Time-Series Linear Search for Video Copies Based on Compact Signature Manipulation and Containment Relation Modeling

Chih-Yi Chiu, *Member, IEEE,* and Hsin-Min Wang, *Senior Member, IEEE*

***Abstract*—This paper presents a novel time-series linear search (TLS) method for detecting video copies. The method utilizes a sliding window to locate window sequences that are near-duplicates of a given query sequence. We address two issues of the conventional TLS method in order to strengthen its video copy detection capability. First, to accelerate the TLS process, we use a sequence-level signature as a compact representation of a video sequence based on the min-hash theory, and develop an efficient heap manipulation technique for fast generation of each window sequence's signature. Second, to improve the robustness of the TLS method, we use two techniques, namely, window length estimation and threshold transform, to resolve the containment relation problem caused by various types of video transformation and editing, such as frame cropping and speed change. The results of experiments on the MUSCLE-VCD-2007 dataset demonstrate that the proposed method is efficient and robust against different types of video transformation and editing.**

***Index Terms*—Video copy detection, content-based retrieval, near-duplicate, fingerprint identification**

I. Introduction

Detecting copies of digital documents, such as text, image, and video content, has been an active research area for decades. Copy detection techniques for text documents have been used in many applications of databases and search engines [4][14]. In recent years, the maturity of hardware and software has generated enormous amounts of image/video content, creating the need for copy detection techniques for multimedia. Such techniques allow content owners to monitor their image/video content for infringement detection and data mining [25]; blog operators to identify near-duplicate images/videos for piracy removal, search result aggregation, or tag suggestion [33]; and TV advertisers to check if their commercials are broadcast as contracted by TV channels [13]. In this paper, we focus on content-based video copy detection (CBVCD).

Manuscript received February 9, 2010; revised July 21, 2010. This work was supported in part by the National Science Council of Taiwan under Grants NSC 99-2221-E-415-011 and NSC 99-2631-H-001-020.

C. Y. Chiu is with the Department of Computer Science and Information Engineering, National Chiayi University, Chiayi City, 60004, Taiwan (phone: +886-5-2717228; fax: +886-5-2717705; e-mail: cychiu@mail.ncyu.edu.tw).

H. M. Wang is with the Institute of Information Science and Research Center for Information Technology Innovation, Academia Sinica, Taipei, 11529, Taiwan (phone: +886-2-27883799; fax: +886-2-27824814; e-mail: whm@iis.sinica.edu.tw).

A. Related Work

We first define two terms, namely "copy" and "near-duplicate", used in this paper. A video copy corresponds to a video sequence derived by applying video transformation and editing on a source sequence; while a video near-duplicate means its content is considered highly similar to a source sequence. A near-duplicate might not be a copy of a source sequence, whereas a copy would be a near-duplicate. The above definitions are generally accepted in most studies and competitions, although a slightly different viewpoint considers a video copy a transformed video sequence of the source sequence rather than an identical or a near-duplicated one [20]. Recently, higher-level semantic content has been involved in the definition of a video copy. For example, the same scene recaptured with different capturing configurations might be considered a copy of the original concept [2][28][33]. Furthermore, Cherubini *et al*. [6] proposed a user-centric-based definition that video sequences are considered near-duplicates if they are visually similar and semantically related, while identical video sequences with relevant complementary information are not considered near-duplicates.

In order to enrich and facilitate the discussion in this paper, we do not distinguish between "copy" and "near-duplicate"; the two terms will be used interchangeably hereafter. As the video copy definition is still an open issue, we adopt the general definition that a video copy corresponds to a video sequence derived by applying video transformation and editing on a source sequence. Some widely used types of video transformation and editing are categorized as follows:

(1) Preserved-frame-region transformation and editing. This category includes *brightness enhancement*, *compression*, *noise addition*, and *frame resolution change*, which modify the content of the whole frame without discarding any frame regions.

(2) Discarded-frame-region transformation and editing. This category includes *frame cropping* and *zoom-in*, which discard parts of the frame region and modify the remaining content.

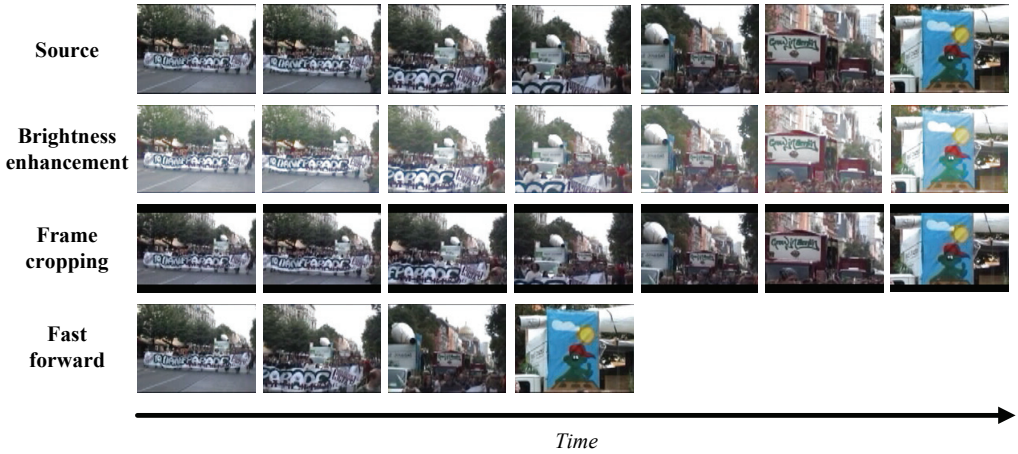


Fig. 1. Examples of video transformation and editing.

(3) Changed-frame-number transformation and editing. This category includes *frame rate change* and *video speed change* (fast forward and slow motion), which increase or decrease the number of frames in a certain period.

Figure 1 gives some examples of video transformation and editing. The first row lists the source video frames, and the remaining rows list the copy frames by applying brightness enhancement, frame cropping, and fast forward, respectively. Note that categories (2) and (3) induce the *containment relation* between the source and the copy, i.e., one video is a subset of the other. It will be discussed in the later paragraph.

A number of feature representations have been proposed to resist some types of video transformation and editing. For example, global descriptors, such as the ordinal measure [7][8][18], the color-shift and centroid-based signature [15], and the spatial correlation descriptor [34], are used to model the properties of an entire frame region. The merit of global descriptors is that their dimensions are very compact; only a 9-dimensional ordinal measure or spatial correlation descriptor is needed for a 3×3-block frame, and a 2-dimensional color-shift and centroid-based signature is needed for a single frame. Although they have been shown to be robust against preserved-frame-region transformation and editing in many studies, the global descriptors of the modified frame might be totally different from those of the source frame if the source frame region is partial discarded.

Due to the limitation of global descriptors, some researchers have employed local descriptors, such as scale-invariant feature transform (SIFT) [22] and speeded up robust features (SURF) [3], to capture the local region properties of the keypoints in a frame. Some researchers further aggregated local descriptors in a "bag-of-words" form [7][9][16][24][29][35]. Wu *et al*. [33] employed the color histogram as the global descriptor for fast rejection of unlikely video clips, and utilized the SIFT descriptor as the local descriptor for subsequent comparison of the remaining video clips. Basharat *et al*. [2] extracted SIFT trajectories to form spatiotemporal volumes. Local descriptors are more robust than global descriptors in handling discarded-frame-region transformation and editing. However, the dimensionality of the local descriptor-based feature is usually much higher than that of the global descriptor-based feature.

Video matching methods can be divided into two categories. One is the elementary unit search method, in which the elementary unit can be a shot, a trajectory, or a keyframe. The conventional shot-based video retrieval technique is naturally used in detecting near-duplicate shots [9][33]. Law-To *et al*. [20] used a trajectory feature and matched trajectories by registering their spatiotemporal relation. Shen *et al*. [27] transformed the keyframe matching task to a bipartite graph problem and solved it by the maximum size matching algorithm. Tan *et al*. [30] modeled the spatiotemporal consistency among keyframes as a network maximum flow problem. These methods usually rely on efficient indexing techniques (e.g., tree and hash techniques) to maintain a shot/trajectory/keyframe collection. However, the indexing techniques are apt to deteriorate significantly when dealing with the high-dimensional and large-scale dataset [32].

