Learning Binary Code for Fast Nearest Subspace Search

Lei Zhou[a], Xiao Bai[a,*], Xianglong Liu[a], Jun Zhou[b], Edwin R. Hancock[c]

[a]*School of Computer Science and Engineering, Beihang University, Beijing, China*
[b]*School of Information and Communication Technology, Griffith University, Nathan, Australia*
[c]*Department of Computer Science, University of York, York, U.K.*

Abstract

Subspace is widely used to represent objects under different viewpoints, illuminations, identities, and more. Due to the growing amount and dimensionality of visual contents, fast search in a large-scale database with high-dimensional subspaces is an important task in many applications, such as image retrieval, clustering, video retrieval, and visual recognition. This can be facilitated by approximate nearest subspace (ANS) search which requires effective subspace representation. All existing methods for this problem represent a subspace by a point in the Euclidean or the Grassmannian space before applying the approximate nearest neighbor (ANN) search. However, the efficiency of these methods is not guaranteed because the subspace representation step can be very time consuming when coping with high-dimensional data. Moreover, the subspace to point transforming process may cause subspace structural information loss which influences the search accuracy. In this paper, we present a new approach for hashing-based ANS search which can directly binarize a subspace without transforming it into a vector. The proposed method learns the binary codes for subspaces following a similarity preserving criterion, and simultaneously leverages the learned binary codes to train matrix classifiers as hash functions. Experiments on face and action recognition and video retrieval applications show that our method outperforms several state-of-the-art methods in both efficiency and accuracy. Moreover, we also compare our method with vector-based hashing methods. Results also show the superiority of our subspace

*Corresponding author
Email addresses: leizhou@buaa.edu.cn (Lei Zhou), baixiao@buaa.edu.cn (Xiao Bai), xlliu@nlsde.buaa.edu.cn (Xianglong Liu), jun.zhou@griffith.edu.au (Jun Zhou), edwin.hancock@york.ac.uk (Edwin R. Hancock)

matrix based search scheme.

Keywords: Nearest subspace search, learning binary code, hashing, matrix classifier

1. Introduction

Subspace is important for data representation in many computer vision and pattern recognition tasks. It has demonstrated excellent capability in capturing the structural information of specific data, e.g. face images under different illuminations [1, 2, 3, 4], sequential frames in the video clips [5, 6], different identities [7, 8], and classes of similar objects [9, 10]. In a high-dimensional space, data can be either represented as subspaces or vectors. In practice, due to the high dimensionality and large-scale nature, it is better to represent these data by subspace rather than vectorized features because the intrinsic dimension of high-dimensional data is often much smaller than the ambient dimension [11]. Subspace representation has the ability to retain important information with significant dimensionality reduction [12, 13]. This has motivated the development of subspace representation based visual methods.

Given a searching problem, when a query is represented as a vectorized image and a dataset is composed of vectors, the problem can be solved by the nearest neighbor search. Due to the recent growth of visual contents, rapid search in a large dataset is highly demanded. Because of the high-computational cost, traditional linear search (or exhaustive search) is not appropriate to be applied on a large-scale dataset. Instead of linear search, many practical strategies have been proposed for approximate nearest neighbor (ANN) search [14, 15, 16, 17, 18]. The best known approach is hashing methods, which can efficiently solve high-dimensional and large-scale problems [19, 20, 21, 22, 23]. A hashing algorithm aims to seek compact binary codes for high-dimensional data so that the Hamming distances between binary codes preserve the pairwise similarities of the data points. Recently, with the rise of interest in deep learning technology, there are also many deep hashing methods [24, 25, 26, 27, 28] which have achieved great success. Since deep learning methods require a large number of labeled training samples, these deep hashing methods may be ineffective when used in an unsupervised way or used with small datasets. These problems can be ef-

fectively solved by adopting a subspace representation.

Compared to the vectorized representations, subspace is a more generic data type in practice [29]. There are many benefits of using subspace methods. First, comparing subspaces is cheaper than comparing two datasets directly. Second, they are more robust to missing data which can be filled-in by interpolation [30]. Third, subspace representation has lower dimension [31] than the original representation and the distance of subspaces can be efficiently measured by principal angles [32]. Given a query image (or a set of images) represented as a point (or a subspace) in a high-dimensional space, searching the subspace nearest to the query is much faster than a brute sequential search over the entire database with vectorized representation. When subspace is used as the representation, how to efficiently find the nearest subspace for a given query image or subspace has to be studied. This problem is related to a wide range of computer vision and pattern recognition applications such as face recognition, image approximation, speaker recognition, action recognition, and video retrieval. In recent years, some efficient methods have been proposed for approximate nearest subspace (ANS) search. Basri *et al.* [33, 34, 10] proposed to transform subspaces to points, then solve the subspace searching problem by well-studied ANN search on points. Motivated by this work, some recent approaches [35, 36] utilized the idea of locality sensitive hashing (LSH) [37] to accelerate the searching process. More recently, Xu *et al.* [38] proposed a video retrieval framework, which defined a similarity-preserving distance metric between an image and its orthogonal projection in the subspace of the video. Since manifold is also widely used for the subspace analysis [39, 40], Grassmannian based subspace search methods have been proposed [41, 42, 43]. These approaches, however, represent subspaces by points in the Euclidean space or Grassmannian space before applying the ANN methods. This operation increases the dimension of data since a subspace of dimension d will be transformed into a vector of size $\Theta(d^2)$. In addition, the subspace to point transformation may incur subspace structural information loss and computational cost. Therefore, they can not efficiently solve the ANS search problem, specifically for very high-dimensional data in a lager-scale dataset.

The superiority of hashing algorithm in information retrieval comes from two aspects [44]. First, binary signatures generated by hashing can significantly reduce the

storage space for large-scale data. In addition, pairwise data similarity can be fast estimated by calculating the Hamming distance between the corresponding binary signatures. Motivated by these two properties, in our previous work [46], we proposed a matrix classifier based binary coding method which solves the high-dimensional and large-scale subspace retrieval problem. However, there are several issues with this approach. First, the binary codes learning objective function does not consider the constraint that makes the hamming distance between dissimilar pair large. Second, the previous work cannot well handle the condition that query subspaces with different dimensions. In order to solve these issues, in this paper, we extend our previous work and present a new method for linear subspace search based on hashing algorithm. First we learn the binary codes for a given subspace set following defined similarity preserving criterion. In order to solve the out-of-sample problem (a new query which is not used in the training set), our algorithm leverages the learned binary codes to train a set of matrix classifiers to predict the code for query subspace. Here each bit of the binary code for subspace is considered as a class label for that subspace. Assume that we learn r bits binary code for each subspace, then we train r binary classifiers for the r bits binary code. For any given out-of-sample query subspace, the corresponding r bits binary code can be efficiently generated by the r trained binary classifiers. We choose to use the Support Matrix Machines (SMM) [45] as the binary matrix classifier because of its excellent capability in capturing the structural information within the matrix data. As SMM is a linear classifier which requires the dimensionality of the training and testing samples to be the same, we further extend the SMM to a kernelized version. Then our proposed method can address the circumstances in which the query subspaces have different dimensions.

Compared with the previous similar works [10, 42, 36, 38], the main advantages of our method are a) that it achieves fast computation without converting a subspace into a vector in the data space, and b) the matrix classifier can efficiently generate effective compact hash codes using structural information for the subspace. In the offline training stage, we learn the binary codes for a set of given subspaces with their similarities, and simultaneously train binary classifiers by the learned codes. All calculations in this stage are done directly on subspace matrices. In the online query stage, the query

subspace matrix is directly binarized by the trained classifiers. Then we can use the Hamming distance between binary codes to quickly search the ANS. In this way, our method can efficiently solve the ANS search problem even with a high-dimensional and lager-scale subspace dataset.

The main technical extensions and contributions of this paper are as follows:

1. The objective function for learning binary codes is different with our previous work [46]. We design a more strict objective function which not only considers the hamming distances between similar pairs, but also constrains the hamming distances between dissimilar pairs. Therefore, we can learn more discriminative binary codes for the subspace searching problem.
2. To adapt to complex applications, in which the query subspaces have different dimensions, we further extend our method with a kernelized version to address the circumstances in which the query subspaces have different dimensions.
3. Experiments on several public databases with different tasks are presented. The results show that our method outperforms all the state-of-the-art subspace search methods in both searching accuracy and efficiency. Moreover, we also compare the proposed method with several vector-based hashing methods. The results also show the superiority of our subspace matrix based searching scheme.

In the rest of the paper, a review of related work is given in Section 2. In Section 3, we define the similarity between subspaces. We then propose an efficient hashing based subspace search method with SMM classifier. We further extend the SMM to a kernelized version for query subspace with different dimensions in this section. In Section 4, we show the results of three sets of experiments on five public datasets for face, action, gesture recognition and video retrieval. The conclusions are drawn in Section 5.

2. Related Work

In this section, we review the related work on ANS search. One of the earliest method in this area was proposed by Basri, Hassner and Zelnik-Manor (BHZ) [10],

which represents a subspace $\mathcal{X}$ as its orthogonal bases matrix X[1]. Then a scalable mapping is defined to map X to a vector by taking the entries of the upper triangular portion of XX^T with the diagonal entries scaled by $1/\sqrt{2}$. This allows traditional ANN vector search methods be applicable to the data. Based on the mapping, this method can solve the ANS problem when both query and database elements are either points or subspaces of different dimensions. However, when the dimension d of the subspace is very high, the corresponding vector has size $d(d+1)/2$, so the efficiency of this method suffers from high-dimensional data.

