Vocabulary-based Hashing for Image Search

Yingyu Liang Jianmin Li Bo Zhang
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University
liangyy08@mails.tsinghua.edu.cn {lijianmin,dcszb}@mail.tsinghua.edu.cn

ABSTRACT

This paper proposes a hash function family based on feature vocabularies and investigates the application in building indexes for image search. Each hash function is associated with a set of feature points, i.e. a vocabulary, and maps an input point to the ID of the nearest one in the vocabulary. The function family can be employed to build a high-dimensional index for approximate nearest neighbor search. Then we concentrate on its application in image search. Guiding rules for the construction of the vocabularies are derived, which improve the effectiveness of the approach in this context by taking advantage of the data distribution. The rules are applied to design an algorithm for vocabulary construction in practice. Experiments show promising performance of the approach and the effectiveness of the guiding rules. Comparison with the popular Euclidean locality-sensitive hashing also shows the advantage of our approach in image search.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Storage and Retrieval; I.4.9 [**Computing Methodologies**]: Image Processing and Computer Vision

General Terms

Algorithms, Design

Keywords

visual vocabulary, hashing index, image search

1. INTRODUCTION

High-dimensional nearest neighbor search indexes are essential for various kinds of multimedia applications, where the multimedia objects are represented as sets of elements, and similarity between them is evaluated by searching near neighbors for each element. Such applications include content-based copy detection [6, 9] and song intersection [2].

MM'09, October 19–24, 2009, Beijing, China.

Similar image search is another typical such application. Similar images are defined as images of the same object or scene viewed under different imaging conditions [4]. Most previous works involve search for similar local invariant feature points such as SIFT[7]. Various typical approximate nearest neighbor search algorithms, for example [1, 3], show high performance in relatively small datasets, but do not fit in large scale scenes. The bag-of-features (BOF) method [10] is introduced in this context. A set of feature points, called visual words and usually generated by clustering the dataset, forms a visual vocabulary. Each feature point is quantized by mapping to the ID of the nearest word. The approach can be interpreted as an approximate nearest neighbor search: the space is partitioned into Voronoi cells, and points in the same cell are treated as neighbors of each other.

Some recent works extends BOF[4, 8]. For example, [4] improves BOF by adding to the points carefully designed binary signatures from Hamming embedding. Those points in the same cell and within a Hamming distance threshold are treated as neighbors. However, the time spent on computing Hamming distance of signitures grows linearly with the size of the dataset. On the other hand, hashing schemes [3, 5, 11] can fetch points in buckets directly as neighbors, for example, the popular Euclidean locality sensitive hashing based on p-stable distributions (E2LSH)[3]. The problem is E2LSH typically requires hundreds of bytes for each point, which prevents it from usage in large scale datasets.

Inspired by the vocabulary methodology and hashing index scheme and aiming to improve the defects, we propose a hash function family based on feature vocabularies and investigate the application in building indexes for image search. Each hash function is associated with a vocabulary, and maps an input point to the ID of the nearest one in the vocabulary. The function family can be employed to build a high-dimensional index for approximate nearest neighbor search. Guiding rules for the construction of the vocabularies are derived, which improve the effectiveness of the approach in the context of image search by taking advantage of the data distribution. The rules are applied to design an algorithm in practice. Although evaluated only in image search, the scheme is suitable for general near neighbor search in multimedia applications where the objects are represented as sets of elements.

This paper is organized as follows. The vocabulary-based hashing index is described in Section 2. Section 3 presents the experimental results and Section 4 provides some discussions. Section 5 concludes the paper.

2. VOCABULARY-BASED HASHING

In this section we first describe the hash function family based on visual vocabularies and its application in building an index. Guiding rules for constructing the vocabularies in the context of image search are then discussed, and are applied to design a practical algorithm.

2.1 Vocabulary-based Hash Functions

Denote a hash function family mapping a domain S into U as $\mathcal{H} = \{h : S \to U\}$. The hash function family can be interpreted as a randomized space partition approach, for each $h \in \mathcal{H}$ partitions the space into $C_h^i = \{p : h(p) = i\}(i \in U)$. On the other hand, the bag-of-features approach can be viewed as an approximate nearest neighbor search method, for the space is partitioned into Voronoi cells and points in the same cell are treated as neighbors of each other. So it is natural to use vocabularies to define hash functions. Formally, A hash function $h \in \mathcal{H}$ is defined as

$$h(q) = \underset{0 \le i < t}{\operatorname{argmin}} D(q, w_h^i), w_h^i \in V_h$$

where

$$V_h = \{w_h^i, 0 \le i < t\}$$

is a vocabulary associated with h, t is the size of the vocabulary and $D(q, w)$ is the distance between points q and w. Note that the functions are specific to the distance measure used. As the Euclidean distance is widely used in applications, we focus on this case of special interest.

