ELSEVIER

Signal Processing: *Image Communication* 16 (2001) 477–500

SIGNAL PROCESSING: IMAGE COMMUNICATION

www.elsevier.nl/locate/image

Temporal video segmentation: A survey

Irena Koprinska[a], Sergio Carrato[b,*]

[a]*Institute for Information Technologies, Acad. G. Bonchev Str., Bl. 29A, 1113 Sofia, Bulgaria*
[b]*Department of Electrical Engineering and Computer Science (D.E.E.I), Image Processing Laboratory, University of Trieste, via Valerio 10, 34127 Trieste, Italy*

Received 27 July 1999; received in revised form 8 February 2000; accepted 15 February 2000

Abstract

Temporal video segmentation is the first step towards automatic annotation of digital video for browsing and retrieval. This article gives an overview of existing techniques for video segmentation that operate on both uncompressed and compressed video stream. The performance, relative merits and limitations of each of the approaches are comprehensively discussed and contrasted. The gradual development of the techniques and how the uncompressed domain methods were tailored and applied into compressed domain are considered. In addition to the algorithms for shot boundaries detection, the related topic of camera operation recognition is also reviewed.

Keywords: Temporal video segmentation; Shot boundaries detection; Camera operations; Video databases

1. Introduction

Recent advances in multimedia compression technology, coupled with the significant increase in computer performance and the growth of Internet, have led to the widespread use and availability of digital video. Applications such as digital libraries, distance learning, video-on-demand, digital video broadcast, interactive TV, multimedia information systems generate and use large collections of video data. This has created a need for tools that can efficiently index, search, browse and retrieve relevant material. Consequently, several content-based retrieval systems for organizing and managing video databases have been recently proposed [8,26,34].

As shown in Fig. 1, temporal video segmentation is the first step towards automatic annotation of digital video sequences. Its goal is to divide the video stream into a set of meaningful and manageable segments (*shots*) that are used as basic elements for indexing. Each shot is then represented by selecting key frames and indexed by extracting spatial and temporal features. The retrieval is based on the similarity between the feature vector of the query and already stored video features.

A shot is defined as an unbroken sequence of frames taken from one camera. There are two basic types of shot transitions: *abrupt* and *gradual*. Abrupt transitions (*cuts*) are simpler, they occur in a single frame when stopping and restarting the

* Corresponding author. Tel.: + 39 040-676-7147; fax: + 39-040-676-3460.

E-mail addresses: irena@iinf.bas.bg (I. Koprinska), carrato@ipl.univ.trieste.it (S. Carrato).

0923-5965/01/$ - see front matter
PII: S0923-5965(00)00011-4

camera. Although many kinds of cinematic effects could be applied to artificially combine two shots, and thus to create gradual transitions, most often *fades* and *dissolves* are used. A *fade out* is a slow decrease in brightness resulting in a black frame; a *fade in* is a gradual increase in intensity starting from a black image. *Dissolves* show one image superimposed on the other as the frames of the first shot get dimmer and those of the second one get brighter. Fig. 2 shows an example of dissolve and cut. Fade out followed by fade in is presented in Fig. 3.

Gradual transitions are more difficult to detect than cuts. They must be distinguished from camera operations (Fig. 4) and object movement that exhibit temporal variances of the same order and cause false positives. It is particularly difficult to detect dissolves between sequences involving intensive motion [14,44,47].

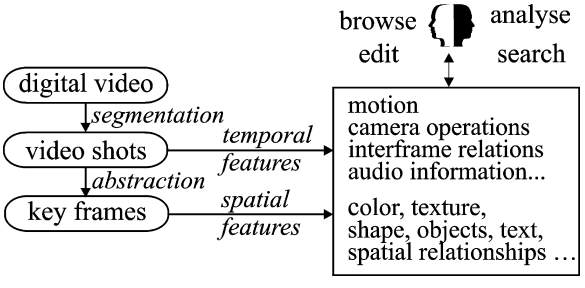

Fig. 1. Content-based retrieval of video databases.

Camera operation recognition is an important issue also for another reason. As camera operations usually explicitly reflect how the attention of the viewer should be directed, the clues obtained are useful for key frame selection. For example, when a camera pans over a scene, the entire video sequence belongs to one shot but the content of the scene could change substantially, thus suggesting the use of more than one key frame. Also, when the camera zooms, the images at the beginning and end of the zoom may be considered as representative of the entire shot. Furthermore, recognizing camera operations allows the construction of salient video stills [38] – static images that efficiently represent video content.

Algorithms for shot boundaries detection were already discussed in several review papers. Ananger and Little [4] presented a survey in video indexing, including some techniques for temporal video segmentation mainly in uncompressed domain. Idris and Panchanathan [15] surveyed methods for content-based indexing in image and video databases focusing on feature extraction. A review of video parsing is presented but it mainly includes methods that operate on uncompressed domain and detect cuts. The goal of this paper is to provide a comprehensive taxonomy and critical survey of the existing approaches for temporal video segmentation in both uncompressed and compressed video. The performance, relative merits and shortcomings of each

Fig. 2. Dissolve, cut.

Fig. 3. Fade out followed by fade in.

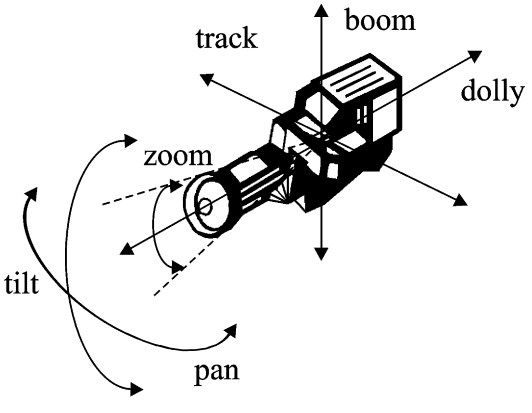

Fig. 4. Basic camera operations: fixed, zooming (focal length change of a stationary camera), panning/tilting (camera rotation around its horizontal/vertical axis), tracking/booming (horizontal/vertical transverse movement) and dollying (horizontal lateral movement).

approach are discussed in detail. A special attention is given to the gradual development and improvement of the techniques, their relationships and similarities, in particular how the uncompressed domain methods were tailored and imported into the compressed domain. In addition to the algorithms for shot boundaries detection, the related topic of camera operation recognition is also discussed.

The paper is organized as follows. In the next section we review shot boundaries detection techniques starting with approaches in uncompressed domain and then moving to compressed domain via an introduction to MPEG fundamentals. An overview of methods for camera operation recognition is presented in Section 3. Finally, a summary with future directions concludes the paper.

2. Temporal video segmentation

More than eight years of temporal video segmentation research have resulted in a great variety of algorithms. Early work focus on cut detection, while more recent techniques deal with the harder problem – gradual transitions detection.

2.1. Temporal video segmentation in uncompressed domain

The majority of algorithms process uncompressed video. Usually, a similarity measure

between successive images is defined. When two images are sufficiently dissimilar, there may be a cut. Gradual transitions are found by using cumulative difference measures and more sophisticated thresholding schemes.

Based on the metrics used to detect the difference between successive frames, the algorithms can be divided broadly into three categories: pixel, block-based and histogram comparisons.

2.1.1. Pixel comparison

Pair-wise pixel comparison (also called template matching) evaluates the differences in intensity or color values of corresponding pixels in two successive frames.

The simplest way is to calculate the absolute sum of pixel differences and compare it against a threshold [18]:

$$D(i,i+1)=\frac{\sum_{x=1}^{X}\sum_{y=1}^{Y}|P_i(x,y)-P_{i+1}(x,y)|}{XY}$$

for gray level images,

$$D(i,i+1)=\frac{\sum_{x=1}^{X}\sum_{y=1}^{Y}\sum_{c}|P_i(x,y,c)-P_{i+1}(x,y,c)|}{XY}$$

for color images, (1)

where i and $i+1$ are two successive frames with dimension $X\times Y$, $P_i(x,y)$ is the intensity value of the pixel at the coordinates (x,y) in frame i, c is index for the color components (e.g. $c\in\{R,G,B\}$ in case of RGB color system) and $P_i(x,y,c)$ is the color component of the pixel at (x,y) in frame i.

A cut is detected if the difference $D(i,i+1)$ is above a prespecified threshold T. The main disadvantage of this method is that it is not able to distinguish between a large change in a small area and a small change in a large area. For example, cuts are misdetected when a small part of the frame undergoes a large, rapid change. Therefore, methods based on simple pixel comparison are sensitive to object and camera movements.

A possible improvement is to count the number of pixels that change in value more than some threshold and to compare the total against a second threshold [25,45]:

$$\mathrm{DP}(i,i+1,x,y)=\begin{cases}1 & \text{if } |P_i(x,y)-P_{i+1}(x,y)|>T_1,\\ 0, & \text{otherwise,}\end{cases}$$

$$D(i,i+1)=\frac{\sum_{x=1}^{X}\sum_{y=1}^{Y}\mathrm{DP}(i,i+1,x,y)}{XY}. \quad (2)$$

If the percentage of changed pixels $D(i,i+1)$ is greater than a threshold T_2, a cut is detected.

Although some irrelevant frame differences are filtered out, these approaches are still sensitive to object and camera movements. For example, if camera pans, a large number of pixels can be judged as changed, even though there is actually a shift with a few pixels. It is possible to reduce this effect to a certain extent by the application of a smoothing filter: before the comparison each pixel is replaced by the mean value of its neighbors.

2.1.2. Block-based comparison

In contrast to template matching that is based on global image characteristic (pixel by pixel differences), block-based approaches use local characteristic to increase the robustness to camera and object movement. Each frame i is divided into b blocks that are compared with their corresponding blocks in $i+1$. Typically, the difference between i and $i+1$ is measured by

$$D(i,i+1)=\sum_{k=1}^{b}c_k\,\mathrm{DP}(i,i+1,k), \quad (3)$$

where c_k is a predetermined coefficient for the block k and $\mathrm{DP}(i,i+1,k)$ is a partial match value between the kth blocks in i and $i+1$ frames.

In [17] corresponding blocks are compared using a likelihood ratio

$$\lambda_k=\frac{\left[\frac{\sigma_{k,i}+\sigma_{k,i+1}}{2}+\left(\frac{\mu_{k,i+1}-\mu_{k,i+1}}{2}\right)^2\right]^2}{\sigma_{k,i}\cdot\sigma_{k,i+1}}, \quad (4)$$

where $\sigma_{k,i}$, $\sigma_{k,i+1}$ are the mean intensity values for the two corresponding blocks k in the consecutive frames i and $i+1$, and $\sigma_{k,i}$, $\sigma_{k,i+1}$ are their variances, respectively. Then, the number of blocks for which the likelihood ratio is greater than

a threshold T_1 is counted,

$$\mathrm{DP}(i, i+1, k) = \begin{cases} 1 & \text{if } \lambda_k > T_1, \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

A cut is declared when the number of changed blocks is large enough, i.e. $D(i, i+1)$ is greater than a given threshold T_2 and $c_k = 1$ for all k.

Compared to template matching, this method is more tolerant to slow and small object motion from frame to frame. On the other hand, it is slower due to the complexity of the statistical formulas. Additional potential disadvantage is that no change will be detected in the case of two corresponding blocks that are different but have the same density function. Such situations, however, are very unlikely.

