Similar Image Search with a Tiny Bag-of-Delegates Representation*

Weiwen Tu
Dept. of Computer Science
Sun Yat-sen University,
Guangzhou, China
tuywen@gmail.com

Rong Pan
Dept. of Computer Science
Sun Yat-sen University,
Guangzhou, China
panr@sysu.edu.cn

Jingdong Wang
Microsoft Research Asia
Beijing, China
jingdw@microsoft.com

ABSTRACT

Similar image search over a large image database has been attracting a lot of attention recently. The widely-used solution is to use a set of codes, which we call bag-of-delegates, to represent each image, and to build inverted indices to organize the image database. The search can be conducted through the inverted indices, which is the same to the way of using texts to index images for search and has been shown to be efficient and effective.

In this paper, we propose a tiny bag-of-delegates representation that uses a small amount of delegates with a high search performance guaranteed. The main advantage is that less storage is required to save the inverted indices while having a high search accuracy. We propose an adaptive forward selection scheme to sequentially learn more and more inverted indices that are constructed based on subspace partition, e.g. using spatial partition trees. Experimental results demonstrate that our approach can require a smaller number of delegates while achieving the same accuracy and taking similar time.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing—*Indexing methods*

General Terms

Algorithms, Experimentation

Keywords

Bag-of-delegates, spatial partition trees, adaptive forward selection, similar image search

1. INTRODUCTION

Similar image search over a large image database has been a hot yet challenging problem in image retrieval. It plays a complementary role to text-based image search using associated texts for indexing. For instance, similar image search can help find images that look like an existing image at hand, but it is uneasy to use textual queries to describe the search intention and hence text-based image search cannot return satisfactory results.

There are two key issues in similar image search[1]. One is the visual similarity evaluation, which is related to feature selection. It is still unclear what kinds of features (e.g., texture, shape, or color) and what kind of similarity functions should be used. The other one is how to efficiently search in the large database, which is related to nearest neighbor search. Both are important to similar image search [5]. This paper focuses on the latter issue.

The straightforward solution to nearest neighbor search is to exhaustively compare the query with each image in the database to find exact nearest neighbors. This is apparently infeasible for a large database due to the high computational cost. Therefore, most research efforts are turned to approximate nearest neighbor (ANN) search. There are two main categories of ANN search methods. One is hashing. Representative methods include locality sensitive hashing [3], spectral hashing [16], and complementary hashing [17]. The other one is spatial partition trees, including kd-trees [1], PCA-trees [10], hierarchial k-means [8] and so on.

To search over the large database, the ANN search techniques, hashing or spatial partition trees, usually organize the images using inverted indices. Each inverted index consists of a set of inverted lists, each corresponding to a subset of points contained in a subspace, e.g., a bucket in a hash table or a leaf node in a tree, and representing the points in the bucket or the leaf node. Each inverted index can also be viewed as a vocabulary, with each word corresponding to a list. Using multiple inverted indices yields a bag-of-delegates representation, each delegate corresponding to a word in one vocabulary. To save an inverted index we need to save the indices of the lists and the indices of the images in each list. The main storage cost comes from the indices of all the images if the number of points in each list is not too small (e.g., not smaller than 100 for $1M$ images). Experiments show that spatial partition trees usually perform better in terms of accuracy and efficiency [7]. However, because of too many overlapped contributions to ANN search among spatial partition trees, this scheme requires too many

*This work was performed at Microsoft Research Asia.

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

[1]In this paper, we differentiate similar image search from particular object retrieval [9] that intends to search the images containing the same object to the one appearing in the query image.

trees to get a good performance, and consequently takes a large amount of storage and time costs.

In this paper, we propose a tiny bag-of-delegates descriptor to describe an image, which uses a small amount of spatial partition trees but achieves a good search performance. We present an adaptive forward selection approach, which sequentially chooses more and more spatial partition trees adaptively according to previously-identified trees. Experimental results demonstrate that the proposed approach is very efficient and effective and requires a smaller amount of delegates. Searching for top 100 similar images over $1M$ tiny images at the accuracy 0.75 (Figure 2(a)), our approach only requires about 50 inverted indices, $\frac{3}{5}$ of the number of indices required by traditional random PCA-tree-based method. A demo [14] based on the technology in [13] and this paper will be shown in ACMMM12.