Motivated by the above method, Wang *et al.* proposed a Grassmannian-based hashing solution to the ANS search problem [42]. This method represents the subspace as a point on a Grassmannian manifold, and is referred to as GLH (Grassmannian-based Locality Hashing). The LSH takes advantage of the relative geometric positions of different subspaces which are encoded in their principal angles with random lines. As a result this method provides good results when the ambient dimension is high. However, instability still exists due to the strong randomness of LSH, which makes the search accuracy decreased when the training dataset is small [37]. A similar approach proposed by Stephen and Bruce introduces a subspace forest for determining the approximate nearest neighbors of points on the Grassmann manifolds [41], which can rapidly classify actions by a set of labeled samples. The subspace forest works in a manner that is conceptually similar to randomized forests for ANN computations, but maintains the Grassmann manifold geometry.

Ji *et al.* presented a method to produce similarity-preserving binary signatures for subspaces (BSS) [36]. In this method, the angular similarity and angular distance between subspaces is first defined. Then a sign-random-projection function is applied to generate binary signatures for the subspace. It is proved that the Hamming distance between the binary signatures is an unbiased estimator of the pairwise angular distance. Hence, LSH is applied to the ANN vector search problem. Since this method transforms the subspace matrices into higher dimensional vectors, it needs long binary codes to preserve the similarity between subspaces. As shown in Figure 1, the

[1] In our method, all operations on a subspace are done on its orthogonal bases matrix.

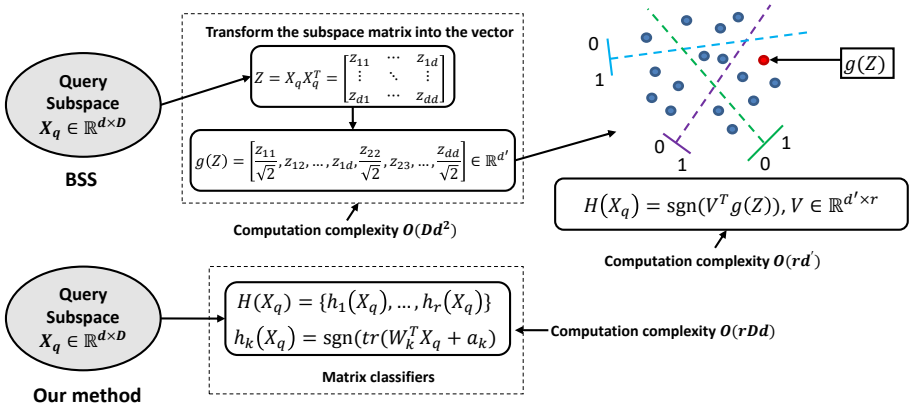


Figure 1: The comparison of search process between BSS [36] and our proposed method. For a query subspace, BSS first transforms the subspace matrix into higher dimensional vector $g(Z) \in \mathbb{R}^{d'}$, where $d' = d(d+1)/2$. Then a sign-random-projection function is applied to generate binary signatures for subspace. Our method directly binarize the query subspace by the trained matrix classifiers without transforming the subspace matrix into vector form. As shown in the figure, the computation complexity of search process for BSS is $O(Dd^2 + rd') = O((D+r)d^2)$. For our method, the computation complexity is $O(rDd)$. And experiments show that BSS needs more than $r = 2000$ bits binary codes to achieve a good result. But our method achieves a good result only with $r = 64$ bits binary codes. Therefore, our method significantly outperforms BSS in the query time.

subspace matrix to vector transforming process will cause information loss and calculational cost.

In order to solve the video retrieval problem with a subspace representation, Li *et al.* [43] have proposed a method which learns hash functions using a max-margin framework across both Euclidean space and a Riemannian manifold (HER). More recently, Xu *et al.* [38] proposed a video retrieval framework, which define a similarity-preserving distance metric between an image and its orthogonal projection in the subspace of the video. They first represented videos as subspaces of frames, and then asymmetrically projected images and videos into a common Hamming space, where they could efficiently retrieve the most relevant videos provided with an image query. Actually, their method also converts the subspace matrix into vector, and then learns similarity-preserving binary codes by the Euclidean distance between images and video subspaces.

The drawbacks of existing methods are solved in our algorithm. We leverage the binary classifier as the hash function to encode a subspace into a binary code. On one hand, the compact binary code for the subspace can significantly reduce the storage space and computational cost for large-scale data. On the other hand, experiments show that our method achieves less information loss than previous works.

3. Proposed Method

In this section, we describe a hashing based method that solves the ANS search problem. We first define the similarity between subspaces and construct an affinity matrix for learning the binary code. Then the binary codes for given subspace set can be learned by a similarity preserving criterion. In order to solve the out-of-sample problem, our algorithm leverages the learned binary codes to train a set of binary classifiers as hash functions. Here each bit of the binary code for subspace is considered as a class label for that subspace. Then we can train r binary classifiers for the r bits binary code. For any given out-of-sample query subspace, the corresponding r bits binary code can be efficiently generated by the r trained binary classifiers.

A subspace is generally represented as a matrix composed of the orthonormal bases of the data set. Our proposed method aims to utilize the significant structural information residing in a subspace to learn effective binary codes. Motivated by previous vector-based hashing methods [20, 47], our method makes three significant novel improvements to the subspace search problem. First, in common with most existing hashing methods, the newly proposed objective function for learning the binary codes not only guarantees that the Hamming distance for similar data is small but additionally guarantees that the Hamming distance for dissimilar data is large. This makes the learned binary codes more discriminative. Second, we leverage the matrix classifiers and use them as hash functions to directly binarize the subspace matrix. The nuclear norm term in the matrix classifier imposes a low-rank constraint on the regression matrix which allows us to utilize more effectively the structural information residing in the subspace. Thirdly, we extend our method to obtain a novel kernelized method for subspace query which can gauge the similarity of data residing in subspaces with

different dimensions. By contrast, most existing hashing methods require that the di-
190 mensionality of the training and testing data be the same. Our method is detailed in the remainder of this section.

3.1. Similarity Between Subspaces

First we define the similarity between linear subspaces. In our method, we use the principal angles proposed in [48] which reveal the relative positions of two subspaces. Let $\mathcal{F}$ and $\mathcal{G}$ be linear subspaces in $\mathbb{R}^d$, whose dimensions satisfy

$$p = \dim(\mathcal{F}) \geq \dim(\mathcal{G}) = q \geq 1 \tag{1}$$

The principal angles $\{\theta_i\}_{i=1}^q \in [0, \pi/2]$ between these two subspaces are defined recursively by

$$\cos\theta_k = f_k^T g_k = \max_{\substack{f\in\mathcal{F}, \|f\|_2=1 \\ f^T[f_1,\ldots,f_{k-1}]=0}} \max_{\substack{g\in\mathcal{G}, \|g\|_2=1 \\ g^T[g_1,\ldots,g_{k-1}]=0}} f^T g \tag{2}$$

Let F be a $d \times p$ matrix whose columns are orthonormal bases for subspace $\mathcal{F}$. Let G be a $d \times q$ matrix whose columns are orthonormal bases for subspace $\mathcal{G}$. It is easy to prove that the cosine of each principal angle equals a singular value of $F^T G$. Assume that the SVD of $F^T G$ is

$$F^T G = U diag(\sigma_1, \sigma_2, ..., \sigma_q) V^T \tag{3}$$

where $1 \geq \sigma_1 \geq \sigma_2 \geq ... \geq \sigma_q \geq 0$. Assume that the principal angles satisfy $0 \leq \theta_1 \leq \theta_2 \leq ... \leq \theta_q \leq \pi/2$, we can get

$$\cos\theta_i = \sigma_i, \quad i = 1, 2, ..., q \tag{4}$$

The first principal angle θ_1 is the smallest angle between a pair of unit vectors from two subspaces. The cosine of the principal angle σ_1 is the first canonical correlation [49]. The i-th principal angle and canonical correlation are defined recursively. It is known that the principal angles are related to the geodesic distance [50]. Then the similarity between subspaces $\mathcal{F}$ and $\mathcal{G}$ can be defined as

$$S_{\mathcal{F},\mathcal{G}} = \cos\theta_{\mathcal{F},\mathcal{G}} = \frac{\sum_{i=1}^q \cos^2\theta_i}{\sqrt{p}\sqrt{q}} = \frac{\sum_{i=1}^q \sigma_i^2}{\sqrt{p}\sqrt{q}} \tag{5}$$

Since the singular values have the property that $\sum_{i=1}^{q} \sigma_i^2 = \|F^T G\|_F^2$, where $\|A\|_F$ is the Frobenius norm of A, we can rewrite the subspace similarity as

$$S_{\mathcal{F},\mathcal{G}} = \frac{\|F^T G\|_F^2}{\sqrt{p}\sqrt{q}} \tag{6}$$

Given the defined similarity between subspaces, we can construct an $n \times n$ affinity matrix S for a subspace set $\{\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_n\}$. Where $S_{ij} \in [0, 1]$ is the similarity between $\mathcal{X}_i$ and $\mathcal{X}_j$.

3.2. Model Formulation

Given a set of linear subspaces $\mathcal{X} = \{\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_n\} \subset \mathbb{R}^d$, the goal is to search the approximate nearest subspace in $\mathcal{X}$ for a given query subspace $\mathcal{X}_q$. In order to achieve rapid search, we transform the subspaces into binary codes. Here we assume that X_i is a matrix whose columns are orthonormal bases of $\mathcal{X}_i$ and b_i is the corresponding r bit binary code for X_i. Our goal is to learn a set of hash functions which can well preserve the similarities between subspaces in $\mathcal{X}$.