Another block-based technique is proposed by Shahraray [32]. The frame is divided into 12 non-overlapping blocks. For each of them the best match is found in the respective neighborhoods in the previous image based on image intensity values. A non-linear order statistics filter is used to combine the match values, i.e. the weight of a match value in Eq. (3) will depend on its order in the match value list. Thus, the effect of camera and object movements is further suppressed. The author claims that such similarity measure of two images is more consistent with human judgement. Both cuts and gradual transitions are detected. Cuts are found using thresholds like in the other approaches that are discussed while gradual transitions are detected by identifying sustained low-level increase in match values.

Xiong et al. [41] describe a method they call *net comparison*, which attempts to detect cuts inspecting only part of the image. It is shown that the error will be low enough if less than half of so called base windows (non-overlapping square blocks, Fig. 5) are checked. Under an assumption about the largest movement between two images, the size of the windows can be chosen large enough to be indifferent to a non-break change and small enough to contain the spatial information as much as possible. Base windows are compared using the difference between the mean values of their gray-level or color values. If this difference is larger than a threshold, the region is considered changed. When the number of changed windows is greater than another threshold, a cut is declared. The experiments demonstrated that the approach is faster and more accurate than pixel pair-wise, likelihood and local histogram methods. In their subsequent paper [40], the idea of video subsampling into space is further extended to subsampling in both space and time. The new *Step-variable* algorithm detects both abrupt and gradual transition comparing frames i and j, where $j = i + \mathrm{myStep}$. If no significant change is found between them, the move is with half step forward and the next comparison is between $i + \mathrm{myStep}/2$ and $j + \mathrm{myStep}/2$. Otherwise, binary search is used to locate the change. If i and j are successive and their difference is bigger than a threshold, cut is declared. Otherwise, edge differences between the two frames are compared against another threshold to check for gradual transition. Obviously, the performance depends on the proper setting of myStep: large steps are efficient but increase the number of false alarms, too small steps may result in missing gradual transition. In addition, the approach is very sensitive to object and camera motion.

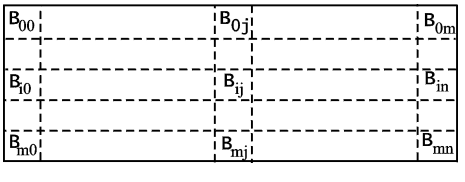

Fig. 5. Net comparison algorithm: base windows B_{ij}.

2.1.3. Histogram comparison

A step further towards reducing sensitivity to camera and object movements can be done by comparing the histograms of successive images. The idea behind histogram-based approaches is that two frames with unchanging background and unchanging (although moving) objects will have little difference in their histograms. In addition, histograms are invariant to image rotation and change slowly under the variations of viewing angle and scale [35]. As a disadvantage one can note that two images with similar histograms may have completely different content. However, the probability for such events is low enough, moreover techniques for dealing with this problem have already been proposed in [28].

A gray level (color) histogram of a frame i is an n-dimensional vector $H_i(j)$, $j = 1, \ldots, n$, where n is the number of gray levels (colors) and $H(j)$ is the number of pixels from the frame i with gray level (color) j.

2.1.3.1. Global histogram comparison. The simplest approach uses an adaptation of the metrics from Eq. (1): instead of intensity values, gray level histograms are compared [25,39,45]. A cut is declared if the absolute sum of histogram differences between two successive frames $D(i, i+1)$ is greater than a threshold T,

$$D(i, i+1) = \sum_{j=1}^{n} |H_i(j) - H_{i+1}(j)|, \tag{6}$$

where $H_i(j)$ is the histogram value for the gray level j in the frame i, j is the gray value and n is the total number of gray levels.

Another simple and very effective approach is to compare color histograms. Zhang et al. [45] apply Eq. (6) where j, instead of gray levels, denotes a code value derived from the three color intensities of a pixel. In order to reduce the bin number (3 colors $\times 8$ bits create histograms with 2^{24} bins), only the upper two bits of each color intensity are used to compose the color code. The comparison of the resulting 64 bins has been shown to give sufficient accuracy.

To enhance the difference between two frames across a cut, several authors [25] propose the use of the χ^2 test to compare the (color) histograms $H_i(j)$ and $H_{i+1}(j)$ of the two successive frames i and $i+1$,

$$D(i, i+1) = \sum_{j=1}^{n} \frac{|H_i(j) - H_{i+1}(j)|^2}{H_{i+1}(j)}. \tag{7}$$

When the difference is larger than a given threshold T, a cut is declared. However, experimental results reported in [45] show that χ^2 test not only enhances the difference between two frames across a cut but also increases the difference due to camera and object movements. Hence, the overall performance is not necessarily better than the linear histogram comparison represented in Eq. (6). In addition, χ^2 statistics requires more computational time.

Gargi et al. [12] evaluate the performance of three histogram based methods using six different color coordinate systems: *RGB*, *HSV*, *YIQ*, *L*a*b**, *L*u*v** and Munsell. The *RGB* histogram of a frame is computed as three sets of 256 bins. The other five histograms are represented as a 2-dimensional distribution over the two non-intensity based dimensions of the color spaces, namely: H and S for the *HSV*, I and Q for the *YIQ*, a^* and b^* for the *L*a*b**, u^* and v^* for the *L*u*v** and hue and chroma components for the Munsell space. The number of bins is 1600 (40×40) for the *L*a*b**, *L*u*v** and *YIQ* histograms and 1800 (60 hues $\times$ 30 saturations/chromas) for the *HSV* and Munsell space histograms. The difference functions used to compare histograms of two consecutive frames are defined as follows:

- bin-to-bin differences as in Eq. (6)
- histogram intersection:

$$D(i, i+1) = 1 - \text{Intersection}(H_i, H_{i+1}) = 1 - \frac{\sum_{j=1}^{n} \min(H_i(j) - H_{i+1}(j))}{\sum_{j=1}^{n} \max(H_i(j) - H_{i+1}(j))}. \tag{8}$$

Note that for two identical histograms the intersection is 1 and the difference 0 while for two frames which do not share even a single pixel of the same color (bin), the difference is 1.

- weighted bin differences

$$D(i, i+1) = \sum_{j=1}^{n} \sum_{k \in N(k)} W(k) \cdot (H_i(j) - H_i(k)), \tag{9}$$

where $N(k)$ is a neighborhood of bin j and $W(k)$ is the weight value assigned to that neighbor. A 3×3 or 3 neighborhoods are used in the case of 2-dimensional and 1-dimensional histograms, respectively.

It is found that in terms of overall classification accuracy *YIQ*, *L*a*b** and Munsell color coordinate spaces perform well, followed by HSV, *L*u*v** and *RGB*. In terms of computational cost of conversion from *RGB*, the *HSV* and *YIQ* are the least expensive, followed by *L*a*b**, *L*u*v** and the Munsell space.

So far only histogram comparison techniques for cut detection have been presented. They are based on the fact that there is a big difference between the

frames across a cut that results in a high peak in the histogram comparison and can be easily detected using one threshold. However, such one-threshold based approaches are not suitable to detect gradual transitions. Although during a gradual transition the frame to frame differences are usually higher than those within a shot, they are much smaller than the differences in the case of cut and cannot be detected with the same threshold. On the other hand, object and camera motions might entail bigger differences than the gradual transition. Hence, lowering the threshold will increase the number of false positives. Below we review a simple and effective two-thresholds technique for gradual transition recognition.

The *twin-comparison* method [45] takes into account the cumulative differences between frames of the gradual transition. In the first pass a high threshold T_h is used to detect cuts as shown in Fig. 6(a). In the second pass a lower threshold T_l is employed to detect the potential starting frame F_s of a gradual transition. F_s is then compared to subsequent frames (Fig. 6(b)). This is called an accumulated comparison as during a gradual transition this difference value increases. The end frame F_e of the transition is detected when the difference between consecutive frames decreases to less than T_l, while the accumulated comparison has increased to a value higher than T_h. If the consecutive difference falls below T_l before the accumulated difference exceeds T_h, then the potential start frame F_s is dropped and the search continues for other gradual transitions. It was found, however, that there are some gradual transitions during which the consecutive difference falls below the lower threshold. This problem can be easily solved by setting a tolerance value that allows a certain number of consecutive frames with low difference values before rejecting the transition candidate. As it can be seen, the twin-comparison detects both abrupt and gradual transitions at the same time. Boreczky and Rowe [6] compared several temporal video segmentation techniques on real video sequences and found that twin comparison is a simple algorithm that works very well.

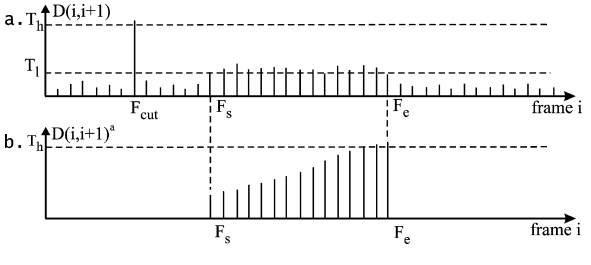

Fig. 6. Twin comparison: (a) consecutive and (b) accumulated histogram differences.

2.1.3.2. Local histogram comparison. As it was already discussed, histogram-based approaches are simple and more robust to object and camera movements but they ignore the spatial information and, therefore, fail when two different images have similar histograms. On the other hand, block-based comparison methods make use of spatial information. They typically perform better than pair-wise pixel comparison but are still sensitive to camera and object motion and are also computationally expensive. By integrating the two paradigms, false alarms due to camera and object movement can be reduced while enough spatial information is retained to produce more accurate results.

The frame-to-frame difference of frame i and frame $i+1$ is computed as

$$D(i, i+1) = \sum_{k=1}^{b} \mathrm{DP}(i, i+1, k),$$
$$\mathrm{DP}(i, i+1, k) = \sum_{j}^{n} |H_i(j, k) - H_{i+1}(j, k)|, \tag{10}$$

where $H_i(j, k)$ denotes the histogram value at gray level j for the region (block) k and b is the total number of the blocks.

For example, Nagasaka and Tanaka [25] compare several statistics based on gray-level and color pixel differences and histogram comparisons. The best results were obtained by breaking the image into 16 equal-sized regions, using χ^2 test on color histograms for these regions and discarding the largest differences to reduce the effects of noise, object and camera movements.

