Available online at www.sciencedirect.com
SCIENCE DIRECT®
Real-Time Imaging 9 (2003) 73–98
www.elsevier.com/locate/rti

A review of real-time segmentation of uncompressed video sequences for content-based search and retrieval

Sébastien Lefèvre[a,b,*], Jérôme Holler[a], Nicole Vincent[a]

[a] *Laboratoire d'Informatique, E3i, Université de Tours, 64, Avenue Portalis, 37200 Tours, France*
[b] *AtosOrigin, 19, Rue de la Vallée Maillard, BP 1311, 41013 Blois Cedex, France*

Abstract

We present in this paper a review of methods for segmentation of uncompressed video sequences. Video segmentation is usually performed in the temporal domain by shot change detection. In case of real-time segmentation, computational complexity is one of the criteria which has to be taken into account when comparing different methods. When dealing with uncompressed video sequences, this criterion is even more significant. However, previous published reviews did not involve complexity criterion when comparing shot change detection methods. Only recognition rate and ability to classify detected shot changes were considered. So contrary to previous reviews, we give here the complexity of most of the described methods. We review in this paper an extensive set of methods presented in the literature and classify them in several parts, depending on the information used to detect shot changes. The earliest methods were comparing successive frames by relying on the most simple elements, that is to say pixels. Comparison could be performed on a global level, so methods based on histograms were also proposed. Block-based methods have been considered to process data at an intermediate level, between local (using pixels) and global (using histograms) levels. More complex features can be involved, resulting in feature-based methods. Alternatively some methods rely on motion as a criterion to detect shot changes. Finally, different kinds of information could be combined together in order to increase the quality of shot change detection. So our review will detail segmentation methods based on the following information: pixel, histogram, block, feature, motion, or other kind of information.

Contents

*Corresponding author. Laboratoire d'Informatique, E3i, Université de Tours, 64, Avenue Portalis, 37200 Tours, France.
E-mail addresses: lefevre@univ-tours.fr (S. Lefèvre), vincent@univ-tours.fr (N. Vincent).

1077-2014/03/$ - see front matter © 2003 Elsevier Science Ltd. All rights reserved.
doi:10.1016/S1077-2014(02)00115-8

1. Introduction

Multimedia information is more and more used, thanks mostly to increasing computation resources. One of the main processings needed when dealing with multimedia data is multimedia sequence indexing. The importance of this research field is shown by the number of recent communications and publications on the subject. In order to index multimedia data, we may need a preprocessing, the aim of which is to temporally segment the videos, that is to say detect the shot changes present in the video sequences.

The number of shot change detection methods is now important and several reviews of these methods have been made [1–14]. These reviews often present the different methods and their efficiency based on some quality measures. So they are very useful when one wants to select and implement a shot change detection method for a global video processing which could be done on-line. When processing has to be done off-line, the selection of a particular method should also consider computation time. This is especially true when dealing with uncompressed video sequences which contain a huge quantity of data. If the method has to be implemented on common hardware architecture, computation time is directly linked with complexity of the method. So in this paper we review most of the methods presented in the literature and focus on their complexity.

In the first part of this paper, before we present a large number of methods, we situate our contribution. We give some references to previous reviews on the subject. It is also necessary to recall and describe the different forms a shot change can take. We give a few details on quality evaluation and introduce the way we compute the complexity of the methods. We also define the notations used in this paper. The following parts will deal with a description of the encountered methods. Finally some conclusions will be made about shot change detection methods.

1.1. Related works

Several reviews have already been published in the literature. Ahanger and Little [1] discuss the

requirements and global architectures for video indexing. Some video segmentation methods are presented in this framework. Idris and Panchanathan [2] deal with image and video indexation. Image features and video processing algorithms useful for indexation are described. Shot change detection is one of the video processing needed to characterize a video sequence. Brunelli et al. [3] also present video indexation, and describe main algorithms including shot change detection. They are particularly involved in a video indexing system. Aigrain et al. [4] review techniques for video content analysis. Shot change detection is one of these techniques. Koprinska and Carrato [5] review algorithms for shot change detection and camera operation recognition. Lienhart [6] presents the different kinds of shot changes and some dedicated detection methods. Jiang et al. [7] propose a review based on three categories which are uncompressed video-, compressed video- and model-based algorithms.

Some reviews compare a few algorithms based on author's implementation. Boreczky and Rowe [8] compare the performances of five algorithms using a common evaluation methodology. Lienhart [9] compares the methods and characterizes their ability to correctly determine the kind of shot changes that have been detected. Dailianas et al. [10] compare several segmentation methods and introduce a filtering algorithm in order to limit the false detections. Some information is also given about the complexity of the evaluated methods. Yusoff et al. [11] compare several methods and propose improved versions using an adaptive threshold.

Finally, some papers are reviewing only a specific kind of method, as those from Gargi et al. [12,13] which are, respectively, dedicated to colour histogram-based methods and MPEG and motion-based methods for temporal segmentation of video. Mandal et al. [14] focused on methods working in compressed domain.

Contrary to other approaches, we review and compare in this paper uncompressed video segmentation methods following their computational complexity and not their detection or error rates, which has already done in papers presented in this section. We base the classification of the presented methods on the basic elements used in the segmentation process: pixels, histograms, blocks, features, motion, and combination of several approaches.

1.2. Shot change description

A shot is defined as a continuous video acquisition (with the same camera). When the video acquisition is done with another camera, there is a shot change. The simplest way to perform a change between two shots is called a *cut*. In this case, the last frame of the first video sequence is directly followed by the first frame of the second video sequence. This kind of shot change is also called abrupt change. Because of their simplicity, cuts are often the easiest shot changes to be detected.

More complex shot changes are now available for video editing, thanks to improvement of the video production softwares. Instead of cutting and pasting the second video next to the first one, it is possible to insert an effect, as a *wipe*, a *fade*, or a *dissolve*. A wipe is obtained by progressively replacing the old image by the new one, using a spatial basis. A dissolve is a transition where all the images inserted between the two video sequences contain pixels whose values are computed as linear combination of the final frame of the first video sequence and the initial frame of the second video sequence. Fades are special cases of dissolve effects, where a monochrome frame replaces the last frame of the first shot (*fade in*) or the first frame of the second shot (*fade out*). There are also other kinds of effects (combining, for example, wipe and zoom), but actually most of the shot change detection methods are concerned only with the effects described previously in their indexing task.

1.3. Quality evaluation

The recognition rate is the most used quality criterion in order to compare shot change detection methods. Some work has been done to define some standard quality measures and to discuss existing ones [12,15–17]. Most of the time, quality is evaluated thanks to computation of the quantity of correctly detected shot changes, missed shot changes, and false detections. Indeed, to be fair, the evaluation should be achieved on a universal benchmark. This is not the case.

A similar background for two consecutive shots often results in missing the shot change. False detections appear when there is a significant content change. Camera motion, moving objects, illumination changes can be sources of false detections.

Our review does not focus on this aspect of quality evaluation of the methods. We will insist here on complexity of the methods, because this work has not been done for a complete set of methods yet.

In order to compare video segmentation methods, it is also possible to take into account the number of thresholds or parameters which have to be set. Learning capabilities of these thresholds or parameters can also be used as comparison criteria. Comparison of uncompressed video segmentation methods based on these criteria is out of the scope of this paper.

1.4. Complexity computation

As mentioned previously, quality evaluation is not the only criterion to evaluate and compare shot change detection methods if we are concerned with real-time

(or near real-time) processing using common hardware. In this case, one should also have to consider complexity of the evaluated methods. A work on complexity of shot change detection methods has been done by Dailianas et al. [10] but it was limited to few methods.

In this paper, the complexity was computed considering a cost of one for any logical or arithmetic operation (including absolute value). We do not consider other operations as, for example, memory access time or branching operations (e.g. if … then). In order to compute the complexity of the methods, we define $\mathscr{N}$ as the possible number of levels for pixel value, which is equivalent to the number of bins for histogram-based methods. We also introduced $\mathscr{P}$ as the number of pixels per frame. In case of block-based methods, we use the notation $\mathscr{B}$ to represent the number of blocks defined in the frame.

Complexity measurements given in this paper represent the number of operations needed to process one frame. Temporal subsampling of the video sequences is not taken into account. However, when values obtained for a given frame can be used to process the next frame, complexity measurements are optimised and given considering the use of previous results.

1.5. Notations

Video sequences are composed of successive frames or images. We define I_t the frame of the video obtained at time t. So it is possible to define $P(I_t, i, j)$ the intensity of the pixel with coordinates i and j in the frame I_t. We assume that the size of the images is X-by-Y pixels, so we have $1 \leqslant i \leqslant X$ and $1 \leqslant j \leqslant Y$.

When methods are dealing with colour images, the notation $P(I_t, C_k, i, j)$ will be used. C_k represents the colour component numbered k. As an example, we can consider that C_1, C_2, and C_3, respectively, represent the R, G, and B components in the RGB colour space. So $P(I_t, C_k, i, j)$ represents the value of the colour component C_k for the pixel with coordinates i and j in frame I_t.

Some methods deal with histograms. So we define $H(I_t, v)$ the number of pixels of the image I_t with an intensity equal to v, with $v \in [0, V]$ where V is the maximum gray-level value. If we consider colour images, indexing methods can use several histograms, one for each colour component. We then use the notation $H(I_t, C_k, v)$ to define the number of pixels with an intensity value of v for the colour component C_k in the image I_t.

Another common approach for video segmentation is to use block-sampled images. Let us note B the number of blocks b in each frame.

Finally, because a lot of methods use some thresholds for shot change detection, we have also noted T some threshold fixed by the user. Several authors [18,19] propose a learning procedure in order to use an appropriate threshold value.

As can easily be imagined from this introductory part, the works dealing with video sequence segmentation are quite numerous. We report 93 entries in our bibliography. Of course, some others exist but we consider covering the main ways used to solve the problem. Even if the complexity of the methods is naturally increasing along time we have not chosen a chronological thread to present the various methods. Rather we have sorted them according to the basic elements they are rely on. We have organized them from the most simple, the pixel in the image, to the most sophisticated ones, those that use a combination of methods. More precisely we have distinguished six large categories characterized by the respective use of:

- pixel characterization,
- histogram of the frames,
- partition of the image in blocks,
- features,
- motion during the sequence, and
- combination of approaches.

2. Pixel-based methods

Shot change detection can be performed by comparing successive frames. The simplest way to compute the dissimilarity between two frames is to compare corresponding pixels from two successive images [20]. As we will see, some improvements of the initial pixel comparison have been proposed. First, we present the methods considering two consecutive frames and then those that extend the study to a longer temporal interval.

2.1. Pixel comparison between two successive frames

One of the first method described in literature was from Nagasaka and Tanaka [20] in 1991. Shot changes are detected using a simple global interframe difference measure, defined as

$$\text{Detection if : } \left(\left| \sum_{i=1}^{X} \sum_{j=1}^{Y} P(I_t, i, j) - \sum_{i=1}^{X} \sum_{j=1}^{Y} P(I_{t-1}, i, j) \right| \right) > T \quad (1)$$

resulting in $\mathscr{O}(\mathscr{P})$ operations per frame (as the second term of the difference has been already obtained after the processing of the previous frame I_{t-1}).

Nagasaka and Tanaka [20] also introduced a shot change detection method based on pixel pair difference called template matching. For every two successive frames, differences of intensities are computed on pixels having the same spatial position in the two frames. Then

the cumulated sum of differences is compared to a fixed threshold in order to determine if a shot change has been detected:

Detection if :

$$\left(\sum_{i=1}^{X}\sum_{j=1}^{Y}|P(I_t,i,j)-P(I_{t-1},i,j)|\right)>T. \quad (2)$$

The number of operations per frame is equal to $O(3\mathscr{P})$. A colour version (of higher complexity $O(9\mathscr{P})$) has also been presented:

Detection if :

$$\left(\sum_{i=1}^{X}\sum_{j=1}^{Y}\sum_{k=1}^{3}|P(I_t,C_k,i,j)-P(I_{t-1},C_k,i,j)|\right)>T. \quad (3)$$

A couple of years later, Zhang et al. [21] compared the pixels of two successive frames on a Boolean basis. The fact that pixels are different is noted:

$$D(I_t,I_{t-1},i,j)=\begin{cases}1 & \text{if } P(I_t,i,j)\neq P(I_{t-1},i,j),\\ 0 & \text{otherwise,}\end{cases} \quad (4)$$

for the gray-level case and requires one operation per couple of pixels. Definition is quite similar for colour images. In order to allow some variations on pixel intensities, a better (but more complex as it needs three operations instead of one) definition is:

$$D(I_t,I_{t-1},i,j)=\begin{cases}1 & \text{if } |P(I_t,i,j)-P(I_{t-1},i,j)|>T_D,\\ 0 & \text{otherwise,}\end{cases} \quad (5)$$

where T_D is considered as the tolerance value. The amount of different pixels is computed and is compared to a given threshold, which results in the detection or not of a shot change:

$$\text{Detection if : }\left(\sum_{i=1}^{X}\sum_{j=1}^{Y}D(I_t,I_{t-1},i,j)\right)>T \quad (6)$$

resulting in complexity of $O(2\mathscr{P})$ or $O(4\mathscr{P})$ according to the condition used to compare pixels. In order to avoid false detections due to motion in the video sequence, they also propose to smooth the images with a filter of size 3×3 before computing the D values. The filter limits the effects due to noise and camera motion.

Several other statistical measures have been proposed in the literature [22]. The normalized difference energy and the normalized sum of absolute differences can be used for shot change detection, as shown by the following equations:

Detection if :

$$\left(\frac{\sum_{i=1}^{X}\sum_{j=1}^{Y}(P(I_t,i,j)-P(I_{t-1},i,j))^2}{(\sum_{i=1}^{X}\sum_{j=1}^{Y}P(I_t,i,j)^2)(\sum_{i=1}^{X}\sum_{j=1}^{Y}P(I_{t-1},i,j)^2)}\right)>T, \quad (7)$$

Detection if :

$$\left(\frac{\sum_{i=1}^{X}\sum_{j=1}^{Y}|P(I_t,i,j)-P(I_{t-1},i,j)|}{\sum_{i=1}^{X}\sum_{j=1}^{Y}P(I_t,i,j)+\sum_{i=1}^{X}\sum_{j=1}^{Y}P(I_{t-1},i,j)}\right)>T. \quad (8)$$

These measures are, respectively, characterized by a complexity equal to $O(5\mathscr{P})$ and $O(4\mathscr{P})$. Indeed, in both methods, the second part of the denominator has been obtained after processing the previous frame I_{t-1} and so does not need to be computed once again.

