J Braz Comput Soc (2011) 17:175–192
DOI 10.1007/s13173-011-0036-4

ORIGINAL PAPER

Identification of video subsequence using bipartite graph matching

Silvio Jamil Ferzoli Guimarães · Zenilton Kleber Gonçalves do Patrocínio Jr.

Received: 16 February 2011 / Accepted: 13 July 2011 / Published online: 7 September 2011

Abstract Subsequence identification consists of identifying real positions of a specific video clip in a video stream together with the operations that may be used to transform the former into a subsequence from the latter. To cope with this problem, we propose a new approach, considering a bipartite graph matching to measure video clip similarity with a target video stream which has not been preprocessed.

The main contributions of our work are the application of a simple and efficient distance to solve the subsequence identification problem along with the definition of a hit function that identifies precisely which operations were used in query transformation. Experimental results demonstrate that our method performances achieve 90% recall with 93% precision, though it is done without preprocessing of the target video.

Keywords Video retrieval · Graph bipartite · Video clip localization

1 Introduction

Traditionally, multimedia information has been analogically stored and manually indexed. Due to advances in multimedia technology, video retrieval techniques are increasing. Unfortunately, the recall and precision rates of these systems depend on the similarity measure used to retrieve information. Nowadays, due to improvements on digitalization and compression technologies, database systems are used to store images and videos, together with their metadata and associated taxonomy. Thus, there is an increasing search for efficient systems to process and to index image, audio and video information, mainly for information retrieval purposes.

Automatic segmentation, indexing, and retrieval of large amount of video data have important applications in archive management, entertainment, media production, rights control, surveillance, and many more.

We see the complex task of video segmenting (mainly in the presence of gradual transitions and motion) and indexing faces and the challenges of coping with the exponential growth of the Internet, which has resulted in a massive publication and sharing of video content and in an increase in the number of duplicated documents; and the distribution across communication channels, like TV, resulting in thousands of hours of streaming broadcast media. Additionally, one important application of video content management is broadcast monitoring for market analysis [8, 10, 11].

The video clip localization problem, as it will be referred to throughout this paper, has arisen in the domain of broadcast television, and consists of identifying the real locations of a specific video clip in a target video stream (see Fig. 1). The main issues that must be considered during video clip localization are:

(i) The definition of the dissimilarity measures in order to compare frames/shots of video clips
(ii) The processing time of the algorithms due to the huge amount of information that must be analyzed
(iii) The insertion of intentional and non-intentional distortions

A previous version of this paper appeared at WEBMEDIA 2010, the Brazilian Symposium on Multimedia and the Web.

S.J.F. Guimarães (✉) · Z.K.G. do Patrocínio Jr.
Department of Computer Science, Pontifícia Universidade Católica de Minas Gerais, Rua Walter Ianni, 255, 31980-110 São Gabriel Belo Horizonte, Brazil
e-mail: sjamil@pucminas.br

Z.K.G. do Patrocínio Jr.
e-mail: zenilton@pucminas.br

(iv) Different frame rates; and
(v) Edition of the videos

The selection of the feature used to compute the dissimilarity measure has an important role to play in content-based image retrieval and has been largely explored [25]. In [7], the authors showed that the performance of the features is task dependent, and that it is hard to select the best feature for an specific task without empirical studies. Nevertheless low-complexity features and matching algorithms can work together to increase matching performance.

Current methods for solving the video retrieval/localization problem can be grouped in two main approaches:

(i) Computation of video signatures after temporal video segmentation, as described in [10, 17, 20]; and
(ii) Use of matching algorithms after transformation of the video frame content into a feature vector, as described in [1, 9, 13, 18, 26]

When video signatures are used, methods for temporal video segmentation must be applied before the signature calculation [2]. Although temporal video segmentation is a widely studied problem, it represents an important issue that has to be considered, as it increases the complexity of the algorithms and it affects the matching performance. For methods based on string matching algorithms, the efficiency of these algorithms must be taken into account, when compared to image/video identification algorithms. In [1] and

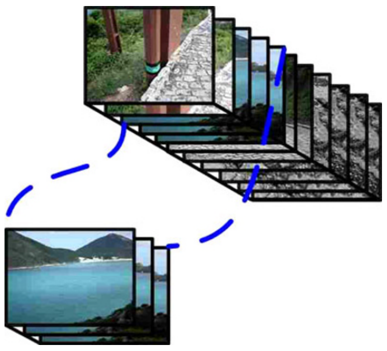


Fig. 1 Problem of identifying the real position of a specific video clip in a target video stream

[18], the authors have successfully applied the longest common substring (LCS) algorithm to deal with the problem. However, it has space and time cost $O(mn)$, in which m and n represent the size of the query and target video clips, respectively. In [9], the authors have proposed a new approach to cope with the problem of video clip localization using the maximum cardinality matching of a bipartite graph; however, this approach deals only with exact video matching. In [26] bipartite graph matching is also used; however, the first step of the algorithm is related to analysis of the video query and video target to identify frame similarities and candidate locations of query occurrence (clip filtering).

In the present paper, we present a modified version of our previous approach [9], which is able to deal with the general case in video clip localization problem, allowing insertion, removal and replacement operations on the video clip. Our method not only solves the approximate video clip (or subsequence) localization problem, but it also gives a precise description of the operations that are necessary to transform the query video into a clip of the target video stream. Our approach deals with the problem of video clip localization using the maximum cardinality matching of a bipartite graph. It is important to note that we use this information to shift over the video target in order to identify candidate locations of the matching. For the set of frames from a query video and from a clip of target video, a bipartite graph is constructed based on a similarity measure between each pair of frames (illustrated in Fig. 2(a)). The size of the maximum cardinality matching of this graph defines a video similarity measure that is used for video clip identification. This information together with the maximum distance between any two frames of the target video clip which belong to the maximum cardinality matching are used to analyze the hit occurrences and to identify precisely which operations are necessary to transform the query video into the target video content. A preliminary version of this work was presented in [12].

This paper is organized as follows. In Sect. 2, we discuss related work. In Sect. 3, the problem of video subsequence identification is described, together with some formal definitions of the field. In Sect. 4, we present a methodology to identify the locations of a query video clip using bipartite graph matching. In Sect. 5, we discuss experiments and the

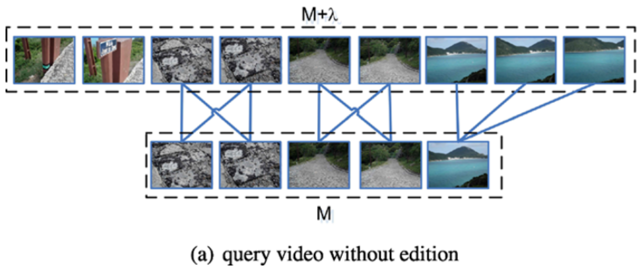

M+λ
M
(a) query video without edition
M+λ
M
(b) edited (frame-reordered) query video

Fig. 2 Frame similarity graph

Table 1 Comparison of some approaches for video clip localization (adapted from [24])

	Sliding window [3]	Temporal order [16, 27]	Vstring edit [1]	Multi-level [19]	Graph approach [24]	Dense [26]	BMH [13]	Our proposed method
Shot/Frame Matching	shot	shot	shot	shot	shot	frame	frame	frame
Temporal order	no	yes	yes	yes	yes	possible	yes	possible
Clip filtering	no	no	no	no	yes	yes	no	no
Online Clip Segment.	yes	no	no	no	yes	yes	no	no
Preprocessing	yes	yes	yes	yes	yes	yes	no	no
Video edition	no	no	no	no	no	yes	no	yes

setting of the parameters of the algorithm. Finally, in Sect. 6, we draw some conclusions and suggest further work.

2 Related works

In general, video clip similarity can be obtained using matching between shots. Besides relying on shot similarity, clip similarity could also be dependent on the temporal order. Table 1 presents a comparison between some approaches found in the literature. In [3], a window is slid across the target video to identify the matched shots. The number of similar shots in the window is used to form a one-dimensional signal. The relevant clips are then segmented by locating the local maxima of this signal (i.e., online clip segmentation). The major disadvantage of the method presented in [3] is that the granularity (which models the degree of one-to-one shot matching between two clips), temporal order, and interference (which models the percentage of unmatched shots) are not taken into account. One typical example is that the similarity of two clips with one-to-one shot matching can be the same as two clips with one-to-many matching [24].

In [16, 27], shots in two clips are matched by preserving their temporal order. For instance, in [27] the authors have employed dynamic programming to align two sequences in time order and to measure the similarity accordingly. Sometimes, in clip retrieval, shot matching by time preserving may not be appropriate, since shots in different clips tend to appear in various orders due to editing effects [24]. In [16], a nontemporal preserving matching is also proposed. The similarity of clips is reduced to the similarity of two most similar shots by bidirectional matching. Nevertheless, this approach is prone to noise and, furthermore, the similarity ranking is ineffective, since two clips can be considered similar even if only one pair of shots is matched.

Another approach for clip retrieval is proposed in [1], where various factors, including granularity, temporal order, and interference are taken into account. Several new edit operations (swap, fusion and break) are proposed, along with the parametric edit distances which consider the weights of traditional and video-specific operations for similarity ranking. One major problem with this approach is the high dependence on the robustness of features used. Moreover, if the indices are not invariant under certain changes, the computed edit distance may not be very reliable. In [13], a modified version of the Boyer–Moore–Horspool (BMH) algorithm is proposed for exact string matching [14, 21], to deal with the problem of video location and counting; however, it is a very hard task to translate video features into a meaningful alphabet.

As indicated in Table 1, most approaches assume that video clips are presegmented and always available for matching. In addition, the capabilities of filtering irrelevant clips prior to similarity ranking are usually not considered. In [24], the authors have adopted clip filtering and online segmentation. First, the candidate clips are located and segmented from videos, while the irrelevant clips are rapidly filtered out. In the second stage, the detailed similarity ranking is conducted by using a quality measure of matching (determined jointly by the granularity, temporal order, and interference factors). In [26], the authors have proposed two algorithms for clip localization, an optimum and a sub-optimum approach, based on bipartite graph matching analysis, and they present good experimental results. This work adopted a clip filtering strategy in order to identify the candidate locations; however, this could increase mismatches since two or more video clips may be merged into one.

Recently, an online system to detect near-duplicate occurrences into a video stream has been proposed in [15]. In that work, some video editions are allowed; however, temporal reorder and combination of video edit operations are not treated. In [6], a video retrieval algorithm based on ensemble similarity was proposed. An ensemble similarity is used to calibrate the similarity between a user given query video clip and each video clip in the database. In [4], the authors present a method based on a time-series linear search for detecting video copies. Their method utilizes a sliding window to locate sequences that are near-duplications of a given query. Experimental results demonstrate that their proposed method is robust against different types of video transformation and editing. Finally, the work presented in [29] considers the copy detection problem in the case of a continuous query stream, for which precise temporal localization and some complex video transformations like frame insertion and video editing need to be handled. The authors present a frame fusion approach based on a Viterbi-like algorithm which converts video copy detection to a frame similarity search and frame fusion under a temporal consistency assumption. The test results show that their proposal achieves high localization accuracy even when a query video undergoes some complex transformations.

