Adaptive Hashing With Sparse Matrix Factorization

Huawen Liu, *Member, IEEE*, Xuelong Li, *Fellow, IEEE*, Shichao Zhang, *Senior Member, IEEE*, and Qi Tian, *Fellow, IEEE*

***Abstract*—Hashing offers a desirable and effective solution for efficiently retrieving the nearest neighbors from large-scale data because of its low storage and computation costs. One of the most appealing techniques for hashing learning is matrix factorization. However, most hashing methods focus only on building the mapping relationships between the Euclidean and Hamming spaces and, unfortunately, underestimate the naturally sparse structures of the data. In addition, parameter tuning is always a challenging and head-scratching problem for sparse hashing learning. To address these problems, in this article, we propose a novel hashing method termed adaptively sparse matrix factorization hashing (SMFH), which exploits sparse matrix factorization to explore the parsimonious structures of the data. Moreover, SMFH adopts an orthogonal transformation to minimize the quantization loss while deriving the binary codes. The most distinguished property of SMFH is that it is adaptive and parameter-free, that is, SMFH can automatically generate sparse representations and does not require human involvement to tune the regularization parameters for the sparse models. Empirical studies on four publicly available benchmark data sets show that the proposed method can achieve promising performance and is competitive with a variety of state-of-the-art hashing methods.**

***Index Terms*—Binary code, hashing, image retrieval, matrix factorization, sparse learning.**

I. Introduction

WITH the rapid advancement of information technologies and the prevalence of social networks, considerable quantities of data, including images, videos, and texts collected from different application domains, such as computer vision, machine learning, and information retrieval, are increasing in a marvelous and unprecedented way. As the data grow explosively, how to efficiently organize and index the large-scale data is a challenging and fundamental issue [1]. From the viewpoint of practical applications, efficiently retrieving relevant information from such large-scale data with good scalability has become an emerging need for the communities of computer vision and information retrieval.

Nearest neighbor (NN) search, also known as similarity search, is of particular interest and receives significant attention in computer vision, pattern recognition, recommendation systems, and information retrieval [2]–[4]. Given a query, NN aims to identify those similar or nearest data points from a large data collection by using some distances or similar measurements [5]. Nonetheless, traditional NN algorithms are neither scalable nor efficient for large-scale data because the time complexity of finding exact NNs is linear with respect to the size of data [6]. Especially when the data have a magnitude of millions or billions, NN algorithms are usually computationally prohibitive and infeasible for practical applications.

Approximate NN (ANN) search is an alternative solution to end the aforementioned problem. It exploits heuristic strategies, such as tree-based structures [7] and hash functions [8], to obtain ANNs with sublinear or even constant-time complexity [9], [10]. Due to its efficiency, many effective ANN methods have been developed in the literature. Broadly, they can be grouped into three different categories, i.e., tree-based, vector quantization, and hashing methods [11]. The tree-based methods reorganize the data as indexing-tree structures, such as KD-tree, M-tree, and R-tree [7], [12], while the vector quantization methods transfer the data into a schema of vectors [13]. Both schemes work well when the data dimensionality is low. However, they may suffer from the curse of dimensionality and memory constraints if the dimensionality is high [12].

The hashing methods encode the high-dimensional data as the representation of binary codes by a series of hash functions while preserving the neighborhood relationships (e.g., similarities or distances) of the data. As a result, the derived binary codes corresponding to the data points that are close in the Euclidean space should also have small Hamming distances, which can be obtained efficiently, usually with $O(1)$ time complexity [4]. Benefiting from bit operations, the binary-code representation can not only enable neighbor retrieval to be extremely efficient without greatly degrading the performance but also take much less storage space compared to the original data [1]. This, however, is especially crucial for large-scale data, where low storage and efficient computation are always appealing.

Manuscript received December 5, 2018; revised July 16, 2019 and October 10, 2019; accepted November 15, 2019. Date of publication December 30, 2019; date of current version October 6, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1107400, in part by the National NSF of China (NSFC) under Grant 61572443, Grant 61871470, Grant 61761130079, and Grant 61976195, in part by the Key Project of NSFC under Grant 61836016, and in part by the NSF of Zhejiang Province under Grant LY18F020019. *(Corresponding author: Huawen Liu.)*

H. Liu is with the Department of Computer Science, Zhejiang Normal University, Jinhua 321004, China (e-mail: hwliu@zjnu.edu.cn).

X. Li is with the School of Computer Science and the Center for OPTical IMagery Analysis and Learning (OPTIMAL), Northwestern Polytechnical University, Xi'an 710072, China (e-mail: xuelong_li@nwpu.edu.cn).

S. Zhang is with the School of Computer Science and Engineering, Central South University, Changsha 410083, China (e-mail: zhangsc@gxnu.edu.cn).

Q. Tian is with Huawei Noah's Ark Lab, Shenzhen 518129, China (e-mail: tian.qi1@huawei.com).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2954856

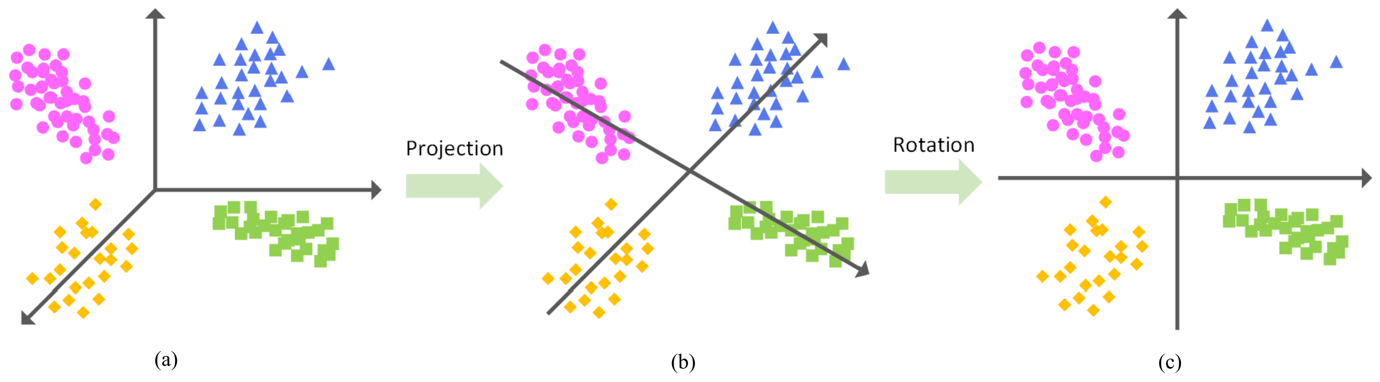

Fig. 1. Toy example of SMFH (see Section III for details). The basic encoding scheme of our method consists of two stages: sparse projection and optimal rotation. (a) Data points from four clusters in a 3-D feature space. (b) Sparse projection: the data are projected into a low-dimensional space via adaptively sparse matrix factorization, and each axis corresponds to a principal component direction. (c) Optimal rotation: the projected space is rotated with respect to the structure of the data to achieve low quantization errors for hash encoding. (a) Original space. (b) Projected space. (c) Hamming space.

Motivated by the potential advantages, hashing learning has shown great promise, and a rich number of hashing methods, including data-independent hashing and data-dependent hashing, have been proposed in recent decades [1] [4] [8]. The data-independent hashing (also known as random projection-based hashing) randomly generates a set of hash functions without involving the data. Representative examples include locality-sensitive hashing (LSH) [14] and its variants, e.g., KLSH [15] and SKLSH [16]. Theoretically, the similar property of the data can be progressively approximated if the generated hash codes are long enough. In contrast, the data-dependent hashing (also known as learning-based hashing) seeks projection functions to capture the underlying geometries of data. Typical examples of such methods are PCA hashing (PCAH) [17], spectral hashing (SH) [18], iterative quantization (ITQ) [19], circulant binary embedding (CBE) [20], density sensitive hashing (DSH) [21], spherical hashing (SpH) [22], and sparse embedding and least variance encoding (SELVE) [23]. Generally, the data-dependent methods have a promising performance with shorter hash codes, making them more popular. Nevertheless, learning the projection functions from the data is difficult and time consuming.

In this article, we also focus our attention on hashing learning and propose a novel two-stage hashing method, dubbed as adaptively sparse matrix factorization hashing (SMFH). Fig. 1 gives a toy example of SMFH. It mainly consists of two stages: sparse projection and optimal rotation. First, the projection stage exploits the technique of matrix factorization to automatically learn the latent and local structural information of the data so that the similarity property of the data can be retained after projection. Since the weight coefficients obtained by transformation are continuous and require a large sum of memory to store, we further make them sparse without losing considerable information. In routine sparse models, the regularization parameters are often tuned empirically or determined experimentally. This, however, is time consuming and heavily data dependent, resulting in the overfitting problem. To end this, here, we exploit heuristic strategies to avoid the head-scratching problem of the tuning parameter. As a result, SMFH is adaptive and parameter-free, where no parameter is required to be tuned. The second stage aims at rotating principal directions of the data projected from the original feature space to optimally encode the data into binary codes with low quantization errors.

In summary, we propose a simple yet effective learning method for hashing, which brings potential benefits to learning systems for large-scale data. The major contributions of this article are briefly highlighted as follows.

