ISSN:2320-0790

An International Journal of Advanced Computer Technology

Image Indexing and Retrieval

Ms. Snehal S. Bhamre [1], Prof. N. M. Shahane [2]

[1]Department of Computer Engineering, KKWIEER, Nashik, University of Pune, Maharashtra, India.
[2] Associate Professor, Dept. of Computer Engineering, KKWIEER, Nashik, University of Pune, Maharashtra, India.

Abstract: Scalable content based image search based on hash codes is hot topic nowadays. The existing hashing methods have a drawback of providing a fixed set of semantic preserving hash functions to the labelled data for the images. However, it may ignore the user's search intention conveyed through the query image. Again these hashing methods embed high-dimensional image features into hamming space performing real time search based on hamming distance. This paper introduces an approach that generates the most appropriate binary codes for different queries. This is done by firstly offline generating bitwise weights of the hash codes for a set of predefined semantic classes. At query time, query adaptive weights are computed online by finding out the proximity between a query and the semantic concept classes. Then these images can be ranked by weighted Hamming distance at a finer-grained hash code level rather than the original Hamming distance level.

Keywords: hash codes; high-dimensional image features; scalability; bitwise weights; weighted Hamming distance.

I. INTRODUCTION

Since there are lots of images on the Internet, there is strong need to develop techniques for effective and efficient image search. While traditional image search mechanisms highly rely on textual words associated to the images, scalable content based image search is becoming popular.

Generally a large-scale image search system consists of two key components—an effective image feature representation and an efficient search mechanism. The quality of search results relies heavily on the representation power of image features. An efficient search mechanism is critical since existing image features are mostly of high dimensions and current image databases are huge, on top of which exhaustively comparing a query with every database sample is computationally prohibitive.

In this work, images are represented using the popular bag-of-visual-words (BoW) framework, where local invariant image descriptors are extracted and quantized based on a set of visual words. The BoW features are then embedded into compact hash codes for efficient search. For this, a hashing technique including semi-supervised hashing and semantic hashing with deep belief networks is considered. Hashing is preferable over tree-based indexing structures as it generally requires greatly reduced memory and also works better for high-dimensional samples. With the hash codes, image similarities can be efficiently measured.

II. LITERATURE SURVEY

There are many surveys on general image retrieval task. Many people adopted simple features such as color and texture in systems developed in the early years, while more effective features such as GIST [1] and SIFT [4] have been popular recently.

Lowe introduced the Scale-Invariant Feature Transform (SIFT) descriptor [Lowe 1999] in 1999. The basic idea is to extract interesting features from an image and be able to compare them to template features, regardless of a change in scale or orientation.

Inverted index was initially proposed and is still very popular for document retrieval in the informational retrieval community [3]. A key difference of document retrieval from visual search, however, is that the textual queries usually contain very few words. While in the BoW representation, a single image may contain hundreds of visual words, resulting in a large number of candidate images (from the inverted lists) that need further verification. This largely limits the application of inverted files for large scale image search. While increasing visual vocabulary size in BoW can reduce the number of candidates, it will also significantly increase memory usage [6]. For example, indexing 1 million BoW features of 10 000 dimensions will need 1 GB memory with a compressed version of the inverted file. In contrast, for the binary representation in hashing methods, the memory consumption is much lower (e.g., 48 MB for 1 million 48-bit hash codes).

Indexing with tree-like structure has been frequently applied to fast visual search. Nister and Stewenius [5] used a visual vocabulary tree to achieve real-time object retrieval in 40 000 images. Muja and Lowe [8] adopted multiple randomized d-trees [7] for SIFT feature matching in image applications. One drawback of the classical tree-based methods is that they normally do not work well with

high-dimensional feature. In view of the limitations of both inverted file and tree-based indexing, embedding high-dimensional image features into hash codes has become very popular recently. Hashing satisfies both query time and memory requirements as the binary hash codes are compact in memory and efficient in search via hash table lookup or bitwise operations.