The first difference between our proposed approach and the others is associated with the use of frame matching to compute video clip similarity. Most contributions assume that the target video has been preprocessed and online/offline segmented into video clips which are used by the search procedure, while ours can be applied directly to a target video stream without any preprocessing, since it uses frame-based similarity measures. With the exponential growth of the Internet, the storage of segmented videos may become an intractable problem. Our approach allows us to perform video localization over a streaming media downloaded directly from the Internet, while most of the others need to download, segment and store segmented video clips before starting to deal with video clip localization.

Moreover, our approach can be applied without considering temporal order constraints, which allows us to locate the position of the query video even if the video has been edited and its frames reordered (illustrated in Fig. 2(b)). The current version of our algorithm also deals with insertion, removal and replacement of frames/shots, and it also allows for changes in temporal order of query video clip frames/shots. However, our approach can be applied to the traditional (exact) video clip localization problem using dynamic programming to compute temporal order similarity. On the other hand, clip editing and reordering have become desired features in the new context of online video delivery. As mentioned in [23], users expect to be able to manipulate video content based on choices such as desired portions of video, ordering and "crop/stitch" of clips. New coding schemes that consider this novel scenario have been included in most recent standards such as MPEG-7 and MPEG-21 [28]. Nevertheless, since our approach is based on frame similarity measures, it may present an efficiency problem. This issue has been addressed by employing a *shift strategy* based on the size of the maximum cardinality matching, as will be discussed later.

Finally, one should note that our approach does not require a complete match between the query video and a given target video clip, since we are only trying to identify video subsequences. During the search, when a matching is identified (even if there is an error whose size is less than or equal to the allowed video edit distance), our method does not search for an exact subsequence. In fact, we are interested not only in finding locations at the target video of a near-similar clip of the query, but also in identifying which operations are necessary to transform the query video into a subsequence of the target.

3 Problem definition

Let $\mathbb{A} \subset \mathbb{N}^2$, $\mathbb{A} = \{0, \ldots, \mathrm{W}-1\} \times \{0, \ldots, \mathrm{H}-1\}$, where W and H are the width and height of each frame, respectively, and, $\mathbb{T} \subset \mathbb{N}$, $\mathbb{T} = \{0, \ldots, N-1\}$, where N is the length of a video. Frame, video, and video clip can be defined as follows.

Definition 1 (Frame) A frame f is a function from $\mathbb{A}$ to $\mathbb{Z}^3$, where for each spatial position (x, y) in $\mathbb{A}$, $f(x, y)$ represents a color value at pixel location (x, y). Without loss of generality, a frame can be defined by a function from $\mathbb{A}$ to $\mathbb{Z}$ to represent a grayscale value at location (x, y).

Definition 2 (Video) A video V_N, in domain $\mathbb{A} \times \mathbb{T}$, can be seen as a temporally ordered sequence of frames f. It is described by

$$\mathrm{V}_N = (f)_{t \in \mathbb{T}}, \tag{1}$$

where N is the number of frames contained in the video.

Definition 3 (Video clip) Let V_N be a video. A j-sized video clip $\mathrm{C}_{k,j}$ is a temporally ordered subsequence of frames from V_N which starts at frame k with j frames. It can be described by

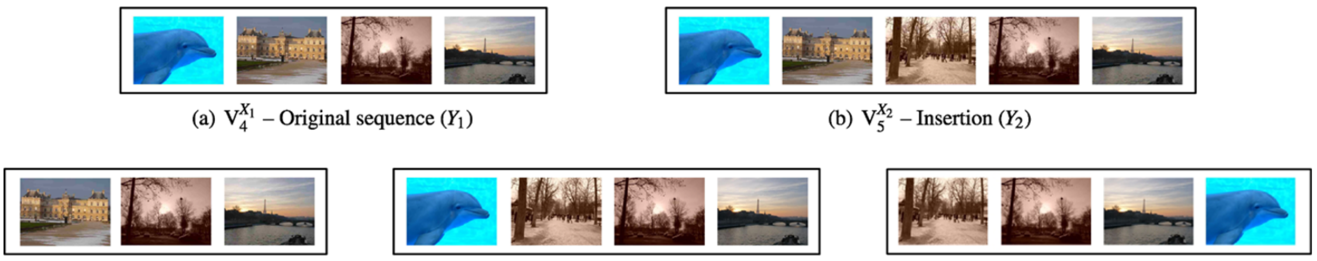
$$\mathrm{C}_{k,j} = (f_t \mid f_t \in \mathrm{V}_N)_{t \in [k, k+j-1]} \tag{2}$$

in which $k \leq N - j$.

It is obvious that $\mathrm{C}_{k,0} = \emptyset, \forall k$, and $\mathrm{C}_{0,N} = \mathrm{V}_N$. Moreover, from two distinct videos one can produce distinct video clips, so a superscript will be used to indicate which video is the frame source. Therefore video clip $\mathrm{C}^X_{.,.}$ contains frames f^X_t from video V^X_N, and, $\mathrm{C}^X_{0,N} = \mathrm{V}^X_N$.

A dataset may contain an altered version of a specified video clip, in which some frames may have been added, removed, or even replaced by others. Consequently, we can define a distance between two video clips based only on the number of frame insertions, removals, and replacements, as follows. It is important to note that frame reordering is permitted without changing the video edit distance. But it is possible to consider temporal order (if necessary) adopting a measure for temporal order as used in [24].

Definition 4 (Video clip edit distance—$\mathrm{d}(\mathrm{C}^{X_1}_{k_1,i}, \mathrm{C}^{X_2}_{k_2,j})$) Let $\mathrm{C}^{X_1}_{k_1,i}$ and $\mathrm{C}^{X_2}_{k_2,j}$ be two video clips of sizes i and j from videos V^{X_1} and V^{X_2}, respectively. Then the edit distance



(a) $V_4^{X_1}$ – Original sequence (Y_1)

(b) $V_5^{X_2}$ – Insertion (Y_2)

(c) $V_3^{X_3}$ – Removal (Y_3)

(d) $V_4^{X_4}$ – Replacement (Y_4)

(e) $V_4^{X_5}$ – Reordering (Y_5)

Fig. 3 Video clip modifications

between $C_{k_1,i}^{X_1}$ and $C_{k_2,j}^{X_2}$ is equal to the minimum number of operations (insertions, removals and replacements) needed to transform $C_{k_1,i}^{X_1}$ into $C_{k_2,j}^{X_2}$, and vice versa, regardless of frame temporal ordering.

Let $V_4^{X_1}$ be the original video Y_1 presented by Fig. 3(a), so $C_{0,4}^{X_1} = V_4^{X_1}$. Let $Y_2 = V_5^{X_2}$, $Y_3 = V_3^{X_3}$, $Y_4 = V_4^{X_4}$, and let $Y_5 = V_4^{X_5}$ be the altered versions presented by Figs. 3(b), 3(c), 3(d), and 3(e), respectively.

The edit distance between Y_1 and Y_2 is equal to 1 since both videos contain the same frames except for $f_2^{X_2}$, which is not present in the original video Y_1, i.e., $C_{0,2}^{X_1} = C_{0,2}^{X_2}$ and $C_{2,2}^{X_1} = C_{3,2}^{X_2}$. Therefore, $d(Y_1, Y_2) = d(C_{0,4}^{X_1}, C_{0,5}^{X_2}) = 1$ because only one insertion is need to transform Y_1 into Y_2. Analogously, the edit distance between Y_1 and Y_3 is also equal to 1, since both videos contain the same frames except for $f_0^{X_1}$, which is not present in altered video Y_3, i.e., $C_{1,3}^{X_1} = C_{0,3}^{X_3}$. Hence, $d(Y_1, Y_3) = d(C_{0,4}^{X_1}, C_{0,3}^{X_3}) = 1$, because only one removal is needed to transform Y_1 into Y_3. The edit distance between Y_1 and Y_4 is also equal to 1, since frame $f_1^{X_1}$ has been replaced by $f_1^{X_4}$, i.e., $C_{0,1}^{X_1} = C_{0,1}^{X_4}$ and $C_{2,2}^{X_1} = C_{2,2}^{X_4}$. Therefore, $d(Y_1, Y_4) = d(C_{0,4}^{X_1}, C_{0,4}^{X_4}) = 1$, because one replacement is needed to transform Y_1 into Y_4. Finally, the edit distance between Y_1 and Y_5 is equal to 0, since there is only a temporal reordering of frames without any insertion, removal or replacement (if temporal order is considered, the video edit distance is not equal to zero). Therefore, $d(Y_1, Y_5) = d(C_{0,4}^{X_1}, C_{0,4}^{X_5}) = 0$, because no insertion, removal and replacement is needed to transform Y_1 into Y_5. However, it is possible to consider temporal order as an assumption of our method (see Sect. 4.3).

In the previous example, two frames were considered equal if they were exactly alike, but this may not be always true due to minor differences in vector quantization used in digital representation for both videos. In order to establish if two distinct frames of two distinct video clips are similar, we define *frame similarity* as follows.

Definition 5 (Frame similarity) Let $f_{t_1}^{X_1}$ and $f_{t_2}^{X_2}$ be two video frames at locations t_1 and t_2 from video clips $C_{k_1,i}^{X_1}$ and $C_{k_2,j}^{X_2}$, respectively. Two frames are similar if the distance measure $\mathcal{D}(f_{t_1}^{X_1}, f_{t_2}^{X_2})$ between them is smaller than a specified threshold δ. The frame similarity is defined as

$$\mathrm{FS}\big(f_{t_1}^{X_1}, f_{t_2}^{X_2}, \delta\big) = \begin{cases} 1, & \text{if } \mathcal{D}(f_{t_1}^{X_1}, f_{t_2}^{X_2}) \le \delta; \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

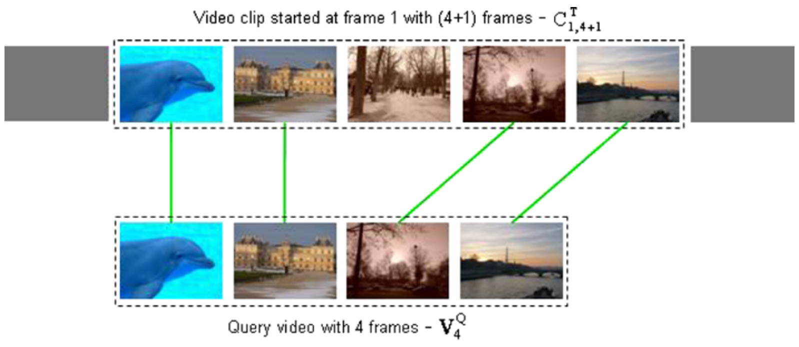
There are several choices for $\mathcal{D}(f_{t_1}^{X_1}, f_{t_2}^{X_2})$, i.e., the distance measure between two frames; e.g. histogram/frame difference, histogram intersection, difference of histograms means, and others. In our experiments, we have adopted histogram intersection as the distance measure between frames. After selecting one distance measure, it is possible to construct a frame similarity graph based on a query video V_M^Q and a $(M + \lambda)$-sized video clip of target video $C_{k,M+\lambda}^T$, in which λ is the maximum allowed edit distance, as follows.