Another approach based on local histogram comparison is proposed by Swanberg et al. [36]. The partial difference $\mathrm{DP}(i, i+1, k)$ is measured by comparing the color *RGB* histograms of the blocks

using the following equation:

$$\mathrm{DP}(i, i+1, k) = \sum_{c\in\{R,B,G\}} \sum_{l=1}^{n} \frac{(H_i^c(l) - H_{i+1}^c(l))^2}{H_i^c(l) - H_{i+1}^c(l)}. \quad (11)$$

Then, Eq. (3) is applied where c_k is $1/b$ for all k. Lee and Ip [22] introduce a *selective HSV histogram* comparison algorithm. In order to reduce the frame-to-frame differences caused by change in intensity or shade, image blocks are compared in *HSV* (hue, saturation, value) color space. It is the use of hue that makes the algorithm insensitive to such changes since hue is independent of saturation and intensity. However, as hue is unstable when the saturation or the value are very low, selective comparison is proposed. If a pixel contains rich color information (i.e. a high V and a high S), it is classified into a discrete color based on its hue (Hue), otherwise on its intensity value (Gray). The selective histograms $H_i^{\mathrm{hue}}(h, k)$, $H_i^{\mathrm{gray}}(g, k)$ and the frame-to-frame difference for the block k with dimensionality $X \times Y$ are formulated as follows:

$$H_i^{\mathrm{hue}}(h, k) = \sum_{x}^{X} \sum_{y}^{Y} I_i^{\mathrm{hue}}(x, y, h),$$

$$H_i^{\mathrm{gray}}(g, k) = \sum_{x}^{X} \sum_{y}^{Y} I_i^{\mathrm{gray}}(x, y, g),$$

$$I_i^{\mathrm{hue}}(x, y, h) = \begin{cases} 1 & \text{if } S_i(x, y, h) > T_s \text{ and } V_i(x, y, h) > T_v, \\ 0 & \text{otherwise,} \end{cases}$$

$$I_i^{\mathrm{gray}}(x, y, g) = \begin{cases} 1 & \text{if } (S_i(x, y, g) \leqslant T_s \text{ or } V_i(x, y, g) \leqslant T_v), \\ 0 & \text{otherwise,} \end{cases}$$

$$D(i, i+1, k) = \sum_{h=1}^{N} |H_i^{\mathrm{hue}}(h, k) - H_{i+1}^{\mathrm{hue}}(h, k)| + \sum_{g=1}^{M} |H_i^{\mathrm{gray}}(g, k) - H_{i+1}^{\mathrm{gray}}(g, k)|, \quad (12)$$

where h and g are indexes for the hue and gray levels, respectively; T_s and T_v are thresholds and x, y are pixel coordinates.

To further improve the algorithm by increasing the differences across a cut, local histogram comparison is performed. It is shown that the algorithm outperforms both histogram (gray level global and local) and pixel differences based approaches. However, none of the algorithms gives satisfactory performance on very dark video images.

2.1.4. Clustering-based temporal video segmentation

The approaches discussed so far rely on suitable thresholding of similarities between successive frames. However, the thresholds are typically highly sensitive to the type of input video. This drawback is overcome in [13] by the application of *unsupervised clustering* algorithm. More specifically, the temporal video segmentation is viewed as a 2-class clustering problem ("scene change" and "no scene change") and the well-known K-means algorithm [27] is used to cluster frame dissimilarities. Then the frames from the cluster "scene change" which are temporary adjacent are labeled as belonging to a gradual transition and the other frames from this cluster are considered as cuts. Two similarity measures based on color histograms were used: χ^2 statistics and the histogram difference defined in Eq. (6), both in *RGB* and *YUV* color spaces. The experiments show that the χ^2-*YUV* detects the larger number of correct transitions but the histogram difference-*YUV* is the best choice in terms of overall performance (i.e. number of false alarms and correct detections). As a limitation we can note that the approach is not able to recognize the type of the gradual transitions. The main advantage of the clustering-based segmentation is that it is a generic techniques that not only eliminates the need for threshold setting but also allows multiple features to be used simultaneously to improve the performance. For example, in their subsequent work Ferman and Tekalp [10] incorporate two features in the clustering method: histogram difference and pair-wise pixel comparison. It was found that when filtered these features supplement one another, which results in both high recall and precision. A technique for clustering-based temporal segmentation on-the-fly was introduced as well.

2.1.5. Feature based temporal video segmentation

An interesting approach for temporal video segmentation based on features is described by Zabih et al. [44]. It involves analyzing intensity edges

between consecutive frames. During a cut or a dissolve, new intensity edges appear far from the locations of the old edges. Similarly, old edges disappear far from the location of new edges. Thus, by counting the entering and exiting edge pixels, cuts, fades and dissolves are detected and classified. To obtain better results in case of object and camera movements, an algorithm for motion compensation is also included. It first estimates the global motion between frames that is then used to align the frames before detecting entering and exiting edge pixels. However, this technique is not able to handle multiple rapidly moving objects. As the authors have pointed out, another weakness of the approach are the false positives due to the limitations of the edge detection method. In particular, rapid changes in the overall shot brightness, and very dark or very light frames, may cause false positives.

2.1.6. Model driven temporal video segmentation

The video segmentation techniques presented so far are sometimes referred to as *data driven*, *bottom–up* approaches [14]. They address the problem from data analysis point of view. It is also possible to apply *top–down* algorithms that are based on mathematical models of video data. Such approaches allow a systematic analysis of the problem and the use of several domain-specific constraints that might improve the efficiency.

Hampapur et al. [14] present a shot boundaries identification approach based on the mathematical model of the video production process. This model was used as a basis for the classification of the video edit types (cuts, fades, dissolves).

For example, fades and dissolves are chromatic edits and can be modeled as

$$S(x, y, t) = S_1(x, y, t)(1 - \tfrac{t}{l_1}) + S_2(x, y, t)(1 - \tfrac{t}{l_2}), \tag{13}$$

where $S_1(x, y, t)$ and $S_2(x, y, t)$ are two shots that are being edited, $S(x, y, t)$ is the edited shot and l_1, l_2 are the number of frames for each shot during the edit.

The taxonomy along with the models are then used to identify features that correspond to the different classes of shot boundaries. Finally, feature vectors are fed into a system for frames classification and temporal video segmentation. The approach is sensitive to camera and object motion.

Another model-based technique, called differential model of motion picture, is proposed by Aigrain and Joly [1]. It is based on the probabilistic distribution of differences in pixel values between two successive frames and combines the following factors: (1) a small amplitude additive zero-centered Gaussian noise that models camera, film, digitizer and other noises; (2) an intrashot change model for pixel change probability distribution resulting from object and camera motion, angle, focus and light change; (3) a shot transition model for the different types of abrupt and gradual transitions. The histogram of absolute values of pixel differences is computed and the number of pixels that change in value within a certain range determined by the models is counted. Then shot transitions are detected by examining the resulting integer sequences. Experiments show 94–100% accuracy for cuts and 80% for gradual transitions detection.

Yu et al. [43] present an approach for gradual transitions detection based on a model of intensity changes during fade out, fade in and dissolve. At the first pass, cuts are detected using histogram comparison. The gradual transitions are then detected by examining the frames between the cuts using the proposed model of their characteristics. For example, it was found that the number of edge pixels have a local minimum during a gradual transition. However, as this feature exhibits the same behavior in case of zoom and pan, additional characteristics of the fades and dissolves need to be used for their detection. During a fade, the beginning and end image is a constant image. Hence the number of edge pixels will be close to zero. Furthermore, the number of edge pixels gradually increases going away from the minimum in either side. In order to distinguish dissolves, the so called double chromatic difference curve is examined. It is based on the idea that the frames of a dissolve can be recovered using the beginning and end frames. The approach has low computational requirements but works under the assumption of small object movement.

Boreczky and Wilcox [7] use hidden Markov models (HMM) for temporal video segmentation.

Separate states are used to model shot, cut, fade, dissolve, pan and zoom. The arcs between states model the allowable progressions of states. For example, from the shot state it is possible to go to any of the transition states, but from a transition state it is only possible to return to a shot state. Similarly, the pan and zoom states can only be reached from the shot state, since they are subsets of the shot. The arcs from a state to itself model the length of time the video is in that particular state. Three different types of features (image, audio and motion) are used: (1) a standard gray-level histogram distance between two adjacent frames; (2) an audio distance based on the acoustic difference in intervals just before and just after the frames and (3) an estimate of object motion between the two frames. The parameters of the HMM, namely the transition probabilities associated with the arcs and the probability distributions of the features associated with the states, are learned by training with the Baum–Welch algorithm. Training data consists of features vectors computed for a collection of video and labeled as one of the following classes: shot, cut, fade, dissolve, pan and zoom. Once the parameters are trained, segmenting the video is performed using the Viterbi algorithm, a standard technique for recognition in HMM.

Thus, thresholds are not required as the parameters are learned automatically. Another advantage of the approach is that HMM framework allows any number of features to be included in a feature vector. The algorithm was tested on different video databases and has been shown to improve the accuracy of the temporal video segmentation in comparison to the standard threshold-based approaches.

2.2. Temporal video segmentation in MPEG compressed domain

The previous approaches for video segmentation process uncompressed video. As nowadays video is increasingly stored and moved in compressed format (e.g. MPEG), it is highly desirable to develop methods that can operate directly on the encoded stream. Working in the compressed domain offers the following advantages. First, by not having to perform decoding/re-encoding, computational complexity is reduced and savings on decompression time and decompression storage are obtained. Second, operations are faster due to the lower data rate of compressed video. Last but not least, the encoded video stream already contains a rich set of pre-computed features, such as motion vectors (MVs) and block averages, that are suitable for temporal video segmentation.

Table 1
Six groups of approaches for temporal video segmentation in compressed domain based on the information used

	Group					
Information used	1	2	3	4	5	6
DCT coefficients	✓			✓		
DC terms		✓	✓			
MB coding mode			✓	✓	✓	✓
MVs			✓	✓	✓	
Bit-rate						✓

Several algorithms for temporal video segmentation in the compressed domain have been reported. According to the type of information used (see Table 1), they can be divided into six non-overlapping groups – segmentation based on (1) DCT coefficients; (2) DC terms; (3) DC terms, macroblock (MB) coding mode and MVs; (4) DCT coefficients, MB coding mode and MVs; (5) MB coding mode and MVs and (6) MB coding mode and bit-rate information. Before reviewing each of them, we present a brief description of the fundamentals of MPEG compression standard.

2.2.1. MPEG stream

The Moving Picture Expert Group (MPEG) standard is the most widely accepted international standard for digital video compression. It uses two basic techniques: MB-based motion compensation to reduce temporal redundancy and transform domain block-based compression to capture spatial redundancy. An MPEG stream consists of three types of pictures – I, P and B – which are combined in a repetitive pattern called group of picture (GOP). Fig. 7 shows a typical GOP and the predictive relationships between the different types of frames.

Intra (I) frames provide random access points into the compressed data and are coded using only information present in the picture itself by Discrete Cosine Transform (DCT), Quantization (Q), Run Length Encoding (RLE), and Huffman entropy coding, see Fig. 8. The first DCT coefficient is called DC term and is 8 times the average intensity of the respective block.

P (predicted) frames are coded with *forward* motion compensation using the nearest previous reference (*I* or *P*) pictures. *Bi-directional (B)* pictures are also motion compensated, this time with respect to both past and future reference frames. In the case of motion compensation, for each 16×16 MB of the current frame the encoder finds the best matching block in the respective reference frame(s), calculates and DCT-encodes the residual error and also transmits one or two MVs, see Figs. 9 and 10. During the encoding process a test is made on each MB of *P* and *B* frame to see if it is more expensive to use motion compensation or intra coding. The latter occurs when the current frame does not have much in common with the reference frame(s). As a result each MB of a *P* frame could be coded either intra or forward while for each MB of a *B* frame there are four possibilities: intra, forward, backward or interpolated. For more information about MPEG see [16].

