Multimedia Tools and Applications, 1, 89–111 (1995)
© 1995 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Video Parsing and Browsing Using Compressed Data

HONGJIANG ZHANG, CHIEN YONG LOW AND STEPHEN W. SMOLIAR zhj@iss.nus.sg
Institute of Systems Science, National University of Singapore, Heng Mui Keng Terrace, Kent Ridge, Singapore 0511, Republic of Singapore

Abstract. Parsing video content is an important first step in the video indexing process. This paper presents algorithms to automate the video parsing task, including partitioning a source video into clips and classifying those clips according to camera operations, using compressed video data. We have developed two algorithms and a hybrid approach to partitioning video data compressed according to the JPEG and MPEG standards. The algorithms utilize both the video content encoded in DCT (Discrete Cosine Transform) coefficients and the motion vectors between frames. The hybrid approach integrates the two algorithms and incorporates multi-pass strategies and motion analyses to improve both accuracy and processing speed. Also, we present content-based video browsing tools which utilize the information, particularly about the shot boundaries and key frames, obtained from parsing.

Keywords: video parsing, video compression, MPEG, JPEG, video retrieval, video browsing, motion analysis, camera operation, video database, multimedia systems

1. Introduction

Currently, the practical use of video as an information resource is seriously limited by a lack of viable indexing and retrieval systems to enable easy and effective organization and retrieval. The current state of the art requires a process known as *logging*, which basically reduces a video source to some form of structured text which may then be entered into a database. Unfortunately, text descriptions can be both inadequate and inaccurate; and, in the absence of any effective tools, logging is an extremely tedious operation. Overcoming these difficulties requires tools which support at least three tasks [8, 15]. First, we have to parse the video into an appropriate set of units for indexing, which we shall call *clips*. Second, we need to identify both low-level image properties and semantic properties of individual clips, using these properties as a representation of content from which we can construct an index. Third, but not the least important, we need a set of retrieval and browsing tools to select video material resulting from a query. The last task is especially important in light of the high volume of video data.[1]

Up to now, a variety of algorithms and systems have been developed to successfully support the first task [5, 13]. However, they have been based on uncompressed video, usually digitized from analog sources. On the other hand, different schemes of video compression have been studied and proposed to reduce the data volume of video for storage and transmission, among which MPEG [2], JPEG [12] and H.261 [7] have become standards. As a result, more and more video data have been and will continue to be stored and distributed in compressed digital formats. It would therefore be advantageous for the tools we envisage to operate directly on compressed representations, saving on the computational cost of decompression and lowering the overall magnitude of the data which must be processed. This paper reports our work which contributes to developing tools to support video parsing, retrieval and browsing based on such compressed data, especially on video compressed using MPEG.

In Section 2 we first review previous work related to video parsing systems. Section 3 presents a novel approach to parsing MPEG video data, along with an experimental evaluation. Section 4 discusses a set of content-based video browsing tools. Our conclusions are given in Section 5.

2. Video parsing: principles and previous work

The first task in video parsing is to partition a given video stream into basic units to be identified and indexed. By way of analogy to sentences in a body of text, these elements will take the form of continuous segments from the time-line of the source video. Appropriate index terms (which need not be restricted to text and may include, for example, one or more representative frames) may then be assigned to each such segment, based on a variety of approaches to content analysis. We consider a camera shot (also called a *clip*), consisting of a sequence of contiguous frames representing a continuous action in time and space, as the basic indexing unit for video. Parsing thus begins with the process of detecting the boundaries between consecutive camera shots.

Consecutive frames on either side of a camera break generally display a significant change in content. Therefore, what is required is a quantitative measure which captures the qualitative difference between such a pair of frames. Then, if that measure exceeds a given threshold, it may be interpreted as indicating a segment boundary. Hence, establishing suitable metrics is the key issue in automatic partitioning. The most commonly used metrics can be divided into two types: those based on comparing corresponding pixels or block regions, and those based on derived features, such as a histogram of intensity values. Experiments have shown that pixel histograms, based on either color or gray-level intensities, provide a very robust metric [5, 13].

When techniques such as dissolve, wipe, fade-in, and fade-out are employed, the boundary between two shots no longer lies between two successive frames; instead, it is spread out across a sequence of frames. Our approach to partitioning requires identifying the start and end points of these transitions, since they should not receive the same subsequent content analysis which we apply to "pure" clips. However, the quantitative difference between any two consecutive frames in such a transition is much less than the difference value which indicates a camera break; so the simple difference metric and a single threshold which detect camera breaks cannot also detect these gradual transitions. Zhang et al. have developed a *multi-pass*, *twin-comparison* approach to solve this problem [13]. This technique performs a first pass to locate *potential* boundaries for both simple breaks and gradual transitions. This pass does not examine all consecutive frames but employs a *skip factor* which defines a constant interval of time between the frames which are compared. A second pass then performs two more refined comparisons for gradual transition detection: the first uses a reduced threshold to detect the potential starting frame of a transition, and the second uses a higher threshold to detect the ending frame of the transition. Experiments have shown that this is a very effective approach [13].

A major problem in partitioning is that camera operations and fast object motion may introduce false positives, since they tend to exhibit quantitative differences of the same order as those of gradual transitions. Thus, it is necessary to distinguish changes associated with those transitions from changes introduced by camera movement. This problem has been addressed by Zhang et al. [13] and Ueda et al. [11], using a camera operation detection algorithm based on motion vectors between consecutive frames.

These approaches all require that each video frame be represented as an array of pixel intensities. If the input is a compressed digital video, then it must first be decompressed in its entirety. Since the algorithms are already compute-intensive, additional computation time is far from desirable. A more effective approach is to develop tools that can work directly on compressed representations. Arman et al. [1] have proposed partitioning JPEG video with a difference metric based on correlation between the Discrete Cosine Transform (DCT) coefficients of consecutive frames. The algorithm is based on the fact that the DCT coefficients of each frame are mathematically related to the spatial domain and represent the content of the frames; therefore, they can be used to detect qualitative differences between two video frames. In other words, the DCT coefficients can be used to detect shot boundaries if we can construct a suitable metric.

To improve both accuracy and speed, Arman et al. also applied a similar multi-pass approach. The first pass uses a skip factor and its own threshold value for isolating the regions of potential cut points. The frames which cannot be classified based on DCT coefficient comparison may be decompressed for further examination by color histogram comparison. However, while the approach is effective in detecting camera breaks, it fails to address the detection of gradual transitions, camera operations, and object motion, all of which significantly affect partitioning accuracy. Also, the DCT comparison metric and the subsequent decompression are computationally expensive.

Our own approach uses two fast yet effective algorithms: The first one applies a simpler difference metric to the comparison of DCT coefficients; the second utilizes motion information encoded in MPEG data. We have developed a novel *hybrid* approach, which exploits the advantages of both of our algorithms, as well as Arman's, while overcoming their respective shortcomings. This hybrid technique also incorporates robust motion analysis. As will be discussed in Section 3, the result is a powerful parser with the ability to detect gradual transitions, camera operations, and object motion.

We have also addressed the problem of providing content-based retrieval and browsing tools. Arman et al. [1] have proposed that key frames which represent the content of each shot can facilitate browsing; but they fail to address how such frames may be automatically selected. We shall briefly discuss a solution to this problem, again based on a compressed video source, along with a set of associated tools, in Section 4.

3. Video parsing based on compressed data

In this Section we begin with a brief review of the JPEG and MPEG video compression standards. We then discuss three difference metrics for partitioning either JPEG or MPEG video: two employ the DCT coefficients used in both JPEG and MPEG representations, while the third uses MPEG motion vectors. After this, we discuss how to combine the power of the two approaches, incorporating multi-pass, twin-comparison strategies and motion analysis which enable the detection of gradual transitions and camera operations.

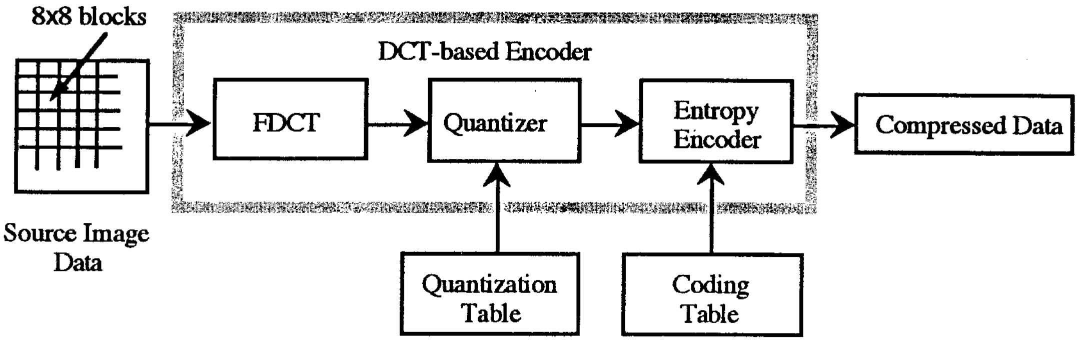


Figure 1. A JPEG encoder.

