Multi-IDF Scheme for large scale image retrieval

G.KAVITHA[1], *PG Scholar, RVS College of Engineering and Technology, Coimbatore, Tamilnadu, India*
gkavitha1991@gmail.com
E.HARI PRASATH[2], *Assistant Professor, RVS College of Engineering and Technology, Coimbatore, Tamilnadu, India,*
erhariprasath@gmail.com
P.ARUL PRAKASH[3], *Assistant Professor, RVS College of Engineering and Technology, Coimbatore, Tamilnadu, India, arulprakash247@gmail.com*

Abstract: **Visual matching could be a crucial step in image retrieval supported the bag-of-words (Bow) model. Within the baseline technique, 2 key points are thought of as an identical try if their SIFT descriptors are amount to identical visual word. However, the SIFT visual word has 2 limitations. First, it loses most of its discriminative power throughout division. Second, SIFT solely describes the native texture feature. Each drawbacks impair the discriminative power of the Bow model and result in false positive matches. To tackle this downside, multiple binary options are embedded at assortment level. To model correlation between options, a multi-IDF scheme [8] is introduced, through that completely different binary options are coupled into the inverted file. Matching verification methods [5] are supported binary options. The joint integration of the SIFT visual word and binary options greatly enhances the preciseness of visual matching, reducing the impact of false positive matches. The planned technique considerably improves the baseline approach. Additionally, massive scale experiments indicate that the planned technique needs acceptable memory usage and question time compared with alternative approaches**.

Index terms: **SIFT, CN Descriptor, Hamming Embedded, Multi-IDF, image retrieval.**

1. INTRODUCTION

This paper focuses on the task of enormous scale partial duplicate image retrieval. Given a question image, our target is to seek out pictures containing constant options in an exceedingly massive info in real time. a picture retrieval system could be a computing system for browsing, looking and retrieving pictures from an outsized info of digital pictures. Most ancient and customary ways of image retrieval utilize some technique of adding data such as captioning, keywords, or descriptions to the pictures so retrieval will be performed over the annotation words.

One of the foremost well-liked approaches to perform such a task is that the Bag-of-Words (BoW) model. The introduction of the SIFT descriptor has enabled correct partial-duplicate image retrieval supported feature matching. Specifically, the BoW model 1st constructs a codebook via unattended bunch algorithms. Then, a picture is depicted as a bar graph of visual words, created by feature division. Every bin of the bar graph is weighted with tf-idf score or its variants. With the inverted file arrangement, pictures area unit indexed for economical retrieval.

Essentially, one key issue of the BoW model involves visual word matching between pictures. Accurate feature matching ends up in high image retrieval performance. However, 2 drawbacks compromise this procedure. First, in quantization, a 128-D double SIFT feature is quantal to a single integer [2]. The discriminative power of SIFT feature is basically lost. Options that lie off from one another may very well comprise an equivalent cell, so manufacturing false positive matches. Second, the progressive systems accept the SIFT descriptor, that solely describes the native gradient distribution, with rare description of alternative characteristics, like color, of this native region [4]. As a result, regions that area unit similar in texture area however totally different in color area might also be thought-about as a real match. Each drawbacks cause false positive matches and impair the image retrieval accuracy. Therefore, it's undesirable to require visual word index because the solely price tag to visual matching. Typically, the binary options area unit extracted alongside SIFTS, and embedded into the inverted file [6]. The explanation why binary feature will be used for matching verification is two-fold. First, compared with floating-point vectors of an equivalent length, binary options consume a lot of less memory. As an example, for a 128-D vector, it takes 512 bytes and sixteen bytes for the floating-point and binary options, severally. Second, throughout matching verification, the playing distance between 2 binary options will be expeditiously calculated via xor operations, whereas the Euclidian distance between floating-point vectors is extremely dear to figure. Previous work of this line includes playing Embedding (HE) [3] and its variants that use binary SIFT options for verification. Meanwhile, binary options additionally embody abstraction context, heterogeneous feature.

In lightweight of the effectiveness of binary options, this paper proposes to refine visual matching via the embedding of multiple binary options. On one hand, binary options give complementary clues to build the discriminative power of SIFT visual word. On the opposite hand, during this feature fusion method, binary options are coupled by links derived from a virtual multi-index structure. During this structure, SIFT visual word and different binary options are combined at categorization level by taking every feature joined dimension of the virtual multi-index. Therefore, the image retrieval method votes for candidate pictures not solely similar in native texture feature, however conjointly consistent in different feature areas. With the thought of multi-index, a unique IDF theme, referred to as multi-IDF, is introduced. We have a tendency to show that binary feature verification strategies like performing Embedding [5], will be effectively incorporated in our framework. Moreover, we have a tendency to extend the planned framework by embedding binary color feature.