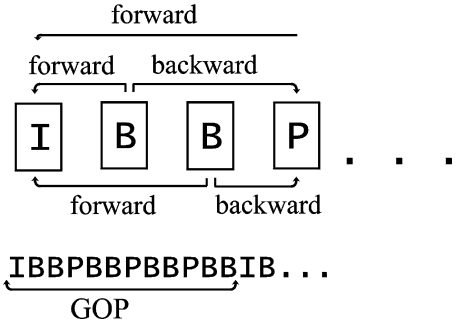

Fig. 7. Typical GOP and predictive relationships between *I*, *P* and *B* pictures.

2.2.2. Temporal video segmentation based on DCT coefficients

The pioneering work on video parsing directly in compressed domain is conducted by Arman et al. [5] who proposed a technique for cut detection based on the DCT coefficients of *I* frames. For each frame a subset of the DCT coefficients of a subset of the blocks is selected in order to construct a vector $V_i = \{c_1, c_2, c_3, \ldots\}$. V_i represents the frame i from the video sequence in the DCT space. The normalized inner product is then used to find the difference between frames i and $i + \varphi$,

$$D(i, i + \varphi) = \frac{V_i \cdot V_{i+\varphi}}{|V_i||V_{i+\varphi}|}. \tag{14}$$

A cut is detected if $1 - |D(i, i + \varphi)| > T_1$, where T_1 is a threshold. In order to reduce false positives due to camera and object motion, video cuts are examined more closely using a second threshold T_2 $(0 < T_1 < T_2 < 1)$. If $T_1 < 1 - |D(i, i + \varphi)| < T_2$, the two frames are decompressed and examined by comparing their color histograms.

Zhang et al. [46] apply a pair-wise comparison technique to the DCT coefficients of corresponding blocks of video frames. The difference metric is similar to pixel comparisons [25,45], see Section 2.1.1. More specifically, the difference of

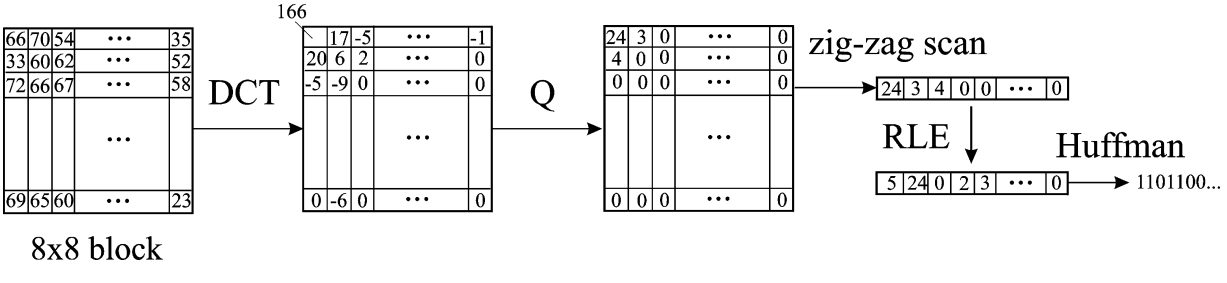

Fig. 8. Intra coding.

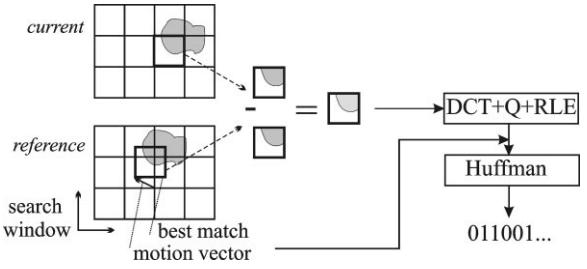

Fig. 9. Forward prediction for P frames.

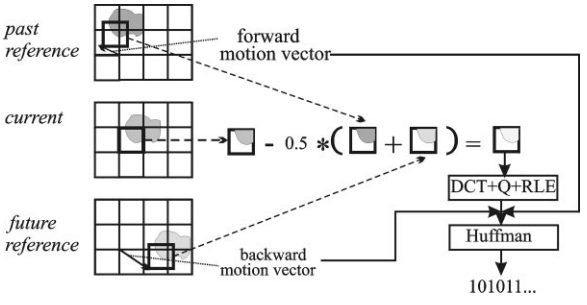

Fig. 10. Interpolated prediction for B frames.

block l from two frames which are φ frames apart is given by

$$\mathrm{DP}(i, i+\varphi, l) = \frac{1}{64}\sum_{k=1}^{64}\frac{|c_{l,k}(i) - c_{l,k}(i+\varphi)|}{\max[c_{l,k}(i), c_{l,k}(i+\varphi)]} > T_1, \qquad (15)$$

where $c_{l,k}(i)$ is the DCT coefficient of block l in the frame i, $k = 1, \ldots, 64$ and l depends on the size of the frame.

If the difference exceeds a given threshold T_1, the block l is considered to be changed. If the number of changed blocks is larger than another threshold T_2, a transition between the two frames is declared. The pair-wise comparison requires far less computation than the difference metric used by Arman. The processing time can be further reduced by applying Arman's method of using only a subset of coefficients and blocks.

It should be noted that both of the above algorithms may be applied only to I frames of the MPEG compressed video, as they are the frames fully encoded with DCT coefficients. As a result, the processing time is significantly reduced but the temporal resolution is low. In addition, due to the loss of the resolution between the I frames, false positives are introduced and, hence, the classification accuracy decreases. Also, neither of the two algorithms can handle gradual transitions or false positives introduced by camera operations and object motion.

2.2.3. Temporal video segmentation based on DC terms

For temporal video segmentation in MPEG compressed domain the most natural solution is to use the DC terms as they are directly related to the pixel domain, possibly reconstructing them for P and B frames, when only DC terms of the residual errors are available. Then, analogous to the uncompressed domain methods, the changes between successive frames are evaluated by difference metrics and the decision is taken by complex thresholding.

For example, Yeo and Liu [42] propose a method where so called DC-images are created and compared. DC-images are spatially reduced versions of the original images: the (i, j) pixel of the DC-image is the average value of the (i, j) block of the image (Fig. 11).

As each DC term is a scaled version of the block's average value, DC-images can be constructed from DC terms. The DC terms of I frames are directly available in the MPEG stream while those of B and P frames are estimated using the MVs and DCT coefficients of previous I frames. It should be noted that the reconstruction techniques is computationally very expensive – in order to compute the DC term of a reference frame ($\mathrm{DC}_{\mathrm{ref}}$) for each block, eight 8×8 matrix multiplications and 4 matrix summations are required. Then, the pixel differences of DC-images are compared and a sliding window is used to set the thresholds because the shot transition is a local activity.

In order to find a suitable similarity measure, the authors compare metrics based on pixel differences and color histograms. They confirm that when full images are compared, the first group of metrics is more sensitive to camera and object movements but computationally less expensive than the second one. However, when DC-images are compared, pixel-differences-based metrics give satisfactory results as DC-images are already smoothed versions of the corresponding full images. Hence, as in the

Fig. 11. A full image (352 × 288 pixels) and its DC image (44 × 36 pixels).

pixel domain approaches (e.g. Eq. (1)), abrupt transitions are detected using a similarity measure based on the sum of absolute pixel differences of two consecutive frames (DC-images in this case):

$$D(l, l+1) = \sum_{i,\,j} (|P_l(i,j) - P_{l+1}(i,j)|), \tag{16}$$

where l and $l+1$ are two consecutive DC-images and $P_l(i,j)$ is the intensity value of the pixel in lth DC-image at the coordinates (i,j).

In contrast to the previous methods for cut detection that apply global thresholds on the difference metrics, Yeo and Liu propose to use local thresholds as scene changes are local activities in the temporal domain. In this way false positives due to significant camera and object motions are reduced. More specifically, a *sliding window* is used to examine m successive frame differences. A cut between frames l and $l+1$ is declared if the following two conditions are satisfied: (1) $D(l, l+1)$ is the maximum within a symmetric sliding window of size $2m-1$ and (2) $D(l, l+1)$ is n times the second largest maximum in the window. The second condition guards against false positives due to fast panning or zooming and camera flashes that typically manifest themselves as sequences of large differences or two consecutive peaks, respectively. The size of the sliding window m is set to be smaller than the minimum duration between two transitions, while the values of n typically range from 2 to 3.

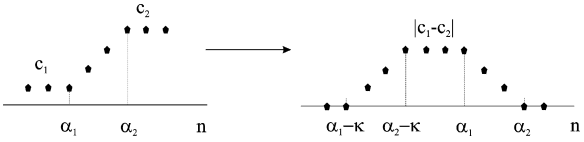

Fig. 12. g_n and $D_{g_n}(l, l+k)$ in the dissolve detection algorithm of Yeo and Liu.

Gradual transitions are detected by comparing each frame with the following kth frame where k is larger than the number of frames in the gradual transition. A gradual transition g_n in the form of linear transition from c_1 to c_2 in the time interval (α_1, α_2) is modeled as

$$g_n = \begin{cases} c_1, & n < \alpha_1, \\ \dfrac{c_2 - c_1}{\alpha_2 - \alpha_1}(n - \alpha_2) + c_2, & \alpha_1 \leqslant n < \alpha_2, \\ c_2, & n \geqslant \alpha_2. \end{cases} \tag{17}$$

Then if $k > \alpha_2 - \alpha_1$, the difference between frames l and $l+k$ from the transition g_n will be

$$D_{g_n}(l, l+k) = \begin{cases} 0, & n < \alpha_1 - k, \\ \dfrac{|c_2 - c_1|}{|\alpha_2 - \alpha_1|}[n - (\alpha_1 - k)], & \alpha_1 - k \leqslant n < \alpha_2 - k, \\ |c_2 - c_1|, & \alpha_2 - k \leqslant n < \alpha_1, \\ -\dfrac{|c_2 - c_1|}{|\alpha_2 - \alpha_1|}(n - \alpha_2), & \alpha_1 \leqslant n < \alpha_2, \\ 0, & n \geqslant \alpha_2. \end{cases} \tag{18}$$

As $D_{g_n}(l, l+k)$ corresponds to a symmetric plateau with sloping sides (see Fig. 12), the goal of the gradual transition detection algorithm is to identify such plateau patterns. The algorithm of Yeo and Liu needs 11 parameters to be specified.

In [33] shots are detected by color histogram comparison of DC term images of consecutive frames. Such images are formed by the DC terms of the DCT coefficients for a frame. DC terms of I pictures are taken directly from the MPEG stream, while those for P and B frames are reconstructed by the following fast algorithm. First, the DC term of the reference image (DC_{ref}) is approximated using the weighted average of the DC terms

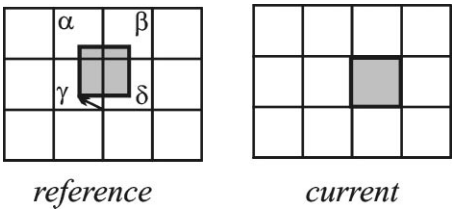

Fig. 13. DC term estimation in the method of Shen and Delp.

