Scalar Quantization for Large Scale Image Search

Wengang Zhou[1], Yijuan Lu[2], Houqiang Li[3], Qi Tian[1]
Dept. of Computer Science, University of Texas at San Antonio[1], Texas, TX 78249
Dept. of Computer Science, Texas State University[2], Texas, TX 78666
Dept. of EEIS, University of Science and Technology of China[3], Hefei, P.R. China
wengang.zhou@utsa.edu[1], yl12@txstate.edu[2], lihq@ustc.edu.cn[3], qitian@cs.utsa.edu[1]

ABSTRACT
Bag-of-Words (BoW) model based on SIFT has been widely used in large scale image retrieval applications. Feature quantization plays a crucial role in BoW model, which generates visual words from the high dimensional SIFT features, so as to adapt to the inverted file structure for indexing. Traditional feature quantization approaches suffer several problems: 1) high computational cost—visual words generation (codebook construction) is time consuming especially with large amount of features; 2) limited reliability—different collections of images may produce totally different codebooks and quantization error is hard to be controlled; 3) update inefficiency—once the codebook is constructed, it is not easy to be updated. In this paper, a novel feature quantization algorithm, *scalar quantization*, is proposed. With scalar quantization, a SIFT feature is quantized to a descriptive and discriminative bit-vector, of which the first tens of bits are taken out as *code word*. Our quantizer is independent of collections of images. In addition, the result of scalar quantization naturally lends itself to adapt to the classic inverted file structure for image indexing. Moreover, the quantization error can be flexibly reduced and controlled by efficiently enumerating nearest neighbors of code words.

The performance of scalar quantization has been evaluated in partial-duplicate Web image search on a database of one million images. Experiments reveal that the proposed scalar quantization achieves a relatively 42% improvement in mean average precision over the baseline (hierarchical visual vocabulary tree approach), and also outperforms the state-of-the-art Hamming Embedding approach and soft assignment method.

Categories and Subject Descriptors
I.2.10 [**Vision and Scene Understanding**]: VISION

General Terms
Algorithms, Experimentation, Verification

Keywords
Large-scale image retrieval, scalar quantization, SIFT.

1. INTRODUCTION
The last decade has witnessed the great advance in content-based image retrieval on large-scale database. Most state-of-the-art approaches utilize SIFT features [1] to represent images and leverage the BoW model [2] to index large-scale image dataset for scalable retrieval. Some post-processing techniques, such as spatial verification [3] [4] and query expansion [5] [24], are also explored to further boost the retrieval accuracy. Of them, one of the key steps is feature quantization, which first generates visual words from the high dimensional SIFT features, and then quantize features to the corresponding visual words for indexing.

The most popular feature quantization method is vector quantization. Originally used in lossy data compression, vector quantization divides a large set of training SIFT features into groups. Each group corresponds to a sub-space in the feature space, and is represented by its center, which is called visual word [2]. All visual words constitute a visual codebook. Then, given a novel feature, vector quantization assigns it the visual word ID of the sub-space where the feature falls in. The most popular visual codebook generation approach is *k*-means [2] clustering. When the visual codebook size becomes very large (e.g. 1 million), it is infeasible to train the codebook with *k*-means, and hierarchical *k*-means [6] is more preferred to improve codebook generation speed and enhance feature quantization efficiency.

Traditional vector quantization suffers several problems. 1) **High computational cost**: visual codebook generation is computationally expensive especially with a large amount of features. For example, in order to train a large visual codebook containing 1 million visual words, usually about 50 million SIFT features may need, considering both feature coverage and affordable memory size. However, for the SIFT descriptor space with as large as 128 dimensions, it is still unknown whether 50 million SIFT features are enough to capture the feature distribution. Even if the memory would afford several orders of magnitude more training features, it would take intolerable time cost to finish the clustering for codebook generation. 2) **Limited reliability**: codebook construction in vector quantization relies on the collection of image features and codebook generation methods. Different collections of image features may produce totally different codebooks. Even with the same collection of images and the same clustering methods, generated codebook may be still different due to the variability of *k*-means. Therefore, quantization error is hard to be controlled. 3) **Update inefficiency**: with many new features collected, the codebook/quantizer should be updated accordingly. However, the codebook updating needs lots of effort. The huge amount of features have to be re-clustered, which is computationally inefficient.

To address the above problems, in this paper, a novel quantization strategy, *scalar quantization*, is proposed. Distinguished from the traditional vector quantization methods, the proposed scalar quantization approach does not involve any form of visual codebook training or clustering. Instead, it transforms each feature to a bit-vector with a quantizer, which is independent of collections of image features. Our quantization operation is very simple and requires low computational cost. The bit-vector

MM'12, October 29–November 2, 2012, Nara, Japan.

generated by scalar quantization achieves more compact representation of the original SIFT descriptors, but still keeps the discriminative power of SIFT feature. Since our quantization method is independent of collections of images, even with new collected features, there is no need to update our quantizer.

Moreover, scalar quantization can index features to the classic inverted file structure easily by extracting the first tens of bits from the quantized bit-vector to generate code word. And the remaining bits of the quantized bit-vector are stored in the inverted file list for matching verification. Furthermore, a novel soft quantization strategy is applied in scalar quantization to address the quantization loss by enumerating the nearest neighbors of the code word. Consequently, more candidate matches are included for matching verification, which greatly boost the retrieval accuracy in large-scale image database.

In this paper, we only focus on the feature quantization step. To further improve the image retrieval performance, our approach can also be flexibly integrated with many other algorithms, such as weak geometric consistency [15], fast spatial matching [3], geometric verification [4] [17], and query expansion [5], *etc*.

To summarize, the main contributions of this paper lie in three aspects:

(1) We propose a new scalar quantization method to quantize the SIFT descriptor to a compact bit-vector, with the discriminative power kept. No visual codebook is needed to be trained in our quantization scheme.

(2) We adapt the quantized bit-vectors to the popular inverted index file structure for scalable image search.

(3) We propose a soft quantization scheme based on our indexing structure to reduce the quantization error.

The rest of the paper is organized as follows. Section 2 reviews related work in large-scale image search. Section 3 discusses the proposed algorithm in details. Experimental results are given in Section 4. Section 5 makes discussions on three issues. Finally, the conclusion is provided in Section 6.

2. RELATED WORK

In large-scale content-based image search applications, Bag-of-Words (BoW) model based on local features has been widely adopted. Generally, in those BoW-based approaches, there are four major key components: local feature representation, feature quantization, index strategy, and post-processing. In this section, we make a review of related work in each component.

Local Feature Representation Extraction of local feature usually involves two steps, *i.e.* interest point detection and feature description. The detected interest points are expected to have high repeatability over various changes. Popular detectors include Difference of Gaussian (DoG) [1], MSER [7], and Hessian affine [8]. After interest point detection, a descriptor is extracted to represent the visual appearance of the local region centered at the interest point. Usually, the descriptor should be invariant to rotation and scale, and also robust to affine distortion, addition of noise, and illumination changes, *etc*. The most popular choice with the above merits is SFIT feature [1]. As a variation, SURF [9] demonstrates good performance but achieves better efficiency. Recently, a binary feature BRIEF [10] and its variation ORB [11] have been proposed and attracted lots of attention.