Definition 6 (Frame similarity graph—$G_{k,\lambda}^{\delta}$) Let V_M^Q and V_N^T be a query video with M frames and a target video with N frames, respectively, and let $C_{k,M+\lambda}^T$ be a $(M + \lambda)$-sized video clip which starts at frame k of target video with $(M + \lambda)$ frames. A frame similarity graph $G_{k,\lambda}^{\delta} = (N^Q \cup N_{k,\lambda}^T, E_{k,\lambda}^{\delta})$ is a bipartite graph. Each node $v_{t_1}^Q \in N^Q$ represents a frame from the query video $f_{t_1}^Q \in C_{0,M}^Q (= V_M^Q)$ and each node $v_{t_2}^T \in N_{k,\lambda}^T$ represents a frame from the target video clip $f_{t_2}^T \in C_{k,M+\lambda}^T$. There is an edge $e \in E_{k,\lambda}^{\delta}$ between $v_{t_1}^Q$ and $v_{t_2}^T$ if frame similarity of associated frames is equal to 1, i.e.,

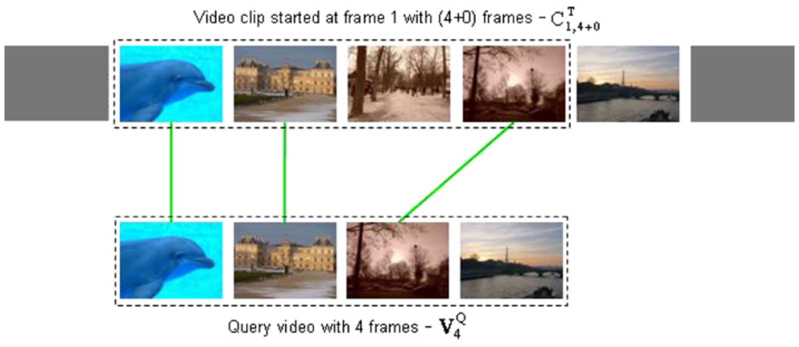
$$E_{k,\lambda}^{\delta} = \big\{\big(v_{t_1}^Q, v_{t_2}^T\big) \mid v_{t_1}^Q \in N^Q, v_{t_2}^T \in N_{k,\lambda}^T, \mathrm{FS}\big(f_{t_1}^Q, f_{t_2}^T, \delta\big) = 1\big\}. \tag{4}$$

As illustrated in Fig. 2, in order to allow up to λ operations of insertion into the query video clip, we match the query video to a video clip of the target video stream whose size (number of frames) is equal to the query video size plus the maximum allowed edit distance (λ). Figure 4 illustrates three different frame similarity graphs based on a query video equals to the original video sequence Y_1 (see Fig. 3(a)) and a video target containing Y_2 (see Fig. 3(b))

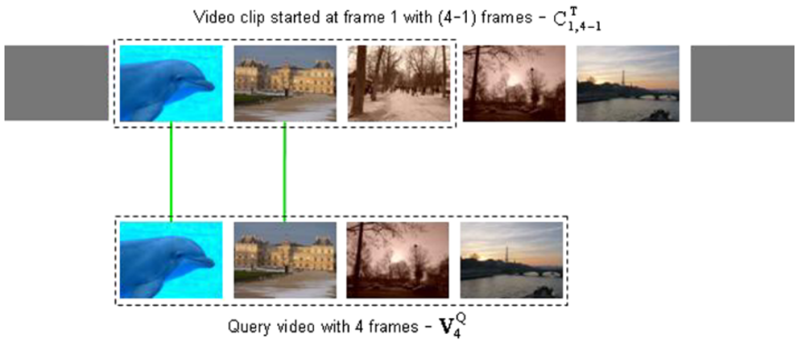
Fig. 4 Frame similarity graphs based on a query video equals to the original video sequence Y_1 of Fig. 3(a) (i.e., $M = 4$) and a video target containing Y_2 of Fig. 3(b) (with only one insertion, i.e. $\lambda = 1$) for different sizes of target video clip


Video clip started at frame 1 with (4+1) frames - $C^T_{1,4+1}$
Query video with 4 frames - V^Q_4

(a) Frame similarity graph for a target video clip of 5 ($= M + \lambda$) frames.


Video clip started at frame 1 with (4+0) frames - $C^T_{1,4+0}$
Query video with 4 frames - V^Q_4

(b) Frame similarity graph for a target video clip of 4 ($= M + 0$) frames.


Video clip started at frame 1 with (4-1) frames - $C^T_{1,4-1}$
Query video with 4 frames - V^Q_4

(c) Frame similarity graph for a target video clip of 3 ($= M - \lambda$) frames.

with only one insertion, i.e. $d(Y_1, Y_2) = 1$. For this case, the query video has 4 frames (i.e., $M = 4$) and the maximum allowed edit distance should be at least equal to 1 (i.e., $\lambda = 1$). One can easily see that a frame similarity graph between all the relevant frames is only obtained when the number of frames of the target video clip is at least equal to 5 ($= M + \lambda$) frames—see Fig. 4(a). If the size of the target video clip is smaller than 5 frames, relevant frames may not be consider in the construction of the frame similarity graph—see Fig. 4(b) and 4(c). If the target video clip has more than 5 frames, unnecessary frames may be used in the construction of the frame similarity graph, and this could increase the number of misses.

Conversely, in order to allow up to λ operations of removal from the query video clip, we match the query video to a video clip of the target video stream whose size (number of frames) is equal at least to the query video size minus the maximum allowed edit distance (λ). Figure 5 illustrates three different frame similarity graphs based on a query video equal to the original video sequence Y_1 (see Fig. 3(a)) and a video target containing Y_3 (see Fig. 3(c)) with only one removal, i.e. $d(Y_1, Y_3) = 1$. Again, the query video has 4 frames (i.e., $M = 4$) and the maximum allowed edit distance should be at least equal to 1 (i.e., $\lambda = 1$). One can see that, in this case, a frame similarity graph between all the relevant frames is obtained when the number of frames of the target video clip is at least equal to 3 ($= M - \lambda$) frames—see Fig. 5(c). If the size of the target video clip is greater than 3 frames, unnecessary frames may be considered in the construction of the frame similarity frame—see Fig. 5(a) and 5(b). However, it the target video has less than 3 frames,

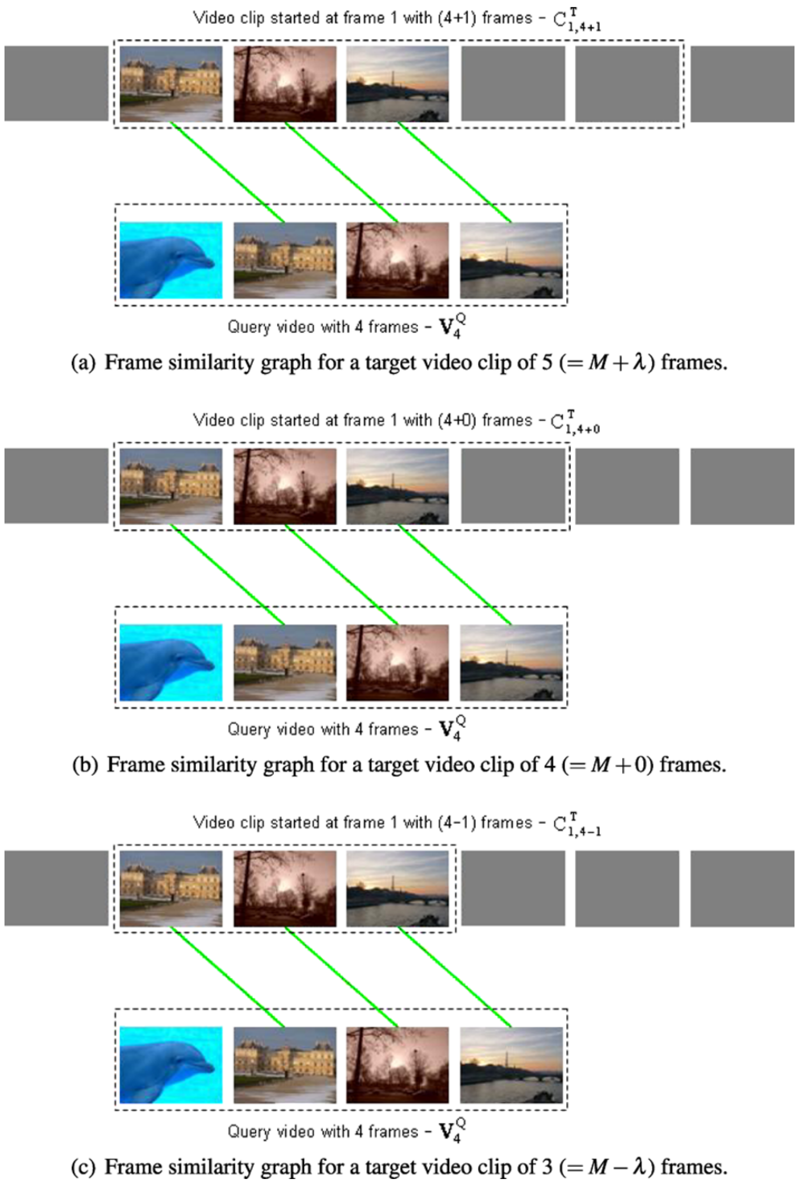

Video clip started at frame 1 with (4+1) frames - $C^T_{1,4+1}$
Query video with 4 frames - V^Q_4
(a) Frame similarity graph for a target video clip of 5 ($= M+\lambda$) frames.
Video clip started at frame 1 with (4+0) frames - $C^T_{1,4+0}$
Query video with 4 frames - V^Q_4
(b) Frame similarity graph for a target video clip of 4 ($= M+0$) frames.
Video clip started at frame 1 with (4-1) frames - $C^T_{1,4-1}$
Query video with 4 frames - V^Q_4
(c) Frame similarity graph for a target video clip of 3 ($= M-\lambda$) frames.

Fig. 5 Frame similarity graphs based on a query video equals to the original video sequence Y_1 of Fig. 3(a) (i.e., $M = 4$) and a video target containing Y_3 of Fig. 3(c) (with only one removal, i.e. $\lambda = 1$) for different sizes of target video clip

relevant frames may not be used in the construction of the frame similarity graph.

