Spatial Pyramid Covariance-Based Compact Video Code for Robust Face Retrieval in TV-Series

Yan Li, *Student Member, IEEE*, Ruiping Wang, *Member, IEEE*, Zhen Cui, *Member, IEEE*, Shiguang Shan, *Senior Member, IEEE*, and Xilin Chen, *Fellow, IEEE*

***Abstract*— We address the problem of face video retrieval in TV-series, which searches video clips based on the presence of specific character, given one face track of his/her. This is tremendously challenging because on one hand, faces in TV-series are captured in largely uncontrolled conditions with complex appearance variations, and on the other hand, retrieval task typically needs efficient representation with low time and space complexity. To handle this problem, we propose a compact and discriminative representation for the huge body of video data, named compact video code (CVC). Our method first models the face track by its sample (i.e., frame) covariance matrix to capture the video data variations in a statistical manner. To incorporate discriminative information and obtain more compact video signature suitable for retrieval, the high-dimensional covariance representation is further encoded as a much lower dimensional binary vector, which finally yields the proposed CVC. Specifically, each bit of the code, i.e., each dimension of the binary vector, is produced via supervised learning in a max margin framework, which aims to make a balance between the discriminability and stability of the code. Besides, we further extend the descriptive granularity of covariance matrix from traditional pixel-level to more general patch-level, and proceed to propose a novel hierarchical video representation named spatial pyramid covariance along with a fast calculation method. Face retrieval experiments on two challenging TV-series video databases, i.e., *the Big Bang Theory* and *Prison Break*, demonstrate the competitiveness of the proposed CVC over the state-of-the-art retrieval methods. In addition, as a general video matching algorithm, CVC is also evaluated in traditional video face recognition task on a standard Internet database, i.e., *YouTube Celebrities*, showing its quite promising performance by using an extremely compact code with only 128 bits.**

***Index Terms*— Face video retrieval, covariance matrix, spatial pyramid covariance, compact video code, binary code learning.**

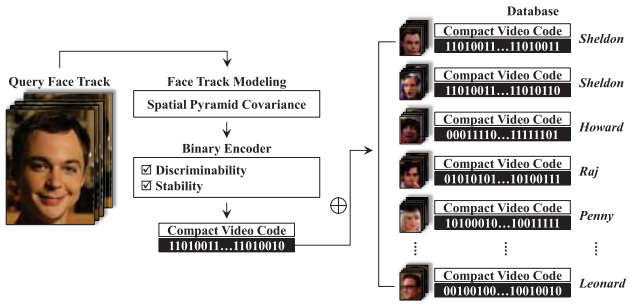


Fig. 1. Illustration of the proposed method. Given a face track of one character as query, we extract the proposed Compact Video Code (CVC) to represent it and use Hamming distance to retrieve face tracks of the specific character in database, which are also encoded in the form of CVC.

I. Introduction

FACE video retrieval in general is to retrieve shots containing particular person given one face track (i.e., faces detected in consecutive frames that share a large enough number of tracked points) of him/her [1]. It is a promising research direction with increasing demands, especially in the era of Internet multimedia considering huge body of video data (e.g., various types of videos can be found on the video sharing sites like YouTube, such as movies, newscasts, sitcoms, sports, commercials, and homemade videos, etc). Finding a specific person in videos is crucial to understand and retrieve videos. There are a wide range of applications relying on it, for example: 'intelligent fast-forwards' - where the video jumps to the next shot containing the specific character; retrieval of all the shots containing a particular family member from thousands of short videos captured by a digital camera; and rapid locating and tracking of suspects from masses of city surveillance videos. In this paper, we mainly focus on the problem of face video retrieval in TV-series with character's one face track as query, as depicted in Fig. 1.

The key technique for face video retrieval is face recognition which has long been established as one of the most active research areas in computer vision. However, face video retrieval has its unique features compared with traditional face recognition tasks. Specifically, different from traditional image based face recognition, video provides rich and redundant information, which can be exploited to resolve the inherent

Manuscript received March 31, 2016; revised August 16, 2016 and September 26, 2016; accepted September 27, 2016. Date of publication October 10, 2016; date of current version October 28, 2016. This work was supported in part by the 973 Program under Contract 2015CB351802, in part by the Natural Science Foundation of China under Contract 61390511, Contract 61222211, Contract 61379083, and Contract 61271445, in part by the Strategic Priority Research Program of CAS under Grant XDB02070004, in part by the Youth Innovation Promotion Association CAS under Grant 2015085, and in part by the Natural Science Foundation of Zhejiang Province, China, under Contract LQ15F020005. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Stefan Winkler. *(Corresponding author: Shiguang Shan.)*

Y. Li, R. Wang, and X. Chen are with the Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China, and also with the University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: yan.li@vipl.ict.ac.cn; wangruiping@ict.ac.cn; xlchen@ict.ac.cn).

Z. Cui is with Southeast University 210096, China (e-mail: zhen.cui@seu.edu.cn).

S. Shan is with the Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China, also with the University of Chinese Academy of Sciences, Beijing 100049, China, and also with the CAS Center for Excellence in Brain Science and Intelligence Technology, Shanghai 200031, China (e-mail: sgshan@ict.ac.cn).

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

Digital Object Identifier 10.1109/TIP.2016.2616297

ambiguities of still image based recognition like sensitivity to variations caused by lighting, pose, resolution and occlusion [1]. However, how to effectively utilize the rich information of video needs to be considered adequately. In this paper, we utilize the second-order statistic covariance matrix which has been proved natural and efficient for front-end video representation [2]–[4]. Moreover, to overcome the shortcoming of raw covariance matrix, i.e., high dimensionality and alignment sensitivity, we extend the descriptive granularity of covariance matrix (will be written as covariance henceforth, for short) from traditional pixel-level to more general patch-level with a fast calculation method based on theoretical derivation. On this basis, we proceed to propose a complete multi-granularity video representation by organizing a series of different parameterized patch-level covariances in a hierarchical manner named by us as Spatial Pyramid Covariance (SPC). Another unique feature of face video retrieval is the strong demand for compact and discriminative video representation for both time fast and space saving search. Despite covariance has a certain degree of discrimination ability due to its modeling of video data variations, it does not involve any supervised information which is crucial for retrieval. Besides, it is not compact enough due to its high dimension for fast retrieval especially when the database is very large. To this end, we further encode the covariance-based video representation to Compact Video Code (CVC) as the final video signature, which comes in the form of a much lower-dimensional binary code, where each bit of the code is learned by explicitly optimizing for discrimination in a max margin framework. By doing this discriminability and stability are considered jointly.

To verify the effectiveness of the proposed method, we conduct retrieval experiments on two challenging TV-series databases, i.e., *the Big Bang Theory* and *Prison Break*. Experimental results demonstrate the superiority of the proposed CVC over state-of-the-art retrieval methods. In addition, the proposed method is also evaluated in traditional video face recognition task on a standard Internet benchmark, i.e., *YouTube Celebrities* [5]. It is shown that, as a general video matching algorithm, the proposed method shows quite promising performance by using a rather compact code with only 128 bits.

The rest of this paper is organized as follows: Section II discusses the related work of the proposed method. Then Section III describes the covariance-based video modeling as well as the metrics of covariance. Next, Section IV presents the binary code learning framework along with two constraints, i.e., discriminability and stability. After that, Section V exhaustively evaluates the proposed method on different video classification tasks. Finally, we end with a summary of conclusions and future work in Section VI.

II. Related Work

A. Video-Related Face Applications

Recent years have witnessed more and more studies on video-related face applications [6]–[15], especially the entertainment videos, e.g., films, TV-series. For instance, Everingham *et al*. [7] studied the problem of labelling appearances of characters in TV-series and films; Arandjelović and Cipolla [8] addressed the problem of automatically determining the cast of feature-length film; Cinbis *et al*. [9] investigated the identification problem for face tracks of TV-series; Parkhi *et al*. [13] dealt with the problem of video face verification with a compact and discriminative vector representation; Sivic *et al*. [6] attempted to retrieve shots containing particular person in video using an imaged face as query; Li *et al*. [14] extended video-video retrieval to a cross-modality scenario, i.e., face video retrieval with image query, and modeled such problem as a heterogeneous hash learning cross Euclidean space (still images) and Riemannian manifold (videos); and Dong *et al*. [15] developed a new deep convolutional neural network to learn discriminative and compact binary face representations for face video retrieval, and the network integrated feature extraction and hash learning into a unified optimization framework for the optimal compatibility of feature extractor and hash functions. More recently, an increasing number of peripheral information are utilized for performance boosting, e.g., Ortiz *et al*. [11] studied the problem of face track identification using a collected large dictionary of still face images for assist; and Bäuml *et al*. [10] took advantage of subtitles and fan transcripts to implement character identification in TV-series. While most of such previous works [6]–[13] have been devoted to an end-to-end system, including those preprocessing stages such as shot boundary detection, face detection, facial landmark localization, tracking, and face track extraction, etc., it is generally believed that the pivotal technical components of video-related face applications (especially face video retrieval) lie in the video data modeling and the subsequent discriminant and compact representation learning, which are the exact topics of this paper.

B. Video Modeling

As video is comprised of frames (i.e., images), in practice it is often treated as image set. Compared with treating video as separated frames and processing it frame by frame, holistic modeling methods [2]–[4], [16]–[21] gradually exhibit their advantages of not only compact representation but also superior performance, after the pioneering work of Yamaguchi *et al*. [16]. One class of prevalent methods is to use subspace learning techniques to account for the image set variability globally either by a single linear subspace [17], or by a more sophisticated manifold [18], [19]. However, the linear subspace modeling cannot well accommodate the case in real world when the set is of small size but has complex data variations. As also indicated in [20], linear subspace modeling has the limitation that it incorporates only relatively weak information, i.e., subspace angles, about the location and boundary of instances in the input space. Another class of prevalent methods is based on affine subspace [20], [21], where image set is approximated with a more theoretically principled affine subspace model and closest virtual points are matched via a convex optimization. While intra-class variations can be effectively handled, such methods are still susceptible to the presence of outliers and have relatively high computational cost due to their inherent single sample-based

matching mechanism [2], [17]. More recently, covariance-based methods [2]–[4] show their superiority for image set modeling. As the raw second-order statistic of image set, covariance which lies on a Riemannian manifold provides a natural representation for any set size and any image feature, and therefore characterizes the complicated set structure more faithfully [2]–[4]. Also, as indicated in [2], linear subspace models originate from an eigen-decomposition of the covariance while discarding some important information. Taking such into consideration, we resort to covariance for representing videos in this paper.

C. Binary Code Learning

While above methods have gained successes in image set classification, high-dimensional representation limits their applicability to the video retrieval scenario which typically requires not only accurate but also compact and efficient representation of video for fast search. To obtain compact representation, binary code (a.k.a., hash/hashing code) is a natural solution, as it is quite efficient to match with sub-linear time complexity, and is able to index a huge size of data with very short code length [22]. The basic idea of hashing is to learn similarity preserving binary codes for data representation [23], i.e., each data point will be hashed into a compact binary string, and similar data points in the original feature space should be hashed into close points in the binary code space (always Hamming space).

