Weighted Component Hashing of Binary Aggregated Descriptors for Fast Visual Search

Ling-Yu Duan, *Member, IEEE*, Jie Lin, Zhe Wang, Tiejun Huang, *Senior Member, IEEE*, and Wen Gao, *Fellow, IEEE*

***Abstract*—Towards low bit rate mobile visual search, recent works have proposed to aggregate the local features and compress the aggregated descriptor (such as Fisher vector, the vector of locally aggregated descriptors) for low latency query delivery as well as moderate search complexity. Even though Hamming distance can be computed very fast, the computational cost of exhaustive linear search over the binary descriptors grows linearly with either the length of a binary descriptor or the number of database images. In this paper, we propose a novel weighted component hashing (WeCoHash) algorithm for long binary aggregated descriptors to significantly improve search efficiency over a large scale image database. Accordingly, the proposed WeCoHash has attempted to address two essential issues in Hashing algorithms: "what to hash" and "how to search." "What to hash" is tackled by a hybrid approach, which utilizes both image-specific component (i.e., visual word) redundancy and bit dependency within each component of a binary aggregated descriptor to produce discriminative hash values for bucketing. "How to search" is tackled by an adaptive relevance weighting based on the statistics of hash values. Extensive comparison results have shown that WeCoHash is at least 20 times faster than linear search and 10 times faster than local sensitive hash (LSH) when maintaining comparable search accuracy. In particular, the WeCoHash solution has been adopted by the emerging MPEG compact descriptor for visual search (CDVS) standard to significantly speed up the exhaustive search of the binary aggregated descriptors.**

***Index Terms*—Aggregating local features, hamming space, hashing, visual search.**

I. Introduction

VISUAL search regards the discovery of images contained within a large database that depict the same objects/scenes as those depicted by query images. In general, state-of-the-art image search systems [21]–[26] are built upon a visual vocabulary model with an inverted indexing structure, which quantizes local features (e.g., SIFT [3] or SURF [4]) of query and database images into visual words. Each database image is then represented as a Bag-of-Words (BoW) histogram and is invert indexed by quantized words of local features in the image. Recently, the Fisher Vectors (FV) [1], [5] and the Vector of Locally Aggregated Descriptors (VLAD) [2], [30] have extended the BoW by encoding higher-order statistics of the distribution of local features. Most of the state-of-the-art image retrieval systems are built upon the aggregated descriptors such as BoW, FV and VLAD. Compared to the BoW with a large vocabulary (e.g., 1 million visual words), both FV and VLAD have achieved the state-of-the-art search performance at a much smaller visual vocabulary (e.g., hundreds of visual words) [30].

The high-dimensional image-level representations such as FV and VLAD are often compressed to binary aggregated descriptors, without degrading the discriminative power [5]–[7], [39]. Binary aggregated descriptors allow for fast Hamming distance computation as well as light storage of visual descriptors extracted from either query or database images. To tackle the query latency issue of mobile visual search [10], [36], [11], recent work [6], [7] focuses on directly extracting binary aggregated descriptors of query images at the mobile end and sending compact descriptors to the server end. At the server end, an exhaustive linear search is performed by computing the Hamming distance between the query and database images based on the binary aggregated descriptors. The retrieval accuracy can be further improved by weighting the Hamming distance [37]. In particular, the topic of compact descriptors is closely related to the MPEG standardization of Compact Descriptors for Visual Search (CDVS) [8], [9], [15]. A scalable compressed Fisher Vector (SCFV) [7] has been adopted by the emerging MPEG CDVS standard to specify the technology of aggregating local features' descriptors to form an image level global descriptor.

For the state-of-the-art binary aggregated descriptors, even though Hamming distance can be computed very fast, the accumulated computational cost from exhaustive linear search between query and database images grows linearly with the descriptor length as well as the scale of image database. In this work, our goal is to develop fast approximate nearest neighbor (ANN) search algorithms for the binary aggregated descriptors towards large scale visual search.

Generally speaking, state-of-the-art ANN search approaches may be categorized into three groups: tree-structured search (e.g., randomized KD-trees [14]), clustering (e.g., Hierarchical

Manuscript received August 07, 2014; revised December 13, 2014 and March 21, 2015; accepted March 23, 2015. Date of publication April 06, 2015; date of current version May 13, 2015. This work was supported by the Chinese Natural Science Foundation under Contract 61271311 and Contract 61390515, and by the National Hightech R&D Program of China (863 Program) under Grant 2015AA016302. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Cees Snoek.

The authors are with the Institute of Digital Media, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China (e-mail: lingyu@pku.edu.cn; jielin@pku.edu.cn; zhew@pku.edu.cn; tjhuang@pku.edu.cn; wgao@pku.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2015.2419973

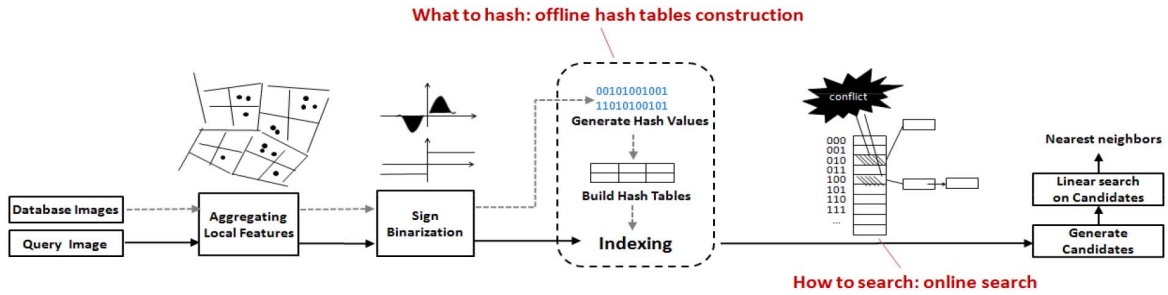


Fig. 1. Image search framework by applying hashing algorithms to binary aggregated descriptors like the binarized Fisher vectors (FV) or vector of locally aggregated descriptors (VLAD). "What to hash" in the offline stage relates to the construction of hash tables, while "How to search" in the online stage relates to the hash value collision based candidates retrieval as well as the hamming distance computing based linear search over the short list of recalled candidates.

K-Means [22] or Product Quantization [40]) and hashing[1] [12]–[14]. The tree-structured and clustering based ANN approaches are designed basically for real-valued descriptor vectors in Euclidean space, which would seriously underperform in dealing with binary vectors, which will be revisited in Section II. By contrast, hashing algorithms, such as the well-known Locality Sensitive Hashing (LSH) for binary vectors [12], can significantly alleviate the exhaustive Hamming distance computing by reducing the search space from the whole database to a small subset of candidate images. Specifically, any hashing approach involves offline hash tables construction and online search (see Fig. 1). Each hash table corresponds to a hash key formed by selecting a whole binary vector or part of a binary vector. Each hash key relates to a set of buckets, by which an inverted index structure is employed to index the binary vectors of database images. In online search, given a query binary vector, a subset of candidate images can be figured out by the collision in the buckets, and then linear search using the whole binary vector is performed over the subset of candidate images to return the final results.

However, to solve the problem of ANN search with the binary aggregated descriptors, hashing techniques have yet to address two key issues of (1) generating discriminative hash keys to index binary vectors effectively and (2) reducing the required number of candidate images for exhaustive linear search. One may simply increase the number of hash keys (each incurring a hash table) to improve search accuracy, but slow search speed and heavy memory cost of hash tables would result. This will be qualitatively discussed in Section II and quantitatively validated in Section IV. Distinct from existing hashing algorithms, the proposed WeCoHash applies the statistics of binary aggregated descriptors, including visual words redundancy to generate discriminative hashing keys towards an optimal hashing strategy.

To address an optimal hashing strategy for binary aggregated descriptor, we study two key issues of "What to hash" and "How to search", which are essential to implement an hashing algorithm [16], [20], [41]. The former works on the offline hash tables construction and the latter online search. "What to hash" aims to deploy discriminative hash values into buckets. Firstly, each hash value involves a subset of bits from a binary aggregated descriptor. As the discriminative power of each bit would vary with its statistical importance, not all hash values are of equal importance to distinguish an image. To discard the noisy hash values we may remove the image specific redundancy in terms of informative hash values. Secondly, by using the global bit statistics over the image collection, the bit-wise dependency between binary aggregated descriptors can be further removed to reduce the dimensionality of hash values. "How to search" aims to maximize the search accuracy, while minimizing the size of candidates for linear search during online search. Our practice is to exploit the statistical relevance of hash values between query and database images, to weigh the similarity score for generating the shortlist of "good" candidates.

In this paper, we propose a novel weighted component hashing (WeCoHash) algorithm for indexing the state-of-the-art long binary aggregated descriptors. Extensive experiments have shown that WeCoHash significantly improves search speed as well as reduces the memory footprint of hash tables, while search accuracy is well maintained. Below, we brief the WeCoHash based image search framework: First, we extract the aggregated descriptor FV [1] (or VLAD [2]), followed by sign binarization [5] to generate the high-dimensional binary aggregated descriptors, which represents the state-of-the-art compact yet discriminative visual descriptors [5]–[7]. Second, we address the issues of "What to hash" and "How to search" by the proposed WeCoHash approach. In the offline stage (see Fig. 2), we partition a binary aggregated descriptor into disjoint components by visual words. Given a binary aggregated descriptor, the informative components with high statistical reliability are adaptively selected, followed by de-correlated bits selection to produce dim-reduced hash values. In the online search, we calculate an adaptive weight w.r.t. each query hash value, and introduce score weighting to improve search performance.

In summary, our contributions are three-fold, as follows.

1. Towards "What to hash", we propose a novel hybrid approach to produce discriminative hash values for the state-of-the-art binary aggregated descriptors by taking advantages of the image specific local redundancy statistics and the collection based global bit dependency statistics. To the best of our knowledge, this approach serves as the first to investigate the issue of discriminative hash values to construct compact hash tables, for the high-dimensional binary aggregated descriptor to accomplish fast visual search.

[1]The term "hashing" in this work refers to data structure for indexing high-dimensional binary aggregated descriptors, in which effective algorithms have to be developed to realize fast search in a collision mechanism. This is different from another popular Hashing topic that learns binary embedding to covert real-valued descriptors into compact binary codes. More discussion of these two terminologies can be found in [31].