The other category is the time-series linear search (TLS) method, which has also been widely used in CBVCD [7][8][15][17][18] because of its simple computation process. Basically, it employs a fixed-length sliding window to scan video streams and computes the similarity between the query and window sequences. Since TLS does not utilize indexing techniques, its performance would not be affected seriously when processing the high-dimensional and large-scale dataset [19]. However, the method might experience problems when handling video transformation and editing events that induce the containment relation. For example, consider the source and its fast forward copy shown in Figure 1. It is clear that, with a fixed-length window, the source content and the copy content do not synchronize; hence, the similarity between the two sequences could be very low. Moreover, even though local descriptor-based features are expected to be robust to frame cropping, the discarded part might still result in a decline in the similarity between the source and its cropped copy. The above two cases would induce the increment of *false negatives*, i.e., real copies that are not detected.

Another research issue of TLS is the efficiency. Instead of exhaustive frame-by-frame scanning, Kashino *et al*. [17] proposed a histogram pruning algorithm to accelerate the search process by skipping unnecessary frame scanning. The number

of frames that can be skipped is proportional to the difference between the similarity and the threshold. The main drawback is that histogram pruning does not guarantee a stable acceleration because it would be not very efficient if the difference between the similarity and the threshold is very small.

B. The Contribution of this Work

We present a novel TLS method for video copy detection. Specifically, a compact signature derived based on the min-hash theory is used to represent a video sequence. In our experiments, a 50-dimensional min-hash signature is sufficient to represent a 60-frame sequence, where each dimension represents the min-hash value of an integer between 1 and 1024. Therefore, only 500 bits are needed to represent 60 frames (bit rate: 8.3 bits/frame). The proposed min-hash signature is a sequence-level feature that differs from the frame-level min-hash sketch proposed by Chum *et al*. [9]. As the number of frames of the query sequence increases, the frame-level feature would become more inefficient than the sequence-level feature.

We further develop an efficient algorithm for generating the min-hash signature with the help of heap manipulation. A heap [11] is used to maintain the min-hash signature of the current window sequence. Each time the sliding window moves forward to the next frame, the min-hash signature of the next window sequence can be obtained through a series of light-weight heap operations. The speedup is due to the fact that there is a substantial amount of overlap content between two adjacent window sequences, which can be reused to generate the next signature. Unlike the histogram pruning algorithm [17], the acceleration of our algorithm is not affected by the similarity or the threshold. With the compact signature representation and efficient signature generation process, the proposed method can be implemented effectively and efficiently.

We use two techniques to resolve the containment relation problem. First, to alleviate the content synchronization problem, we develop a *window length estimation* technique based on the motion movement relation between the query and window sequences. Second, we propose an adaptive threshold metric to assess the containment relation. Broder [5] employed random permutation sketches to estimate the containment relation in document retrieval. However, the estimation between a short document and a relatively larger one becomes unstable. We utilize a *threshold transform* approach to reflect the containment relation, which changes the threshold dynamically based on the query and current window sequences. Experiment results show that window length estimation and threshold transform complement each other and jointly improve the accuracy.

The remainder of this paper is organized as follows. In Section II, we introduce the formulation and the flow of the TLS-based CBVCD task. In Section III, we describe the compact signature and the associated manipulation used to accelerate the TLS process. In Section IV, we present the window length estimation and threshold transform techniques for modeling the containment relation. We discuss the experiment results in Section V. Section VI summarizes our conclusion.

II. The TLS-Based CBVCD Task

The TLS-based CBVCD task is formulated as follows. Let $Q = \{q_i \mid i = 1, 2, \ldots, n_Q\}$ be a query sequence with n_Q frames, where q_i is the i-th query frame; and let $T = \{t_j \mid j = 1, 2, \ldots, n_T\}$ be a target sequence with n_T frames, where t_j is the j-th target frame, and $n_Q << n_T$. A sliding window is used to scan over T to search for a subsequence whose content is identical or similar to Q. Let $W = \{t_j, t_{j+1}, \ldots, t_{j+n_W-1}\}$ be a subsequence of T extracted by a sliding window with n_W frames, denoted as a window sequence. Our goal is to devise an efficient and effective mechanism to measure the similarity between Q and W in T. Any W with similarity higher than a predefined threshold is considered a copy of Q. Figure 2 briefly illustrates the flow of the TLS-based CBVCD task.

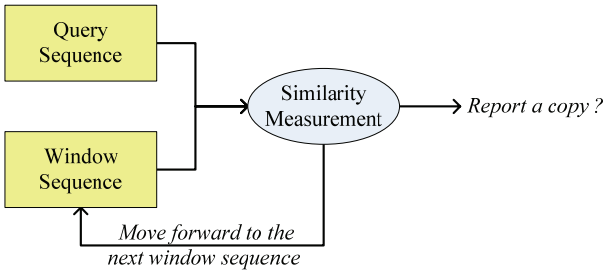


Fig. 2. The flow of the CBVCD task.

III. Compact Signature Manipulation

To address the efficiency issue in matching, a compact signature is proposed based on the *bag-of-words* (BoW) model. The BoW model has been widely used to represent textual features in information retrieval, and has grabbed a great attention in pattern recognition and multimedia retrieval recently. Sivic and Zisserman [29] used *k*-means clustering to train a quantization codebook from a corpus of SIFT descriptors, and the resulting codewords were called visual words. Each SIFT descriptor extracted from a video frame was quantized to the nearest codeword, and an object was represented as a set of visual words, i.e., a BoW. Then, two objects were matched based on the function of term frequencies and inverse document frequencies (i.e., the tf-idf weighting scheme used in information retrieval) of their BoWs. Nistér and Stewénius [24] proposed a vocabulary tree structure built by hierarchical *k*-means clustering. Each SIFT descriptor was quantized through the vocabulary tree to yield a path signature, and an ad-hoc variant of tf-idf weighting scheme was applied to match two objects' BoWs. Zhang *et al*. [35] further aggregated visual word pairs to visual phrases. Jiang and Ngo [16] built the visual word hierarchy by agglomerative clustering. A soft-weighting function, which assigned each SIFT descriptor to multiple visual words "softly", was proposed to assess a visual word's weight in an object. Their experiment results indicated that the hierarchical structure unveiled the relation between visual words, and thus improved the retrieval accuracy. However, in our method, since the proposed min-hash indexing assumes that visual words are generated randomly and are independent

to each other, a flat clustering-based BoW model would be more appropriate than a hierarchical clustering-based one.

Given a SIFT codebook with L codewords, the i-th query frame q_i can be represented by a histogram with L bins as $H_{q_i} = \{h_{q_i,1}, h_{q_i,2}, ..., h_{q_i,l}, ..., h_{q_i,L}\}$, where $h_{q_i,l}$ is the number of q_i's SIFT descriptors classified into the l-th cluster (bin). Q's histogram is denoted as $H_Q = \{h_{Q,1}, h_{Q,2}, \dots, h_{Q,l}, \dots, h_{Q,L}\}$, where $h_{Q,l}$ is calculated by

$$h_{Q,l} = \sum_{i=1}^{n_Q} h_{q_i,l} . \quad (1)$$

For the j-th target frame t_j, its histogram is denoted as $H_{t_j} = \{h_{t_j,1}, h_{t_j,2}, ..., h_{t_j,l}, ..., h_{t_j,L}\}$. The histogram $H_W = \{h_{W,1}, h_{W,2}, \dots, h_{W,l}, \dots, h_{W,L}\}$ for the window sequence W can be constructed by

$$h_{W,l} = \sum_{p=0}^{n_W-1} h_{t_{j+p},l} . \quad (2)$$

H_Q and H_W are modeled in sequence-level BoW forms of Q and W, respectively. Their similarity measurement is defined in the following.