Based on the definition of subspace similarity in Eq. (6), we first calculate the affinity matrix S for $\mathcal{X}$. Let $H(\cdot)$ be a set of r hash functions to be learned, and $H(\cdot) = \{h_1(\cdot), h_2(\cdot), ..., h_r(\cdot)\} \in \{-1, 1\}^r$. The binary codes and hash functions can be learned simultaneously. In other words, our method aims to learn a set of binary codes $B = \{b_i\}_{i=1}^n \in \{-1, 1\}^{r \times n}$ which well preserves the similarities between subspaces in $\mathcal{X}$ and simultaneously learns a set of hash functions that map $\mathcal{X}$ to the learned binary codes.

In our previous work [46], the objective function to learn binary codes is

$$\min_{B \in \{-1,1\}^{r \times n}} \sum_{i=1}^{n} \sum_{j=1}^{n} S_{ij} \frac{r - b_i^T b_j}{2} \tag{7}$$

where $\frac{r - b_i^T b_j}{2}$ is the Hamming distance between b_i and b_j. However, minimizing objective function (7) can only make the Hamming distance between similar pairs small. It does not consider the Hamming distance between dissimilar pairs. In other words, the discrimination of binary codes of dissimilar pairs cannot be guaranteed.

In this paper, we design a more strict objective function to solve this problem. When $\mathcal{X}_i$ and $\mathcal{X}_j$ are similar, it is expected that the Hamming distance between two

binary codes $H(X_i)$ and $H(X_j)$ be small. Correspondingly, $H(X_i)^T H(X_j)$ is close to r. Conversely, when $\mathcal{X}_i$ and $\mathcal{X}_j$ are dissimilar, the Hamming distance between $H(X_i)$ and $H(X_j)$ shall be large, i.e., $H(X_i)^T H(X_j)$ is close to $-r$. Formally, the similarity preserving criterion can be defined as

$$H(X_i)^T H(X_j) = \begin{cases} r & \text{when } S_{ij} \text{ is close to 1} \\ -r & \text{when } S_{ij} \text{ is close to 0} \end{cases} \tag{8}$$

To meet this similarity preserving criterion, the optimization objective function can be written as

$$\min_H \sum_{i=1}^{n} \sum_{j=1}^{n} (H(X_i)^T H(X_j) - r(2S_{ij} - 1))^2 \tag{9}$$

Considering these two situations to minimize objective function (9): a) When S_{ij} is close to 1, i.e., subspaces $\mathcal{X}_i$ and $\mathcal{X}_j$ are similar, $r(2S_{ij} - 1)$ is close to r. Then it is expected that $H(X_i)^T H(X_j)$ is close to r, i.e., the Hamming distance between $H(X_i)$ and $H(X_j)$ is near 0. b) When S_{ij} is close to 0, i.e., subspaces $\mathcal{X}_i$ and $\mathcal{X}_j$ are dissimilar, $r(2S_{ij} - 1)$ is close to $-r$. Then it is expected that $H(X_i)^T H(X_j)$ is close to $-r$, i.e., the Hamming distance between $H(X_i)$ and $H(X_j)$ is close to r.

Optimizing Eq. (9) directly for learning hash functions is difficult. Hence we decompose problem (9) into two simple sub-problems:

$$\min_{B \in \{-1,1\}^{r \times n}} \sum_{i=1}^{n} \sum_{j=1}^{n} (b_i^T b_j - r(2S_{ij} - 1))^2 \tag{10}$$

and

$$\min_H \sum_{i=1}^{n} \sum_{k=1}^{r} \delta(b_i^{(k)} = h_k(X_i)) \tag{11}$$

where $b_i^{(k)}$ is the k-th bit of b_i and $\delta \in \{0, 1\}$ indicates whether the relationship between $b_i^{(k)}$ and $h_k(X_i)$ is defined. The solution to problem (10) is the optimal binary codes which can preserve the similarity between subspace set $\mathcal{X}$, and problem (11) is to learn a set of functions $H(\cdot) = \{h_1(\cdot), h_2(\cdot), ..., h_r(\cdot)\} \in \{-1, 1\}^r$ which map the subspaces to corresponding binary codes.

3.3. Optimization

Learning binary codes. Problem (10) can be rewritten into a matrix form

$$\min_{B\in\{-1,1\}^{r\times n}} \|B^T B - r(2S-1)\|_F^2 \tag{12}$$

Since $r(2S-1)$ is a constant term, this becomes a least-squares problem. Let $B^{(k)}$ be the k-th row of matrix B, then $B^{(k)} = (b_1^{(k)}, b_2^{(k)}, ..., b_n^{(k)})$. In this way, Eq. (12) becomes

$$\min_{B\in\{-1,1\}^{r\times n}} \|\sum_{k=1}^{r} B^{(k)T} B^{(k)} - r(2S-1)\|_F^2 \tag{13}$$

We can use a greedy method to solve $B^{(k)}$ sequentially. In each step, it only solves one vector $B^{(k)}$ given the previously solved vectors $B^{(1)}, B^{(2)}, ..., B^{(k-1)}$. Let a residue matrix be defined as

$$R_{k-1} = r(2S-1) - \sum_{t=1}^{k-1} B^{(t)T} B^{(t)} (R_0 = r(2S-1)) \tag{14}$$

$B^{(k)}$ can be solved by minimizing the following cost function

$$\begin{aligned} &\|B^{(k)T} B^{(k)} - R_{k-1}\|_F^2 \\ =&(B^{(k)} B^{(k)T})^2 - 2B^{(k)} R_{k-1} B^{(k)T} + tr(R_{k-1}^2) \\ =&n^2 - 2B^{(k)} R_{k-1} B^{(k)T} + tr(R_{k-1}^2) \\ =&-2B^{(k)} R_{k-1} B^{(k)T} + const \end{aligned} \tag{15}$$

Discarding the constant term, we get a compact objective

$$\max_{B^{(k)}\in\{-1,1\}^n} B^{(k)} R_{k-1} B^{(k)T} \tag{16}$$

Since the constraint $B^{(k)} \in \{-1,1\}^n$ is discrete, Eq. (16) is a nonconvex problem. Here we apply the spectral relaxation method to relax this constraint. Then the following quadratic problem can be defined

$$\begin{aligned} \max_{B^{(k)}} \quad & B^{(k)} R_{k-1} B^{(k)T} \\ \text{s.t. } & B^{(k)} B^{(k)T} = 1 \end{aligned} \tag{17}$$

The solution to this quadratic problem is the eigenvector corresponding to the largest eigenvalue of R_{k-1}.

The solution to Eq. (17) generates real-values vectors $\bar{B}^{(k)} = (\bar{b}_1^{(k)}, \bar{b}_2^{(k)}, ..., \bar{b}_n^{(k)})$. Next we need to transform $\bar{B}^{(k)}$ into binary codes via a proper threshold. Following the Spectral Hashing method [51], a good binary code shall satisfy that each bit has 50% chance of being 1 or -1. Therefore, we select the median value of $\bar{B}^{(k)}$ as a threshold. If $\bar{b}_i^{(k)}$ is larger than the median value, $b_i^{(k)} = 1$ ($b_i^{(k)}$ is the k-th bit of b_i). Otherwise, $b_i^{(k)} = -1$. Then discriminative binary codes B can be obtained, which well preserves the similarities between subspaces in $\mathcal{X}$.

Learning hash functions. Although we have learned the binary codes for subspace set $\mathcal{X}$, it does not mean that the ANS search problem has been solved. For a given new query subspace $\mathcal{X}_q$, if we add it into the subspace set $\mathcal{X}$ and update all binary codes according to the method presented above, the high computational cost obviously can not be accepted for practical applications. According to the idea of hashing algorithm, we need a set of hash functions which can map the query subspace to the corresponding binary code. The solution to problem (11) is the required hash functions.

Problem (11) can be treated as a binary classification problem. Each bit of the binary codes for the subspace can be considered as a class label (class '1' or class '-1') for that subspace. A binary classifier can be trained for each bit of the learned binary codes. The subspace is generally represented as a matrix composed the orthonormal bases which contain significant structural information concerning the subspace. If we simply concatenate the column vectors of X into a single long column vector, the subspace structural information will be disrupted. Therefore, instead we leverage the matrix classifiers as hash functions. To this end, we train r matrix classifiers by the learned r-bit binary codes in problem (10). Then we use the r binary matrix classifiers to obtain the r-bit binary code for any query subspace.

In this paper, we choose to use the Support Matrix Machine (SMM) [45] as the matrix classifier due to its excellent capability in capturing the structural information within the matrix data. To capture the correlation among columns or rows within the matrix data, SMM imposes a low-rank constraint on the regression matrix W to leverage the structural information. Since the nuclear norm $\|W\|_*$ is the best convex approximation of rank(W) over the unit ball of matrices, and the hinge loss is well fit for

large margin principle, the formulation of SMM model is

$$\min_{W,a} \frac{1}{2}\|W\|_F^2 + \tau\|W\|_* + C\sum_{i=1}^{n}\{1 - y_i[tr(W^T X_i) + a]\}_+ \tag{18}$$

where $\{1-e\}_+ = \max\{0, 1-e\}$ is the hinge loss, W is the matrix of regression coefficients, a is an offset term, C is a regularization parameter, and y_i is the class label of X_i. SMM is based on a penalty function which is the combination of the squared Frobenius norm $\|W\|_F^2$ and the nuclear norm $\|W\|_*$.

Because each bit of the binary code is independent of each other, the binary classifiers of each bit can be trained independently. For the bit $b_i^{(k)}$, the corresponding SMM can be trained by the following formulation

$$\min_{W_k,a_k} \frac{1}{2}\|W_k\|_F^2 + \tau\|W_k\|_* + C\sum_{i=1}^{n}\{1 - b_i^{(k)}[tr(W_k^T X_i) + a_k]\}_+ \tag{19}$$

Here we consider $b_i^{(k)}$ as a class label for X_i. After learning the binary classifier, the corresponding hash function can be formulated as $h_k(X) = sgn(tr(W_k^T X) + a_k)$, where the sign function $sgn(x)$ returns 1 if $x > 0$ and -1 otherwise.

