ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

A Comprehensive Survey on Image Search Using Binary Hash Codes

Shroff Rahul D[1], Dhage Jaykumar S[2]

M. Tech Scholar, Department of Computer Science & Engineering, Maharashtra Institute of Technology (MIT), Aurangabad, India [1]

Assistant Professor, Department of Computer Science & Engineering, Maharashtra Institute of Technology (MIT), Aurangabad, India [2]

Abstract: Effective content based Image Search based on hash codes is a very acceptable for efficient similarity search, due to its query time and storage efficiency. In binary hashing technique, first the high-dimensional visual features are extracted from images, then these extracted features are embedded into Hamming space and the distance or similarity of two points are approximately calculated by the Hamming distance between their binary codes. Though the Hamming distance calculation is efficient in practice, a query result is ambiguous as it returns multiple results sharing the same Hamming distance and poses a critical issue for similarity search where ranking is important. This paper presents a survey of various states-of-the-art- hashing techniques that allows faster visual similarity search in hamming space. And also discussed concept of weighted Hamming distance, which improve ranking performance of binary hash code so that result images can be ranked at fined grained level.

Keywords: Feature Extraction, Approximate Nearest Neighbour Search, Binary Hash Codes, Weighted Hamming Distance.

I. INTRODUCTION

Rapidly increasing smartphones users (ultimately the internet users & usage) have resulted in explosive growth of digital images over web. As a result, content based search engine are in demand to find exact similar image from huge image database. Last was the decade of many famous text based search engines which are now migrating to Content Based Image Retrieval (CBIR) System. Unlike text based search engines, CBIR systems take an image as search input and try to return its similar images from a given database using pre specified feature extraction and distance measure techniques. This paper concentrates on a technique which is able to retrieve the most similar images from a large and possibly distributed database of images.

Any search method is expected to be memory efficient, allowing to store millions of images, with an ability to perform a fast similarity search. The most successful methods can be mainly divided into two different categories:

1) Tree based indexing methods
2) Hashing methods.

Tree based indexing methods store the reference samples in a tree structure. It provides an approximate nearest neighbour search in logarithmic time with respect to the number of samples. A critical drawback of this is that the images are represented using a very high dimensional vector. This results in increased need of backtracking to explore all the nodes. This is when the Hashing techniques come in picture with a solution to the approximate nearest neighbour search in high dimensional spaces. Hashing methods map the high-dimensional representation into a binary representation with a fixed number of bits. Moreover, computing the hamming distance for binary codes is very fast, as it can be performed efficiently by using bit XOR operation and counting the number of set bits.

Though hashing proved to be effective for visual similar search in several existing works, they lack in providing a good ranking which is vital for image search. There is possibility of having multiple hash codes sharing a same distance to a query in a high dimensional Hamming space resulting in hundreds or even thousands of images sharing the same ranking in search result list, but very unlikely to be equally related to the queried image. In this paper, we discuss various hashing methods used for similarity search and also discuss the concept of Weighted Hamming Distance Ranking.

The rest of this paper is organized as follows. System framework of Image Search using hash codes discus in Section II. The Hashing technique used for similarity image search discussed in Section III and Give an overview Weighted Hamming Distance Ranking in Section VI. Finally, Section V concludes this paper.

II. SYSTEM FRAMEWORK

The flow diagram of image search using hash codes is shown in Fig. 1. It requires the user to enter image as query with minimum effort and accordingly return its similar images from a given database using Hamming distance ranking. The system works as following:

1) First, visual features are extracted from individual images from database. The extracted features are described by feature vectors. These extracted feature vectors are embedded into hash codes and stored in feature database.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

2) For finding similar images from database user has to enter image as query. For a given query image, similarly extract its features and form a hash code. This hash code of query image is compares with already store hash codes of all images in feature database. This process usually called similarity matching and is based on calculation of hamming distance between the query image hash code and hash codes from database. Finally, results images are retrieved based on shorter hamming distance between images.