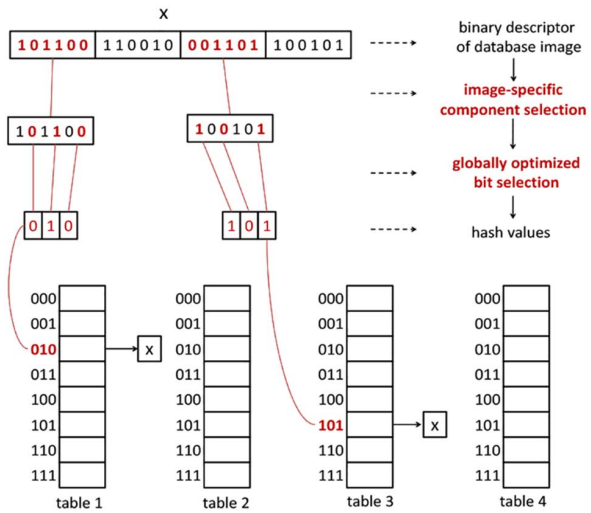


Fig. 2. Offline constructing hash tables via discriminative hash values derived from image specific component selection (a sort of image-level local statistics) and de-correlated bits selection (a sort of global statistics over an image collection).

2. Towards "How to search", we propose an adaptive relevance weighting to boost the online search performance by exploiting the statistics of hash values. Distinct from existing hashing algorithms, our method calculates variable weights to adapt the Hamming distance according to the hash values of a query. The proposed weighting approach significantly reduces the number of candidates for linear search, while maintaining comparable search accuracy. Hence, the search process can be greatly accelerated due to the reduced number of candidates for linear search.
3. Finally, the proposed WeCoHash solution has been adopted by the emerging MPEG CDVS standard [8], [9], [15] as a core technique to solve the problem of scalable visual search based on high performance and low complexity standardized compact descriptors, where Fisher vector [5], [7] is applied to yield the compact binary aggregated descriptor. Extensive comparison experiments over a variety of benchmark datasets have shown the advantages of significantly improved search efficiency, promising search accuracy, as well as much reduced memory cost of Hash tables.

This paper is organized as follows. Section II introduces the related work. In Section III, we describe the proposed WeCoHash for indexing high-dimensional binary aggregated descriptors for fast visual search over large database. In Section IV, we present the experimental results of WeCoHash and compare with the state-of-the-art. Finally, we conclude this paper in Section V.

II. Related Work

In this section, we firstly review the state-of-the-art aggregated descriptors and the schemes of compressing high-dimensional descriptors into binary vectors. Then, we compare various hashing algorithms in solving the problem of ANN search with binary vectors. An extended discussion will be given to typical non-hashing ANN search approaches.

Aggregated Descriptors. The BoW histogram [21] is the most popular image-level descriptor aggregated from local invariant features (e.g., SIFT and SURF). Each bin of the BoW histogram counts the number of the local feature distribution quantized to the corresponding visual word in a visual vocabulary, which encodes the 0-order statistics of local features (unlike FV or VLAD). Many research efforts have been made to improve the BoW model, such as training a large vocabulary for fine-grained partition of the local feature space (e.g., 1 million visual words) [22], [23], Hamming embedding for a coarse-to-fine quantization of local features [26], soft assignment of descriptors to multiple visual words for reducing quantization errors [25], and geometric consistency check (GCC) to re-rank the retrieval results [28]. Despite performance improvements, those conventional BoW approaches bring about heavy memory cost from a large visual vocabulary as well as a big index file in dealing with a large scale image database. For example, the memory cost for inverted indexing 1.1 million images at a vocabulary of 1 million visual words may reach up to 4.3 GB [23].

Perronnin *et al.* proposed FV [1] to extend BoW by encoding high-order statistics of the local feature distribution. FV employs a Gaussian Mixture model (GMM) to estimate the distribution of local features over a training set. Given an image, the gradient vectors of all local features w.r.t. the parameters (i.e., mean or variance) of each Gaussian are aggregated into a vector representation. FV is finally formed by concatenating the 1-order (mean) and/or 2-order (variance) aggregated vectors of all Gaussians. Jégou *et al.* [2] proposed the VLAD to aggregate the residual vectors between local features and their quantized visual words, which can be regarded as a non-probabilistic version of FV. Compared to the BoW, both FV and VLAD significantly reduce the dimensionality of aggregated descriptors and their computational complexity, for instance, from a 10^6-dimensional BoW histogram down to a FV with thousands of dimension. In particular, the discriminative power of FV and VLAD can be further improved by injecting vector normalization [43], [45], vocabulary adaptation [43], spatial information [43], fusion with attribute features [44], orientation covariant aggregation [49], or democratic aggregation of local features [46].

In this work, our focus is on the accelerated matching between high-dimensional binary codes derived from visual words based aggregated descriptors, subject to the performance maintenance against the exhaustive linear search. The proposed visual words based WeCoHash is supposed to be compliant with the state-of-the-art aggregated descriptors including a variety of improved FV or VLAD methods like the latest democratic aggregated descriptor [46].

Binary Aggregated Descriptors. There are lots of work on learning binary codes from real-valued descriptors, such as ORB [17], SH [32], ITQ [34], MLH [35], USB [18], BRIEF [19] and BPBC [39]. The basic idea is to setup an intermediate embedding of original descriptors (say, dimension reduction by PCA), then binarize the projected vectors into binary codes. However, most of them work on relatively low-dimensional descriptors such as SIFT and GIST [29]. In addition, the projection matrices in the embedding stage incur considerable memory (e.g., hundreds of megabytes) and bring about extra computation for the dimension reduction of original descriptors.

An alternative is to directly binarize the original descriptors [21], [5], [6], which can save out a lot of computation and memory resources for computing the dim-reduced descriptors and storing the big projection matrices. Sivic *et al.* [21] proposed to binarize the BoW histogram entries, without incurring any significant performance loss for large vocabularies. Perronnin *et al.* [5] proposed sign binarization to binarize FV. Recent works [6], [7] have further verified the discriminative power of these binarzied descriptors, which obtains comparable search accuracy to original descriptors. These results have shown that a high-dimensional (long) code is necessary to preserve discriminative power of aggregated descriptors, while the binarized operation has less impact on the performance.

However, exhaustively computing the distance between long binary descriptors in Hamming space over a very large database is demanding. The traditional hashing algorithms cannot decently address the efficiency and effectiveness issues of ANN search over the high-dimensional binarized FV (or VLAD) descriptors. In this work, the proposed WeCoHash came up with a novel hashing mechanism to accomplish remarkable search speed up and significantly reduced memory footprint, while maintaining promising search accuracy.

Hashing Binary Vectors for ANN Search. As introduced before, hashing binary vectors for ANN search can be divided into two stages: offline hash tables construction and online search over the hash tables. Considering the length of a binary vector, hashing algorithms can be classified into "Single key Hashing" and "Multiple keys Hashing":

1) "Single key Hashing" (denoted as S-Hashing) refers to directly use the whole binary vector as a hash key when its length is small (i.e., $l \leq 32$). Thus, there is a single hash table to build up. For example, recent works on learning short binary codes (such as Spectral Hashing [32], and Minimal Loss Hashing [35]) adopt S-Hashing to accomplish the offline hash table construction in which the length of binary codes is no longer than 32.
2) "Multiple keys Hashing" (denoted as M-Hashing) refers to build up multiple hash tables when binary vector is long (i.e., a binarized FV descriptor with length $l >> 32$). Each hash table corresponds to a hash key of a binary vector. However, the memory resource limits the choice of hash keys, as the number of buckets increases exponentially with the length of a hash key [41].

Most of existing M-Hashing methods rely on the LSH for binary vectors [12], which selects random subsets of bits from a binary vector as hash keys. Random hash keys generation may lead to unbalanced bit selection, for example, some bits may be selected more frequently than others, or even some of them may not be selected at all [16]. One can simply increase the number of random hash keys to alleviate the problem, but this leads to more search time and heavier memory use. To the best of our knowledge, there is few work studying how to generate optimal hash keys, which, as shown in our work for the binary aggregated descriptors, is crucial for improving the search speedup as well as reducing the memory footprint of hash tables.

Beyond hash tables construction, Esmaeili *et al.* [20] proposed error weighted hashing (EWH) to enhance the online search of LSH. EWH differs from LSH in the way that the candidates are chosen. In LSH, any database image that presents a hash value identical to the query (i.e., $e = 0$) in at least one hash key is adopted as a candidate, while EWH uses e-bit difference of hash values to find candidate images (i.e., $e > 0$). For each hash table, EWH first enumerates all the hash values $\{b(r)\}$ $(0 \leq r \leq e)$ having r-bit difference with the corresponding hash value of a query, and add a weighted similarity score to the database images in the buckets corresponding to $\{b(r)\}$. This procedure is repeated for all hash tables, and results in a list of database images ranked by similarity scores. Finally, a similarity threshold is applied to select the database images with higher scores as candidates. Compared to LSH, EWH can generate a much smaller subset of candidates but yield better search accuracy by tolerating erroneous hash values of the query [20]. However, the weights only relate to Hamming distance e and are empirically fixed, while the statistics of hash values are ignored.

While we focus on ANN search, there also exist hashing algorithms for finding exact nearest neighbors. Norouzi *et al.* [41] proposed Multi-Index Hashing (MIH) to find r-neighbors. The search radius r is meant to constrain the search range to be smaller or equal to r, where the range is measured by the Hamming distance between the query and its nearest neighbors. MIH contiguously partitions a long binary code into m disjoint sub-vectors, builds up multiple hash tables using these sub-vectors as hash indices, and finds r/m-neighbors for each sub-vector. MIH has sub-linear search time for uniformly distributed codes when the search radius r is small. But if we apply MIH to ANN search with high-dimensional binarized VLAD or FV (e.g., thousands of bits), a large search radius is normally needed for satisfactory search accuracy, which exponentially increases search time in scanning the buckets.

Approximate Nearest Neighbor Search. Besides the classical hashing, many other ANN algorithms have been proposed for indexing and searching real-valued descriptors. The most representative are tree-structured search and clustering algorithms. Tree-structured search algorithms like KD-tree [51] or randomized KD-tree [52] build up a set of trees independently, where each tree is constructed by recursively splitting a collection of descriptor vectors along the dimension randomly chosen from the top ranked dimensions with the largest variance at each level. Given a query, it traverses each tree down to a leaf node, followed by backtracking to check other nodes for better candidates. However, the tree-structured search approaches performs very poor when dealing with binary vectors because the query vector can easily move to the wrong branch if a single bit is flipped [16].

Clustering algorithms such as k-means [21], Hierarchical K-Means (HKM) [22] and Product Quantization (PQ) [40] approximate the nearest neighbors search by vector quantizing descriptors. For example, PQ [40], [53]–[56] partitions an original descriptor into disjoint sub-vectors and quantize each sub-vector separately with a pre-trained codebook. The original descriptor is thus represented by short codes composed of the quantizer indices. The distance between original descriptors is approximated by the distance between their quantized visual words, which can be efficiently read from a lookup table. However, to obtain satisfactory search accuracy, PQ generally

requires fine quantization with short sub-vectors associated with large codebooks. As a consequence, it would lead to slow search speed and heavy memory footprint of lookup tables during online search, especially for high-dimensional binary vectors. This will be quantitatively validated in Section IV.

