Hierarchical Hybrid Statistic based Video Binary Code and Its Application to Face Retrieval in TV-Series

Yan Li[1,2], Ruiping Wang[1], Shiguang Shan[1], Xilin Chen[1,3]
[1] Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
[3] Department of Computer Science and Engineering, University of Oulu, Oulu 90570, Finland
yan.li@vipl.ict.ac.cn, {wangruiping, sgshan, xlchen}@ict.ac.cn

***Abstract*— We address the problem of video face retrieval in TV-Series, which searches video clips based on the presence of particular character, given one video clip of his/hers. 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 highly efficient representation with low time and space complexity. To handle such problems, we propose a compact and discriminative binary representation for the huge body of video data based on a novel hierarchical hybrid statistic. Our method, named Hierarchical Hybrid Statistic based Video Binary Code (HHSVBC), first utilizes different parameterized Fisher Vectors (FVs) as frame representation that can encode multi-granularity low level variation information within the frame, and then models the video by its frame covariance matrix to capture high level variation information among video frames. To incorporate discriminative information and obtain more compact video signature, the high-dimensional video representation is further encoded to a much lower-dimensional binary vector, which finally yields the proposed HHSVBC. Specifically, each bit of the code, is produced via supervised learning in a max margin framework, which aims to make a trade-off between code discriminability and stability. Face retrieval experiments on two challenging large scale TV-Series video databases demonstrate the competitiveness of the proposed HHSVBC over state-of-the-art retrieval methods.**

I. INTRODUCTION

Video face retrieval in general is to retrieve shots containing a particular person given one video clip of his/hers [1]. It is a promising problem which has received an increasing amount of attention, especially in the era of Big Data, where huge body of videos can be found on the user-generated video sharing sites like *YouTube*, such as news programs, dramas, and homemade videos, etc. Technically speaking, video face retrieval is a crucial part for video analyzing and understanding. There exist a wide range of applications relying on it, e.g., '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 a number of short videos captured by a digital camera; and rapid locating and tracking of criminal suspects from masses of surveillance videos. In this paper, we mainly focus on the problem of video face retrieval in TV-Series with character's one video clip as query.

Undoubtedly, the core technique for video face retrieval is face recognition, which has long been established as one of the most active research directions in computer vision. However, compared with traditional face recognition, video face retrieval has its unique characteristics. In particular, different from traditional image based recognition, video provides much more information, which could be exploited to resolve the inherent ambiguities of image based recognition, like sensitivity to appearance variations caused by head pose, lighting, occlusion, and resolution [1]. Nevertheless, how to utilize the rich video information effectively needs to be considered adequately.

In this paper, we first utilize Fisher Vector (FV) [2], [3], [4] to adequately describe video frames, considering that FV has been widely recognized as a compelling discriminative descriptor. Then, we represent each video as the second-order covariance of all the frame descriptors. Though covariance matrix has been proved natural and efficient for video modeling [5], [6], [7], [8], it is not exploited as a pooling strategy to further integrate FV-like statistics. Therefore, our method essentially represents a video as statistic (covariance) of statistic (Fisher Vectors), named by us Hybrid Statistic (HS), where the FV encodes low level statistical information within each video frame, and the covariance matrix captures high level statistical information among video frames. Moreover, by exploiting different numbers of Gaussian modes in FV, we further obtain a series of HS in a coarse-to-fine multi-granularity hierarchical structure. It is also in this sense that our method is named by us Hierarchical Hybrid Statistic (HHS), serving as the first-stage modeling of video.

Although HHS as a general video descriptor has a certain descriptive capability by means of its complete modeling of video variations, it does not involve any supervised information which is crucial for retrieval. Besides, it is not compact enough due to its high dimensionality for fast retrieval, especially when the database is very large. To solve this problem, we further encode the HHS into a much lower-dimensional binary code as the final video signature, i.e., Hierarchical Hybrid Statistic based Video Binary Code (HHSVBC), where each bit of the code is learned by explicitly optimizing for discrimination in a max margin framework, inspired by a recent attribute learning method [9]. By doing this, *discriminability* and *stability* are considered jointly. To verify the effectiveness of our method, we set up two challenging large scale hit TV-Series video

978-1-4799-6026-2/15/$31.00 ©2015 IEEE

databases, i.e., *the Big Bang Theory* (BBT) and *Prison Break* (PB), and then conduct video face retrieval experiments on them. Experimental results show the competitiveness of the proposed HHSVBC over the state-of-the-art retrieval methods.

The rest of this paper is organized as follows: Section II discusses the related work of the proposed method. Then Section III describes the Hierarchical Hybrid Statistic (HHS) designed for video representation. Section IV shows the binary code learning and optimization algorithm. Next, Section V extensively evaluates our method on two large scale TV-Series databases. Finally, we end with a summary of conclusions and future work in Section VI.

II. Related Work

Recent years have witnessed more and more studies on video-related face classification [10], [11], [12], [13], [14], [15], [16], [17], especially the entertainment videos, e.g., movies, TV-Series. For instance, Everingham *et al*. [11] investigated the problem of labelling appearances of characters in TV-Series and films; Arandjelović and Zisserman [12] addressed the problem of automatically determining the cast of feature-length film; Cinbis *et al*. [13] addressed the identification problem for face tracks of TV-Series; Parkhi *et al*. [16] investigated the problem of video face verification with a compact and discriminative vector representation; and Sivic *et al*. [10] tried to retrieve shots containing particular person in video using an imaged face as query. More recently, various of extra information is utilized to expect performance boosting, e.g., Ortiz *et al*. [15] carried out the problem of identifying a face track using a collected large dictionary of still face images for assist; and Bäuml *et al*. [14] took advantage of subtitles and fan transcripts to implement person identification in TV-Series. While most of such previous works have been devoted to an end-to-end system, including those preprocessing stages such as shot boundary detection, face detection, tracking, and face track extraction, etc, it is generally believed that the pivotal technical components of video face retrieval lie in two aspects, i.e., the video data modeling and the final retrieval process, which are the exact topics of this paper.