This paper argues that feature fusion by coupled binary feature embedding considerably enhances the Discriminative power of SIFT visual word [5]. First, SIFT binary feature retains a lot of info from the first feature, providing effective check for visual word matching. Second, color binary feature offers complementary clues to SIFT feature. Each aspects serve to enhance feature matching accuracy. Intensive experiments on totally different class image retrieval datasets make sure that the planned methodology dramatically improves image retrieval accuracy.

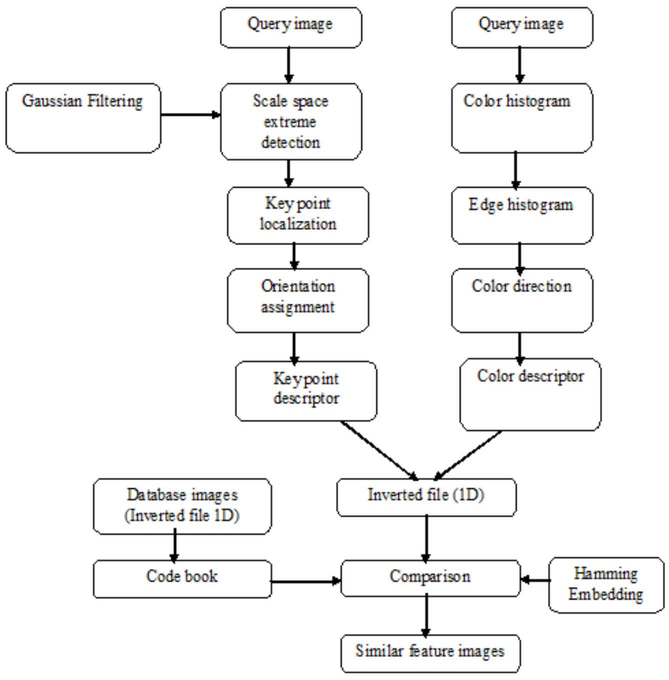


Fig.1 System Design

3. SYSTEM DESIGN

The following above Fig 1 illustrate the design of the system for retrieving similar features images.

The system has two modules

(1) SIFT (Scale Invariant Feature Transform)
(2) CN Descriptor

A. Module 1

This module is used to extracting the features from an image. SIFT extracts image features that are stable over image translation, rotation and scaling and somewhat invariant to changes in the illumination and camera viewpoint. The SIFT algorithm has four major phases,

1) Extrema Detection,
2) Keypoint Localization,
3) Orientation Assignment,
4) Keypoint Descriptor Generation.

The first section, Extrema Detection, examines the image beneath numerous scales and octaves to isolate points of the image that are totally different from their surroundings. These points, known as extrema, are potential candidates for image options.

The next section, Key point Detection, starts with the extrema and selects a number of these points to be key points that are a whittled down a group of feature candidates. This refinement rejects extrema that are caused by edges of the image and by low distinction points.

The third section, Orientation Assignment, converts every key point and its neighborhood into a group of vectors by computing a magnitude and a direction for them.

The last section, Key point Descriptor Generation, takes a set of vectors within the neighborhood of every key point and consolidates this info into a group of eight vectors known as the descriptor. every descriptor is regenerate into a feature by computing a normalized total of those vectors.

1) Scale-Space Extrema Detection

This is the primary section of the SIFT algorithm [8]. Here the formula identifies the points that square measure stable with relevance image rotation, translation and people that square measure minimally stricken by noise and little distortions. Detective work these points may be accomplished by checking out stable options across all attainable scales. The formula computes "scale," "difference of Gaussian's, and "extrema" over many "octaves."

Scale:

Let I be an N × N image and for 0 ≤ x, y < N, let $G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$ be the discrete two-dimensional Gaussian function. Then the scale of the image I is defined as

$L(\sigma) = \{G(x,y,\sigma) * I(x,y) : 0 \leq x,y < N\}$

where * is the two-dimensional convolution operation and I(x, y) is the pixel at row x and column y of image I(x, y). In general, the k^{th} scale of the image, for k≥ 1 is defined as $L(k\sigma) = \{G(x,y,k\sigma) * I(x,y) : 0 \leq x,y < N\}$. For each image point I(x, y), the scale is computed by applying a scalar product between the point I(x, y) and a w×w Gaussian weighted window placed over that point.

