Soft Comput (2011) 15:469–482
DOI 10.1007/s00500-009-0527-9

FOCUS

Automatic video temporal segmentation based on multiple features

Shiguo Lian

Published online: 1 November 2009

Abstract This paper investigates automatic video temporal segmentation techniques, also named shot boundary detection (SBD) techniques. Firstly, the existing SBD algorithms are reviewed in detail. Then, a new SBD algorithm is proposed aiming to obtain fast and accurate detection, and its performances are evaluated and compared with existing works. This algorithm computes the frame difference/similarity by such simple features as pixel difference and histogram difference, adopts motion-based difference to resist camera or object movements in the same shot and uses the flash detection to avoid false positives caused by light changes or flashes. The adopted features are computational efficient, and the combination of various features improve the detection accuracy. These properties make the algorithm suitable for real-time applications, such as broadcasted news segmentation.

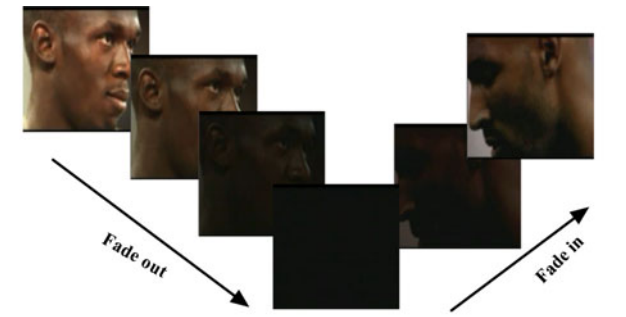
Keywords Video temporal segmentation · Shot boundary detection · News segmentation · Multimedia analysis

1 Introduction

With the rapid development of multimedia technology and network technology, multimedia information enriches the daily life of human beings, and the human–computer interface becomes more and more intelligent. The increased availability and usage of online digital video has created a need for automated video content analysis techniques (Divakaran 2009; Gong and Xu 2007; Wang et al. 2000). Most research on video content involves automatically detecting the boundaries between camera shots. A shot is an unbroken sequence of frames from one camera. Thus, a movie sequence that alternates between views of two people would consist of multiple shots. A scene is defined as a collection of one or more adjoining shots that focus on an object or objects of interest. For example, a person walking down a hallway into a room would constitute one scene, even though different camera angles might be shown. Three camera shots showing three different people walking down a hallway might constitute one scene if the important object was the hallway and not the people.

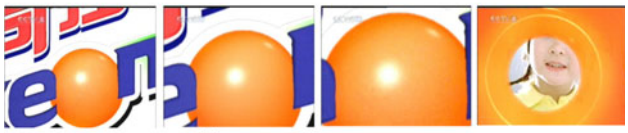
Generally, there is a transition between two adjacent shots. That is, the shot boundaries denote the points where the shot transition happens. Shot boundaries can be broadly classified into two types, i.e., abrupt transitions and gradual transitions (Ekin et al. 2003; Yuan et al. 2007). Abrupt transitions are instantaneous transitions from one shot to the subsequent shot. This is also called a cut. Gradual transitions include dissolves and fades, in which one frame is transformed into another as the proportion of each frame is gradually changed; wipes, in which the boundary between shots moves spatially; and special effects (zoom in/out, pan/ban, rotation, etc.), in which shapes and positions are transformed three dimensionally during the transition. For example, Fig. 1a shows a fade that is a slow change in brightness usually resulting in or starting with a solid black frame. Figure 1b shows a wipe that occurs when pixels from the second shot replace those of the first shot in a regular pattern, such as in a line from the left edge of the frames, and Fig. 1c shows a zoom out that occurs when the regions of frame are zoomed continuously. Of course, many other types of gradual transition are possible.

S. Lian (✉)
France Telecom R&D (Orange Labs), Beijing, China
e-mail: shiguo.lian@orange-ftgroup.com



(a) An example of fade in/out

(b) An example of wipe



(c) An example of zoom

Fig. 1 Examples of some gradual transitions

Intuitively, there are high similarities between adjacent frames in the same shot, while low ones between different shots. Shot boundary detection aims to detect the boundaries by computing and comparing the similarity or difference between adjacent frames. Generally, it is composed of two steps, i.e., computing frame similarity/difference, and making the decision. Generally, the correct detection rate and detection speed are two important metrics for evaluating a shot boundary detection algorithm's performances (Han et al. 2007; Kawai et al. 2007).

- *The detection rate* is often tested by recall rate (R), precision rate (P) and F-measure (F) (Han et al. 2007). They are commonly used in the field of information retrieval: Recall rate is defined as the percentage of desired items that are retrieved, and precision rate is defined as the percentage of retrieved items that are desired items. Here, in shot boundary detection, recall rate is defined as the percentage of desired boundaries that are detected, and precision rate is defined as the percentage of detected boundaries that are desired boundaries. They are defined as below. Here, *hit* denotes the number of detected boundaries, *miss* denotes the number of missed boundaries, *false* denotes the number of boundaries that should not be detected, and F-measure is the mean of recall rate and precision rate.

$$\begin{cases} R = \text{hit}/(\text{hit} + \text{miss}) \\ P = \text{hit}/(\text{hit} + \text{false}) \\ F = 2 \cdot R \cdot P/(R + P) \end{cases}. \quad (1)$$

- *Detection speed* is in close relation with the SBD algorithm's computational complexity. It is often measured by two metrics, i.e., SBD processing seed, and SBD processing time ratio (Kawai et al. 2007). The former one is defined as the number of frames that can be processed by the SBD algorithm in 1-s. The latter one is defined as the ratio between the SBD time cost and the video length.

Till now, various works (Ekin et al. 2003; Yuan et al. 2007) have been done to detect shot boundaries automatically, which have different properties in detection accuracy and detection speed. However, due to the varieties of video contents (movie, music video, television program, sports video, news video, etc.), it is still difficult to get ideal performances. Additionally, for real-time applications, e.g., broadcasted news segmentation, both high detection accuracy and efficient detection speed are expected. However, there is often a trade-off between the two aspects, and the existing works cannot solve this problem in a satisfactory manner (Yuan et al. 2007).

This paper aims to summarize existing works and proposes an SBD algorithm that can get a good trade-off between detection accuracy and detection speed. In detail, this paper reviews the existing SBD algorithms, analyzes and compares their performances, points out the open issues, and presents and evaluates the innovative algorithm. The rest of the paper is organized as follows. Section 2 introduces the existing methods to compute the similarity/difference between adjacent frames. The decision methods that decide whether the boundary exists are introduced in Sect. 3. In Sect. 4, the advantages and disadvantages of existing works are analyzed and compared. Then, the innovative SBD algorithm is proposed in Sect. 5, and its performances are evaluated in Sect. 6. Finally, in Sect. 7, some conclusions are drawn and open issues are given.

2 Similarity/difference between adjacent frames

There has been a good deal of research on automatic video content extraction. Early techniques focused on cut detection, and more recent work has focused on detecting gradual transitions. The major techniques that have been used for shot boundary detection include pixel differences, statistical differences, histogram comparisons, edge differences, mutual information between pixels, compression differences and motion vectors (Zhang et al. 1993; Nagasaka and Tanaka 1992; Zabih et al. 1993).

2.1 Pixel differences

The easiest way to detect if two frames are significantly different is by counting the number of pixels that change in value more than some threshold, t. This total is compared against a second threshold, T, to determine if a shot boundary has been found. This method is also named pair-wise comparison (Zhang et al. 1993). It is composed of two steps:

Step 1) Compute the difference between pixels' intensity value

$$\mathrm{DP}_i(k,l) = \begin{cases} 1 & \text{if} |P_i(k,l) - P_{i+1}(k,l)| > t \\ 0 & \text{otherwise} \end{cases}. \quad (2)$$

Here, $P_i(k, l)$ represents the pixel in the k-row, l-column of the ith frame.