Since video can be naturally treated as a collection of frame images, how to effectively represent a single image serves as the first basis step to the success of video modeling. Technically, image representation has always been of crucial importance in most computer vision tasks. A number of local descriptors have been invented for this purpose, and widely used examples include the classic Local Binary Pattern (LBP) [18], Histogram of Oriented Gradient (HOG) [19], and Scale-Invariant Feature Transform (SIFT) [20], etc. Another popular class of image representation is Bag-of-Visual words (BoV) [21], which consists in extracting a set of local descriptors, e.g., SIFT, in an image and in assigning each to the closest entry of the visual vocabulary, i.e., a codebook learned offline by clustering a large set of local descriptors. Recently, a kind of extensions of BoV appears which involves the use of more reasonable coding techniques based on soft assignment. Representative methods include Locality-constrained Linear Coding (LLC) [22], Super Vector (SV) [23], Kernel CodeBook (KCB) encoding [24], and Fisher Vector (FV) [2], [3], [4]. Although all these encoding methods have attracted lots of attention with their appealing performance in image classification, among them FV is shown to have an edge over the others on a number of image recognition benchmarks [25], [4], as it simultaneously utilizes the zero, first and second-order statistic to characterize the rich information within images. Due to its appealing property, in this paper we adopt the sophisticated FV as the basic image representation.

As a video is comprised of frames (i.e., images), in practice it is often to be treated as an image set. Compared with treating video as separated frames and processing it frame by frame, holistic modeling methods gradually exhibit their advantages of not only compact representation but also superior performance, after the pioneering work of Yamaguchi *et al*. [26]. Typical image set modeling methods include linear subspaces [26], [27], affine subspaces [28], covariance matrices [6], [7], [8]. Among these, covariance matrix which lies on a Riemannian manifold, as the raw second-order statistic of the image set, provides a natural representation for a video with any number of frames and any type of image features, and characterizes the complicated video structure more faithfully [6], [7], [8]. Also, as indicated in [6], linear subspace models originate from an eigen-decomposition of the covariance matrix while discarding some important information. Taking such into consideration, we resort to covariance matrices for representing videos in this paper.

Though the above representation methods (whether image or video) have gained success in various classification tasks, the high dimensionality limits their applicability to the retrieval scenario, which typically requires not only accurate but also compact representation for fast and large-scale search. Binary code is a natural solution to overcome such shortcoming, as it is quite convenient to match, and the storage capacity of very short codes is so large that all the digital images in this world can be indexed [29]. Having these properties, binary codes have been widely used as hash keys for retrieval, and important examples include: Locality Sensitive Hashing (LSH) [30], Spectral Hashing (SH) [31], ITerative Quantization (ITQ) [32]. However, none of these binary code learning methods would necessarily result in discriminative codes. In order to incorporate discriminability, recently a couple of supervised binary code learning methods form a blowout. Representative methods include: Semi-Supervised Hashing (SSH) [33], Kernel-based Supervised Hashing (KSH) [34], Supervised ITerative Quantization (SITQ) [32], etc.

III. Hierarchical Hybrid Statistic for Video Modeling

A. Fisher Vector Encoding for Image

The Fisher Vector (FV) is a high-dimensional image representation derived from Fisher Kernel [2], [3], [4], and it

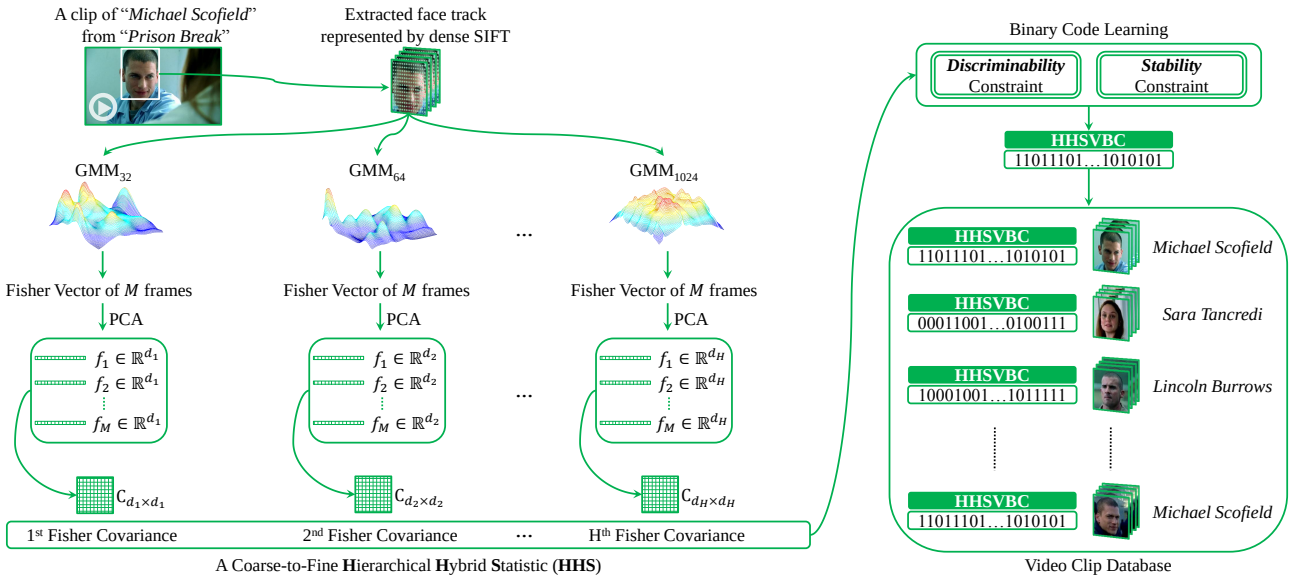


Fig. 1: Illustration of the proposed method. Given a video clip of one character as query, we extract the proposed Hierarchical Hybrid Statistic based Video Binary Code (HHSVBC) to represent it and use Hamming distance to retrieve video clips containing the specific character in database, which are also encoded in the form of HHSVBC.

