Sensitivity based image filtering for multi-hashing in large scale image retrieval problems

Article *in* International Journal of Machine Learning and Cybernetics · July 2015
DOI: 10.1007/s13042-015-0402-1

Wing W. Y. Ng
South China University of Technology
155 PUBLICATIONS **1,229** CITATIONS

Int. J. Mach. Learn. & Cyber. (2015) 6:777–794
DOI 10.1007/s13042-015-0402-1

ORIGINAL ARTICLE

Sensitivity based image filtering for multi-hashing in large scale image retrieval problems

Wing W. Y. Ng[1] · Jinchen Li[2] · Shaoyong Feng[1] · Daniel S. Yeung[1] · Patrick P. K. Chan[1]

Received: 10 February 2015 / Accepted: 14 July 2015 / Published online: 26 July 2015

Abstract Hashing is an effective method to retrieve similar images from a large scale database. However, a single hash table requires searching an exponentially increasing number of hash buckets with large Hamming distance for a better recall rate which is time consuming. The union of results from multiple hash tables (multi-hashing) yields a high recall but low precision rate with exact hash code matching. Methods using image filtering to reduce dissimilar images rely on Hamming distance or hash code difference between query and candidate images. However, they treat all hash buckets to be equally important which is generally not true. Different buckets may return different number of images and yield different importance to the hashing results. We propose two descriptors, bucket sensitivity measure and location sensitivity measure, to score both the hash bucket and the candidate images that it contains using a location-based sensitivity measure. A radial basis function neural network (RBFNN) is trained to filter dissimilar images based on the Hamming distance, hash code difference, and the two proposed descriptors. Since the Hamming distance and the hash code difference are readily computed by all hashing-based image retrieval methods, and both the RBFNN and the two proposed sensitivity-based descriptors are computed offline when hash tables become available, the proposed sensitivity based image filtering method is efficient for a large scale image retrieval. Experimental results using four large scale databases show that the proposed method improves precision at the expense of a small drop in the recall rate for both data-dependent and data-independent multi-hashing methods as well as multi-hashing combining both types.

Keywords Multi-Hashing · RBFNN · Image filtering · CBIR

✉ Jinchen Li
kingsion21@163.com

Wing W. Y. Ng
wingng@ieee.org

1 School of Computing Science and Engineering, South China University of Technology, Guangzhou, China

2 College of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou, China

1 Introduction

An efficient similarity measure between images in a large scale database is a key to the success of content-based image retrieval (CBIR) [1–3]. For a given query image E, the task is to find candidate images (Is) which either have similarity measures, $sim(E, I)$,larger than a given threshold, or rank among the top k images with the largest $sim(E, I)$. The similarity measure could be either Euclidean based or a semantically learnt one [4]. Obviously it is both inefficient and unnecessary to compute similarity measures between the query and all images in a large scale database. Two major approaches have been proposed to find a subset which is much smaller than the whole database and contains images similar to the E: tree-based and hashing methods. The tree-based methods, e.g. kd-tree [5], are inefficient for high dimensional data and they require a large amount of memory to store the data structure [6]. The hashing methods project images onto a Hamming space such that each image is represented by a b-bit hash code, where b is usually much smaller than the number of bits required for image descriptors to describe an image. Images having the same hash code as E does, i.e. Hamming distance $(h) = 0$, are treated as Is similar to E. The computation of the Hamming distance is extremely fast, especially in comparison with the computation of the Euclidean based distance. Therefore, hashing methods are more appropriate for large scale image retrieval problems.

However, hashing methods using a single hash table may yield unsatisfactory recall rate when only Is in buckets with $h = 0$ are returned. Therefore, Is in hash buckets with $h > 0$ are also returned when a higher recall rate is required. For a 64-bit hash code, $h \leq 2$ will require a visit of 2081 hash buckets which significantly reduces the efficiency of the hashing method. One way to overcome this problem is to use multiple hash tables to return Is in buckets with $h = 0$ from m different hash tables (multi-hashing). In this way, only m hash buckets are visited to return a sufficient number of similar images.

Current multi-hashing methods taking the union of Is from all hash buckets with $h = 0$ will return a large number of both similar and dissimilar Is. Hence an image filtering technique is needed to reduce the number of dissimilar Is. The one proposed in the Bayesian locality sensitive hashing (BLSH) [7] computes the Hamming distances between the E and Is in all hash tables. Its major drawback is the assumption of equal importance of all hash bits and all hash tables. In some hashing methods, each bit may have a different importance. Moreover, some hash buckets may have a higher chance to contain more similar images. On the other hand, owing to the geometry of sets of Is returned by different hash buckets, Is located farther away from the intersection area of all $h = 0$ buckets have a higher chance to be dissimilar. These observations motivate us to propose a new image filtering method based on geometric sensitive descriptors in the image descriptor space which defines hash buckets.

In this work, we propose a sensitivity based image filtering method (SIF) using a radial basis function neural network (RBFNN) to approximate the similarity of I with respect to an E. Four descriptors are used to describe an I: Hamming Distance (h) between the I and the E, exclusive-OR (XOR) difference (H) between hash codes of the I and the E, the location sensitivity measure (LSM) of I and the bucket sensitivity measure (BSM) of I. For a m table multi-hashing, there are $m(b + 3)$ descriptors to be computed, where b denotes the number of hash bits. The RBFNN takes $m(b + 3)$ inputs and outputs the values which will be used to filter Is with respect to the E. The RBFNN is used for SIF because of its fast training for large scale datasets. The RBFNN is trained via a minimization of the localized generalization error model [8] for a better robustness with respect to future unseen samples. Both the LSM and the BSM are computed offline when hash tables and database become available. The RBFNN is also trained offline and it is efficient in reaching an image filtering decision. Hence the overall time complexity of the image filtering task using SIF is not worse than other filtering methods which only compute the h and H. Moreover, the proposed SIF, the BSM and the LSM will work for any hash method with well defined hash buckets.

The major contribution of the proposed SIF to hashing based image retrieval problems is to reduce the need of time consuming image similarity computation, which requires hard disk reading and exact Euclidean distance computation, by filtering out dissimilar images using a RBFNN with fast computed h and H, and pre-calculated LSM and BSM for multi-hashing.

This paper is organized as follows: Section 2 discusses the related works. The four descriptors and the SIF training method are presented in Sect. 3. Section 4 shows the detailed experiments on four large databases. This work is concluded in Sect. 5.

2 Related works

Section 2.1 gives a brief survey on major hashing methods. Sections 2.2 and 2.3 discuss how to build multiple hash tables (multi-hashing) and current image filtering methods to enhance the precision rate, respectively.

2.1 Hashing methods for large scale similarity search

This section discusses three major hashing methods: the locality-sensitive hashing (LSH), the principle component hashing (PCH), and the semi-supervised hashing (SSH).

The most important property of hashing is to project similar images into the same hash code (F), or at least two Fs within a small difference. LSH provides a theoretical foundation of preserving the locality similarity of two images [9]. Given images x_l and x_j, a hash code F created by LSH preserves the following property [10]:

$$Pr(F(X_l) = F(x_j)) = sim(x_l, x_j) \tag{1}$$

where $Pr(.)$ denotes the probability and $sim(.)$ is a similarity measure. The ith bit hash function of the F is in the form of a hyperplane $(f_i = o_i x + a_i)$ where o_i and a_i are randomly sampled from a p-stable distribution with $p \in (0, 2]$. Gaussian distribution is widely adopted for the l_2-norm distance measure. Usually, a similarity measure in LSH is based on the Euclidean distance. This is efficient when the Euclidean based ground truth is adopted for the CBIR similarity. The performance of a LSH based method increases when the number of hash bits increases. However, a long hash code will reduce the recall rate for the hash lookup because a higher similarity is needed for two images to share the same hash code. The LSH hash functions can be generated quickly, but they ignore the structural and semantic information provided by the database. Cosine similarity measure is another widely used replacement of Euclidean distance for the LSH generation [10]. In many CBIR problems, semantic similarity is more important while the Euclidean distance between image descriptors can not always provide a good estimation for the semantic similarity [11]. In [12], the Chi-Square χ^2 distance is proposed to replace the Euclidean one. In [13] the Euclidean distance is replaced by a learned metric based on the Mahalanobis distance function capturing pairwise similarity and dissimilarity information between images. An adaptation of the LSH to a kernel based similarity measure is proposed in [1].

The major advantage of the data-independent LSH is that two images with a large similarity yields a small Hamming distance between their LSH hash codes. However, the LSH and its variants require a large number of bits to preserve the locality similarity. To fully utilize label or pairwise similarity information in a CBIR database, data-dependent methods are proposed with hashing methods based on, for instance, the principal component analysis [14] or the semi-supervised learning [2]. Data-dependent methods require fewer bits to yield higher precision and recall rates. A two-phase mapping hashing [15] is proposed to map real-valued image descriptors to a high dimensional Hamming space and then map it to a low dimensional Hamming space to find a compact hash code. The asymmetric cyclical hashing [16] is proposed to reduce the storage cost of huge volume of images while maintaining high precision and recall rates.

When images in a database are not uniformly distributed, some areas in the image descriptor space will be sparse while others may be dense. More hash buckets may be required in the dense area to provide a better resolution of similarity between images in the Hamming space. The principal component analysis (PCA) provides an efficient way to capture the image distribution. The principal component hashing (PCH) [17] projects an image from the image descriptor space onto a space of selected principal components. Then each selected principal component axis is divided into a given number of buckets equally in terms of the probability of the training image occurrence along the axis. However, not every principal component is equally important since those with higher variances may contain more variation information. Therefore if all principal components are encoded with the same number of hash bits, the search performance may degrade. Three hashing methods have been proposed to deal with this issue: the iterative quantization for PCH (ITQ) [14], the semi-supervised hashing (SSH) [2, 18] and the spectral hashing (SH) [18–20].

The ITQ first performs a random orthogonal projection to the resulting PCH to balance the differences in variances of different principal components. It then refines the rotation via a minimization of the quantization error of hashed images to push images forward to the closest vertex of the $(\pm1)^b$ hypercube in the Hamming space. The ITQ works well in the Euclidean based ground truth and it has been extended to the semantic ground truth by replacing the PCA with the Canonical Correlation Analysis.

The SH constructs hash codes by partitioning a spectral graph which is created based on the pairwise similarity information between images. The spectral graph is computed based on the Euclidean distance in [21]. The Sparse SH [19] adopts a sparse PCA and formulates the hashing problem by thresholding a subset of eigenvectors of the Laplacian graph by constraining the number of non-zero features. In [20], the hash code is constructed by directly optimizing the graph Laplacian to provide a better representation of the similarity between images. The SH requires the supervised information about the pairwise similarities between images which is obtained by either a threshold on the Euclidean based distance between images [19, 21] or semantic labels [20]. Other supervised data-dependent hashing methods include LDAHash [3], minimal loss hashing [22], binary reconstruction error [23] and semantic hashing [24].

The SSH is proposed to take advantage of both the supervised and the unsupervised learning for hashing. It minimizes the empirical error over the labeled images regularized by an information theoretic regularizer over both labeled and unlabeled images. There are three different ways to create hash functions in SSH [2]: orthogonal, non-orthogonal and sequential projection. The sequential projection learning (SPLH) iteratively adjusts the pairwise label matrix to implicitly incorporate bit correlation. In [18] images are projected nonlinearly before performing a bootstrap SPLH to reduce the quantization error between real valued outputs of the hash function and the corresponding hash codes.