Both examples discussed before illustrate scenarios in which mismatches and improper hit type classifications may increase, if the size of the target video clip is not carefully chosen. Thus, the size of the target video clip should be restricted to the range $[M-\lambda, M+\lambda]$, in order to reduce the number of mismatches and improper hit type classifications. But, as we want to ensure that all possible combinations of insertions, removals and replacements allowed by the maximum edit distance are considered, we should generate a frame similarity graph based on a query video clip of size M and a target video clip of size $M+\lambda$ (i.e., the upper bound stated above). Later, the lower bound $(M-\lambda)$ is enforced by the adoption of a hit function (see Definition 10).

In this paper, the video subsequence identification problem with (or without) any changes in the video content (including changes in its temporal ordering) will be addressed using *maximum cardinality matching*. To do so, we define *matching* and *maximum cardinality matching* as follows.

Definition 7 (Matching—$\mathrm{M}^{\delta}_{k,\lambda}$) Let $\mathrm{G}^{\delta}_{k,\lambda}$ be a frame similarity graph, i.e., $\mathrm{G}^{\delta}_{k,\lambda} = (\mathrm{N}^{Q} \cup \mathrm{N}^{T}_{k,\lambda}, \mathrm{E}^{\delta}_{k,\lambda})$. A subset $\mathrm{M}^{\delta}_{k,\lambda} \subseteq \mathrm{E}^{\delta}_{k,\lambda}$ is a matching if any two edges in $\mathrm{M}^{\delta}_{k,\lambda}$ are not adjacent. The size of matching $\mathrm{M}^{\delta}_{k,\lambda}$ is the number of edges in $\mathrm{M}^{\delta}_{k,\lambda}$, written as $|\mathrm{M}^{\delta}_{k,\lambda}|$.

Definition 8 (Maximum cardinality matching—$\overline{\mathrm{M}^{\delta}_{k,\lambda}}$) Let $\overline{\mathrm{M}^{\delta}_{k,\lambda}}$ be a matching in a frame similarity graph $\mathrm{G}^{\delta}_{k,\lambda}$. So,

$\overline{\mathrm{M}^{\delta}_{k,\lambda}}$ is the maximum cardinality matching (MCM) if there is no other matching $\mathrm{M}^{\delta}_{k,\lambda}$ in $\mathrm{G}^{\delta}_{k,\lambda}$ such that $|\mathrm{M}^{\delta}_{k,\lambda}| > |\overline{\mathrm{M}^{\delta}_{k,\lambda}}|$.

During the search, a hit can be associated with the size of the MCM, i.e., if $M - \lambda \leq |\overline{\mathrm{M}^{\delta}_{k,\lambda}}| \leq M$, then a hit may have been found. The lower bound is associated with a scenario of λ operations of removal, so the size of matching $\mathrm{M}^{\delta}_{k,\lambda}$ is at least $M - \lambda$. The upper bound is related to the size of the query video (i.e., M), since the frame similarity graph $\mathrm{G}^{\delta}_{k,\lambda}$ is a bipartite graph (and it is impossible to find a matching with size greater than $M = \min\{|\mathrm{N}^{Q}|, |\mathrm{N}^{T}_{k,\lambda}|\}$).

However, in order to prevent false positives, we need to evaluate the largest distance between any two frames of the target video clip associated with the MCM found. This evaluation is necessary to correctly identify which operations are necessary to transform the query video into a subsequence of the target video, since the size of MCM may be smaller than the video query size M, e.g. for removal and replacement operations.

Let $\mathcal{F}$ be the set of frames associated with the MCM $\overline{\mathrm{M}^{\delta}_{k,\lambda}}$ and it contains frames from the query and the target video clips, i.e., $\mathcal{F} = \mathcal{F}^{Q} \cup \mathcal{F}^{T}_{k,\lambda}$, in which $\mathcal{F}^{Q}$ and $\mathcal{F}^{T}_{k,\lambda}$ are frame sets from the query and the target video clips, respectively. So, $\mathcal{F}^{Q} = (f^{Q}_{t} \mid f^{Q}_{t} \in \mathrm{C}^{Q}_{0,M}, t \in [0, M-1])$, while $\mathcal{F}^{T}_{k,\lambda} = (f^{T}_{t} \mid f^{T}_{t} \in \mathrm{C}^{T}_{k,M+\lambda}, t \in [k, k+M+\lambda-1])$. It is also easy to verify that $|\overline{\mathrm{M}^{\delta}_{k,\lambda}}| = |\mathcal{F}^{Q}| = |\mathcal{F}^{T}_{k,\lambda}|$. So, the maximum distance between any two frames of the target video clip associated with the MCM found could be defined as follows.

Definition 9 (Maximum MCM distance—$D(\overline{\mathrm{M}^{\delta}_{k,\lambda}})$) Let $\overline{\mathrm{M}^{\delta}_{k,\lambda}}$ be the MCM in a frame similarity graph $\mathrm{G}^{\delta}_{k,\lambda}$ and let $\mathcal{F} = \mathcal{F}^{Q} \cup \mathcal{F}^{T}_{k,\lambda}$ be the set of frames associated with the MCM. Let t_f and t_l represent the locations of the first and the last frames of the target video clip that are related to the MCM $\overline{\mathrm{M}^{\delta}_{k,\lambda}}$, so $t_f = \min\{\, t \mid f^{T}_{t} \in \mathcal{F}^{T}_{k,\lambda}\}$ and $t_l = \max\{\, t \mid f^{T}_{t} \in \mathcal{F}^{T}_{k,\lambda}\}$. The maximum distance between any two frames of the target video clip associated with the MCM is defined as

$$D\big(\overline{\mathrm{M}^{\delta}_{k,\lambda}}\big) = t_l - t_f + 1. \tag{5}$$

Using the maximum MCM distance together with the size of the MCM, we can identify not only a hit occurrence but also its type. The type of a hit occurrence represents which operations are necessary to transform the query video into the target video clip. Therefore, we can define 06 (six) types of hit occurrence as follows.

- **T1—Exact hit (ExHit):** represents an exact match between the query video and the target video clip, therefore $|\overline{\mathrm{M}^{\delta}_{k,\lambda}}| = M$ in order to guarantee that every frame of the query video appears in the target, i.e., every frame of the query video is similar to at least one distinct frame of the target video clip, and $D(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) = |\overline{\mathrm{M}^{\delta}_{k,\lambda}}|$ to prevent any additional frame from having been inserted among the target frames;
- **T2—Insertion hit (InsHit):** represents a match between the query video and the target video clip in which additional frames have been inserted into the target, thus $|\overline{\mathrm{M}^{\delta}_{k,\lambda}}| = M$ in order to guarantee that every frame of the query video appears in the target video clip, and $|\overline{\mathrm{M}^{\delta}_{k,\lambda}}| < D(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) \leq M + \lambda$ to ensure that no more than λ additional frames have been inserted among the target frames;
- **T3—Removal hit (RemHit):** represents a match between the query video and the target video clip in which some frames have been removed, so $M - \lambda \leq |\overline{\mathrm{M}^{\delta}_{k,\lambda}}| < M$ in order to guarantee that some frames of the query video do not appear in the target, and $D(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) = |\overline{\mathrm{M}^{\delta}_{k,\lambda}}|$ to prevent any additional frame from having been inserted among the target frames;
- **T4—Replacement hit (RepHit):** represents a match between the query video and the target video clip in which some frames have been replaced, so $M - \lambda \leq |\overline{\mathrm{M}^{\delta}_{k,\lambda}}| < M$ in order to guarantee that some frames of the query video do not appear in the target, and $D(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) = M$ to ensure that every missing target frame has been replaced by another one;
- **T5—Replacement and Removal hit (RepRHit):** represents a match between the query video and the target video clip in which some frames have been replaced and others have been removed, so $M - \lambda \leq |\overline{\mathrm{M}^{\delta}_{k,\lambda}}| < M$ in order to guarantee that some frames of the query video do not appear in the target, and $|\overline{\mathrm{M}^{\delta}_{k,\lambda}}| < D(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) < M$ to ensure that some of the frames have been removed since $D(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) < M$ (it is not only a replacement) but also that some frames have been replaced, since $D(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) > |\overline{\mathrm{M}^{\delta}_{k,\lambda}}|$ (it is not only a removal);
- **T6—Replacement and Insertion hit (RepIHit):** represents a match between the query video and the target video clip in which some frames have been replaced and others have been inserted, so $M - \lambda \leq |\overline{\mathrm{M}^{\delta}_{k,\lambda}}| < M$ in order to guarantee that some frames of the query video do not appear in the target (they have been replaced), and $M < D(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) \leq |\overline{\mathrm{M}^{\delta}_{k,\lambda}}| + \lambda$ to ensure that additional frames have been inserted among the target frames since $D(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) > M$ (it is not only a replacement) but also to limit the edit distance to a maximum of λ operations, i.e., $D(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) - |\overline{\mathrm{M}^{\delta}_{k,\lambda}}| \leq \lambda$.

Table 2 presents some examples of hit occurrences with a maximum edit distance $\lambda = 2$. Without loss of generality, each frame is represented only by a single feature value (an

Table 2 Examples of hit occurrence with $\lambda = 2$

Query	Target	Hit type	$\lvert\overline{\mathrm{M}^{\delta}_{k,\lambda}}\rvert$	$D(.)$
1 2 3 4	…1 2 3 4 – –…	ExHit	4	4
1 2 3 4	…1 2 3 5 4 –…	InsHit	4	5
1 2 3 4	…1 2 3 5 5 4…	InsHit	4	6
1 2 3 4	…1 2 – – – –…	RemHit	2	2
1 2 3 4	…1 2 3 – – –…	RemHit	3	3
1 2 3 4	…1 2 5 4 – –…	RepHit	3	4
1 2 3 4	…1 5 5 4 – –…	RepHit	2	4
1 2 3 4	…1 5 4 – – –…	RepRHit	2	3
1 2 3 4	…1 2 5 5 4 –…	RepIHit	3	5
1 2 3 4	…1 2 5 5 5 4…	No hit	3	6

integer one), and two frames are similar if their features are exactly the same. The first column shows the sequence of frames from a query video clip, while the second column presents the sequence of frames from the target video associated with a hit occurrence. Hit type is presented in the third column, followed by MCM size and maximum MCM distance in the fourth and fifth columns, respectively. Lines are separated if they represent different hit types. Therefore, using MCM size and maximum MCM distance, a function to detect a hit occurrence can be defined as follows.