Both tree-structured search and clustering are designed for real-valued descriptor vectors, while few work attempted to apply these algorithms to binary descriptors, especially for large scale image search. However, hashing with binary descriptors exhibits sub-linear search time and light storage of hash tables for uniformly distributed binary descriptors with a proper search radius. More importantly, it supports exact nearest neighbors search [41], which guarantees the best search accuracy.

III. Weighted Component Hashing

A. Problem Definition

Assume that an image is represented as a binary aggregated descriptor $b \in \{0,1\}^l$ with l elements, each element is either 0 or 1. The similarity between binary aggregated descriptors is measured by Hamming distance. The objective of hashing is to construct (multiple) hash tables for indexing the binary aggregated descriptors of database images, and support efficient image search. Accordingly, the problems of "What to hash" and "How to search" can be defined as follows (see Fig. 1):

- *What to hash* generates the i-th hash key function h_i by selecting a subset of bits $b(i)$ from binary aggregated descriptor b

$$h_i(b) = b(i), \quad b(i) \in \{0,1\}^z. \tag{1}$$

  As the quality of selected bits impacts search accuracy and memory cost of hash tables, the hash value $b(i)$ derived by h_i from an image shall be discriminative for describing that image.
- *How to search* matches individual hash values $b(i), i \in [1,s]$ between query b^q and database image b^j with an ANN search and accumulates the matching scores to generate similarity score s_j as

$$s_j = \sum_{i=1}^{s} s(b^q(i), b^j(i)), \quad j \in [1,n] \tag{2}$$

  where $s(b^q(i), b^j(i))$ denotes the matching score of hash values $b^q(i)$ and $b^j(i)$, s and n denote the number of hash keys (hash tables) and the number of database images, respectively. Given a query, a subset of candidate images is produced by collecting database images with high similarity score. To maintain the search accuracy of exhaustive search and reduce the number of shortlisted candidate images, the matching score $s(\cdot)$ shall be informative to distinguish the nearest neighbors of the query image.

To address these two problems with the state-of-the-art binary aggregated descriptors, we propose the WeCoHash solution as presented in the next subsections. First, Section III-B briefs the extraction of binary aggregated descriptors in this work. Then, we present the WeCoHash in two parts (1) the offline hash tables construction based on image-specific local statistics in Section III-C and globally optimized bit selection in Section III-D, and (2) the online search based on adaptive relevance weighting in Section III-E. Finally, we analyze the complexity of WeCoHash in Section III-F.

B. Binary Aggregated Descriptors

Let $X = \{x_1, \ldots, x_t\}$ denote a collection of local features extracted from an image. In this work, we use a variant of SIFT descriptor, RootSIFT [42]. RootSIFT simply applies square root to each element of SIFT. Principal Component Analysis (PCA) is employed to project the dimensionality of 128-dimensional RootSIFT to dimension d [2]. These pre-processing operations are beneficial to the overall performance [42], [2], [7]. As discussed in Section II, the 0-order, 1-order and 2-order statistics of transformed local feature distribution are employed to generate the raw aggregated descriptors. Concretely speaking, FV is applied to perform the aggregation, while VLAD is a simplified non-probabilistic version of FV.

FV [38], [1] firstly employs a Gaussian Mixture model (GMM) with k Gaussians (visual words) to estimate the distribution of local features over a training set. We denote the set of Gaussian parameters as: $q_i = \{\omega_i, \mu_i, \sigma_i^2\}, i = 1, \ldots, k$, where ω_i, μ_i and σ_i^2 are the weight, mean vector and variance vector of the ith Gaussian, respectively. For each Gaussian, the gradient vectors of all local features w.r.t. the mean μ_i are aggregated (averaged) into a d-dimensional vector

$$g(i) = \sum_{x \in X} \gamma(x,i)\sigma_i^{-1}(x - \mu_i) \tag{3}$$

where $\gamma(x,i) = \omega_i p_i(x) / \sum_{j=1}^{k} \omega_j p_j(x)$ denotes the posterior probability of local feature x being assigned to the ith Gaussian. FV g is formed by concatenating the sub-vectors $g = (g(1), \ldots, g(k))$ of all Gaussians and is therefore $l = kd$-dimensional.

As proved in [30], the VLAD [2] is a simplified non-probabilistic version of FV. In other words, VLAD can be derived from FV by replacing the GMM soft clustering with k-means clustering. Accordingly, (3) degenerates to the form $g(i) = \sum_{x \in X_i} x - \mu_i$, where X_i denotes the subset of local features in an image that are assigned to the ith visual word.

Finally, we generate binary aggregated descriptors by quantizing each dimension of FV or VLAD into a single bit 0/1 based on a sign function. Formally speaking, we project each element g_i of descriptors g to 1 if $g_i > 0$; otherwise, 0

$$sgn(g_i) = \begin{cases} 1, & \text{if } g_i > 0 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

which yields a binary descriptor $b = (b(1), \ldots, b(k))$ with $l = kd$ bits that will be used for the subsequent hashing.

C. Image-Specific Component Selection

For multiple hash tables construction, both query and database images are supposed to be represented by a common set of hash keys, each hash key comprising of a subset of bits from a binary aggregated descriptor [5], [6], [7]. "What to hash" is meant to tackle the problem that not all hash keys equally

contribute to describing an image. Given a pool of hash keys, image-specific component selection uses local statistics to figure out part of discriminative hash keys to establish hash table buckets in an adaptive way.

As shown in (3), the aggregated descriptors are formed by concatenating residual vectors computed for all visual words, while each residual vector is aggregated from local features being assigned to the corresponding visual word. Let us define the occurrence of a visual word as the number of local features quantized (the nearest) to that visual word. The occurrence of different visual words may vary within an image, while the visual words with low occurrence are supposed to be less discriminative for describing the image [21]. In an extreme case, if none of local features is assigned to a visual word, all the elements of the corresponding residual vector are zero, which means that this visual word shall not be taken into account for image description.

Therefore, a natural way is to partition a binary aggregated descriptor into disjoint components by visual words and regard each component as a hash key. The goal is to adaptively select part of discriminative components for each image, through selecting the subset of visual words with high reliability. In this paper, we define the reliability $r(i)$ as a soft-weighted occurrence $o(q_i)$ of the ith visual word q_i

$$r(i) = o(q_i). \tag{5}$$

Specifically, for the continuous FV model, $o(q_i)$ is defined as the sum of posterior probabilities of local features $\{x_1, \ldots, x_t\}$ being assigned to the ith visual word: $o(q_i) = \sum_{j=1}^{t} \gamma(x_j, i)$. For the discrete VLAD model: $o(q_i) = \sum_{j=1}^{c} w_j * \#(q_i, j)$. Each local feature can be soft quantized to multiple visual words. Given a local feature, we rank all the visual words based on the distance between the local feature and the centroid of each visual word. The visual word ranked at the j-th position is called the j-th nearest neighbor visual word of the local feature. $\#(q_i, j)$ denotes the number of local features whose the j-th nearest neighbor visual word is q_i, and w_j the associated weight. The smaller j is, the larger w_j will be. The soft-weighted occurrence is more stable as the rank information is considered. In the experiments, we empirically set the weights as $w_1 = 4, w_2 = 2, w_3 = 1$.

We have empirically studied the impact of component selection on the distribution of Hamming distance of match/non-match image pairs. We generate 2,550 match image pairs and 25,500 non-match image pairs from the UKbench dataset (see Section IV). As shown in Fig. 3, it is easier to separate match and non-match image pairs, when Hamming distance is computed over the selected components rather than all the components. This demonstrates the necessity of discarding "noisy" components within a binary aggregated descriptor.

For offline hash tables construction, we compute the reliability of components for each database image using (5) and select a subset of components with high reliability. For each selected component, we store the database image ID in the hash bucket with an identical hash value. The rest of components are discarded. In online search, we apply component selection

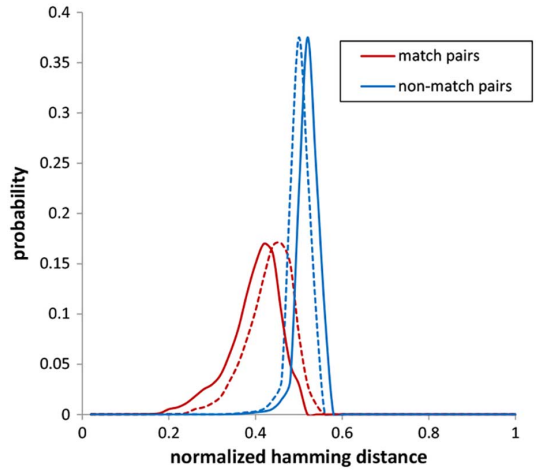


Fig. 3. Effects of image-specific component selection on the probability distribution of Hamming distance of 2,550 match image pairs and 25 500 non-match image pairs from the UKbench dataset (solid lines are with component selection, while dotted lines without component selection). The binaried FV is with $k = 128$ and $d = 64$ in this experiment.

to query images as well and perform search using the selected components. In practice, we apply a uniform number of selected components for all images. The selected components may vary in different images.

By selecting a subset of components for hashing, the number of database image IDs stored in the buckets may decrease, thereby reducing the memory cost of hash tables. Meanwhile, as "noisy" entries are removed from the buckets, the required number of shortlisted candidates can be largely reduced. Accordingly, search speedup can be further improved, while comparable retrieval accuracy is maintained.

D. Globally Optimized Bit Selection

After image-specific component selection, each component (visual word) is regarded as a hash key and the dimensionality of each hash key equals to the dimensionality of a visual word. Ideally, for a d-dimensional hash key, there exist 2^d hash values (buckets) in the corresponding hash table. If d is large, it costs huge memory to maintain the data structure of buckets. For instance, when $d = 32$, the memory cost of buckets (from memory address) is $2^{32} * 4$ bytes (4 bytes for each bucket [41][2]). Besides, the number of buckets to check increases near exponentially with search radius r, resulting in less efficient search. More importantly, even though we perform visual word level component selection to remove redundancy from a binary aggregated descriptor, there probably exists bit-level redundancy among hash values within each selected component. Therefore, we propose bit selection to further reduce the dimensionality of components, while maintaining search performance as well.

A naive solution is to apply random bit selection, which, however, ignores the bit correlation within a component. Given the

[2]As analyzed in the MIH paper [41], the data structure of a hash table is actually an array of pointers. The size of each pointer (corresponding to a bucket) is 4 bytes (for a 32-bits operating system). Even a bucket is empty, a 4 bytes pointer should be occupied; otherwise additional data structure is needed to identify which pointers are dismissed due to the null bucket, while extra complexity is incurred.

