Unsupervised Local Feature Hashing for Image Similarity Search

Li Liu, Mengyang Yu, Student Member, IEEE, and Ling Shao, Senior Member, IEEE

Abstract—The potential value of hashing techniques has led to it becoming one of the most active research areas in computer vision and multimedia. However, most existing hashing methods for image search and retrieval are based on global feature representations, which are susceptible to image variations such as viewpoint changes and background cluttering. Traditional global representations gather local features directly to output a single vector without the analysis of the intrinsic geometric property of local features. In this paper, we propose a novel unsupervised hashing method called unsupervised bilinear local hashing (UBLH) for projecting local feature descriptors from a high-dimensional feature space to a lower-dimensional Hamming space via compact bilinear projections rather than a single large projection matrix. UBLH takes the matrix expression of local features as input and preserves the feature-to-feature and image-to-image structures of local features simultaneously. Experimental results on challenging data sets including Caltech-256, SUN397, and Flickr 1M demonstrate the superiority of UBLH compared with state-of-the-art hashing methods.

Index Terms—Hashing, image similarity search, local feature, unsupervised learning.

I. INTRODUCTION

LEARNING to hash has received substantial attention due to its potential in various applications such as data mining, pattern recognition, and information retrieval [1]–[7]. Compact hashing enables significant efficiency gains in both storage and retrieval speed for large-scale databases. Generally speaking, greedy-searching-based retrieval on a data set with \( N \) samples is infeasible because linear complexity \( O(N) \) is not scalable to realistic applications on large-scale data. Meanwhile, most vision tasks also suffer from the curse of dimensionality, because visual descriptors usually have hundreds or even thousands of dimensions. Due to above reasons, hashing techniques are proposed to effectively embed data from a high-dimensional feature space into a similarity-preserved low-dimensional Hamming space where an approximate nearest neighbor (ANN) of a given query can be found with sublinear time complexity.

Currently, both conventional unsupervised and supervised hashing algorithms are primarily designed for global representations, e.g., GIST [8]. For realistic visual retrieval tasks, however, these global hashing techniques cannot cope with different complications appearing in the images such as cluttering, scaling, occlusion, and change of lighting conditions. However, these aspects are more invariant in local feature-based representations such as bag-of-features [9], [10], since such representations are statistical distributions of image patches and tend to be more robust in challenging and noisy scenarios. Fig. 1 illustrates the comparison of a global representation-based hashing method (GIST+ITQ) and our proposed local feature-based hashing method [SIFT+UBLH] for relatively complex scene retrieval. We can observe that the top four retrieved images via GIST+ITQ only contain the desert. They are indeed relevant to the query image, but not exactly what we want to search. While, the top four retrieved images via our approach, i.e., SIFT+UBLH, include all the detailed information, i.e., man, camel, and desert, as in the query image. The possible reason of the difference on the above retrieval task is that global representation-based hashing methods are...
good at extracting the global intensity, color, texture, and gradient information of images, but will ignore the detailed information in the query image without analyzing the intrinsic geometric property of local features. This problem may heavily limit the effectiveness on applications that demand more accurate retrieval results for complex scene/object images. Thus, inspired by advantages (e.g., invariance for cluttering, scaling, and occlusion) of local representations, in this paper, we intend to develop a local feature-based hashing method for improving the retrieval results. If keypoints are well detected, local hash codes are able to avoid the limitations such as background variations, occlusions, and shifts in global representations.

In this paper, we propose an UBLH framework for large-scale visual similarity search, in which the feature-to-feature (F2F) and image-to-image (I2I) structures are successfully combined and preserved together. Specifically, the F2F structure considers the pairwise relationship between local features in the original feature space, which is always considered in manifold learning techniques [11]–[16]. From a higher-level aspect, I2I structure reflects the connection between images when each of them is represented by a set of local features. In particular, the I2I distance can provide a feasible way to measure the connection between two images, which is derived from [17]. It measures the distance between two images using the set representation of local features of the image. Since the raised problem of UBLH is regarded as nonconvex and discrete, our objective function is then optimized via an alternate way with relaxation.

Furthermore, motivated by [18]–[20], a bilinear projection is employed to make the algorithm more efficient. To be specific, the bilinear projection applies two projection matrices to local features, which have much smaller sizes than the original single projection matrix. Since the computational complexity of eigen decomposition is cubic degree on the dimension of the matrix, the effect of applying smaller matrices is quite conspicuous. Beyond that, most local features are based on histograms and bins, e.g., an SIFT feature is computed from 16 histograms, each of which has eight bins. Therefore, the bilinear scheme can explore two different kinds of data structure from the views of histograms and bins simultaneously. More crucially, when local features are transformed from the vector form to the matrix form, a factorization of integral for dimensionality is needed. These two different views provide a natural factorization.

The outline of our proposed UBLH is demonstrated in Fig. 2. Considering that our method is specifically designed for local feature-based hashing, the original Hamming ranking and Hamming table cannot be directly applied on local features for visual indexing. Thus, in this paper, we also introduce an image indexing/searching scheme called local hashing voting (LHV), which has been demonstrated to be efficient and accurate for image similarity search in our experiments.