Since the objective function Eq. (19) is convex in both W and a, we use the Alternating Direction Method of Multipliers (ADMM) [52] to optimize this convex problem. The objective function Eq. (19) can be equivalently written as

$$\begin{gathered}\min_{W_k,a_k,V_k} H(W_k, a_k) + G(V_k) \\ \text{subject to} \quad V_k - W_k = 0\end{gathered} \tag{20}$$

where

$$H(W_k, a_k) = \frac{1}{2}\|W_k\|_F^2 + C\sum_{i=1}^{n}\{1 - b_i^{(k)}[tr(W_k^T X_i) + a_k]\}_+ \tag{21}$$

and

$$G(V_k) = \tau\|V_k\|_* \tag{22}$$

Then the augmented Lagrangian function of Eq. (20) is

$$\begin{aligned}L(W_k, a_k, V_k, \Lambda) =& H(W_k, a_k) + G(V_k) + tr[\Lambda^T(V_k - W_k)] \\ &+ (\rho/2)\|V_k - W_k\|_F^2\end{aligned} \tag{23}$$

where $\rho > 0$ is an augmented Lagrangian parameter, and Λ is a penalty parameter. ADMM solves Eq. (23) with the following iterative equations

$$(W_k^{(i+1)}, a_k^{(i+1)}) = \min_{W_k, a_k} L(W_k, a_k, V_k^{(i)}, \Lambda^{(i)}) \tag{24}$$

$$V_k^{(i+1)} = \min_{V_k} L(W_k^{(i+1)}, a_k^{(i+1)}, V_k, \Lambda^{(i)}) \tag{25}$$

$$\Lambda^{(i+1)} = \Lambda^{(i)} + \rho(V_k^{(i+1)} - W_k^{(i+1)}) \tag{26}$$

The condition of stopping iteration is that the primal residual $r^{(i+1)} = \|V_k^{(i+1)} - W_k^{(i+1)}\|_F^2$ and the dual residual $t^{(i+1)} = \rho\|V_k^{(i+1)} - V_k^{(i)}\|_F^2$ converge to 0.

From the solution of Eq. (19), we can successively obtain r hash functions $\{h_1(\cdot), h_2(\cdot), ..., h_r(\cdot)\}$, where

$$h_k(X) = sgn(tr(W_k^T X) + a_k) \tag{27}$$

Then we can efficiently get the corresponding r bits binary code for the query subspace by these hash functions.

We name our proposed model Hashing based Subspace Search with SMM (HSS-SMM). The proposed method is summarized in Algorithm 1.

3.4. For Different Query Dimension

In practice, the dimension of the query subspace may be different from that of the training subspace [53]. However, due to the restriction by the dimension of classification-plane, linear matrix classifiers such as bilinear SVM [54] and SMM [45] both require that the dimensionality of the training and testing samples be the same. In our proposed HSS-SMM model, the dimension of query subspace must be equal to the training subspace due to the restriction by W_k.

To solve this problem, we extend the SMM to a kernelized version for query subspace with different dimensions. Here we use the projection kernel for subspace which is defined as

$$\kappa(X_i, X_j) = tr(\phi(X_i)^T \phi(X_j)) = \|X_i^T X_j\|_F^2 \tag{28}$$

Algorithm 1: The proposed HSS-SMM model.

Input: Training subspaces $\mathcal{X} = \{\mathcal{X}_i\}_{i=1}^n \subset \mathbb{R}^d$ and the hash bits r.

Output: Hash functions $H = \{h_i\}_{i=1}^r$ and the binary codes $B = \{b_i\}_{i=1}^n$ of $\mathcal{X}$.

Steps:

1. Construct orthonormal bases $X_i \subset \mathbb{R}^{d\times D}$ from subspaces $\mathcal{X}_i$.

2. Compute similarity between subspaces by Eq. (6).

3. Construct affinity matrix S for X.

4. Optimize the quadratic problem Eq. (17), the solution is $\bar{B} = \{\bar{b}_i\}_{i=1}^n$

5. Choose the r median values $v_1, v_2, ..., v_r$ of the r eigenvectors obtained in step 3 as threshold.

for $i = 1, ..., n$ **do**

  for $l = 1, ..., r$ **do**

    If the l-th element of $\bar{b}_i$ is larger than v_l, $b_i^{(l)} = 1$.

    Otherwise, $b_i^{(l)} = -1$.

  end

end

6. Optimize Eq. (19) with B fixed.

for $k = 1, ..., r$ **do**

  Optimize Eq. (19) by the ADMM algorithm, achieving W_k and a_k.

  $h_k(X) = sgn(tr(W_k^T X) + a_k)$.

end

where $X_i \subset \mathbb{R}^{d\times p}$, $X_j \subset \mathbb{R}^{d\times q}$, and $p \neq q$. From the subspace distance definition in equation (6), the projection kernel can measure the similarity between subspaces with different dimensions.

Then, for the k-th bit binary code, the training model of kernel SMM is

$$\begin{aligned} &\min_{W_k, a_k} \frac{1}{2}\|W_k\|_F^2 + \tau\|W_k\|_* \\ &\text{s.t. } b_i^{(k)}(tr(W_k^T\phi(X_i)) + a_k) \geq 1,\ i = 1, 2, ..., n. \end{aligned} \tag{29}$$

$W_k = \sum_{i=1}^n \omega_{k,i}\phi(X_i)$ and a_k can be obtained by solving the dual problem of E-

Algorithm 2: The search process of our method.

Input: Query subspace $\mathcal{X}_q \in \mathbb{R}^d$, the dimension D and binary codes B of subspace dataset.

Output: The binary code b_q of $\mathcal{X}_q$ and the approximate nearest subspace for the given query subspace.

Steps:

1. Construct orthonormal bases $X_q \subset \mathbb{R}^{d \times D_q}$ from subspaces $\mathcal{X}_q$.

2. Learn the binary code for query subspace.

while $D_q = D$ **do**
  Calculate the binary code of X_q by Eq. (27),
  $b_q = (h_1(X_q), h_2(X_q), ..., h_r(X_q))$.
end

while $D_q \neq D$ **do**
  Calculate the binary code of X_q by Eq. (30),
  $b_q = (h_1(X_q), h_2(X_q), ..., h_r(X_q))$.
end

3. Calculate the Hamming distances between b_q and B.

4. Rank the Hamming distances and return the approximate nearest subspace with the smallest Hamming distance to the query subspace.

q. (29). Then the hash function learned by the kernel SMM can be written as

$$\begin{aligned} h_k(X) &= sgn(tr(W_k^T \phi(X)) + a_k) \\ &= sgn(\sum_{i=1}^{n} \omega_{k,i} tr(\phi(X_i)^T \phi(X)) + a_k) \\ &= sgn(\sum_{i=1}^{n} \omega_{k,i} \kappa(X, X_i) + a_k) \end{aligned} \tag{30}$$

Similarly, we can train r matrix classifiers for each bit of the r bits binary code. Then we get the hash functions $H(\cdot) = \{h_1(\cdot), h_2(\cdot), ..., h_r(\cdot)\}$. We call this model HSS-KSMM. Given two query dimensionality settings, the search process of our method is summarized in Algorithm 2.

3.5. Differences with Vector-based Hashing

Our subspace search framework is motivated by the two-steps hashing algorithm. But there are some novel differences. First, our method is designed to directly binarize the subspace matrix. To the best of our knowledge, it is the first work to directly binarize the matrix from data. Second, we extend our method to give a kernelized version for query subspaces with different dimensions. This is achieved by using the projection kernel designed for measuring subspace similarity. Most existing hashing algorithms require that the dimensionality of the training and testing samples be the same.

When we transform the subspace matrix into vector form, this problem can be solved with vector-based hashing algorithms. We emphasize that subspace structural information will be utilized by the nuclear norm term in objective function Eq. (18). If we simply concatenate each column vector of X into a single long column vector, subspace structural information will be disrupted. We have designed experiments to compare our method with vector-based hashing algorithms. Results show the superiority of our subspace matrix based search scheme.

3.6. Complexity Analysis

Here we analyse the time complexity of our algorithm. In the offline training process, for a subspace set $X = \{X_i\}_{i=1}^{n} \subset \mathbb{R}^{d\times D}$, the matrix multiplication for distance between subspaces in Eq. (6) needs $O(dD^2)$. The affinity matrix can be computed in $O(n^2dD^2)$.

In the online searching process, calculating the r bit binary code by Eq. (27) for a query subspace costs $O(rDd)$. Therefore, the encoding time for the query subspace in our algorithm is linearly dependent on the dimension d of training subspaces, regardless the size of training set. Experiments show that our method outperforms the previous works in the query time.

4. Experiments

4.1. Baseline Methods

To evaluate the effectiveness of the proposed method, several state-of-the-art or classical nearest subspace search methods were taken as the baseline algorithms, including BHZ [10], BSS [36], IBC/BBC [38], and manifold methods GLH [42], HER [43]. As HER, and IBC/BBC are designed for query-by-image video retrieval, we only compared with them for video retrieval experiments. We also present the results of our previous work BCMC (Binary Coding by Matrix Classifier) [46]. Moreover, to show the superiority of our subspace matrix based search scheme, we compared it with vector-based unsupervised hashing methods including SH [51], ITQ [55], TSH [56], DH [24], DeepBit [26], and HashGAN [57]. For these methods, we first need to transform the subspace matrix into the vector form. For fair comparison, we investigated two ways to transform the subspace matrix into the vector form. In the first method X is mapped to a vector by taking the entries of the upper triangular portion of of the outer-product XX^T with the diagonal entries scaled by $1/\sqrt{2}$. Since the transformation can preserve the similarity between subspaces, i.e, the distance between subspace matrices is equal to the distance between their vector form obtained from this transformation (the proof is shown in BSS [36]). This subspace transformation has been widely used in the previous works on subspace search. We refer to these methods as SH-1, ITQ-1, TSH-1, DH-1, DeepBit-1 and HashGAN-1. A second method is simply to concatenate each column vector of X to form a single long column vector. We refer to these methods as SH-2, ITQ-2, TSH-2, DH-2, DeepBit-2 and HashGAN-2. We then utilized the resulting vector-based hashing methods to search for the approximate nearest subspace.