Here we describe the hashing index scheme using a given function family [3]. First the discriminant power is amplified by concatenating several functions. In particular, given a parameter k, define a function family $\mathcal{G} = \{g : S \to U^k\}$ such that $g(p) = (h_1(p), \ldots, h_k(p))$, where $h_i \in \mathcal{H}$. Then for a given parameter L, choose L functions $g_1, \ldots, g_L$ from $\mathcal{G}$ independently and uniformly at random. During the construction of the index, each data point p is stored in the buckets $g_j(p)$, for $j = 1, \ldots, L$. To find neighbors for a query point q, search all buckets $g_1(q), \ldots, g_L(q)$ and return all the points encountered. Thus, the functions $g_1, \ldots, g_L$ define a hashing index and different hashing function family $\mathcal{H}$ leads to different index. For example, E2LSH employs locality-sensitive hashing functions based on 2-stable distributions[1]. Our vocabulary-based hashing index is constructed by employing the vocabulary-based hash functions. Obviously, the vocabulary construction plays a key role for the performance of the index, which is discussed in following subsections. For simplicity, we call $V_g = (V_{h_1}, \ldots, V_{h_k})$ a vocabulary associated with $g = (h_1, \ldots, h_k)$ and let $V = (V_{g_1}, \ldots, V_{g_L})$.

2.2 Guiding Rules for Vocabulary Construction

In the context of image search, the approximate nearest neighbors retrieved are used for computing image-level similarity, typically used to perform a voting on images in the dataset[10, 4]. In order to produce best search results, a high-quality search index should return ground truth points and filter noise points with high probability at the same time. Therefore, the quality of an approximate nearest neighbor search index can be measured in the following two criteria: 1) the average recall for the ground truth neighbors and 2) the average rate of points that are filtered in the dataset [4]. Aiming to improve these criteria for fixed k and L, we discuss some guiding rules which provide useful hints for constructing the vocabulary V.

If $g(q)$ returns too much points for a query q, the filtering rate of the search can not be improved in the following step. Therefore, a greedy strategy is adopted: we first seek to maximize filtering rate for each g and then maximize recall for the union of all the buckets $g_i (1 \le i \le L)$.

Denote the dataset as $\mathcal{P}$ and $N = \#\mathcal{P}, N(i) = \#\{p : g(p) = i, p \in \mathcal{P}\}$. Here we assume that the query shares the same distribution with the dataset, which leads to $Pr[g(q) = i] \approx \frac{N(i)}{N}$. The filtering rate of g for a query q is defined as $R_f = 1 - \frac{N(g(q))}{N}$. It is expected to be

$$\begin{aligned} E\big[R_f\big] &= 1 - \sum_{i \in U} Pr[g(q) = i] \frac{N(i)}{N} \\ &\approx 1 - \sum_{i \in U} \left(\frac{N(i)}{N}\right)^2. \end{aligned}$$

Now consider maximizing recall for the union of all the buckets $g_i (1 \le i \le L)$. The recall is relevant to the distribution of the ground truth neighbors, which is complicated. We simplify the analysis by turning to maximize the expected number of distinct points in the buckets. This makes sense since points in the buckets are near neighbors, i.e. candidate ground truth neighbors for the query. Let $N_i^s = \#\{p : g_s(p) = i, p \in \mathcal{P}\}$ and $N_{i,j}^{s,t} = \#\{p : g_s(p) = i, g_t(p) = j, p \in \mathcal{P}\}$. Let $L = 2$ for illustration. The number N_r of distinct points in the buckets is expected to be