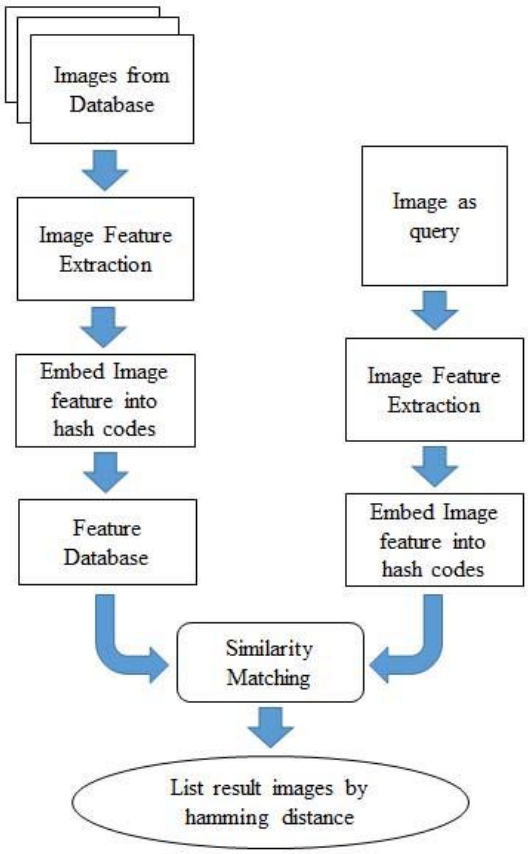

Fig. 1. Image Search Using Binary Hash Codes

III. HASHING TECHNIQUES

Generally, hashing methods are classified into two categories:

1) Supervised methods
2) Unsupervised methods.

Our initial discussion starts with unsupervised binary hashing methods that design hash function using unlabeled data to generate binary codes. Locality Sensitive Hashing (LSH) is the most popular and effective technique among unsupervised methods and uses random projections of the data. We further continue discussing the kernelled version of LSH which works on kernel distance. Our discussion after this also includes another method called Spectral Hashing (SH) that generates hash based on data distribution and ensures that projection are orthogonal and sample number is balanced across different buckets.

Finally we discuss supervised methods that use labelled data. We focus on few approaches like Binary reconstructive embedding (BRE) that minimizes the reconstruction error between the original metric and hamming space, Restricted Boltzmann Machines (RBMs) and Sematic Hashing method which uses deep belief network to learn hash codes. Our discussion then concentrates on Semi Supervised hashing (SSH) method that can leverage the sematic similarity using label data while remaining robust to over fitting.

The following common notion is used in this paper: There are n images given for training. Each query or training item is represented by a feature vector x of dimension d and the data matrix is represented as $X \in \mathbb{R}^{d \times n}$, where each column is a data point. Each binary code of the input is indicated by $b = [b_1, b_2, ...b_k]$ with k bits and a single bit p in the b is denoted by b_p where $b\{-1,1\}^k . B\{-1,1\}^{k \times n}$ represents the binary code matrix where each column is a binary code b of the data point x. $H = [h_1; h_2; ...; h_k]$ is the set of sequential hash functions where each hash function $h_p(x)$ gives a single bit value. Notice that the data and binary matrices are column matrix.

A. Locality Sensitive Hashing

In 1999, Indyk and Motwani introduced the term called Locality sensitive hashing [1], [2] is one of the most widely accepted methods. This method try to hash the input items so that similar items are assigned to the same buckets with high probability. More precisely, the hash functions used for finding approximate nearest neighbours that shows the property $Pr[h(p) = h(q)] = sim(p,q)$ where the term $sim(p,q) \in [0,1]$ is similarity function of interest. Definition: A family H is called (R, cR, P_1, P_2) sensitive if for any $p, q \in \mathbb{R}^d$.