4.2. Datasets and Evaluation Criteria

Datasets: To evaluate the performance of our HSS-SMM and HSS-KSMM models, we undertook three sets of experiments on five public datasets. Since the subspace is generated from similar image set, it is infeasible to compare subspace search methods on the datasets in which each category contains only one or two example images. We chose several public datasets in which each category contains sufficient images to

Figure 2: Images on the left are sample faces under varying illuminations from the CMU Multi-PIE Face Database. The images on the right are cropped frames randomly chosen from the video clips in the YouTube Faces Database.

construct reliable and stable subspaces. The first experiment performed was face recognition using two datasets, namely a) the CMU Multi-PIE Face Database [58] and b) the YouTube Faces Database [59]. The CMU Multi-PIE Face Database was published in 2010. It consists of more than 750,000 images collected from 337 people under 15 view points and 19 illumination conditions with a multitude of different facial expressions. The YouTube Faces Database was published in 2011. It contains 3425 videos of 1595 different people. For each subject, there are 2.15 videos on average. We used the frame-images database which contains sequential frames extracted from the videos, and cropped only the face from each image (see Figure 2).

The second experiment performed was action and gesture recognition on the UCF101 Dataset [60] and the Cambridge Gestures Dataset [61]. The UCF101 was published in 2012. It contains 101 action categories collected from 13320 videos (see examples in Figure 3). The Cambridge Gestures Dataset contains 900 image sequences of 9 gesture classes with three kinds of hand shapes (flat, spread, v-shape) and three kinds of motions (leftward, rightward, contraction). The combination of hand shapes and motions yield 9 classes of gestures. Each class has 100 image sequences (10 arbitrary motions of 2 subjects under 5 different illuminations).

The last one was video retrieval on the Big Bang Theory (BBT) [43]. The Big Bang Theory is a sitcom (about 20 minutes each episode) which includes many full-

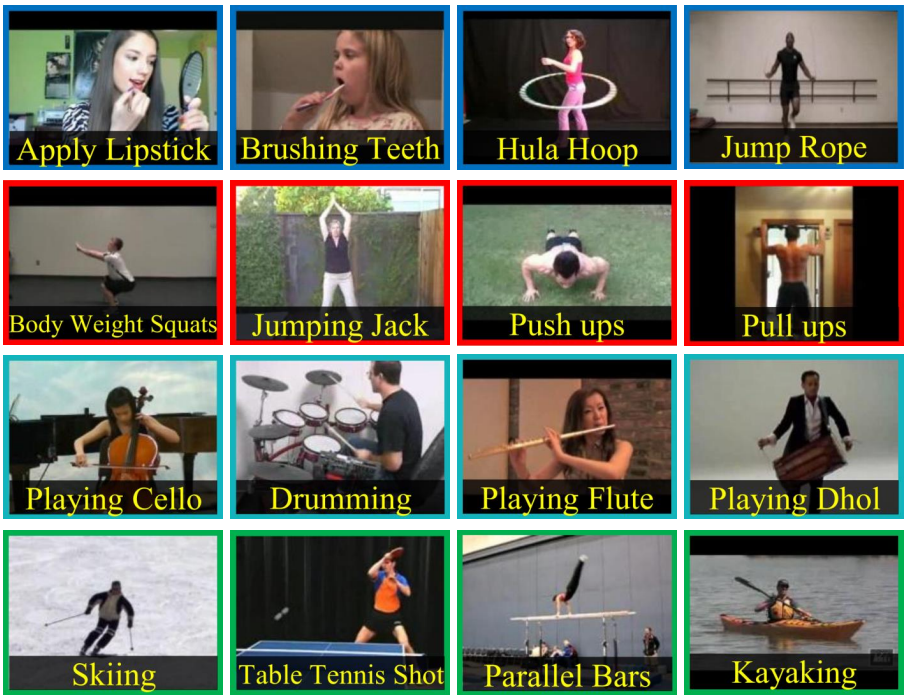


Figure 3: Some frames randomly chosen from the video clips in the UCF101 Dataset.

view shots of about 5 to 8 characters at a time. In our experiment, we used the BBT1 (The Big Bang Theory Season 1) which consists of 3,341 face videos of the first 6 episodes that are represented by block Discrete Cosine Transformation (DCT) feature as used in [62], which forms a 240×240 covariance video representation.

Evaluation criteria: We used the top 1 search accuracy (recognition accuracy), query time for one search, and storage cost of dataset to evaluate the performance of all subspace search methods under comparison. The query time for one search is the average of all test which includes both encoding time and search time for the query subspace. The storage cost of dataset includes both the training set and testing set.

4.3. Face Recognition

Since subspace is commonly used to capture the appearance of faces under varying illuminations, we test the performance of our method by searching the approximate nearest neighbors on the CMU Multi-PIE Face Database and the YouTube Faces

Table 1: The top 1 search accuracy of different methods on two face databases, with different dimension $d_q = 4, 9, 13$ of query subspaces. Here we show the accuracy of BSS with binary code length $r = 2,000$. The accuracy of our methods and vector-based hashing methods are with binary code length $r = 32$. '-' denotes that the method can not deal with the different query dimension.

Method	CMU Multi-PIE Face Database			YouTube Faces Database		
	$d_q = 4$	$d_q = 9$	$d_q = 13$	$d_q = 4$	$d_q = 9$	$d_q = 13$
HSS-KSMM	**0.9214**	**0.9321**	**0.9311**	**0.8626**	**0.8912**	**0.8745**
HSS-SMM	-	0.9316	-	-	0.8863	-
BCMC [46]	0.9023	0.9301	0.9075	0.8414	0.8821	0.8572
BSS [36]	0.8624	0.8925	0.8751	0.8072	0.8251	0.8062
GLH [42]	0.8317	0.8549	0.8401	0.7693	0.8014	0.8130
BHZ [10]	0.7925	0.7735	0.7516	0.7318	0.7015	0.6803
HashGAN-1 [57]	-	0.9055	-	-	0.8570	-
DeepBit-1 [26]	-	0.8602	-	-	0.8357	-
DH-1 [24]	-	0.8514	-	-	0.8315	-
TSH-1 [56]	-	0.8385	-	-	0.8225	-
ITQ-1 [55]	-	0.8352	-	-	0.8031	-
SH-1 [51]	-	0.8214	-	-	0.7842	-
HashGAN-2 [57]	-	0.8712	-	-	0.8152	-
DeepBit-2 [26]	-	0.8326	-	-	0.7634	-
DH-2 [24]	-	0.8210	-	-	0.7505	-
TSH-2 [56]	-	0.7924	-	-	0.7412	-
ITQ-2 [55]	-	0.7865	-	-	0.7265	-
SH-2 [51]	-	0.7815	-	-	0.7243	-

Database.

Experiment setup: In our experiment, we first resized all the face images to 32×32. For the CMU Multi-PIE Face Database, we constructed 20 9-dimensional training subspaces by randomly choosing 600 face images for each people, i.e, for each subspace with 30 face images from one person, we vectorized them to 30 1,024-

dimensional vectors and computed their first 9 principal components. The rest of the images were used for constructing query subspaces in the same manner. In practice, the dimension of the query subspace may be different from the training subspace. Therefore, according to the empirical set in previous works [10, 42, 36], we constructed query subspaces with different dimensions $d_q = 4, 9, 13$ which included dimensions smaller than, equal to, or larger than the dimensions of the training subspaces. For the YouTube Faces Database, we randomly chose 30 frame-images from each video as the training set, and the rest of the images were used for the query set. Then we constructed training subspaces and query subspaces by the same manner as used in the CMU Multi-PIE Face Database.

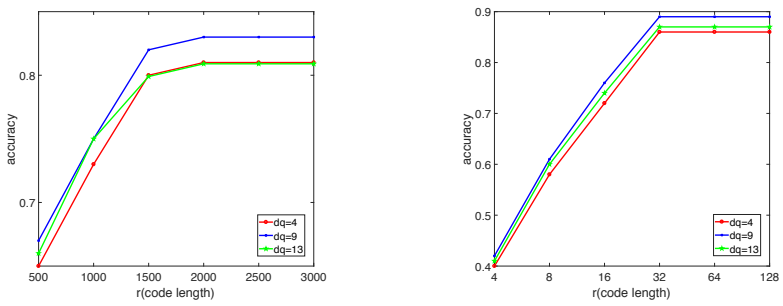
Table 2: The query time (second) for one search by different methods on two face databases, with different dimension $d_q = 4, 9, 13$ of query subspaces. Here we show the query time of BSS with binary code length $r = 2,000$. The query time of our methods and vector-based hashing methods are with binary code length $r = 32$. '-' denotes that the method can not deal with different query dimensions.