2.2. Pixel intensity time variation

The previous two-frame study can be generalized by analysing variations of intensities through time. Taniguchi et al. [23] label pixels with respect to the evolution of their intensities on several successive frames. The labels used are "constant", "step (I_t)", "linear (I_{t_1},I_{t_2})", and "no label". These labels represent, respectively, pixels with constant values, pixels with a change in value at frame I_t, pixels with a progressive change in value between frames I_{t_1} and I_{t_2}, and finally pixels with random values due to motion. Two Boolean conditions $\Theta_1(I_{t_1},I_{t_2},i,j)$ and $\Theta_2(I_{t_1},I_{t_2},i,j)$ (needing, respectively, four and six operations per pixel) are introduced in order to define the constancy of a set of pixel values $P(I_t,i,j)$ with $t_1\leqslant t\leqslant t_2$:

$$\Theta_1(I_{t_1},I_{t_2},i,j)=\begin{cases}\text{true} & \text{if } \left(\max\limits_{t_1\leqslant t\leqslant t_2}P(I_t,i,j)-\min\limits_{t_1\leqslant t\leqslant t_2}P(I_t,i,j)\right)<T,\\ \text{false} & \text{otherwise,}\end{cases} \quad (9)$$

$$\Theta_2(I_{t_1},I_{t_2},i,j)=\begin{cases}\text{true} & \text{if } \left(\begin{array}{l}\max\limits_{t_1\leqslant t\leqslant t_2}(P(I_t,i,j)+(t-t_1)\theta_{t_1,t_2})\\ -\min\limits_{t_1\leqslant t\leqslant t_2}(P(I_t,i,j)+(t-t_1)\theta_{t_1,t_2})\end{array}\right)<T,\\ \text{false} & \text{otherwise,}\end{cases} \quad (10)$$

with θ_{t_1,t_2} defined as

$$\theta_{t_1,t_2}=\frac{|P(I_{t_1},i,j)-P(I_{t_2},i,j)|}{t_2-t_1} \quad (11)$$

which requires three operations per pixel to be obtained since t_2-t_1 is a constant value, computed only once per couple of frames. These similarity conditions Θ_1 and Θ_2 are then used to determine the label $L(I_{t_0},I_{t_f},i,j)$ of each pixel of a video sequence involving $f+1$ frames, using

the following scheme:

$$L(I_{t_0}, I_{t_f}, i, j) = \begin{cases} \text{constant} & \text{if } \Theta_1(I_{t_0}, I_{t_f}, i, j), \\ \text{step}(I_t) & \text{if } \begin{pmatrix} & \Theta_1(I_{t_0}, I_{t-1}, i, j) \\ \wedge & \Theta_1(I_t, I_{t_f}, i, j) \\ \wedge \ \neg & \Theta_1(I_{t-1}, I_t, i, j) \end{pmatrix}, \\ \text{linear}(I_{t_1}, I_{t_2}) & \text{if } \begin{pmatrix} & \Theta_1(I_{t_0}, I_{t_1}, i, j) \\ \wedge & \Theta_1(I_{t_2}, I_{t_f}, i, j) \\ \wedge \ \neg & \Theta_1(I_{t_1}, I_{t_2}, i, j) \\ \wedge & \Theta_2(I_{t_1}, I_{t_2}, i, j) \end{pmatrix}, \\ \text{no label} & \text{otherwise}, \end{cases} \tag{12}$$

which can also be defined as

$$L_{label}(I_{t_0}, I_{t_f}, i, j) = \begin{cases} 1 & \text{if } L(I_{t_0}, I_{t_f}, i, j) \text{ is of kind ``label''}, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

Quantities of pixels associated with each label are computed. Cuts (respectively dissolves) are detected thanks to the analysis of the ratio between quantity of pixels labeled "step" (respectively "linear") and quantity of pixels labeled (i.e. with a label different of "no label"). A cut is detected at frame I_t if

$$\text{Detection if : } \left(\frac{\sum_{i=1}^{X} \sum_{j=1}^{Y} L_{step(I_t)}(I_{t_0}, I_{t_f}, i, j)}{XY - \sum_{i=1}^{X} \sum_{j=1}^{Y} L_{no\ label}(I_{t_0}, I_{t_f}, i, j)} \right) > T. \tag{14}$$

A dissolve is detected between frames I_{t_1} and I_{t_2} if

$$\text{Detection if : } \left(\frac{\sum_{i=1}^{X} \sum_{j=1}^{Y} L_{linear(I_{t_1}, I_{t_2})}(I_{t_0}, I_{t_f}, i, j)}{XY - \sum_{i=1}^{X} \sum_{j=1}^{Y} L_{no\ label}(I_{t_0}, I_{t_f}, i, j)} \right) > T. \tag{15}$$

Considering a number of operations per pixel, respectively, equal to atleast four for Eq. (9), six for Eq. (10), three for Eq. (11), and two for Eq. (14) or (15), the overall complexity is then equal to $O(15\mathscr{P})$.

Lawrence et al. [24] use evolution of temporal derivative of the pixel intensities as a criterion for shot change detection. First pixels with high spatial derivative are discarded in order to avoid motion effect. A pixel $P(I_t, i, j)$ is considered if and only if the following condition holds:

$$\max(|P(I_t, i, j) - P(I_t, i-1, j)|, |P(I_t, i, j) - P(I_t, i, j-1)|) < T. \tag{16}$$

This condition requires six operations per pixel to be verified. A convolution process involving remaining pixels and a Gaussian mask is then performed to obtain temporal derivative of $P(I_t, i, j)$. Absolute values of these derivatives are summed up in order to define the distance measure (needing two operations per pixel) which will be analysed through time. Shot boundaries correspond to local maxima of this distance measure. False detections due to noise or motion are limited if the neighbourhoods of the local maxima obtained previously are further analysed. Method complexity is equal to $O(8\mathscr{P})$ without considering the Gaussian filtering.

3. Histogram-based methods

The previous section was dedicated to pixel-based methods. It is also possible to compare two images based on global features instead of local features (pixels). Histogram is a global image feature widely used in image processing. The main advantage of histogram-based methods is their global aspect. So they are more robust to camera or object motion. The main drawback appears when we compare two different images having a similar histogram. It will often result in missing a shot change.

Different uses of the histogram can be distinguished. Some methods only compute differences between histograms and then the quality of the result is linked to the kind of histogram considered. A first extension is the use of weighted differences between histograms. Another approach consists in the definition of an intersection operator between histograms or the definition of different distances or similarity measures.

3.1. Histogram difference

Tonomura and Abe [25] proposed a method based on gray-level histograms. Images are compared by computing a distance (of complexity $O(3\mathscr{N})$) between their histograms, as shown in the following equation:

$$\text{Detection if : } \left(\sum_{v=0}^{V} |H(I_t, v) - H(I_{t-1}, v)| \right) > T. \tag{17}$$

Nagasaka and Tanaka [20] propose a similar method using only 64 bins for colour histograms (2 bits for each colour component of the RGB space). Using the notation $H_{64}(I_t, v)$ the detection is defined by

$$\text{Detection if : } \left(\sum_{v=0}^{63} |H_{64}(I_t, v) - H_{64}(I_{t-1}, v)| \right) > T. \tag{18}$$

Gargi and Kasturi [12] apply histogram difference to other colour spaces (HSV, YIQ, L*a*b*, L*u*v*, and Munsell). More precisely, only non-intensity components are used (i.e. Hue and Saturation for HSV, I and Q for YIQ, a* and b* for L*a*b*, u* and v* for L*u*v*, and Hue and Chroma for the Munsell space). Shot change detection is then defined by

Detection if :

$$\left(\sum_{k=1}^{2}\sum_{v=0}^{V}|H(I_t,C_k,v)-H(I_{t-1},C_k,v)|\right)>T. \quad (19)$$

As it uses two colour components instead of only one, complexity is twice higher (i.e. equal to $O(6\mathcal{N})$).

Pye et al. [26] compute three histogram differences, considering separately the three colour components of the RGB space. The highest value is compared to a threshold for a shot change detection of complexity $O(9\mathcal{N})$:

Detection if :

$$\left(\max_{k\in\{R,G,B\}}\sum_{v=0}^{V}|H(I_t,C_k,v)-H(I_{t-1},C_k,v)|\right)>T. \quad (20)$$

Ahmed et al. [27] present several shot change detection algorithms using colour histograms. The first algorithm compares two frames using histograms computed on the Hue component C_H. So the detection needs $O(4\mathcal{N})$ operations and can be represented by

Detection if :

$$\left(\frac{\sum_{v=0}^{V}|H(I_t,C_H,v)-H(I_{t-\Delta},C_H,v)|}{\sum_{v=0}^{V}H(I_{t-\Delta},C_H,v)}\right)>T, \quad (21)$$

where Δ is the temporal skip between two frames.

The second algorithm by Ahmed et al. is based on reduced RGB space histograms. As in [20], histograms are composed of only 64 bins, using 2 bits for each colour component. The detection is done through a computation similar to the previously mentioned method:

Detection if :

$$\left(\frac{\sum_{v=0}^{63}|H_{64}(I_t,v)-H_{64}(I_{t-\Delta},v)|}{\sum_{v=0}^{63}H_{64}(I_{t-\Delta},v)}\right)>T, \quad (22)$$

resulting in a similar complexity.

O'Toole et al. [28] detect shot changes using a cosine similarity measure computed between two histograms. First three 64-bin histograms representing, respectively, the Y, U, and V components are obtained from each frame. Next the three histograms are concatenated into a single one in order to get only one 192-bin histogram per frame. Then two successive frames are compared based on their histogram using a cosine similarity measure to perform shot change detection:

Detection if :

$$\left(\frac{\sum_{v=0}^{V}(H_{YUV}(I_t,v)\ H_{YUV}(I_{t-1},v))}{\sum_{v=0}^{V}H_{YUV}(I_t,v)^2\sum_{v=0}^{V}H_{YUV}(I_{t-1},v)^2}\right)>T. \quad (23)$$

The method complexity is $O(4\mathcal{N})$. A similar work has been done by Cabedo and Bhattacharjee [29].

Chiu et al. [30] rely their video segmentation on a genetic algorithm using colour histogram differences of complexity $O(9\mathcal{N})$. Possible shot boundaries are evaluated with similarity adjacency functions. In order to limit the optimization cost of these functions, a genetic algorithm is used instead of traditional methods. A video sequence is encoded as a string of binary values, 1 and 0 representing, respectively, the presence or not of a shot boundary in the current frame. The fitness function used in the algorithm is defined as a similarity adjacency function based on colour histogram differences. Finally, crossover and mutation processes are derived from classical genetic algorithms in order to be adapted to video segmentation task.

Zhang et al. [21] propose a method called twin comparison. Successive frames are compared using a histogram difference metric of complexity $O(3\mathcal{N})$. The difference values obtained are compared with two thresholds. Cuts are detected when the difference is higher than a high threshold T_H. Possible starts of gradual transition are detected when difference is higher than a low threshold T_L. In this case, an accumulated difference is computed until the current difference is below T_L. Finally, the accumulated difference is compared to the high threshold T_H for shot change detection. The two thresholds can be automatically set using standard deviation and mean of the interframe differences in the whole video sequence.

Li and Lu [31] use also a two step method, detecting successively the location of the end of the transition and its start. Frames are compared using the colour ratio histogram metric [32]. First, two frames I_{t_1} and I_{t_2} (with $t_2=t_1+\Delta$) are compared using this metric. While the difference is below a given threshold T, t_2 is set to t_2+1. When the difference is above T, the transition end has been obtained. In order to determine the transition start, t_1 is set to t_2-1. The difference between frames I_{t_1} and I_{t_2} is then computed and compared to the threshold T. While the difference is below T, t_1 is set to t_1-1. When the difference is above T, the transition start is also obtained.

Several other statistical measures have been reviewed in [22]. The quadratic histogram difference can be computed between histograms from two successive frames, whereas the Kolmogorov–Smirnov statistic is computed between cumulative histograms from two successive frames. These two measures are detailed below, using the notation $H_C\ (I_t,v)$ to represent the cumulative histogram up to bin v for the frame I_t.

Detection if :

$$\left(\sum_{v=0}^{V}\frac{(H(I_t,v)-H(I_{t-1},v))^2}{(H(I_t,v)+H(I_{t-1},v))^2}\right)>T, \quad (24)$$

Detection if :

$$\left(\max_{v\in[0,V]}\left(|H_C(I_t,v)-H_C(I_{t-1},v)|\right)\right)>T. \quad (25)$$

The two methods are characterized by a complexity, respectively, equal to $O(6\mathcal{N})$ and $O(3\mathcal{N})$.

3.2. Weighted difference

In colour images, some colour components may have a bigger influence than others. So it is possible to detect shot changes by weighting the histograms of each colour component depending on their importance [10]:

Detection if :

$$\left(\sum_{k=1}^{3}\sum_{v=0}^{V}\frac{L(I_t,C_k)}{L_{mean}(I_t)}|H(I_t,C_k,v)-H(I_{t-1},C_k,v)|\right)>T, \quad (26)$$

where $L(I_t,C_k)$ and $L_{mean}(I_t)$ are, respectively, the luminance for the kth colour component of the frame I_t and the average luminance of the frame I_t considering all the colour components. The method complexity is equal to $O(3\mathcal{P}+9\mathcal{N})$.

Zhao et al. [33] use a learning procedure to determine the best weight values for weighted histogram difference computation. They first compute the original histogram difference with complexity $O(3\mathcal{N})$ defined by Eq. (17). Then a learning step formulated as a minmax optimization problem is performed in order to select the best weights to use in weighted histogram differences. The detection process relies finally on the following equation which requires 12 operations per histogram bin:

Detection if :

$$\left(\sum_{k=1}^{3}\sum_{v=0}^{V}w(k,v)|H(I_t,C_k,v)-H(I_{t-1},C_k,v)|\right)>T, \quad (27)$$

where $w(k,v)$ represents the best weight selected after the learning step. The overall complexity is then $O(15\mathcal{N})$.

Gargi and Kasturi [12] presented a method based on the difference of average colours of a histogram, which can as well be considered as a weighted histogram difference. The shot change detection can then be represented by

Detection if :

$$\left(\sum_{k=1}^{3}\left(\sum_{v=0}^{V}H(I_t,C_k,v)v-\sum_{v=0}^{V}H(I_{t-1},C_k,v)v\right)^2\right)>T \quad (28)$$

and requires $O(6\mathcal{N})$ operations.

Another method by Gargi and Kasturi using colour histograms has also been described in [12]. More precisely, it uses a reference colour table as a frame difference measure. Reference colour table can be seen as a coarse quantization of RGB colour space into 27 different colour triples which are used as bins for a three-dimensional (3D) colour histogram H_{ref}. The shot change detection needs five operations per bin and can be represented by

Detection if :

$$\left(\sum_{v=0}^{V}w(v,t)\sqrt{(H_{ref}(I_t,v)-H_{ref}(I_{t-1},v))^2}\right)>T, \quad (29)$$

where the weight $w(v,t)$ is defined as

$$w(v,t)=\begin{cases}H_{ref}(I_{t-1},v) & \text{if } (H_{ref}(I_t,v)>0)\wedge(H_{ref}(I_{t-1},v)>0)\\ 1 & \text{otherwise}\end{cases} \quad (30)$$

and requires three operations per bin. The overall complexity is then equal to $O(8\mathcal{N})$.