Existing hashing methods [23]–[38] can be roughly divided into two types, i.e., data independent and data dependent. Representative data independent hashing methods include the pioneering Locality-Sensitive Hashing (LSH) [24] and its variants [25], [26], and Shift-Invariant Kernel Hashing (SIKH) [27]. LSH and its variants utilize random projections as hash functions which are independent from training data, and slightly different from LSH, SIKH utilizes a shifted cosine function to generate hash codes. Due to their inherent property that the original metrics are asymptotically preserved in the target Hamming space with increasing code length, LSH-related methods usually achieve satisfactory performance at the expense of long codes. To overcome such disadvantage, another type, i.e., data dependent methods, have come into being with their hash functions being learned from training data. Representative methods include Spectral Hashing (SH) [28], and ITerative Quantization (ITQ) [29], *etc*. SH is calculated by thresholding a subset of eigenvectors of the Laplacian of the similarity graph, ITQ is implemented by iteratively minimizing the quantization error of projecting data from the original feature space to the target Hamming space.

Most recently, an increasing number of methods attempt to integrate discriminant supervised information into the hash functions learning for performance boosting, and these form a sub-branch of data dependent hashing methods, a.k.a., supervised hashing methods. Representative methods include Semi-Supervised Hashing (SSH) [30], Discriminative Binary Code (DBC) [31], Kernel-based Supervised Hashing (KSH) [32], Supervised ITerative Quantization (SITQ) [29], and Minimal Loss Hashing (MLH) [36].

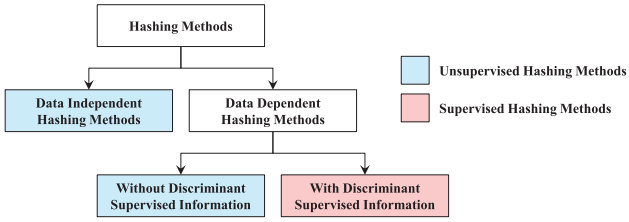


Fig. 2. Types of hashing methods.

Technically, SSH formulates its learning as minimizing empirical error on the labeled data while maximizing variance and independence of hash bits over the labeled and unlabeled data; DBC can be used for visual attribute discovery by jointly optimizing the discriminability and predictability of hash codes; KSH makes use of kernel-based hash functions by dexterously utilizing the algebraic equivalence between a Hamming distance and a code inner product; SITQ is implemented by simply replacing the Principal Component Analysis (PCA) [39] preprocessing in ITQ with Canonical Correlation Analysis (CCA) [40], and MLH as a general purpose hashing method is appropriate for both unsupervised and supervised scenarios by formulating the approach to learn similarity-preserving binary codes based on structured prediction with latent variables and a hinge-like loss function, and the learning algorithm of MLH is online, efficient, and scales well to large code lengths. We summarize the above types of hashing methods in Fig. 2.

III. Video Modeling

As mentioned in Section I, we decompose our task into two steps, i.e., the front-end video modeling and the back-end binary code learning. In this section, we focus on the first step by organizing it as follows: in Section III-A, we briefly introduce the classical sample covariance for video modeling; and then we extend the pixel-level covariance to patch-level in Section III-B and further to Spatial Pyramid Covariance (SPC) in Section III-C; in the end of this section, metrics of covariance are introduced in Section III-D.

A. Sample Covariance

Let $\boldsymbol{F} = [\boldsymbol{f}^1, \boldsymbol{f}^2, \cdots, \boldsymbol{f}^N]$ be the data matrix of a face track with N frames, where $\boldsymbol{f}^k \in \mathbb{R}^d$ denotes the k^{th} frame with d-dimensional representation. We represent the face track with the $d \times d$ sample covariance:

$$\boldsymbol{C} = \frac{1}{N-1}\sum_{k=1}^{N}(\boldsymbol{f}^k - \bar{\boldsymbol{f}})(\boldsymbol{f}^k - \bar{\boldsymbol{f}})^T, \tag{1}$$

where $\bar{\boldsymbol{f}} \in \mathbb{R}^d$ is the mean representation of all frames. The diagonal entries of the covariance $\boldsymbol{C}$ represent the variance of each individual dimensionality of the frame representation, and the off-diagonal entries are their respective correlations.

While it is rather simple to derive and compute, there exist several advantages to model a face track with its sample covariance: (a) as the raw second-order statistic of a set of video frames, the covariance makes no assumption about

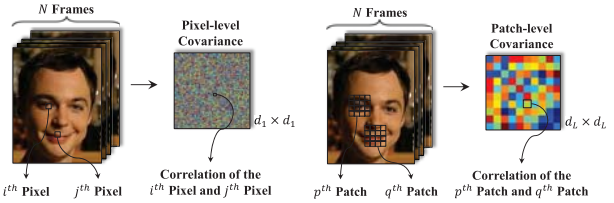


Fig. 3. Illustration of (a) pixel-level and (b) patch-level covariance.

the set data distribution, compared with the early parametric modeling methods which seek to represent each face track with some parametric distribution function, e.g., single Gaussian [41] or Gaussian Mixture Models (GMM) [42], thus providing a natural and flexible way to represent a face track with any frame number and any frame representation; (b) the covariance modeling shows stronger resistance to outliers, since it is a statistic of all frames and noise-corrupting elements are largely filtered out with an average filter during covariance computation [2]–[4].

B. Pixel-Level Covariance to Patch-Level Covariance

Let's define that the $\boldsymbol{f}^k \in \mathbb{R}^d$ in Eqn. (1) is the vectorization of frame intensities, where d indicates the total number of frame pixels. Therefore, $\boldsymbol{C}$ in Eqn. (1) is generally considered as the pixel-level covariance (denoted by $\boldsymbol{C}^{\text{pix}}$), i.e., elements in $\boldsymbol{C}^{\text{pix}}$ describe the correlations between individual pixel. In this subsection, we propose to extend the traditional pixel-level covariance to more general patch-level covariance.

To this end, we first rewrite the pixel-level covariance $\boldsymbol{C}^{\text{pix}}$ in the element-wise form:

$$\boldsymbol{C}^{\text{pix}}_{ij} = \frac{1}{N-1}\sum_{k=1}^{N}(f_i^k - \bar{f}_i)(f_j^k - \bar{f}_j), \tag{2}$$

where f_i^k ($\bar{f}_i$) is the i^{th} element of $\boldsymbol{f}^k$ ($\bar{\boldsymbol{f}}$) corresponding to the i^{th} frame pixel. Inspired by [43], we then give the definition of patch-level covariance as:

$$\boldsymbol{C}^{\text{pat}}_{pq} = \frac{1}{N-1}\sum_{k=1}^{N}(\boldsymbol{f}_p^k - \bar{\boldsymbol{f}}_p)^T(\boldsymbol{f}_q^k - \bar{\boldsymbol{f}}_q), \tag{3}$$

where $\boldsymbol{C}^{\text{pat}}_{pq}$ describes the correlation between the p^{th} and the q^{th} patches of the face track (please see Fig. 3 for better understanding). $\boldsymbol{f}_p^k$ and $\bar{\boldsymbol{f}}_p$ in Eqn. (3) are defined as follows.

$$\boldsymbol{f}_p^k = [f_{p_1}^k, f_{p_2}^k, \cdots, f_{p_n}^k]^T, \tag{4}$$

$$\bar{\boldsymbol{f}}_p = [\bar{f}_{p_1}, \bar{f}_{p_2}, \cdots, \bar{f}_{p_n}]^T, \tag{5}$$

where n is the number of pixels within the p^{th} (q^{th}) patch (of course, the p^{th} and q^{th} patches are required to have the same size). Based on the above definitions, we further rewrite Eqn. (3) as follows:

$$\begin{aligned}
\boldsymbol{C}^{\text{pat}}_{pq} &= \frac{1}{N-1}\sum_{k=1}^{N}(\boldsymbol{f}_p^k - \bar{\boldsymbol{f}}_p)^T(\boldsymbol{f}_q^k - \bar{\boldsymbol{f}}_q) \\
&= \frac{1}{N-1}\sum_{k=1}^{N}(\boldsymbol{f}_p^{kT}\boldsymbol{f}_q^k - \boldsymbol{f}_p^{kT}\bar{\boldsymbol{f}}_q - \bar{\boldsymbol{f}}_p^T\boldsymbol{f}_q^k + \bar{\boldsymbol{f}}_p^T\bar{\boldsymbol{f}}_q) \\
&= \frac{1}{N-1}\sum_{k=1}^{N}(\sum_{w=1}^{n}f_{p_w}^k f_{q_w}^k - \sum_{w=1}^{n}f_{p_w}^k \bar{f}_{q_w} - \sum_{w=1}^{n}\bar{f}_{p_w} f_{q_w}^k + \sum_{w=1}^{n}\bar{f}_{p_w}\bar{f}_{q_w}) \\
&= \frac{1}{N-1}\sum_{k=1}^{N}(\sum_{w=1}^{n}(f_{p_w}^k f_{q_w}^k - f_{p_w}^k \bar{f}_{q_w} - \bar{f}_{p_w} f_{q_w}^k + \bar{f}_{p_w}\bar{f}_{q_w})) \\
&= \sum_{w=1}^{n}(\frac{1}{N-1}\sum_{k=1}^{N}(f_{p_w}^k f_{q_w}^k - f_{p_w}^k \bar{f}_{q_w} - \bar{f}_{p_w} f_{q_w}^k + \bar{f}_{p_w}\bar{f}_{q_w})) \\
&= \sum_{w=1}^{n}(\frac{1}{N-1}\sum_{k=1}^{N}(f_{p_w}^k - \bar{f}_{p_w})(f_{q_w}^k - \bar{f}_{q_w})) \\
&= \sum_{w=1}^{n}\boldsymbol{C}^{\text{pix}}_{p_w q_w}.
\end{aligned} \tag{6}$$

Here, a quite appealing result is observed that the patch-level covariance $\boldsymbol{C}^{\text{pat}}$ can be treated as the sum-pooling form of the pixel-level covariance $\boldsymbol{C}^{\text{pix}}$. Moreover, compared with the traditional pixel-level covariance, patch-based version has several advantages as follows: (a) ***representation volume***, as illustrated in Fig. 3, by upgrading from pixel to patch, the size of covariance drops from $d \times d$ to the more concise $d' \times d'$, where d and d' respectively denote the total number of pixels and patches; (b) ***semantic interpretability***, with appropriate patch size, patch-level covariance has the potential to explicitly depict all kinds of correlations between facial components (e.g., forehead, eyes, nose, mouth), whereas pixel-wise correlations contain relatively weaker and ambiguous semantic interpretation; (c) ***mis-alignment invariance***, the sum-pooling operation strengthens the robustness against mis-alignment, which has always been regarded as an intractable noise for face images. Based on these, patch-level covariance is expected to admit a relatively wider range of application demands compared with the pixel-level version.

Theoretically, the proposed patch-level covariance has the potential to compatible with any off-the-shelf visual feature. The only requirement is a reasonable definition of patches. More specifically, the patches defined on low-level intensity have the natural semantic correspondence to spatial regions. Nevertheless, deep-based representation as a high-level feature does not explicitly have the spatial region information. As a consequence, we choose the raw intensity as entry level prototype to illustrate the proposed model, and a straightforward scheme to employ other feature is dividing the feature vector at regular intervals to form patches.

