SHOT AGGREGATING STRATEGY FOR NEAR-DUPLICATE VIDEO RETRIEVAL

Vignesh Srinivasan, Frederic Lefebvre and Alexey Ozerov

Technicolor
975 avenue des Champs Blancs,
CS 17616, 35576 Cesson Sévigné, France
{vignesh.srinivasan, frederic.lefebvre, alexey.ozerov}@technicolor.com

ABSTRACT
In this paper, we propose a new strategy for near-duplicate video retrieval that is based on shot aggregation. We investigate different methods for shot aggregation with the main objective to solve the difficult trade-off between performance, scalability and speed. The proposed short aggregation is based on two steps. The first step consists of keyframes selection. And the second one is the aggregation of the keyframes per shot. The aggregation is performed by applying Fisher vector on the descriptors computed on the selected keyframes. We demonstrate that the scalability and the speed are tackled by a sparse video analysis approach (i.e. extracting only few keyframes) combined with shot aggregation, while the performance is discussed around the choice of the aggregation strategy. The performance is evaluated on the CC_WEB_VIDEO dataset that is designed for the near-duplicate video retrieval assessment and for which some experiments have been conducted by different authors.

1. INTRODUCTION

Over the last decade, several web services offer online-storage for personal video backup, such as Dropbox, OneDrive, or for video sharing such as Youtube, Dailymotion. The problem of efficient storage, retrieval, and copyright infringement has motivated the online storage providers to develop the concept of near-duplicates in their platforms. Near-duplicate video content (NDVC) is often defined as identical or approximately identical videos but different in file formats, encoding parameters, photometric variations, editing operations, lengths, and certain modifications [1] such as variations in camera viewpoint, setting [2] and camera motion [3]. Cherubini et al. [4] in agreement with [3] articulates the definition of near-duplicate video clip from a user's perspective by taking into account the participants activity on video sharing websites. Jiang et al. [5] defines NDVC as the same scenes originally captured from two different cameras. Since there is no unique consensus on the NDVC definition in the literature, particular approaches rely on some datasets, where it is annotated whether two videos are near-duplicate or not. In this work we will follow this data-driven definition.

There are a number of datasets that challenge video fingerprint and give data-driven definitions of near-duplicate videos. The Muscle-VCD dataset [6] and the TRECVID dataset provided by the National Institute of Science and Technology [7] are essentially for video copy detection. The CC_WEB_VIDEO dataset [1] does not consider the large scale but proposes NDVC content from real world rather than artificially created content. UQ_VIDEO [2] is a combined dataset created by injecting more videos to the existing CC_WEB_VIDEO. VCDB is a new dataset [5] for partial copy detection in videos containing about 100,000 videos downloaded from the internet. The evaluation procedure needs to compare the scores obtained by our algorithm with state of the art in the same test collection. CC_WEB_VIDEO dataset benefits to some relevant comparison studies [8], [9] in near duplicate use case. For this reason, we have selected CC_WEB_VIDEO as the reference dataset for our experiments.

To achieve a good trade-off between performance, scalability and speed, most of the video fingerprint algorithms manage the video as a succession of frames. For each selected frame, some attributes or descriptors are extracted in a sparse or dense manner. The quality of attributes affects the accuracy of a near-duplicate video retrieval (NDVR) system while their size and number impact the scalability and the speed of retrieval. Color histograms [1] used as a global video signature are computationally expensive and also fail in case of photometric variations in the videos. An improvement over this issue was proposed [1, 2], which consists in using a hierarchical approach employing local signatures. Shang et al. [9] introduced a video signature method based on a binary spatio-temporal feature. A temporal approach is also elaborated in [10] by finding temporal relations among patterns. To improve scalability and the speed of retrieval, a temporal sparse approach consisting of detecting keyframes is employed [11]. Ordinal relations are extracted only on these keyframes using conditional entropy and local binary pattern methods. However, this method is not robust to spatial editing and hence brings the performance down. Visual descriptors aggregation methods [12–20] became more and more popular because of their robustness to different transformations or editing in the videos. Aggregated descriptors tackle the sparsity of the image descriptor but do not handle the temporal aspect. Note also that selecting “relevant” keyframes is a way to improve the scalability and speed in video description. Uniformly sampled keyframes are used in [9, 16, 18, 20, 21] which leads to an extra storage cost because of redundant information. One keyframe per shot (the center of the shot) is extracted in [1, 13, 15]. Such a method of keyframe extraction leads to a loss of information over the entire shot and hence to a performance drop, especially in the case of long shots. To improve the accuracy, a method consists in increasing the number of features [22] but this leads also to an extra cost storage.