3.3. Histogram intersection

Similarity between two images can also be evaluated thanks to histogram intersection. Histogram intersection is computed using different operators, for example a min function. In this case the computational cost is $O(2\mathcal{N})$. Similarity ratio belonging to interval $[0,1]$ is then compared with a given threshold. This comparison allows the detection of shot changes:

Detection if :

$$\left(1-\frac{1}{XY}\sum_{v=0}^{V}\min(H(I_t,v),H(I_{t-1},v))\right)>T, \quad (31)$$

where XY represents the number of pixels in frames processed.

Another version of the histogram intersection-based shot change detection method is defined in [12] using the following equation:

Detection if :

$$\left(1-\frac{1}{XY}\sum_{v=0}^{V}\frac{\min(H(I_t,v),H(I_{t-1},v))}{\max(H(I_t,v),H(I_{t-1},v))}\right)>T \quad (32)$$

with a complexity equal to $O(4\mathcal{N})$.

Haering et al. [34] apply histogram intersection defined in Eq. (31) to HSV (Hue, Saturation, Value) colour space, using 16 bins for Hue component and 4 bins each for Saturation and Value components.

An extension of [34] has been proposed by Javed et al. [35]. Hue is represented using only 8 bins. Instead of thresholding the histogram intersection of two successive frames, they compute the difference between two successive histogram intersection values and compare this derivative to a threshold.

3.4. Use of χ^2 test

Nagasaka and Tanaka have also proposed [20] a 64-bin histogram comparison based on χ^2 test. The shot change detection is then defined by

Detection if :

$$\left(\sum_{v=0}^{63}\frac{(H_{64}(I_t,v)-H_{64}(I_{t-1},v))^2}{H_{64}(I_t,v)}\right)>T \tag{33}$$

with the assumption $H_{64}(I_t,v)\neq 0$. If this assumption does not hold, we use the following equation instead:

Detection if :

$$\left(\sum_{v=0}^{63}\frac{(H_{64}(I_t,v)-H_{64}(I_{t-1},v))^2}{H_{64}(I_{t-1},v)}\right)>T \tag{34}$$

with the assumptions $H_{64}(I_{t-1},v)\neq 0$ and $H_{64}(I_t,v)=0$. This method is considered as more efficient than simple histogram comparison-based methods. Its complexity is equal to $O(5\mathcal{N})$.

A modification has been proposed by Dailianas et al. [10] where the detection is represented by

Detection if :

$$\left(\sum_{v=0}^{V}\frac{(H(I_t,v)-H(I_{t-1},v))^2}{\max(H(I_t,v),H(I_{t-1},v))}\right)>T \tag{35}$$

with a similar complexity.

Gunsel et al. [36] perform a K-means clustering algorithm to determine the location of shot boundaries. Successive frames are compared using χ^2 test or histogram difference on YUV histograms, resulting in a complexity equal to $O(15\mathcal{N})$. Every interframe difference value is classified into shot change or non-shot change.

3.5. Similarity measures between normalized histograms

Several measures computed on normalized histograms have been reviewed by Ren et al. [22] and by Kim and Park in [37]. Using the notation $H_N(I_t,v)$ to represent the probability of intensity v in the frame I_t, cross entropy, divergence, Kullback Liebler distance, and Bhattacharya distance are, respectively, defined as

Detection if :

$$\left(\sum_{v=0}^{V}\left(H_N(I_t,v)\log\frac{H_N(I_t,v)}{H_N(I_{t-1},v)}\right)\right)>T, \tag{36}$$

Detection if :

$$\left(\begin{array}{l}\displaystyle\sum_{v=0}^{V}\left(H_N(I_t,v)\log\frac{H_N(I_t,v)}{H_N(I_{t-1},v)}\right)\\ \displaystyle+\sum_{v=0}^{V}\left(H_N(I_{t-1},v)\log\frac{H_N(I_{t-1},v)}{H_N(I_t,v)}\right)\end{array}\right)>T, \tag{37}$$

Detection if :

$$\left(\begin{array}{l}\displaystyle\sum_{v=0}^{V}\left(H_N(I_t,v)\log\frac{H_N(I_t,v)}{H_N(I_{t-1},v)}\right)\\ \displaystyle+\sum_{v=0}^{V}\left((1-H_N(I_t,v))\log\frac{1-H_N(I_t,v)}{1-H_N(I_{t-1},v)}\right)\end{array}\right)>T, \tag{38}$$

Detection if :

$$\left(-\log\left(\sum_{v=0}^{V}\sqrt{H_N(I_t,v)H_N(I_{t-1},v)}\right)\right)>T \tag{39}$$

with complexities, respectively, equal to $O(4\mathcal{N})$, $O(8\mathcal{N})$, $O(11\mathcal{N})$, and $O(3\mathcal{N})$.

All these methods are based on a uniform process all over the image. The heterogeneity present within a frame led to use block-based methods.

4. Block-based methods

Block sampling of the video frames can be performed in order to increase the quality of shot change detection but also to decrease the computation time. Once block representation has been obtained from original images, it is possible to perform some algorithms derived from pixel or histogram-based methods presented previously. Use of blocks allows a processing which is intermediate, between local level like pixel-based methods and global level as histogram-based methods. The main advantage of block-based methods is their relative insensitivity to noise and camera or object motion. We have distinguished between several approaches all working on blocks.

4.1. Block similarity

Kasturi and Jain [38] perform a similarity test on block-sampled images. Like in pixel-based methods, pairs of blocks (with same spatial coordinates) from two successive frames are compared. The similarity is based on block features like mean and variance, which can be computed on a complete frame considering a complexity of $O(\mathcal{P}+\mathcal{B})$ and $O(2\mathcal{P}+3\mathcal{B})$. The likelihood rate L (of complexity $O(9\mathcal{B})$) is defined for a block b as

$$L(I_t,I_{t-1},b)=\frac{((\sigma_{t,b}^2+\sigma_{t-1,b}^2)/2+((\mu_{t,b}-\mu_{t-1,b})/2)^2)^2}{\sigma_{t,b}^2\sigma_{t-1,b}^2}, \tag{40}$$

where $\mu_{t,b}$ and $\sigma_{t,b}^2$ are, respectively, the mean and the variance of block b pixel values in image I_t. Then thresholded values L_D of L are defined by the

equation

$$L_D(I_t, I_{t-1}, b) = \begin{cases} 1 & \text{if } L(I_t, I_{t-1}, b) > T_D, \\ 0 & \text{in other cases,} \end{cases} \quad (41)$$

where T_D is considered as a tolerance value. A detection is obtained when

$$\text{Detection if : } \sum_{b=1}^{B} c_b L_D(I_t, I_{t-1}, b) > T, \quad (42)$$

where c_b is used to give more or less importance to block b. Most of the time c_b is set to 1 for all the blocks. Overall complexity is estimated to $O(3\mathscr{P} + 15\mathscr{B})$ considering required operations for estimation of the block mean, variance, likelihood rate, and thresholding, and the final cost of the detection. This likelihood ratio can also be used directly on full-frames, as proposed in [22].

Another well-known measure involving variance is the Yakimovsky likelihood ratio which can be applied also on blocks or frames directly [22]. For each block this ratio is computed as

$$L'(I_t, I_{t-1}, b) = \frac{(\sigma^2_{\{t,t-1\},b})^2}{\sigma^2_{t-1,b}\sigma^2_{t,b}}, \quad (43)$$

where $\sigma^2_{t,b}$ and $\sigma^2_{t-1,b}$ represent the variances of the pixel intensity values in the frames I_t and I_{t-1} considering a block b. The notation $\sigma^2_{\{t,t-1\},b}$ is used to denote the variance of the pooled data from both frames for a block b. Knowing $\sigma^2_{t,b}$ and $\sigma^2_{t-1,b}$ computation of $\sigma^2_{\{t,t-1\},b}$ needs four operations per block. When $\mu_{t,b}$ is not available, computation of $\sigma_{t,b}$ is characterized by a cost of $O(3\mathscr{P} + 4\mathscr{B})$. The overall complexity $O(3\mathscr{P} + 13\mathscr{B})$ is obtained by adding the cost of Eq. (43) (i.e. three operations per block).

The Freund statistic can also be used to detect shot changes. Distance measure is then defined by

$$L''(I_t, I_{t-1}, b) = \frac{\mu_{t,b} - \mu_{t-1,b}}{\sqrt{(\sigma^2_{t,b} + \sigma^2_{t-1,b})/XY}} \quad (44)$$

resulting in a complexity of $O(3\mathscr{P} + 11\mathscr{B})$ obtained by replacing cost of Eq. (40) by cost of Freund statistic, i.e. $O(5\mathscr{B})$.

Lee et al. [39] perform shot change detection using block differences computed in the HSV colour space. First RGB images are converted to HSV in order to avoid camera flashes. Then the mean values of Hue and Saturation components are computed for each block (with a cost of $O(\mathscr{P} + \mathscr{B})$ for each colour component). Two successive blocks are compared using these mean values:

$$D_H(I_{t_1}, I_{t_2}, b) = |\mu(I_{t_1}, b, C_H) - \mu(I_{t_2}, b, C_H)|, \quad (45)$$

$$D_S(I_{t_1}, I_{t_2}, b) = |\mu(I_{t_1}, b, C_S) - \mu(I_{t_2}, b, C_S)|, \quad (46)$$

where $\mu(I_t, b, C_k)$ is the mean of a block b in the frame I_t considering the colour component C_k. D_H and D_S represent, respectively, differences for Hue and Saturation colour component. These two distances need two operations per block each and are used to determine if each block has changed:

$$D(I_{t_1}, I_{t_2}, b) = \begin{cases} 1 & \text{if } (D_H(I_{t_1}, I_{t_2}, b) > T_H) \vee (D_S(I_{t_1}, I_{t_2}, b) > T_S), \\ 0 & \text{otherwise,} \end{cases} \quad (47)$$

which requires three operations per block. Finally, the ratio between the number of changed blocks and the total number of blocks is compared to another threshold in order to detect shot changes:

$$\text{Detection if : } \frac{1}{B}\sum_{b=1}^{B} D(I_t, I_{t-\Delta}, b) > T, \quad (48)$$

where Δ represents the temporal skip used in the shot change detection process. The overall complexity of this method is $O(2\mathscr{P} + 10\mathscr{B})$.

4.2. Histogram comparison

Swanberg et al. [40] present a method detecting shot changes thanks to the comparison of colour histograms computed on the blocks of the images noted $H(I_t, b, C_k, v)$. The detection process is then defined as:

$$\text{Detection if : } \left(\sum_{k=1}^{3}\sum_{b=1}^{B}\sum_{v=0}^{V} \frac{(H(I_t, b, C_k, v) - H(I_{t-1}, b, C_k, v))^2}{H(I_t, b, C_k, v) + H(I_{t-1}, b, C_k, v)}\right) > T \quad (49)$$

and has a computational cost of $O(15\mathscr{N}B + 3\mathscr{B})$.

Nagasaka and Tanaka [20] extend their histogram comparison to images divided in 4×4 blocks. Every pair of blocks from two successive frames is compared using the χ^2 test on 64-bin histograms:

$$\chi(I_t, b) = \left(\sum_{v=0}^{63} \frac{(H_{64}(I_t, b, v) - H_{64}(I_{t-1}, b, v))^2}{H_{64}(I_t, b, v)}\right) \quad (50)$$

which requires for each block four operations per histogram bin. The values obtained are then sorted in an ascending way and the eight lowest are kept. The sum of these values is computed and compared to a threshold to detect shot changes:

$$\text{Detection if : } \left(\sum_{b=1}^{8} \chi_s(I_t, b)\right) > T \quad (51)$$

where the χ_s values represent ascending sorted values of χ (i.e. for $b \in [1, 16]$ we have $\chi_s(I_t, 1) \leqslant \chi_s(I_t, b) \leqslant \chi_s(I_t, 16)$). Considering Eqs. (50) and (51), the overall complexity of the method is $O(4\mathscr{N}B + 0.5\mathscr{B})$.

Ueda et al. [41] proposed to use the rate of correlation change instead of the magnitude of correlation change

proposed in [20]. Each value $\chi(I_t,b)$ obtained from a pair of blocks is compared to a threshold. The detection depends on the number of significant values $\chi(I_t,b)$ instead of the sum of the highest $\chi(I_t,b)$, resulting in a complexity equal to $O(4\mathcal{N}B+2\mathcal{B})$.

Ahmed et al. [27] propose also a block-based version of their method using the six most significant RGB bits as described in Eq. (22). They compare histograms computed on blocks instead of global histograms. The sum of the histogram differences obtained for each block is computed and compared to a predefined threshold in order to detect shot changes, as shown in

Detection if :

$$\left(\sum_{b=1}^{B}\frac{\sum_{v=0}^{63}|H_{64}(I_t,b,v)-H_{64}(I_{t-\Delta},b,v)|}{\sum_{v=0}^{63}H_{64}(I_{t-\Delta},b,v)}\right)>T \quad (52)$$

where Δ represents the temporal skip between two successive frames to be analysed. The computation cost is also equal to $O(4\mathcal{N}B+2\mathcal{B})$.

Ahmed and Karmouch proposed [42] an improved version of their algorithm described previously. Instead of comparing two frames considering a fixed temporal skip, the method is based on an adaptive temporal skip. First, two images I_{t_1} and I_{t_2} are compared according to Eq. (52). Then if the difference is greater than a threshold, t_2 is replaced by $(t_1+t_2)/2$ and the frames are again compared. If the difference is still greater than the threshold, t_1 is also set to $(t_1+t_2)/2$ (considering the current values of t_1 and t_2) and frames are compared. This process is repeated until $t_1+1=t_2$ which represents a shot change between frames t_1 and t_2.

Lee and Ip [43] introduce a selective HSV histogram comparison algorithm. First, pixels are classified with respect to their colour level. If a pixel is characterized by high values for V and S, it is classified into a discrete colour using H component. Otherwise the classification is based on the intensity (or gray-level) value. For a given pixel $P(I_t,b,i,j)$, two complementary states are defined:

$$S_{hue}(I_t,b,i,j)=\begin{cases}1 & \text{if } \begin{pmatrix}(P(I_t,b,C_S,i,j)>T_S)\\ \wedge\ (P(I_t,b,C_V,i,j)>T_V)\end{pmatrix},\\ 0 & \text{otherwise,}\end{cases} \quad (53)$$

$$S_{gray}(I_t,b,i,j)=1-S_{hue}(I_t,b,i,j). \quad (54)$$

Computation of these states requires four operations per pixel. For each block two selective histograms $H_{hue}(I_t,b)$ and $H_{gray}(I_t,b)$ are then computed in a classical way considering the two states previously defined. The notation $H_{hue}(I_t,b,v)$ (respectively, $H_{gray}(I_t,b,v)$) represents the number of pixels in block b of frame I_t with state S_{hue} (resp. S_{gray}) equal to 1 and with hue (resp. intensity or gray-level) value equal to v. These histograms are used for shot change detection:

Detection if :

$$\left(\sum_{b=1}^{B}\left(\begin{array}{l}\sum_{v=0}^{V}|H_{hue}(I_t,b,v)-H_{hue}(I_{t-1},b,v)|\\ +\sum_{v=0}^{V}|H_{gray}(I_t,b,v)-H_{gray}(I_{t-1},b,v)|\end{array}\right)\right)>T \quad (55)$$

resulting in a complexity equal to $O(6\mathcal{N}B+2\mathcal{B}+4\mathcal{P})$.