$$\begin{aligned} E\big[N_r\big] &= \sum_{i \in U^k} \sum_{j \in U^k} Pr[h_1(q) = i, h_2(q) = j]\big(N_i^1 + N_j^2 - N_{i,j}^{1,2}\big) \\ &= \sum_{i \in U^k} Pr[h_1(q) = i] N_i^1 \\ &\quad + \sum_{j \in U^k} Pr[h_2(q) = j] N_j^2 \\ &\quad - \sum_{i \in U^k} \sum_{j \in U^k} Pr[h_1(q) = i, h_2(q) = j] N_{i,j}^{1,2} \\ &\approx \frac{1}{N}\left[\sum_{i \in U^k} \big(N_i^1\big)^2 + \sum_{j \in U^k} \big(N_j^2\big)^2 - \sum_{i \in U^k} \sum_{j \in U^k} \big(N_{i,j}^{1,2}\big)^2\right]. \end{aligned}$$

A similar analysis can be applied to $L > 2$, but the result is more awkward. By cutting the tail terms which are comparatively negligible, the expression is simplified to

$$\begin{aligned} E\big[N_r\big] &\approx \frac{1}{N} \sum_{s=1}^{L} \sum_{i \in U^k} \big(N_i^s\big)^2 \\ &\quad - \frac{1}{N} \sum_{s \ne t, 1 \le s,t \le L} \sum_{i,j \in U^k} \big(N_{i,j}^{s,t}\big)^2. \end{aligned}$$

2.3 Vocabulary Construction

Here we apply the guiding rules to design a preliminary algorithm, and then turn it into a more practical one. The algorithm is described as follows: $\mathcal{P}$ is the dataset, t, k and L are parameters of the index, C is a parameter indicating the number of repetitions.

[1]Instead of all points in the buckets, only those within a distance threshold are returned by E2LSH.

Subprocedure: ConstructVg$(\mathcal{P}, t, k, C)$

1 Compute the mean m of $\mathcal{P}$ and its bounding box, i.e. minimum and maximum value in each dimension

2 For $i = 1$ to C, $j = 1$ to k
Draw t random points $p^i_{j,s} (0 \leq s < t)$ within the bounding box of $\mathcal{P}$ uniformly and randomly

3 Centralize to $w^i_{j,s} = p^i_{j,s} - \frac{1}{t}\sum_{s=0}^{t-1} p^i_{j,s} + m$

4 Let $V^i_{h_j} = \{w^i_{j,s}\}, V^i_g = (V^i_{h_1}, \ldots, V^i_{h_k})$

5 Return V^i_g that maximizes $E\big[R_f\big]$

Procedure: ConstructVocabulary$(\mathcal{P}, t, k, L, C)$

1 For $i = 1$ to C, $j = 1$ to L
$V^i_{g_j} =$ **ConstructVg**$(\mathcal{P}, t, k, C)$

2 Let $V^i = (V^i_{g_1}, \ldots, V^i_{g_L})$

3 Return V^i that maximizes $E\big[N_r\big]$

When the vocabulary is large, constructing the vocabularies and computing the hash functions are time-consuming. We adopt a hierarchical approach: the dataset is partitioned into t_1 subsets and vocabulary construction is performed for each subset. More specifically, during the vocabulary construction step, the dataset $\mathcal{P}$ is first clustered into t_1 points, which form the first level vocabulary $\widehat{V} = \{\hat{w}^i, 0 \leq i < t_1\}$ for all $h \in \mathcal{H}$. Then we hash points on $\widehat{V}$ and each bucket forms a subset $\mathcal{P}_i$. The algorithm **ConstructVocabulary** uses each $\mathcal{P}_i$ as input dataset to construct vocabularies $V_i = (V_{i,g_1}, \ldots, V_{i,g_L}), V_{i,g} = (V_{i,h_1}, \ldots, V_{i,h_k}), V_{i,h} = \{w^s_{i,h}, 0 \leq s < t_2\}$, which form the second level. During the search step, we hash the query point on $\widehat{V}$, find which $\mathcal{P}_i$ it falls in, and use V_i to find its approximate nearest neighbors. Equivalently, each $h \in \mathcal{H}$ is associated with a two-level vocabulary tree, as show in Figure 1($t_1 = 3, t_2 = 2$ for illustration). Note the guiding rules are applied locally in $\mathcal{P}_i$. Still, experiments presented later show the effectiveness.