In general for a w × w window with odd w, the image points located around the point I(x, y) are I(x + u, y + v) Where $-\frac{w-1}{2} \leq u,v \leq \frac{w-1}{2}$. Here the scale of I(x, y) is

$$L(x,y,\sigma) = \sum_{u=-\frac{w-1}{2}}^{\frac{w-1}{2}} \sum_{v=-\frac{w-1}{2}}^{\frac{w-1}{2}} G(u,v) I(x+u, y+v) \quad \text{eqn 1}$$

Let $\sigma_0$be the initial value of σ in the Gaussian filter. Define $\sigma_i = k^i\sigma$ for 0 ≤i < s + 3.Let $L_0^0 = I$be the original image (the superscript is explained later

Octaves

The sequence of scales is called an octave.L_{s+1}^0as part of the first octave. This is a blurred image from the original image I. The next step requires a reduction in image resolution. The resolution of an image can be reduced1 by a factor of 2 in each dimension by sampling every other pixel of the image in a checkerboard pattern. Let L_0^1be L_{s+1}^0 reduced in resolution by a factor of 2 (the superscript j here denotes Octave 1 for $L_0^1$and Octave 0 forL_{s+1}^0).

We now define a new octave (second octave) analogous to the Octave 0.

$$L_0^1 \xrightarrow{G_\sigma} L_1^1 \xrightarrow{G_{\sigma_1}} L_2^1 \dots \xrightarrow{G_{\sigma_s}} L_{s+1}^1 \xrightarrow{G_{\sigma_{s+1}}} L_{s+2}^1 \xrightarrow{G_{s+2}} L_{s+3}^1 \qquad \text{eqn 2}$$

If there are $\hat{s}$octaves, then in general for $0 < j \leq \hat{s} - 1$, L_0^j is L_{s+1}^{j-1} reduced in resolution by a factor of 2 in each dimension and

$$L_0^j \xrightarrow{G_\sigma} L_1^j \xrightarrow{G_{\sigma_1}} L_2^j \dots \xrightarrow{G_{\sigma_s}} L_{s+1}^j \xrightarrow{G_{\sigma_{s+1}}} L_{s+2}^j \xrightarrow{G_{s+2}} L_{s+3}^j \qquad \text{eqn 3}$$

The time complexity for computing all (s + 3) scales of the image over one octave is $\Theta(N^2 w^2 (s+3))$and repeating this for $\hat{s}$ octaves gives a time complexity of $\Theta\left(\sum_{j=0}^{\hat{s}-1} \frac{N^2}{2^j} w^2 s\right) = \Theta(N^2 w s)$where the down sampling needed to start each octave requires another $\Theta\left(\frac{N^2}{2^j}\right)$operations. Therefore the overall complexity of this phase is $\Theta(N^2 w^2 s)$. Considering constants and unit time for all operations, this phase requires approximately $4N^2 w^2 s$time.

Difference of Gaussians

At this point, we have (s + 3) scales L_i^jover all $\hat{s}$octaves where $0 \leq j < s + 3$ and $0 \leq j < \hat{s}$. For any fixed octave j and$0 \leq j < s + 2$, define the ith difference of Gaussians over octave j as:

$$D_i^j = L_{i+1}^j - L_i^j \qquad \text{eqn 4}$$

Where the difference is for each pair of corresponding $L_{i+1}^j(x,y)$and $L_i^j(x,y)$ .The scales in octave j are $\frac{N}{2^j} \times \frac{N}{2^j}$ images. Then each of L_{i+1}^j and $L_i^j$is an$\frac{N}{2^j} \times \frac{N}{2^j}$ image and so computing D_i^jrequires finding $\frac{N^2}{2^j}$differences.

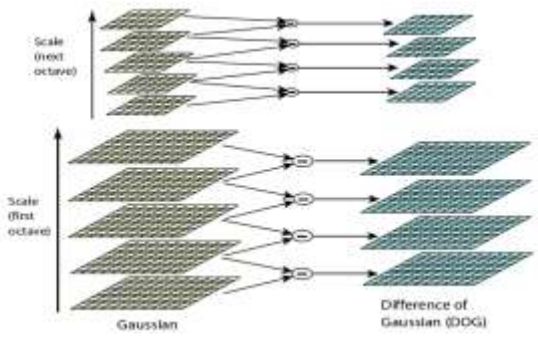


Fig 2 Scale, octaves and difference of Gaussians

In the above Fig 2 shows that difference of two scale-space images in single octaves. The same process is repeated till three octaves.

