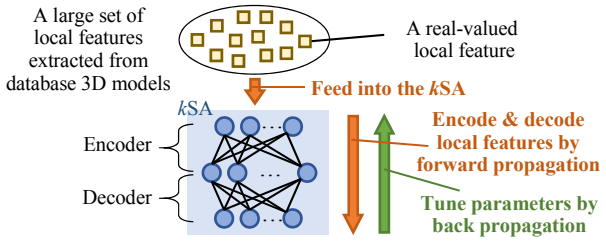
III. Proposed Algorithm

A. Overview of the Algorithm

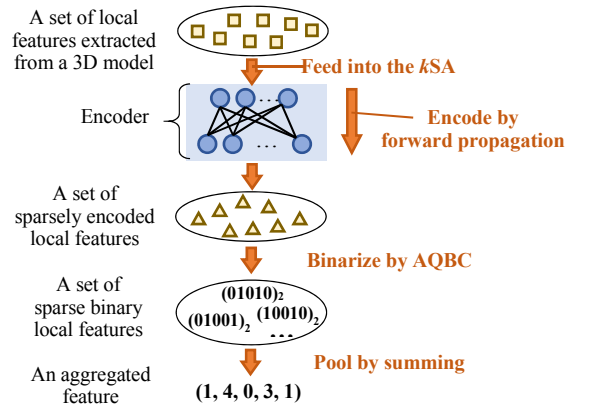
Toward more efficient yet accurate aggregation of local features, we propose novel feature aggregation algorithms that employ sparse and binary representation of local features. One of the proposed algorithms is called *Sum of Sparse Binary codes* (*SSB*) aggregation. The SSB (Fig. 1b) encodes a set of real-valued local features into a set of sparse binary codes. Sparsifying local features enhances saliency of the local features. Binarization that follows produces a compact representation for the local features. The set of sparse binary codes are efficiently aggregated into a single feature vector per 3D model by simple sum-pooling. We employ *k*-Sparse Autoencoder (*k*SA) [3] to generate sparse codes for the local features. The *k*SA is expected to produce accurate sparse codes of the local features since it jointly optimizes a codebook and feature encoding. To accurately and efficiently binarize the sparse features, we use Angular Quantization-based Binary Codes (AQBC) algorithm [23]. The AQBC efficiently converts real vectors into binary codes with low quantization error.

While the feature encoding of the SSB is memory-efficient and accurate, it becomes time-consuming especially when the *k*SA contains a large number of codewords. Computation time for feature encoding increases linearly to the number of codewords of the *k*SA. To alleviate this bottleneck, we also propose a faster-to-compute approximation of the SSB, which we call *fast SSB* (*fSSB*) aggregation (Fig. 1c), by using an index structure in the local feature space. The fSSB is able to encode a

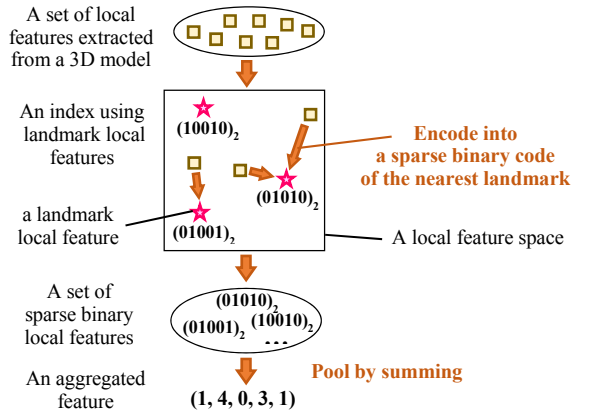
local feature in constant time regardless of number of codewords. To be specific, as a pre-processing, a set of "landmark" local features is selected and their sparse binary codes are computed by using the *k*SA and the AQBC algorithm. Also, the set of landmark local features is indexed by using a *k*d-tree. At the feature encoding stage, each local feature is vector quantized into its nearest landmark local feature and is encoded into a sparse binary code for the landmark. This approximation of feature coding for the fSSB can be computed very quickly since the coding only need to find the nearest landmark for the local feature via a tree-structured index. As with the original SSB, an aggregated feature for the fSSB is formed by summing the sparse binary codes.



(a) Training a *k*SA (pre-processing).



(b) Sum of Sparse Binary codes (SSB) aggregation.



(c) fast SSB (fSSB) aggregation.

Figure 1. Overview of the proposed feature aggregation algorithms.

B. Sum of Sparse Binary Codes (SSB) Aggregation

1) Aggregating Local Features by SSB

As with the other existing feature aggregation algorithms, the SSB aggregation proceeds in two steps, that are, *encoding* and *pooling* of local features. Pre-processing for the SSB, i.e., training *k*SA, will be described in Section III-B-2.

