Effective Multiple Feature Hashing for Large-Scale Near-Duplicate Video Retrieval

Jingkuan Song, Yi Yang, Zi Huang, Heng Tao Shen, and Jiebo Luo

***Abstract*—Near-duplicate video retrieval (NDVR) has recently attracted much research attention due to the exponential growth of online videos. It has many applications, such as copyright protection, automatic video tagging and online video monitoring. Many existing approaches use only a single feature to represent a video for NDVR. However, a single feature is often insufficient to characterize the video content. Moreover, while the accuracy is the main concern in previous literatures, the scalability of NDVR algorithms for large scale video datasets has been rarely addressed. In this paper, we present a novel approach—Multiple Feature Hashing (MFH) to tackle both the accuracy and the scalability issues of NDVR. MFH preserves the local structural information of each individual feature and also globally considers the local structures for all the features to learn a group of hash functions to map the video keyframes into the Hamming space and generate a series of binary codes to represent the video dataset. We evaluate our approach on a public video dataset and a large scale video dataset consisting of 132,647 videos collected from YouTube by ourselves. This dataset has been released (http://itee.uq.edu.au/shenht/UQ_VIDEO/). The experimental results show that the proposed method outperforms the state-of-the-art techniques in both accuracy and efficiency.**

***Index Terms*—Hashing, manifold learning, near-duplicate video retrieval, optimization, video indexing.**

I. Introduction

WITH the rapid development of the Internet techniques, video capturing devices, and video editing softwares, the number of online videos continues to grow at an astonishing speed. Meanwhile, the video-related services, such as video sharing, monitoring, advertising and recommendation, inspire online users' interests and participation to video-related activities, including uploading, downloading, commenting, and searching. A substantial number of videos are uploaded and shared on social websites every day. It has been shown that there are a large number of near-duplicate videos (NDVs) on the Web, which are generated in different ways, ranging from simple reformatting, to different acquisitions, transformations, editions, and mixtures of different effects [26], [28], [41]. The presence of massive NDVs imposes strong demand for effective near-duplicate video retrieval (NDVR) in many novel applications, such as copyright enforcement, video tagging, online video usage monitoring, video database cleansing, cross-modal divergence detection, video result re-ranking, and so on. To list a few, a typical scenario could be that a Web user wants to get some novel videos but ends up with lots of NDVs in the top-ranked search results returned by the search engine. Another situation could be that a video producer expects to avoid their copyright protected videos being shared on the Internet. Both occasions require NDVR techniques to achieve their goals.

During recent years, much research effort have been made to NDVR. Most of the existing algorithms usually adopt the following framework for NDVR [26], [31], [43], [48]. First, a video is divided into a sequence of keyframes extracted by time sampling or shot-boundary detection algorithms. These keyframes are represented by their visual features, such as color histogram, Local Binary Pattern (LBP), etc. The sequence of the keyframes' features is then regarded as the signature of the video. For NDVR, the system needs to compare the similarity between the signatures of the query video and each dataset video and return the dataset videos which are most similar to the query example. Typically, both spatial and temporal information is used to measure the similarity between two videos [6], [10], [26], [43]. There are also some existing works which summarize the whole video clip with a single and global signature to achieve real-time retrieval [27], [28]. However, they are usually not effective in representing long videos. More recently, pair-wise frame correlation between two videos has also been exploited to measure the near-duplicate relationship [18]. A recent survey on NDVR can be found in [17].

Intuitively, the multiple features of video clips, each of which reflects the specific information of video data, are complementary to each other. Simultaneously utilizing the multiple features is helpful for disambiguation. For example, local signature is less robust to the changes in frame rate, video length, captions and global signature is sensitive to changes in contrast, brightness, scale, camera viewpoint, and so on. It is difficult, if not impossible, to find a single visual feature which is robust to all types of variations. It is therefore particularly important to exploit multiple features for NDVR, where various variations and distortions could take place between NDVs.

The scalability of NDVR algorithm is becoming a more and more critical research issue as the number of videos is growing rapidly and most of the video content features are highly complex. How to construct efficient indexing structures to facilitate fast search over large scale video datasets is important for online NDVR, which demands real-time response. Many indexing methods in database literature have been proposed to support

Manuscript received August 26, 2012; revised January 22, 2013; accepted April 15, 2013. Date of publication July 03, 2013; date of current version November 13, 2013. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Alan Hanjalic.

J. Song, Z. Huang, and H. T. Shen are with the School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane 4072, Australia (e-mail: jk.song@itee.uq.edu.au; huang@itee.uq.edu.au; shenht@itee.uq.edu.au).

Y. Yang is with the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: yiyang@cs.cmu.edu).

J. Luo is with the Department of Computer Science, University of Rochester, Rochester, NY 14627 USA (e-mail: jluo@cs.rochester.edu).

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/TMM.2013.2271746

multimedia search, such as tree-structures [11], data reduction [9], hashing [5], and others [4]. The interest of the indexing methods is normally evaluated with the trade-offs offered with respect to search quality and efficiency, but efficiency is their main concern. How to better preserve the near-duplicate relationships among videos in the indexing structure needs to be further studied.

In this work, we propose a new hashing algorithm, namely Multiple Feature Hashing (MFH), for accurate and efficient NDVR based on multiple visual features. To this end, multiple features (i.e., global and local visual features) of videos are analyzed to obtain the hash codes, which are more accurate to represent video content. Here we regard a video as a sequence of keyframes. Inspired by the recent study which shows that it is beneficial for multimedia semantics understanding to exploit the non-linear manifold structure of multimedia data [16], [44], [45], we propose to make use of the manifold information of keyframes when learning the hash codes for keyframes. Different from [39], [47] which can only make use of single feature to learn the hash code of keyframes, the individual structural information of each feature type is preserved in MFH proposed in this paper. Therefore MFH is more capable to make best use of the available information contained in different feature types for a more accurate NDVR. To cope with large scale data, MFH simultaneously learns the hash codes of the training data as well as a series of hash functions to infer the hash code of the videos which are outside the training set, making it apparently different from existing related work such as self-taught hashing [47] and spectral hashing [39]. NDVs are expected to have the same or similar hash codes. Once the hash codes are obtained, only Hamming distance calculation is performed to compute the similarity between videos. Such an operation is very fast, making it possible to perform NDVR over large scale video datasets in real time. It is worth highlighting the follow aspects of this paper.

1) We propose a new framework to exploit multiple local and global features of video data for NDVR. While most of the traditional NDVR algorithms use a single local or global feature, we use multiple features to improve the NDVR performance. We show that the combination of multiple local and global features is more capable to characterize the video content and thus yields better performance.
2) We propose a new hashing algorithm MFH which learns hash codes of the training videos and a group of hash functions to generate hash codes for the videos outside the training set. As far as we know, it is the first hashing algorithm on multiple features which is able to preserve the local structural information of each individual feature and also globally consider all the local structures in the optimization.
3) We utilize the group information of the keyframes within the same video in the learning process. Lots of existing works [42], [30] learn the hash codes by using the keyframe individually. However, the keyframes within the same video are usually highly related. Hence, we incorporate this group information into our learning framework to further improve the accuracy.
4) We have constructed a large scale video dataset consisting of 132,647 videos which have 2,570,554 keyframes. This dataset is released to public so that other researchers will be able to use it as a test bed. Experiment results validate the proposed algorithm in both efficiency and accuracy.

The remainder of this paper is organized as follows. In Section II, we briefly discuss the related work. The details of MFH is presented in Section III. Extensive experiment results are given in Section IV. Lastly, we draw a conclusion in Section V. A preliminary version of this paper has appeared in [30]. Here we further extend it by incorporating the video group information in the learning framework, providing complexity analysis of the framework, and performing more extensive experimental study.

II. Related Work

A. NDVR

NDVR and its related applications such as video copy and similarity detection have been actively studied in all kinds of real world applications [3], [14]. During recent years, various approaches using different features and matching algorithms have been proposed for NDVR [26], [28], [40], [48]. The existing literatures on NDVR can be roughly divided into two groups, i.e., global feature based approaches and local feature based approaches respectively.

Many of the existing global feature based NDVR approaches emphasize the rapid identification of NDVs [28], [40]. In these works, videos are represented by compact global features. These approaches perform quite well in handling almost identical videos. For example, in [40], the authors adopt HSV to represent their keyframes and further generate a video signature by cumulating all the keyframes in the video. This representation achieves fast retrieval speed as well as high accuracy in their dataset. However, a limitation is that global feature based approaches usually become less effective in handling video copies with layers of editing cosmetics [41]. Meanwhile, global feature based approaches rely heavily on the selected feature types.