This paper aims at unsupervised linear (bilinear) hashing for local features, which makes UBLH effective and practical for real-world applications without class label information. With the bilinear projection learning, the complexity of the eigen decomposition, which is the cubic form of the dimensionality, will be significantly reduced. Once the projections are learned, they can be efficiently used on the test data. Additionally, UBLH simultaneously preserves the F2F and I2I structures which can be regarded as the local and global structures respectively in the original feature space.

II. RELATED WORK

In terms of bilinear hashing, Gong et al. [18] applied the bilinear scheme to the global representations by minimizing the angles between the rotated features and the corresponding binary codes. Although this scheme can effectively solve the high-dimensional hashing problem with less computational complexity, it still has some deficiency in the optimization process. Specifically, the angle between the vector in a continuous
space and that in a discrete space would bring quantization errors in the optimization. Moreover, their scheme lacks considering the relationship among features.

To explore hashing in the early time, random projections are always used to construct randomized hash functions. A most well-known representative is locality-sensitive hashing (LSH) [21], [22], which can preserve similarity information and map data points close in a Euclidean space to similar codes. It is theoretically guaranteed that as the code length increases, the Hamming distance between two codes will asymptotically approach the Euclidean distance between their corresponding data points. Furthermore, a kernel trick, which allows the use of a wide class of similarity functions, was combined with LSH to generalize LSH with arbitrary kernel functions [23]. Beyond that, principled linear projections such as PCA hashing (PCAH) [24] and its rotational variant have been introduced for better quantization rather than random projection hashing. Spectral hashing (SpH) [25] was proposed to preserve the data locality relationship to keep neighbors in the input space as neighbors in the Hamming space. Anchor graphs hashing (AGH) [26] is adopted to obtain tractable low-rank adjacency matrices for efficient similarity search. Kernel reconstructive hashing [27] was proposed to preserve the similarity defined by an arbitrary kernel using compact binary code. Compressed hashing (CH) [28] has been effectively applied for large-scale data retrieval tasks as well. All these hashing techniques mentioned above are regarded as unsupervised methods which may lead to worse retrieval precision for the data sets with noise. To achieve better results, researchers have developed supervised hashing methods which could attain higher search accuracy, since the label information is involved in the learning phase. A simple supervised hashing method is linear discriminant analysis hashing (LDAH) [29] which can tackle supervision via easy optimization but still lacks adequate performance due to the use of orthogonal projection in hash functions. Beyond that, some more complicated methods have been proposed such as binary reconstructive embeddings (BRE) [30], minimal loss hashing (MLH) [31], and kernel supervised hashing (KSH) [32]. Although these supervised methods can achieve promising results, they impose difficulties in the optimization. Moreover, their scheme lacks considering the relationship among features.

A. Notations and Problem Statement

We are given $N$ local features $x_{1}, \ldots, x_{N} \in \mathbb{R}^{D}$ from $n$ images. For image $i$, we use $X_{i} = \{x_{1}, \ldots, x_{m_{i}}\}$ to represent its local feature set. Bilinear projection is to multiply projection matrices on both sides of data. It can explore the matrix structure of features to enhance the effectiveness of projection. First, we factor integer $D$ as $D = D_{1} \times D_{2}$. Then we reorganize vector $x_{i}$ into matrix $X_{i} \in \mathbb{R}^{D_{1} \times D_{2}}$ such that $vec(X_{i}) = x_{i}$, where $vec(\cdot)$ represents the vectorization of a matrix. And we also have the inverse map of vectorization $vec^{-1}(X_{i}) = X_{i}$, since the vectorization is a one-to-one correspondence if $D_{1}$ and $D_{2}$ are given. To make the transformation more efficient, in this matrix form of local features, we define our hash function using two matrices $\Theta_{1} \in \mathbb{R}^{D_{1} \times d_{1}}$ and $\Theta_{2} \in \mathbb{R}^{D_{2} \times d_{2}}$

$$H(X_{i}) = sgn(vec(\Theta_{1}^{T}X_{i}\Theta_{2})) \in \{-1, +1\}^{d_{1}d_{2}}. \tag{1}$$

In fact, we notice that $vec(\Theta_{1}^{T}X_{i}\Theta_{2}) = (\Theta_{1}^{T} \otimes \Theta_{2}^{T})vec(X_{i}) = (\Theta_{1}^{T} \otimes \Theta_{2}^{T})x_{i}$, where $\otimes$ is the Kronecker product, thus a bilinear projection is simply a special case of the single matrix projection $\Theta$ which can be decomposed as $\Theta = \Theta_{2} \otimes \Theta_{1}$. Besides, it is easy to show that if $\Theta_{1}$ and $\Theta_{2}$ are orthogonal, i.e., $\Theta_{1}^{T}\Theta_{1} = D_{1}I_{d_{1}}$ and $\Theta_{2}^{T}\Theta_{2} = D_{2}I_{d_{2}}$, then $\Theta$ is orthogonal, as well. The bilinear projection leads to a more efficient eigen decomposition on matrices with much smaller sizes $D_{1} \times D_{1}$ and $D_{2} \times D_{2}$ rather than $D_{1}D_{2} \times D_{1}D_{2}$ for single projection. Additionally, the space complexity for bilinear projections is $O(D_{1}^{2} + D_{2}^{2})$, while the single one needs $O((D_{1} \times D_{2})^{2})$. Besides, since most of the local features are represented as concatenated histogram vectors, they can be intrinsically decomposed by two data structures. For instance, 128-dim SIFT is computed on $4 \times 4$ grids and for each grid
a 8-bin histogram is calculated. In this way, a 128-dim SIFT is formed by concatenating $16 \times 8$-bin histograms. Thus, for SIFT feature, we can naturally decomposed it via $16 \times 8$ in our bilinear codes learning.