1) If $\| p - q \| \leq R$ then $\Pr_H[h(p) = h(q)] \geq P_1$
2) If $\| p - q \| \geq cR$ then $\Pr_H[h(p) = h(q)] \leq P_2$

LSH is useful if it satisfy $P_1 > P_2$ and any hash function of such family holds the similarity property. A typical category of hash function $h_c(x) \in H$ for the inner product similarity $sim(x_i, x_j) = x_i^T x_j$ based on rounding the output of a product with a random hyper plane:

$$h_c(x) = \begin{cases} 1 & w_c^T x > 0 \\ 0 & otherwise \end{cases}$$

Where $w_c = [w_{c1}, w_{c2}, ...w_{cd}]^T$ is a random hyper plane from a zero-mean multi-variant Gaussian $N(0,1)$ of dimension d. Efficient codes in practice for large collections could be drawn independently and uniformly from a normal distribution by the weights shown in both [1], [2]. The main advantage of this method is that random projections is used to maintain the input distances with in the specified range as the number of hash bits increases; Simultaneously, it has been recognized that the large number of hash bits may require to maintain the distances for some cases. Practically, this technique is applicable to

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

many problem domain and works well for large number of bits.

B. Kernelized Locality Sensitive Hashing

In order to satisfy the locality sensitive hashing property several hash functions are designed for instances where similarity referred by using an l_p norm [1], Mahalanobis metric[3] or inner product [4]. In [11], Kulis et al proposed a similar technique that designs hash function for similarity function is an arbitrary kernel function $\kappa : sim(x_i, x_j) = \kappa(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ for some unknown embedding function. Input data is only accessible through the embedding function it is requires that $w^T \phi(x)$ to be calculated using the kernel function to preserve the w is from $N(0,1)$. Only one way to achieve this by constructing w as a weighted sum of a subset of database items where w is roughly Gaussian. The following algorithm shows computing w, where kernel matrix K of input items defined using any kernel κ:

1) First, choose p points and create K as a kernel matrix using this database items.

2) Create the hash table using database items: to form e_S for each hash function $h(\phi(x))$ by selecting indices at random from $[1...p]$ then form $w = K^{-1/2e_S}$, and allocate bits using $h(\phi(x)) = sign(\sum_i (w(i)\kappa(x_i, x_j))$

3) These hash functions are used to create a hash key for each query and apply existing LSH techniques to find ANN.

At the cost of computation, this techniques shows superior performance than traditional LSH. It required $O(p^3)$ time for computing K^{-1} most expensive step. It need $O(p^2)$ time to calculate w and finally each bit of codes is calculated in $O(p)$ time. The algorithm would sub linear search times by selecting $p = O(\sqrt{(n)})$. This algorithms is not suitable for large datasets because it based on a random sample of the data and required more bit to achieve considerable performance.

C. Spectral Hashing

To overcome the disadvantages of hashing techniques that based on random projections, machine learning techniques is used to enhance the quality of hashing methods. Especially, Spectral Hashing was recently introduced to create compact binary hash codes for approximate nearest neighbors search. In addition, the desired property of placing neighbours in input space like neighbours in the Hamming space the basic Spectral Hashing formulation needs the codes to be balanced and uncorrelated. The hash functions $H(x) = \{h_k(x)\}, k = 1,...,K$ must satisfy the following criteria [5]:

$$\min \sum_{ij} sim(x_i, x_j) \| H(x_i) - H(x_j) \|^2$$

$$\text{Subject to: } h_k(x_i) \in \{-1,1\}$$

$$\sum_i h_k(x) = 0, k = 1,...K$$

$$\sum_i h_k(x_i) h_l(x_i) = 0, \text{ for } k \neq l$$

Due to the balanced graph partition problem is NP hard, there is non trivial solutions to above optimization even for single bits. The constraints of pairwise independence makes combination of *K*-bit balanced partitioning harder. After relaxing the constraints, analysis of spectral graph is used to solve the above optimization [6]. Particularly, with the supposition of uniform data distribution, 1D Laplacian eigen functions is used for efficiently calculating the spectral solution [5].