3.1. JPEG and MPEG: a brief introduction

To provide some background, we briefly summarize the principles of JPEG and MPEG encoding before introducing our algorithms. Figure 1 illustrates the basic processes of JPEG, the major standard for still picture compression. The key idea is to reduce spatial redundancy. Compression begins by dividing an image into a set of 8×8-pixel blocks [2, 12, 7]. The intensity levels in each block are then transformed by the forward Discrete Cosine Transform (DCT) into 64 coefficients, which are then quantized and run-length, Huffman entropy encoded. The process can then be reversed for decompression. If we can encode and decode JPEG fast enough to manage video rates, we can apply it to individual video frames, achieving what is known as *Motion JPEG*. Because each frame is transformed independently, no process is performed to reduce temporal redundancy.

MPEG, the standard for motion video compression proposed by the Moving Picture Experts Group, reduces spatial redundancy with the same technique as JPEG but also uses motion compensation coding to reduce temporal redundancy. However, this approach entails a tradeoff between efficient coding and fast random access. To facilitate random access, certain frames are encoded like still JPEG images without any motion compensation encoding; these frames are called intra-pictures (**I**) [2, 7]. Two types of frames are motion compensation encoded: predictively coded (**P**) frames and bi-directional predictively coded (**B**) frames. All three types are illustrated in Fig. 2. There are two sets of motion vectors (one for every 8×8 block), forward and backward, associated with each **B** frame and a single set for each **P** frame. The residual differences associated with motion compensation of each block in **P** and **B** frames are then DCT-coded in the same way as blocks in JPEG frames. Let us now turn to the use of DCT and motion compensation data for video parsing.

3.2. Two algorithms based on DCT coefficients

As stated before, the algorithm developed by Arman et al. [1] is based on the correlation between DCT coefficients of consecutive frames of JPEG compressed video. A vector representation for each frame is constructed by a subset of the DCT coefficients of a subset of the blocks in the frame:

$$V_f = \{c_1, c_2, c_3, \ldots, c_k\}. \tag{1}$$

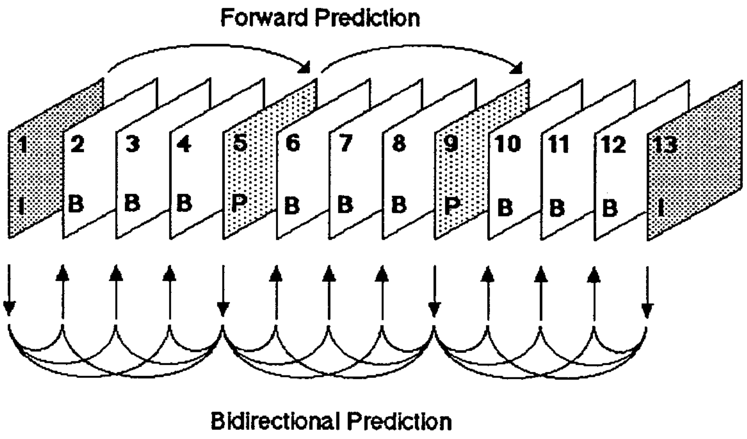


Figure 2. Three types of frames in MPEG: intra-pictures (**I**), predicted frames (**P**), and interpolated, bidirectional predicted frames (**B**). Here there are eleven frames between successive **I** frames, and the ratio of **B** frames to the union of **I** and **P** frames is three to one. These ratios are flexible and are controlled by parameters such as coding display.

The difference metric between the frames is then defined in terms of a normalized inner product:

$$\Psi = 1 - \frac{|V_f \bullet V_{f+\varphi}|}{|V_f||V_{f+\varphi}|} \tag{2}$$

where φ is the number of frames between the two frames being compared.

Following the approach of pair-wise comparison [13], we have developed a pair-wise block algorithm which compares the DCT coefficients of corresponding blocks of consecutive video frames. More specifically, let $c_{l,k}(i)$ be a DCT coefficient of block l in frame i, where k ranges from 1 through 64 and l depends on the size of the frame; then the content difference of block l in two frames which are φ frames apart can be measured as:

$$\text{Diff}_l = \frac{1}{64}\sum_{k=1}^{64} \frac{|c_{l,k}(i) - c_{l,k}(i+\varphi)|}{\max[c_{l,k}(i), c_{l,k}(i+\varphi)]} \bullet 100\% \tag{3}$$

If this difference exceeds a given threshold t:

$$\text{Diff}_l > t \tag{4}$$

then we can say that the block has changed across the two frames. If $D(i, i+\varphi)$ is defined to be the percentage of blocks which have changed, then a segment boundary is declared if

$$D(i, i+\varphi) > T_b \tag{5}$$

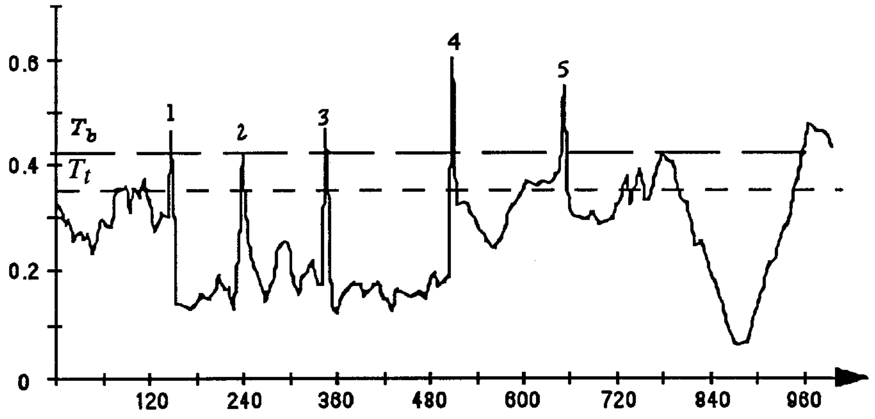
where T_b is the threshold for camera breaks. This difference metric is thus analogous to pair-wise pixel comparison [13], using DCT coefficients instead of pixel intensities.

Selecting appropriate values for t and T_b may also employ the same techniques discussed in [13]. t tends not to vary across different video sources and can easily be determined experimentally. T_b, on the other hand, is best computed in terms of the overall statistics for values of $D(i, i + \varphi)$ as a value which exceeds the mean by about five standard deviations.

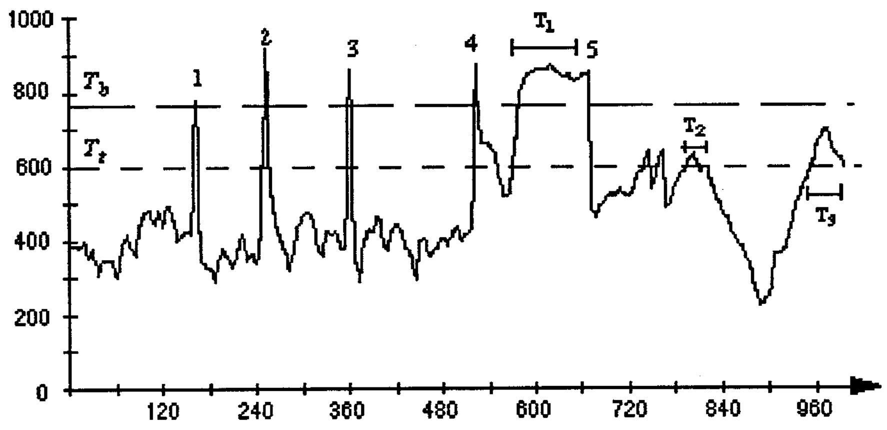
For implementation purposes processing time can be reduced significantly by applying Arman's technique of using only a subset of coefficients and blocks. However, our algorithm requires far less computation than the difference metric defined by (2). Also, it is more sensitive to gradual changes, which makes it more useful than (2) for the hybrid approach to detecting transitions discussed in Section 3.4.

The above two algorithms may be applied to every video frame compressed with Motion JPEG; but in an MPEG video it may only be applied to **I** frames, since those are the only frames encoded with DCT coefficients. Since only a small portion of all video frames are **I** frames, this significantly reduces the amount of processing which goes into computing differences. On the other hand, the loss of temporal resolution between **I** frames may introduce false positives which have to be handled with subsequent processing.