Note that during the learning stage, we use $\{-1, +1\}$ to represent the output of hash functions and employ centralized data $x_i - (1/N) \sum_{j=1}^{N} x_j$ instead of $x_i, \forall i$. In the indexing phase, we use $\{0, 1\}$ to represent codes for hash lookup.

B. Feature-to-Feature Structure Preserving

To obtain meaningful hash codes for local features, let us first consider the geometric structure of the entire local feature set $\mathcal{F} = \{X_1, \ldots, X_N\}$. We are concerned about the individual relationship between local features in the high-dimensional space, which should also be retained in the lower-dimensional space. Specifically, for similar (dissimilar) pairs, their distance is expected to be minimized (maximized) in the Hamming space. Since the class labels are unavailable for unsupervised method, we first employ $K$-means clustering on $\mathcal{F}$ to obtain some weak label information. Then the pairwise label of $(X_i, X_j)$ is defined as

$$\ell_{ij} = \begin{cases} +1, & X_i \text{ and } X_j \text{ are in the same cluster} \\ -1, & \text{otherwise} \end{cases}$$

Since different pairs have different importance in the embedding, for pair $(X_i, X_j)$, we assign a weight which is related to the pairwise distance with parameter $\sigma$

$$w_{ij}^F = \exp\left(-\frac{\ell_{ij} + 1}{2\sigma^2} \frac{\|X_i - X_j\|^2}{\|X_i - X_j\|^2}\right)$$

where $\|\cdot\|$ is Frobenius norm. We can find that $w_{ij}^F \in (0, 1)$ and for a positive pairwise label, $w_{ij}^F$ is decreasing as the distance $\|X_i - X_j\|$ increases and vice versa. In other words, the positive pair is more important when they are close to each other, and the negative pair is more important when they are far away from each other. We denote $P = \{(i, j)|X_i, X_j \in \mathcal{F}\}$. Therefore, preserving the F2F structure is to maximize

$$\sum_{(i, j) \in P} w_{ij}^F \ell_{ij} \langle H(X_i), H(X_j) \rangle.$$  

The above function reaches its maximum value when $w_{ij}^F \ell_{ij} H(X_i)$ and $H(X_j)$ are similarly sorted due to the rearrangement inequality [43].

C. Image-to-Image Structure Preserving

Now we take a higher level connection, i.e., the relationship between images, into account since source information is also crucial to local features. For image $i$, we still use $X_i$ to represent the local feature set $\{X_{i1}, \ldots, X_{in}\}$ in matrix form. Derived from [17], the I2I distance from image $i$ to image $j$ is defined as

$$d_{ij} = \sum_{X \in X_i} \|X - NN_j(X)\|^2$$

where $NN_j(X)$ is the nearest neighbor of the local feature $X$ in image $j$. Although the number of local features in one image is much smaller than $N$, the nearest neighbor search (NN-search) for all images is still time-consuming. We hope to use the cluster information in the above F2F section for the reduction of complexity. We denote the clusters of the $K$-means clustering on $\mathcal{F}$ by $C_1, \ldots, C_K$. Without loss of generality, supposing the local features of image $j$ are in $C_1, \ldots, C_{K_j}$ and the order of distances from corresponding centroids to $X \in X_i$ is from nearest to farthest, then the range of NN-search in $X_j$ is reduced to $(C_{[\delta N]}) \cap X_j$, where $0 < \delta < 1$ and $[\cdot]$ is the ceiling function. This reduction of range is based on the assumption that the centroid of the cluster where the true nearest neighbor locates is also close to $X$. In fact, it holds when $K \rightarrow N$. After the reduction of searching range, the average complexity is reduced from $O(N^2)$ to $O(N(K1+\delta))$ and we only need to compute the distances from $X$ to the cluster centroids, which has been done in the $K$-means.

In a general situation, $d_{ij} \neq d_{ji}$. Thus, to ensure symmetry, we update the I2I distance as

$$D_{ij} = \frac{1}{2}(d_{ij} + d_{ji}).$$

Via a Gaussian function, we have the following I2I similarity:

$$w_{ij}^I = \exp\left(-\frac{D_{ij}^2}{2\sigma_I^2}\right), \quad i, j = 1, \ldots, n$$

where $\sigma_I$ is the smooth parameter. After applying UBLH, we have the I2I distance in the Hamming space

$$\hat{D}_{ij} = \frac{1}{2}\left(\sum_{X \in X_j} \|H(X) - NN_j(H(X))\|^2 + \sum_{X \in X_i} \|H(X) - NN_i(H(X))\|^2\right).$$

To preserve the I2I structure of the original space, a reasonable objective function is to minimize

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \hat{D}_{ij} \cdot w_{ij}^I.$$

The above function reaches the minimum value when $(\hat{D}_{ij})$ and $[w_{ij}^I]$ are oppositely sorted due to the rearrangement inequality [43]. With the F2F part in (4) and orthogonal constraints on $\Theta_1$ and $\Theta_2$, i.e., $\Theta_1^T \Theta_1 = I$ and $\Theta_2^T \Theta_2 = I$, we have the final optimization problem

$$\arg\max_{\Theta_1^T \Theta_1 = I, \Theta_2^T \Theta_2 = I} \sum_{(i, j) \in P} w_{ij}^F \ell_{ij} \langle H(X_i), H(X_j) \rangle - \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} \hat{D}_{ij} \cdot w_{ij}^I$$

where $\lambda$ is the balance parameter.