Step 2) Compute the ratio of changed pixels

$$\frac{\sum_{k,l=1}^{M,N} \mathrm{DP}_i(k,l)}{MN} \times 100 > T. \quad (3)$$

Here, M and N are the width and height of a frame.

This method is sensitive to camera motion. The improved method (Zhang et al. 1993) implements this method with the additional step of using a 3 × 3 averaging filter before the comparison to reduce camera motion and noise effects. The authors found that by selecting a threshold tailored to the input sequence good results were obtained, although the method was somewhat slow. We note that manually adjusting the threshold is unlikely to be practical. Another method (Shahraray 1995) divides the images into 12 regions, and finds the best match for each region in a neighborhood around the region in the other image. This matching process duplicates the process used to extract motion vectors from an image pair. The pixel differences for each region were sorted, and the weighted sum of the sorted region differences provided the image difference measure. Gradual transitions were detected by generating a cumulative difference measure from consecutive values of the image differences. Additionally, the method in (Hampapur et al. 1994) computes what it calls chromatic images by dividing the change in gray level of each pixel between two images by the gray level of that pixel in the second image. During dissolves and fades, this chromatic image assumes a reasonably constant value. It also computes a similar image that detects wipes. Unfortunately, this technique is very sensitive to camera and object motion.

2.2 Statistical differences

Statistical methods expand on the idea of pixel differences by breaking the images into regions and comparing statistical measures of the pixels in those regions. For example, the method (Kasturi and Jain 1991) computes a measure based on the mean and standard deviation of the gray levels in regions of the images. The likelihood ratio (the second-order statistical characteristics of the regions' pixels) (Nam and Tewfik 2005) is computed according to

$$\frac{\left[\frac{S_i+S_{i+1}}{2} + \left(\frac{m_i-m_{i+1}}{2}\right)^2\right]^2}{S_i S_{i+1}} > T. \quad (4)$$

where m_i and S_i are the ith frame's mean of intensity values and variance, respectively. This method is reasonably tolerant to noise, but is slow due to the complexity of the statistical formulas. It also generates many false positives (i.e., changes not caused by a shot boundary).

2.3 Histogram comparisons

The frames in different shots often have different histograms. The frames in the same shot often have similar histograms, but different in different shots. Thus, histograms are the most common method used to detect shot boundaries. The simplest histogram method computes gray level or color histograms of the two images. If the bin-wise difference between the two histograms is above a threshold, a shot boundary is assumed. For example, the method in (Ueda et al. 1991) uses the color histogram change rate to find shot boundaries. Generally, two kinds of histogram difference (Zhang et al. 1993; Boreczky and Rowe 1996; Truong et al. 2000) are used. The first one computes the absolute distance bin by bin according to

$$\mathrm{SD}_i = \sum_{j=1}^{G} |H_i(j) - H_{i+1}(j)|. \quad (5)$$

Here, G represents the total number of histogram bins, and $H_i(j)$ represents the number of pixels in frame i contained in the jth bin. The second one computes the relative distance according to

$$\mathrm{SD}_i = \sum_{j=1}^{G} \frac{|H_i(j) - H_{i+1}(j)|}{H_{i+1}(j)}. \quad (6)$$

The paper (Nagasaka and Tanaka 1992) compares several simple statistics based on gray level and color histograms. It finds the best results by breaking the images into 16 regions, using a c2 test on color histograms of those regions and discarding the 8 largest differences to reduce the effects of object motion and noise. Another method (Swanberg et al. 1993) uses gray-level histogram differences in regions, weighted by how likely the region is to change in the video sequence. This worked well because the test video (CNN Headline News) had a very regular spatial structure. Some simple shot categorizations are done by comparing shots with the known types (e.g., anchor person shot) in a

database, and the shots are grouped into higher level objects, such as scenes and segments, by matching the shot types with the known temporal structure. The method in (Ekin et al. 2003) detects dominant colors in soccer video frames, which are then used to determine the existence of shot boundaries. The method in (Zhang et al. 1993) compares pixel differences, statistical differences and several different histogram methods and shows that the histogram methods are a good trade-off between accuracy and speed.

2.4 Compression differences

The features extracted from compression domain can also be used to detect the boundaries. For example, the method in (Little et al. 1993) uses differences in the size of JPEG compressed frames to detect shot boundaries as a supplement to a manual indexing system. The method in Arman et al. (1994) detects shot boundaries by comparing a small number of connected regions. It uses differences in the discrete cosine transform (DCT) coefficients of JPEG compressed frames as their measure of frame similarity, thus avoiding the need to decompress the frames. A further speedup is obtained by sampling the frames temporally and using a form of binary search to find the actual boundary. Potential boundaries are checked using a color histogram difference method. Another method (Joyce and Liu 2006) uses the distribution of DCT coefficients to detect the boundary. In the compression domain, the DC coefficients and several AC coefficients are extracted and used to form a vector, and then the vectors extracted from adjacent frames are compared to get the correlation value

$$\rho_k = \frac{\langle d^l_{k+l}, d^l_k \rangle}{\sqrt{\left\|d^l_{k+l}\right\|^2 \left\|d^l_k\right\|^2}}. \tag{7}$$

Here, d^l_k is the kth frame's DCT vector. Finally, the correlation is compared with a threshold to tell the boundary (Joyce and Liu 2006).

2.5 Edge tracking

For adjacent frames, there are often mutual edges, as shown in Fig. 2. By computing the non-mutual edges, the transition is detectable. In Zabih et al. (1993), the algorithms based on color histograms, chromatic scaling and edge detection are compared. The authors aligned consecutive frames to reduce the effects of camera motion and compared the number and position of edges in the edge-detected images. The percentage of edges that enter and exit between the two frames was computed. Shot boundaries were detected by looking for large edge change percentages. Dissolves and fades were identified by looking at the relative values of the entering and exiting edge percentages.

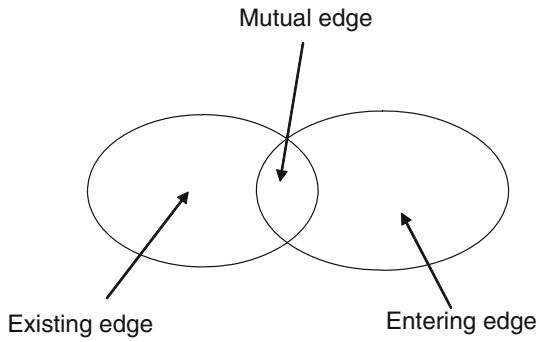


Fig. 2 Mutual edges between adjacent frames

The improved method (Zabih et al. 1999) was reported to be more accurate at detecting cuts than histograms and much less sensitive to motion than chromatic scaling.