As an alternative to global feature based approaches, local feature based approaches have also attracted much research attention. Features such as color, texture and shape extracted at the keyframe level are further segmented into multiple region units, and are particularly suitable for retrieving NDVs with complex variations. The well-known local feature based approaches include keypoint based local feature detection approach (e.g., SIFT [21]) and some others [2]. However, these local feature based approaches utilize pair-wise comparison which are computationally expensive and are not applicable to large scale NDVR. Some recent works proposed to produce a compact representation from local descriptors [23], [7], which are very efficient, but the compact representations are generated with information loss, which will degrade the retrieval quality. In [41], Wu *et al.* propose to construct a hierarchy structure to take advantage of both local feature and global feature. They first filter out part of the videos according to color histogram, and then employ pairwise comparison among keyframes by interest points matching. This hierarchy method improves the performance of NDVR. However, pairwise comparison among keyframes by local feature matching is impractical for large scale video datasets due to the heavy computation cost. Besides, using the global feature HSV only to filter out a large portion of videos might be inaccurate, because some NDVs may have quite different HSV feature, as shown in Fig. 1(a) and Fig. 1(b). Some recent proposals (e.g., [10], [25]) have also considered designing new features (such as spatial-temporal features) or new similarity measuring methods to be adapted to videos characters. Nonetheless, they are single feature methods. Some (e.g., [24]) has also considered scalability issue. They use keyframes to query within large-scale video archive. However, their main contribution is to investigate different keyframe samplings of the reference video database in order to evaluate the possible trade-off between accuracy and scalability.

B. Multiple Feature Fusion

Given that multimedia data can be represented by multiple features, it becomes an important research topic to properly combine the evidences derived from different features. The key problem lying here is how to identify the similarity or correlation between two observations represented by multiple features. Late fusion strategies and early fusion strategies are the traditional methods for multiple-source fusion. The late fusion algorithms [33], [35], [37] first generate separate results from different features, and then combine these results together by different strategies. The methods do not consider the correlation among features. Besides, as indicated in [29], late fusion is computationally more expensive for training. The early fusion strategies try to combine multiple features at the input stage [29]. For example, the algorithms proposed in [4] project different features into a unified space in which their applications can be performed. However, the individual structural information of each individual feature can not be well preserved and the computation cost is heavy. Some others (e.g., [43]) utilize tensor to incorporate multiple features, but they focus on transductive learning.

C. Hashing

Hashing is an important approach to achieve fast similarity search. Many hashing methods have been proposed. Hashing methods can be generally divided into two main categories: random projection based hashing methods and machine learning based hashing methods. The difference between these two categories lies in the generation of the hash functions.

Random projection based hashing methods use random vectors (with some specific distribution, e.g., standard Gaussian distribution) as the hashing function bases. Locality sensitive hashing (LSH) [5], [1] is undoubtedly one of the most representative random projection based hashing methods. LSH uses a family of locality sensitive hash functions composed of linear projection over random directions in the feature space. The intuition behind is that for at lease one of the hash functions, nearby data points have high probability of being hashed into the same bucket. However, in practice, many hash tables have to be built for a high search accuracy, which consumes a large amount of memory space. To reduce the number of hash tables, multi-probe LSH [20] is proposed. It can obtain the same search quality with much less tables. Tao *et al.* [32] have recently proposed the idea of LSB-forest to further improve the performance. The hash codes are represented as one-dimensional Z-order values and indexed in the B^+-tree. Multiple trees forming an LSB-forest can be built to improve the search quality. However, since the hash functions in all these hashing methods are randomly generated and data independent, they are not very effective. To achieve an acceptable accuracy, it usually needs long hash codes in each hash table. However, with the increase of the code length, the collision probability decreases very fast, then many tables are needed to get a good recall.

On the other hand, machine learning based hashing methods utilize machine learning to improve the hashing functions' quality by learning a group of hash functions. These methods can be further divided into three subcategories: supervised hashing [12], [25], [34], semi-supervised hashing [36] and unsupervised hashing [39], [47]. Supervised hashing utilizes label information. In [12], the authors combine LSH with a learned Mahalanobis metric to reflect semantic indexing, and the Mahalanobis metric is learned in an image dataset using supervised learning. In [25], the authors propose Boosting Similarity Sensitive Coding (BoostSSC) to learn a series of weighted hashing functions from labeled data. A deep neural network stacked with Restricted Boltzmann Machines (RBMs) has been recently applied to learn compact binary codes from high dimensional inputs [34], which has shown superior performance over BoostSSC. One problem of the supervised hashing stems from limited or noisy training data. The performance of these methods degrades with less training data due to the over fitting problem. Semi-supervised hashing makes use of both labeled and unlabeled data for training. Wang *et al.* [36] propose a semi-supervised hashing method that is formulated as minimizing empirical error on the labeled data while maximizing variance and independence of hash codes over both the labeled and unlabeled data. Unsupervised hashing uses just unlabeled data, such as spectral hashing [39], self-taught hashing [47], and so on. Spectral hashing utilizes spectral graph partitioning in the learning phase to get the hash codes of training data, which is similar to self-taught hashing. But to calculate the binary codes for a new data point, spectral hashing assumes that the data are uniformly distributed in a hype-rectangle, which is very restrictive, and self-taught hashing has to learn new classification models based on the result of learning phase. None of them takes consideration of multiple features. In this paper, we propose the Multiple Feature Hashing (MFH) algorithm to learn a group of hash functions which can be utilized to map the keyframes represented by multiple features into the Hamming space.

III. Multiple Feature Hashing

The proposed framework based on multiple feature hashing (MFH) for NDVR comprises of two phases. In the first phase which is offline, we use the proposed MFH algorithm to learn a series of s hash functions $\{h_1(\cdot),\ldots,h_s(\cdot)\}$, each of which generates one bit hash code for a keyframe according to the given multiple features. Each keyframe has s bits. Using the derived hash functions $\{h_1(\cdot),\ldots,h_s(\cdot)\}$, each keyframe for a dataset video can be represented by the generated s-sized hash codes in linear time. In the second phase which is online, the query video's keyframes are also represented by s-sized hash codes mapped from the s hash functions. NDVR can be efficiently achieved where only efficient XOR operation and bit count operation on the hash codes are performed to compute the similarity between two videos. Our hashing method, like most machine learning based hashing methods, is a binary code representation method, and it utilizes only one table for retrieval. Given the hash codes of a query data point, it does not only search the bucket with identical hash codes, but also search nearby buckets until all the neighbors are found. In this work, we are doing whole video clip near-duplicate detection, without supporting partial near-duplicate detection.

The key research issue is how to learn the hash functions which affect both accuracy and efficiency. In this section, we detail the proposed MFH algorithm.

A. Terms and Notations

Let v be the number of features.[1] Given an integer $g \leq v$, $(x^g)_t \in \mathbb{R}^{d_g*1}$ denotes the g-th feature of t-th training data, where d_g is the dimensionality of the g-th feature. Suppose there are n training keyframes in total. $X^g = [(x^g)_1, (x^g)_2, \ldots, (x^g)_n] \in \mathbb{R}^{d_g*n}$ is the feature matrix corresponding to the g-th feature of all the training data. $x_t = [(x^1)_t^T, (x^2)_t^T, \ldots, (x^v)_t^T]^T \in \mathbb{R}^{d\times 1}$, where $d = \sum_{g=1}^{v} d_g$, is the vector representation of the t-th training keyframe using all of the features and $X = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{d*n}$. MFH aims to learn a series of hash functions $\{h_1(\cdot), \ldots, h_s(\cdot)\}$, where s is the number of hash functions, i.e., the length of the hash code. Each of the s hash functions generates one bit hash code of the input keyframe. We further denote $Y = [(y_1)^T, \ldots, (y_n)^T]^T \in^{n\times s}$ as the hash codes of the training keyframes corresponding to all features and $Y^g = [(y_1^g)^T, \ldots, (y_n^g)^T]^T \in^{n\times s}$ as the hash codes of the training keyframes derived from the g-th feature.

B. The Objective Function of MFH

As demonstrated in [16], [22], [38], [39], [45], [47], it is beneficial to exploit the local structure of the training data to infer an accurate and compact representation, especially for multimedia data. While in [39], [47], the local structural information has been utilized to learn the hash codes, there are still two major limitations. First, both the algorithms proposed in [39], [47] are two-step training approaches. During the training phase, the hash codes of the training data are first generated according to which the hash functions are then defined. This may incur overfitting when generating the hash codes of training data because only local structure information is considered. Second, the algorithms proposed in [38], [46] can not deal with multiple features. Although a straightforward way is to concatenate the multiple features to produce a high dimensional vector, the structural information of each feature type will be lost.

To exploit the individual structural information of each individual feature, we define v affinity matrices $A_1^g \in \mathbb{R}^{n\times n}$ $(1 \leq g \leq v)$, one for each feature as follows:

$$(A_1^g)_{pq} = \begin{cases} 1, \text{ if } (x^g)_p \in \mathcal{N}_k((x^g)_q) \text{ or } (x^g)_q \in \mathcal{N}_k((x^g)_p) \\ 0, \text{ else} \end{cases} \tag{1}$$

where $\mathcal{N}_k(\cdot)$ is the k-nearest-neighbor set and $1 \leq (p,q) \leq n$.