2.2 Multi-hashing

LSH based methods require long hash code to provide high similarity precision rate, but it also significantly reduces the recall rate of the similarity search with hash lookup. Moreover, a 32-bit hash code yields $2^{32} \approx 4$ billions of hash buckets which create a lot of empty hash buckets even for a database consisting of one billion images. The exponential increase of the number of hash buckets visited with respect to the increase of h in the hash ranking reduces the efficiency in a large scale similarity search. Multiple hash tables are created randomly to increase the recall rate of the LSH with hash lookup. When m b-bit hash tables are created, LSH preserves the locality similarity of two images (x_l and x_j) as follows [2]:

$$Pr\left(f(x_l) = f(x_j)\right) \propto m\left(1 - \frac{\cos^{-1}\left(x_l^T x_j\right)}{\pi}\right)^b \qquad (2)$$

Note that instead of reducing b, one can increase m to achieve a higher recall rate.

Data-dependent methods can yield a better precision rate comparing to the data-independent LSH based methods. However, they suffer from a low recall rate when only a single hash table is used. The complementary hashing (CH) is a data-dependent multi-hashing method which creates multiple hash tables with boosting [25]. These hash tables created by complementary hashing are dependent because each table, except the first one, is boosted to compensate the error created by previous tables. The dual complementary hashing (DCH) combines the SPLH and the CH to build multiple hash tables [26]. The learning method of an individual hash table is similar to the SPLH's while each new hash table is constructed to compensate error made by previous hash tables using a boosting based method. When the similarity of two images is learned correctly by the current hash table, its corresponding entry in the weight matrix of the boosting will be set to zero. Therefore, after a number of hash tables being built, new hash table built by the DCH will be similar to the immediately preceding one. It restricts the DCH from gaining better precision and recall rates by creating more hash tables.

2.3 Image filtering

Returning all retrieved images to user directly can yield a poor user experience owing to a large number of irrelevant images (i.e. low precision). Re-ranking of retrieved images has been applied in text-based image retrieval problems [27] to enhance the precision of relevance between query text and images belonging to semantic concepts. On the other hand, query-adaptive weights are proposed to weight hash codes of the E in different semantic classes differently [28]. A weight vector is learned for the hash codes of each semantic class. The semantic class of the E is approximated using a k-nearest neighbors approach. Then, a weighted Hamming distance between hash codes of the E and images in the database is computed to enhance the retrieval precision. This approach is limited to the semantic query only and inappropriate for the large scale image retrieval problems because the computation of weighted Hamming distance of all images in the database is very time consuming.

A high recall rate is achieved by taking the union of all retuned Is from multiple hash tables. However, this will reduce the efficiency of exact similarity search or final ranking results of CBIR after hashing. Image filtering can be used to reduce redundant dissimilar Is prior to exact similarity comparison. The Bayesian LSH (BLSH) [7] proposes a Bayesian based method to evaluate each I from the multi-hashing with a randomly selected subset of hash codes from all hash tables. The filtering method which computes weighted Hamming distances of all images in the database [28] is not suitable for large scale image retrieval problems. The BLSH and our proposed method only compute Hamming distances of all images in the database and are more suitable for large scale problems. Even the exhaustive computation of millions of Hamming distances will not take a second using a single CPU [29].

However, the BLSH assumes all hash bits of all hash tables to be independent. This assumption is invalid for the PCA based and the semi-supervised methods which do not create hash functions randomly and independently. Moreover, hash buckets should not be treated equally. Therefore, we are motivated to propose a RBFNN based image filtering method and two new descriptors to score both the candidate image and buckets that contain it.

3 Sensitivity based image filtering

The SIF consists of two parts: a RBFNN trained via a minimization of its localized generalization error (L-GEM) and four descriptors to capture the similarity of a candidate image (I) to the query image (E), i.e. $sim(E, I)$. The four descriptors were mentioned in the Introduction section and they are: the Hamming distance (h), the hash code difference (H) the LSM and the BSM. The h measures the approximated dissimilarity between the E and the I, the H shows the hash bits yielding the non-zero h, the LSM measures the location of the I with respect to the boundary of its hash bucket, while the BSM measures the goodness of a hash bucket based on the ratio of its images located near the bucket boundary. For each of m hash tables of the multi-hash, all descriptors are computed for each table, so there are $m(3 + b)$ inputs of the RBFNN for each I. Each of these descriptors alone will not give a clear picture on the $sim(E, I)$. A RBFNN is used to learn the nonlinear relationship between the $sim(E, I)$ and the descriptors. The RBFNN cannot find the exact $sim(E, I)$. Instead, the major contribution of the SIF is to build an RBFNN to approximate $sim(E, I)$ with a fast computation to filter out a large portion of dissimilar Is using h, H, and the pre-calculated LSM and BSM. As aforementioned, the computations of both h and H are fast and they are required for any CBIR using a hashing method, so their computational costs can be ignored as far as filtering is concerned.

The proposed SIF method can be applied to any hashing method which creates hash buckets in the original descriptor space, e.g. the LSH, the PCH, the DCH, the SH and the SSH. The hash bucket may be either a closed or an open bucket as long as it contains a finite number of images. As long as the original descriptor of the images in the database is numerical, e.g. the GIST and the SIFT for images and term frequency for documents, the SIF can be used.

When a hash table is changed, its BSMs and the corresponding LSMs need to be recomputed and the RBFNN must be retrained with a new training set. If a new image is added to the database, the BSM and the LSM related to this image must be computed, but the re-training of the RBFNN in this situation is optional.

The computation of descriptors and the L-GEM based RBFNN training method are shown in Sects. 3.1 and 3.2, respectively. The time complexity of the SIF is analyzed in Sect. 3.3.

3.1 *H*, *h*, and two new descriptors for SIF

For a given query E, each of the m hash tables computes the hash code of E with respect to its own b-bit hash functions. The Is with the same hash code of E in at least one table are returned to form the Union of all Is from the m hash tables. The problem of SIF is to filter out dissimilar Is from the Union of Is to speed up the final exact match or to reduce the list of images returned to the users. It is impossible to estimate the location of the E within a hash bucket without computing its Euclidean distance with respect to hash functions, which will be expensive for a large scale CBIR. The h, H, and the two new descriptors are proposed to provide approximated measures to the Is in the Union. The h answers the question whether an I and the E are located in the same bucket. The H finds hash bits which the hash codes of the E and the I are different. The LSM (Sect. 3.1.1) measures the relative location of an I with respect to its own bucket, and the BSM (Sect. 3.1.2) quantifies the chance of an I to be similar to E.

For any hashing-based image retrieval methods, the H is computed by the exclusive-OR (XOR) operator between hash codes of the I and the E. The h is computed by counting the number of bits being different in the hash codes of the I and the E, i.e. the number of 1s in the H. Therefore, both the H and the h are computed online when the E arrives and the hashing method provides them to the SIF without any extra computation. Owing to the fact that the intersection of hash buckets having $h = 0$ usually cannot return enough Is, a multi-hashing method yields a better result by returning Is having $h = 0$ in at least one hash table among the m tables. These two values provide important references to the approximated distances between Is and the E.

3.1.1 *Computation of LSM*

Figure 1 shows a two dimensional artificial image descriptor space with three hash tables (F_1, F_2 and F_3) which are represented by the three buckets containing the E (i.e. $h = 0$). Any I falls in the blue circle centered at the E (the red star) is regarded as similar to the E in the sense of the Euclidean ground truth. As shown in Fig. 1, if an I yields $h = 0$ for all three tables (the black triangle and the Is in the shaded area), then it has a very high chance to be similar to the E. However, two Is yielding the same h values for all hash tables may not necessary yield the same similarity to the E. The green dot and the orange square demonstrate such a situation. For instance, an I located at the boundary of its hash bucket with large h for some hash tables, is dissimilar to the E. Moreover, if an I located very close to the decision hyperplanes of many hash functions, it has a higher chance to be mis-hashed to its current bucket. An I is said to be mis-hashed into its current bucket mistakenly if it has similar images located in another bucket. Mis-hashing is an important research issue in hashing methods. In this case, we avoided computing real hash functions and ignored the computation of the H for simplicity.

By the locality preserving principle, images located close to each other (similar) should be hashed into the same bucket. So, we define a local neighborhood of I, i.e. yellow circles centered at the green dot and the orange square (see Fig. 1), to find out the chance of an I being mis-hashed. The yellow circle represents the area which contains images similar to I at the center and they are expected to be hashed in the same bucket. When the yellow circle (the one centered at the green dot) is cut by at least one hash function, images in it have a chance to be mis-hashed. If the yellow circle is cut by many hash functions, a small perturbation in

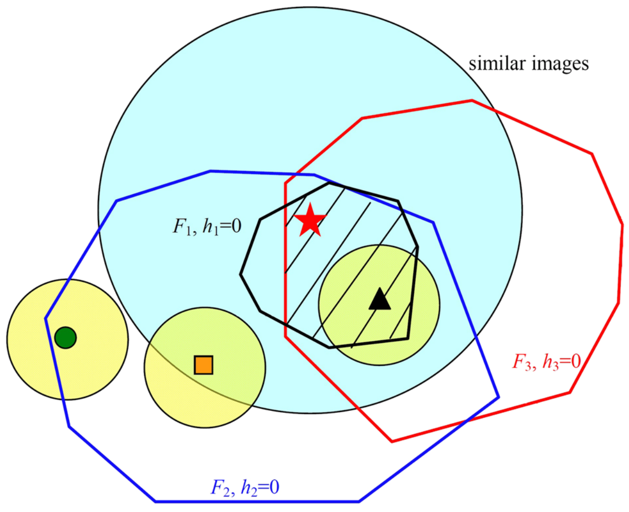

Fig. 1 Images returned by 3 hash tables

the original image descriptors of the *I* will push it to another hash bucket with a high *h* value, i.e. *I* is unreliable. A small perturbation could be due to either noise or minor change in the descriptor features, e.g. brightness or color. The LSM is defined as follows:

Algorithm 1 The computation of LSM for an image and a hash table

```
Require: An image $x$ in the database, the similarity threshold $C, b$ hash functions of a table $F() = [f_1(), f_2(), \cdots, f_b()]$;
Ensure: LSM
  for $i = 1$ to $b$ do
    Compute distance $d_i$ between the image $x$ and $f_i(x)$
  end for
  Count the number of $d_i < C$ and denote this number as $c_1$;
  LSM $= c_1/b$
  Retrun LSM
```

In Fig. 1, with respect to the hash table F_2, the LSM of the green dot is 2 / *b* while the LSM of the orange square is zero. A non-zero LSM value means that the *I* is located within the margin determined by *C*. The LSM is only a relative measurement. With a fixed *C* value, a larger LSM value of an *I* indicates that it is in a region with a higher chance of probably mis-hashed (i.e. similar becomes dissimilar). The pre-selected threshold *C* indicates the percentage of image descriptor value that we can tolerate for artificial similar images near *I*. It needs to be selected empirically. By cross-validation, we found 0.01 to be a reasonable choice which yields good results in our experiments. Further fine tuning of *C* may produce a better result. The value 0.01 means a 1 % difference from the *I* in image descriptor values when the image descriptors are scaled to [0, 1]. For problems with Euclidean ground truth, the Euclidean distance based range between an image and its farthest similar image can be used as the value of the threshold *C*.