The aim of our proposal is to maintain a high performance while increasing the scalability and speed of retrieval. First, the keyframes are selected using the method described in [23, 24]. Shot boundaries are found in a video and stable keyframes are extracted from inside these shots in a non-uniform manner. The scale-invariant feature transform (SIFT) [25] descriptors are extracted from each of the keyframes. A feature vector is calculated per shot by aggregating
all extracted descriptors in a shot in a single Fisher vector, which is a simplifications of the Fisher kernel [26]. Finally, the retrieved videos are ranked using two different strategies: a voting strategy or a hidden Markov model (HMM)-based strategy that allows exploiting temporal coherence between sequences of shots. The main contribution of this paper is to combine a sparse video analysis approach, i.e., selecting just few keyframes, with different aggregation methods that should lead a good trade-off between performance, scalability and speed. The rest of the paper is organized as follows. Section 2 describes our proposed shot aggregation method in detail. Section 3 is devoted to experiments. Conclusions are drawn in section 4.

2. PROPOSED METHOD

A general scheme of the proposed method is represented in Fig. 2. In the following subsections the method is described step by step.

2.1. Keyframe Selection

The goal of this step is to find the best frames in a video to fingerprint with a reasonable density rate. A shot being a temporal section where the video activity shall be constant, the method we use [23, 24] consists in finding shot boundaries and the best stable frames in each shot. Shot boundaries are the two frames that surround the shot while the best stable frames are the frames with the smallest content variation along the shot. The activity is captured by analysing a perceptual distance between successive frames [24]. This perceptual distance is defined as the Euclidean distance between perceptual hash (a global descriptor) of neighbour frames. Perceptual hash we used is a so-called Radon soft hash algorithm (RASH) [27] that is a simplifications of the Fisher kernel [26]. Finally, the retrieved hash (a global descriptor) of neighbour frames. Perceptual hash we used is a so-called Radon soft hash algorithm (RASH) [27] that is a simplifications of the Fisher kernel [26]. Finally, the retrieved Fisher vector can be derived as a special case of the Fisher kernel [26].

The Fisher vector can be derived as a special case of the Fisher kernel [26]. Consider a $D$-dimensional feature vector $X = [x_1, x_2, \ldots, x_D]^T$ (here we use dense SIFT) extracted from an image and the parameters of the Gaussian mixture model (GMM) to be $\theta = \{\mu_k, \Sigma_k, \pi_k\}_{k=1}^K$, where $\mu_k$, $\Sigma_k$ and $\pi_k$ denote respectively the mean vector, the covariance matrix and the weight of $k$-th component. The GMM associates each feature vector $X$ to a component $k$ in the mixture with a strength given by the posterior probability. Thus, the assignment to a given GMM component is done in a soft manner, which is a fundamental difference between the Fisher vector and the BOVW model that is based on a hard assignment. We here compute the Fisher vector with respect to GMM means $\mu_k$ only [29], and thus the Fisher vector length is

$$L = D \times K.$$  

For example for a SIFT descriptor of size 128 and a GMM with 64 components, we get a Fisher vector of length $128 \times 64 = 8192$.