The complexity to compute this quantity across all $\hat{s}$octaves is$\Theta\left(\sum_{j=0}^{\hat{s}-1} \frac{sN^2}{2^j}\right) = \Theta(sN^2)$. With normally used values of SIFT parameters, the number of operations is approximately$4N^2 s$.

Extrema Detection

Suppose that we have the setsD_{i-1}^j, D_i^j, D_{i+1}^j of difference of Gaussian images in an octave j. For each octave j where $0 \leq j < \hat{s}$ and for$0 < j < s + 1$, place the difference of gaussiansD_{i-1}^j, D_i^j, D_{i+1}^jin three adjacent layers. Now element $D_i^j(x,y)$ has 26 neighboring difference of Gaussian elements. Element $D_i^j(x,y)$ is an extremum iff it is strictly larger (in pixel value) than all of the neighboring elements. Detecting whether$D_i^j(x,y)$ is an extremum takes at most 26 comparisons each requiring constant time. For all elements of D_i^j in an octave j, the time needed is$\Theta\left(\frac{N^2}{2^j}\right)$. For all difference of Gaussians over all $\hat{s}$octaves, the time needed$\Theta\left(\sum_{j=0}^{\hat{s}-1} \frac{(s+2)N^2}{2^j}\right) = \Theta(sN^2)$.

Before we proceed to the next phase (Keypoint Detection), we touch upon how the Scale-space Extrema Detection phase is executed. However each of these (scales, difference of Gaussians, extrema) has relatively local dependencies. That is, to determine the scale of a point, one needs to know only the w×w neighborhood of the point. To determine the difference of Gaussian, we only need two corresponding points and to check whether a point is extremum, we only need 26 differences of Gaussians points spread over three scales around it. Thus, it is possible to execute these operations over octaves in many different ways. The original algorithm of Lowe uses the structure. The program we used for this work uses the modified flow of algorithm.

2) Keypoint Detection

Recall that the algorithm first determines αN2extrema and then further distills them into αβN2 keypoints that will ultimately become keypoints of the image. The Scale-Space Extrema Detection phase of the algorithm identifies αN2 potential candidates for keypoints. Some of these candidates may lie along an edge of the image or may correspond to points of low contrast. These are generally not useful as features as they are unstable over image variation. Hence these points are rejected. For rejecting low contrast points, each extremum is examined using a method that involves solving a system of 3×3 linear equations and so it takes constant time. To detect the extrema on edges, a 2 × 2 matrix is generated and simple computations performed on it, to generate a ratio of principle of curvatures. This quantity is simply compared with a threshold value to decide whether an extremum is to be rejected or not. Thus, this phase runs in time over all octaves. Considering constants into the account this phase takes approximately operations. After the elimination of extrema points, the points that remain are called keypoints. Now nominally have key points.

3) Orientation Assignment

The nominal number of key points at the start of this phase is $\alpha\beta N^2$. This phase adds to the set of keypoints (those that may be missed in the previous phases) on the basis of their magnitude and orientation. The magnitude

$m_i^j(x,y)$ and orientation $\theta_i^j(x,y)$ for each point $L_i^j(x,y)$ can be calculated as follows:

$$m_i^j(x,y) = \sqrt{\left(L_i^j(x+1,y) - L_i^j(x-1,y)\right)^2 + \left(L_i^j(x,y+1) - L_i^j(x,y-1)\right)^2} \quad \text{eqn 5}$$

$$\theta_i^j(x,y) = \tan^{-1}\frac{L_i^j(x,y+1) - L_i^j(x,y-1)}{L_i^j(x+1,y) - L_i^j(x-1,y)} \quad \text{eqn 6}$$

Non-keypoint points whose magnitudes are close to the peak magnitude are added as new keypoints. The number of points examined is $N^2 - \alpha\beta N^2 \cong N^2$ as α andβ are small fractions. Of these, a fraction are added back. Thus, the total number of keypoints at the end of this phase is $\alpha\beta N^2 + \gamma(N^2 - \alpha\beta N^2) = \alpha\beta N^2(1-\gamma) + \gamma N^2 \cong N^2(\alpha\beta + \gamma)$ again because is a small fraction. Clearly the computation for $m_i^j(x,y)$ and $\theta_i^j(x,y)$ can be done over constant time. The overall complexity for all points over all octaves is $\Theta(sN^2)$. Considering the constants, the number of operations is approximately $48sN^2$

4) Keypoint Descriptor Generation

In this phase, the algorithm computes a descriptor for each keypoint identified so far. The descriptor is a collection of information in an 2x × 2x Neighborhood of the keypoint. The following tasks are undertaken for each Key point.