Feature Quantization Usually, several hundred or thousand local features are extracted from a single image. To achieve a compact representation, high-dimensional local features are quantized to visual words, and an image can be represented as a "bag" of visual words. Therefore, a visual codebook containing visual words needs to be generated first. The most intuitive visual codebook generation method is *k*-means [2] or hierarchical *k*-means [6] for large size visual codebook generation.

With visual codebook defined, feature quantization is to assign a visual word ID to each feature. The most naive choice is finding the closest (the most similar) visual word of a given feature by linear scan, which, however, suffers expensive computational cost. Usually, approximate nearest neighbor (ANN) search methods are adopted to speed up the searching process, with sacrifice of accuracy to some extent. In [1], a *k*-d tree [21] is utilized with a best-bin-first modification to find approximate nearest neighbors to the descriptor vector of the query. In [6], based on hierarchical vocabulary tree, an efficient approximate nearest neighbor search is achieved by propagating the query feature vector from the root node down the tree by comparing the corresponding child nodes and choosing the closest one. In [12], a *k*-d forest approximation algorithm is proposed with reduced time complexity. To reduce the quantization loss, a descriptor-dependent soft assignment scheme [13] is proposed to map a feature vector to a weighted combination of several visual words. In [14], the high dimensional SIFT descriptor space is partitioned into regular lattices. Although demonstrated to work well in image classification, in [13], regular lattice quantization is reported working significant worse than [6][13] in large-scale image search. In [27], a novel scheme is proposed to jointly optimize the dimension reduction and indexing. In [28], a compact image signature, called Residual Enhanced Visual Vector, is designed via quantization residue aggregation and classification-aware dimensionality reduction. In [29] and [30], descriptive and contextual visual vocabularies are generated respectively for large-scale image applications, such as image search.

In [15], for each feature quantized to a visual word, feature dimension is further performed and a binary signature is generated with a pre-trained median vector. Such binary signature will be used for feature matching verification in on-line retrieval. In [26], a variation of Hamming Embedding [15], *i.e.*, the Asymmetric Hamming Embedding scheme, is proposed to better exploit the information conveyed by the binary signature.

Index Strategy Inspired by the success of text search engines, inverted file structure [23] has been successfully used for large-scale image search [2][3][4][5][6][13][15][16]. In essence, inverted file structure is a compact representation of a sparse matrix, whose row and column denote visual word and image, respectively. In on-line retrieval, only those images sharing common visual words with the query image need to be checked. Therefore, the number of candidate images to be compared is greatly reduced, achieving efficient response.

In inverted file structure, each visual word is followed by an inverted file list of entries. Each entry stores the ID of image where the visual word appears, and some other clues for verification or similarity measurement. For instance, Hamming Embedding [15] generates a 64-bit Hamming code for each feature to verify descriptor matching. Bundled Feature [16] stores the *x*-order and *y*-order of each SIFT feature located in the bundled area. The geometric clues, such as feature position, scale, and orientation, are also stored in inverted file list for verification of geometric consistency [3][4][15][16][17].

To further reduce the memory cost of inverted file structure, a visual word vector is mapped to a low-dimensional representation by a group of min-hash functions [18] [19]. Consequently, only a small constant amount of data per image needs to be stored.

Post Processing The initially returned result list can be further refined by exploring the spatial context or enhancing the original query. Spatial verification [3] [4] [15] [17] [19] and query expansion [5] [24] are two of the most successful post-processing techniques to boost the accuracy of large-scale image search.

Spatial context is an important clue to remove false positive visual matches. Lots of work has been done on spatial verification. In [2], locally spatial consistency is imposed to filter visual-word matches with low support. In [15], weak geometric consistency on scale and orientation is imposed to quickly filter potential false matches. In [3], global spatial verification is performed based on a variation of RANSAC [20]. An affine model is estimated to filter local matches that fail to fit the model. In [4] [17], the geometric context among local features is encoded into binary maps. And then it recursively removes geometrically inconsistent matches by analyzing those coding maps.

Query expansion, leveraged from text retrieval, reissues the initial highly-ranked results to generate new queries. Some relevant features, which are not present in the original query, can be used to enrich the original query to further improve the recall performance. Several strategies, such as average query expansion, transitive closure expansion, recursive expansion, intra-expansion, and inter-expansion, *etc*. have been discussed in [5] [24].

3. METHOD

In this paper, we focus on feature quantization, which plays a key role of BoW model. We first introduce our scalar quantization strategy in Section 3.1. Then, in Section 3.2, we discuss how to adapt the scalar quantization result to the classic inverted file structure for scalable image search. In Section 3.3, we discuss a soft quantization scheme to further reduce the quantization error in on-line query stage. Finally, a summary of our quantization algorithm is given in Section 3.4.

3.1 Scalar Quantization

High dimensional SIFT descriptors (L_2-normalized 128-D vectors [1]) are extracted from images for discrimination. Each dimension of the descriptor vector corresponds to a bin of concatenated orientation histograms. Generally, similar SIFT features have relatively smaller distances than different features. Features from the same source, e.g. image patch, may not be exactly same due to image noise. But their values on the 128 bins usually share some common patterns, e.g., the pair-wise differences between most of bins are similar and stable. Therefore, it can be easily extended that the differences between bins and a predefined threshold are stable for most bins. Based on such observation, we propose a scalar quantization strategy.

Given a high dimensional feature vector $f = (f_1, f_2, \cdots, f_d)^T \in R^d$, where $f_i \in R, (i = 1,2,\cdots,d)$, and d denotes the feature dimension size. For SIFT descriptor, $d = 128$. We define a quantization function $q(\cdot)$ to transform f to a bit vector $b = (b_1, b_2, \cdots, b_d)^T$, as follows:

$$b_i = \begin{cases} 1 & \text{if } f_i > \hat{f} \\ 0 & \text{if } f_i \le \hat{f} \end{cases} \qquad (i = 1,2,\cdots,d) \qquad (1)$$

where $\hat{f}$ is a threshold determined by vector f.

The threshold $\hat{f}$ is an important parameter, which determines the discriminative power of the quantization results. If the discriminative power of SIFT is well kept in scalar quantization, the Hamming distance between scalar vectors b should be consistent with the L_2 distance between original feature vectors f. There are many methods to choose the threshold $\hat{f}$. In this paper, we choose $\hat{f}$ as the median value of vector f. The philosophy behind it is that, the median value is relatively stable to changes in some bins of a long vector. The quantization function $q(\cdot)$ is a kind of hashing. Unlike classic LSH methods involving many hashing tables and functions [8] [19], our scheme needs only one hash function and therefore is much simpler and more efficient.

With each high dimension feature quantized to a bit-stream vector, the feature comparison is transformed to the comparison of binary vectors, which can be efficiently accomplished by exclusive-OR operation and measured by Hamming distance.