C. Spatial Pyramid Covariance

Nevertheless, the biggest practical problem encountered when using patch-level covariance is how to set an appropriate patch size, and usually it is almost impossible to assign a

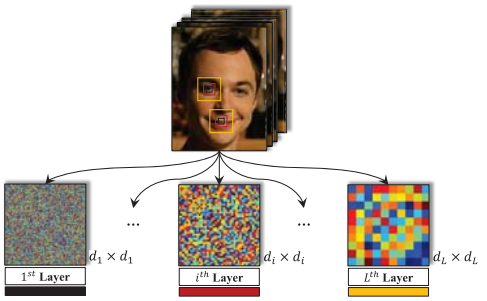


Fig. 4. Illustration of constructing SPC. The leftmost black (1^{st}) layer is the pixel-level covariance, the rightmost yellow (L^{th}) layer is the patch-level covariance with the maximal patch size, where d_1 and d_L denote the total number of pixels and patches in corresponding layer respectively. Obviously, from left to right, covariance has more concise volume and more explicit semantic interpretability. Moreover, different layers are organized in a hierarchical multi-granularity manner.

general-purpose parameter for distinct scenarios. Such being the case, we might consider whether it is possible to intelligently combine a series of different parameterized patch-level covariances by fully exploring the complementarity among them. Inspired by the classical technique, Spatial Pyramid Matching (SPM) [44], [45], we propose a novel hierarchical multi-granularity video representation, named Spatial Pyramid Covariance (SPC).

SPM is first explored by Grauman and Darrell [45] to find an approximate correspondence between two feature sets, and then extended by Lazebnik *et al.* [44] to the problem of natural scene recognition. In particular, SPM is complemented by subdividing an image and computing histograms of local features at increasingly fine resolutions. Similar idea is adopted in this work by integrating multiple layers of patch-level covariance of different patch size as:

$$\boldsymbol{C}_i^{\mathrm{SPC}} = \{\boldsymbol{C}_i^1, \boldsymbol{C}_i^2, ..., \boldsymbol{C}_i^L\}, \tag{7}$$

where $\boldsymbol{C}_i^{\mathrm{SPC}}$ denotes the final SPC representation of the i^{th} instance, $\boldsymbol{C}_i^l$ is the l^{th} layer patch-level covariance ($\boldsymbol{C}_i^1$ is the fundamental pixel-level covariance, and it can be also regarded as a special case of patch-level covariance with 1×1 patch size), and L is the total number of layers. Fig. 4 shows a toy example for better understanding. Although such spatial pyramid structure may certainly result in a higher-dimensional representation, it preserves complete video information, i.e., from weak-semantic but subtle pixel-level description to strong-semantic yet coarse patch-level description. In a nutshell, such multi-granularity structure effectively encodes all kinds of appearance variations into a unified model, and then naturally suits for complex video representation.

D. Metrics of Covariance

It is well known that the nonsingular covariances,[1] a.k.a., symmetric positive definite (SPD) matrices, do not lie in a Euclidean space but on a Riemannian manifold $\mathcal{M}$ [46]–[48]. However, it is not trivial to design classifier on the manifold since typical learning algorithms are devoted to operating in Euclidean space. To handle such problem, Log-Euclidean Distance (LED) [49] as a well-studied metric is utilized to bridge the gap between Riemannian manifold and Euclidean space:

$$d_{\mathrm{LED}}(\boldsymbol{C}_i, \boldsymbol{C}_j) = \left\| \log(\boldsymbol{C}_i) - \log(\boldsymbol{C}_j) \right\|_F, \tag{8}$$

where $\boldsymbol{C}_i$, $\boldsymbol{C}_j$ are two $d \times d$ nonsingular covariances, $\log(\cdot)$ is the ordinary matrix logarithm operator, and $\|\cdot\|_F$ denotes the matrix Frobenius norm. Let $\boldsymbol{C} = \boldsymbol{U}\sum\boldsymbol{U}^T$ be the eigen-decomposition of SPD matrix $\boldsymbol{C}$, its log is a symmetric matrix and can be easily computed by

$$\log(\boldsymbol{C}) = \boldsymbol{U}\log(\sum)\boldsymbol{U}^T, \tag{9}$$

where $\log(\sum)$ is the diagonal matrix of the eigenvalue logarithms. By doing this, a point $\boldsymbol{C}$ on the Riemannian manifold $\mathcal{M}$ is projected to a Euclidean space via the logarithm map:

$$\Psi_{\log} : \mathcal{M} \mapsto T_{\boldsymbol{I}}, \quad \boldsymbol{C} \to \log(\boldsymbol{C}). \tag{10}$$

The image $\Psi_{\log}(\mathcal{M})$ is the tangent space $T_{\boldsymbol{I}}$ of the manifold $\mathcal{M}$ at the point of the identity matrix $\boldsymbol{I}$, which is a vector space spanned by $d \times d$ symmetric matrices. The LED metric thus simply reduces to a Euclidean distance in $\mathbb{R}^{d\times d}$. By computing the inner product in the Euclidean space $T_{\boldsymbol{I}}$, Wang *et al.* [2] further derived a Riemannian kernel function on the manifold $\mathcal{M}$:

$$k_{\log}(\boldsymbol{C}_i, \boldsymbol{C}_j) = \mathrm{trace}[\log(\boldsymbol{C}_i) \cdot \log(\boldsymbol{C}_j)], \tag{11}$$

where trace[·] denotes the matrix trace. Benefited from this explicit kernel mapping, any learning method originally developed for vector spaces can be used, by taking either the Log-mapped covariances as input to its linear formulation or the derived kernel function as input to its kernel formulation.

IV. Binary Code Leaning

For each face track, we can obtain a complete SPC representation as introduced in Section III, but meanwhile such representation magnifies the feature dimension, which certainly leads to higher space and time complexity, especially conflicting with the demand of retrieval task. Moreover, the current SPC representation does not incorporate any supervision information, which will definitely favor the retrieval accuracy. To address these problems, our strategy is to discriminatively map the high-dimensional SPC representation into a much lower-dimensional Hamming space to produce a binary vector for each face track. Two advantages can be induced by doing this, the concise space demand, and the low time cost (only bit-wise XOR operation). Next we will discuss how to efficiently learn binary codes from those high-dimensional representations.

[1]For image set classification, it is often the case that the number of images (frames) is less than the feature dimension, thus leading to the singularity of the covariance. To tackle this singularity, a simple method is adding a small perturbation to the covariance [2].

A. Discriminability and Stability

First, the ***discriminability*** of binary codes in Hamming space is expected most. To this end, we further decompose the discriminability constraint into two components, i.e., intra-class (within-class) compactness and inter-class (between-class) separability. That is, instances from the same class should have similar codes, and instances from different classes should have better separability in the target Hamming space. Formally, let $\boldsymbol{B} \in \{-1, 1\}^{K\times M}$ denotes the binary codes of training instances, where K and M are the binary code length and the total number of training instances, respectively. $\boldsymbol{b}_i \in \{-1, 1\}^{K\times 1}$ denotes the binary code of the i^{th} training instance. Then the distance measures of within-class S_W and between-class S_B can be formulated as:

$$S_W = \sum_{c\in\{1:R\}} \sum_{y_i, y_j = c} \text{dis}(\boldsymbol{b}_i, \boldsymbol{b}_j), \tag{12}$$

$$S_B = \sum_{\substack{c\in\{1:R\} \\ y_i = c}} \sum_{\substack{c'\in\{1:R\} \\ c\neq c', y_j = c'}} \text{dis}(\boldsymbol{b}_i, \boldsymbol{b}_j), \tag{13}$$

where R is the total number of classes, y_i denotes the class label of the i^{th} training instance, dis$(\cdot)$ can be any available distance measurement in Hamming space. Thus, to implement a strong discrimination, we minimize the following energy function E_{disc} as:

$$E_{disc} = S_W - \lambda S_B. \tag{14}$$

Second, ***stability*** (sometimes also called as generality, learnability, or predictability) is another crucial constraint in binary code learning. Intuitively, it is the concern about similarity preserving, i.e., visually similar instances should be mapped to similar binary codes within a short Hamming distance. In some sense, the above discriminability constraint, i.e., E_{disc}, only minimizes the empirical risk on the training instances, and here we add the stability constraint to achieve structural risk minimization. Fig. 5 illustrates the relationship between discriminability and stability. Let's imagine each bit of binary code as a split (hyperplane) in the original feature space that separates training instances into two half-spaces that have binary code value -1 v.s. the ones that have value 1, and we want the most stable splits. Specifically, a split is stable when it has large margins from instances around it. Think about such a disappointing situation where a split crosses an area with dense instances, many actually neighboring instances will be inevitably assigned different binary code values.

Technically, we resort to the classical Support Vector Machine (SVM) [50] with its inherent max-marginal property to implement the binary code learning, i.e., hash functions learning. In addition, to maximally exploit the complementarity among the patch-level covariances in SPC, we dexterously embed the combination coefficients learning into the stability constraint. To that end, Multiple Kernel Learning (MKL) [51] is utilized along with the sophisticated Riemannian kernel in Eqn. (11). In particular, we build K splits (each corresponding to one bit of the binary code) by training K kernel SVMs individually. More concretely, we denote the k^{th} split by

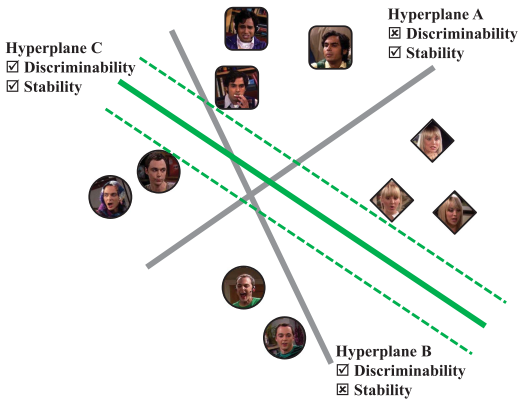


Fig. 5. Illustration of the two constraints when learning 1-bit binary code, i.e., ***discriminability*** and ***stability***, where the two gray splits only satisfy either of the constraints, and the green split is the one we need.

$\boldsymbol{\omega}^k$ $(k = 1, \cdots, K)$, and the energy function can be formulated as follow.

$$\begin{aligned} E_{stab} = & \frac{1}{2} \sum_{k\in\{1:K\}} \left\| \boldsymbol{\omega}^k \right\|^2 + \delta \sum_{\substack{k\in\{1:K\} \\ i\in\{1:M\}}} \max(1 - \boldsymbol{b}_i^k (\boldsymbol{\omega}^k{}^T \varphi^k(\boldsymbol{C}_i^{\text{SPC}})), 0), \\ s.t. \quad & <\varphi^k(\boldsymbol{C}_i^{\text{SPC}}), \varphi^k(\boldsymbol{C}_j^{\text{SPC}})> = \sum_{l\in\{1:L\}} \boldsymbol{\beta}_l^k k_{\log}(\boldsymbol{C}_i^l, \boldsymbol{C}_j^l), \\ & \forall k \in \{1{:}K\}, \\ & \sum_{l\in\{1:L\}} \boldsymbol{\beta}_l^k = 1, \forall k \in \{1{:}K\}, \\ & \boldsymbol{\beta}^k = [\boldsymbol{\beta}_1^k, \boldsymbol{\beta}_2^k, ..., \boldsymbol{\beta}_L^k]^T \geq 0, \forall k \in \{1{:}K\}, \end{aligned} \tag{15}$$

where $\varphi^k(\cdot)$ denotes the k^{th} mapping function to map $\boldsymbol{C}_i^{\text{SPC}}$ to a Reproducing Kernel Hilbert Space (RKHS) [52] (by means of the kernel trick theory), $\boldsymbol{b}_i^k \in \{-1, 1\}$ indicates in which side of the k^{th} split the i^{th} instance lies, and $\boldsymbol{\beta}^k$ is the combination coefficient of the L kernel matrices (corresponding to the L layers of patch-level covariance in SPC) for the k^{th} bit.