2.6 Motion vectors

The methods in Ueda et al. (1991) and Zhang et al. (1993) uses motion vectors determined from block matching to detect whether or not a shot is a zoom or a pan. Another method in Shahraray (1995) uses the motion vectors extracted as part of the region-based pixel difference computation described above to decide if there is a large amount of camera or object motion in a shot. Because shots with camera motion can be incorrectly classified as gradual transitions, detecting zooms and pans increases the accuracy of a shot boundary detection algorithm. Motion vector information can also be obtained from MPEG compressed video sequences. However, the block matching performed as part of MPEG encoding selects vectors based on compression efficiency and thus often selects inappropriate vectors for image processing purposes. Generally, two kinds of methods are used to estimate the motion direction or motion compensation. The first one is based on optical flow computation, which detects the motion direction by computing the optical flows (Zhang et al. 1993). Given the reference direction, θ_m, the motion direction, θ_k, can be determined if it satisfies

$$\sum_k^N |\theta_k - \theta_m| \le \theta_p. \tag{8}$$

Here, θ_P is the threshold for decision. The second one is based on block motion matching. For the kth block in a frame, the matched block in the previous frame is searched by computing the cost function

$$P_i(k) = \sum_m^M \sum_n^N F[b_i(m,n) - b_{i+1}(n,m)]. \tag{9}$$

Here, $b_i(m, n)$ is the block in the ith frame. The matched block is searched when the minimal cost function F is

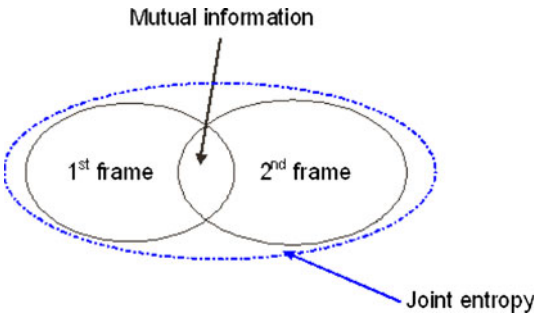


Fig. 3 Boundary detection based on entropy

obtained. In that way, the motion vector can also be obtained.

2.7 Mutual information between pixels

Intuitively, there is mutual information between adjacent frames. Similar with mutual edge detection, mutual information also exists in the pixels. Figure 3 shows the entropy-based boundary detection method (Cernekova et al. 2006), where the cuts are detected by mutual information, and the fades detected by joint entropy.

3 Decision methods

After computing the similarity/difference between adjacent frames, the decision should be made according to a suitable method. Generally, four kinds of decision methods are often used, i.e., threshold-based method, B-spline-based method, Fuzzy c-means method and SVM-based method.

3.1 Threshold-based method

It is the straightforward method that compares the computed similarity/difference metric with the thresholds (Gao and Tang 2002). Generally, more than one threshold is used to detect both cuts and graduals. The study (Zhang et al. 1993) shows the example based on twin comparison. Here, the histogram differences between adjacent frames are computed. To properly detect gradual transitions such as wipes and dissolves, they used two thresholds. Generally, the cuts can be detected by the first threshold, while the graduals should be detected by the second threshold that is often smaller than the first one. If the histogram difference falls between the thresholds, it is tentatively marked as the beginning of a gradual transition sequence, and succeeding frames are compared against the first frame in the sequence. If the running difference exceeds the larger threshold, the sequence is marked as a gradual transition.

Additionally, the threshold can be changed according to the properties of video content. Thus, adaptive threshold scheme are proposed in Yusoff et al. (2000). In this scheme, three kinds of models are proposed, i.e., constant variance model, proportional variance model and the Dugad model (Dugad et al. 1998). In constant variance model, the adjacent frames' dissimilarity is distributed in a unimodal manner, and thus, the threshold satisfies

$$T = \mu + T_c.$$

Here, T, μ and T_c are the threshold, mean of dissimilarity, and offset, respectively. Generally, T_c is determined by experimenting with a range of values on a training set of video material for which ground truth information is available. It is easy to understand that the threshold is added to the mean of samples. In proportional variance model, the threshold is computed by

$$T = T_p \mu.$$

Here, T_p is also determined from experimentation. It is easy to understand that the threshold is multiplied by the mean of samples. In the Dugad model, the threshold is computed by

$$T = \mu + T_d \sqrt{\sigma}.$$

Here, σ is the standard deviation of the adjacent frames' dissimilarity, and T_d is also determined from experimentation. It is easy to understand that the threshold is multiplied by the standard deviation of the samples and added to the mean of samples.

3.2 B-spline-based method

Generally, according to gradual transitions' properties, the frame difference in gradual changes gradually compared with the frame difference of the cut. The change curve can be modeled as a B-spline curve (Ngo 2003). Thus, after computing the frame differences, by doing B-spline interpolation, the gradual transition can be detected.

3.3 Fuzzy c-means method

For different transitions or adjacent frames, the similarity/difference has different statistical properties. Using such classification method as fuzzy c-means, the transitions can be detected or even classified. Generally, this kind of method (Matsumoto et al. 2006) consists of two steps, i.e., computing the difference features in spatial or histogram and classifying based on fuzzy c-means.

3.4 SVM-based method

Supported vector machine can be used to distinguish shot boundaries adaptively (Han et al. 2007). Generally, it is

composed of three steps, i.e., computing the features (e.g., in DCT, wavelet or histogram, etc.), training the SVM with video sequences and using the trained SVM to detect shot boundaries (Feng et al. 2005).

4 Performance comparison

4.1 Difficulties of SBD algorithms

Due to the varieties of video contents and special effects, there are some difficulties for SBD. Some typical ones are listed as follows.

- *Small moving object* This kind of shot often contains only a small moving object and a large background. Thus, although the shot transition happens, the similarity/difference between adjacent frames is so small that it may not be regarded as a transition. In this case, the transitions may be missed, which leads to the loss of recall rate.
- *Flash or light changes* The flash or immediate light changes often increase greatly the differences between adjacent frames. Thus, they often lead to false positive detection. Till now, no single metric (mentioned in Sect. 2) can tell how this happens correctly.
- *Fast motion* In some video contents, e.g., sports video, the fast motions of camera/object may lead to large differences between adjacent frames, during which, shot boundaries are sometimes detected in a mistake (false positives).
- *Blurred frames* Sometimes, there are blurs in the video frames. Generally, blurs happen in two cases. The first case is the very fast moving of camera or object. In this case, the camera's capturing speed cannot catch up with the moving speed. The second case is the special effects added by editors. The blurs often make the working of difference metric impossible, which leads to false positive in shot boundary detection.
- *Varieties of gradual transitions* As we know, shot boundaries are classified into two types, i.e., abrupt transition (cut) and gradual transition. It is easy to explain the cut. However, with the development of video processing techniques, various gradual transitions have appeared, e.g., fade in/out, dissolve, zoom in/out, wipe, pan/ban, rotation, etc. Additionally, some special effects appear, e.g., 3D objects and motions, and the combination of various gradual operations. For example, the 3D object flies in, and, at the same time, it is rotated uniformly. Some existing works aim to detect each kind of gradual transition with certain methods. However, with the variety increasing, the detection becomes more and more difficult.
- *Varieties of video contents* Till now, there are many kinds of video contents, e.g., movie, music video, TV program, news program, sports video, etc., and they often have different properties. For example, there are many more special effects in television commercials than in other videos, there are many shot boundaries in news program, flashes in music video and many more fast motions in sport videos. It cannot be confirmed whether the decision parameters are suitable for all kinds of contents.

4.2 Comparison of existing SBD algorithms

According to the above analysis, we summarize the performances of different SBD algorithms. Here, the considered SBD algorithms include B-spline-based method, Fuzzy c-means method, SVM-based method, and 7 threshold-based algorithms. The considered performances (Boreczky 1996; Boreczky and Rowe 1996) include the ability to detect cuts, the ability to detect graduals, the sensitivity to flash/light changes, the sensitivity to fast motion, and detection speed. Seen from Table 1, no algorithms can get good performances in both detection ability and detection speed, and are robust against flash/light changes and fast motions. For threshold-based methods, some of them can detect both cuts and graduals, some of them are efficient in detection speed, but most of them are sensitive to flash/light changes. The B-spline method is difficult to detect cuts, and spends much time on curve interpolation. The method based on fuzzy c-means is difficult to detect graduals accurately. The SVM-based method needs to train the classifier beforehand.