Locality Sensitive Hashing (LSH) [10] is one of the most well-known unsupervised hashing methods. Recently, Kulis and Grauman [2] extended LSH to work in arbitrary kernel space, and Chum et al. [9] proposed min-Hashing to extend LSH for sets of features. In [3], Kulis and Darrell proposed a supervised hashing method to learn hash functions by minimizing reconstruction error between original feature distances and Hamming distances of hash codes. In [12], Salakhutdinov and Hinton proposed a method called semantic hashing, which uses deep belief networks [5] to learn hash codes.

All these hashing methods (either unsupervised or supervised) have one limitation when applied to image search. The Hamming distance of hash codes cannot offer fine-grained ranking of search results, which is very important in practice.

III. Details Of Dissertation Work

A. Mathematical Model

The proposed system accesses a query in image form and provides the search results in terms of similar images from stored database.

The proposed system S is defined as,

S = {I, SF, H, C, W, O, F}

Where,

I = {I1, I2, I3…... IN} set of N input images.

SF = {sf1, sf2, sf3…... sfN} set of SIFT features vector.

SFi = {sf1, sf2, sf3…...} set of features of single image.

H = {H1, H2, H3…... HN} set of hash code.

C = {C1, C2, C3 ……..} set of BoW classes.

W = {W1, W2, W3 ……. WN} set of hamming weights.

O= {O1, O2, O3…... Ok} set of top k relevant images searched.

F = {f1, f2, f3, f4, f5} set of functions.

The system design includes main functions which are given below:

1. Function f1 takes images as input and generates 128 bit vector of SIFT features.
   f1 (Ii) → SFi
2. Function f2 takes SIFT features as input and generates hash code.
   f2 (SFi) →Hi (1 ≤ i ≤ N)
3. Function f3 classify images into BoW class and assign class tag to the images.
   f3 (Hi) → Cj (1 ≤ j ≤ no of classes of images)
4. Function f4 takes hash code as input and assign hamming weights to the images.
   f4 (Hi) → Wi
5. Function f5 compares hash code and hamming weight of query image with the data stored in database and generates top k relevant images searched.
   f5 (Hq, Wq) ⟶ Ok

B. Data Dependence and Data Flow architecture

1. Functional dependency graph:

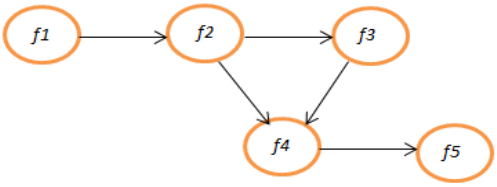

Fig 1: functional dependency graph

2. Level 0 Data Flow Diagram:

Level 0 DFD for hash code based image search is as shown in the figure given bellow. The images in the database are given as the input to the system. And this system is responsible to generate output search result.

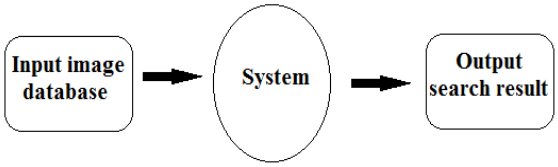

Fig 2: level 0 data flow diagram

3. Level 1 Data Flow Diagram:

Level 1 data flow diagram gives a detailed view of the flow of data in the proposed system, in which all the function, database needed for the system are shown. Feature extraction, hashing and finding Hamming weights are the main functions of the proposed system.

