Scalable Image Search with Multiple Index Tables

Junjie Cai[†,*], Qiong Liu[‡], Francine Chen[‡], Dhiraj Joshi[‡] and Qi Tian[†]
[†]Department of Computer Science, University of Texas at San Antonio
[‡]FX Palo Alto Laboratory, California
{caijunjieustc}@gmail.com, {liu,chen,dhiraj}@fxpal.com, qitian@cs.utsa.edu

ABSTRACT

Motivated by scalable partial-duplicate visual search, there has been growing interest in a wealth of compact and efficient binary feature descriptors (e.g. ORB, FREAK, BRISK). Typically, binary descriptors are clustered into codewords and quantized with Hamming distance, following the conventional bag-of-words strategy. However, such codewords formulated in Hamming space do not present obvious indexing and search performance improvement as compared to the Euclidean codewords. In this paper, without explicit codeword construction, we explore the use of partial binary descriptors as direct codebook indices (addresses). We propose a novel approach to build multiple index tables which concurrently check for collision of the same hash values. The evaluation is performed on two public image datasets: DupImage and Holidays. The experimental results demonstrate the indexing efficiency and retrieval accuracy of our approach.

Categories and Subject Descriptors

H.3.3 [**Information Retrieval**]: Information Retrieval and Indexing

General Terms

Algorithm, Measurement, Experimentation

Keywords

Image search, Multiple index tables, Binary features

1. INTRODUCTION

With the rapid advance in digital techniques, billions of images have been shared online. To find images containing a particular object or partially similar object using a query image is like finding a needle in a haystack[12][10]. In large-scale image retrieval systems, many state-of-the-art approaches leverage scalable textual retrieval techniques (e.g. bag-of-words) for image search[4][3][7]. Similar to text words in information retrieval, local features (e.g. SIFT[1]) are quantized into visual words, where all visual words constitute a visual codebook. However, conventional feature quantization has several drawbacks: a) **Necessity of visual codebook training:** the visual codebook has to be trained in advance and the training process is computationally expensive, especially with a large amount of sample features. b) **Limited reliability**: the effectiveness of codebook construction relies on the collection of image features and codebook generation methods (e.g. Vocabulary Tree[8], k-means). c) **Update inefficiency**: with many new features collected, updating a large visual codebook requires a wealth of effort and re-clustering features is computationally inefficient[20][21].

Instead of being trapped in such dilemmas, researchers have explored many different ways to improve search performance[5][13][16][17][9]. Jegou *et al.*[5] proposed the Vector of Locally Aggregated Descriptors(VLAD) which are derived from local descriptors to replace the bag-of-words histogram. However, search results of the VLAD pipeline are severely limited to the cluster center of training sets. Any cluster inconsistency can severely impact search performance. Zhou *et al.*[13] proposed a scalar quantization strategy, which transforms a conventional SIFT feature to a binary bit vector by setting a threshold on each dimension. Their results utilize and preserve the Euclidean similarity between SIFT feature pairs to some extent. However, the threshold is not easily chosen among various image datasets. A joint indexing approach[9] was shown to optimize the quantization process. They indicate that multiple quantizers are effective in improving the indexing recall. Moreover, there has been growing interest in many compact and efficient binary feature descriptors(e.g. ORB[11], FREAK[14], BRISK[15]) for visual matching applications. Typically, binary local descriptors are clustered into codewords and quantized with Hamming distance, following the conventional bag-of-words strategy. However, such codewords formulated in Hamming space do not present obvious indexing and search efficiency improvements as compared to the Euclidean ones.

In this paper, without explicit codeword training, we explore the use of partial binary descriptors as direct codebook indices (addresses). Intuitively, if a codeword address is represented with t bits, the maximum total number of codewords or index table size is 2^t. However, the number

[*]This work was done when Junjie Cai was an intern at FX Palo Alto Laboratory.