At the encoding stage, a set of local features extracted from a 3D model is encoded into a set of sparse binary codes. Each local feature is first fed into the trained *k*SA to generate sparse representation of the local feature. We use the following equation to sparsify each local feature;

$$\mathbf{h} = \sigma(\mathbf{y}) = \sigma(\mathbf{x}\mathbf{W}_e) \tag{1}$$

where $\mathbf{x}$ is a local feature represented as a real-valued and dense vector, and $\mathbf{W}_e$ indicates a weight matrix for the encoder of *k*SA. $\mathbf{h}$ is unit activation in the hidden layer of the *k*SA, which is a sparse representation of the local feature of $\mathbf{x}$. $\mathbf{h}$ is an n-dimensional vector, where n is equal to the number of units (or codewords) in the hidden layer, and $\mathbf{h}$ has only k non-zero elements. σ is a non-linear activation function. The *k*SA combines Rectified Linear Units (ReLU) [2] and k-largest value selection for activation function. Specifically, each element y_i of $\mathbf{y}$ is rectified by ReLU, i.e., $z_i = max(0, y_i)$. And k largest elements among $\{z_i \mid i = 1, \ldots, n\}$ are kept while the rest are set to zero.

The set of sparsified local features is then binarized by using the AQBC algorithm. In the n-dimensional feature space, the AQBC binarizes each sparse local feature by vector-quantizing it into a vertex of n-dimensional unit hyper-cube that has maximum Cosine similarity with the local feature. Finding the most similar vertex to the local feature can be computed very efficiently when the feature is highly sparse [23]. Each binary code is represented as an n-bit string. Since the local features encoded by *k*SA are sparse, the binarized codes generated by the AQBC are also sparse. That is, most bits within the n-bit binary code are '0'. Note that the number of '1' bits in the binary code produced by the AQBC is not necessarily equal to k. In practice, however, the AQBC most often generates a binary code having k '1's.

The set of sparse binarized local features produced by the method described above is very compact. We only need to store bit positions of '1's within the binary code instead of storing the whole n-bit code. For example, if we were to use 4 bytes to store a bit position of '1' and the *k*SA operates with k=3, each sparse binarized local feature occupies only 4×3=12 bytes, and the feature size does not depend on the length n of the binary code.

At the pooling stage, the set of binarized sparse local features is efficiently aggregated into a single feature vector per 3D model by sum-pooling. The aggregated feature is an n-dimensional vector whose element is either zero or non-zero integer. To effectively compare the aggregated features, each aggregated feature vector is normalized, in succession, by power-normalization and L2-normalization, as with [7]. Comparison of a pair of normalized aggregated features is done by employing Cosine similarity.

The SSB aggregation has two hyper-parameters that need to be manually chosen, i.e., the number of units n in the hidden layer of kSA and the number non-zero elements k of unit activation. As we will show in the experimental section, n should be large (e.g., n=8,000) and k should be small (e.g., k=3) to obtain high retrieval accuracy. Small k is essential for a compact sparse binary code.

2) Training kSA

Prior to training the kSA, we first normalize the set of local features by using PCA-whitening. The set of local features is projected onto a lower dimensional linear subspace by using PCA, and then whitened so that each pair of dimensions in the subspace is uncorrelated. We fix the dimension of the subspace d to 64 throughout the experiments.

The kSA used in the SSB aggregation has a single layer, which consists of an input layer, a hidden layer, and an output layer (see Fig. 1a). The input layer and the output layer contains d units while the hidden layer has n units. Each pair of units between two adjacent layers are connected by a weighted edge. The encoder associates the input layer with the hidden layer while the decoder associates the hidden layer with the output layer. The kSA is trained so that it can reconstruct the local features with minimal error. We use the following objective function for training.

$$E = \frac{1}{T}\sum_{i=1}^{T}\left\|\mathbf{x}_i - \hat{\mathbf{x}}_i\right\|_2^2 + \lambda\left(\left\|\mathbf{W}_\mathrm{e}\right\|_2 + \left\|\mathbf{W}_\mathrm{d}\right\|_2\right) \qquad (2)$$

In the equation above, two matrices $\mathbf{W}_\mathrm{e}$ and $\mathbf{W}_\mathrm{d}$ are weight matrix for the encoder and the decoder, respectively, $\mathbf{x}_i$ is one of T training local features, $\hat{\mathbf{x}}_i$ is reconstructed local feature generated by kSA of the input $\mathbf{x}_i$. $\hat{\mathbf{x}}$ is computed by decoding the sparsely encoded local feature $\mathbf{h}$ in the hidden layer, i.e., $\hat{\mathbf{x}} = \mathbf{h}\mathbf{W}_\mathrm{d}$. In (2), the first term is mean squared reconstruction error of T training local features, while the second term regularizes the training. These two terms are balanced by a hyper-parameter λ, which we fix at 0.0001 throughout the experiments. We use T=250k local features randomly selected from the database for training.

