FOREST HASHING: EXPEDITING LARGE SCALE IMAGE RETRIEVAL

Jonathan Springer[†], *Xin Xin*[†], *Zhu Li*[‡], *Jeremy Watt*[†], *Aggelos Katsaggelos*[†]

[†]Department of Electrical Engineering & Computer Science, Northwestern University, Evanston, IL
[‡]Multimedia Standards Research, Samsung Telecommunications America, Richardson, TX

ABSTRACT

This paper introduces a hybrid method for searching large image datasets for approximate nearest neighbor items, specifically SIFT descriptors. The basic idea behind our method is to create a serial system that first partitions approximate nearest neighbors using multiple kd-trees before calling upon locally designed spectral hashing tables for retrieval. This combination gives us the local approximate nearest neighbor accuracy of kd-trees with the computational efficiency of hashing techniques. Experimental results show that our approach efficiently and accurately outperforms previous methods designed to achieve similar goals.

Index Terms— image retrieval, kd-tree, spectral hashing, forest hashing.

1. INTRODUCTION

Efficiently finding similar images based on the content within an image has been of interest to multiple research communities for some time [4, 11]. Recently the Scale Invariant Feature Transform (SIFT) was demonstrated to effectively model image content for use as descriptors to help locate similarities between images [8]. However, SIFT descriptors leave users operating in a high-dimensional space leaving efficient search through large databases impractical. The Motion Pictures Experts Group (MPEG) is interested in minimizing the size and memory burden posed and are working towards a standard Compact Descriptors for Visual Search (CDVS) [9, 15, 17]; an end goal being an algorithm that can be executed on existing mobile devices, efficiently and accurately.

Several machine learning or retrieval techniques have been implemented and are thoroughly discussed in [4, 16, 18] to help circumvent this issue. It has been proven that approximate nearest neighbor (ANN) search works well for this task [7], and since this requires no offline training it is practical for many applications with continually expanding databases. However as the database size grows the search time increases significantly.

Bentley proved in [1] that binary tree structures known as kd-trees are particularly useful for efficiently finding nearest neighbors on a small scale. These trees are built by splitting datasets using a hyper plane into two equal parts. Depending on the user-defined height (*ht*) of the tree, this is iterated creating 2^{ht} different *leaves* at the bottom of the tree. Therefore if one were to choose a value for ht where $2^{ht} = N$, where N is the number of data points in the dataset, then every single data point would end up in its own leaf.

Another method, hashing - a general procedure for indexing and organizing a large quantity of data for convenient retrieval using functions - has become attractive to researchers as an alternative. There have been different approaches as to which hashing function best minimizes the distance between neighbor pairs [3, 12, 13, 14], but it seems spectral hashing performs well. Spectral Hashing takes a more holistic approach constructing hash functions based on both the structure of the data and ideal hash properties. For a formal review of spectral hashing, refer to Weiss et al [13]. Generally after matches are establish a geometric verification step is performed [4]. In this stage, location information from the query and from database features is used to confirm that the feature matches are consistent with a change in viewpoint between matched images. In this paper, we chose to focus on the improving the approximate nearest neighbor matching rather than enhancing the geometric verification step.

The rest of this paper is organized as follows. In Section II we describe related word. In Section III we describe in detail our method. In Section IV we discuss the experiments we conducted and evaluate our method and compare it against the current state of the art. Section V offers a conclusion and discusses future direction.

2. RELATED WORK

Annotations are introduced in part to facilitate later discussions. Dataset X consists of N data points $\{x_i\}$, where $x_i \in \mathbb{R}^d$. K hash functions maps a data point x to a K bit hash code $H(x) = [h_1(x), h_2(x), \dots, h_K(x)]$. In this section, we review two widely used hashing methods,

namely, Locality Sensitive Hashing (LSH) and Spectral Hashing(SH).

2.1. Locality Sensitive Hashing

Locality Sensitive Hashing constructs K hash tables using the hash projections, $H = [h_1, h_2, \dots, h_K]$. The k-th hash function is in the form of:

$$h_k(x) = \text{sgn}({w_k}^T x + b_k)$$

where w_k is a random hyper plane and b_k is a random threshold. With these hash functions, each data points are projected to K bit hash codes. To perform query search, a query point q is mapped to K hash codes and nearest neighbors of the hash codes are retrieved from hash table. In practice, because LSH suffers from severe redundancy of the hash bits, K has to be sufficiently large to achieve satisfactory precision. This leads to the big storage burden of holding the hash tables and high computational cost of projecting the query to the hash codes.