Definition 10 (Hit function—$\mathrm{H}(\overline{\mathrm{M}^{\delta}_{k,\lambda}})$) Let $\overline{\mathrm{M}^{\delta}_{k,\lambda}}$ be the MCM in a frame similarity graph $\mathrm{G}^{\delta}_{k,\lambda}$. Then a function $\mathrm{H}(\overline{\mathrm{M}^{\delta}_{k,\lambda}})$ to detect a hit occurrence can be defined as

$$\mathrm{H}(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) = \begin{cases} 1, & \text{if } \lvert\overline{\mathrm{M}^{\delta}_{k,\lambda}}\rvert = M \text{ and } \\ & \quad D(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) = \lvert\overline{\mathrm{M}^{\delta}_{k,\lambda}}\rvert; \\ 2, & \text{if } \lvert\overline{\mathrm{M}^{\delta}_{k,\lambda}}\rvert = M \text{ and } \\ & \quad \lvert\overline{\mathrm{M}^{\delta}_{k,\lambda}}\rvert < D(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) \le M + \lambda; \\ 3, & \text{if } M - \lambda \le \lvert\overline{\mathrm{M}^{\delta}_{k,\lambda}}\rvert < M \text{ and } \\ & \quad D(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) = \lvert\overline{\mathrm{M}^{\delta}_{k,\lambda}}\rvert; \\ 4, & \text{if } M - \lambda \le \lvert\overline{\mathrm{M}^{\delta}_{k,\lambda}}\rvert < M \text{ and } \\ & \quad D(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) = M; \\ 5, & \text{if } M - \lambda \le \lvert\overline{\mathrm{M}^{\delta}_{k,\lambda}}\rvert < M \text{ and } \\ & \quad \lvert\overline{\mathrm{M}^{\delta}_{k,\lambda}}\rvert \le D(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) < M; \\ 6, & \text{if } M - \lambda \le \lvert\overline{\mathrm{M}^{\delta}_{k,\lambda}}\rvert < M \text{ and } \\ & \quad M < D(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) \le \lvert\overline{\mathrm{M}^{\delta}_{k,\lambda}}\rvert + \lambda; \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The hit function returns a value greater than zero for a hit occurrence, while zero corresponds to a miss. Moreover, the hit function value corresponds to the type of the hit occurrence, i.e., $\mathrm{H}(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) = 1$ for a **T1 hit** (or ExHit), $\mathrm{H}(\overline{\mathrm{M}^{\delta}_{k,\lambda}}) = 2$ for a **T2 hit** (or InsHit), and so on. As described before, the proposed hit function ignores temporal reordering of frames, since we are interested in identifying video subsequence with similar content. But it is possible to consider temporal order as an assumption of our method (see Sect. 4.3).

Finally, the video subsequence identification problem can be stated.

Definition 11 (Video subsequence identification problem) Let $\overline{\mathrm{M}^{\delta}_{k,\lambda}}$ be a MCM of a frame similarity graph $\mathrm{G}^{\delta}_{k,\lambda}$ with a maximum edit distance λ and a threshold δ. The video subsequence identification (VSI) problem corresponds to identify the locations of subsequences (or video clips) of the target video V^{T}_{N} that are similar (or near-similar) to a query video V^{Q}_{M} together with the operations necessary to transform the query video into a subsequence from the target. A hit at location k is found if there is a $(M+\lambda)$-sized video clip $\mathrm{C}^{T}_{k,M+\lambda}$ of V^{T}_{N} that matches with $\mathrm{V}^{Q}_{M} (= \mathrm{C}^{Q}_{0,M})$ according to the hit function $\mathrm{H}(\overline{\mathrm{M}^{\delta}_{k,\lambda}})$. Thus, the VSI problem can be stated as

$$\begin{aligned} &\mathrm{VSI}\bigl(\mathrm{V}^{Q}_{M}, \mathrm{V}^{T}_{N}, \delta, \lambda\bigr) \\ &\quad = \bigl\{k \in \bigl[0, N - (M+\lambda) - 1\bigr] \bigm| \mathrm{H}\bigl(\overline{\mathrm{M}^{\delta}_{k,\lambda}}\bigr) \neq 0\bigr\}. \end{aligned} \tag{7}$$

One should note that our definition for the VSI problem does not require a complete match between the query video and a given target video clip, since we are only trying to identify video subsequences. During the search, when a matching is identified (even if there is an error whose size is less than or equal to the allowed video edit distance), our method does not search for an exact subsequence. In fact, we are interested not only in finding locations at the target video of a near-similar clip of the query, but also in identifying which operations are necessary to transform the query video into a subsequence of the target. Since we have adopted a linear search for the query video (as will be discussed in Sect. 4), our identification procedure may produce two hits (e.g., two removal hits or a removal hit followed by a replacement hit) instead of an exact hit. But this could be eliminated (if necessary) with a post-processing step.

4 Methodology

As described before, the main goal of the VSI problem is to identify occurrences of the subsequences of the target video that are similar to a query video together with the operations necessary to transform the query video into a subsequence from the target; see Fig. 6. One of the key steps of the process is feature extraction. Choosing an appropriate feature that enhances the performance of a matching algorithm is not a trivial task. Therefore, empirical studies are the best way to get insights in which feature should be used for each case.

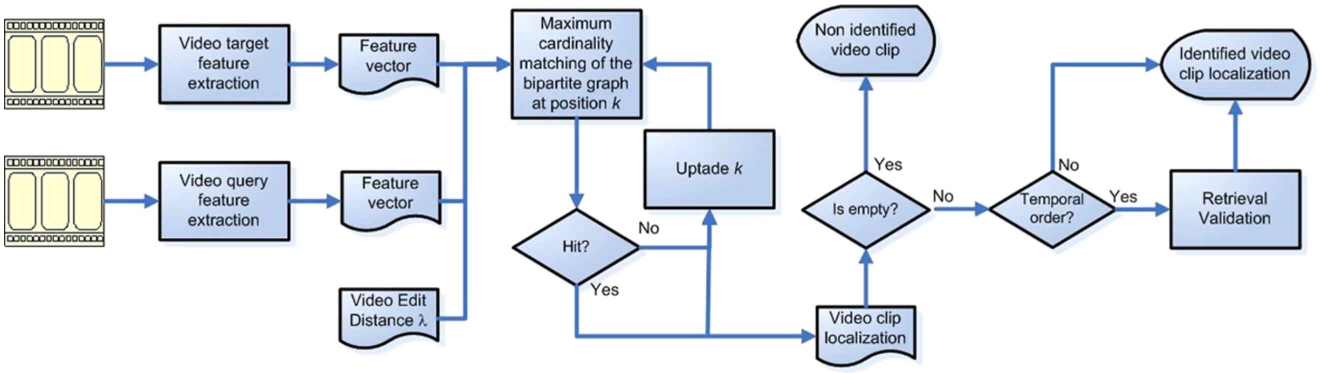

Video target feature extraction
Feature vector
Video query feature extraction
Feature vector
Video Edit Distance λ
Maximum cardinality matching of the bipartite graph at position k
Hit?
No
Yes
Uptade k
Video clip localization
Is empty?
Yes
Non identified video clip
No
Temporal order?
No
Identified video clip localization
Yes
Retrieval Validation

Fig. 6 Workflow for subsequence identification

Algorithm 1: Identification procedure.

```
Input:   Target video $V_N^T$; query video $V_M^Q$; threshold
         value $\delta$; maximum edit distance $\lambda$.
Output:  Hit locations ≡ pos[]; size of each hit ≡
         size[]; and hit occurrence types ≡ type[].

1  count = 0; $k = 0$;
2  while ($k < N - M - \lambda$) do
3      Construct $G_{k,\lambda}^{\delta}$;
4      Calculate $\overline{M_{k,\lambda}^{\delta}}$ and $\mathcal{F}$ for $G_{k,\lambda}^{\delta}$;
5      Calculate $t_f$ and $t_l$ for $\mathcal{F}$;
6      if ($H(\overline{M_{k,\lambda}^{\delta}}) \neq 0$) then
7          // Query was found at
              location k
8          pos[count] $= t_f$;
9          size[count] $= D(\overline{M_{k,\lambda}^{\delta}})$;
10         type[count] $= H(\overline{M_{k,\lambda}^{\delta}})$;
11         count = count + 1;
12         $k = t_l + 1$;
13     else
14         // Query was not found at
              location k
15         if ($|\overline{M_{k,\lambda}^{\delta}}| = 0$) then
16             $k = k + (M + \lambda)$;
17         else
18             $k = \max\{k + M - (|\overline{M_{k,\lambda}^{\delta}}| + \lambda), t_f\}$;
19         end
20     end
21 end
22 return (pos[], size[], type[]);
```

4.1 Identification procedure

Algorithm 1 presents our identification procedure. It scans over the target video stream, looking for a video clip that matches the query video, i.e., one that generates a frame similarity graph (*line 3*) which has a MCM that corresponds to a hit occurrence according to the hit function $H(\overline{M_{k,\lambda}^{\delta}})$ (*lines 4–6*). If a hit is found, its location, size and type are saved (*lines 7–11*).

It is also important to describe the *shift strategy* adopted (at *lines 12*, *16* and *18* of Algorithm 1). After locating a hit occurrence, the procedure ensures a jump to the location after the last frame of the target video clip that belongs to the MCM (*line 12*), since one should not expect to find the query video inside itself. In this case, the minimum shift value is $M - \lambda$, because it corresponds to the minimum value of the maximum $\overline{\text{MCM}}$ distance for a **T1** or a **T2 hit**. Moreover, $t_l + 1 = D(\overline{M_{k,\lambda}^{\delta}}) + t_f$ and t_f is equal at least to the previous value of k. Therefore, *line 12* could also be written as $k = k + D(\overline{M_{k,\lambda}^{\delta}})$. This not only contributes to an acceleration of the search but it also helps in reducing the number of false positives.

In case of a mismatch, the shift value depends on the MCM size. If the MCM size is equal to zero, i.e., there is no match between query and target frames, so the maximum shift value ($M + \lambda$) is employed (*line 16*). But if the MCM size is greater than zero, the shift value is set to the difference between $M - \lambda$ and the size of the MCM, i.e., the number of unmatched frames necessary to detect a hit occurrence (*line 18*). In order to ensure positive shift value, the value of k is set to t_f at least.

In spite of being a conservative approach, this setting allows our search procedure to perform better than the naive (brute force) algorithm, and it could result in a great improvement depending on query content and size, e.g., the search would be faster for videos that are more dissimilar and/or for lower values of edit distance (λ)—see the experimental results for more on that.

It is also important that the search procedure does not miss a *hit* position. Adjusting the shift value to the number of unmatched frames avoids this event by using a conservative approach which assumes that all mismatches occurred in the

Table 3 Example of subsequence identification procedure

Video Information																		$\lvert\mathrm{M}^{\delta}_{k,\lambda}\rvert$	$D(.)$	Shift	Iteration	Hit
Target video	1	5	6	2	4	2	1	3	5	1	2	3	7	6	1	4	2	–	–	–	–	
$\lambda = 0$	1	2	3															1	1	2	1	no
			1	2	3													1	1	2	2	no
					1	2	3											2	2	1	3	no
						1	2	3										**3**	3	3	4	ExHit
									1	2	3							2	2	1	5	no
										1	2	3						**3**	3	3	6	ExHit
													1	2	3			1	1	2	7	no
															1	2	3	2	3	–	8	no
Average																		1.87		2.00	–	
Target video	1	5	6	2	4	2	1	3	5	1	2	3	7	6	1	4	2	–	–	–	–	
$\lambda = 1$	1	2	3	–														2	4	1	1	no
		1	2	3	–													1	1	1	2	no
			1	2	3	–												1	1	1	3	no
				1	2	3	–											2	4	1	4	no
					1	2	3	–										3	3	4	5	ExHit
									1	2	3	–						3	3	4	6	ExHit
													1	2	3	–		1	1	2	7	no
															1	2	3	2	3	-	8	RepHit
Average																		1.87	–	2.00	–	

beginning of the video clip $C^T_{k,M+\lambda}$ of the target video. Thus, it is necessary to shift the target video clip at least the same number of unmatched frames in order to be feasible to find a new *hit* occurrence (**T3**, **T4**, **T5**, or **T6 hit**).