Method	CMU Multi-PIE Face Database			YouTube Faces Database		
	$d_q = 4$	$d_q = 9$	$d_q = 13$	$d_q = 4$	$d_q = 9$	$d_q = 13$
HSS-KSMM	1.357e-4	1.357e-4	1.357e-4	4.723e-4	4.723e-4	4.723e-4
HSS-SMM	-	**7.872e-5**	-	-	**2.735e-4**	-
BCMC [46]	**7.872e-5**	**7.872e-5**	**7.872e-5**	**2.735e-4**	**2.735e-4**	**2.735e-4**
BSS [36]	6.517e-3	6.517e-3	6.517e-3	0.0221	0.0221	0.0221
GLH [42]	0.1125	0.1125	0.1839	0.2415	0.2415	0.3757
BHZ [10]	0.4357	0.9245	1.425	0.9517	2.174	3.381
TSH [56]	-	1.628e-3	-	-	5.125e-3	-
ITQ [55]	-	1.041e-3	-	-	2.453e-3	-
SH [51]	-	9.256e-4	-	-	2.132e-3	-

Results: For each query subspace, we first encoded it by the learned hash functions Eq. (27) or Eq. (30). Then we searched for the nearest subspace in the training set in terms of Hamming distance between the binary codes. The top 1 search accuracy of our

Table 3: The storage cost (byte) on two face databases (include the training set and the query set). Here we show the storage cost of the state-of-the-art method BSS with binary code length $r = 2,000$, and the storage cost of our HSS-KSMM model with binary code length $r = 32$.

Method	Multi-PIE Face	YouTube Face
Original subspace	7.354e8	6.290e9
BSS [36]	4.263e6	3.209e7
HSS-KSMM	**8.175e4**	**6.846e5**



(a) BSS (b) HSS-KSMM

Figure 4: (a) The accuracy of BSS method with different code lengths on the YouTube Faces Database. (b) The accuracy of our HSS-KSMM model with different code lengths on the YouTube Faces Database.

HSS-SMM and HSS-KSMM models, compared with BHZ [10], GLH [42], BSS [36], BCMC [46] and vector-based hashing methods SH [51], ITQ [55], TSH [56],DH [24], DeepBit [26], HashGAN [57] are shown in Table 1. Here we show the accuracy of BSS with binary code length $r = 2,000$. The accuracy of our methods and the vector-based hashing methods are with binary code length $r = 32$. Due to the strong capability of classifiers to express the mapping of subspaces and binary codes, our method achieves better accuracy than the state-of-the-art subspace search method BSS with shorter code. We can see from Table 1 that both HSS-KSMM and HSS-SMM outperform the state-of-the-art unsupervised deep hashing method HashGAN. In addition, HSS-KSMM can not only deal with query subspace of different dimensions, but also achieve better accuracy than HSS-SMM and our previous work BCMC.

As the state-of-the-art method BSS also produces binary codes for subspaces, we compared our method with BSS by using different code lengths on the Youtube Faces Database. As shown in Figure 4, when $r = 32$, $d_q = 9$, our method achieves an accuracy of more than 89%. But BSS needs more than 2,000 bit binary codes to achieve an accuracy of 85%. In our experiment, BHZ stores subspace with a vector in $1,024 \times (1,024+1)/2 = 524,800$ dimensions, GLH stores subspace in 1024 dimensions, BSS stores subspace with 2,000 bits binary code, and our method can store subspace with only 32 bits binary code. Thus our method has superiority in both query time and storage cost. Table 2 and Table 3 show the query time and storage cost respectively, where the query time includes both encoding time and search time for the query subspace. It can be seen from Table 2 that our method HSS-SMM is about 80 times faster than the state-of-the-art in query time due to the shorter code length. And the storage cost is also significantly reduced by our method from Table 3. Therefore, our method can efficiently solve the high-dimensional data problem.

4.4. Action and Gesture Recognition

Subspace is also a good representation of video clips. In this section, we show the superiority of our method by action and gesture recognition experiments on the UCF101 Dataset and Cambridge Gestures Dataset.

Experiment setup: In our experiments, for the UCF101 Dataset, we randomly chose 30 videos from each action category for training, and the rest videos were used for querying. We resized all the videos to $20 \times 20 \times 30$ by taking the middle 30 frames for each sequence. For the $20 \times 20 \times 30$ data cube, we constructed 9-dimensional training subspace by vectorizing the 30 20×20 images to 30 400-dimensional vectors and computing their first 9 principal components. The query subspaces were in different dimensions $d_q = 4, 9, 13$.

For the Cambridge Gestures Dataset, we randomly chose 60 image sequences for training, and the rest 40 image sequences were used for querying. Then we constructed training subspaces and query subspaces by the same manner used in the UCF101 dataset. For each video sample, we resized it to $20 \times 20 \times 30$ by taking the middle 30 frames for each sequence (see examples in Figure 5).

Figure 5: Some image sequences randomly chosen from the Cambridge Gestures Dataset.

Results: We calculated the top 1 search accuracy for each method respectively. The results are shown in Table 4. Here we show the accuracy of BSS with binary code length $r = 3,000$. The accuracy of our methods and vector-based hashing methods are with binary code length $r = 64$. We can see from Table 4 that our proposed methods HSS-SMM and HSS-KSMM both achieve better recognition accuracy than previous works BHZ, GLH, BSS, BCMC and the state-of-the-art unsupervised deep hashing method HashGAN. The search accuracies of our methods and BSS with different binary code lengths are shown in Figure 6. It can be seen that our methods can generate better recognition accuracy with shorter binary code. Table 5 and Table 6 show the query time and storage cost of all the methods on the UCF101 dataset and the Cambridge Gestures dataset, where the query time includes both encoding time and search time for the query subspace. Table 5 shows that our method HSS-SMM is about 80 times faster than the state-of-the-art in query time thanks to the shorter code length. Table 6 shows that both HSS-SMM and HSS-KSMM significantly reduce the storage cost of subspace data. This suggests that our method is potentially more

Table 4: The top 1 search accuracy of different methods on the UCF101 dataset and Cambridge Gestures dataset, with different dimensions $d_q = 4, 9, 13$ of query subspaces. Here we show the accuracy of BSS with binary code length $r = 3,000$. The accuracy of our methods and vector-based hashing methods are with binary code length $r = 64$. '-' denotes that the method can not deal with the different query dimensions.

Method	UCF101			Cambridge Gestures		
	$d_q = 4$	$d_q = 9$	$d_q = 13$	$d_q = 4$	$d_q = 9$	$d_q = 13$
HSS-KSMM	**0.7345**	**0.7526**	**0.7501**	**0.9410**	**0.9535**	**0.9468**
HSS-SMM	-	0.7483	-	-	0.9424	-
BCMC [46]	0.7058	0.7365	0.7184	0.9165	0.9402	0.9260
BSS [36]	0.7014	0.7155	0.7123	0.9136	0.9342	0.9255
GLH [42]	0.6612	0.6825	0.6814	0.8415	0.8532	0.8463
BHZ [10]	0.6085	0.6138	0.6105	0.7518	0.7715	0.7603
HashGAN-1 [57]	-	0.7235	-	-	0.9210	-
DeepBit-1 [26]	-	0.7005	-	-	0.9016	-
DH-1 [24]	-	0.6849	-	-	0.8852	-
TSH-1 [56]	-	0.6524	-	-	0.8710	-
ITQ-1 [55]	-	0.6306	-	-	0.8342	-
SH-1 [51]	-	0.6279	-	-	0.8217	-
HashGAN-2 [57]	-	0.6618	-	-	0.8725	-
DeepBit-2 [26]	-	0.6385	-	-	0.8475	-
DH-2 [24]	-	0.6327	-	-	0.8326	-
TSH-2 [56]	-	0.6143	-	-	0.7911	-
ITQ-2 [55]	-	0.5925	-	-	0.7683	-
SH-2 [51]	-	0.5812	-	-	0.7526	-

suitable than previous methods on large-scale dataset with high dimensionality for real world applications.

Table 5: The query time (second) for one search by different methods on the UCF101 dataset and Cambridge Gestures dataset, with different dimensions $d_q = 4, 9, 13$ of query subspaces. Here we show the query time of BSS with binary code length $r = 3,000$. The query time of our methods and vector-based hashing methods are with binary code length $r = 64$. '-' denotes that the method can not deal with the different query dimensions.

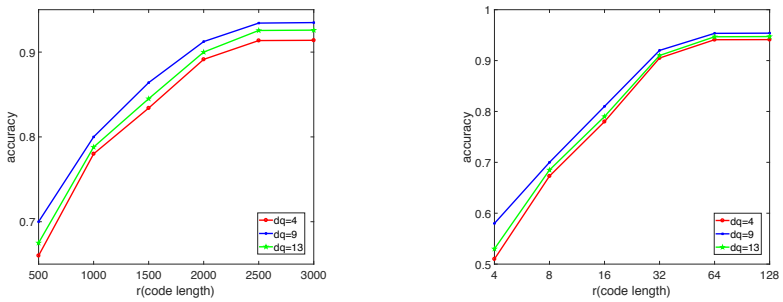
Method	UCF101			Cambridge Gestures		
	$d_q = 4$	$d_q = 9$	$d_q = 13$	$d_q = 4$	$d_q = 9$	$d_q = 13$
HSS-KSMM	7.934e-4	7.934e-4	7.934e-4	1.215e-3	1.215e-3	1.215e-3
HSS-SMM	-	**6.755e-4**	-	-	**9.025e-4**	-
BCMC [46]	**6.755e-4**	**6.755e-4**	**6.755e-4**	**9.025e-4**	**9.025e-4**	**9.025e-4**
BSS [36]	6.127e-2	6.127e-2	6.127e-2	6.921e-2	6.921e-2	6.921e-2
GLH [42]	0.9251	0.9251	1.436	1.316	1.316	1.935
BHZ [10]	4.125	8.274	12.08	4.312	9.235	15.036
TSH [56]	-	1.025e-2	-	-	1.835e-2	-
ITQ [55]	-	8.120e-3	-	-	1.147e-2	-
SH [51]	-	7.836e-3	-	-	9.752e-3	-

Table 6: The storage cost (byte) of the UCF101 dataset (include training set and query set). Here we show the storage cost of BSS with binary code length $r = 3,000$, and the storage cost of our method with binary code length $r = 64$.