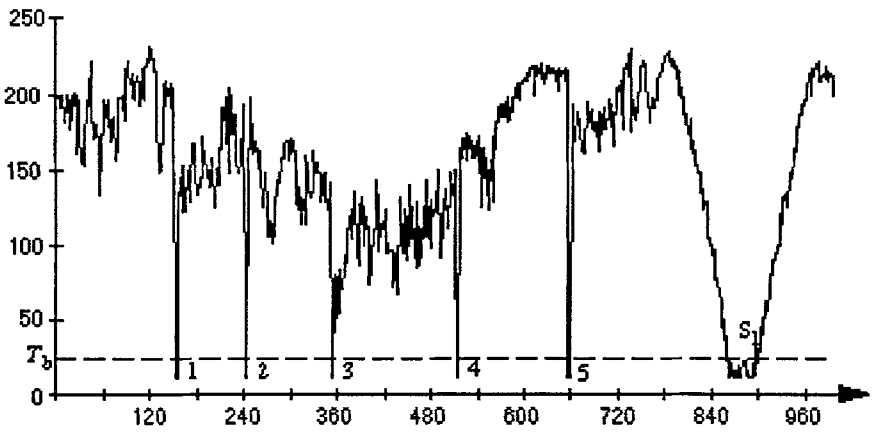
The sequences of differences between all successive **I** frames from an MPEG compressed documentary video, defined by both DCT correlation and pair-wise block comparison, are shown in two parts in Figs. 3 and Fig. 4, (a) and (b), respectively. This particular sequence



(a)



(b)



(c)

Figure 3. Video parsing result using three algorithms on the first half of a test video compressed in MPEG: (a) DCT coefficient correlation using difference value Ψ, as computed by equation (2); (b) pair-wise block comparison of DCT coefficients based on difference value $D(i, i + \varphi)$, as used in (5); and (c) motion-based comparison based on M, the number of valid motion vectors, as used in (6). In (a) and (b) difference values are computed between successive **I** frames. There is 1 **I** frame, 1 **P** frame and 4 **B** frames in every 6 frames in the MPEG representation; and motion vectors for (c) are examined in both **P** and **B** frames. T_b is the threshold for detecting a camera break. T_t is the lower threshold for the twin-comparison approach discussed in Section 3.4.1.

contains sharp breaks, gradual transitions, camera operations, and object motion, making it an excellent test case. While it is only about 1.3 minutes long, there are 25 shots separated by 17 sharp breaks and 7 gradual transitions. Also 4 shots involve camera panning. The 17 sharp breaks are labeled 1–17, and they can all be detected by exceeding an appropriately set break threshold T_b. Specific images from this sequence will be presented in the discussion in Section 3.4.1.

However, neither of these two algorithms can handle gradual transitions or false positives introduced by camera operations and object motion. Both Fig. 3(b) and Fig. 4(b) illustrate examples of these problems, labeled $\mathbf{T}_1$ through $\mathbf{T}_7$; but we shall not discuss how they may be solved until Section 3.4. Instead, we now turn to motion vectors as a source of partitioning information.

3.3. Partitioning based on motion vectors

Apart from intensity values and distributions, motion of both objects and the camera are the major elements of video content. In general within a single camera shot, the field of motion vectors should show relatively continuous changes, while this continuity will be disrupted between frames across different shots. Thus, a continuity metric for a sequence of motion vector fields should be able to serve as an alternative criterion for detecting segment boundaries.

In an MPEG data stream, as was illustrated in Fig. 2, there are two sets of motion vectors associated with each **B** frame, forward and backward, and a single set of motion vectors associated with each **P** frame. That is, each **B** frame is predicted and interpolated from its

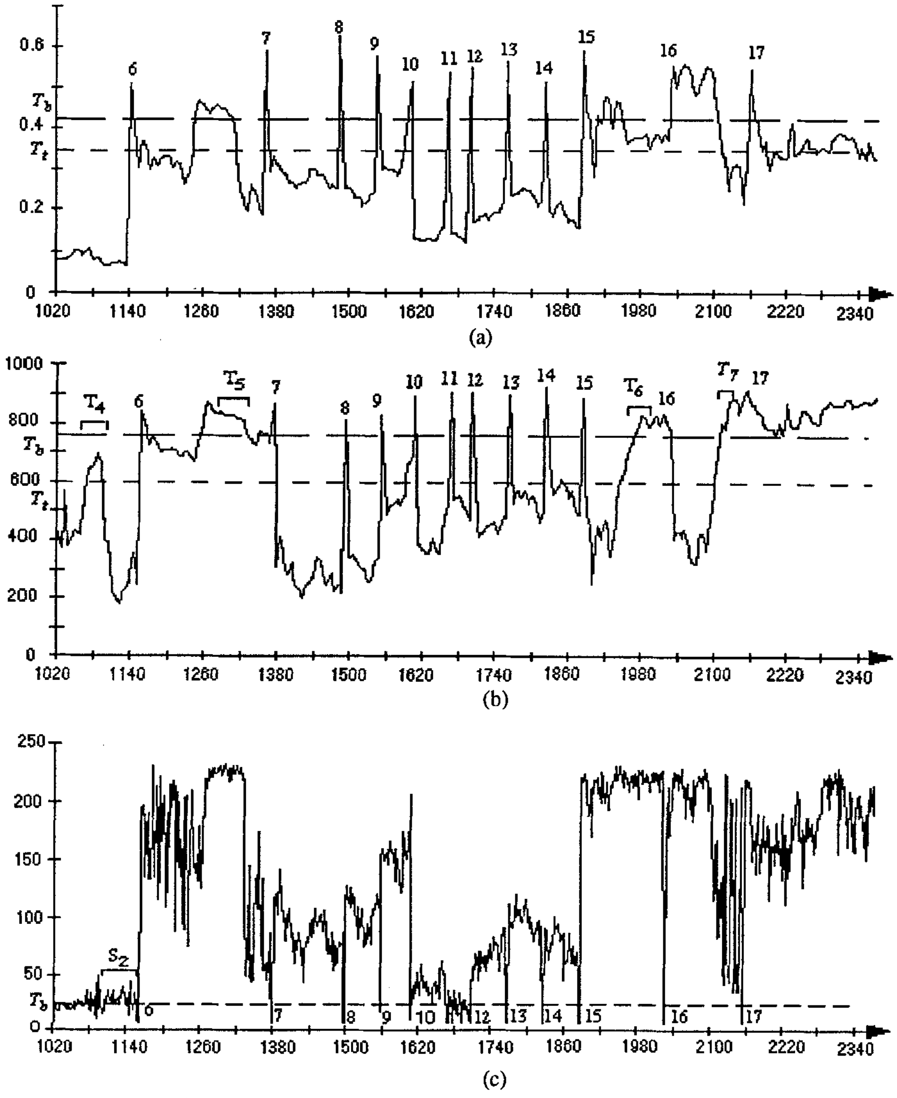


Figure 4. Video parsing results using three algorithms on the second half of a test video compressed in MPEG: (a) DCT coefficient correlation using difference value Ψ, as computed by equation (2); (b) pair-wise block comparison of DCT coefficients based on difference value $D(i, i + \varphi)$, as used in (5); and (c) motion-based comparison based on M, the number of valid motion vectors, as used in (6). T_b is the threshold for detecting a camera break. T_t is the lower threshold for the twin-comparison approach discussed in Section 3.4.1.

preceding and succeeding **I/P** frames by motion compensation; and each **P** frame is similarly predicted from its preceding **I/P** frame. In both cases the residual error after motion compensation is then transformed into DCT coefficients and coded. However, if this residual error exceeds a given threshold for certain blocks, motion compensation prediction is abandoned; and those blocks are represented by DCT coefficients, as in **I** frames. Subsequently, there will be no motion vectors associated with those blocks in the **B/P** frames. Such high residual error values are likely to occur in many, if not all, blocks across a camera shot boundary; and we have developed an algorithm which uses this information.

Let M be the number of valid motion vectors for each **P** frame and the smaller of the numbers of valid forward and backward motion vectors for each **B** frame, and let T_b be a threshold value close to zero. Then

$$M < T_b \tag{6}$$

will be an effective indicator of a camera boundary before or after (depending on whether interpolation is forward or backward) the **B/P** frame. The difference sequences in Fig. 3(c) and Fig. 4(c) illustrate these values of M for the same video example used for data in the other graphs in these illustrations. In this case the camera breaks are all accurately represented as valleys below the threshold level, labeled 1 through 17.

However, this algorithm fails to detect gradual transitions because the number of motion vectors during such a transition sequence is often much higher than the threshold. The algorithm also yields false positives, indicated by the $\mathbf{S_1}$ and $\mathbf{S_2}$ valleys. These correspond to sequences of repeating static frames: When there is no motion, all the motion vectors will be zero, yielding a false detection. A simple resolution to this problem is to measure the width of the valley. For a camera break, the valley is usually very deep and narrow, since the small number of motion vectors will be confined to at most two successive frames; on the other hand, any static sequence which is longer than two frames will yield a wider valley. However, this may eliminate some gradual transitions with a long sequence of frames with very few motion vectors. The robust solution is to use difference metric (5) to examine the potential false positives, since $D(i, i + \varphi)$ will be close to zero between the stationary frames.

3.4. A new hybrid approach

We now present a new hybrid approach which integrates the approaches discussed in Sections 3.2 and 3.3 and incorporates the multi-pass strategies and motion analyses proposed by Zhang et al. [14] to improve detection accuracy and processing speed; we shall base our discussion on video compressed with MPEG.

3.4.1. Multiple passes and multiple comparisons. The first step is to apply a DCT comparison, such as the one defined by (5), to the **I** frames with a large skip factor φ to detect regions of potential gradual transitions, breaks, camera operations, or object motion. The large skip factor reduces processing time by comparing fewer frames. Furthermore, gradual transitions are more likely to be detected as potential breaks, since the difference between two more "temporally distant" frames could be larger than the break threshold, T_b. The drawbacks of using a large skip factor, false positives and low temporal resolution for shot boundaries, are then recovered by a second pass with a smaller skip factor (which may be 1, i.e. comparing consecutive **I** frames); but the second pass is only applied to the neighborhood of the potential breaks and transitions. Thus a high processing speed is achieved without losing either detection accuracy or temporal resolution of segment boundaries.