Table 3 presents an example of subsequence identification procedure for a query video with 3 frames and two different values for the maximum edit distance $\lambda = 0$ and $\lambda = 1$ ($\equiv$30% of the query video size), in order to illustrate an exact and an approximate matching. For an exact match (i.e., $\lambda = 0$), only two occurrences were found; and for $\lambda = 1$, the number of hit occurrences rises to three. In both cases an average shift value of 2 ($\equiv$66% if the query video size) was adopted.

4.2 Computation cost analysis

Generation of the frame similarity graph (*line 3*) and calculation of the MCM (*line 4*) are the most time consuming steps of Algorithm 1. Graph generation needs $O(M^2 + \lambda M)$ operations, in which M represents the query video size and λ is the maximum edit distance. Moreover, for practical scenarios, the maximum edit distance should be set to a value between 0 and $M/2$, i.e., $0 \leq \lambda \leq M/2$; thus, $\lambda = O(M)$. Therefore, the total time spent on graph generation is $O(NM^2)$, if the shift value is set to its worst possible value, i.e., if it is equal to 1.

Solving the MCM on a bipartite graph could be done with $O(E\sqrt{V})$ operations [22], in which V and E represent the number of nodes and edges, respectively. The number of nodes is always equal to $2M + \lambda$, while the number of edges depends on frame similarity measures and threshold. It could be close to zero, but it also could be equal to $M^2 + \lambda M$ in the worst case scenario (in which all query video frames are similar to all target video frames).

One should notice that at least $M - \lambda$ edges are needed in order to find a *hit* occurrence, i.e., the size of the MCM has to be equal to $M - \lambda$ at least (for **T3**, **T4**, **T5**, or **T6 hit**). And, for practical reasons, one should consider the number of edges to be at least $O(M - \lambda) = O(M)$ in the iteration that locates a *hit* occurrence. So the MCM should need at least $O((M - \lambda)\sqrt{2M + \lambda}) = O(M\sqrt{M})$ operations and the total time spent on maximum cardinality matching is $O(NM\sqrt{M})$. Unfortunately, in the worst case scenario, it could take $O((M^2 + \lambda M)\sqrt{2M + \lambda}) = O(M^2\sqrt{M})$.

Assuming that all query video frames are similar to all target video frames is quite unrealistic, since frame similarity measure and threshold should reduce that number. Moreover, this worst case scenario always leads to near optimal shift value, i.e., a shift value that is equal at least to the query video size because MCM for a $K_{M,M+\lambda}$ (i.e., a complete bipartite graph) has size equal to M and the minimum value for the maximum MCM distance is also M for **T1** and **T2**

hit. The search algorithm runs faster when the near optimal shift value is used (only $O(N/M)$ locations need to be tested), and the total time spent on the maximum cardinality matching calculation is $O(NM\sqrt{M})$, even in the worst case scenario.

Thus, our search procedure has a time complexity of $O(NM^2)$ since it is dominated by the total time spent on the graph generation step.

4.3 Retrieval validation

After query location candidates are selected, they may be validated to ensure other assumptions, like temporal ordering. In order to verify this assumption, we can use the dynamic programming (DP), as proposed by the authors in [24], for computing the longest common subsequence between video clips. We define a temporal order similarity as follows.

Definition 12 (Temporal order similarity—TS_k^δ) Let i be used to represent the ith frame of the query video V_M^Q, and j be used to represent the jth frame of a $\overline{M}$-sized video clip $\mathrm{C}^T_{t_f,\overline{M}}$ of the target video, in which $\overline{M} = D(\overline{\mathrm{M}^\delta_{k,\lambda}})$. Temporal order similarity $\mathrm{TS}_k^\delta(\mathrm{V}_M^Q, \mathrm{C}^T_{t_f,\overline{M}})$ between the query video and a $\overline{M}$-sized target video clip is equal to $\mathrm{T}^\delta(M,\overline{M})$ which is calculated using DP and a specified frame similarity threshold δ using the following recurrence relation:

$$\mathrm{T}^\delta(i,j) = \begin{cases} 0 & \text{if } i=0 \text{ or } j=0; \\ \mathrm{T}^\delta(i-1,j-1)+1 & \text{if } \mathrm{FS}(i,j,\delta)=1; \\ \max\{\mathrm{T}^\delta(i,j-1), \mathrm{T}^\delta(i-1,j)\} & \text{otherwise.} \end{cases} \tag{8}$$

Using the temporal order similarity TS_k^δ, we can validate location candidates by ensuring that its value is greater than a threshold, i.e., $\mathrm{TS}_k^\delta(\mathrm{V}_M^Q, \mathrm{C}^T_{t_f,\overline{M}}) \geq \Delta$. The threshold Δ represents the minimum number of similar frames in correct temporal order needed to accept a location candidate. No temporal order changes are allowed, if $\Delta = |\overline{\mathrm{M}^\delta_{k,\lambda}}|$ (i.e., the MCM size).

Table 4 and Table 5 illustrate the calculation of the temporal order similarity using DP and the recurrence relation of (8) between a target video clip and a given query video. The value of temporal order similarity appears at the shaded cell. Table 4(a) and Table 4(b) correspond to hit occurrences identified in Table 3 for $\lambda = 0$ (two **ExHit**s). In the first case, only 2 frames are in correct temporal order, while in the second case all 3 frames are in correct temporal order. Table 4(c) corresponds to the last hit occurrence identified in Table 3 for $\lambda = 1$ (a **RepHit**) with only 2 frames in the correct temporal order, and Table 4(d) shows a case with a temporal reordering of the same target video frames presented in Table 4(c) and only 1 frame is presented in correct temporal order. Finally, Table 5(a) and Table 5(b) correspond to hit occurrences associated with an insertion (i.e., **InsHit**) and a removal (i.e., **RemHit**), respectively, for video edit distance $\lambda = 1$—which did not appear in the example described in Table 3. In the first case, all 3 frames are in correct temporal order, while in the second case only 1 frame is presented in correct temporal order.

Table 4 Temporal order similarity for **ExHit**s and **RepHit**

(a)

			Target video frames		
			2	1	3
Query frames	$i\backslash j$	0	1	2	3
	0	0	0	0	0
1	1	0	0	1	1
2	2	0	1	1	1
3	3	0	1	1	**2**

(b)

			Target video frames		
			1	2	3
Query frames	$i\backslash j$	0	1	2	3
	0	0	0	0	0
1	1	0	1	1	1
2	2	0	1	2	2
3	3	0	1	2	**3**

(c)

			Target video frames		
			1	4	2
Query frames	$i\backslash j$	0	1	2	3
	0	0	0	0	0
1	1	0	1	1	1
2	2	0	1	1	2
3	3	0	1	1	**2**

(d)

			Target video frames		
			2	4	1
Query frames	$i\backslash j$	0	1	2	3
	0	0	0	0	0
1	1	0	0	0	1
2	2	0	1	1	1
3	3	0	1	1	**1**

5 Experiments

In this section, we present some experiments to show the effectiveness of our method, and we also compare it to Shen's approach [26]. We illustrate also an example of video editing in which the operations of insertion, removal and reordering are analyzed. The purposes of our experiments are:

(i) To show the effectiveness of our method
(ii) To compare it with a reference method
(iii) To understand the tuning of parameters

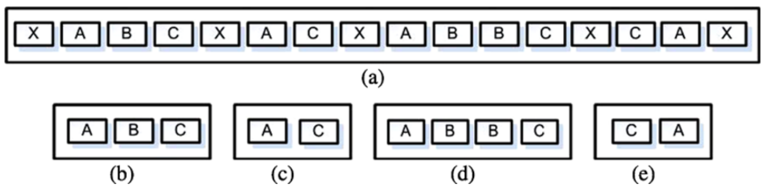


Fig. 7 Edited videos: (**a**) target; (**b**) exact query (Q1); (**c**) removal query (Q2); (**d**) inserted query (Q3); and (**e**) removal and re-ordering query (Q4)

Table 5 Temporal order similarity for **InsHit** and **RemHit**

(a)

			Target video frames			
			1	4	2	3
Query frames	$i \setminus j$	0	1	2	3	4
	0	0	0	0	0	0
1	1	0	1	1	1	1
2	2	0	1	1	2	2
3	3	0	1	1	2	**3**

(b)

			Target video frames	
			2	1
Query frames	$i \setminus j$	0	1	2
	0	0	0	0
1	1	0	0	1
2	2	0	1	1
3	3	0	1	**1**

Table 6 Video corpora

Video dataset	Time length	Video queries	Frame rate	Genre
TV broadcast 1	1h 00 min 04 s	08	30 fps	News
TV broadcast 2	35 min 02 s	02	30 fps	Cartoon
TV broadcast 3	31 min 50 s	03	30 fps	Series
TV broadcast 4	33 min 13 s	05	30 fps	Series
TV broadcast 5	30 min 27 s	07	30 fps	General
Internet retrieved video	19 min 52 s	21	25 fps	Advertisement
Total	3h 30 min 29 s	46	–	

(iv) To understand its behavior when facing video edit operations; and
(v) To understand the performance issues

5.1 Experimental setup

The video dataset has been split into two sub-sets, since two different experiments were conducted:

(i) Real videos; and
(ii) Edited videos

The first one is used to analyze the implemented methods, our method and Shen's method (which was fully implemented from scratch), and consists of TV broadcast, recorded directly and continuously from a Brazilian cable TV channel, and Internet retrieved video. The first dataset is composed of videos of different genres, like cartoon, news, advertisement and series. Table 6 shows information about the dataset (including video length and the number of queries)—there are 46 query videos without repetition. The experiments searched for 92 occurrences of video clips in our dataset (54 for TV broadcast and 38 for Internet retrieved video), with lengths varying from 9 to 61 seconds. It is important to note that we extracted some video queries from the original videos to create our query video dataset.