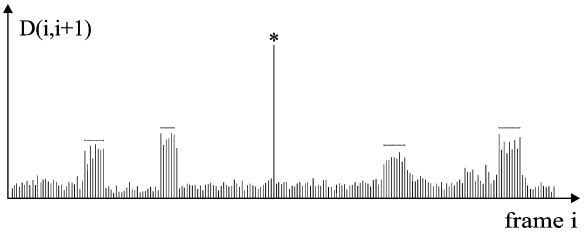

Fig. 14. Histogram difference diagram ((∗) cut; (- - - -) dissolve).

of the blocks pointed by the MVs, Fig. 13:

$$\mathrm{DC}_{\mathrm{ref}} = \frac{1}{64} \sum_{\alpha \in E} N_\alpha \, \mathrm{DC}_\alpha, \tag{19}$$

where DC_α is the DC term of block α, E is the collection of all blocks that are overlapped by the reference block and N_α is the number of pixels in block α that is overlapped by the reference block.

Then, the approximated DC terms of the predicted pictures are added to the encoded DC terms of the difference images in order to form the DC terms of P and B pictures,

$$\mathrm{DC} = \mathrm{DC}_{\mathrm{diff}} + \mathrm{DC}_{\mathrm{ref}}$$

(only forward or only backward prediction),

$$\mathrm{DC} = \mathrm{DC}_{\mathrm{diff}} + \tfrac{1}{2}(\mathrm{DC}_{\mathrm{ref1}} + \mathrm{DC}_{\mathrm{ref2}})$$

(interpolated prediction). (20)

In this way the computations are reduced to at most 4 scalar multiplications and 3 scalar summations for each block to determine $\mathrm{DC}_{\mathrm{ref}}$.

The histogram difference diagram is generated using the measure from Eq. (6) comparing DC term images. As it can be seen from Fig. 14, a break is represented by a single sharp pulse and a dissolve entails a number of consecutive medium-heighted pulses. Cuts are detected using a static threshold. For the recognition of gradual transitions, the histogram difference of the current frame is compared with the average of the histogram differences of the previous frames within a window. If this difference is n times larger than the average value, a possible start of a gradual transition is marked. The same value of n is used as a soft threshold for the following frames. End of the transition is declared when the histogram difference is lower than the threshold. Since during a gradual transition not all of the histogram differences may be higher than the soft threshold, similarly to the twin comparison, several frames are allowed to have lower difference as long as the majority of the frames in the transition have higher magnitude than the soft threshold.

As only the DC terms are used, the computation of the histograms is 64 times faster than that using the original pixel values. The approach is not able to distinguish rapid object movement from gradual transition. As a partial solution, a median filter (of size 3) is applied to smooth the histogram differences when detecting gradual transitions. There are 7 parameters that need to be specified.

An interesting extension of the previous approach is proposed by Taskiran and Delp [37]. After the DC term image sequence and the luminance histogram for each image are obtained, a two-dimensional feature vector is extracted from each pair of images. The first component is the dissimilarity measure based on the histogram intersection of the consecutive DC term images,

$$x_{1i} = 1 - \mathrm{Intersection}(H_i, H_{i+1}) = \frac{\sum_{j=1}^{n} \min(H_i(j), H_{i+1}(j))}{\sum_{j=1}^{n} H_{i+1}(j)}, \tag{21}$$

where $H_i(j)$ is the luminance histogram value for the bin j in frame i and n is the number of bins used. Note that the definition of the histogram intersection is slightly different from that used in [12, Section 2.1.3.1].

The second feature is the absolute value of the difference of standard deviations σ for the luminance component of the DC term images, i.e. $x_{i2} = |\sigma_i - \sigma_{i+1}|$. The so called *generalized sequence trace d* for a video stream composed of n frames is defined as $d_i = \|x_i - x_{i+1}\|, i = 1, \ldots, n$.

These features are chosen not only because they are easy to extract. Combining histogram-based and pixel-based parameters makes sense as they complement some of their disadvantages. As it was discussed already, pixel-based techniques give false alarms in case of camera and object movements. On the other hand, histogram-based techniques are less sensitive to these effects but may miss shot transition if the luminance distribution of the frames do not change significantly. It is shown that there are different types of peaks in the generalized trace plot: wide, narrow and middle corresponding to a fade out followed by a fade in, cuts and dissolves, respectively. Then, in contrast to the other approaches that apply global or local thresholds to detect the shot boundaries, Taskiran and Delp pose the problem as a one-dimensional edge detection and apply a method based on mathematical morphology.

Patel and Sethi [29,30] use only the DC components of I frames. In [30] they compute the intensity histogram for the DC term images and compare them using three different statistics: Yakimovski likelihood ratio, χ^2 test and Kolmogorov–Smirnov statistics. The experiments show that χ^2 test gives satisfactory results and outperforms the other techniques. In their consequent paper [29], Patel and Sethi compare local and global histograms of consecutive DC term images using χ^2 test, Fig. 15.

The local row and column histograms X_i and Y_j are defined as follows:

$$X_i = \frac{1}{M}\sum_{j=1}^{M} b_{0,0}(i,j), \quad Y_j = \frac{1}{N}\sum_{j=1}^{N} b_{0,0}(i,j), \qquad (22)$$

where $b_{0,0}(i,j)$ is the DC term of block (i,j), $i = 1,\ldots,N, j = 1,\ldots,M$. The outputs of the χ^2 test are combined using majority and average comparison in order to detect abrupt and gradual transitions.

As only I frames are used, the DC recovering is eliminated. However, the temporal resolution is low as in a typical GOP every 12th frame is an I frame and, hence, the exact shot boundaries cannot be labeled.

2.2.4. Temporal video segmentation based on DC terms and MB coding mode

Meng et al. [24] propose a shot boundaries detection algorithm based on the DC terms and the type of MB coding, Fig. 16. DC components only for P frames are reconstructed. Gradual transitions are detected by calculating the variance σ^2 of the DC term sequence for I and P frames and looking for parabolic shapes in this curve. This is based on the fact that if gradual transitions are linear mixture of two video sequences f_1 and f_2 with intensity variances σ_1 and σ_2, respectively, and are characterized by $f(t) = f_1(t)[1-\alpha(t)] + f_2(t)\alpha(t)$ where $\alpha(t)$ is a linear parameter, then the shape of the variance curve is parabolic: $\sigma^2(t) = (\sigma_1^2 + \sigma_2^2)\alpha(t) - 2\sigma_1^2\alpha(t) + \sigma_1^2$. Cuts are detected by the computation of the following three ratios:

$$R_p = \frac{\text{intra}}{\text{forw}}, \quad R_b = \frac{\text{back}}{\text{forw}}, \quad R_f = \frac{\text{forw}}{\text{back}}, \qquad (23)$$

where intra, forw and back are the number of MBs in the current frame that are intra, forward and backward coded, respectively.

If there is a cut on a P frame, the encoder cannot use many MBs from the previous anchor frame for motion compensation and as a result many MBs will be coded intra. Hence, a suspected cut on P frame is declared if R_p peaks. On the other hand,

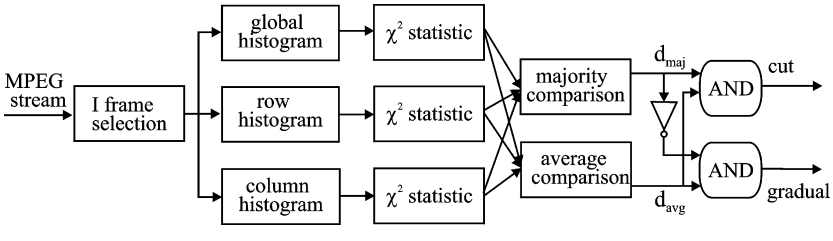

Fig. 15. Video shot detection scheme of Patel and Sethi.

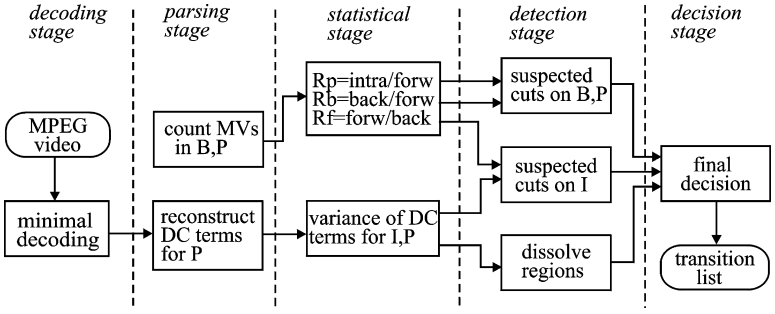

Fig. 16. Shot detection algorithm of Meng et al.

if there is a cut on a B frame, the encoding will be mainly backward. Therefore, a suspected cut on B frame is declared if there is a peak in R_b. An I frame is a suspected cut frame if two conditions are satisfied: (1) there is a peak in $|\Delta\sigma^2|$ for this frame and (2) the B frames before I have peaks in R_f. The first condition is based on the observation that the intensity variance of the frames during a shot is stable, while the second condition prevents false positives due to motion.

This technique is relatively simple, requires minimum encoding and produces good accuracy. The total number of parameters needed to implement this algorithm is 7.

2.2.5. Temporal video segmentation based on DCT coefficients, MB coding mode and MVs

A very interesting two-pass approach is taken by Zhang et al. [47]. They first locate the regions of potential transitions, camera operations and object motion, applying the pair-wise DCT coefficients comparison of I frames (Eq. (15)) as in their previous approach (see Section 2.2.2). The goal of the second pass is to refine and confirm the break points detected by the first pass. By checking the number of MVs M for the selected areas, the exact cut locations are detected. If M denotes the number of MVs in P frames and the smaller of the numbers of forward and backward nonzero MVs in B frames, then $M < T$ (where T is a threshold close to zero) is an effective indicator of a cut before or after the B and P frame. Gradual transitions are found by an adaptation of the twin comparison algorithm utilizing the DCT differences of I frames. By MV analysis (see Section 3.1 for more details), though using thresholds, false positives due to pan and zoom are detected and discriminated from gradual transitions.

Thus, the algorithm only uses information directly available in the MPEG stream. It offers high processing speed due to the multipass strategy, good accuracy and also detects false positives due to pan and zoom. However, the metric for cut detection yields false positives in the case of static frames. Also, the problem of how to distinguish object movements from gradual transitions is not addressed.

2.2.6. Temporal video segmentation based on MB coding mode and MVs

In [21] cuts, fades and dissolves are detected only using MVs from P and B frames and information about MB coding mode. The system follows a two-pass scheme and has a hybrid rule-based/neural structure. During the rough scan peaks in the number of intra coded MBs in P frames are detected. They can be sharp (Fig. 17) or gradual with specific shape (Fig. 18) and are good indicators of abrupt and gradual transitions, respectively.