The two matrices $\mathbf{W}_\mathrm{e}$ and $\mathbf{W}_\mathrm{d}$ are tuned so that the objective function is minimized through the training. Optimization of (2) is performed by using Stochastic Gradient Descent (SGD) with mini-batch. Each mini-batch includes 200 training local features randomly chosen from the set of T training local features. To adaptively assign a learning rate to each parameter in $\mathbf{W}_\mathrm{e}$ and $\mathbf{W}_\mathrm{d}$, we use AdaGrad algorithm [10] with an initial learning rate = 0.2. The optimization is iterated for 50 epochs.

C. Fast SSB (fSSB) Aggregation

1) Aggregating Local Features by fSSB

The feature encoding using kSA (i.e., computing (1)) is time-consuming when the kSA has a large number (e.g., thousands) of units in the hidden layer, even if the process is accelerated by using a GPU. To reduce the temporal cost for feature encoding, we propose fSSB, which is an approximation of the SSB that uses a tree-structured index in the local feature space. The fSSB aggregation removes the kSA from the feature encoding step of the SSB.

The tree-structured index is constructed as follows. Firstly, a large set of landmark local features is selected. We use L=250k landmark local features randomly sampled from the set of local features extracted from all the 3D models in the database. Each landmark feature is then encoded into a sparse binary code by using the method described in Section III-B-1. Each sparse binary code is associated with its corresponding landmark feature. Also, a kd-tree is built in the d-dimensional local feature space to index the set of L landmark features.

Once the tree-structured index is constructed, encoding of local feature can be computed very efficiently. Given a set of local features extracted from a 3D model, each local feature is vector-quantized into its nearest landmark local feature, and is encoded into the sparse binary code associated with the nearest landmark. Finding nearest landmarks for the local features can be computed quickly by using the kd-tree. After feature encoding, pooling is performed identically to the SSB. That is, the set of sparse binary codes for the 3D model is aggregated into a single feature vector by sum-pooling. The pooled feature vector is power-normalized and then L2-normalized for comparison.

2) Computational Complexity

Compared to the SSB, the fSSB has an increased spatial complexity for it needs to store the index structure on memory. However, memory usage of the fSSB is reasonable for practical use. Assume that, for example, we use L=250k landmark local features, each of which is a d=64 dimensional floating point vector. Assume also that we use a kSA with the number of non-zero elements in the hidden layer k=3. The set of landmark local features occupies 250k×64×4byte=64Mbyte. And the set of sparse binary codes for the landmarks requires 250k×3×4byte=3Mbyte. The index structure consumes about 70Mbyte including the kd-tree. The index fits easily in a main memory of a resource-constrained computer.

Temporal cost of the fSSB for encoding a set of f local features is $O(f \times \log L)$, dominated by the cost of approximate nearest neighbor search by using a kd-tree. In comparison, the SSB, which uses a kSA having n units in the hidden layer in place of the index structure, requires $O(f \times n)$ for feature encoding by computing (1). Note that time complexity of feature encoding for the fSSB is independent of n. We will show, in the experiments, that the fSSB aggregation is done in near-constant time regardless of n.

IV. Experiments and Results

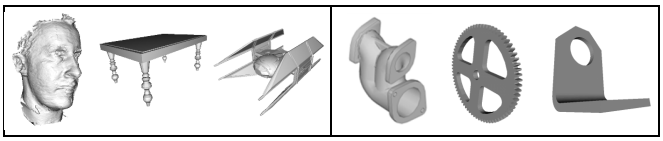
A. Experimental Setup

1) Benchmark Databases

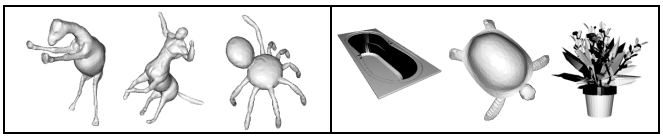
To evaluate efficiency and accuracy of the proposed feature aggregation algorithms, we use four benchmark datasets for 3DMR, that are, the Princeton Shape Benchmark (PSB) [14], the Engineering Shape Benchmark (ESB) [15], the SHREC 2011 Non-Rigid watertight meshes dataset (SH11NR) [28], and the SHREC 2014 Large-scale Comprehensive 3D shape retrieval dataset (SH14LC) [4]. Fig. 2 shows examples of 3D models contained in the benchmark datasets. In Fig. 2, N_m is the number of 3D models in the dataset and N_c is the number of semantic categories. The PSB and SH14LC contain diverse and rigid 3D models of animals, plants, vehicle, furniture, building, etc. The

ESB consists of 3D CAD models of mechanical parts. The SH11NR has non-rigid 3D models of humans or animals. For each of these benchmarks, a 3D model in the dataset is used as the query to retrieve the rest of the 3D models in the dataset. Retrieval accuracy measured in average precision is averaged over all the models in a benchmark to obtain a Mean Average Precision (MAP) [%] value for the benchmark.

