# Parallel Implementation Strategy for Content Based Video Copy Detection Using a Multi-core Processor

**Kaiyang Liao[1], Fan Zhao[1] and Mingzhu Zhang [2]**
[1] School of Printing and Packaging Engineering, Xi'an University of Technology
Xi'an 710048, China
E-mail: liaokaiyang@xaut.edu.cn
[2] Department of Public Courses, Xi'an Fanyi University
Xi'an 710005, China
*Corresponding author: Kaiyang Liao

*Received May 8, 2014; revised July 29, 2014; accepted September 10, 2014; published October 31, 2014*

## Abstract

Video copy detection methods have emerged in recent years for a variety of applications. However, the lack of efficiency in the usual retrieval systems restricts their use. In this paper, we propose a parallel implementation strategy for content based video copy detection (CBCD) by using a multi-core processor. This strategy can support video copy detection effectively, and the processing time tends to decrease linearly as the number of processors increases. Experiments have shown that our approach is successful in speeding up computation and as well as in keeping the performance.

***Keywords:*** Large-scale indexing, parallel video copy detection, parallel video retrieval, parallel clustering

The authors would like to thank the anonymous reviewers for valuable comments. This work was supported by funds from National Natural Science Foundation of China (NSFC Project No. 61173110 and NSFC Project No. 11272253).

**http://dx.doi.org/10.3837/tiis.2014.10.014**

## 1. Introduction

Video copy detection methods have emerged in recent years for a variety of applications. However, the lack of speed in the usual retrieval systems restricts their use [1, 2]. Furthermore, the advances in both computing and communication technologies have unleashed an unprecedented growth in the amount of videos. This presents a big challenge in video content management and retrieval solutions.

Parallel computers can provide the appropriate setting where to efficiently execute the program of retrieval systems from large-scale databases[3]. Recently, there has been an increasing interest in parallel implementations of retrieval systems. Plaza et al. proposed a commodity cluster-based parallel processing of hyperspectral imagery [4]. Emmanuel et al. presented a parallel approach for content based medical image retrieval system [5]. Liu et al. proposed a hybrid parallel computing framework for video understanding and retrieval [6].

The distributed parallel approaches have greatly raised the retrieval system's efficiency, but these parallel approaches cannot give full play to the effect of parallel because of the speed limit of network communication [7]. For multicore computer systems, Sutter and Larus point out that multicore computer mostly benefits concurrent applications, meaning ones where there is little communication between the cores [8]. We are approaching an era of increasing numbers of cores per chip. However, to the best of our knowledge, there is as yet no good video copy detection system to take the advantage of massive numbers of cores.

In this paper, we propose a parallel implementation strategy for CBCD by using a multi-core processor. In the parallel version of CBCD system, the SPMD (single program multiple data) approach is exploited by the parallel feature extraction, parallel data clustering and parallel searching on different processors.

The remainder of this paper is organized as follows. Section 2 introduces the parallel version of the content based video copy detection (PCBCD) system. Section 3 presents experimental setup. Section 4 describes the experimental results and analysis. Finally, the paper is concluded in Section 5.

## 2. Parallel Implementation of CBCD

The task of Content Based video Copy Detection is to locate segments within a video (query) that are copies or modifications from a large archive of reference videos. If the CBCD system finds a matching video segment, it will return the name of the database video and the time stamp at which the copy is executed. The key challenge in CBCD is how to efficiently and precisely localize the pair of a copy and its original clip in both the query video stream and the reference database despite various video transformations on the copy.

A general overview on the overall process of a CBCD framework is outlined in the following: 1) Frame sampling: to extract key frames from the videos. The aim of frame sampling is data reduction, since it is computationally intractable to index each of the reference frames. 2) Feature extraction from the key frames. 3) Video data clustering using the extracted features: using clustering algorithm to efficiently build “visual words”. 4) Building video index: using the extracted features and “visual words” to build a semantic video index. For each of the indexed frames, every feature is quantized to its closest “visual words”. In order to search efficiently, the entire set of quantized features of the video datasets is stored in a structure

similar to the inverted file used in the text retrieval system or the image search system of [9]. 5) Searching: the video database is searched for the desired videos using the index and some video similarity measures including (BOF-based searching, Weighting and Reranking and Temporal Consistency etc.).

Parallel execution of CBCD algorithms requires splitting up the computation so that each processing unit can perform a part of the CBCD task in parallel with the other processing units of a parallel computer [10]. Thus the results are achieved as fast as possible.

In computing, SPMD is a technique employed to achieve parallelism, and it is the most common style of parallel programming. On a multicore computer, messages can be sent by depositing their contents in a shared memory area. This is often the most efficient way to program shared memory computers with a large number of processors.

Among the main steps of the CBCD system, the steps of feature extraction, data clustering, quantization and searching are the most computationally intensive ones. We analyzed the time spent on these steps and they cost about 98 percent of the total time. Thus, we identified these four steps as the functions where parallelism must be exploited to speed up the performance of the CBCD system. **Fig. 1** illustrates the high level block diagram of PCBCD.

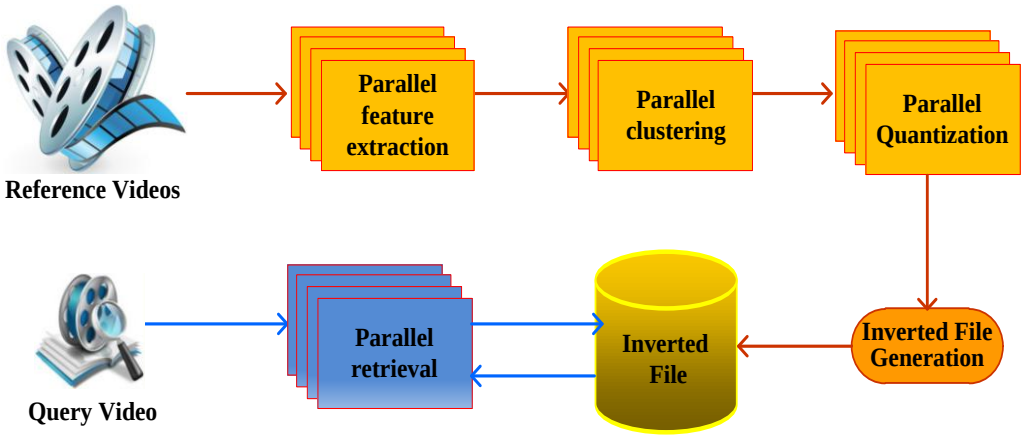



**Fig. 1.** The proposed PCBCD Framework

## 2.1 Parallel Implementation of Feature Extraction

The parallel version of the feature extraction function can concurrently execute the program from two aspects. The first one is video level parallel, and the other is frame level parallel. The video level parallel can be used in off-line processing and can also be used in batch searching for a large quantity of query videos. Frame level parallel can be used in a single video for rapid processing. The parallel version of this function first maps the problem that manipulates the n videos/frames into p instances of a program so that each instance manipulates n/p videos/frames. Then, each processor executes a loop where it extracts the SIFT features[11, 12] from each sampled frame of a video. Because the tasks are independent of each other, the processors do not need additional communication. This strategy is described in **Fig. 2**.