The final algorithm of spectral hashing has three basic steps:

1) First, The directions of maximum variance are extracted using principal component analysis over the data;

2) The direction selection is used to make favour for partition projections using large spread and small spatial frequency;

3) The Projected data is partition by using a sinusoidal function with lastly calculated angular frequency.

Because of the vital principal component analysis directions are selected many times to form binary bits spectral hashing has been proved to be effective to encode low-dimensional data. Nevertheless, for high dimensional problems $(D >> K)$ where a large number of directions contain enough variance, generally each principal component analysis direction is chose only once.

A low spatial frequency is used due to the top few projections share similar range. Here, spectral hashing nearly replicates a principal component analysis projection followed by a mean partition. In spectral hashing, the directions of projection are depend on data but learning is done in an unsupervised manner. Besides, the supposition of uniform data distribution is generally not accurate for real-world data.

D. Binary Reconstructive Embedding

Kulis and Darrell [10] proposed a technique called Binary Reconstruction Embedding that design the hash functions by explicitly minimizing the reconstruction error between the actual distances and the Hamming distances of the corresponding binary encoding. In brief, it minimize the $L2$ loss between distance and hamming space. It employs the kernelized hash function similar to a kernelized version of LSH given as $h_p(x) = sign(\sum_{i=1}^{q} W_{pq} k(x, x_q))$ where q is a subset of points randomly selected so that q can vary for each hash function. The objective function is as follows:

$$\min \sum (i, j) \in N (d(x_i, x_j) - \tilde{d}(x_i, x_j))^2$$

Where $d(x_i, x_j) = 1/2 \| x_i - x_j \|^2$ is the Euclidean distance and $d(x_i, x_j) = 1/k \| \tilde{x_i} - \tilde{x_j} \|^2$ is the hamming distance and N is the number of pairs. $\tilde{x_i} = [h_1(x_i), h_2(x_i)....h_k(x_i)]$ gives the binary code $\tilde{x} = B$.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

In order to solve the problem it uses the coordinate descent algorithm which require $O(nk(q+\log n))$ time to update all the hash functions where q is number of pairs per point. Although the solution does not provide guarantees to be globally optimal as the index for each update is selected randomly and it takes a large time for computations.

E. Restricted Boltzmann Machines

In order to obtain binary codes for similarity image search, deep belief networks is learn [13] using stacking Restricted Boltzmann Machines (RBMs) [12]. Deep network is used to acquire high order correlation between different layers of network. By selecting structure that successively decrease the number of units in each layer, the high-dimensional input vector can be assigned to a smaller compact binary output vector.

Practically, in RBMs network is trained using two basic stages: first, the unsupervised pre-training phase is successively executed from input layer to output layer in greedy way. After attaining convergence of the parameters of a layer through contrastive divergence, the activation probabilities of a layer are fixed and used as input to the above layer for driving network. Second, in the supervised fine-tuning stage, the trained network is refined by using the labeled data via back-propagation. First, a cost function defined to calculate the number of points that correctly classified from the training data set [14].Then, this objective function is maximize by refining the network weights via gradient descent. Let's For example, in [15] the RBMs structure consist of five layers of size 512-512-256-32 nodes needs total of 663552 weights for learning. This need very costly training procedure, and enough training data for fine tuning.

F. Semantic Hashing

In order to overcome drawback of previous hashing technique Salakhutdinov and Hinton introduced a nearest-neighbours technique for binary vectors termed as Semantic Hashing whose speed is independent of the number of data points [16]. Each binary vector corresponds to an address in memory. To find matches to a query vector by taking the query vector and systematically disturbing bits within it, so exploring a Hamming ball around the original vector. Any neighbours in the database that fall within this ball will be declared as neighbours. Semantic Hashing possess two main advantages: (i) if the radius of the Hamming ball is small, it is extremely quick and (ii) as compared to kd-tree type data structures, constructing the database in this is very fast.