2. TINY BAG-OF-DELEGATES

Given a large image set $\mathcal{I} = \{I_1, I_2, \cdots, I_N\}$, the goal is to find a bag-of-delegates descriptor to represent each image, e.g., I_i is represented by M delegates, $\{w_{i1}, \cdots, w_{iM}\}$. To this end, we need to build M vocabularies $\{\mathcal{V}_1, \cdots, \mathcal{V}_m, \cdots, \mathcal{V}_M\}$, where $\mathcal{V}_m = \{v_{m1}, \cdots, v_{mJ}\}$ with J being the size of each vocabulary. From each vocabulary an element is selected as a delegate for each image.

To conduct similar image search, we organize the image set using M inverted indices $\{T_1, \cdots, T_m, \cdots, T_M\}$, each corresponding to a vocabulary, respectively. An inverted index T_m is composed of J lists $\{T_{m1}, \cdots, T_{mJ}\}$, with each corresponding to a word, e.g., T_{mj} corresponds to v_{mj}. Each list consists of images that are mapped to the word v_{mj}.

Given an image I_i whose bag-of-delegates is defined as $\{w_{i1}, \cdots, w_{iM}\}$, we find the lists $\{F_1(w_{i1}), \cdots, F_M(w_{iM})\}$, where $F_m(w)$ is a function mapping from the delegate w to the list in the inverted index T_m, then regard all the images in these lists as the candidate similar images, and finally order these images according to their similarities computed with their features.

2.1 Background

The basic requirement to vocabulary construction is to map similar images to the same word. A single vocabulary often is not satisfactory. Thus, multiple vocabularies are exploited so that a pair of similar images are most likely to be mapped to the same word at least in one vocabulary.

One of the common solutions to multiple vocabulary construction is locality sensitive hashing (LSH) [3]. LSH builds multiple hash tables and maps each image into a bucket in each hash table. Here, a table is equivalent to a vocabulary and a bucket is equivalent to a word. The main characteristics of LSH include that (1) a hash table with a code length l uses only l hash functions, and (2) the tables are constructed independently on the data set. The two points lead to a worse performance than spatial partition tree-based methods [1, 10], and even learning-based hashing methods [16, 17] are still not good due to the first characteristic.

It has been shown that in practice spatial partition trees produce promising performance for approximate nearest neighbor search [7] and hence for similar image search. Similar to LSH, multiple random spatial partition trees are usually constructed. A larger amount of trees yield a better performance, while requiring more storage to save the inverted indices and accordingly more query time. We observed that on average there exist about 20% overlaps among a pair of spatial partition trees, while 12% in our approach. The overlap between two trees is computed based on the number of the true neighbor pairs appearing in both trees.

2.2 Vocabulary Construction

In this paper, we investigate the problem of using a small amount of trees but with the search performance guaranteed, which is less studied before to the best of our knowledge. Our idea is to construct the trees using the supervised information. In similar image search, the supervised information is the true nearest neighbors of each image. We denote the true neighbors of an image I_i by a list, $L_i = [n_{i_1}\ n_{i_2}\ \cdots\ n_{i_g}]$, where n_{i_j} is the index of a similar image. The nearest neighbor candidates discovered from the bag-of-delegates representation can be written as $C_i = F_1(w_{i1}) \cup \cdots \cup F_M(w_{iM})$. The criterion checking if C_i is satisfactory is to check if the recall $r_i = \frac{|C_i \cap L_i|}{|L_i|}$ is maximized. The whole criterion is to find trees to maximize the average recall $\bar{R} = \mathrm{E}(r_i) = \frac{1}{N}\sum_{i=1}^{N} r_i$.

The problem is formally described as follows. Given a set of Z candidate inverted indices $\mathcal{T} = \{T_1, \cdots, T_Z\}$, the goal is to find M ($M \leqslant Z$) inverted indices such that the recall is maximized. This is equivalent to selecting as few inverted indices as possible to guarantee that the recall is not lower than the target recall. For convenience, we will describe our solution in terms of the former one.

We denote a set of M candidates as $\mathcal{T}_M = \{T_{i_1}, \cdots, T_{i_M}\}$, and the recall from them as $\bar{R}(\mathcal{T}_M)$. The objective function is then written as

$$\mathcal{T}_M^* = \arg\max_{\mathcal{T}_M \subset \mathcal{T}} \bar{R}(\mathcal{T}_M). \quad (1)$$

This optimization problem is a combinatorial problem and NP-hard. Selecting the optimal inverted indices resembles the ensemble learning problem in pattern recognition that selects a subset of classifiers to yield the best classification performance. We propose to adopt the greedy search schemes that are studied in ensemble learning, because they seem to be particularly computationally advantageous and robust against overfitting. There are two basic schemes: forward selection and backward elimination. In forward selection, the inverted indices are progressively incorporated, yielding larger and larger subsets, whereas in backward elimination one starts with the set of all indices and progressively eliminates the least promising ones.