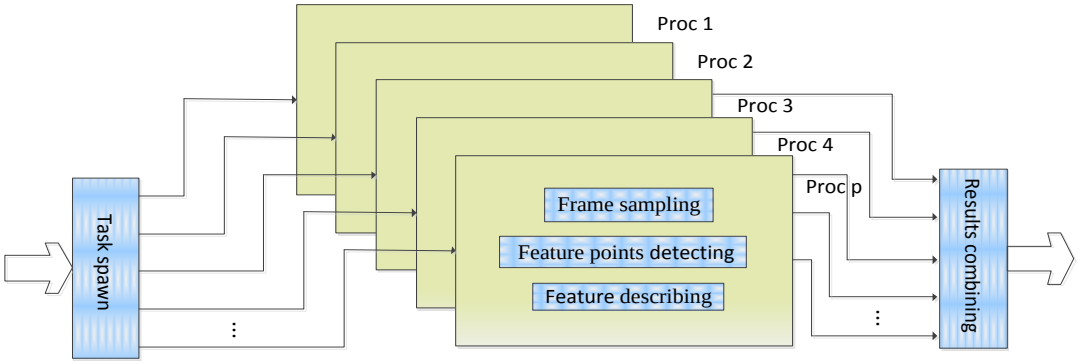



**Fig. 2.** Procedure for parallel feature extraction.

## 2.2 Parallel N-path Labeling HKM (PNHKM) Clustering

The main steps of the HKM clustering algorithm are described in **Algorithm 1**. After program starting and structure initialization, the main part of the algorithm is devoted to data classification (step 3), data partition (step 4).

| **Algorithm 1** HKM clustering |
|---|
| 1. Program start and file reading; |
| 2. Data structures initialized; |
| 3. According to the number of nodes calls the K-means to generate new nodes at the next level; |
| 4. Assign the data items to the new nodes; |
| 5. Check the stopping conditions, if they are not verified to go to step 3, else go to step 6; |
| 6. Store results on the output files. |

To improve the clustering results, we propose a Greedy N-best Paths Labeling (GNPL) method to guide building the vocabulary tree in the step 4. Besides, among those steps, the step 3 and 4 are the most computationally intensive ones. To maintain the same semantics of the sequential clustering algorithm, we design the parallel version of these steps by partitioning data and local computation on each of P processors of a shared memory multicore computer and by exchanging the local variables among the processors that contribute to form the global values of a cluster.

### 2.2.1 Greedy N-best Paths Labeling

In step 4 of HKM algorithm, the clustering process needs to partition the data items into an appropriate number of subsets, where a new data item is assigned by descending the tree. Instead of assigning each data item to the single leaf node on the bottom of the tree, the data item is additionally assigned to some internal nodes which their path from the root to leaf passes through followed by one branch at each level. This is a single path labeling method, which has the advantages of high efficiency. However, the drawback is easy to produce error classification. Schindler et al. [13] proposed a Greedy N-Best Paths (GNP) method, which follows multiple branches at each level rather than just the branch whose parent is closest to

the query feature. The GNP method was used in a retrieval algorithm to search on a vocabulary tree. In their experiments, they showed very good performance in image retrieval. We propose a Greedy N-best Paths Labeling (GNPL) method similar in spirit to GNP, which exploits the unique properties of trees to help to improve the accuracy of the HKM clustering algorithm. First, the data item is additionally assigned to some internal nodes that their paths from the root to leaf pass through followed by N (N≥1 and N≤ branching factor k) branches at each level. Then, we choose one node which is closest to the data item from the last N nodes on the bottom of the tree and assigning the data item to it. In contrast to previous work, rather than searching a vocabulary tree, we propose using GNPL to guide building the vocabulary tree instead. This generalization of the Greedy N-best Paths Labeling method is described in **Algorithm 2**.

**Algorithm 2** Greedy N-Best Paths Labeling

```
Given data item q, level l = 1, and depth m
Compute distance from q to all k children of the root node
While (l < m) {
   Candidates=children of closest N nodes at the level l
   Compute distance from q to all kN candidates
   l = l + 1}
Assign the data item q to the closest candidate
Return
```

For the branching factor k and depth m, the normal labeling algorithm for a tree performs k comparisons between the data item and each of the nodes in the tree at each of the m levels, a total of km comparisons. Our labeling algorithm performs k comparisons at the top level, and kN comparisons at each of the remaining m−1 levels, for a total of k +kN(m−1) comparisons. Of course, the increase of path N has an associated cost that will increase the computational complexity. Even though, we have to consider the tradeoff between accuracy and efficiency. The experimental results indicate that for most of the datasets that we considered, with a moderate size of N, we are able to achieve very good clusters. Note that the traditional labeling algorithm is just the specific case in which N = 1.

### 2.2.2 Parallel Data Classification

The HKM is a top-down and divisive hierarchical clustering algorithm. It divides the vectors in data set $U$ into $k$ groups: $U = \{U_0, U_1, ...U_{k-1}\}$, each of which consists of the vectors closest to the particular cluster center. Then each of the groups $U_i, i = 0,...k-1$, is clustered respectively and sequentially by K-means. The parallel version of this function first divides k groups into p blocks, each of which, with a k/p size of the original data, is imposed by an instance of the program. Then, each processor executes a loop where it does a certain amount of local computation. Finally, all the local results are collected on each processor and are used for computing the global results. This strategy is described in **Fig. 3**.

