Multimedia Systems (1993) 1:10–28

Multimedia Systems

Automatic partitioning of full-motion video

HongJiang Zhang, Atreyi Kankanhalli, Stephen W. Smoliar

Institute of Systems Science, National University of Singapore, Heng Mui Keng Terrace, Kent Ridge, Singapore 0511, Republic of Singapore
e-mail: zhj@iss.nus.sg

Received January 10, 1993/Accepted April 10, 1993

Abstract. Partitioning a video source into meaningful segments is an important step for video indexing. We present a comprehensive study of a partitioning system that detects segment boundaries. The system is based on a set of difference metrics and it measures the content changes between video frames. A twin-comparison approach has been developed to solve the problem of detecting transitions implemented by special effects. To eliminate the false interpretation of camera movements as transitions, a motion analysis algorithm is applied to determine whether an actual transition has occurred. A technique for determining the threshold for a difference metric and a multi-pass approach to improve the computation speed and accuracy have also been developed.

Key words: Image difference analysis – Video partitioning – Video indexing – Multimedia

1 Introduction

Advances in multimedia technologies have now demonstrated that video in computers is a very important and common medium for applications as varied as education, broadcasting, publishing, and military intelligence (Mackay and Davenport 1989). The value of video is partially due to the fact that significant information about many major aspects of the world can only be successfully managed when presented in a time-varying manner. However, the effective use of video sources is seriously limited by a lack of viable systems that enable easy and effective organization and retrieval of information from those sources. Also, the time-dependent nature of video makes it a very difficult medium to represent and manage. Thus, much of the vast quantity of video that has been acquired sits on a shelf for future use without indexing (Nagasaka and Tanaka 1991). This is due to the fact that indexing requires an operator to view the entire video package and to assign index terms manually to each of its scenes. Between the abundance of unindexed video and the lack of sufficient manpower and time, such an approach is simply not feasible. Without an index, information retrieval from video requires one to view the source during a sequential scan, but this process is slow and unreliable, particularly when compared with analogous retrieval techniques based on text. Unfortunately, when it comes to the problem of developing new techniques for indexing and searching of video sources, the intuitions we have acquired through the study of information retrieval do not translate very well into non-text media.

Clearly, research is required to improve this situation, but relatively little has been achieved towards the development of video editing and browsing tools (Mackay and Davenport 1989; Tonomura 1991). A growing interest in digital video has led to new approaches to image processing based on the creation and analysis of compressed data (Netravali and Haskell 1988; Le Gall 1991) but these activities are in their earliest stages. The Video Classification Project at the Institute of Systems Science (ISS) of the National University of Singapore (NUS) is an effort to change this situation. It aims to develop an intelligent system that can automatically classify the content of a given video package. The tangible result of this classification will consist of an index structure and a table of contents, both of which can be stored as part of the package. In this way a video package will become more like a book to anyone interested in accessing specific information.

When text is indexed, words and phrases are used as index entries for sentences, paragraphs, pages or documents. This process only selects certain key words or phrases as index terms. Similarly, a video index will require, apart from the entries used in text, key frames or frame sequences as entries for scenes or stories. Therefore, automated indexing will require the support of tools which can *detect* and *isolate* such meaningful segments in any video source. Content analysis can then be performed on individual segments in order to identify appropriate index terms.

In our study, a segment is defined as a single, uninterrupted camera shot. This reduces the partitioning task to detecting the boundaries between consecutive camera shots. The simplest transition is a *camera break*. Figure 1 illustrates a sequence of four consecutive video frames with a camera break occurring between the second and third frames. The

Correspondence to: H.J. Zhang

Fig. 1a–d. Four frames across a camera break from a documentary video. The first two frames are in the first camera shot, and the third and fourth frame belong to the second camera shot. There are significant content changes between the second and third frames

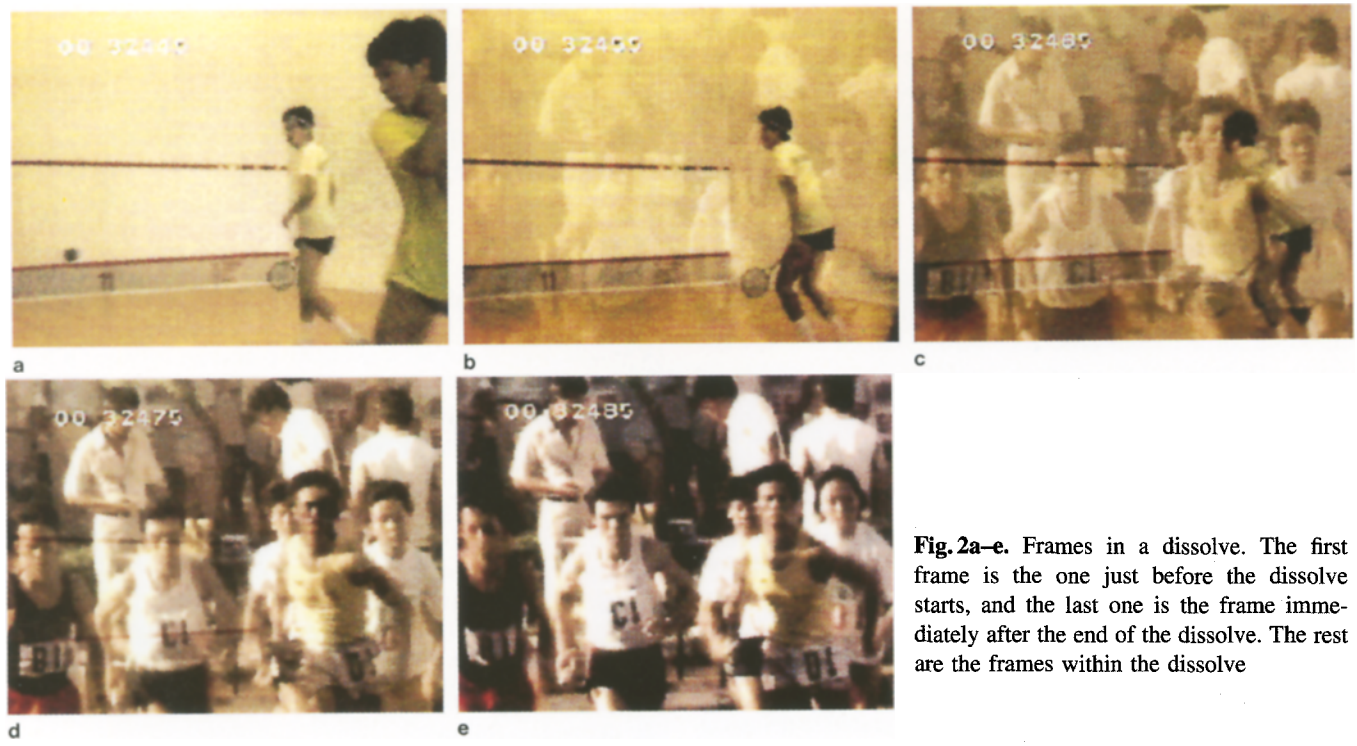

Fig. 2a–e. Frames in a dissolve. The first frame is the one just before the dissolve starts, and the last one is the frame immediately after the end of the dissolve. The rest are the frames within the dissolve

significant qualitative difference in content is readily apparent. If that difference can be expressed by a suitable metric, then a segment boundary can be declared whenever that metric exceeds a given threshold. Hence, establishing such metrics and techniques for applying them is the first step in our efforts to develop tools for the automatic partitioning of video packages.

The segmentation problem is also important for applications other than indexing. It is a key process in video editing, where it is generally called *scene change detection*. It also figures in the motion compensation of video for compression, where motion vectors must be computed *within* segments, rather than *across* segment boundaries (Liou 1991). However, accuracy is not as crucial for either of these appli-

cations; for example, in compression, false positives only increase the number of reference frames. By contrast, in video segmentation for indexing, such false positives would have to be corrected by manual intervention. Therefore, high accuracy is a more important requirement in automating the process.

A camera break is the simplest transition between two shots. More sophisticated techniques include dissolve, wipe, fade-in, and fade-out (Bordwell and Thompson 1993). Such special effects involve much more gradual changes between consecutive frames than does a camera break. Figure 2 shows five frames of a dissolve from a documentary video: the frame just before the dissolve begins, three frames within the dissolve, and the frame immediately after the dissolve. This sequence illustrates the gradual change that downgrades the power of a simple difference metric and a single threshold for camera break detection. Indeed, most changes are even more gradual, since dissolves usually last more than ten frames. Furthermore, the changes introduced by camera movement, such as pan and zoom, may be of the same order as that introduced by such gradual transitions. This further complicates the detection of the boundaries of camera shots, since the artefacts of camera movements must be distinguished from those of gradual shot transitions.

While there has been related work on camera break detection by other researchers (Tonomura 1991; Nagasaka and Tanaka 1991; Ruan et al. 1992), few experimental data have been reported. Also, the implementation of the difference metrics for practical video processing and an automatic determination of the cutoff threshold have not yet been addressed in the current literature. This paper presents a comprehensive experimental study of three difference metrics for video partitioning: pair-wise comparison of pixels, comparison of pixel histograms, and "likelihood ratio" comparison, where the likelihood ratio metric has not been previously discussed. It also discusses the automatic selection of the threshold for a chosen difference metric, which is a key issue in obtaining high segment accuracy, and a multi-pass technique that upgrades performance efficiency.

So far, there has been no report on algorithms that successfully detect gradual transitions. As a major contribution to the video segmentation problem, we suggest that this problem can be solved by a novel technique called *twin-comparison*. This technique first uses a difference metric with a reduced threshold to detect the potential frames where a gradual transition can occur; then the difference metric is used to compare the first potential transition frame with each following frame until the accumulated difference exceeds a second threshold. This interval is then interpreted to delineate the start and end frames of the transition. Motion detection and analysis techniques are then applied to distinguish camera movements from such gradual transitions. Experiments show that the approach is very effective and that it achieves a very high level of accuracy.

Before discussing the detection of gradual transitions, we first present a set of the difference metrics and their applications in detecting camera breaks in Sect. 2. In Sect. 3 we discuss how the use of a difference metric can be adapted to accommodate gradual transitions and present the twin-comparison approach. A motion analysis technique for eliminating false detection of transitions resulting from the artefacts of camera movements is presented in Sect. 3. This is followed by a discussion of automatic selection of threshold values and the multi-pass approach to improving computational efficiency in Sect. 4. Algorithmic implementation, evaluation of algorithm performance, and the application of these techniques to some "real-world" video examples are given in Sect. 5. Finally, Sect. 6 presents our conclusions and a discussion of anticipated future work.

2 Difference metrics for video partitioning

As observed in Sect. 1, the detection of transitions involves the quantification of the difference between two image frames in a video sequence. To achieve, this, we need first to define a suitable metric, so that a segment boundary can be declared whenever that metric exceeds a given threshold. Difference measures used to partition video can be divided into two major types: the pair-wise comparison of pixels or blocks, and the comparison of the histograms of pixel values. The blocks are compared with the *likelihood ratio*, a statistic calculated over the area occupied by a given super-pixel (Kasturi and Jain 1991). These metrics can be implemented with a variety of different modifications to accommodate the idiosyncrasies of different video sources (Nagasaka and Tanaka 1991; Ruan et al. 1992) and have been used successfully in camera break detection. Details will now be discussed.