An *I* having a large LSM value is not necessarily dissimilar to *E*. For example, *E* (red star) itself appears to have a LSM = 2 / *b* with respect to the F_2 hash table. However, in practical situations, it is impossible to know either the location or the relative direction of the *E* with respect to a hash bucket. The LSM only provides a partial knowledge about the location of *I* itself. Additional information such as the *h* and the BSM (to be discussed in the next section) will be needed to provide a better picture of the approximated similarity between *I* and *E*.

3.1.2 Computation of BSM

The BSM is intended to measure the “reliability” of a bucket. When *I*s and the *E* are hashed into the same hash bucket (code), all such *I*s are included in the Union of the returned images. A bucket is considered “unreliable“ if a large portion of its returned *I*s is likely to be mis-hashed (LSM > 0), i.e. with a high chance to be dissimilar to the *E*. The BSM gives the percentage of *I*s in a bucket with LSM > 0. It provides a fast, intuitive and practical way to assess the approximated similarity between an *I* and the *E* located in the same bucket. Again the BSM should not be used as the sole indicator for such a purpose. It is computed as follows:

Algorithm 2 The computation of BSM values for all buckets of a hash table:

```
Require: The similarity threshold $C, b$ hash functions of a table $F() = [f_1(), f_2(), \cdots, f_b()]$, an integer $c_2$;
Ensure: BSM values for all non-empty buckets of the hash table
  for each nonempty bucket $B_i$ do: do
    Return all $N_i$ images in $B_i$;
    $C_2 = 0$
    for each image $x$ in $B_i$ do
      if $x$ yields LSM ¿ 0, then $c_2 = c_2 + 1$
    end for
    BSM$(B_i) = c_2/N_i$;
  end for
  Retrun BSM
```

The LSM and the BSM predict the relevance of the *I* with respect to the *E* from two different points of view. In contrast to the LSM which measures the chance of an *I* being hashed into the current bucket by mis-hashing at the image level, the BSM measures the overall percentage of *I*s located in a hash bucket with a chance of mis-hashing at the bucket level. When the BSM = 0, it implies that all images located within the bucket yield LSM = 0 and none of these images is located near the boundary of the bucket, i.e. not likely being mis-hashed. When the BSM = 1, all images in the bucket is located near the boundary which may indicate a very small hash bucket (in size) or images are not hashed well. However, when the $1 \geq$ BSM > 0, no implication on the values of the LSM can be said.

Assume that the database shown in Fig. 1 consists of three images only (i.e. the green dot, the black triangle and the orange square). BSM values of the bucket F_1, F_2, and F_3 are 1, 1/3, and 0, respectively. Images located within the bucket F_1 is highly likely to be similar to the *E* (red star). However, the bucket F_3 yields a zero BSM value, but it contains the same image as the F_1 does. This shows that the BSM alone does not give a full picture about the similarity between the *I* and the *E*. In this case, the black triangle yields $h = H = 0$ for all hash tables, so it is the most relevant images. However, as aforementioned, when either $H > 0$ or $h > 0$, the BSM and the LSM help to provide additional evidences to the relevance of the *I* with respect to the *E*. Hence it will be difficult to provide a simple model to combine all these four descriptors to form the best image filtering model. The SIF is proposed to use a L-GEM-based RBFNN to relate the four descriptors to the similarity between an *I* and the *E*.

3.2 L-GEM based RBFNN training method

The BSM is computed per bucket in each hash table and the LSM is computed for each *I* per hash table. Both the *H* and the *h* are computed between the *E* and the *I* for each hash table. So, for each *I* in a multi-hashing problem with *m* tables of *b*-bit hash functions, there is a $m(3+b)$ input vector (z) for the RBFNN and the learning target output (R) is the decision of dissimilarity ($R=0$) or similarity ($R=1$) between the *I* and the *E*. The *R* takes either 0 or 1 for simplicity. User can use the exact similarity value (*sim*(*E*, *I*)) for the *R*, but it may not be necessary as we target to learn an approximated discrete decision of image filtering only. The major objective of the RBFNN learning is to output a value near 1 for a similar image and 0 for a dissimilar image. The RBFNN output value approximates *sim*(*E*, *I*). In addition to fast training and response, the RBFNN is also adaptable to learn different ground truths, databases and hashing methods.

A training dataset is created by applying a number of queries to the database with a given ground truth of similarity decisions. For each query, we label all returned *I*s for their similarity to the *E* according to the ground truth being used. The total number of *I*s for all queries is denoted by *N*. With the sampled training set of input-output pairs, we train a RBFNN via a minimization of its Localized Generalization Error (L-GEM) [8]. A RBFNN trained via a minimization of L-GEM yields smaller generalization error for future unseen images and is more robust to perturbations in image descriptors. Both robustness and generalization capability are vital to the RBFNN in a large scale CBIR. Feature selection on the $m(3+b)$ input features may further improve the accuracy and efficiency of the RBFNN [30]. In our experiments, all $m(3+b)$ input features are used.

The number of training samples collected from the database (*N*) is small, in comparison to the large size of the database. *N* is selected by user. A larger *N* may result in a better filtering but need more training and data collection times. Indeed there are $m2^b$ buckets for an *m*-table *b*-bit multi-hashing, and we may not be able to collect samples from all possible hash buckets. Any future query may create descriptor values different from those of the training samples. The L-GEM based RBFNN training algorithm minimizes both the training error on the training *I*s and also the sensitivity of RBFNN with respect to the changes of the input features. These changes may occur when a new *I* appears which is likely to be different from any training *I*, but similar to some of them. The difference between the new *I* and the training ones is defined as input perturbations in the L-GEM. The training algorithm of the RBFNN is given in Algorithm 1 and we will briefly introduce the L-GEM of the RBFNN here.

First, we define a *Q*-neighborhood of each training image as follows:

$$S_Q(z_i) = \{z | z = z_i + \Delta z; ||\Delta z||_\infty \le Q\} \tag{3}$$

where z_i and Δz denote the descriptor vector extracted from the *i*th image and the perturbation to original image descriptors, respectively. We select the value of 0.01 for the *Q*, same as the similarity threshold of *C*. Further fine tuning on the *Q* value by cross-validation is expected to yield better results. A *Q*-Union (S_Q) is defined as the union of *Q*-neighborhoods of all training images. Every perturbed training image in the *Q*-Union represents a non-training unseen image. Δz is assumed to follow a uniform distribution because, without further distribution information about unseen samples, every unseen sample in the *Q*-neighborhood has the same chance to occur. The sensitivity measure (SM) of a classifier is defined as the average value of squared output differences (Δr) between training images and the unseen samples in the *Q*-neighborhoods. A RBFNN (r) is defined as follows:

$$r(z) = \sum_{j=1}^{M} w_j exp\left(\frac{\sum_{k=1}^{n}(z_i - u_{jk})^2}{-2v_j^2}\right) \tag{4}$$

where M, n, z_k, w_j, v_j and u_{jk} denote the number of hidden neurons, the number of features in image descriptors, the *k*th feature value of the image *z*, the connection weight between the output neuron and the *j*th hidden neuron, the width parameter and the *k*th feature value of the center parameter of the *j*th hidden neuron, respectively. The maximum value of *M* is bounded by 300 in Algorithm 1. This ad-hoc choice is selected via a cross-validation and it gives good results in our experiments. The SM of the RBFNN is defined as follows:

$$\begin{aligned}
E((\Delta r)^2) &= \frac{1}{N}\sum_{i=1}^{N}\int_{S_Q(z_i))}(r(z_i) - r(z_i + \Delta z))^2 p\Delta z d\Delta z \\
&= \sum_{j=1}^{M}\varphi_j \left(\begin{array}{c} exp\Big(4\Big(\sum_{k=1}^{n}\sigma_{\Delta z_k}^2(\sigma_{z_k}^2 + (\mu_{z_k} - u_{jk})^2 \\ +0.2\sigma_{\Delta z_k}^2)\Big)/2v_j^4 - 2\sum_{k=1}^{n}\sigma_{\Delta z_k}^2/2v_j^2\Big) \\ -2exp\Big(\sum_{k=1}^{n}\Big(\sigma_{\Delta z_k}^2(\sigma_{z_k}^2 + (\mu_{z_k} - u_{jk})^2 \\ +0.2\sigma_{\Delta z_k}^2)/2v_j^4\Big) - \sum_{k=1}^{n}\sigma_{\Delta z_k}^2/2v_j^2\Big) + 1 \end{array}\right) \\
&\approx \sum_{j=1}^{M}\varphi_j \frac{\sum_{k=1}^{n}\sigma_{\Delta z_k}^2\left(\sigma_{z_k}^2 + (\mu_{z_k} - u_{jk})^2 + 0.2\sigma_{\Delta z_k}^2\right)}{v_j^4} \\
&= R_{SM}^*(Q)
\end{aligned} \tag{5}$$

where $\Delta r = (r(z_i) - r(z_i + \Delta z))$,
$\varphi_j = (w_j)^2 exp\Big(var(\|z-u_j\|^2)/2v_j^4 - E(\|z-u_j\|^2)/v_j^2\Big)$,
$var(\|z-u_j\|^2) = \sum_{k=1}^{n}\Big(E((z_k-\mu_{z_k})^4) - (\sigma_{z_k}^2)^2$
$+4\sigma_{z_k}^2(\mu_{z_k} - u_{jk})^2 + 4E((z_k - \mu_{jk})^3)(\mu_{z_k} - u_{jk})\Big)$,
$E(\|z-u_j\|^2) = \sum_{k=1}^{n}(\sigma_{z_k}^2 + (\mu_{z_k} - u_{jk})^2)$, and μ_{z_k} and $\sigma_{z_k}^2$ denote the mean and the variance of the z^{th} feature of image descriptors, respectively.

Similar to all machine learning methods, the performance of the trained classifier depends on the quality of training samples. One cannot expect a classifier to correctly classify unseen samples that are very different from its training samples. Therefore, the SM is combined with the training error to form the L-GEM which provides an upper bound for the Mean Square Error (MSE) of a RBFNN for unseen samples located within the Q-Union of training samples [8]. The L-GEM is defined as follows:

$$R_{SM}(Q) = \int_{S_Q}(R(z) - r(z))^2 p(z)dz \tag{6}$$

where $R(z)$ and $p(z)$ denote the target decision of image filtering of the image z and the true unknown probability density function of the z. The $p(z)$ is unknown and therefore Eq. 6 can not be computed exactly. Its upper-bound is estimated using a Hoeffdings inequality [8]. With a probability of $(1-\eta)$, we have

$$R_{SM}(Q) \le (\sqrt{R_{emp}} + \sqrt{E((\Delta r)^2)} + \varepsilon) = R_{SM}^*(Q) \tag{7}$$

where $\varepsilon = B\sqrt{\ln\eta/(-2N)}$. R_{emp}, A, B and N denote the training MSE, the difference between the maximum and minimum values of the outputs, the maximum possible value of the MSE and the number of training samples, respectively. A and B do not affect the training significantly because they are constants. In our experiments, $A = B = 1$.