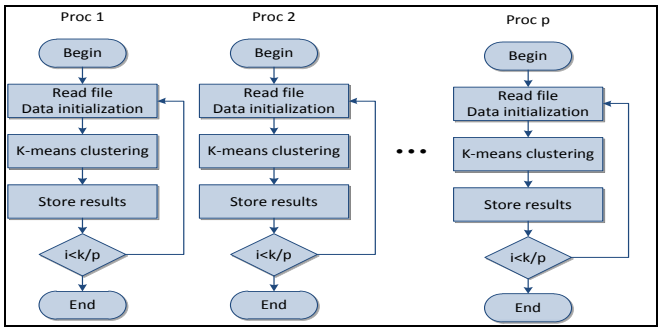



**Fig. 3.** Procedure for parallel data clustering.

### 2.2.3 Parallel Data Partition

During the whole HKM clustering process, it must divide the training data into the current level clusters based on the Euclidean distance. Thus, it generates a smaller data set for each cluster at the current level. In HKM algorithm, the data partition function sequentially computes the distance between a vector and a cluster center and designates the vector to the closest cluster. We proposed a Greedy N-Best Paths Labeling (GNPL) method to help improve the accuracy of division in Section 2.2.1, which follows multiple branches at each level rather than just the branch whose parent is closest to the data item. In the parallel version of GNPL based HKM, the data partition in GNPL is executed in parallel on all the processors. The parallel version of this function first divides the dataset into p blocks, each of which, with a 1/p size of the original data, is imposed by an instance of the program. When the processors write data to a shared file they need a certain amount of local communication for the process mutex is necessary. **Fig. 4** shows the scheme of the parallel version of the function. Note that the principle and method of parallel quantization are the same as the parallel data partition, so, we will no longer introduce it anymore.

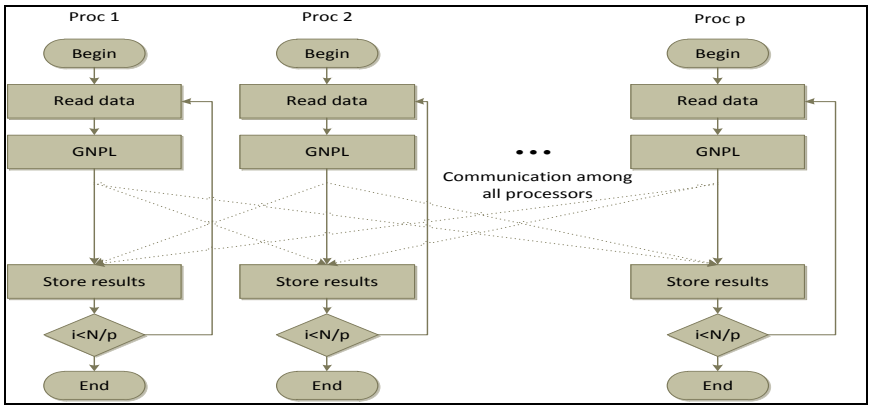



**Fig. 4.** Procedure for parallel data partition.

## 2.3 Parallel Implementation of Retrieval Processing

In order to deal with large video datasets, our PCBCD system builds upon the bag-of-features (BOF) representation. In this section, we will introduce how video search based on BOF vectors works, which can be interpreted as a voting system that matches individual descriptors

with an approximate nearest neighbor (ANN) search [14, 15]. **Fig. 5** shows an illustration of the BOF-based voting process.

Given a query frame represented by its local descriptors $y$ and a set of database frames $j = 1,...,n$ represented by their local descriptors $x_j$, the voting approach can be summarized as:

1. For the descriptors $y_l, l = 1,...,m_1$ of the query video frame and for the descriptors $x_{i,j}, i = 1,...,m_2$ of the database, calculate the score $s_j$ of the corresponding frame image by

$$s_j = \sum_{l=1}^{m_1} \sum_{i=1}^{m_2} f(x_{i,j}, y_l) \tag{1}$$

where $f$ is a matching function which reflects the similarity between descriptors $x_{i,j}$ and $y_l$. The descriptor quantization is used by BOF video search. A quantizer $q$ is formally a function:

$$\begin{aligned} q : R^d &\rightarrow [1,k] \\ x_{i,j} &\mapsto q(x_{i,j}) \end{aligned} \tag{2}$$

which maps a descriptor $x_{i,j} \in R^d$ to an integer index. The quantizer $q$ is obtained by performing the PNHKM clustering on database descriptors. The quantizer $q(x_{i,j})$ is then the index of the centroid closest to the descriptor $x_{i,j}$. So, two descriptors $x_{i,j}$ and $y_l$ that are very close in the descriptor space satisfy $q(x_{i,j}) = q(y_l)$ with a high probability. At this point, the matching function $f$ can be defined as

$$f(x_{i,j}, y_l) = \delta_{q(x_{i,j}),q(y_l)} \tag{3}$$

which allows the efficient comparison of the two descriptors based on their quantized index.

2. The final image score $s_f$ used for ranking is obtained from $s_j$ by applying a post-processing function:

$$s_f = s_j / \sqrt{m_1 m_2} \tag{4}$$

which takes into account the number of image descriptors .

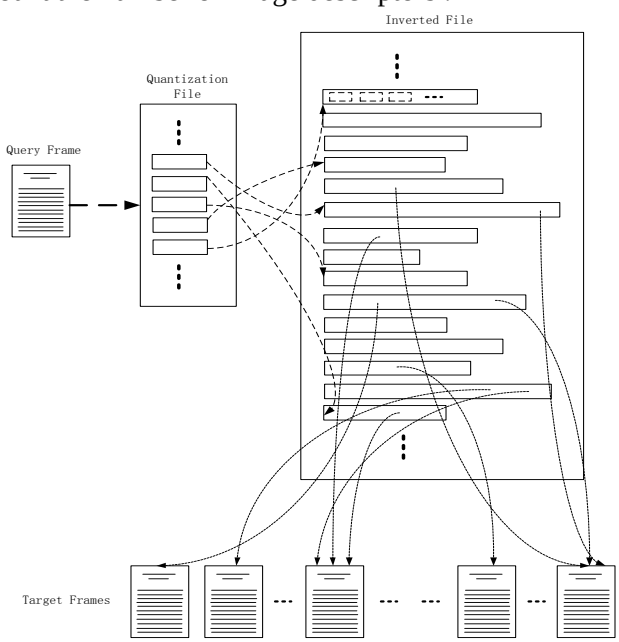



**Fig. 5.** Illustration of the BOF-based voting process.

### 2.3.1 Weighting and Reranking

The voting approach described above is represented by a vector of word frequencies. However, it is usually to use a weighting at the components of this vector, rather than employ the frequency vector directly in indexing. Here we describe the standard weighting which is employed to our PCBCD system.