To learn the hash codes $(y^g)_t$ from the g-th feature, a reasonable criterion is that similar items in the original space should have similar binary codes, which can be formulated as the following minimization problem:

$$\min_{Y^g} \sum_{p,q=1}^{n} (A_1^g)_{pq} \left\|(y^g)_p - (y^g)_q\right\|^2. \tag{2}$$

In our preliminary work [30], the keyframes are considered individually to learn the hash codes. However, a video may contain several keyframes, and the keyframes within the same video are usually highly related.

To exploit the group information of keyframes within the same video for each feature, we define v affinity matrices $G^g \in \mathbb{R}^{n\times n}$ $(1 \leq g \leq v)$, one for each feature as follows:

$$(A_2^g)_{pq} = \begin{cases} 1, \text{ if } (x^g)_p \; and \; (x^g)_p \; are \; in \; the \; same \; video \\ 0, \text{ else} \end{cases} \tag{3}$$

Embedded with the video group information, the objective function can be further formulated as the following minimization problem:

$$\min_{Y^g} \sum_{p,q=1}^{n} \left((A_1^g)_{pq} + \lambda (A_2^g)_{pq}\right) \|(y^g)_p - (y^g)_q\|^2 \tag{4}$$

where λ is a parameter to balance the impact of matrix A_1^g and A_2^g. We define $A^g = A_1^g + \lambda A_2^g$, then (4) can be rewritten as:

$$\min_{Y^g} \sum_{p,q=1}^{n} (A^g)_{pq} \|(y^g)_p - (y^g)_q\|^2 \tag{5}$$

After we get the hash codes on each feature type, we need to learn the overall hash code. Given a training keyframe x_t, we hope the overall hash codes y_t should be consistent with the hash codes $y_t^g|_{g=1}^v$ derived from each feature. We use an example to illustrate why the consistent assumption holds. Suppose we have two NDVs: Fig. 1(a) and Fig. 1(b), and each of them has three types of features, HSV, LBP and SIFT. According to our knowledge, Fig. 1(a) and Fig. 1(b) have similar LBP and SIFT features, but different HSV feature. The hash codes for Fig. 1(a) and Fig. 1(b) learnt from each feature type are y_a^1, y_a^2, y_a^3 and y_b^1, y_b^2, y_b^3. Since LBP and SIFT for both videos are similar, we can assume $y_a^2 = y_b^2$ and $y_a^3 = y_b^3$. The overall hash codes are $y_a = \mu^1 y_a^1 + \mu^2 y_a^2 + \mu^3 y_a^3$ and $y_b = \mu^1 y_b^1 + \mu^2 y_b^2 + \mu^3 y_b^3$, where μ^1, μ^2, μ^3 are the weights for each feature type. If HSV feature performs unsatisfactorily, its corresponding weight μ^1 will be adjusted to a small value, leading to that $y_a \approx y_b$. Therefore, we have the following minimization problem:

$$\begin{aligned} \min_{Y^g, Y} \quad & \sum_{g=1}^{v} \sum_{p,q=1}^{n} \mu^g (A^g)_{pq} \|(y^g)_p - (y^g)_q\|^2 \\ & + \gamma \sum_{g=1}^{v} \sum_{t=1}^{n} \mu^g \|y_t - (y^g)_t\|^2 \end{aligned} \tag{6}$$

where γ is the parameter to balance the two parts, and μ^g is a parameter to balance each feature type. It is easy to see from (6) that the individual manifold structure of each feature type is preserved (the first term) and all the manifold structures are also globally considered in the optimization (the second term).

[1]In our system, each keyframe is represented by the local feature LBP and the global feature HSV color histogram. Therefore, $v = 2$ here. Yet, MFH is able to deal with more feature types when $v > 2$.

Recall that we intend to learn the hash codes of the training keyframes and a series of hash functions $\{h_1(\cdot),\dots,h_s(\cdot)\}$ in a joint framework. We propose to simultaneously learn the hash codes of the training keyframes and the hash functions by minimizing the empirical error of the hash functions *w.r.t.* the learnt hash codes Y. The proposed final objective function of MFH is given by:

$$\begin{aligned}\min_{Y,Y^g,W,b}\quad & \sum_{g=1}^{v}\sum_{p,q=1}^{n}\mu^g(A^g)_{pq}\,\|(y^g)_p-(y^g)_q\|^2\\ & +\gamma\sum_{g=1}^{v}\sum_{t=1}^{n}\mu^g\|y_t-(y^g)_t\|^2\\ & +\alpha\sum_{l=1}^{s}\left(\sum_{t=1}^{n}\|h_l(x_t)-y_{tl}\|^2+\beta\Omega(h_l)\right)\\ \text{s.t.}\quad & \begin{cases} y_t\in\{-1,1\}^s,(y^g)_t\in\{-1,1\}^s\\ Y^TY=I\end{cases}\end{aligned} \tag{7}$$

where $\Omega(h_l)$ is a regularization function over h_l, α and β are the parameters, and y_{tl} is l-th bit hash code for x_t. In (7), $y_t\in\{-1,1\}^s$ and $(y^g)_t\in\{-1,1\}^s$ enforce the hash codes of y_t and $(y^g)_t$ to be binary codes. The constraint $Y^TY=I$ is imposed to avoid the trivial solution. For the simplicity of implementation for large scale datasets, we adopt the linear transformation as the hash function. More specifically, given a keyframe represented by its feature x_t, the l-th ($1\le l\le s$) hash function h_l is defined as:

$$h_l(x_t)=w_l^Tx_t+b_l \tag{8}$$

where $w_l\in\mathbb{R}^{d\times 1}$ is the transformation matrix and $b_l\in\mathbb{R}$ is the bias term.

Let us define $W=[w_1,\mathrm{w}_2,\dots,w_{\mathrm{s}}]\in\mathbb{R}^{d\times\mathrm{s}}$, $b=[b_1,b_2,\dots,b_{\mathrm{s}}]\in\mathbb{R}^{1\times\mathrm{s}}$ and $\mathbf{1}\in\mathbb{R}^{n\times 1}$ is a column vector with all ones. Then we rewrite our objective function as follows:

$$\begin{aligned}\min_{Y,Y^g,W,b}\quad & \sum_{g=1}^{v}\sum_{p,q=1}^{n}\mu^g(A^g)_{pq}\,\|(y^g)_p-(y^g)_q\|^2\\ & +\gamma\sum_{g=1}^{v}\sum_{t=1}^{n}\mu^g\|y_t-(y^g)_t\|^2\\ & +\alpha\left(\|X^TW+\mathbf{1}b-Y\|_F^2+\beta\|W\|_F^2\right)\\ \text{s.t.}\quad & \begin{cases} y_t\in\{-1,1\}^s,(y^g)_t\in\{-1,1\}^s\\ Y^TY=I\end{cases}\end{aligned} \tag{9}$$

In (9), the term $\|X^TW+\mathbf{1}b-Y\|_F^2$ learns the hash functions to generate hash codes for the data. Meanwhile, it can be also interpreted as a regularizer in which global information is enclosed. Compared with the hashing algorithms proposed in [39], [47] where only local information is considered, it may eliminate the problem of overfitting, making the learning more robust. The objective function shown in (9) is equivalent to the balanced graph partitioning and it is an NP-hard problem. Following [39], we relax the constraints $y_t\in\{-1,1\}^s, y_t^g\in\{-1,1\}^s$ to make the problem computationally tractable. The relaxed objective function is given by:

$$\begin{aligned}\min_{Y,Y^g,W,b}\quad & \sum_{g=1}^{v}\sum_{p,q=1}^{n}\mu^g(A^g)_{pq}\,\|(y^g)_p-(y^g)_q\|^2\\ & +\gamma\sum_{g=1}^{v}\sum_{t=1}^{n}\mu^g\|y_t-(y^g)_t\|^2\\ & +\alpha\left(\left\|X^TW+\mathbf{1}b-Y\right\|_F^2+\beta\left\|W\right\|_F^2\right)\\ \text{s.t.}\ & Y^TY=I\end{aligned} \tag{10}$$

C. Solution of MFH

In this subsection, we detail the approach to optimize the objective function of MFH. By setting the derivative of (10) *w.r.t.* b to zero, we get:

$$\begin{aligned}&\mathbf{1}^T(X^TW+\mathbf{1}b-Y)=0\\ &\Rightarrow b=\frac{1}{n}(\mathbf{1}^TY-\mathbf{1}^TX^TW)\end{aligned} \tag{11}$$

By setting the derivative of (10) *w.r.t.* W to zero, we get:

$$X(X^TW+\mathbf{1}b-Y)+\beta W=0. \tag{12}$$

Substituting b in (11) by (13), we have

$$\begin{aligned}&XX^TW+X\mathbf{1}\left(\frac{1}{n}(\mathbf{1}^TY-\mathbf{1}^TX^TW)\right)-XY\\ &+\beta W=0\rightarrow W=(XL_cX^T+\beta I)^{-1}XL_cY,\end{aligned} \tag{13}$$

where $L_c=I-\frac{1}{n}\mathbf{1}\mathbf{1}^T$ is the centering matrix.