2.2 Spectral Hashing

Spectral hashing is brought in to learn hash functions from input data by minimizing the following function:

$$J(H) = \sum_{i,j=1}^{N} ||H(x_i) - H(x_j)||^2$$
$$s.t. \sum_{i=0}^{N} H(x_i) = 0$$
$$\frac{1}{N}\sum_{i=1}^{N} H(x_i)H(x_j)^T = \mathbf{I}$$
$$H(x_i) \in \{1, -1\}$$

The first two constraints give the following two good properties. 1) The hash codes are efficient: each hash bit partitions the data points into two balanced parts and 2) The hash codes are compact: the bits of a hash code are uncorrelated. Due to the constraints, SH is able to produce better hash codes than LSH in practice. Besides, this minimization problem can be converted to an eigenvalue decomposition problem and easily solved.

3. FOREST HASHING

In this section, we present our scalable image retrieval method, i.e. *Forest Hashing* (FH). Our method is so named due to its combination of hashing techniques and the use of multiple kd-trees. The basic idea behind FH is to combine the accuracy of local nearest neighbor retrieval of kd-trees with the efficiency of spectral hashing. A visualization of this is show in Figure 1.

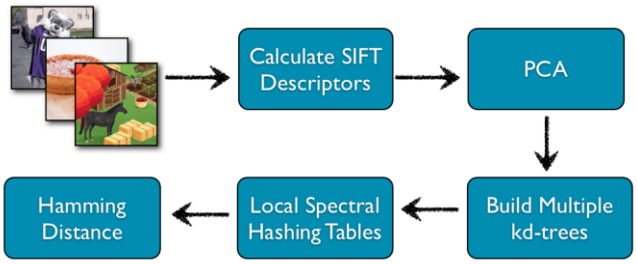

Figure 1 - Forest Hashing System

In our implementation of FH, we start with some large number N of SIFT descriptors, X = $\{x_i\}$, i = 1, 2, ..., N, $x_i \in R^D$, where generally D = 128 for the number of SIFT features per descriptor. Before system development is started, we find each descriptor's global nearest neighbors using a Euclidean distance measurement and save up different percentages of them to be used to gather optimal choices for precision and recall measurements found later.

3.1. Building multiple *kd*-trees

kd-trees were initially designed as space partitioning data structures useful for nearest neighbor searches that partition subspaces linearly [1], however, *kd*-trees vastly fail in high dimensional spaces. Our motivation for utilizing multiple trees was to incorporate information from different dimensions of the dataset in an efficient manner.

To make the tree structure more manageable we initially reduce the N descriptors using principal component analysis (PCA) on the dataset to lower its dimension considerably down to some small number L (e.g. L = 12) [2]. Next, we build multiple trees (up to four for our purposes) from organized subsets of the L-dimensioned data as follows:

$$Tree_{dim}(t) = L_{dim}\{ t : [L - (T - 1)] + t - 1\}$$

where t is the t^{th} set of dimensions in the build Trees and T is the total overall number of trees to build. This ensures that as long as $L > T$, that is the number of dimensions in the PCA reduced data is greater than the total number of trees being built, that individual trees will be constructed with at least 50% of the same dimensions, but never the exact same dimensions. We believe that building trees with different subsets of dimensions and utilizing these trees to find similar descriptors will give a higher number of accurate results in retrieval and will avoid a common pitfall of *kd*-trees - being they find local, not global approximate nearest neighbors.

Our experiments were all performed on *kd*-trees with a fixed height of ht = 8. It is a future intention to vary the height of trees to gather performance metrics.

3.2. Spectral Hashing

As mentioned earlier, hashing functions exceed at efficiently finding information in large sets, and lately are becoming more accurate as well [13]. Our motivation for incorporating a hashing method into our design, was to further improve the efficiency of the local approximate nearest neighbor search in each individual leaf while also creating a more effective method of parsing the global results for neighbors.

After the desired number, T, of *kd*-trees are built, local spectral hashing functions are built for each leaf; this step is the memory overhead for our method and simply consists of storing a matrix of size:

$$N/2^{ht} \text{ x } T \text{ - 16 bit hashing codes}$$

Which for N = 1,000,000, ht = 8, and T = 4 gives us roughly 30kB. As long as the quotient of the first term ($N/2^{ht}$) does not exceed 65536 one can use 16 bit hash codes. These local hashing functions ensure that the subspace within each leaf is properly partitioned to group true neighbors and non-neighbors.