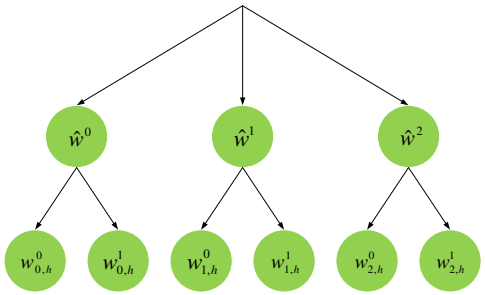

Figure 1: The vocabulary tree for $h \in \mathcal{H}$

3. EXPERIMENTS

Settings. We perform our experiments on the *Holidays* and *Flickr*60*k* datasets from [4]. The *Holidays* contains 4.456M points extracted from 1491 images, divided into 500 groups, each consisting of one query and its ground truth for evaluation. To show more accurately the behavior in large scale datasets the vocabularies are constructed on a distinct dataset *Flickr*60*k*. The index is built as follows: for each g_i, keep a table with $t_1 t_2^k$ buckets, and put points into corresponding buckets. Since in image search the neighbors retrieved are used for voting, we only keep the identity of the image that the feature point comes from.

Table 1: Performance

Method	Parameters	perf@100	perf@5
BOF [4]		0.673	
HE+WGC [4]		0.855	
VBH	$t_2 = 2, k = 4, L = 4$	0.804	0.634
VBH	$t_2 = 2, k = 4, L = 8$	0.814	0.633
VBH	$t_2 = 2, k = 8, L = 8$	0.843	0.719
VBH	$t_2 = 2, k = 8, L = 16$	0.860	0.731
VBH	$t_2 = 2, k = 8, L = 32$	0.863	0.739
VBH	$t_2 = 3, k = 6, L = 16$	0.858	0.712
VBH	$t_2 = 3, k = 6, L = 32$	0.869	0.712
VBH	$t_2 = 4, k = 4, L = 32$	0.856	0.694
no rules	$t_2 = 2, k = 8, L = 16$	0.798	0.585
no rules	$t_2 = 2, k = 8, L = 32$	0.799	0.580

Performance measure. To evaluate the performance of the index, we focus on its contribution to the image search result. Voting is performed with our implementation of the approach in [4]. There is usually a post-verification step of the top n positions in practice, using the geometry of the matched feature points, especially in large scale scenes where the voting results need further refinements. So rate of true positives returned in the top n positions after voting (perf@n) serves as a suitable performance measure [8, 4] and is adopted for our evaluation.

Results. The results for different settings of parameters are presented in Table 1. BOF (bag-of-feature) and HE+WGC (Hamming embedding + weak geometrical consistency) both use a vocabulary of 200000 words; HE+WCG adds 75 bit binary signature to each point. Their results are from [4]. In our VBH (vocabulary-based hashing) approach, the vocabularies are constructed with $C = 10$ and $t_1 = 20000$.

A set of small parameters such as $t_2 = 2, k = 4, L = 4$, results in a significant improvement over BOF. This indicates the effectiveness of our index scheme. A reasonable set of parameters such as $t_2 = 2, k = 8, L = 16$, produces better result than HE+WGC and removes the time growing linearly with the size of the dataset. And better performance is observed when larger k, L used. Larger t_2 produces similar results when the bucket number t_2^k is similar, so we focus on $t_2 = 2$. Comparison between perf@100 and perf@5 shows the ground truth images mainly rank in top positions.

Experiments are also performed to verify the effectiveness of the guiding rules, using random vocabularies with words drawn from the bounding box of the dataset uniformly and independently at random. The results are presented in the rows "no rules". The results produced by using the same parameters and trained vocabularies are significant better than those by using random vocabularies. This suggests the guiding rules and the subsequent algorithm indeed make contribution to the improvement of the performance.

Comparison with E2LSH. Figure 2 presents comparison with E2LSH. For E2LSH occupies a significant amount of memory, the experiments are performed on subsets of *Holidays*, each consisting 100 groups. And as *tf-idf* is not defined in E2LSH scheme, it is not used in the voting step