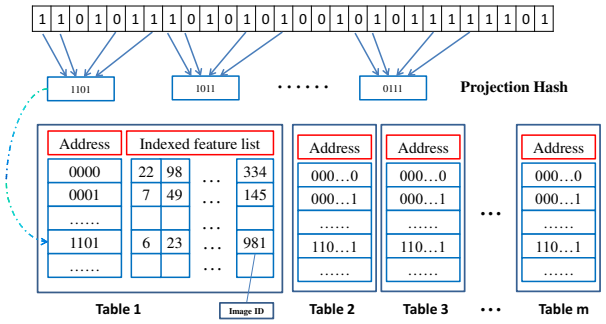

Figure 1: Flowchart of adding a binary descriptor to the index table.

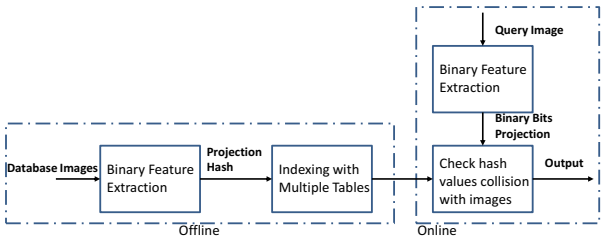

Figure 2: Flowchart of our proposed search strategy based on binary descriptors.

of codewords needed for 1 million image dataset is normally much less than the table size when the binary descriptor length is longer than 32 bits. Inspired in part by [13], we investigate building multiple index tables corresponding to various parts of binary bits. Such a strategy highly preserves sufficient similarity in Hamming space among binary feature pairs. Figure 1 illustrates a toy example of our proposed approach.

The main contributions of this paper can be summarized as follows:

- Instead of explicit codewords, we explore the use of partial binary descriptors as direct codebook addresses.
- We propose a novel approach to building multiple index tables which concurrently check for partial collision of binary features.

The rest of this paper is organized as follows. In Section 2, we describe the formulation of our proposed method, including constructing multiple index tables and indexing with the inverted-file structure. Experimental results on two publicly available data sets are reported in Section 3, followed by the conclusions in Section 4.

2. SEARCH WITH MULTIPLE INDEX TABLES

In this section, we illustrate our approach to two aspects: constructing multiple index tables offline and searching with inverted file structure in real-time. Figure 2 illustrates the flowchart of our proposed search strategy.

2.1 Multiple index table construction

Assume that the length of binary visual descriptor x is l, each one is partitioned into m disjoint binary substrings, $x^1, \ldots, x^m$ and each of length $\lfloor l/m \rfloor$ bits. We suppose that the binary bits are evenly distributed in various substrings. Given a dataset of binary descriptors, let m index tables be created for initialization. We conduct a projection from each binary feature to equal segments in the dataset which is denoted as "projection hash". Any complex hash function could be incorporated into our proposed pipeline, while in this paper we adopt the simplest one and select l/m bits to form m hash vectors in left-to-right order. The hash substrings determine where a certain binary descriptor should be stored in the index table. The descriptors which have identical hash vectors are stored in the same entry of the index table. As an entry may contain more than one descriptor, it is referred to as a codeword. Each index table has M codewords, corresponding to M different values that a t-bit hash vector can have. Ideally, $M = 2^t$ and $t = l/m$, but if t is large, the table size can be too large to be handled. Fortunately, a single index table can contain up to 2^{32} entries when t is set to 32 while the number of unique codewords after projection is much less than 10^8. In this paper, for a 256-bit ORB binary feature, we choose 8 index tables and t is 32 bits. For a 512-bit FREAK binary feature, we set m, t be equal to 16 and 32 respectively.