But it comes has a major drawback that, it breaks down for long code vectors, since the mean Hamming distance between points becomes large and the volume of the Hamming ball becomes prohibitive to explore. Let's say for example, we have a code length of 100 bits. The mean distance to a query's closest neighbour may well be quite large, e.g., differing in 7 bits or more. However, if one can afford only a Hamming ball radius search of 3, a query will not find any neighbours within that restricted search volume. Another drawback with Semantic Hashing is that it requires a contiguous block of memory, which becomes impractical for vectors length beyond 32.

G. Semi-Supervised Hashing

The term Semi-Supervised Hashing is proposed by Wang et al [17] that minimizes empirical error on the labeled data and at the same time it maximizing variance and independence of hash bits over the both labeled and unlabeled data. Mathematically, choose a subset l of points from a set of n points for which supervised information is calculated. A pair $(x_i, x_j) \in M$ is denoted as a neighbor-pair where x_i and x_j are neighbors in metric space or share common class-labels. Likewise, $(x_i, x_j) \in C$ is denoted as non-neighbor pair where x_i and x_j are far away in metric space or have different class labels. Let us denote data matrix by $X_l \in \mathbb{R}^{d\times l}$ formed by X set of l samples. The p^{th} hash function is written as $h_p(x_i) = sign(w_p^T x_i)$. The objective function is defined in such way that $W = [w1;.....;wk] \in \mathbb{R}^{d\times k}$ learn to gives the same bits for $(x_i, x_j) \in M$ and different bits for $(x_i, x_j) \in C$

$$J(\mathrm{H}) = \sum_p \left\{ \sum_{(x_i,x_j)\in M} h_p(x_i)h_p(x_j) - \sum_{(x_i,x_j)\in C} h_p(x_i)h_p(x_j) \right\}$$

Using the similarity matrix S notation, the objective function is represented as

$$J(W) = \frac{1}{2} tr\{sign(W^T X_l) S sign(W^T X_l)^T\}$$

The above optimization tries to over fit the data for $l << n$ and hence a new objective function is designed so as to minimize the empirical error of the supervised data and maximize the independence of individual bits and balanced properties. According to the lemma a hash function with a maximum variance on data X must satisfy the balancing constraint and vice-versa. The following objective function is obtained:

$$J(W) = \frac{1}{2} tr[W^M W]$$

Where $M = X_l S X_l^T + \eta X X^T$, relaxing the decorrelation of bits to orthogonally constraints on the projection directions, combined with unit-norm $\| w\| = 1$ consideration leads to constraints $W^T W = I$. In brief, the first part in M tries to approximate the empirical error using supervised information and the second part provides regularization that gives those directions that amplify the projections variance subject to the orthogonally constraints. The work shows that there is no need to have orthogonally constraints and in order to obtain a better solution for the real data weights are computed by relaxing these orthogonally constraints.Weights of the non-orthogonal constraints are given as $W_{nonorth} = LU_k$ where L is obtained from the cholesky decomposition $Q = LL^T$ and Q is positive definite and given as $Q = (1+1/\rho M)$.

ISSN (Online) : 2278-1021
ISSN (Print) : 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 3, Issue 11, November 2014

As compared to BRE this technique gives us optimization solutions and is of the technique that learn compact binary codes by combining supervised and unsupervised information.

IV. WEIGHTED HAMMING DISTANCE RANKING

All these hashing techniques (either unsupervised or supervised) that we have reviewed in previous section brought one limitation when applied to image search. The traditional hamming distance of hash codes cannot provide fine grained ranking on search results, which is very vital in practice.

In practice, to implement the concept of weighted hamming distance for visual similarity search, we have to assign specific weights to each bit of binary hash codes. Let's say for example, we have three images P, Q and R with their binary hash codes 1010, 1111 and 0000 respectively. The P, Q and P, R are having equal Hamming distance, without considering the reality that R varies from P in the first and third bits while Q vary in the second and last bits. Because of this behaviour of the Hamming distance, practically there can be multiple images having the equal hamming distance to a query image which makes critical issue of finding similar images. To solve this by assigning different weights to each bit of binary hash codes, so that each bit of binary hash has a different meaning. Returning to the example, for this consider the first and third bits are important for P, then Q must be ranked first than R if P is the query image.