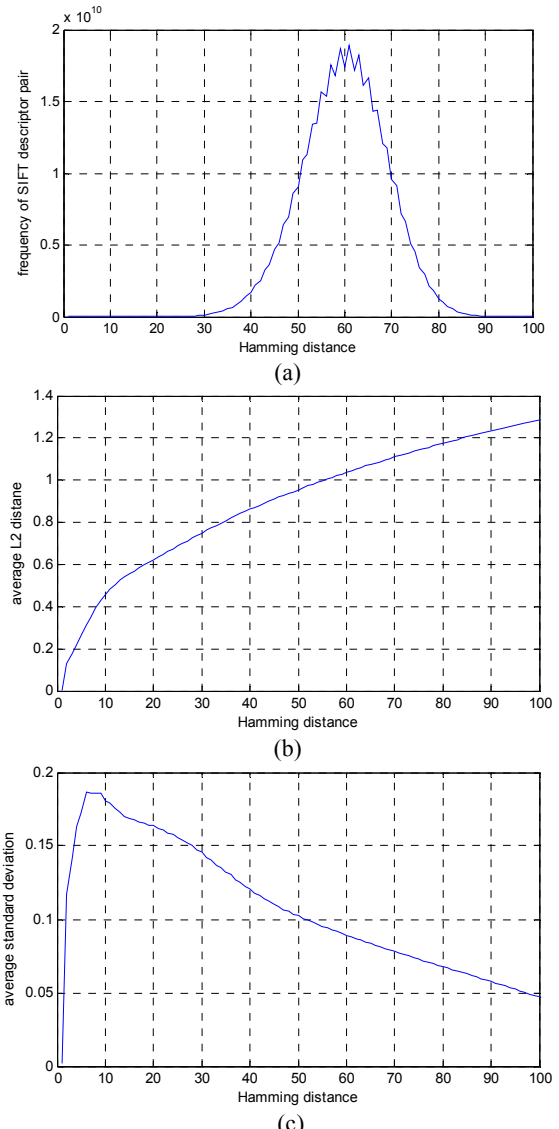
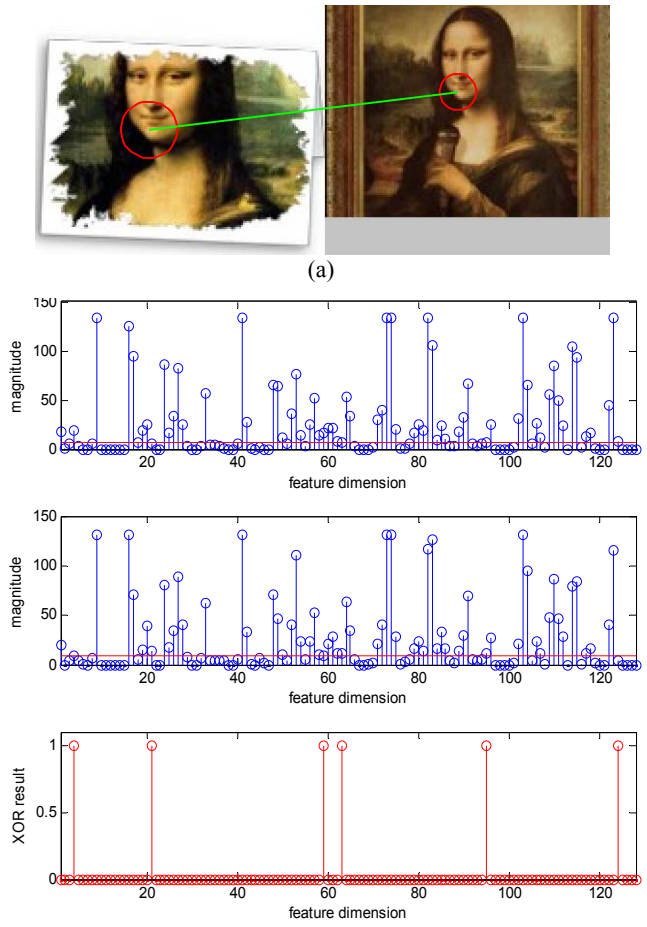


Figure 1. The statistics on 4.08×10^{11} pairs of SIFT descriptors. (a) Descriptor pair frequency vs. Hamming distance; (b) The average L_2-distance vs. Hamming distance; (c) The average standard deviation vs. Hamming distance.

To demonstrate the discriminative power of SIFT descriptors is well kept in our scalar quantization, we have made a statistical study on 4.08×10^{11} SIFT descriptor pairs, which include every SIFT pair extracted from image pairs randomly sampled from a large image dataset. For each descriptor pair, its L_2 distance before scalar quantization and Hamming distance after scalar quantization are calculated. As shown in Fig. 1(a), the distribution of Hamming distances between these descriptors exhibits a Gaussian-like distribution. From Fig. 1 (b) and Fig. 1(c), it is observed that the Hamming distance between our quantized bit-vectors is consistent with the average L_2-distance, with relatively small standard deviation (computed on the unit-normalized descriptors). To further reduce the deviation, we use a variation of Eq. (1) and transform the descriptor vector to a 256-bit vector, which will be discussed at the end of this section.



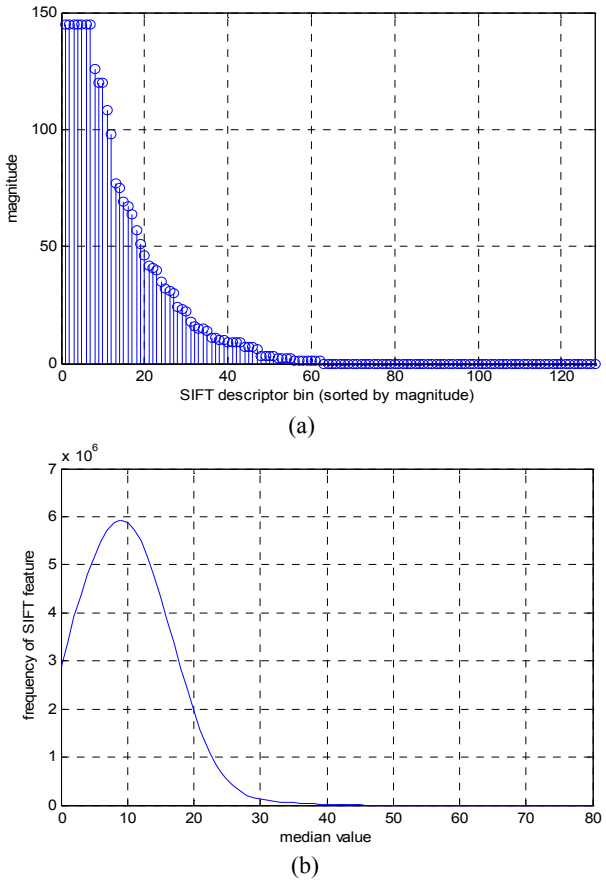
(a)

(b)

Figure 2. Example of feature matches. (a) A local match between two images. The endpoints of the green line denote the key point positions of two SIFT features. The radius of the red circle centered at the key points is proportional to the SIFT feature's characteristic scale. (b) top: the 128-D descriptor of the matched SIFT feature in the left image; middle: the 128-D descriptor of the matched SIFT feature in the right image; bottom: the XOR result of the binary SIFT features from the two matched SIFT features. The red horizontal lines in the "top" and "bottom" figure denote the median values of the two SIFT descriptors, respectively.

It should be noted that our approach is different from the SIFT quantization methods proposed in lattice quantization [14] and Hamming Embedding [15]. In [14], the descriptor space is arbitrarily split along dimension axes into regular lattice. In [15], for each bin/dimension, a median value of all training features on that bin in the reduced dimensional space is computed for binarizing the corresponding dimension. Both two approaches ignore the unique property of every individual SIFT descriptor.