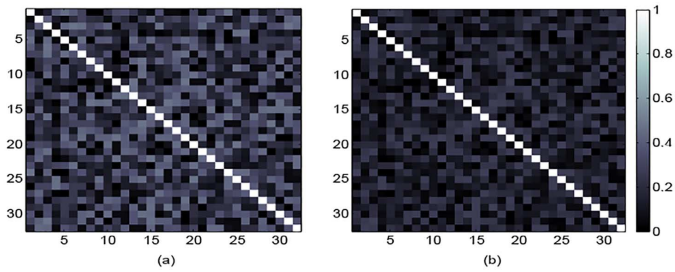


Fig. 4. Visualization of bit correlations of (a) randomly selected 32 bits and (b) globally optimized selected 32 bits from a selected component with $d = 64$ of binarized FV descriptors (in which 64-dimensional PCA reduced RootSIFT is applied) over the UKbench dataset (10,200 images). Bright colors indicate strong bit correlation.

ith component $b(i) = (b_1, \ldots, b_d)$ with dimension d, we define the bit correlation $c(b_i, b_j)$ between bit b_i and b_j as

$$c(b_i, b_j) = \frac{I(b_i; b_j)}{H(b_i, b_j)} \tag{6}$$

where $H(b_i, b_j)$ and $I(b_i; b_j)$ denote the joint entropy and the mutual information of a pair of bits b_i and b_j, respectively. Due to limited space, we add these definitions into Appendix. Fig. 4(a) visualize the exemplar statistics of bit correlations between any two bits within a component, where 32 bits are randomly selected from a 64-dimensional hash key. As one can see, the majority of bits are correlated more or less with each other.

In this paper, we propose a globally optimized bit selection based on mutual information minimization [47] to produce dim-reduced informative hash values. Formally speaking, for the ith component $b(i)$ from a binary aggregated descriptor, the goal is to generate a hash key h_i by selecting z ($z < d$) bits that carry as much information as possible, let us denote the hash key h_i as the position of selected bits $h_i = \{u(1), \ldots, u(z)\}$ with $u(j) \in [1, \ldots, d]$. The globally optimized bit selection is formulated as an entropy maximization problem

$$h_i = *argmax_{u^*(1),\ldots,u^*(z)} H(b_{u^*(1)}, \ldots, b_{u^*(z)}), \quad u^*(j) \in [1, \ldots, d] \tag{7}$$

where $u^*(j)$ denotes the position (ranging from 1 to d) of the selected bits from a component, and $b_{u^*(j)}$ denotes the actual bit element of that position. As this combinatorial optimization is computationally intractable, we employ a greedy algorithm [48] to solve (7)

$$u(1) = *argmax_i H(b_i), \quad i \in [1, \ldots, d] \tag{8}$$

$$u(a+1) = *argmin_{i \notin \{u(j)|1 \le j \le a\}} \sum_{j=1}^{a} I(b_i; b_{u(j)}), \quad 1 \le a < z. \tag{9}$$

We firstly choose the single bit $u(1)$ with maximum entropy from a component, then iteratively pick up the bit $u(a+1)$ which minimizes its mutual information with the already chosen bits $\{u(1), \ldots, u(a)\}$. Equation (9) can be solved efficiently as we only compute the entropy of two variables (bits). The mutual information minimization de-correlates the selected bits towards discriminative hash values of reduced dimension. Fig. 4(b) visualizes bit correlations between 32 selected bits from a 64-dim component by the global optimization. By comparing Fig 4(a) and (b), one can see that the mutual information minimization based selected bits are much less correlated than random bit selection.

Finally, we summarize the proposed multiple hash tables construction method combining image-specific component selection and globally optimized bit selection in Algorithm 1.

Algorithm 1: The offline hash tables construction through image-specific component selection and globally optimized bit selection.

Input: Binary aggregated descriptors of database images $b^i (1 \le i \le n)$. Number of components (visual words) k.
Output: Hash tables $T = \{T_1, \ldots, T_k\}$.
1: Clear each hash table $T_i \leftarrow NULL, i \in [1, k]$.
2: **for** i from 1 to n **do**
3: **for** j from 1 to k **do**
4: **if** the j-th component of b^i is selected based on its reliability $r(j)$ **then**
5: Generate hash value $b^i(j)$ from b^i based on the corresponding hash key (i.e., bit selection mask) h_j.
6: Insert image ID i into hash bucket $T_j(b^i(j))$.
7: **end if**
8: **end for**
9: **end for**
10: $T = T_1 \cup T_2 \cup \ldots \cup T_k$.
11: return T.

E. Adaptive Relevance Weighting

To answer "How to search", we propose an adaptive relevance weighting to boost the online search performance by exploiting the hash value relevance between query and database images. Recall that existing weighting approaches such as EWH [20] extend the traditional e-bit difference mechanism by setting different weights for different Hamming distance r ($0 \le r \le e$). That is, when using multiple hashing tables, a higher weight is empirically assigned to the similarity score derived from a collided hash value that generates a lower Hamming distance to that of a query. However, the weights just relate to the Hamming distance, while the statistical information from hash values is ignored; moreover, the empirically fixed weights cannot adapt to different hash values. In this work, we propose to incorporate the statistics of hash values into the weighting process, resulting in adaptive weights with respect to different hash values.

We attempt to model the statistical relevance between query and database images in terms of hash values, which is useful for adaptive weighting of hash values. The basic idea is that the most promising nearest neighbors returned by hashing collision are supposed to fall into the buckets with low Hamming distance to query hash value. Meanwhile, for those low Hamming

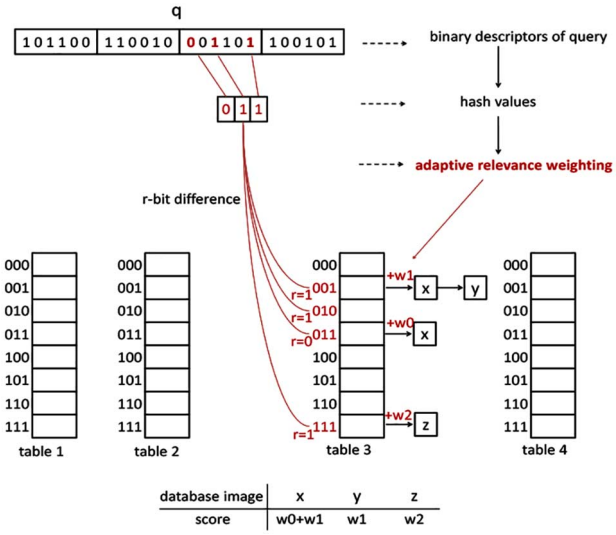


Fig. 5. Online search based on the adaptive relevance weighting. Given a query hash value from a selected component, we generate all binary vectors (i.e., hash values) $\{b(r)\}$ $(0 \leq r \leq 1)$ having r-bit difference with the query hash value, each binary vector corresponding to a bucket in the hash table and each bucket counting the hash value collisions of database images (e.g., x, y, z) for the adaptive weighting (e.g., w_0, w_1, w_2).

distance buckets, a smaller number of collision means that the corresponding hash value tends to be more informative in distinguishing the "true" nearest neighbors from "noisy" database images, which follows a similar idea of inverted document frequency (IDF) [21] in retrieval. Specifically, the matching score between dim-reduced hash value $b^q(i)$ and $b^j(i)$ in (2) is rewritten as follows:

$$s(b^q(i), b^j(i)) = \sum_{r=0}^{e} w^r \left(b^q(i), b^j(i)\right) \tag{10}$$

where $w^r(b^q(i), b^j(i))$ is defined as the adaptive relevance weight between hash values $b^q(i)$ and $b^j(i)$ with r-bit difference

$$w^r(b^q(i), b^j(i)) = \begin{cases} \log(\frac{n}{\#(b^q(i), r)}), & Ha(b^q(i), b^j(i)) = r \\ 0, & \text{otherwise} \end{cases}$$

where $Ha(\cdot)$ denotes the Hamming distance between two binary vectors; $\#(b^q(i), r)$ the total number of database images falling into colliding buckets of the ith component, each bucket corresponding to a hash value with Hamming distance $r \in [0, e]$ to $b^q(i)$; n the number of database images.

As illustrated in Fig. 5, given a query, we initially reset the similarity scores of all database images to zero. For the query hash value $b^q(i)$ from the i-th selected component, we firstly generate all hash values $\{b(r)\}$ $(0 \leq r \leq e)$ having r-bit difference with $b^q(i)$, each hash value corresponding to a bucket in the hash table and each bucket accumulating the hash value collisions of database images for the adaptive weighting in (10). For each selected component, this procedure is repeated for all hash values that differ from $b^q(i)$ by $0, 1, \ldots, e$ bits, respectively; then this procedure is repeated for all the selected components of the query. As a result, a list of similarity scores of the database images are generated. A threshold is applied to select the images with high similarity scores as candidates. Finally, an exhaustive linear search is performed on the shortlisted candidates with a full-length binary descriptor to find out the nearest neighbors. The online search process is summarized in Algorithm 2.

During online search, we have to calculate the weight in (10) for all the query hash values of selected components and all Hamming distance r, which inevitably slows down the search process. To address the issue, we propose a fast look-up table method. For each component, we enumerate all the possible query hash values and compute their weights at different Hamming distance r using (10). Thus, there are in total $k{*}2^z{*}(e+1)$ elements to be stored in the lookup table. In online search, given a query hash value of a component and Hamming distance r, we can quickly retrieve the weight value from the lookup table. The memory cost of a lookup table is mainly determined by the number of selected bits z, which is usually a small value (e.g., less than 16). For instance, when $k = 128$, $z = 12$, $e = 2$, only 6 megabytes is needed to store the lookup table.

In summary, the proposed relevance weighting as the similarity scoring weights, differs from EWH [20] in two aspects (1) the adaptive relevance weighting incorporates the statistical information of hash values in addition to the Hamming distance. (2) the adaptive weighting employs the hashing value statistics of both query and database images to distinguish discriminative hash values for improving search performance.

Algorithm 2: The online search based on adaptive relevance weighting.