3.3 Time complexity of SIF

When an E is presented, its hash code is computed. Then Is located in the union of $h = 0$ hash buckets from all hash tables are returned. The Euclidean distances between the E and Is will be computed and they will be used to sort the Is in decreasing distances for returning to the user. Without any filtering, the time complexity for sorting the Is in this way is $O(nmb + Jn)$, where n, m, b and J denote the number of original descriptor features, the number of hash tables, the number of hash bits and the number of Is in the Union of exact match for all hash tables respectively. The $O(nmb)$ is needed for the computation of the hash code of the E, and the $O(Jn)$ is needed for computing the Euclidean distances between the E and all J candidate images. The major drawback of computing distances between the Is and the E in comparison with image filtering is the need of time consuming hard disk access for retrieving the original descriptor values for all Is.

So, the major contribution of filtering is to reduce the number of Is (J) to a smaller value (J^*) for fewer hard disk accessing. The time complexity of the BLSH is $O(nmb + Jmb^{'} + J^*n)$ where $b^{'}$ denotes the number of bits used in the BLSH which is smaller than b. The time complexity of the SIF is $O(nmb + JmbM + J^*n)$ where M denotes the number of hidden neurons of the RBFNN. Overall, time complexities of the SIF and the BLSH are similar. Hash codes of images and data required for the SIF and other filtering methods can be stored in memory. The additional time of filtering is small in comparison to time required for the hard disk access and the computation of exact similarity for those filtered images.

Before being deployed for query, all hash methods require the computation of the hash code for every image in the database. This requires a time complexity of $O(Tnbm)$, where T denotes the number of images in the database. Each hash function is defined by a hyperplane:

Algorithm 3 L-GEM based RBFNN Training Method

```
Inputs: The size of neighborhood $Q$, Training samples in form of input-output pairs$(z, R)$, The number of training samples $N$;
Outputs: RBFNN$(r)$
  Step1: Set $M = 10$;
  Step2: Train a RBFNN with $M$ hidden neurons using the training images;
      Step2.1: Perform k-means clustering with $M$ clusters to find the center vectors $(u_j)$;
      Step2.2: The width parameter $(v_j)$ is computed by the average of the distances among all centers;
      Step2.3: Connection weights are computed by a pseudo-inverse method minimizing the least square error;
  Step3: Calculate the $R^*_{SM}(Q)$ value using Equation 7 for the newly trained RBFNN;
  Step4: If $M <$ threshold (e.g. 300), then $M = M + 10$ and go to Step 2;
  Step5: $r =$ RBFNN yielding the minimum $R^*_{SM}(Q)$ value;
  Retrun $r$;
```

$f(x) = (ox + a)$. The hash code is computed by the $sign(f(x))$ where $sign(f(x)) = 1$ if $f(x) > 0$ and 0 otherwise. The distance between the hash function and a given image for computing the LSM is equal to the absolute value of $f(x)/\|o\|$ where $\|o\|$ is a constant after the hash function is generated. So, the computation of the BSM and the LSM can be combined with the computation of hash code for each image in the database to reduce the total computational time. The total time complexity of computing all hash codes, BSMs and LSMs for all images in the database is still $O(Tnbm)$ only.

The time complexity of the RBFNN training method presented in Algorithm 1 is $O(tMNmb + M^3 + MN)$, where t and N denote the number of clustering iterations of the k-means algorithm and the number of training samples, respectively. In our experiments, 500 artificial queries are made to generate the training sample set for the RBFNN training and $N \ll T$. Both t and M are small values comparing to T and the time complexity of RBFNN training is independent to the number of images in the database (T). So, the training time of RBFNN is not very long in comparing to the computation of all hash codes for images in the database.

Overall, the time complexity of the SIF for online processing during queries is fast and similar to existing candidate image filtering methods while the time complexity of SIF for offline training of RBFNN is scalable. As aforementioned, the computations of both the LSM and the BSM only take a few extra times to the computation of hash codes for all images.

4 Experiments

In this section, experimental comparisons are carried out to compare precision and recall rates of the proposed SIF with three image filtering methods: the BLSH, the Intersection and the Hash Code Difference (HCD) for multi-hashing. Image filtering methods are tested on five multi-hashing methods using four large scale databases: the

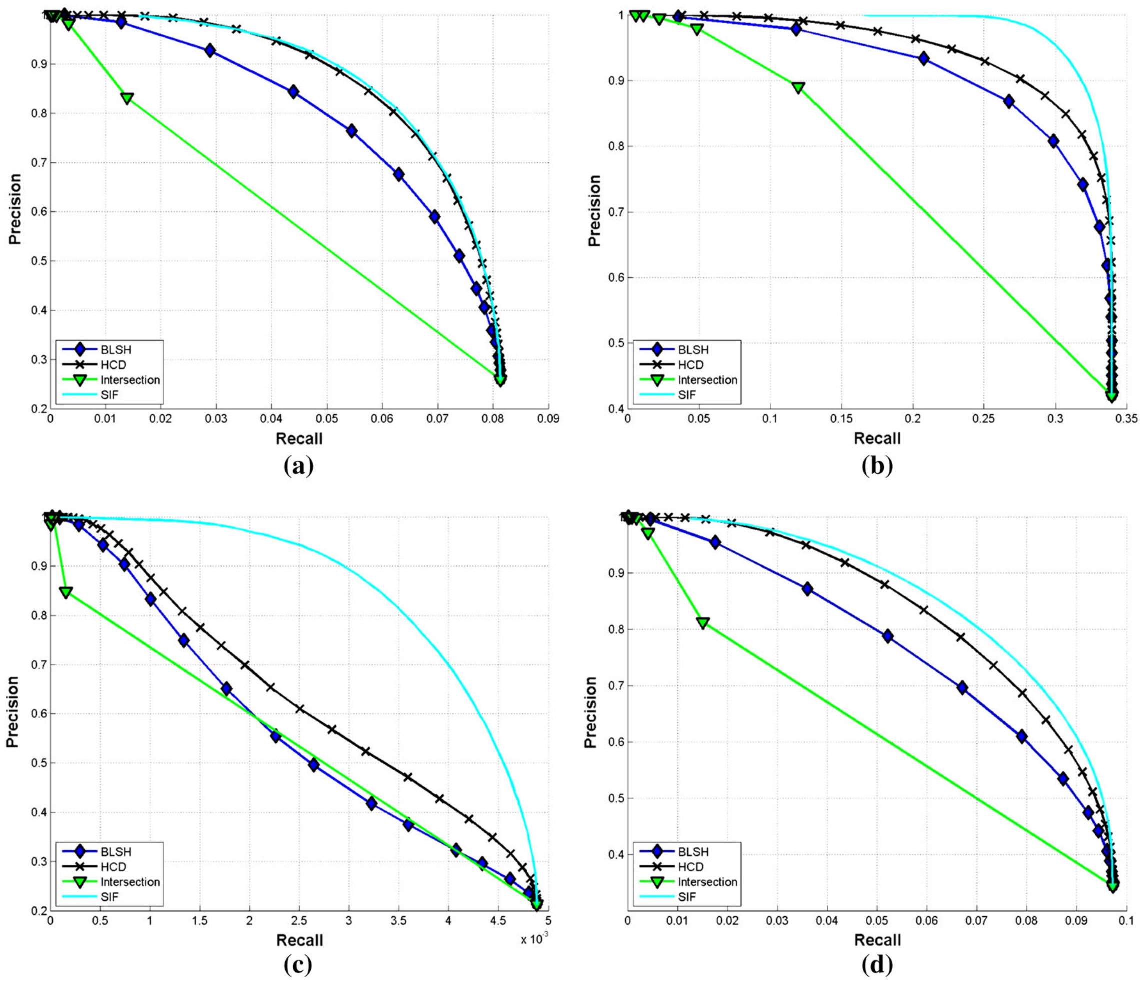

Fig. 2 Result of 32-table LSH_{multi} for CIFAR10 (**a**), MNIST (**b**), Nuswide (**c**) and SIFT1M (**d**)

MNIST (70K images) [31], the CIFAR-10 (60K images) [32], the Nuswide (around 270K images) [33] and the SIFT1M (1.02 million images) [2]. The five multi-hash methods are data-independent multi-table LSH (LSH_{multi}), data-dependent unsupervised multi-table ITQ (ITQ_{multi}), boosting based multi-table data-dependent semi-supervised hashing ($SPLH_{multi}$), a multi-table hashing combining all these three hashing methods ($Combined_{multli}$) and the DCH [26]. The LSH, the ITQ and the SPLH cover three major categories of current hashing methods while the combined multi-hashing is a new attempt to test the adaptability of the SIF.

Fifteen hundred images are randomly selected and removed from each database. They are randomly divided into 1000 query images and 500 training query images. Experimental results presented in the next section are average values over 1000 queries which are performed using the 1000 query images. For the RBFNN training, the 500 training query images are used to perform artificial queries to get *I*s for manual labeling of similarities. Pairs of *I*s and corresponding similarity labels form the training dataset for the RBFNN in the SIF. The number of training samples being used for each database depends on the number of *I*s returned.

BLSH: The BayesLSH lite algorithm is adopted for the BLSH and we select the set of parameters yielding the largest area under the recall-precision curve following setup in [7].

Interaction: *I*s are filtered out in an increasing order of the number of $h = 0$ buckets containing them among all m hash tables. Different precision and recall rates can be computed for different numbers of $h = 0$ buckets returning the *I*. *I*s having the number of $h = 0$ buckets equal to $0(m)$ will be the first (last) batch of images being filtered out.

HCD: *I*s are filtered according to the total Hamming distance over all hash tables between hash codes of *I*s and the E in a decreasing order. *I*s with the total Hamming distance equal to $0(mb)$ will be the last (first) batch of images being filtered out.