Bertini et al. [44] compare in the HSI colour space histograms of successive frames divided into nine blocks. In order to improve robustness to change in lighting conditions, the intensity component is not used. The detection can then be represented by the following equations:

$$D_{HS}(I_t,I_{t+1},b)=\sum_{k\in\{H,S\}}\sum_{v=0}^{V}(H(I_t,b,C_k,v)-H(I_{t+1},b,C_k,v)), \quad (56)$$

$$D'_{HS}(I_t)=\sum_{b=1}^{B}D_{HS}(I_t,I_{t+1},b)-\sum_{b=1}^{B}D_{HS}(I_{t-1},I_t,b), \quad (57)$$

Detection if :

$$\begin{pmatrix}((D'_{HS}(I_t)>0)\wedge(D'_{HS}(I_{t+1})<0))\\ \vee\ ((D'_{HS}(I_t)<0)\wedge(D'_{HS}(I_{t+1})>0))\end{pmatrix}. \quad (58)$$

The two distances $D_{HS}(I_t,I_{t+1},b)$ and $D'_{HS}(I_t)$ require respectively, two operations per bin per block per colour and one operation per block. The overall complexity is then $O(4\mathcal{N}B+\mathcal{B})$. In order to improve the detection, a minimum temporal distance is defined between two successive cuts.

Chahir and Chen [45] based their method on histogram intersection computed on frames divided into 24 blocks. The colour space used in their method is L*u*v*. For each block, color histogram intersection is computed between two successive frames requiring 12 operations per bin. A comparison with a threshold allows to determine whether the block has been changed or not. The number of changed blocks is then compared to another threshold in order to detect a shot change, resulting in an overall complexity equal to $O(12\mathcal{N}B+2\mathcal{B})$.

4.3. Combination of histogram differences and likelihood rate

This method proposed by Dugad et al. [46] is based on two successive steps to detect cuts and other transitions. Shot changes are detected using successively histogram difference and likelihood ratio. In this method, three

thresholds have to be set. Histogram difference step (whose complexity is $O(3\mathcal{N})$) is defined as in Eq. (17) and is compared with two thresholds. The difference is first compared to a high threshold in order to avoid false alarms. If it is lower than this threshold, it is compared to a low threshold. If it is higher than this low threshold, the final decision is taken by computing likelihood ratio values. In this case, the two frames to be compared are divided in 64 blocks and the 16 central blocks are kept. For each block, the likelihood ratio is computed between the block $P(I_t, b)$ and the blocks $P(I_{t-1}, b')$ where b' belongs to the neighbourhood of b, and the minimum of the likelihood value is kept. Then a mean of the 16 minimum likelihood ratios is computed and is compared with the third threshold, which may result in a shot change detection. The respective costs for computation of the mean and variance of each block and estimation of the likelihood ratio in a 3×3 neighbourhood are $O(3\mathcal{P} + 4\mathcal{B})$ and $O(72\mathcal{B})$. Computation of the minimum values for each block and the mean value on all block requires, respectively, seven and one operations per block. As we are using only a quarter of the blocks, we divide the number of operations per block by 4. The overall complexity is then $O(3\mathcal{P} + 3\mathcal{N} + 21\mathcal{B})$.

4.4. Use of neighbourhood colour ratio

Adjeroh et al. [47] compare two successive frames using neighbourhood colour ratios. A local averaging step (with a cost of $O(5\mathcal{P} + \mathcal{B})$) is first performed in order to obtain one value $P'(I_t, b)$ per block:

$$P'(I_t, b) = \prod_{i=2}^{X-1} \prod_{j=2}^{Y-1} \frac{1}{4P(I_t, b, i, j)} \times \begin{pmatrix} P(I_t, b, i-1, j) + P(I_t, b, i+1, j) \\ +P(I_t, b, i, j-1) + P(I_t, b, i, j+1) \end{pmatrix}. \quad (59)$$

Pairs of blocks from two different frames are then compared using this measure:

$$D'(I_t, I_{t-\Delta}, b) = 1 - \min\left(\frac{P'(I_t, b)}{P'(I_{t-\Delta}, b)}, \frac{P'(I_{t-\Delta}, b)}{P'(I_t, b)}\right) \quad (60)$$

which requires four operations per block and where Δ represents the temporal skip. Shot changes are finally detected if the number of significant D' values for all selected blocks is higher than a fixed threshold, or if the average value of D' is higher than another threshold. These two conditions need, respectively, two and one operations per block to be verified. The overall complexity of this method is then $O(5\mathcal{N}B + 8\mathcal{B})$.

4.5. Evolution of block dissimilarity

Shot changes can also be detected by analysing the evolution of block dissimilarity. Demarty and Beucher [48] compute locally a distance criterion (of cost $O(9\mathcal{P})$) in RGB colour space between blocks of two successive images. Result obtained consists in distance values between the two images for every block. Then the sum of these values is computed (which requires three operations per block) and an evolution curve of this sum is built. This evolution curve is filtered using a top-hat morphological operation and is finally compared with a threshold in order to detect shot changes. The complexity is equal to $O(9\mathcal{P} + 3\mathcal{B})$.

Lefèvre et al. [49] proposed a method using HSV colour space on block-sampled images in order to avoid false detection due to illumination effects. A value is obtained for each block from Hue and Saturation components with a cost of $O(2\mathcal{P} + 5\mathcal{B})$. Then a block-based difference (requiring three operations per block) is computed between two frames based on the block values. This difference is tracked through time as well as its derivative. Analysis of this derivative allows cut detection, whereas the initial (non-derivated) difference values are used to initialize a cumulative sum computation of the derivated values. This allows detection of gradual transitions. This method is characterized by a computational cost of $O(2\mathcal{P} + 8\mathcal{B})$.

4.6. Temporal and spatial subsampling

Xiong and Lee [50] propose to subsample the video sequence in both space and time. An abrupt change is detected between two frames I_t and $I_{t+\Delta}$ if

$$\text{Detection if : } \left(\sum_{b \in B'} D_\mu(I_t, I_{t+\Delta}, b)\right) > T, \quad (61)$$

where B' represents a set of *a priori* selected blocks and $D_\mu(I_{t_1}, I_{t_2}, b)$ is defined as

$$D_\mu(I_{t_1}, I_{t_2}, b) = \begin{cases} 1 & \text{if } |\mu_{t_1,b} - \mu_{t_2,b}| > T_\mu, \\ 0 & \text{in other cases.} \end{cases} \quad (62)$$

The two equations need, respectively, one and three operations per block. As the block mean computation is linked with a cost of $O(\mathcal{P} + \mathcal{B})$, the overall complexity is equal to $O(\mathcal{P} + 5\mathcal{B})$. Gradual transitions are detected using an edge-based frame-to-frame difference measure. If a shot change is detected, a binary search is performed reducing Δ to determine the exact shot boundaries. The method proposed is called "Net Comparison" and has also been tested in HSV colour space.

5. Feature-based methods

All the methods we have already presented were using features, but they can be qualified of trivial features. Here we are considering more sophisticated ones. We

consider:

- the moments computed on the image,
- the contour lines extracted from the image,
- some feature points extracted using Hough Transform,
- the planar points,
- colour transition,
- modeling of the video transition effects,
- the use of some decision process as Bayesian methods,
- features computed from classical statistical approaches,
- and the use of Hidden Markov Models.

5.1. Moment invariants

Arman et al. [51] use moment invariants combined with histogram intersection to detect shot changes. Moment invariants have properties such as invariance to scale change, rotation, and translation. The moments of a frame I_t are defined as

$$m_{p,q} = \sum_{i=1}^{X} \sum_{j=1}^{Y} i^p j^q (I_t, i, j). \tag{63}$$

In [10], shot changes are detected thanks to the computation of the usual Euclidean distance between two frames using a vector composed of the first three moment invariants, defined as

$$\vec{\Phi} = \begin{pmatrix} m_{2,0} + m_{0,2} \\ (m_{2,0} - m_{0,2})^2 + 4m_{1,1}^2 \\ (m_{3,0} - 3m_{1,2})^2 + (3m_{2,1} - m_{0,3})^2 \end{pmatrix}. \tag{64}$$

Considering Eq. (63), all moments used in Eq. (64) require three operations per pixel, excepting $m_{1,2}$ and $m_{2,1}$ which need four operations per pixel. The detection can be finally defined as:

$$\text{Detection if : } (\|\overrightarrow{\Phi(I_t)} - \overrightarrow{\Phi(I_{t-1})}\|^2) > T \tag{65}$$

resulting in an overall complexity of $O(23\mathscr{P})$.

5.2. Edges

Zabih et al. [52] use edge extraction to detect shot changes. Global motion compensation is performed on successive frames. Next, edges are extracted using Canny algorithm and dilated. Normalized proportions of entering edges and exiting edges are then computed (with a cost of $O(3\mathscr{P})$ each) using the following equations:

$$P(C_{out}, I_t) = 1 - \frac{\sum_{i=1}^{X} \sum_{j=1}^{Y} E(I_{t-1}, i + \alpha_{t-1,t}, j + \beta_{t-1,t}) E_d(I_t, i, j)}{\sum_{i=1}^{X} \sum_{j=1}^{Y} E(I_{t-1,}, i, j)}, \tag{66}$$

$$P(C_{in}, I_t) = 1 - \frac{\sum_{i=1}^{X} \sum_{j=1}^{Y} E_d(I_{t-1}, i + \alpha_{t-1,t}, j + \beta_{t-1,t}) E(I_t, i, j)}{\sum_{i=1}^{X} \sum_{j=1}^{Y} E(I_{t-1,}, i + \alpha_{t-1,t}, j + \beta_{t-1,t})}, \tag{67}$$

where E and E_d are, respectively, the contour image and its dilated version, and $(\alpha_{t-1,t}, \beta_{t-1,t})$ represents the global motion translation vector between the two successive images I_t and I_{t-1}. Then a dissimilarity measure $ECF(I_t)$ called edge change fraction is computed by

$$ECF(I_t) = \max(P(C_{out}, I_t), P(C_{in}, I_t)). \tag{68}$$

Finally this value is compared to a threshold to detect shot changes:

$$\text{Detection if : } ECF(I_t) > T \tag{69}$$

resulting in a complexity at least equal to $O(26\mathscr{P})$ when considering edge detection requires 20 operations per pixel.

Smeaton et al. [53] proposed an evolution of the previous method where the detection is based on the evolution of the edge change fraction on several frames instead of the analysis of this dissimilarity measure on only one frame. Detection can then be defined by:

$$\text{Detection if : } (ECF(I_t) - ECF(I_{t-1})) > T. \tag{70}$$

Lienhart [9] also uses edge information to perform dissolve detection. First, edges extracted with the Canny edge detector are confronted with two thresholds to determine weak and strong edges:

$$E_w(I_t, i, j) = \begin{cases} E(I_t, i, j) & \text{if } T_w \leqslant E(I_t, i, j) \leqslant T_s, \\ 0 & \text{in other cases,} \end{cases} \tag{71}$$

$$E_s(I_t, i, j) = \begin{cases} E(I_t, i, j) & \text{if } T_s \leqslant E(I_t, i, j), \\ 0 & \text{in other cases,} \end{cases} \tag{72}$$

where T_w and T_s are, respectively, the lowest and highest thresholds for detecting weak and strong edges. E_w and E_s represent the weak and strong edge images and need, respectively, two and one operations per pixel to be computed. Then the edge-based contrast EC is obtained for a frame I_t according to the equation

$$EC(I_t) = 1 + \frac{\sum_{i=1}^{X} \sum_{j=1}^{Y} E_s(I_t, i, j) - \sum_{i=1}^{X} \sum_{j=1}^{Y} E_w(I_t, i, j) - 1}{\sum_{i=1}^{X} \sum_{j=1}^{Y} E_s(I_t, i, j) + \sum_{i=1}^{X} \sum_{j=1}^{Y} E_w(I_t, i, j) + 1} \tag{73}$$

with a cost of $O(2\mathscr{P})$. Finally dissolves are detected when the current value of EC is a local minimum. The overall complexity $O(25\mathscr{P})$ is obtained by adding edge detection cost.

Yu et al. [54] use edge information to detect gradual transitions. Cuts are first detected using a histogram

difference measure (of cost $O(3\mathscr{P})$) computed between two successive subsampled frames. Then a second pass is necessary for detecting gradual transitions. For every frame I_t between two successive cuts at time t_1 and t_2, the number $Q_E(I_t)$ of edge points present in the image is computed (considering 21 operations per pixel) and temporally smoothed. Then for every local minima $Q_E(I_{t'})$ which is below a predefined threshold, a search of the two closest local maxima $Q_E(I_{t'_1})$ and $Q_E(I_{t'_2})$ is performed with $t'_1 < t' < t'_2$. A fade-out effect is detected between t'_1 and t' if

$$\text{Detection if : } \left(\frac{1}{t' - t'_1} \sum_{t=t'_1}^{t'} |Q_E(I_t) - Q_E(I_{t-1})| \right) \in \left[T_{low}^{fade\ \ out}, T_{high}^{fade\ \ out} \right]. \quad (74)$$

Similarly, a fade-in effect is detected between t' and t'_2 if

$$\text{Detection if : } \left(\frac{1}{t'_2 - t'} \sum_{t=t'}^{t'_2} |Q_E(I_t) - Q_E(I_{t-1})| \right) \in [T_{low}^{fade\ \ in}, T_{high}^{fade\ \ in}]. \quad (75)$$

For dissolve effect detection, a new measure called double chromatic difference is computed for every frame belonging to the interval $[t'_1, t'_2]$:

$$DCD(I_t) = \sum_{i=1}^{X} \sum_{j=1}^{Y} \left| \frac{1}{2} P(I_{t'_1}, i, j) + \frac{1}{2} P(I_{t'_2}, i, j) - P(I_t, i, j) \right| \quad (76)$$

which requires six operations per pixel. If the frame number t_{min} corresponding to the global minimum value $DCD\,(I_{t_{min}})$ is close enough to t' (i.e. $|t' - t_{min}| < \varepsilon$ with ε defined as a small number), a dissolve effect is assumed to be found between frames $I_{t'_1}$ and $I_{t'_2}$. The overall complexity of this method is equal to $O(3\mathscr{N} + 27\mathscr{P})$.