1) We formulate the learning method for hashing as an optimization problem and then solve it by using matrix factorization, which is good at capturing the latent structural information of the data. Furthermore, an orthogonal rotation is performed on the projection coefficients to generate more discriminative and representative binary codes.
2) An effective sparsity-inducing function, which can automatically yield a parsimonious model according to the data at hand, is introduced to make the projection coefficients sparse, reducing their storage consumption.
3) More importantly, a heuristic strategy for tuning the regularization parameters is introduced, making the sparsity-inducing function adaptive and parameter-free. Thus, the regularization parameters are no longer required when generating sparse models without considering the head-scratching problem of parameter tuning in SMFH.

The rest of this article is organized as follows. Section II provides a brief review of hashing algorithms. In Section III, we first present the notations used in this article and the problem formulation and then provide the details of our adaptively SMFH method. Experimental results on four publicly used data sets are reported in Section IV, followed by the concluding remarks of this article in Section V.

II. Related Work

As mentioned earlier, hashing learning has received considerable attention due to its low storage cost and high computational efficiency, and great endeavors have been attempted in recent decades [1], [4], [8]. The existing hashing learning methods generally follow two lines: the data-independent hashing and the data-dependent hashing. For the former, the projection functions are constructed independently of the data. The most classic example is LSH [14], which randomly projects the data points in the Euclidean space to the corresponding binary codes in the Hamming space. Theoretically, the closer the data points, the higher the collision probability of the corresponding binary codes, and vice versa, as the length of the binary codes is long enough. Because of its efficiency, LSH has become very popular, and the similarity measurements embedded have also been extended to the p-norm distance [24], the Mahalanobis distance [25], the Hamming affinity [14], the kernel similarity [15], and the cosine similarity [30]. Unfortunately, the LSH-related methods have not considered the intrinsic structural information of the data. In addition, an acceptable performance for LSH requires longer binary codes [8].

The data-dependent or learning-based methods derive shorter binary codes while retaining the characteristics of the data. Since the generated codes are more effective and compact, the data-dependent methods have recently flourished. Depending on the availability of label information, they can be further divided into three major subcategories: unsupervised hashing, supervised hashing, and semisupervised hashing (SSH) [8]. In the semisupervised and supervised hashings, the data label information is required to learn hashing functions or codes with more discriminative capabilities. Typical examples include, but are not limited to, SSH [26], kernel-based supervised hashing (KSH) [27], fast supervised discrete hashing (FSDH) [28], supervised discrete hashing with relaxation (SDHR) [29], and angular reconstructive embeddings (AREs) [30]. As an example, Ding *et al.* [31] first transferred data labels to binary values via LSH and then took several binary classifiers as hashing functions to reap the binary codes.

Auxiliary contexts of data can also benefit to generating binary codes if available [32]. Indeed, the same objects in reality often exhibit different representations, e.g., images or videos are described or accompanied by text information. Taking the contexts into consideration, Zhu *et al.* [33] leveraged contextual modalities to retain visual similarities when transferring, yielding the binary codes with strong semantics. Yu *et al.* [34] obtained meaningful binary codes after mapping local features of data points at both the image and text levels into a common space. Jin *et al.* [35] encoded rank structures of features into ordinal representation and then used it to generate discriminative codes, while Wang *et al.* [36] transformed multimodal data into latent semantic spaces, where the same labels from different models have the same semantic and representations. RFDH [37] utilizes a discrete matrix decomposition technique, coupled with an $\ell_{2,1}$-norm, to minimize the quantization errors among multimodal data and then projects data into effective binary codes with two hash functions. Zheng *et al.* [38] measured the similarities between each pair of modalities by three order random walks and then transformed into the problem of regularized support vector learning for the sake of efficiency.

Unsupervised hashing exploits underlying structures or distributions of the data to learn the binary codes without involving label information, which is not always available in real-world applications. During the past decades, a variety of unsupervised hashing methods, such as DSH [21], PCAH [17], SH [18], and SpH [22], have been witnessed. For example, Jin *et al.* [39] applied a ranking-based LSH on fingerprint templates to develop the Gaussian random projection-based and uniformly random permutation-based hashing schemes for fingerprint biometric protection. Huang *et al.* [40] proposed an online hashing model for sequential or stream data by using a similarity loss function, which measures the difference of the binary codes of a pair of data points in the Hamming space. Unlike SH [18], reversed SH (ReSH) [41] defines the similarities of the data points as those of hash codes and interchanges the input and output of SH. In this way, the similar data points are encoded into adjacent binary codes, while the dissimilar data points were separated from each other. KRH [42] takes a normalized Gaussian kernel to preserve the local distribution of data and then uses a low-rank approximation technique to obtain optimal binary codes.

The ITQ is an intensively studied branch of unsupervised hashing. It iteratively quantifies the continuous vectors of data points into discrete ones to achieve compact binary codes. The most representative examples of such kind are ITQ and its supervised variants (e.g., CCA-ITQ [19]). Cao *et al.* [43] integrated deep learning with the quantization approach to jointly learn deep visual-semantic embeddings and quantizers using hybrid networks. To handle the dilemma with insufficient data, Zhou *et al.* [44] introduced a concept called transfer hashing by extending ITQ for transfer learning. Duan *et al.* [45] treated projection and quantization as a whole and solved it by using the minimal reconstruction bias of signals. Yuan *et al.* [46] adopted k-means to quantize codes on each feature that is generated by using feature clustering.

Matrix factorization is another promising technique for hashing learning because it can effectively capture latent and intrinsic locality structures of data [47]. Recently, it has attracted increasing interest from the community of hashing learning, and a rich number of hashing methods adopt matrix factorization to obtain discriminative hash codes. For example, Ding *et al.* [48] utilized collective matrix factorization to learn latent factor models that were subsequently used to produce unified binary codes. Similarly, with the help of collective matrix factorization, Tang *et al.* [49] took both label consistencies and local geometric consistencies as a mixed-graph Laplacian regularization term to obtain the discriminative binary codes. Lai *et al.* [50] integrated the ideas of anchor graph, dimension reduction, and rotation operation together to minimize the quantization errors between binary codes and low-dimensional feature spaces. Besides, the $\ell_{2,1}$-norm regularization term was conducted to obtain a jointly sparse projection matrix for feature selection. Lu *et al.* [12] first obtained latent semantic features via matrix decomposition and then combined them with a minimum encoding loss to generate more discriminative binary codes.

In this article, we also focus our concentrations on the matrix factorization technique for unsupervised hashing. Unlike the aforementioned work, our contributions will be made on its sparse variant without involving regularization parameters for the sake of saving memory and time consumption.

III. Sparse Matrix Factorization Hashing

In this section, we deliberately present the implementation details about the proposed method termed SMFH. As illustrated in Fig. 1, the key idea of SMFH consists of projection and rotation stages. The former maps the data points into a low-dimensional feature subspace with a sparse representation by using an adaptively sparse matrix factorization, while the latter derives the projected vectors by optimally rotating the projection directions with orthogonal transformation so that the data points can be encoded as the semantic-preserving binary codes.

A. Notations and Problem Statements

Throughout this article, we use bold-faced uppercase letters, such as $\mathbf{X}$, to denote matrices and bold-faced lowercase letters, such as $\mathbf{x}$, to represent (column) vectors, respectively. The (i, j)th element in $\mathbf{X}$ is denoted as x_{ij}. $\mathbf{X}^T$ is the transpose of $\mathbf{X}$. For the vector $\mathbf{x}$, x_i denotes the ith element of $\mathbf{x}$. The Frobenius norm of $\mathbf{x}$ is denoted as $\|\mathbf{x}\|_2$. Let $\mathcal{I}_k \subseteq \{1, \ldots, k\}$ be an index set, and the notation '$\sqsubseteq$' be a sparse structure relationship. As an example, $\bar{\mathbf{x}} \sqsubseteq \mathbf{x}$ implies that $\bar{\mathbf{x}}$ is a parsimonious vector of $\mathbf{x}$, where both vectors have the same number of elements and each element $\bar{x}_i$ of $\bar{\mathbf{x}}$ equals to zero or x_i of $\mathbf{x}$. For clarity, the letter $\mathbf{x}$ also represents a random variable (feature vector). Given a data set $\mathbf{X} \in \mathcal{R}^{n\times p}$ consisting of n data points, i.e., $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{n}$, each point $\mathbf{x}_i \in \mathcal{R}^p$ is represented by a p-dimensional feature vector.

The hashing problem has a similar representation to matrix factorization [51]. Mathematically, the data matrix $\mathbf{X} \in \mathcal{R}^{n\times p}$ can be approximately reconstructed as follows:

$$\begin{aligned}&\arg\min \frac{1}{2}\|\mathbf{X} - \mathbf{U}\mathbf{D}\mathbf{V}^T\|_2^2\\ &\text{s.t. } \mathbf{U}^T\mathbf{U} = \mathbf{I}, \quad \mathbf{V}^T\mathbf{V} = \mathbf{I}\end{aligned} \tag{1}$$

where $\mathbf{D} = diag(\delta_1, \delta_2, \ldots, \delta_k)$ is a diagonal matrix of singular values, such that $\delta_1 \geq \delta_2 \geq \cdots \geq \delta_k$, and $\mathbf{U}$ and $\mathbf{V}$ are the left and right singular matrices corresponding to the k singular values, respectively. Considering a sequential way, (1) can be equivalently reorganized as

$$\begin{aligned}&\arg\max_{\mathbf{u},\mathbf{v}} \mathbf{u}^T\mathbf{X}\mathbf{v}\\ &\text{s.t. } \mathbf{u}^T\mathbf{u} = 1, \quad \mathbf{v}^T\mathbf{v} = 1.\end{aligned} \tag{2}$$

B. Adaptively Sparse Matrix Factorization

Most conventional matrix factorization techniques focus only on the aspect of computational efficiency rather than the physical meanings of the constructed models. In many real-world domains, such as medical diagnosis, business decision-making, and information retrieval, the interpretability of the derived models plays a critical role because it can help users understand the models better and make right decisions. However, as the dimensionality of the data increases, interpreting the derived models becomes challenging and even impossible in some cases.