Motivated by previous work on multi-index hashing[19], we cut the long codes into smaller segments and build multiple index tables using these substrings as addresses. For each table, the dimension of Hamming space and radius of the Hamming ball to be searched are only a fraction of that without projection hash, which greatly reduces the number of buckets to be checked. During the feature matching for each table, threshold r decreases to r/m accordingly and the candidate result must satisfy the following condition in at least one substring k, $1 \leq k \leq m$, such that

$$\left\| x^k - y^k \right\|_H \leq r/m \tag{1}$$

where $\|\cdot\|_H$ denotes Hamming norm and the proof can be found in [19]. Taking the previous example, suppose we separate each 256-bit binary code into 8 segments of 32-bit codes, the total number of buckets to be checked is much less than a single table using the whole binary code as indices. Algorithm 1 summarizes the process of constructing index tables.

2.2 Search with the inverted-file structure

In image search, the problem of feature matching between images can be considered as finding a feature's nearest or approximately nearest neighbors within a certain range[22]. When the feature amount becomes very large, say, over one million, it is too computationally expensive to find the nearest neighbors by linearly comparing all features' binary vectors. To address this issue, an inverted file structure, which is leveraged from text retrieval, can be used for scalable indexing of a large-scale image dataset.

In traditional inverted-file structure for image search, a group of visual words are pre-trained. Each visual word is followed with an entry list of image features which were quantized to visual word. Each indexed feature in the list records its image ID and some other clues[2]. In our work, for a descriptor q in a query image, we divide it into m substrings $\{q^i\}_{i=1}^m$. For the i^{th} substring, we search the corresponding index table for entries within a Hamming distance

Algorithm 1 Indexing with Multiple Hash Table

Input: A set of distinct binary descriptors with l-bit length,
Output: Constructed m index tables, each with 2^t rows, where $t = l/m$

1: Initialize similarity score to 0 for all the descriptors in the image data set.
2: **repeat**
3: For a l-D binary descriptor x from an image to be indexed, each one is partitioned into m disjoint binary substrings $x^1, x^2, \ldots, x^m$.
4: Identify codeword position of individual index table by each substring of binary descriptor.
5: In the inverted image list, store the ID of the image where the descriptor appears.
6: **until** binary features of all the images have been indexed

r/m of q^i and retrieve candidates $\mathcal{N}_i(q)$. The union of the m multiple tables $\mathcal{N}(q) = \bigcup_i \mathcal{N}_i(q)$ constitute matching results. When r increases, more candidate true matches could be included. But when r is too large, many noisy matches are also included and therefore degrade the search performance. The parameter r was empirically set to 32.

Given a query image, matching images are looked up in the multiple index tables. We formulate the image retrieval as a voting problem. Each verified feature casts a vote to the corresponding target image. We define the similarity between two images by cardinality of matched feature set. Finally, the target images are ranked by their similarity scores and returned to the users as retrieved results.

3. EXPERIMENTS

We conducted experimental comparisons with several state-of-the-art approaches, including Vocabulary Tree(VT)[8] and VLAD[5]. We also evaluate performance using only the first index table instead of multiple ones and call it Single Index Table. The evaluation is performed on two publicly available datasets, DupImage[6] and Holidays[7]. We implemented the experiments on a server with a 16-core 2.4GHz CPU and 16 GB memory. Similar to [3][5], mean Average Precision(mAP) is adopted to evaluate the performance of all methods.

3.1 Evaluation on DupImage Dataset

The DupImage dataset contains 1104 images collected from 33 groups, including Mona Lisa, StarBucks Logo etc. We mix the dataset with one million images crawled from the Web. To evaluate the performance with respect to the size of dataset, we construct three smaller dataset ($50K$, $200K$, and $500K$) by sampling the dataset. For baseline approaches of Vocabulary Tree and VLAD, the following parameter setting is utilized: we adopt one-million visual codebook(a vocabulary tree with branch factor as 10 and level number as 6). And for VLAD, the dictionary size is 64 and the parameter of power-law normalization is 0.5. As aforementioned, we utilize 256-bit ORB descriptor to build 8 indexing tables, labeled as Multiple Index Table. Figure 3 provides the performance comparison of different approaches in terms of mAP. We can see that our scheme based on the ORB-feature outperforms both VT and VLAD approaches. Our approach demonstrates consistent superiority over conventional approaches for all tested dataset sizes.