D. Alternate Optimization Via Relaxation

In this section, we derive the projections of the optimization problem (10). Motivated by [25] and [32], to gain an optimal solution, we first relax the discrete sign function to a real-valued continuous function by using its signed magnitude,
i.e., $\text{sgn}(x) \approx x$. In this case, the objective function of the F2F part, i.e., (4) becomes
\[
\sum_{(i,j) \in P} w_{ij}^f \ell_{ij}(\text{vec}(\Theta_1^T X_1), \text{vec}(\Theta_2^T X_2))
\]
\[
= \sum_{(i,j) \in P} w_{ij}^f \ell_{ij} \text{Tr}\left( (\Theta_1^T X_1)(\Theta_2^T X_2)^T \right)
\]
\[
= \sum_{(i,j) \in P} w_{ij}^f \ell_{ij} \text{Tr}\left( \Theta_1^T X_1 \Theta_2 \Theta_2^T X_1^T \Theta_1 \right). \tag{11}
\]

Besides, we also make a statistical approximation on the computation of projected I2I distances due to the large number of local features. In other words, we exchange the operation of the pairwise structure has been preserved in the F2F objective function (4), which ensures the correctness of the exchange operation. Hence, the projected $d_{ij}$ in (5) becomes
\[
d_{ij} \approx \sum_{X \in X_i} \| \Theta_1^T X \Theta_2 - \Theta_1^T N(X) \Theta_2 \|^2
\]
\[
= \sum_{X \in X_i} \| \Theta_1^T (X - N(X)) \Theta_2 \|^2
\]
\[
= \sum_{k=1}^{m_i} \text{Tr}\left( \Theta_1^T \Delta X_{ik} \Theta_2 \Theta_2^T (\Delta X_{ik}^T \Theta_1) \right)
\]
\[
= \sum_{k=1}^{m_i} \text{Tr}\left( \Theta_1^T \Delta X_{ik} \Theta_2 \Theta_2^T (\Delta X_{ik}^T \Theta_1) \right). \tag{12}
\]

where $\Delta X_{ik} := X_{ik} - N(X_{ik})$, $k = 1, \ldots, m_i$, $i, j = 1, \ldots, n$. And we also have the similar derivation for $d_{ji}$. Then the projected I2I distance $\bar{d}_{ij}$ can be written as
\[
\bar{d}_{ij} \approx \frac{1}{2} \sum_{k=1}^{m_i} \text{Tr}\left( \Theta_1^T \Delta X_{ik} \Theta_2 \Theta_2^T (\Delta X_{ik}^T \Theta_1) \right)
\]
\[
+ \frac{1}{2} \sum_{k=1}^{m_j} \text{Tr}\left( \Theta_1^T \Delta X_{jk} \Theta_2 \Theta_2^T (\Delta X_{jk}^T \Theta_1) \right). \tag{13}
\]

Since it is a nonconvex optimization problem, to the best of our knowledge, there is no direct way to output the projections $\Theta_1$ and $\Theta_2$ simultaneously. We derive an alternate iteration algorithm to update one projection when given the other, i.e., we optimize $\Theta_1$ when $\Theta_2$ is fixed and we fix $\Theta_1$ to update $\Theta_2$ iteratively. Combined with (11) and (13), let us denote the objective function by
\[
L(\Theta_1, \Theta_2)
\]
\[
= \sum_{(i,j) \in P} w_{ij}^f \ell_{ij} \text{Tr}\left( \Theta_1^T X_1 \Theta_2 \Theta_2^T X_2^T \Theta_1 \right)
\]
\[
- \frac{1}{2} \lambda \sum_{k=1}^{m_i} w_{ij}^f \text{Tr}\left( \Theta_1^T \Delta X_{ik} \Theta_2 \Theta_2^T (\Delta X_{ik}^T \Theta_1) \right)
\]
\[
+ \frac{1}{2} \sum_{k=1}^{m_j} w_{ij}^f \text{Tr}\left( \Theta_1^T \Delta X_{jk} \Theta_2 \Theta_2^T (\Delta X_{jk}^T \Theta_1) \right). \tag{14}
\]

Algorithm 1: Unsupervised Bilinear Local Hashing

**Input:** The local feature set $\mathcal{F}$ of training images, the number of centroids $K$ in the K-means, the parameter $\delta$ for the NN-search in the I2I distance and the balance parameter $\lambda$.