As shown in Fig. 1, the original features' difference in Euclidean distance can be well captured by their Hamming distance after scalar quantization. Fig. 2 shows a real instance of local descriptor match across two images with scalar quantization. From Fig. 2(b), it can be observed that these two SIFT descriptors have similar magnitude in the corresponding bins with some small variations before quantization. After scalar quantization, they differ from each other in six bins. With a proper threshold, it can be easily determined whether the local match is true or false just by the exclusive-OR (XOR) operation between the quantized bit-vectors. Obviously, the error in the exclusive-OR result is likely to occur in those bins with magnitude around the median value. Intuitively, the median threshold could be increased to some upper level, which can make the Hamming distance between similar SIFT descriptors smaller. However, such modification will also reduce the Hamming distance between irrelevant descriptors and cause false descriptor matches.



(a)

(b)

Figure 3. Statistics of SIFT descriptors. (a) A typical SIFT descriptor with bins sorted by magnitude in each dimension; (b) The frequency distribution of median value of the descriptor vector among 100 million SIFT descriptors.

Another statistical study on the distribution of median value of SIFT descriptor is also performed. 100 million SIFT descriptors are sampled from a large dataset, and the median value of each 128-D descriptor vector is computed. As shown in Fig. 3, the median value of most SIFT descriptors is relatively small, around 10, but the maximum magnitude in some bins still can reach more than 140. This may incur potential quantization loss since those bins with magnitude above the median are not well distinguished. To address this issue, the same scalar quantization strategy could be conducted again on those bins with magnitude above the median. Intuitively, such operation can be performed recursively. However, it will cause additional storage cost. In our implementation, we only perform the scalar quantization twice, *i.e.*, first on the whole 128 elements, and second on those elements with magnitude above the median value. Consequently, a SIFT descriptor $f = (f_1, f_2, \cdots, f_{128})^T \in R^{128}$ is quantized to a 256-bit vector $\tilde{b} = (b_1, b_2, \cdots, b_{256})^T$, as follows:

$$(b_i, b_{i+128}) = \begin{cases} (1,\ 1) & \text{if } f_i > \hat{f}_2 \\ (1,\ 0) & \text{if } \hat{f}_1 < f_i \le \hat{f}_2 \\ (0,\ 0) & \text{if } f_i \le \hat{f}_1 \end{cases} \qquad (i = 1,2,\cdots,128) \qquad (2)$$

where $\hat{f}_1 = \frac{g_{64} + g_{65}}{2}, \hat{f}_2 = \frac{g_{32} + g_{33}}{2}$, $(g_1, g_2, \cdots, g_{128})$ is the sorted vector from $(f_1, f_2, \cdots, f_{128})$ in descending order. With Eq. (2), each dimension of SIFT descriptor is divided into three parts, and two bits are used to encode each part.

With scalar quantization by Eq. (2), the comparison of SIFT descriptors in L_2-distance is captured by the Hamming distance of the corresponding 256-bit vectors. Since our target is large-scale image search, how to adapt our scalar quantization result to the classic inverted file structure for scalable image search needs to be explored.

3.2 Indexing with Inverted File Structure

In image search, the problem of feature matching between images can be regarded as finding feature's nearest or approximate-nearest neighbors. When the feature amount becomes very large, say, over one billion, it is too computationally expensive to find the nearest neighbors by linearly comparing all features' binary vectors. To address this problem, leveraged from text retrieval, inverted file structure can be used for scalable indexing of large-scale image dataset.

In traditional inverted file structure for image search, a group of visual words are pre-trained. And each visual word is followed with an entry list of image features, which are quantized to this followed visual word. Each indexed feature in the list records its image ID and some other clues.

To adapt to the classic inverted file structure to index image features, we define *code word*[1] by the first t bits of the binary code generated by scalar quantization result. Then, the rest bits of features are recorded in the entry list of the corresponding code word. In fact, any other t bits of the binary code are expected to be equivalent. A toy example is shown in Fig. 4. Intuitively, if a code word is represented with t bits, the total number of code words could be amounted up to 2^t. However, it is found from experiments that, when t increases above 20, the amount of non-empty code words becomes much smaller than 2^t, as shown in Fig. 5. For example, when t increases to 32, the total number of code words could be up to $2^{32} \approx 4 \times 10^9$ (4 billion). However, the number of unique code words generated by scalar quantization (on one million image database) is even much less than 10^8.

Generally, the more code words are generated, the shorter the average length of indexed feature list becomes, and the less the time cost is needed to query a new feature. However, in our method, we will introduce a soft quantization scheme (Section 3.3) to expand more code words for each query feature. And the number of expanded indexed feature lists is polynomial to t. To make a tradeoff, in our experiments, we select $t = 32$, and 46.5 million "code words" are obtained.

Fig. 6 shows the distribution of code word occurrence on one million image database. It can be observed that, of the 46.5 million code words, only the top few thousand code words have very high frequency. Those code words are prevalent in many images, and their distinctive power is weak. As suggested by [2], we apply a stop-list to ignore those high frequency code words that occur in more than 0.11% of the total image dataset. Experiments reveal that a proper stop-list may not affect the search accuracy, but does avoid checking many code word lists and achieves gain in efficiency.

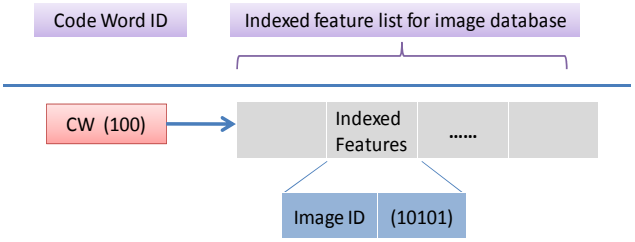


Figure 4. A toy example of image feature indexed with inverted file structure. The scalar quantization result of the indexed feature is an 8-bit vector (1001 0101). The first three bits denote its code word ID (100), and the remaining 5 bits (10101) are stored in the inverted file list.

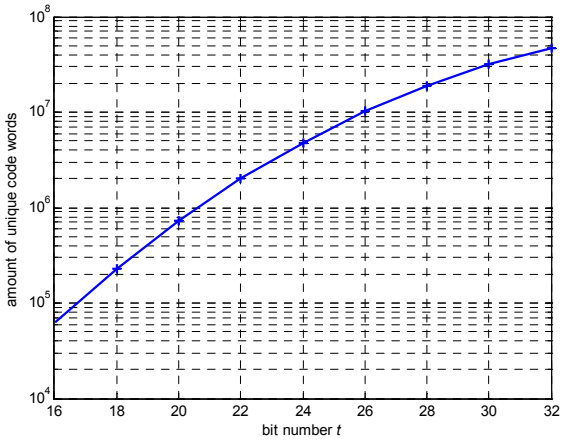


Figure 5. The amount of unique code words (top t bits from 256-bit vector) for different t on 1-million image database.

[1] It should be noted that our code word is different from the traditional visual word [2] [6].

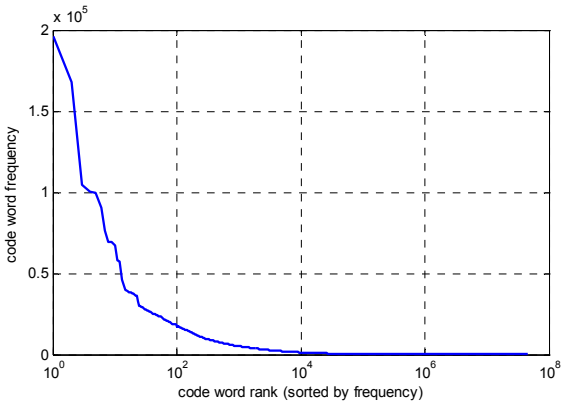


Figure 6. Frequency of code words among one million images.