V. CONCLUSION

In this paper, we have represented our semantic literature review on various recent states-of-the-art hashing technique used for visual similar image search. These techniques were presented in a way highlighting their advantages and limitations in terms of efficiency and time-taken. We also discussed concept of weighted hamming distance.

The review of this paper will support our future research on improving image search quality using weighted hamming distance, which provide fine grained ranking on return images.

REFERENCES

[1] Moses S. Charikar, *"Similarity estimation techniques from rounding algorithms,"* in Proceedings of the thirty-fourth annual ACM symposium on Theory of computing, STOC '02, pages 380–388, New York, NY, USA, 2002. ACM.

[2] Piotr Indyk and Rajeev Motwani, *"Approximate nearest neighbors: towards removing the curse of dimensionality,"* In Proceedings of the thirtieth annual ACM symposium on Theory of computing, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

[3] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S.Mirrokni, *"Locality-sensitive hashing scheme based on p-stable distributions,"* in Proceedings of the twentieth annual symposium on Computational geometry, SCG '04, pages 253–262, New York, NY, USA, 2004. ACM.

[4] Prateek Jain, Brian Kulis, and Kristen Grauman, *"Fast image search for learned metrics,"* in CVPR, 2008.

[5] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. of Advances in Neural Information Processing Systems*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., 2008, vol. 21, pp. 1753–1760.

[6] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, *"Spectral grouping using the Nystrom method," IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 2, pp. 214–225, 2004.

[7] Mikhail Belkin and Partha Niyogi, *"Towards a theoretical foundation for laplacian based manifold methods,"* in Int. J. Comput. Syst. Sci., 74(8):1289–1308, 2008.

[8] Yoshua Bengio, Olivier Delalleau, Nicolas Le Roux, Jean-Franois Paiement, Pascal Vincent, and Marie Ouimet, *"Learning eigen functions links spectral embedding and kernel PCA,"* in Neural Computations, 16:2004, 2004.

[9] R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, F. Warner,and S. Zucker. *"Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps,"* in Proceedings of the National Academy of Sciences, pages 7426–7431, 2005.

[10] Brian Kulis and Trevor Darrell, *"Learning to hash with binary reconstructive embeddings,"* in Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, Advances in Neural Information Processing Systems 22, pages 1042– 1050. 2009.

[11] Brian Kulis and Kristen Grauman, *"Kernelized locality-sensitive hashing,"* in IEEE Trans. Pattern Anal. Mach. Intell., 34(6):1092–1104, 2012.

[12] Geoffrey Hinton and Ruslan Salakhutdinov, "*Reducing the dimensionality of data with neural networks,*" Science, 313(5786):504 – 507, 2006.

[13] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye The, *"A fast learning algorithm for deep belief nets,"* Neural Comput., 18(7):1527–1554, July 2006.

[14] Jacob Goldberger, Sam Roweis, Geoff Hinton, and Ruslan Salakhutdinov, "Neighbourhood components analysis," in Advances in Neural Information Processing Systems 17, pages 513–520. MIT Press, 2004.

[15] Antonio Torralba, Robert Fergus, and Yair Weiss, *"Small codes and large image databases for recognition"*, In CVPR, 2008.

[16] Ruslan Salakhutdinov and Geoffrey Hinton, *"Semantic hashing,"* Int. J. Approx. Reasoning, 50(7):969–978, July 2009.

[17] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang ,*"Semi-supervised hashing for scalable image retrieval,"* in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, USA, June 2010.

BIOGRAPHY

Shroff Rahul D, Pursuing M. Tech from M.I.T, Aurangabad. He has accomplished B.E (Computer Science and Engineering) from P.E.S College of Engineering, Aurangabad. His current research interest include Digital Image processing and retrieval, Machine Learning and Pattern recognition.