The solution is then refined by a precise scan over the frames of the respective neighborhoods. The "simpler" boundaries (cuts and black fade edges) are recognized by the rule-based module, while the decisions for the "complex" ones (dissolves and non-black fade edges) are taken by the neural part. The precise scan also reveals cuts that remain hidden for the rough scan, e.g. B_{24}, I_{49}, B_{71} and B_{96} in Fig. 17. The rules for the exact cut location are based on the number of backward and forward MBs while those for the fades black edges detection use the number of interpolated and

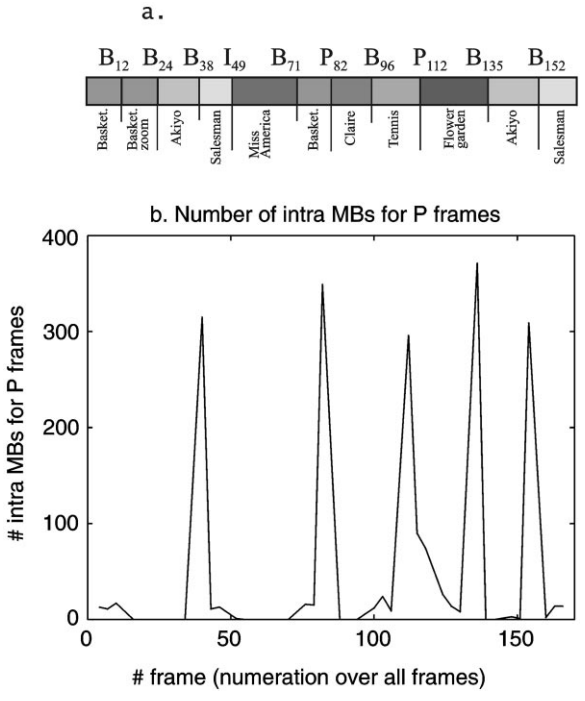

Fig. 17. Cuts: (a) video structure, (b) number of intra-coded MBs for *P* frames.

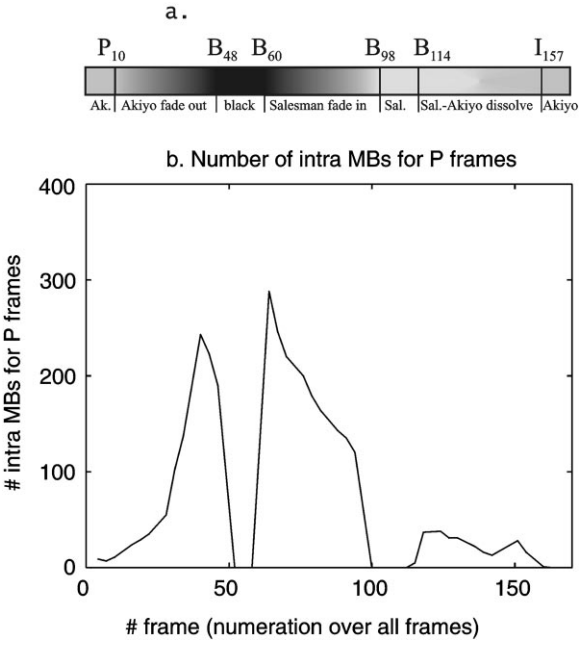

Fig. 18. Fade out, fade in, dissolve: (a) video structure, (b) number of intra-coded MBs for *P* frames.

backward coded MBs. There is only one threshold in the rules that is easy to set and not sensitive to the type of video. The neural network module learns from pre-classified examples in the form of MV patterns corresponding to the following 6 classes: stationary, pan, zoom, object motion, tracking and dissolve. It is used to distinguish dissolves from object and camera movements, find the exact location of the "complex" boundaries of the gradual transition and further divide shots into sub-shots. For more details about the neural network see Section 3.3.

The approach is simple, fast, robust to camera operations and very accurate when detecting the exact locations of cuts, fades and simple dissolves. However, sometimes dissolves between busy sequences are recognized as object movement or their boundaries are not exactly determined.

2.2.7. Temporal video segmentation based on MB coding mode and bit-rate information

Although limited only to cut detection, a simple and effective approach is proposed in [9]. It only uses the bit-rate information at MB level and the number of various motion predicted MBs. A large change in bit-rate between two consecutive *I* or *P* frames indicates a cut between them. Similarly to [24], the number of backward predicted MBs is used for detecting cuts on *B* frames. Here, the ratio is calculated as $R_b = \text{back/mc}$, where back and mc are the number of backward and all motion compensated MBs in a *B* frame, respectively. The algorithm is able to locate the exact cut locations. It operates hierarchically by first locating a suspected cut between two *I* frames, then between the *P* frames of the GOP and finally (if necessary) by checking the *B* frames.

2.2.8. Comparison of algorithms for temporal video segmentation in compressed domain

In [11] the approaches of Arman et al. [5], Patel and Sethi [29], Meng et al. [24], Yeo and Liu [42] and Shen and Delp [33] are compared along several parameters: classification performance (recall and precision), full data use, ease of implementation, source effects. Ten MPEG video sequences containing more than 30 000 frames connected

with 172 cuts and 38 gradual transitions are used as an evaluation database. It is found that the algorithm of Yeo and Liu and those of Shen and Delp perform best when detecting cuts. Although none of the approaches recognizes gradual transitions particularly well, the best performance is achieved by the last one. As the authors point out, the reason for the poor gradual transition detection is that the algorithms expect some sort of ideal curve (a plateau or a parabola) but the actual frame differences are noisy and either do not follow this ideal pattern or do not do this smoothly for the entire transition. Another interesting conclusion is that not processing of all frame types (e.g., like in the first two methods) does decrease performance significantly. The algorithm of Yeo and Liu is found to be easiest for implementation as it specifies the parameter values and even some performance analysis is already carried out by the authors. The dependence of the two best performing algorithms on bit-rate variations is investigated and shown that they are robust to bit-rate changes except at very low rates. Finally, the dependence of the algorithm of Yeo and Liu on two different software encoder implementations is studied and significant performance differences are reported.

3. Camera operation recognition

As stated previously, at the stage of temporal video segmentation gradual transitions have to be distinguished from false positives due to camera motion. In the context of content-based retrieval systems, camera operation recognition is also important for key frame selection, index extraction, construction of salient video stills and search narrowing. Historically, motion estimation were extensively studied in the field of computer vision and image coding and used for tracking of moving objects, recovering of object movement, and motion compensation coding. Below we review approaches for camera operation recognition related to shot partitioning and characterization.

3.1. Analysis of MVs

Camera movements exhibit specific patterns in the field of MVs, as shown in Fig. 19. Therefore, many approaches for camera operation recognition are based on the analysis of MV fields.

Zhang et al. [46] apply rules to detect pan/tilt and zoom in/zoom out. During a pan most of the MVs will be parallel to a modal vector that corresponds to the movement of the camera. This is expressed by the following inequality:

$$\sum_{b=1}^{N} |\theta_b - \theta_m| \leqslant T, \tag{24}$$

where θ_b is the direction of the MV for block b, θ_m is the direction of the modal vector, N is the total number of blocks into which the frame is partitioned and T is a threshold near zero.

In the case of zooming, the field of MVs has focus of expansion (zoom in) or focus of contraction (zoom out). Zooming is determined on the basis of "peripherial vision", i.e. by comparing the vertical components v_k of the MVs for the top and bottom rows of a frame, since during a zoom they have opposite signs. In addition, the horizontal components u_k of the MVs for the left-most and right-most columns are analyzed in the same way. Mathematically these two conditions can be expressed in

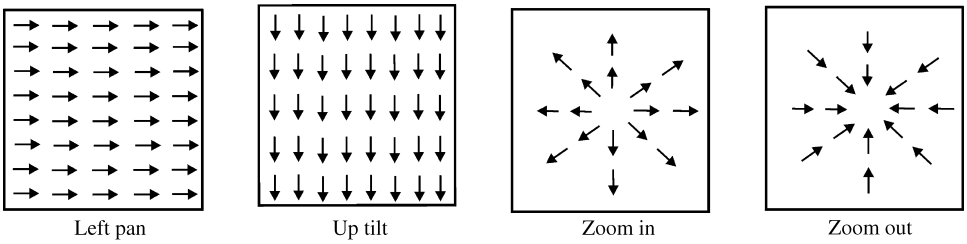

Fig. 19. MV patterns resulting from various camera operations.

the following way:

$$|v_k^{top} - v_k^{bottom}| \geqslant \max(|v_k^{top}|, |v_k^{bottom}|),$$
$$|u_k^{left} - u_k^{right}| \geqslant \max(|u_k^{left}|, |u_k^{right}|). \quad (25)$$

When both conditions are satisfied, a zoom is declared.

3.2. Hough transform

Akutsu et al. [3] characterize the MV patterns corresponding to the different types of camera operations by two parameters: (1) the magnitude of MVs and (2) the divergence/convergence point, see Table 2.

The algorithm has three stages. During the first one, a block matching algorithm is applied to determine the MVs between successive frames. Then, the spatial and temporal characteristics of MVs are determined. MVs are mapped to a polar coordinate space by the Hough transform. A Hough transform of a line is a point. A group of lines with point of convergence/divergence (x_0, y_0) is represented by a curve $\rho = x_0 \sin \varphi + y_0 \cos \varphi$ in the Hough space. The least-squares method is used to fit the transformed MVs to the curve represented by the above equation. There are specific curves that correspond to the different camera operations, e.g. zoom is characterized by a sinusoidal pattern, pan by a straight line. During the third stage these patterns are recognized and the respective camera operations are identified. The approach is effective but also noise sensitive and with high computational complexity.

Table 2
MV patterns characterization

Camera operation	MV origin	MV magnitude
Still	No	Zero
Panning	Infinity	Constant
Tracking	Infinity	Changeable
Tilting	Infinity	Constant
Booming	Infinity	Changeable
Zooming	Image center	Constant
Dollying	Image center	Changeable

3.3. Supervised learning by examples

An alternative approach for detecting camera operations is proposed by Patel and Sethi [29]. They apply induction of decision trees (DTs) [31] to distinguish among the MV patterns of the following six classes: stationary, object motion, pan, zoom, track and ambiguous. DTs are simple, popular and highly developed technique for supervised learning. In each internal node a test of a single feature leads to the path down the tree towards a leaf containing a class label, see Fig. 20. To build a decision tree, a recursive splitting procedure is applied to the set of training examples so that the classification error is reduced. To classify an example that has not been seen during the learning phase, the system starts at the root of the tree and propagates the example down the leaves.

After the extraction of the MVs from the MPEG stream, Patel and Sethi generate a 10-dimensional feature vector for each P frame. Its first component is the fraction of zero MVs and the remaining components are obtained by averaging the column projection of MV directions. In order to develop a decision tree classifier, the MV patterns of 1320 frames have been manually labeled. The results have shown high classification accuracy at a low computational price. We note that as only MVs of P frames are used, the classification resolution is low. In addition, there are problems with the calculation of the MV direction due to the discontinuity at $0/360^\circ$.

The above limitations are addressed in [21] where a neural supervised algorithm is applied. Given a set of pre-classified feature vectors (training examples), Learning Vector Quantization (LVQ) [19] creates a few prototypes for each class, adjusts their positions by learning and then

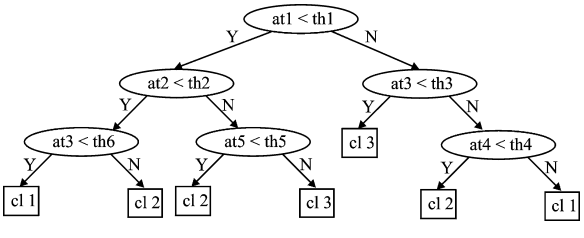

Fig. 20. Decision tree.