A. Similarity Measurement

The similarity measure between Q and W can be computed by the *Jaccard coefficient* of H_Q and H_W:

$$J(Q,W) = \frac{|H_Q \cap H_W|}{|H_Q \cup H_W|} = \frac{\sum_{l=1}^{L} \min(h_{Q,l}, h_{W,l})}{\sum_{l=1}^{L} \max(h_{Q,l}, h_{W,l})} . \quad (3)$$

The cost of computing the Jaccard coefficient is comprised of **(1)** $O(n_W \cdot L)$ for constructing H_W by summing n histograms of L dimensions, and **(2)** $O(L)$ for calculating the histogram intersection and the union between Q and W; thus, the total cost is $O((n_W+1) \cdot L)$. However, the computational cost for constructing the *next window sequence*'s histogram, denoted as $H_{W'}$, may be much lower. Suppose that the current sliding window is shifted forward one frame so that frame t_j slides out of the window and frame t_{j+n_W} slides into it. Then, the next window sequence will be $W' = \{t_{j+1}, t_{j+2}, ..., t_{j+n_W}\}$ [1], and its histogram $H_{W'}$ can be generated by simply updating the current window sequence's histogram H_W

$$H_{W'} = H_W - H_{t_j} + H_{t_{j+n_W}} . \quad (4)$$

The l-th bin of $H_{W'}$ is updated by $h_{W',l} = h_{W,l} - h_{t_j,l} + h_{t_{j+n_W},l}$, $l = 1, 2, \dots, L$. Since we only need to subtract and add one histogram for each dimension, the cost for constructing $H_{W'}$ is $O(2L)$ rather than $O(n_W \cdot L)$. Therefore, the total cost of computing the Jaccard coefficient between the query and the next window sequence becomes $O(3L)$. The memory space used for each window sequence is L memory cells storing the histogram bin values.

To reduce the computational cost further, we can use an approximate Jaccard coefficient, called *min-hash indexing*. Min-hash indexing is a locality sensitive hashing (LSH) technique that can solve the nearest neighbor search problem efficiently [1]. The basic concept is that, given a binary feature vector, we randomly permute its indices and record the position where the first "1" occurs as the min-hash. However, for the sake of efficiency, Cohen *et al*. [10] proposed associating each element with a hash value and taking the minimum as the min-hash value. In this study, we adopt the min-hash concept to represent a video sequence.

Assume the above-mentioned histogram construction is an independent and random process, which acts as a hash function that assigns each SIFT descriptor to a histogram bin. Therefore, we can treat the index number of the bin as the hash value. For the query and window sequences, we take their indices of the first k non-zero histogram bins as the hash values, which form the signatures S_Q and S_W respectively:

$$\begin{aligned} S_Q &= \min_k(\{l \mid h_{Q,l} > 0, l = 1, 2, \dots, L\}), \\ S_W &= \min_k(\{l \mid h_{W,l} > 0, l = 1, 2, \dots, L\}), \end{aligned} \quad (5)$$

where $\min_k(A)$ returns the k smallest values of set A in ascending order. If the size of A is not larger than k, $\min_k(A)$ returns A in ascending order. The min-hash similarity between Q and W is estimated through the following expression [7]:

$$M(Q,W) = |S_Q \cap S_W| / k . \quad (6)$$

B. Signature Manipulation

A fast approximation approach to generate S_w without histogram construction is proposed. For each target frame t_j, we maintain at most g min-hash values, $g < k$,:

$$S_{t_j} = \min_g(\{l \mid h_{t_j,l} > 0, l = 1, 2, \dots, L\}) . \quad (7)$$

Then S_W is approximated by:

$$S_W^* = \min_k(\{S_{t_j}, S_{t_{j+1}}, ..., S_{t_{j+n_W-1}}\}) . \quad (8)$$

S_W^* can be implemented efficiently by using a heap structure and associated operations. A heap is a complete binary tree and its operations, i.e., sorting, insertion, and deletion, can be implemented efficiently in an array. In this study, we employ the min-heap in which, for every node X other than the root, the value of X cannot be smaller than that of X's parent node.

Let W be the current window sequence. Our task is to generate the min-hash signature and construct the corresponding heap for W. First, we define the abstract data types used for heap manipulation:

```
define type Heap {
  HeapNode node[g×n_w];
  function insert(min_hash_value, target_frame_index);
  function delete(heap_index);
  function extractMin();
  function heapify(heap_index);
};
define type HeapNode {
  int min_hash_value;
  int target_frame_index;
};
int inverted_index[g×n_w];
int min_hash_signature[k];
```

[1] Here, we simplify the discussion by assuming that the window lengths of W and W' are the same and identical to n_W. In Section IV.A, we estimate the window length according to the video content.

The data type `Heap` contains an array of $g \times n_w$ nodes, each of which is a type of `HeapNode`. Suppose that l_q, the q-th min-hash value of S_{t_j}, $t_j \in W$, is to be stored in the p-th heap node. Then we set `node[`p`].min_hash_value` = l_q and `node[`p`].target_frame_index` = j. At the same time, we set `inverted_index[`$g \times rem(j, n_w)+q$`]` = p, where $rem(j, n_W)$ returns the remainder of j divided by n_W. `min_hash_signature` is an array of k integers denoted as the min-hash signature of W. Note that there are four basic heap functions defined in the `Heap` type. Due to space limitations, we do not include the algorithms in this paper. Readers may refer to data structure or algorithm textbooks for details [11].

The steps of heap construction and min-hash signature generation for W are detailed in Algorithm 1. The algorithm inserts all min-hash values of W into the heap, and then extracts the k minima from the heap to generate a sorted min-hash signature. The time complexity of Algorithm 1 is $O((g \cdot n_W) \cdot \lg(g \cdot n_W) + k \cdot \lg(g \cdot n_W))$, since there are totally $g \cdot n_W$ min-hash values to be inserted into the heap and k min-hash values to be extracted from the heap, and the time complexity of each insertion or extraction operation is $O(\lg(g \cdot n_W))$. Therefore, the cost of computing the min-hash similarity between Q and W becomes $O((g \cdot n_W) \cdot \lg(g \cdot n_W) + k \cdot \lg(g \cdot n_W) + k)$, which would be less than $O((n_W+1) \cdot L)$ for computing the Jaccard similarity with a suitable choice of g and k that are much smaller than L. Figure 3 gives an example to illustrate Algorithm 1. In this example, the current window sequence W contains the first three target frames; each frame maintains two min-hash values (i.e., $g = 2$); and the min-hash signature is a three-dimensional (i.e., $k = 3$) vector. The top row of Figure 3 shows that we insert a total of six min-hash values to an initially empty heap. The bottom row shows that the three smallest hash values are extracted from the heap as the min-hash signature of W.

For the next window sequence W', its signature $S^*_{W'} = \min_k(\{S_{t_{j+1}}, S_{t_{j+2}}, \ldots, S_{t_{j+n_w}}\})$ can be obtained from the heap by deleting the previous frame elements and inserting the new frame elements. Algorithm 2 describes how the heap is updated to generate the min-hash signature of W'. We illustrate Algorithm 2 in Figure 4 using the example in Figure 3. The next window sequence W' excludes the first frame and includes the fourth frame. The top row of Figure 4 shows the result after deleting and inserting the corresponding min-hash values in the heap, while the bottom row shows the resulting min-hash signature of W'.

The time complexity of Algorithm 2 is analyzed as follows. The time complexity of Step (1), which deletes g min-hash elements of S_{t_j}, is $O(g \cdot \lg(g \cdot n_W))$. Similarly, the time complexity of Step (2) is $O(g \cdot \lg(g \cdot n_W))$. The time complexity of Steps (3) and (4) is $O(g \cdot \lg(g \cdot n_W))$, since the heapify and extractMin functions are executed at most g times. The time complexity of the sorting algorithm in Step (5) is $O(k \cdot \lg(k))$. Hence, the total time complexity of Algorithm 2 is $O(3g \cdot \lg(g \cdot n_W) + k \cdot \lg(k))$, which is less than that of Algorithm 1.

Therefore, after the first heap has been constructed by Algorithm 1, the min-hash signatures of the subsequent window sequences can be generated more efficiently by Algorithm 2. The cost of computing the min-hash similarity between Q and W' becomes $O(3g \cdot \lg(g \cdot n_W) + k \cdot \lg(k) + k)$, which is less than $O(3L)$ for computing the Jaccard similarity.