However, it should be pointed out that choosing a proper skip factor depends on the dynamic features of the movie to be parsed. In our test example the rapid changes due to moving objects and camera motion make a large skip factor relatively inadequate. Since,

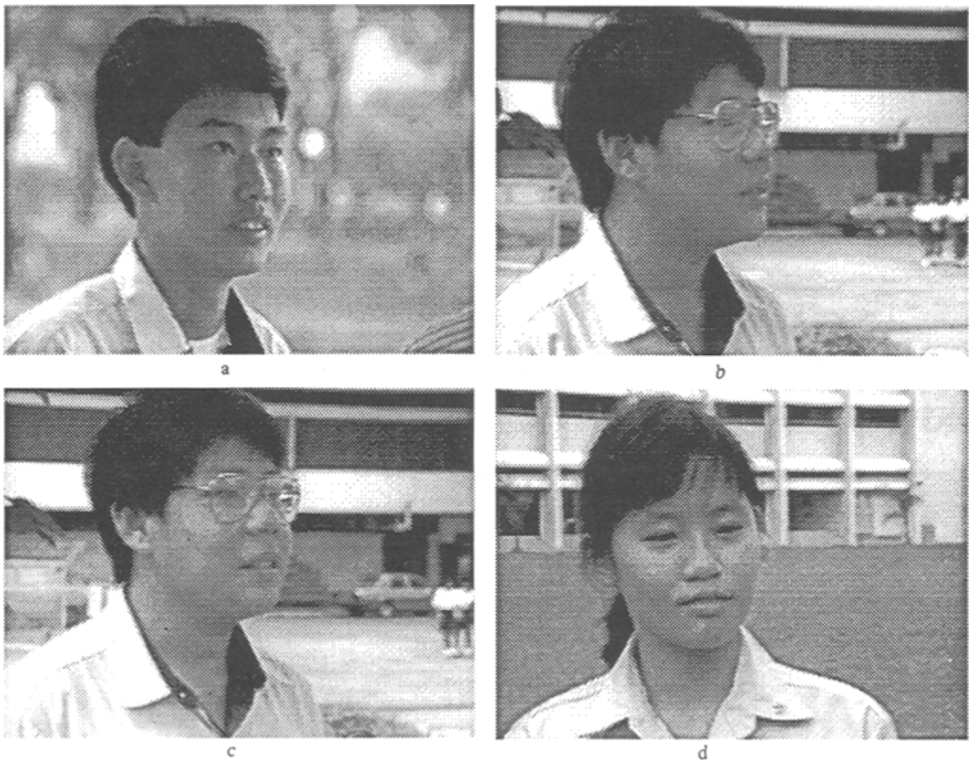


Figure 5. Transitions for breaks 2 and 3 as indicated in the graphs in Fig. 3. Break 2 is illustrated in the upper pair of images. Break 3 is illustrated in the lower. Break 3 was detected by DCT coefficient correlation, while break 2 was missed but was detected by motion vector comparison.

in general, the frame interval between consecutive **I** frames in MPEG movies is larger than 6 frames, an even larger skip factor will not increase the overall processing speed because of the number of false positives introduced.

What makes this a hybrid approach is that the motion-based comparison metric is also applied as another pass, either on the entire video or only on the sequences containing potential breaks and transitions, to complement and verify the results from DCT comparison and to improve further the accuracy and confidence of the detection results. More specifically, difference metric (6) is applied to all the **B** and **P** frames of those selected sequences to confirm and refine the break points and transitions detected by the DCT comparison. Conversely, the DCT results provide information for distinguishing between sequences of static frames and transition frames which could be confused by motion-based detection as observed in Section 3.3. A good example of an advantage of this combined approach is break 2 in Fig. 3(a), which is missed by the DCT correlation algorithm but is clearly recognized from motion vector comparison. The images compared across this break (as well as those across break 3 for control) are illustrated in Fig. 5.

Gradual transitions may be detected by an adaptation of our twin-comparison approach [13]. For example $\mathbf{T}_4$ in Fig. 4(b) can be correctly detected as a gradual transition (illustrated in Fig. 6), since the DCT differences between **I** frames exceed the lower threshold, T_t,

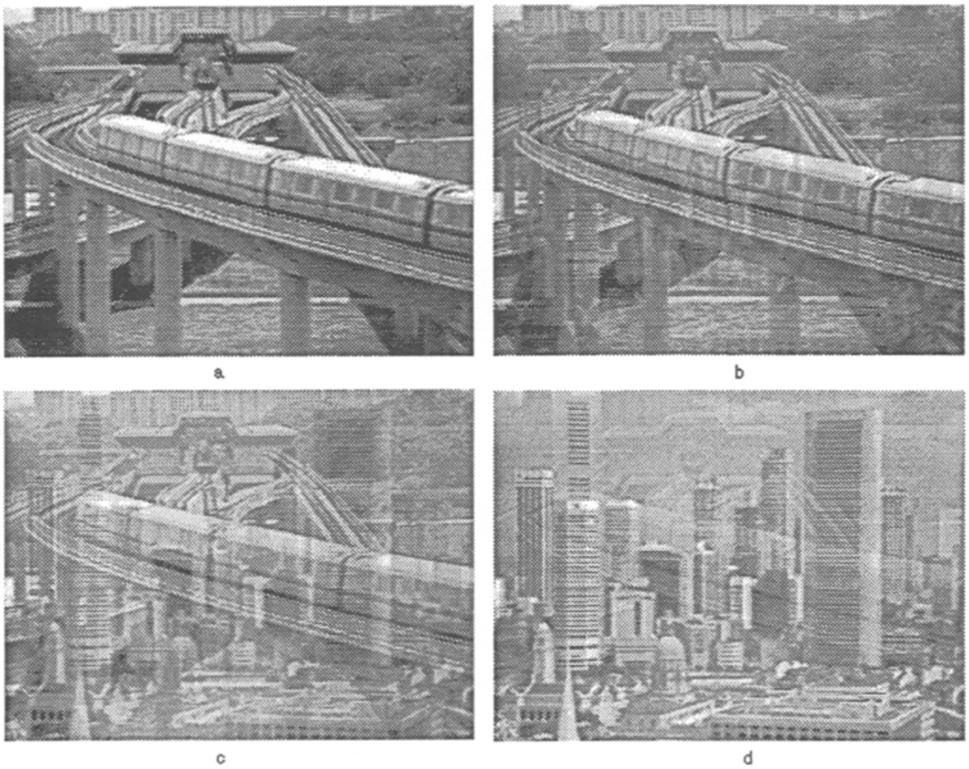


Figure 6. A gradual transition corresponding to segment $\mathbf{T}_4$ in Fig. 4(b).

introduced by the twin-comparison approach [13]. On the other hand $\mathbf{T}_1$ in Fig. 3(b) is a camera pan (illustrated in Fig. 7) whose difference values are higher than the break threshold, due to the large frame interval between **I** frames (6 in our case). However, because motion vector analysis is very robust in detecting sharp breaks, this false positive can be detected. Therefore, by fusion of the information gathered from different passes with different comparison metrics, the false positives and missing transitions, resulting from using any of the three metrics alone, may be resolved.

3.4.2. Camera operation and object motion detection in the hybrid approach. Another video parsing problem consists in distinguishing sequences involving camera operations from those due to gradual transitions, since the former tend to induce temporal variations in frame content of the same order as do the latter. Camera operation information figures significantly in the analysis and classification of video shots, since camera operations often explicitly reflect the communication intentions of the director. Detecting camera operation is also very important in constructing images which effectively represent video content, such as *salient stills* [9].