3.3. Retrieving Similar Descriptors

After the multiple trees and hashing tables have been constructed, we parse these structures to find similar descriptors. Given an input query to the system, the T *kd*-trees are searched to find in which leaf this new query would be placed. Once these T leaves are found (one in each tree), a hashing function is used to construct T hash codes for the input query to compare against the other hash codes within each corresponding leaf. Since these are all multi-bit representations of the data, a simple hamming distance measure is used to compare to find most similar descriptors [10]. An illustration using 2 *kd*-trees is shown in the in Figure 3.

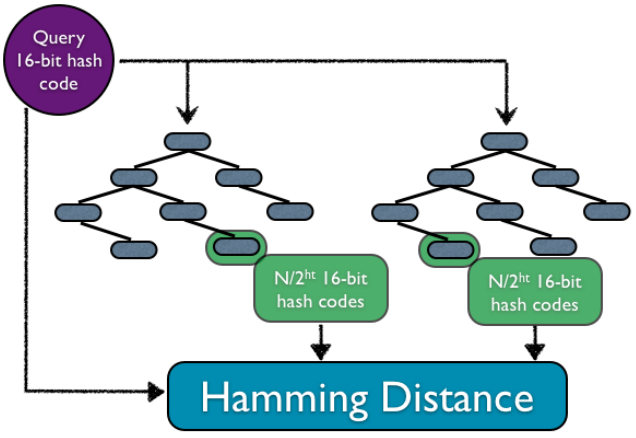

Figure 3 - Hamming Distance Calculation on Locally Hashed Leaves

What we have done is drastically reduced the calculation time of a distance comparison for two points. Assuming L = 12 dimensions after PCA, a comparison would have required comparing 12 different doubles (typically 64 bits each). Using spectral hashing allows us to make this comparison using 16 bits total per query; roughly 48 times quicker.

4. EXPERIMENTS

4.1. Datasets

In this paper, two datasets are used in the experiments. 1 million 128-dimensional SIFT descriptors are extracted from random images [12]. From the SIFT dataset, we select 10K SIFT points as test points in a fashion to make these pints uniformly distributed among *kd*-tree leaves. The ground truth neighbors of a query are obtained by the brute force search and a data point is regarded as a true neighbor if it lies within 0.03 percent points closest to the query. In this case, a query point usually has 300 neighbors.

4.2. ANN Recall Rate with KD-Tree

In order to test our method we ran multiple retrieval algorithms on the same dataset and gathered measurements. In all of our experiments we tested 10% of the entire dataset (100,000 SIFT descriptors) that were incorporated in the trained dataset of 1,000,000 descriptors.

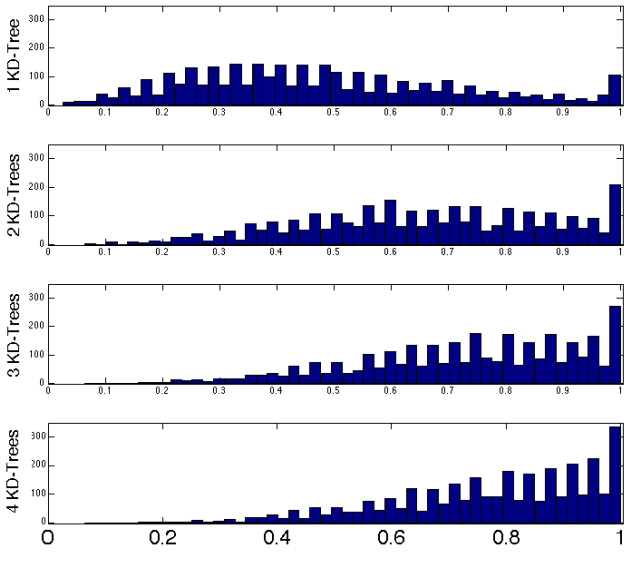

Figure 2 - Recall Rate using Multiple KD-Trees

First we tested the recall rate of a system designed to only use multiple *kd*-trees and not perform spectral hashing.

The graphic in Figure 2 shows histograms of recall rates when using 1 - 4 *kd*-trees; one can clearly see recall rates increasing as the number of trees is increased.