Input: Binary aggregated descriptor of query b^q. Hash tables $T = \{T_1, \ldots, T_k\}$. Hash keys (bit selection masks) $\{h_1, \ldots, h_k\}$. Size of database n. Look-up table computed from (10). Similarity threshold t.
Output: K approximate nearest neighbors of the query.

```
Initialize the similarity scores to 0, s_j ← 0, 1 ≤ j ≤ n.
for i from 1 to k do
    if i-th component of query q is selected based on its reliability r(i) then
        Generate hash value b^q(i) from b^q based on the corresponding hash key h_i.
        for r from 0 to e do
            Enumerate {b(r)}, the set of hash values for all binary vectors that have r-bit difference with b^q(i).
            Compute s(b^q(i), b^j(i)) using look-up table and add it to s_j that are within the corresponding buckets T_i(b(r)) of hash table T_i.
        end for
    end if
end for
Collect all the database images whose similarity scores are greater than threshold t as candidates.
Compute the Hamming distance between the query and candidates, and return the top K nearest neighbors.
```

F. Analysis of Time and Memory Complexity

Let us denote the number of database images as n, the number of visual words k, the dimensionality of a visual word d, the number of selected components for each image s, the length of dim-reduced hash value z, the number of candidate images K and the computation time cost of Hamming distance between d-dimensional binary vectors ϕ, we analyze the search time and memory cost of the proposed WeCoHash algorithm as follows:

1) *Search time*. Given a query, search time cost mainly consists of two parts: (1) $O(ms(C_z^0 + C_z^1 + \ldots + C_z^e))$: visit all buckets from s hash tables that are within e-bit difference to the hash value of a query, where $m = n/2^z$ denotes the average number of entries (database image IDs) in each bucket and C_z^i the number of hash values having i-bit difference with query. The entries in the buckets are considered as candidates. (2) $O(K*\phi)$: Hamming distance based exhaustive linear search on the K candidate images to return nearest neighbors. The first part can be ignored when e is small (e.g., $e < 3$), search time is then determined by the cost of exhaustive search on the candidates, i.e., the number of returned candidates.
2) *Memory cost*. The memory cost of hash tables consists of two parts: (1) $O(2^z*k)$: the cost of data structure to maintain the buckets. (2) $O(n*s)$: the entries in the buckets. In general, the length of dim-reduced hash value z and the number of visual words k are small values, so the first part can be ignored, and the memory footprint mainly depends on the number of entries in the buckets (i.e., the number of database images n as well as the number of selected components s).

IV. Experiments

A. Datasets

To evaluate the advantages of WeCoHash algorithm, we perform extensive comparison experiments over publicly available benchmark datasets, including UKbench, INRIA Holidays and Stanford Mobile Visual Search (SMVS) dataset, combined with 1 million distractor images collected from the Flickr website (FLICKR1M).

UKbench [22] contains 10,200 images with 2,550 objects. There are 4 images per object involving variances in viewpoints, lightings, occlusions and affine transforms. All the 10,200 images are used as reference images and each image is used as a query as well.

INRIA Holidays dataset [26] is a collection of 1,491 holiday photos. There are 500 image groups where the first image of each group is used as a query. The remaining 991 images are used as the reference images.

Stanford Mobile Visual Search (SMVS) dataset [50] contains 4,500 phone camera images of products, CDs, books, outdoor landmarks, business cards, text documents, museum paintings and video clips. The dataset has posed several challenges of recognizing rigid objects in the context of mobile imaging such as: widely varying lighting conditions, perspective distortion, clutter, realistic ground-truth reference data, and heterogeneous query data collected from low and high-end camera phones. There are 2,800 queries and 1,700 reference images.

FLICKR1M dataset contains 1 million distractor images, which are merged with the reference images of the benchmark datasets including UKBench, INRIA Holidays, and SMVS to evaluate the retrieval performance over a large scale database.

In addition, an independent image dataset, Oxford buildings [23], is used in all training stages, including PCA projection matrix training of RootSIFT, visual vocabulary training based on k-means or GMM clustering, bit selection model based on mutual information minimization, etc.

B. Evaluation Criteria

Search Accuracy. For all experiments we use the mean Average Precision (mAP) to measure the search accuracy. mAP is defined as follows:

$$mAP = \frac{1}{N_q}\sum_{i=1}^{N_q}\left(\frac{\sum_{r=1}^{N} P(r)}{\#\_relevant\_images}\right)$$

where N_q is the number of queries; N the number of relevant images for the ith query; $P(r)$ is the precision at rank r.

Besides mAP, we employ the Success rate for Top Match (STM) to measure the precision at rank 1, which is defined as: STM =(number of times the top retrieved image is relevant)/(number of queries).

Note that for UKbench dataset, we report the average number of relevant images in top 4 returns, i.e., $4 \times$ Recall@4, which is the commonly used measure over this dataset [22].

Search time cost is defined as

$$T = T_{buckets} + T_{candidates}$$

where $T_{buckets}$ denotes the time cost for visiting the buckets in multiple hash tables and $T_{candidates}$ the linear search time cost for Hamming distance computation on the candidate images to return the nearest neighbors.

Memory cost is measured by

$$S = S_{buckets} + S_{indices}$$

where $S_{buckets}$ and $S_{indices}$ denote the memory cost of buckets (addresses) and inverted indices in the buckets, respectively.

C. Comparison Baselines

Evaluation of "What to hash" and "How to search". We present group of comparisons to evaluate the technical advantages of the Image-specific Component Selection (denoted as H_{ics}), the globally optimized bit selection based on Mutual Information Minimization (denoted as H_{mim}), and the ADAptive relevance weighting (denoted as S_{ada}) for online search. A few alternatives in each stage are selected for comparison. (1) For H_{ics}, we setup two intuitive component selection strategies: RANdom Component selection (denoted as H_{ranc}), and TF-IDF based component selection [21] (denoted as H_{tfidf}); (2) For H_{mim}, we compare it with two bit selection baselines: RANdom Bit selection (denotes as H_{ranb}) adopted by the classical hashing algorithms LSH [12], and UNIform bit selection (denotes as H_{uni}) proposed by [16] to improve H_{ranb}; (3) For S_{ada}, we setup two weighting baselines for online search: e-bit

difference based on FIXed weights (denoted as S_{fix}) used in the EWH algorithm [20], and the IDF-based weights (denoted as S_{idf}).

Specifically, for "What to hash", we compare different component selection methods when bit selection is fixed (i.e., $H_{ics+mim}$, $H_{ranc+mim}$) and $H_{tfidf+mim}$), and different bit selection methods when component selection is fixed (i.e., $H_{ics+mim}$, $H_{ics+ranb}$ and $H_{ics+uni}$). Based on the optimal "What to hash", we evaluate "How to search" in terms of without weighting (i.e., $H_{ics+mim}$), and with different weighting schemes (i.e., $H_{ics+mim} + S_{ada}$, $H_{ics+mim} + S_{fix}$ and $H_{ics+mim} + S_{idf}$). In addition, we compare the performance of the proposed WeCoHash (i.e., $H_{ics+mim} + S_{ada}$) with the exhaustive linear search by using Euclidean distance over uncompressed aggregated descriptors, in which the latter generally serves as the performance upper bound of ANN search accuracy.

Comparison with the state of the art. We establish the state-of-the-art approaches for extensive comparison with the proposed WeCoHash, including (1) the exhaustive linear search on the whole database; (2) the LSH algorithm [12] with random hash keys generation and "collision" mechanism ($e = 0$) to generate candidate images; (3) the Uniform LSH algorithm [16] differing from LSH in uniformly generated random hash keys; (4) the EWH algorithm [20] differing from LSH in weighted e-bit difference ($e > 0$) search strategy; (5) the MIH algorithm [41] originally designed for exact nearest neighbors search; (6) the HKM [22] with depth 2 and branch 10 producing 100 visual words; (7) the PQ [40], each block consisting of 16 dimensions and 256 visual words setup for each block.

Note that we implemented LSH, Uniform LSH, EWH with C++ as there is no publicly available source code release. For MIH, we directly use the source code provided by the author. We utilize the implementation of HKM and PQ in the open source library VLFeat.[3]

D. Implementation Details

All images are converted to gray images. If at least one of the dimensions of the original image is greater than 640 pixels then the original image shall be spatially resampled, maintaining the aspect ratio, so that the largest of the vertical and horizontal image dimensions is equal to 640 pixels. We use a variant of SIFT descriptor, RootSIFT [42], which simply applies square root to each component of SIFT extracted by the VLFeat[3] library. The dimension of RootSIFT is reduced to $d = \{16, 32, 48, 64\}$, respectively, by applying different PCA project matrixes. We employ the state-of-the-art approaches FV and VLAD to aggregate the dim reduced RootSIFT local feature descriptors, with the number of visual words $k = 128$. k-means clustering or GMM model are applied to VLAD or FV. As introduced before, sign binarization is employed to compress the original FV (or VLAD) to binary aggregated descriptors with $l = kd$ bits. Except as expressly stated, a fixed setting of $k = 128, d = 64$ is applied across all experiments. All experiments are performed on a Dell Precision workstation 7400-E5440 with 2.83 GHz Intel XEON CPU and 16G of RAM in a mode of single core and single thread.

[3][Online]. Available: http://www.vlfeat.org

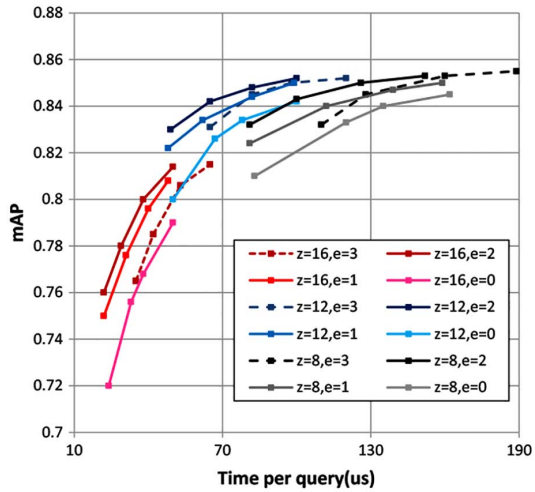


Fig. 6. Impact of key parameters of the proposed WeCoHash algorithm, including the number of selected components s, the number of selected bits z, and the search radius e, in terms of average search time (us) and retrieval mAP on the SMVS dataset. Note that the different numbers of selected components $s = \{50, 60, 70, 80\}$ (from left to right for each curve) are applied to different configuration settings of selected bit and search radius.

E. Impact of Parameters

We firstly evaluate the impact of key parameters of the proposed WeCoHash algorithm, including the number of selected components s, the number of selected bits z and the search radius e, in terms of average search time and retrieval mAP on the SMVS dataset, as shown in Fig. 6. Similar trends are obtained on the INRIA Holidays and UKbench datasets.

Number of selected components s. By fixing the number of selected bits z and the search radius e, as shown in Fig. 6, increasing the number of selected components has consistently improved the retrieval mAP while more search time cost is incurred. For example, when $z = 12, e = 2$, the retrieval mAP is improved from 83% to 85.2% while the search time increases from 49 us to 100 us, by increasing the number of selected components from 50 to 80.

Number of selected bits z. By fixing the number of selected components s and the search radius e, decreasing the number of selected bits z significantly increases the search time. When z becomes smaller, the number of entries (database image IDs) in each bucket is increased on average, which leads to a fast growing number of candidates for linear search. The Hamming distance computing of more candidates undoubtedly bring about considerable computation cost. When z becomes too large, the conflict probability between the hash values of query and reference images would decrease dramatically and fewer entries in each buckets result, which would seriously degenerate mAP. In other words, it is meaningful to figure out a proper number of selected bits.