is encoded by aggregating a large set of local features, e.g., dense SIFT features. In a nutshell, this is implemented by fitting a parametric generative model, always the Gaussian Mixture Model (GMM), to the local features, and then encoding the derivatives of the log-likelihood of the model with respect to its parameters. As in [3], we train a GMM with diagonal covariances, and only the derivatives with respect to the mean and variance of each Gaussian component are considered. This yields the representation which captures the first-order and second-order differences between the dense local features and each of the GMM centres:

$$\Phi_k^{(1)} = \frac{1}{T\sqrt{w_k}}\sum_{t=1}^{T}\gamma_t(k)\left(\frac{x_t-\mu_k}{\sigma_k}\right), \quad (1)$$

$$\Phi_k^{(2)} = \frac{1}{T\sqrt{w_k}}\sum_{t=1}^{T}\gamma_t(k)\frac{1}{\sqrt{2}}\left[\frac{(x_t-\mu_k)^2}{\sigma_k^2}-1\right]. \quad (2)$$

Here, T is the total number of local features of the target image, $\{\mu_k, \sigma_k, w_k\}$ refer to the mean, diagonal covariance, and mixture weight of the k^{th} Gaussian of GMM, which is computed on the training set, and $\gamma_t(k)$ is the soft assignment distribution of the t^{th} local feature x_t to the k^{th} Gaussian.

An FV ϕ is finally obtained by stacking the differences, i.e., $\phi = [\Phi_1^{(1)}, \Phi_1^{(2)}, ..., \Phi_K^{(1)}, \Phi_K^{(2)}]$, where K denotes the Gaussian number of GMM. To summarize, the above FV encoding describes how the distribution of local features of a particular image differs from the distribution fitted to the local features of all the training images.

B. Frame Covariance Matrix Encoding for Video

As mentioned in Section II, covariance as the raw second-order statistic of the video frames, provides a natural representation for a video with any number of frames and any type of image features, and therefore characterizes the video structure more faithfully [6]. Formally, let $F = [f_1, f_2, ..., f_M]$ be the data matrix of a video with M frames, where $f_i \in \mathbb{R}^d$ denotes the i^{th} frame with d-dimensional image representation. So we can represent the video with its $d \times d$ frame covariance matrix:

$$C = \frac{1}{M-1}\sum_{i=1}^{M}(f_i-\bar{f})(f_i-\bar{f})^T, \quad (3)$$

where $\bar{f}$ is the mean vector of all image frames. The diagonal entries of the covariance matrix encode the variance of each individual feature, and the off-diagonal entries encode their respective correlations. It is well known that the nonsingular covariance matrices lie on a Riemannian manifold rather than the well studied Euclidean space [35].

C. A Novel Hierarchical Hybrid Statistic

FV is a sound descriptor proposed for depicting the distribution of local features *within* a frame, whereas frame covariance matrix acts as a promising statistic depicts the appearance variance *among* frames. It would be wonderful if we can have a robust video representation that can characterize the frame-wise variance without losing the complete characterization of each frame. This motivates us to propose a novel Hybrid Statistic (HS) by taking into account the advantages of both FV and frame covariance matrix to yield the so called fisher covariance. Technically, we use FV as the frame representation when computing the frame covariance matrix, please see Fig. 1 for more intuitive understanding.

Let us view from a further perspective. The GMM in FV can be understood as a *probabilistic visual vocabulary*, where each Gaussian represents a word of the visual vocabulary: w_k encodes the relative frequency of the k^{th} word, μ_k the mean of the word, and σ_k the variation around the mean [36]. Therefore, the size of the visual vocabulary/codebook, i.e., the Gaussian number in GMM, is a crucial parameter closely related to the final performance of FV, and intuitively more Gaussians always lead to finer description, whereas

less Gaussians usually give a relatively coarser representation. However, it doesn't necessarily mean that larger vocabulary/codebook brings more competitive performance. Actually, that is highly correlated with the real data, which is also supported by the experimental results in Section V.

According to the above analysis, it might be difficult and inadvisable to set the vocabulary/codebook to a fixed size. As a consequence, we then further propose a new Hierarchical Hybrid Statistic (HHS), which integrates a series of Hybrid Statistic, i.e., fisher covariance matrices, with coarse-to-fine Gaussian numbers (please refer to Fig. 1). Formally, a target video can be represented as a set of Hybrid Statistic as follow.

$$C_i = \{C_{i1}, C_{i2}, ..., C_{iH}\}, \tag{4}$$

where C_i denotes the HHS of the i^{th} video, C_{ih} denotes each Hybrid Statistic, and H is the size of GMM parameters set.

IV. Binary Code Leaning

For each video, we can obtain a complete representation of Hierarchical Hybrid Statistic (HHS) as Eqn. (4), but meanwhile it magnifies the feature dimension, which certainly leads to higher space and time complexity, especially conflicting with the demand of retrieval task. Moreover, the HHS build above does not incorporate any supervision information, which will definitely favor the retrieval accuracy. To address this problem, our strategy is to discriminatively map the high-dimensional representation HHS into a much lower-dimensional Hamming space to produce a binary vector for each video. Two advantages can be induced by doing this, the concise space demand, and the low time cost (only bit-wise XOR operation). Inspired by the recent binary code learning work for attribute discovery [9], next we will discuss how to efficiently learn binary codes from those high-dimensional features.