Imposing regularization terms on the optimization problem of (2) as follows seems to be a feasible solution to alleviate the above-mentioned problem:

$$\begin{aligned}&\arg\max_{\mathbf{u},\mathbf{v},\Theta} \mathbf{u}^T\mathbf{X}\mathbf{v} + g(\mathbf{u}, \mathbf{v}, \Theta)\\ &\text{s.t. } \mathbf{u}^T\mathbf{u} = 1, \quad \mathbf{v}^T\mathbf{v} = 1\end{aligned} \tag{3}$$

where $g(\mathbf{u}, \mathbf{v}, \Theta)$ is a regularization function on $\mathbf{u}$ and $\mathbf{v}$, and Θ is a set of regularization factors. Typical regularization terms include the ℓ_0-norm, ℓ_1-norm (also known as Lasso), ℓ_2-norm, and their variants. Note that these norms have different characteristics. For example, the ℓ_0-norm can induce sparse models, but its solution is often nonconvex, and the computational cost is relatively high, making it impracticable even in middle-scale problems. Contrastively, the ℓ_2-norm has good statistical properties, while the derived models tend to be overfitting [52].

The ℓ_1-norm offers an effective solution for improving the interpretability of the derived models. It enforces those small coefficients of the derived results to zero via the regularization functions, yielding parsimonious structures for the models. Assume that $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ are the corresponding parsimonious structures of $\mathbf{u}$ and $\mathbf{v}$, respectively. The sparse variant of (2) can be formally represented as

$$\begin{aligned}&\arg\max_{\bar{\mathbf{u}},\bar{\mathbf{v}}} \bar{\mathbf{u}}^T\mathbf{X}\bar{\mathbf{v}}\\ &\text{s.t. } \bar{\mathbf{u}}^T\bar{\mathbf{u}} = 1, \quad \bar{\mathbf{v}}^T\bar{\mathbf{v}} = 1\end{aligned} \tag{4}$$

where $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ can be obtained by using sparse-inducing functions, such as smoothly clipped absolute deviation (SCAD), Lasso, adaptive Lasso, and fused Lasso [52]. As an example, the soft-thresholding function enforces $\bar{u}_i$ (or $\bar{v}_i$) to zero if the corresponding absolute value u_i (or v_i) is less than λ, a regularization parameter, or $u_i - \lambda$ (or $v_i - \lambda$) or $u_i + \lambda$ (or $v_i + \lambda$), depending on the sign of u_i (or v_i).

How to determine appropriate values for the regularization parameters, i.e., Θ in (3), in sparse learning is still an open and challenging issue. As a matter of fact, the regularization parameters are data driven and should be carefully tuned according to the available data. Generally, empirically assigning appropriate values with a cross-validation manner or considering prior knowledge for the regularization parameters are the two commonly used strategies. However, their computational costs are relatively high. The worse thing is that the derived models with the optimal values incline to be overfitting, that is, the models with the optimal parameters may perform well in one case but poorly in other cases.

Here, we resort to a heuristic strategy to approximately obtain $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ by exploiting the inherent properties of $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ to their counterparts without involving the regularization parameters Θ. Let us revisit (4). It is a suboptimization problem of (2) to some extent if no parsimonious structures are considered, i.e., $\bar{\mathbf{u}} = \mathbf{u}$ and $\bar{\mathbf{v}} = \mathbf{v}$. Therefore, $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ can be derived from $\mathbf{u}$ and $\mathbf{v}$, respectively, only when (4) holds. Observing from this property, we have the fact that there are two kinds of intrinsic relationships between the parsimonious structures, i.e., intrarelation and interrelation.

The intrarelation refers to the fact that $\bar{\mathbf{u}}$ (or $\bar{\mathbf{v}}$) is an elementwise one-to-one projection of $\mathbf{u}$ (or $\mathbf{v}$, respectively), while the interrelation implies that $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ should be positively correlated with each other with respect to $\mathbf{X}$, such that $\bar{\mathbf{u}}^T\mathbf{X}\bar{\mathbf{v}}$ is maximal. Inspired by the idea of the hard-thresholding function, both $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ can be similarly derived from $\mathbf{u}$ and $\mathbf{v}$, respectively.

First, let us turn our focus on the intrarelation of $\bar{\mathbf{u}}$ (or $\bar{\mathbf{v}}$) to $\mathbf{u}$ (or $\mathbf{v}$, respectively). For any two random vectors $\mathbf{x}\in \mathcal{R}^n$ and $\mathbf{y}\in \mathcal{R}^n$, we have the following property.

Lemma 1: Given two vectors $\mathbf{x}\in \mathcal{R}^n$ and $\mathbf{y}\in \mathcal{R}^n$ with the same number of elements, $\mathbf{x}^T\mathbf{y}$ is maximal if their corresponding elements have the same signs, i.e., $sgn(x_i) = sgn(y_i)$.

Proof: This lemma can be proved easily in an intuitive manner. Let n be the number of elements within $\mathbf{x}$ and $\mathbf{y}$. For any $i = 1 \ldots n$, $x_i y_i \geq 0$ if both x_i and y_i have the same signs. Thus, $\Sigma_{i=1\ldots n} x_i y_i$ is maximal. Otherwise, let the signs of x_k and y_k be different, and we have $x_i y_i \leq 0$. In this case, $\mathbf{x}^T\mathbf{y}$ is not maximal because $\Sigma_{i=1\ldots n} x_i y_i \leq \Sigma_{i=1\ldots k-1,k+1\ldots n} x_i y_i$. ■

The lemma enlightens us to make a vector sparse straightforwardly. Based on the lemma above, a parsimonious structure $\bar{\mathbf{x}}$ of the vector $\mathbf{x}$ can be obtained through the following lemma.

Lemma 2: Given a vector $\mathbf{x}\in \mathcal{R}^n$ and a reference vector $\mathbf{y}\in \mathcal{R}^n$, $\bar{\mathbf{x}} \sqsubseteq \mathbf{x}$ is a parsimonious structure of $\mathbf{x}$, such that $\bar{\mathbf{x}}^T\mathbf{y}$ is the maximal value of $\mathbf{x}^T\mathbf{y}$ if

$$\bar{x}_i = \begin{cases} x_i, & \text{if } x_i y_i \geq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

Proof: It can be easily observed that $\bar{\mathbf{x}}$ is a parsimonious structure of $\mathbf{x}$, i.e., $\bar{\mathbf{x}} \sqsubseteq \mathbf{x}$, because for each element $\bar{x}_i \in \bar{\mathbf{x}}$ $(i = 1, \ldots, n)$, its value equals to the corresponding element $x_i \in \mathbf{x}$ if it has the same sign to the reference element y_i, otherwise $\bar{x}_i = 0$.

Let us prove the maximal sum problem. Assume that $\mathcal{I}_t \subseteq \{1, \ldots, t\}$ is the index subset from one to t, and S_t is the maximal subsum of $\mathbf{x}_{\mathcal{I}_t}$ and $\mathbf{y}_{\mathcal{I}_t}$ from the first to the tth element, i.e., $S_t = \arg\max_{\mathcal{I}_t}(\mathbf{x}_{\mathcal{I}_t}^T\mathbf{y}_{\mathcal{I}_t})$. For the first element $x_1 \in \mathbf{x}$, we have $\bar{x}_1 = x_1$ if $x_1 y_1 \geq 0$, otherwise $\bar{x}_1 = 0$ according to the definition. Thus, $S_1 = \bar{x}_1 y_1 \geq x_1 y_1$ and $\mathcal{I}_1 = \{1\}$.

Assume that we have $\mathcal{I}_{t-1} \subseteq \{1 \ldots t-1\}$ and $S_{t-1} = \bar{\mathbf{x}}^T\mathbf{y} = \arg\max_{\mathcal{I}_{t-1}}(\mathbf{x}_{\mathcal{I}_{t-1}}^T\mathbf{y}_{\mathcal{I}_{t-1}})$. According to the definition, we can observe that $S_t = \max(S_{t-1}, \mathbf{x}_{1\ldots t}^T\mathbf{y}_{1\ldots t}) \leq S_{t-1} + x_t y_t$. Therefore, if x_t has the same sign to y_t, it should be preserved, i.e., $\bar{x}_t = x_t$, as it can increase the maximal sum S_{t-1}. On the contrary, x_t will decrease S_{t-1} when $x_t y_t \leq 0$. In this case, $\bar{x}_t = 0$ will not affect the fact that S_{t-1} is maximal. Similarly, we can handle the other element $\bar{x}_i$ and, ultimately, generate the parsimonious vector $\bar{\mathbf{x}}$ of $\mathbf{x}$, such that $\bar{\mathbf{x}}^T\mathbf{y} = \arg\max_{\mathcal{I}_n} \mathbf{x}^T\mathbf{y}$. ■

The lemma provides an intuitive description of the intrarelation of vectors and offers us a good clue to generate a parsimonious structure for any vector. It should be noted that the lemma requires both the vector and its reference to have the same number of elements. This implies that we cannot straightforwardly apply the lemma to generate the parsimonious structure $\bar{\mathbf{u}}$ (or $\bar{\mathbf{v}}$) of $\mathbf{u}$ (or $\mathbf{v}$, respectively) because the number of elements within $\mathbf{u}$ is usually different from that of $\mathbf{v}$, i.e., $n \neq p$. Fortunately, we can make the vectors sparse with the help of the interrelation between $\mathbf{u}$ and $\mathbf{v}$.