Search radius e. As shown in Fig. 6, by fixing the number of selected component s and the number of selected bits z, one can see that the retrieval mAP is consistently improved with the

TABLE I
EVALUATION OF "WHAT TO HASH" AND "HOW TO SEARCH" FOR BOTH BINARIZED FV (BFV) AND BINARIZED VLAD (BVLAD) OVER VARIOUS DATASETS. THE INRIA HOLIDAYS AND SMVS DATASETS USE MAP, WHILE THE UKBENCH DATASET USES 4 × Recall@4

Methods		Holidays	SMVS	UKbench
Uncompressed FV		**0.665**	**0.872**	**3.36**
bFV	H_{ranb}	0.485	0.699	2.87
	H_{uni}	0.493	0.707	2.90
	$H_{ics+ranb}$	0.535	0.739	2.60
	$H_{ics+uni}$	0.561	0.765	2.81
	$H_{ics+mim}$	**0.597**	**0.800**	**2.91**
	$H_{ranc+mim}$	0.545	0.758	2.69
	$H_{tfidf+mim}$	0.577	0.781	2.89
	$H_{ics+mim} + S_{fix}$	0.625	0.835	3.07
	$H_{ics+mim} + S_{idf}$	0.641	0.849	3.11
	$H_{ics+mim} + S_{ada}$	**0.653**	**0.862**	**3.14**
Uncompressed VLAD		**0.631**	**0.851**	**3.25**
bVLAD	H_{ranb}	0.476	0.685	2.77
	H_{uni}	0.490	0.697	2.79
	$H_{ics+ranb}$	0.501	0.700	2.51
	$H_{ics+uni}$	0.528	0.731	2.72
	$H_{ics+mim}$	**0.561**	**0.771**	**2.93**
	$H_{ranc+mim}$	0.512	0.723	2.62
	$H_{tfidf+mim}$	0.550	0.747	2.78
	$H_{ics+mim} + S_{fix}$	0.590	0.802	2.98
	$H_{ics+mim} + S_{idf}$	0.605	0.809	3.02
	$H_{ics+mim} + S_{ada}$	**0.623**	**0.827**	**3.05**

search radius from $e = 0$ to $e = 2$. When increasing e from 2 to 3, the retrieval perform gain (mAP) is very minor, while search efficiency has decreased significantly due to the dramatic increase of buckets to visit, whereas more candidates still cannot be recalled.

Note that we observed a few cases of slight speed up when increasing search radius e (except $e = 3$). As we know, search consists of two stages: hashing based recall and exhaustive linear search over candidates. When e is very small, the time cost on buckets visiting (the first stage) is ignorable, while the time cost on exhaustive linear search (the second stage) would be effected by the number of candidate images for exhaustive search by setting a threshold t. The setup of different empirical thresholds may bring about some variance, which may lead to a few exceptional cases.

Referring to Fig. 6, we apply the optimal setting of $z = 12$ and $e = 2$ in the subsequent experiments, which yields desirable performance in terms of both search accuracy and search efficiency.

F. Evaluation of "What to Hash" and "How to Search"

In this section, we evaluate the performance of WeCoHash from the perspectives of "What to hash", "How to search", and the combination. Table I summarizes extensive comparison results of different combinations of empirical settings of "What to hash" and "How to search", for the state-of-the-art binary aggregated descriptors: binarized FV [5] and VLAD [2], [30].

What to hash. Referring to the listed results in Table I, the combination of component selection and bit selection significantly outperforms solely bit selection approaches H_{ranb} and H_{uni} without component selection. For $H_{ics+mim}$, $H_{ranc+mim}$ and $H_{tfidf+mim}$, different component selection methods are applied while the same bit selection mechanism H_{mim} is applied. From Table I, $H_{ics+mim}$ yields the best retrieval accuracy, e.g., mAP 80% for $H_{ics+mim}$ vs 78.1% for $H_{tfidf+mim}$ vs. 75.8% for $H_{ranc+mim}$ on the SMVS dataset with bFV. This has verified that the effectiveness of our image-specific component selection. In addition, for $H_{ics+ranb}$, $H_{ics+uni}$ and $H_{ics+mim}$, different bit selection methods are applied while the same component selection mechanism H_{ics} is applied. As listed in Table I, $H_{ics+mim}$ has significantly outperformed the rest, which has shown the effectiveness of our globally optimized bit selection.

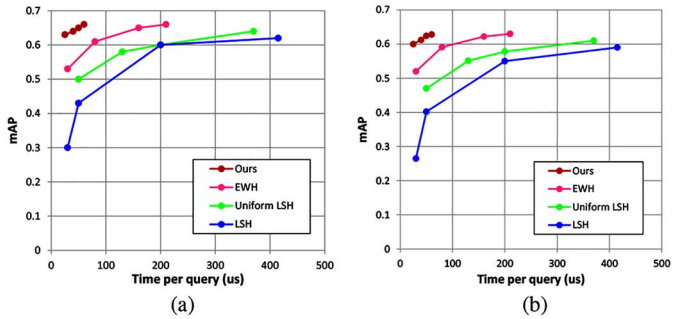


Fig. 7. Comparison in terms of mAP between different hashing methods on the INRIA Holidays dataset, with the binary aggregated descriptor bFV (a) and bVLAD (b), respectively.

How to search. From Table I, $H_{ics+mim}+S_{ada}$, $H_{ics+mim}+S_{fix}$ and $H_{ics+mim}+S_{idf}$ outperform $H_{ics+min}$ without using weighting, which means the weighting mechanism in online search can consistently improve the retrieval accuracy. In addition, $H_{ics+mim} + S_{ada}$ (i.e., the proposed WeCoHash) significantly outperforms $H_{ics+mim}+S_{fix}$ and $H_{ics+mim}+S_{idf}$, e.g., mAP 62.3% for $H_{ics+mim}+S_{ada}$ vs 59% for $H_{ics+mim}+S_{fix}$ and 60.5% for $H_{ics+mim}+S_{idf}$ on the INRIA Holidays dataset with bVLAD.

Finally, we add an important comparison between the proposed WeCoHash (with the binary aggregated descriptor) and the exhaustive linear search with Euclidean distance (with uncompressed aggregated descriptor). From Table I, the WeCoHash obtains comparable search accuracy with the uncompressed FV or VLAD. This demonstrates the effectiveness of WeCoHash.

G. Comparison With the State of the Art

In this section, we firstly compare the proposed WeCoHash approach with the state-of-the-art hashing techniques over the aggregated descriptors of binarized FV and binarized VLAD which encode the high-order statistics of local feature distribution in a probabilistic and non-probabilistic way, respectively. Secondly, regarding the state-of-the-art hashing techniques, we further study the effects of dimensionality of binary aggregated descriptors and image database size on search speedup, when retrieval accuracy is maintained comparable as exhaustive linear search. Finally, we extend the experiments to a large scale setting with the 1 million FLICKR1M dataset and analyze search speedup as well as memory footprint.

Performance comparison. Fig. 7, Fig. 8 and Fig. 9 show the retrieval mAP vs search time on the datasets of INRIA Holidays, SMVS and UKbench, respectively. For fair comparison with LSH, Uniform LSH, and EWH, we apply the same number of hash tables as well as the same length of hash value for all hashing algorithms. Optimal parameters tuning is allowed

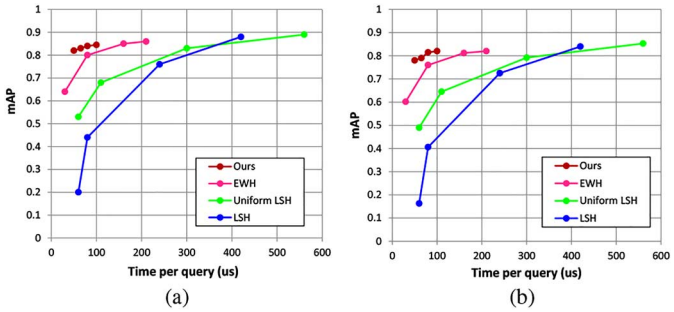


Fig. 8. Comparison in terms of mAP between different hashing methods on the SMVS dataset, with the binary aggregated descriptor bFV (a) and bVLAD (b), respectively.

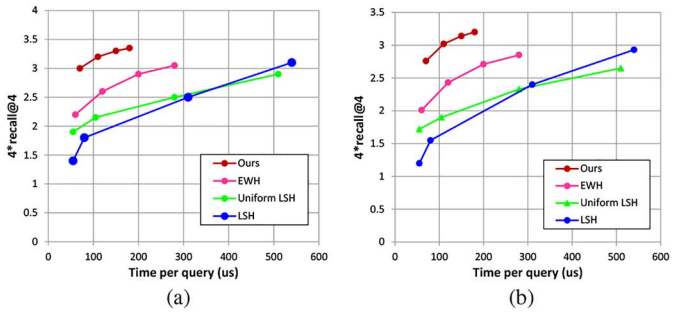


Fig. 9. Comparison in terms of $4 \times$ Recall@4 between different hashing methods on the UKbench dataset, with the binary aggregated descriptor bFV (a) and bVLAD (b), respectively.

for each baseline. As shown in Fig. 7, on the INRIA Holidays dataset, at comparable retrieval mAP, our WeCoHash approach is up to $5 \sim 7$ times faster than LSH and Uniform LSH, and $2 \sim 3$ times faster than EWH. At comparable search time, our WeCoHash approach achieves much better retrieval mAP than LSH and Uniform LSH (+20.0%) and EWH (+10%). Similar performance gains and search speedup can be observed on SMVS and UKBench in Fig. 8 and 9 as well.

Effects of the dimensionality of binary aggregated descriptors on search efficiency. Fig. 10(a) presents the search time vs descriptor dimensionality of WeCoHash, LSH, Uniform LSH, EWH and Linear Search over the UKbench dataset combined with 1 million FLICKR1M. We apply a fixed number of components (visual words) $k = 128$, and different dimensionality of binary aggregated descriptors $d \in \{16, 32, 48, 64\}$. The optimal parameters are setup for all hashing algorithms to obtain comparable retrieval mAP (mAP drop $< 1\%$) with linear search. As shown in Fig. 10(a), the time cost of linear search increases linearly with the dimensionality of binary aggregated descriptors, while the time cost of hashing algorithms increases slowly (sub-linearly) with the dimensionality of binary aggregated descriptors, especially for WeCoHash. For instance, when the dimension increases from 16×128 to 48×128, the search speed decreases by 2.2 times (from 0.04s to 0.09s) for WeCoHash vs 2.7 times (from 0.37s to 0.98s) for LSH.