Typically, some algorithms' detection rate and detection speed are compared here. For the algorithm in Kawai et al. (2007), the detection rate of cut is $R = 0.961$, $P = 0.939$, $F = 0.948$, and the detection rate of gradual is $R = 0.578$, $P = 0.768$, $F = 0.660$. For the algorithm in Ekin et al. (2003), the detection rate of cut is $R = 0.973$, $P = 0.917$, and for gradual is $R = 0.853$, $P = 0.866$. For the algorithm in (Han et al. 2007), the detection rate of cut is $F = 0.983$ and for gradual is $F = 0.848$. For the algorithm in Kawai et al. (2007), the ratio between SBD time and video sequence length is 1/123 (Intel Core 2 Duo E6600 2.40 GHz processor and 2 GB RAM). For the algorithm in Han et al. (2007), the SBD algorithm's processing speed is 25–30 FPS (PC with P4-1.7 GHz CPU and 512 MB RAM).

According to the above comparison and analysis, it is difficult to get good performances based on only one of the similarity/difference metrics. For some applications without the chance of training, the supervised method based on learning, e.g., SVM, is not practical. Additionally, considering that video content is often of large volumes, high

Table 1 Performance comparison of existing SBD algorithms

SBD algorithms	Performances				
	Cut detection	Gradual detection	Flash/light changes	Fast motion	Detection speed
Pixel differences + Threshold	Yes	Yes	Sensitive	Sensitive	High
Statistical differences + Threshold	Yes	Yes	Sensitive	Sensitive	Low
Histograms + Threshold	Yes	Yes	Sensitive	Robust	High
Compression differences + Threshold	Yes	Yes	Sensitive	Sensitive	Low
Edge tracking + Threshold	Yes	Yes	Robust	Sensitive	Low
Motion vector + Threshold	Yes	No	Sensitive	Robust	Low
Mutual pixel information + Threshold	Yes	Yes	Sensitive	Sensitive	High
Features + B-spline	No	Yes	Sensitive	Sensitive	Low
Features + Fuzzy c-means	Yes	No	Sensitive	Sensitive	Middle
Features + SVM	Yes	Yes	Sensitive	Robust	Low (need training)

detection speed is expected, which makes SBD available for real-time applications. The potential solution is based on the combination of various features. For example, the method (Kawai et al. 2007) presented in TRECVID 2007 gets good performances in detection speed and detection rate. The method can detect not only cuts, but also some gradual transitions, including fades in/out and dissolves. The major advantages include: using only the features with low cost, combining various metrics together, skipping the frames that are clearly not shot boundaries, and avoiding the complex operations based on classification or learning. The disadvantages include: it detects the cuts and gradual transitions in a parallel manner, and detects only two kinds of gradual transitions in a serial manner.

5 Proposed SBD algorithm

5.1 Principles of algorithm design

Motivated by the method in Kawai et al. (2007), we propose a lightweight SBD algorithm. It can overcome the disadvantages of the method (Kawai et al. 2007). This algorithm is based on the following principles:

(1) Uses various simple features. The similarity/difference metric is computed based on pixels, histograms, blocks, etc.
(2) Omits the frames that are clearly not shot boundaries by using the simplest feature, such as pixel comparison.
(3) Detects cuts and gradual transitions in serial manners.
(4) Detects various gradual transitions at the same time.
(5) No complex features are used, e.g., pixel distribution, DCT coefficient, etc.
(6) No classification or training algorithm is used, e.g., fuzzy c-mean, SVM, etc.

5.2 Architecture of the algorithm

The architecture of the algorithm is shown in Fig. 4. Here, the uncompressed video sequence (in YUV format) is processed by cut detection and gradual detection serially. That is, if the cut happens, then the gradual transition is not detected. Cut is detected for each two frames. Differently, since gradual transitions often happens with a duration (a sequence of frames), the gradual transition is detected for every group of frames. Both cut detection and gradual detection are controlled by the parameters determined by experiments. Finally, based on the two steps, the shot boundaries (cuts or gradual transitions) are produced.

5.3 Cut detection

In cut detection, each two frames are processed by several steps of operations. As shown in Fig. 5, it consists of four steps: computing pixel-based frame difference, histogram-based frame difference, motion-based frame difference and detecting flash or light changes. In each step, if the frame pair is clearly not a cut, then the following steps will be skipped.

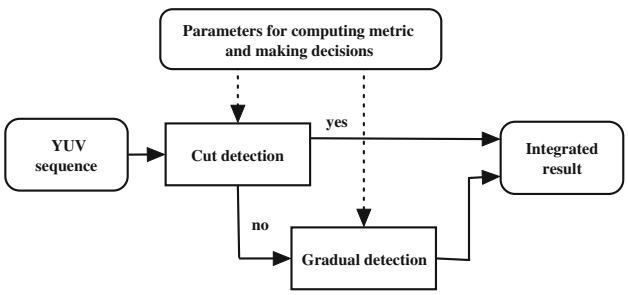


Fig. 4 The architecture of the proposed SBD algorithm

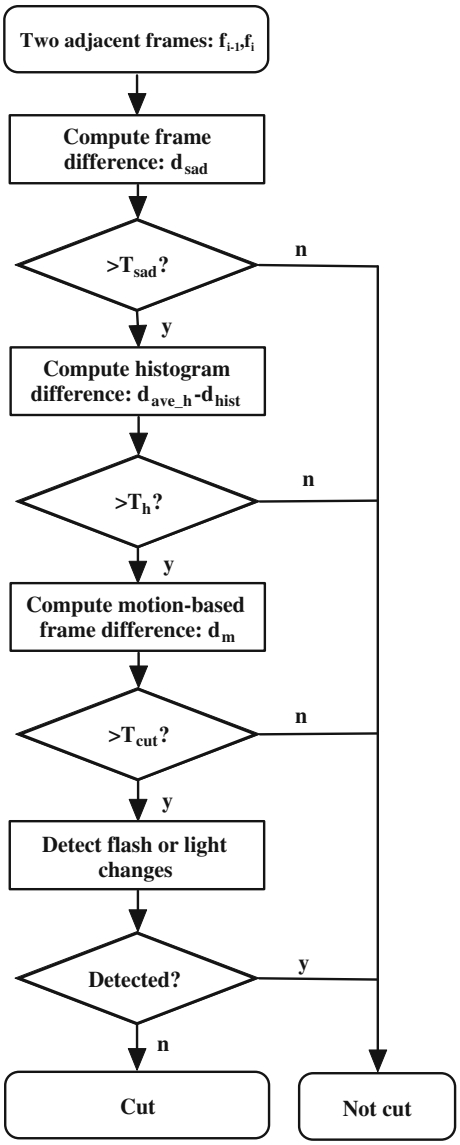


Fig. 5 Details of cut detection steps

Step 1: Compute pixel-based frame difference

First, frames that show no significant change are determined by using the sum of the absolute differences between pixel R, G and B values, d_{sad}. Here, d_{sad} is computed by

$$d_{\text{sad}}(f_{i-1},f_i) = \frac{1}{|\mathbf{F}|}\sum_{\mathbf{r}\in\mathbf{F}} |f_{i-1}(\mathbf{r}) - f_i(\mathbf{r})|. \tag{10}$$

$f_i(\boldsymbol{r})$ indicates the pixel value at coordinates $\boldsymbol{r}$ of the ith frame. $\mathbf{F}$ represents the pixels of the overall frame, and $|\mathbf{F}|$ represents the total number of pixels in a frame. The formula described here has been simplified to avoid undue complexity, but in practice the sum of differences of each of the R, G and B values is required. If d_{sad} is less than a threshold T_{sad}, it is judged that the frame cannot be a shot boundary, and the remaining processing is skipped.