**Output:** The bilinear projection matrices $\Theta_1$ and $\Theta_2$.

1. Preprocessing: centralize $x_i \leftarrow (x_i - \frac{1}{N} \sum_{j=1}^{N} x_j)$, $i = 1, \ldots, N$;
2. Transform all the local features $x_i$ into matrix form $X_i$, $i = 1, \ldots, N$;
3. Employ K-means clustering on $\mathcal{F}$;
4. Construct local feature pairing set $\mathcal{P}$ and their corresponding pairwise labels $\ell_{ij}$ according to Eq. (2);
5. Compute F2F weight $w_{ij}^f$ and I2I similarity $w_{ij}^s$ by Eqs. (3) and (7), respectively;
6. Initialize $\Theta_2$ randomly;
7. **repeat**
8. $\Theta_1 \leftarrow$ the eigenvectors corresponding to the largest $d_1$ eigenvalues of $M_2(\Theta_2)$ by Eq. (15);
9. $\Theta_2 \leftarrow$ the eigenvectors corresponding to the largest $d_2$ eigenvalues of $M_1(\Theta_1)$ by Eq. (16);
10. **until** $L(\Theta_1, \Theta_2)$ converges.
11. **return** $\Theta_1$ and $\Theta_2$.

By simple algebraic derivation, we have the following form:
\[
L(\Theta_1, \Theta_2) = \text{Tr}(\Theta_1^T M_2(\Theta_2) \Theta_1) = \text{Tr}(\Theta_2^T M_1(\Theta_1) \Theta_2)
\]
where
\[
M_2(\Theta_2) = \sum_{(i,j) \in P} w_{ij}^f \ell_{ij} X_i \Theta_2 \Theta_2^T X_j^T
\]
\[
- \frac{1}{2} \lambda \sum_{k=1}^{m_i} w_{ij}^f \text{Tr}(\Theta_1^T \Delta X_{ik} \Theta_2 \Theta_2^T (\Delta X_{ik}^T \Theta_1) )
\]
\[
+ \frac{1}{2} \sum_{k=1}^{m_j} w_{ij}^f \text{Tr}(\Theta_1^T \Delta X_{jk} \Theta_2 \Theta_2^T (\Delta X_{jk}^T \Theta_1) ) \tag{15}
\]
and
\[
M_1(\Theta_1) = \sum_{(i,j) \in P} w_{ij}^f \ell_{ij} X_i \Theta_1 \Theta_1^T X_j
\]
\[
- \frac{1}{2} \lambda \sum_{k=1}^{m_i} w_{ij}^f (\Delta X_{ik}^T \Theta_1 \Theta_1^T \Delta X_{ik})
\]
\[
+ \frac{1}{2} \sum_{k=1}^{m_j} w_{ij}^f (\Delta X_{jk}^T \Theta_1 \Theta_1^T \Delta X_{jk}) \tag{16}
\]
are two symmetric matrix-valued functions with their codomains $\mathbb{R}^{D_1 \times D_1}$ and $\mathbb{R}^{D_2 \times D_2}$, respectively. Consequently, for fixed $\Theta_1$, the optimal $\Theta_2$ is given as the eigenvectors corresponding to the largest $d_2$ eigenvalues of $M_1(\Theta_1)$, and likewise, for fixed $\Theta_2$, the optimal $\Theta_1$ is given as the eigenvectors corresponding to the largest $d_1$ eigenvalues of $M_2(\Theta_2)$. Although the number of the local features is usually huge, the sizes of our final matrices $M_1$ and $M_2$ used for decomposition are small enough ($D_1$ and $D_2$ are always less than 100). This property mainly guarantees the efficiency and feasibility.
Therefore, for \( t = 0 \), we randomly initialize \( \Theta_1^{(t)} \); for the \( t \)th step, we have the update rules
\[
\Theta_1^{(t)} \leftarrow \text{the first } d_1 \text{ eigenvectors of } M_2(\Theta_2^{(t-1)})\\
\Theta_2^{(t)} \leftarrow \text{the first } d_2 \text{ eigenvectors of } M_1(\Theta_1^{(t)}).
\]
For the \( t \)th step (\( t \geq 1 \)), we have the following inequality:
\[
\mathcal{L}(\Theta_1^{(t-1)}, \Theta_2^{(t-1)}) \leq \mathcal{L}(\Theta_1^{(t)}, \Theta_2^{(t)}) \leq \mathcal{L}(\Theta_1^{(0)}, \Theta_2^{(0)}).
\]
Thus \( \mathcal{L}(\Theta_1^{(t)}, \Theta_2^{(t)}) \) is monotonically nondecreasing as \( t \to \infty \). And continuous function \( \mathcal{L}(\Theta_1, \Theta_2) \) is bounded in the closed district \( \{ (\Theta_1, \Theta_2) | \Theta_1^T \Theta_1 = I, \Theta_2^T \Theta_2 = I \} \). Then we stop the iteration when the difference \( |\mathcal{L}(\Theta_1^{(t)}, \Theta_2^{(t)}) - \mathcal{L}(\Theta_1^{(t-1)}, \Theta_2^{(t-1)})| \) is less than a small threshold or the number of iteration reaches a maximum. We summarize UBLH in Algorithm 1.