Heng and Ngan [55] also propose a method based on edge information. They introduce the notion of edge object, considering the pixels close to the edge. Occurrences of every edge object are matched on two successive frames. Shot changes are detected using the ratio between the amount of edge objects persistent over time and the total amount of edge objects.

Nam and Tewfik [56] propose a coarse-to-fine shot change detection method based on wavelet transforms. Image sequences are first temporally subsampled. Frames processed are also spatially reduced using a spatial two-dimensional (2D) wavelet transform. Intensity evolution of pixels belonging to coarse frames is analysed using a temporal one-dimensional (1D) wavelet transform. Sharp edges define possible shot change locations. Video frames around these locations are further processed at full-rate. Temporal 1D wavelet transform is applied again on the full-rate video sequence. Edge detection is also performed on every coarse frame and the number of edge points is computed on a block-based basis. Difference between two successive frames is computed using the number of edge points for each block. True shot boundaries are located on sharp edges in the 1D wavelet transform and high values of interframe difference considering block-based amount of edge points. An extension to wipe transitions detection has been proposed in [57].

5.3. Feature points

Ardebilian et al. [18] detect shot changes by comparing feature points extracted from two successive images. They use Hough transform to extract feature points. Success or not of the feature points matching between two successive frames results directly in cut detection.

5.4. Planar points

Silva et al. [58] perform shot change detection using spatio-temporal representation of an image sequence. A video sequence $\mathscr{V}$ is represented in $\mathbb{R}^4$ as an hypersurface:

$$\mathscr{V} = \{i, j, t, P(I_t, i, j)\}. \quad (77)$$

The amount of planar points in every frame is considered as the measure for detecting cuts. We recall that planar points are points contained in a flat neighbourhood of the hypersurface. For a given frame I_t, planar points are determined using the characteristic polynomial coefficients of the Hessian matrix of $P(I_t, i, j)$. Then the percentage of planar points is computed. A cut is detected (in a four frame interval) when this value is greater than three times the temporal variance of the percentage of planar points. The overall complexity $O(51\mathscr{P})$ is obtained by adding the cost associated with the Hessian matrix ($O(27\mathscr{P})$), the three characteristic polynomial coefficients ($O(18\mathscr{P})$), the classification of a point as planar ($O(5\mathscr{P})$), and the final decision ($O(\mathscr{P})$).

5.5. Colour transitions

Sanchez et al. [59] compare between two successive frames using colour histograms computed on specific regions. These regions are defined from the most significant colour transitions of the image, considered as high values of its multispectral gradient and computed with Sobel approximation. Colour histograms are compared between regions of two successive frames to determine the coherence of the region through time. Shot changes are finally detected if the amount of changed regions is above a given threshold.

5.6. Transition modeling

Some shot changes are created from production effects. These transitions can be modeled explicitly with mathematical tools in order to be detected. Several methods using these assumptions are presented below.

Hampapur et al. [16] model several kinds of fades and wipes with mathematical functions. Knowing the two last shots and their respective durations, it is possible to estimate the duration of the current shot. Detection of shot changes can rely on a constancy measure (of cost $O(\mathscr{P})$) defined for frame I_t as

$$C(I_t) = \frac{\sum_{i=1}^{X}\sum_{j=1}^{Y} S_{binary}(I_t,i,j)}{\sigma_t(1+|X_c(I_t)-(X/2)|+|Y_c(I_t)-(Y/2)|)}, \quad (78)$$

where σ_t represents the standard deviation of pixel intensities in frame I_t computed with a cost of $O(3\mathscr{P})$. The binary state $S_{binary}(I_t,i,j)$ of a pixel is defined as

$$S_{binary}(I_t,i,j) = \begin{cases} 1 & \text{if } P(I_t,i,j)\neq 0, \\ 0 & \text{otherwise} \end{cases} \quad (79)$$

requiring one operation per pixel and the components of the centroid of image I_t are noted $X_c(I_t)$ and $Y_c(I_t)$. They are computed as

$$X_c(I_t) = \frac{\sum_{i=1}^{X}\sum_{j=1}^{Y} iP(I_t,i,j)}{\sum_{i=1}^{X}\sum_{j=1}^{Y} P(I_t,i,j)}, \quad (80)$$

$$Y_c(I_t) = \frac{\sum_{i=1}^{X}\sum_{j=1}^{Y} jP(I_t,i,j)}{\sum_{i=1}^{X}\sum_{j=1}^{Y} P(I_t,i,j)} \quad (81)$$

and need together five operations per pixel. The overall complexity is then $O(10\mathscr{P})$.

Adami and Leopardi [60] perform dissolve detection applying a model of dissolve effects on frame histograms. For every frame, two specific histograms are computed:

$$\tilde{H}(I_t) = H_{1/2}(I_t) * H_{1/2}(I_t), \quad (82)$$

$$H_{\Delta}(I_t) = H_{1/2}(I_{t-\Delta}) * H_{1/2}(I_{t+\Delta}), \quad (83)$$

where $H_{1/2}(I_t)$ represents the histogram of the frame I_t scaled by half, Δ is a fixed parameter, and the operator $*$ figures a convolution. Both operations are characterized by a cost of $O(\mathscr{N}^2)$. It is then possible to compute a dissimilarity measure using these histograms:

$$R(I_t) = \frac{\chi^2(H(I_t),\tilde{H}(I_t))}{\chi^2(H(I_t),H_{\Delta}(I_t))} - 1, \quad (84)$$

where the χ^2 operator (of complexity $O(5\mathscr{N})$) is computed between two histograms. Maxima values of $R(I_t)$ indicate dissolve locations:

$$\text{Detection if : } R(I_t) > R(I_{t'}) \quad \forall t' \text{ in the neighbourhood of } t. \quad (85)$$

The overall complexity is then $O(2\mathscr{N}^2 + 10\mathscr{N})$.

Aigrain and Joly [61] detect shot changes using the assumption of the absence of important motion in the different shots. Their method is based on a motion differential model and uses a density function estimation as the difference between two images. First, two successive images are reduced spatially and normalized using histogram equalization. Then the histogram of pixel-pair difference is computed and is simplified to two values, which are, respectively, the amount of differences belonging to the interval $[128,255]$ computed on normalized images and the amount of differences belonging to the interval $[1,40]$ computed on non-normalized images. Both values require four operations per pixel in order to be computed. Cut and gradual transition detections are, respectively, based on local maxima of the first and second value described previously. The method complexity is equal to $O(8\mathscr{P})$.

Lienhart [9] relies on fade modeling from [16] to perform fade detection. The proposed algorithm uses the standard deviation of pixel intensities as an estimation of the scaling function introduced in fade effects. First, all monochrome frames are located as they are fade start or end candidates. These frames are characterized by a standard deviation $\sigma(I_t)$ close to zero. Fades are then detected by searching in both temporal directions for a linear increase in the standard deviation.

Alattar [62] bases also the shot change detection on variance of pixel intensities. Fades are detected using a two-step scheme. First, local minimum values of the second-order difference of the pixel intensity spatial variance time series are obtained. Then a test is performed to determine whether the first-order difference of the pixel intensity mean is relatively constant in the neighbourhood of the local variance minimum or not. In the positive case, a fade is assumed to be found. A similar method [63] has been proposed for dissolve.

Truong et al. [64] combine approaches from [9,62]. First, all monochrome frames are detected. Then only monochrome frames which are next to a local minimum of the intensity variance second-order difference are kept. Finally, a test is performed on the first-order difference of the mean intensity. If this value is constant through time and other conditions are satisfied, a fade would be detected. The other conditions correspond to comparison between thresholds and the absolute value of the first-order difference and the intensity variance of the first or last frame. Dissolve detection is performed using the evolution of the variance first-order difference

through time. This difference value should be monotonically increasing from a negative value up to a positive value. So zero crossings are used to locate dissolve frames.

Fernando et al. [65] use also mean and variance of the luminance signal to determine fade and dissolve transitions locations. For every frame, the mean and the variance of the luminance is computed. The ratio between the second temporal derivative of the variance and the first temporal derivative of the mean is then compared between two successive frames. Shot changes are located when this ratio is constant through time.

All the approaches based on effect modeling using variance and mean of successive frames are characterized by a cost of $O(3\mathscr{P})$.

5.7. Bayesian approaches

Vasconcelos and Lippman [66] propose a segmentation method using a Bayesian model of the editing process. For each frame a local activity measure is computed based on a tangent distance. In order to detect shot changes, this measure is compared (following a Bayesian framework) to an adaptive threshold, depending on the a priori duration of a shot and on the time elapsed between the previous shot change and the current frame I_t.

Hanjalic and Zhang [67] use also a statistical framework for shot change detection, which is modeled as a probability minimization problem of the average detection error. Detection criteria are linked with visual content discontinuity (based on motion compensation) and knowledge about shot length distribution.

Han et al. [68] base their detection on gray-level or colour histogram differences computed between successive frames using Eq. (17) or (19). A filtering step combining an average clipping operation and a subsequent local convolution is used to improve the shot change detection. The evolution curve of the filtered histogram difference value is analysed and decision for the detection of a shot change is taken following a Bayesian framework. Detections of a cut or a gradual effect are, respectively, linked with the presence in the evolution curve of a rectangular or triangular shape.

5.8. Statistical approaches

Yilmaz and Shah [69] use Principal Coordinate System to perform shot change detection on RGB frames. First, image rows are concatenated in order to obtain only one row vector per colour component for each frame. We use the notations $V(I_t, C_R)$, $V(I_t, C_G)$, and $V(I_t, C_B)$ for the row vectors associated with the Red, Green and Blue components. Then the 3×3 covariance matrix $M(I_t)$ of the RGB colour space is computed as:

$$M(I_t) = \begin{pmatrix} V(I_t, C_R) \\ V(I_t, C_G) \\ V(I_t, C_B) \end{pmatrix} (V(I_t, C_R)^T \;\; V(I_t, C_G)^T \;\; V(I_t, C_B)^T) \quad (86)$$

which requires 18 operations per pixel. Next the vector representing the principal axis is selected and noted $V_{\lambda_{max}}(I_t)$. We recall this vector is the eigenvector associated with the maximum eigenvalue λ_{max} of the covariance matrix. Finally, two successive frames are compared with respect to the angle between their respective principal axes following the equation

$$\text{Detection if : } \left(1 - \frac{V_{\lambda_{max}}(I_t)^T V_{\lambda_{max}}(I_{t-1})}{\|V_{\lambda_{max}}(I_t)\| \, \|V_{\lambda_{max}}(I_{t-1})\|}\right) > T; \quad (87)$$

so the actual complexity is equal to $O(18\mathscr{P})$.

Han and Tewfik [70] use also a principal component analysis to perform shot change detection. First frames are subsampled and represented as column vectors. Then successive frames are grouped in a temporal window. The mean vector μ and the empirical covariance matrix M of this window are computed. Then the unique set of orthonormal eigenvectors of M and their associated eigenvalues are obtained. Each frame in the window is then projected onto the K eigenvectors corresponding to the K largest eigenvalues. Finally, shot changes are detected using the temporal variations of angle and length of the K first principal components.

Li and Wei [71] based their algorithm on the computation of joint probability images between frames. They use a spatio-temporal representation of the successive joint probability images obtained in order to detect shot changes. First a joint probability image is computed (with a cost of $O(2\mathscr{P})$) between two frames, which consists in the frequency of the co-occurrences of intensity or chrominance values. Two similar images I_{t_1} and I_{t_2} will be characterized by a joint probability image $J(I_{t_1}, I_{t_2})$ composed of non-zero values on the diagonal. A distance measure is then defined between two frames using the joint probability image:

$$D_J(I_{t_1}, I_{t_2}) = 1 - \frac{\sum_{(i,j)\in\mathscr{J}'} J(I_{t_1}, I_{t_2}, i, j)}{\sum_{(i,j)\in\mathscr{J}} J(I_{t_1}, I_{t_2}, i, j)}, \quad (88)$$

where $\mathscr{J}$ and $\mathscr{J}'$ represent, respectively, the set of all pixels (i,j) in the joint probability image and the set of all pixels near the diagonal line with a given tolerance δ (i.e. $\mathscr{T}' = \{(i,j) : |i-j| < \delta\}$). As $\mathscr{J}$ and $\mathscr{J}'$ contain about V^2 and V values, the computation of D_J has a cost of $O(\mathscr{N}^2 + \mathscr{N})$. If the value D_J obtained is higher than a fixed threshold, several algorithms are used in order to confirm the presence of a shot change. Dissolve effects are detected using histogram intersection performed on spatio-temporal representations of joint

probability images. This method is characterized by a complexity equal to $O(2\mathscr{P} + \mathscr{N}^2 + \mathscr{N})$.

Gong and Liu [72] perform shot change detection using the Singular Value Decomposition. Every frame is divided in 3×3 blocks on which a 3D colour histogram composed of 125 bins is computed. A vector of 1125 components is then obtained for every frame. The video sequence is represented by a matrix which is processed by a singular value decomposition algorithm. The K largest singular values are kept and are used to compute a similarity measure between two frames. Detection of a shot change is done by comparing the similarity computed between the two frames I_{t_1} and I_{t_2} with a low and a high threshold. If the similarity measure is below the low threshold, no shot change would be detected. On the contrary, if the measure is higher than the high threshold, a shot change is assumed to be found. In the last case (i.e. the similarity measure is between the two thresholds), a refinement step is performed involving frames between I_{t_1} and I_{t_2}.

5.9. Hidden Markov models

Eickeler and Müller [73] use Hidden Markov Models to perform video indexing. Some of the classes represent shot boundary frames. Several features are defined to describe each frame, but only some of them characterize shot boundary frames:

$$d_1(I_t) = \min\left(\begin{array}{c}\dfrac{\sum_{i=1}^{X}\sum_{j=1}^{Y}|P(I_t,i,j)-P(I_{t-1},i,j)|}{XY},\\ \dfrac{\sum_{i=1}^{X}\sum_{j=1}^{Y}|P(I_{t+1},i,j)-P(I_{t-2},i,j)|}{XY}\end{array}\right), \tag{89}$$

$$d_2(I_t) = \sum_{v=0}^{V}\left(\begin{array}{c}|H(I_t,v)-H(I_{t-1},v)|\\ -\text{median}\left(\begin{array}{c}|H(I_{t-1},v)-H(I_{t-2},v)|,\\ |H(I_t,v)-H(I_{t-1},v)|,\\ |H(I_{t+1},v)-H(I_t,v)|\end{array}\right)\end{array}\right), \tag{90}$$

$$d_3(I_t) = \sum_{i=1}^{X}\sum_{j=1}^{Y}\frac{P(I_t,i,j)-P(I_{t-1},i,j)}{P(I_t,i,j)-0.5P(I_{t-1},i,j)+P(I_{t+1},i,j)}, \tag{91}$$

where $d_1(I_t)$, $d_2(I_t)$, and $d_3(I_t)$ represent, respectively, the intensity of motion, the median filtered intensity of difference histograms, and the ratio between the difference pixel and the difference from interpolated pixel. These three measures need, respectively, six operations per pixel, six operations per histogram bin, and five operations per pixel, resulting in an overall complexity equal to $O(11\mathscr{P} + 6\mathscr{N})$. After a learning step using the Baum–Welch algorithm, segmentation is performed using the Viterbi algorithm.