The second set consists of a video with 11:05 minutes. Consider the target video sequence T illustrated in Fig. 7(a) composed of small video subsequences. In that sequence, the X represents a don't-care (i.e., any video subsequences that could be ignored during the search). From the target video sequence, we extracted three parts, A, B and C, with 11 sec, 12 sec and 6 sec, respectively, as illustrated in Fig. 7(b). Afterwards, we produce four video subsequences from these segments:

(i) An exact query (Q1)
(ii) A query with a removal operation (Q2)
(iii) A query with an insertion operation (Q3); and
(iv) A query with a removal operation and temporal reordering (Q4)

Figures 7(b), 7(c), 7(d) and 7(e) illustrate exact, removal, insertion and removal with reordering queries, respectively, obtained by editing the original video subsequence. This dataset is used in the experiment that aims to analyze the effectiveness of our method to deal with video edit operation.

In Table 7, we present all values of the parameters used in our experiments. For simplicity, some parameters are written as relative values with respect to query size; for example, $\lambda = 10\%$ corresponds to a maximum allowed video edit distance of 10% of the query size. In order to compare the two methods using similar features, we ignore temporal subsampling, which was considered in [26]. Shen's method has the following parameters:

(i) α, related to highest permitted number of consecutive dissimilar frames at the target with respect to the query
(ii) β, the minimum allowed video edit distance; and
(iii) δ, the similarity threshold between two frames

Table 7 Parameter setting

(a) Our method

Parameter	Values
λ	10%, 20%, 30%, 40%, 50%
δ	10%, 20%, 30%
Sub-sampling	No
Dissimilarity function	Histogram intersection

(b) Shen's method

Parameter	Values
α	10%, 20%, 30%, 40%, 50%
β	10%, 20%, 30%, 40%, 50%
δ	10%, 20%, 30%
Sub-sampling	No
Dissimilarity function	Histogram intersection

5.2 Precision–Recall analysis

In order to evaluate the results, it is necessary to define some measures. We denote by *#Occurrences* the number of query video occurrences, by *#Video clip identified* the number of query video occurrences that are properly identified and by *#Falses* the number of video occurrences that do not represent a correct identification. Based on these values, we consider the following quality measures.

Definition 13 (Recall and precision rates) The recall rate represents the ratio of correct and the precision value relates correct to false detections. They are given by

$$\mathcal{R} = \frac{\mathit{Video\ clip\ identified}}{\#\mathit{Occurrences}}; \quad \text{(Recall)} \tag{9}$$

$$\mathcal{P} = \frac{\#\mathit{Video\ clip\ identified}}{\#\mathit{Falses} + \#\mathit{Video\ clip\ identified}}. \quad (\mathit{Precision}) \tag{10}$$

Precision–Recall (PR) curves give a more informative picture of the algorithm performance, since they group information about hits, miss, false positives and false negatives [5]. An optimal algorithm should have a precision-recall value of (1,1) (which means 100% of recall with 100% of precision), i.e. it managed to identify all video clip occurrences with no false positives.

Definition 14 (F measure) The F measure is a weight harmonic mean of precision and recall rates, and it is given by

$$\mathrm{F} = \frac{2 \times \mathcal{P} \times \mathcal{R}}{\mathcal{P} + \mathcal{R}}. \tag{11}$$

Table 8 shows the quality measures for our approach related to precision and recall rates, together with F measures for different values of edit distance (λ) and threshold (δ).

Table 8 Precision and recall rates

Video edit	Threshold value (δ)								
distance	10%			20%			30%		
(λ)	$\mathcal{R}$	$\mathcal{P}$	F	$\mathcal{R}$	$\mathcal{P}$	F	$\mathcal{R}$	$\mathcal{P}$	F
10%	51%	100%	68%	86%	99%	92%	91%	92%	92%
20%	62%	100%	77%	90%	93%	92%	92%	79%	85%
30%	67%	100%	81%	91%	91%	91%	96%	77%	85%
40%	73%	99%	84%	91%	88%	90%	96%	70%	81%
50%	84%	99%	91%	95%	89%	92%	93%	62%	75%

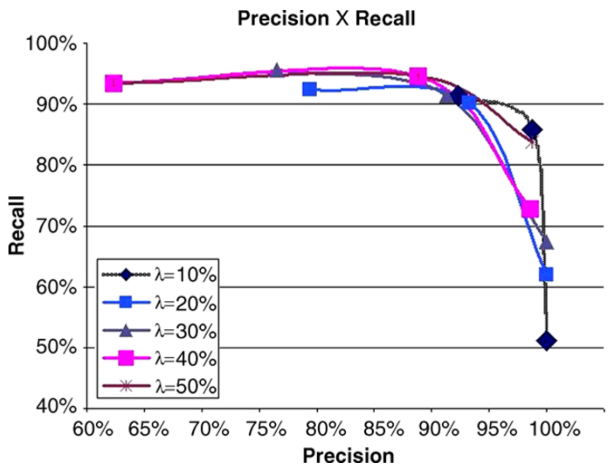


Fig. 8 Precision-recall curves

Figure 8 illustrates the precision-recall curves. Best results are associated with $\lambda = 20\%$ and $\delta = 20\%$ and our method achieves 90% recall with 93% precision (see Table 8), which is similar to (and even better than) the approach proposed in [26] with the capacity of identifying video edit operations.

One of the features of our algorithm that contributes to those results is the use of the MCM size. The algorithm only considers a video clip position as a positive localization if the size of the MCM is at least equal to the query video size minus the maximum allowed edit distance, and the number of frame insertions in the target video clip must also be smaller than the error. Using the size of the MCM prevents the algorithm from finding versions of query video that have additional frames/shots, or target video clips with suppressed parts (frames/shots) of the query video, when we search for exact video, i.e., $\lambda = 0$. A relaxation of the edit distance may imply finding edited video clips, but it may also contribute to a rising number of false positives, as can be seen in Table 8 and Fig. 8.

Another parameter that is related to high precision detection is the threshold (δ), which was kept very low during all experiments.

Table 9 Average shift value percentage

Video edit distance (λ)	Threshold value (δ) 10%	20%	30%
10%	94%	87%	71%
20%	97%	84%	65%
30%	98%	81%	59%
40%	99%	76%	52%
50%	100%	72%	45%

5.3 Tuning of parameters

One should notice that both parameters, similarity threshold (δ) and video edit distance (λ), have influenced the size of frame similarity graph ($G^{\delta}_{k,\lambda}$), and consequently, the quality of the results.

5.3.1 Tuning of video edit distance

The recall and precision rates are directly influenced by the tuning of video edit distance value. As one can see in Table 8, the recall rate increases together with high video edit distance values. This should be expected since a relaxation of the edit distance may imply finding a great number of query video occurrences (with or without editing). Conversely, the precision rate decreases for high values of the edit distance, since more false positives may be identified. With respect to the shift value (see Table 9), for high values of the edit distance, the shift value decreases, since we need less frames to find a hit occurrence according to the query video, except when using a very low value for the similarity threshold ($\delta = 10\%$).

5.3.2 Tuning of similarity threshold

The recall and precision rates are also directly influenced by the tuning of the similarity threshold value. As one can observe in Table 8, the recall rate decreases when the similarity threshold value becomes lower. That should be expected, since we impose more restrictions on the similarity between frames. Conversely, the precision rate increases for low values of similarity threshold, since less false positives are identified. With respect to the shift value (see Table 9), for high values of similarity threshold, the shift value decreases, since we need more frames to find a hit occurrence according to the query video.

According to Table 8, an outlier for the recall rate is identified for $\lambda = 50\%$ and $\delta = 30\%$. This can be explained by the large number of edges in the frame similarity graph, due to both huge relaxation of the similarity threshold and the size of target video clip used for matchings (as a consequence of a large video edit distance).

5.4 Comparative analysis

In [26], the authors have proposed two algorithms for clip localization, an optimum and a sub-optimum approach based on bipartite graph matching analysis, and they present good experimental results. Considering that our method is based on the calculation of the optimal MCM for a bipartite graph, we decided to implement the optimum method proposed by Shen et al. The main differences between the two methods are:

(i) Filtering step
(ii) Bipartite graph construction; and
(iii) The shift strategy

In order to reduce the number of constructed graphs, Shen's method filters out some video segments which did not have some properties. Afterwards, a graph is constructed for each dense segment which was not filtered out in the previous step. As discussed before, our method adopts a shift strategy while Shen's approach does not use such a mechanism. Thus, our method does not have any filtering step, and the graph construction depends on the shift strategy which helps us in localizing all frames that must match.

Some results for Shen's method are illustrated in Table 10, and as one can see the best results according to F1 measure are obtained for $\alpha = 10\%$, $\beta = 50\%$ and $\delta = 10\%$. Both methods (ours and Shen's) present quite similar results, with respect to precision, recall and F1 measure. However, our method produces more detailed results, since it can describe which video edit operations were done. One should note that the recall rate decreases for high values of α. With respect to β, the recall rate increases for high values.

5.5 Video editing analysis

As described before, our method has two main features:

(i) Temporal reordering invariance; and
(ii) Video editing invariance

The former can be considered as a special case of the latter. In order to illustrate these features, we conduct an experiment with the second dataset (which was described before).

The query videos represent video edit operations and will be used to show the behavior of our approach under temporal reordering and/or video editing; e.g., Q4 is generated by temporal reordering of Q2. Each query video occurs exactly once; however, depending on the video edit distance some other occurrences will be identified. For example, a search for Q1 with $\lambda = 50\%$ will return as hits all four subsequences Q1, Q2, Q3 and Q4 (see the groundtruth for edited video queries in Table 11). Moreover, one should pay attention to the fact that subsequences Q2 and Q4 have the same frames in a different order.