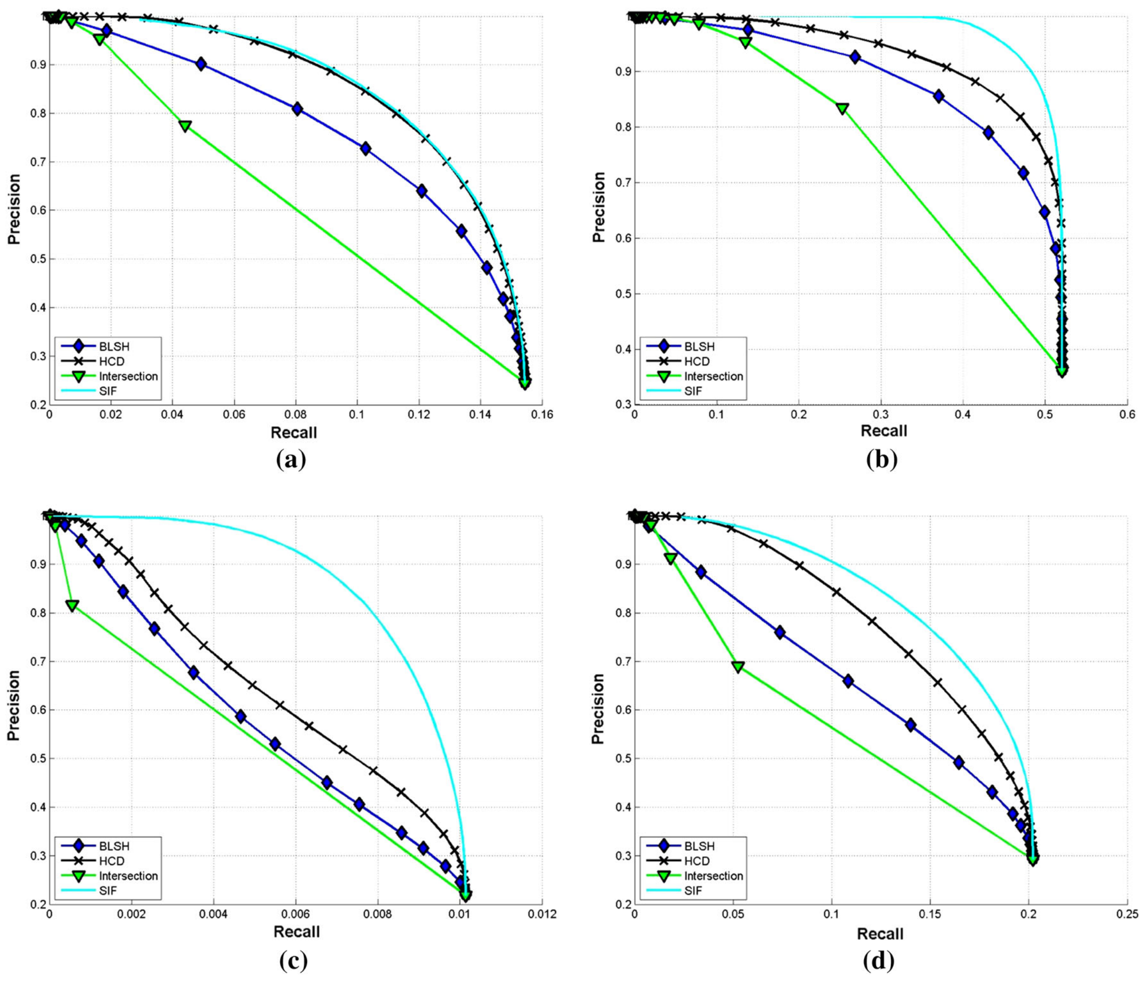

Fig. 3 Result of 64-table LSH_{multi} for CIFAR10 (**a**), MNIST (**b**), Nuswide (**c**) and SIFT1M (**d**)

SIF: *I*s are filtered by the output of the RBFNN of the SIF method and larger RBFNN outputs indicates more similar to the *E*. So, the SIF filter out *I*s in an increasing order of the output of the RBFNN.

All multi-hashing methods only return images from hash bucket which exactly match the hash code of the *E*, i.e. hash lookup. For the CIFAR-10, the MNIST and the SIFT1M databases, each individual table of multi-hashing uses 16 hash bits. 64-bit individual hash table is used for the SPLH hashing method for the Nuswide database owing to the poor performance of 16-bit tables. The LSH_{multi} uses 32 and 64 LSH tables as in [10]. The ITQ_{multi} is created by randomly selecting *K* principal components instead of the largest K. We extended the original SPLH to a multi-hash using a boosting method for the $SPLH_{multi}$. 10, 20 and 30 tables are compared for both the ITQ_{multi} and the $SPLH_{multi}$. The $Combined_{multi}$ takes 10 tables from the LSH, the ITQ and the SPLH, respectively. The $Combined_{multi}$ is intended to demonstrate the applicability of the SIF instead of proposing a new, powerful multi-hashing method. Owing to the boosting nature of the DCH, it cannot create a large number of tables. Moreover, we find that when the DCH tries to build the 6_{th} hash table, it is very similar to the 5_{th} hash table because the weight matrix of boosting is almost filled with zeros. In the DCH scheme, a zero is assigned to the weight matrix when the current hash table "correctly classifies" the pairwise similarity of images. So the weight matrix becomes a zero matrix quickly. Therefore, in our experiments, the DCH uses 3, 5 and 10 tables with 16 bits.

Both Euclidean and semantic ground truths are applicable for experiments of the CIFAR-10, the MNIST and the Nuswide while the SIFT1M uses Euclidean ground truth only. When the semantic ground truth is applied, *I*s sharing the same class label with the *E* are regarded as similar. For the Euclidean ground truth, a nominal threshold of the average distance of the 50 nearest neighbors of all images in the database is used [14] for the CIFAR-10, the MNIST and the Nuswide databases. *I*s with Euclidean distances less than the threshold from the *E* are regarded as similar images to the *E*. Other images are dissimilar. For

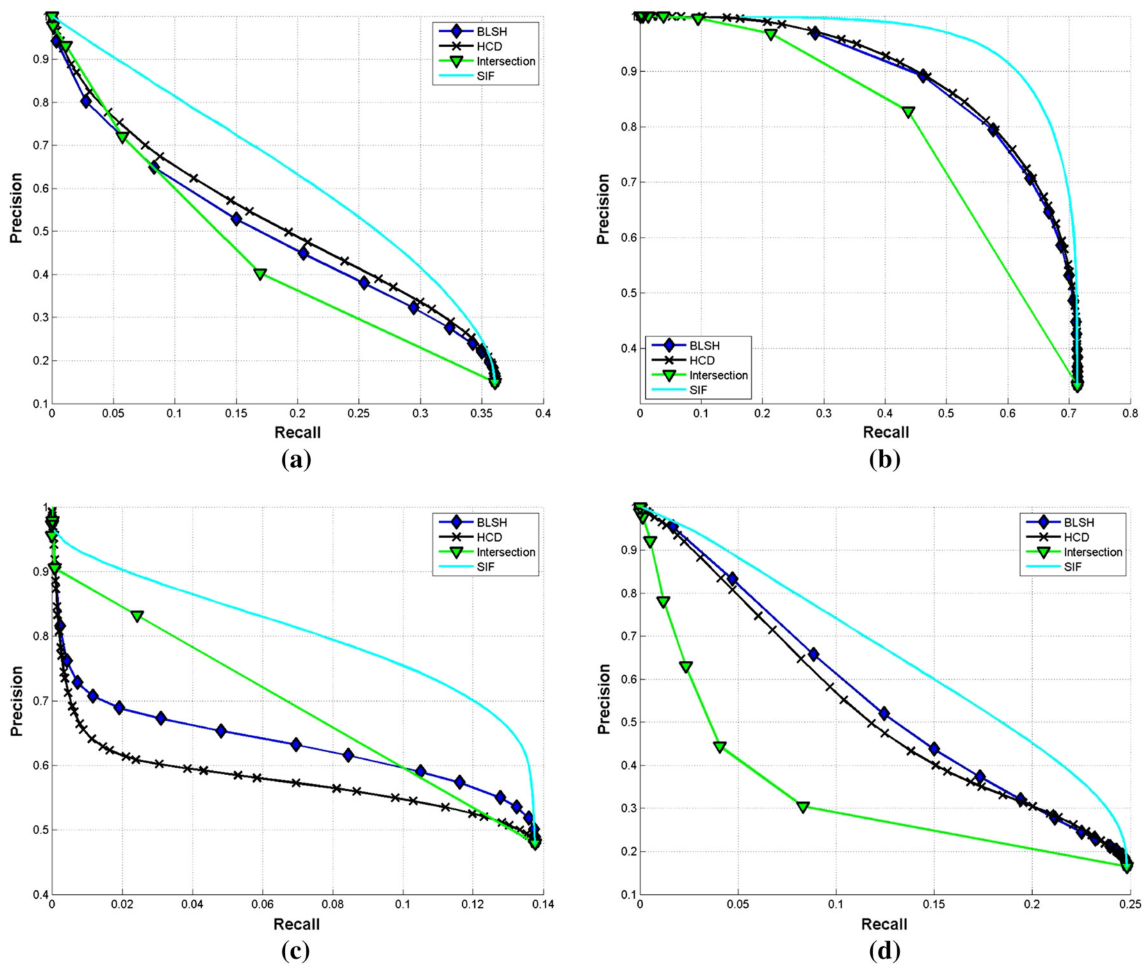

Fig. 4 Result of 10-table ITQ_{multi} for CIFAR10 (**a**), MNIST (**b**), Nuswide (**c**) and SIFT1M (**d**)

the SIFT1M, 20,000 nearest neighbors of an image are regarded as similar images [2] with the Euclidean ground truth.

4.1 Experimental results

We present experimental results of the LSH_{multi}, the ITQ_{multi} and the $SPLH_{multi}$ methods with different number of hash tables in Figs. 2, 3, 4, 5, 6, 7, 8 and 9, respectively. Experiments of the LSH_{multi} and the ITQ_{multi} use the Euclidean ground truth while experiments of the $SPLH_{multi}$ use the semantic ground truth. Experimental results of the $Combine_{multi}$ for both the Euclidean and the semantic ground truths are shown in Figs. 10 and 11, respectively. Figures 12, 13 and 14 show experimental results of multi-hashing created by the DCH with the semantic ground truths. Table 1 shows the number of *I*s being filtered out by different methods (the best results are bolded).

In summary, the proposed SIF outperforms all other filtering methods using different number of hash tables for all databases in experiments with different number of images. In experiments of the Nuswide database, the proposed method yields very high rates of both precision and recall using both the LSH_{multi} and the ITQ_{multi} while other image filtering methods using the ITQ_{multi} drop quickly. By comparing the precision-recall curves of all experiments, the proposed method outperforms both the HCD and the BLSH. Given the fact that both the HCD and the BLSH are essentially equivalent to using *h* only for image filtering, this shows that both the BSM and the LSM improve the approximation on similarity of *I*s significantly.

For the CIFAR-10 database, the HCD performs similarly well in comparison to the SIF using LSH_{multi}. By a PCA on samples in the CIFAR-10 database, we find that their principle components (variances) are either nearly zero or with similar values. This shows that samples are distributed almost as a hypersphere, i.e. no single dimension has significantly larger variance than others. For LSH based multi-hashing which generates hash functions and tables randomly and independently, both the HCD and the