Camera operations may be classified according to whether or not camera position changes. The camera is fixed for panning and tilting (horizontal or vertical rotation) and zooming (focal length change); the analogous operations in which the camera moves are tracking

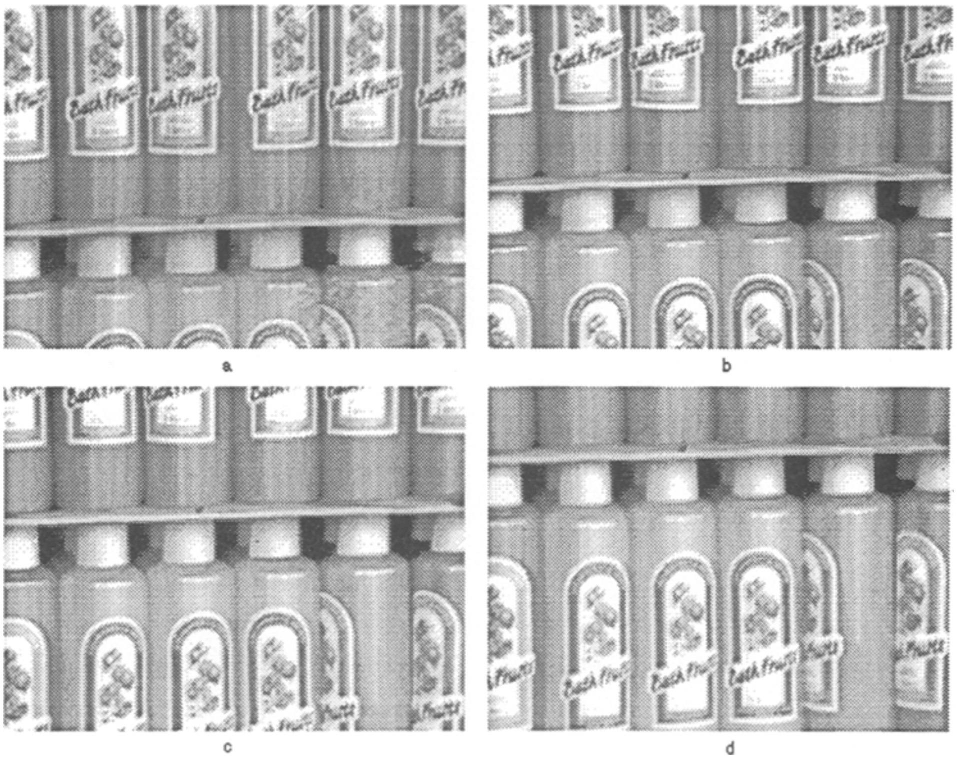


Figure 7. A camera pan corresponding to segment $\mathbf{T}_1$ in Fig. 3(b).

and booming (horizontal and vertical transverse movement) and dollying (horizontal lateral movement). These operations may also be applied in combinations. Each of these operations induces a specific pattern in the field of motion vectors; so by detecting those different patterns, we can derive the corresponding camera operations. We have implemented two algorithms: motion vector pattern analysis for isolating camera operation sequences; and a regression algorithm to calculate panning and zooming parameters once the sequences are located, if these parameters are needed.

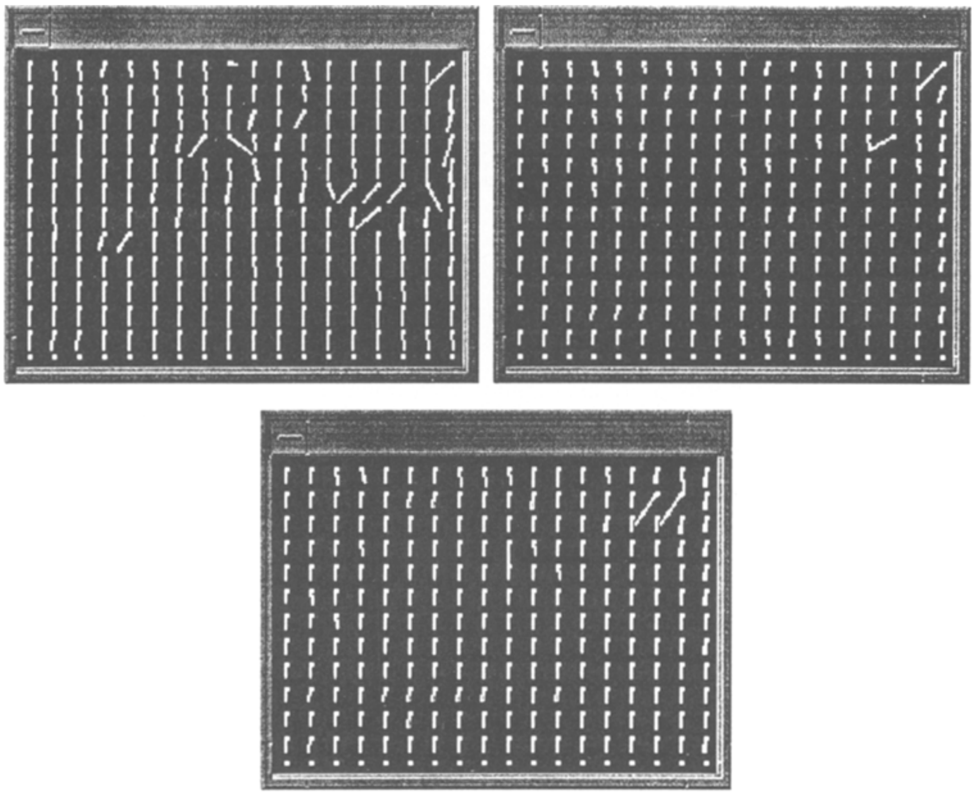
Figure 8 shows the sorts of representative patterns which analysis must be able to detect [13]: these are MPEG **B** frames extracted from a tilting sequence and a zooming sequence. The motion vector field for any combination of panning and tilting will exhibit a single strong modal vector value which corresponds to the direction of the camera movement; so most of the motion vectors will be parallel to this modal vector. This may be expressed by the formula

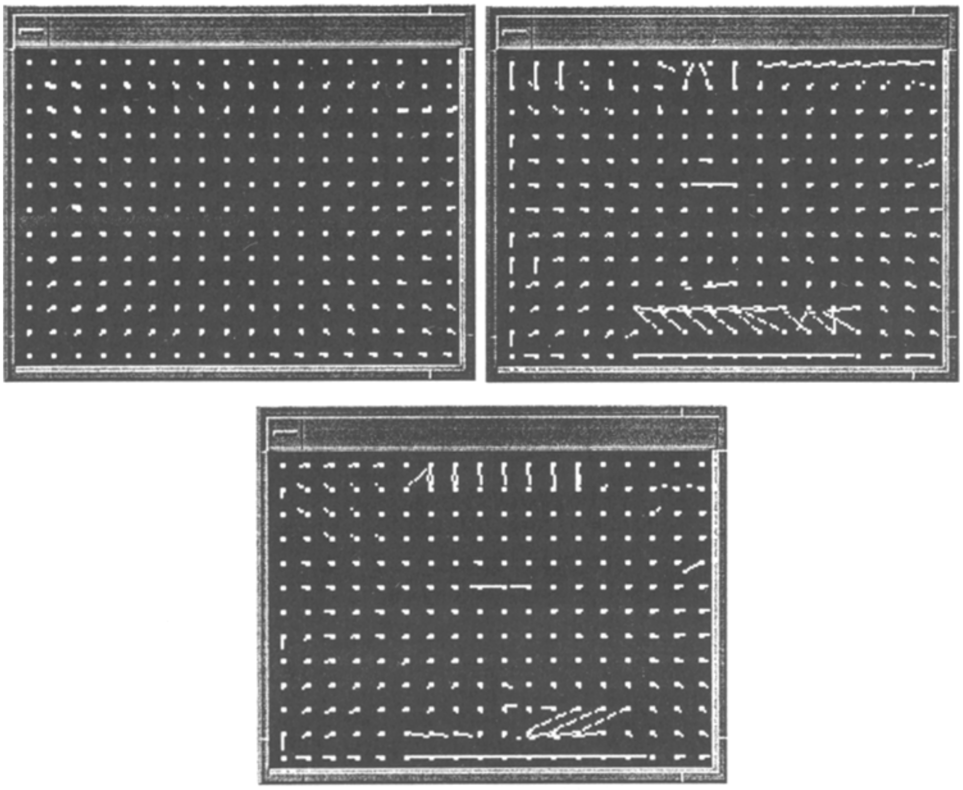
$$\sum_{k}^{N} |\theta_k - \theta_m| \leq T_P \tag{7}$$

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. (7) thus counts the total variation in direction of all motion vectors from the direction of the modal vector; and a camera pan/tilt is declared if this variation is less than T_P.

On the other hand, the field of motion vectors resulting from zooming creates a focus center. As one can observe from Fig. 8(b), the vertical components of motion vectors in the top and bottom rows of a frame will have opposite signs if there is a zooming. Mathematically, this means that, in every column, the magnitude of the difference between these vertical components will always exceed the magnitude of both components. The horizontal components of the motion vectors for the leftmost and rightmost columns can then be analyzed the same way. Therefore, a zooming can be detected by comparing the components of motion vectors; and a zoom is detected if the number of positive comparisons exceeds a threshold T_Z. However, if there is a combination of panning and zooming, the motion pattern will become much more complicated and difficult to detect. Fortunately, such combinations are not often used in movie making, and one operation is usually much more rapid than the other. Thus such sequences will be classified into one of the two operations, rather than the combination.

A potential problem is that pan/tilt detection may yield false positives from video sequences containing object motion rather than camera motion. This will happen when there is a large object or a group of small objects moving in the same direction, thus resulting


(a)



(b)

Figure 8. Typical motion vector field patterns resulting from (a) camera tilting and (b) zooming-out, plotted from motion vectors associated with three **B** frames.

in the same pattern of a large group of motion vectors with the same direction. To be able to distinguish between these two types of sequences, we further divide the motion field of each frame into a number of macro blocks and then apply the motion analysis to each block. The sequence is only declared a panning sequence if the mode directions of all of the blocks agree. Figure 9 shows an example of a motion field pattern resulting from object motion. However, if the moving object(s) cover most of the frame, then the algorithm will falsely interpret the movement as a camera pan/tilt.