IV. INDEXING VIA LOCAL HASHING VOTING

Once the bilinear projection matrices \( \{ \Theta_1, \Theta_2 \} \) are obtained, we can embed the training data into binary hash codes by (1). And for a query local feature \( \hat{x} \), its hash code is obtained by
\[
H(\hat{x}) = \text{sgn} \left( \text{vec} \left( \Theta_1^T \left( \hat{x} - \frac{1}{N} \sum_{j=1}^{N} X_j \right) \Theta_2 \right) \right) \tag{17}
\]
with the input of centralized data, where \( \hat{x} \) is the matrix form of \( \hat{x} \). For an upcoming query, a common way to find the similar samples in the training set by using Hamming distance ranking. However, for our local feature hashing scenario, traditional linear search (e.g., Hamming distance ranking) with complexity \( O(N) \) is not fast any more, since \( N \) denotes the total number (at least 3M for a large-scale database)\(^1\) of local features. To accomplish the local feature-based visual retrieval, in this paper, we introduce a fast indexing scheme via LHV as shown in Fig. 3. We first build the Hamming lookup table (also known as the hashing table) into our LHV scheme. Given a query, we can find the bucket of corresponding hash codes in near constant time \( O(1) \), and return all the data in the bucket as the retrieval results.

After construction of the Hamming lookup table over the training set, we store the corresponding indices for the hash codes of all local features. For instance, given a bucket with hash code “1100101001,” we store the indices of the images, which contain the same local feature hash code with this bucket. In this way, we search the hash code \( H(\text{vec}^{-1}(q_i)) \) for each local feature \( \mathbf{q}_k \in \mathcal{Q} \) in the query image \( \mathcal{Q} = \{ \mathbf{q}_1, \ldots, \mathbf{q}_m \} \) over the Hamming lookup table within Hamming radius \( r \) and return the possible images’ indices. It is noteworthy that the same bucket in the Hamming lookup table may store the indices from different images. Finally, we vote and accumulate the times of each image’s indices appearing in relevant buckets and then rank them in decreasing order. The final retrieved samples are returned according to the relevant ranking generated by LHV, which is depicted in Algorithm 2.

\(^{1}\)For large-scale database retrieval, the total number of local features is always very huge. In practice, for a 10K images training set, if each image contains 300 local feature, \( N \) would be \( 300 \times 10K = 3M \).
V. Complexity Analysis

The time complexity of UBLH mainly contains three parts. The first part is computing the F2F weight $w_{ij}$, which costs $O(|P|D+NKTD)$, where $T$ is the number of iterations in the K-means. The second part is constructing the I2I similarity $w_{ij}$. Using the reduction strategy in NN-search, the average time complexity of this part is $O(NK^{1+\delta}D)$. The last part is the eigen decomposition for the bilinear projection matrices via alternate optimization. The updates of $\Theta_1$ and $\Theta_2$ have the time complexity of $O(D_1^3)$ and $O(D_2^3)$, respectively. The test phase is image indexing via the LHV. For the hash code of each local feature, the index complexity is $O(1)$ using the Hamming lookup table. Thus, it only needs $O(m)$ time for a query with $m$ local features in the search phase. In total, if we select local feature pairs less densely such that $|P| \sim O(N)$ and set $\delta = 0.5$ for the reduction of complexity, the time complexity of UBLH is at most $O(ND + NKTD + NK^{1+\delta}D) + O(D_1^3 + O(D_2^3))NT + O(m)$, where $NT$ is the number of the iteration for alternate optimization. In the experiments, $NT$ is always less than 10.

VI. Experiments

In this section, the proposed UBLH algorithm is evaluated for the image similarity search problem. Three realistic image data sets are used in our experiments: Caltech-256 [44], SUN397 [45], and Flickr 1M. The Caltech-256 data set consists of 30,607 images associated with 256 object categories. We further randomly choose 1000 images as the query set and the rest of data set is regarded as the training set. The SUN397 data set contains 108,754 scene images in total from 397 well-sampled categories with at least 100 images per category. Seventy samples are randomly selected from each category to construct the training set and the rest of samples are the query set. Thus, there are total numbers of 27,790 and 80,964 in the training set and query set, respectively. For the Flickr 1M data set, it contains one million Web images collected from the Flickr. We take 1K images as the queries by random selection and use the remaining to form the gallery database. Considering the huge cost of computation, in this experiment, only 150,000 randomly selected samples from the gallery database form the training set. Furthermore, for image searching tasks, given an image, we would like to describe it with a set of local features extracted from it. In our experiments, we extract 128-D SIFT as the local feature to describe the images and then learn to hash these local descriptors with all compared methods. Particularly, considering the computational cost, we limit the maximum number of local features extracted from one image with 700.

In the querying phase, using LHV as the retrieval tool, a returned point is regarded as a neighbor if it lies in the top ranked 200, 200, and 1000 points for Caltech-256, SUN397, and Flickr 1M, respectively. Specifically in LHV, we only consider the local hash codes lying in the buckets that fall within a small Hamming radius $r = 2$ (following [25]) in the Hamming lookup table which is constructed using the training codes.

We evaluate the retrieval results in terms of the mean average precision (MAP) and the precision-recall curve by changing the number of top ranked points in LHV. Additionally, we also report the training time and the test time (the average searching time used for each query) for all methods. Our experiments are completed using MATLAB 2013a on a server configured with a 12-core processor and 128 GB of RAM running the Linux OS.