2.1 Pair-wise comparison

A simple way to detect a qualitative change between a pair of images is to compare the corresponding pixels in the two frames to determine how many pixels have changed. This approach is known as *pair-wise comparison*. In the simplest case of monochromatic images, a pixel is judged as changed if the difference between its intensity values in the two frames exceeds a given threshold t. This metric can be represented as a binary function $DP_i(k,l)$ over the domain of two-dimensional coordinates of pixels, (k,l), where the subscript i denotes the index of the frame being compared with its successor. If $P_i(k,l)$ denotes the intensity value of the pixel at coordinates (k,l) in frame i, then $DP_i(k,l)$ may be defined as follows:

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

The pair-wise segmentation algorithm simply counts the number of pixels changed from one frame to the next according to this metric. A segment boundary is declared if more than a given percentage of the total number of pixels

(given as a threshold T) have changed. Since the total number of pixels in a frame of dimensions M by N is $M*N$, this condition may be represented by the following inequality:

$$\frac{\sum_{k,l=1}^{M,N} DP_i(k,l)}{M*N} * 100 > T \,. \tag{2}$$

A potential problem with this metric is its sensitivity to camera movement. For instance, in the case of camera panning, a large number of objects will move in the same direction across successive frames; this means that a large number of pixels will be judged as changed even if the pan entails a shift of only a few pixels. This effect may be reduced by the use of a smoothing filter: before comparison each pixel in a frame is replaced with the mean value of its nearest neighbours. (We are currently using a 3×3 window centred on the pixel being "smoothed" for this purpose.) This also filters out some noise in the input images.

2.2 Likelihood ratio

To make the detection of camera breaks more robust, instead of comparing individual pixels we can compare corresponding *regions* (blocks) in two successive frames on the basis of second-order statistical characteristics of their intensity values. One such metric for comparing corresponding regions is called the *likelihood ratio* (Kasturi and Jain 1991). Let m_i and m_{i+1} denote the mean intensity values for a given region in two consecutive frames, and let S_i and S_{i+1} denote the corresponding variances. The following formula computes the likelihood ratio and determines whether or not it exceeds a given threshold t:

$$\frac{\left[\frac{S_i+S_{i+1}}{2} + \left(\frac{m_i-m_{i+1}}{2}\right)^2\right]^2}{S_i * S_{i+1}} > t \,. \tag{3}$$

Camera breaks can now be detected by first partitioning the frame into a set of sample areas. Then a camera break can be declared whenever the total number of sample areas whose likelihood ratio exceeds the threshold is sufficiently large (where "sufficiently large" will depend on how the frame is partitioned). An advantage that sample areas have over individual pixels is that the likelihood ratio raises the level of tolerance to slow and small object motion from frame to frame. This increased tolerance makes it less likely that effects such as slow motion will mistakenly be interpreted as camera breaks.

The likelihood ratio also has a broader dynamic range than does the percentage used in pair-wise comparison. This broader range makes it easier to choose a suitable threshold value t for distinguishing changed from unchanged sample areas. A potential problem with the likelihood ratio is that if two sample areas to be compared have the same mean and variance, but completely different probability density functions, no change will be detected. Fortunately, such a situation is very unlikely.

2.3 Histogram comparison

An alternative to comparing corresponding pixels or regions in successive frames is to compare some feature of the entire image. One such feature that can be used in segmentation algorithms is a histogram of intensity levels. The principle behind this algorithm is that two frames having an unchanging background and unchanging objects will show little difference in their respective histograms. The histogram comparison algorithm should be less sensitive to object motion than the pair-wise pixel comparison algorithm, since it ignores the spatial changes in a frame. One could argue that there may be cases in which two images have similar histograms but completely different content. However, the probability of such an event is sufficiently low that, in practice, we can tolerate such errors.

Let $H_i(j)$ denote the histogram value for the ith frame, where j is one of the G possible grey levels. (The number of histogram bins can be chosen on the basis of the available grey-level resolution and the desired computation time.) Then the difference between the ith frame and its successor will be given by the following formula:

$$SD_i = \sum_{j=1}^{G} |H_i(j) - H_{i+1}(j)| \,. \tag{4}$$

If the overall difference SD_i is larger than a given threshold T, a segment boundary is declared. To select a suitable threshold, SD_i can be normalized by dividing it by the product of G and $M*N$, the number of pixels in the frame.

Figure 3 shows grey-level histograms of the first three images shown in Fig. 1. Note the difference between the histograms across the camera break between the second and the third frames, while the histograms of the first and second frames are almost identical. Figure 4 illustrates the application of histogram comparison to a documentary video. The graph displays the sequence of SD_i values defined by Eq. 4 between every two consecutive frames over an excerpt from this source. Equation 4 was applied to grey-level intensities computed from the intensities of the three colour channels by the National Television System Committee (NTSC) conversion formula (Sect. 5.1). The graph exhibits two high pulses that correspond to two camera breaks. If an appropriate threshold is set, the breaks can be detected easily.

Equation 4 can also be applied to histograms of individual colour channels. A simple but effective approach is to use colour histogram comparison (Nagasaka and Tanaka 1991; Zhang et al. 1993): Instead of grey levels, j in Eq. 4 denotes a code value derived from the three colour intensities of a pixel. Of course, if 24 bits of colour data were translated into a 24-bit code word, that would create histograms with 2^{24} bins, which is clearly unwieldy. Consequently, only the two or three most significant bits of each colour component tend to be used to compose a colour code. A 6-bit code, providing 64 bins, has been shown to give sufficient accuracy. This approach is also more efficient for colour source material because it eliminates the need to first convert the colour intensities into grey levels.

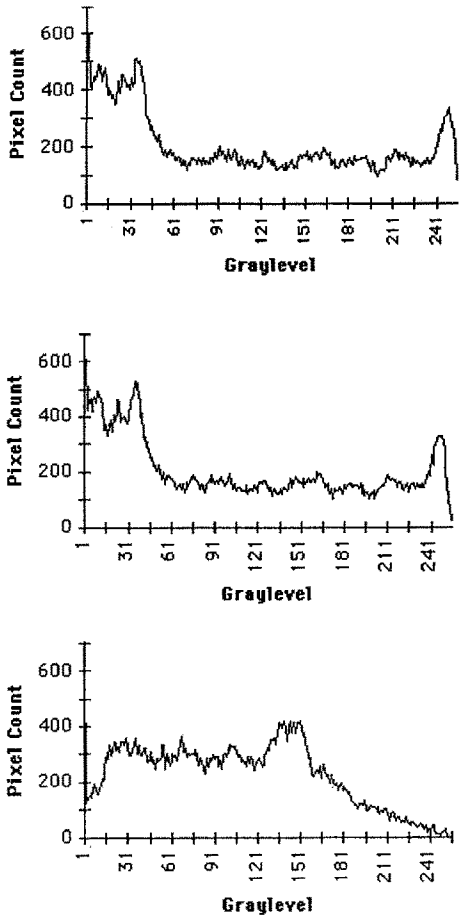

Fig. 3. Histograms of grey-level pixel values corresponding to the first three frames shown in Fig. 1

Nagasaka and Tanaka (1991) have also used the following χ^2-test equation,

$$SD_i = \sum_{j=1}^{G} \frac{|H_i(j) - H_{i+1}(j)|^2}{H_{i+1}(j)} \tag{5}$$

to make the histogram comparison reflect the difference between two frames more strongly. However, our experiments show that while this equation enhances the difference between two frames across a camera break, it also increases the difference between frames representing small changes due to camera or object movements. Therefore, the overall performance is not necessarily better than that achieved by using Eq. 4, while Eq. 5 also requires more computation time.

2.4. Limitations

In addition to the weaknesses of each of the individual metrics that have already been cited, all three types of difference metric face a severe problem if there are moving objects (of either large size or of high speed) or a sharp illumination change between two frames in a common shot, resulting in a false detection of a camera break. Flashing lights and flickering objects (such as video monitors) are common sources of errors, as will be seen in Sect. 5.

When the illumination does not change over the entire frame, the problem can be overcome by a robust approach developed by Nagasaka and Tanaka (1991) to accommodate momentary noise. It is based on the assumption that such noise usually influences no more than half of an entire frame. Therefore, a frame can be divided into a 4×4 grid of 16 rectangular regions; and, instead of comparing entire frames, corresponding regions are compared. This yields 16 difference values, and the camera break detection is only based on the sum of the eight lowest difference values. Such a selection of data is meant to eliminate the errors introduced by momentary noise. We plan to test this technique, but we anticipate that its performance might be made more robust by taking the sum of the median eight difference values (eliminating the four extremes from both ends).

3 Gradual transition detection

We now present the twin-comparison approach that adapts a difference metric to accommodate gradual transitions, which is our main contribution to automatic video partitioning. This discussion will be based on the histogram-comparison difference metric. We also discuss how a technique based on motion detection and analysis can be used to identify intervals of camera movement that may be confused with gradual transitions between camera shots.

3.1 The twin-comparison approach for detecting special effects

As one can observe from Fig. 4, the graph of the frame-to-frame histogram differences for a sequence exhibits two high pulses that correspond to two camera breaks. It is easy to select a suitable cutoff threshold value (such as 50) for detecting these camera breaks. However, the inset of this graph displays another sequence of pulses the values of which are higher than those of their neighbous but are significantly lower than the cutoff threshold. This inset displays the difference values for the dissolve sequence shown in Fig. 2 and illustrates why a simple application of this difference metric is inadequate.

The simplest approach to this problem would be to lower the threshold. Unfortunately, lower thresholds cannot be effectively employed, because the difference values that occur during the gradual transition implemented by a special effect may be smaller than those that occur between the frames within a camera shot. For example, object motion, camera panning, and zooming also entail changes in the computed difference value. If the cutoff threshold is too low, such changes may easily be registered as "false positives". The problem is that a single threshold value is being made to account for all segment boundaries, regardless of context. This appears to be asking too much of a single number, so a new approach has to be developed.