We use the forward selection scheme as it may loose the requirement, enumerating all the candidate indices at the beginning. We progressively generate the candidate inverted indices for each step. At the beginning, we randomly generate a set of candidate inverted indices $\mathcal{T}^{(1)} = \{T_1, \cdots, T_j, \cdots, T_{M_1}\}$. We evaluate each inverted index T_j to compute the recall, and then identify the first inverted index that corresponds to the largest recall. Denote the identified inverted index as T_1^* and the current solution is $\mathcal{T}_1^* = \{T_1^*\}$.

The later steps sequentially find the inverted indices one by one. Considering the $(t+1)$ step, we have found t indices $\mathcal{T}_t^* = \{T_1^*, \cdots, T_t^*\}$. We generate a set of new candidates $\bar{\mathcal{T}}^{(t+1)} = \{T_{M_t+1}, \cdots, T_{M_{t+1}}\}$ and form the whole candidates $\mathcal{T}^{(t+1)} = \mathcal{T}^t \cup \bar{\mathcal{T}}^{(t+1)}$. The objective of identifying the $(t+1)$-th index is as follows,

$$T_{(t+1)}^* = \arg\max_{T_{(t+1)} \in \mathcal{T}^{(t+1)}} \bar{R}(\mathcal{T}_t^* \cup \{T_{(t+1)}\}). \quad (2)$$

Algorithm 1 Adaptive forward selection

// L: the set of all pairs of neighboring points;
// Q: a subset of L, maintaining the pairs of images that do not appear in the same leaf node yet;
// e: the cardinality of Q; R: the obtained trees;

1. Initialization:
   $Q \leftarrow L$, $t \leftarrow 0$, $e \leftarrow |Q|$, $R \leftarrow \emptyset$;
2. **repeat**
3. Candidate proposal:
   Randomly generate spatial partition trees $\bar{\mathcal{T}}$ per Q;
4. Candidate selection:
   Choose the spatial partition tree T from $\bar{\mathcal{T}}$ that keeps the large number of pairs in Q;
5. Update:
   Discard all the pairs of points lying in the same bucket in T from Q;
   $t \leftarrow t + 1$;
   $e \leftarrow |Q|$;
   $R \leftarrow R \cup \{T\}$;
6. **until** $e \leqslant \epsilon$ && $t \geqslant \tau$;
7. **return** R;

To generate more helpful candidates $\bar{\mathcal{T}}^{(t+1)}$, we exploit previously-identified indices to adaptively generate the candidate indices. To this end, we maintain a set $\mathcal{Q} = \{(I_i, I_j)\}$, where I_i and I_j are truly neighboring points but not appear in the same bucket in the previously-identified indices $\mathcal{T}_t^*$. $\mathcal{Q}$ initially includes all the pairs of neighboring points. This set becomes smaller when more indices are discovered.

The following describes how to use $\mathcal{Q} = \{(I_i, I_j)\}$ to build a random spatial partition tree, thus yielding an inverted index. In the branch of each internal node v in the spatial partition tree, we randomly sample a set of points, compute top r sparse principal directions from them, and then form r candidate partition hyperplanes, each formed by a principal direction and the medium of the projections of the associated data points along this direction. We select the best partition hyperplane from the r ones so that the maximum number of pairs of points, which appear in $\mathcal{Q}$ and belong to the set of points associated with the internal node, lie in the same side. The algorithm is outlined in Algorithm 1.

Our algorithm is related to adaboost [4] for ensemble learning and complementary hashing [17]. It is different from them in that the problem we intend to solve is to combine tree-based indices. Moreover, our problem is more challenging because there are no closed-form solutions.

3. EXPERIMENTS

Data set. We conduct our experiments on the tiny images data set [11]. The tiny images data set consists of around $80M$ images, each being a 32×32 color image. We sample $1M$ tiny images to form the reference data set. We use a global GIST descriptor to represent each image, a 384D vector describing the texture within localized grid cells.

Evaluation criteria. We adopt the average accuracy score to evaluate the quality of search with inverted indices. Given an image I_i, we first collect the points that lie in the same bucket to I_i at least in one of the M inverted indices, and reorder them according to the distances computed between them and I_i to find the top K images R_i, where $K = |L_i|$ is the number of true nearest neighbors. The accuracy is computed as $a_i = \frac{|R_i \cap L_i|}{|L_i|}$. The average accuracy is computed as $\bar{a} = \frac{1}{N}\sum_{i=1}^{N} a_i$. We compare the performances in terms of two aspects: accuracy vs. # inverted indices and accuracy vs. # accessed point numbers. The former aims to show that our approach requires fewer number of inverted indices to reach the same accuracy, and the latter aims to show that our approach requires to access fewer number of images, equivalently less time cost, to get the same accuracy.