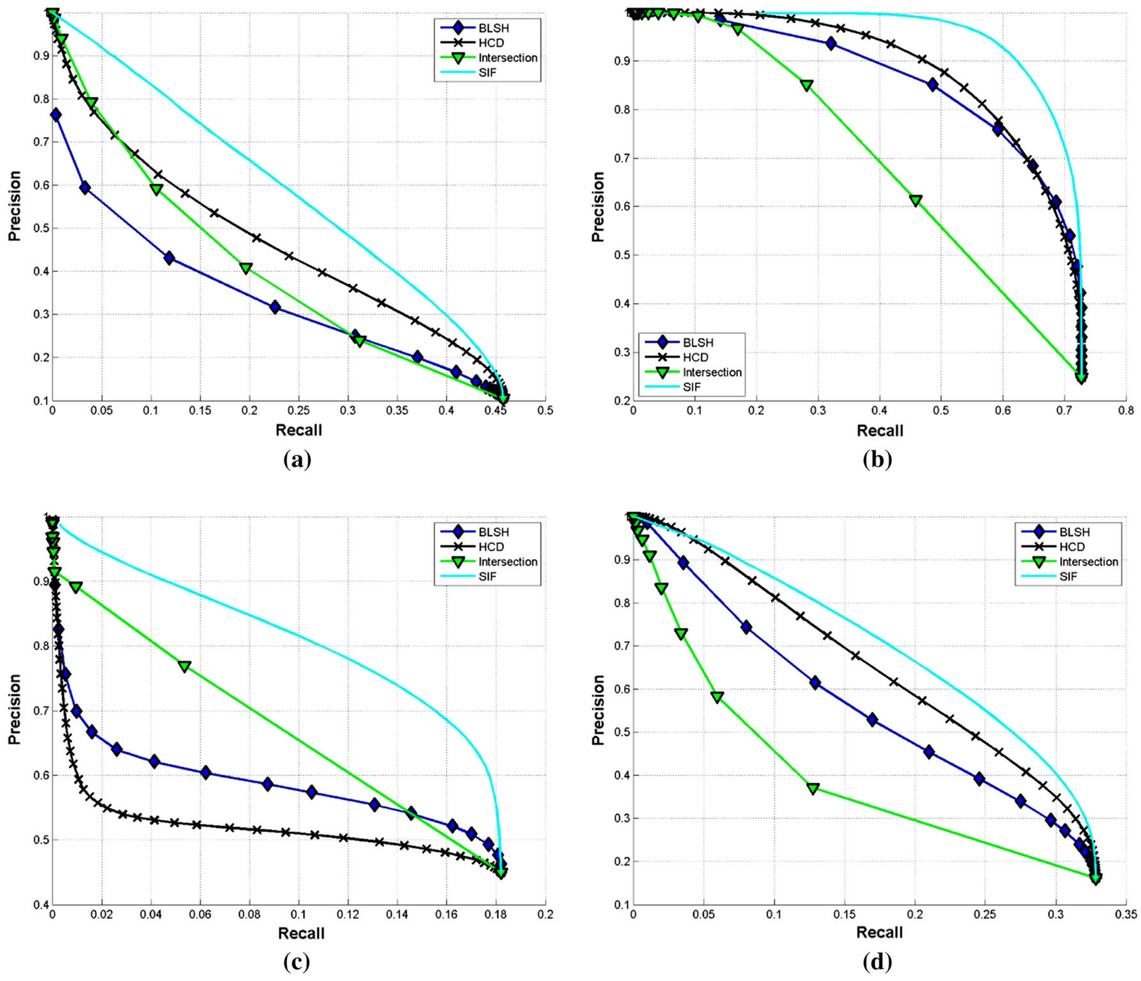

Fig. 5 Result of 20-table ITQ_{multi} for CIFAR10 (**a**), MNIST (**b**), Nuswide (**c**) and SIFT1M (**d**)

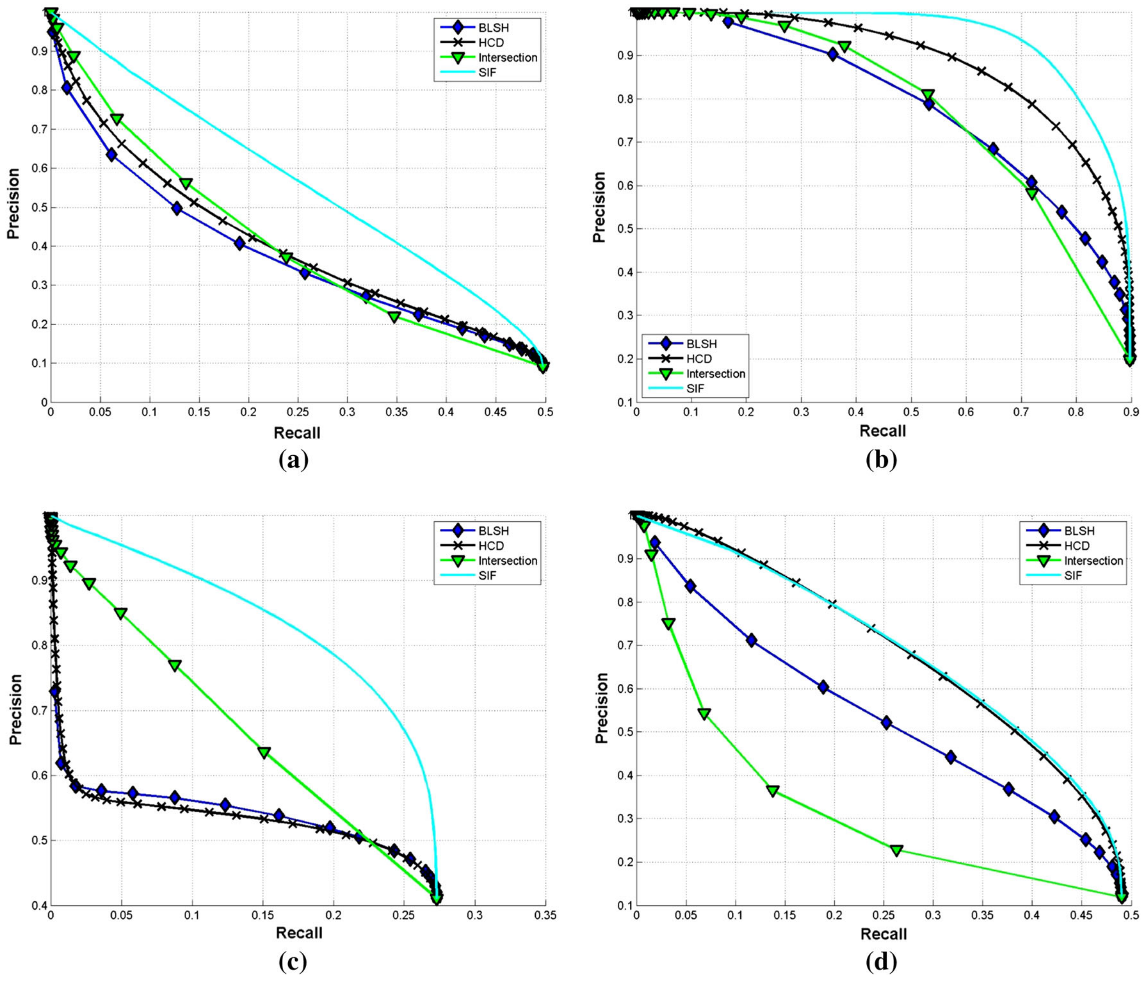

Fig. 6 Result of 30-table ITQ_{multi} for CIFAR10 (**a**), MNIST (**b**), Nuswide (**c**) and SIFT1M (**d**)

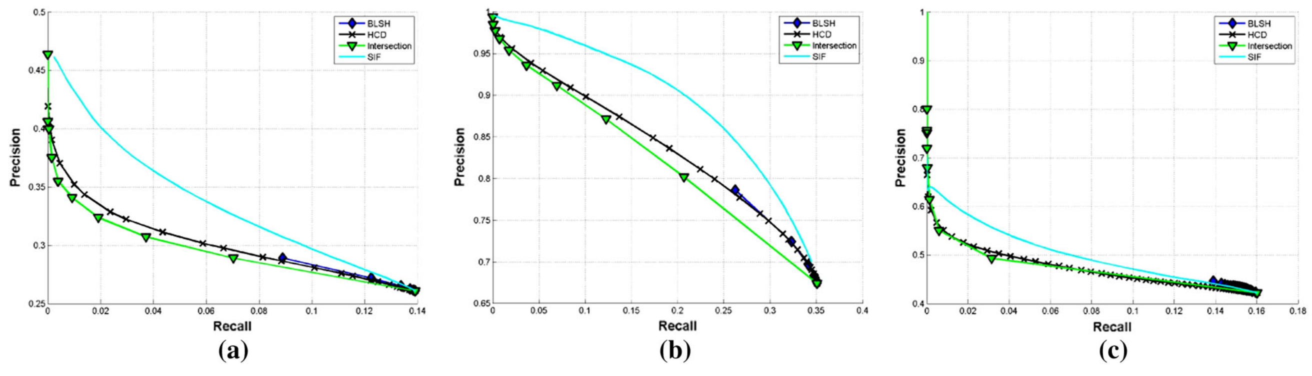

Fig. 7 Result of 10-table $SPLH_{multi}$ for CIFAR10 (**a**), MNIST (**b**), Nuswide (**c**)

BLSH provide a good estimate to the maximum likelihood of similarity between *I*s and the *E* [7]. Hence the SIF method does not have an obvious advantage over the HCD and the BLSH. In this case, the HCD outperforms the BLSH owing to the use of all bits of hash codes in the computation of the similarity. The Intersection method yields very high precision but very low recall rate for all databases.

In experiments of the $SPLH_{multi}$, when the number of hash tables increases, the BLSH outperforms the HCD in our experiments in the region of high recall rates. However, the curves of the BLSH stop early in low recall regions in

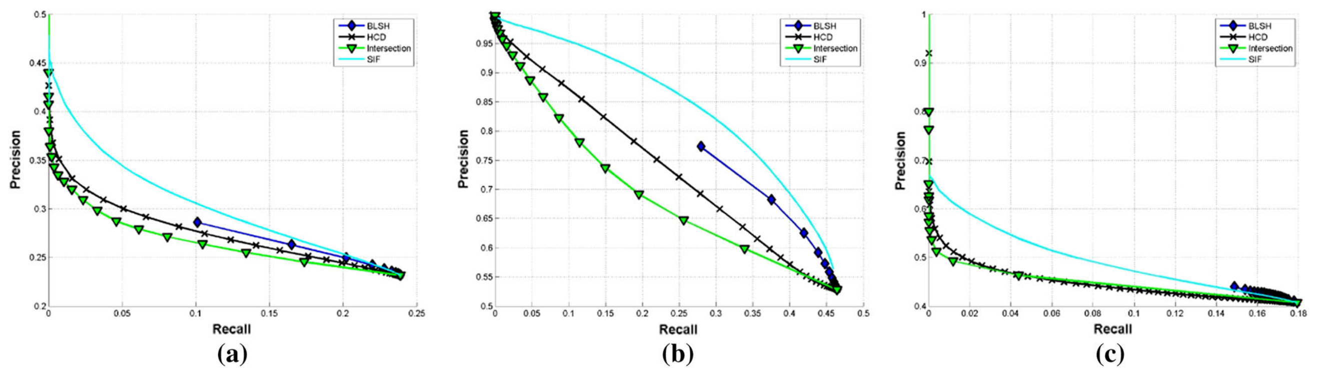

Fig. 8 Result of 20-table $SPLH_{multi}$ for CIFAR10 (**a**), MNIST (**b**), Nuswide (**c**)

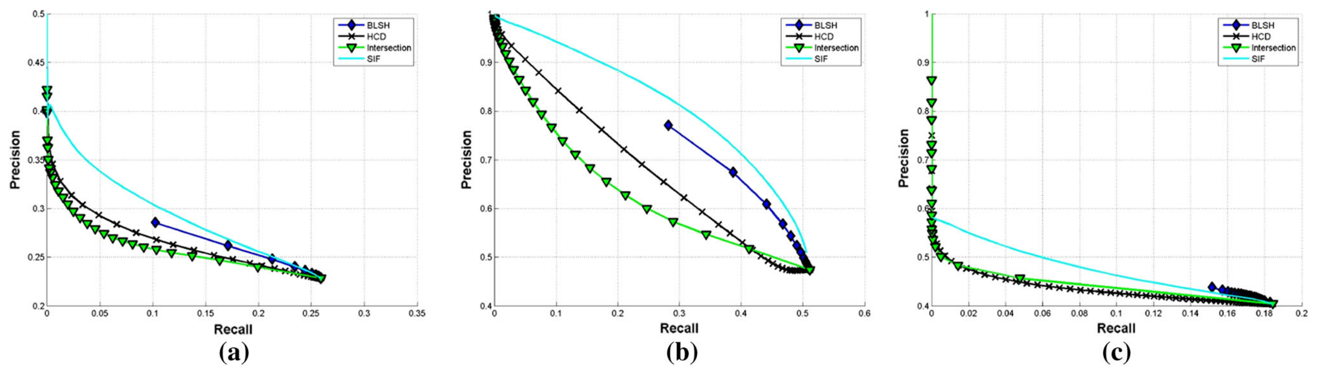

Fig. 9 Result of 30-table $SPLH_{multi}$ for (**a**), MNIST (**b**), Nuswide (**c**)