After the above analysis, we can reach the final objective function by combining Eqn. (14) and Eqn. (15) to simultaneously consider the discriminability and stability of the target binary code:

$$\min_{\boldsymbol{\omega}, \boldsymbol{\beta}, \boldsymbol{B}} \quad E_{disc} + \alpha E_{stab}. \tag{16}$$

B. Optimization Algorithm

Since the objective function in Eqn. (16) is non-convex, it is infeasible to find a global analytical solution. In practice, we utilize block coordinate descent method [53] to independently optimize each individual item for iteratively updating $\boldsymbol{\omega}$, $\boldsymbol{\beta}$, and $\boldsymbol{B}$. The pseudo-code of optimization can be found in Algorithm 1. Next, we give a detailed discussion. Assume that we have M training instances of R classes, and for each instance we have the computed SPC representation $\boldsymbol{C}_i^{\text{SPC}}$ with its class label $y_i \in \{1, 2, ..., R\}$.

Initialization: First, computing the L kernel matrices, i.e., $\boldsymbol{K}_l \in \mathbb{R}^{M\times M}, l \in \{1, 2, ..., L\}$, with the Riemannian

Algorithm 1 Optimization

Input: Training instances $\boldsymbol{C}_i^{\text{SPC}}$ and corresponding class labels $y_i \in \{1, 2, ..., R\}$, where $i \in \{1, 2, ..., M\}$.
Output: $\boldsymbol{B} \in \{-1, 1\}^{K \times M}$.
Initialization:
1. Compute L kernel matrices $\boldsymbol{K}_l \in \mathbb{R}^{M \times M}, l \in \{1, 2, ..., L\}$ with Eqn. (11);
2. $\boldsymbol{\beta}^k = [\frac{1}{L}, \frac{1}{L}, ..., \frac{1}{L}]^T, k \in \{1, 2, ..., K\}$;
3. $\boldsymbol{K}^k = \sum_{l=1}^{L} \boldsymbol{\beta}_l^k \boldsymbol{K}_l, k \in \{1, 2, ..., K\}$;
4. Randomly initialize $\boldsymbol{B} \in \{-1, 1\}^{K \times M}$;
Repeat several times
5. Fix $\boldsymbol{B}$ to optimize $\boldsymbol{\omega}$ (i.e., $\boldsymbol{U}$) and $\boldsymbol{\beta}$ with Eqn. (15);
6. Fix $\boldsymbol{\omega}$ (i.e., $\boldsymbol{U}$) and $\boldsymbol{\beta}$ to optimize $\boldsymbol{B}$ with Eqn. (14);
End
7. $\boldsymbol{K}^k = \sum_{l=1}^{L} \boldsymbol{\beta}_l^k \boldsymbol{K}_l, k \in \{1, 2, ..., K\}$;
8. $\boldsymbol{b}^k = sgn(\boldsymbol{U}^{k^T} \boldsymbol{K}^k), k \in \{1, 2, ..., K\}$;
9. $\boldsymbol{B} = [\boldsymbol{b}^1, \boldsymbol{b}^2, ..., \boldsymbol{b}^K]^T$.

kernel defined in Eqn. (11); ***second***, for each bit, initializing the combination coefficient of the L kernel matrices, i.e., $\boldsymbol{\beta}^k$, as $[\frac{1}{L}, \frac{1}{L}, ..., \frac{1}{L}]^T$; ***third***, for each bit, computing the corresponding integrated kernel matrix $\boldsymbol{K}^k \in \mathbb{R}^{M \times M}$ by $\boldsymbol{K}^k = \sum_{l=1}^{L} \boldsymbol{\beta}_l^k \boldsymbol{K}_l$; ***lastly***, randomly initializing the binary codes $\boldsymbol{B}$.

Fix B to Optimize ω and β: for each bit, we use $\boldsymbol{b}^k \in \{-1, 1\}^{1 \times M}$ as training label to train a kernel SVM with $\boldsymbol{K}^k$ as training data. Since we also need to learn the kernel combination coefficient $\boldsymbol{\beta}^k$, an off-the-shelf MKL method, i.e., SimpleMKL [51], is adopted to simultaneously optimize $\boldsymbol{\omega}^k$ and $\boldsymbol{\beta}^k$ for each bit. Moreover, as kernel trick is applied to handle the non-linear mapping, in practice each $\boldsymbol{\omega}^k$ is learned in an equivalent form of projection $\boldsymbol{U}^k \in \mathbb{R}^{M \times 1}$ according to the Riesz representation theorem [54], [55], which further forms the integrated projection matrix $\boldsymbol{U} = [\boldsymbol{U}^1, \boldsymbol{U}^2, ..., \boldsymbol{U}^K] \in \mathbb{R}^{M \times K}$ corresponding to $\boldsymbol{\omega}$.

Fix ω and β to Optimize B: having the learned $\boldsymbol{\omega}$ (i.e., $\boldsymbol{U}$) and $\boldsymbol{\beta}$, we then use them to predict $\boldsymbol{B}$ and further optimize it. For each bit, we first compute $\boldsymbol{K}^k = \sum_{l=1}^{L} \boldsymbol{\beta}_l^k \boldsymbol{K}_l$, and then use the learned split $\boldsymbol{\omega}^k$ (i.e., $\boldsymbol{U}^k$) to predict $\boldsymbol{b}^k$ by quantizing the kernel SVM's outputs as $\boldsymbol{b}^k = sgn(\boldsymbol{U}^{k^T} \boldsymbol{K}^k)$. After this, we feed the predicted $\boldsymbol{B} = [\boldsymbol{b}^1, \boldsymbol{b}^2, ..., \boldsymbol{b}^K]^T$ into a subgradient descend based binary code optimization method proposed in [31] to update it. It is during this step that the discriminability of binary codes is guaranteed.

Convergence Criteria: the whole optimization is looped by iteratively update $\boldsymbol{\omega}$, $\boldsymbol{\beta}$, and $\boldsymbol{B}$, and in practice we find that usually two or three times iterations can make the objective function converge. For better understanding, we design a toy example to illustrate the optimization process of one bit in Fig. 6.

V. Experiments

In this section, we evaluate the proposed method in the face video retrieval task on two TV-series video databases. Besides, as a general video matching algorithm, we also evaluate the proposed method in the traditional video face recognition task.

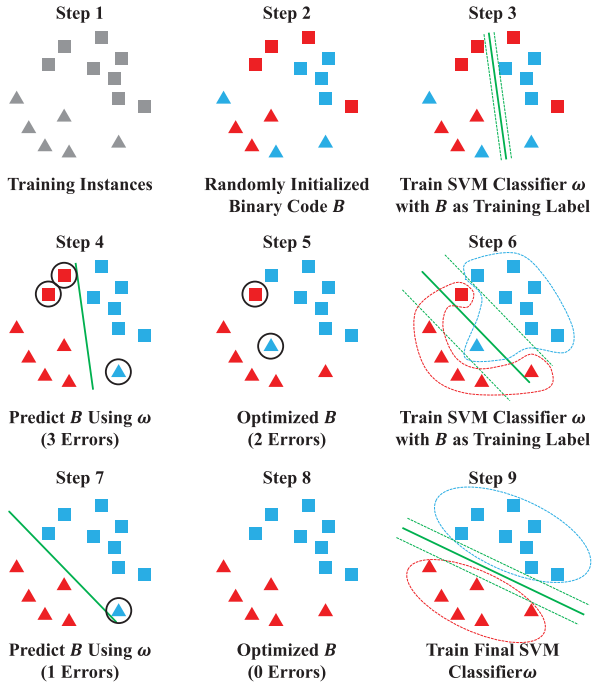


Fig. 6. Illustration of the iterative optimization process of a one-bit two-class case. The square and triangular shapes represent two classes, and the blue and red colors represent two code values (i.e., 1 and -1) respectively. The green solid line indicates the learned split.

A. Databases Description and Settings

We construct two large scale video databases[2] from two hit American shows, i.e., *the Big Bang Theory* (BBT), and *Prison Break* (PB). These two TV-series are quite different in their filming styles: BBT is a sitcom about 20 minutes per episode with a main cast of 5 characters and mostly takes place indoors; on the other hand, PB has an average length of about 42 minutes per episode, where many shots are set outside, resulting in a large range of different illumination. To extract face tracks, several techniques, e.g., shot boundary detection, face detection, tracking, and facial landmark localization are involved. To guarantee the purity of databases, we also invite 5 fans of each TV-series to annotate every extracted face track. Specifically, we deal with the whole first season of both TV-series, i.e., 17 episodes for BBT, and 22 episodes for PB, and finally we collect 4,667 and 9,435 face tracks from BBT and PB, respectively. Fig. 7 shows some exemplar face tracks of the two constructed TV-series video databases, and the face track distributions are listed in Table I. Considering the space complexity of covariance matrix, faces in both TV-series are cropped from the central region of original square images (Fig. 7), and normalized to 20×16 gray images with left eye center located at (4, 8) and right eye center located at (11, 8) respectively. For simplicity, histogram equalization is the only pre-processing operation employed to eliminate lighting effects.

For retrieval performance evaluation, we randomly partition each TV-series into three sets, i.e., training set, query set, and database set. More specifically, for each TV-series,

[2]The databases and Matlab code can be downloaded at http://vipl.ict.ac.cn/resources.

(a) (b)

Fig. 7. Some exemplar face tracks of the two constructed TV-series databases, i.e., (a) *the Big Bang Theory*, characters from top to bottom are *Sheldon Cooper*, *Howard Wolowitz*, *Penny*, *Leonard Hofstadter*, and *Raj Koothrappali*, and (b) *Prison Break*, characters from top to bottom are *Michael Scofield*, *Lincoln Burrows*, *Paul Kellerman*, *Brad Bellick*, and *Theodore Bagwell*.