Algorithm 1. Heap construction and min-hash signature generation for the current window sequence W

(1) Declare `A` to be the type of `Heap`.

(2) For each min-hash value $l \in S_{t_j}$ and $t_j \in W$, execute `A.insert(`l, j`)`.

(3) Execute u = `A.extractMin()` k times and store each u in `min_hash_signature` sequentially to generate a sorted min-hash signature of W.

Algorithm 2. Heap updating and min-hash signature generation for the next window sequence W'

(1) For the q-th min-hash value l_q of S_{t_j}, $q = 1, 2, \ldots, g$, execute `A.delete(inverted_index[`$g \times \text{rem}(j, n_w)+q$`])`.

(2) For each min-hash value $l \in S_{t_{j+n_w}}$, execute `A.insert(`$l, j+n_W$`)`.

(3) Let u be the smallest value in heap `A` and v be the largest value in `min_hash_signature`. If $u < v$, then swap u and v and execute `A.heapify(1)`. Repeat this step until $u \geq v$.

(4) For each empty slot of `min_hash_signature` where the min-hash value is deleted in Step (1), Execute u = `A.extractMin()` and store u in the empty slot.

(5) Sort `min_hash_signature` to generate a sorted min-hash signature of W'.

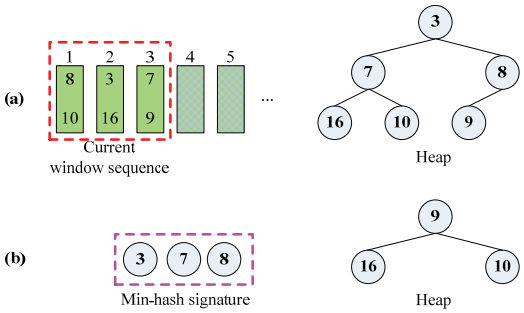


Fig. 3. An example of Algorithm 1: (a) heap construction of Step (2); (b) signature generation of Step (3).

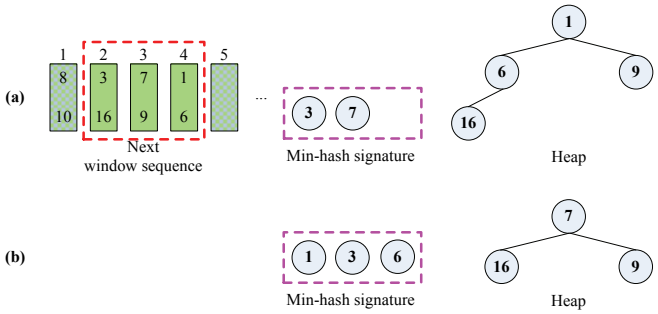


Fig. 4. An example of Algorithm 2: (a) heap updating of Steps (1) and (2); (b) signature generation of Steps (3), (4), and (5).

IV. Containment Relation Modeling

To address the containment relation problem, we incorporate two techniques in TLS, namely, window length estimation and threshold transform, as detailed in the following.

A. Window Length Estimation

To alleviate the content synchronization problem of the query and window sequences, we estimate an appropriate length of the sliding window according to the sequences' content. The estimation is based on the intuition that the movement distance of a video block should be consistent despite video transformation and editing. We adopt Hoad and Zobel's signature, called the centroid-based signature [15], as the movement distance. In their implementation, the lightest and darkest 5% of pixels in the i-th frame are located and their average coordinates are respectively computed as:

$$\begin{aligned} \overline{P}_i^{lightest} &= avg(\{(x,y) \mid \arg\max_{(x,y)}{}_{5\%}(I_{i,(x,y)})\}) \\ \overline{P}_i^{darkest} &= avg(\{(x,y) \mid \arg\min_{(x,y)}{}_{5\%}(I_{i,(x,y)})\}) \end{aligned} \quad (9)$$

where $I_{i,(x,y)}$ is the intensity of pixel location (x,y) of the i-th frame; $\max_{5\%}(A)$ and $\min_{5\%}(A)$ return the maximum and minimum 5% subset of A, respectively; and $avg(A)$ returns the average of A. The Euclidean distance between the average coordinates of adjacent frames is calculated and normalized based on the frames' resolution. The centroid-based signature of the i-th frame, denoted as d_i, is thus computed by

$$d_i = NE(\overline{P}_i^{lightest}, \overline{P}_{i+1}^{lightest}) + NE(\overline{P}_i^{darkest}, \overline{P}_{i+1}^{darkest}), \quad (10)$$

where $NE(x, y)$ returns the normalized Euclidean distance between x and y. Hoad and Zobel showed that the centroid-based signature is insensitive to several types of video transformation and editing, and the computational cost is lower than that of conventional motion estimation algorithms.

Let d_{q_i} and d_{t_j} be the movement distances (i.e., centroid-based signatures) of query frame q_i and target frame t_j, respectively. The movement distance of query sequence Q, denoted as d_Q, is computed by summing all query frames' movement distances:

$$d_Q = \sum_{i=1}^{n_Q} d_{q_i} \cdot \quad (11)$$

Suppose that t_j is the first frame of window sequence W. The window length of W, n_W, should satisfy the following criterion:

$$\sum_{p=1}^{n_W-1} d_{t_{j+p-1}} \le d_Q < \sum_{p=1}^{n_W} d_{t_{j+p-1}} = d_W \cdot \quad (12)$$

In other words, we want to find the window sequence $W = \{t_j, t_{j+1}, \ldots, t_{j+n_w-1}\}$ with n_W frames whose movement distance (i.e., d_W) is the closest to that of query sequence Q (i.e., d_Q). n_W is estimated by inserting target frames t_j, t_{j+1}, t_{j+2}, … sequentially to an initially empty set W until the criterion in (12) is met. The time cost of finding n_W is approximately $O(n_W)$.

Estimating the length of the next window sequence can be even more efficient. Recall that $W = \{t_j, t_{j+1}, \ldots, t_{j+n_W-1}\}$ and $W' = \{t_{j+1}, t_{j+2}, \ldots, t_{j+n_{W'}}\}$. Initially, we assume that $n_{W'}$ is equal to n_W, and calculate the movement distance of W' by

$$d_{W'} = d_W - d_{t_j} + d_{t_{j+n_w}}, \quad (13)$$

Then, we compare the movement distances of Q and W'. If $d_{W'}$ is greater (resp. smaller) than d_Q, $n_{W'}$ will be decreased (resp. increased) until $d_{W'}$ is smaller (resp. greater) than d_Q. In most cases, we only require a few frames to update $d_{W'}$ since the movement distance of the next window sequence usually alters slightly; the estimation is thus very efficient.

B. Threshold Transform

The purpose of threshold transform is to dynamically change the detection threshold according to the search context. That is, the threshold is updated to capture the containment relation of the query and current window sequences. Threshold transform is derived based on the connection between the containment coefficient and the Jaccard coefficient. The containment coefficient [5] between Q and W is defined as

$$C(Q,W) = \frac{|H_Q \cap H_W|}{\min(|H_Q|, |H_W|)}, \quad (14)$$

which can be transformed into the form of the Jaccard coefficient. Since $|H_Q \cup H_W| = |H_Q| + |H_W| - |H_Q \cap H_W|$, by replacing $|H_Q \cup H_W|$ in the denominator of (3) with $|H_Q| + |H_W| - |H_Q \cap H_W|$, we obtain

$$|H_Q \cap H_W| = \frac{(|H_Q| + |H_W|) \cdot J(Q,W)}{1 + J(Q,W)}. \quad (15)$$

Therefore, (14) can be rewritten as

$$C(Q,W) = \frac{(|H_Q| + |H_W|)}{\min(|H_Q|, |H_W|)} \cdot \frac{J(Q,W)}{1 + J(Q,W)}. \quad (16)$$