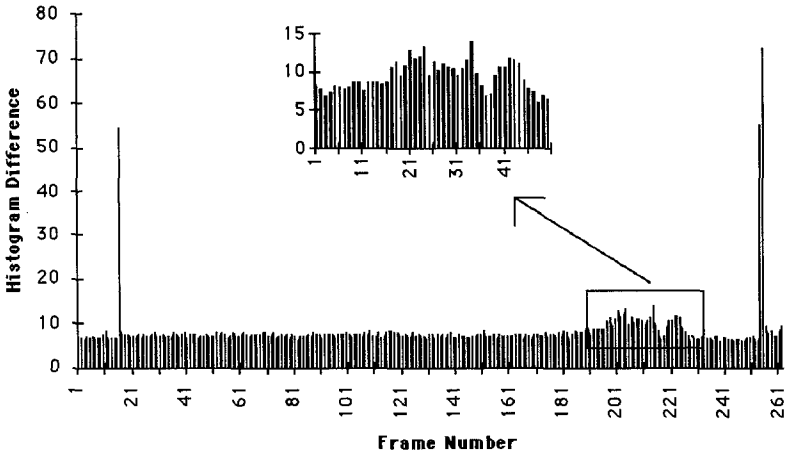

Fig. 4. A sequence of frame-to-frame histogram differences obtained from a documentary video, where differences corresponding to both camera breaks and transitions implemented by special effects can be observed

In Fig. 2 it is obvious that the first and the last frame are different, even if all consecutive frames are very similar in content. In other words the difference metric applied in Sect. 2.3 with the threshold derived from Fig. 4 would still be effective were it to be applied to the first and the last frame directly. Thus, the problem becomes one of detecting these first and last frames. If they can be determined, then each of them may be interpreted as a segment boundary, and the period of gradual transition can be isolated as a segment unto itself. The inset of Fig. 4 illustrates that the difference values between most of the frames during the dissolve are higher, although only slightly, than those in the preceding and following segments. What is required is a threshold value that will detect a dissolve *sequence* and distinguish it from an ordinary camera shot. A similar approach can be applied to transitions implemented by other types of special effects. We shall present this *twin-comparison* approach in the context of an example of dissolve detection using Eq. 4 as the difference metric.

Twin-comparison requires the use of two cutoff thresholds: T_b is used for camera break detection in the same manner as was described in Sect. 2. In addition, a second, lower, threshold T_s is used for special effect detection. The detection process begins by comparing consecutive frames using a difference metric such as Eq. 4. Whenever the difference value exceeds threshold T_b, a camera break is declared, e.g., F_B in Fig. 5a. However, the twin-comparison also detects differences that are smaller than T_b but larger than T_s. Any frame that exhibits such a difference value is marked as the potential start (F_s) of a gradual transition. Such a frame is labelled in Fig. 5a. This frame is then compared to subsequent frames, as shown in Fig. 5b. This is called an *accumulated comparison* since, during a gradual transition, this difference value will normally increase. The end frame (F_s) of the transition is detected when the difference between consecutive frames decreases to less than T_s, while the accumulated comparison has increased to a value larger than T_b.

Note that the accumulated comparison is only computed when the difference between consecutive frames exceeds T_s. If the consecutive difference value drops below T_s before the accumulated comparison value exceeds T_b, then the potential start point is dropped and the search continues for other

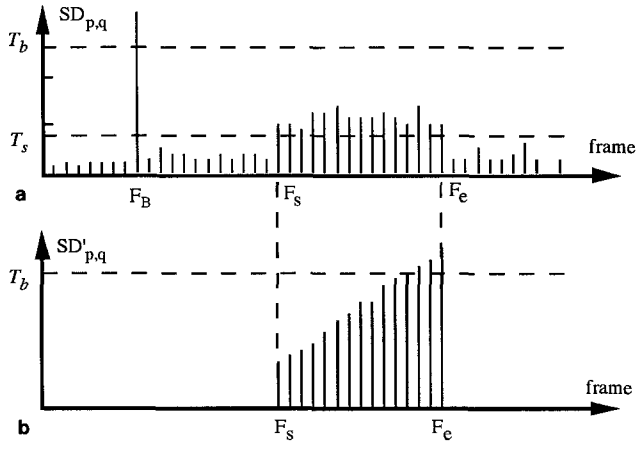

Fig. 5a,b. Illustration of twin-comparison. $SD_{p,q}$, the difference between consecutive frames defined by the difference metric; $SD'_{p,q}$, the accumulated difference between the current frame and the potential starting frame of a transition; T_s, the treshold used to detect the starting frame (F_s) of a transition; T_b, the thresold used to detect the ending frame (F_e) of a transition. T_b is also used to detect camera breaks and F_B is such a camera break. $SD'_{p,q}$ is only calculated when $SD_{p,q} > T_s$

gradual transitions. The key idea of twin-comparison is that two distinct threshold conditions be satisfied at the same time. Furthermore, the algorithm is designed in such a way that gradual transitions are detected *in addition* to ordinary camera breaks.

A problem with twin-comparison is that there are some gradual transitions during which the consecutive difference value *does* fall below T_s. This problem is solved by permitting the user to set a tolerance value that allows a number (such as two or three) of consecutive frames with low difference values before rejecting the transition candidate. This approach has proven to be effective when tested on real video examples.

3.2 Distinguishing special effects from camera movements

Given the ability to detect gradual transitions such as those that implement special effects, one must distinguish changes associated with those transitions from changes that are intro-

duced by camera panning or zooming. Changes due to camera movements tend to induce successive difference values of the same order as those of gradual transitions, so that the problem cannot be resolved by introducing yet another cut-off threshold value. Instead, it is necessary to detect patterns of image motion that are induced by camera movement.

The specific feature that serves to detect camera movements is *optical flow*, a technique rooted in computer vision. The optical flow fields resulting from panning and zooming are illustrated in Fig. 6. In contrast, transitions implemented by special effects, such as dissolve, fade-in and fade-out, will not introduce such motion fields. Therefore, if such motion vector fields can be detected and analysed, changes introduced by camera movements can be distinguished from those due to special-effect transitions.

Optical flow computation involves representing the difference between two consecutive frames as a set of motion vectors. As is illustrated in Fig. 6a, during a camera pan these vectors will predominantly have the same direction. (Clearly, if there is also object movement in the scene, not all vectors need share this property.) Thus, the distribution of motion vectors in an entire frame resulting from a camera panning should exhibit a single strong modal value that corresponds to the movement of the camera. In other words, most of the motion vectors will be parallel to the modal vector. This leads to

$$\sum_{k}^{N} |\theta_k - \theta_m| \leq \Theta_p \tag{6}$$

where θ_k is the direction of motion vector k, θ_m is the direction of the modal vector, and N is the total number of motion vectors in a frame. $|\theta_k - \theta_m|$ is zero when the two vectors are exactly parallel. Equation 6 thus counts the total variation in direction of all motion vectors from the direction of the modal vector, and a camera pan is declared if this variation is smaller than Θ_p. Ideally, Θ_p should be zero, but we use a non-zero threshold to accommodate errors such as those that may be introduced by object motion.

In the case of zooming, the field of motion vectors has a minimum value at the focus centre, focus of expansion (FOE) in the case of zoom out, or focus of contraction (FOC) in the case of zoom in. Indeed, if the focus centre is located in the centre of the frame and there is no object movement, then the mean of all the motion vectors will be the zero vector. Unfortunately, determining an entire frame of motion vectors with high spatial resolution is a very time consuming process, so locating a focus centre is no easy matter.

Since we are only interested in determining that a zoom takes place over a sequence of frames, it is not necessary to locate the focus centre. If we assume that, in general, the focus centre of a camera zoom will lie within the boundary of a frame, we may apply a simpler vector comparison technique. In this approach we compare the vertical components of the motion vectors for the top and bottom rows of a frame, since during a zoom these vertical components will have opposite signs. Mathematically, this means that in every column the magnitude of the difference between these vertical components will always exceed the magnitude of both components. That is,

$$\left|v_k^{\text{top}} - v_k^{\text{bottom}}\right| \geq \max\left(\left|v_k^{\text{top}}\right|, \left|v_k^{\text{bottom}}\right|\right) . \tag{7.1}$$

One may again modify this "always" condition with a tolerance value, but this value can generally be low, since there tends to be little object motion at the periphery of a frame. The horizontal components of the motion vectors for the left-most and right-most columns can then be analysed the same way. That is, the vectors of two blocks located at the left-most and right-most columns but at the same row will satisfy the following condition:

$$\left|u_k^{\text{top}} - u_k^{\text{bottom}}\right| \geq \max\left(\left|u_k^{\text{top}}\right|, \left|u_k^{\text{bottom}}\right|\right) . \tag{7.2}$$

A zoom is said to occur when both conditions 7.1 and 7.2 are satisfied for the majority of the motion vectors.

The field of motion vectors that is required for this analysis can be computed by the *block-matching* algorithm commonly used for motion compensation in video compression (Netravali and Haskell 1988). This algorithm requires the partitioning of the frame into blocks, and motion vectors are computed for each block by finding the minimum value of a cost function over a set of trial vectors. Suppose each block is M lines high and N pixels wide; and, within a block, let $b(m, n)$ denote the intensity of the pixel with coordinates (m, n). Then one may define a cost function for block k between two consecutive frames, i and $i + 1$, in terms of an error-power function F as follows:

$$P_i(k) = \sum_{m}^{M} \sum_{n}^{N} F[b_i(m, n) - b_{i+1}(n, m)] . \tag{8}$$

Given a set of trial vectors, the vector that minimizes $P_i(k)$ can be assigned as the motion vector for block k. The calculation time to find a motion vector depends mainly on the number of trial vectors, which, in turn, is based on the possible maximum velocity of camera movement and the resolution of the field of motion vectors depends on the size of the blocks compared.

Apart from the approach just presented, one can also identify camera motion based on sophisticated optical flow analyses (Kasturi and Jain 1991). However, such a technique will require a much greater density of motion vectors (one vector per pixel, in principle). Computing such a resolution of motion vectors is very time consuming, requiring either iterative refinement of a gradient-based algorithm (Horn and Schunck 1981) or the construction of a hierarchical framework of cross-correlation (Anandan 1989). More importantly, although optical flow calculations have been under investigation by many researchers, besides the difficulties in obtaining high accuracy and robustness, the recovery of motion from optical flow and the application of smoothness constraints are still research issues (Kasturi and Jain 1991). Therefore, instead of waiting for more reliable algorithms for recovering camera motion from optical flow,

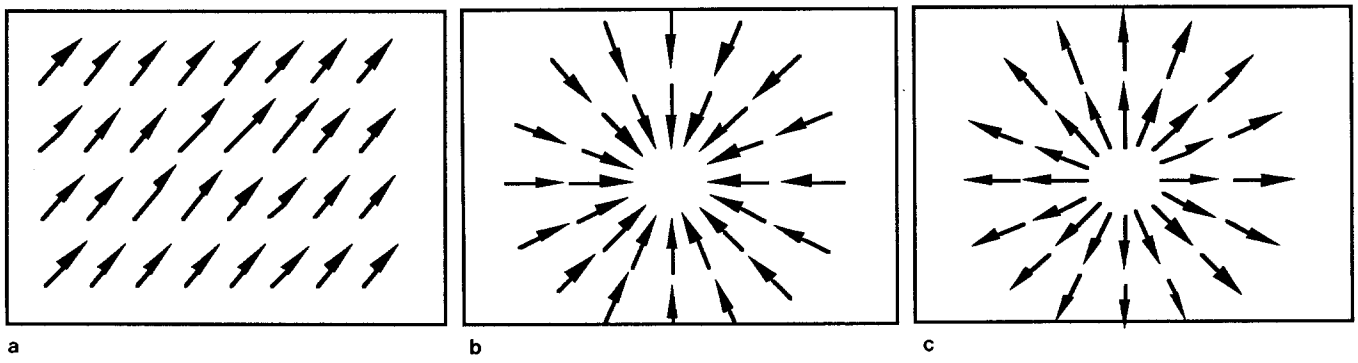

Fig. 6a–c. Motion vector patterns resulting from camera panning and zooming. **a** ↙ Camera panning direction. **b** Camera zoom-out. **c** Camera zoom-in

our current, more limited, motion vector analysis algorithm can provide reasonable accuracy, as one can see from the experimental results presented in Sect. 5.