Note that $\mathrm{L}_c=(L_c)^T=\mathrm{L}_c(L_c)^T$. Replacing W and b by (11) and (13) respectively, we have

$$\begin{aligned}&X^TW+\mathbf{1}b-Y\\ &=X^TW+\mathbf{1}\frac{1}{n}(\mathbf{1}^TY-\mathbf{1}^TX^TW)-Y\\ &=L_cX^TW-L_cY\\ &=L_cX^T(XL_cX^T+\beta I)^{-1}XL_cY-L_cY\end{aligned} \tag{14}$$

Let $M=(XL_cX^T+\beta I)^{-1}$. $\|X^TW+\mathbf{1}b-Y\|_F^2+\beta\|W\|_F^2$ can be rewritten as:

$$\begin{aligned}&\|X^TW+\mathbf{1}b-Y\|_F^2+\beta\|W\|_F^2\\ &=tr(Y^T(L_cX^TMXL_c-L_c)^T(L_cX^TMXL_c-L_c)Y)\\ &+\beta tr(W^TW)\\ &=tr(Y^T(L_cX^TMXL_cL_cX^TMXL_c-2L_cX^TMXL_c\\ &+L_c)Y)+\beta tr(Y^TL_cX^TMMXL_cY)\\ &=tr(Y^T(L_cX^TMM^{-1}MXL_c-2L_cX^TMXL_c+L_c)Y)\\ &=tr(Y^T(L_c-L_cX^TMXL_c)Y)\\ &=tr(Y^TBY)\end{aligned}$$

where B is defined as:

$$B=(L_c-L_cX^T(XL_cX^T+\beta I)^{-1}XL_c) \tag{15}$$

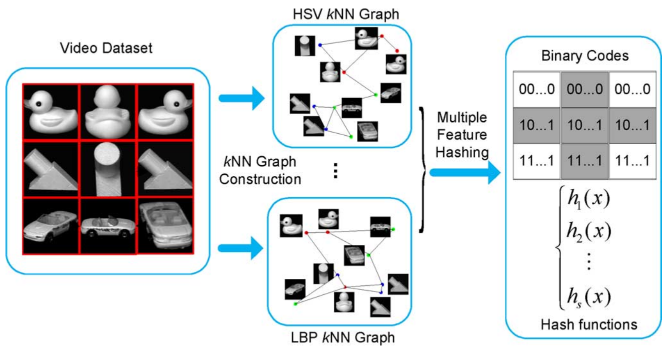


Fig. 1. Multi-Feature Hashing.

Meanwhile, we have:

$$\sum_{p,q=1}^{n} (A^g)_{pq} \|(y^g)_p - (y^g)_q\|_F^2 = tr((Y^g)^T L^g Y^g) \quad (16)$$

where $L^g = N^g - A^g$ is the Laplacian matrix of the g-th feature and N^g is the diagonal matrix with its diagonal element $N^g_{ii} = \sum_j A^g_{ij}$. Note that $\sum_{g=1}^{v} \sum_{t=1}^{n} \mu^g \|y_t - (y^g)_t\|_F^2$ can also be written as $\sum_{g=1}^{v} \mu^g \|Y - Y^g\|_F^2$. Then the objective function shown in (10) becomes:

$$\begin{aligned} \min_{Y,Y^g} \quad & \sum_{g=1}^{v} \left(tr((Y^g)^T \mu^g L^g Y^g) + \gamma \mu^g \|Y - Y^g\|_F^2 \right) \\ & + \alpha tr(Y^T BY) \\ \textbf{s.t.} \quad & Y^T Y = I. \end{aligned} \quad (17)$$

Setting the derivative of (17) *w.r.t.* Y^g to be zero, we have

$$\begin{aligned} & 2L^g Y^g - 2\gamma(Y - Y^g) = 0 \\ & \Rightarrow Y^g = \gamma(L^g + \gamma I)^{-1} Y \end{aligned} \quad (18)$$

Let $C^g = \gamma(L^g + \gamma I)^{-1}$. Note that $C^g = (C^g)^T$. Then we arrive at

$$Y^g = C^g Y \quad (19)$$

Replace Y^g with (19), the objective function in (17) becomes:

$$\min_{Y^T Y = I} tr(Y^T DY) \quad (20)$$

where D is defined as

$$\begin{aligned} D &= \sum_{g=1}^{v} \mu^g (C^g L^g C^g + \gamma(I - C^g)^2) + \alpha B \\ &= \gamma \sum_{g=1}^{v} \mu^g (I - C^g) + \alpha B \end{aligned} \quad (21)$$

The optimization problem shown in (20) can be solved by performing the eigen-value decomposition of D. Y can be obtained by the s eigenvectors of D corresponding to the s smallest eigenvalues.[2] In summary, the training processing of the proposed MFH algorithm is listed in Algorithm 1.

[2]The trivial solution corresponding to the eigenvalue of zero is removed

Algorithm 1 The MFH algorithm.

```
for $g = 1$ to v do
    for $p = 1$ to n do
        for$q = 1$ to n do
            Compute $L^g_{p,q}$ according to 16;
        end for
    end for
end for
Compute $B$ according to (15);
Compute $D$ according to (21);
Compute $Y$ by performing eigenvalue decomposition on $D$.
Output $W$ and $b$ according to (13) and (11).
```

After training, we can get the overall hash code for the training data, and also the hash functions W and b simultaneously. To get the hash codes of the new data points, we concatenate feature vectors into one vector, and map this vector into the Hamming space using the learned hash functions. For example, given a keyframe with concatenated feature x_t, we first compute its relaxed hash code y_t in continuous domain by $y_t = W^T x_t + b$. Then, we binarize it by comparing each dimension of y_t with its median of all dimensions. If it is greater than the median, we set it as 1. Otherwise, we set it as -1. To represent a video clip, Wu *et al.* proposed to averaging the visual feature of all its keyframes as the representation the whole video clip [40]. Following Wu's way, we can also generate the hash codes for a video clip by averaging the relaxed codes of all its keyframes and then binarize them. As a result, each video clip is represented by a single s-bit hash code. In this work, we use the above video representation to avoid pair-wise keyframe comparisons. The number of common bits along all dimensions shared by two videos is approximated as the video similarity.

For better understanding of MFH, we give an illustration in Fig. 1. Given a set of videos, to explore the individual manifold structure, we first construct kNN graph on each individual feature space. Then we learn the hash codes of each video under the criterion that similar items in the original space should have similar hash codes by solving the objective function (17) which preserves the local structure information of each individual feature and also globally considers the local structures among all the features. Meanwhile, a series of hash functions are simultaneously learned.

D. Complexity Analysis

In the previous subsections, we have proposed a multi-feature hashing algorithm MFH for NDVR. During the training phase, to compute the matrix A^g_1, we need to perform kNN search for each feature type, whose time complexity is $O(k \times n^2)$. Since we have v feature types, the time complexity is $O(v \times k \times n^2)$. Because $v \ll n$ and $k \ll n$, the time complexity for computing A^g_1 is $O(n^2)$. To compute the matrix A^g_2, we need to search the n training data points for each data point, whose time complexity is $O(n^2)$. Also, we need to perform an eigenvalue decomposition to get Y, whose time complexity is n^3. Then, we need to solve a linear system when computing W and b, whose time complexity is $O(n^2)$ approximately. Therefore, the time complexity for the training of MFH is $O(n^3)$ approximately. Once the hash functions, i.e., W and b, are obtained, we only need to compute the hash code to map the whole database into the Hamming space with a linear time complexity. All of the above steps are done offline.

Then the online process is to find the near-duplicate videos for a query. The query cost mainly comes from searching the hash table and accessing the candidates. Typically the code length l is dependent to the data size and the number of features. Hash code comparison in the hash table is done by the quick XOR operation. Since our search algorithm only checks very few buckets, the number of candidates is reasonably small. Given that the hash table is much smaller than the original dataset and the XOR operation is far more efficient than the full distance computation, the query cost in RHLM is expected to be highly sub-linear to the data size. In the experiments, we will see how the query cost changes with respect to the code length.

E. Differences From Existing Hashing Methods

Next, we briefly discuss the connections and differences between our MFH and some other related works. Many machine learning based hashing methods have been proposed, and there are a group of hashing methods based on graph. Although the motivations of all these graph-based hashing algorithms vary, their objectives are similar, that is, to encode the relationships of training data points in the original spaces into graphs, and then derive the hash codes based on these graphs.

The difference among various graph-based hashing methods lies in two aspects: how to construct this similarity graph and how to learn the hash functions. SpH [39] constructs global similarity graph using Gaussian similarity function, and STH [47] constructs local similarity graph using cosine similarity on the Euclidean space. SHSC [15] and some metric learning methods [13] argue that Euclidean distance may not reflect the semantic similarity, and they learnt a distance metric which can better reflect the semantic similarity. SSH [36] utilizes the label information to construct the similarity graph. If two data points in the original space have the same label, their entry will be 1, and if they belong to different classes, their entry will be -1, otherwise, their entry will be 0. Spherical hashing [8] utilizes spatial information to help constructing similarity graph. To learn the hash functions, STH utilizes SVMs and SSH adopts RBMs. Other methods [18], [46] extend existing methods using kernels.