Once all features of an image dataset have been indexed with the inverted file structure, given a new SIFT descriptor, it will be first quantized to a 256-bit vector with scalar quantization. Then through the top 32 bits, the corresponding code word can be located. And only the indexed features following the matched code word will be checked. Therefore, the searching space is greatly reduced. Finally, the exclusive-OR operation is performed on the remained 224 bits of the query vector and those of indexed features recorded in the entry list of the matched code word. A threshold κ on the Hamming distance between 256-bit vectors needs to be set for true-match judgment, such that those matches with Hamming distance no larger than κ will be accepted as true matches. The impact of κ will be studied in Section 4.1.

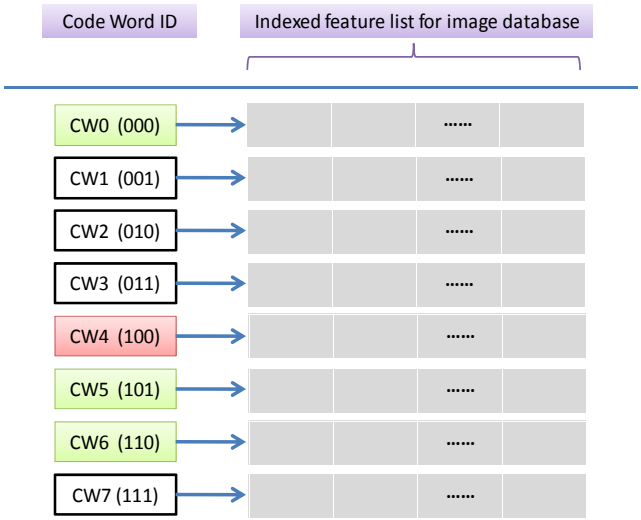


Figure 7. A toy example of soft quantization with bit-stream code words. There are eight code words, each represented with a three-bit vector. Each code word is followed by an indexed image feature list. (Best viewed in color PDF)

3.3 Reduction of Quantization Error

In Section 3.2, we define code word by the top 32 bits of the bit vector from scalar quantization. However, such simple processing will exclude some candidate features that have some flipping bits among the top 32 bits (e.g., 0 changes to 1) due to noise. To address this issue, we propose a soft strategy to reduce the quantization error. Assuming such flipping happens only to very few dimensions, features before and after the flipping should be still very similar, *i.e.*, small Hamming distance. To identify these candidate features, it is desired to quickly enumerate all of its possible nearest neighbors within a predefined Hamming distance d, just by alternatively flipping some bits. This is equivalent to a tolerant expansion of the original code word. The impact of expansion-bit number d will be studied in Section 4.1.

As shown in the toy example in Fig. 7, the code word of a new query feature is a bit-vector 100, *i.e.*, CW4 in pink color. To identify all of candidate features, its possible nearest neighbors (e.g., Hamming distance d=1) will be obtained by flipping one bit in turn, which generates three additional code words (in green color): CW0 (000), CW5 (101) and CW6 (110). These code words are nearest neighbors of CW4 in the Hamming space. Then, besides CW4, the indexed feature lists of these three expanded code words will be also considered as candidate true matches, and all features in these expanded lists will be further compared on their rest bit-codes.

Off-line Indexing with Scalar Quantization

1) Given a 128-D SIFT descriptor from an index image, convert it to a 256-bit vector by Eq. (2);
2) Identify code word ID V_d by the first 32 bits of the 256-bit vector.
3) In the inverted image list of V_d, store both the ID of the image where the feature appears, and the remaining 224 bits of the quantized 256-bit vector.

Figure 8. The general steps of off-line indexing with scalar quantization.

On-line Querying with Scalar Quantization

1) Given a 128-D SIFT descriptor in a query image, convert it to a 256-bit vector by Eq. (2);
2) Identify its code word ID V_q by the first 32 bits of the 256-bit vector.
3) For each feature in the inverted image list linked to V_q, compare its indexed 224-bit vector with the query feature. If the total Hamming distance in 256-bit is not greater than κ, accept the indexed feature as true match.
4) Expand the V_q to include its nearest code words $\{V_q^i, i=1,2,\cdots\}$ with Hamming distance no greater than d.
5) For each V_q^i, repeat step (3).

Figure 9. The general steps of on-line querying with scalar quantization.

3.4 Algorithm Summary

Overall, the proposed scalar quantization consists of two stages: offline indexing and on-line querying. In this section, we summarize the general steps of these two stages in Fig. 8 and Fig. 9, respectively. Given a query image, after looking up the index

file with the proposed on-line querying steps, we can obtain the matching results between the query image and target images. Finally, we formulate image retrieval as a voting problem and define the similarity between two images by the cardinality of matched feature set.

4. EXPERIMENTS

Our basic dataset is built by crawling one million images from the Web. We take the partial-duplicate image dataset released in [17] as the ground-truth dataset, which contains 1104 images from 33 groups, including "Mona Lisa", "KFC logo", "American Gothic Painting", "Seven-eleven logo", *etc*. To evaluate the performance with respect to the size of dataset, we construct three smaller datasets (50K, 200K, and 500K) by sampling the basic dataset. From the ground truth dataset, 108 representative query images are randomly selected for evaluation comparison. Similar to [3][14][16], mean average precision (mAP) is selected to evaluate the accuracy performance of all methods.

We use the standard SIFT feature [1] for image representation. Key points are detected with the Difference-of-Gaussian (DoG) detector. A 128-D orientation histogram (SIFT descriptor) is extracted to capture the visual appearance of local patch centered at each key point. Before feature extraction, large images are scaled to have a maximum axis size of 400.

4.1 Parameter Analysis

There are two parameters in our approach: Hamming distance threshold κ and expansion-bit number d. To study the impact of these two parameters on search performance and computational cost, we compare the mAP performance and average time cost per query under different parameter settings of κ and d on the 1-M image dataset. The results are shown in Fig. 10.

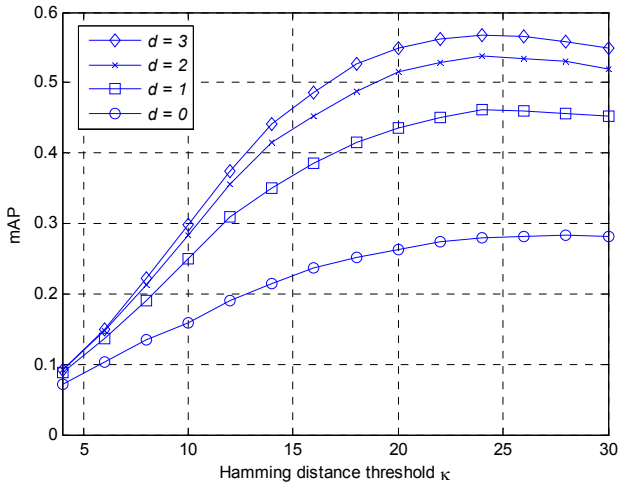
From Fig. 10(a), it can be observed that when the Hamming distance threshold κ increases, the mAP performance first increases and then keeps stable and gradually drops a little after it reaches the peak, where $\kappa = 24$. This is intuitive, since increasing κ always includes more candidate true matches, but when κ is too large, many noisy matches are also included and pollute the results. On the other hand, when expansion-bit number d increases, the mAP gradually increases. This is due to the fact that more candidate code word lists are involved in matching verification, and more true matches will be kept.