4.3. Comparison with Single Basis Hashing

When running our Forest Hashing algorithm, we chose only a few variables as parameters: T, the total number of trees to build, and R, the percentage of the ground truth calculated global nearest neighbors in which to compare our retrieved results. As can be seen from the below tables, we varied T from 1 - 4 and R from 0.1 to 0.8 find optimal recall rates given computation and time constraints.

5. CONCLUSION

In this paper we have presented a method known as Forest Hashing to improve accuracy and diminish computation time for image retrieval systems using a hybrid method involving multiple kd-trees and spectral hashing. Our system's performance demonstrates the advantages of using more than one kd-tree built on a variation of different dimensions from a SIFT descriptor. Our implementation of spectral hashing suggests performance improvements on the order of (64 x L/M), where L is the number of dimensions in your data and M is the number of bits assigned to each descriptor's hash code.

Experimental results showed a slight increase in recall rates using Forest Hashing as the number of trees increased. However, precision rates dropped slightly. One possibility is that adding trees is actually adding more false positives at corresponding hamming distances than true positives giving a drop in performance. Further investigation is required to verify this reason for this.

Our intention is to expand on these experiments to find a true optimal number of various parameters including: number of descriptor dimensions, number of kd-trees used, and number of bits assigned to hash codes. Furthermore, comparisons of various hashing techniques (e.g. Complementary Hashing) with Spectral Hashing may give an improvement in performance. Source code and demonstrations will be made available at [5].

6. REFERENCES

[1] L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, 18(9):509-517, 1975.

[2] Duda, Richard O., Peter E. Hart, and David G. Stork. "Pattern Classification and Scene Analysis 2nd ed." (1995).

[3] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," *Proceedings of the International Conference on Very Large Data Bases*, pages 518-529, 1999.

[4] B. Girod, V. Chandrasekhar, D.M. Chen, Ngai-Man Cheung, R. Grzeszczuk, Y. Reznik, G. Takacs, S.S. Tsai,R. Vedantham, "Mobile Visual Search," *Signal Processing Magazine, IEEE* , vol.28, no.4, pp.61-76, July 2011.

[5] *Image And Video Processing Laboratory Home Page.* (2007). Retrieved November 15, 2012 from <http:// http://ivpl.eecs.northwestern.edu/>

[6] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Machine Intell*., 2010.

[7] Y. Jia; J. Wang; G. Zeng; H. Zha; X. Hua; , "Optimizing kd-trees for scalable visual descriptor indexing," *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, vol., no., pp.3392-3399, 13-18 June 2010.

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[9] MPEG CDVS website <http://wg11.sc29.org/visual-search/>

[10] J. G. Proakis, Digital Communications, 5 ed. New York: McGrawHill, 2007.

[11] G. Schroth, R. Huitl, D. Chen, M. Abu-Alqumsan, A. Al-Nuaimi, E. Steinbach, "Mobile Visual Location Recognition," *Signal Processing Magazine, IEEE* , vol.28, no.4, pp.77-89, July 2011.

[12] J. Wang, S. Kumar, and S.F. Chang, "Semi-Supervised Hashing for Scalable Image Retrieval," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2010.

[13] Y. Weiss, A. Torralba, and R. Fergus, "Spectral Hashing," *Proc.Advances in Neural Information Processing Systems* 2008.

[14] Xu, H., Wang, J., Li, Z., Zeng, G., Li, S., Yu, N.: "Complementary hashing for approximate nearest neighbor search," *International Conference on Computer Vision*. (2011) 1631

[15] Xin Xin, Zhu Li, Aggelos K. Katsaggelos. "Laplacian embedding and key points topology verification for large scale mobile visual identification". Signal Processing: Image Communication, Available online 4 February 2013, ISSN 0923-5965, 10.1016/j.image.2012.11.003.

[16] Tzu-Jui Liu, Hye Jung Han, Xin Xin, Zhu Li, Aggelos K Katsaggelos "A ROBUST AND LIGHTWEIGHT FEATURE SYSTEM FOR VIDEO FINGERPRINTING". Eusipco 2012

[17] Xin Xin, Aggelos K Katsaggelos. "A novel image retrieval framework exploring inter cluster distance". ICIP 2010.

[18] Xin Xin, Zhu Li, Aggelos K Katsaggelos. "LAPLACIAN SIFT IN VISUAL SEARCH "
http://www.mirlab.org/conference_papers/International_Conference/ICASSP%202012/pdfs/0000957.pdf