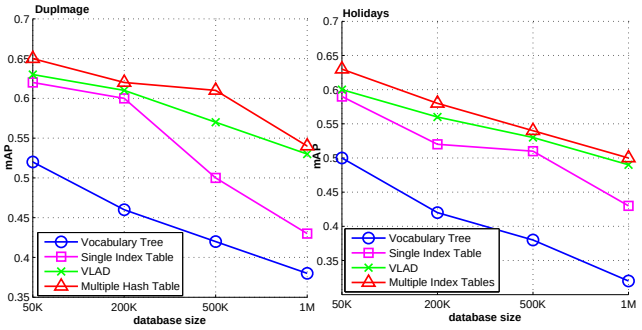

Figure 3: Image retrieval performance comparison of various methods with different database sizes.

Table 1: mAP performance with various number of index tables for one million database size.

Index Tables	1	2	4	6	8
DupImage	0.43	0.46	0.47	0.50	**0.54**
Holidays	0.43	0.44	0.44	0.46	**0.50**

In addition, we observe from the comparison that a single index table demonstrates less effectiveness than the proposed multiple indexing tables. this is mainly due to two reasons. First, a single index table is constructed by choosing one substring as the index. Such bit-selection leads to losing important image clues. Second, the Hamming distance between feature pairs is also severely affected as fewer bits instead of all feature are chosen. Moreover, Table 1 illustrates mAP comparisons when we increase the number of index tables. We can see that it achieves the best performance and yields an mAP of 0.54 when 8 index tables are fully used. It shows that search performance significantly improves when we fully utilize binary feature with multiple index tables.

3.2 Evaluation on Holidays Dataset

To further test the effectiveness of our algorithm, we validate it on the Holidays dataset. This dataset contains personal holiday photos provided by INRIA[7]. The dataset itself contains 1491 images. A subset of 500 images serves as queries. Each query is compared to the other 1490 images in a leave-one-out fashion. To evaluate the performance on a large scale, a distractor dataset of 1 million images downloaded from Flickr is also provided.

We use the same parameter setting as selected in the previous subsection. Figure 3 shows the experimental comparison results of our approach with baseline approaches. When the size of dataset is $50K$, our method achieves a mAP of 0.64 and obtains 28.0% and 8.3% relative improvement compared with VT and VLAD, respectively. It can also be observed that our approach shows consistent improvement to VLAD when the dataset size increases to $1M$. It is interesting to note that, as dataset size increases, the accuracy improvement of our approach over VLAD becomes smaller. This is partially due to the tradeoff between parameters r and m. Also, we employ a simple hash function instead of complex ones which might contribute to performance improvement. In Figure 4, we show some visualized results on a large-scale image dataset for various queries, such as

Table 2: Time cost comparison to index one million features for all approaches. ORB and FREAK refer to multiple indexing tables with the corresponding features, respectively.

Methods	VT	VLAD	ORB	FREAK
Time(second)	53.72	46.32	**4.23**	**7.23**
codebook size(bytes)	156M	324B	**NA**	**NA**

"StarBucks Logo", "Mona Lisa", etc. The left most images are the queries and returned images before the first false positive are shown in the list. The images that can not be retrieved by VLAD are highlighted by solid color boxes.

3.3 Efficiency Analysis

We also evaluate our approach from the aspect of efficiency. Table 2 shows the average time cost per query for all three retrieval approaches. Benefiting in part from binary features, it can be observed that our approach is more efficient in indexing one million features, which takes 4.23 and 7.23 seconds to index one million ORB and FREAK features. Our approach with ORB features is 13 and 10 times faster than VT and VLAD, respectively. It should be noted that another distinctiveness of our approach from others is that no visual codebook needs to be trained which could save much computation. In contrast, for the VT method, a large visual codebook containing millions of codewords has to be trained in a time-consuming way. Moreover, it is still unclear whether such a codebook could capture and model the sample distribution in high dimensional feature space (128-D)[13]. For instance, a hierarchical vocabulary tree requires $156M$ memory to be loaded into main memory. And the memory cost of VLAD approach for codebook is 324 bytes. In contrast, our approach based on ORB or FREAK features does not need memory to store a pre-trained codebook.