A similar approach using Hidden Markov Models has been proposed by Boreczky and Wilcox [74]. The model is based on image, audio, and motion features. Classification of a frame into a shot boundary class is done using only the image feature of the frame. This feature is defined as a luminance 64-bin histogram difference similar to the one described in Eq. (17).

6. Motion-based methods

As the nature of motion is usually continuous in a video sequence, it can also be used as a criterion to detect shot changes. Based on this fact, several approaches using motion information were proposed in the literature. We review here methods based on global (or camera) motion, motion vectors, optical flow, and correlation in the frequency domain.

6.1. Global motion

Cherfaoui and Bertin [75] detect shot changes in two steps. First the global motion parameters are estimated using an affine transformation model. The estimated motion is then used to classify a shot as fixed, pan, or zoom. If the motion is not coherent through time, a shot change is assumed to be found.

Bouthemy et al. [76] based their detection on a dominant multiresolution motion estimation. This estimation uses a global 2D parametric model composed of six parameters. Once the dominant motion has been estimated, a coefficient $\omega_{i,j}$ is also available for every pixel (i,j). It represents the coherence of the pixel with the dominant motion estimated. Using a predefined threshold, it is possible to define the set of dominant motion-coherent pixels in each frame. The evolution of the set size through time is analysed in order to detect shot changes.

Zugaj and Bouthemy [77] extend the previous method to wipe detection. Here only pixels which are non-coherent with the estimated dominant motion are considered. For each frame, two histograms are computed based on the number of non-coherent pixels along horizontal and vertical axes. For every couple of frames, absolute differences between corresponding histograms are computed and result in two other histograms. The correlation between two successive absolute difference histograms is then measured along the two axes. If one of the two correlation values exceeds a predefined threshold, an horizontal or vertical wipe is detected.

Mann and Picard [78] proposed a method where global motion estimation is performed using an eight-parameter model. They define "video orbits" as a collection of pictures starting from one picture and applying all possible projective coordinate transformations to that picture using the eight motion parameters. Two frames belonging to the same scene will lie in the same orbit or nearly so. So shot changes are detected when the distance between the orbits of successive frames is higher than a threshold.

Complexity of global or dominant motion computation is usually estimated at $O(20\mathscr{P})$ when considering only 2D translations.

6.2. Motion vectors

Akutsu et al. [79] use a motion smoothness measure to detect shot changes. The indexing method uses subsampled video sequences and processes only one frame every k frames. Then the selected frame is divided into 8×8 blocks and each block is matched to a block in the next chosen frame. Motion vector is estimated thanks to the closest matched neighbouring block, which is also used to compute the value of the correlation coefficient. An interframe similarity measure can be defined as the average of these correlations. Another measure called motion smoothness is defined as the ratio between the number of blocks which have significantly moved and the displacement of these blocks. Shot changes are finally detected when occur local extrema in the correlation and motion smoothness ratio values.

Shahraray [80] proposed a similar method. Subsampled video sequences are used and every frame is divided in 12 blocks. A search is performed to match each block from one frame to the most similar block (in a spatial neighbourhood) in the next frame. Motion vector and correlation value are computed in a way similar to the Akutsu et al. method. The main difference from the previous method is the use of a non-linear digital order statistic filter. Correlation values are sorted into ascending order and the first values and their respective motion vectors are used for the computation of an average value which is considered as a similarity measure. As in [79], a local temporal extremum in the similarity measure means shot change detection. A motion-controlled temporal filter is used to avoid false detection due to motion.

Liu et al. [81] based their method on motion-compensated images obtained from motion vector information. First motion vectors of frame I_{t-1} are used to create a motion-compensated version I'_t of the frame I_t. The next step is luminance normalization. The motion-compensated frame I'_t is normalized in order to get the same energy as the original frame I_t. Normalized image is noted I''_t. The original frame I_t is then compared to the two modified frames I'_t and I''_t using a modified version of the χ^2 test applied on their histograms. The result $\chi(I_t, I'_t)$ is compared to an adaptive threshold in order to detect cuts. Fade detection is based on the comparison between $\chi(I_t, I'_t)$ and $\chi(I_t, I''_t)$ which are the two histogram differences computed previously.

When motion vectors are obtained using a fast block-matching technique (as the three step search method or the 2D log search method), and considering a search size of 16×16 pixels, the complexity is estimated at $O(75\mathscr{P})$.

6.3. Optical flow

Fatemi et al. [82] use optical flow as information to detect shot changes. First, the video sequence is divided into overlapping subsequences, defined as three consecutive frames and a fourth predicted frame. Every frame is then divided into B blocks, which are predicted from the first frame to the second one, and from the second frame to the third one. Finally, a set of three matching blocks from the first three frames is used for block prediction into the last frame. If the block prediction does not correctly estimate the content of the last frame, a shot change is assumed to be found.

Optical flow computation is usually characterized in the literature by a cost of $O(105\mathscr{P})$.

6.4. Frequency domain correlation

Porter et al. [83] propose a technique inspired by motion-based algorithms. Correlation between two successive frames is computed and used as a shot change detection measure. In order to compute the interframe correlation, a block-based approach working in the frequency domain is taken. Frames are divided into blocks of 32×32 pixels. Every block in a frame I_{t-1} is matched with a neighbouring block in frame I_t by first computing the normalized correlation between blocks and then seeking and locating the correlation coefficient with the largest magnitude. The normalized correlation is computed in the frequency domain instead of the spatial domain to limit computation time. The average correlation is then obtained for a couple of frames. Shot changes are detected in the presence of local minima of this value.

Complexity of motion estimation obtained by computing the correlation in the frequency domain is assumed to be similar to complexity of fast blockmatching algorithms, i.e. $O(75\mathscr{P})$.

7. Combination of several methods

Even if most of shot change detection methods working in the uncompressed domain can be categorized under pixel-, histogram-, block-, feature-, and

motion-based methods, some cannot. In this section we will present some methods which do not fit into the previously defined categories.

Two systems have been proposed by Quénot and Mulhem [84]. The first one is dedicated to cut detection and based on colour histogram comparison by χ^2 test [20] followed by a simplified version of the contour analysis algorithm [52] where the motion compensation step is replaced by a dilation of the contour points. The two steps require, respectively, five operations per histogram bin and 26 operations per pixel, so the overall cost is equal to $O(26\mathscr{P}+5\mathscr{N})$. In the second method, motion compensation is performed using optical flow. A direct analysis of the resulting images is done in order to detect cuts, whereas first and second temporal derivatives of the images are compared for progressive transition detection.

Gauch et al. [85] detect shot changes thanks to the combination of three image features: the average brightness of each video frame, the change in pixel values and the change in colour distribution from two different frames. Shot changes are detected if at least one of the two following conditions holds:

$$\sum_{i=1}^{X}\sum_{j=1}^{Y} P(I_t,i,j) < T_1, \tag{92}$$

$$\left(\begin{array}{ll} & \left(\sum_{i=1}^{X}\sum_{j=1}^{Y} |P(I_t,i,j)-P(I_{t-\Delta},i,j)| > T_2\right) \\ \wedge & \left(\sum_{v=0}^{255} |H_{256}(I_t,v)-H_{256}(I_{t-\Delta},v)| > T_3\right) \end{array}\right). \tag{93}$$

where Δ represents the temporal skip and H_{256} the histograms quantified to 256 uniformly distributed colours. The two conditions are, respectively, characterized by a cost of $O(\mathscr{P})$ and $O(3\mathscr{P}+3\mathscr{N})$, resulting in an overall complexity of $O(4\mathscr{P}+3\mathscr{N})$. Thresholds T_1, T_2, and T_3 are initialized manually at the beginning of the video using a priori knowledge and updated dynamically during the video analysis using statistical information.

Yusoff et al. [86] perform shot change detection by combining five algorithms. The methods used are the average intensity measurement [16], the Euclidean distance [87], the histogram comparison [21], the likelihood ratio [38], and the motion estimation method characterized, respectively, by a cost of $O(3\mathscr{P})$, $O(2\mathscr{B})$, $O(3\mathscr{P}+9\mathscr{N})$, $O(3\mathscr{P}+15\mathscr{B})$, and $O(75\mathscr{P})$. The overall complexity is then $O(85\mathscr{P}+9\mathscr{N}+17\mathscr{B})$. The last algorithm uses prediction error computed as the sum of absolute differences between the original frame and the reconstructed frame built using motion information. Final decision is obtained using a majority vote (there should be at least three algorithms detecting a shot change to validate this hypothesis).

Sabata and Goldsmizdt [88] perform fusion of multiple cues to parse video. The features used are from different origins. One is linked to colour distribution, another is a set of responses to several texture filters, some tracking information is also involved, and finally edges in spatio-temporal volumes are computed. Colour feature is analysed through the use of weighted histogram, defined for a temporal window $[t_1,t_2]$ with $t_1<t_2$ as

$$H(I_{t_1},I_{t_2},v)=\frac{1}{V}\sum_{t=t_1}^{t_2} w(t)H(I_t,v) \tag{94}$$

with the weight $w(t)$ computed using the 1D Gaussian function $G(t)$ multiplied by a normalization factor. Colour feature for the frame I_t is then defined as

$$D^{colour}(I_t)=\sum_{v=0}^{V}\frac{(H(I_{t-\Delta},I_t,v)-H(I_t,I_{t+\Delta},v))^2}{H(I_{t-\Delta},I_t,v)+H(I_{t+\Delta},I_t,v)}. \tag{95}$$

Texture feature is analysed in a similar way using a set of 12 Gabor filters instead of colour histogram. The 12 Gabor filters are computed using four values for θ and three for λ, which represent, respectively, the orientation and the wavelength of the filter. The next feature is linked to results of feature tracking, where a score is assigned to the tracking in every frame. It is computed by weighting the contribution of each feature from the last frame to the current frame using the weight $1-e^{-\delta_i/k}$, where the constant k determines the sensitivity to the history of the track and δ_i represents the number of frames in the past through which the ith feature was tracked. The distance measure between two sets of frames $[I_{t-\Delta},I_t]$ and $[I_t,I_{t+\Delta}]$ is then computed using the difference between their average track scores and their missed tracks ratios. The last feature used is linked to spatio-temporal volumes. These volumes are built thanks to the projection of the video data along the (x,t) and the (y,t) planes, as in [89]. For a given sequence, edges perpendicular to the t-axis may indicate shot changes. So an average value of the fraction of pixels covered by the horizontal edges for every set of frames to be analysed is computed and is used as a probability measure of the existence of a shot change. Once all the features are obtained, the final decision is taken according to the result of a fusion process performed using a Bayesian network.

Naphade et al. [90] detect shot changes using a K-means clustering algorithm using two metrics per frame couple. The two metrics are, respectively, based on histogram difference and pixel intensity difference. Then all couples of successive frames are classified, based on these two metrics and using a K-means clustering algorithm with two clusters. The cluster characterized by higher values represents frames containing shot changes. A final step is necessary to eliminate false positive. Frames with a local maximum value for the

histogram difference metric are selected as shot change boundaries. The complexity has been evaluated to $O(4\mathscr{P}+9\mathscr{N})$.

Oh et al. [91] propose a three-step method based on a Gaussian pyramid representation of the background area of images. Background area is defined as a concatenation of three regions located, respectively, on the left, the top, and the right of the image. The first step, called pixel matching, is based on the computation of the pixel difference ratio, defined as

$$\left(\frac{\sum_{i=1}^{X}\sum_{j=1}^{Y} D(I_t, I_{t-1}, i, j)}{XY}\right) \geqslant 0.1, \tag{96}$$

where $D(I_t, I_{t-1}, i, j)$ is defined in Eq. (4). If this condition is satisfied, there may be a shot change. So in a next step a matching is performed between signs of two successive frames. A sign is defined as the pixel on the top of the Gaussian pyramid. The matching can be represented by

$$\left(\frac{1}{256}\max_{k\in\{R,G,B\}}\begin{pmatrix} \operatorname{sign}^{BA}(I_t, C_k) \\ - \quad \operatorname{sign}^{BA}(I_{t-1}, C_k)\end{pmatrix}\right) \geqslant 0.1, \tag{97}$$

where $\operatorname{sign}^{BA}(I_t, C_k)$ denotes the background area sign value of the C_k colour component for the frame I_t. If this second condition is satisfied, the detection process continues with a final step called background tracking. This is done by comparing signatures of two successive frames. Signatures are 1D arrays extracted from background areas using Gaussian pyramid. The comparison is done by shifting the two arrays in opposite directions. In each matching step, only the overlapping pixels in the matching window are compared to determine the maximum continuous match, which is the maximum amount of consecutive pixels matched in the two frames. This measure is finally compared to a certain threshold in order to determine the presence of a shot change.

Ferman and Tekalp [92] extend the K-means-based method proposed in [36]. The unsupervised clustering process uses, for each couple of frames, two features which are filtered histogram difference (derivated from Eq. (17)) and filtered pair-wise pixel comparison (derivated from Eqs. (5) and (6)). For a given distance measure $D(I_t)$, the filtering is done in two steps:

$$D'(I_t) = \max\left(D(I_t) - \underset{t'\in[t-\Delta_1,t+\Delta_1]}{\operatorname{mean}} D(I_{t'})\right), \tag{98}$$

$$D''(I_t) = \max\left(D'(I_t) - \underset{t'\in[t-\Delta_2,t+\Delta_2]}{\operatorname{median}} D'(I_{t'})\right), \tag{99}$$

where Δ_1 and Δ_2 are temporal window lengths which have to be set. This method has a complexity similar to [90], i.e. $O(4\mathscr{P}+9\mathscr{N})$.

Lee et al. [93] also use a K-means clustering algorithm to perform shot change detection. Every couple of frames is represented by a vector of two components which are linked to pixel-based and histogram-based distance measures. The measures used are normalized versions of those described in Eqs. (2) and (17). A vector is then composed of two features:

$$\begin{pmatrix} \dfrac{1}{255XY}\displaystyle\sum_{i=1}^{X}\sum_{j=1}^{Y}|P(I_t,i,j)-P(I_{t-1},i,j)| \\ \dfrac{1}{2XY}\displaystyle\sum_{v=0}^{V}|H(I_t,v)-H(I_{t-1},v)| \end{pmatrix} \tag{100}$$

which need, respectively, three operations per pixel and three operations per histogram bin. The complexity is then equal to $O(3\mathscr{P}+3\mathscr{N})$. Every couple of frames (I_{t-1}, I_t) is classified using a 2-class K-means clustering algorithm applied on pairs of frames from (I_{t_0}, I_{t_0+1}) to (I_{t_f-1}, I_{t_f}) where I_{t_0} and I_{t_f} represent, respectively, the first and final frames of the video sequence. The two classes represent the set of couples of frames where a cut is present and its complement.