In spite of the computational expense, there are still advantages to working with the block-matching algorithm. In the future we plan to make use of motion vectors retrieved from files of video that have been compressed by the motion-compensation technique. Such files can be computed already during a real-time scan of video input by video compression hardware, such as an MPEG (Moving Pictures Experts Group) chip (Ang et al. 1991).

4 Applying the comparison techniques

Before the difference metrics for transition detection can be applied to full-motion video, two practical issues have to be considered. First, how does one select the appropriate threshold values for a given video source and difference metric? Secondly, comparing each pair of consecutive frames of a video source (54 000 frames for half an hour's worth of material) is a very time consuming process; hence, a scheme to speed up the partitioning process needs to be considered. We discuss the solutions to these two issues.

4.1 Threshold selection

Selection of appropriate threshold values is a key issue in applying the segmentation algorithms described in Sect. 2 and 3. Thresholds must be assigned that tolerate variations in individual frames while still ensuring a desired level of performance. A "tight" threshold makes it difficult for "impostors" to be falsely accepted by the system, but at the risk of falsely rejecting true transitions. Conversely, a "loose" threshold enables transitions to be accepted consistently, at the risk of falsely accepting "impostors". In order to achieve high accuracy in video partitioning, an appropriate threshold must be found.

The threshold t, used in pair-wise comparison for judging whether a pixel or super-pixel has changed across successive frames, can be easily determined experimentally and it does not change significantly for different video sources. However, experiments have shown that the threshold T_b for determining a segment boundary using any of the algorithms varies from one video source to another. For instance, "camera breaks" in a cartoon film tend to exhibit much larger frame-to-frame differences than those in a "live" film. Obviously, the threshold to be selected must be based on the distribution of the frame-to-frame differences of the video sequence.

Considerable research has been done on the selection of thresholds for the spatial segmentation of static images. A good summary can be found in *Digital Picture Processing* by Rosenfeld and Kak (1982). Typically, selecting thresholds for such spatial segmentation is based on the histogram of the pixel values of the image. The conventional approaches include the use of a single threshold, multiple thresholds and variable thresholds. The accuracy of single threshold selection depends upon whether the histogram is bimodal, while multiple threshold selection requires clear multiple peaks in the histogram. Variable threshold selection is based on local histograms of specific regions in an image. In spite of this variety of techniques, threshold selection is still a difficult problem in image processing and is most successful when the solution is application dependent. In order to set an appropriate threshold for temporal segmentation of video sequences, we draw upon the same feature, i.e., the histogram of the frame-to-frame differences. Thus, it is necessary to know the distribution of the frame-to-frame differences across camera breaks and gradual transitions.

The automatic selection of threshold T_b is based on the normalized frame-to-frame differences over an entire given video source. The dashed curve (M) in Fig. 7 shows a typical distribution of difference values. This particular example is based on the difference metric for comparison of colour code histograms obtained from a documentary video. The range of difference values is given on the horizontal axis, and the frequency of occurrence of each difference value is represented as a percentage of the total number of frame-to-frame differences on the vertical axis. This particular distribution exhibits a high and sharp peak on the left corresponding to a large number of consecutive frames that have a very small difference between them. The long tail to the right corresponds to the small number of consecutive frames between which a significant difference occurs. Because this histogram has only a single modal point, the approaches for threshold

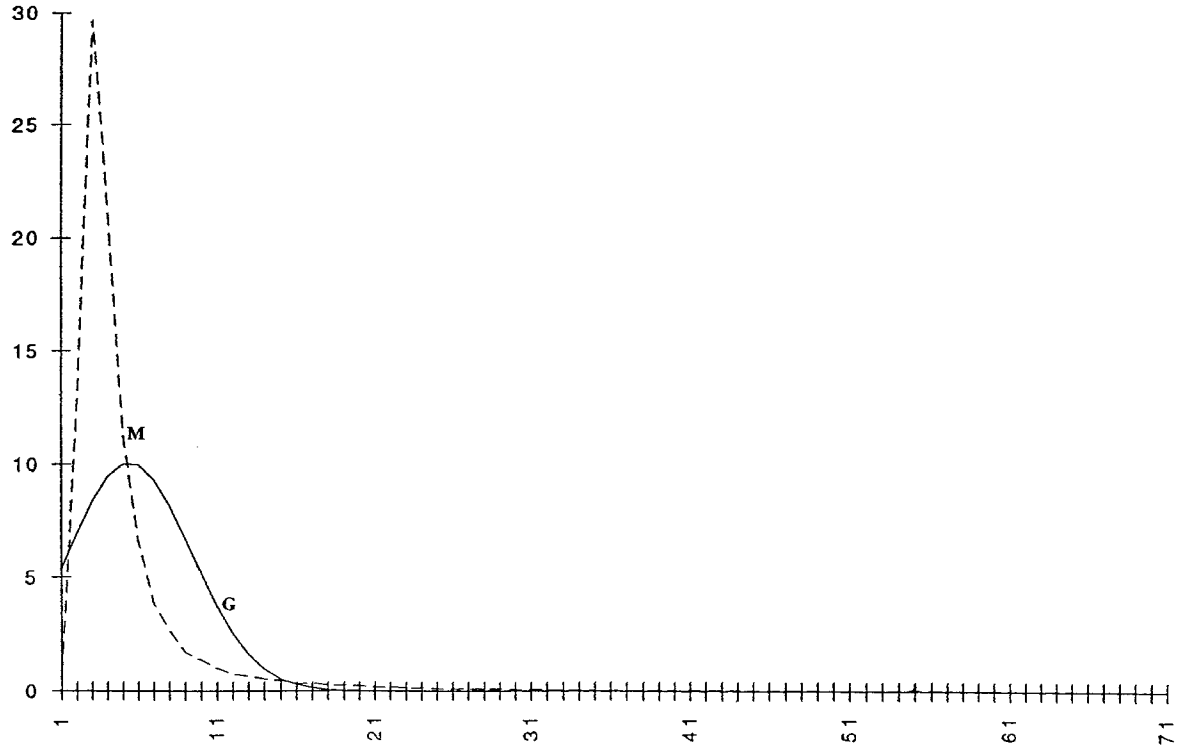

Fig. 7. M, distribution of computed frame-to-frame differences based on a colour code histogram for all frames of the documentary video; G, Gaussian distribution derived from the mean and variance of distribution M

selection already mentioned are not applicable. To solve this problem, we propose an alternative approach to selecting the threshold value T_b as follows.

If there is no camera shot change or camera movement in a video sequence, the frame-to-frame difference value can only be due to three sources of noise: noise from digitizing the original analog video signal, noise introduced by video production equipment, and noise resulting from the physical fact that few objects are perfectly still. All three sources of noise can be assumed to be Gaussian. Thus, the distribution of frame-to-frame differences can be decomposed into a sum of two parts: the Gaussian noises and the differences introduced by camera breaks, gradual transitions, and camera movements. Obviously, differences due to noise have nothing to do with transitions. Statistically, the second sum accounts for less than 15% of the total number of frames in a documentary video that we have used as a source of experimental data.

Let σ be the standard deviation and μ the mean of the frame-to-frame differences. If the only departure from μ is due to Gaussian noise, then the probability integral

$$P(x) = \int_0^x \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \tag{9}$$

(taken from 0 since all differences are given as absolute values) will account for most of the frames within a few standard deviations of the mean value. In other words, the frame-to-frame differences from the non-transition frames will fall in the range of 0 to $\mu + \alpha\sigma$ for a small constant value α. For instance, if we chose $\alpha = 3$, Eq. 9 will account for 99.9% of all difference values. The black curve (G) in Fig. 7 shows the Gaussian distribution obtained from σ and μ for the frame-to-frame differences from which the M curve was calculated. Therefore, the threshold T_b can be selected as

$$T_b = \mu + \alpha\sigma \; . \tag{10}$$

That is, difference values that fall out of the range from 0 to $\mu + \alpha\sigma$ can be considered indicators of segment boundaries. From our experiments, the value α should be chosen between five and six when the histogram comparison metric is used. Under a Gaussian distribution, the probability that a non-transition frame will fall out of this range is practically zero.

For detecting gradual transitions, another threshold, T_s, defined in Fig. 5, also needs to be selected. Experiments have shown that T_s should be selected along the right slope of the M distribution shown in Fig. 7. Furthermore, T_s should generally be larger than the mean value of the frame-to-frame differences of the entire video package. After examining three documentary videos, we observed that T_s does not vary significantly; and a typical value usually lies between eight and ten.

4.2 Multi-pass approach

Once threshold values have been established, the partitioning algorithms can be applied in "delayed real-time". Only one pass through the entire video package is required to determine all the camera breaks. However, the delay can be quite substantial given 30 fps of colour source material. In

addition, such a single-pass approach has the disadvantage that it does not exploit any information other than the threshold values. Therefore, this approach depends heavily on the selection of those values.

A straightforward approach to the reduction of processing time is to lower the resolution of the comparison. This can be done in two ways – either spatially or temporally. In the spatial domain, one may sacrifice resolution by examining only a subset of the total number of pixels in each frame. However, this is clearly risky since if the subset is too small, the loss of spatial detail may result in a failure to detect certain segment boundaries. Also, resampling the original image may even increase the processing time, and that would run contrary to our goal of applying spatial resampling to reduce the processing time.

Alternatively, to sacrifice temporal resolution by examining fewer frames is a better choice, since in motion video temporal information redundancy is much higher than spatial information redundancy. For example, one could apply a "skip factor" of ten (examining only three frames/s of video time from a 30-frames/s source). This will reduce the number of comparisons (and, therefore, the associated processing time) by the same factor. An advantage of this approach is that it will also detect some of the transitions implemented by special effects as camera breaks, since the difference between two frames that are ten frames apart across such a transition could be larger than the threshold T_b. A drawback of this approach is that the accuracy of locating the camera break decreases with the same factor as the skip. Also, if the skip factor is too large, the change during a camera movement may be so great that it leads to a false detection of a camera break. (This was observed experimentally in a system that was limited to "grabbing" only one frame/s.)

To overcome the problems of low resolution in locating segment boundaries, we have developed a multi-pass approach that improves processing speed and achieves the same order of accuracy. In the first pass resolution is sacrificed temporally to detect *potential* segment boundaries. In this process twin-comparison for gradual transitions is not applied. Instead, a lower value of T_b is used; and all frames across which there is a difference larger than T_b are detected as potential segment boundaries. Due to the lower threshold and large skip factor, both camera breaks and gradual transitions, as well as some artefacts due to camera movement, will be detected; but any number of false detections will also be admitted, as long as no *real* boundaries are missed. In the second pass all computation is restricted to the vicinity of these potential boundaries. Increased resolution is used to locate all boundaries (both camera breaks and gradual transitions) more accurately. Also, motion analysis is applied to distinguish camera movements from gradual transitions.