Step 2: Compute histogram-based frame difference

Set d_{hist} as the sum of absolute histogram differences. d_{hist} is calculated by creating a frequency histogram of pixel values for each block region and then computing the sum of the absolute differences for each bin. Here, d_{hist} is calculated by

$$d_{\text{hist}}(f_{i-1},f_i) = \frac{1}{N_c}\sum_{c=1}^{N_c} |H_{i-1}(c) - H_i(c)|. \tag{11}$$

Here, N_c represents the total number of histogram bins, and $H_i(c)$ represents the number of pixels in frame i contained in the cth bin. $d_{\text{ave\_h}}$ is defined as the mean of previous histogram differences:

$$d_{\text{ave\_h}} = \frac{1}{i-1}\sum_{k=1}^{i-1} d_{\text{hist}}(f_{k-1},f_k). \tag{12}$$

Then, the difference, $d_{\text{ave\_h}} - d_{\text{hist}}$, is computed and compared with the threshold T_{h}. If the difference is not bigger than the threshold, then the frame pair is not a cut and the following steps will be skipped.

Step 3: Compute motion-based frame difference

The frame difference based on block matching, d_{bm} is used, which is robust against camera operations (e.g. zoom in/out, pan and tilt) or object motions. Then, it is determined whether a cut transition exists between frames based on the frame difference d_{bm}. Our method detects a cut when the increase in frame difference d_{bm} exceeds a certain threshold, because d_{bm} can vary significantly due to strong movements of the camera or the object, resulting in false detections. The following condition is used for this decision.

$$d_{\text{m}} = d_{\text{bm}}(f_{i-1},f_i) - d_{\text{bm}}(f_{i-2},f_{i-1}) > T_{\text{cut}}. \tag{13}$$

If the condition is true, a cut is detected between frame $i-1$ and frame i. Otherwise, the processing is terminated here.

In the block matching method, the current frame is divided into non-overlapping block regions, and then the previous frame is searched to find the position of minimum inter-block cost for each block of the current frame. The frame difference is calculated by adding up the total number of blocks for which the cost is higher than a threshold. The d_{bm} is calculated by

$$d_{\text{bm}}(f_{i-1},f_i) = \frac{1}{N}\sum_{n=1}^{N} 1 \quad \text{if} \quad \lambda_n(f_{i-1},f_i) > T_\lambda. \tag{14}$$

Here, N represents the total number of blocks, and λ_n represents the minimum value of the inter-block cost for the nth block. T_λ is the threshold. The sum of squared differences or the sum of absolute differences is generally

used as the inter-block cost to achieve motion vectors for video encoding. In contrast, we adopt the sum of absolute histogram differences, which is more robust to camera motions. λ_n is calculated by

$$\lambda_n(f_{i-1},f_i) = \min_{\mathbf{v}\in\mathbf{S}_n}\{d_{\text{hist}}(f_{i-1}(\mathbf{r}+\mathbf{v}),f_i(\mathbf{r})),\mathbf{r}\in\mathbf{B}_n\}. \quad (15)$$

Here, $\mathbf{S}_n$ indicates the search range, and $\mathbf{B}_n$ indicates the pixels contained within the nth block region. d_{hist} represents the sum of absolute histogram differences. As mentioned in Step 2, d_{hist} is calculated by creating a frequency histogram of pixel values for each block region and then computing the sum of the absolute differences for each bin.

Step 4: Detect flash or light changes

Considering that the flash or light changes will lead to great changes on frame similarity/difference, the properties should be considered to avoid the potential false positives. The most important property of flash or light changes is that there is often a duration in flash or light changes. Thus, we adopt histogram-based method to compare different frame pairs during a certain period. Here, two frame sets are selected:

$$\{f_{i-Nf},f_{i-\text{Nf}+1},\ldots,f_{i-1}\} \quad \text{and} \quad \{f_i,f_{i+1},\ldots,f_{i+\text{Nf}-1}\}.$$

Then, the histogram-based frame difference is computed for each frame pair (one frame is selected from the first set, and the other from the second set), compared with the average difference d_{ave_h}, and finally, gives the decision by comparing with the threshold T_h. The process is shown in the following pseudo code:

For k=0 to N_f-1
  Compute $d_{hist}(f_{i-k-1}, f_{i+k})$;
  Compute d_{ave_h}-$d_{hist}(f_{i-k-1}, f_{i+k})$;
  If d_{ave_h}-$d_{hist}(f_{i-k-1}, f_{i+k})$ T_h
    The flash/light changes are detected;
    Stop;
  Else
    The flash/light changes are not detected;
    Continue;
  End If
End For

5.4 Gradual transition detection

Compared with cuts, gradual transitions often consist of a sequence of frames. To detect this kind of transition, we propose the method to compare not two adjacent frames, but the frames with N_d-distance. As shown in Fig. 6, the detection process consists of four steps: computing the pixel-based frame difference to skip non-transitions, computing histogram-based difference to skip non-transitions, computing the similarity between frames to skip false positives, and computing motion-based frame difference to skip fast camera/object motions.

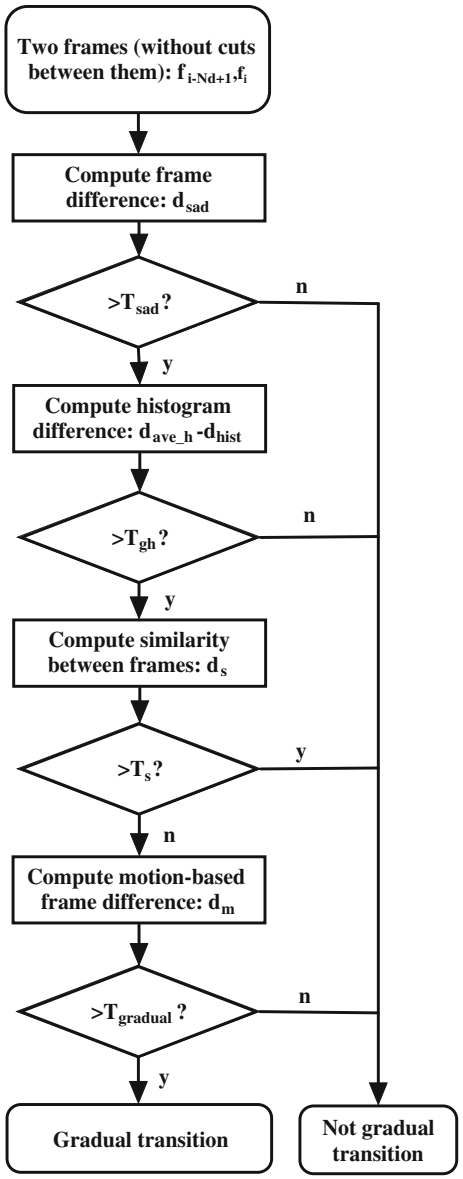


Fig. 6 Details of gradual detection

Step 1: Compute pixel-based frame difference

It is similar with the Step 1 of cut detection. Differently, the difference is computed based on two noncontinuous frames (with the distance of N_d), and between them, there is no cut. The pixel-based frame difference is computed by

$$d_{\text{sad}}(f_{i-N_d+1},f_i) = \frac{1}{|\text{F}|}\sum_{r\in F}|f_{i-N_d+1}(\text{r}) - f_i(\text{r})|. \quad (16)$$

Step 2: Compute histogram-based frame difference

It is similar with the Step 2 of cut detection. But, there are two differences: (1) the difference is computed based on two noncontinuous frames

$$d_{\text{hist}}(f_{i-N_d+1}, f_i) = \frac{1}{N_c}\sum_{c=1}^{N_c} |H_{i-N_d+1}(c) - H_i(c)|. \quad (17)$$

(2) The threshold is changed from T_h to T_{gh}. Here, T_{gh} is slightly bigger than T_h.