The standard weighting is known as 'term frequency– inverse document frequency' (TF-IDF), which weights the visual words according to their frequency [16]. Suppose there is a vocabulary of $k$ visual words, then each video frame is represented by a vector $V_d = (w_1, ..., w_i, ...w_k)^T$ of weighted visual word frequencies with all components

$$tf_i = f_{ij} \Big/ \sum_{t=1}^{k} f_{tj} \tag{5}$$

$$idf_i = \log \frac{N}{n_i} \tag{6}$$

$$w_i = tf_i \cdot idf_i \tag{7}$$

where $f_{ij}$ is the number of occurrences of visual word $i$ in video frame $j$, $n_i$ is the number of video frames in which visual word $i$ occurs and $N$ is the number of video frames in the whole database. It's obviously that $tf_i$ weights visual words occurring often in a particular frame, while the $idf_i$ down weights visual words that appear often in the database.

Take into account the TF-IDF scheme, the matching function (3) and (4) will be modified:

$$f_{\text{tf-idf}}(x_{i,j}, y_l) = (w_{q(y_l)} \cdot w_{q(x_{i,j})}) \delta_{q(x_{i,j}), q(y_l)} \tag{8}$$

$$s_f = s_j \Big/ \sqrt{\sum_{l=1}^{m_1} w_{q(y_l)}{}^2 \sum_{i=1}^{m_2} w_{q(x_{i,j})}{}^2} \tag{9}$$

So the TF-IDF weight is applied to both the query and the dataset video frames in the BOF inner product. This weighting scheme normalizes the number of votes by the visual word histogram of the database image.

### 2.3.2 Temporal Consistency

If one video is a copy of others, then the two videos would have a consistent time shift between the matched frames [17]. This is the basic ideal of the temporal consistency. According to this principle, we can find the time shift and we only calculate the scores of the matched frames whose time shift meet the conditions. Thus we can remove a large part of the errors in matching.

The output of the frame level query is a set of frame matches

$$(t_q, d, t_d, s_f) \tag{10}$$

where $t_d$ and $t_q$ are the timestamps of the matched database and query frames, $d$ is the database video, and $s_f$ is the final score of the frame match, computed by (9).

Each matching results in a candidate $(d, \delta t)$, where $\delta t = t_d - t_q$, which aligns the database $d$ with the query video $q$. Candidates vote in the temporal Hough space of the corresponding database video $d$ with $\delta t$. The time shift for each database video is obtained as the local maximum in the Hough space (the 1-D histogram $H_d$ of $\delta t$). The histograms have a bin size of $t_{bin} = 1s$. We group together the bins, which are near to the local maximum bins, into short segments with a size of $t_{seg} = 60s$. A shortlist of $(d, \delta t)$ candidates is obtained as the segment

with the highest scores. A video score $s_v$ is then obtained as the sum of the scores of the frames in the shortlist divided by the number of query video frames $n$.

$$s_v = \frac{1}{n}\sum s_f \tag{11}$$

This method separates matches that have different time shifts, which removes the influence of incorrect noncontiguous matches. **Fig. 6** shows the temporal consistency checking process. On the left, it shows the frames matching between a query video and two database videos; on the other side, it shows the corresponding $\delta t$ time shift histograms. There are more frames matched between the query and video 2, but the matched frames with video 1 are temporally consistent.

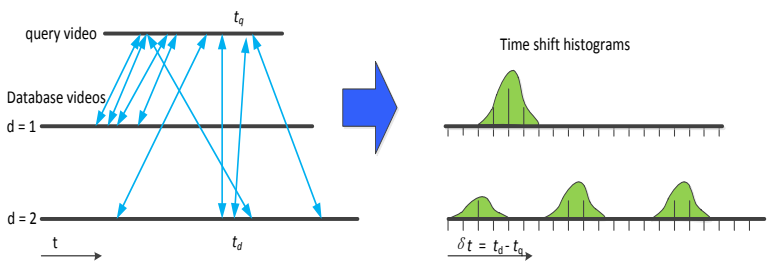



**Fig. 6.** Illustration of the temporal consistency checking process.

To maintain the same semantics of the sequential retrieval algorithm, we design the parallel version of the retrieval algorithm, the search processing is executed in parallel on all the processors. We first divide the number of videos N into p instances of a program so that each instance works on an N/p sized block of the original data. Then each processor executes an independent program on different data. So the processors do not need additional communication. Finally, local partial results are collected among all the processors to have global values on every processor. **Fig. 7** shows the scheme of the parallel version of the function.

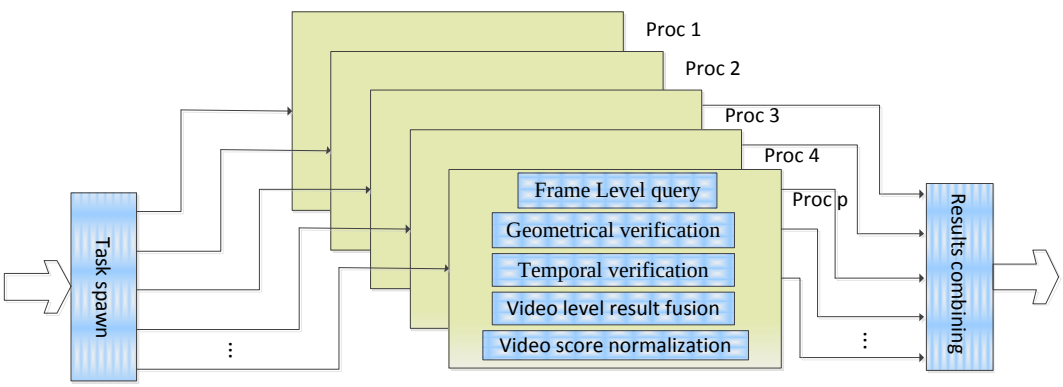



**Fig. 7.** Procedure for parallel retrieval.

# 3. Experimental Setup

## 3.1 Datasets

The video sequences used to construct the reference databases come from the so-called SNC database stored at the French Institut National de l'Audiovisuel (INA), whose main tasks include collecting and exploiting French television broadcasts.

In order to evaluate the performance of the PCBCD system we use the specific ground-truth databases which are randomly selected subparts of the SNC database and we note $R_H$ a randomly selected subpart containing H hours of videos. **Table 1** summarizes the relative sizes of the datasets.