TABLE I

DISTRIBUTIONS OF FACE TRACKS IN *the Big Bang Theory* (BBT) AND *Prison Break* (PB). SPECIFICALLY, THERE ARE 4,667 FACE TRACKS OF 15 CHARACTERS AND 9,435 FACE TRACKS OF 20 CHARACTERS IN BBT AND PB, RESPECTIVELY, WHERE EXTRAS (USUALLY APPEAR IN THE BACKGROUND) ARE LABELED AS "*Unknown*"

Idx.	*the Big Bang Theory*		*Prison Break*	
	Character Name	Num.	Character Name	Num.
1	*Althea*	14	*Benjamin Franklin*	294
2	*Dmitri*	19	*Brad Bellick*	309
3	*Eric Gablehauser*	31	*Caroline Reynolds*	126
4	*Howard Wolowitz*	413	*Charles Patoshik*	88
5	*Kurt*	11	*Charles Westmoreland*	223
6	*Lalita Gupta*	30	*Daniel Hale*	93
7	*Leonard Hofstadter*	824	*David Apolskis*	174
8	*Leslie Winkle*	72	*Fernando Sucre*	461
9	*Mary Cooper*	58	*Henry Pope*	350
10	*Missy Cooper*	65	*John Abruzzi*	414
11	*Penny*	917	*Lincoln Burrows*	740
12	*Raj Koothrappali*	453	*LJ Burrows*	145
13	*Sheldon Cooper*	1,528	*Maricruz Delgado*	49
14	*Toby Loobenfeld*	23	*Michael Scofield*	1,965
15	*Unknown*	209	*Nick Savrinn*	245
16			*Paul Kellerman*	225
17			*Sara Tancredi*	490
18			*Theodore Bagwell*	472
19			*Veronica Donovan*	637
20			*Unknown*	1,935
	All Characters	4,667	All Characters	9,435

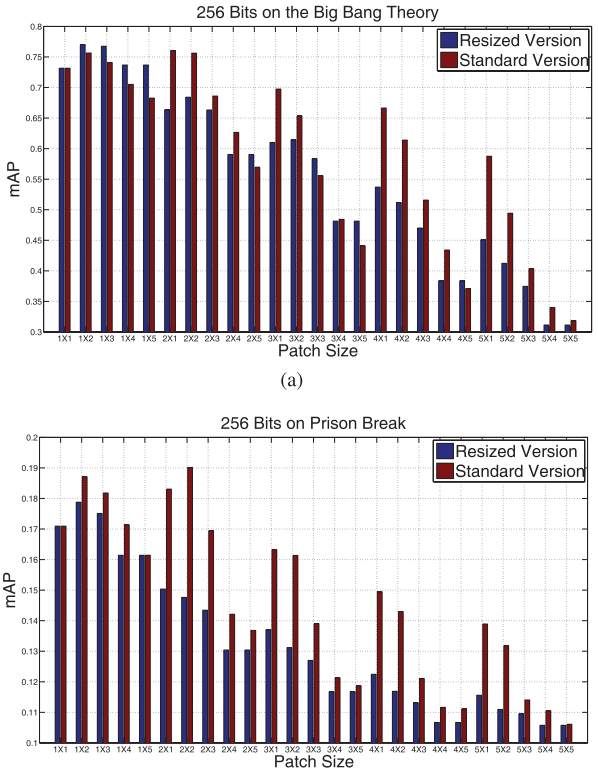


Fig. 8. Comparison of two computation schemes of patch-level covariance on (a) *the Big Bang Theory* and (b) *Prison Break*. Standard Version: first computing the pixel-level covariance with the original-size face track, and then sum-pooling it as proposed in Section III-B; Resized Version: directly computing the pixel-level covariance on the resized face track.

we randomly select 10 face tracks per character as training set (not including the *Unknown* ones, because it's impossible to label the extra identities), and leave the rest as test data. Then we further select 10 face tracks per character randomly from test data to form query set, the rest of test data is regarded as database set (for the characters whose images less than 20, e.g., *Althea*, *Dmitri*, *Kurt*, we will not choose images from them to the query set). To sum up, three sets have no overlap images, and training set is used for model training, and query set acts as key words for retrieving from database set. For quantitative evaluation, we use the standard mean Average Precision (mAP) [56] and the precision recall curve [57] as measurements.

B. Evaluation on Patch-Level Covariance

In this subsection, we evaluate different patch-level covariances. As defined in Section III-B, a patch-level covariance can be treated as a sum-pooling form (or down-sampling form) of the fundamental pixel-level covariance, and this raises a natural baseline comparison. Which of the following two patch-level covariance computation schemes makes more sense: (a) first computing the pixel-level covariance with

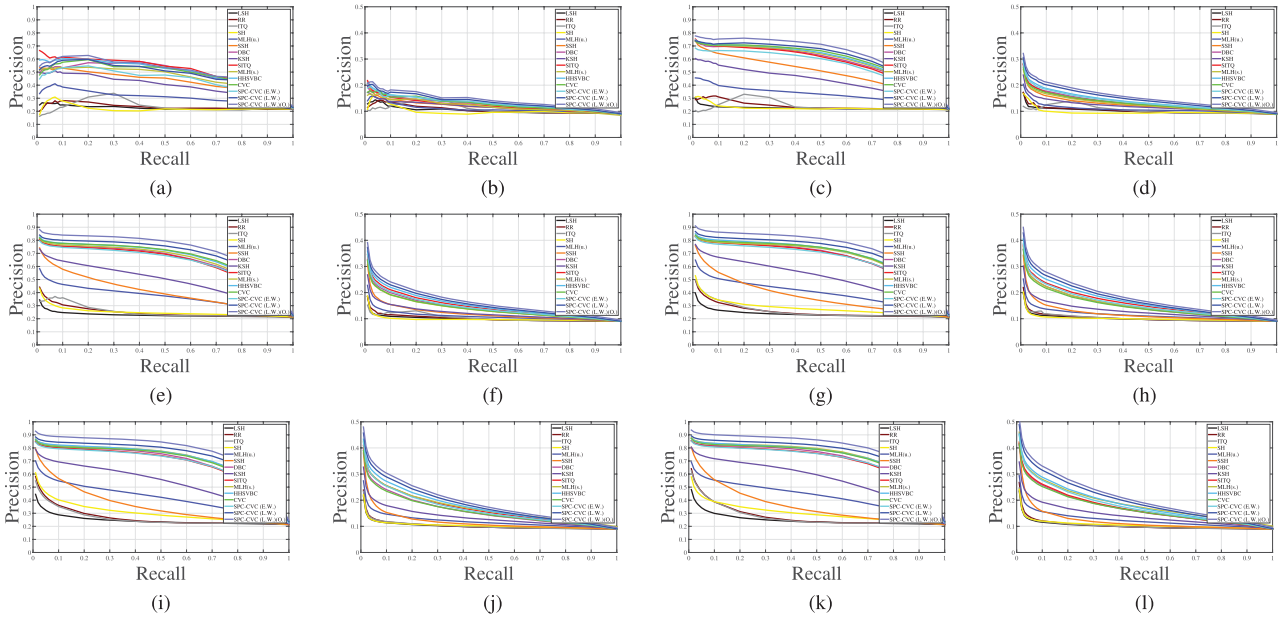


Fig. 9. Comparison with state-of-the-art binary code learning methods in precision recall curve on *the Big Bang Theory* and *Prison Break*, where E.W. and L.W. are the abbreviations of equal weight and learned weight respectively, and O. is the initial of overlap. (a) K=8, the Big Bang Theory. (b) K=8, Prison Break. (c) K=16, the Big Bang Theory. (d) K=16, Prison Break. (e) K=32, the Big Bang Theory. (f) K=32, Prison Break. (g) K=64, the Big Bang Theory. (h) K=64, Prison Break. (i) K=128, the Big Bang Theory. (j) K=128, Prison Break. (k) K=256, the Big Bang Theory. (k) K=256, the Big Bang Theory.

the original-size face track, and then down-sampling it as the proposed method; (b) directly computing the pixel-level covariance on the resized face track[3] (e.g., computing the patch-level covariance with 2×2 patch size, it seems, can be implemented by computing the pixel-level covariance on the resized face track which has 50% height and width of the original size). Intuitively, the above two schemes seems equivalent to each other.

For this and the following experiments, we fix the number of patch types as 25, ranging from 1×1 (i.e., pixel-level covariance), 1×2 to 5×5. Moveover, different-sized patch results in different-sized covariance, e.g., 1×1 and 2×2 patch sizes respectively lead to 320×320 and 80×80 patch-level covariances with the provided face size (i.e., 20×16). Fig. 8 shows the comparison between the two schemes in mAP. It is quite obviously to find that the proposed scheme (standard version) performs better than the baseline scheme (resized version) in most cases, especially in the ones with large patch size, and that is mainly because the resizing of original face track inevitably leads to information loss at the starting line, thus can only providing a limited basis for the subsequent covariance computation. In contrast, the proposed scheme first computes an over-complete pixel-level covariance with the original-size face track, and the sum-pooling operation is placed in the second step, which is supposed to make more sense. In addition, an interesting phenomenon is found that for patches with small sizes, the comparison results are opposite on the two TV-series (i.e., for *the Big Bang Theory*, resized version is better, and for *Prison Break*, standard version wins). Such phenomenon is mainly caused by the different face image conditions (resolutions) of the two TV-series. As two TV-series with different filming style, *the Big Bang Theory* has plenty of close-up shots which lead to relatively high face resolution, while *Prison Break* consists of lots of night and outdoors scenes which bring relatively bad face quality. Thus, *Prison Break* suffers more than *the Big Bang Theory* about information loss caused by image resizing. As a consequence, the proposed method in general exhibits its competitiveness on low quality faces.

[3]In this work, we used Matlab function *imresize*() for face track resizing. Specifically, we have tried *bilinear* and *bicubic* interpolation strategies, and for the current face size, i.e., 20×16, there was no obvious performance difference between them (we adopted the *bicubic* strategy in this work). That is mainly because of the relatively low resolution, and we do believe as the resolution increases, the advantage of *bicubic* will become increasingly evident.