Since the min-hash similarity is an approximation of the Jaccard coefficient, hereafter, we replace $J(Q, W)$ with $M(Q, W)$ and continue the discussion. Clearly, $M(Q, W) \ge \theta_{\mathbf{MH}}$ if and only if $\frac{M(Q,W)}{1+M(Q,W)} \ge \frac{\theta_{\mathrm{MH}}}{1+\theta_{\mathrm{MH}}}$; therefore, we have the following inequality condition:

$$M(Q,W) \ge \theta_{\mathrm{MH}} \quad \Leftrightarrow \quad C(Q,W) \ge \frac{(|H_Q| + |H_W|)}{\min(|H_Q|, |H_W|)} \cdot \frac{\theta_{\mathrm{MH}}}{(1+\theta_{\mathrm{MH}})} \quad (17)$$

Let $\theta_{\mathbf{CC}} = \frac{(|H_Q| + |H_W|)}{\min(|H_Q|, |H_W|)} \cdot \frac{\theta_{\mathbf{MH}}}{(1+\theta_{\mathbf{MH}})}$. The above Equation can be rewritten as

$$C(Q,W) \ge \theta_{\mathbf{CC}} \quad \Leftrightarrow \quad M(Q,W) \ge \frac{\theta_{\mathbf{CC}} \cdot \min(|H_Q|, |H_W|)}{|H_Q| + |H_W| - \theta_{\mathbf{CC}} \cdot \min(|H_Q|, |H_W|)} \quad (18)$$

The detection criterion thus becomes: if the min-hash similarity $M(Q, W)$ exceeds the threshold derived in (18), the containment coefficient $C(Q, W)$ will be greater than $\theta_{\mathbf{CC}}$.

Equation (18) shows that the threshold for $M(Q, W)$ changes dynamically according to the cardinalities of the query and window sequences, i.e., $|H_Q|$ and $|H_W|$. The threshold transform process is very efficient, since only $|H_W|$ has to be re-calculated for each window sequence.

Note that the proposed similarity measure does not consider the temporal relation between Q and W. It would increase *false positives*, which are negative examples incorrectly labeled as positives. To remedy this, we apply the spatio-temporal matching technique proposed by Chiu *et al*. [7] to further examine a window sequence whose min-hash similarity exceeds the threshold. Our previous study showed that this technique is very effective for removing false positives and it can be completed in a very short constant time.

V. Experiments

A. Dataset

We evaluated the proposed method on the MUSCLE-VCD-2007 dataset [23], which is the benchmark dataset of the TRECVID video copy detection task [31]. The dataset was augmented with video clips covering a wide variety of content to yield a target dataset of 50 hours. All the movie clips were converted into the following uniform format: MPEG-1, 320×240 pixels, and 30 frames per second (fps). The eight types of video transformation and editing listed in Table I were applied to 31 video sequences, each of thirty seconds duration, excerpted from the ST2 movie clip set of the MUSCLE-VCD-2007 dataset. The resulting 248 (31×8) sequences were used as queries to detect the corresponding segments in the 50-hour target sequence dataset.

Although the frame rate of the video dataset is 30 fps, it is not necessary to use such a high frame rate for copy detection because many frames are near-duplicate. We re-sampled the video dataset to be 2 fps in the experiment. A SIFT codebook with size $L = 1024$ was used in our BoW model.

B. Parameter Configuration

1) Window length Estimation

We implement two window length estimation approaches; one uses the whole frame region to extract the movement distance, and the other only uses 36% (60% length × 60% width) of the central frame region. The objective of the latter is to avoid the effects caused by cropping or zoom-in. The experiment was conducted on the query dataset. Since each query sequence was derived from a 30-second source sequence, the ideal window length estimation should be $n_W = 60$ (30 sec. × 2 fps) irrespective of the type of video transformation and editing. As shown in Table II, for the preserved-frame-region and discarded-frame-region categories, the central region approach yields very good estimations compared to the whole region approach. For the changed-frame-number category, the query length of the slow motion (resp. fast forward) version is n_Q = 120 (resp. 30). If we were to employ a fixed-length sliding window, the contents of the query and window sequences do not synchronize because the two sequences are in different speed scales. Table II also shows that, under the two window length estimation approaches, although the estimation in the changed-frame-number category is not as good as that in other categories, it is much better than fix-length method. The results demonstrate that 1) both approaches alleviate the content synchronization problem to some extent; and 2) the central region approach outperforms the whole region approach. Hence, in the subsequent experiments, we used the central region approach for window length estimation.

TABLE I
Video Transformation and Editing Applied in the Experiments

Type	Description
Brightness	Enhance the brightness by 20%.
Compression	Set the compression quality at 50%.
Noise	Add random noise (10%).
Resolution	Change the frame resolution to 120×90 pixels.
Cropping	Crop the top and bottom frame regions by 10% each.
Zoom-in	Zoom in the frame 10%.
Slow motion	Halve the video speed.
Fast forward	Double the video speed.

TABLE II
The Results of Window Length Estimation

	Whole region		Central region	
	mean	*std.*	*mean*	*std.*
Brightness	44.68	15.50	59.73	4.36
Compression	42.56	12.36	60.98	4.62
Noise addition	39.39	9.24	55.00	5.37
Resolution	47.20	16.73	68.90	8.11
Cropping	34.33	9.22	60.88	4.75
Zoom-in	50.76	9.09	67.15	5.62
Slow motion	68.67	13.76	80.64	9.11
Fast forward	31.10	13.90	46.01	7.40

2) Min-hash signature

Recall that, in Section III, the min-hash signature is generated based on k and g, which are the maximum numbers of min-hash values required to represent a sequence and a frame respectively. In this subsection, we discuss the impact of k and g on the min-hash signature.

For each query sequence Q in the query dataset, we generated a min-hash signature S_Q and an approximate min-hash signature S_Q^* with various combinations of (k, g), where $k \in \{10, 20, \ldots, 100\}$ and $g \in \{1, 2, \ldots, 10\}$. Figure 5(a) shows the similarity between S_Q and S_Q^* calculated by their Jaccard coefficient. For the case where $k = 50$ and $g = 6$, the similarity between S_Q and S_Q^* is 91.56%. It is worth noting that, under this configuration, the computation time of S_Q^* was reduced by 92.93% compared with that of S_Q. That is, with a suitable choice of k and g, S_Q^* provides a highly satisfactory approximation to S_Q at a substantially lower computational cost.

Next, we investigated the relation between k, g, and the detection accuracy. The accuracy metric is defined by the *recall* and *precision* rates:

$$recall = TP / (TP + FN), \quad (19)$$

$$precision = TP / (TP + FP), \quad (20)$$

where True Positives (TP) refer to the number of positive examples correctly labeled as positives; False Negatives (FN) refer to the number of positive examples incorrectly labeled as negatives; and False Positives (FP) refer to the number of negative examples incorrectly labeled as positives. Figure 5(b)

shows the accuracy variation of each (k, g) pair. The accuracy variation is defined as

$$accuracy\_variation_{(k,g)} = |\, recall_{(k,g)} - recall_{(k,g-1)} \,| + |\, precision_{(k,g)} - precision_{(k,g-1)} \,|. \quad (21)$$

It is clear that, as g grows, the curve of k approaches zero and becomes more stable. Besides, the curve of a larger k needs a larger g to get stable. Without loss of generality, we selected the following (k, g) configurations for each k: (10, 4), (20, 4), (30, 5), (40, 5), (50, 6), (60, 6), (70, 7), (80, 7), (90, 7), and (100, 8). Table III lists the recall and precision rates of our method, derived by combining all the proposed techniques under various k. θ_{MH} and θ_{CC} were set at 0.7. Overall, the proposed method yields a consistent performance when $k \geq 30$. The insensitivity to k is a good characteristic of the min-hash signature; we do not need to pay much attention to tune the signature length. In the subsequent experiments, we use (k, g) = (50, 6).

C. Accuracy Comparison

We compared the accuracy of six methods:

(1) The min-hash signature (abbreviated as "MH"),
(2) MH + window length estimation ("MH+WE"),
(3) MH + threshold transform ("MH+TT"),
(4) MH + WE + TT ("ALL"),
(5) Hoad and Zobel's method [15] ("HZ"), and
(6) Chum *et al*'s method [9] ("CHUM").