To measure computation times for the database pre-processing and the query processing stages, we use a PC having an *Intel Core i7-5930K* CPU, a *GeForce GTX Titan X* GPU, and 32GB DRAM.



(a) PSB (N_M =907, N_C =92) (b) ESB (N_M =867, N_C =45)



(c) SH11NR (N_M =600, N_C =30) (d) SH14LC (N_M=8,987, N_C=171)

Figure 2. Example of 3D models contained in benchmarks datasets.

2) *Local Feature Aggregation Algorithms*

We compare the SSB and the fSSB against six existing feature aggregation algorithms, i.e., BF [8], LL [12], FV [7], VLAD (VL) [9], SV [22], and DM [19]. To learn the codewords, *k*-means clustering algorithm is used for BF, LL, and VL, while Gaussian Mixture Model clustering algorithm is used for FV and SV. For sparse coding-based aggregation methods including BF and LL, we learn 8k codewords. For higher-order statistics-based methods including VL, FV, and SV, the number of codewords are determined so that dimensionality of aggregated feature vector becomes about 8k. For DM, we use 8k local features to generate a manifold graph for aggregation. All the aggregated feature vectors are power-normalized and then L2-normalized. We use Cosine similarity for comparison among aggregated feature vectors.

For both SSB and fSSB aggregations, we set the hyper-parameters n=8k, k=3, and L=250k unless otherwise stated. Experimental evaluation of effects these hyper-parameters have on retrieval accuracy and efficiency will be presented below.

3) *Local Features*

For the 3DMR experiment, we use six local features, that are; Position and Orientation Distribution (POD) [19], Spin Image (SI) [1], Local Statistical Feature (LSF) [26], Rotational Projection Statistics (RoPS) [24], Dense SIFT (DSIFT) [17], and Multi-Orientation One SIFT (MO1SIFT) [18]. They can be classified into two groups; local 3D geometric feature including POD, SI, LSF, RoPS, and local 2D visual feature including DSIFT and MO1SIFT.

In this paper, we follow the experimental settings by Furuya et al. [19]. For local 3D geometric features, i.e., POD, SI, LSF, and RoPS, a 3D model is first converted into an oriented point set for feature extraction. We sample 3k oriented points from surfaces of a 3D model. Then, a local geometric feature is computed from a local sphere of interest. Each local sphere has random radius and its center is one of the 3k oriented points. Each 3D model is thus described by a set of 3k local features. For local 2D visual features, i.e., DSIFT and MO1SIFT, a 3D model is first converted into a set of 42 depth images by rendering the 3D model from 42 viewpoints spaced uniformly in solid angle. DSIFT densely extracts a set of 300 SIFT [5] features at random positions from a rendered image. A set of 13k SIFT features is thus extracted from a 3D model. MO1SIFT extracts a SIFT feature per rendered image. For improved rotation invariance, each rendered image is rotated, in-plane, to 16 orientations prior to global SIFT extraction. Each 3D model is thus described by a set of 42×16=672 MO1SIFT features.

B. *Experimental Results*

1) *Hyper-parameters and Retrieval Accuracy*

In this section, we investigate influences of hyper-parameters of *k*SA used for the fSSB aggregation. We use the PSB dataset for this evaluation. Please note that, while this section presents results for the fSSB only, results similar to these have been observed for the SSB aggregation as well.

Fig. 3 plots retrieval accuracies of the fSSB against the number of units n in the hidden layer of *k*SA. Another hyper-parameter for *k*SA, i.e., the number of non-zero hidden units k, is fixed at 3 for the experiment. In Fig. 3, retrieval accuracies saturate at around n=5k~10k for most local features. Note that, in *k*SA, more hidden units leads to more computation time for feature aggregation. We will evaluate accuracy and efficiency of the approximation method used in fSSB in the next section.

Fig. 4 shows relationship between the number of non-zero hidden units k for *k*SA and retrieval accuracy. Here, the number of hidden units is fixed at n=8k. We observe that retrieval accuracies have peaks at k=2 or k=3 for most local features for the value of n. The DSIFT is an exception with its highest retrieval accuracy at k=1. These results suggest that, *k*SA with its k-sparseness, works best with a small k.