Effects of database size on search efficiency. Fig. 10(b) shows the growth of search time with the database size when applying WeCoHash, LSH, Uniform LSH, EWH and linear search on the UKbench dataset combined with different number of distractors randomly selected from FLICKR1M, in which the parameter

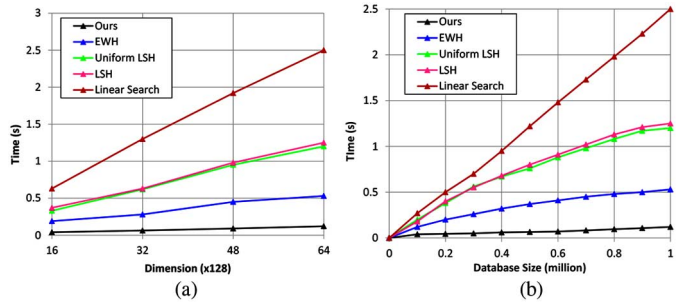


Fig. 10. (a) Effects of the dimensionality of binary aggregated descriptors on search time cost when the visual words size $k = 128$ and the dimension d varies from 16 to 64. (b) Effects of database size on search time cost. The experiments are evaluated over the UKbench dataset combined with different number of distractors from FLICKR1M.

tuning of each hashing algorithm are allowed to yield comparable retrieval mAP (mAP drop $< 1\%$) with linear search. As shown in Fig. 10(b), the search time cost of linear search increases linearly with the size of database. Different from linear search, the search speed of hashing algorithms decreases slower as the size of database becoming large, especially for WeCoHash. For example, when the database size increases from 100 K to 500 K, the search speed decreases by 1.6 times (from 0.04s to 0.065s) for WeCoHash vs 3.1 times (from 0.12s to 0.37s) for EWH.

Performance evaluation over large-scale experiments. Table II compares the retrieval accuracy in terms of mAP (or $4 \times$ Recall@4), Top Match (STM) as well as search time (in seconds) of the proposed WeCoHash approach against LSH, Uniform LSH, EWH, MIH, HKM, PQ and Linear Search over various types of datasets combined with 1 million FLICKR1M. Both binarized VLAD (bVLAD) and binarized FV (bFV) are evaluated.

Firstly, WeCoHash is significantly faster than linear search over all datasets, with comparable retrieval accuracy. For instance, WeCoHash obtains over 20 times speedup than linear search at the cost of a minor STM drop $\sim 0.6\%$ on SMVS dataset for bVLAD. Secondly, WeCoHash outperforms LSH, Uniform LSH and EWH in terms of mAP(or $4 \times$ Recall@4), respectively. For example, compared to LSH, WeCoHash improves the measure of $4 \times$ Recall@4 from 3.06 to 3.20 on UKbench dataset for bFV. More importantly, WeCoHash is up to 10 times faster than LSH and Uniform LSH, and up to 5 times faster with EWH. Finally, compared to clustering algorithms such as HKM and PQ, WeCoHash performs significantly better than HKM (e.g., +18% mAP on Holidays dataset) with comparable search time, and obtains over 20 times speedup than PQ (e.g., 0.09s for WeCoHash vs. 3.08s for PQ on UKbench dataset) with comparable search accuracy. This demonstrates that the proposed WeCoHash is more suitable for ANN search with binary descriptors.

Memory cost. Table III shows the memory cost of hash tables for indexing 1 million FLICKR1M dataset with WeCoHash, LSH, Uniform LSH, EWH and MIH. As shown in Table III, thanks to the component selection, the memory use of WeCoHash is much less than the baselines, which is $\sim 50\%$ less than LSH, Uniform LSH and EWH, and 30% less than MIH.

TABLE II
COMPARISON OF THE PROPOSED WECOHASH, LSH, UNIFORM LSH, EWH, MIH, HKM, PQ AND LINEAR SEARCH IN TERMS OF MAP (OR 4 × Recall@4) AS WELL AS TOP MATCH (STM) AND SEARCH TIME IN SECONDS OVER VARIOUS TYPES OF DATASETS COMBINED WITH FLICKR1M. BOTH BINARIZED VLAD (BVLAD) AND BINARIZED FV (BFV) ARE EVALUATED

		UKbench			INRIA Holidays			SMVS		
Feature	Method	$4 \times recall@4$	STM	time (s)	mAP	STM	time (s)	mAP	STM	time (s)
bFV	Linear Search	3.25	94.4	2.59	63.59	69.85	2.23	83.56	85.6	2.75
	HKM	2.12	61.5	0.12	45.2	47.8	0.12	53.7	56.5	0.12
	PQ	3.10	93.25	3.08	62.53	68.15	2.98	83.24	84.9	3.45
	LSH	3.06	93.17	1.18	62.69	68.88	1.06	82.84	84.3	1.32
	Uniform LSH	3.10	93.33	1.17	62.58	68.72	1.00	82.94	84.3	1.37
	EWH	3.18	93.56	0.56	61.23	67.99	0.43	82.59	84.0	0.60
	MIH	3.25	94.4	0.57	63.50	69.80	0.53	83.50	85.6	0.55
	Ours	3.20	93.81	**0.09**	63.50	69.5	**0.14**	82.17	85.1	**0.13**
bVLAD	Linear Search	3.19	92.8	2.63	62.23	68.2	2.41	82.77	83.9	2.67
	HKM	2.01	58.7	0.09	43.7	45.6	0.11	50.1	53.5	0.15
	PQ	2.98	91.21	3.18	60.15	65.23	2.95	82.24	83.9	3.27
	LSH	3.00	92.3	1.09	61.15	67.5	1.25	82.00	83.2	1.30
	Uniform LSH	2.95	92.2	1.04	61.35	67.4	1.15	82.30	83.3	1.37
	EWH	2.92	91.9	0.49	60.46	67.1	0.47	82.24	82.5	0.51
	MIH	3.10	92.8	0.56	62.20	68.2	0.50	82.77	83.9	0.56
	Ours	3.15	92.5	**0.11**	62.05	67.9	**0.08**	82.19	83.3	**0.12**

TABLE III
MEMORY COST OF CONSTRUCTING HASH TABLES TO INDEX 1 MILLION FLICKR1M DATASET WITH HASHING METHODS

	WeCoHash	LSH	Uniform LSH	EWH	MIH
Memory (Mb)	**339**	610	635	590	513

TABLE IV
PERFORMANCE OF STATE-OF-THE-ART VISUAL SEARCH METHODS ON INRIA HOLIDAY DATASET. IN THIS TABLE, MA INDICATES IF A METHOD APPLIES WITH MULTIPLE ASSIGNMENT AND $|D|$ INDICATES THE VOCABULARY SIZE

	MA	$\|D\|$	Accuracy	Complexity
VLAD		64	55.6	$121ms$
Fisher		64	59.5	$132ms$
BoW		200K	54.0	$< 1ms$
HE[27]		65K	51.7	$< 1ms$
HE[27]	√	65K	56.1	$< 1ms$
HE-BURST[58]		65K	64.5	$< 1ms$
HE-BURST[58]	√	65K	67.4	$< 1ms$
Fine Vocabulary[59]		65K	74.2	$< 1ms$
ASMK[28]	√	65K	82.2	$2.5s$
ASMK*[28]	√	65K	81.0	$2.3s$
TE+DA[47]		64	77.1	$18.9s$
VLAD+WeCoHash		128	66.7	$< 1ms$
VLAD+WeCoHash	√	100K	80.5	$< 1ms$

Accuracy versus efficiency. Our WeCohash mainly focuses on the search efficiency which aims to accelerate the search speed of the aggregated descriptors. As accuracy is always important for visual search, we further study the results of WeCohash from the perspectives of both search accuracy and search efficiency by comparison with state-of-the-art visual search methods. Table IV shows the results of state-of-the-art visual search methods on INRIA Holiday dataset. From Table IV, we find that the raw VLAD and Fisher Vector (FV) cost much time due to the nature of high dimensionality. By utilizing the inverted list for indexing image features, BoW and its varieties (such as HE [26], BURST [57] and Fine Vocabulary [58]) are much more efficient on search speed. Recently, researchers have imposed kernal technologies on VLAD and FV to further improve retrieval performance. Two typical kernal methods ASMK [27] and TE+DA [46] have shown their advantages to improve search accuracy. As shown in Table IV, ASMK(ASMK{\ast}) [27] with multiple assignment achieves 82.2%(81.0%) mAP and TE+DA [46] achieves 77.1% mAP on INRIA Holiday dataset. Compared with Bow and its varieties, kernal technologies bring about better search accuracy. Unfortunately, both of ASMK(ASMK{\ast}) and TE+DA would introduce much more additional computational complexity. In a single query, ASMK costs about 2.5 seconds and TE+DA even costs up to 18.9 seconds, which are much more time consuming than other methods. Our WeCohash algorithm achieves a promising search accuracy of 80.5% mAP on INRIA Holidays at vocabulary size 100 K. This is just slightly worse than ASMK(ASMK{\ast}), but better than other methods. What's more, WeCohash doesn't incur noticeable complexity which may maximize the joint strength in terms of search efficiency and accuracy. In WeCoHash, a query can be finished within just 1 ms on average. Considering the joint benefits of high accuracy and more efficiency, WeCoHash deserves one of the most promising and practically useful algorithms in dealing with large scale visual search.

V. CONCLUSIONS

Aggregating the statistics of local feature distribution is the state-of-the-art approach to scalable visual search. In this paper, we have proposed an emerging problem of approximate nearest neighbor search with a long binary aggregated descriptor. We have proposed a novel WeCoHash algorithm to address the ANN problem with the long binary aggregated descriptor from two aspects of "What to hash" and "How to search". Accordingly, the efficient and effective WeCoHash solution has been developed for binary FV and VLAD descriptors. The proposed WeCoHash implementation has been rigorously validated in the evaluation framework of the emerging MPEG CDVS standard. The CDVS adopted WeCoHash technique is meant to address the scalability issue of compact binary aggregated descriptors towards fast visual search. In particular, more extensive comparison experiments have studied the impact of key elements, including discriminative components, bits de-correlation, hash value adaptive weighting, etc., with a high-dimensional binary aggregated descriptor in accomplishing high performance and low complexity visual search.

APPENDIX

Let b denotes a binary vector, $b(i)$ and $b(j)$ denotes the i-th and j-th bit element respectively. The entropy $H(b_j)$, mutual

information $I(b_i; b_j)$ and joint entropy $H(b_i, b_j)$ are defined as follows:

$$H(b_j) = -\sum_{x\in\{0,1\}} p(x)\log_2 p(x),$$
$$H(b_i, b_j) = H(b_i) + H(b_j) - I(b_i; b_j),$$
$$I(b_i; b_j) = -\sum_{x\in\{0,1\}}\sum_{y\in\{0,1\}} p(x,y)\log\left(\frac{p(x,y)}{p(x|y)p(y|x)}\right)$$

where $p(x)$ denotes the probability of element x.

REFERENCES

[1] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2007, pp. 1–8.

[2] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2010, pp. 3304–3311.

[3] D. G. Lowe, "Distinctive image features from scale invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[4] H. Bay *et al.*, "SURF: Speeded up robust features," in *Proc. ECCV*, 2006, pp. 404–417.

[5] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, "Large-scale image retrieval with compressed Fisher vectors," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2010, pp. 3384–3391.