From the aforementioned statement, the interrelation of two vectors indicates that they are positively correlated with each other. According to the lemma, one may observe that the derived parsimonious vector (e.g., $\bar{\mathbf{u}}$ or $\bar{\mathbf{v}}$) is the most positive correlated vector for the corresponding reference vector (e.g., $\mathbf{u}$ or $\mathbf{v}$, respectively). Indeed, the maximal subsum of two vectors can be considered as a measurement of their positive correlation to some extent. More importantly, according to the optimization problems of (2) and (4), we have the fact that $\mathbf{u}$ and $\mathbf{v}$ derived from (2) are correlated with each other with respect to $\mathbf{X}$. Meanwhile, $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ derived from (4) should also be correlated with respect to $\mathbf{X}$, such that $\bar{\mathbf{u}}^T\mathbf{X}\bar{\mathbf{v}}$ is maximal.

Based on the interrelations, we can seek an alternating way to approximately obtain $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$, where (5) is the basis of our sparse operations. Before making the vector (e.g., $\mathbf{u}$) sparse, its corresponding reference vector with the same length should be given in advance. Thus, it is natural to take the data matrix $\mathbf{X}$ into consideration when generating the parsimonious structures. Specifically, $\mathbf{Xv}$ will be considered as the reference vector of $\mathbf{u}$ when making it sparse because they should be positively correlated and have the same length. In a similar way, once the sparse vector $\bar{\mathbf{u}}$ is available, it, together with $\mathbf{X}$, can also be used as the reference vector (i.e., $\mathbf{X}^T\bar{\mathbf{u}}$) of $\mathbf{v}$ to generate the corresponding sparse vector $\bar{\mathbf{v}}$.

C. Binary Coding With Rotation

With the sparse vectors $\bar{\mathbf{V}} = [\bar{\mathbf{v}}_1, \bar{\mathbf{v}}_2, \ldots, \bar{\mathbf{v}}_k]$ available, $\mathbf{X}$ can be represented as $\mathbf{X}\bar{\mathbf{V}}$ after projected into $\bar{\mathbf{V}}$. In this context, the general schema of hashing learning can be formally organized as

$$\mathcal{L}(\mathbf{B}, \mathbf{R}) = \frac{1}{2}\|\mathbf{B} - \mathbf{MR}\|_2^2 \tag{6}$$

where $\mathbf{M} = \mathbf{X}\bar{\mathbf{V}}$ and $\mathbf{R}$ is an orthogonal matrix with $k \times k$. Thus, minimizing the quantization loss can preserve the original local structure of the data. The binary matrix $\mathbf{B}$ and the rotation matrix $\mathbf{R}$ can be obtained with alternative iterations. Specifically, in each iteration, we first obtain $\mathbf{R}$ with $\mathbf{B}$ fixed. Once $\mathbf{R}$ is available, we can solve $\mathbf{B}$ alternatively.

*1) Update **R** With **B** Fixed:* It is noticeable that the formulation of (6) is the orthogonal procrustes problem when $\mathbf{B}$ is fixed. Thus, the objective function $\mathcal{L}(\mathbf{R})$, which tries to seek the most optimal rotation $\mathbf{R}$, can be solved by performing partial derivative operation in a routine manner. Unfortunately, it is time consuming, and the matrix inverse may be unavailable. Alternatively, here, we resort to matrix factorization to obtain $\mathbf{R}$. Let $\mathbf{B}^T\mathbf{M} = \mathbf{UDV}^T$, and (6) can be rewritten as

$$\begin{aligned}\mathbf{R} &= \arg\min_{\bar{\mathbf{R}}} \frac{1}{2}\|\mathbf{B} - \mathbf{M}\bar{\mathbf{R}}\|_2^2 \\ &= \arg\max_{\bar{\mathbf{R}}} \ \operatorname{tr}(\mathbf{B}^T\mathbf{M}\bar{\mathbf{R}})\end{aligned}$$

Algorithm 1 Adaptively SMFH

```
Input: The data $\mathbf{X} \in \mathcal{R}^{n\times p}$ and the length of hash codes $k$;
Output: The generated binary codes $\mathbf{B} \in \{-1, 1\}^{n\times k}$;
1:  Normalize $\mathbf{X}$, and $\mathbf{Y}$=$\mathbf{X}$;
2:  For $i = 1,\ldots,k$ do
3:      Get $\mathbf{u}$ and $\mathbf{v}$ from $\mathbf{Y}$ via matrix factorization;
4:      Make $\mathbf{u}$ and $\mathbf{v}$ sparse according to Eq.(4) in Lemma 2;
5:      $\mathbf{U}$=$\mathbf{U}\cup\{\mathbf{u}\}$ and $\mathbf{V}$=$\mathbf{V}\cup\{\mathbf{v}\}$;
6:      Update $\mathbf{Y}$ as $\mathbf{Y} \leftarrow \mathbf{Y} - tr(\mathbf{Y}\mathbf{v}\mathbf{u}^T)\mathbf{u}\mathbf{v}^T$;
7:  End
8:  $\mathbf{M}$=$\mathbf{X}\mathbf{V}$;
9:  Obtain the optimized rotation $\mathbf{R}$ according to Eq.(8);
10: $\mathbf{B}$=sign($\mathbf{M}\mathbf{R}$);
11: Return $\mathbf{B}$ as the binary codes.
```

$$\begin{aligned} &= \arg\max_{\bar{\mathbf{R}}}\ \mathrm{tr}(\mathbf{U}\mathbf{D}\mathbf{V}^T\bar{\mathbf{R}}) \\ &= \arg\max_{\bar{\mathbf{R}}}\ \mathrm{tr}(\mathbf{D}\mathbf{V}^T\bar{\mathbf{R}}\mathbf{U}). \end{aligned} \tag{7}$$

Since $\mathbf{V}^T\bar{\mathbf{R}}\mathbf{U}$ is orthogonal, the above-mentioned formulation is maximized when it is the identity matrix $\mathbf{I}$, i.e., $\mathbf{V}^T\mathbf{R}\mathbf{U} = \mathbf{I}$. Thus

$$\mathbf{R} = \mathbf{V}\mathbf{U}^T. \tag{8}$$

*2) Update **B** With **R** Fixed:* Once the rotation matrix $\mathbf{R}$ is available, $\mathbf{M}\mathbf{R}$ is fixed because $\mathbf{M} = \mathbf{X}\mathbf{V}$. Let $\tilde{\mathbf{M}} = \mathbf{M}\mathbf{R}$. In this case, (6) can be rewritten as

$$\begin{aligned} \mathcal{L}(\mathbf{B}) &= \frac{1}{2}\|\mathbf{B} - \tilde{\mathbf{M}}\|_2^2 \\ &= \frac{1}{2}(\mathrm{tr}(\mathbf{B}^T\mathbf{B}) + \mathrm{tr}(\tilde{\mathbf{M}}^T\tilde{\mathbf{M}})) - \mathrm{tr}(\mathbf{B}\tilde{\mathbf{M}}^T). \end{aligned} \tag{9}$$

As stated earlier, each element $b_{ij} \in \mathbf{B}$ is either -1 or 1, i.e., $b_{ij} \in \{-1, 1\}$. Thus, the first two terms of the above-mentioned formulation are constant because $\tilde{\mathbf{M}}$ is also fixed. Besides, for the last term, we have $\mathrm{tr}(\mathbf{B}\tilde{\mathbf{M}}^T) = \sum_{ij} b_{ij}\tilde{m}_{ji}$. This implies that minimizing $\mathcal{L}(\mathbf{B})$ is equivalent to maximize $\sum_{ij} b_{ij}\tilde{m}_{ji}$. To achieve this purpose, each element b_{ij} should have the same sign as that of $\tilde{m}_{ji}$, i.e., $b_{ij} = 1$ if $\tilde{m}_{ji} \geq 0$ and -1 otherwise.

D. Algorithm Implementation

Based on the above-mentioned analysis, the implemented details of our hashing learning method are summarized in Algorithm 1. The schema comprises two stages: sparse decomposition (from Steps 1 to 7) and binary encoding (from Steps 8 to 10). The algorithm can be easily understood. First, the original data are normalized to zero mean and unit variance before performing the decomposition operation (Step 3); notwithstanding, scaling does not affect the optimization model. Then, the k sparse projection vectors are derived from the data in a sequential manner. At the beginning of each iteration, an optimized projection direction of the data is obtained by nonlinear iterative partial least squares (NIPALS), which is a classic matrix factorization technique. The reason for adopting NIPALS here is that it has higher efficiency than SVD in computing approximately optimal values that are enough for deriving the sparse model. It should be noted that $\mathbf{u}$ and $\mathbf{v}$ can be initially assigned any values, except zeros, as NIPALS can achieve their optimal values. Subsequently, the parsimonious structures of $\mathbf{u}$ and $\mathbf{v}$ can be obtained according to the principle of Lemma 2. After each sparse projection vector has been derived, its encoding information should be removed from the data (see Step 6). Otherwise, similar vectors would be made, resulting in a useless model. Finally, the sparse projection vectors are rotated to seek projection directions, which can help generate optimal and compact binary codes.