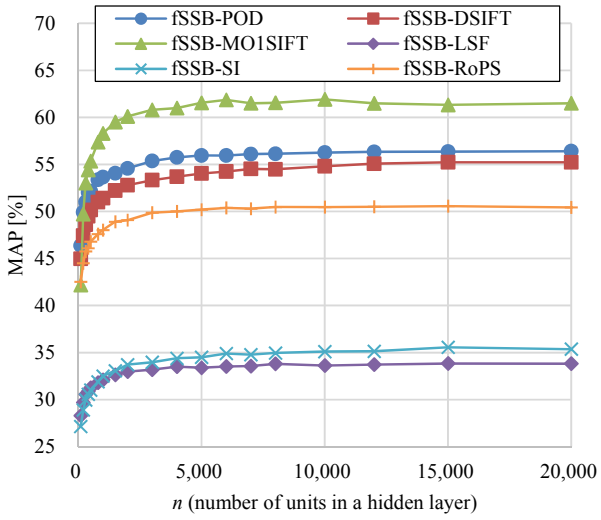


Figure 3. Number of hidden units and retrieval accuraccy (PSB).

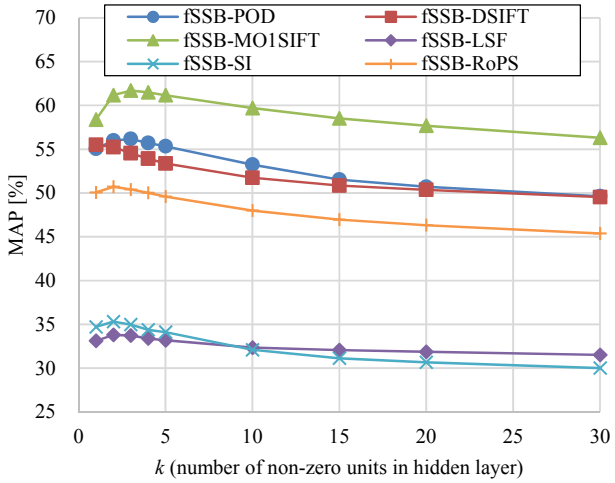


Figure 4. Number of non-zero hidden units and retrieval accuracy (PSB).

2) *Effectiveness of fSSB Aggregation*

We evaluate effectiveness of the fSSB, which is an approximation of the SSB aggregation, both in terms of computational efficiency and retrieval accuracy.

Fig. 5 plots feature aggregation time against the number of hidden units n for kSA. We use a set of 3k POD features, each of which is dimension reduced by PCA to d=64 dimensions, for feature aggregation. Fig. 5 indicates that computation time for the SSB roughly linearly increases with n. Note that feature aggregation of the SSB is accelerated by using a GPU. More computation time would be necessary if a CPU is used for the SSB aggregation. In comparison, feature aggregation time for the fSSB is nearly constant regardless of n. The fSSB replaces kSA with a nearest neighbor search that uses an efficient spatial index structure to find a landmark feature closest to a given local feature to be encoded. For example, when n=8k, the fSSB takes about 0.04s for feature aggregation, which is about 10 times faster than feature encoding using the SSB aggregation that takes about 0.39s.

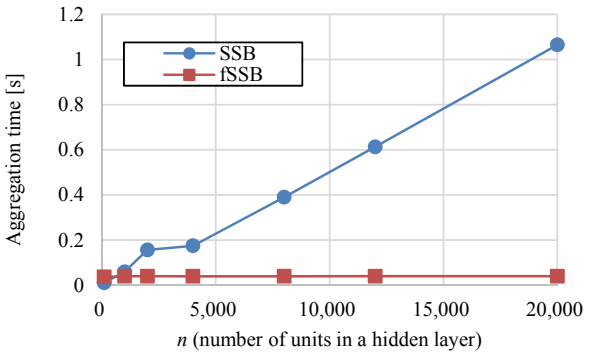


Figure 5. Feature aggregation time per 3D model.

Fig. 6 plots retrieval accuracies of the fSSB aggregation against the number of landmark local features L. Fig. 6 also includes plots of retrieval accuracies of the SSB aggregation as the base line. We can observe that the fSSB is able to approximate the SSB well if a sufficient number (e.g., more than 200k) of landmark local features are used. Using L=250k landmark local features, the fSSB yields retrieval accuracies almost equal to those of the SSB aggregation for all the three local features in Fig. 6.

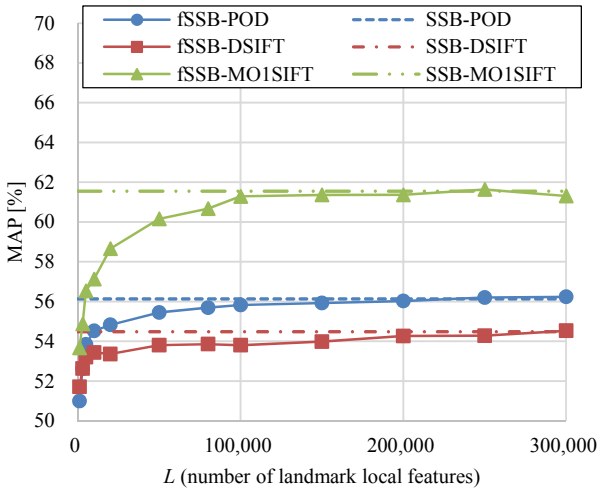


Figure 6. Number of landmark features and retrieval accuracy (PSB).