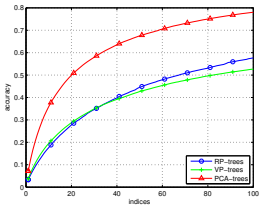

Figure 1: Performance comparison showing the superiority of PCA-trees over RP-trees and VP-trees.

We justify the proposed approach by showing the improvement for the indices constructed from random PCA-trees. It is theoretically shown in [12] that the principal-direction-based way to hierarchically partition the points can guarantee that the diameters of the subsets are reduced quickly. In our implementation, the principle direction is computed by the Lanczos algorithm [6]. Compared with other space partitioning ways, e.g., random projections trees (RP-trees) [2], vantage-point trees (VP-trees), our experiments [15] show that the principal-direction-based way is more efficient and effective, as illustrated in Figure 1.

Results. The comparison of the accuracy performance is shown in Figure 2. We show the performance over various numbers of indices. The horizontal axis corresponds to the number of the trees. The vertical axis corresponds to the accuracy score. To make the comparison more concrete, we show the performance for searching different numbers of true nearest neighbors. It can be seen that our approach is consistently superior for both different numbers of trees and various numbers of true nearest neighbors. As shown in Figure 2(a) with the buckets size (the number of points in a leaf node) being 500, searching for 100 NNs, our approach gets a 0.08 improvement when using 60 trees. Considering the case that we want to get a 0.7 accuracy, our approach only needs 40 trees, while random PCA-trees need 60 trees.

We also report the performance against different sizes of buckets. Using the same number of trees, the size of the bucket roughly determines how many candidates there are for an image. From Figures 2(b) and 2(c) with the bucket sizes being 300 and 100, it can be seen that our approach consistently performs better.

We also show the comparisons in Figure 3 according to accuracy vs. # accessed images, which can roughly reflect the time cost when achieving the same accuracy. We can see that our approach consistently performs better. From Figure 3(a), searching for 100 NNs, our approach gets about 0.04 improvement when searching $16,000$ images.

Out-of-sample search. To demonstrate the performance of query by an out-of-sample image, we sample an additional $10K$ images from the tiny images data set to form an out-of-sample data set, with the guarantee that there is no overlap with the $1M$ reference images. We build the ground truth nearest neighbors for the out-of-sample images by comparing them with the $1M$ images in a brute-force manner. The comparisons are shown in Figures 4 and 5. It can be seen that our approach consistently performs better.

4. CONCLUSIONS

In this paper, we study the problem of similar image search and propose a tiny bag-of-delegates representation approach. This approach uses a small number of inverted

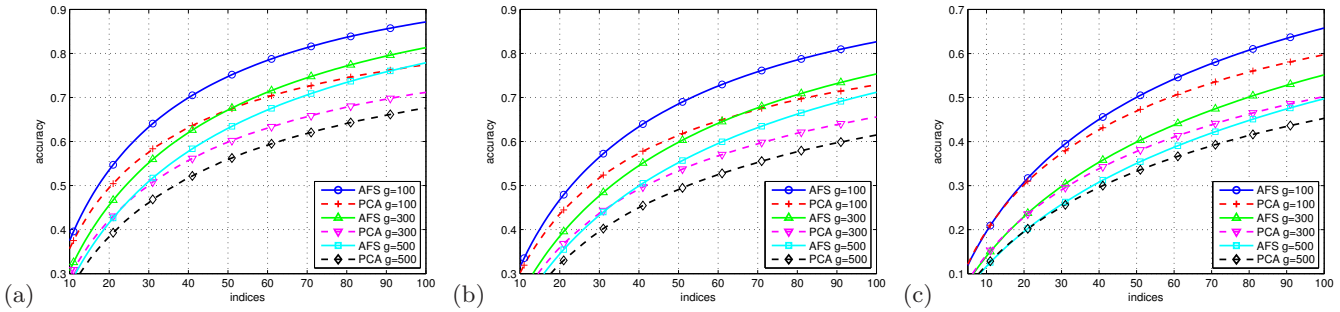

Figure 2: Accuracy vs. #indices over different bucket sizes: (a) 500, (b) 300 and (c) 100. g is the number of target NNs. AFS means our approach. PCA means random PCA-trees.