Suppose that there are n data points with p dimensions for the training data $\mathbf{X}$ and the number of bits for the final binary codes is k. The computational complexity of SMFH during the training stage is predominated by four steps: NIPALS (Step 3), sparse operation (Step 4), updating data (Step 6), and rotation operation (Step 9). As we know, the time complexity of NIPALS is $O(cnp)$, where c is the number of iterations toward convergence. Generally, it is normally assigned as a constant (e.g., 20 in our later experiments). Note that both the sparse operation and the data updating are scaling linearly with respect to the size of $\mathbf{X}$. For the rotation operation, $O(np^2 + p^3)$ time is required. Thus, the overall computational complexity of SMFH is $O(kcnp + np^2 + p^3)$. Given a query point, it needs $O(p)$ to encode the query point to a k-bits binary code during the test stage.

IV. Evaluation Experiments

In this section, we elaborate extensive evaluation experiments to show the effectiveness of our proposed approach. Initially, we present experimental settings, including experimental data sets, evaluation metrics, and baselines. Then, the comparison results of SMFH with the state-of-the-art hashing algorithms are given. The simulation experiments were conducted on a PC with a 4.0-GHz Intel Core i7-6700K CPU with 8 GB of main memory.

A. Data Sets and Settings

To make a comprehensive comparison, we adopted four publicly available image data sets, including CALTECH265, CorelDB, GHIM10K, and INRIA Holidays, to verify the effectiveness of the proposed method. These four data sets cover different scales and are widely used to evaluate hashing methods in the literature. The brief descriptions of the experimental data sets are provided as follows.

1) *CALTECH256*[1]*:* It consists of 29 780 color images in 256 categories (over 80 images per category), such as flag, backpack, bear, beer-mug, butterfly, and calculator. Each image belongs to one of the 256 categories.
2) *CorelDB*[2]*:* It contains 10 800 color images from the Corel Photo Gallery associating with 80 concept groups, e.g., autumn, castle, cloud, dog, elephant, ship, tiger, and waterfall; each group has more than 100 images.

[1]http://www.vision.caltech.edu/Image_Datasets/Caltech256

[2]https://sites.google.com/site/dctresearch/Home/content-based-image-retrieval

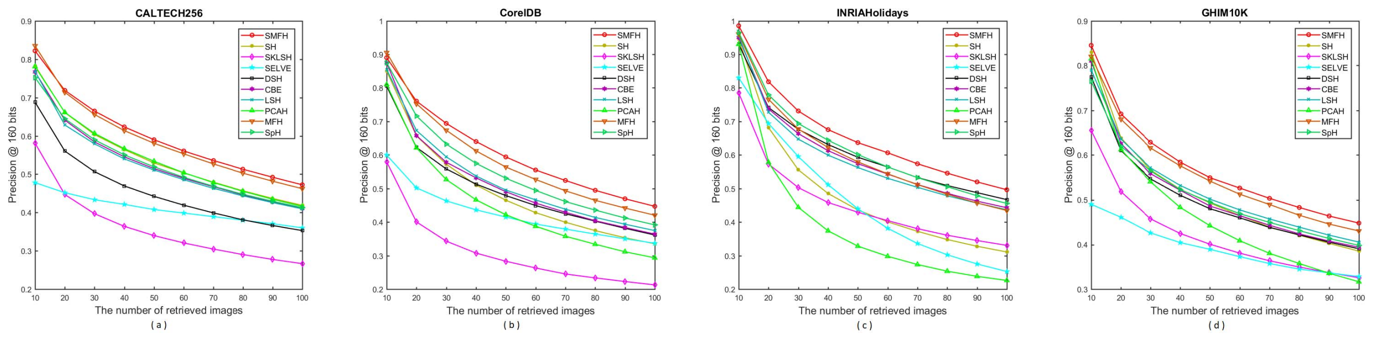

Fig. 2. Precision curves of the generated binary codes with 160 bits on the data sets (the number of retrieved images ranges from 10 to 100). (a) CALTECH256. (b) CorelDB. (c) INRIA Holidays. (d) GHIM10K.

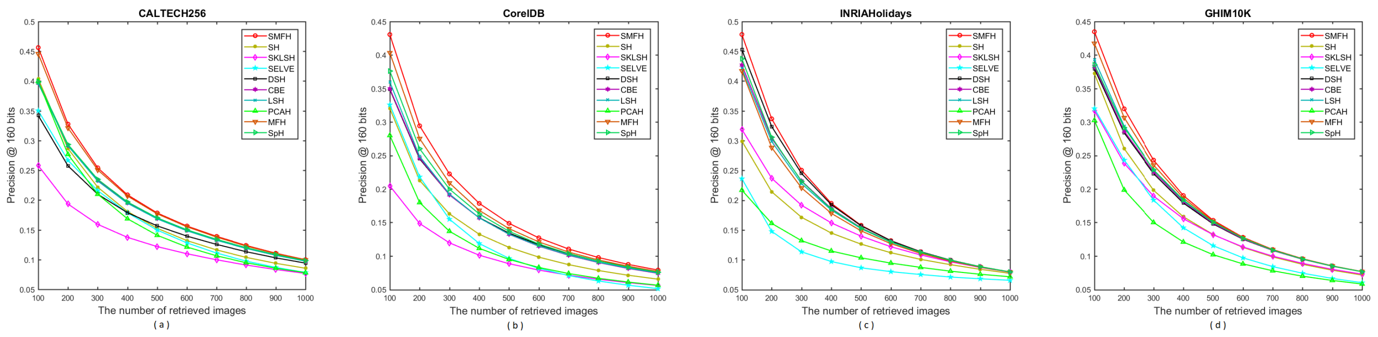

Fig. 3. Precision curves of the generated binary codes with 160 bits on the data sets (the number of retrieved images ranges from 100 to 1000). (a) CALTECH256. (b) CorelDB. (c) INRIA Holidays. (d) GHIM10K.

3) *GHIM10K*[3]*:* It includes 10 000 color images (300 $\times$ 400) collected from the web and cameras. They are annotated by 20 categories covering diverse contents, such as sunset, ship, flower, building, car, mountains, and insect. Each category has 500 images.
4) *INRIA Holidays*[4]*:* It is a photo collection of personal holidays. It has 1491 images in total with a large variety of scene types, covering 500 scenes, such as natural, water, mountain, building, sea, waterfall, tree, and sunset.

Recent studies show that the convolutional neural network (CNN), which provides an abstract and holistic representation of an image via a set of perceptual features, has received increasing attention in image retrieval and scene recognition. In our experiments, we also utilized a fully connected CNN to extract representation features from each image by activating the last layer of the network, and each image is represented with 1024 CNN features.

Following the experimental settings in recent work, we randomly selected 1000 images to serve as query points, and the remaining data served as training samples for each data set. During the whole experiments, we simply took the class label of each query as the ground truth if it was annotated. Otherwise, the ground truth was defined as the average Euclidean distance of the 50th NN. In this way, a returned point for a given query was taken as a true neighbor if it was close enough to the query. The experiments were repeated ten times, and the average performance over all runs was reported in this article.

[3]http://www.ci.gxnu.edu.cn/cbir/Dataset.aspx
[4]http://lear.inrialpes.fr/ jegou/data.php

B. Baselines and Metrics

We compared the proposed method to nine popular hashing algorithms, including CBE [20], DSH [21], LSH [14], MFH [48], PCAH [17], SELVE [23], SH [18], SKLSH [16], and SpH [22]. As discussed earlier, these algorithms stand for different kinds of hashing learning techniques.

1) *CBE [20]:* It projects the data with a circulant matrix and then uses the technique of the fast Fourier transformation to improve the encoding efficiency of the binary codes.
2) *DSH [21]:* It exploits geometric structures, i.e., densities, of the data to determine the projection vectors rather than purely random selection in LSH.
3) *LSH [14]:* It generates the projection vectors by randomly sampling from a Gaussian distribution matrix, in the context of preserving the locality with high probability.
4) *MFH [48]:* It applies collective matrix factorization to learn latent factor models that are subsequently used to produce unified binary codes.
5) *PCAH [17]:* It takes the top principal directions derived by PCA as the projective vectors to encode the binary codes to preserve the most variance of the data.
6) *SELVE [23]:* It learns the binary codes from the coefficients of the least variance encoding model that is built on sparse embedding vectors derived by spectral clustering.