C. Evaluation on Spatial Pyramid Covariance

In this subsection, we evaluate the effectiveness of SPC, i.e., integrating the above 25 patch-level covariances. Table II and Fig. 9 respectively exhibit this evaluation on two databases in mAP and precision recall curve, where CVC [12] denotes the implementation of single pixel-level covariance, and SPC-CVC denotes the implementation of SPC (E.W. and L.W. are the abbreviations of equal weight and learned weight, respectively indicating two integrating strategies for the 25 different parameterized patch-level covariances in SPC). Please observe these results along with Fig. 8. Quite consistent with our intuition, by fully exploring the inherent complementary among different patch-level covariances, SPC-CVC (L.W.) is shown to be more competitive than any single patch-level covariance, and straightforwardly treating different patch-level covariance with equal concern as SPC-CVC (E.W.) only results in limited performance (even lower than the fundamental CVC). For better understanding, we also visualize the combination coefficient of the 25 covariances in Fig. 10,

TABLE II

COMPARISON WITH STATE-OF-THE-ART BINARY CODE LEARNING METHODS IN mAP ON *the Big Bang Theory* AND *Prison Break*, WHERE U., S.U., AND S. INDICATE UNSUPERVISED, SEMI-SUPERVISED, AND SUPERVISED METHOD RESPECTIVELY, E.W. AND L.W. ARE THE ABBREVIATIONS OF EQUAL WEIGHT AND LEARNED WEIGHT RESPECTIVELY, AND O. IS THE INITIAL OF OVERLAP

Method	*the Big Bang Theory*						*Prison Break*					
	8 bits	16 bits	32 bits	64 bits	128 bits	256 bits	8 bits	16 bits	32 bits	64 bits	128 bits	256 bits
LSH (u.) [24]	0.2265	0.2259	0.2310	0.2385	0.2474	0.2509	0.1034	0.1023	0.1021	0.1030	0.1037	0.1047
RR (u.) [29]	0.2389	0.2418	0.2519	0.2559	0.2675	0.2753	0.1062	0.1031	0.1032	0.1047	0.1054	0.1067
ITQ (u.) [29]	0.2403	0.2411	0.2526	0.2579	0.2619	0.2721	0.1090	0.1075	0.1049	0.1045	0.1051	0.1055
SH (u.) [28]	0.2220	0.2267	0.2516	0.2829	0.3061	0.3043	0.1001	0.0972	0.1002	0.1033	0.1066	0.1093
MLH (u.) [36]	0.2768	0.2914	0.3096	0.3120	0.3342	0.3477	0.1183	0.1224	0.1260	0.1279	0.1300	0.1335
SSH (s.u.) [30]	0.4305	0.4974	0.4085	0.3724	0.3618	0.3572	0.1194	0.1268	0.1231	0.1212	0.1206	0.1229
DBC (s.) [31]	0.4939	0.5843	0.6430	0.6619	0.6932	0.7127	0.1275	0.1336	0.1434	0.1530	0.1612	0.1713
KSH (s.) [32]	0.3963	0.4326	0.4897	0.5120	0.5340	0.5626	0.1176	0.1210	0.1242	0.1317	0.1373	0.1443
SITQ (s.) [29]	0.5195	0.5766	0.6360	0.6632	0.6869	0.7050	0.1241	0.1321	0.1496	0.1626	0.1713	0.1732
MLH (s.) [36]	0.4727	0.5810	0.6557	0.6813	0.7154	0.7392	0.1198	0.1286	0.1406	0.1553	0.1759	0.1891
HHSVBC (s.) [38]	0.5099	0.5934	0.6718	0.6821	0.7170	0.7401	0.1388	0.1445	0.1560	0.1629	0.1784	0.1982
CVC [12]	0.4881	0.6050	0.6698	0.6844	0.7083	0.7316	0.1281	0.1326	0.1415	0.1508	0.1611	0.1709
SPC-CVC (E.W.)	0.4461	0.5536	0.6317	0.6545	0.6846	0.7049	0.1314	0.1386	0.1468	0.1586	0.1687	0.1805
SPC-CVC (L.W.)	0.4926	0.6183	0.6993	0.7169	0.7486	0.7683	0.1389	0.1494	0.1596	0.1752	0.1863	0.2015
SPC-CVC (L.W.)(O.)	**0.5202**	**0.6471**	**0.7325**	**0.7543**	**0.7740**	**0.7899**	**0.1401**	**0.1525**	**0.1674**	**0.1903**	**0.2099**	**0.2287**

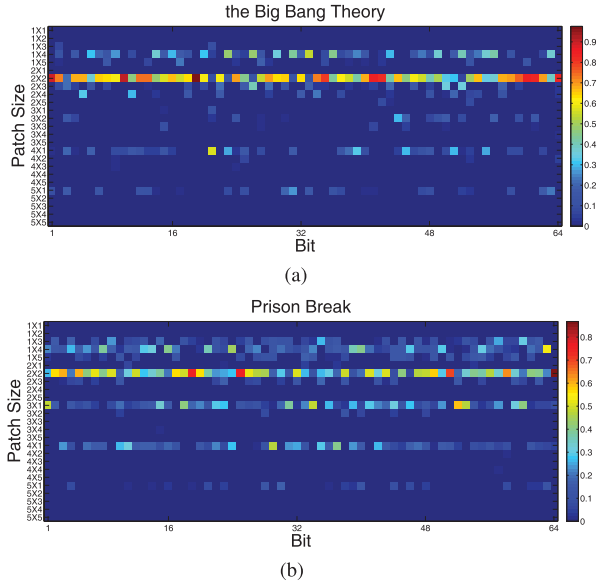


Fig. 10. Visualization of the learned combination coefficients for the 25 patch-level covariances on the two databases, i.e., (a) *the Big Bang Theory* and (b) *Prison Break*. In this visualization, larger weight is shown in warmer color, and for space limitation only 64 bits are shown here.

and for space limitation only 64 bits are shown here. As mentioned in Section III-B, patch-level covariances can be treated as derivatives of the ancestral pixel-level covariance, but with more concise representation volume, more explicit semantic interpretability, and more robust mis-alignment invariance. When we attempt to combine pixel-level covariance and different patch-level covariances, the theory of natural selection successfully selects representative patch-level covariances to reconstruct and then substitute the over-complete pixel-level covariance (almost zero weight).

D. Evaluation on Binary Code Learning

After the evaluation of front-end video modeling, in this subsection, we mainly evaluate the binary code learning part. To this end, we select several state-of-the-art hashing methods as comparison, i.e., the unsupervised Locality-Sensitive Hashing (LSH) [24], Random Rotation (RR) [29], ITerative Quantization (ITQ) [29], Spectral Hashing (SH) [28], Minimal Loss Hashing (MLH) [36], the Semi-Supervised Hashing (SSH) [30], and the supervised Discriminative Binary Code (DBC) [31], Kernel-based Supervised Hashing (KSH) [32], Supervised ITerative Quantization (SITQ) [29], supervised Minimal Loss Hashing (MLH) [36], and Hierarchical Hybrid Statistic based Video Binary Code (HHSVBC) [38]. It's important to note that MLH is designed to be compatible with both unsupervised and supervised training. Technically, training data for MLH are organized in the form of pairs along with binary similarity labels, i.e., 1 for similar pairs and 0 for dissimilar pairs, and such design naturally enables the extensibility from unsupervised to fully supervised learning. That is to say, if class labels are not available for training, binary similarity labels can be obtained by thresholding pairwise distances based on some specific metric (e.g., Euclidean distance), while binary similarity labels could be easily generated with given class labels (i.e., 1 for pairs in which elements come from the same class, and 0 for otherwise).

For fair comparison, we fix the front-end video model as 320×320 pixel-level covariance. However, most of the competitive methods are designed on Euclidean space and do not have the kernel version. Hence, here we use the Log-Euclidean Distance (LED) as [49] to map the Riemannian covariance to Euclidean space, in which all the competitive methods can handle. Table II and Fig. 9 respectively show the comparison results on the two databases in mAP and precision recall curve. It is obvious to find that supervised methods generally achieve higher retrieval accuracy than those unsupervised and semi-supervised methods, which mainly attribute to the full use of supervised information. Besides, among all the unsupervised hashing methods, MLH performs best with its hinge-like loss function and the latent structural SVM framework, and such characteristics indicate us that the stability or generality (constrained by max-margin) is of great importance for hash function learning. Please also note that,

for all the listed methods, we carefully tuned the parameters through cross validation according to the suggestions of the original literatures and released codes.

Compared with those state-of-the-art supervised hashing methods, the proposed method achieves comparable or even better performance. A possible interpretation is that the proposed method also incorporates the stability while considering the discriminability, which makes the learned codes better generality on unseen data. Again, MLH exhibits its effectiveness among all the supervised hashing methods (i.e., DBC, KSH, SITQ, MLH, CVC), especially with large code lengths (128 and 256 bits). Compared with MLH (optimal parameters: ρ is set to the value that guarantees at least 50% recall on validation set, λ is set to 0.7, and 2,000 training pairs with 1,000 epochs are employed), CVC achieves better performance with small code lengths (8, 16, 32 bits), yet loses with larger code lengths. This phenomenon can be mainly explained by the different binary codes initialization strategies of the two methods, that is, MLH initializes binary codes with LSH which makes each bit equal significance, while CVC utilizes signified PCA projections for code initialization which endows each bit different role at the starting line.

For further boosting the retrieval performance of CVC, we could upgrade the current non-overlapped patch partitioning to overlapped version. Specifically, we straightforward set the step size of patch sliding (horizontal/vertical) to half of the patch size (width/height). It can be seen that the overlapping strategy, i.e., SPC-CVC (L.W.)(O.), indeed make the performance higher, even though accordingly increase the computational complexity. Apart from above, it's not difficult to find a huge performance difference between the two databases. The reasons lie in: (a) different filming style, i.e., *the Big Bang Theory* is an indoors sitcom with plenty of close-up shots, whereas *Prison Break* is a crime theme show with lots of night and outdoors scenes, which leads to distinct face qualities; (b) different amount of extras, i.e., *the Big Bang Theory* as a sitcom has limited number of extras (less than 20), whereas *Prison Break* has much more (roughly more than 200). Among all the comparative methods, HHSVBC is the most closest one with the proposed method. It is designed by first utilizing different parameterized fisher vectors as frame representation that can encode multi-granularity low-level variation information within the frame, and then modeling the video by its frame covariance matrix to capture high-level variation information among video frames. Thus, it is supposed to be unfair to compare HHSVBC (high-level fisher vector feature) with the proposed method (raw-level intensity feature). However, SPC-CVC still exhibits its competitiveness, especially under the overlap patch setting. A reasonable interpretation is that fisher vector requires relatively high-quality/resolution image to capture image local details. In addition, we also exhibit two real retrieval cases on *the Big Bang Theory* and *Prison Break* in Fig. 13.

E. Evaluation on Method Generality

For better evaluating the generality of the proposed method, in this subsection we conduct *cross-training* experiments on the two TV-series, i.e., testing on *the Big Bang Theory* with *Prison Break* as training data, and vice versa. To be fair, we only take supervised hashing methods as comparative methods, including Discriminative Binary Code (DBC) [31], Kernel-based Supervised Hashing (KSH) [32], Supervised ITerative Quantization (SITQ) [29], supervised Minimal Loss Hashing (MLH) [36], and Hierarchical Hybrid Statistic based Video Binary Code (HHSVBC) [38]. Experimental results can be found in Table III, where 'Self' denotes training and testing on the same TV-series, and 'Cross' represents *cross-training*, e.g., the sub-column 'Cross' under column '*the Big Bang Theory*' means testing on *the Big Bang Theory* with the training data from *Prison Break*. From the table, we can notice that: (a) most of the list methods get worse with the *cross-training* strategy, as it is obvious that open-set compared with close-set makes the learning problem much more challenging; (b) methods based on max-margin principle, e.g., DBC, MLH, HHSVBC, perform much stable compared with the others without max-margin idea, i.e., KSH and SITQ; (c) the proposed method performs best, and even slight improvement can be detected under the setting testing *the Big Bang Theory* with *Prison Break*'s training data. Such improvement is mainly caused by the increasement of training subjects, i.e., *the Big Bang Theory* has only 14 characters for training, while *Prison Break* has 19 characters.