With the multi-pass approach different detection algorithms can be applied in different passes to increase confidence in the results. For instance, for a given video package, we can apply either pair-wise or histogram comparison in the first, low resolution, pass with a large skip factor and a low value of T_b. Then, in the second pass, *both* comparison algorithms are applied independently to the potential boundaries detected by the first pass. The results from the two algorithms can then be used to verify each other, and positive results from both algorithms will have sufficiently high confidence to be declared as segment boundaries. As will be seen from the following experimental results, the multi-pass approach can achieve both high speed and accuracy.

Adaptive sampling in time can also be applied in the segmentation process to reduce processing time. That is, we only examine those frames with a high likelihood of crossing a segment boundary and skip the frames with a low likelihood. For instance, we can skip a number of frames after a camera shot boundary is detected since there are not likely to be two boundaries in very close succession. However, in general, it is difficult to obtain a priori knowledge about such likelihood. More importantly, we currently calculate the mean and variance for those frame-to-frame differences taken over an entire video sequence and sampled according to a constant skip factor; and those mean and variance values are used to determine the threshold for segment boundaries. That is, we need to calculate the threshold from the video sequence before the sequence is partitioned. An adaptive sampling technique that introduces variable skip factors, would destroy the statistical properties of those calculations, possibly resulting in an inadequate threshold. In addition, variable thresholds would be needed if adaptive sampling were to be applied, which would further complicate the processing. However, the detection process in the second pass (when the two-pass approach is used with the same metric in both passes) is, in fact, an adaptive sampling process in time, as only the frames close to the potential boundaries are examined in the second pass, while other frames between the potential boundaries are skipped. In this case, however, the threshold value is obtained from the first pass.

5 Implementation and evaluation

The partitioning algorithms described have been implemented and applied on a variety of video materials. The output of the segmentation process of a given video package is a sequence of segment boundaries. These boundaries are compared with those detected manually from the same video sources as a basis for evaluation. We shall now discuss the implementation of these algorithms, the experimental results, and the comparative value of each approach.

5.1 System and algorithm implementation

The implementation platform consists of a Macintosh IIfx connected to a Pioneer LD-V8000 video laser disc player, as shown in Fig. 8. The interface between the laser disc player and the computer is provided by a RasterOps Colorboard 364 video board, which combines the functions of a true colour frame buffer with hardware pan and zoom, "frame grabbing", and real-time video display at a rate of 30 frames/s. A software control panel for the video disc player has been developed and integrated with a user-friendly interface to control

frame searching, input of algorithm parameters, monitoring of the segmentation process and display and storage of segmentation results.

During segmentation, the frames to be compared are directly grabbed from the laser disc rather than stored as a file of compressed digital video. The frames are of resolution 278 × 208 with each pixel represented by 32 bits of data: 8 bits for each of the three colour components and the first 8 bits unused. The colour intensities can then be transformed into a single 8-bit value of grey level or a 6-bit colour code, composed from the most significant 2 bits of each of the three colour components, as illustrated in Fig. 9. If grey-level intensities are used in the segmentation, they are computed from the three colour components according to the NTSC standard:

$$I = 0.299R + 0.587G + 0.114B\ . \tag{11}$$

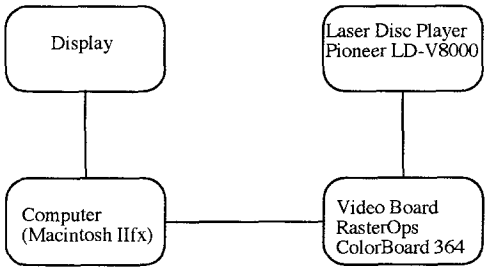

Fig. 8. Architecture of the automatic video segmentation system

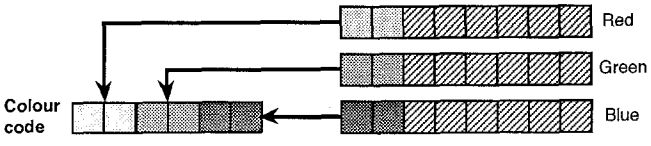

Fig. 9. Diagrammatic representation of colour code generation

In this formula R, G and B stand for the intensities of the red, green and blue components, respectively. This formula also gives the impact of individual colour changes on the overall grey level and indicates the significance of the green component. To increase the processing speed by avoiding the computation of this transformation, it is possible to compute differences using only the green component. However, this approach will miss any difference due to large changes in the red and blue components but a small change in the green component. Whether or not this information loss can be tolerated will depend on the nature of the video source.

Input to the segmentation system is provided by grabbing the frames for the sequence to be partitioned and digitizing each frame in the video board. The system first processes these input data by calculating the frame-to-frame differences between those frames determined by the skip factor; these difference values are then stored in an array. This array is then used to compute the mean and variance of those difference values. Then the segmentation threshold is determined from these two quantities, as described in Sect. 4.2. Finally, segment boundaries are detected using the difference values in the array.

5.2 Experimental results

The algorithms described in this paper have been tested on a number of video sequences, including documentaries, educational material, and cartoons. However, we shall only present the results of the partitioning of a 7-min animated cartoon and two documentaries: The first is a 20-min local production about the Faculty of Engineering at NUS; the second is a 40-min commercial travelogue about Singapore. These video packages provide representative material for testing the algorithms. The cartoon employs few special effects, so while all "camera shots" are actually simulations in animation, they are all separated relatively conventionally. Therefore, this was excellent material for the initial testing and evaluation of our algorithms for camera break detection. The NUS documentary video provides a combination of live footage and a few animated sequences, along with the use of various special effects. The entre documentary consists of approximately 200 camera shots, and the transitions are implemented by both dissolves and camera breaks. There are also several sequences of camera panning and zooming, as well as object motion. This video thus provides suitable material for test and evaluation of algorithms for both camera break and gradual transition detection, as well as for the technique for camera movement detection. In the travel documentary there are many more camera breaks and camera movements, as well as more sequences where moving objects and lighting changes arising from flashing lights and the flickering of some objects occur. Thus, this video provides more data for testing the robustness and limitations of the partitioning algorithms.

Three experiments will now be discussed. The first is for proof-of-concept, testing the effectiveness of the difference metrics for segment boundary detection on the cartoon video. The second one presents both a much more in-depth analysis of the difference metrics and a test of twin-comparison for detecting gradual transitions. The comparison of histograms based on colour codes is further analysed by subjecting the complete travel documentary video to both twin-comparison and subsequent analysis for camera movement detection. Presentation of these results will include a detailed error analysis. The final experiment concentrates on the effectiveness of the camera movement detection techniques presented in Sect. 3.2.

5.2.1 Example 1: Camera break detection on a cartoon video

Table 1 lists the results of applying each of four difference metrics in a single pass to partition the cartoon video (12 990 frames). Since we were only concerned with testing the difference metrics, we were only interested in detecting camera

Table 1. Segmentation results of single pass algorithms

Algorithms	T_b	N_d	N_m	N_f	τ (seconds)
Pair-wise pixel, grey	77	51	2	6	71320
Likelihood, grey	28	49	4	27	103023
Histogram, grey	45	44	9	16	32475
Histogram, colour	50	48	5	5	20861

T_b, the threshold selected; N_d, the total number of camera breaks correctly detected by each algorithm; N_m, the number of boundaries missed; N_f, the number of boundaries misdetected; τ, the processing time

Table 2. Segmentation results of multi-pass algorithms

Algorithms	T_b	N_d	N_m	N_f	τ (seconds)
Pair-wise pixel, grey	77	49	4	6	17386
Likelihood	28	49	4	25	19546
Histogram	45	43	10	11	15752
Hybrid	–	42	11	7	22183

breaks. The performance was evaluated with respect to those segment boundaries detected by a manual analysis. The processing time (τ) for each algorithm is also shown in the table for the comparison of the processing load among the four difference metrics. Table 1 indicates that all algorithms detect camera breaks with satisfactory accuracy. The leading success of pair-wise comparison is partially due to the fact that its threshold was optimally tuned, which was not the case for the other algorithms. Nevertheless, it missed two transitions implemented as dissolves and falsely detected six artefacts of camera and object movements. Due to the high requirement of computation time for Eq. 3, the likelihood-ratio algorithm is slow. Of the four algorithms, the colour histogram gives the most promising overall result: both high speed and accuracy. The major sources of error in this case are due to camera panning, object motion, and improper threshold selection.

We have also tested the multi-pass concept on the cartoon video and the results are shown in Table 2. The first three results were obtained by using the same difference metric in both passes, while with the hybrid method, both the histogram comparison and likelihood criterion were applied; and the results from each of the algorithms were used to verify each other. As presented in Sect. 4.2, in the first pass a lower temporal resolution (skip factor of 3) was used. In using pairwise comparison, lower spatial resolution (averaging of 3×4 pixels) was also used in the first pass. The results in Table 2 indicate that this approach performs much faster than that of a single pass, with comparable or better accuracy. In the hybrid approach, a segment boundary is declared only if both algorithms detect it as a boundary. As indicated in the table, such a hybrid approach allows for high-confidence camera breaks to be detected and reduces the number of false detections.

5.2.2 Example 2: Transition detection on a documentary video

A more general test of segmentation, based on three different types of histogram comparison, was applied to the NUS documentary video. The twin-comparison approach was applied to detect both camera breaks and gradual transitions implemented by special effects. The reason that only histogram comparison algorithms were used in this experiment is that they are insensitive to object motion and they are faster than the two types of pair-wise comparison algorithms. The segmentation results are summarized in Table 3.

As in the case of Table 1 and Table 2, the camera breaks detected algorithmically, as well as those missed and misdetected, are listed. In addition, the same information is provided for gradual transitions. It should be noted that camera movements are also included here that may be subsequently detected by the technique discussed in example 3. Similar results, obtained from the differences of colour code histograms for the travel documentary are given in Table 4 as further evaluation of the performance and robustness of this particular algorithm, which will be seen later to be the optimal technique.

The first two rows of Table 3 show the results of applying difference metrics (4) and (5), respectively, to grey level histograms. The third row shows the results of applying difference metric (4) to histograms of the 6-bit colour code. In order to reduce computation time, only every second frame is compared (skip factor is 2), providing a temporal resolution of 15 frames/s. The last set of results were obtained by a two-pass algorithm: in the first pass the colour-code histogram was applied with a skip factor of 10 and a reduced threshold, and the second pass then used a skip factor of 2.

Table 3 shows that histogram comparison, based on either grey level or colour code, gives very high accuracy in detecting both camera breaks and gradual transitions. In fact, no effort has been made to tune the thresholds to obtain these data. Approximately 90% of the breaks and transitions are correctly detected. Among the three single-pass algorithms, colour gives the most promising result: besides the high accuracy, it is also the fastest of the three algorithms. (The 6-bit colour code requires only 64 histogram bins, instead of the 256 bins required for the 8-bit grey level.) The much smaller number of missing breaks suggests that the colour histogram is better than the grey level in detecting qualitative differences in content. In other words it would appear that 6 bits of a colour code provide more *effective* information than 8 bits of grey level. Similar accuracy in detecting both camera breaks and transitions is exhibited in Table 4, again using the colour histogram algorithm, though there are many more breaks and camera movements in this video.