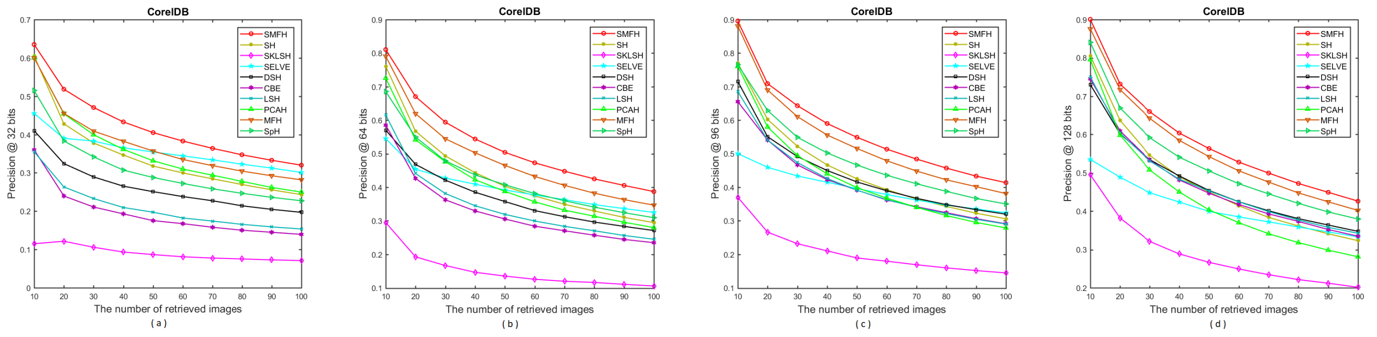

Fig. 4. Precision curves on the CorelDB data set. (a)–(d) Performances of the generated binary codes with 32, 64, 96, and 128 b, respectively.

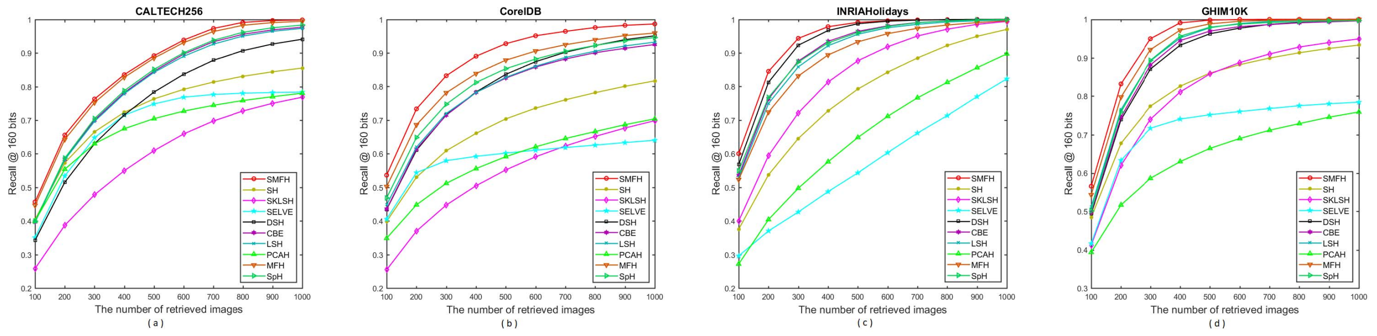

Fig. 5. Recall curves of the generated binary codes with 160 b on the different data sets. (a) CALTECH256. (b) CorelDB. (c) INRIA Holidays. (d) GHIM10K.

7) *SH [18]:* It constructs a Laplacian matrix with spectral embedding and then performs eigenvalue decomposition operation to quantize the binary codes.
8) *SpH [22]:* It projects the spatially coherent data into spherical planes and generates binary codes by iteratively adjusting the spherical planes with the spherical Hamming distances.
9) *SKLSH [16]:* It derives the binary codes by random projections with distribution-free encoding for approximating shift-invariant kernels. In our experiments, a Gaussian kernel was utilized, and its bandwidth was set to the average distance to the fifth NN.

To make a fair comparison, we adopted four commonly used metrics, top-k precision (Pre@k), top-k recall (Recall@k), precision–recall (PR) curve, and mean average precision (mAP), to quantitatively evaluate the performance of the hashing learning algorithms for retrieval tasks. Their definitions are briefly given as follows.

1) *Pre@k:* This refers to the precision of the top-k retrieval images, that is, it is the ratio of the relevant images to the retrieved top-k images in terms of the Hamming distance.
2) *Rec@k:* It is the ratio of the retrieved relevant images to all the relevant images in the retrieved top-k images in terms of the Hamming distance.
3) *PR:* It denotes the retrieval accuracies at different recall levels. The area under the PR curve is good at evaluating the overall performance in information retrieval.
4) *mAP:* It is the average precision at the retrieval ranks where the recall changes. Specifically, it represented as

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{M}\sum_{t=1}^{M}P_i(t)I_i(t) \qquad (10)$$

where N is the number of query points, M is the number of retrieved images, $P_i(t)$ is the top-i precision of the tth query, and $I_i(t)$ indicates whether the ith images is relevant to the tth query.

For the above-mentioned metrics, the larger the value, the better the retrieval performance.

C. Results and Discussion

To test the effectiveness of SMFH, a series of comparison experiments with popular hashing algorithms were carried out, in which the number of bits of the generated binary codes is varied from 16 to 160. We first compared the hashing algorithms from the perspective of precision, which is a fundamental criterion in information retrieval. The experimental results are shown in Figs. 2 and 3, where the length of the generated binary codes was fixed to 160 b. As illustrated in Figs. 2 and 3, we can observe that the precision of the hashing algorithms decreased as the number of retrieved images increased. This, however, is reasonable because the precision is sensitive to true positive images of the query. In addition, the proposed method, SMFH, outperformed the baselines in most cases. The most comparable hashing algorithm was MFH that also adopts matrix factorization to generate the binary codes. On the INRIA Holidays data set [see Fig. 3(c)], DSH also achieved good performance. The reason is that each image within this data set has less than five truly similar images.

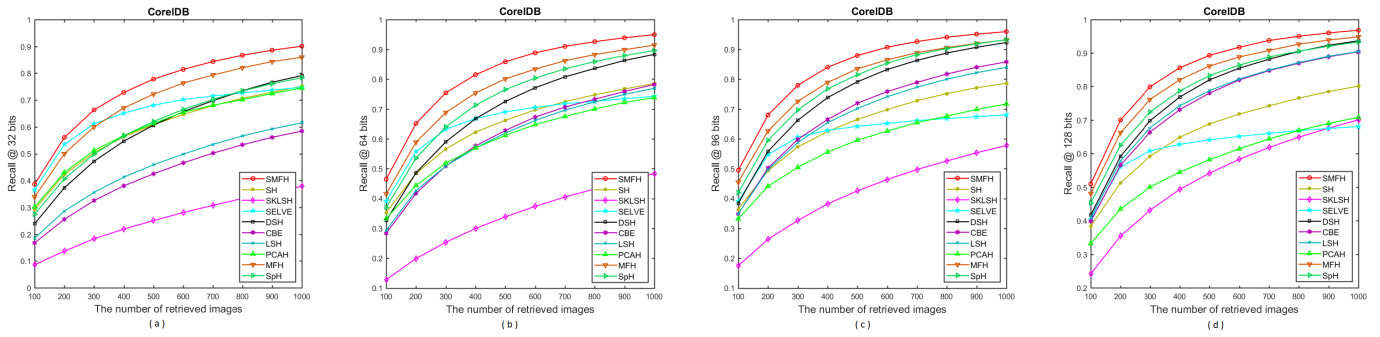

Fig. 6. Recall curves on the CorelDB data set. (a)–(d) Performances of the generated binary codes with 32, 64, 96, and 128 b, respectively.

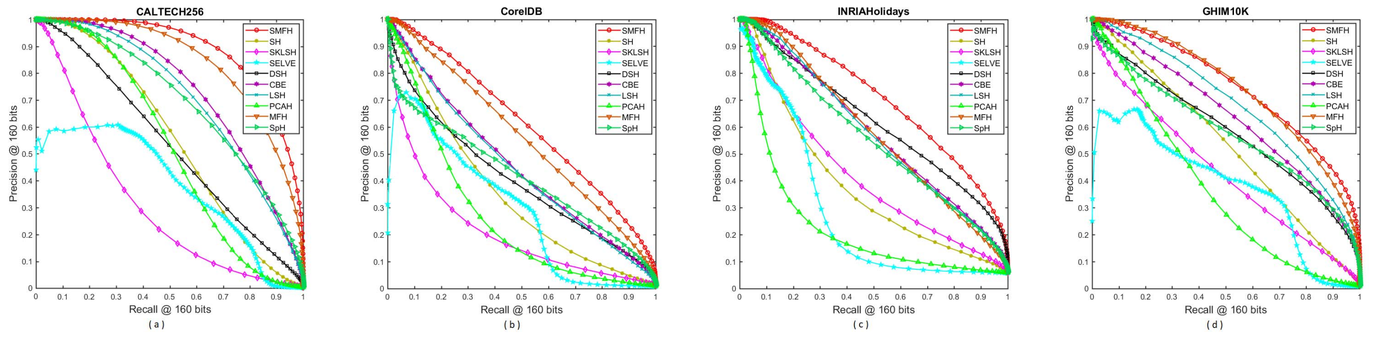

Fig. 7. PR curves of the generated binary codes with 160 b on the different data sets. (a) CALTECH256. (b) CorelDB. (c) INRIA Holidays. (d) GHIM10K.

For different bits of the generated binary codes, SMFH also exhibited higher precision than the others. Fig. 4 presents the precision comparison of the generated binary codes with different lengths on the CorelDB data set. From the plots shown in Fig. 4, one can safely conclude that SMFH was superior to the popular hashing methods and derived binary codes with high quality. Besides, the plots shown in Fig. 4(a)–(d) indicate that the performance decreased as the number of retrieved images increased, notwithstanding the different lengths of the generated binary codes. Another interesting fact is that for the same number of the retrieved images, the precision rates of the hashing methods with more bits of the binary codes were higher than those with fewer bits. This is consistent with our expectation, that is, the binary codes with more bits encoded more information. The same cases can be found on the rest of the data sets. For space limitations, we have not provided all of them here.