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 compactness and inter-class separability. That is, videos from the same category should have similar codes, and videos from different categories should have better separability in the target Hamming space. Formally, let $b \in \{-1,1\}^{S\times N}$ denotes the binary codes of training instances, where S and N are the binary code length and the total number of training video instances, respectively. $b_i \in \{-1,1\}^{S\times 1}$ 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,

$$S_W = \sum_{c\in\{1:R\}} \sum_{i,j\in c} \mathrm{dis}(b_i, b_j), \tag{5}$$

$$S_B = \sum_{\substack{c'\in\{1:R\} \\ i\in c'}} \sum_{\substack{c''\in\{1:R\} \\ c'\neq c'', j\in c''}} \mathrm{dis}(b_i, b_j), \tag{6}$$

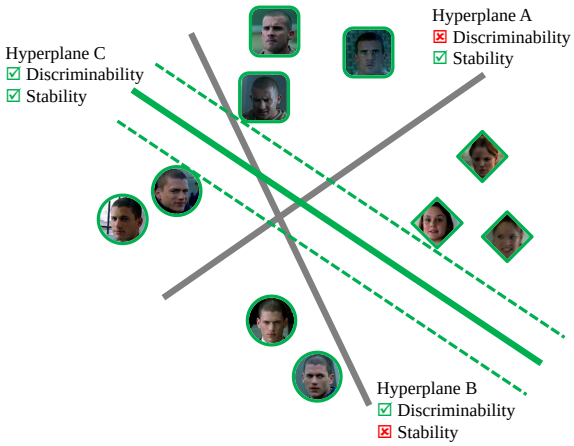


Fig. 2: Illustration of two constraints when learning binary code, i.e., discriminability and stability, where the two gray hyperplanes only satisfy either of the constraints, and the green hyperplane is the one that we need.

where R is the total number of categories, $\mathrm{dis}(\cdot)$ can be any available distance measurement in Hamming space. Thus, to implement a strong discrimination, we minimize the following energy function E_{disc}.

$$E_{disc} = S_W - \lambda_1 S_B. \tag{7}$$

Second, another crucial constraint in binary code learning is similarity preserving, and in this paper it is referred to ***stability***, which indicates that 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. 2 illustrates the relationship between discriminability and stability. In particular, we resort to the classic Support Vector Machine (SVM) to implement the binary code learning. More specifically, each bit of the binary code can be thought of as a split of the original feature space into two half-spaces, and each bit can also be visualized as a hyperplane that separates instances that have binary code value -1 v.s. the ones that have value 1.

Before the binary code learning, there is still one remaining problem that the proposed Hybrid Statistic, i.e., fisher covariance matrix, lies on a specific Riemannian manifold rather than the well-studied Euclidean space which a lot of off-the-shelf learning algorithms apply to. To address this problem, we recur to Reproducing Kernel Hilbert Space (RKHS) [37] with a sophisticated Riemannian kernel proposed in [6] for help as Eqn. (8).

$$K_{ij} = \mathrm{trace}[\log(P_i)\cdot\log(P_j)], \tag{8}$$

where P_i, P_j are Symmetric Positive Definite (SPD) matrices lying on Riemannian manifold, $\log(\cdot)$ is the ordinary matrix logarithm operator, and $\mathrm{trace}[\cdot]$ denotes the matrix trace.

As mentioned in Section III-C, the final Hierarchical Hybrid Statistic (HHS) is composed of a series of Hybrid

Statistic, i.e., fisher covariance matrices with different numbers of Gaussian. Hence, to mining their complementary to a wide extent, Multiple Kernel Learning (MKL) is used when training SVM. In particular, we build S hyperplanes/splits each corresponding to one bit of the binary code by training S kernel SVMs individually. More concretely, we denote the s^{th} hyperplane by ω^s ($s = 1, \cdots, S$), and the energy function can be formulated as follow.

$$\begin{aligned} E_{stab} &= \delta \sum_{\substack{s\in\{1:S\}\\ i\in\{1:N\}}} \max(1 - b_i^s(\omega^{sT}\varphi^s(C_i)), 0) \\ &\quad + \frac{1}{2}\sum_{s\in\{1:S\}} \|\omega^s\|^2 + \eta \sum_{s\in\{1:S\}} \|\beta^s\|_p, \\ s.t. &\quad < \varphi^s(C_i), \varphi^s(C_j) >= \sum_{h\in\{1:H\}} \beta_h^s K_h(C_{ih}, C_{jh}), \\ &\quad \beta^s \geq 0, \forall s \in \{1:S\}, \end{aligned} \tag{9}$$

where C_i, C_j, C_{ih}, C_{jh}, H are consistent with the definitions in Eqn. (4), $\varphi^s(\cdot)$ denotes the s^{th} mapping function to map C_i to a RKHS, $b_i^s \in \{-1, 1\}$ indicates in which side of the s^{th} hyperplane the i^{th} instance lies, K_h is the h^{th} kernel matrix computed by Eqn. (8), $\beta^s = [\beta_1^s, \beta_2^s, ..., \beta_H^s]^T$ is the combination coefficients of the H kernel matrices for the s^{th} bit, and arbitrary norm can be assigned by setting the subscript p.

After the above analysis, we can reach the final objective function by combining Eqn. (7) and Eqn. (9) to simultaneously consider the discriminability and stability of the target binary code:

$$\min_{\omega,\beta,b} \quad E_{disc} + \lambda E_{stab}. \tag{10}$$

B. Optimization Algorithm

Since the objective function Eqn. (10) is non-convex, it is infeasible to find a global analytical solution. In practice, block coordinate descent method to independently optimize each individual component for iteratively updating ω, β, and b. The pseudo-code of optimization can be found in Algorithm 1. Next, we will give a detailed discussion. Assume that we have N training video instances of R categories, and for each instance we have the computed Hierarchical Hybrid Statistic (HHS) C_i with its ground truth label $l_i \in \{1, 2, ..., R\}$, where $i \in \{1, 2, ..., N\}$.