Note that the χ^2-test histogram comparison algorithm does not yield a better result, even after tuning the threshold. In fact the number of missing and false detections in the second row of Table 3 is dramatically greater than those in the first row. This is contrary to the conclusions of Nagasaka and Tanaka (1991), where no experimental data were presented.

Table 3. Detection results for camera breaks and gradual transitions based on four types of twin-comparison algorithms applied to the NUS documentary video

Type of transitions	Camera breaks			Transitions + camera movements				
Algorithms used	N_d	N_m	N_f	N_d	N_m	N_f	N_P	N_Z
Grey level comparison	65	13	2	101	8	9	2	1
χ^2 grey level comparison	60	18	16	93	16	9	2	3
Colour code comparison	73	5	3	95	14	13	1	2
Colour code, 2 passes	71	7	3	88	21	18	3	2

N_d, the total number of camera breaks correctly detected by each algorithm; N_m, the number of boundaries missed; N_f, the number of boundaries misdetected; N_P, the number of "transitions" actually due to camera panning; N_Z, the number of "transitions" actually due to camera zooming

Also, χ^2-test comparison requires the longest computation time among the three algorithms. Therefore, we shall not discuss this algorithm any further, concentrating our attention on simple histogram comparison.

The multi-pass approach achieves similar accuracy, but it is almost three times faster than the single-pass colour code algorithm. The increased missing transitions were due to errors in the first pass and can be improved by either lowering the threshold or increasing the skip factor for the first pass. Accuracy may also be improved by increasing the search vicinity around the potential boundaries in the second pass.

Table 4. Results of applying the twin-comparison approach to camera break and transition detection and the camera movement detection approach on the travel documentary video. Differences were calculated from colour histograms

Camera breaks			Transitions + camera movements				
N_d	N_m	N_f	N_d	N_m	N_f	N_Z	N_P
280	2	10	104	6	19	20	22

N_d The total number of camera breaks correctly detected by each algorithm; N_m, the number of boundaries missed; N_f, the number of boundaries misdetected; N_Z, the number of "transitions" actually due to camera zooming; N_P, the number of "transitions actually due to camera panning

The missed camera breaks mainly result from the fact that the differences between some frames across a camera break are lower than the given threshold. Both the grey level and colour histogram comparison algorithms failed to detect the two frames across a camera break (Fig. 10), because the histograms of the two frames are similar. All missed camera breaks based on simple histogram comparison listed in Table 3 have resulted from this problem. The missing gradual transitions are mainly due to the same problem, as shown in Fig. 11. However, this problem cannot be solved simply by lowering the threshold because this will lead to an increase in the number of false detections. We have to accept the fact that, as in spatial segmentation in static image processing (Rosenfeld and Kak 1982), it is almost impossible to obtain a threshold that will achieve 100% accuracy. For this reason our segmentation system also provides a set of editing tools to enable the user to correct those few errors that result from the automatic partitioning process.

The second class of errors, false camera break detections, are mainly due to sharp changes in lighting arising from flashing lights and flickering objects, such as a computer screen, in the image. This type of error accounts for all the false breaks listed in all the rows of Table 3, except for some instances based on the χ^2-test histogram comparison, and four false breaks in Table 4. Such problems may be solved by using the robust algorithms presented in Sect. 4.2 if the large illumination changes only occur in a restricted portion of the entire frame. The rest of the false breaks listed in Table 4 result from dramatic content changes in short transitions of a length of 20 frames or less. That is, a short transition may be detected as one or more breaks due to the large content changes within the transition. This fact also accounts for four missing transitions that were falsely detected as breaks.

Flashing lights and flickering objects are also the major source of false detection of transitions: 9 out of 19 false transitions in listed in Table 4 resulted from such situations. In fact, gradual transitions are more sensitive than camera breaks to such changes, since even a gradual change of lighting can cause a false detection of a transition. Figure 12 shows an example of such a false transition where four frames from a shot of a colourful water fountain were detected as a transition due to the colour lighting changes.

Object movement is another main source of false detection of gradual transitions, and it accounts for most of the false transitions in both Table 3 and Table 4. Figure 13 shows an example of such a false detection. Movement of the large object results in large changes between consecutive frames that may far exceed the thresholds set for transitions. However, object motion in general tends not to induce the regular patterns in the motion field that arise from camera movements. Therefore, object motion cannot be detected by the technique developed for camera movement detection. An alternative is to identify a moving object as a cluster in the motion field and then use that cluster of vectors to track the object. One can then assume that a transition will not take place in the middle of an object's trajectory and thus eliminate false detection. Hence, detecting object motion is

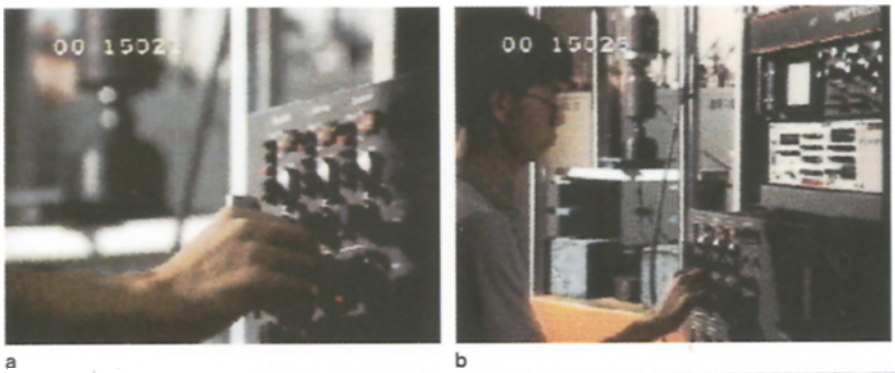

Fig. 10a,b. Missed camera breaks: two frames across a camera break that both the grey-level and the colour-histogram comparison algorithms failed to detect because the histograms of the two frames are similar

Fig. 11a–d. A missed gradual transition: four frames from a dissolve transition that the twin-comparison approach, using the colour-histogram comparison as the difference metric, failed to detect, because the colour histograms of the two frames in the two shots are very close to each other

an important part of our efforts to improve the accuracy of video partitioning algorithms.

As a supplement to the data in Table 3, note that only nine of the errors of missing or false breaks or transitions are common to the analyses based on grey level and colour code histograms, respectively. Such overlap will be less likely if we compare the results from using pair-wise comparison and histogram comparison. In other words the results from the two different algorithms may compensate each other if we combine them in a proper way, rather than the exclusive verification used in the hybrid multi-pass approach presented in Table 2. However, if we use an opposite scheme where a segment boundary is declared whenever it is detected by *either* of the two algorithms, the number of false detections may increase. To avoid this, we are currently investigating the use of a *confidence measure* based on both the difference across two frames and the selected threshold. The concept of "fuzzy boundary" may be used in combining the segment boundaries obtained from different algorithms. While it should be relatively easy to improve the accuracy of camera break detection, improving the accuracy in detecting gradual transitions will probably be much more difficult.

Table 5. Detection of camera movements by motion detection and analysis algorithms

Camera movements	N_d	N_m	N_f
Panning (skip = 1)	14	0	3
Zooming (skip = 1)	3	4	0
Zooming (skip = 5)	7	0	0

N_d, The number of camera movements detected by the algorithms; N_m, the number of camera movements missed by the algorithms; N_f, the number of false detections of camera movements

5.2.3 Example 3: Camera movement detection

Camera pan and zoom sequences are distinguished from gradual transitions by using the motion analysis techniques discussed in Sect. 3.2. Twenty blocks of 15×15 pixels, uniformly distributed in a frame as four rows of five columns,

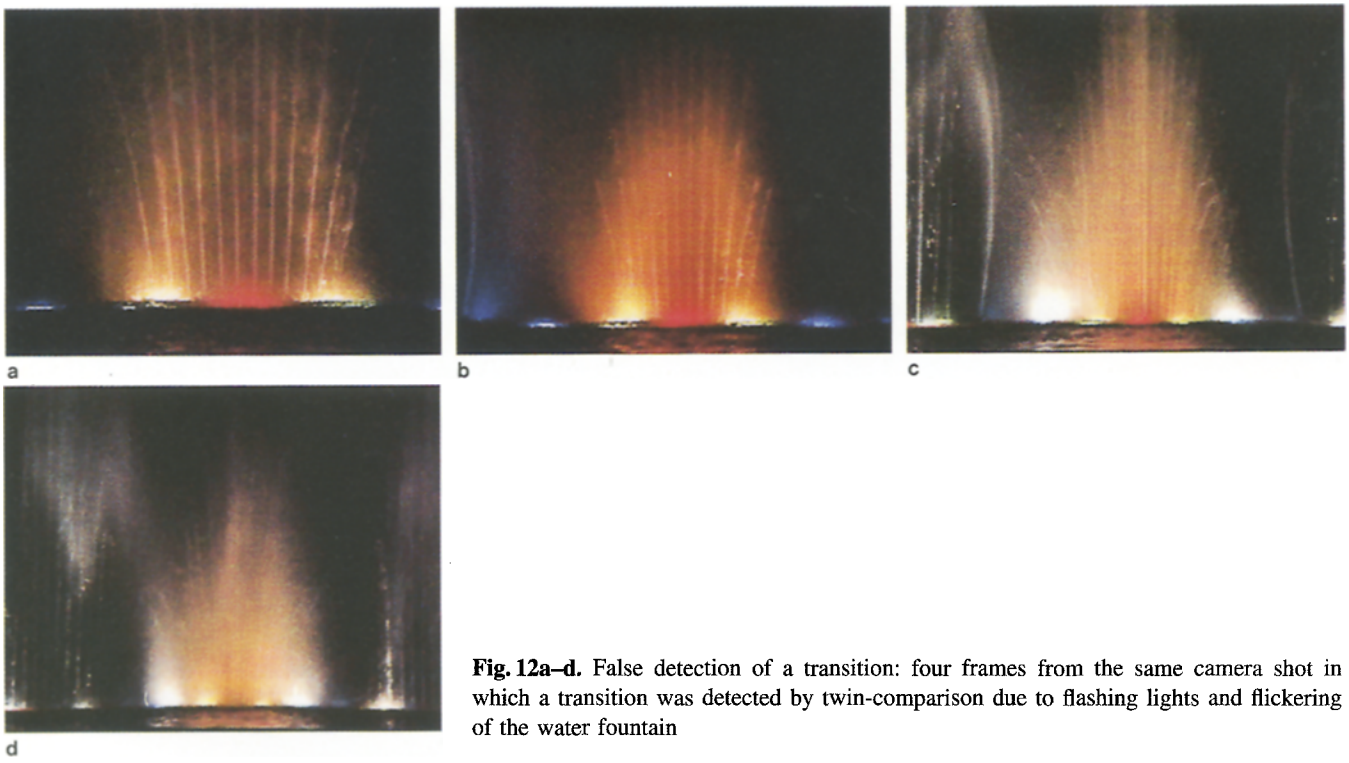

Fig. 12a–d. False detection of a transition: four frames from the same camera shot in which a transition was detected by twin-comparison due to flashing lights and flickering of the water fountain

are used to compute the motion vectors with the following cost function:

$$P_i(k) = \sum_{m}^{M} \sum_{n}^{N} |b_i(m,n) - b_{i+1}(m,n)| \ . \tag{12}$$

The search area for minimizing this cost function is crucial for obtaining accurate motion vectors and acceptable computation speed. In order to accommodate the video rate and the maximum velocity of camera movement, we have chosen 9 pixels per frame as the search area. It is not necessary to compare every pair of consecutive frames in a potential transition to determine if they resulted from a camera movement. Instead, only two pairs of consecutive frames are used to compute motion vectors, and the second pair is used to verify the first. If the motion vectors detected from the two pairs oppose each other, then they are probably not due to camera movements. If a potential transition lasts a large number of frames, it may be necessary to compute more than two pairs of frames in order to get higher confidence in the results.