A. Compared Methods and Settings

Since our UBLH is the work of unsupervised linear hashing for local features, for fair comparison, the other hashing techniques originally proposed for global features are also directly applied to local features. In our experiments, we first compare the proposed method against twelve prevailing hashing algorithms, including four supervised methods: 1) LDAH [29]; 2) BRE [30]; 3) MLH [31]; and 4) KSH [32], and eight unsupervised methods: 1) LSH [21]; 2) WTA [46]; 3) PCAH [24]; 4) SpH [25]; 5) AGH [26]; 6) CH [28]; 7) ITQ [47]; and 8) spherical hashing (SpherH) [48]. Besides, HE [38], which is a nonlinear local feature-based hashing method, is also included in our comparison. We use the publicly available codes of BRE, MLH, LDAH, SpH, AGH, ITQ, and SpherH, and implement LSH, PCAH, KSH, CH, HE, and WTA ourselves. All of the above methods are then evaluated on six different lengths of codes (16, 32, 48, 64, 80, and 96). Under the same experimental setting, all the parameters used in the compared methods have been strictly chosen according to their original papers.

For our UBLH, to obtain the weak label information, the parameter $K$ of the $K$-means in the proposed method for each data set is selected from one of {300, 400, ..., 1200} with the step of 100 by 10-fold cross-validation on the training sets. The parameter $\delta$ is always fixed to 0.5. Besides, we set $D_1 = 16$ and $D_2 = 8$ for the transformation of 128-D SIFT local features (see Section III). Additionally, the optimal balance parameter $\lambda$ is chosen from one of {0.05, 0.1, ..., 0.5} with the step of 0.05 via the training sets, as well.

B. Results and Analysis

We demonstrate MAP curves on the Caltech-256, SUN397, and Flickr 1M data sets compared with different methods in Fig. 4. All the results are calculated via the proposed LHV ranking algorithm under the same setting. From the general tendency, accuracies on the SUN397 data set are lower than those on the other two data sets, since more categories and large intraclass variations exist in SUN397. Our UBLH algorithm consistently outperforms all the compared methods in every length of code. Especially on the Caltech256 data set, the improvement is near 5% between UBLH and the top supervised method KSH on each code length. Beyond that, we can observe that due to the available label information in the learning phases, the supervised methods, such as KSH, BRE, and MLH, always achieve better performance than the compared unsupervised methods on all three data sets. Interestingly, the results of LDAH always climb up then go down when the length of code increases. The same tendency also appears with
BRE, KSH, and PCAH. The two unsupervised hashing methods, ITQ and SpherH, generally outperform other compared unsupervised methods, while achieve worse accuracies than the supervised methods. Our UBLH achieves dramatically better performance than all other unsupervised methods and also reaches higher accuracies compared with the supervised ones over three data sets. This is because we consider the geometry structure of local features and the global relationship between images simultaneously. Besides, in Table II we also illustrate the results computed via LFBC under different local feature decomposition. Since SIFT feature is intrinsically composed via 16 grid with 8-bin histograms, the best naturally bilinear decomposition, i.e., $8 \times 16 = 128$ dim, can achieve the better results than other decomposition ways.

To make the comparison more convincing, some hashing schemes based on global representations are also included in our comparison. For all three data sets, we first use the $K$-means scheme to construct the codebooks with size of 500 and 1000, respectively, and then encode SIFT features into global representations via VLAD [49], which are proved to be more discriminative than original BoW representations. After that, two best performed hashing methods, i.e., KSH and BRE, are used to learn the hash codes on these global representations. Additionally, we also list the search performance via directly using the global feature GIST with KSH and BRE. In Fig. 5, it shows that our local hashing method UBLH with LHV achieves better results than the global representation-based hashing schemes with the ordinary hash table ($r = 2$).
This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

LIU et al.: UNSUPERVISED LOCAL FEATURE HASHING FOR IMAGE SIMILARITY SEARCH

---

**Fig. 7.** Performance comparison (MAP) on the three data sets at 32-bit. UBLH(F2F) and UBLH(I2I) represent the proposed method only using the F2F term or the I2I term, respectively.

**TABLE I**

<table>
<thead>
<tr>
<th>Methods</th>
<th>MAP at 32-Bit</th>
<th>Training Time</th>
<th>Test Time (with LHV)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caltech-256</td>
<td>SUN397</td>
<td>Flickr 1M</td>
<td></td>
</tr>
<tr>
<td>KSH*</td>
<td>0.196</td>
<td>4741.1s</td>
<td>57.4μs</td>
</tr>
<tr>
<td>BRE*</td>
<td>0.188</td>
<td>2.84 × 10^4</td>
<td>32.9μs</td>
</tr>
<tr>
<td>MLH*</td>
<td>0.180</td>
<td>1.19 × 10^4</td>
<td>18.3μs</td>
</tr>
<tr>
<td>LDAH*</td>
<td>0.195</td>
<td>15.3s</td>
<td>11.2μs</td>
</tr>
<tr>
<td>AGH</td>
<td>0.172</td>
<td>764.2s</td>
<td>97.4μs</td>
</tr>
<tr>
<td>SpH</td>
<td>0.154</td>
<td>282.7s</td>
<td>103.4μs</td>
</tr>
<tr>
<td>PCAH</td>
<td>0.142</td>
<td>17.4s</td>
<td>16.1μs</td>
</tr>
<tr>
<td>LSH</td>
<td>0.104</td>
<td>9.1s</td>
<td>8.3μs</td>
</tr>
<tr>
<td>WTA</td>
<td>0.122</td>
<td>1004.2s</td>
<td>57.4μs</td>
</tr>
<tr>
<td>ITQ</td>
<td>0.180</td>
<td>811.6s</td>
<td>93.0μs</td>
</tr>
<tr>
<td>SphER</td>
<td>0.185</td>
<td>957.3s</td>
<td>105.6μs</td>
</tr>
<tr>
<td>CH</td>
<td>0.180</td>
<td>1712.5s</td>
<td>64.5μs</td>
</tr>
<tr>
<td>RE</td>
<td>0.174</td>
<td>1565.0s</td>
<td>87.2μs</td>
</tr>
<tr>
<td>UBLH</td>
<td>0.239</td>
<td>1148.1s</td>
<td>56.1μs</td>
</tr>
</tbody>
</table>