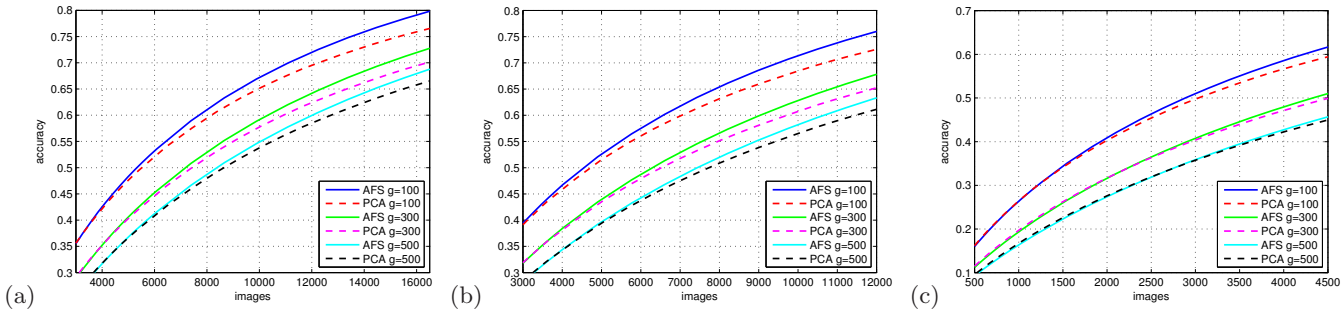

Figure 3: Accuracy vs. #(accessed images) over different bucket sizes: (a) 500, (b) 300 and (c) 100.

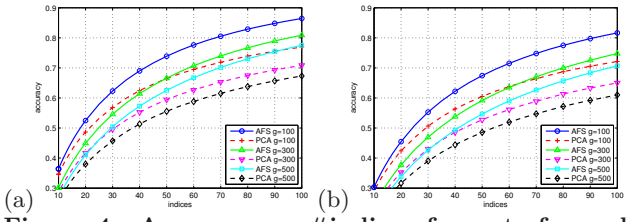

Figure 4: Accuracy vs #indices for out-of-sample queries over bucket sizes: (a) 500 and (b) 300.

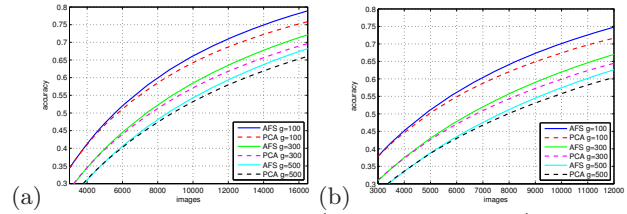

Figure 5: Accuracy vs #(accessed images) for out-of-sample queries over bucket sizes: (a) 500 and (b) 300.

indices leading to a small amount storage cost, but guarantees high search quality and efficiency. We present an adaptive forward selection scheme to effectively build the indices. Empirical results justify the powerfulness of our approach.

Acknowledgements

Weiwen Tu and Rong Pan were supported by National Natural Science Foundation of China (61003140, 61033010).

5. REFERENCES

[1] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975.

[2] S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In *STOC*, pages 537–546, 2008.

[3] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Symposium on Computational Geometry*, pages 253–262, 2004.

[4] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[5] B. Kulis, P. Jain, and K. Grauman. Fast similarity search for learned metrics. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(12):2143–2157, 2009.

[6] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4):255–282, 1950.

[7] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP (1)*, pages 331–340, 2009.

[8] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *CVPR (2)*, pages 2161–2168, 2006.

[9] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[10] R. F. Sproull. Refinements to nearest-neighbor searching in k-dimensional trees. *Algorithmica*, 6(4):579–589, 1991.

[11] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1958–1970, 2008.

[12] N. Verma, S. Kpotufe, and S. Dasgupta. Which spatial partition trees are adaptive to intrinsic dimension? In *UAI*, pages 565–574, 2009.

[13] J. Wang and S. Li. Query-driven iterated neighborhood graph search for large scale indexing. In *ACM Multimedia*, 2012.

[14] J. Wang, J. Wang, X.-S. Hua, and S. Li. Scalable similar image search by joint indices. In *ACM Multimedia*, 2012.

[15] J. Wang, J. Wang, G. Zeng, Z. Tu, R. Gan, and S. Li. Scalable k-nn graph construction for visual descriptors. In *CVPR*, pages 1106–1113, 2012.

[16] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.

[17] H. Xu, J. Wang, Z. Li, G. Zeng, S. Li, and N. Yu. Complementary hashing for approximate nearest neighbor search. In *ICCV*, pages 1631–1638, 2011.