Let (u_k, v_k) be a motion vector, where k ranges from 1 to N. Following the analysis in Sect. 3.2, let (u_m, v_m) be the modal value of these N motion vectors. Then a camera pan is detected if the fraction of motion vectors equal to the mode value exceeds a given threshold T_p. Camera zooms are detected using the comparison approach described in Sect. 3.2 that examines only the border rows and columns.

Our experiments show that those camera movements detected as potential transitions in Table 3 and Table 4 are correctly detected as camera pan or zoom instances. Figure 14 shows frames from a camera panning (moving from up to down) and zooming (moving from close to far), respectively, that were detected as transitions first and were subsequently recognized as camera movement. This demonstrates that motion detection and analysis are effective in distinguishing camera movements from gradual transitions with a high degree of accuracy.

While identifying such distinctions was our primary purpose, we have also tried to classify camera movements over an entire video. This could provide useful information in video content analysis and in selecting representative frames for a detected segment. For instance, the end frame of a zoom can serve as the representative frame of a segment.

Our experiment consisted of applying the algorithm directly to the video package without first identifying potential transition frames. That is, the motion vector field between each pair of consecutive frames was computed and analysed to determine if there was a camera movement during a sequence of the frames. The results are summarized in Table 5. Two different skip factors were tested, and it was found that it is difficult to detect small zoom effects correctly with no skip (using every frame).

The motion of a large object in a camera shot is the major source of error for camera pan detection: two false detections of camera pans listed in Table 5 are due to this problem. Another one is due to erroneous motion vectors computed by the block-matching algorithm, which is inherently inaccurate when there are multiple motions inside a block. Therefore, to improve the accuracy of detecting camera movements, we need to apply a more accurate algorithm to determine motion vectors. Block-matching is particularly problematic during a zoom sequence because the area covered by the camera angle expands or contracts during a zoom. Also, *combinations* of pans and zooms make the detection of camera movement even more difficult. A more general-

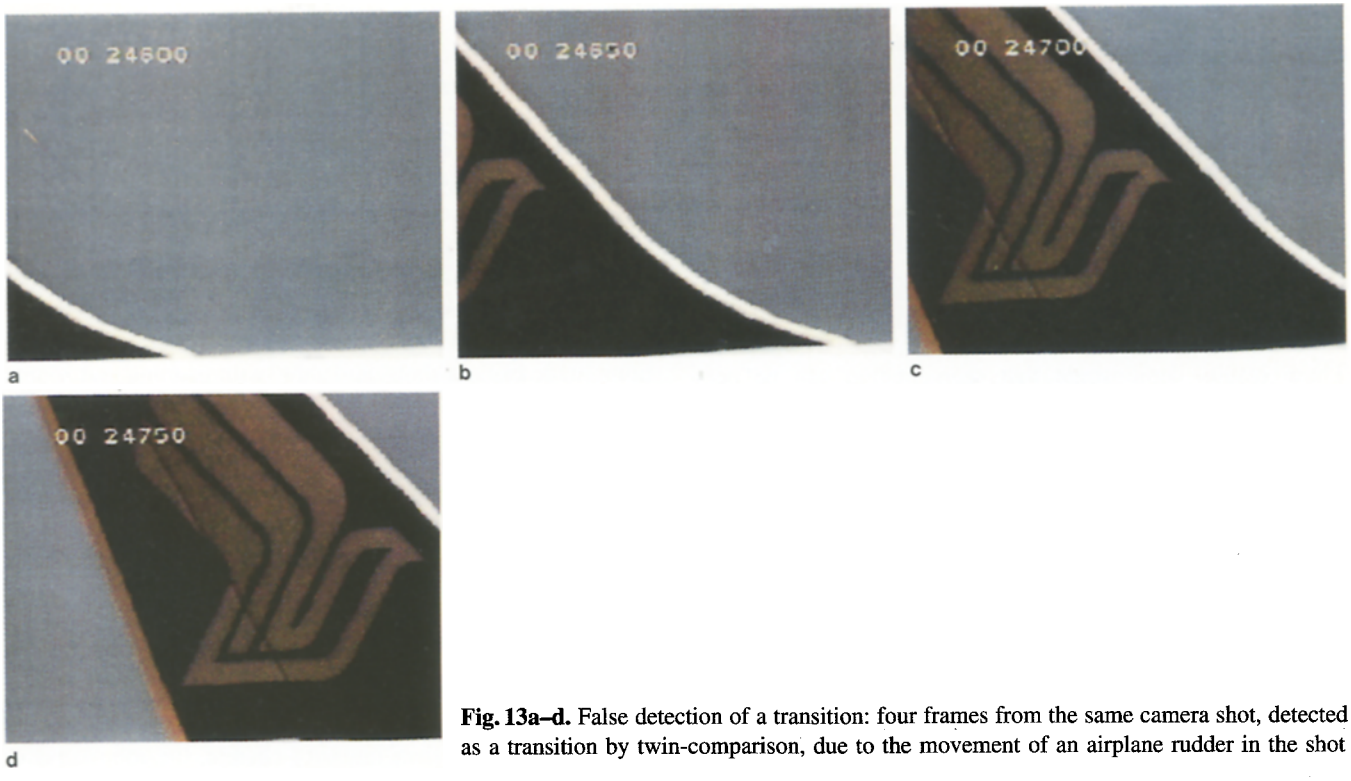

Fig. 13a–d. False detection of a transition: four frames from the same camera shot, detected as a transition by twin-comparison, due to the movement of an airplane rudder in the shot

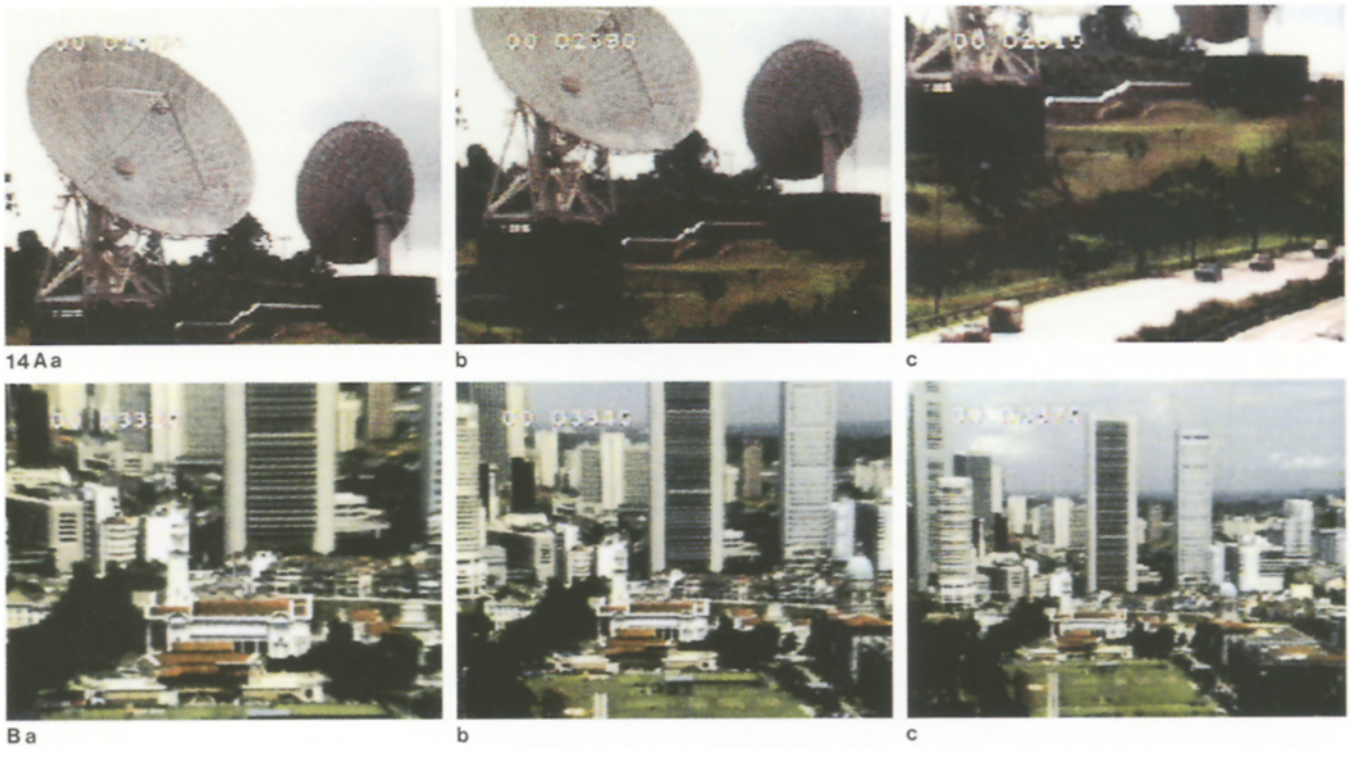

Fig. 14Aa–c, Ba–c. Camera panning and zooming detected by motion analysis: **Aa–c** three frames from a panning sequence; **Ba–c** three frames from a zooming sequence

ized motion classification approach is necessary, such as the global zoom/pan estimation algorithm proposed in Tse and Bakler (1991), which requires analysis of the motion field of a much higher density – a very time consuming process.

Another problematic situation occurs when the camera follows a moving object. The size of the object may be such that the background or foreground changes due to the camera panning exceed the threshold T_s; the sequence is then detected as a potential transition. If the moving object is not

too large, the sequence can be detected as camera panning by motion vector analysis. However, if the object covers more than half the frame, there will not be a clear mode of motion vectors, and block matching will fail to detect camera panning. This accounts for a few of the false detections listed in Table 5. Such cases can no longer be treated as simple camera panning; rather the problem is one of *motion recovery*, which needs a more sophisticated analysis.

5.3 Summary and further discussion

These results demonstrate that segmentation can be performed with satisfactory accuracy. It has also been shown that, among a variety of difference metrics, comparison of colour code histograms is the optimal choice; it achieves both high accuracy and speed. The major sources of error in detecting breaks and transitions are object motion, lighting changes and a few inherent limitations of the segmentation processing, the effects of which need to be reduced to improve the performance of the algorithms further.