Initialization: first, we compute the H kernel matrices just once, i.e., $K_h \in \mathbb{R}^{N\times N}, h \in \{1, 2, ..., H\}$, using Eqn. (8); second, the kernel combination coefficients of each bit, i.e., $\beta^s, s \in \{1, 2, ..., S\}$, is initially set to equal weight as $[\frac{1}{H}, \frac{1}{H}, ..., \frac{1}{H}]^T$; third, for each bit compute the corresponding integrated kernel matrix $K^s \in \mathbb{R}^{N\times N}, s \in \{1, 2, ..., S\}$ by $K^s = \sum_{h=1}^H \beta_h^s K_h$; lastly, we conduct the initialization of binary codes $b \in \{-1, 1\}^{S\times N}$ just randomly.

Fix b to optimize ω and β: in this step, we use $b^s \in \{-1, 1\}^{1\times N}$ as training labels to train the s^{th} bit's kernel SVM hyperplane ω^s with $K^s \in \mathbb{R}^{N\times N}$ as training input. As we also need to learn the kernel combination coefficients β^s of each bit, here we adopt an off-the-shelf MKL method, i.e., SimpleMKL [38], to simultaneously learn ω^s and β^s for each bit. One more point, as we use kernel trick to handle the non-linear mapping problem, the hyperplane ω^s of each bit is learnt in the form of projection $U^s \in \mathbb{R}^{N\times 1}$ according to Riesz representation theorem, which further forms the integrated $U = [U^1, U^2, ..., U^S] \in \mathbb{R}^{N\times S}$ corresponding to ω.

Fix ω and β to optimize b: having the learned ω and β, in this step we use them to predict b and further optimize it. For each bit, we first compute $K^s = \sum_{h=1}^H \beta_h^s K_h$, and then use the learned hyperplane ω^s (i.e., U^s) to predict b^s by quantizing the kernel SVM's outputs as $b^s = sgn(U^{sT}K^s)$. Next, we feed the newly predicted $b = [b^1, b^2, ..., b^S]^T \in \{-1, 1\}^{S\times N}$ into a subgradient descend-based binary code optimization method proposed in [9]. It is during this step that the discriminability of binary codes is guaranteed.

Convergence criteria: the whole optimization is looped by iteratively update ω, β, and b, and in practice we find that usually two or three times iterations can make the objective function converge.

Parameters setting: the objective function in Eqn. (10) is used just as a conceptual formulation to depict the discriminability and stability. As the optimization is conducted separately in an iterative manner as in Algorithm 1, the parameter λ mainly plays a role of balancing the two components E_{disc} and E_{stab}, and is simply set to equally weight them. Besides, the only substantial parameters are the soft margin parameters δ in Eqn. (9), which is simply set to 1 as standard SVM, and the MKL parameter η in Eqn. (9), which is also set to the default value as recommended by the authors of SimpleMKL [38], i.e., 10^{-8}. As for λ_1 in Eqn. (7), which just serves as a normalization factor, and it is set to the value to balance the numbers of data pairs in S_W and S_B.

Algorithm 1 Optimization

Input: Training instances $C_i = \{C_{i1}, C_{i2}, ..., C_{iH}\}$ and ground truth labels $l_i \in \{1, 2, ..., R\}$, where $i \in \{1, 2, ..., N\}$.
Output: $b \in \{-1, 1\}^{S\times N}$.
Initialization:
1. Compute $K_h \in \mathbb{R}^{N\times N}, h \in \{1, 2, ..., H\}$ with Eqn. (8);
2. $\beta^s = [\frac{1}{H}, \frac{1}{H}, ..., \frac{1}{H}]^T, s \in \{1, 2, ..., S\}$;
3. $K^s = \sum_{h=1}^H \beta_h^s K_h, s \in \{1, 2, ..., S\}$;
4. Randomly initialize $b \in \{-1, 1\}^{S\times N}$;
Repeat several times
5. Fix b to optimize ω (i.e., U) and β with Eqn. (9);
6. Fix ω (i.e., U) and β to optimize b with Eqn. (7);
End
7. $K^s = \sum_{h=1}^H \beta_h^s K_h, s \in \{1, 2, ..., S\}$;
8. $b^s = sgn(U^{sT}K^s), s \in \{1, 2, ..., S\}$;
9. $b = [b^1, b^2, ..., b^S]^T$.

V. Experiments

A. Two Large Scale TV-Series Video Databases

To investigate the video face retrieval problem, we set up two large scale video databases[1] from two hit American

[1]The databases can be downloaded from http://vipl.ict.ac.cn/.

Fig. 3: Some exemplar face tracks of the two constructed video databases, where the top three rows come from *the Big Bang Theory*, and the rest three come from *Prison Break*.

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 (some examples are shown in Fig. 3). The original videos are acquired from Blu-ray discs with 720P resolution. To extract the final face tracks, several technologies, e.g., shot boundary detection, face detection, tracking, and facial landmark localization are conducted. To guarantee the purity of databases, we invite 5 fans of each TV-Series to annotate every extracted face track. More detail, we deal with the first season of both TV-Series, i.e., 17 episodes for BBT, and 22 for PB, and finally we collect 4667 and 9435 face tracks from BBT and PB, respectively.