Our methods (Methods (1)-(4)) evaluated different combinations of the proposed techniques. We incorporated the spatio-temporal matching technique in our methods to filter out false positives, as described in Section IV.B.

Hoad and Zobel's method [15] is a TLS method that uses a fixed-length sliding window. It is implemented as follows. From each frame, the color-shift signature is extracted by using 16 bins for each of the three color channels in YCbCr color space; and the Manhattan distance is used to calculate the histogram distance of two adjacent frames. The centroid-based signature, described in Section IV.A, and the color-shift signature are combined as a two-dimensional feature vector to represent a frame, and an approximate string matching algorithm is applied for copy detection.

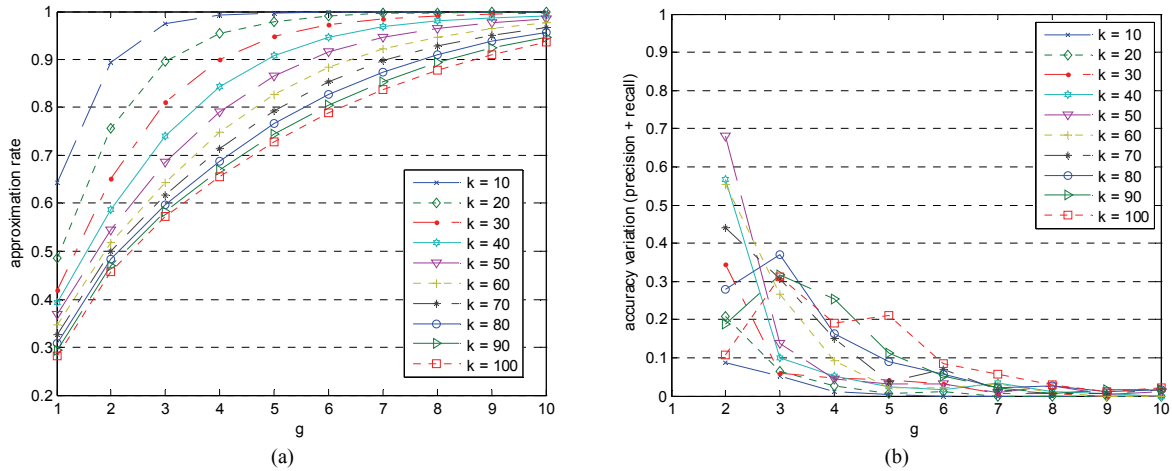
Chum *et al*'s method [9] is based on the BoW model and the min-hash concept. First, it extracts and quantizes SIFT descriptors for each frame in the same way as our method. Then, 64 min-hash sketches, each of which is a 3-tuple min-hash set, are generated as the frame representation. In the detection step, each query frame searches for its near-duplicates. The frames whose sketch similarities are greater than a predefined threshold (35% in their paper) are considered near-duplicates, and each one votes for its corresponding shot in the video dataset.

Table IV lists the recall and precision rates of the six compared methods for eight types of transformation and editing.

1) Preserved-frame-region transformation and editing

This category includes brightness enhancement, compression, noise addition, and frame resolution change. Methods (2)-(4) generally perform well, except for the frame resolution change type in terms of the recall rate. We observe that there is a discrepancy between the SIFT descriptors extracted from the same content with different frame resolutions (320×240 vs. 120×90). The discrepancy might induce a mismatch between the source and its copy, and thus increase false negatives. On the other hand, Hoad and Zobel's method performs well for the compression, noise addition, and frame resolution change types. However, it does not perform as well for the brightness enhancement type because its color-shift signature counts the histograms of color channels, which vary widely after applying brightness enhancement.

Chum *et al*.'s method yields a very bad precision rate because it only counts the frame space similarity without considering the temporal relation; hence, a lot of false positives are retrieved from the dataset. In fact, without the spatio-temporal matching technique, our methods would face the same problem.



(a) (b)

Fig. 5. (a) The approximation rates of different (k, g) pairs; (b) the accuracy variations of different (k, g) pairs.

TABLE III
THE RECALL AND PRECISION RATES OF THE PROPOSED METHOD UNDER VARIOUS *K*

		k=10	*k*=20	*k*=30	*k*=40	*k*=50	*k*=60	*k*=70	*k*=80	*k*=90	*k*=100
Brightness	**R**	0.9355	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	P	0.9355	0.9118	0.9118	0.8857	0.9394	0.9394	0.9118	0.9394	0.9394	0.9686
Compression	**R**	0.7742	0.9677	1.0000	0.9677	0.9677	0.9677	0.9677	0.9677	0.9677	0.9677
	P	0.9231	0.9091	0.8577	0.8824	0.9091	0.8824	0.8824	0.8571	0.8571	0.8571
Noise	**R**	0.7742	0.9355	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9677	0.9677
	P	0.9231	0.9355	0.9394	0.9394	1.0000	0.9394	0.9118	0.9118	0.9091	0.9091
Resolution	**R**	0.6452	0.7419	0.7097	0.8387	0.8065	0.7742	0.8387	0.7419	0.7742	0.7742
	P	0.9091	0.9200	0.9565	0.9286	0.9259	0.9231	0.9286	0.9200	0.9231	0.9231
Cropping	**R**	0.9032	0.9355	0.9355	0.9677	0.9677	0.9677	0.9677	1.0000	0.9355	0.9355
	P	0.9655	0.9667	0.9036	0.9375	0.9375	0.9375	0.9375	0.9412	0.9355	0.9355
Zoom-in	**R**	0.7097	0.8710	0.8387	0.8710	0.9355	0.9355	0.9355	0.9677	0.9677	0.8710
	P	0.9167	0.9643	0.9286	0.9310	0.9355	0.9355	0.9355	0.9375	0.9375	0.9000
Slow motion	**R**	0.8387	0.9032	0.9355	0.8710	0.9355	0.9032	0.9677	0.9355	0.9677	0.9355
	P	1.0000	0.9655	0.9355	0.8710	1.0000	0.9655	0.9677	0.9355	0.9091	0.9063
Fast forward	**R**	0.9677	1.0000	1.0000	0.9677	1.0000	0.9677	0.9677	0.9677	0.9677	0.9355
	P	0.8824	0.8649	0.8649	0.8333	0.8378	0.8333	0.8333	0.8108	0.8108	0.7838

TABLE IV
THE RECALL AND PRECISION RATES OF THE SIX METHODS FOR VIDEO TRANSFORMATION AND EDITING

		(1) *MH*	(2) *MH+WE*	(3) *MH+TT*	(4) *ALL*	(5) *HZ*	(6) *CHUM*
Brightness	**R**	0.9459	0.9459	1.0000	1.0000	0.8065	0.8710
	P	0.9476	0.9476	0.9261	0.9394	0.8065	0.1627
Compression	**R**	0.9189	0.9189	0.9729	0.9677	0.9355	0.9032
	P	0.9460	0.9729	0.8803	0.9091	0.9355	0.1228
Noise	**R**	0.8649	0.9189	0.8649	1.0000	0.9032	0.8065
	P	0.9422	1.0000	0.8691	1.0000	0.9032	0.1042
Resolution	**R**	0.5676	0.5946	0.6756	0.8065	0.9032	0.6129
	P	0.9238	0.9271	0.9033	0.9259	0.9032	0.0465
Cropping	**R**	0.8649	0.8648	0.9677	0.9677	0.3226	0.7742
	P	0.9162	0.9711	0.8571	0.9375	0.3226	0.1127
Zoom-in	**R**	0.8378	0.8378	0.9032	0.9355	0.8387	0.6774
	P	0.9402	0.9690	0.8750	0.9355	0.8387	0.0824
Slow motion	**R**	0.8108	0.9189	0.8379	0.9355	0.0645	0.9355
	P	0.9356	1.0000	0.9379	1.0000	0.0645	0.1835
Fast forward	**R**	0.8919	0.9459	0.8919	1.0000	0.1935	0.9355
	P	0.9189	0.9476	0.7905	0.8378	0.1935	0.0755

2) Discarded-frame-region transformation and editing

This category includes frame cropping and zoom-in. Hoad and Zobel's method performs poorly for frame cropping. This is because frame cropping, which yields black borders on the top and bottom frame regions (see Figure 1), would vary the source's color-shift and centroid-based signatures substantially. Overall speaking, Hoad and Zobel's method does not perform as good as our methods due to the limited capability of the global descriptors. Our methods using the SIFT descriptors are less affected in this category.