Method	UCF101	Cambridge Gestures
Original subspace	5.247e9	3.521e8
BSS [36]	5.047e7	3.231e6
HSS-KSMM	**9.851e5**	**7.231e4**

4.5. Video Retrieval

In this section, we show the superiority of our method by a video retrieval experiment on the Big Bang Theory (BBT). We compared our method with the state-of-the-



(a) BSS (b) HSS-KSMM

Figure 6: (a) The accuracy of state-of-the-art method BSS with different code lengths on the Cambridge Gestures Dataset. (b) The accuracy of our HSS-KSMM model with different code lengths on the Cambridge Gestures Dataset.

art query-by-image video retrieval methods HER [43], and IBC, BBC [38].

We constructed two sets of experiments, query-by-image and query-by-video video retrieval. For query-by-image experiment, the experiment set of HER, IBC, and BBC are as done in [43] and [38]. For our method, we randomly extracted 300 videos from the 3341 videos for training and 100 videos from the rest as queries for 10 trials. Then we calculated the average retrieval accuracy on 10 testing sets. We first represent training videos as subspaces and learn the corresponding binary codes. Then we train a set of classifiers as hash functions. The binary codes of query images can be efficiently obtained by the learned classifiers, i.e. Eq. (30), where an image is considered as one-dimensional subspace. The average retrieval accuracy with different length of hash code is shown in Table 7. Our proposed method outperforms the state-of-the-art methods.

Although HER, IBC, and BBC are designed for query-by-image video retrieval, they can also be applied to query-by-video video retrieval. We use their definition of distance between projected image and video subspace to calculate the distance between video subspaces, and then apply their coding method. For our method, we first represent training and query videos as subspaces and then use our ANS retrieval algorithm. Table 7 shows the result that our method significantly performs better than the state-of-the-art methods.

Table 7: Comparison with the state-of-the-art methods on BBT1 database with different length of hash code.

Method	Query-by-image				Query-by-video			
	16-bit	32-bit	64-bit	128-bit	16-bit	32-bit	64-bit	128-bit
HSS-KSMM	**0.5576**	**0.5973**	**0.6052**	**0.6136**	**0.6051**	**0.6486**	**0.6604**	**0.6652**
HSS-SMM	0.6024	0.6458	0.6587	0.6602	0.6024	0.6458	0.6587	0.6602
BCMC [46]	0.5520	0.5915	0.6014	0.6083	0.6013	0.6415	0.6523	0.6589
BBC [38]	0.5080	0.5401	0.5643	0.5718	0.5378	0.5537	0.5752	0.5968
IBC [38]	0.5152	0.5369	0.5561	0.5645	0.5425	0.5542	0.5736	0.5904
HER [43]	0.5049	0.5227	0.5490	0.5539	0.5214	0.5431	0.5614	0.5854
HashGAN-1 [57]	0.4832	0.5250	0.5389	0.5670	0.5023	0.5350	0.5602	0.5900
DeepBit-1 [26]	0.4455	0.4731	0.4928	0.5292	0.4716	0.4982	0.5376	0.5611
DH-1 [24]	0.4225	0.4468	0.4816	0.5127	0.4524	0.4817	0.5139	0.5475
TSH-1 [56]	0.3613	0.3752	0.3910	0.4154	0.3626	0.3912	0.4165	0.4321
ITQ-1 [55]	0.3310	0.3421	0.3573	0.3812	0.3428	0.3547	0.3761	0.3923
SH-1 [51]	0.3019	0.3096	0.3215	0.3541	0.3064	0.3186	0.3421	0.3596
HashGAN-2 [57]	0.4125	0.4370	0.4632	0.4810	0.4502	0.4886	0.4980	0.5103
DeepBit-2 [26]	0.3728	0.3886	0.4107	0.4239	0.4079	0.4305	0.4567	0.4793
DH-2 [24]	0.3758	0.3852	0.4096	0.4125	0.4058	0.4267	0.4513	0.4689
TSH-2 [56]	0.3325	0.3407	0.3528	0.3627	0.3198	0.3367	0.3452	0.3526
ITQ-2 [55]	0.3142	0.3158	0.3263	0.3381	0.3153	0.3241	0.3387	0.3504
SH-2 [51]	0.2932	0.3015	0.3124	0.3256	0.2916	0.3104	0.3221	0.3268

5. Conclusion

We have presented a hashing based method to solve the ANS search problem. Our method first learns the binary codes for training subspaces following a similarity p-reserving criterion. Then the learned binary codes are used to train a set of matrix binary classifiers as hash functions. The compact binary code for query subspace can be efficiently generated by these matrix classifiers. Our method is further expanded to cope with the case that query subspaces have different dimensions from the target database. The main contribution of our proposed method is that we do not transform

subspace into vector form, which fully utilize the subspace structural information. Several experiments on face datasets and video datasets verify that our method can achieve significant improvement in query time and storage cost with superior accuracy, when compared with several state-of-the-art approaches. Furthermore, we also compare our method with several vector-based hashing methods. The results also show the superiority of our subspace matrix based searching scheme. For future work, we will expand this method with manifold based subspace similarity measurement such as Grassmann kernels defined by KL-divergence.

References

[1] X. Wang, X. Tang, Random sampling for subspace face recognition, International Journal of Computer Vision 70 (1) (2006) 91–104.

[2] X. Zhang, Y. Jia, A linear discriminant analysis framework based on random subspace for face recognition, Pattern Recognition 40 (9) (2007) 2585–2591.

[3] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, Y. Ma, Robust face recognition via sparse representation, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2) (2009) 210–227.

[4] L. Wang, X. Wang, J. Feng, Subspace distance analysis with application to adaptive bayesian algorithm for face recognition, Pattern recognition 39 (3) (2006) 456–464.

[5] M. Blank, L. Gorelick, E. Shechtman, M. Irani, R. Basri, Actions as space-time shapes, in: International Conference on Computer Vision, Vol. 2, 2005, pp. 1395–1402.

[6] G. Liu, S. Yan, Latent low-rank representation for subspace segmentation and feature extraction, in: International Conference on Computer Vision, 2011, pp. 1615–1622.

[7] A. W. Fitzgibbon, A. Zisserman, Joint manifold distance: a new approach to appearance based clustering, in: Computer Vision and Pattern Recognition, Vol. 1, 2003.

[8] H. Zhang, A. C. Berg, M. Maire, J. Malik, Svm-knn: Discriminative nearest neighbor classification for visual category recognition, in: Computer Vision and Pattern Recognition, Vol. 2, 2006, pp. 2126–2136.

[9] V. Blanz, T. Vetter, Face recognition based on fitting a 3d morphable model, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (9) (2003) 1063–1074.

[10] R. Basri, T. Hassner, L. Zelnikmanor, Approximate nearest subspace search, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (2) (2011) 266–278.

[11] R. Vidal, Subspace clustering, IEEE Signal Processing Magazine 28 (2) (2011) 52–68.

[12] D. S. Broomhead, M. J. Kirby, The whitney reduction network: A method for computing autoassociative graphs, Neural Computation 13 (11) (2001) 2595–2616.

[13] D. Broomhead, M. Kirby, Dimensionality reduction using secant-based projection methods: The induced dynamics in projected systems, Nonlinear Dynamics 41 (1-3) (2005) 47–67.

[14] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, A. Y. Wu, An optimal algorithm for approximate nearest neighbor searching fixed dimensions, Journal of the ACM (JACM) 45 (6) (1998) 891–923.

[15] E. Kushilevitz, R. Ostrovsky, Y. Rabani, Efficient search for approximate nearest neighbor in high dimensional spaces, SIAM Journal on Computing 30 (2) (2000) 457–474.

[16] M. Muja, Fast approximate nearest neighbors with automatic algorithm configuration, in: International Conference on Computer Vision Theory and Application Vissapp, 2009, pp. 331–340.

[17] A. Andoni, P. Indyk, R. Krauthgamer, H. L. Nguyen, Approximate line nearest neighbor in high dimensions, in: Proceedings of ACM-SIAM Symposium on Discrete Algorithms, 2009, pp. 293–301.

[18] B. Wang, X. Liu, K. Xia, K. Ramamohanarao, D. Tao, Random angular projection for fast nearest subspace search, in: Pacific Rim Conference on Multimedia, Springer, 2018, pp. 15–26.

[19] M. Datar, N. Immorlica, P. Indyk, V. S. Mirrokni, Locality-sensitive hashing scheme based on p-stable distributions, in: Symposium on Computational Geometry, 2004, pp. 253–262.

[20] D. Zhang, J. Wang, D. Cai, J. Lu, Self-taught hashing for fast similarity search, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2010, pp. 18–25.

[21] X. Liu, J. He, B. Lang, Multiple feature kernel hashing for large-scale visual search, Pattern Recognition 47 (2) (2014) 748–757.

[22] J. Song, L. Gao, L. Liu, X. Zhu, N. Sebe, Quantization-based hashing: a general framework for scalable image and video retrieval, Pattern Recognition 75 (2018) 175–187.

[23] X. Bai, C. Yan, H. Yang, L. Bai, J. Zhou, E. R. Hancock, Adaptive hash retrieval with kernel based similarity, Pattern Recognition 75 (2018) 136–148.

[24] V. E. Liong, J. Lu, G. Wang, P. Moulin, J. Zhou, Deep hashing for compact binary codes learning, in: Computer Vision and Pattern Recognition, 2015, pp. 2475–2483.