Compared with most existing hashing methods, firstly, our method is unsupervised. Secondly, in our this work, we put our focus more on how to utilize multiple features to significantly improve the performance of NDVR. To the best of our knowledge, this is the first effort on multiple feature hashing. Thirdly, our method provides the flexibility of adjusting the weights of different features for better performance on different datasets, by incorporating the importance of different features into the objective function. Fourthly, the tradeoff between embedding individual features and globally aligning all features is also considered in our method.

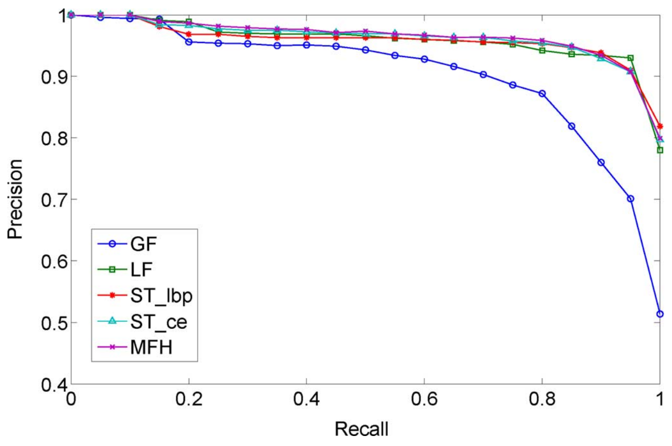


Fig. 2. Comparison of precision and recall on CC_WEB_VIDEO.

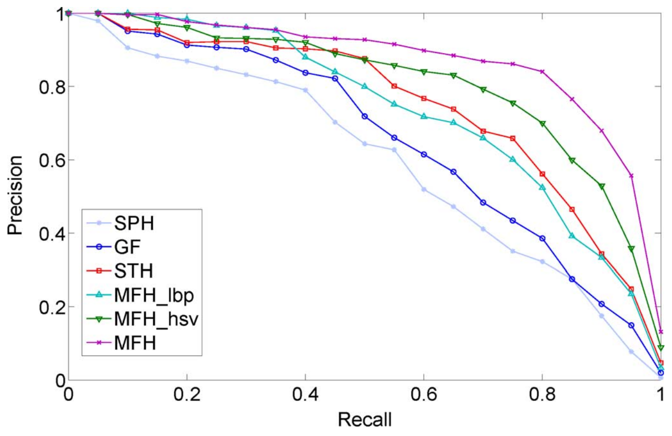


Fig. 3. Comparison of precision and recall on the Combined Dataset.

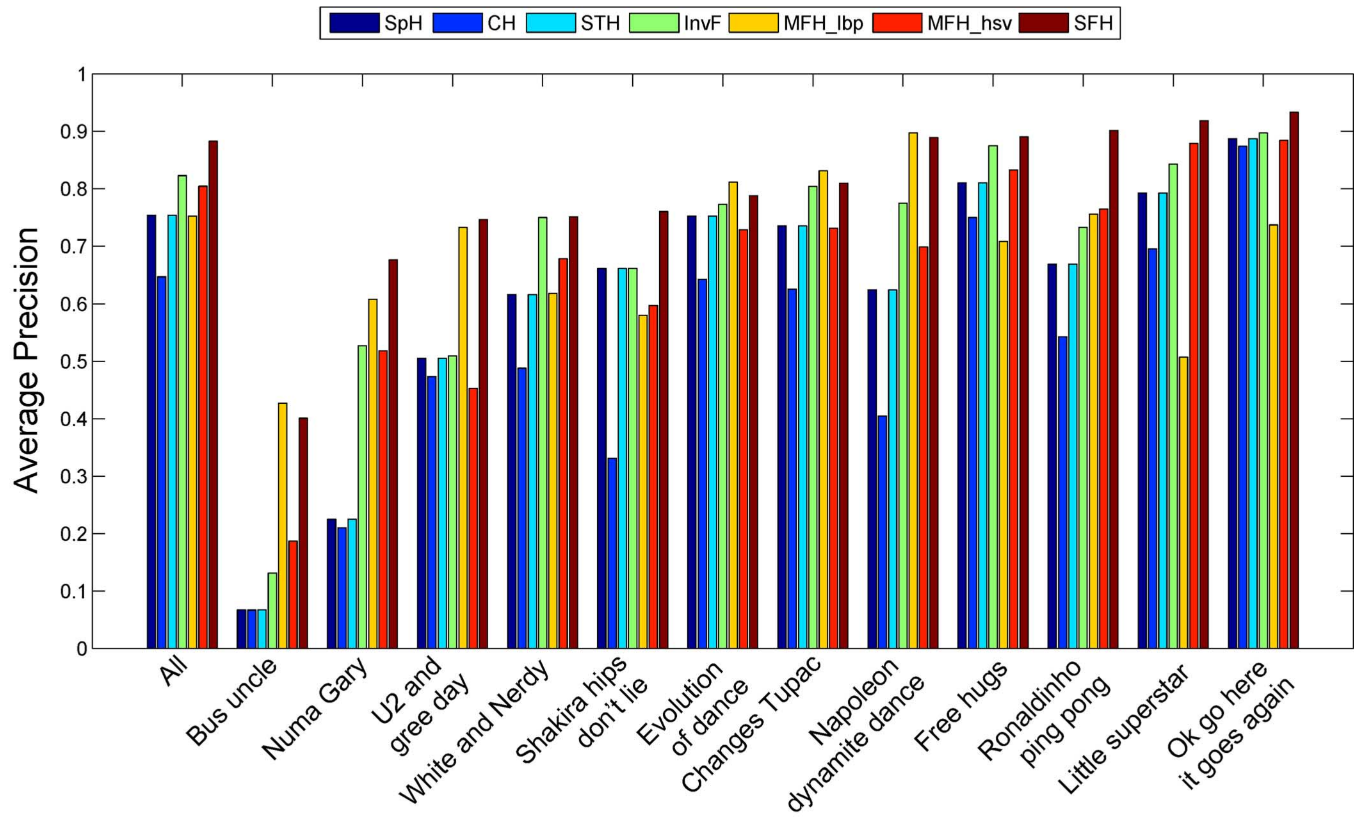
IV. Experiments

A. Experiment Setting

1) Video Datasets: **CC_WEB_VIDEO Dataset**[3] is provided by CityU of Hong Kong and CMU. It consists of 24 sets of video clips (totally 13,129 video clips) downloaded from video sharing websites including Google Video, Yahoo! Video, and YouTube using specific keywords. For each set of videos, the most popular video is selected as the query video, and all the other videos within the set are manually labeled to get the ground truth.

Combined Dataset is a larger video dataset created by ourselves by adding CC_WEB_VIDEO to our video dataset downloaded from YouTube. We choose the most popular 400 queries to query the videos from YouTube. Those queries are selected from Google Zeitgeist. Each year, Google examines billions of queries that people around the world have typed into Google search to discover the Zeitgeist saved in Google Zeitgeist Archives. We collect Google Zeitgeist Archives from 2004 to 2009, and choose the most popular 400 queries to search YouTube. The downloaded number of videos for each query is up to 1000. We crawled more than 150 K YouTube videos from July 2010 to September 2010. After filtering out

[3]CC_WEB_VIDEO: Near-Duplicate Web Video Dataset. http://vireo.cs.cityu.edu.hk/webvideo