Unlike the previous works [9, 16, 18, 20, 21], where either only one frame per shot or uniformly sampled frames were considered, our method considers several non-uniformly selected stable frames, aka keyframes, per shot. By not considering uniformly sampled frames, our method eliminates redundant information and this helps in its scalability. Once dense SIFT and Fisher vector are applied on keyframes, the shot aggregation of the local features is performed. We consider two alternative ways for that:

1. Shot level SIFT aggregation ($S_{AGG}$): Dense SIFT features computed for each of the keyframes in a shot are aggregated into a single Fisher vector per shot:

$$FV_{shot} = G\left(\left\{X_{m,n}\right\}_{n=1}^{N_k(m)}\right)^{M_s}_{m=1},$$  

where $X_{m,n}$ is the $n$-th SIFT vector in the $m$-th keyframe, $N_k(m)$ is the total number of SIFT vectors in the $m$-th keyframe, $M_s$ is the total number of keyframes in the shot, and $G(\cdot)$ is the Fisher vector aggregation function.

2. Shot level Fisher averaging ($F_{AV}$): For each keyframe one Fisher vector is computed from the corresponding dense SIFT features. Then, Fisher vectors belonging to a single shot are averaged to obtain a single Fisher vector for this shot:

$$FV_{shot} = \frac{1}{M_s} \sum_{m=1}^{M_s} G\left(\left\{X_{m,n}\right\}_{n=1}^{N_k(m)}\right).$$  

Once the keyframes are selected, they have to be described. SIFT local descriptors aggregated by Fisher vector techniques are dedicated to this task.

2.2. Fisher Vector Encoding

The Fisher vector [19] combines the advantages of methods based on generative statistical models and those of discriminative methods. The Fisher vector is the normalized gradient of the log-likelihood of the data sample with respect to the model parameters, the model being pre-trained from some training data. The Fisher vector can be derived as a special case of the Fisher kernel [26].

Consider a $D$-dimensional feature vector $X = [x_1, x_2, \ldots, x_D]^T$ (here we use dense SIFT) extracted from an image and the parameters of the Gaussian mixture model (GMM) to be $\theta = \{\mu_k, \Sigma_k, \pi_k\}_{k=1}^K$, where $\mu_k$, $\Sigma_k$ and $\pi_k$ denote respectively the mean vector, the covariance matrix and the weight of $k$-th component. The GMM associates each feature vector $X$ to a component $k$ in the mixture with a strength given by the posterior probability. Thus, the assignment to a given GMM component is done in a soft manner, which is a fundamental difference between the Fisher vector and the BOVW model that is based on a hard assignment. We here compute the Fisher vector with respect to GMM means $\mu_k$ only [29], and thus the Fisher vector length is

$$L = D \times K.$$  

For example for a SIFT descriptor of size 128 and a GMM with 64 components, we get a Fisher vector of length $128 \times 64 = 8192$.

Unlike the previous works [9, 16, 18, 20, 21], where either only one frame per shot or uniformly sampled frames were considered, our method considers several non-uniformly selected stable frames, aka keyframes, per shot. By not considering uniformly sampled frames, our method eliminates redundant information and this helps in its scalability. Once dense SIFT and Fisher vector are applied on keyframes, the shot aggregation of the local features is performed. We consider two alternative ways for that:

1. Shot level SIFT aggregation ($S_{AGG}$): Dense SIFT features computed for each of the keyframes in a shot are aggregated into a single Fisher vector per shot:

$$FV_{shot} = G\left(\left\{X_{m,n}\right\}_{n=1}^{N_k(m)}\right)^{M_s}_{m=1},$$  

where $X_{m,n}$ is the $n$-th SIFT vector in the $m$-th keyframe, $N_k(m)$ is the total number of SIFT vectors in the $m$-th keyframe, $M_s$ is the total number of keyframes in the shot, and $G(\cdot)$ is the Fisher vector aggregation function.

2. Shot level Fisher averaging ($F_{AV}$): For each keyframe one Fisher vector is computed from the corresponding dense SIFT features. Then, Fisher vectors belonging to a single shot are averaged to obtain a single Fisher vector for this shot:

$$FV_{shot} = \frac{1}{M_s} \sum_{m=1}^{M_s} G\left(\left\{X_{m,n}\right\}_{n=1}^{N_k(m)}\right).$$  

$^1$In case of image representation this training data may be a set of descriptors extracted from a big set of images.
2.3. Ranking