Table 10 Precision and recall rates for Shen's method

(a) $\alpha = 10$

Video edit distance (β)	Threshold value (δ) 10% $\mathcal{R}$	10% $\mathcal{P}$	10% F	20% $\mathcal{R}$	20% $\mathcal{P}$	20% F	30% $\mathcal{R}$	30% $\mathcal{P}$	30% F
10%	52%	100%	69%	82%	95%	88%	85%	85%	85%
20%	63%	100%	77%	87%	92%	89%	89%	77%	82%
30%	67%	100%	81%	89%	87%	88%	89%	70%	78%
40%	75%	100%	86%	89%	86%	88%	89%	64%	74%
50%	85%	99%	91%	91%	79%	85%	90%	52%	66%

(b) $\alpha = 30$

Video edit distance (β)	Threshold value (δ) 10% $\mathcal{R}$	10% $\mathcal{P}$	10% F	20% $\mathcal{R}$	20% $\mathcal{P}$	20% F	30% $\mathcal{R}$	30% $\mathcal{P}$	30% F
10%	51%	100%	68%	76%	92%	83%	70%	82%	75%
20%	62%	100%	77%	80%	88%	84%	72%	77%	74%
30%	66%	100%	80%	83%	85%	84%	72%	68%	70%
40%	74%	100%	85%	83%	84%	84%	72%	59%	65%
50%	84%	99%	91%	85%	78%	81%	73%	48%	58%

(c) $\alpha = 50$

Video edit distance (β)	Threshold value (δ) 10% $\mathcal{R}$	10% $\mathcal{P}$	10% F	20% $\mathcal{R}$	20% $\mathcal{P}$	20% F	30% $\mathcal{R}$	30% $\mathcal{P}$	30% F
10%	48%	98%	64%	68%	90%	78%	55%	78%	65%
20%	58%	96%	72%	73%	86%	79%	57%	73%	64%
30%	62%	97%	75%	75%	83%	79%	57%	63%	60%
40%	70%	97%	81%	75%	82%	78%	57%	54%	55%
50%	78%	96%	86%	77%	76%	76%	58%	47%	52%

(d) average values for α between [10%, 50%]

Video edit distance (β)	Threshold value (δ) 10% $\mathcal{R}$	10% $\mathcal{P}$	10% F	20% $\mathcal{R}$	20% $\mathcal{P}$	20% F	30% $\mathcal{R}$	30% $\mathcal{P}$	30% F
10%	50%	99%	67%	76%	93%	83%	70%	82%	75%
20%	61%	99%	75%	80%	89%	84%	73%	76%	74%
30%	65%	99%	79%	83%	86%	84%	73%	67%	70%
40%	73%	99%	84%	83%	84%	83%	73%	59%	65%
50%	82%	98%	89%	85%	78%	81%	74%	49%	59%

As can be seen in Table 12, the recall rate of this experiment is 100%; however, the precision is not so high, mainly due to the size of video subsequence Q2 (and Q4). For example, there are false positives related to query video Q1 with $\lambda = 40\%$, since the size of Q2 (and Q4) is only 0.4 seconds ($\equiv$12 frames) smaller than the size of Q1 in this case, i.e., when the video edit distance is set to 40% ($\lambda = 40\%$). Table 13 represents minimum and maximum sizes (in seconds) of the video subsequences searched in the target video according to the video edit distance.

5.6 Performance issues

The performance of the algorithm is directly related to the shift value that is adopted during the identification procedure. Once the size of the MCM is calculated, the conservative approach used in this work defines the shift value as a function of the number of unmatched frames. This approach assumes that all mismatches occurred in the beginning of the frame similarity graph. In other words, the algorithm assumes that all matches might be used in the next iteration, preventing the algorithm from shifting at larger steps.

Table 9 shows the average shift value of the performed experiments. A number of 100% means that the shift value is equal to the query video length. It can be seen that the average shift value is higher for lower values of δ when the edit distance is the same. This effect is expected, since a lower value of δ increases the number of mismatched frames. In addition, the shift value is also higher for lower edit distances, since the expected number of mismatched frames is higher.

6 Conclusions

In this work, we present an approach capable of coping with a general video clip localization problem, allowing insertion, removal and replacement operations on the video clip. Our method not only solves the approximate subsequence localization problem but also gives a precise description of the set of operations that are necessary to transform the query video into the target content. In order to do so, it uses a bipartite graph matching to identify query location candidates. The main contributions of our work are the application of a simple and efficient distance to solve the subsequence identification problem along with the definition of a hit function that identifies precisely which operations were used in query transformation.

According to the experimental results, our method performance (90% recall with 93% precision) is similar to (and even better than) the approach proposed in [26] but it is done without preprocessing of the target video. However, subsequence identification results may be highly dependent on the testing material, which is usually scarce and not especially representative. Moreover, choosing an appropriate feature that enhances performance of a matching algorithm is not a trivial task. Therefore, as future work, we will consider more relevant/robust descriptors and study their impact on the precision and recall rates.

Finally, we also intend to adapt our method to cope with another important problem known as near-duplicate video

Table 11 Groundtruth for edited video queries

Query videos	Video edit distance (λ) 0%	10%	20%	30%	40%	50%
Query 1 (Q1)	Q1	Q1	Q1	Q1	Q1	Q1, Q2, Q3, Q4
Query 2 (Q2)	Q2, Q4	Q2, Q4	Q2, Q4	Q2, Q4	Q2, Q4	Q2, Q4
Query 3 (Q3)	Q3	Q3	Q3	Q1, Q3	Q1, Q3	Q1, Q3
Query 4 (Q4)	Q2, Q4	Q2, Q4	Q2, Q4	Q2, Q4	Q2, Q4	Q2, Q4

Table 12 Precision and recall rates for edited video queries

Video edit	Threshold value (δ)								
distance	10%			20%			30%		
(λ)	$\mathcal{R}$	$\mathcal{P}$	F	$\mathcal{R}$	$\mathcal{P}$	F	$\mathcal{R}$	$\mathcal{P}$	F
10%	100%	100%	100%	100%	100%	100%	100%	42%	60%
20%	100%	100%	100%	100%	85%	92%	100%	40%	57%
30%	100%	63%	73%	100%	63%	77%	100%	43%	60%
40%	100%	58%	73%	100%	70%	82%	100%	43%	60%
50%	100%	83%	90%	100%	62%	76%	90%	55%	68%

Table 13 Minimum and maximum sizes of video subsequences (in seconds) searched in target video

	Video edit distance (λ) 0%	10%	20%	30%	40%	50%
Q1	29	[26.1,31.9]	[23.2,34.8]	[20.3,37.7]	[17.4,40.6]	[14.5,43.5]
Q2	17	[15.3,18.7]	[13.6,20.4]	[11.9,22.1]	[10.2,23.8]	[8.5,25.5]
Q3	41	[36.9,45.1]	[32.8,49.2]	[28.7,53.3]	[24.6,57.4]	[20.5,61.5]
Q4	17	[15.3,18.7]	[13.6,20.4]	[11.9,22.1]	[10.2,23.8]	[8.5,25.5]

clip/copy detection. Near-duplicate video copies are those video copies derived from the same original copy, by some global transformation such as video re-formatting and color shifting, or some local changes such as frame editing; and it has a wide range applications in TV broadcasting monitoring, copyright enforcement, and content-based video search.

Acknowledgements The authors are grateful to PUC Minas—Pontifícia Universidade Católica de Minas Gerais, CT-Info/MCT/CNPq (Project 551005/2007-6) and FAPEMIG (Project CEX PPM 126/08) for the financial support of this work. We are also grateful to the anonymous reviewers for their various contributions.

References

1. Adjeroh DA, Lee MC, King I (1999) A distance measure for video sequences. Comput Vis Image Underst 75(1–2):25–45
2. Bimbo AD (1999) Visual information retrieval. Morgan Kaufmann, San Francisco
3. Chen L, Chua TS (2001) A match and tiling approach to content-based video retrieval. In: ICME. IEEE Comput Soc, Los Alamitos
4. Chiu CY, Wang HM (2010) Time-series linear search for video copies based on compact signature manipulation and containment relation modeling. IEEE Trans Circuits Syst Video Technol 20(11):1603–1613
5. Davis J, Goadrich M (2006) The relationship between precision-recall and roc curves. In: Proc of the 23rd international conference on machine learning, Pittsburgh, PA
6. Deng L, Jin LZ (2010) A video retrieval algorithm based on ensemble similarity. In: IEEE international conference on intelligent computing and intelligent systems (ICIS), vol 3, pp 638–642
7. Deselaers T, Keysers D, Ney H (2004) Features for image retrieval—a quantitative comparison. In: DAGM 2004, pattern recognition, 26th DAGM symposium, Tübingen, Germany. Lecture notes in computer science, pp 228–236
8. Diakopoulos N, Volmer S (2003) Temporally tolerant video matching. In: Proc of the ACM SIGIR Workshop on multimedia information retrieval, Toronto, Canada
9. do Patrocínio ZKG Jr, Guimarães SJF, de Paula HB (2007) Bipartite graph matching for video clip localization. In: SIBGRAPI, pp 129–138
10. Gauch J, Shivadas A (2005) Identification of new commercials using repeated video sequence detection. In: International conference on image processing, vol III, pp 1252–1255
11. Gauch JM, Shivadas A (2006) Finding and identifying unknown commercials using repeated video sequence detection. Comput Vis Image Underst 103:80–88
12. Guimarães SJF, do Patrocínio ZKG Jr (2010) Identification and analysis of video subsequence using bipartite graph matching. In: 16th WebMedia Brazilian symposium on multimedia and the web
13. Guimarães SJF, Kelly R, Torres A (2006) Counting of video clip repetitions using a modified bmh algorithm: preliminary results. In: Proc of the IEEE ICME, Toronto, Canada, pp 1065–1068
14. Horspool RN (1980) Practical fast searching in strings. Softw Pract Exp 10(6):501–506
15. Huang Z, Shen HT, Shao J, Cui B, Zhou X (2010) Practical online near-duplicate subsequence detection for continuous video streams. IEEE Trans Multimed 12(5):386–398
16. Jain AK, Vailaya A, Xiong W (1999) Query by video clip. Multimed Syst 7(5):369–384
17. Joly A, Frelicot C, Buisson O (2005) Content-based video copy detection in large databases: A local fingerprints statistical similarity search approach. In: International conference on image processing, vol I, pp 505–508
18. Kim Y, Chua T (2005) Retrieval of news video using video sequence matching. In: MMM, pp 68–75

19. Lienhart R, Effelsberg W, Jain R (1999) Visualgrep: A systematic method to compare and retrieve video sequences. Multimed Tools Appl 10(1):47–72
20. Naturel X, Gros P (2005) A fast shot matching strategy for detecting duplicate sequences in a television stream. In: Proceedings of the 2nd ACM SIGMOD international workshop on computer vision meets DataBases
21. Navarro G (2001) A guided tour to approximate string matching. ACM Comput Surv 33(1):31–88
22. Papadimitriou CH, Steiglitz K (1982) Combinatorial optimization: algorithms and complexity. Prentice-Hall, Upper Saddle River
23. Pedro JS, Denis N, Domínguez S (2005) Video retrieval using an edl-based timeline. In: Marques JS, de la Blanca NP, Pina P (eds) IbPRIA (1). Lecture notes in computer science, vol 3522. Springer, Berlin, pp 401–408
24. Peng Y, Ngo CW (2006) Clip-based similarity measure for query-dependent clip retrieval and video summarization. IEEE Trans Circuits Syst Video Technol 16(5):612–627
25. Rubner Y, Puzicha J, Tomasi C, Buhmann JM (2001) Empirical evaluation of dissimilarity measures for color and texture. Comput Vis Image Underst 84(1):25–43
26. Shen HT, Shao J, Huang Z, Zhou X (2009) Effective and efficient query processing for video subsequence identification. IEEE Trans Knowl Data Eng 21(3):321–334
27. Tan YP, Kulkarni SR, Ramadge PJ (1999) A framework for measuring video similarity and its application to video query by example. In: ICIP (2), pp 106–110
28. Tseng BL, Lin CY, Smith JR (2004) Using MPEG-7 and MPEG-21 for personalizing video. IEEE Multimed 11(1):42–53
29. Wei S, Zhao Y, Zhu C, Xu C, Zhu Z (2011) Frame fusion for video copy detection. IEEE Trans Circuits Syst Video Technol 21(1):15–28