(The test time indicates the average searching time used for each query. * denotes the supervised hashing methods.)

---

**Fig. 8.** Illustration of convergence on Caltech-256 with the code length 32. (a) Objective function value versus number of iteration. (b) Differences of $\Theta_1$ and $\Theta_2$ versus number of iteration.

---

for retrieval. Moreover, precision-recall curves of all the compared methods on these data sets with the code length of 96 bits are presented in Fig. 6 as well. From all these figures, we can further discover that, for all three data sets, UBLH achieves significantly better performance than other unsupervised methods and still slightly outperforms supervised ones by comparing the MAP and area under the curve. Fig. 8 illustrates the convergence of the proposed ULBH on Caltech-256 with the code length of 32. We can clearly observe that the objective function value is stable when the number of iteration is larger than 3. Besides, it is easy to show that with the increase in the number of iteration, the differences of $\Theta_1$ and $\Theta_2$, i.e., $\|\Theta_1(t) - \Theta_1(t-1)\|$ and $\|\Theta_2(t) - \Theta_2(t-1)\|$, where $t$ is the number of iteration, dramatically drop down. Additionally, in Fig. 9, we also compare the performance of UBLH with respect to the parameter $K$ in the $K$-means and balance parameter $\lambda$ via cross-validation on the training sets.

In addition, to illustrate the effectiveness of F2F and I2I terms in our method, we compare the algorithm only using the F2F term or the I2I term in Fig. 7. The results indicate that preserving the I2I similarity is more effective than preserving the similarity between features. Meanwhile, combining them together could always gain better performance.

---

**TABLE II**

<table>
<thead>
<tr>
<th>Decomposition methods</th>
<th>Caltech-256</th>
<th>SUN397</th>
<th>Flickr 1M</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1 \times 128$</td>
<td>0.225</td>
<td>0.115</td>
<td>0.235</td>
</tr>
<tr>
<td>$2 \times 64$</td>
<td>0.229</td>
<td>0.117</td>
<td>0.239</td>
</tr>
<tr>
<td>$4 \times 32$</td>
<td>0.231</td>
<td>0.120</td>
<td>0.241</td>
</tr>
<tr>
<td>$8 \times 16$</td>
<td>0.239</td>
<td>0.122</td>
<td>0.250</td>
</tr>
</tbody>
</table>
Finally, the training time and test time for different algorithms on three data sets are also illustrated in Table I. Considering the training time, supervised methods always need more time for the hash learning except for LDAH. In particular, BRE and MLH spend the most time to train hash functions. The random projection-based algorithms are relatively more efficient, especially the LSH. Our UBLH costs significantly less time than KSH, BRE, MLH, and CH but more than other methods for training. In terms of the test phase, LSH, LDAH, and PCAH are the most efficient methods due to the simple matrix multiplication or thresholding in binary code learning, while AGH has the comparable searching time with SpH. Our UBLH costs similar time as WTA. More details can be seen in Table I.

VII. SCALABILITY FOR VERY LARGE DATA SETS

For more realistic retrieval on very large-scale data sets (e.g., the Google database), the proposed approach would become time-consuming for training due to a huge number of local features generated and involved in our computation. To avoid the heavy burden of computation and make our method practical in such cases, we can use anchor point quantization (APQ) to reduce the computational complexity of our UBLH. Inspired by [50], we can first extract the anchor points from all local features via clustering techniques (e.g., K-means) as we have stated in our algorithm. Then, each local feature in the training set can be quantized to one anchor point. In this way, we replace all local features with their corresponding anchor points in the training phase. In particular, F2F preserving can be effectively transferred to anchor point to anchor point preserving. Similarly, we can also use anchor points for I2I preserving. Thus, APQ can be applied on very large-scale data collections to enable more efficient training than directly using a huge number of original local features.

VIII. CONCLUSION

In this paper, we have presented a novel unsupervised hashing framework, namely UBLH, to learn highly discriminative binary codes on local feature descriptors for large-scale image similarity search. The bilinear property of UBLH lets it explore the matrix representation of local features. Considering the pairwise and source information of local features, as a result, the F2F and I2I structures have been simultaneously preserved in UBLH. We have systematically evaluated our method on the Caltech-256, SUN397, and Flickr 1M data sets and shown promising results compared to state-of-the-art hashing methods.

REFERENCES