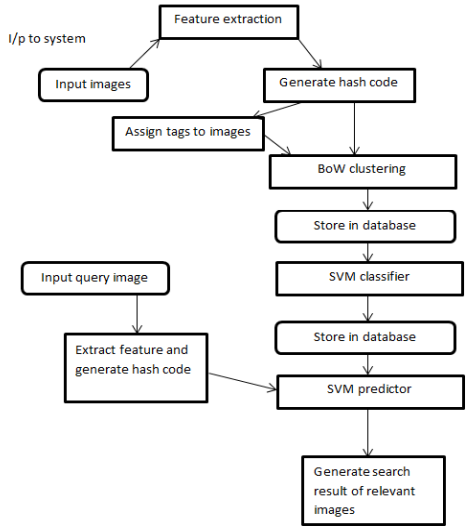

Fig 3: level 1 data flow diagram

C. Process Block Diagram

The System architecture is as shown below:

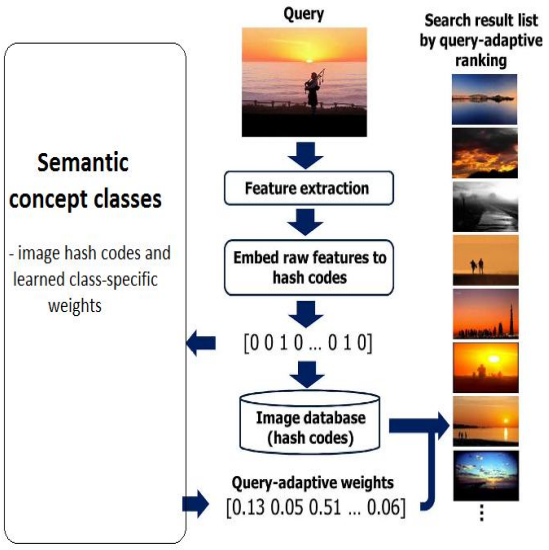

Fig 4: System Architecture

The system works in two parts: 1) Offline processing and 2) Online processing.

In offline processing, we have a database of images. First step will be of feature extraction of images by using SIFT/SURF/ORB algorithm. After that these features are embedded into hash codes. Images are then assigned tags as per the features and are classified into different classes using clustering. These classes together form Bag-Of-Visual Words (BoW). All this data is stored in the database.

The flowchart of offline processing is as shown in figure.

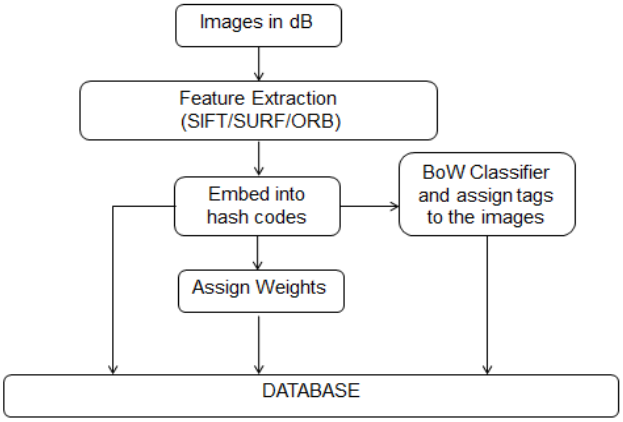

Fig.3. Offline processing of classifying and assigning hash codes to the images in the database.

In online processing, when query image is fired, feature extraction of that image is carried out. Then these features are embedded into hash code. These hash code along with the assigned weight are compared with the data stored in the database and list of relevant images is produced. These images are ranked based on the hamming distance. And thus we get an efficient search result.

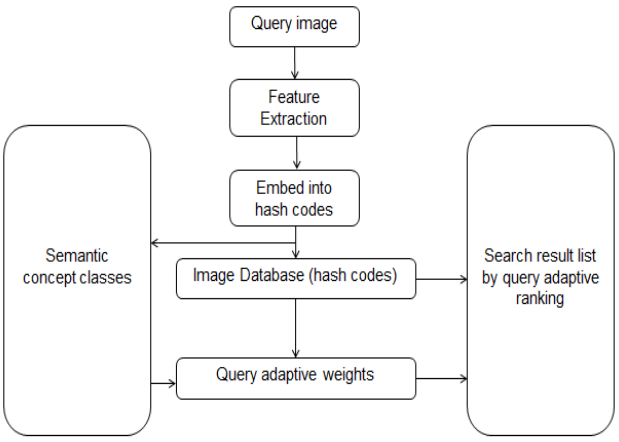

Fig.4. Online processing of searching results for the fired query image in the database.