Comparing Methods (1) and (3), we observe that applying threshold transform improves the recall rate noticeably. Although the precision rate declines because more false positives satisfy the transformed threshold, Method (4) shows generally better accuracy than Methods (2) and (3). Thus, we consider the proposed threshold transform technique is helpful to reflect the containment relation between two video sequences.

3) Changed-frame-number transformation and editing

This category includes slow motion and fast forward. Hoad and Zobel's method performs poorly in this category due to the content synchronization problem. Even the approximate string matching scheme can not compensate for the large discrepancy between the query and window contents. Another reason is that their method produces a very different signature pattern from the original in this category.

Methods (2) and (4) yield better precision and recall rates than Method (1), which shows that the window length estimation technique can alleviate the content synchronization problem induced in this category.

4) Summary

The results of the above experiments demonstrate that the proposed techniques improve the detection accuracy; window length estimation alleviates the content synchronization problem, and threshold transform captures the containment relation in similarity measurement. Method (4), which integrates all these techniques, yields a consistently robust performance for the three categories of video transformation and editing. The excellent results are achieved with a very compact min-hash signature; specifically, a 50-dimensional signature (i.e., $k = 50$) is satisfactory for representing a 60-frame sequence.

D. Speed Comparison

We implemented four TLS-based methods for speed comparison:

(1) the SIFT histogram ("SH"),
(2) the min-hash signature ("MH"),
(3) MH + histogram pruning ("MH+HP"), and
(4) MH + heap manipulation ("MH+HM").

Method (1) used the SIFT histogram (defined in (1) and (2)) of the 1024-dimensional vector as the video frame feature, while Method (2) employed the proposed k-dimensional min-hash signature, with $k = 10, 20, \ldots, 100$. The two methods applied a conventional TLS scheme that scanned the target sequence linearly with a sliding window without any speedup technique. Method (3) integrated Kashino *et al.'s* histogram pruning algorithm [17] into Method (2); and Method (4) integrated the proposed heap manipulation into Method (2). All four methods were implemented with window length estimation, threshold transform, and spatio-temporal matching.

The program was implemented in C++, and run on a PC with a 2.8 GHz CPU and 2GB RAM. The time costs of Methods (1)-(4) with respect to k are shown in Figure 6. They are measured by using a thirty-second query sequence to scan the fifty-hour target dataset. For example, Methods (1)-(4) take 69.997, 20.108, 3.317, and 1.375 seconds, respectively, at k = 50. Interestingly, the time required by Method (3) first decreases when k increases from 10 to 40, and then increases when k further increases to 100 (except for k = 70). Usually, although a higher k increases the computational cost in matching, it yields a lower min-hash similarity score and thus magnifies the difference between the similarity score and the predefined threshold. Hence, a larger number of frames would be skipped by the histogram pruning algorithm. Method (3) reduces the computation time by 50-85% compared with Method (2), and the reduction approximately corresponds to the number of frames that are skipped. Method (4) has the most efficient detection process among the compared methods. It is clear that heap manipulation can lower k's influence on the time cost, comparing Method (4) with Method (2).

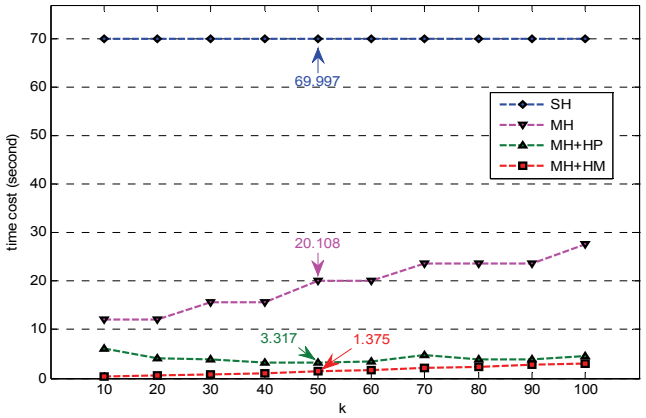


Fig. 6. The time costs of the four methods with respect to k.

E. Discussion on Scalability

Dealing with a large-scale dataset has attracted a great interest in information retrieval and data mining. To address the scalability issue in CBVCD, there have been some studies on the compact feature representation and efficient matching strategy [21][25][30]. Our experiment results have demonstrated the compactness and efficiency of the proposed sequence-level signature and associated manipulation. In particular, as shown in Figure 6, the time cost of the min-hash-based signature grows only linearly with k rather than exponentially. Therefore, the proposed method has good scalability to the high dimensionality, which is usually accompanied by the large-scale dataset. In addition, the nature of TLS makes it apparently scalable to deal with the large-scale dataset, as discussed in our related work.

One potential direction to further improve the scalability in CBVCD is distributed computing. Recently, Dean and Ghemawat [12] introduced a programming model called MapReduce, which provides a simple but powerful interface to realize parallelization, fault-tolerance, data distribution, and load balancing of large-scale computations. The MapReduce model is implemented by the map and reduce functions. The model can be applied in our method in the following way. The map function emits the window sequence that matches the query sequence as the intermediate data, and then the reduce function outputs the identity of the intermediate data. With the MapReduce utility, we can use PC clusters to deal with the large-scale dataset effectively.

VI. Conclusion

In this paper, we have proposed a novel TLS method for efficient and effective content-based video copy detection. To accelerate the search process, the method integrates a compact signature representation of a video sequence based on the min-hash theory and an efficient signature generation process based on the heap structure and operations. To improve the detection accuracy, the method integrates window length estimation and threshold transform to model the containment relation between the source and copy sequences. The proposed method is robust against various types of video transformation and editing, and the computational cost is low compared to existing methods.

Acknowledgment

The authors would like to thank the anonymous reviewers for their thoughtful comments and suggestions that have advanced the quality of this article.

References

[1] A. Andoni, and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimension," *Comm. ACM*, Vol. 51, No. 1, pp. 117-122, 2008.
[2] A. Basharat, Y, Zhai, and M. Shah, "Content based video matching using spatiotemporal volumes," *Comput. Vis. Image Understand.*, Vol. 110, No. 3, pp. 360-377, 2008.
[3] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "SURF: speeded up robust features," *Comput. Vis. Image Understand.*, Vol. 110, No. 3, pp. 346-359, 2008.
[4] S. Brin, J. Davis, and H. Garcia-Molina, "Copy detection mechanisms for digital documents," *Proc. ACM Int'l Conf. Management of Data (SIGMOD)*, San Jose, USA, May. 22-25, 1995.