TABLE III

EVALUATION ON METHOD GENERALITY WITH CROSS-TRAINING STRATEGY IN mAP ON *the Big Bang Theory* AND *Prison Break* WITH FIXED 256 BITS, WHERE 'SELF' DENOTES TRAINING AND TESTING ON THE SAME TV-SERIES, AND 'CROSS' REPRESENTS CROSS TRAINING, e.g., THE SUB-COLUMN 'CROSS' UNDER COLUMN '*the Big Bang Theory*' MEANS TESTING ON *the Big Bang Theory* WITH THE TRAINING DATA FROM *Prison Break*

Method	*the Big Bang Theory*		*Prison Break*	
	Self	Cross	Self	Cross
DBC [31]	0.7127	0.6872	0.1713	0.1489
KSH [32]	0.5626	0.4310	0.1443	0.0933
SITQ [29]	0.7050	0.5323	0.1732	0.1220
MLH [36]	0.7392	0.7388	0.1891	0.1779
HHSVBC [38]	0.7401	0.6994	0.1982	0.1768
CVC [12]	0.7316	0.7299	0.1709	0.1563
SPC-CVC (L.W.)	0.7683	0.7707	0.2015	0.1895

F. Evaluation on Parameter Sensitivity

In this subsection, we focus on the parameter sensitivity. Generally speaking, although several parameters appear in Section IV, the proposed method is parameter insensitive. Specifically, the objective function Eqn. (16) is just used as a conceptual formulation to depict the proposed two constraints. As it is infeasible to find a global analytical solution, we had to optimize each component separately in an iterative manner as in Algorithm 1. That is, the parameter α mainly plays the role of balancing each component, and was simply set to 1 indicating equal importance. Besides, another balance parameter λ in Eqn. (14) is used to manage the trade-off between S_W and S_B, and in practice we set it to the pre-computed value according to training data to equally handle each within-class and between-class pair. Besides, the only

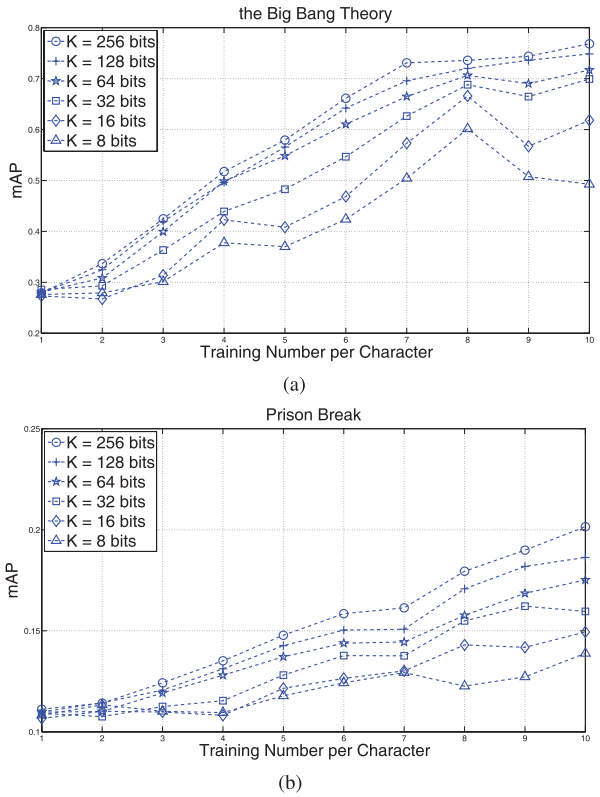


Fig. 11. Retrieval performance against training number per character on the two databases, i.e., (a) *the Big Bang Theory* and (b) *Prison Break*. Without loss of generality, we take all the possible code lengths into consideration.

substantial parameter is the soft margin parameter δ, which was simply set to 1 as standard SVM.

Next, we analyse the impact of a practical parameter on retrieval performance, i.e., the training number of each character. To this aim, we vary the instance number from 1 to 10 (we set the upper bound to 10, because some characters have only dozen of instances), and a ten-folder cross validation is adopted to make the curves more gentle. Fig. 11 shows the retrieval performance against training number per character on the two databases. It is quite clear that the proposed method performs better with more training instances per character, as rich data offers the possibility of fully exploration of code discriminability. Besides, we can also notice that some of the curves reveal their decreasing tendency with a large number of training samples. The only reason for this is over fitting on the training data. Therefore, the optimal parameter should perfectly balance the method discriminability and generality.

G. Evaluation on Computational Complexity

Since the proposed method can be decomposed into two parts, i.e., the front-end video modeling and the back-end binary code learning, here we list the computational complexity separately for each part. The time-consuming is measured in seconds on a PC with Intel Core i7 processor of 3.40GHz.

Generally speaking, the proposed method is computationally efficient. More specifically, the computation of front-end video modeling takes 0.56s in average for each face track (roughly 50 frames), including 0.20s for the fundamental pixel-level covariance and 0.36s for the patch-level covariances (which can be efficiently derived from the just computed pixel-level covariance). With the pre-computed kernel matrices, the back-end binary code learning takes 0.35s in average of each bit with standard LIBSVM [58].

Fig. 12. Some exemplar video frames of the *YouTube Celebrities* database, and celebrities from top to bottom are *Adam Sandler*, *Bruce Willis*, *Al Pacino*, *Bill Clinton*, and *Angelina Jolie*.

TABLE IV

COMPARISON WITH PREVALENT IMAGE SET CLASSIFICATION METHODS ON *YouTube Celebrities*

Method	Representation Size	Mean Accuracy
MSM [16]	hundreds of floating points	0.611
DCC [17]	hundreds of floating points	0.648
MMD [18]	hundreds of floating points	0.629
MDA [19]	hundreds of floating points	0.653
AHISD [20]	hundreds of floating points	0.637
CHISD [20]	hundreds of floating points	0.663
SANP [21]	hundreds of floating points	0.684
CDL [2]	hundreds of floating points	0.701
CVC	128 bits	0.706
SPC-CVC (E.W.)	128 bits	0.697
SPC-CVC (L.W.)	128 bits	0.716
SPC-CVC (L.W.)(O.)	128 bits	**0.719**

H. Evaluation on Traditional Video Face Identification

Although the proposed method is designed for face video retrieval, it is also a general video matching algorithm, but with more concise features. In this part, we carry out an additional evaluation on traditional video face recognition task, i.e., video face identification, on *YouTube Celebrities*.

YouTube Celebrities (YTC) is a widely studied and challenging benchmark [5] and contains of 1,910 face tracks involving 47 celebrities collected from *YouTube*. Each face track contains hundreds of frames, which are mostly highly compressed, with low resolution and covering large intra-class variations. Fig. 12 shows some exemplar video frames of the *YouTube Celebrities* database. We use the data features and 10-fold cross validation splits exactly the same as [2]. Specifically, each face is resized into 20×20 and only histogram equalization is used for pre-processing. In each folder, one subject has 3 randomly chosen face tracks for the gallery and 6 for the probe.

In this task, we mainly compare the proposed method with the prevalent image set classification methods, includ-

Fig. 13. Two real retrieval cases on *the Big Bang Theory* (upper) and *Prison Break* (lower). For space limitation, only top ten retrieved videos are shown here, and green (red) box indicates relevant (non-relevant) face track.

ing linear subspace based methods, Mutual Subspace Method (MSM) [16], Discriminative Canonical Correlations (DCC) [17]; nonlinear manifold based methods, Manifold-Manifold Distance (MMD) [18], Manifold Discriminant Analysis (MDA) [19]; affine subspace based methods, Affine Hull based Image Set Distance (AHISD) [20], Convex Hull based Image Set Distance (CHISD) [20], Sparse Approximated Nearest Points (SANP) [21] and a covariance-based method Covariance Discriminative Learning (CDL) [2].

For fair comparison, important parameters of each method are empirically tuned according to the recommendations in the original literatures as well as the source codes provided by the authors. Finally, we set the parameters as follows. In DCC, the eigen subspace dimension of each vector set is set to 10. In MDA, the number of between-class NN local models and the subspace dimension are specified as [19]. For both AHISD and CHISD, we use their linear version. The error penalty in CHISD is set to $C = 100$ as [20]. For SANP, we adopt the same weight parameters as [21] for the convex optimization. For CDL, we use Partial Least Squares (PLS) as the back-end classifier. For the proposed method, we simply keep all the parameters consistent with the previous experiments.

Table IV shows the comparison with the above prevalent image set classification methods on *YouTube Celebrities*. From this table, we are a little surprised that the proposed CVC with extremely compact 128 bits achieves a comparable or even superior performance compared with these state-of-the-art real-valued image set classification methods, which are elaborately designed for video face recognition. The reason behind such superiority lies in twofold: (a) covariance, as a natural statistical model, faithfully captures the original video information, containing appearance textures and dynamic actions; (b) the proposed binary code learning framework dexterously incorporates the stability constraint while considering discriminability, which makes the learned binary codes better generality on the unseen test data. Similarly, we further upgrade the non-overlapped SPC-CVC (L.W.) to overlapped version SPC-CVC (L.W.)(O.), with setting the step size of patch sliding (horizontal/vertical) to half of the patch size (width/height). Unsurprisingly, higher performance is achieved.

VI. Conclusion and Discussions

In this paper, we address the problem of face video retrieval in TV-series. To solve this problem, we propose a Compact Video Code (CVC) which at first models the face track by its sample covariance, then forms the compact binary representation by jointly optimizing for discriminability and stability which are particularly crucial for retrieval. Based on such framework, we extend the original covariance from pixel-level to patch-level, and further devise a novel video representation, named Spatial Pyramid Covariance (SPC), which is a composite of different parameterized patch-level covariances in a coarse-to-fine hierarchical structure. Accordingly, the optimizing of SPC configuration parameter is embedded into the discriminative binary code learning by means of multiple kernel learning technique. The learned CVC is computationally efficient and has been successfully applied to different classification tasks, including face video retrieval and identification with rather compact code.

As CVC is a binary representation, from another perspective, each bit of CVC can be regarded as an attribute classifier which shows the presence or absence of specific attribute of face tracks. Although these attributes have been proven discriminative but they cannot be described by human beings, that is to say, there is nothing of explicit semantic information. In the future, we intend to explore the connection between CVC and semantic attributes for more convenient and practical retrieval applications, not limited to faces, but extend to more

general objects, such as human behavior retrieval from massive surveillance data.

ACKNOWLEDGMENT

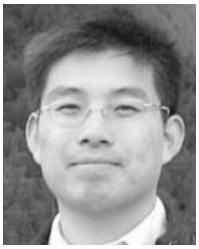
The authors would like to thank Mr. Shishi Qiao for giving us great help in collecting and processing the two TV-series databases.