In terms of efficiency, as shown in Fig. 10(b), the average time cost per query increases when κ increases. This is due to that, when κ is larger, we have to make more exclusive-OR operations, until the Hamming distance between two 224-bit vectors is above a threshold. As d increases, the querying time cost rises significantly. This is because the expanded code word list number is exponential to the expansion-bit number d.

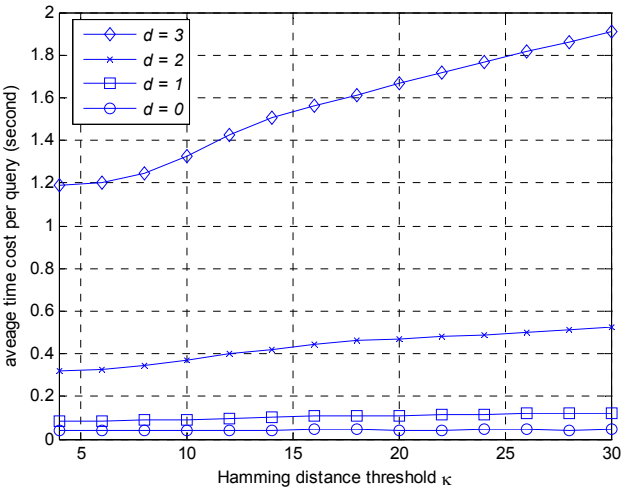
Considering the tradeoff between mAP performance and time cost, κ is set as 24 and d is set as 2 in the rest experiments.

4.2 Evaluation

Comparison Algorithms: We compare our approach with three state-of-the-art feature quantization algorithms in large-scale image search. The BoW approach with visual vocabulary tree [6] is selected as the "baseline" method. We test various sizes of visual word vocabulary, and the 1-million vocabulary gives the best overall performance. As suggested in [2][6], the stop-list strategy is also adopted to improve efficiency. To enhance the baseline, two other algorithms, *i.e.*, soft assignment [13] and Hamming embedding [15], are also compared.



(a)



(b)

Figure 10. (a)The mAP performance and (b) average time cost per query under different parameter settings of κ and d on the 1-million image dataset.

Soft assignment [13] identifies a local feature with a weighted combination of three nearby visual words. We use the default parameters as set in [13]. The "nearby" visual words for a given feature are found by the approximate nearest neighbor search algorithm *k*-d tree [21][25]. A public library for approximate nearest neighbor (ANN) searching [22] is used in our experiments. To make a tradeoff between accuracy and efficiency, we select the error bound parameter [22] as five.

Hamming embedding [15] generates additional Hamming codes (64 bits) to filter more candidate features which are quantized to the same visual word but have large hamming distance to the query feature. We denote this method as "HE". We have tested different thresholds for the Hamming distance in HE, and the best performance is achieved when the threshold is selected as 12. Since the focus of this paper is feature quantization, the weak geometric consistency scheme proposed in the Hamming embedding approach is not added in the experiments.

Accuracy: From Fig. 11, it can be observed that our approach outperforms all the other three methods on large image databases. On the 1-million dataset, The mAP of the baseline is 0.38. Our

approach hits 0.54, a relatively 42.1% improvement. Since Hamming codes can effectively filter false featurs, the Hamming Embdding approach achieves a mAP of 0.43, but still 11% lower than our approach. The mAP improvement of soft assignment approach is higher than HE. It reaches a mAP of 0.48. Compared with soft assignment, our approach still enjoys a relatively 12.5% improvement. Such improvement stems from the distance based thresholding for matching verification of our approach. It is interesting to note that, when the database size decreases to 50 K, our approach is worse than the soft assignment. This is due to our tradeoff selection on the expansion bit number and efficiency.

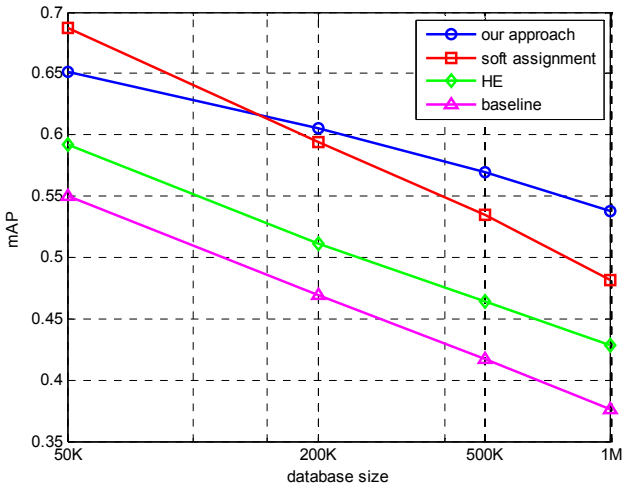


Figure 11. Performance (mAP) comparison of different methods with different database sizes.

Efficiency: The experiments are performed on a server with 3.4 GHz CPU and 16 GB memory. We compare efficiency in both off-line indexing and on-line query. From Table 1, it can be observed that our approach is the most efficient one in indexing image features. It takes our approach 18.86 seconds to index one million SIFT features, which is 2 times, 2.5 times, and 40 times faster than the baseline, HE and soft assignment approach, respectively. Fig. 12 shows the average time cost per query of all four approaches. It should be noted that the time cost of SIFT feature extraction is not included for all approaches. It takes the baseline 0.12 second in average to perform one query. HE is the most time-efficient one and costs only 0.05 second to finish one query in average. Soft assignment is the most time-consuming approach, consuming 0.52 second in average per query. Although our approach costs more time than the baseline approach, it may still meet user's expectation of fast response time (average 0.48 second per query) but with much higher search accuracy. It is slightly more efficient than the soft assignment approach, with 0.04 second less in average per query.

It should be noted that a distinctive characteristic of our approach from other three comparison methods is that, no visual codebook needed to be trained before feature quantization, which could save a lot of computational time. As a contrast, all three comparison algorithms have to train a large visual codebook containing as many as one million visual words, which usually costs days of time. In order to train a visual codebook of one million in size, usually about 100 million SFIT descriptors are needed as training samples. However, even with so many training samples, it is still unclear whether these training samples are enough to generate desired visual words to capture the sample distribution in the so large 128-D descriptor space. Moreover, when more new features are indexed, it may be necessary to update the visual codebook accordingly (*e.g.*, re-cluster all the features), which is always time consuming and computationally expensive. On the contrary, our scalar quantization just needs to incrementally add new code words to the existing visual codebook (code word set).

Table 1. Time cost to index 1 million SIFT features for four approaches in off-line stage.

Method	baseline	HE	soft assignment	our approach
Time cost (second)	53.72	64.82	771.09	18.86

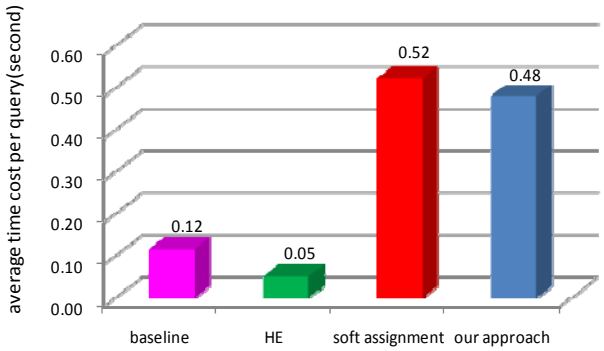


Figure 12. Comparison of average query time cost of different methods on the 1M database. (Not including the time cost for SIFT feature extraction)

Table 2. Memory cost for four approaches.