**Table 1.** The number of keyframes, features and descriptor sizes for each dataset.

| Dataset | # Keyframes | # features | Size of descriptors |
|---|---|---|---|
| $R_{50}$ | 42 118 | 16 847 500 | 5.6G |
| $R_{100}$ | 105 530 | 38 212 342 | 11.8G |
| $R_{200}$ | 341 762 | 116 251 837 | 36.8G |
| $R_{400}$ | 672 114 | 268 845 792 | 89.4G |

As for the query videos we consider 10 categories of visual transformations (including insertions of logo/text, re-encoding, change of gamma, blur, change of contrast, white noise, horizontal mirroring, horizontal or vertical shift, picture in picture, letterbox). We build the validation dataset as follows: 1000 video clips are randomly selected in the $R_H$ database and then transformed with a combination of previous transformations. The transformation parameters and the number of transformations are randomly selected for each video clip. This query video set is referred as $Q_H$, and these 1000 transformed video clips represent the true probes. The false probes come from online video repository www.youtube.com that is supposed to never broadcast on the SNC database. In order to have a more realistic precision measure, this query video set is ten times longer than $Q_H$, and it is referred as $Q_F$. The total length of all the queries ($Q_H$+ $Q_F$) is 169 000 s (46 h 56 min).

Our experiments also made use of the publicly available TRECVID video sets (tv.2007, tv.2009 and tv.2010). The TRECVID databases stored at the servers of U.S. National Institute of Standards and Technology (NIST), whose main goal is to promote progress in content-based analysis of and retrieval from digital video via open, metrics-based evaluation. We use the original data of TRECVID copy detection task to compare with the state-of-arts methods.

## 3.2 Experimental evaluation criteria

The precision-recall curves are standard evaluation criteria of measuring the performance of an information retrieval system. We have generated the curves for our PCBCD system. Recall and precision metrics are defined as

$$\text{Recall} = \frac{\text{number of true positives}}{\text{total number of true}} \tag{12}$$

$$\text{Precision} = \frac{\text{number of true positives}}{\text{total number of positives}} \tag{13}$$

The average precision (AP) for a single query q is the mean over the precision scores at each relevant item:

$$AP(q) = \frac{1}{g(q)} \sum_{k=1}^{g(q)} \frac{k}{r(k)} \tag{14}$$

where g(q) is the total number of the ground truth videos for the query q. Consider a query q and assume that the kth ground truth video is found to be the Rth result of the retrieval. Then r(k) = R. Consequently, the mean average precision (MAP) is the mean of the average precision scores over all queries:

$$MAP = \frac{1}{Q} \sum_{q \in Q} AP(q) \tag{15}$$

where Q is the set of queries q. In the perfect retrieval case MAP = 1 and as the number of the nonrelevant videos ranked above a retrieved relevant video increases, the MAP approaches the value 0, $MAP \in [0,1]$.

We also use the average retrieval time (ART) measure for a set of query videos. The average retrieval time is defined as the ratio of total retrieval time to the total number retrieved videos.

$$\text{Average Retrieval Time } (\text{ART}) = \frac{\text{total retrieval time}}{\#\text{retrieved total videos}} \tag{16}$$

In order to evaluate the performance of PNHKM clustering algorithm we only compute the average retrieval time from inputting the query visual vocabularies to the end of the retrieve.

## 4. Experimental results

In order to evaluate the performance of the PNHKM clustering method, we compare the performance of four different clustering methods: (a) PNHKM, (b) hierarchical K-means (HKM) [10], (c) parallel version of HKM (PHKM), and (d) GNPL based HKM (NHKM). The experiments are carried out on the $R_{50}$, $R_{100}$, $R_{200}$ and $R_{400}$ datasets. The number of available cores p is 30 (p=30), and the branches in PNHKM and NHKM is three (N=3). The results are shown in **Table 2** and **Table 3**. Through **Table 2** and **Table 3** we can see that the PHKM performance is almost identical to the HKM in MAP but the clustering time is much less. PHKM and HKM have adopted the same hierarchical tree structure, but the PHKM method is much faster than the best HKM method. This is attributed to the effects of the parallel clustering algorithm. It is obvious that GNPL based HKM method gives very good performance in MAP. From the figures in the column of Clustering time, we can see that HKM is faster than the GNPL based HKM, which is due to the fact that GNPL is somewhat time-consuming. We can see that the PNHKM performance is superior to the HKM both in MAP and the clustering time. PNHKM and HKM have adopted the same hierarchical tree structure, but the PNHKM method outperforms the best HKM method. This is attributed to the effects of GNPL clustering algorithm. And PNHKM is much faster than HKM, which is attributed to the effects of using the parallel method.

**Table 2.** Comparison of the MAP performance of PNHKM, HKM, PHKM, and NHKM on four datasets ($R_{50}$, $R_{100}$, $R_{200}$ AND $R_{400}$). The branches in PNHKM and NHKM is three (N=3), and the number of available cores p is 30 (p=30).

| Dataset | MAP | | | |
|---|---|---|---|---|
| | HKM | NHKM (N=3) | PHKM (p=30) | PNHKM (N=3,p=30) |
| $R_{50}$ | 0.948 | 0.975 | 0.947 | 0.975 |
| $R_{100}$ | 0.937 | 0.966 | 0.937 | 0.965 |
| $R_{200}$ | 0.921 | 0.958 | 0.922 | 0.959 |
| $R_{400}$ | 0.912 | 0.947 | 0.913 | 0.948 |

**Table 3.** Comparison of the Clustering time performance of PNHKM, HKM, PHKM, and NHKM on four datasets ($R_{50}$, $R_{100}$, $R_{200}$ AND $R_{400}$). The branches in PNHKM and NHKM is three (N=3), and the number of available cores p is 30 (p=30).

| Dataset | Clustering time (hour) | | | |
|---|---|---|---|---|
| | HKM | NHKM (N=3) | PHKM (p=30) | PNHKM (N=3,p=30) |
| $R_{50}$ | 36.5 | 70.8 | 1.7 | 3.1 |
| $R_{100}$ | 70.7 | 132.6 | 3.5 | 6.6 |
| $R_{200}$ | 127.2 | 242.3 | 6.1 | 11.3 |
| $R_{400}$ | 210.8 | 398.9 | 11.6 | 20.4 |

In order to evaluate the parallel performance of the proposed PCBCD system, we now compare the processing time in off-line processing and on-line processing to discuss the cost of parallel computation on the $R_{50}$, $R_{100}$, $R_{200}$ and $R_{400}$ datasets with a different number of processors. As shown in the **Table 4**, the 4 cores server has about a 4x speedup over the sequential system on off-line processing. The speedup can increase to about 8x when we use the PCBCD system on an 8 cores server. The processing time will be shortened 20x if we put the system on a 32 cores server. It can be seen that the processing time tends to decrease linearly as the number of processors increases. **Table 4** also shows the speedup for different data sets with varying data sizes.