8. Complexity results

We reviewed in previous sections a great number of methods for segmentation of uncompressed video sequences. In order to compare the different methods, we have to select a criterion. Quality evaluation, as detailed earlier, depends directly on the selection of optimal parameters for each method. So we decide to compare reviewed methods based on a complexity point of view rather than a quality one. As we are focusing on real-time video parsing, this criterion is also of highest importance.

In this section, we will present complexity of reviewed methods. More precisely, theoretical and experimental results will be given.

8.1. Theoretical complexity

We present in Tables 1–6 results of complexity computation for the different categories of video segmentation methods presented in this paper. Complexity computation results are given for most of the methods reviewed here. However, complexity was not computed for some methods providing insufficient available information, as for example video segmentation methods dealing with wavelet transform, Hough transform, or motion compensation.

Methods with complexity annoted by * are characterized by higher complexity because of the explicit use of colour. Other methods were considered in a gray-level framework. Indeed, colour-based methods dealing with a subsampled colour space (for example, 64-bin colour histogram-based methods) are considered as gray level for complexity computation. For several methods, the complexity computed does not include some specific processing. So the actual complexities of the concerned

Table 1
Complexity and relative computation time of pixel-based methods

Method and reference(s)	Complexity	Time
Simple interframe difference [20]	$O(\mathscr{P})$	1.00
Template matching [20]	$O(3\mathscr{P})$	2.11
Colour template matching [20]	$O(9\mathscr{P})^*$	5.29
Boolean difference [21]	$O(2\mathscr{P})$	1.80
Improved Boolean difference [21]	$O(4\mathscr{P})$	2.60
Normalized difference energy [22]	$O(5\mathscr{P})$	3.00
Normalized sum of absolute differences [22]	$O(5\mathscr{P})$	2.58
Pixel labeling [23]	$O(15\mathscr{P})$	
Evolution of temporal derivative of pixel intensities [24]	$O(8\mathscr{P})_+$	

Table 2
Complexity and relative computation time of histogram-based methods

Method and reference(s)	Complexity	Time
Histogram creation	$O(\mathscr{P})$	
Color histogram creation	$O(3\mathscr{P})$	
Histogram difference [20,25]	$O(3\mathscr{N})$	4.29
Chrominance histogram difference [12]	$O(6\mathscr{N})^*$	55.62
Highest histogram difference [26]	$O(9\mathscr{N})^*$	16.47
Hue histogram difference [27]	$O(4\mathscr{N})$	111.56
Normalized histogram difference [27]	$O(4\mathscr{N})$	12.18
Cosine similarity measure [28,29]	$O(4\mathscr{N})$	45.84
Genetic algorithm [30] (colour histogram only)	$O(9\mathscr{N})^*_+$	
Twin comparison [21]	$O(3\mathscr{N})$	
Squared histogram difference [22]	$O(6\mathscr{N})$	11.18
Kolmogorov–Smirnov statistic [22]	$O(3\mathscr{N})$	4.29
Ponderation relative to colour importance [10]	$O(3\mathscr{P}+9\mathscr{N})^*$	19.58
Ponderation after learning step [33]	$O(15\mathscr{N})^*_+$	
Average colour [12]	$O(6\mathscr{N})^*$	21.56
Reference colour table [12]	$O(8\mathscr{N})$	12.29
Histogram intersection [12,34,35]	$O(2\mathscr{N})$	4.29
Normalized histogram intersection [12]	$O(4\mathscr{N})$	12.18
Original χ^2 test [20]	$O(5\mathscr{N})$	12.29
χ^2 test normalized by max [10]	$O(5\mathscr{N})$	12.27
χ^2 test on colour histograms [36]	$O(15\mathscr{N})^*$	
Cross-entropy [22,37]	$O(4\mathscr{N})$	12.29
Divergence [22,37]	$O(8\mathscr{N})$	12.27
Kullback Liebler distance [22,37]	$O(11\mathscr{N})$	12.29
Bhattacharya distance [22]	$O(3\mathscr{N})$	12.27

methods are in fact higher than values given here. We use the notation + in this case.

We also have to notice that complexity of methods dealing with histograms (mainly those in Table 2) does not include the cost of histogram computation. It is then necessary to add a cost of $O(\mathscr{P})$ to every method using histograms.

Block-based methods (Table 3) may not be characterized by a lower complexity. It is mainly due to extraction of block features which need a processing of all pixels in the image. Some of these methods (like [49]) can be used to segment compressed video sequences using, for example, the DC term as a block average value. In this case, the real complexity is much lower than using uncompressed frames. The same remark applies to several motion-based methods which detect shot changes using motion vectors. These vectors can be directly available in compressed video sequences whereas they have to be computed in uncompressed video sequences.

The theoretical estimations given here have been verified by actual experiments.

8.2. Experimental complexity

We estimated theoretically the complexity of most of the reviewed methods. In order to verify these estimations, an important part of reviewed methods were

Table 3
Complexity and relative computation time of block-based methods

Method and reference(s)	Complexity	Time
Similarity ratio [38]	$O(3\mathscr{P}+15\mathscr{B})$	85.29
Yakimovsky likelihood ratio [22]	$O(3\mathscr{P}+13\mathscr{B})$	88.67
Freund statistic	$O(3\mathscr{P}+11\mathscr{B})$	79.20
Block differences in the HSV colour space [39]	$O(2\mathscr{P}+10\mathscr{B})^*$	179.05
Colour histogram comparison [40]	$O(15\mathscr{N}\mathscr{B})+3\mathscr{B})^*$	206.42
Selective RGB histogram comparison [20]	$O(4\mathscr{N}\mathscr{B}+0,5\mathscr{B})_+$	208.42
Rate of correlation change [41]	$O(4\mathscr{N}\mathscr{B}+2\mathscr{B})$	
Block-based normalized histogram difference [27]	$O(4\mathscr{N}\mathscr{B}+2\mathscr{B})$	233.38
Selective HSV histogram comparison [43]	$O(6\mathscr{N}\mathscr{B}+2\mathscr{B}+4\mathscr{P})^*$	300.00
Block-based HSI histogram comparison [44]	$O(4\mathscr{N}\mathscr{B}+\mathscr{B})^*$	
L*u*v* histogram intersection [45]	$O(12\mathscr{N}\mathscr{B}+2\mathscr{B})^*$	
Histogram difference and likelihood ratio [46]	$O(3\mathscr{P}+3\mathscr{N}+21\mathscr{B})$	
Neighbourhood colour ratio [47]	$O(5\mathscr{P}+8\mathscr{B})$	45.93
RGB block dissimilarity [48]	$O(9\mathscr{P}+3\mathscr{B})^*$	
HSV block dissimilarity [49]	$O(2\mathscr{P}+8\mathscr{B})^*$	114.25
Temporal and spatial subsampling [50]	$O(\mathscr{P}+5\mathscr{B})$	36.45

Table 4
Complexity and relative computation time of feature-based methods

Method and reference(s)	Complexity	Time
Moment invariants [51]	$O(23\mathscr{P})$	
Edge change ratio [52,53]	$O(26\mathscr{P})_+$	240.67
Edge-based contrast [9]	$O(25\mathscr{P})$	
Histogram difference, edge points count, double chromatic difference [54]	$O(27\mathscr{P}+3\mathscr{N})$	
Planar points [58]	$O(51\mathscr{P})$	
Transition modeling using centroids [16]	$O(10\mathscr{P})$	11.89
Transition modeling using histograms [60]	$O(2\mathscr{N}^2+10\mathscr{N})$	
Histogram of pixel-pair differences [61]	$O(8\mathscr{P})_+$	
Transition modeling using mean and variance [9,62–65]	$O(3\mathscr{P})$	
Principal coordinate system [69]	$O(18\mathscr{P})^*$	35.95
Joint probability images [71]	$O(2\mathscr{P}+\mathscr{N}^2+\mathscr{N})$	
Hidden Markov Model with three features [73]	$O(13\mathscr{P}+11\mathscr{N})$	

Table 5
Complexity and relative computation time of motion-based methods

Method and reference(s)	Complexity	Time
Global motion [75–78]	$O(20\mathscr{P})_+$	80.80
Motion vectors [79–81]	$O(75\mathscr{P})_+$	2312.73
Optical flow [82]	$O(105\mathscr{P})_+$	
Frequency domain correlation [83]	$O(75\mathscr{P})_+$	105.65

implemented. Experimental results are also given in Tables 1–6.

As the computation time depends on the programming language and the computing architecture used, results are given in a relative form rather than an absolute one. Simple interframe difference [20] being the fastest method, it is used as a reference computation time.

Tests have been performed using gray-level or RGB colour video sequences. Images are composed of 192 × 144 pixels. The implementation of the reviewed methods has been performed using Matlab programming environment. As we compare methods on a relative basis, the absolute computation time is not critical. So we use Matlab rather than C or C++ in order to limit implementation time.

Due to the Matlab implementation, some apparently strange results have to be explained. All methods performing a colour space conversion (chrominance histogram difference, hue histogram difference, cosine similarity measure, block difference in the HSV colour space, selective HSV histogram comparison, and HSV block dissimilarity) are characterized by higher computation time as this kind of operation is quite slow using Matlab. Block-based methods were implemented using Matlab `blkproc` and `colfilt` functions to avoid loops, which perform really quite slow in Matlab. As `colfilt` performs faster than `blkproc` (ratios observed experimentally are between 135% and 175 %), it was used as

Table 6
Complexity and relative computation time of other methods

Method and reference(s)	Complexity	Time
χ^2 test and edge analysis [84]	$O(26\mathscr{P}+5\mathscr{N})$	243.07
Average brightness, pixel value change and colour distribution change [85]	$O(4\mathscr{P}+3\mathscr{N})$	16.47
Combination of five algorithms [86]	$O(85\mathscr{P}+9\mathscr{N}+17\mathscr{B})_+$	
K-means using colour histogram and pixel intensity differences [90,92]	$O(4\mathscr{P}+9\mathscr{N})^*$	
K-means using gray-level histogram and pixel intensity differences [93]	$O(3\mathscr{P}+3\mathscr{N})$	

often as possible. However, even if we use adequate Matlab functions, computation time of block-based methods stays high. Finally, motion-based methods are characterized by very variable computation times, depending on the use or not of Matlab optimized functions. As motion vectors are obtained using the block matching technique which requires several nested loops, methods [79–81] using motion vectors show actual computation time worse than theoretical complexity.

9. Conclusion

We presented in this paper a great number of methods for segmentation of uncompressed video sequences. We focused on the complexity of these methods, contrary to other previously published reviews. This criterion is particularly important for two main reasons. Firstly, uncompressed video contains a huge quantity of data, which is not the case of compressed video. So video parsing method dealing with uncompressed video will be characterized by their computational intensive aspect. Secondly, when dealing with real-time video segmentation, we have to use methods known for their low computational cost in order to process video at frame rate.

Results of this study show that the best method for real-time segmentation of uncompressed video sequences should be selected considering efficiency and computational cost. Simple methods like interframe difference is one of the fastest method but it is characterized by a poor quality. On the opposite, feature- or motion-based methods or methods combining several algorithms are more robust and lead to better quality, but they are also known to be greedy in computational resources. Another important criterion to choose a method is the way of computing threshold values and the number of thresholds. Future work will deal with the study of this criterion.

References

[1] Ahanger G, Little TDC. A survey of technologies for parsing and indexing digital video. Journal of Visual Communication and Image Representation 1996;7(1):28–43.
[2] Idris F, Panchanathan S. Review of image and video indexing techniques. Journal of Visual Communication and Image Representation 1997;8(2):146–66.
[3] Brunelli R, Mich O, Modena CM. A survey on the automatic indexing of video data. Journal of Visual Communication and Image Representation 1999;10(2):78–112.
[4] Aigrain P, Zhang HJ, Petkovic D. Content-based representation and retrieval of visual media: a state-of-the-art review. International Journal of Multimedia Tools and Applications 1996; 3(3):179–202.
[5] Koprinska I, Carrato S. Temporal video segmentation: a survey. Signal Processing: Image Communication 2001;16(5):477–500.
[6] Lienhart R. Reliable transition detection in videos: a survey and practitioner's guide. International Journal of Image and Graphics 2001;1(3):469–86.
[7] Jiang H, Helal AS, Elmagarmid AK, Joshi A. Scene change detection techniques for video database systems. Multimedia Systems 1998;6(3):186–95.
[8] Boreczky JS, Rowe LA. Comparison of video shot boundary detection techniques. Journal of Electronic Imaging 1996;5(2): 122–8.
[9] Lienhart R. Comparison of automatic shot boundary detection algorithms. In: SPIE Conference on Storage and Retrieval for Image and Video Databases, vol. 3656, San Jose, CA, January 1999. p. 290–301.
[10] Dailianas A, Allen RB, England P. Comparison of automatic video segmentation algorithms. In: SPIE Conference on Integration Issues in Large Commercial Media Delivery Systems, vol. 2615, Philadelphia, PA, October 1995. p. 2–16.
[11] Yuso Y, Christmas W, Kittler J. Video shot cut detection using adaptative thresholding. In: British Machine Vision Conference, Bristol, UK, September 2000. p. 362–71.
[12] Gargi U, Kasturi R. An evaluation of color histogram-based methods in video indexing. In: International Workshop on Image Databases and Multimedia Search, Amsterdam, The Netherlands, August 1996. p. 75–82.
[13] Gargi U, Kasturi R, Strayer SH. Performance characterization of video-shot-change detection methods. IEEE Transactions on Circuits and Systems for Video Technology 2000;10(1):1–13.
[14] Mandal MK, Idris F, Panchanathan S. A critical evaluation of image and video indexing techniques in the compressed domain. Image and Vision Computing 1999;17(7):513–29.
[15] Eickeler S, Rigoll G. A novel error measure for the evaluation of video indexing systems. In: International Conference on Acoustics, Speech, and Signal Processing, Istanbul, Turkey, June 2000.
[16] Hampapur A, Jain RC, Weymouth T. Production model based digital video segmentation. International Journal of Multimedia Tools and Applications 1995;1(1):9–46.
[17] Ruiloba R, Joly P, Marchand-Maillet S, Quénot G. Towards a standard protocol for the evaluation of video-to-shots segmentation algorithms. In: European Workshop on Content Based Multimedia Indexing, Toulouse, France, October 1999. p. 41–8.
[18] Ardebilian M, Tu X, Chen L. Robust 3d clue-based video segmentation for video indexing. Journal of Visual Communication and Image Representation 2000;11(1):58–79.