Figures 10 and 11 show the detection of camera panning and zooming on the same video sequence shown in Figs. 3 and 4. No zooming has been detected by the algorithm which is consistent with the content of this particular portion of the video. On the other hand, several panning sequences are detected, marked as $\mathbf{P_1}$ to $\mathbf{P_4}$. ($\mathbf{P_1}$ is the same as the $\mathbf{T_1}$ sequence from Fig. 3(b), illustrated in Fig. 7.) The camera operation information is added to our video parsing process to improve the capability of the hybrid approach in eliminating the false detection of transitions, as shown in Figs. 3 and 4.

In case we need to describe the camera operation quantitatively, we use an algorithm developed for detecting global motion between frames [10]. This algorithm is derived by

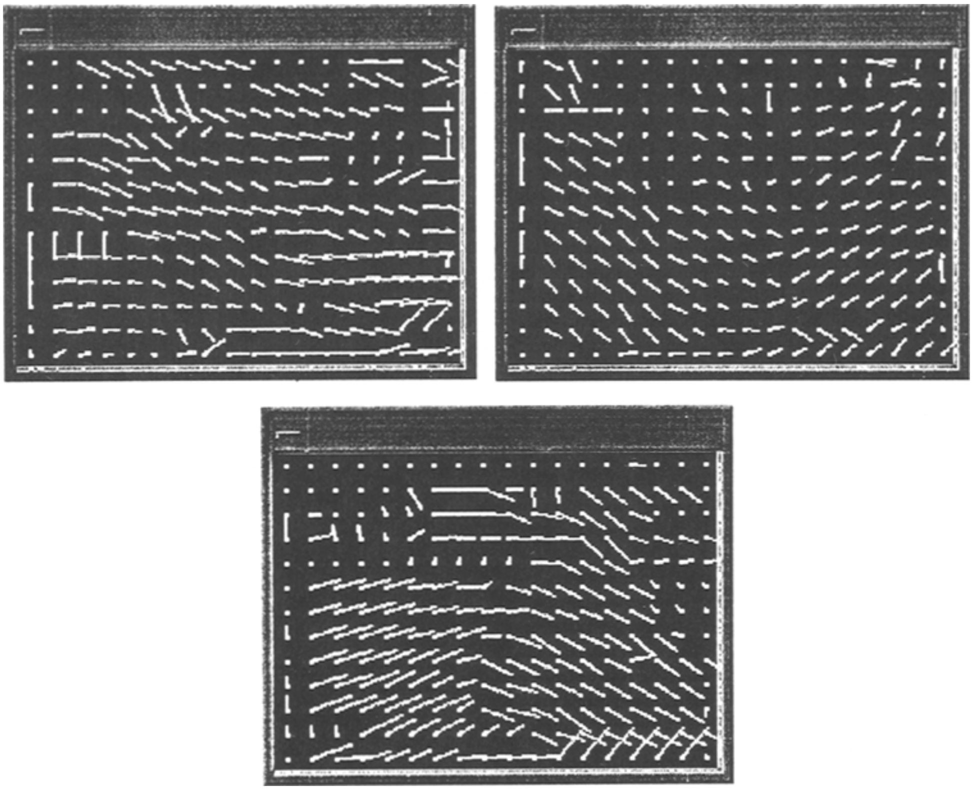


Figure 9. An example of motion vector field patterns resulting from moving objects: note the different motion patterns in different regions of the frames.

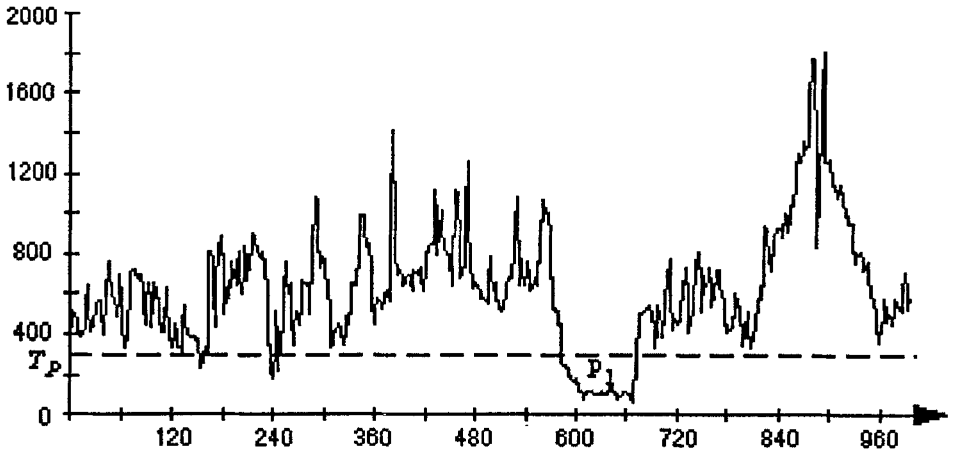
analyzing the relations between the camera coordinates and the image-space coordinates, and the changes of the pixel coordinates introduced by camera panning and zooming. A zoom can be represented by a *zoom factor*, $\mathbf{f_z}$; and a camera pan/tilt can be represented by a *panning vector*, **p**. Thus if we have a set of N designated points in a frame, if $\mathbf{U_i}$ represents the horizontal and vertical coordinates of the **i**th of these points in that frame, and $\mathbf{U'_i}$ represents the coordintates of the same point as displaced in the following frame, then the effect of combination of these operations can then be expressed by

$$\mathbf{U'_i} = \mathbf{f_z U_i} + \mathbf{p} \tag{6}$$

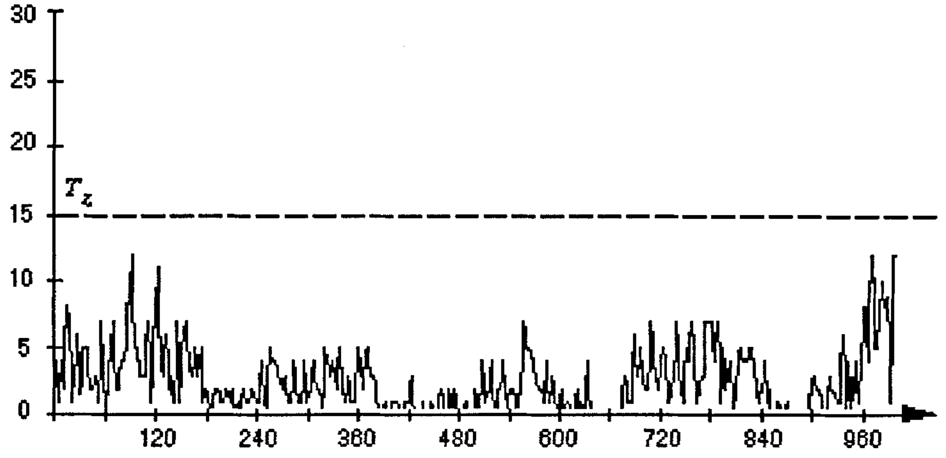
$\mathbf{f_z}$ and **p** can be calculated by an iterative algorithm based on the set of all N displaced points [10]:

$$\mathbf{f_z} = \frac{\sum_i |\mathbf{U_i} \bullet \mathbf{U'_i}| - \frac{1}{N} |\sum_i \mathbf{U_i} \bullet \sum_i \mathbf{U'_i}|}{\sum_i |\mathbf{U'_i} \bullet \mathbf{U'_i}| - \frac{1}{N} |\sum_i \mathbf{U'_i} \bullet \sum_i \mathbf{U'_i}|} \tag{7.1}$$

$$\mathbf{p} = \frac{1}{N}\left(\sum_{\mathbf{i}} \mathbf{U'_i} - \mathbf{f_z} \sum_{\mathbf{i}} \mathbf{U_i}\right) \tag{7.2}$$



(a)

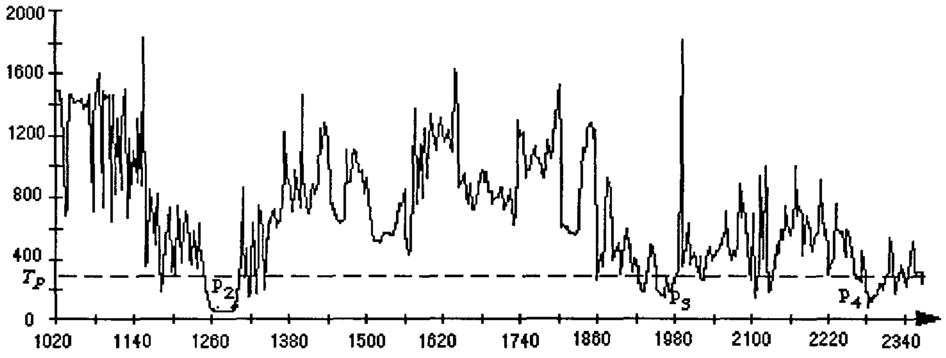


(b)

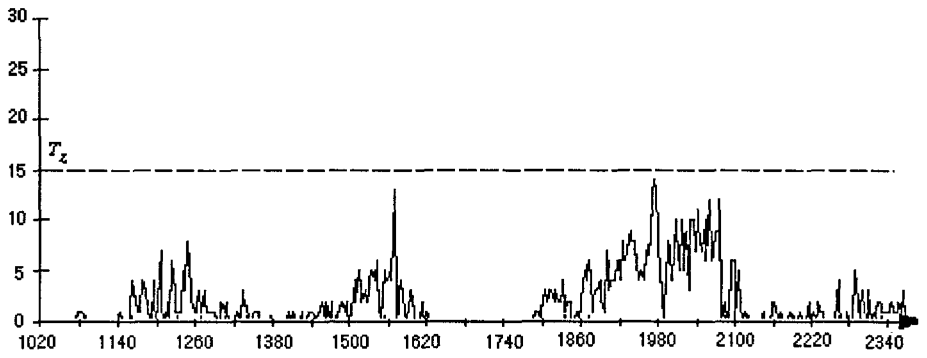
Figure 10. (a) A sequence of differences in motion direction from the modal vector direction for the frames graphed in Fig. 3. T_P is the threshold for panning detection. (b) A sequence of numbers of blocks whose motion vectors satisfy the criterion for zooming detection for the frames graphed in Fig. 3. The threshold T_Z is set at 15.