**Table 4.** Influence of the number of processors on the speed of PCBCD

| Parallel task | # processors | Processing time (h) | | | |
|---|---|---|---|---|---|
| | | $R_{50}$ | $R_{100}$ | $R_{200}$ | $R_{400}$ |
| Off-line processing | 1 | 76 | 150 | 285 | 552 |
| | 4 | 20 | 39 | 73 | 138 |
| | 8 | 9.5 | 19 | 38 | 72 |
| | 32 | 3.8 | 7.5 | 15 | 28 |
| On-line processing | 1 | 49 | 50 | 52 | 54 |
| | 4 | 12.3 | 12.5 | 13 | 13.5 |
| | 8 | 6.2 | 6.3 | 6.5 | 6.9 |
| | 32 | 2.4 | 2.6 | 2.7 | 2.8 |

**Fig. 8** shows the relationship of the acceleration ratio and the number of processors. The ratio of acceleration can be used to evaluate the effectiveness of parallel computation. It can be formally described: $S_p = T_s / T_p$. Where p denotes the number of processors, $T_p$ denotes the time

consumed by parallel computation with p nodes, $T_s$ denotes the time consumed by a single processor. Parameter $S_p$ reflects the speed of parallel computation. An inflexion point occurred when 23 computing nodes were used, with an acceleration ratio of more than 19. This is probably due to the fact that more communication cost, resource competition and IO (Input/Output) cost were needed as the number of processors increased, which lead to a lower acceleration ratio. The acceleration ratio is one of the key indexes of parallel processing, and gaining of a higher acceleration ratio has relation not only with the task partition and the IO cost but also with the communication between processors and the resource competition. In general, when the data size of parallel programs is fixed, the acceleration ratio increases with the increasing of the total number of processors. But in Multi-core parallel, all processors share one memory and physical transmission media; as a result, the acceleration ratio will not increase with the increasing of processors when the total number of processors exceeds a score. In our experiment, when the number of processors increased to 23 the increasing of acceleration ratio is not obvious, and then increasing the processor is not desired.

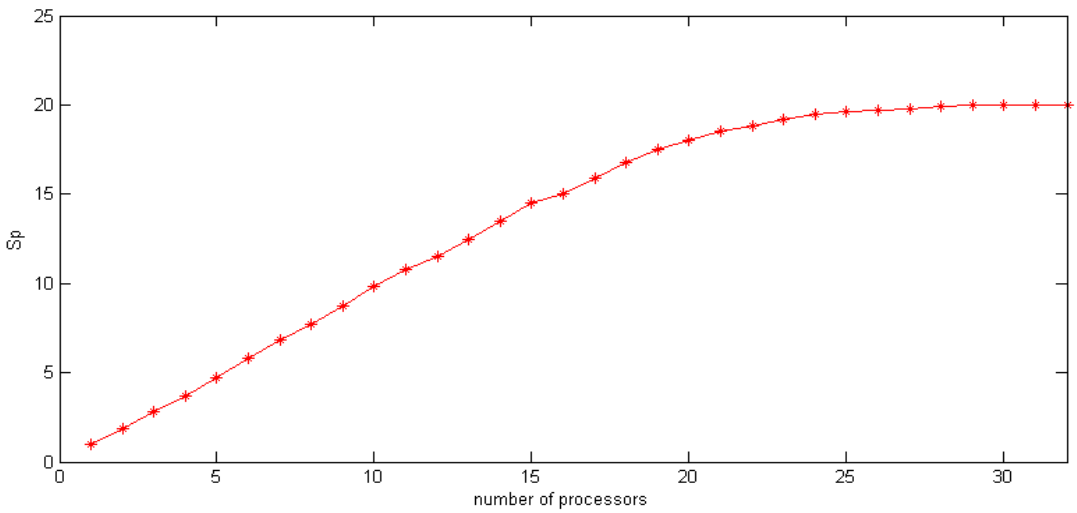



**Fig. 8.** Relationship of acceleration ratio to number of processors.

**Fig.9** describes the precision/recall curves of the proposed PCBCD system for the validation dataset. Despite the differences of the two orders of magnitude between the smallest and the largest database, the recall and the precision of the proposed system do not significantly degrade. This robustness to database size is owing to the use of local features and to the discrimination of BOF based voting strategies. Even if the number of irrelevant features provided by the retrieval for each local feature is growing linearly, the impact on the final result is very limited for the reason that the number of consistent matches remains very low.

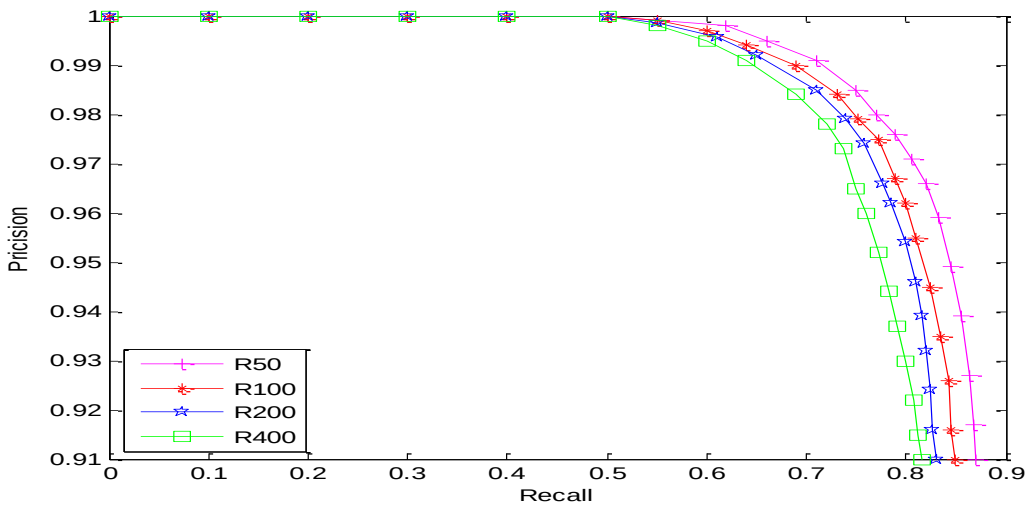



**Fig. 9.** ROC curves for different database sizes.

In the following, we compare to the state-of-the-art on the SNC datasets and the TRECVID databases. The comparison is performed with Video Google [18], VQ-Based Linear Search-Method (VQLS) [19], Frame Fusion for Video Copy Detection (FF)[20], and video copy detection using inclined video Tomography and bag-of-visual-words (TBOW)[21], which to our knowledge are the state-of-the-art for this benchmark.