[5] A. Z. Broder, "On the resemblance and containment of documents," *Proc. Int'l Conf. Compression and Complexity of Sequences*, Salerno, Italy, Jun. 11-13, 1997.
[6] M. Cherubini, R. d. Oliveira, and N. Oliver, "Understanding near-duplicate videos: a user-centric approach," *Proc. ACM Int'l Conf. Multimedia (ACM-MM)*, Beijing, China, Oct. 19-24, 2009.
[7] C. Y. Chiu, H. M. Wang, and C. S. Chen, "Fast min-hashing indexing and robust spatio-temporal matching for detecting video copies," *ACM Trans. on Multimedia Comput. Comm. Applications*, Vol. 6, No. 2, pp. 10:1-23, 2010.
[8] C. Y. Chiu, C. S. Chen, and L. F. Chien, "A framework for handling spatiotemporal variations in video copy detection," *IEEE Trans. on Circuits Syst. Video Technol.*, Vol. 18, No. 3, pp. 412-417, 2008.
[9] O. Chum, J. Philbin, M. Isard, and A. Zisserman, "Scalable near identical image and shot detection," *Proc. ACM Int'l Conf. Image and Video Retrieval (CIVR)*, Amsterdam, The Netherlands, Jul. 9-11, 2007.
[10] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. Ullman, and C. Yang, "Finding interesting associations without support pruning," *Proc. IEEE Int'l Conf. Data Engineering (ICDE)*, Feb. 28-Mar. 3, 2000.
[11] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT Press, 1996.
[12] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Comm. ACM*, Vol. 51, No. 1, pp. 107-113, 2008.
[13] L. Y. Duan, J. Wang, Y. Zheng, J. S. Jin, H. Lu, and C. Xu, "Segmentation, categorization, and identification of commercials from TV streams using multimodal analysis," *Proc. ACM Int'l Conf. Multimedia (ACM-MM)*, pp. 201-210, Santa Barbara, USA, Oct. 23-27, 2006.
[14] M. Henzinger, "Finding near-duplicate web pages: a large-scale evaluation of algorithms," *Proc. ACM Int'l Conf. Information Retrieval (SIGIR)*, Seattle, USA, Aug. 6-11, 2006.
[15] T. C. Hoad and J. Zobel, "Detection of video sequence using compact signatures," *ACM Trans. on Inform. Syst.*, Vol. 24, No. 1, pp. 1-50, 2006.
[16] Y. G. Jiang and C. W. Ngo, "Visual word proximity and linguistics for semantic video indexing and near-duplicate retrieval," *Comput. Vis. Image Understand.*, Vol. 113, No. 3, pp. 405-414, 2009.
[17] K. Kashino, T. Kurozumi, and H. Murase, "A quick search method for audio and video signals based on histogram pruning," *IEEE Trans. Multimedia*, Vol. 5, No. 3, pp. 348-357, 2003.
[18] C. Kim and B. Vasudev, "Spatiotemporal sequence matching for efficient video copy detection," *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 15, No. 1, pp. 127-132, 2005.
[19] A. Kimura, K. Kashino, T. Kurozumi, and H. Murase, "A quick search method for audio signals based on a piecewise linear representation of feature trajectories," *IEEE Trans. Audio, Speech, and Language Process.*, Vol. 16, No. 2, pp. 396-407, 2008.
[20] J. Law-To, O. Buisson, V. Gouet-Brunet, and N. Boujemaa, "Robust voting algorithm based on labels of behavior for video copy detection," *Proc. ACM Int'l Conf. Multimedia (ACM-MM)*, pp. 835-844, Santa Barbara, USA, Oct. 23-27, 2006.
[21] Z. Liu, T, Liu, D. Gibbon, and B Shahraray, "Effective and scalable video copy detection," *Proc. ACM Int'l Conf. Multimedia Information Retrieval (MIR)*, Philadelphia, USA, Mar. 29-31, 2010.
[22] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Inter. J. of Comput. Vision*, Vol. 60, No. 2, pp. 91-110, 2004.
[23] MUSCLE-VCD-2007, http://www-rocq.inria.fr/imedia/civr-bench/index.html
[24] D. Nistér and H. Stewénius, "Scalable recognition with a vocabulary tree," *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition (CVPR)*, New York, USA, Jun. 17-22, 2006.
[25] S. Poullot, M. Crucianu, and O. Buisson, "Scalable mining of large video databases using copy detection," *Proc. ACM Int'l Conf. Multimedia (ACM-MM)*, pp. 61-70, Vancouver, Canada, Oct. 26-31, 2008.
[26] H. T. Shen, B. C. Ooi, X. Zhou, and Z. Huang, "Towards effective indexing for very large video sequence database," *Proc. ACM Int'l Conf. Management of Data (SIGMOD)*, pp. 730-741, Baltimore, USA, Jun. 14-16, 2005.
[27] H. T. Shen, J. Shao, Z. Huang, and X. Zhou, "Effective and efficient query processing for video subsequence identification," *IEEE Trans. Knowl. Data Eng.*, Vol. 21, No. 3, pp. 321-334, 2009.
[28] H. T. Shen, X. Zhou, Z. Huang, J. Shao, and X. Zhou, "UQLIPS: a real-time near-duplicate video clip detection system," *Proc. Int'l Conf. Very Large Data Bases (VLDB)*, pp. 1374-1377, Vienna, Austria, Sep. 23-27, 2007.
[29] J. Sivic and A. Zisserman, "Video Google: a text retrieval approach to object matching in videos," *Proc. IEEE Int'l Conf. Computer Vision (ICCV)*, Nice, France, Oct. 14-17, 2003.
[30] H. K. Tan, C. W. Ngo, R. Hong, and T. S. Chua, "Scalable detection of partial near-duplicate videos by visual-temporal consistency," *Proc. ACM Int'l Conf. Multimedia (ACM-MM)*, Beijing, China, Oct. 19-24, 2009.
[31] TRECVID 2010 Guidelines, http://www-nlpir.nist.gov/projects/tv2010/tv2010.html#ccd
[32] R. Weber, H. Schek, and S. Blott, S, "A quantitative analysis and performance study for similarity-search methods in high dimensional spaces," *Proc. Int'l Conf. Very Large Data Baases (VLDB)*, pp. 194-205, New York, USA, Aug. 24-27, 1998.
[33] X. Wu, A. G. Hauptmann, and C. W. Ngo, "Practical elimination of near-duplicates from web video search," *Proc. ACM Int'l Conf. Multimedia (ACM-MM)*, pp. 218-227, Augsburg, Germany, Sep. 23-28, 2007.
[34] M. C. Yeh and K. T. Cheng, "A compact, effective descriptor for video copy detection," *Proc. ACM Int'l Conf. Multimedia (ACM-MM)*, Beijing, China, Oct. 19-24, 2009.
[35] S. Zhang, Q. Tian, G. Hua, Q. Huang, and S. Li, "Descriptive visual words and visual phrases for image applications," *Proc. ACM Int'l Conf. Multimedia (ACM-MM)*, pp. 75-84, Beijing, China, Oct. 19-24, 2009.

Chih-Yi Chiu (M'10) received the B.S. degree in information management from National Taiwan University, Taiwan, in 1997, and the M.S. degree in computer science from National Taiwan University, Taiwan, in 1999, and the Ph.D. degree in computer science from National Tsing Hua University, Taiwan, in 2004.

From January 2005 to July 2009, he was with Academia Sinica as a Postdoctoral Fellow. In August 2009, he joined National Chiayi University, Taiwan, as an assistant professor in the Department of Computer Science and Information Engineering. His current research interests include multimedia retrieval, human-computer interaction, and digital archiving.

Hsin-Min Wang (SM'04) received the B.S. and Ph.D. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 1989 and 1995, respectively.

In October 1995, he joined the Institute of Information Science, Academia Sinica, Taipei, Taiwan, as a Postdoctoral Fellow. He was promoted to Assistant Research Fellow, Associate Research Fellow, and then Research Fellow in 1996, 2002, and 2010, respectively. He was an Adjunct Associate Professor with National Taipei University of Technology and National Chengchi University. He was a board member, chair of academic council, and secretary-general of ACLCLP. He currently serves as a standing board member of ACLCLP and as an editorial board member of the International Journal of Computational Linguistics and Chinese Language Processing. His major research interests include speech processing, natural language processing, multimedia information retrieval, and pattern recognition.

Dr. Wang was a recipient of the Chinese Institute of Engineers (CIE) Technical Paper Award in 1995. He is a life member of ACLCLP and IICM and a member of ISCA.

Footnote

[1] Here, we simplify the discussion by assuming that the window lengths of W and W' are the same and identical to n_W. In Section IV.A, we estimate the window length according to the video content.

Affiliation of Author

C. Y. Chiu is with the Department of Computer Science and Information Engineering, National Chiayi University, Chiayi City, 60004, Taiwan (phone: +886-5-2717228; fax: +886-5-2717705; e-mail: cychiu@mail.ncyu.edu.tw).

H. M. Wang is with the Institute of Information Science and Research Center for Information Technology Innovation, Academia Sinica, Taipei, 11529, Taiwan (phone: +886-2-27883799; fax: +886-2-27824814; e-mail: whm@iis.sinica.edu.tw).

Acknowledgment of Financial Support

This work was supported in part by the National Science Council of Taiwan under Grants NSC 99-2221-E-415-011 and NSC 99-2631-H-001-020.