[25] Z. Dong, S. Jia, T. Wu, M. Pei, Face video retrieval via deep learning of binary hash representations., in: AAAI Conference on Artificial Intelligence, 2016, pp. 3471–3477.

[26] K. Lin, J. Lu, C. S. Chen, J. Zhou, Learning compact binary descriptors with unsupervised deep neural networks, in: Computer Vision and Pattern Recognition, 2016, pp. 1183–1192.

[27] H. Liu, R. Wang, S. Shan, X. Chen, Deep supervised hashing for fast image retrieval, in: Computer Vision and Pattern Recognition, 2016, pp. 2064–2072.

[28] Z. Dong, C. Jing, M. Pei, Y. Jia, Deep cnn based binary hash video representations for face retrieval, Pattern Recognition 81 (2018) 357–369.

[29] X. Wang, X. Tang, A unified framework for subspace face recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (9) (2004) 1222–1228.

[30] J. Hamm, D. D. Lee, Grassmann discriminant analysis: a unifying view on subspace-based learning, in: International Conference on Machine Learning, 2008, pp. 376–383.

[31] J.-M. Chang, M. Kirby, H. Kley, C. Peterson, B. Draper, J. R. Beveridge, Recognition of digital images of the human face at ultra low resolution via illumination spaces, in: Asian Conference on Computer Vision, Springer, 2007, pp. 733–743.

[32] J. R. Beveridge, B. A. Draper, J.-M. Chang, M. Kirby, H. Kley, C. Peterson, Principal angles separate subject illumination spaces in ydb and cmu-pie, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2) (2008) 351–363.

[33] R. Basri, T. Hassner, L. Zelnikmanor, Approximate nearest subspace search with applications to pattern recognition, in: Computer Vision and Pattern Recognition, 2007, pp. 1–8.

[34] R. Basri, T. Hassner, L. Zelnik-Manor, A general framework for approximate nearest subspace search, in: International Conference on Computer Vision Workshops, 2009, pp. 109–116.

[35] J. Ji, J. Li, S. Yan, Q. Tian, B. Zhang, Similarity-preserving binary signature for linear subspaces., in: AAAI Conference on Artificial Intelligence, 2014, pp. 2767–2772.

[36] J. Ji, J. Li, Q. Tian, S. Yan, B. Zhang, Angular-similarity-preserving binary signatures for linear subspaces, IEEE Transactions on Image Processing 24 (11) (2015) 4372–80.

[37] A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, in: International Conference on Very Large Data Bases, 1999, pp. 518–529.

[38] R. Xu, Y. Yang, F. Shen, N. Xie, H. T. Shen, Efficient binary coding for subspace-based query-by-image video retrieval, in: Proceedings of the 2017 ACM on Multimedia Conference, ACM, 2017, pp. 1354–1362.

[39] R. G. Baraniuk, M. B. Wakin, Random projections of smooth manifolds, Foundations of computational mathematics 9 (1) (2009) 51–77.

[40] T. Marrinan, J. R. Beveridge, B. Draper, M. Kirby, C. Peterson, Flag manifolds for the characterization of geometric structure in large data sets, in: Numerical Mathematics and Advanced Applications-ENUMATH 2013, Springer, 2015, pp. 457–465.

[41] S. O'Hara, B. A. Draper, Scalable action recognition with a subspace forest, in: Computer Vision and Pattern Recognition, 2012, pp. 1210–1217.

[42] X. Wang, S. Atev, J. Wright, G. Lerman, Fast subspace search via grassmannian based hashing, in: International Conference on Computer Vision, 2013, pp. 2776–2783.

[43] Y. Li, R. Wang, Z. Huang, S. Shan, X. Chen, Face video retrieval with image query via hashing across euclidean space and riemannian manifold, in: Computer Vision and Pattern Recognition, 2015, pp. 4758–4767.

[44] A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, in: IEEE Symposium on Foundations of Computer Science, 2006, pp. 459–468.

[45] L. Luo, Y. Xie, Z. Zhang, W. Li, Support matrix machines, International Conference on Machine Learning (2015) 938–947.

[46] L. Zhou, X. Bai, X. Liu, J. Zhou, Binary coding by matrix classifier for efficient subspace retrieval, in: Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval, ACM, 2018, pp. 82–90.

[47] F. Shen, C. Shen, W. Liu, H. Tao Shen, Supervised discrete hashing, in: Computer Vision and Pattern Recognition, 2015, pp. 37–45.

[48] G. H. Golub, C. F. Van Loan, Matrix computations. 1996, Johns Hopkins University, Press, Baltimore, MD, USA (1996) 374–426.

[49] H. Hotelling, Relations between two sets of variates, in: Breakthroughs in statistics, Springer, 1992, pp. 162–190.

[50] A. Edelman, T. A. Arias, S. T. Smith, The geometry of algorithms with orthogonality constraints, SIAM journal on Matrix Analysis and Applications 20 (2) (1998) 303–353.

[51] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing., in: Conference on Neural Information Processing Systems, 2008, pp. 1753–1760.

[52] T. Goldstein, Donoghue, S. Setzer, Fast alternating direction optimization methods, Siam Journal on Imaging Sciences 7 (3).

[53] J. Sun, Y. Zhang, J. Wright, Efficient point-to-subspace query in ℓ_1 with application to robust face recognition, European Conference on Computer Vision (2012) 416–429.

[54] H. Pirsiavash, D. Ramanan, C. Fowlkes, Bilinear classifiers for visual recognition, in: International Conference on Neural Information Processing Systems, 2009, pp. 1482–1490.

[55] Y. Gong, S. Lazebnik, Iterative quantization: A procrustean approach to learning binary codes, in: Computer Vision and Pattern Recognition, 2011, pp. 817–824.

[56] G. Lin, C. Shen, D. Suter, A. V. D. Hengel, A general two-step approach to learning-based hashing, in: International Conference on Computer Vision, 2013, pp. 2552–2559.

[57] K. Ghasedi Dizaji, F. Zheng, N. Sadoughi, Y. Yang, C. Deng, H. Huang, Unsupervised deep generative adversarial hashing network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3664–3673.

[58] R. Gross, I. Matthews, J. Cohn, T. Kanade, S. Baker, Multi-pie., Image and Vision Computing 28 (5) (2010) 807–813.

[59] L. Wolf, T. Hassner, I. Maoz, Face recognition in unconstrained videos with matched background similarity, in: Computer Vision and Pattern Recognition, 2011, pp. 529–534.

[60] K. Soomro, A. Roshan Zamir, M. Shah, UCF101: A dataset of 101 human actions classes from videos in the wild, in: CRCV-TR-12-01, 2012.

[61] T.-K. Kim, S.-F. Wong, R. Cipolla, Tensor canonical correlation analysis for action classification, in: Computer Vision and Pattern Recognition, 2007, pp. 1–8.

[62] M. Bauml, M. Tapaswi, R. Stiefelhagen, Semi-supervised learning with constraints for person identification in multimedia data, in: Computer Vision and Pattern Recognition, 2013, pp. 3602–3609.

*Author Biography

Lei Zhou received the Bachelor's degree in 2016 from the School of Mathematics and Systems Science, Beihang University, Beijing, China, where he is currently working toward the Ph.D. degree at the School of Computer Science and Engineering.

His current research interests include machine learning, computer vision, and image processing.

Xiao Bai received the B.Eng. degree in computer science from Beihang University of China, Beijing, China, in 2001, and the Ph.D. degree in computer science from the University of York, York, U.K., in 2006.

He was a Research Officer (Fellow, Scientist) with the Computer Science Department, University of Bath, until 2008. He is currently a Full Professor with the School of Computer Science and Engineering, Beihang University. He has authored or co-authored more than 60 papers in journals and refereed conferences. His current research interests include pattern recognition, image processing, and remote sensing image analysis.

Xianglong Liu received the B.S. and Ph.D. degrees in computer science from Beihang University, Beijing, in 2008 and 2014. From 2011 to 2012, he visited the Digital Video and Multimedia Laboratory, Columbia University as a joint Ph.D. student. He is currently an Assistant Professor with Beihang University.

His research interests include machine learning, computer vision, and multimedia information retrieval.

Jun Zhou received the B.S. degree in computer science and the B.E. degree in international business from Nanjing University of Science and Technology, Nanjing, China, in 1996 and 1998, respectively. He received the M.S. degree in computer science from Concordia University, Montreal, Canada, in 2002, and the Ph.D. degree from the University of Alberta, Edmonton, Canada, in 2006. He is a senior lecturer in the School of Information and Communication Technology at Griffith University, Nathan, Australia. Previously, he had been a research fellow in the Research School of Computer Science at the Australian National University, Canberra, Australia, and a researcher in the Canberra Research Laboratory, NICTA, Australia.

His research interests include pattern recognition, computer vision and remote sensing with their applications to spectral imaging and environmental informatics.

Edwin R. Hancock holds a BSc degree in physics (1977), a PhD degree in high-energy physics (1981) and a D.Sc. degree (2008) from the University of Durham, and a doctorate Honoris Causa from the University of Alicante in 2015. He is Professor in the Department of Computer Science, where he leads a group of some faculty, research staff, and PhD students working in the areas of computer vision and pattern recognition. His main research interests are in the use of optimization and probabilistic methods for high and intermediate level vision. He is a fellow of the International Association for Pattern Recognition and the IEEE. He is currently Editor-in-Chief of the journal Pattern Recognition, and was founding Editor-in-Chief of IET Computer Vision from 2006 until 2012. He has also been a member of the editorial boards of the journals IEEE Transactions on Pattern Analysis and Machine Intelligence, Pattern Recognition, Computer Vision and Image Understanding, Image and Vision Computing, and the International Journal of Complex Networks. He is currently Vice President of the IAPR.