B. Experimental Settings

For the proposed method, i.e., Hierarchical Hybrid Statistic based Video Binary Code (HHSVBC), we fix the hierarchical parameter H to 6, which ranges within 32, 64, 128, 256, 512, 1024 Gaussian numbers, mainly considering the trade-off between retrieval performance and computation complexity. The face size is fixed at 80×64 pixels. For each database, we randomly select 10 face tracks per character for training, and leave the rest as test data. Query set of each database is consist of 10 face tracks per main character. For quantitative evaluation, we use the standard mean Average Precision (mAP) [39] and the precision recall curve as measurements. For fair comparison, important parameters of the competitive methods are empirically tuned according to the recommendations in the original literatures as well as the source codes provided by the authors.

C. Evaluation on Different Gaussian Numbers

As discussed in Section III, the crucial parameter of Hybrid Statistic (HS), i.e., the Gaussian number, has a high correlation with the accuracy of target representation. We are more willing to believe that finer visual vocabulary/codebook (i.e., more Gaussians) can capture more detailed structure compared with coarser vocabulary/codebook (i.e., less Gaussians), especially in the face-related problem which needs very precise description. Table I shows the retrieval performance in mAP with different Gaussian numbers for Hybrid Statistic (HS) computation. An interesting phenomenon is observed that the variation tendencies of this parameter on BBT and PB present completely opposite. In BBT, which contains many relatively simple close-up shots, finer Hybrid Statistic (HS) brings better performance, whereas in PB, which has relatively lower facial resolution, variable lighting, and frequent occlusion, coarser Hybrid Statistic (HS) has more advantages in turn. The last two rows of Table I show the performance evaluation of the integrated Hierarchical Hybrid Statistic (HHS), where subscript E.W. means using equal weights to combine the kernel matrices, and L.W. means learned weight with the proposed method. It is obviously to see that compared with straightforward summing, combining with learned weight fully takes advantage of the complementarity among Hybrid Statistic (HS) of different parameters. For more intuitive understanding, we visualize the learned weights of the first 16 bits on each database, and please see Fig. 4.

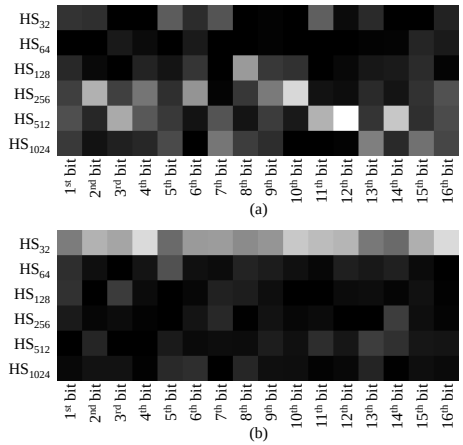


Fig. 4: Visualization of the learned kernel weights, i.e., combination coefficients of the 6 Hybrid Statistic (HS) of the first 16 bits on two databases, i.e., (a) *the Big Bang Theory* and (b) *Prison Break*. In this visualization, larger weight is shown in lighter color.

D. Evaluation on Different Frame Representations

To further evaluate the effectiveness of the proposed Hybrid Statistic (HS), here we compare Fisher Vector (FV) with different front-end features as frame representations to compute covariance matrix. In this part, raw gray feature, histogram equalized gray feature, Local Binary Pattern (LBP) [18], Histograms of Oriented Gradients (HOG) [19], and Dense Scale-Invariant Feature Transform (DSIFT) [20] are taken into consideration. As the superiority of FV against other BoV encoding methods has been verified in plenty of literatures [3], [25], [4], we will not take such features into account. To simplify reproducibility, the HOG, DSIFT, FV

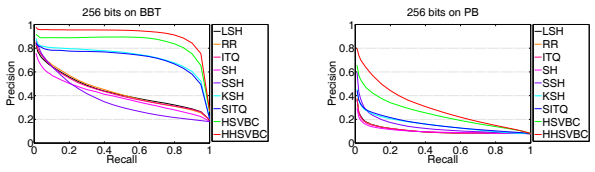
TABLE I: Evaluation on different frame representations used in covariance computation with mAP on two databases.

Feature	*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
Gray	0.3104	0.3510	0.3786	0.4032	0.4172	0.4430	0.1010	0.1018	0.1042	0.1075	0.1135	0.1189
Gray (HE)	0.3167	0.3662	0.4208	0.4666	0.4873	0.5120	0.1013	0.0996	0.1054	0.1103	0.1134	0.1187
LBP	0.3839	0.4653	0.5162	0.5332	0.5489	0.5678	0.1277	0.1382	0.1507	0.1667	0.1801	0.2043
HOG	0.4057	0.4874	0.5639	0.5998	0.6209	0.6479	0.1181	0.1273	0.1464	0.1619	0.1726	0.1924
DSIFT	0.4601	0.5319	0.6094	0.6453	0.6611	0.6803	0.1189	0.1307	0.1518	0.1714	0.1837	0.2008
HS_{32}	0.5729	0.6524	0.7325	0.7505	0.7774	0.7961	0.1265	0.1458	0.1758	0.2069	0.2270	0.2616
HS_{64}	0.5555	0.6663	0.7435	0.7731	0.8029	0.8131	0.1306	0.1533	0.1853	0.2130	0.2321	0.2598
HS_{128}	0.5856	0.7186	0.7829	0.8071	0.8257	0.8406	0.1329	0.1519	0.1857	0.2167	0.2323	0.2577
HS_{256}	0.5783	0.7213	0.7909	0.8195	0.8426	0.8600	0.1344	0.1547	0.1822	0.2012	0.2189	0.2434
HS_{512}	0.6208	0.7918	0.8494	0.8573	0.8663	0.8730	0.1393	0.1552	0.1824	0.1996	0.2158	0.2365
HS_{1024}	0.6530	0.8078	0.8597	0.8660	0.8731	0.8779	0.1450	0.1608	0.1829	0.2009	0.2119	0.2303
$HHS_{E.W.}$	0.6194	0.7333	0.8194	0.8214	0.8374	0.8512	0.1372	0.1590	0.1925	0.2121	0.2235	0.2490
$HHS_{L.W.}$	**0.7177**	**0.8763**	**0.9113**	**0.9078**	**0.9116**	**0.9172**	**0.1703**	**0.1950**	**0.2279**	**0.2585**	**0.2743**	**0.3035**