To rank the retrieved videos, we propose two strategies: the voting strategy and the HMM-based strategy. Both strategies are using a similarity measure that is the Euclidean distance between the descriptors, i.e., Fisher vectors, of the query video and the descriptors of all the videos in the dataset. Let \( D = [d_{ij}]_{i=1}^{I} \times {J}_{j=1}^{J} \), matrix of similarity measures between the query video and a video from the reference dataset, where \( d_{ij} \) is the Euclidean distance between the \( i^{th} \) descriptor of the query and the \( j^{th} \) descriptor of the reference datasets, indices \( i \) and \( j \) enumerate to either the shots or the keyframes depending on the method, and \( I \) and \( J \) are the corresponding total numbers of shots or keyframes.

2.3.1. Majority voting

We rely on the voting strategy introduced in [30]. Given \( D \) the voting similarity between the query video and a video from the reference dataset is computed as

\[
V = \sum_{i=1}^{I} \left( \max_{k=1,\ldots,K} [\tilde{d}_{ik}] - \min_{j=1,\ldots,J} [d_{ij}] \right),
\]

where \( \tilde{D} = [\tilde{d}_{ik}]_{i=1}^{I} \times {K}_{k=1}^{K} \) is the matrix of similarity measures between the query video and all videos from the reference dataset, with index \( k = 1,\ldots,K \) enumerating over all videos the shots or the keyframes depending on the method. The videos can now be ranked according to the votes received.

2.3.2. HMM-based strategy

Voting similarity described in the previous section does not take into account temporal consistency between different keyframes or shots. However, in the near-duplicate content the keyframes or shots keep usually the same order in the query and the reference videos, and at the same time this does not happen in 100%, since some shot swaps, insertions or deletions are possible as well. Here we would like exploiting such a temporal consistency, while having an approach that tolerates temporally non-consist exceptions. For that we are using an HMM-based strategy that allows taking into account temporal consistency via a most likely path decoded within similarity matrix \( D \) that usually relies distances \( d_{ij} \) in a monotone manner (see red path on Fig. 2), while non-monotone behaviour is possible as well. HMM-based strategy was also used to detect copied segments in [31].

This is achieved as follows. Given a similarity matrix \( D \) between a query video and one reference video, we consider an \( I \)-length observation sequence decoded within a \( J \)-state HMM. The similarity measure between these two videos is now computed as the log-likelihood of the most likely sequence of states \( q = \{q_i\}_{i=1}^{I} \) (\( q_i \in \{1,\ldots,J\} \)). In other words

\[
V = \max_q \log p(q),
\]

where

\[
\log p(q) = -c \times \sum_{i=1}^{I} d_{qi} + \sum_{i=2}^{I} \log p(q_i|q_{i-1}) + \log p(q_1),
\]

is a log-likelihood of a sequence \( q \) that is defined by replacing the observation log-likelihood at time \( i \) and state \( j \) by \( c d_{ij} \) with \( c \) being a positive constant (here we use \( c = 0.001 \)), initial probabilities \( p(j) \) are fixed to equal values and transition probabilities \( p(j|j') \) are defined so as to favour short state transitions forward, while not forbidding any other state transitions, e.g., backward or far forward (and example of transition probabilities values can be found on Fig. 2). This allows temporal consistency check, while being tolerant to exceptional cases where this consistency does not hold (e.g., due to video editing).

This maximum log-likelihood (5) can be efficiently computed by dynamic programming or Viterbi algorithm [32] relying on forward and backward propagation. Since we only need the maximum log-likelihood value and do not really need the most likely state sequence \( q \), we are using forward propagation only.

![Fig. 2. HMM-based strategy](image)

3. EXPERIMENTS

Here we evaluate our methods on the CC_WEB_VIDEO dataset, since it is a popular benchmark for the NDVR task and results of state-of-the-art methods for this benchmark are available [8, 9]. In this dataset 24 different queries are issued from YouTube, Google Video, and Yahoo! and the corresponding search results are collected to form a dataset consisting of 12790 videos which is split into 24 subsets based on the queries. We pre-trained a GMM for the Fisher vector computation using a completely different dataset, built with stable frames collected from thirty hours of videos, and for varying number of components: 64, 128, 256. The keyframes are then densely sampled and SIFT features are computed on each patch. Fisher vectors are then computed based on the two shot aggregation strategies, S_AGG and P_AW, described in section 2.2. To evaluate the performance of our methods, we have two additional experimental set-ups, A_KF and C_SH:

- **A_KF**: Fisher vectors for all the \( M \) keyframes from a shot are computed and no shot aggregation is performed.