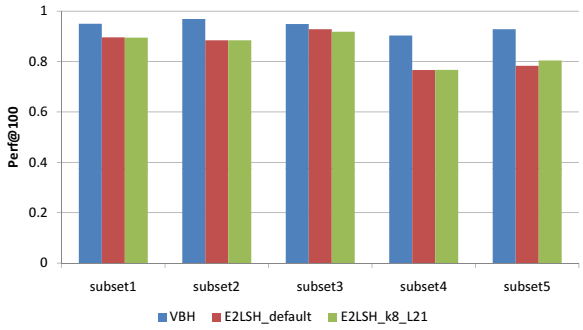

Figure 2: Comparison with E2LSH

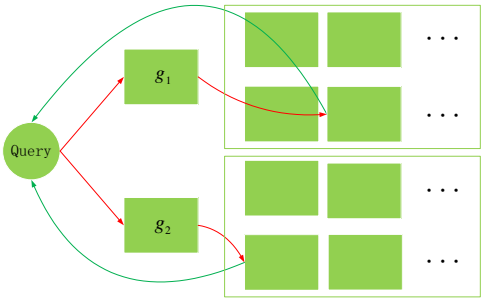

Figure 3: Parallelized vocabulary-based hashing

of the comparison, i.e. we vote without the *tf-idf* weighting term. For VBH, $t_2 = 2, k = 8, L = 16$. For E2LSH_default, $k = 14, L = 105$ is automatically computed by E2LSH; for E2LSH_k8_L21, we manually designate $k = 8, L = 21$ similar to VBH for comparison. For both E2LSH settings, we choose the neighbor distance threshold that produces best results and use default values for the other parameters. It is observed that VBH outperforms both E2LSH settings. We attribute this to the fact that VBH makes exploration of the data distribution and is specifically improved for image search by the guiding rules.

4. DISCUSSIONS

As described in the settings, the index keeps only identities of images, so each point requires $4L$ bytes, about 5 times compared to Hamming embedding (12 bytes), and significantly less than E2LSH (the point itself and $12L$ bytes). The search time consists of two parts: hashing on $\widehat{V}$ needs $O(t_1)$ if brute-force search is adopted; hashing on V_i needs $O(t_2 kL)$. So with fixed vocabulary, the time remains constant, which is a desired quality for large scale datasets. For a vocabulary of t words and b bit signature, the search time of Hamming embedding consists of three parts: hashing on the vocabulary needs the same $O(t)$ as our approach; embedding needs $O(b)$; computing Hamming distance needs approximately $O(n/t)$, growing linearly with the dataset size. For fixed table size t E2LSH also consumes linearly growing time $O(nL/t)$ since it computes distances between the query and points in the corresponding buckets.

Additionally, the vocabulary-based hashing can be parallelized naturally for the tables work in parallel. The parallelized version of the index is illustrated in Figure 3($L = 2$). The query is sent to each table and further forwarded to the corresponding bucket. Points in those buckets are then returned. The second part of the search time is reduced to $O(t_2 k)$. This should be a significant advantage over many other indexes, such as Hamming embedding. And although our index occupies more space as a whole, each table can hold more points than Hamming embedding, thus the index can deal with larger datasets after parallelization.

5. CONCLUSION

This paper has introduced a hash function family based on feature vocabularies and employed the functions to build an approximate nearest neighbor search index. Guiding rules for vocabulary construction have been derived to improve the effectiveness of the approach in the context of image search. Experiments show promising performance of the approach and the effectiveness of the guiding rules.

The approach shows desired qualities for large scale applications, so for future study, we plan to evaluate it on large scale datasets, especially the parallelized version.

6. ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under the grant No. 60621062 and 60605003, the National Key Foundation R&D Projects under the grant No. 2003CB317007, 2004CB318108 and 2007CB311003. The authors would like to thank Herve Jegou for providing the *Holidays* and *Flickr*60*k* datasets, and thank Alexandr Andoni for providing the E2LSH code.

7. REFERENCES

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 1998.

[2] M. Casey and M. Slaney. Song intersection by approximate nearest neighbour search. In *ISMIR*, 2006.

[3] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG*, 2004.

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

[5] A. Joly and O. Buisson. A posteriori multi-probe locality sensitive hashing. In *MM*, 2008.

[6] Y. Ke, R. Sukthankar, and L. Huston. Efficient near-duplicate detection and sub-image retrieval. In *MM*, 2004.

[7] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.

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

[9] S. Poullot, O. Buisson, and M. Crucianu. Z-grid-based probabilistic retrieval for scaling up content-based copy detection. In *CIVR*, 2007.

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

[11] H. Yang, Q. Wang, and Z. He. Randomized sub-vectors hashing for high-dimensional image feature matching. In *MM*, 2008.