As was observed in the experiments, object motion is a major obstacle to segmentation and detection of camera movement. Fast motions of a single object or motion of several objects may induce a frame-to-frame difference that would be detected as a camera break, while any object motion will disrupt the visual flow required to identify pans and zooms. Distinguishing object motions from camera motion is much more difficult than distinguishing camera movement from gradual transitions, since object motions in a sequence of frames are usually not as regular as those introduced by camera movements. This problem is best solved by algorithms for object tracking. Indeed, continuous tracking of a set of objects can serve as an alternative criterion for setting segment boundaries, while the objects themselves are fundamental units for any indexing task. Therefore, the next phase of our project will involve an extensive study of object tracking algorithms. Again, optical flow analysis developed for robot navigation can be used for object tracking, though this particular analysis is still a very difficult task. Further study of optical flow analysis has been initiated as an important part of the project for both object tracking and camera movement detection.

Currently, computation of the field of motion vectors is implemented in software, but chips developed for video compression and transmission, using the H.261 (Ang et al. 1991) and MPEG (Le Gall 1991) standards, can be used for the same purpose. These chips compute a motion vector for every block of 8×8 pixels for motion-compensated interframe coding for video compression. When a video compression chip processes input from any video source, motion vectors are computed in real-time, thereby increasing the efficiency of computation with much higher spatial resolution than has been implemented in software. In addition, if the video material is stored as a compressed file, we can retrieve the motion vectors directly from that file. Once the dense motion field can be obtained accurately, algorithms to recover motion from optical flow (Kasturi and Jain 1991) may also be used for camera movement detection. Such analysis may provide more accurate information about camera movement when both camera and object motion exist in the same frame sequence.

The study of object tracking and motion analysis is of importance not only for camera shot boundary detection but also for the classification of camera shots. Initially, the twin-comparison approach was introduced to detect gradual transitions but, as we have seen, it can also be used to detect potential sequences of camera and object motion. Such sequences can then be analysed further and classified as transitions, static camera shots, and shots with panning and zooming. Such a classification will provide useful information for content analysis of the video sequence.

If lighting changes only result in changes of pixel intensity, the problem of false detection can be avoided by using a colour histogram comparison algorithm suggested by Arman et al. (1993). This algorithm uses the two-dimensional hue and saturation (HS) histogram based on the hue, saturation and intensity (HSI) colour space. Only the hue and saturation information are used because past studies have shown that the hue of an object surface remains the same under different lighting intensities. However, since illumination changes may involve more than intensity change, the robustness of this algorithm may be limited.

6 Conclusions and future work

As an initial effort towards the automatic partitioning of a video package into meaningful segments, techniques for detecting camera breaks and gradual transitions implemented by special effects as well as the effects of camera movement, have been developed and implemented. The resulting algorithms have been tested on a variety of video sources. The segment boundaries detected by these algorithms have been compared with those detected by manual logging, and it has been demonstrated that the algorithms perform with satisfactory accuracy. Some future improvements were suggested in Sect. 5.3. We shall now consider several other important aspects of future work, including the development of more computer tools for video indexing.

6.1 Further enhancement of the video partitioning system

An important enhancement for video indexing is the capability to select a representative frame automatically so that it can be used as a visual cue for each segment determined by the partitioning process. Such cues are particularly valuable if one wishes to build an index based on a content analysis of each segment. Alternatively, an array of these representative frames can be used as an index itself through which a user can browse when searching for video source material. Our current effort toward automatic selection of representative frames focuses on developing algorithms that do not require any sophisticated (or computationally intensive) attempts at content analysis. For instance, in case there is a

camera movement in a shot, then the end frame of the pan or zoom can be selected as the representative frame. Algorithms that find an average frame in a sequence may serve our purpose in case there are no other useful features.

As more and more video material becomes available in a compressed digital form, it would be advantageous to perform segmentation directly on that digital source in order to save on the computational cost of decompressing every frame. This is an important aspect of our current research. In general, compression of a video frame begins with dividing each colour component of the image into a set of 8×8 pixel blocks (Le Gall, 1991). The pixels in the blocks are then coded by the forward discrete cosine transform (DCT). The resulting 64 coefficients are then quantized and subjected to Huffman entropy encoding. This process is then reversed during decompression. Since these coefficients are mathematically related to the spatial domain, they can be used to detect the sorts of changes associated with segmentation. That is, the segment boundaries are detected by correlating the DCT coefficients of a set of blocks from each frame (Arman et al. 1993). Since only the differences between the *key frames* or *Intrapictures* in video compressed using MPEG, H.261 or Apple QuickTime standard contain scene change information (Le Gall, 1991; Liou, 1991; Apple Computer 1991), video partitioning can be based on only these frames. Thus, the processing time for correlating DCT coefficients can be reduced further. Our initial studies have shown promising results.

6.2 Video indexing: challenges

As we have already observed, the sort of automatic video partitioning reported here is only the first step in our effort towards computer-assisted video indexing. The next important step is to perform content analysis on individual segments to identify appropriate index terms. This is obviously a more challenging task. More sophisticated image processing techniques will be required to provide useful tools for automatic identification of index entries and content-based video retrieval.

Object tracking and motion analysis will not only improve the performance of video segmentation and camera shot classification, as pointed out in Sect. 5.3, but also they will supply the content information required for indexing. For instance, if we can extract a moving object from its background and track its motion, we should be able to construct a description of that motion that can be employed for subsequent retrieval of the camera shot. There is also the value of *identifying* the object, once it has been extracted; but even without sophisticated identification techniques, merely constructing an icon from the extracted image may serve as a valuable visual index term.

Besides images in video, another important source of information in most video packages is the audio track. As any film-maker knows, the audio signal provides a very rich source of information to supplement the understanding of any video source (Metz 1985); this information may also be engaged for tasks of segmentation and indexing. For instance, significant changes in spectral content may serve as segment boundary cues. Tracking audio objects will also provide useful information for segmentation and indexing. Therefore, an effective analysis of the audio track and its integration with information obtained from image analysis will be an important part of our future work on video segmentation and indexing.

Before a video index system can be developed, it will be necessary to develop an architectural specification. If one is working with a dynamic medium like video, any index that ultimately takes the form of a printed document is likely to be of limited value. It is preferable to view the index itself as a piece of computer software through which the user may interact with the video source material. This is a vision we share with the Visual Information System project (Swanberg et al. 1993), along with their specification of an architecture that integrates databases, knowledge bases, and vision systems. We can view the index construction process as one of data insertion in a video database. The index construction system (video parser) will first use the partitioning techniques to segment the video stream into representative clips. Once those clips are generated, the knowledge module should be able to feed components of the video sample into a parser that can compare the sample to the clips. When the parser identifies a matching clip or episode, it communicates its results with the knowledge base to determine an index term that can then be assigned. That is, the knowledge module should provide the connection between the vision system and a database of index categories. To support this activity, the parser should be able to both calculate similarities between video samples and extract a representative sample from a set of video clips. The building of such a parser system is the primary challenge we are facing, and it is the focus of our future work in developing a computer system for video indexing.

Acknowledgements. This research is carried out with support from the National Science and Technology Board of Singapore.

References

Anandan P (1989) A computational framework and an algorithm for the measurement of visual motion. Int J Comput Vision 2:283–310

Ang PH, Ruetz PA, Auld D (1991) Video compression makes big gains. IEEE Spectrum 28:16–19

Apple Computer (1991) Quick time developer's guide. Cupertino

Arman F, Hsu A, Chiu M-Y (1992) Feature management for large video databases. Proc SPIE Conf on Storage and Retrieval for Image and Video Databases, San Diego

Bordwell D, Thompson K (1993) Film art: An introduction, McGraw-Hill, New York

Horn BKP, Schunck BG (1981) Determining optical flow, Artif Intell 17:185–203

Kasturi R, Jain R (1991) Dynamic vision. In: Computer Vision: Principles, Kasturi R, Jain R (eds) IEEE Computer Society Press, Washington, pp. 469–480

Le Gall D (1991) MPEG: A video compression standard for multimedia applications. Commun ACM 34:47–58

Liou M (1991) Overview of the px64 kbit/s video coding standard. Commun ACM 34:59–63

Mackay W, Davenport G (1989) Virtual video editing in interactive multimedia applications. Commun ACM 32:802–810

Metz C (1985) Aural objects. In: Film sound: theory and practice. Weis E, Belton J (eds) Columbia University Press, New York, pp. 154–161

Nagasaka A, Tanaka Y (1991) Automatic video indexing and full-video search for object appearances. Proc 2nd Working Conf Visual Database Systems, pp. 119–133

Netravali AN, Haskell BG (1988) Digital pictures: representation and compression. Plenum, New York

Rosenfeld A, Kak AC (1982) Chapter 10: Segmentation. Digital Picture Processing, 2nd edn. Academic Press New York, pp. 57–190

Ruan LQ, Smoliar SW, Kankanhalli A (1992) An analysis of low-resolution segmentation techniques for animate video. Proc ICARCV '92. 1:CV-16.3.1–16.3.5

Swanberg D, Shu C-F, Jain R (1992) Knowledge guided parsing in video databases. Proc IS&T/SPIE's Symp Electronic Imaging: Sci Tech, San Jose

Tonomura Y (1991) Video handling based on structured information for hypermedia systems. Proc Int Conf Multimedia Information Syst, Singapore, pp. 333–344

Tse YT, Bakler RL (1991) Global zoom pan estimation and compensation for video compression. Proc ICASSP '91, pp. 2725–2728

Zhang HJ, Kankanhalli A, Smoliar SW (1993) Automatic video partitioning and indexing. Proc. IFAC '93, to appear. Detecting camera breaks in full-motion video, available from the authors upon request

HONGJIANG ZHANG obtained his Ph.D. from Technical University of Denmark in 1991 and his BSc from ZhengZhou University, China, in 1981, both in Electrical Engineering. He had also worked as an engineer at Shijiaz huang Communications Institute, China, before his Ph.D work. He joined the Institute of Systems Science at the National University of Singapore in December 1991 and is presently working on projects on video classification and moving object tracking and classification. His current research interests include image processing, computer vision, multimedia, digital video, and remote sensing.

ATREYI KANKANHALLI obtained the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Delhi, in 1986 and the M.S. degree in electrical engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1987. From 1988 to 1989, she was a graduate research assistant in the Biomedical Engineering Department at RPI. Since 1990, she is with the Institute of Systems Science, National University of Singapore, where she is working in the areas of image segmentation and object tracking.

STEPHEN WILLIAM SMOLIAR obtained his PhD in Applied Mathematics and his BSc in Mathematics from MIT. He has taught Computer Science at both the Technion, Israel, and the University of Pennsylvania. He has worked on problems involving specification of distributed systems at General Research Corporation and has investigated expert systems development at both Schlumberger and the Information Sciences Institute (University of Southern California). He is currently interested in applying artificial intelligence to advanced media technologies. He joined the Institute of Systems Science at the National University of Singapore in May 1991 and is presently leading a project on video classification.