The recall criterion is often used to indicate the successfully retrieved ratio of the relevant images in image retrieval. There is no exception in our comparisons. We also took it as one of the criteria to evaluate the performance of the hashing algorithms. The performance comparisons are illustrated in Figs. 5 and 6. Similar to the precision criterion, the same situations can be found for the recall criterion. For example, SMFH played a predominant role in encoding the binary codes on these four data sets. MFH was a comparable hashing method because both SMFH and MFH use the technique of matrix factorization. The difference is that SMFH exploits an adaptively sparse strategy without involving regularization parameters. However, MFH was relatively worse than DSH and SpH on the INRIA Holidays data set. In addition, for the same number of retrieved images (see Fig. 6), the hashing algorithms with more bits of binary codes had higher recall than the corresponding algorithms with fewer bits.

The precision and recall metrics represent positive prediction and sensitivity, respectively. They stand for different sides for retrieval tasks. However, a good retrieval algorithm should have both high precision and high recall. To compromise this issue, the PR curve is introduced, which measures the tradeoff relationship between precision and recall for possible cutoffs. A large area under the curve indicates a low false-positive rate and false-negative rate. Fig. 7 gives the PR curves of the binary codes with 160 b generated by the hashing algorithms on the four data sets. As the PR curves show, the variation trends of precision and recall are opposite, that is, the precision decreases smoothly as the recall increased. From the perspective of the area under the curve, SMFH was better than the others because the curve of SMFH was higher than those of the competing algorithms. This implies that SMFH retrieved more images similar to the query at the top of the retrieved images. On the GHIM10K data set, MFH also achieved comparable performance when the length of the binary codes was fixed to 160. However, with fewer bits (see Fig. 8), MFH was worse than SMFH. This shows that SMFH is good at deriving more compact binary codes than the other methods.

The mAP measurement can be used to comprehensively evaluate retrieval performance. We also employed it as a criterion to verify the effectiveness of SMFH in our experiments. The experimental results of the binary codes generated by the hashing algorithms are provided in Fig. 9, where the lengths of the binary codes were varied from 16 to 160. From the mAP scores, we know that most of the plots corresponding

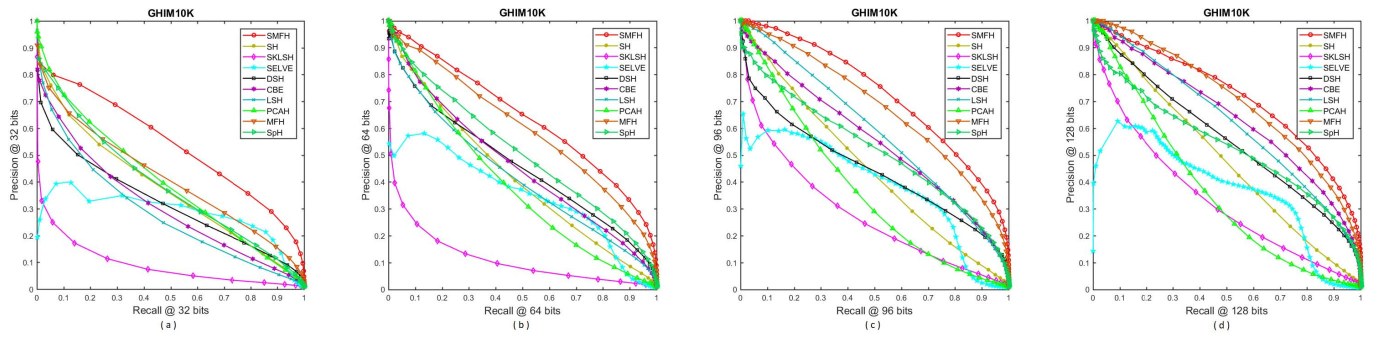

Fig. 8. PR curves of the generated binary codes with different bits on the GHIM10K data set. (a)–(d) Performances of the binary codes with 32, 64, 96, and 128 b, respectively.

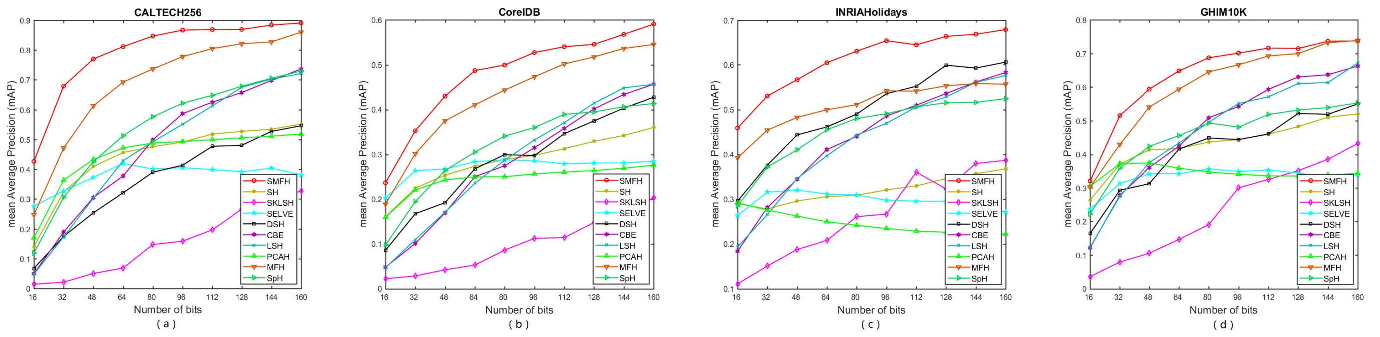

Fig. 9. mAP scores of the generated binary codes on the different data sets. (a) CALTECH256. (b) CorelDB. (c) INRIA Holidays. (d) GHIM10K.

to the hashing algorithms were similar, that is, increasing the length of the binary codes led to higher mAP scores. PCAH and SELVE seemed to be the exceptions on INRIA Holidays, as there were too many categories in this data set and each category contained very few similar images. It should be mentioned that SMFH achieved very promising performance with different code bits on all data sets, and MFH was still comparative. This indicates that the matrix factorization can effectively explore the relationships of the data to some extent. In contrast, SKLSH was less stable and had poor performance on all data sets. The reason is that SKLSH did not perform dimension reduction, resulting in poor performance for the high-dimensional data.

V. Conclusion

In this article, we propose a novel hashing method that exploits sparse matrix factorization to capture the latent structures of data. Unlike traditional sparse learning techniques that require explicitly choosing appropriate sparsity-inducing functions and carefully tuning the corresponding regularization parameters, our method is adaptive and parameter-free, without involving the troublesome issue of parameter tuning. In addition, an orthogonal transformation is also considered to minimize the quantization loss while deriving the binary codes. As a result, the binary codes generated by the proposed method are more compact and have more semantic properties. Extensive experiments on four widely used benchmark image data sets were carried out. The experimental results demonstrated the promising performance of the proposed method compared to the state-of-the-art hashing methods.

Acknowledgment

The authors would like to thank the anonymous referees and the associate editor for their valuable comments and suggestions and their help in significantly improving this article.

References

[1] J. Wang, W. Liu, S. Kumar, and S. F. Chang, "Learning to hash for indexing big data: A survey," *Proc. IEEE*, vol. 104, no. 1, pp. 34–57, Jan. 2016.
[2] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient kNN classification with different numbers of nearest neighbors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1774–1785, May 2018.
[3] X. Fang, N. Han, W. K. Wong, S. Teng, J. Wu, S. Xie, and X. Li, "Flexible affinity matrix learning for unsupervised and semisupervised classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 1133–1149, Apr. 2019.
[4] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 769–790, Apr. 2018.
[5] L. Liu and L. Shao, "Sequential compact code learning for unsupervised image hashing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2526–2536, Dec. 2016.
[6] Q. Wang, Z. Qin, F. Nie, and X. Li, "Spectral embedded adaptive neighbors clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 4, pp. 1265–1271, Apr. 2019.
[7] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.
[8] L. Chi and X. Zhu, "Hashing techniques: A survey and taxonomy," *ACM Comput. Surv.*, vol. 50, no. 1, Apr. 2017, Art. no. 11.
[9] D. Cai, "A revisit of hashing algorithms for approximate nearest neighbor search," May 2018, *arXiv:1612.07545*. [Online]. Available: https://arxiv.org/abs/1612.07545
[10] M. Wang, W. Zhou, Q. Tian, and H. Li, "A general framework for linear distance preserving hashing," *IEEE Trans. Image Process.*, vol. 27, no. 2, pp. 907–922, Feb. 2018.