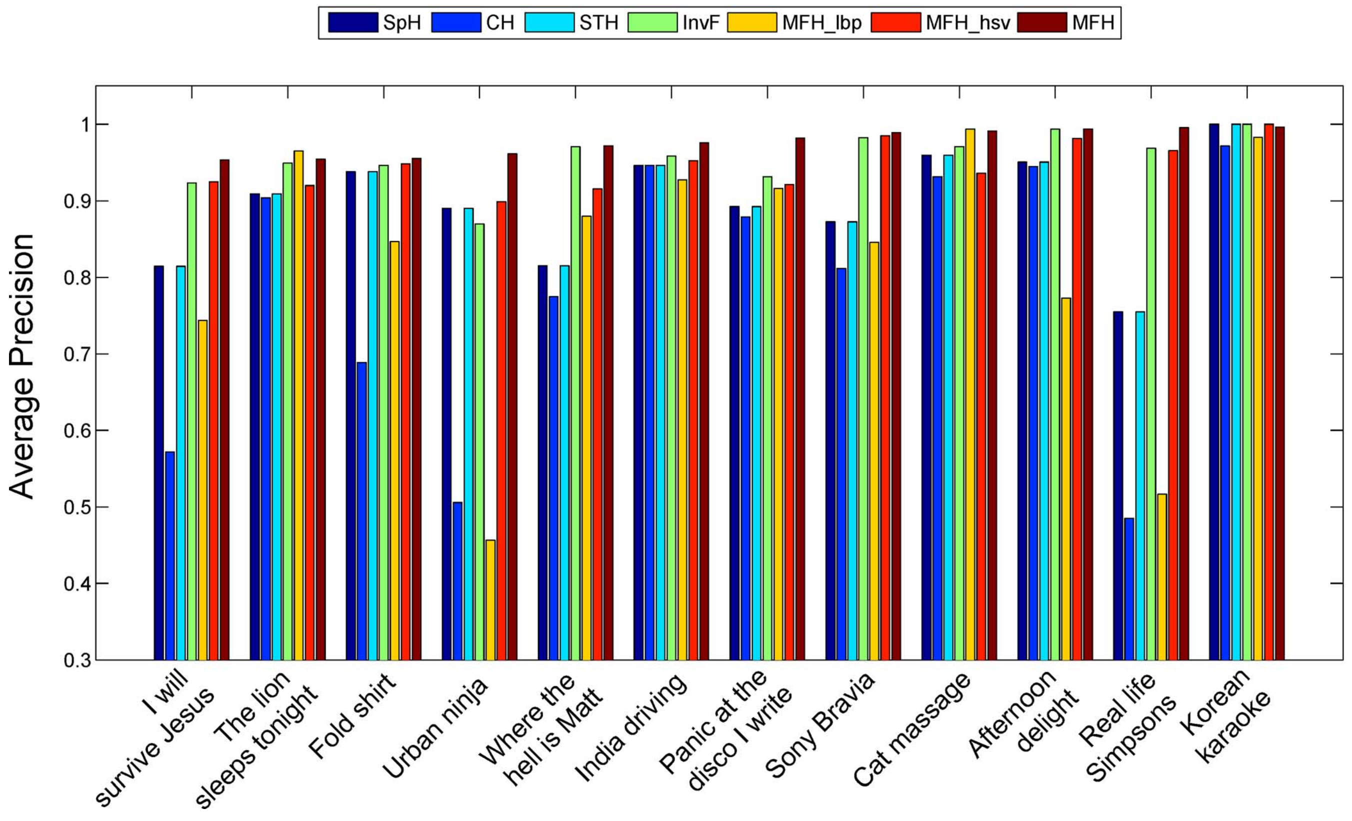


Fig. 4. Comparison of AP corresponding to each query on the Combined Dataset. 'All' indicates the MAP of all the 24 queries.

the videos whose sizes are greater than 10 M, the Combined Dataset contains 132,647 videos in total, which is more than twice as big as the one used in [26]. To our best knowledge, this is the biggest Web video dataset for experimental purpose. We further extract 2,570,554 keyframes from these videos. This dataset has been released.

2) Visual Features: On the CC_WEB_VIDEO dataset, shot-based sampling method is used to extract key-frames and there are 398,078 keyframes extracted in total. These keyframes are provided in the dataset. We further extract LBP feature on the keyframes as a local feature, which will be used together with the existing global feature HSV provided by dataset.

On the Combined Dataset, we firstly detect the shots from which the keyframes are extracted. Then we extracted two features for each keyframe, which are HSV and LBP, to be used as a global feature and a local feature respectively. HSV is a global color histogram with 162 dimensions and LBP is a local content feature with 256 dimensions.

3) Algorithms for Comparison: To evaluate NDVR accuracy and efficiency, we present an extensive comparison of the proposed method with a set of existing NDVR methods, which are briefly described as follows. **Global feature based method**: We use the global color histogram method as a baseline method.

Local feature based method: Wu *et al.* [40] proposed a hierarchical method to accelerate the NDVR.

Spatiotemporal feature based method: Shang *et al.* [26] introduced a compact spatiotemporal feature to represent the videos for efficient NDVR.

Self-taught hashing: Zhang *et al.* [47] designed compact binary codes for documents and here we extend it for video representation.

Spectral hashing: Weiss *et al.* [39] generate compact binary codes of training data points and learn hash functions for new data points.

We also implemented single feature MFH, which is a special case of our MFH, to prove the significance of using multiple features. For simplicity, we let MFH_hsv and MFH_lbp represent the single feature cpase of our MFH. Similarly, let ST_lbp and ST_ce represent the two spatiotemporal feature based methods used in [26], and GF, LF, STH and SPH represent global feature based method, local feature based method, self-taught hashing and spectral hashing respectively.

4) System Environment and Evaluation Metrics: All of the experiments are implemented on a computer which has Intel(R) Core(TM) i7 2.93 GHz 8 processors, 16 GB RAM and 64-bit Windows 7 Enterprise N operating system. Following the work in [26], we also adopt the metric MAP (Mean Average Precision) to evaluate the performance. Among evaluation measures, MAP has been shown to have especially good discrimination and stability. We further use precision-recall curve to evaluate the performance on both datasets. Time efficiency is calculated on Matlab R2010a for all the comparison methods.

B. Results on CC_Web_Video

We first test different methods on the CC_WEB_VIDEO dataset. The same 24 queries as in [40], [26] are used. The MAP results and precision-recall curves of different methods can be seen in Table I and Fig. 2 respectively.

From Table I, we can see that the proposed MFH achieves the best results in terms of MAP, although the improvement is small. Fig. 2 shows that GF performs poorly compared with other four methods which have quite similar performance. This is probably because the dataset is not big enough and thus less noises can be introduced for most methods. The performance of different methods cannot be fully explored. Next, we focus on the large dataset.

TABLE I
COMPARISON OF MAP ON CC_WEB_VIDEO

Methods	GF	LF	ST_lbp	ST_ce	MFH
MAP	0.892	0.952	0.953	0.950	0.958

TABLE II
COMPARISON OF MAP AND TIME ON COMBINED DATASET

Methods	MAP	Time(s)
SPH	0.6015	0.4962
GF	0.6397	1.3921
STH	0.7516	0.6231
MFH_lbp	0.7613	0.6336
MFH_hsv	0.8051	0.4613
MFH	0.8825	0.5529

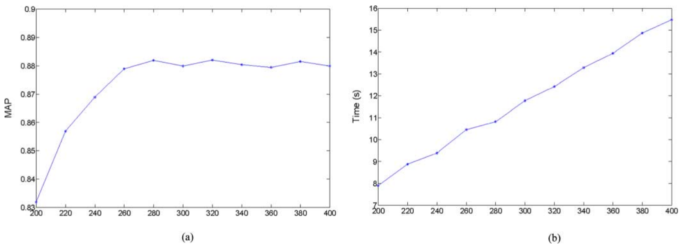


Fig. 5. The performance variations of different code lengths. (a) The MAP of different code length, (b) The time cost of different code length.

C. Results on Combined Dataset

To further test the accuracy and scalability of MFH, we test MFH on the Combined Dataset and compare it with existing methods. More specifically, we use the same 24 queries and groundtruth as in [40], [26] to search the whole Combined Dataset. We randomly select 1,000 videos as the training data set and repeat the experiment for three times. Since different videos contain different number of keyframes, the size of the training keyframes may differ each time. There are several parameters need to be tuned for some of the algorithms. For each parameter we have a set of candidate values, and then we iteratively combine the parameters to form various parameter settings. For each parameter combination we run the same experiment to evaluate its impact on the performance. Finally, the parameter setting which results in the best performance is selected for each method. For the parameters in MFH, as reported in [44] and verified in [30], the performance is not sensitive to the local regularization parameter β. We didn't tune this parameter and fix it as 1. For the other parameter in MFH including γ,α, λ, we tune them from $10^{-6}, 10^{-3}, 10^{0}, 10^{3}, 10^{6}$ and tune the length of the binary code from 200 to 400. On this large Combined Dataset, we cannot compare with local feature based method LF proposed in [40] because this method utilizes interest points matching to measure the similarity of two keyframes. It is impractical in such a large dataset with 2,570,554 keyframes. We are also unable to extract the spatiotemporal features ST_lbp and ST_ce, because it takes extremely long time for our computer to extract the features from 132,647 videos. As a result, only methods including GF, STH, SPH, MFH_hsv, MFH_lbp, and MFH are compared. The results on MAP, search time are shown in Table II. To have a clear visualization, averaged precision-recall curves are depicted in Fig. 3. We also report the Average Precision (AP) of each query in Fig. 4. From Table II and Fig. 3, we have the following observations.