[6] D. Chen *et al.*, "Residual enhanced visual vector as a compact signature for mobile visual search," *Signal Process.*, vol. 93, no. 8, pp. 2316–2327, 2013.

[7] J. Lin *et al.*, "Rate-adaptive compact fisher codes for mobile visual search," *IEEE Signal Process. Lett.*, vol. 21, no. 2, pp. 195–198, Feb. 2014.

[8] *Evaluation Framework for Compact Descriptors for Visual Search*, ISO/IEC JTC1/SC29/WG11/N12202, 2011.

[9] *Text of ISO/IEC DIS 15938-13 Compact Descriptors for Visual Search*, ISO/IEC JTC1/SC29/WG11/W14392, 2014.

[10] B. Girod *et al.*, "Mobile visual search," *IEEE Signal Process. Mag.*, vol. 28, no. 4, pp. 61–76, Jul. 2011.

[11] W. Zhou *et al.*, "Towards codebook-free: Scalable cascaded hashing for mobile image search," *IEEE Trans. Multimedia*, vol. 28, no. 4, pp. 61–76, Jul. 2014.

[12] A. A. and P. I. , "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Proc. FOCS*, 2006, pp. 459–468.

[13] M. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proc. STOC*, 2002, pp. 380–388.

[14] C. Silpa-Anan and R. Hartley, "Optimised KD-trees for fast image descriptor matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2008, 2008, pp. 1–8.

[15] L.-Y. Duan, J. Lin, J. Chen, T. Huang, and W. Gao, "Compact descriptors for visual search," *IEEE Multimedia Mag.*, vol. 21, no. 3, pp. 30–40, Jul.–Sep. 2014.

[16] T. Trzcinski *et al.*, "Thick boundaries in binary space and their influence on nearest-neighbor search," *Pattern Recog. Lett.*, vol. 33, no. 16, pp. 2173–2180, Dec. 2012.

[17] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. IEEE Int. Conf. Comput. Vis.*, Nov. 2011, pp. 2564–2571.

[18] S. Zhang, Q. Tian, Q. Huang, W. Gao, and Y. Rui, "USB: Ultrashort binary descriptor for fast visual matching and retrieval," *IEEE Trans. Image Process.*, vol. 23, no. 8, pp. 3671–3683, Aug. 2014.

[19] M. Calonder *et al.*, "Brief: Binary robust independent elementary features," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 778–792.

[20] M. M. Esmaeili, R. K. Ward, and M. Fatourechi, "A fast approximate nearest neighbor search algorithm in the hamming space," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 2481–2488, Dec. 2012.

[21] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2003, vol. 2, pp. 1470–1477.

[22] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2006, vol. 2, pp. 2161–2168.

[23] J. Philbin, O. Chum, and M. Isard *et al.*, "Object retrieval with large vocabularies and fast spatial matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2007, pp. 1–8.

[24] 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. IEEE Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.

[25] 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. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2008, pp. 1–8.

[26] H. Jégou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *Proc. ECCV*, 2008, pp. 304–317.

[27] G. Tolias, Y. Avrithis, and H. Jégou, "To aggregate or not to aggregate: Selective match kernels for image search," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 1401–1408.

[28] H. Jégou, M. Douze, and C. Schmid, "Improving bag-of-features for large scale image search," *Int. J. Comput. Vis.*, vol. 87, no. 3, pp. 316–336, May 2010.

[29] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, 2001.

[30] H. Jégou *et al.*, "Aggregating local images descriptors into compact codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1704–1716, Sep. 2012.

[31] F. Perronnin and H. Jégou, "CVPR tutorial: Large-scale visual recognition," presented at the IEEE Conf. Comput. Vis. Pattern Recog. Tut., Jun. 2012.

[32] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. NIPS*, 2008, pp. 1753–1760.

[33] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large databases for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2008, pp. 1–8.

[34] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2011, pp. 817–824.

[35] M. Norouzi and D. Fleet, "Minimal loss hashing for compact binary codes," in *Proc. ICML*, 2011, pp. 353–360.

[36] R. Ji *et al.*, "Location discriminative vocabulary coding for mobile landmark search," *Int. J. Comput. Vis.*, vol. 96, no. 3, pp. 290–314, Feb. 2012.

[37] Y.-G. Jiang, J. Wang, X. Xue, and S.-F. Chang, "Query-adaptive image search with hash codes," *IEEE Trans. Multimedia*, vol. 15, no. 2, pp. 442–453, Feb. 2013.

[38] T. Jaakkola and D. Haussler, "Exploiting generative models in discriminative classifier," in *Proc. NIPS*, 1999, pp. 487–493.

[39] Y. Gong, S. Kumar, H. Rowley, and S. Lazebnik, "Learning binary codes for high-dimensional data using bilinear projections," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 484–491.

[40] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, Jan. 2011.

[41] M. Norouzi, A. Punjani, and D. Fleet, "Fast search in hamming space with multi-index hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 3108–3115.

[42] R. Arandjelovic and A. Zisserman, "Three things everyone should know to improve object retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 2911–2918.

[43] R. Arandjelovic and A. Zisserman, "All About VLAD," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 1578–1585.

[44] M. Douze, A. Ramisa, and C. Schmid, "Combining attributes and Fisher vectors for efficient image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2011, pp. 745–752.

[45] J. Delhumeau *et al.*, "Revisiting the VLAD image representation," in *Proc. ACM Multimedia*, 2013, pp. 653–656.

[46] H. Jégou and A. Zisserman, "Triangulation embedding and democratic aggregation for image search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2014, pp. 3310–3317.

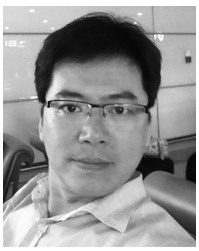
[47] T. Cover and J. Thomas, *Elements of Information Theory*. New York, NY, USA: Wiley, 1991.

[48] F. Fleuret, "Fast binary feature selection with conditional mutual information," *J. Mach. Learn. Res.*, vol. 5, pp. 1531–1555, Dec. 2004.

[49] G. Tolias, T. Furon, and H. Jégou, "Orientation covariant aggregation of local descriptors with embeddings," in *Proc. ECCV*, 2014, pp. 382–397.

[50] V. Chandrasekhar *et al.*, "The Stanford mobile visual search dataset," in *Proc. ACM Multimedia Syst. Conf.*, 2011, pp. 117–122.

[51] J. Friedman, J. Bentley, and R. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, no. 3, pp. 1977209–226.
[52] C. Silpa-Anan and R. Hartley, "Optimised KD-trees for fast image descriptor matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2008, pp. 1–8.
[53] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization for approximate nearest neighbor search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 2946–2953.
[54] Y. Kalantidis and Y. Avrithis, "Locally optimized product quantization for approximate nearest neighbor search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2014, pp. 2329–2336.
[55] K. He, F. Wen, and J. Sun, "K-means hashing: An affinity-preserving quantization method for learning binary compact codes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 2938–2945.
[56] J.-P. Heo, Z. Lin, and S.-E. Yoon, "Distance encoded product quantization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2014, pp. 2139–2146.
[57] M. Jain *et al.*, "Asymmetric hamming embedding: Taking the best of our bits for large scale image search," in *Proc. ACM Multimedia*, 2011, pp. 1441–1444.
[58] A. Mikulik *et al.*, "Learning a fine vocabulary," in *Proc. ECCV*, 2010, pp. 1–14.



Ling-Yu Duan (M'09) received the M.Sc. degree in automation from the University of Science and Technology of China, Hefei, China, in 1999, the M.Sc. degree in computer science from the National University of Singapore, Singapore, in 2002, and the Ph.D. degree in information technology from The University of Newcastle, Callaghan, N.S.W., Australia, in 2007.

From 2003 to 2008, he was a Research Scientist with the Institute for Infocomm Research, Singapore. Since 2008, he has been with Peking University, Beijing, China, where he is currently an Associate Professor with the School of Electrical Engineering and Computer Science. He is the leader of the Visual Search Group with the Institute of Digital Media, Peking University. Since 2012, he has been Deputy Director with the Rapid-Rich Object Search (ROSE) Laboratory, a joint laboratory between Nanyang Technological University, Singapore, and Peking University, Beijing, China. He has authored or coauthored more than 100 publications. His research interests include visual search and augmented reality, multimedia content analysis, and mobile media computing.

Dr. Duan is a member of the Association for Computing Machinery.

Jie Lin received the Ph.D. degree from the School of Computer Science and Technology, Beijing Jiaotong University, Beijing, China, in 2014.

He was a visiting Ph.D. student with the School of EEE, Nanyang Technological University, Singapore, in 2010, and the Institute of Digital Media, Peking University, Beijing, China, from 2011 to 2013. He was a Research Engineer with the Rapid-Rich Object Search Laboratory, Nanyang Technological University, Singapore, in 2014. He is currently a Research Scientist with the Institute of Infocomm Research, Singapore. His research interests include mobile visual search, large scale image/video retrieval, and deep learning.

Zhe Wang received the Bachelor degree in software engineering from Beijing Jiaotong University, Beijing, China, in 2012, and is currently working towards the M.S. degree at the School of Electrical Engineering and Computer Science, Peking University, Beijing, China.

His current research interests include large-scale image retrieval and fast approximate nearest neighbor search.

Tiejun Huang (M'01–SM'12) received the B.S. and M.S. degrees in automation from Wuhan University of Technology, Wuhan, China, in 1992, and the Ph.D. degree from the School of Information Technology and Engineering, Huazhong University of Science and Technology, Wuhan, China, in 1999.

He was a Postdoctoral Researcher and a Research Faculty Member with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, from 1999 to 2001. He was also the Associated Director (from 2001 to 2003) and the Director (from 2003 to 2006) of the Research Center for Digital Media, Chinese Academy of Sciences. He is currently a Professor with the National Engineering Laboratory for Video Technology, School of Electronics Engineering and Computer Science, Peking University, Beijing, China. His research interests include digital media technology, digital library, and digital rights management.

Dr. Huang is a member of the Association for Computing Machinery.

Wen Gao (S'87–M'88–SM'05–F'09) received the Ph.D. degree in electronics engineering from the University of Tokyo, Tokyo, Japan, in 1991.

He was a Professor of Computer Science with the Harbin Institute of Technology, Harbin, China, from 1991 to 1995, and a Professor with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. He is currently a Professor of Computer Science with the Institute of Digital Media, School of Electronic Engineering and Computer Science, Peking University, Beijing, China.

Dr. Gao has served on the Editorial Boards for several journals, such as the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE TRANSACTIONS ON AUTONOMOUS MENTAL DEVELOPMENT, the *Eurasip Journal of Image Communications*, and the *Journal of Visual Communication and Image Representation*. He has chaired a number of prestigious international conferences on multimedia and video signal processing, such as IEEE ICME and ACM Multimedia, and also served on the advisory and technical committees of numerous professional organizations.