Because this computation is very intensive, we restrict it to the sequences which have already been identified by the simple motion analysis algorithm.

So far, we have not tried to distinguish panning/tilting from tracking/booming or zooming from dollying. This is because our primary interest has been in separating camera operation sequences from gradual transitions. A more thorough classification requires a more detailed analysis of variations in vector magnitudes which reflect the distances between different objects and the camera. This is a far more compute-intensive process, which may be further complicated by object motion. More sophisticated motion recovery algorithms are needed to tackle these problems [3]. Similarly, we have not yet undertaken a systematic study of combinations of gradual transitions, such as a fade out of one shot while the succeeding shot



(a)



(b)

Figure 11. (a) A sequence of differences in motion direction from the modal vector direction for the frames graphed in Fig. 4. T_P is the threshold for the panning detection. (b) A sequence of numbers of blocks whose motion vectors satisfy the criterion for zooming detection for the frames graphed in Fig. 4. The threshold T_Z is set at 15.

wipes in. We anticipate that such combinations will be recognized as gradual transitions by our current techniques, but this hypothesis remains to be tested.

3.5. Summary of results

The results presented in Figs. 3 and 4 may now be summarized in tabular form. (Recall the summary of the test video at the end of Section 3.2.) First let us consider the detection of camera breaks:

	Detected Camera Breaks	Undetected Camera Breaks	Falsely Detected Camera Breaks
DCT1	16	1	4
DCT2	17	0	4
motion vector	17	0	0
hybrid	17	0	0

In this chart DCT1 refers to the DCT correlation comparison of [1], while DCT2 is our own pair-wise block comparison technique. We may summarize the detection of gradual transitions in a similar chart as follows:

	Detected Transitions	Undetected Transitions	Falsely Detected Transitions
DCT1	3	5	1
DCT2	4	4	1
motion vector	0	0	0
hybrid	7	0	0

Finally, our technique for detecting camera movements, as illustrated in Figs. 10 and 11, detected all four of the camera pans in our test data.

4. Content-based browsing tools for compressed video

The results of our video parsing techniques can be utilized to support indexing and browsing. The segment boundaries serve as pointers to each basic indexing unit of video data. The descriptions of camera operations can be used for representative frame construction in shot content representation, and they also provide another level of indexing that can be used as a retrieval key. We now look at how we utilize the information obtained from video parsing in content-based video retrieval and fast browsing.

An important function to facilitate video information retrieval is a tool for fast browsing of any video material retrieved from a query. Such browsing should take place at two levels of temporal granularity—overview and detail—and should be implemented in a way which facilitates movement between these two levels. Detailed browsing should be based on VCR-like functions, particularly fast forward and reverse scanning. This requires decompression of all frames of a compressed source, which is computationally intensive but not content-based. Overview browsing, on the other hand, can be based on content change information obtained from the algorithms discussed in Section 3.

Decompressing and displaying MPEG **I** frames is one of the simplest approaches to fast browsing. However, since the **I** frames are uniformly distributed in time, it is not content-based. A more content-based approach would decompress only the representative frames, or key frames, of each shot of the parsed video data. However, if we select only one key frame for every shot, regardless of the content or how it changes in the shot, the browsing may be too coarse; and, more importantly, the dynamic feature of the video may be lost. Thus, the problem is how to detect an appropriate set of key frames.

Using key frames to represent video content has been proposed by many researchers, such as in [6]; but the problem of how to effectively extract key frames based on video content automatically has not really been addressed. We have developed a unique and robust content-based key frame extraction technique which utilizes the results of video parsing.[2] The number of key frames extracted for each shot depends on the shot content, its variations, and the camera operations involved. The technique is based on the use of a selected difference metric, similar to those used for detecting camera breaks. However, in this case all computation takes place within a single camera shot and differences between consecutive frames are not computed. Instead, the first frame is proposed as a key frame; and consecutive frames are compared against that candidate. A two-threshold technique,

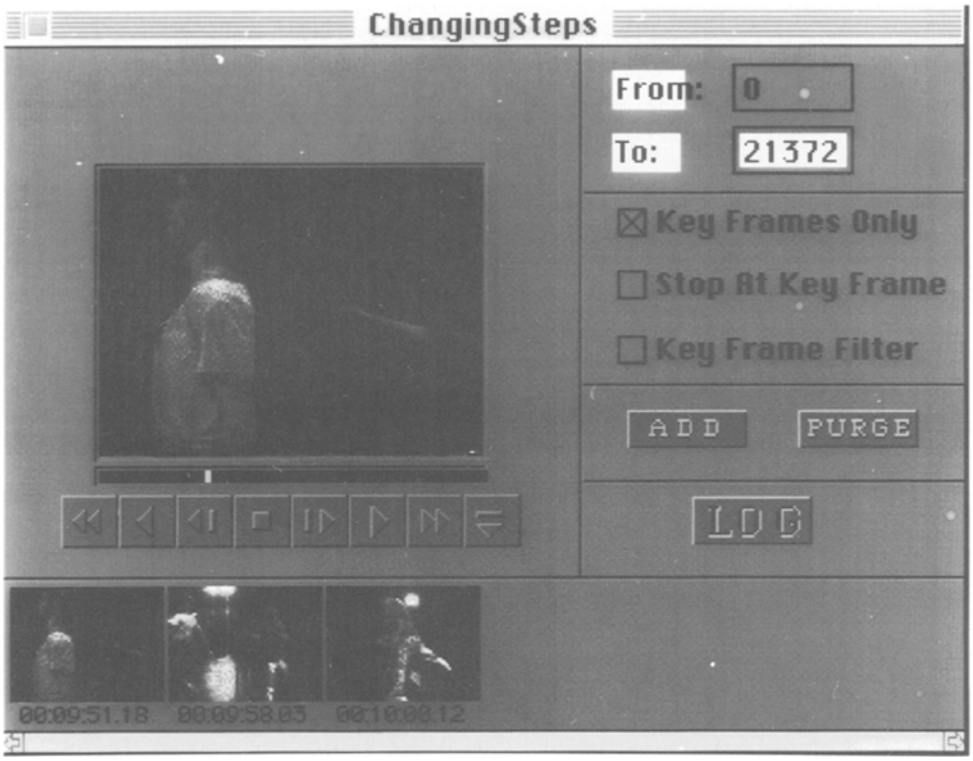


Figure 12. A video browser based on extracted key frames.

similar to the twin-comparison method in [13], is applied to identify a frame significantly different from the candidate; and that frame is proposed as another candidate, against which successive frames are compared. Users can specify the density of the detected key frames by adjusting the two threshold values.

Once the key frames have been extracted, they may be viewed with two types of browsing tools: a sequential browser and a hierarchical browser. The sequential browser resembles a "virtual" VCR, supporting conventional play modes; but the fast forward and reverse play will decompress only the **I** frames of an MPEG video. However, as shown in Fig. 12, there is a special mode that plays only the key frames of each shot. This mode achieves a much faster play speed than normal fast forward or reverse play, while the content of the video is still preserved. A quantitative example is that a half-hour stock footage video may be represented by about 200 key frames, and the content of this video can be viewed within less than 7 seconds if we play at 30 frames per second. As one can see from Fig. 12, the key frames of each shot can also be viewed in detail, as they can be displayed in the bottom of the display window while the video is playing.