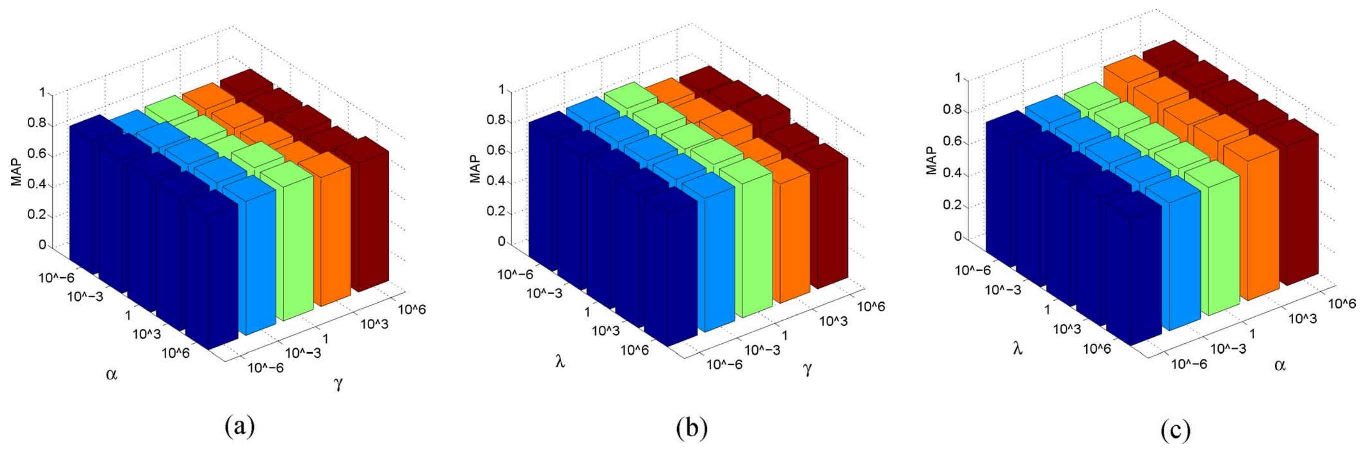


Fig. 6. The MAP variations of different parameter settings. (a) λ is fixed as 10^3. (b) α is fixed as 10^3. (c) γ is fixed as 10^0.

- MFH achieves the highest MAP and outperforms existing methods by large margins. Note that the MAP for MFH is 88.25% while that for STH is 75.16%.
- SPH, STH and MFH all construct binary codes for the videos and perform the search using in the Hamming space. Even in the Matlab environment, they can search the large dataset within 1 second. However, SPH's performance is really unsatisfactory probably because the video dataset is not in accordance with the assumption that all the data points are uniformly distributed in a hyper-rectangle in the high-dimensional space. This also shows that MFH is more capable of preserving local data distribution information.
- GF performs search using the Euclidean distance and its performance is not good either. Its accuracy is much worse than that of MFH. Furthermore, its search time is about 2–3 times longer than hashing methods.
- MFH outperforms MFH_lbp and MFH_hsv in general, which demonstrates the effectiveness of combining multiple features in NDVR.

From Fig. 4, we also observe that MFH_lbp is worse than MFH_hsv for the overall MAP. However, for some queries, such as 'Bus uncle' and 'U2 and green day', MFH_lbp achieves much better accuracy than MFH_hsv. This shows that different types of features can character some part of the visual content, and they can only perform well in some certain situations.

D. Parameter Sensitivity Study for MFH

In this subsection we test different parameter settings for our MFH to see the performance variation. There are three parameters to be tuned in our proposed model: λ, α and γ, as shown in (10). In our paper, by fixing the code length $s = 320$ and $\mu^1 = \mu^2 = 1$, we test $\lambda, \alpha, \gamma = 10^{-6}, 10^{-3}, 10^0, 10^3, 10^6$.

We show the MAP variations on λ, α and γ in Fig. 6. Generally speaking, in order to obtain better performance, α should not be smaller than 10^3, as shown in Fig. 6(c) and Fig. 6(a). Next, we study the importance of each part. When α is fixed as 10^3, as is shown in Fig. 6(b), using embedding part only $(\gamma \ll \alpha, \gamma \ll \lambda)$, i.e., $\gamma = 10^{-3}, 10^{-6}$ and $\lambda \geq 10^0$, can obtain a relatively good performance. On the other hand, if we use agreement part only, $(\lambda \ll \alpha, \lambda \ll \gamma)$, i.e., $\lambda = 10^{-3}, 10^{-6}$ and $\gamma \geq 10^3$, the performance is unsatisfactory. This verifies that embedding part is actually more important than agreement part. If we use the combination of both, i.e., $\gamma \approx \lambda$, the performance is more sensitive to α. When $\gamma = \lambda \ll \alpha$, the results are good. By contrast, if $\gamma = \lambda \geq \alpha$, the performance gets bad. The best performance is achieved when we use the combination of $\lambda = 10^3, \alpha = 10^3$ and $\gamma = 1$. This conclusion is also verified in Fig. 6(a) and Fig. 6(c). However, like any learning methods, the importance of these parameters is data-dependent.

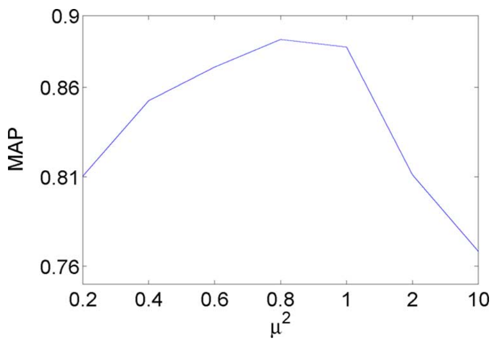


Fig. 7. The effect of feature importance on Combined dataset.

E. Effect of Code Length

By fixing $\lambda = 10^3, \alpha = 10^3$ and $\gamma = 10^0$, we show the performance variations on different code lengths in Fig. 5. When the binary code length is less than 280, the MAP improves dramatically with the increase of binary code length. After that, the MAP keeps steady until reaching peak when $s = 320$. Then, the MAP fluctuates with slight drop until $s = 400$. This indicates that the longest binary code length can not guarantee the best MAP. When $s > 320$, the MAP declines slightly probably because a longer code may contain more noises. An approximately linear increasing trend for query time is illustrated in Fig. 5(b). This is reasonable since it has a linear relationship with the code length.

F. Effect of Feature Importance

In this experiment, we add the parameters of weights on the features to study the effect of feature importance on the results' quality. We fix $\mu^1 = 1$, and test $\mu^2 = 0.2, 0.4, 0.6, 0.8, 1, 2,$ and

10 respectively. From Fig. 7, we can see that the best result occurs when $\mu^2 = 0.8$, which means that HSV is slightly more important than LBP feature on the tested dataset. Generally, combining multiple features will improve the performance. However, different features may play different roles in the overall performance. As they are data dependent, properly tuning the weights of different feature types can help achieve the best results for different datasets.

V. Conclusion and Future Work

In this paper, we have proposed a new multiple feature hashing (MFH) method for large scale NDVR. Compared with the existing algorithms, MFH has two major advantages. First, instead of using single feature, MFH employs a machine learning approach to exploit the local structure of each individual feature and fuse multiple features in a joint framework. Such a method is more stable to well characterize the video content. Second, the MFH method is proposed here to also accelerate the query speed. The extensive experiments verify the superior performance of our method on efficiency and accuracy over existing methods. In the future we intend to incorporate the temporal correlation of the keyframes within the same video to further improve the performance.

References

[1] A. Andoni and P. Indyk, “Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions,” *Commun. ACM*, vol. 51, no. 1, pp. 117–122, 2008.

[2] H. Bay, T. Tuytelaars, and L. V. Gool, “Surf: Speeded up robust features,” in *Proc. ECCV*, 2006, pp. 404–417.

[3] M. Cherubini, R. de Oliveira, and N. Oliver, “Understanding near-duplicate videos: A user-centric approach,” in *Proc. ACM Multimedia*, 2009, pp. 35–44.

[4] B. Cui, A. K. H. Tung, C. Zhang, and Z. Zhao, “Multiple feature fusion for social media applications,” in *Proc. SIGMOD*, 2010, pp. 435–446.

[5] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, “Locality-sensitive hashing scheme based on p-stable distributions,” in *Proc. Symp. Computational Geometry*, 2004, pp. 253–262.

[6] M. Douze, H. Jegou, and C. Schmid, “An image-based approach to video copy detection with spatio-temporal post-filtering,” *IEEE Trans. Multimedia*, vol. 12, no. 4, pp. 257–266, 2010.

[7] M. Douze, H. Jégou, C. Schmid, and P. Pérez, “Compact video description for copy detection with precise temporal alignment,” in *Proc. ECCV*, 2010, pp. 522–535.

[8] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, “Spherical hashing,” in *Proc. CVPR*, 2012, pp. 2957–2964.

[9] Z. Huang, H. T. Shen, J. Liu, and X. Zhou, “Effective data co-reduction for multimedia similarity search,” in *Proc. SIGMOD*, 2011, pp. 1021–1032.

[10] Z. Huang, H. T. Shen, J. Shao, B. Cui, and X. Zhou, “Practical online near-duplicate subsequence detection for continuous video streams,” *IEEE Trans. Multimedia*, vol. 12, no. 5, pp. 386–398, 2010.

[11] Z. Huang, H. T. Shen, J. Shao, X. Zhou, and B. Cui, “Bounded coordinate system indexing for real-time video clip search,” *TOIS*, vol. 27, no. 3, 2009.

[12] P. Jain, B. Kulis, and K. Grauman, “Fast image search for learned metrics,” in *Proc. CVPR*, 2008.

[13] B. Kulis, P. Jain, and K. Grauman, “Fast similarity search for learned metrics,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2143–2157, 2009.