features are computed by using the publicly available VLFeat toolbox [40] (version 0.9.17) with the defaults options. The LBP implementation is acquired from Oulu CMV with default parameters of uniformed patterns. Moreover, for fair comparison, the dimensionalities of all front-end features are reduced to the same value by PCA (300 for all the experiments). The comparison results can be found on Table I. The proposed Hybrid Statistic (HS) shows its overwhelming superiority against the other features, and this implies that covariance matrix possibly favors the representation based on higher-order statistic, like FV. Besides, we can also find that among the competitive features, DSIFT performs best.

E. Evaluation with State-of-the-art Binary Code Learning Methods

Apart from the novel video representation, in this paper we also proposed a binary code learning method. To evaluate its performance, in this part we select several representative binary code learning methods for comparison, including Locality Sensitive Hashing (LSH) [30], Spectral Hashing (SH) [31], Random Rotation (RR) [32], Semi-Supervised Hashing (SSH) [33], Kernel-based Supervised Hashing (KSH) [34], ITerative Quantization (ITQ) [32], and Supervised ITerative Quantization (SITQ) [32]. For fair comparison, we fix the video representation part for all the methods by using the proposed Hybrid Statistic (HS) with 256 Gaussians as suggested in [4]. However, most of the competitive methods are based on Euclidean space and do not have the kernel version. Hence, here we use the Log-Euclidean Distance (LED) as [6] to map the Riemannian Hybrid Statistic (HS) to Euclidean space, in which all the competitive methods can handle. Table II and Fig. 5 show the comparison results in mAP and PR curve (please find more in supplementary materials). 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. Compared with those state-of-the-art supervised hashing methods, ours achieves more excellent performance. A possible interpretation is our method also incorporates the stability while considering the discriminability, which makes the learned model better stability on unseen data.



(a) (b)

Fig. 5: Comparison with state-of-the-art binary code learning methods with precision recall curve on two databases, i.e., (a) *the Big Bang Theory* (BBT) and (b) *Prison Break* (PB). Without loss of generality, we only show the comparison with 256 bits, please find more in supplementary materials.

F. Evaluation of Computational Complexity

In this subsection, we discuss about the computational complexity of the proposed binary code learning method. Compared with SH, RR, ITQ, SSH, SITQ, which contain the operation of matrix decomposition, the proposed method is computational efficient. More specifically, with pre-computed kernel matrices, the whole training process of our method takes less than 10 seconds for 256 bits on a PC with Intel Core i7 processor of 3.40GHz. As for the front-end Hierarchical Hybrid Statistic (HHS) computation, most of the time is spent on computing the low-level descriptors for the video frames (i.e., DSIFTs: 100ms per frame).

VI. Conclusion

In this paper, we address the problem of video face retrieval. To solve this problem, we first decompose it into two parts, i.e., complete video modeling and discriminative binary code learning. For the former part, we take advantages of Fisher Vector (FV) for frame encoding and frame covariance matrix for video encoding, and propose a novel video representation, named Hybrid Statistic (HS), which can also be regarded as the statistic of statistic. Moreover, to make full use of the complementarity among different parameterized HS, we further extend the basic HS to a coarse-to-fine Hierarchical Hybrid Statistic (HHS), which involves multi-granularity information. To fit the proposed HHS to retrieval task, a binary code learning method which jointly optimizes discriminability and stability is proposed to map HHS to the final Video Binary Code, i.e., HHSVBC. The learned HHSVBC has been successfully applied to character retrieval on two challenging video databases. Viewing from another

TABLE II: Comparison with state-of-the-art binary code learning methods with mAP on two databases.

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 [30]	0.3533	0.3783	0.4093	0.4148	0.4414	0.4383	0.0959	0.0998	0.1048	0.1081	0.1078	0.1101
RR [32]	0.3885	0.4207	0.4042	0.4507	0.4622	0.4407	0.0985	0.0981	0.1018	0.1042	0.1065	0.1105
ITQ [32]	0.3434	0.3445	0.4033	0.4257	0.4428	0.4324	0.0999	0.1129	0.1083	0.1114	0.1095	0.1098
SH [31]	0.4086	0.4225	0.3802	0.3809	0.3765	0.3972	0.0914	0.0901	0.0978	0.0964	0.1048	0.1059
SSH [33]	0.3401	0.3134	0.2830	0.2757	0.2878	0.3656	0.1138	0.1527	0.1488	0.1417	0.1409	0.1436
KSH [34]	0.4981	0.5799	0.6506	0.6965	0.7094	0.7300	0.1218	0.1571	0.1546	0.1619	0.1630	0.1599
SITQ [32]	0.5384	0.6185	0.6702	0.6891	0.7006	0.7165	0.1070	0.1211	0.1326	0.1462	0.1578	0.1640
HSVBC	0.6208	0.7918	0.8494	0.8573	0.8663	0.8730	0.1393	0.1552	0.1824	0.1996	0.2158	0.2365
HHSVBC	**0.7177**	**0.8763**	**0.9113**	**0.9078**	**0.9116**	**0.9172**	**0.1703**	**0.1950**	**0.2279**	**0.2585**	**0.2743**	**0.3035**

perspective, each bit of the HHSVBC can be understood as an attribute classifier which indicates the presence or absence of specific attribute of the video. While these attributes have been proven discriminative, they can hardly be described by human beings, i.e., there is nothing of explicit semantic information. In the future, we would explore the inherent connection between HHSVBC and semantic attributes for more convenient and practical computer vision applications.