- **C_SH**: We extract one middle keyframe per shot, similar to [1], but using our shot detection method described in section 2.1. Fisher vectors are computed for all the keyframes.

We also consider two baseline state-of-the-art methods [8, 9]:

- **B_CE**: In [9], keyframes are sampled uniformly every second and a conditional entropy method is applied to all the keyframes.

- **B_CCA**: In [8], a canonical correlation analysis (CCA) is performed between two videos to check for similarity. The CCA is applied on the features extracted from the videos.

For all the methods based on the Fisher vectors, we perform tests with 3 different GMM components: 64, 128, 256. The lower the number of components, the smaller is the Fisher vector size and hence the storage requirements, which leads in turn to a better scalability.
3.1. Results

We evaluated the performances for each of the set-up, and the corresponding precision-recall curves [8, 9] are plotted on Figure 3. The best overall performance of $F_{AV}$ method is obtained with 128 GMM components. $S_{AGG}-128$ performs similar to $F_{AV}-128$, while $A_{KF}-128$ has a slightly higher precision for lower recall values. However, as recall is increased, the precision drops heavily. Since for $C_{SH}-128$ method only the middle keyframes from the shot are considered, the area under the precision-recall curve is relatively smaller than for other methods considered here.

Figure 4 plots the comparison of our methods $F_{AV}-128$ and $F_{AV}-128$-HMM with the baseline state-of-the-art methods $B_{CE}$ and $B_{CCA}$. We can see that our method $F_{AV}-128$ has a performance similar to $B_{CCA}$, while $F_{AV}-128$-HMM has a higher precision than $B_{CCA}$. This is due to the fact that the HMM-based strategy checks the temporal coherence between matched shots or keyframes sequences using a probabilistic model. HMM-based strategy does not improve the recall but reduces the false positive rate. Both $F_{AV}-128$ and $F_{AV}-128$-HMM outperform $B_{CE}$ for lower recall values.

3.2. Storage and Complexity

Our shot aggregation methods require an average 0.210 MB for GMM with 64 components and the storage requirements increase linearly with respect to the GMM size. $F_{AV}-128$ requires a higher storage than $F_{AV}-64$ but performs better compared to it. Since descriptors are computed and stored for many keyframes in each shot, $A_{KF}$ requires the maximum storage space among our methods, thus affecting its scalability. The computational expense is also increased compared to $F_{AV}-128$. Keyframes are sampled uniformly in $B_{CE}$ and hence it requires more storage compared to $F_{AV}-128$. With relatively smaller storage requirements and a less expensive computation, $F_{AV}-128$ provides a similar performance to $B_{CCA}$. $F_{AV}-128$-HMM is computationally more expensive than $F_{AV}-128$ but it improves the performance.

$^2$Precision-recall curves of the baseline methods are taken from the corresponding papers [8, 9].

4. CONCLUSION

This paper introduced several shot aggregation strategies for NDVR. The keyframes from the videos are extracted using a shot boundary detector followed by non-uniform stable frame selection from the shots. Shot aggregation methods are applied to each shot in a video from which one single Fisher vector per shot is computed.

This strategy provides similar or higher performance than the baseline methods that are based on single keyframe extraction from the center of each shot. It also provided better scalability compared to baseline state-of-the-art methods that are based on uniform keyframes sampling. The combination of shot aggregation strategy and an HMM-based strategy for ranking the near-duplicates improved the performance compared to the voting strategy at the expense of a slightly higher computational load. Our best method is robust to temporal and spatial editing, photometric and geometric variations and gives a similar performance compared to the baseline state-of-the-art methods [8, 9] with smaller storage requirements. It also provides a good trade-off between performance, scalability and speed.

REFERENCES