[14] J. Law-To, L. Chen, A. Joly, I. Laptev, O. Buisson, V. Gouet-Brunet, N. Boujemaa, and F. Stentiford, “Video copy detection: A comparative study,” in *Proc. CIVR*, 2007, pp. 371–378.

[15] P. Li, M. Wang, J. Cheng, C. Xu, and H. Lu, “Spectral hashing with semantically consistent graph for image indexing,” *IEEE Trans. Multimedia*, vol. 15, no. 1, pp. 141–152, 2013.

[16] Y.-Y. Lin, T.-L. Liu, and H.-T. Chen, “Semantic manifold learning for image retrieval,” in *Proc. ACM Multimedia*, 2005, pp. 249–258.

[17] J. Liu, Z. Huang, H. Cai, H. T. Shen, C.-W. Ngo, and W. Wang, “An image-based approach to video copy detection with spatio-temporal post-filtering,” *ACM Comput. Surveys*, vol. 45, no. 4, 2013.

[18] J. Liu, Z. Huang, H. T. Shen, and B. Cui, “Correlation-based retrieval for heavily changed near-duplicate videos,” *TOIS*, 2011.

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

[20] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li, “Multi-probe LSH: Efficient indexing for high-dimensional similarity search,” in *Proc. VLDB*, 2007, pp. 950–961.

[21] K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, 2005.

[22] F. Nie, D. Xu, I. W.-H. Tsang, and C. Zhang, “Flexible manifold embedding: A framework for semi-supervised and unsupervised dimension reduction,” *IEEE Trans. Image Process.*, vol. 19, no. 7, pp. 1921–1932, 2010.

[23] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier, “Large-scale image retrieval with compressed fisher vectors,” in *Proc. CVPR*, 2010, pp. 3384–3391.

[24] S. Poullot and S. Satoh, “Detecting screen shot images within large-scale video archive,” in *Proc. ICPR*, 2010, pp. 3203–3207.

[25] G. Shakhnarovich, P. Viola, and T. Darrell, “Fast pose estimation with parameter-sensitive hashing,” in *Proc. ICCV*, 2003, pp. 750–760.

[26] L. Shang, L. Yang, F. Wang, K.-P. Chan, and X.-S. Hua, “Real-time large scale near-duplicate web video retrieval,” in *Proc. ACM Multimedia*, 2010, pp. 531–540.

[27] H. T. Shen, X. Zhou, Z. Huang, and J. Shao, “Statistical summarization of content features for fast near-duplicate video detection,” in *Proc. ACM Multimedia*, 2007, pp. 164–165.

[28] H. T. Shen, X. Zhou, Z. Huang, J. Shao, and X. Zhou, “Uqlips: A real-time near-duplicate video clip detection system,” in *Proc. VLDB*, 2007, pp. 1374–1377.

[29] C. Snoek, M. Worring, and A. W. M. Smeulders, “Early versus late fusion in semantic video analysis,” in *Proc. ACM Multimedia*, 2005, pp. 399–402.

[30] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong, “Multiple feature hashing for real-time large scale near-duplicate video retrieval,” in *Proc. ACM Multimedia*, 2011, pp. 423–432.

[31] H.-K. Tan, X. Wu, C.-W. Ngo, and W.-L. Zhao, “Accelerating near-duplicate video matching by combining visual similarity and alignment distortion,” in *Proc. ACM Multimedia*, 2008, pp. 861–864.

[32] Y. Tao, K. Yi, C. Sheng, and P. Kalnis, “Efficient and accurate nearest neighbor and closest pair search in high-dimensional space,” *ACM TODS*, vol. 35, no. 3, 2010.

[33] S. Tollari and H. Glotin, “Web image retrieval on imageval: evidences on visualness and textualness concept dependency in fusion model,” in *Proc. CIVR*, 2007, pp. 65–72.

[34] A. Torralba, R. Fergus, and Y. Weiss, “Small codes and large image databases for recognition,” in *Proc. CVPR*, 2008.

[35] D. R. Turnbull, L. Barrington, G. Lanckriet, and M. Yazdani, “Combining audio content and social context for semantic music discovery,” in *Proc. SIGIR*, 2009, pp. 387–394.

[36] J. Wang, O. Kumar, and S.-F. Chang, “Semi-supervised hashing for scalable image retrieval,” in *Proc. CVPR*, 2010, pp. 3424–3431.

[37] M. Wang, X.-S. Hua, R. Hong, J. Tang, G.-J. Qi, and Y. Song, “Unified video annotation via multigraph learning,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 5, pp. 733–746, 2009.

[38] M. Wang, X.-S. Hua, J. Tang, and R. Hong, “Beyond distance measurement: Constructing neighborhood similarity for video annotation,” *IEEE Trans. Multimedia*, vol. 11, no. 3, pp. 465–476, 2009.

[39] Y. Weiss, A. Torralba, and R. Fergus, “Spectral hashing,” in *Proc. NIPS*, 2008, pp. 1753–1760.

[40] X. Wu, A. G. Hauptmann, and C.-W. Ngo, “Practical elimination of near-duplicates from web video search,” in *Proc. ACM Multimedia*, 2007, pp. 218–227.

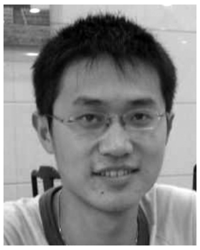
[41] X. Wu, C.-W. Ngo, A. G. Hauptmann, and H.-K. Tan, “Real-time near-duplicate elimination for web video search with content and context,” *IEEE Trans. Multimedia*, vol. 11, no. 2, pp. 196–207, 2009.

[42] X. Wu, W.-L. Zhao, and C.-W. Ngo, “Near-duplicate keyframe retrieval with visual keywords and semantic context,” in *Proc. CIVR*, 2007, pp. 162–169.

[43] Z. Wu, S. Jiang, and Q. Huang, “Near-duplicate video matching with transformation recognition,” in *Proc. ACM Multimedia*, 2009, pp. 549–552.

[44] Y. Yang, D. Xu, F. Nie, J. Luo, and Y. Zhuang, "Ranking with local regression and global alignment for cross media retrieval," in *Proc. ACM Multimedia*, 2009, pp. 175–184.
[45] Y. Yang, Y. Zhuang, F. Wu, and Y. Pan, "Harmonizing hierarchical manifolds for multimedia document semantics understanding and cross-media retrieval," *IEEE Trans. Multimedia*, vol. 10, no. 3, pp. 437–446, 2008.
[46] D. Zhang, J. Wang, D. Cai, and J. Lu, "Extensions to self-taught hashing: Kernelisation and supervision," in *SIGIR*, , vol. 29, no. 38.
[47] D. Zhang, J. Wang, D. Cai, and J. Lu, "Self-taught hashing for fast similarity search," in *Proc. SIGIR*, 2010, pp. 18–25.
[48] X. Zhou, X. Zhou, L. Chen, A. Bouguettaya, N. Xiao, and J. A. Taylor, "An efficient near-duplicate video shot detection method using shot-based interest points," *IEEE Trans. Multimedia*, vol. 11, no. 5, pp. 879–891, 2009.

Jingkuan Song is a Ph.D. student in School of Information Technology and Electrical Engineering, The University of Queensland. His research interests mainly include large-scale multimedia indexing and retrieval.


Yi Yang is a research fellow in School of Computer Science at Carnegie Mellon University. His research interests include machine learning and its applications to multimedia content analysis and computer vision, e.g., multimedia indexing and retrieval, multimedia event detection, multimedia database, etc.

Zi Huang is a Lecturer in School of Information Technology and Electrical Engineering, The University of Queensland. She received her B.Sc. degree from Department of Computer Science, Tsinghua University, China, and her Ph.D. in computer science from School of Information Technology and Electrical Engineering, The University of Queensland. Dr. Huang's research interests include multimedia search, information retrieval, and knowledge discovery.

Heng Tao Shen is a Professor of Computer Science in School of Information Technology and Electrical Engineering, The University of Queensland. He obtained his B.Sc. (with 1st class Honours) and Ph.D. from Department of Computer Science, National University of Singapore in 2000 and 2004 respectively. He then joined the University of Queensland as a Lecturer and became a Professor in 2011. His research interests include Multimedia/Mobile/Web Search and Big Data Management. He is the winner of Chris Wallace Award for outstanding Research Contribution in 2010 from CORE Australasia. He is an Associate Editor of IEEE TKDE, and will serve as a PC Co-Chair for ACM Multimedia 2015.

Jiebo Luo is a Professor in Department of Computer Science in the University of Rochester. His research spans image processing, computer vision, machine learning, data mining, etc. He has been involved in numerous technical conferences, including serving as the program co-chair of ACM Multimedia 2010 and IEEE CVPR 2012. He is the Editor-in-Chief of the *Journal of Multimedia*, and has served on the editorial boards of the IEEE TRANSACTIONS on PAMI, IEEE TRANSACTIONS ON MULTIMEDIA, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, Pattern Recognition, etc. He is a Fellow of the SPIE, IEEE, and IAPR.