3) *Comparison with Other Feature Aggregation Algorithms*

a) *Retrieval Accuracy*

Fig. 7 compares retrieval accuracies of eight feature aggregation algorithms including the SSB and the fSSB by using the POD feature. Horizontal axis of Fig. 7 is the dimensionality of aggregated feature vectors. The maximum dimensionality is set at 10k for we wanted to contain computational cost of feature comparison.

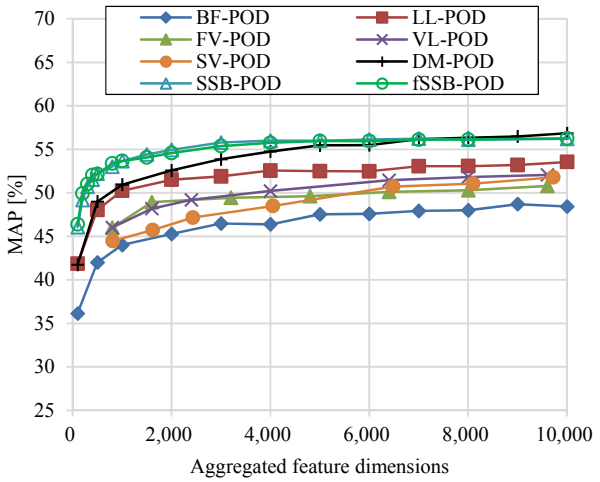


Figure 7. Aggregated feature dimensions and retrieval accuracy (PSB).

Fig. 7 shows that the proposed feature aggregation algorithms, i.e., the SSB and the fSSB, significantly outperform most of the six previous feature aggregation algorithms we have compared against. Among the six, only the DM aggregation seems to compete with the SSB and the fSSB if aggregated feature dimension n is larger than 5k. If the comparison is made for a smaller aggregated feature dimensionality, the SSB and the

fSSB do better than the others. We speculate that the joint optimization of codebook learning and feature encoding using *k*SA contributed to the high retrieval accuracy of the SSB and the fSSB. In comparison, "greedy" approach to codebook learning and feature encoding, which are done independently of each other, has limited the accuracies of the six previous feature aggregation algorithms.

The experiment shown in Figure 7 uses the PSB benchmark and the POD feature. To assess generality of the proposed algorithms against datasets and features, we conducted a comprehensive set of experiments by using 4 benchmarks, 7 feature aggregation algorithms including the fSSB, and 6 local features. The results for the PSB, ESB, SH11NR, and SH14LC benchmarks are summarized in Table I, II, III, and IV. We set the number of dimensions for the aggregated feature vectors to roughly 8k for all the feature aggregation algorithms in these tables. We can observe that the success of the fSSB aggregation depends on local features to be aggregated. For most of the benchmarks, for the features POD, MO1SIFT, and RoPS, the fSSB yields equal or better retrieval accuracies than the state-of-the-art feature aggregation algorithms including LL, FV, VL, SV, and DM. On the other hand, for the features DSIFT, LSF, and SI, retrieval accuracies of the fSSB are less than those of the DM aggregation, which is arguably the most accurate feature aggregation algorithms for 3DMR [19]. However, overall, the fSSB performs about as well or better than the state-of-the-art feature aggregation algorithms despite the fact that the fSSB uses compact sparse binary representation for its feature encoding.

TABLE I. MAP [%] FOR PSB DATASET.

algorithms	POD	DSIFT	MO1SIFT	LSF	SI	RoPS
BF	48.0	51.0	55.5	33.0	38.1	40.5
LL	53.1	57.6	56.5	33.8	41.0	46.4
FV	50.3	56.4	44.8	33.4	40.8	44.3
VL	51.8	56.3	43.5	32.7	41.4	42.6
SV	51.1	54.2	47.6	31.2	42.1	41.5
DM	**56.3**	**59.1**	56.9	**39.0**	**43.1**	47.9
fSSB	56.1	54.5	**61.6**	33.8	35.0	**50.5**

TABLE II. MAP [%] FOR ESB DATASET.

algorithms	POD	DSIFT	MO1SIFT	LSF	SI	RoPS
BF	52.1	53.6	49.1	47.1	48.3	47.2
LL	53.0	56.0	53.8	50.7	50.9	49.1
FV	53.8	56.0	53.1	49.8	52.3	47.9
VL	52.7	54.5	49.7	50.1	51.3	47.4
SV	55.0	54.9	51.1	49.3	53.9	46.4
DM	55.6	**56.5**	56.4	**55.2**	**54.9**	49.3
fSSB	**57.1**	55.2	**59.5**	52.2	51.2	**51.7**