The hierarchical browser is designed to provide random access to any point in a given video: a video sequence is spread in space and represented by frame icons which function rather like a light table of slides. In other words the display space is traded for time to provide a rapid view of the content of a long video. As shown in Fig. 13, at the top of the hierarchy, a whole video is represented by five key frames, each corresponding to an

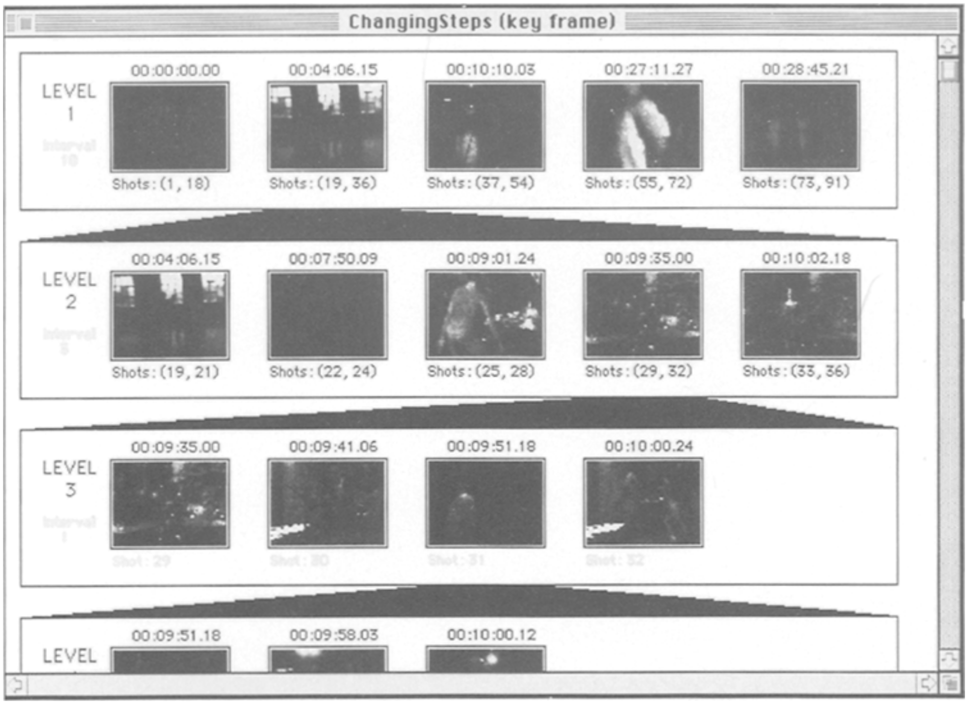


Figure 13. A content-based hierarchical browser.

equal number of camera shots. Unlike the Hierarchical Video Magnifier [4], which inspired our design, our browser is based on the results of video parsing and key frame extraction. As we descend through the hierarchy, our attention focuses on smaller groups of shots, single shots, the representative frames of a specific shot, and finally a sequence of frames represented by a key frame. Of course, we can always open the first type of video player to view sequentially any particular segment of video selected from this browser at any level of the hierarchy, provided that the (compressed) video is stored on line.

Another advantage to incorporating key frames in such browsing tools, including the sequential browser presented above, is that we can browse the video content down to the key frame level without necessarily storing the entire video, either compressed or not. This is particularly advantageous if our storage space is limited. Such a feature is not only very useful in video databases and information systems, but also in video on demand systems to support previewing. Key frames can also be used for visual content based retrieval [15]: one can specify a visual query based on the key frames and retrieve the video sequences represented by those key frames.

5. Concluding remarks

We have presented two novel algorithms for parsing JPEG or MPEG compressed video: a DCT coefficient based pair-wise block comparison and a motion vector comparison.

We also presented a new hybrid approach which integrates the above two algorithms and incorporates a multi-pass strategy and motion analyses to improve accuracy and processing speed. With such an approach, we can effectively detect camera breaks, gradual transitions, camera operations, and object motion using compressed video data directly, all of which are the most important tasks of video parsing and provide necessary information for automatic content-based video browsing and retrieval. Finally, we have presented two content-based video browsing tools which utilize the information, particularly about shot boundaries and key frames, obtained from parsing. Such tools enable us to rapidly browse and locate video sequences from compressed video according to the content of the video. Experimental evaluation of these algorithms has shown that they are effective and able to achieve high accuracy in parsing compressed video. These algorithms are currently being integrated into our video parsing, indexing and retrieval system [15].

Acknowledgments

The MPEG codec used in compressing the video data is from The Berkeley Plateau Research Group at the Department of Electrical Engineering and Computer Science of the University of California at Berkeley. The images in Figs. 5, 6, and 7 were taken from a documentary about engineering at the National University of Singapore. Those in Figs. 12 and 13 are from Merce Cunningham's "Changing Steps" and have been used with permission from the Cunningham Dance Foundation.

Notes

1. This discussion assumes that a video consists only of *image* content. However, most videos also have soundtracks. All three tasks must be addressed in the auditory domain as well: identifying the audio associated with each video chip, analyzing that audio for low-level properties, and supporting browsing and retrieval based on audio content.
2. US Patent pending.

References

1. F. Arman, A. Hsu, and M.-Y. Chiu, "Image Processing on Compressed Data for Large Video Databases," Proc. ACM Multimedia 93, Anaheim, CA, pp. 267–272, 1993.
2. D. Le Gall, "MPEG: A Video Compression Standard for Multimedia Applications," Communications of the ACM, 34 (4), pp. 46–58, 1991.
3. R.M. Haralick and L.G. Shapiro, "Computer and Robot Vision," Vol. 2, Addison-Wesley, Reading, MA, 1993.
4. M. Mills, J. Cohen, and Y.Y. Wong, "A Magnifier Tool for Video Data," Proc. CHI '92, Monterey, CA, pp. 93–98, 1992.
5. A. Nagasaka and Y. Tanaka, "Automatic Video Indexing and Full-Video Search for Object Appearances," Visual Database Systems, II, E. Knuth and L.M. Wegner, editors, North-Holland, pp. 119–133, 1991.
6. B.C. O'Connor, "Selecting Key Frames of Moving Image Documents: A Digital Environment for Analysis and Navigation," Microcomputers for Information Management, 8(2), pp. 119–133, 1991.
7. R. Steinmetz, "Data Compression in Multimedia Computing—Standards and Systems," Multimedia Systems, 1 (4), pp. 187–204, 1994.
8. D. Swanberg, C.-F. Shu, and R. Jain, "Knowledge Guided Parsing in Video Databases," Proc. IS&T/SPIE Conf. on Storage and Retrieval for Image and Video Databases, San Jose, CA, 1993.

9. L. Teodosio and W. Bender, "Salient Video Stills: Content and Context Preserved," Proc. ACM Multimedia 93, Anaheim, CA, pp. 39–46, 1993.
10. Y.T. Tse and R.L. Baker, "Camera Zoom/Pan Estimation and Compensation for Video Compression," Proc. SPIE Conf. on Image Processing Algorithms and Techniques II, Boston, MA, pp. 468–479, 1991.
11. H. Ueda, T. Miyatake, and S. Yoshizawa, "IMPACT: An Interactive Natural-Motion-Picture Multimedia Authoring System," Proc. CHI'91, New Orleans, LA, pp. 343–350, 1991.
12. G.K. Wallace, "The JPEG Still Picture Compression Standard," Communications of the ACM, 34(4), pp. 30–44, 1991.
13. H.J. Zhang, A. Kankanhalli, and S.W. Smoliar, "Automatic Partitioning of Full-motion Video," Multimedia Systems, 1(1), pp. 10–28, 1993.
14. H.J. Zhang, C.Y. Low, Y. Gong, and S.W. Smoliar, "Video Parsing Using Compressed Data," Proc. IS&T/SPIE Conf. on Image and Video Processing II, San Jose, CA, pp. 142–149, 1994.
15. H.J. Zhang and S.W. Smoliar, "Developing Power Tools for Video Indexing and Retrieval," Proc. IS&T/SPIE Conf. on Storage and Retrieval for Image and Video Databases II, San Jose, CA, pp. 140–149, 1994.

HongJiang Zhang received his Ph.D. from the Technical University of Denmark in 1991 and his B.Sc. from Zhengzhou University, China, in 1981, both in Electrical Engineering. He had worked as an engineer at Shijizhuang Communication Institute, China, before his Ph.D. work. He joined the Institute of Systems Science, National University of Singapore in December 1991, and is presently leading research on image/video content analysis, representation and retrieval in the Video Classification Project. He is also leading a project on tracking and classification of moving objects. His current research interests include image/video processing, computer vision, interactive video, and their applications in multimedia databases and systems, and has published over 30 papers in these areas. Dr. Zhang is a member of IEEE, and a member of the Editorial Board of the international journal, "Multimedia Tools and Applications," published by Kluwer Academic Publishers.

Low Chien Yong obtained his B.Eng. (Hons) in Electrical Engineering from the National University of Singapore in 1989. From 1990 to 1992, he was working in Japan on multimedia applications. Since 1993, he has been with the Institute of Systems Science, National University of Singapore, where he is working on projects on video classification, image processing and multimedia.

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, in 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).

Dr. Smoliar also has extensive background in music and is the sole member in Singapore of the Society for Music Theory. His main areas of research interest are in knowledge representation, perceptual categorization and cognitive models. He is currently interested in applying artificial intelligence to advanced media technologies. He joined ISS in May 1991 and is presently leading a project on video classification. He is also leading another project concerned with using information technology as a teaching medium in music schools.