	Memory cost per indexed feature (byte)	Memory cost for quantizer (byte)
baseline	8	142M
HE	12	398M
soft assignment	24	506M
our approach	32	0

Memory Cost: We compare memory cost of all approaches on both indexed feature and quantizer, as listed in Table 2. In terms of memory cost per indexed feature, for each feature, the baseline approach needs 4 bytes to store image ID and another 4 bytes to store the *tf-idf* weight. The soft assignment has to store each indexed features in three visual word lists, therefore it costs 24 bytes, three times the memory cost of the baseline approach. In Hamming Embedding approach, it allocates 4 bytes on image ID and 8 bytes on the 64-bit Hamming code. Compared with the above three methods, our approach consumes more memory. It takes 4 bytes to store image ID and additional 28 bytes to store another 224 bits from quantization results.

Besides indexed feature, all the three comparison methods have to load a large quantizer into main memory. A hierarchical visual vocabulary tree (about 142M bytes) is required for both the baseline and HE. Besides, HE has to store a 64-D median vector (floating point values) for each leaf node. As for soft assignment approach, besides the visual words (leaf nodes of the vocabulary tree, 128M bytes), it also needs to generate a *k*-d tree (about 378M bytes)to quantize features. As a contrast, our approach needs no memory cost on quantizer.

4.3 Sample Results

In Fig. 13, a sample edited image "Apollo" is selected as a query to demonstrate the search performance of all four approaches on the 1-million dataset. For this query, compared with the baseline approach, our approach improves the mAP from 0.36 to 0.846, with relatively 135% improvement. Fig. 13 (c) and (d) show the top images returned by the baseline approach and our approach, respectively. Due to the poor quality of the query image, the returned results of the baseline are polluted by irrelevant images. As for our approach, more relevant images are ranked to the top.

Fig. 14 shows a difficult image sample on which our method works poorly. The query image is the first image with label rank-1 in Fig. 14(a). Since the query contains a large portion of patches containing rich text, the returned results are polluted by many screenshot images of documents or web pages. The matching result between the query and the fifth returned image is shown in Fig. 14(b). Although irrelevant in visual content, they still share many local similar patterns. Such query is also very challenging for the other three comparison methods. To filter such false positives, spatial verification algorithms [3] [17] could be combined with our scalar quantization to re-rank the results.

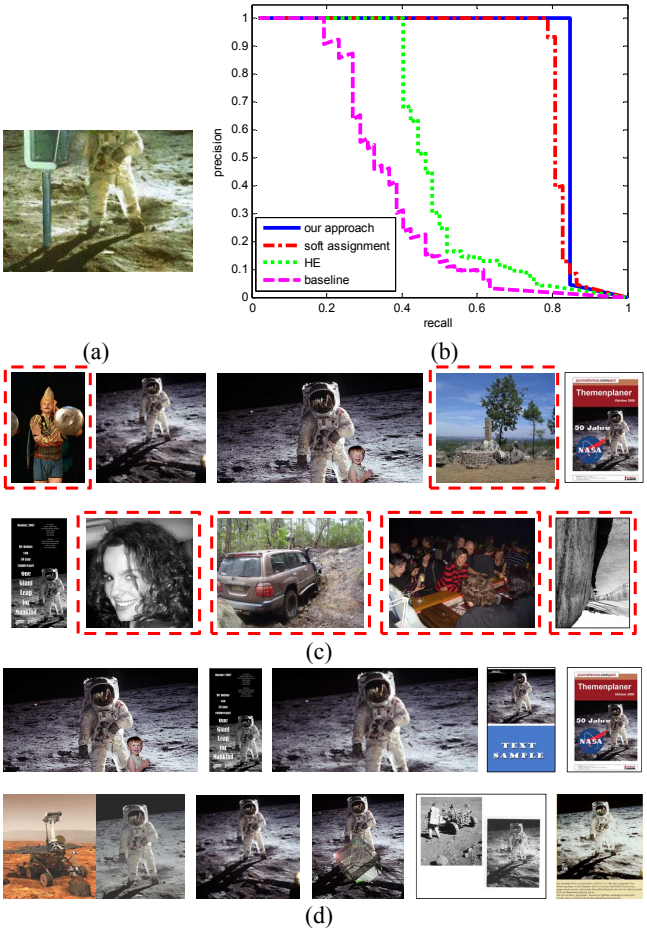
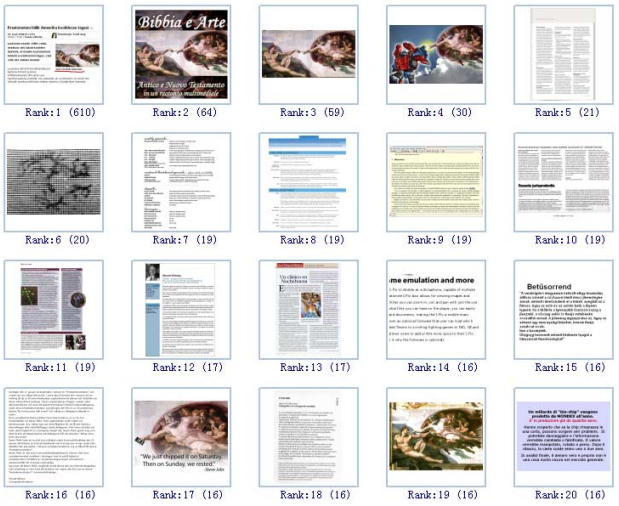
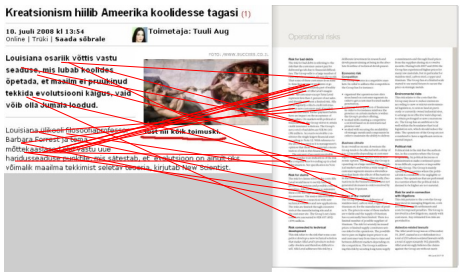


Figure 13. Sample results comparing the baseline and our approach. (a) Query image; (b) Comparison of the precision-recall curves of all four approaches; (c) The top images (rank 11 to rank 20) returned by the baseline; (d) The top images returned by our approach (rank 11 to rank 20). The first 10 images are true positive for both baseline and our approach and therefore are not shown. The false positives are shown with red dashed bounding boxes. (Best viewed in color PDF)



(a)



(b)

Figure 14. Retrieval examples. (a) Retrieval results of the query (Rank 1). The number of local matches is shown in the bracket below each image. (b) Feature matching between the query and the rank 5 result. (Best viewed in color PDF)

5. DISCUSSION

5.1 Threshold in Quantization

In our scalar quantization, the threshold $\hat{f}$ in Eq. (1) is selected as the median value of the feature descriptor, so as in Eq. (2). However, it is still an open question whether the median value is the best choice. There are many alternatives for it, such as mean value of vector f. Alternatively, we can also cluster the 128 elements of a SIFT descriptor into two clusters and choose the cluster boundary as the threshold $\hat{f}$. For some specific applications, $\hat{f}$ can also be learned by training. For example, some positive and negative matching pairs are manually labeled beforehand. Then, the threshold $\hat{f}$ can be learned to yield the best classification performance for discrimination.