[19] Drew MS, Wei J, Li ZN. Illumination-invariant image retrieval and video segmentation. Pattern Recognition 1999;32(8): 1369–88.
[20] Nagasaka A, Tanaka Y. Automatic video indexing and full-video search for object appearances. In: IFIP Working Conference on Visual Database Systems, Budapest, Hungary, October 1991. p. 113–27.
[21] Zhang HJ, Kankanhalli A, Smoliar SW. Automatic partitioning of full-motion video. Multimedia Systems 1993;1(1):10–28.
[22] Ren W, Sharma M, Singh S. Automated video segmentation. In: International Conference on Information, Communication, and Signal Processing, Singapore, October 2001.
[23] Taniguchi Y, Akutsu A, Tonomura Y. Panorama excerpts: extracting and packing panoramas for video browsing. In: ACM International Conference on Multimedia, Seattle, WA, November 1997. p. 427–36.
[24] Lawrence S, Ziou D, Auclair-Fortier MF, Wang S. Motion insensitive detection of cuts and gradual transitions in digital videos. Technical Report 266, DMI, University of Sherbrooke, Canada, May 2001.
[25] Tonomura Y, Abe S. Content oriented visual interface using video icons for visual database systems. Journal of Visual Languages and Computing 1990;1(2):183–98.
[26] Pye D, Hollinghurst NJ, Mills TJ, Wood KR. Audio-visual segmentation for content-based retrieval. In: International Conference on Spoken Language Processing, Sydney, Australia, December 1998.
[27] Ahmed M, Karmouch A, Abu-Hakima S. Key frame extraction and indexing for multimedia databases. In: Vision Interface Conference, Trois-Rivières, Canada, May 1999. p. 506–11.
[28] O'Toole C, Smeaton A, Murphy N, Marlow S. Evaluation of automatic shot boundary detection on a large video test suite. In: Conference on Challenge of Information Retrieval, Newcastle, UK, February 1999.
[29] Cabedo XU, Bhattacharjee SK. Shot detection tools in digital video. In: Workshop on Non-linear Model Based Image Analysis, Glasgow, Scotland, July 1998. p. 121–6.
[30] Chiu P, Girgensohn A, Polak W, Rieffel E, Wilcox L. A genetic algorithm for video segmentation and summarization. In: IEEE International Conference on Multimedia and Expo, vol. 3, New York, NY, July 2000. p. 1329–32.
[31] Li D, Lu H. Efficient scene change detection through stdd (single threshold double direction). In: IASTED International Conference on Modelling and Simulation, Pittsburgh, PA, May 2000.
[32] Kong WX, Ding XF, Lu HQ, Ma SD. Improvement of shot detection using illumination invariant metric and dynamic threshold selection. In: International Conference on Visual Information Systems, Amsterdam, The Netherlands, June 1999. p. 277–82.
[33] Zhao W, Wang J, Bhat D, Sakiewics K, Nandhakumar N, Chang W. Improving color based video shot detection. In: IEEE International Conference on Multimedia Computing and Systems, vol. 2, Florence, Italy, June 1999. p. 752–6.
[34] Haering N, da Vitoria Lobo N, Qian R, Sezan I. A framework for designing event detectors. In: Asian Conference on Computer Vision, Taipei, Taiwan, January 2000.
[35] Javed O, Khan S, Rasheed Z, Shah M. A framework for segmentation of interview videos. In: IASTED International Conference on Internet and Multimedia Systems and Applications, Las Vegas, CA, November 2000.
[36] Gunsel B, Ferman AM, Tekalp AM. Temporal video segmentation using unsupervised clustering and semantic object tracking. Journal of Electronic Imaging 1998;7(3):592–604.
[37] Kim S, Park RH. A novel approach to scene change detection using a cross entropy. In: IEEE International Conference on Image Processing, vol. 3, Vancouver, Canada, September 2000. p. 937–40.
[38] Kasturi R, Jain RC. Dynamic vision. In: Kasturi R, Jain RC, editors. Computer vision: principles. Washington, DC: IEEE Computer Society Press, 1991. p. 469–80.
[39] Lee MS, Yang YM, Lee SW. Automatic video parsing using shot boundary detection and camera operation analysis. Pattern Recognition 2001;34(3):711–9.
[40] Swanberg D, Shu CF, Jain R. Knowledge guided parsing and retrieval in video databases. In: SPIE Conference on Storage and Retrieval for Image and Video Databases, vol. 1908, San Jose, CA, February 1993. p. 13–24.
[41] Ueda H, Miyatake T, Yoshizawa S. Impact: an interactive natural-motion-picture dedicated multimedia authoring system. In: ACM Conference on Human Factors in Computing Systems, New Orleans, LO, April 1991. p. 343–50.
[42] Ahmed M, Karmouch A. Video segmentation using an opportunistic approach. In: Multimedia Modeling, Ottawa, Canada, October 1999. p. 389–405.
[43] Lee CM, Ip MC. A robust approach for camera break detection in color video sequences. In: IAPR International Workshop on Machine Vision Applications, Kawasaki, Japan, December 1994. p. 502–5.
[44] Bertini M, Del Bimbo A, Pala P. Content based indexing and retrieval of TV news. Pattern Recognition Letters 2001;22(5): 503–16.
[45] Chahir Y, Chen L. Automatic video segmentation and indexing. In: SPIE Conference on Intelligent Robots and Computer Vision, vol. 3837, Boston, MA, September 1999. p. 345–56.
[46] Dugad R, Ratakonda K, Ahuja N. Robust video shot change detection. In: IEEE Workshop on Multimedia Signal Processing, Redondo Beach, CA, December 1998. p. 376–81.
[47] Adjeroh DA, Lee MC, Orji CU. Techniques for fast partitioning of compressed and uncompressed video. International Journal of Multimedia Tools and Applications 1997;4(2):225–43.
[48] Demarty CH, Beucher S. Morphological tools for indexing video documents. In: IEEE International Conference on Multimedia Computing and Systems, vol. 2, Florence, Italy, June 1999. p. 991–2.
[49] Lefèvre S, Holler J, Vincent N. Real time temporal segmentation of compressed and uncompressed dynamic colour image sequences. In: International Workshop on Real Time Image Sequence Analysis, Oulu, Finland, August 2000. p. 56–62.
[50] Xiong W, Lee JCM. Efficient scene change detection and camera motion annotation for video classification. Journal of Computer Vision and Image Understanding 1998;71(2):166–81.
[51] Arman F, Depommier R, Hsu A, Chiu MY. Content-based browsing of video sequences. In: ACM International Conference on Multimedia, San Francisco, CA, November 1994. p. 97–103.
[52] Zabih R, Miller J, Mai K. A feature-based algorithm for detecting and classifying production effects. Multimedia Systems 1999;7(2):119–28.
[53] Smeaton AF, Gormley G, Gilvarry J, Tobin B, Marlow S, Murphy N. An evaluation of alternative techniques for automatic detection of shot boundaries in digital video. In: Irish Machine Vision and Image Processing Conference, Dublin, Ireland, September 1999. p. 45–62.
[54] Yu H, Bozdagi G, Harrington S. Feature-based hierarchical video segmentation. In: IEEE International Conference on Image Processing, vol. 2, Santa Barbara, CA, October 1997. p. 498–501.
[55] Heng WJ, Ngan KN. Integrated shot boundary detection using object-based technique. In: IEEE International Conference on Image Processing, vol. 3, Kobe, Japan, October 1999. p. 289–93.

[56] Nam J, Tewfik AH. Combined audio and visual streams analysis for video sequence segmentation. In: International Conference on Acoustics, Speech, and Signal Processing, vol. 4, Munich, Germany, April 1997. p. 2665–8.
[57] Nam J, Tewfik AH. Wipe transition detection using polynomial interpolation. In: SPIE Conference on Storage and Retrieval for Media Databases, vol. 4315, San Jose, CA, January 2001. p. 231–41.
[58] Silva R, Gomes J, Velho L. Segmentation of video sequences using volumetric image processing. In: Eurographics Workshop on Multimedia, Milan, Italy, September 1999.
[59] Sanchez JM, Binefa X, Vitria J, Radeva P. Local color analysis for scene break detection applied to TV commercials recognition. In: International Conference on Visual Information Systems, Amsterdam, The Netherlands, June 1999. p. 237–44.
[60] Adami N, Leonardi R. Identification of editing effect in image sequences by statistical modelling. In: Symposium on Picture Coding, Portland, OR, April 1999. p. 157–60.
[61] Aigrain P, Joly P. The automatic real-time analysis of film editing and transition effects and its applications. Computer and Graphics 1994;18(1):93–103.
[62] Alattar AM. Detecting fade regions in uncompressed video sequences. In: International Conference on Acoustics, Speech, and Signal Processing, Munich, Germany, April 1997. p. 3025–8.
[63] Alattar AM. Detecting and compressing dissolve regions in video sequences with dvi multimedia image compression algorithm. In: IEEE International Symposium on Circuits and Systems, vol. 1, Chicago, IL, May 1993. p. 13–6.
[64] Truong BT, Dorai C, Venkatesh S. New enhancements to cut, fade, and dissolve detection processes in video segmentation. In: ACM International Conference on Multimedia, Los Angeles, CA, October 2000. p. 219–27.
[65] Fernando WAC, Canagarajah CN, Bull DR. Fade and dissolve detection in uncompressed and compressed video sequences. In: IEEE International Conference on Image Processing, Kobe, Japan, October 1999. p. 27AP2.
[66] Vasconcelos N, Lippman A. Statistical models of video structure for content analysis and characterization. IEEE Transactions on Image Processing 2001;9(1):3–19.
[67] Hanjalic A, Zhang HJ. Optimal shot boundary detection based on robust statistical models. In: IEEE International Conference on Multimedia Computing and Systems, vol. 2, Florence, Italy, June 1999. p. 710–4.
[68] Han SH, Kweon IS, Kim CY, Seo YS. Bayesian shot detection using structural weighting. In: IAPR International Workshop on Machine Vision Applications, Tokyo, Japan, November 2000. p. 95–8.
[69] Yilmaz A, Shah M. Shot detection using principle coordinate system. In: IASTED International Conference on Internet and Multimedia Systems and Applications, Las Vegas, CA, November 2000.
[70] Han KJ, Tewfik AH. Eigen-image based video segmentation and indexing. In: IEEE International Conference on Image Processing, Santa Barbara, CA, October 1997. p. 538–41.
[71] Li ZN, Wei J. Spatio-temporal joint probability images for video segmentation. In: IEEE International Conference on Image Processing, vol. 2, Vancouver, Canada, September 2000. p. 295–8.
[72] Gong Y, Liu X. Video shot segmentation and classification. In: IAPR International Conference on Pattern Recognition, vol. 1, Barcelona, Spain, September 2000. p. 860–3.
[73] Eickeler S, Müller S. Content-based video indexing of TV broadcast news using hidden Markov models. In: International Conference on Acoustics, Speech, and Signal Processing, Phoenix, AZ, March 1999. p. 2997–3000.
[74] Boreczky JS, Wilcox LD. A hidden Markov model framework for video segmentation using audio and image features. In: International Conference on Acoustics, Speech, and Signal Processing, vol. 6, Seattle, WA, May 1998. p. 3741–4.
[75] Cherfaoui M, Bertin C. Temporal segmentation of videos: a new approach. In: SPIE Conference on Digital Video Compression: Algorithms and Technologies, vol. 2419, San Jose, CA, February 1995. p. 38–47.
[76] Bouthemy P, Gelgon M, Ganansia F. A unified approach to shot change detection and camera motion characterization. IEEE Transactions on Circuits and Systems for Video Technology 1999;9(7):1030–44.
[77] Zugaj D, Bouthemy P. Wipe detection in the temporal segmentation of video. In: European Workshop on Content Based Multimedia Indexing, Toulouse, France, October 1999.
[78] Mann S, Picard RW. Video orbits of the projective group: a simple approach to featureless estimation of parameters. IEEE Transactions on Image Processing 1997;6(9):1281–95.
[79] Akutsu A, Tonomura Y, Hashimoto H, Ohba Y. Video indexing using motion vectors. In: SPIE Conference on Visual Communications and Image Processing, vol. 1818, Boston, MA, November 1992. p. 1522–30.
[80] Shahraray B. Scene change detection and content-based sampling of video sequences. In: SPIE Conference on Digital Video Compression: Algorithms and Technologies, vol. 2419, San Jose, CA, February 1995. p. 2–13.
[81] Liu T, Zhang X, Wang D, Feng J, Lo KT. Inertia-based cut detection technique: a step to the integration of video coding and content-based retrieval. In: IEEE International Conference on Signal Processing, Beijing, China, August 2000. p. 1018–25.
[82] Fatemi O, Zhang S, Panchanathan S. Optical flow based model for scene cut detection. In: Canadian Conference on Electrical and Computer Engineering, vol. 1, Calgary, Canada, May 1996. p. 470–3.
[83] Porter SV, Mirmehdi M, Thomas BT. Video cut detection using frequency domain correlation. In: IAPR International Conference on Pattern Recognition, vol. 3, Barcelona, Spain, September 2000. p. 413–6.
[84] Quénot G, Mulhem P. Two systems for temporal video segmentation. In: European Workshop on Content Based Multimedia Indexing, Toulouse, France, October 1999. p. 187–94.
[85] Gauch JM, Gauch S, Bouix S, Zhu X. Real time video scene detection and classification. Information Processing and Management 1999;35(3):401–20.
[86] Yusoff Y, Kittler J, Christmas W. Combining multiple experts for classifying shot changes in video sequences. In: IEEE International Conference on Multimedia Computing and Systems, vol. 2, Florence, Italy, June 1999. p. 700–4.
[87] Vellakal A, Kuo CCJ. Joint spatial-spectral indexing for image retrieval. In: IEEE International Conference on Image Processing, vol. 3, Lausanne, Switzerland, September 1996. p. 867–70.
[88] Sabata B, Goldszmidt M. Fusion of multiple cues for video segmentation. In: International Conference on Information Fusion, Sunnyvale, CA, July 1999.
[89] Otsuji K, Tonomura Y. Projection-detecting filter for video cut detection. Multimedia Systems 1994;1(5):205–10.
[90] Naphade MR, Mehrotra P, Ferman AM, Warnick J, Huang TS, Tekalp AM. A high-performance shot boundary detection algorithm using multiple cues. In: IEEE International Conference on Image Processing, vol. 2, Chicago, IL, October 1998. p. 884–7.
[91] Oh JH, Hua KA, Liang N. A content-based scene change detection and classification technique using background tracking. In: SPIE Conference on Multimedia Computing and Networking, vol. 3969, San Jose, CA, January 2000. p. 254–65.

[92] Ferman AM, Tekalp AM. Efficient filtering and clustering for temporal video segmentation and visual summarization. Journal of Visual Communication and Image Representation 1998;9(4): 336–51.

[93] Lee HC, Lee CW, Kim SD. Abrupt shot change detection using an unsupervised clustering of multiple features. In: International Conference on Acoustics, Speech, and Signal Processing, vol. 3, Istanbul, Turkey, June 2000. p. 2015–8.