Step 3: Compute the frame similarity

This similarity is calculated based on the cosine of frame images, which is insensitive to luminance changes over the whole frame. In detail, the similarity is computed by

$$d_s(f_{i-N_d+1}, f_i) = \frac{\sum_{r\in F} f_{i-N_d+1}(r)\cdot f_i(r)}{\sqrt{\sum_{r\in F} f^2_{i-N_d+1}(r)\sum_{r\in F} f^2_i(r)}}. \quad (18)$$

That is, it is compared with the threshold T_s. If the similarity is bigger than the threshold, there is no transition between the considered frames.

Step 4: Compute motion-based difference

It is similar to the Step 4 of cut detection except in two points. 1) The difference is computed between two non-continuous frames and not continuous frames, as shown in the following equation.

$$d_m = d_{bm}(f_{i-N_d+1}, f_i) - d_{bm}(f_{i-N_d}, f_{i-N_d+1}) > T_{\text{gradual}}. \quad (19)$$

(2) The threshold is changed from T_{cut} to T_{gradual}. Here, T_{gradual} is bigger than T_{cut}.

6 Performance evaluation

6.1 Architecture of the test system

We implement the SBD algorithm in C code. In experiments, we use the system composed of two components, i.e., video decoding and SBD. Among them, the video decoding process is implemented by FFmpeg (FFmpeg, a free tool for video transcoding or processing). The system will generally produce two files, i.e., the YUV video sequence (converted from the original video) and the SBD result in XML file, as shown in Fig. 7.

In experiments, we tested the proposed SBD algorithm on news programs, as shown in Table 2, and compared it with the histogram-based algorithm (Zhang et al. 1993) and the serial-detection algorithm (Kawai et al. 2007). For the two previous algorithms, the recommended parameters are used, while our algorithm uses the parameters optimally selected through experiments.

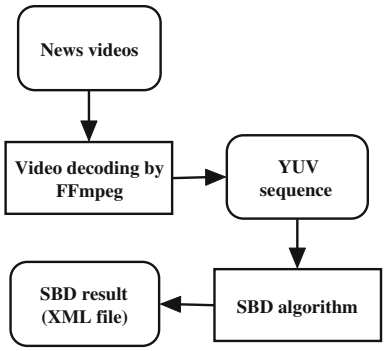


Fig. 7 Architecture of the test system

Table 2 The tested news programs

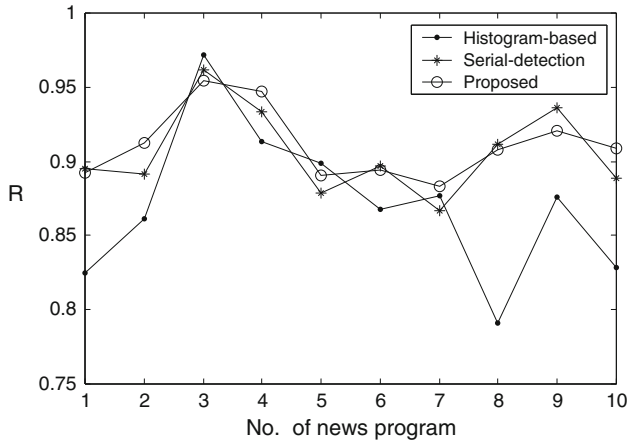
News program	Duration (hh:mm:ss)	Number of Transitions	
		Cut	Gradual
1	0:09:59	103	15
2	0:09:59	124	20
3	0:25:44	319	24
4	0:15:20	221	50
5	0:36:33	465	89
6	0:30:40	406	60
7	0:14:52	253	53
8	0:09:59	100	16
9	0:24:59	309	60
10	0:37:05	445	36

6.2 Cut detection rate

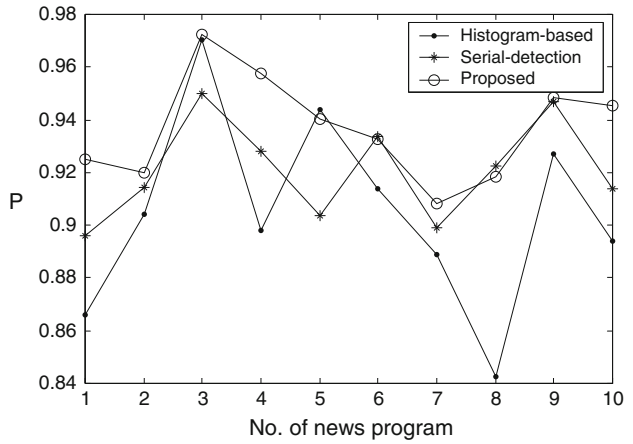
In cut detection, we use the following parameters carefully selected from experiments: $T_{sad} = 10$, $T_h = 0.03$, $T_\lambda = 30$, $T_{cut} = 0.25$ and $N_f = 6$. The detection rates (R: recall rate, P: precision rate, F: F-measure) are computed and compared, as shown in Fig. 8. In detail, the R, P and F rates are shown in Fig. 8a–c, respectively.

Seen from Fig. 8a, for recall rate, the proposed algorithm gets an averaged rate of $R = 0.969$, which is slightly bigger than the histogram-based algorithm ($R = 0.956$) while much bigger than the serial-detection algorithm ($R = 0.931$). It is caused by the serial-detection algorithm's multiple conditions. For example, in the serial-detection algorithm, such cases as small region changes may be missed.

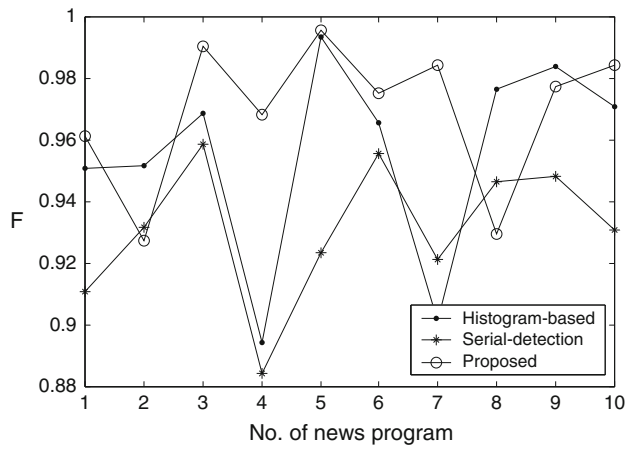
Seen from Fig. 8b, for precision rate, the proposed algorithm gets the averaged rate of $P = 0.911$, which is similar to the serial-detection algorithm ($P = 0.906$), while bigger than the histogram-based algorithm ($P = 0.871$). The proposed algorithm needs less conditions than the serial-detection algorithm, while the serial-detection



(a) Recall rate of cut detection



(b) Precision rate of cut detection



(c) F-measure of cut detection

Fig. 8 Comparisons of cut detection

algorithm cannot detect the continuous flashes. The histogram-based algorithm does not consider flashes or light changes that often happen in news programs. Additionally, for the cases with fast motion, the histogram-based algorithm is worse than the other two.