TABLE III. MAP [%] FOR SH11NR DATASET

algorithms	POD	DSIFT	MO1SIFT	LSF	SI	RoPS
BF	87.3	94.3	75.7	87.5	85.8	89.0
LL	94.7	**97.0**	82.4	93.8	94.2	94.2
FV	95.2	95.1	70.2	88.3	92.2	94.9
VL	94.9	95.7	71.8	**94.2**	94.2	94.3
SV	95.6	95.7	71.7	92.4	94.0	**95.5**
DM	**96.6**	96.2	80.1	**94.2**	**94.7**	94.9
fSSB	93.2	96.0	**86.4**	90.9	91.3	90.2

TABLE IV. MAP [%] FOR SH14LC DATASET

algorithms	POD	DSIFT	MO1SIFT	LSF	SI	RoPS
BF	39.6	38.5	36.5	30.9	32.7	35.8
LL	44.7	**41.7**	39.4	31.4	34.6	39.2
FV	43.3	40.5	32.6	31.9	34.9	37.8
VL	43.9	39.8	31.3	30.9	34.8	36.0
SV	43.8	39.8	33.2	31.5	36.0	36.8
DM	**47.4**	41.5	40.1	**36.2**	**36.5**	40.7
fSSB	47.1	38.2	**41.6**	31.5	31.4	**42.1**

b) Temporal Cost and Spatial Cost

Table V compares efficiency of several feature aggregation algorithms both in terms of computation time and memory footprint. We used the PSB dataset and POD feature for the experiments. In table V, column "Pre-processing" indicates a computation time for codebook learning. The fSSB also includes times for encoding landmark local features and building a tree-structured index in the pre-processing step. "Feature aggregation / 3D model" is a time for aggregating a set of 3k POD features extracted from a 3D model. "Codebook" is memory footprint for storing the codebook. For SSB, a codebook is equivalent to a trained *k*SA with n=8k hidden units. For fSSB, the codebook is the set of 250k landmark local features and its tree-structured index. "Encoded features / 3D model" is memory usage for encoded local features *before pooling*, while "Aggregated feature / 3D model" is memory footprint for a pooled (aggregated) feature vector per 3D model.

Table V shows that the proposed *k*SA-based feature aggregation algorithms, i.e., SSB and fSSB, takes longer time than LL and SV for their pre-processing. Pre-processing time of the SSB and fSSB is mostly dominated by training of the *k*SA. As for the feature aggregation step, the fSSB aggregates faster than LL and SSB.

The fSSB requires the largest memory footprint for its codebook among all the algorithms listed in Table V, as its "codebook" include tree-structured index. However, in practice, memory footprint of about 70Mbyte is acceptable as it easily fits on memory of an ordinary PCs.

One of the advantages for the proposed algorithms is compactness of the encoded local features. Since the SSB and fSSB generates *highly sparse*, *binary* representation of local features, they require only 0.04Mbyte (40kbyte) to store 3k encoded local features of a 3D model. Compared to the LL, whose *k*-sparseness is set to k=15, memory consumption of encoded local features for the SSB and fSSB is only about one-tenth. Compactness of the encoded local features is required in certain applications, e.g., part-based 3DMR, in which an encoded local features need to be kept and reused many times to generate aggregated features for a large number of possibly overlapping sub-regions. Memory usage for the aggregated feature is almost the same among the aggregation algorithms since we set the number of aggregated feature dimensions to roughly 8k.

Table VI shows computation time per query using the proposed SSB and fSSB aggregation. We used the SH14LC since it contains the largest number (i.e., 8,987) of 3D models among the four benchmark datasets we have used in the experiments. In the table, column "Feat." is a computation time

for extracting 3k POD features from a query 3D model, "Agg." is a time for aggregating the set of 3k POD features extracted from the query, and "Sim." indicates a time for computing similarities among the aggregated feature of the query and the aggregated features of the 3D models in the database. Clearly, in this case, feature aggregation time of the SSB (0.39s) is a bottleneck for retrieval. This bottleneck can be removed by the fSSB aggregation, which employs tree-structured index of landmark local features for faster encoding, with no loss of retrieval accuracy as we showed in Section IV-B-2.