VII. Acknowledgements

This work is partially supported by Natural Science Foundation of China under contracts Nos. 61222211, 61379083, 61390511, and the FiDiPro program of Tekes.

References

[1] C. Shan, "Face recognition and retrieval in video," in *Video Search and Mining*. Springer, 2010, pp. 235–260.

[2] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *CVPR*. IEEE, 2007, pp. 1–8.

[3] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *ECCV*. Springer, 2010, pp. 143–156.

[4] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *IJCV*, vol. 105, no. 3, pp. 222–245, 2013.

[5] 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 *BMVC*. BMVA Press, 2014.

[6] R. Wang, H. Guo, L. S. Davis, and Q. Dai, "Covariance discriminative learning: a natural and efficient approach to image set classification," in *CVPR*. IEEE, 2012, pp. 2496–2503.

[7] J. Lu, G. Wang, and P. Moulin, "Image set classification using holistic multiple order statistics features and localized multi-kernel metric learning," in *ICCV*, 2013.

[8] R. Vemulapalli, J. K. Pillai, and R. Chellappa, "Kernel learning for extrinsic classification of manifold features," in *CVPR*. IEEE, 2013, pp. 1782–1789.

[9] M. Rastegari, A. Farhadi, and D. Forsyth, "Attribute discovery via predictable discriminative binary codes," in *ECCV*. Springer, 2012, pp. 876–889.

[10] J. Sivic, M. Everingham, and A. Zisserman, "Person spotting: video shot retrieval for face sets," in *Image and Video Retrieval*. Springer, 2005, pp. 226–236.

[11] M. Everingham, J. Sivic, and A. Zisserman, "Hello! my name is... buffy–automatic naming of characters in tv video," 2006.

[12] O. Arandjelović and R. Cipolla, "Automatic cast listing in feature-length films with anisotropic manifold space," in *CVPR*, vol. 2. IEEE, 2006, pp. 1513–1520.

[13] R. G. Cinbis, J. Verbeek, and C. Schmid, "Unsupervised metric learning for face identification in tv video," in *ICCV*. IEEE, 2011, pp. 1559–1566.

[14] M. Bäuml, M. Tapaswi, and R. Stiefelhagen, "Semi-supervised learning with constraints for person identification in multimedia data," in *CVPR*. IEEE, 2013, pp. 3602–3609.

[15] E. G. Ortiz, A. Wright, and M. Shah, "Face recognition in movie trailers via mean sequence sparse representation-based classification," in *CVPR*. IEEE, 2013, pp. 3531–3538.

[16] O. M. Parkhi, K. Simonyan, A. Vedaldi, and A. Zisserman, "A compact and discriminative face track descriptor," 2014.

[17] L. Best-Rowden, H. Han, C. Otto, B. Klare, and A. K. Jain, "Unconstrained face recognition: Identifying a person of interest from a media collection," *TIFS*, vol. 9, pp. 2144–2157, 2014.

[18] T. Ahonen, A. Hadid, and M. Pietikäinen, "Face recognition with local binary patterns," in *ECCV*. Springer, 2004, pp. 469–481.

[19] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, vol. 1. IEEE, 2005, pp. 886–893.

[20] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, vol. 60, no. 2, pp. 91–110, 2004.

[21] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision, ECCV*, vol. 1, no. 1-22, 2004, pp. 1–2.

[22] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *CVPR*. IEEE, 2010, pp. 3360–3367.

[23] X. Zhou, K. Yu, T. Zhang, and T. S. Huang, "Image classification using super-vector coding of local image descriptors," in *ECCV*. Springer, 2010, pp. 141–154.

[24] J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. Smeulders, "Kernel codebooks for scene categorization," in *ECCV*. Springer, 2008, pp. 696–709.

[25] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman, "The devil is in the details: an evaluation of recent feature encoding methods," in *BMVC*. BMVA Press, 2011, pp. 76.1–76.12.

[26] O. Yamaguchi, K. Fukui, and K.-i. Maeda, "Face recognition using temporal image sequence," in *FG*. IEEE, 1998, pp. 318–323.

[27] T.-K. Kim, J. Kittler, and R. Cipolla, "Discriminative learning and recognition of image set classes using canonical correlations," *PAMI*, vol. 29, no. 6, pp. 1005–1018, 2007.

[28] H. Cevikalp and B. Triggs, "Face recognition based on image sets," in *CVPR*. IEEE, 2010, pp. 2567–2573.

[29] P. Lyman and H. Varian, "How much information 2003?" 2004.

[30] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *VLDB*, vol. 99, 1999, pp. 518–529.

[31] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *NIPS*, 2008, pp. 1753–1760.

[32] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *CVPR*. IEEE, 2011, pp. 817–824.

[33] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *CVPR*. IEEE, 2010, pp. 3424–3431.

[34] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *CVPR*. IEEE, 2012, pp. 2074–2081.

[35] X. Pennec, P. Fillard, and N. Ayache, "A riemannian framework for tensor computing," *IJCV*, vol. 66, no. 1, pp. 41–66, 2006.

[36] F. Perronnin, C. Dance, G. Csurka, and M. Bressan, "Adapted vocabularies for generic visual categorization," in *ECCV*. Springer, 2006, pp. 464–475.

[37] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[38] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "Simplemkl," *JMLR*, vol. 9, pp. 2491–2521, 2008.

[39] A. Turpin and F. Scholer, "User performance versus precision measures for simple search tasks," in *SIGIR*. ACM, 2006, pp. 11–18.

[40] A. Vedaldi and B. Fulkerson, "Vlfeat: An open and portable library of computer vision algorithms," in *MM*. ACM, 2010, pp. 1469–1472.