Seen from Fig. 8c, for F-measure rate, the proposed algorithm gets the averaged rate of $F = 0.937$, which is bigger than both the serial-detection algorithm ($F = 0.921$) and the histogram-based algorithm ($F = 0.905$). It shows that the proposed algorithm obtains the highest average detection rate or best detection accuracy.

There are some error cases for the proposed algorithm. By investigating news programs, we can list some of them. For example, some changes in small regions, as shown in Fig. 9a, may be missed, while the fast dissolve, as shown in Fig. 9b, may be detected as a cut by a mistake. Additionally, the image blurs may lead to false positive, as shown in Fig. 9c.

6.3 Gradual detection rate

In gradual detection, we use the following parameters: $N_d = 6$, $T_{gsad} = 8$, $T_{gh} = 0.75$, $T_s = 0.70$, and $T_{gradual} = 0.5$. For different SBD algorithms, the detection rate (F: F-measure) are computed and compared, as shown in Fig. 10. As seen, the proposed algorithm gets the averaged rate of $F = 0.697$, which is bigger than the serial-detection algorithm ($F = 0.653$) while smaller than the histogram-based algorithm ($F = 0.716$). Considering that the serial-detection algorithm can only detect two kinds of gradual transitions, i.e., fades and dissolves, it fails to detect the wipes that often appear in news programs.

Of course, there are some error cases in gradual detection. For example, the very short dissolve with only three frames, as shown in Fig. 11a, may be missed and detected as a cut by mistake. The frame sequence with frequent flash/light changes and camera/object motions, as shown in Fig. 11b, may be detected as a gradual transition by mistake.

6.4 Detection speed

Detection speed is in close relation with the algorithm's computational complexity. It is often measured by two metrics, i.e., segmentation processing speed and segmentation processing time ratio. The former is defined as the number of frames that can be processed by the segmentation algorithm in 1 s. The latter one is defined as the ratio between the segmentation time cost and the video length. The detection time ratio of the proposed method is tested and compared with other methods, as shown in Table 3. Here, the algorithms are implemented in C code, the computer is of 2.80 GHz CPU and 2.00 GB RAM, and five news programs are tested. As can be seen, the proposed algorithm gets the average processing time ratio, 0.093, which is smaller than the serial-detection algorithm, while bigger than the histogram-based algorithm. Considering that the frame rate is often 25 fps, the detection speed

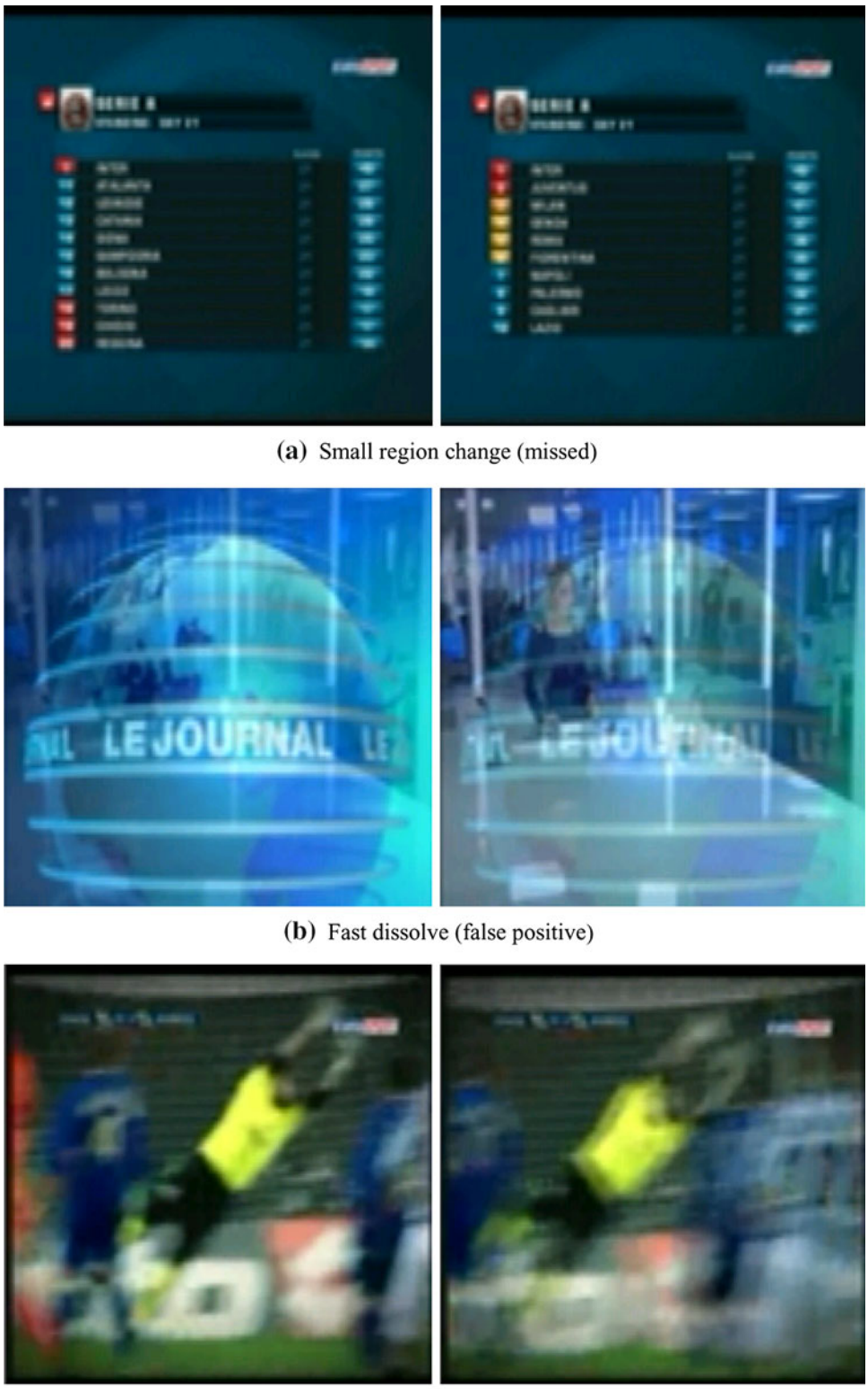


Fig. 9 Error cases of cut detection

(a) Small region change (missed)

(b) Fast dissolve (false positive)

(C) Fast motion + blur (false positive)

(SBD time/video sequence length) is 1/255, which is much smaller than the one (1/123) obtained by the serial-detection algorithm (Kawai et al. 2007). Thus, it is applicable to real-time scenarios.

7 Conclusions and discussions

This study reviews existing video temporal segmentation or shot boundary detection algorithms, analyzes their

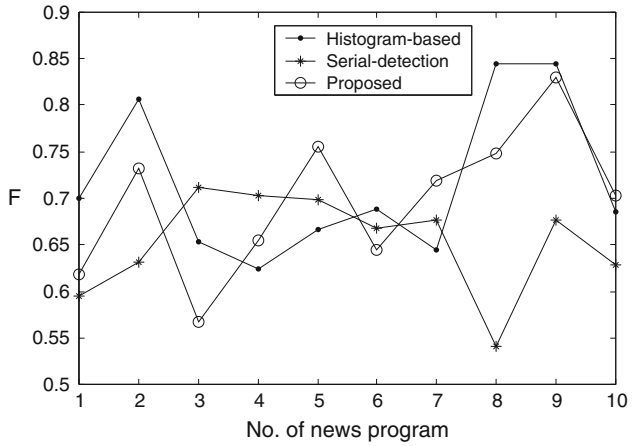


Fig. 10 Comparisons of gradual transition detection (F-measure)

performances, proposes a fast SBD algorithm and tests its performances based on news programs. For the proposed SBD algorithm, we get the following conclusions:

(1) The proposed SBD algorithm gets higher detection rate than some typical algorithms. Especially for cut detection, the proposed algorithm obtains higher R rate, P rate and F-measure. Although, for gradual transition detection, the proposed algorithm's detection accuracy is slightly worse, there are much more cuts than graduals in news programs, and the overall detection accuracy can be kept at a high level.