REFERENCES

[1] C. Shan, "Face recognition and retrieval in video," in *Video Search Mining*. Heidelberg, Germany, Springer, 2010, pp. 235–260.
[2] R. Wang, H. Guo, L. S. Davis, and Q. Dai, "Covariance discriminative learning: A natural and efficient approach to image set classification," in *Proc. CVPR*, Jun. 2012, pp. 2496–2503.
[3] J. Lu, G. Wang, and P. Moulin, "Image set classification using holistic multiple order statistics features and localized multi-kernel metric learning," in *Proc. ICCV*, Dec. 2013, pp. 329–336.
[4] R. Vemulapalli, J. K. Pillai, and R. Chellappa, "Kernel learning for extrinsic classification of manifold features," in *Proc. CVPR*, 2013, pp. 1782–1789.
[5] M. Kim, S. Kumar, V. Pavlovic, and H. Rowley, "Face tracking and recognition with visual constraints in real-world videos," in *Proc. CVPR*, Jun. 2008, pp. 1–8.
[6] J. Sivic, M. Everingham, and A. Zisserman, "Person spotting: Video shot retrieval for face sets," in *Image and Video Retrieval*. Heidelberg, Germany, Springer, 2005, pp. 226–236.
[7] M. Everingham, J. Sivic, and A. Zisserman, "Hello! My name is... Buffy—Automatic naming of characters in TV video," in *Proc. BMVC*, 2006, pp. 899–908.
[8] O. Arandjelovic and R. Cipolla, "Automatic cast listing in feature-length films with anisotropic manifold space," in *Proc. CVPR*, vol. 2. Jun. 2006, pp. 1513–1520.
[9] R. G. Cinbis, J. Verbeek, and C. Schmid, "Unsupervised metric learning for face identification in TV video," in *Proc. ICCV*, Nov. 2011, pp. 1559–1566.
[10] M. Bauml, M. Tapaswi, and R. Stiefelhagen, "Semi-supervised learning with constraints for person identification in multimedia data," in *Proc. CVPR*, 2013, pp. 3602–3609.
[11] E. G. Ortiz, A. Wright, and M. Shah, "Face recognition in movie trailers via mean sequence sparse representation-based classification," in *Proc. CVPR*, 2013, pp. 3531–3538.
[12] Y. Li, R. Wang, Z. Cui, S. Shan, and X. Chen, "Compact video code and its application to robust face retrieval in TV-series,"in *Proc. BMVC*, 2014.
[13] O. M. Parkhi, K. Simonyan, A. Vedaldi, and A. Zisserman, "A compact and discriminative face track descriptor," in *Proc. CVPR*, 2014, pp. 1693–1700.
[14] Y. Li, R. Wang, Z. Huang, S. Shan, and X. Chen, "Face video retrieval with image query via hashing across euclidean space and riemannian manifold," in *Proc. CVPR*, 2015, pp. 4758–4767.
[15] Z. Dong, S. Jia, T. Wu, and M. Pei, "Face video retrieval via deep learning of binary hash representations," in *Proc. AAAI*, 2016, pp. 3471–3477.
[16] O. Yamaguchi, K. Fukui, and K.-I. Maeda, "Face recognition using temporal image sequence," in *Proc. FG*, Apr. 1998, pp. 318–323.
[17] T.-K. Kim, J. Kittler, and R. Cipolla, "Discriminative learning and recognition of image set classes using canonical correlations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1005–1018, Jun. 2007.
[18] R. Wang, S. Shan, X. Chen, and W. Gao, "Manifold-manifold distance with application to face recognition based on image set," in *Proc. IEEE CVPR*, Jun. 2008, pp. 1–8.
[19] R. Wang and X. Chen, "Manifold discriminant analysis," in *Proc. IEEE CVPR*, Jun. 2009, pp. 429–436.
[20] H. Cevikalp and B. Triggs, "Face recognition based on image sets," in *Proc. CVPR*, 2010, pp. 2567–2573.
[21] Y. Hu, A. S. Mian, and R. Owens, "Sparse approximated nearest points for image set classification," in *Proc. IEEE CVPR*, Jun. 2011, pp. 121–128.
[22] P. Lyman and H. Varian, *How Much Information?*, 2004. http://www.citeulike.org/group/626/article/381860
[23] W. Kong and W.-J. Li, "Isotropic hashing," in *Proc. NIPS*, 2012, pp. 1646–1654.
[24] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. VLDB*, vol. 99. 1999, pp. 518–529.
[25] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. SCG*, 2004, pp. 253–262.
[26] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proc. ICCV*, Sep./Oct. 2009, pp. 2130–2137.
[27] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Proc. NIPS*, 2009, pp. 1509–1517.
[28] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. NIPS*, 2008, pp. 1753–1760.
[29] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Proc. IEEE CVPR*. Jun. 2011, pp. 817–824.
[30] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Proc. IEEE CVPR*, Jun. 2010, pp. 3424–3431.
[31] M. Rastegari, A. Farhadi, and D. Forsyth, "Attribute discovery via predictable discriminative binary codes," in *Proc. ECCV*, 2012, pp. 876–889.
[32] 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.
[33] R. Salakhutdinov and G. Hinton, "Semantic hashing," *Int. J. Approx. Reasoning*, vol. 50, no. 7, pp. 969–978, Jul. 2009.
[34] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proc. NIPS*, 2009, pp. 1042–1050.
[35] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, Jan. 2011.
[36] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in *Proc. ICML*, 2011, pp. 353–360.
[37] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. ICML*, 2011, pp. 1–8.
[38] Y. Li, R. Wang, S. Shan, and X. Chen, "Hierarchical hybrid statistic based video binary code and its application to face retrieval in TV-series," in *Proc. FG*, May 2015, pp. 1–8.
[39] I. Jolliffe, *Principal Component Analysis*. Hoboken, NJ, USA: Wiley, 2005.
[40] H. Hotelling, "Relations between two sets of variates," in *Breakthroughs in Statistics*, New York, NY, USA: Springer, 1992, pp. 162–190.
[41] G. Shakhnarovich, J. W. Fisher, and T. Darrell, "Face recognition from long-term observations," in *Proc. ECCV*, 2002, pp. 851–865.
[42] O. Arandjelovic, G. Shakhnarovich, J. Fisher, R. Cipolla, and T. Darrell, "Face recognition with image sets using manifold density divergence," in *Proc. CVPR*, vol. 1. Jun. 2005, pp. 581–588.
[43] C. Kim and C.-H. Choi, "Image covariance-based subspace method for face recognition," *Pattern Recognit.*, vol. 40, no. 5, pp. 1592–1604, 2007.
[44] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2. 2006, pp. 2169–2178.
[45] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *Proc. ICCV*, vol. 2. Oct. 2005, pp. 1458–1465.
[46] S. Jayasumana, R. Hartley, M. Salzmann, H. Li, and M. Harandi, "Kernel methods on the riemannian manifold of symmetric positive definite matrices," in *Proc. CVPR*, 2013, pp. 73–80.
[47] X. Pennec, P. Fillard, and N. Ayache, "A riemannian framework for tensor computing," *Int. J. Comput. Vis.*, vol. 66, no. 1, pp. 41–66, 2006.
[48] S. Sra, "A new metric on the manifold of kernel matrices with application to matrix geometric means," in *Proc. NIPS*, 2012, pp. 144–152.
[49] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, "Geometric means in a novel vector space structure on symmetric positive-definite matrices," *SIAM J. Matrix Anal. Appl.*, vol. 29, no. 1, pp. 328–347, 2007.
[50] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[51] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "SimpleMKL," *J. Mach. Learn. Res.*, vol. 9, pp. 2491–2521, Nov. 2008.
[52] B. Schölkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2002.
[53] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *J. Optim. Theory Appl.*, vol. 109, no. 3, pp. 475–494, 2001.

[54] M. Fréchet, "Sur les ensembles de fonctions et les opérations linéaires," *CR Acad. Sci. Paris*, vol. 144, pp. 1414–1416, 1907.
[55] F. Riesz, "Sur une espèce de géométrie analytique des systèmes de fonctions sommables," *CR Acad. Sci. Paris*, vol. 144, pp. 1409–1411, 1907.
[56] A. Turpin and F. Scholer, "User performance versus precision measures for simple search tasks," in *Proc. SIGIR*, 2006, pp. 11–18.
[57] M. Zhu, "Recall, precision and average precision," Dept. Statist. Actuarial Sci., Univ. Waterloo, Waterloo, ON, Canada, Tech. Rep. 2004-09, 2004.
[58] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm

Yan Li (S'15) received the B.S. degree in computer science and technology from Nankai University, Tianjin, China, in 2010. He is currently pursuing the Ph.D. degree with the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China. He was a Research Scholar with the Lane Department of Computer Science and Electrical Engineering, Benjamin M. Statler College of Engineering and Mineral Resources, West Virginia University. His research interests include computer vision, pattern recognition, image processing, and, in particular, image and video face recognition, face retrieval, and binary code learning.



Ruiping Wang (S'08–M'11) received the B.S. degree in applied mathematics from Beijing Jiaotong University, Beijing, China, in 2003, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 2010. He was a Post-Doctoral Researcher with the Department of Automation, Tsinghua University, Beijing, from 2010 to 2012, and a Research Associate with the Computer Vision Laboratory, Institute for Advanced Computer Studies, University of Maryland, College Park, from 2010 to 2011. He has been a Faculty Member with the Institute of Computing Technology, Chinese Academy of Sciences, since 2012, where he is currently an Associate Professor. His research interests include computer vision, pattern recognition, and machine learning.

Zhen Cui (S'11–M'14) received the B.S. degree from Shandong Normal University in 2004, the M.S. degree from Sun Yat-sen University in 2006, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, in 2014. He was a Research Assistant with Nanyang Technological University from 2012 to 2012, and a Research Fellow with the Department of Electrical and Computer Engineering, National University of Singapore, from 2014 to 2015. He is currently an Associate Professor with Southeast University, China. His research interests cover computer vision, pattern recognition, and machine learning, especially focusing on deep learning, manifold learning, sparse coding, face detection/alignment/recognition, object tracking, image super resolution, and emotion analysis.

Shiguang Shan (M'04–SM'15) received the M.S. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 1999, and the Ph.D. degree in computer science from the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China, in 2004. He joined ICT, CAS, in 2002, and has been a Professor since 2010. He is currently the Deputy Director of the Key Laboratory of Intelligent Information Processing, CAS. He has authored over 200 papers in refereed journals and proceedings in the areas of computer vision and pattern recognition. His research interests cover computer vision, pattern recognition, and machine learning. He especially focuses on face recognition related research topics. He is a recipient of the China's State Natural Science Award in 2015, and the China's State S&T Progress Award in 2005 for his research work. He has served as an Area Chair for many international conferences, including ICCV'11, ICPR'12, ACCV'12, FG'13, ICPR'14, ICASSP'14, and ACCV'16. He is an Associate Editor of several journals, including the IEEE TRANSACTIONS ON IMAGE PROCESSING, COMPUTER VISION, AND IMAGE UNDERSTANDING,*Neurocomputing*, and *Pattern Recognition Letters*.

Xilin Chen (M'00–SM'09–F'16) received the B.S., M.S., and Ph.D. degrees in computer science from the Harbin Institute of Technology, Harbin, China, in 1988, 1991, and 1994, respectively. He was a Professor with the Harbin Institute of Technology from 1999 to 2005. He has been a Professor with the Institute of Computing Technology, Chinese Academy of Sciences, since 2004. He has authored one book and over 200 papers in refereed journals and proceedings in the areas of computer vision, pattern recognition, image processing, and multimodal interfaces. He is a recipient of several awards, including the China's State Natural Science Award in 2015 and the China's State Scientific and Technological Progress Award in 2000, 2003, 2005, and 2012 for his research work. He is a fellow of China Computer Federation. He served as an Organizing Committee/Program Committee Member for more than 50 conferences. He is an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA, a leading Editor of the *Journal of Computer Science and Technology*, and an Associate Editor-in-Chief of the *Chinese Journal of Computers*.