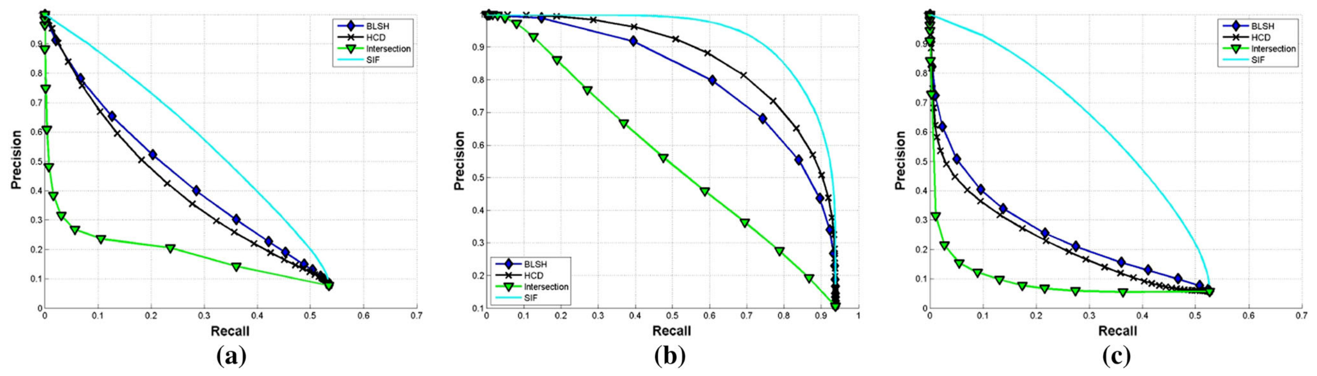

Fig. 10 Result of $Combine_{multi}$ with Euclidean Ground Truth for (**a**), MNIST (**b**), Nuswide (**c**)

Figs. 7, 8 and 9 because the BLSH can not recognize images which are semantically dissimilar.

The proposed method yields very high precision and recall rates on all databases when the LSH_{multi} and the Euclidean ground truth are used. Moreover, good performances in the $SPLH_{multi}$ and the $Combine_{multi}$ show that the proposed SIF also provides a good filtering to images with semantic ground truth.

Figures 10 and 11 show that the proposed SIF method outperforms other image filtering methods using multi-hashing combining different types of hashing methods, especially for the Nuswide database using the Euclidean ground truth. Overall, image retrieval problems with the Euclidean ground truth are easier to achieve high precision and recall rates with different hashing methods. For the semantic ground truth, the $Combine_{multi}$ (Fig. 11) outperforms the $SPLH_{multi}$ (Figs. 7, 8, 9) and yields larger area under curves in the precision-recall curves in Fig. 11. It shows that the SPLH still has room for improvement. The ITQ provides the structural information of the database

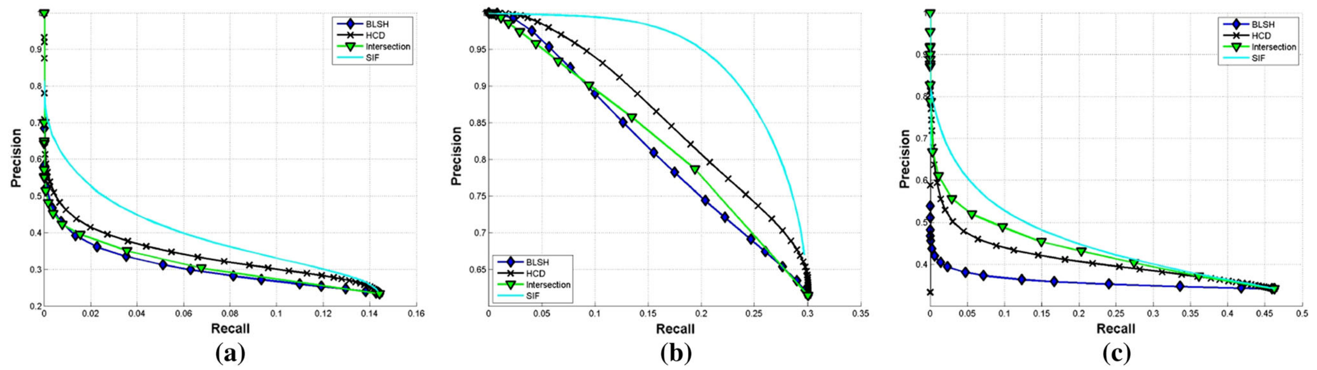

Fig. 11 Result of *Combine*$_{multi}$ with Semantic Ground Truth for (**a**), MNIST (**b**), Nuswide (**c**)

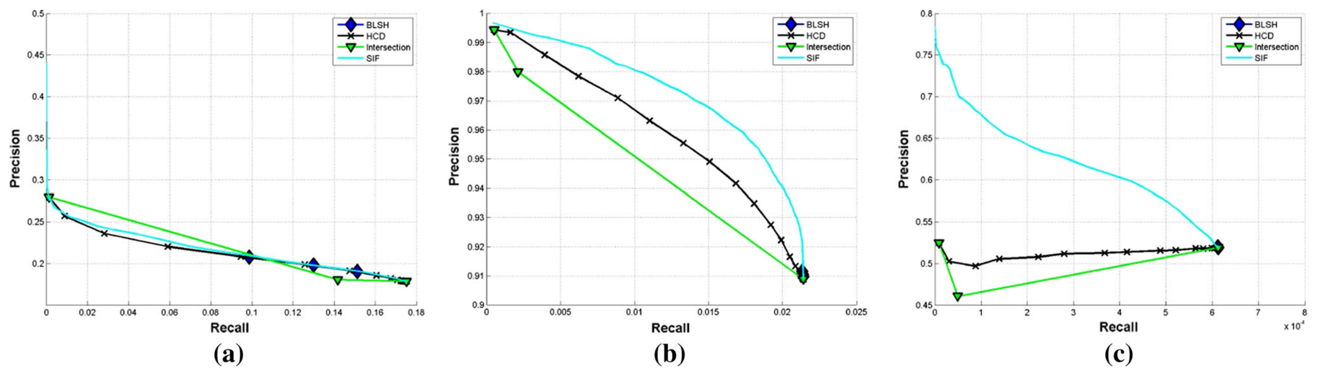

Fig. 12 Result of 3-table DCH for (**a**), MNIST (**b**), Nuswide (**c**)

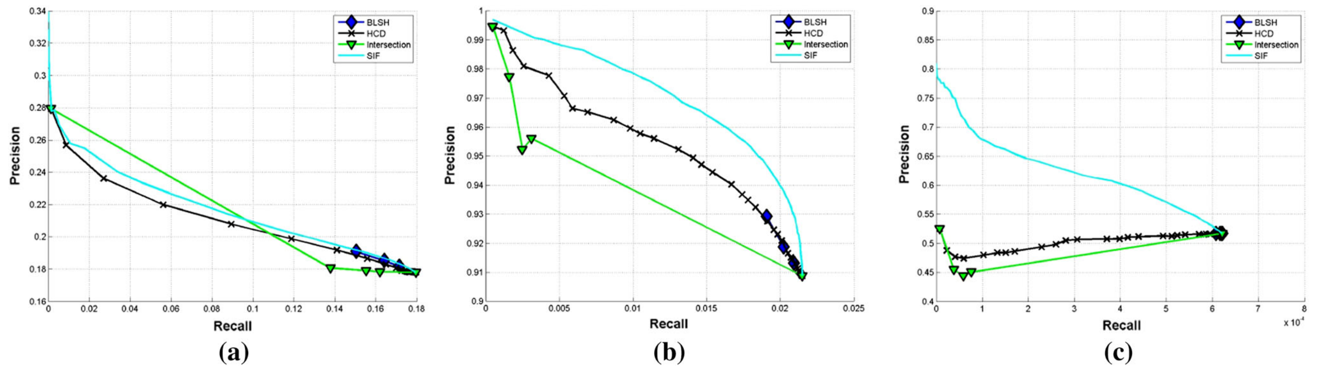

Fig. 13 Result of 5-table DCH for (**a**), MNIST (**b**), Nuswide (**c**)

while the LSH preserves the local similarity among images. Both of them provide complementary information about unlabeled images which is only used to balance the partition in the SPLH only. Hence, a better way to use information carried by unlabeled images is one of the important future works of us for the semi-supervised hashing.

Figures 12, 13 and 14 show that the DCH does not yield a good multi-hashing using the union of exact match. Nonetheless, among these poor results, the SIF still yields the best image filtering performance according to the precision-recall curves. In Figs. 12a, 13a and 14a, the Intersection method does not outperform the SIF, but the curve fitting between two sparse nodes of the precision-recall curve of the Intersection creates a misleading visual effect. None of the points (triangles) on the precision-recall curve of the Intersection is located higher than that of the SIF's with the same recall rate. For experiments of the MNIST and the Nuswide databases, the SIF yields a very high improvement to the precision and recall rates. The DCH creates each hash bit (table) by compensating error

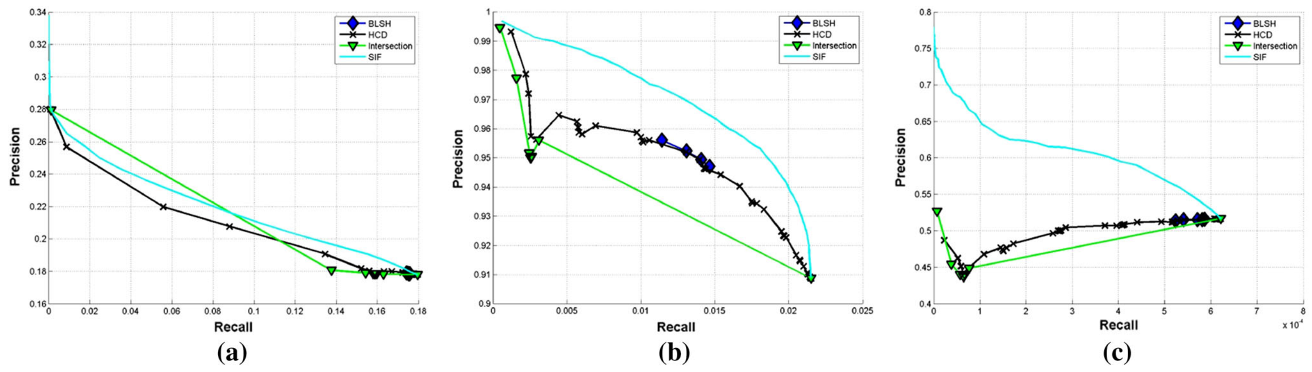

Fig. 14 Result of 10-table DCH for (**a**), MNIST (**b**), Nuswide (**c**)

Table 1 Percentages of *I*s being filtered by different methods for MNIST database by scarifying 2 % of recall rates