(2) The proposed SBD algorithm has a satisfactory detection speed. The average detection speed (SBD time/video sequence length) is 1/255, which is applicable to real-time applications.

Of course, there are still some open issues that need to be investigated:

(1) Gradual detection: as already known, there are various graduals, e.g., fade, dissolve, wipe, zoom, rotation, etc. It is still difficult to detect all of them one by one, especially when they are combined together.

(2) False positive often results when some blurs happen, especially in sports videos. It is still difficult to solve this problem, since existing methods based on pixel, histogram or texture do not work.

(3) Different parameter sets will be selected for different test sets, e.g., news programs and sports programs. It is a challenge to make only one set of parameters suitable for various programs.

(a) Very short dissolve with 3 frames (missed)

(b) Frequent light/flash changes and motions (false positive)

Fig. 11 Error cases in gradual detection

Table 3 Tests of detection speed

News program	Duration (hh:mm:ss)	Proposed algorithm		Histogram-based algorithm		Serial-detection algorithm	
		SBD time (hh:mm:ss)	Time ratio (SBD/Duration)	SBD time (hh:mm:ss)	Time ratio (SBD/Duration)	SBD time (hh:mm:ss)	Time ratio (SBD/Duration)
5	0:36:33 (2,193 s)	0:03:10	0.087	0:02:00	0.055	0:04:49	0.132
7	0:14:52 (892 s)	0:01:48	0.121	0:00:58	0.065	0:02:52	0.193
8	0:09:59 (599 s)	0:00:51	0.085	0:00:31	0.052	0:01:59	0.199
9	0:24:59 (1,499 s)	0:02:08	0.085	0:01:13	0.049	0:03:17	0.131
10	0:37:05 (2225 s)	0:03:09	0.085	0:02:02	0.054	0:04:51	0.131

(4) It is difficult to make comparisons between algorithms based on only recall and precision values. The targeted applications may have to be considered.

Acknowledgments This work was partially supported by the Invenio Project launched by France Telecom.

References

Arman F, Hsu A, Chiu MY (1994) Image processing on encoded video sequences. Multimed Syst 1(5):211–219

Boreczky J (1996) Using video and audio data for content extraction and shot and scene boundary determination. Ph.D. dissertation, University of California Berkeley

Boreczky JS, Rowe LA (1996) Comparison of video shot boundary detection techniques. J Electron Imaging 5(2):122–128

Cernekova Z, Pitas I, Nikou C (2006) Information theory-based shot cut/fade detection and video summarization. IEEE Trans Circ Syst Video Tech 16(1):82–91

Divakaran A (2009) Multimedia content analysis: theory and applications. Springer, Boston

Dugad R, Ratakonda K, Ahuja N (1998) Robust video shot change detection. IEEE Workshop on Multimedia Signal Processing, December 1998

Ekin A, Tekalp AM, Mehrotra R (2003) Automatic soccer video analysis and summarization. IEEE Trans Image Process 12(7):796–807

Feng H, Fang W, Liu S, Fang Y (2005) A new general framework for shot boundary detection based on SVM. In Proc IEEE ICNN&B, IEEE Computer Society 2:1112–1117

Gao X, Tang X (2002) Unsupervised video-shot segmentation and model-free anchorperson detection for news video story parsing. IEEE Trans Circuits Syst Video Technol 12(9):765–776

Gong Y, Xu W (2007) Machine learning for multimedia content analysis. Springer, Secaucus

Hampapur A, Jain R, Weymouth T (1994) Digital video segmentation. Proceedings of the ACM Multimedia 94, San Francisco, CA, October, pp 357–364

Han B, Hu Y, Wang G, Wu W, Yoshigahara T (2007) Enhanced sports video shot boundary detection based on middle-level features and a unified model. IEEE Trans Consumer Electron 53(3):1168–1176

Joyce RA, Liu B (2006) Temporal segmentation of video using frame and histogram space. IEEE Trans Multimed 8(1):130–140

Kasturi R, Jain R (1991) Dynamic vision. In: Kasturi R, Jain R (eds) Computer vision: principles. IEEE Computer Society Press, Washington, DC

Kawai Y, Sumiyoshi H, Yagi N (2007) Shot boundary detection at TRECVID 2007. Proceedings of the TRECVID Workshop 2007. http://www-nlpir.nist.gov/projects/tvpubs/tv7.papers/nhk.pdf

Little TDC, Ahanger G, Folz RJ, Gibbon JF, Reeve FW, Schelleng DH, Venkatesh D (1993) A digital on-demand video service supporting content-based queries. In: Proceedings of the ACM Multimedia 93, Anaheim, CA, August, pp 427–436

Matsumoto K, Naito M, Hoashi K, Sugaya F (2006) SVM-based shot boundary detection with a novel feature. In Proceedings of the IEEE international conference multimedia and expo 2006, pp 1837–1840

Nagasaka A, Tanaka Y (1992) Automatic video indexing and full-video search for object appearances. In: Knuth E, Wegner L (eds) Visual database systems II. Elsevier, Amsterdam, pp 113–127

Nam J, Tewfik AH (2005) Detection of gradual transitions in video sequences using B-spline interpolation. IEEE Trans Multimed 7(4):667–679

Ngo CW (2003) A robust dissolve detector by support vector machine. In: Proceedings of the ACM international conference multimedia 2003, ACM, New York, pp 283–286

Shahraray B (1995) Scene change detection and content-based sampling of video sequences. In: Digital video compression: algorithms and technologies, In: Arturo R, Robert S, Edward D (eds) Proceedings of the SPIE 2419:2–13

Swanberg D, Shu CF, Jain R (1993) Knowledge-guided parsing and retrieval in video databases. In: Wayne N (ed) Storage and retrieval for image and video databases, Proceeding of the SPIE 1908:173–187

Truong BT, Dorai C, Venkatesh S (2000) New enhancements to cut, fade, and dissolve detection processes in video segmentation. In: Proceedings of the ACM multimedia 2000, ACM, New York, pp 219–227

Ueda H, Miyatake T, Yoshizawa S (1991) IMPACT: An interactive natural motion picture dedicated multimedia authoring system. In: Proceedings of CHI 1991 (New Orleans, Louisiana, April–May, 1991) ACM, New York, pp 343–350

Wang Y, Liu Z, Huang JC (2000) Multimedia content analysis: using both audio and visual clues. IEEE Signal Process Mag 17:12–36

Yuan J, Wang H, Xiao L, Zheng W, Li J, Lin F, Zhang B (2007) A formal study of shot boundary detection. IEEE Trans Circ Syst Video Technology 17(2):168–186

Yusoff Y, Christmas W, Kittler J (2000) Video shot cut detection using adaptive thresholding. Proceedings of the 11th British machine vision conference (BMVC2000)

Zabih R, Miller J, Mai, K (1993) A feature-based algorithm for detecting and classifying scene breaks. Proceedings of the ACM Multimedia 95, San Francisco, CA, November, 1993, pp 189–200

Zabih R, Miller J, Mai K (1999) A feature-based algorithm for detecting and classifying production effects. Multimed Syst 7(2):119–128

Zhang HJ, Kankanhalli A, Smoliar SW (1993) Automatic partitioning of full-motion video. Multimed Syst 1(1):10–28