4. CONCLUSION

In this paper, we introduced a novel approach to building multiple index tables for large-scale image search. Without explicit codeword training, we explore the use of partial binary descriptors as direct codebook indices (addresses). Experimental results on two public datasets have demonstrated the effectiveness and efficiency of the proposed approach. Our approach can also be combined and enhanced by several post-possessing techniques[6] to further improve the image retrieval performance.

5. ACKNOWLEDGEMENT

This work was supported in part to Dr. Qi Tian by ARO grant W911NF-12-1-0057, Faculty Research Awards by NEC Laboratories of America, and 2012 UTSA START-R Research Award respectively. This work was supported in part by National Science Foundation of China (NSFC) 61128007.

6. REFERENCES

[1] D. G. Lowe. Distinctive image features from scale-invariant keypoints. In *IJCV*, 2007.

[2] P. Raghavan, C. D. Manning and H. Schtze. An introduction to information retrieval. Cambridge University Press, 2008.

[3] S. Zhang, M. Yang, X. Wang, Y. Lin and Q. Tian Semantic-aware co-indexing for near-duplicate image retrieval. In *ICCV*, 2013.

Figure 4: Retrieved image results based on multiple tables. The left most images are the queries. Returned images before the first false positive are shown. Images failed to be retrieved by VLAD are highlighted by color boxes.

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

[5] H. Jegou, F. Perronnin, M. Douze, J. Sanchez and C. Schmid. Aggregating local descriptors into compact codes. In *PAMI*, 2012.

[6] W. Zhou, Y. Lu, H. Li, Y. Song and Q. Tian. Spatial Coding for large scale partial-duplicate web image search. In *MM*, 2010.

[7] H. Jegou, M. Douze and C. Schmid. Improving bag-of-features for large scale image search. In *IJCV*, 2010.

[8] D. Niester and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

[9] Y. Xia, K. He, F. Wen, and J. Sun. Joint inverted index. In *ICCV*, 2013.

[10] A. Rejia and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[11] E. Rublee, V. Rabaud, K. Konolige and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *ICCV*, 2011.

[12] O. Chum and J. Matas. Fast computation of min-Hash signatures for image collections. In *CVPR*, 2012.

[13] W. Zhou, Y. Lu, H. Li and Q. Tian. Scalar quantization for large scale image search. In *MM*, 2012.

[14] A. Alahi, R. Ortiz and P. Vandergheynst. Freak:fast retina keypoint. In *CVPR*, 2012.

[15] S. StefanLeutenegger, M. Chli and R. Siegwart. Brisk: binary robust invariant scalable keypoints. In *ICCV*, 2011.

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

[17] J. He, J. Feng, X. Liu, T. Cheng, T.-H. Lin, H. Chung and S.-F. Chang. Mobile product serach with bag of hash bits and boundary reranking. In *CVPR*, 2008.

[18] M. Esmaeili, R. Ward and M. Fatourechi. A fast approximate nearest neighbor search algorithm in the Hamming space . In *PAMI*, 2012.

[19] M. Norouzi, A. Punjani and D. Fleet. Fast search in Hamming space with multi-index hashing. In *CVPR*, 2012.

[20] J. Cai, Z. Zha and Q. Tian. Attribute-assisted Reranking for Web Image Retrieval. In *MM*, 2012.

[21] J. Cai, Z. Zha and Q. Tian. Learning attribute-aware dictionary for image classification and search. In *ICMR*, 2013.

[22] X. Yang, Q. Liu, C. Liao and T. Cheng. Large-Scale EMM Identification Based on Geometry-Constrained Visual Word Correspondence Voting. In *ICMR*, 2011.