First we harness a set of semantic concept classes, each with a set of representative images. Low-level features (bag-of-visual-words) of all the images are embedded into hash codes. We first compute hash code of the query image, which is used to search against the images in the predefined semantic classes. From there we pool a large set of images which are close to the query, and use them to predict final search result.

IV. Result And Discussion

A. Dataset

For the experimental purpose, we use a subset of the MIRFLICKR collection. The entire dataset contains 1 million images from the social photo sharing website Flickr. Of the entire collection, 25 thousand images were manually annotated.

B. Result Set

The first step is to extract the features of the images in the database. For this purpose we SIFT (scale invariant feature transform) algorithm. The SIFT algorithm works into four basic steps:

1. Scale-space extrema detection: this step will search overall scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function.

2. Key point localization: At each location, a detailed model is fit to determine location and scale and accordingly Key points are selected.

3. Orientation assignment: One or more orientations are assigned to each key point location based on local image gradient directions.

4. Key point descriptor: The local image gradients are measured and are transformed into a

representation that allows for significant levels of local shape distortion and change in illumination.

Following figure shows a sample image and its extracted features after applying SIFT algorithm.

Fig.5. Sample image

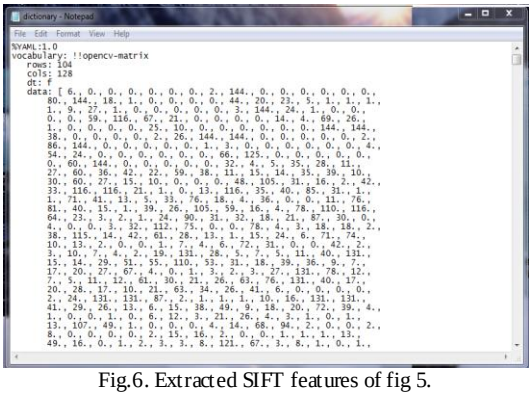

Fig.6. Extracted SIFT features of fig 5.

V. CONCLUSION

A novel framework for query-adaptive image search with hash codes is presented. By harnessing a large set of predefined semantic concept classes, the approach is able to predict query-adaptive bitwise weights of hash codes in real-time, with which search results can be rapidly ranked by at finer-grained hash code level. This capability largely alleviates the effect of a coarse ranking problem that is common in hashing-based image search.

One can further extend framework for query-adaptive hash code selection. Instead of image specific codes, the class specific codes can further improve search performance significantly. One drawback is that nontrivial extra memory is required by the use of additional class-specific codes, and therefore a careful examination of the actual application is needed and hardware environment in order to decide whether this extension could be adopted.

REFERENCES

[1] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," Int. J. Comput. Vision, vol. 42, pp. 145–175, 2001.

[2] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in Proc. IEEE Int. Conf. Computer Vision, 2009.

[3] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in Adv. Neural Inf. Process. Syst., 2009.

[4] D. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vision, vol. 60, no. 2, pp. 91–110, 2004.

[5] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2006.

[6] H. Jegou, M. Douze, and C. Schmid, "Packing bag-of-features," in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2009.

[7] J. L. Bentley, "Multidimensional binary search trees used for associative searching," Commun. ACM, vol. 18, no. 9, pp. 509–517, 1975.

[8] J. Zobel and A. Moffat, "Inverted files for text search engines," ACM Comput. Surveys, vol. 38, no. 2, 2006.

[9] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in Proc. Int. Conf. Computer Vision Theory and Applications, 2009, pp. 331–340.

[10] O. Chum,M. Perdoch, and J. Matas, "Geometric min-hashing: Finding a (thick) needle in a haystack," in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2009.

[11] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in Proc. Symp. Theory of Computing, 1998.

[12] R. Salakhutdinov and G. Hinton, "Semantic hashing," in Proc. Workshop of ACM SIGIR Conf. Research and Development in Information Retrieval, 2007.