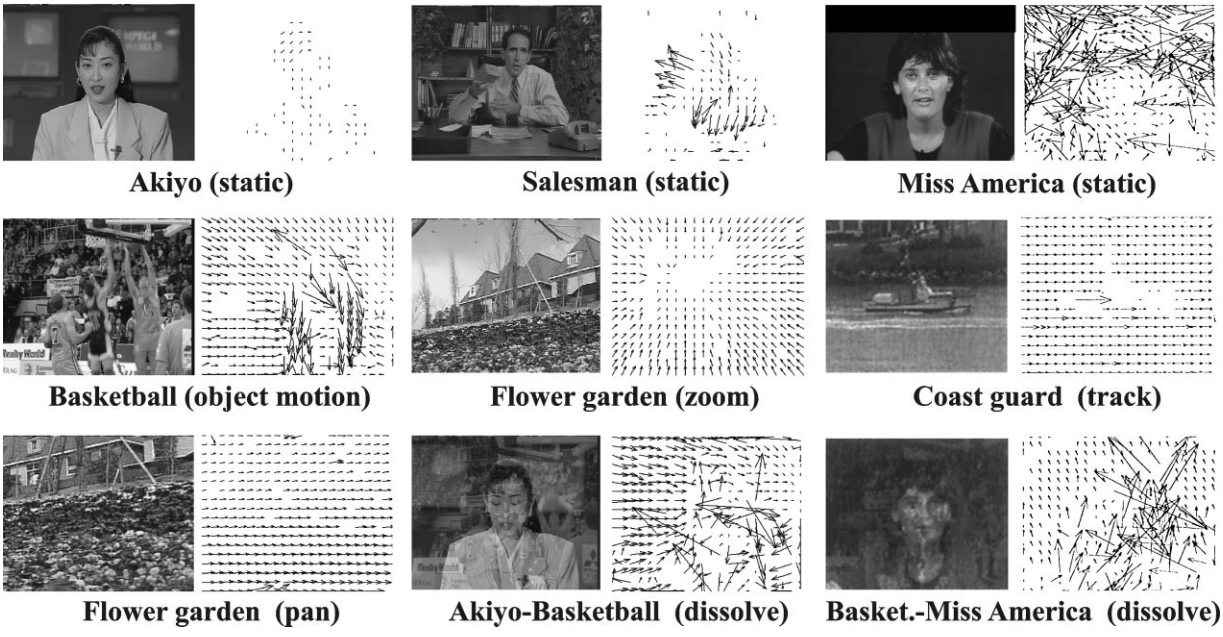

Fig. 21. MV patterns corresponding to different classes.

classifies the unseen examples by means of the nearest-neighbor principle. While LVQ can form arbitrary borders, DTs delineate the concept by a set of axis-parallel hyperplanes which constrains their accuracy in realistic domains. In comparison to the approach of Patel and Sethi, one more class (dissolve) is added (Fig. 21) and the MVs from both P and B frames are used to generate a 22-dimensional feature vector for each frame. The first component is calculated using the number of zero MVs in forward, backward and interpolated areas. Then, the forward MV pattern is sub-divided in 7 vertical strips for which the following 3 parameters are computed: the average of the MV direction, the standard deviation of the MV direction and the average of MV modulus. A technique that deals with the discontinuity of angles at $0/360^\circ$ is proposed for the calculation of the MV direction. Although high classification accuracy is reported, it was found that the most difficult case is to distinguish dissolve from object motion. In [20] MV patterns are classified by an integration between DTs and LVQ. More specifically, DTs are viewed as a feature selection mechanism and only those parameters that appear in the tree are considered as informative and used as inputs in LVQ. The result is faster learning at the cost of a slightly worse classification accuracy.

3.4. Spatiotemporal analysis

Another way to detect camera operations is to examine the so called spatiotemporal image sequence. The latter is constructed by arranging each frame close to the other and forming a parallelepiped where the first two dimensions are determined by the frame size and the third one is the time. Camera operations are recognized by texture analysis of the different faces.

In [2] video X-ray images are created from the spatiotemporal image sequence, as shown in Fig. 22. Sliced x–t and y–t images are first extracted from the spatiotemporal sequence and are then subject to an edge detection. The process is repeated for all x and y values, the slices are summed in the vertical and horizontal directions to produce gray-scale x–t and y–t video X-ray images. There are typical X-ray images corresponding to the following camera operations: still, pan, tilt and zoom. For example, when the camera is still, the video X-ray show lines parallel

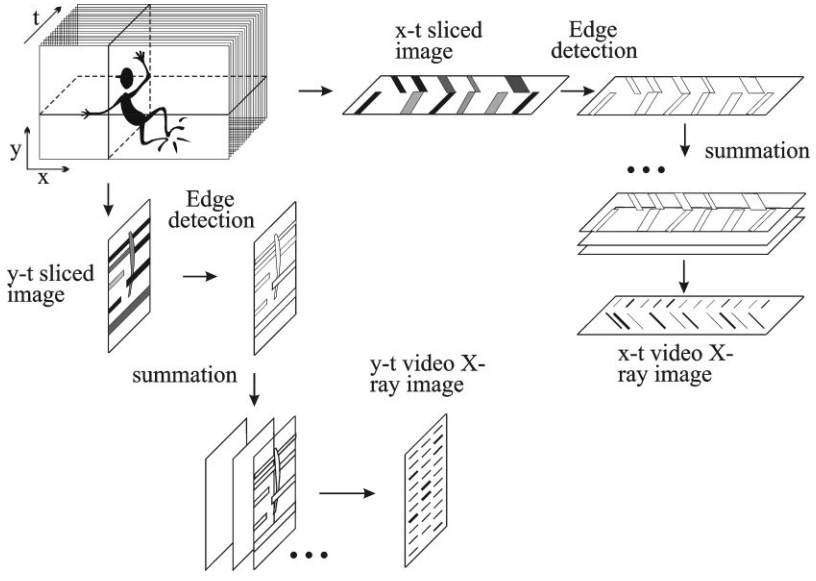

Fig. 22. Creating video X-ray image.

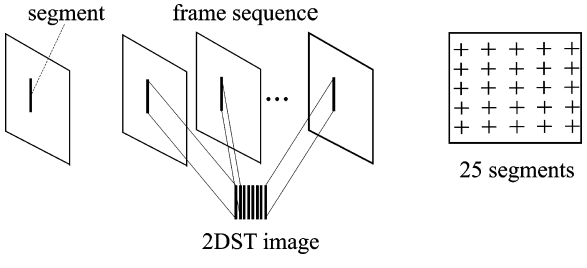

Fig. 23. Producing 2DST image using 25 horizontal and vertical segments.

to the time line for the background and unmoving objects. When the camera pans, the lines become slanted; in the case of zooming, they are spread.

We should note that performing edge detection on all frames in the video sequence is time consuming and computationally expensive.

In [23] the texture of 2-dimensional spatiotemporal (2DST) images is analyzed and the shots are divided into sub-shots described in terms of still scene, zoom and pan. The 2DST images are constructed by stacking up the corresponding segments of the images (Fig. 23). The directivity of the textures are calculated by computing the power spectrum by applying the 2-dimensional discrete Fourier transform.

4. Conclusions

Temporal video segmentation is the first step towards automatic annotation of digital video for browsing and retrieval. It is an active area of research gaining attention from several research communities including image processing, computer vision, pattern recognition and artificial intelligence.

In this paper we have classified and reviewed existing approaches for temporal video segmentation and camera operations recognition discussing their relative advantages and disadvantages. More than eight years of video segmentation research have resulted in a great variety of approaches. Early work focused on cut detection, while more recent techniques deal with gradual transition detection. The majority of algorithms process uncompressed video. They can be broadly classified into five categories, Fig. 24(a). Since the video is likely to be stored in compressed format, several algorithms which operate directly on the compressed video stream were reported. Based on the type of information used they can be divided into six groups, Fig. 24(b). Their limitations, that highlight the directions for further development, can be summarized as follows. Most of the algorithms (1)

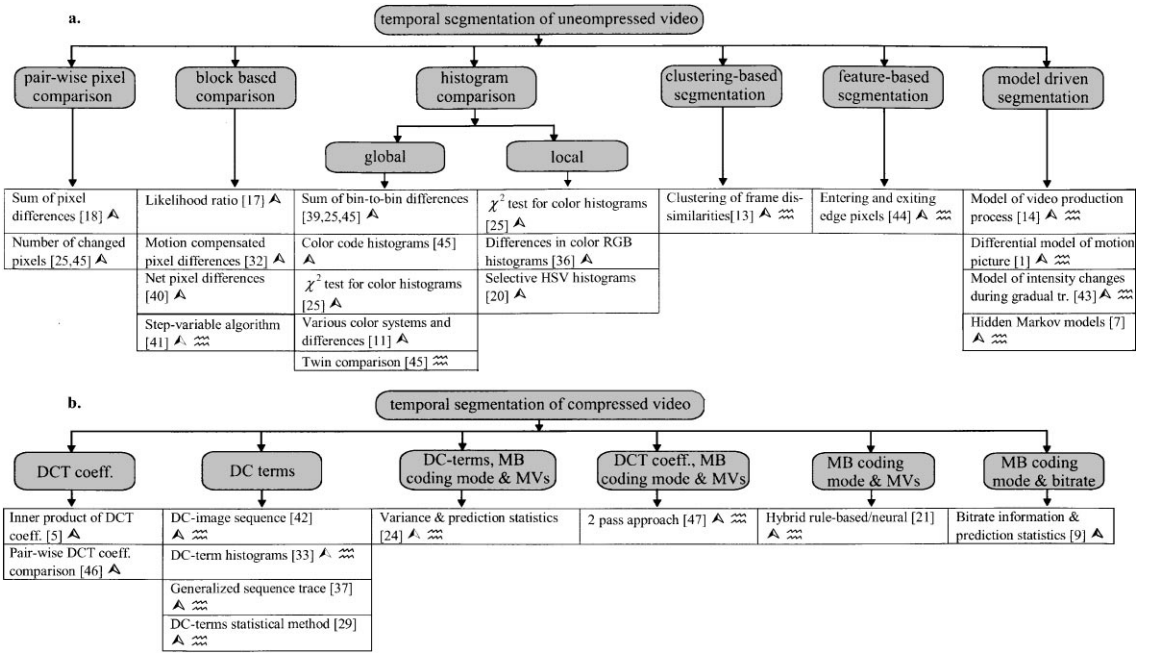

Fig. 24. Taxonomy of techniques for temporal video segmentation that process (a) uncompressed and (b) compressed video ( ) detect cut, ( ) detect gradual transitions.