First, we compare five copy detection systems on different datasets. **Table 5** compares our best results with the state-of-the-art methods using the accuracy measure MAP on the SNC datasets. All the results presented have been obtained for a vocabulary learned on an independent dataset. Our best results on the $S_{50}$, $S_{100}$, $S_{200}$ and $S_{400}$ datasets are better than the state-of-the-art methods. In the results of **Table 5**, the gain due to our method is more significant for $S_{400}$ dataset. This is due to the precision of our PNHKM clustering algorithm, which performs better than other the conventional clustering methods when the reference video dataset is very large. **Table 6** compares our best results with the state-of-the-art methods using the accuracy measure MAP on the TRECVID databases. From **Table 6**, we can see that the overall performance of these algorithms have declined in the TRECVID databases. The main reason is that the transformation of query videos in TRECVID is more complicated than SNC database. Although the performance of the proposed method has declined, it is still better than other methods in MAP.

**Table 5.** Comparison of our best results with the state-of-the-art methods using the accuracy measure MAP on the SNC datasets.

| | $S_{50}$ | $S_{100}$ | $S_{200}$ | $S_{400}$ |
|---|---|---|---|---|
| Video Google (2009) | 0.959 | 0.956 | 0.950 | 0.944 |
| VQLS (2010) | 0.971 | 0.965 | 0.956 | 0.950 |
| FF (2011) | 0.968 | 0.962 | 0.959 | 0.953 |
| TBOW(2012) | 0.970 | 0.965 | 0.952 | 0.954 |
| Proposed method | **0.975** | **0.971** | **0.969** | **0.967** |

**Table 6.** Comparison of our best results with the state-of-the-art methods using the accuracy measure MAP on the TRECVID databases.

| | tv.2007 | tv.2009 | tv.2010 |
|---|---|---|---|
| Video Google (2009) | 0.792 | 0.786 | 0.764 |
| VQLS (2010) | 0.821 | 0.815 | 0.806 |
| FF (2011) | 0.826 | 0.818 | 0.810 |
| TBOW(2012) | 0.834 | 0.816 | 0.809 |
| Proposed method | **0.845** | **0.828** | **0.818** |

We also compare the proposed method with state-of-the-art methods on the time complexity. **Table 7** compares our average retrieval time with the state-of-the-art methods using the accuracy measure ART. The number of available cores p is 24 (p=24), and the branches in PNHKM is three (N=3). From **Table 7**, we can see that our ART results on the S50, S100, S200 and S400 datasets are better than the state-of-the-art methods. This is attributed to the Multi-core parallel processing ability of PCBCD. VQLS is faster than Video Google, FF and TBOW. The reason may be that that VQLS used the global features, and other methods adopted the local features.

**Table 7.** Comparison of our average retrieval time with the state-of-the-art methods using the accuracy measure ART. The branches in PNHKM is three (N=3), and the number of available cores p is 24 (p=24).

| | ART (s) | | | |
|---|---|---|---|---|
| | $S_{50}$ | $S_{100}$ | $S_{200}$ | $S_{400}$ |
| Video Google (2009) | 6.42 | 10.85 | 18.23 | 30.67 |
| VQLS (2010) | 1.25 | 2.16 | 3.45 | 5.28 |
| FF (2011) | 4.63 | 8.14 | 12.96 | 22.53 |
| TBOW(2012) | 3.47 | 5.84 | 9.58 | 16.32 |
| Proposed method | **0.12** | **0.23** | **0.34** | **0.57** |

Second, we compare these systems on the $S_{400}$ dataset for different transformations. Comparison with state-of-the-art copy detection systems on MAP with varied transformations is plotted in **Fig.10**. For the copy detection, the proposed method obtains the best performance for seven transformations, i.e., insertions of logo/text, re-encoding, change of gamma, change of contrast, white noise, crop and shift. The performance for other transformations, i.e., blur, horizontal mirroring, picture in picture and letterbox, is also comparable with the best ones. This means that our proposed copy detection method is indeed robust to various video distortions.

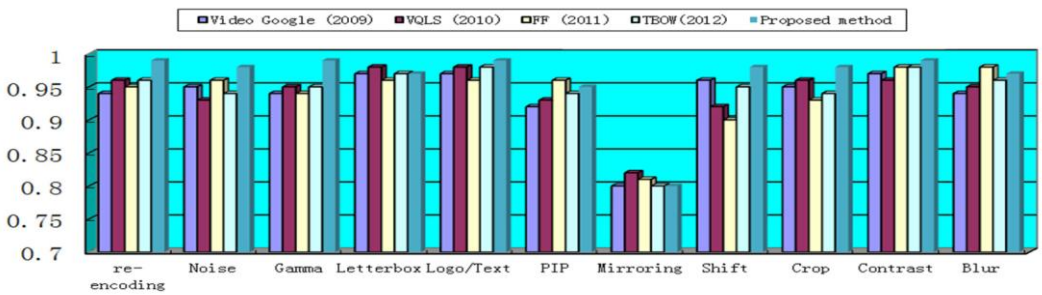



**Fig. 10.** Comparison with state-of-the-art copy detection systems on MAP with varied transformations.

Finally, we show an example of copy detection experiment. The databases for the experiment are the existent databases introduced in sections 3.1. The image matching part of our system is developed to handle pictures of natural scenes seen under different viewing conditions. **Fig. 11** shows a few examples of transformations to which our system is robust. In the experiment, we retrieve the copied videos for about a minute long query in 0.65 second with an average detection accuracy of over 98%. The proposed system can then be applied to detect videos that require copyright protection.


**Fig. 11.** Example frames (right) of the transformed videos that our system recognizes correctly and (left) of the corresponding database videos.

## 5. Conclusion

This paper introduces a parallel implementation strategy for content based video copy detection. We propose a unified parallel computing framework for the feature extraction, data clustering and video searching tasks in the video copy detection system. The unified computing framework is based on the SPMD programming model, which supports shared memory multi-core servers. The proposed multi-core processor computing framework can support video copy detection effectively, and the processing time tends to decrease linearly as the number of processors increases. Experiments have shown that our approach is successful in speeding up computation while keeping the performance.