TABLE V. COMPARISON OF TEMPORAL AND SPATIAL COST.

algorithms	Computation time [s]		Memory footprint [Mbyte]		
	Pre-processing	Feature aggregation / 3D model	Codebook	Encoded features / 3D model	Aggregated feature / 3D model
LL	74.77	0.11	2.56	0.36	0.03
SV	9.16	0.01	0.06	97.20	0.03
SSB	649.24	0.39	2.10	0.04	0.03
fSSB	694.53	0.04	71.02	0.04	0.03

TABLE VI. COMPUTATION TIME PER QUERY FOR SH14LC DATASET.

algorithms	Feat.	Agg.	Sim.	Total
SSB	0.15	0.39	0.02	0.56
fSSB	0.15	0.04	0.02	0.21

V. CONCLUSION AND FUTURE WORK

In this paper, for efficient and accurate aggregation of local features, we proposed the *Sum of Sparse Binary codes* (*SSB*) aggregation and its faster-to-compute approximation *fast SSB* (*fSSB*) aggregation. They employ very compact, sparse, binary representation to encode local features. Aggregation, or pooling, of the encoded features is efficiently performed by summing. Accurate feature encoding of the proposed algorithms is achieved by using *k*-Sparse Autoencoder [3] and Angular Quantization-based Binary Codes algorithm [23]. Experimental evaluations using a whole-based 3D model retrieval scenario demonstrated that both SSB and fSSB is memory-efficient. In addition, their retrieval accuracies are comparable, or sometimes superior, to the existing feature aggregation algorithms including Fisher Vector coding [7], Super Vector coding [22], and Diffusion-on-Manifold aggregation [19]. The fSSB accelerates the SSB aggregation with almost no loss of retrieval accuracy.

As a future work, we will evaluate the proposed algorithms under part-based 3DMR setting that puts the compact sparse binary encoding and efficient sum-pooling of the proposed feature aggregation algorithms to good use.

ACKNOWLEDGMENT

This research is supported by *JSPS Grants-in-Aid for Scientific Research* (C) #26330133.

REFERENCES

[1] A.E. Johnson and M. Hebert "Using spin images for efficient object recognition in cluttered 3D scenes," *PAMI*, **21**(5), pp.433–449, 1999.

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

[3] A. Makhzani and B. Frey, "*k*-Sparse Autoencoders," arXiv:1312.5663, 2012.

[4] B. Li et al., "Large Scale Comprehensive 3D Shape Retrieval," *Proc. EG 3DOR 2014*, pp.131–140, 2014.

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

[6] D. Galvez-Lopez and J.D. Tardos, "Real-time loop detection with bags of binary words," *Proc. IROS 2011*, pp. 51–58, 2011.

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

[8] G. Csurka et al., "Visual Categorization with Bags of Keypoints," *Proc. ECCV 2004 workshop on Statistical Learning in Computer Vision*, pp.59–74, 2004.

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

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

[11] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," *Proc. CVPR 2009*, pp.794–1801, 2009.

[12] J. Wang et al., "Locality-constrained Linear Coding for Image Classification," *Proc. CVPR 2010*, pp.3360–3367, 2010.

[13] L. Liu, L. Wang, and X. Liu, "In defense of soft-assignment coding," *Proc. ICCV 2011*, pp.2486–2493, 2011.

[14] P. Shilane, P. Min, M. Kazhdan, and T. Funkhouser, "The Princeton Shape Benchmark," *Proc. SMI 2004*, pp.167–178, 2004.

[15] S. Jayanti, Y. Kalyanaraman, N. Iyer, and K. Ramani, "Developing an engineering shape benchmark for CAD models," *Proc CAD*, **38**(9), pp.939–953, 2006.

[16] S.M. Prakhya, L. Bingbing, and L. Weisi, "B-SHOT: A binary feature descriptor for fast and efficient keypoint matching on 3D point clouds," *Proc. IROS 2015*, pp.1929–1934, 2015.

[17] T. Furuya and R. Ohbuchi, "Dense sampling and fast encoding for 3D model retrieval using bag-of-visual features," *Proc. ACM CIVR 2009*, Article No. 26, 2009.

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

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

[20] T. Furuya, S. Kurabe, and R. Ohbuchi, "Randomized sub-volume partitioning for part-based 3D model retrieval," *Proc. EG3DOR 2015*, pp. 15–22, 2015.

[21] T. Matsuda, T. Furuya, and R. Ohbuchi, "Lightweight binary voxel shape features for 3D data matching and retrieval," *Proc. BigMM 2015*, pp.20–22, 2015.

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

[23] Y. Gong, S. Kumar, V. Verma, and S. Lazebnik, "Angular quantization-based binary codes for fast similarity search," *Proc. NIPS 2012*, 2012.

[24] Y. Guo et al., "Rotational Projection Statistics for 3D Local Surface Description and Object Recognition," *IJCV*, **105**(1), pp.63–86, 2013.

[25] Y. Huang, Z. Wu, L. Wang, and T. Tan, "Feature Coding in Image Classification, A Comprehensive Study," *PAMI*, **36**(3), pp.493–506, 2014.

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

[27] Y. Uchida and S. Shigeyuki, "Image Retrieval with Fisher Vectors of Binary Features," *Proc. ACPR 2013*, pp.23–28, 2015.

[28] Z. Lian et al., "SHREC'11 Track: Shape Retrieval on Non-rigid 3D Watertight Meshes," *Proc. EG 3DOR 2011*, pp.79–88, 2011