• The magnitudes of all the points in the neighborhood are smoothed by a normalized Gaussian filter with $\sigma = x$. This requires $\Theta(s^2)$ multiplications for each point.

• The neighborhood is divided into 4×4 regions. In each region the vectors (magnitude and direction of points) are histogrammed into 8 buckets covering 360° using trilinear interpolation. Again this requires $\Theta(x^2)$ time for the neighborhood.

• The feature is computed from these descriptors in the neighborhood by computing a normal of the descriptors in the neighborhood.

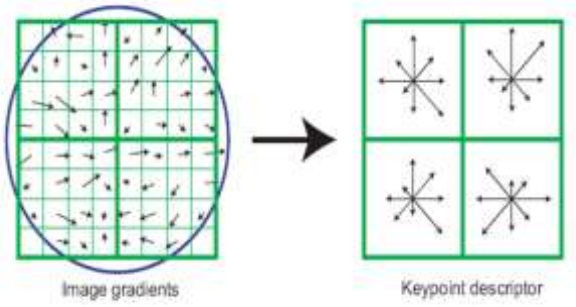


Fig 3 Keypoint descriptor generation

In the above Fig 3 shows that key point descriptors are generated using image gradients.

• The resulting descriptor is represented as a normalized $\frac{x}{2}$x $\frac{x}{2}$descriptor array each with an 8 bucket Histogram of vectors. Thus, the feature is $\log_2 \frac{8x^2}{4} = 2\log x + 1$bits long.

As the time complexity is $\Theta(x^2)$ for each keypoint identified so far, then the overall time complexity for all the keypoints is $\Theta(x^2(\propto \beta + \gamma)N^2)$ .Considering the constants^2, the number of operations is approximately$150x^2(\propto \beta + \gamma)N^2$.

B. Module 2

In this module Color Names (CN) descriptor was employs for two reasons. First, it is shown in that CN has superior performance compared with several commonly used color descriptors such as the robust hue descriptor and Opponent derivative descriptor. Second, although colored SIFT descriptors such as HSV-SIFT and Hue-SIFT provide color information, the descriptors typically lose some invariance properties and are high-dimensional. Basically, the CN descriptor assigns to each pixel an 11-D vector, of which each dimension encodes one of the eleven basic colors: black, blue, brown, grey, green, orange, pink, purple, red, white and yellow. The effectiveness of CN has been validated in image classification and detection application. We further test it in the scenario of image retrieval.

Feature Extraction:

At each keypoint, two descriptors are extracted, *i.e.*, a SIFT descriptor and a CN descriptor. In this scenario, SIFT is extracted with the standard algorithm. As with CN, we first compute CN vectors of pixels surrounding the keypoint, with the area proportional to the scale of the keypoint. Then, we take the average CN vector as the color feature. The two descriptors of a keypoint are individually quantized, binarized, and fed into our model, respectively.

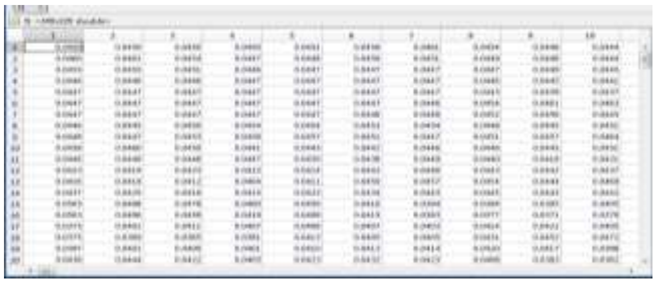
Estimation of Feature Correlation

The feature correlation is calculated between SIFT visual word and binary CN. To this end, we crawled 200*K* high-resolution images uploaded by users. These images are generally high-resolution, with the most common size of 1024 × 768. From the images, we extract over 2×109 (SIFT, CN) feature tuples. For feature tuples with the same/different SIFT visual word, compute the Hamming distance of CN features, and calculate the normalized distance histogram. Finally, The correlation coefficient is calculated between two histograms [5]. Briefly, the intuition is that, for highly independent features, the two histograms should be very similar: whether or not the SIFT visual words of a keypoint pair are the same, the Hamming distance of the other feature is not affected.