Multi-hash methods		LSH 32 tables	LSH 64 tables	ITQ 10 tables	ITQ 20 tables	ITQ 30 tables	SPLH 10 tables	SPLH 20 tables	SPLH 30 Tables	Combined Euclidean ground truth	Combine semantic ground truth
Recall of union		33.95%	52.03 %	71.33 %	72.75 %	89.76 %	35.09 %	46.42 %	51.13 %	93.83 %	30.06 %
Precision of union		41.95 %	36.08 %	33.23 %	24.76 %	19.81 %	67.40 %	52.80 %	47.41 %	18.05 %	61.43 %
Recall after filtering		31.95 %	50.03 %	69.33 %	70.75 %	87.76 %	33.09 %	44.42 %	49.13 %	91.83 %	28.06 %
Average # of candidate images in union		306.8	546.76	813.87	1114.10	1718.03	3596.53	6073.96	7450.39	3380.90	185.55
% of removed images from union	BLSH	46.70 %	45.83 %	42.21 %	55.61 %	45.26 %	10.80 %	12.66 %	12.91 %	71.15 %	11.50 %
	HCD	51.48 %	53.71 %	43.66 %	52.24 %	61.21 %	10.87 %	3.62 %	5.93 %	76.67 %	17.60 %
	Intersection	14.61 %	12.44 %	12.29 %	12.39 %	19.79 %	8.12 %	5.67 %	6.30 %	20.72 %	11.33 %
	SIF	**55.94 %**	**59.07 %**	**54.91 %**	**65.13 %**	**67.96 %**	**13.40 %**	**18.75 %**	**17.18 %**	**82.23 %**	**26.14 %**
Improved precision rate	BLSH	29.05 %	25.89 %	21.06 %	25.16 %	14.50 %	3.86 %	4.75 %	5.22 %	24.42 %	3.36 %
	HCD	39.41 %	38.87 %	24.10 %	25.65 %	30.12 %	3.91 %	0.15 %	0.91 %	33.62 %	8.16 %
	Intersection	4.29 %	3.54 %	3.59 %	2.72 %	4.34 %	1.78 %	0.88 %	1.12 %	2.47 %	3.24 %
	SIF	**47.66 %**	**48.69 %**	**38.39 %**	**44.29 %**	**40.65 %**	**5.99 %**	**8.66 %**	**8.21 %**	**39.91 %**	**16.21 %**

made by the previous bit (table), so they are highly correlated. In this case, the BLSH performs very poorly. For experiments with the Nuswide database, the HCD, the Intersection and the BLSH yield worse precision by filtering with scarification of recall rates. The SIF is the only method in our experiments yielding better precision after the image filtering.

Multi-hashing using the SIF provides a fast retrieval response with high precision and recall rates for large scale image retrieval problems while single hash table based methods may take a long time to gather the required number of similar images with large h between the E and Is. The improvement in precision rate made by the SIF is vital to both web based and mobile based large scale retrieval problems because the bandwidth is both limited and expensive.

After filtering, a reduction in recall rate indicates Is similar to the E being removed incorrectly while an increase in precision indicates a removal of dissimilar Is correctly. In the perfect case, an image filtering method should yield a 100 % precision rate without any drop in recall rate after the candidate image filtering. However, in real situation, we can only settle for a trade off between the precision rate and the recall rate.

Table 1 shows the image filtering by different filtering methods for the MNIST database on the Union of Is from different multi-hashing methods and different number of hash tables. Owing to the poor performance of the DCH, experiments of the DCH are not compared in Table 1. The second and third rows show the precision and recall rates, respectively, yielded by the Union of Is returned by different multi-hash methods. They serve as the baselines of comparison and are the performance of multi-hashing methods without image filtering. Then, we perform the filtering and stop when there is a 2 % drop in the recall rate. The percentage of images being filtered out and the improvement of precision rates by different methods are shown in rows 6–9 and rows 10–13, respectively. The SIF removes the largest number of images from the Union of *I*s from all hash tables. The SIF outperforms other methods significantly and yields an average of 10.05, 8.70 and 30.59 % more reductions in comparison to the BLSH, the HCD and the Intersection, respectively. The SIF removes at least 55 % of images in experiments using the LSH_{multi} and the ITQ_{multi} methods. Improvement of precision rate made by the SIF is at least 39.91 % when the Euclidean ground truth is applied (columns 2–6 and 10). The retrieval problems of $SPLH_{multi}$ and the $Combined_{multi}$ with the semantic ground truth are more difficult than that with the Euclidean ground truth. Hence, precision and recall rates of experiments with the semantic ground truth are usually worse. Nonetheless, for experiments with the semantic ground truths, the SIF still reduces a large number of images from the Union of *I*s and yields the largest improvement in the precision rate. These show that the SIF is able to remove dissimilar images to greatly increase the precision rate with a small sacrifice of the recall rate.

5 Conclusion

In this work, we propose a RBFNN based SIF method for improving the precision rate of multi-hashing methods for the large scale image retrieval problem. Two new descriptors, i.e. the LSM and the BSM, are also proposed to serve as input features describing a candidate image for filtering together with the Hamming distance and the hash code difference. The Radial Basis Function Neural Network outputs scores for candidate images which can be used to both filtering and sorting of candidate images. Experimental results show that the proposed method is efficient and yields high precision and recall rates for different multi-hashing methods.

Currently, there is no guideline for selecting the number of hash tables for multi-hashing methods. Our experiments show that a multi-hashing with more hash tables yields a higher recall rate but also more dissimilar images owing to the expansion of size of the union of $h = 0$ buckets from all hash tables in the original descriptor space. The SIF is able to filter out dissimilar images, but it can not answer the question of how many hash tables is good enough for a given CBIR problem. In our future work, we target to propose a method to find the optimal number of hash tables for a given training set and both the LSM and the BSM to yield a balanced precision and recall rate.

Acknowledgments This work is supported by National Natural Science Foundation of China (61272201), a Program for New Century Excellent Talents in University (NCET-11-0162) of China, and the Fundamental Research Funds for the Central Universities (2015ZZ023). We would like to thank Dr. Jun Wang for providing the source code of SPLH and databases.

References

1. Kulis B, Grauman K (2012) Kernelized locality-sensitive hashing. IEEE Trans Pattern Anal Mach Intell 34(6):1092–1104
2. Wang J, Kumar S, Chung S-F (2012) Semi-supervised hashing for large-scale search. IEEE Trans Pattern Anal Mach Intell 34(12):2393–2406
3. Strecha C, Bronstein AM, Bronstein MM, Fua P (2012) LDAHash: improved matching with smaller descriptors. IEEE Trans Pattern Anal Mach Intell 34(1):66–78
4. Chechik G, Sharma V, Shalit U, Bengio S (2010) Large scale online learning of image similarity through ranking. J Mach Learn Res 11:1109–1135
5. Silpa-Anan C, Hartley R (2008) Optimised KD-tree for fast image descriptor matching. In: Proceedings of IEEE conference on computer vision and pattern recognition, pp 1–8
6. Weber R, Schek HJ, Blott S (1998) A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: Proceedings of the 24th international conference on very large data bases, pp 194–205
7. Satuluri V, Parthasarathy S (2012) Bayesian locality sensitive hashing for fast similarity search. In: Proceedings of the very large database endowment, pp 430–441
8. Yeung DS, Ng WWY, Wang D, Tsang ECC, Wang X-Z (2007) Localized generalization error and its application to architecture selection for radial basis function neural network. IEEE Trans Neural Netw 18(5):1294–1305
9. Indyk P, Motwani R (1998) Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the 30th ACM symposium on theory of computing, pp 604–613
10. Charikar MS (2002) Similarity estimation techniques from rounding algorithms. In: Proceedings of ACM symposium on the theory of computing, pp 380–388
11. Bian W, Tao D (2010) Biased discriminant euclidean embedding for content-based image retrieval. IEEE Transa Image Process 19(2):545–554
12. Gorisse D, Cord M, Precioso F (2012) Locality-sensitive hashing for Chi2 distance. IEEE Trans Pattern Anal Mach Intell 34(2):402–409
13. Kulis B, Jain P, Grauman K (2012) Fast similarity search for learned metrics. IEEE Trans Pattern Anal Mach Intell 31(12):2143–2157
14. Gong Y, Lazebnik S, Gordo A, Perronnin F (2013) Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval. IEEE Trans Pattern Anal Mach Intell 35(12):2916–2929
15. Ng WWY, Lv Y, Yeung DS, Chan PPK (2015) Two-phase mapping hashing. Neurocomputing 151:1423–1429
16. Lv Y, Ng WWY, Zeng Z, Yeung DS, Chan PPK (2015) Asymmetric cyclical hashing for large scale image retrieval. IEEE Trans Multimed 17(8):1225–1235
17. Matsushita Y, Wada T (2009) Principal component hashing: an accelerated approximate nearest neighbor search. Adv Image Video Technol LNCS 5414:374–385
18. Wu C, Zhu J, Cai D, Chen C, Bu J (2013) Semi-supervised nonlinear hashing using bootstrap sequential projection learning. IEEE Trans Knowl Data Eng 25(6):1380–1393
19. Shao J, Wu F, Ouyang C, Zhang X (2012) Sparse spectral hashing. Pattern Recognit Lett 33(3):271–277

20. Li P, Wang M, Cheng J, Xu C, Lu H (2013) Spectral hashing with semantically consistent graph for image indexing. IEEE Trans Multimed 15(1):141–152
21. Weiss Y, Torralba AB, Fergus R (2008) Spectral hashing. In: Proceedings conference on advances in neural information processing systems, pp 1753–1760
22. Norouzi M, Fleet DJ (2011) Minimal loss hashing for compact binary codes. In: Proceedings of international conference on machine learning, pp 353–360
23. Kulis B, Darrell T (2009) Learning to hash with binary reconstructive embeddings. Adv Neural Inf Process Syst 22:1042–1050
24. Salakhutdinov R, Hinton GE (2009) Semantic hashing. Int J Approx Reason 50(7):969–978
25. Xu H, Wang J, Li Z, Zeng G, Li S, Yu N (2011) Complementary hashing for approximate nearest neighbor search. In: Proceedings of IEEE international conference on computer vision, pp 1631–1638
26. Li P, Cheng J, Lu H (2013) Hashing with dual complementary projection learning for fast image retrieval. Neurocomputing 120:83–89
27. Wang X, Qiu S, Liu K, Tang X (2014) Web image re-ranking using query-specific semantic signatures. IEEE Trans Pattern Anal Mach Intell 36:810–823
28. Jiang Y-G, Wang J, Xue X, Chang S-F (2013) Query-adaptive image search with hash codes. IEEE Trans Multimed 15(2):442–453
29. Norouzi M, Punjani A, Fleet DJ (2012) Fast search in Hamming space with multi-index hashing. In: Proceedings of IEEE conference on computer vision and pattern recognition, pp 3108–3115
30. Ng WWY, Yeung DS, Firth M, Tsang ECC, Wang X-Z (2008) Feature selection using localized generalization error for supervised classification problems using RBFNN. Pattern Recognit 41(12):3706–3719
31. LeCun Y, Bottou L, Bengio, Y, Haffner P (1998) Gradient-based learning applied to document recognition. In: Proceedings of the IEEE, pp 2278–2324
32. Krizhevsky A (2009) Learning multiple layers of features from tiny images. Master Thesis, University of Toronto, Toronto. http://www.cs.toronto.edu/ kriz/cifar.html
33. Chua T-S, Tang J, Hong R, Li H, Luo Z, Zheng Y (2009) NUS-WIDE: a real-world web image database from National University of Singapore. In: Proceedings of the ACM international conference on image and video retrieval (Article no. 48)