## References

[1] X. Zhou and L. Chen, "Structure Tensor Series-Based Large Scale Near-Duplicate Video Retrieval," *Multimedia, IEEE Transactions on*. vol.14, no.4, pp. 1220-1233, August, 2012. Article (CrossRef Link)

[2] Y. Mussarat, S. Muhammad, M. Sajjad and I. Isma, "Content Based Image Retrieval Using Combined Features of Shape, Color and Relevance Feedback," *KSII Transactions on internet and information systems,* vol.7, no.12, pp. 3149-3165, July, 2013. Article (CrossRef Link)

[3] G. Barlas "An analytical approach to optimizing parallel image registration/retrieval," *Parallel and Distributed Systems, IEEE Transactions on*. vol.21, no.8, pp. 1074-1088, August, 2010. Article (CrossRef Link)

[4] A. Plaza, D. Valencia, J. Plaza and P. Martinez, "Commodity cluster-based parallel processing of hyperspectral imagery," *Journal of Parallel and Distributed Computing,* vol.66, no.3, pp. 345-358, March, 2006. Article (CrossRef Link)

[5] M. Emmanuel, D. R. R. Babu, J. Jagdale, P. Game and G. Potdar, "Parallel Approach for Content Based Medical Image Retrieval System," *Journal of Computer Science*, vol.6, no.11, pp. 1258-1262, September, 2010. Article (CrossRef Link)

[6] K. Liu, T. Zhang and L. Wang, "A new parallel video understanding and retrieval system," in *Proc. of Multimedia and Expo (ICME), 2010 IEEE International Conference on. IEEE*, July 19-23, 2010. Article (CrossRef Link)

[7] B. Geng, Y. Li, D. Tao, M. Wang, Z. J. Zha and C. Xu, "Parallel Lasso for Large-Scale Video Concept Detection," *Multimedia, IEEE Transactions on*. vol.14, no.1, pp. 55-65, February, 2012. Article (CrossRef Link)

[8] H. Sutter and J. Larus, "Software and the concurrency revolution," *Queue,* vol.3, no.7, pp. 54-62, September, 2005. Article (CrossRef Link)

[9] J. Sivic and A. Zisserman, "Video Google: a text retrieval approach to object matching in videos," in *Proc. of Ninth IEEE International Conference on Computer Vision*, pp. 1470-1477 vol.2, October 13-16, 2003. Article (CrossRef Link)

[10] C. Pizzuti and D. Talia, "P-autoclass: Scalable parallel clustering for mining large data sets," *Knowledge and Data Engineering, IEEE Transactions on*. vol.15, no.3, pp. 629-641, June, 2003. Article (CrossRef Link)

[11] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision,* vol.60, no.2, pp. 91-110, November, 2004. Article (CrossRef Link)

[12] K. Liao, G. Liu and Y. Hui, "An improvement to the SIFT descriptor for image representation and matching," *Pattern Recognition Letters*, pp. 1211-1220, August, 2013. Article (CrossRef Link)

[13] G. Schindler, M. Brown and R. Szeliski, "City-scale location recognition," in *Proc. of Computer Vision and Pattern Recognition*, *2007. CVPR'07. IEEE Conference on. IEEE*, June 17-22, 2007. Article (CrossRef Link)

[14] H. Jegou, M. Douze and C. Schmid, "Improving Bag-of-Features for Large Scale Image Search," *International Journal of Computer Vision*, vol.87, no.3, pp. 316-336, May, 2010. Article (CrossRef Link)

[15] S.-H. Yen andY.-J. Hsieh, "A KD-Tree-Based Nearest Neighbor Search for Large Quantities of Data," *KSII Transactions on Internet & Information Systems*, vol.7, no.3, pp. 459-470, March 2013. Article (CrossRef Link)

[16] W. Zhang, T. Yoshida and X. Tang, "A comparative study of TF*IDF, LSI and multi-words for text classification," *Expert Systems with Applications*, vol.38, no.3, pp. 2758-2765, March, 2011. Article (CrossRef Link)

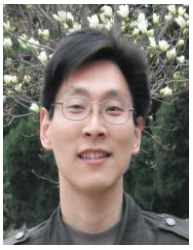
[17] M. Douze, H. Jegou and C. Schmid, "An Image-Based Approach to Video Copy Detection With Spatio-Temporal Post-Filtering," *IEEE Transactions on Multimedia*, vol.12, no.4, pp. 257-266, June, 2010. Article (CrossRef Link)

[18] J. Sivic and A. Zisserman, "Efficient Visual Search of Videos Cast as Text Retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.31, no.4, pp. 591-606, April, 2009. Article (CrossRef Link)

[19] A. Sarkar, V. Singh, P. Ghosh, B. S. Manjunath and A. Singh, "Efficient and Robust Detection of Duplicate Videos in a Large Database," *IEEE Transactions on Circuits and Systems for Video Technology*, vol.20, no.6, pp. 870-885, June, 2010. Article (CrossRef Link)

[20] S. Wei, Y. Zhao, C. Zhu, C. Xu and Z. Zhu, "Frame fusion for video copy detection," *Circuits and Systems for Video Technology*, *IEEE Transactions on*. vol.21, no.1, pp. 15-28, January, 2011. Article (CrossRef Link)

[21] H. Min, S. M. Kim, W. De Neve and Y. M. Ro, "Video Copy Detection Using Inclined Video Tomography and Bag-of-Visual-Words," in *Proc. of Multimedia and Expo (ICME), 2012 IEEE International Conference on. IEEE*, July 9-13, 2012. Article (CrossRef Link)



**Kaiyang Liao** received the B.S. degree in computer science from the XIDIAN University, Xi'an, China, in 2004, the M.S. degree in computer science from the University of Science and Technology LiaoNing, Anshan, China, and the Ph.D. degree in Information and Communication Engineering from the Xi'an Jiaotong University, Xi'an, China, in 2013. He is currently a Full lecturer with the School of Printing and Packaging Engineering, Xi'an University of Technology, Xi'an, China. His research interests include data mining, pattern recognition, video analysis and retrieval.

**Fan Zhao** received the Ph.D. degree in electronics and information engineering from Xi'an Jiaotong University, Xi'an, China, in 2009. She is an associate professor in the Departm nt of Information Science, Xi'an University of Technology, and now working in the department of Computer Science and Engineering as a Postdoctoral Fellow, Xi'an Jiaotong University. Her research interests include data mining, image processing and video compression.

**Mingzhu Zhang** received the B.S. degree in computer science from the XIDIAN University, Xi'an, China, in 2004, the M.S. degree in Management science and Engineering from the Xi'an Technological University, Xi'an, China, in 2011. She is currently a Full lecturer with the Department of Public Courses, Xi'an Fanyi University, Xi'an China. Her research interests include data mining, pattern recognition, video analysis and retrieval.