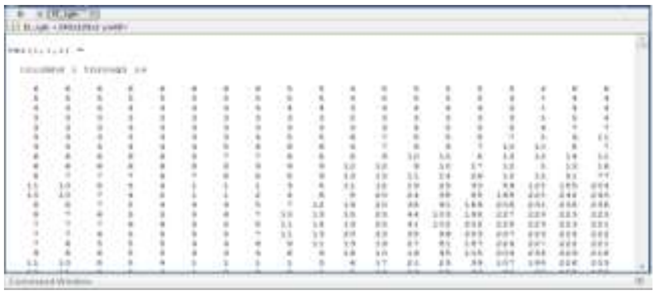
4. EXPERIMENT RESULT

After performing all the extraction of image features using SIFT and CN descriptor similar feature images are retrieved effectively from the databases.

The above image is our query image which is used for retrieving similar feature images after doing embedding of both two features that is SIFTS key and CN descriptor.

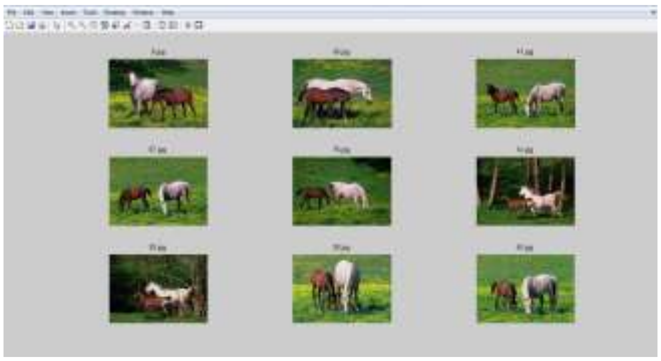


The above figure shows that the coordinate pixel values of query image. Those values are used for finding key values from a query image.



The above figure shows that the gray scale values of the query image. The value up to 0 to 255.

After done the conversion of gray scale octave of scale space is calculated for that image using Gaussian Filter which is used for removing noisy from that image and also getting the scalable pixel values.



The above figure shows that similar feature images are retrieved from the database after perform all the steps.

5. CONCLUSION

Binary Embedding methods are effective for visual matching verification. A coupled binary embedding method using a binary multi-index framework to fuse SIFT visual word with binary features at indexing level is proposed. To model the correlation between different features, a new IDF family is introduced, called the multi-IDF [4], which can be viewed as a weighted sum of individual IDF of each fused feature. In large-scale settings, by storing binary features in the inverted file, the proposed method consumes acceptable memory usage and query time compared with other approaches.

In the future work, more investigation will be focused on the mechanism of how features complement each other and promote visual matching accuracy. Since our method can be easily extended to include other binary features such as recently proposed ORB, BRISK , FREAK , etc, various feature fusion and selection strategies will also be explored to further improve performance.

REFERENCES

1. D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
2. D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2. 2006, pp. 2161–2168.
3. D. Qin and C. W. L. van Gool, "Query adaptive similarity for large scale object retrieval," in *Proc. IEEE Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 1610–1617.
4. David G. Lowe "Distinctive Image Features from Scale-Invariant keypoints"Computer Science Department University of British Columbia Vancouver, B.C., Canada,January 5, 2004.
5. H. Jégou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," in *Proc. 10th Eur. Conf. Comput. Vis. ECCV*, 2008, pp. 304–317.
6. J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. IEEE Int. Conf. Comput. Vis., (ICCV)*, Oct. 2003, pp. 1470–1477.
7. J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, 2007, pp. 1–8.
8. Liang Zheng, Shengjin Wang "Coupled Binary Embedding For Large-Scale Image Retrieval", IEEE transactions on image processing, vol. 23, no. 8, august 2014
9. L. Zheng, S. Wang, Z. Liu, and Q. Tian, "Lp-norm IDF for large scale image search," in *Proc. IEEE Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2013, pp. 1626–1633.
10. L. Zheng, S. Wang, and Q. Tian, "Lp-norm IDF for scalable image retrieval," *IEEE Trans. Image Process.*, to be published.