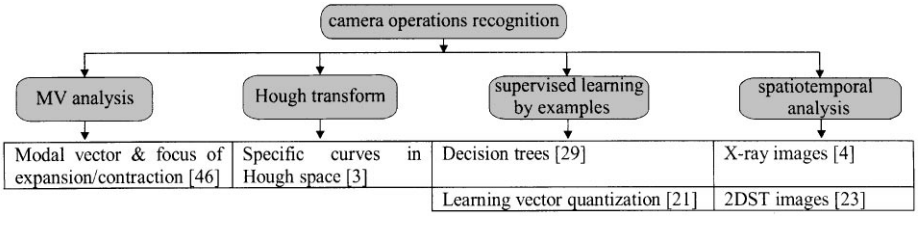

Fig. 25. Taxonomy of techniques for camera operation recognition.

require reconstruction of DC terms of *P* or P&B frames, or sacrifice temporal resolution and classification accuracy; (2) process unrealistically short gradual transitions and are unable to recognize the different types of gradual transitions; (3) involve many adjustable thresholds; (4) do not handle false positives due to camera operations. None of them is able to distinguish gradual transitions from object movement sufficiently well. Some of the ways to achieve further improvement include the use of additional information (e.g. audio features and text captions), integration of different temporal video segmentation techniques and development of methods that can learn from experience how to adjust their parameters. Camera operation recognition is an important issue related to the video segmentation. Fig. 25 presents the taxonomy of camera operation recognition techniques.

This research also confirms the need for benchmark video sequences and unified evaluation criteria that will allow consistent comparison and precise evaluation of the various techniques. The benchmark sequences should contain enough representative data for the possible types of camera operations and shot transitions, including complex gradual transition (i.e. between sequences involving

motion). The evaluation should take into consideration the type of application that indeed may require different trade-off between recall and precision. In case of gradual transition detection, an important evaluation criteria is the algorithm's ability to determine exactly between which frames the transition occurs and to classify the type of the transition (dissolve, fade, etc.). Other essential issues are the sensitivity to the encoder's type and the ease of implementation. Probably the best way for comparison and testing of the different temporal video segmentation techniques is to build a repository that contains Web-executable versions of the algorithms as suggested in [11]. It could be done by either providing an Web interface to the algorithms or by implementing them in a platform-independent language (e.g. Java).

Acknowledgements

The first author was supported by a fellowship from the Consorzio per lo Sviluppo Internazionale dell'Università di Trieste. The work was also supported in part by the European ESPRIT LTR project 20229 "Noblesse" and by the University of Trieste, grant 60%.

References

[1] P. Aigrain, P. Joly, The automatic real-time analysis of film editing and transition effects and its applications, Comput. Graphics 18 (1) (1994) 93–103.

[2] A. Akutsu, Y. Tonomura, Video tomography: an efficient method for camerawork extraction and motion analysis, in: Proceedings of ACM Multimedia'94, 1994, pp. 349–356.

[3] A. Akutsu, Y. Tonomura, H. Hashimoto, Y. Ohba, Video indexing using motion vectors, in: Proceedings of SPIE: Visual Communications and Image Processing 1818, Boston, 1992, pp. 1522–1530.

[4] G. Ananger, T.D.C. Little, A survey of technologies for parsing and indexing digital video, J. Visual Commun. Image Representation 7 (1) (1996) 28–43.

[5] F. Arman, A. Hsu, M.-Y. Chiu, Image processing on compressed data for large video databases, in: Proceedings of First ACM International Conference on Multimedia, 1993, pp. 267–272.

[6] J.S. Boreczky, L.A. Rowe, Comparison of video shot boundary detection techniques, in: Proceedings of IS & T/SPIE International Symposium Electronic Imaging, San Jose, 1996.

[7] J.S. Boreczky, L.D. Wilcox, A hidden Markov model framework for video segmentation using audio and image features, in: Proceedings of International Conference on Acoustics, Speech, and Signal Processing, Vol. 6, Seattle, 1998, pp. 3741–3744.

[8] S.-F. Chang, W. Chen, H.J. Meng, H. Sundaram, D. Zhong, VideoQ: an automated content based video search system using visual cues, in: Proceedings of ACM Multimedia Conference, Seattle, 1997.

[9] J. Feng, K.-T. Lo, H. Mehrpour, Scene change detection algorithm for MPEG video sequence, in: Proceedings of International Conference on Image Processing (ICIP'96), Lausanne, 1996.

[10] A.M. Ferman, A.M. Tekalp, Efficient filtering and clustering for temporal video segmentation and visual summarization, J. Visual Commun. Image Representation 9 (4) (1998) 3368–351.

[11] U. Gargi, R. Kasturi, S. Antani, Performance characterization and comparison of video indexing algorithms, in: Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), 1998.

[12] U. Gargi, S. Oswald, D. Kosiba, S. Devadiga, R. Kasturi, Evaluation of video sequence indexing and hierarchical video indexing, in: Proceedings of SPIE Conference on Storage and Retrieval in Image and Video Databases, 1995, pp. 1522–1530.

[13] B. Günsel, A.M. Ferman, A.M. Tekalp, Temporal video segmentation using unsupervised clustering and semantic object tracking, J. Electron. Imaging 7 (3) (1998) 592–604.

[14] A. Hampapur, R. Jain, T.E. Weymouth, Production model based digital video segmentation, Multimedia Tools Appl. 1 (1) (1995) 9–46.

[15] F. Idris, S. Panchanathan, Review of image and video indexing techniques, J. Visual Commun. Image Representation 8 (2) (1997) 146–166.

[16] ISO/IEC 13818 Draft Int. Standard: Generic Coding of Moving Pictures and Associated Audio, Part 2: video.

[17] R. Kasturi, R. Jain, Dynamic vision, in: R. Kasturi, R. Jain (Eds.), Computer Vision: Principles, IEEE Computer Society Press, Washington DC, 1991, pp. 469–480.

[18] T. Kikukawa, S. Kawafuchi, Development of an automatic summary editing system for the audio-visual resources, Trans. Electron. Inform. J75-A (1992) 204–212.

[19] T. Kohonen, The self-organizing map, Proc. IEEE 78 (9) (1990) 1464–1480.

[20] I. Koprinska, S. Carrato, Camera operation detection in MPEG video data by means of neural networks, in: Proceedings of COST 254 Workshop, Toulouse, 1997, pp. 300–304.

[21] I. Koprinska, S. Carrato, Detecting and classifying video shot boundaries in MPEG compressed sequences, in: Proceedings of IX European Signal Processing Conference (EUSIPCO), Rhodes, 1998, pp. 1729–1732.

[22] C.-M. Lee, D. M.-C. Ip, A robust approach for camera break detection in color video sequences, in: Proceedings

of IAPR Workshop Machine Vision Applications, Kawasaki, Japan, 1994, pp. 502–505.

[23] J. Maeda, Method for extracting camera operations to describe sub-scenes in video sequences, in: Proceedings of IS&T/SPIE Conference on Digital Video Compress, San Jose, Vol. 2187, 1994, pp. 56–67.

[24] J. Meng, Y. Juan, S.-F. Chang, Scene change detection in a MPEG compressed video sequence, in: Proceedings of IS&T/SPIE International Symposium on Electronic Imaging, San Jose, Vol. 2417, 1995, pp. 14–25.

[25] A. Nagasaka, Y. Tanaka, Automatic video indexing and full-video search for object appearances, in: E. Knuth, L.M. Wegner (Eds.), Visual Database Systems II, Elsevier, Amsterdam, 1995, pp. 113–127.

[26] W. Niblack, X. Zhu, J.L. Hafner, T. Breuer, D.B. Ponceleon, D. Petkovic, M.D. Flickner, E. Upfal, S.I. Nin, S. Sull, B.E. Dom, B.-L. Yeo, S. Srinivasan, D. Zivkovic, M. Penner, Updates to the QBIC system, in: Proceedings of IS&T/SPIE Conference on Storage and Retrieval for Image and Video Databases VI, Vol. 3312, 1997, pp. 150–161.

[27] T.N. Pappas, An adaptive clustering algorithm for image segmentation, IEEE Trans. Signal Process. 40 (1992) 901–914.

[28] G. Pass, R. Zabih, Comparing images using joint histograms, Multimedia Systems (1999) in press.

[29] N.V. Patel, I.K. Sethi, Video shot detection and characterization for video databases, Pattern Recognition 30 (1997) 583–592.

[30] I.K. Sethi, N.V. Patel, A statistical approach to scene change detection, in: Proceedings of IS&T/SPIE Conference on Storage and Retrieval for Image and Video Databases III, San Jose, Vol. 2420, 1995, pp. 2–11.

[31] I.K. Sethi, G.P.R. Sarvarayuda, Hierarchical classifier design using mutual information, IEEE Trans. Pattern Anal. Mach. Intell. (1982) 441–445.

[32] B. Shahraray, Scene change detection and content-based sampling of video sequences, in: Proceedings of IS&T/SPIE, Vol, 2419, 1995, pp. 2–13.

[33] K. Shen, E. Delp, A fast algorithm for video parsing using MPEG compressed sequences, in: Proceedings of International Conference on Image Processing (ICIP'96), Lausanne, 1996.

[34] M. Smith, T. Kanade, Video skimming and characterization through the combination of image and language understanding, in: Proceedings of International Workshop Content-Based Access of Image and Video Databases (ICCV'98), Bombay, 1998.

[35] M.J. Swain, Interactive indexing into image databases, in: Proceedings of SPIE Conference on Storage and Retrieval in Image and Video Databases, 1993, pp. 173–187.

[36] D. Swanberg, C.-F. Shu, R. Jain, Knowledge guided parsing in video databases, in: Proceedings of SPIE Conference, Vol. 1908, 1993, pp. 13–24.

[37] C. Taskiran, E. Delp, Video scene change detection using the generalized sequence trace, in: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, Seattle, 1998, pp. 2961–2964.

[38] L. Teodosio, W. Bender, Salient video stills: content and context preserved, in: Proceedings of ACM Multimedia Conference, Anaheim, CA, 1993, pp. 39–46.

[39] Y. Tonomura, Video handling based on structured information for hypermedia systems, in: Proceedings of ACM International Conference on Multimedia Information Systems, 1991, pp. 333–344.

[40] W. Xiong, J.C.-M. Lee, Efficient scene change detection and camera motion annotation for video classification, Comput. Vision Image Understanding 71 (2) (1998) 166–181.

[41] W. Xiong, J.C.-M. Lee, M.C. Ip, Net comparison: a fast and effective method for classifying image sequences, in: Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases III, San Jose, CA, Vol. 2420, 1995, pp. 318–328.

[42] B. Yeo, B. Liu, Rapid scene analysis on compressed video, IEEE Trans. Circuits Systems Video Technol. 5 (6) (1995) 533–544.

[43] H. Yu, G. Bozdagi, S. Harrington, Feature-based hierarchical video segmentation, in: Proceedings of International Conference on Image Processing (ICIP'97), Santa Barbara, 1997, pp. 498–501.

[44] R. Zabih, J. Miler, K. Mai, A feature-based algorithm for detecting and classifying production effects, Multimedia Systems 7 (1999) 119–128.

[45] H.J. Zhang, A. Kankanhalli, S.W. Smoliar, Automatic partitioning of full-motion video, Multimedia Systems 1 (1) (1993) 10–28.

[46] H.J. Zhang, C.Y. Low, Y.H. Gong, S.W. Smoliar, Video parsing using compressed data, in: Proceedings of SPIE Conference on Image and Video Processing II, 1994, pp. 142–149.

[47] H.J. Zhang, C.Y. Low, S.W. Smoliar, Video parsing and browsing using compressed data, Multimedia Tools Appl. 1 (1995) 89–111.