[11] Y. Huang and Z. Lin, “Binary multidimensional scaling for hashing,” *IEEE Trans. Image Process.*, vol. 27, no. 1, pp. 406–418, Jan. 2018.
[12] X. Lu, X. Zheng, and X. Li, “Latent semantic minimal hashing for image retrieval,” *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 355–368, Jan. 2017.
[13] J. Tang *et al.*, “Discriminative deep quantization hashing for face image retrieval,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 6154–6162, Dec. 2018.
[14] A. Gionis, P. Indyk, and R. Motwani, “Similarity search in high dimensions via hashing,” in *Proc. VLDB*, 1999, pp. 518–529.
[15] B. Kulis and K. Grauman, “Kernelized locality-sensitive hashing for scalable image search,” in *Proc. ICCV*, 2009, pp. 2130–2137.
[16] M. Raginsky and S. Lazebnik, “Locality-sensitive binary codes from shift-invariant kernels,” in *Proc. NIPS*, 2009, pp. 1509–1517.
[17] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, “Aggregating local image descriptors into compact codes,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1704–1716, Sep. 2012.
[18] Y. Weiss, A. Torralba, and R. Fergus, “Spectral hashing,” in *Proc. NIPS*, 2008, pp. 1753–1760.
[19] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, “Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.
[20] X. Yu, A. Bhaskara, S. Kumar, Y. Gong, and S. F. Chang, “On binary embedding using circulant matrices,” *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 5507–5536, 2018.
[21] Z. Jin, C. Li, Y. Lin, and D. Cai, “Density sensitive hashing,” *IEEE Trans. Cybern.*, vol. 44, no. 8, pp. 1362–1371, Aug. 2014.
[22] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, “Spherical hashing,” in *Proc. CVPR*, Jun. 2012, pp. 2957–2964.
[23] X. Zhu, L. Zhang, and Z. Huang, “A sparse embedding and least variance encoding approach to hashing,” *IEEE Trans. Image Process.*, vol. 23, no. 9, pp. 3737–3750, Sep. 2014.
[24] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, “Locality-sensitive hashing scheme based on p-stable distributions,” in *Proc. SoCG*, 2004, pp. 253–262.
[25] B. Kulis, P. Jain, and K. Grauman, “Fast similarity search for learned metrics,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2143–2157, Dec. 2009.
[26] J. Wang, S. Kumar, and S.-F. Chang, “Semi-supervised hashing for large-scale search,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2393–2406, Dec. 2012.
[27] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, “Supervised hashing with kernels,” in *Proc. CVPR*, Jun. 2012, pp. 2074–2081.
[28] F. Shen, C. Shen, W. Liu, and H. T. Shen, “Supervised discrete hashing,” in *Proc. CVPR*, Jun. 2015, pp. 37–45.
[29] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan, “Supervised discrete hashing with relaxation,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 3, pp. 608–617, Mar. 2018.
[30] M. Hu, Y. Yang, F. Shen, N. Xie, and H. T. Shen, “Hashing with angular reconstructive embeddings,” *IEEE Trans. Image Process.*, vol. 27, no. 2, pp. 545–555, Feb. 2018.
[31] K. Ding, C. Huo, B. Fan, S. Xiang, and C. Pan, “In defense of locality-sensitive hashing,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 87–103, Jan. 2018.
[32] D. Wang, X. Gao, X. Wang, L. He, and B. Yuan, “Multimodal discriminative binary embedding for large-scale cross-modal retrieval,” *IEEE Trans. Image Process.*, vol. 25, no. 10, pp. 4540–4554, Oct. 2016.
[33] L. Zhu, Z. Huang, Z. Li, L. Xie, and H. Shen, “Exploring auxiliary context: Discrete semantic transfer hashing for scalable image retrieval,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5264–5276, Nov. 2018.
[34] M. Yu, L. Liu, and L. Shao, “Binary set embedding for cross-modal retrieval,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 12, pp. 2899–2910, Dec. 2017.
[35] L. Jin, K. Li, Z. Li, F. Xiao, G.-J. Qi, and J. Tang, “Deep semantic-preserving ordinal hashing for cross-modal similarity search,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 5, pp. 1429–1440, May 2019.
[36] D. Wang, X.-B. Gao, X. Wang, and L. He, “Label consistent matrix factorization hashing for large-scale cross-modal similarity search,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 10, pp. 2466–2479, Oct. 2019.
[37] D. Wang, Q. Wang, and X. Gao, “Robust and flexible discrete hashing for cross-modal similarity search,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 10, pp. 2703–2715, Oct. 2018.
[38] F. Zheng, Y. Tang, and L. Shao, “Hetero-manifold regularisation for cross-modal hashing,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 5, pp. 1059–1071, May 2018.
[39] Z. Jin, J. Y. Hwang, Y.-L. Lai, S. Kim, and A. B. J. Teoh, “Ranking-based locality sensitive hashing-enabled cancelable biometrics: Index-of-max hashing,” *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 2, pp. 393–407, Feb. 2018.
[40] L. Huang, Q. Yang, and W. Zheng, “Online hashing,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2309–2322, Jun. 2018.
[41] Q. Liu, G. Liu, L. Li, X.-T. Yuan, M. Wang, and W. Liu, “Reversed spectral hashing,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2441–2449, Jun. 2018.
[42] X. Bai, C. Yan, H. Yang, L. Bai, J. Zhou, and E. R. Hancock, “Adaptive hash retrieval with kernel based similarity,” *Pattern Recognit.*, vol. 75, pp. 136–148, Feb. 2018.
[43] Y. Cao, M. Long, J. Wang, and S. Liu, “Deep visual-semantic quantization for efficient image retrieval,” in *Proc. CVPR*, Jul. 2017, pp. 916–925.
[44] J. T. Zhou, H. Zhao, X. Peng, M. Fang, Z. Qin, and R. S. M. Goh, “Transfer hashing: From shallow to deep,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 6191–6201, Dec. 2018.
[45] L.-Y. Duan, Y. Wu, Y. Huang, Z. Wang, J. Yuan, and W. Gao, “Minimizing reconstruction bias hashing via joint projection learning and quantization,” *IEEE Trans. Image Process.*, vol. 27, no. 6, pp. 3127–3141, Jun. 2018.
[46] T. Yuan, W. Deng, J. Hu, Z. An, and Y. Tang, “Unsupervised adaptive hashing based on feature clustering,” *Neurocomputing*, vol. 323, pp. 373–382, May 2019.
[47] Z. Li, J. Tang, and X. He, “Robust structured nonnegative matrix factorization for image representation,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1947–1960, May 2018.
[48] G. Ding, Y. Guo, and J. Zhou, “Collective matrix factorization hashing for multimodal data,” in *Proc. CVPR*, Jun. 2014, pp. 2083–2090.
[49] J. Tang, K. Wang, and L. Shao, “Supervised matrix factorization hashing for cross-modal retrieval,” *IEEE Trans. Image Process.*, vol. 25, no. 7, pp. 3157–3166, Jul. 2016.
[50] Z. Lai, Y. Chen, J. Wu, W. K. Wong, and F. Shen, “Jointly sparse hashing for image retrieval,” *IEEE Trans. Image Process.*, vol. 27, no. 12, pp. 6147–6158, Dec. 2018.
[51] Y.-X. Wang and Y.-J. Zhang, “Nonnegative matrix factorization: A comprehensive review,” *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1336–1353, Jun. 2013.
[52] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, “Optimization with sparsity-inducing penalties,” *Found. Trends Mach. Learn.*, vol. 4, no. 1, pp. 1–106, Jan. 2012.

Huawen Liu (M’16) received the M.S. and Ph.D. degrees in computer science from Jilin University, Changchun, China, in 2007 and 2010, respectively.

He is currently a Visiting Scholar with The University of Texas at San Antonio, San Antonio, TX, USA. He is also a Professor with the Department of Computer Science, Zhejiang Normal University, Jinhua, China. His research interests include feature selection, sparse learning, and machine learning.

Xuelong Li (M’02–SM’07–F’12) is currently a Full Professor with the School of Computer Science and the Center for OPTical IMagery Analysis and Learning (OPTIMAL), Northwestern Polytechnical University, Xi’an, China.

Shichao Zhang (M'97–SM'05) received the Ph.D. degree in computer science from Deakin University, Geelong, VIC, Australia.

He is currently a China National Distinguished Professor with the School of Computer Science and Technology, Central South University, Changsha, China. He has published about 80 international journal articles and more than 80 international conference articles. His research interests include information quality and pattern discovery.

Dr. Zhang is also a Senior Member of the IEEE Computer Society. He received 16 national-class grants. He has been serving as an Associate Editor for the ACM TKDD, the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, KAIS, and the IEEE IIB.

Qi Tian (M'96–SM'03–F'16) received the M.S. degree in electrical and computer engineering (ECE) from Drexel University, Philadelphia, PA, USA, in 1996, and the Ph.D. degree in ECE from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2002.

He was a Full Professor with the Department of Computer Science, The University of Texas at San Antonio (UTSA), San Antonio, TX, USA, from 2002 to 2019. He is currently a Chief Scientist in computer vision with Huawei Noah's Ark Lab, Shenzhen, China. He published over 500 refereed journal and conference articles. His Google citation is over 17 200 with an H-index of 65. He was a coauthor of best articles presented at several conferences, including the IEEE ICME 2019, ACM CIKM 2018, ACM ICMR 2015, PCM 2013, MMM 2013, and ACM ICIMCS 2012. His research interests include computer vision, multimedia information retrieval, and machine learning.

Dr. Tian received the Top 10% Paper Award in MMSP 2011, the Student Contest Paper in ICASSP 2006, and was a coauthor of a Best Paper/Student Paper Candidate at the ACM Multimedia 2019, ICME 2015, and PCM 2007. He received the 2017 UTSA President's Distinguished Award for Research Achievement, the 2016 UTSA Innovation Award, the 2014 Research Achievement Award from the College of Science, UTSA, the 2010 Google Faculty Award, and the 2010 ACM Service Award. He is on the Editorial Board of JMM and JMVA. He is also an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, ACM TOMM, and MMSJ.