5.2 Dimension Reduction

Our approach consumes more memory to index each feature (28 bytes) than other three approaches. It is desired that the memory cost can be further reduced without much sacrifice of search accuracy. As discussed in Section 3.1, it can be inferred that the size of scalar quantization result is proportional to the dimension of SIFT descriptors. If SIFT descriptors can be reduced in dimension, the quantization result will be more compact. There are two possible ways to achieve this goal. One method is to project SIFT descriptors to a low dimensional space by PCA. The dimension-reduced features are expected to keep the characteristics of the original SIFT descriptors, so that our scalar quantization can be applied in the new feature space as well.

The second possible method is to change the formulation of SIFT descriptors, instead of performing dimension reduction. The original SIFT descriptor is designed as a combination of 8-D orientation histogram on 16 (4×4) patches. It is possible to reformulate it in a simple way, such as dividing the local region into 3×3 patches instead of 4×4, and extracting an 8-D orientation histogram on each patch. And SIFT descriptor dimension will be reduced from 128 to 72. It is demonstrated by Lowe [1] that, such a design only causes a small drop on the closest neighbor matching. Its impact on large-scale image search still needs to be further studied.

5.3 Extension to General Features

In this paper, we present our scalar quantization based on SIFT descriptors. An intuitive question "can the proposed scalar quantization algorithm be extended to general feature vectors?" may be interesting. SIFT descriptors are very distinctive, with most energy concentrated on relatively few bins, as shown by a typical example in Fig. 3(a). Besides, it captures the local visual appearance with very strict representation. Such property makes our approach work well on SIFT descriptors. Therefore, our scalar quantization can be generally extended to other features with similar property as SIFT feature.

6. CONCLUSION

In this paper, a novel quantization scheme "scalar quantization" is proposed on SIFT descriptors for large-scale image search. Scalar quantization quantizes a SIFT descriptor to a 256-bit vector, which can be easily adapted to the classic inverted file structure for indexing. Distinguished from the traditional vector quantization approaches, the proposed scalar quantization approach does not involve any kind of visual codebook training or clustering. The quantizer is defined by an individual feature itself and is independent of collections of images. Further, soft quantization is proposed to efficiently enumerate the nearest code words for quantization error reduction. Experiments on large-scale image search demonstrate the superiority of scalar quantization on retrieval accuracy over other state-of-the-art methods

In the future, investigation will be performed on developing more compact bit-vector representation in scalar quantization. Moreover, the flipping behavior of bit-vectors of similar SIFT descriptors will be explored. Some insights are expected to be obtained from this study, which may be beneficial for searching space reduction in soft quantization step and consequently improve retrieval efficiency. Further, various choices for threshold selection in scalar quantization will be studied.

7. ACKNOWLEDGMENTS

This work was supported in part to Dr. Lu by Research Enhancement Program (REP), start-up funding from the Texas State University and DoD HBCU/MI grant W911NF-12-1-0057, in part to Dr. Li by NSFC general project "Intelligent Video Processing and Coding Based on Cloud Computing", and in part to Dr. Tian by ARO grant W911NF-12-1-0057, NSF IIS 1052851, Faculty Research Awards by Google, NEC Laboratories of America and FXPAL, UTSA START-R award, respectively.

8. REFERENCES

[1] D. Lowe. Distinctive image features form scale-invariant keypoints. *IJCV*, 20(2): 91-110, 2004.

[2] J. Sivic and A. Zisserman. Video Google: a text retrieval approach to object matching in videos. In *Proc. ICCV*, 2003.

[3] J. Philbin, et al. Object retrieval with large vocabularies and fast spatial matching. In *Proc. CVPR*, 2007.

[4] W. Zhou, Y. Lu, H. Li, Y. Song, and Q. Tian. Spatial coding for large scale partial-duplicate Web image search. In *Proc. ACM Multimedia*, 2010.

[5] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Proc. ICCV*, 2007.

[6] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Proc. CVPR*, 2006.

[7] J. Matas, et al. Robust wide baseline stereo from maximally stable extremal regions. In *Proc. BMVC*, 2002.

[8] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *IJCV*, 1(60):63–86, 2004.

[9] H. Bay, T. Tuytelaars, L. V. Gool. SURF: Speeded up robust features. In *Proc. ECCV*, 2006.

[10] M. Calonder, et al. BRIEF: binary robust independent elementary features. In *Proc. ECCV*, 2010.

[11] Ethan Rublee, et al. ORB: an efficient alternative to SIFT or SURF. In *Proc. ICCV*, 2011.

[12] C. Silpa-Anan and R. Hartley. Localization using an image map. In *Australasian Conf. on Robo. and Auto.*, 2004.

[13] J. Philbin, O. Chum, M. Isard, J. Sivic and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Proc. CVPR*, 2008.

[14] T. Tuytelaars, C. Schmid. Vector quantizing feature space with a regular lattice. In *Proc. ICCV*, 2010.

[15] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Proc. ECCV*, 2008.

[16] Z. Wu, Q. Ke, et al. Bundling features for large scale partial-duplicate Web image search. In *Proc. CVPR*, 2009.

[17] W. Zhou, H. Li, Y. Lu, Q. Tian, Large scale image search with geometric coding. In *Proc. ACM Multimedia*, 2011.

[18] O. Chum, J. Philbin, et al. Near duplicate image detection: min-Hash and tf-idf weighting. In *Proc. BMVC*, 2008.

[19] O. Chum, M. Perdoch, and J. Matas. Geometric min-Hashing: finding a (thick) needle in a haystack. In *Proc. CVPR,* 2009.

[20] M. A. Fischler, et al. Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM*, 24: 381–395, 1981.

[21] J. L. Bentley. K-d trees for semidynamic point sets. In Proc. 6th Ann. *ACM Sympos. Comput. Geom.*, pp. 187-197, 1990.

[22] S. Arya and D. Mount. Ann: Library for approximate nearest neighbor searching. *Available at* http://www.cs.umd.edu/~mount/ANN/.

[23] R. Baeza-Yates and B. Ribeiro-Neto. Modern information retrieval. *ACM Press*, ISBN: 020139829, 1999.

[24] Y. Kuo, K. Chen, et al. Query expansion for hash-based image object retrieval. In *Proc. ACM Multimedia*, 2009.

[25] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209-226, 1977.

[26] M. Jain, H. Jégou and P. Gros. Asymmetric Hamming embedding: taking the best of our bits for large scale image search. In *Proc. ACM Multimedia*, 2011.

[27] H. Jégou, et al. Aggregating local descriptors into a compact image representation. In *Proc. CVPR*, 2010.

[28] D. Chen, S. Tsai, et al. Residual enhanced visual vectors for on-device image matching. In *Asilomar Conference on Signals, Systems, and Computers*, 2011.

[29] S. Zhang, Q. Tian, G. Hua, Q. Huang, and W. Gao. Generating descriptive visual words and visual phrases for large-scale image applications. *TIP*, 20(9): 2664-2677, 2011.

[30] S. Zhang, et al. Building contextual visual vocabulary for large-scale image applications. In *Proc. ACM Multimedia*, 2010.