Multimedia Systems (1994) 1:211–219

Multimedia Systems

Image processing on encoded video sequences

Farshid Arman*, **Arding Hsu, and Ming-Yee Chiu**

Siemens Corporate Research Inc., College Road East, Princeton, NJ 08540, USA

Abstract. This paper presents a novel approach to processing encoded video sequences prior to complete decoding. Scene changes are easily detected using DCT coefficients in JPEG and MPEG encoded video sequences. In addition, by analyzing the DCT coefficients, regions of interest may be isolated prior to decompression, increasing the efficiency of any subsequent image processing steps, such as edge detection. The results are currently used in a video browser and are part of an ongoing research project in creating large video databases. The procedure is detailed with several examples presented and studied in depth.

Key words: Video processing – Scene change detection – Video browsing

1 Introduction

For systems incorporating encoded video, such as video editing systems, various multimedia authoring systems, video-based training systems, and video on demand systems, the ability to *manage* video efficiently is critical. Although many of these systems incorporate many other types of media as well, management of video is particularly challenging because of the vast volume of data associated with it – many megabytes of data per *minute*. The initial steps taken in solving the video management problem have either relied on labor intensive techniques, such as manually entering key words to describe the video contents, or on simple image-processing techniques, such as analyzing histograms. These approaches are neither close to ideal, nor are they efficient in their tasks. Key words have many drawbacks [3], such as inadequate choice of terms to use at search time, the context in which the words are used, and the influence of the operator, while image processing steps cannot be efficiently applied to the hundreds of thousands of images that are usually associated with video. This paper presents techniques aimed at the management of encoded video, such as MPEG [6], JPEG [21], and H.261 [10], which overcome the limitations of traditional image-processing steps, while enhancing key word-based approaches currently in wide use.

* e-mail number:farshid@scr.siemens.com
Correspondence to: F. Arman

Subtasks of video management include the ability to locate a particular video sequenc quickly – high-level video management – *and* the ability to view particular points of interest within the video sequence – low-level video management. The need for management of video exists in many domains from TV news organizations where these capabilities are critical, to home video libraries where these capabilities would be very useful. The focus of this paper is on low-level management techniques for digital video. Currently, the most widely used search technique is to fast-forward and rewind to arrive at the point of interest; this technique is slow and inefficient. More recently, image-processing techniques have been developed to operate on digital video in order to facilitate this task. The first step in solving this problem is to "divide" the video sequence into meaningful segments much like a book can be divided up into sentences. In video, the logical point to partition the video sequence is where the contents of video "change" from one frame to the next – referred to as a *scene change.*

Specifically, this paper presents a novel approach to efficient processing of encoded video for detecting both scene changes of a video sequence and regions of interest within each video frame, which can be used in high-level video management modules. It is reasonable to assume the input to the system is encoded since digital video requirements for storage and communication are extremely high; hence, many standards for video encoding have been developed and are currently in use. Our approach, selective decoding, takes advantage of the information already encoded in a DCT-based compressed video data, and performs many processing steps needed on every frame of a video sequence prior to full decompression. DCT coefficients are analyzed to detect scene changes systematically – referred to as frame selection (see Fig. 1). In the past, expensive operations, such as color histogram analysis, have been performed on *every* frame to achieve scene change detection [12]. Histogram-based procedures require over 10^5 operations per video frame and at least an additional 10^3 operations for comparing the histograms, unlike our procedure which requires many orders of magnitude fewer operations per pair of frames. Once a representative frame has been selected, subregions of the frame may be chosen by further analysis of the DCT coefficients to yield regions of interest which are defined a priori. Frame selection and region selection steps result in

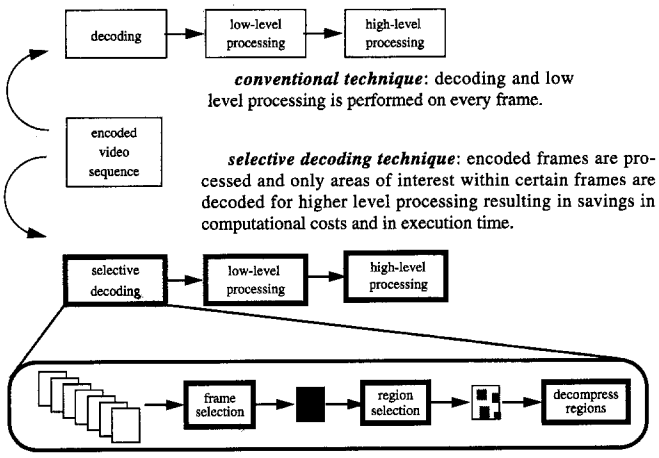


Fig. 1. Conventional technique of using encoded video sequences compared to selective decoding technique

a significant reduction of pixels that need to be processed in subsequent steps, translating into faster processing speeds.

The paper is organized as follows: Sect. 2 briefly surveys previous approaches related to feature management in digital video; Sect. 3 presents the selective decoding approach in-depth. Section 4 outlines our approach to browsing video sequences, and Sect. 5 presents the concluding remarks.

2 Previous approaches

Past research work related to low-level video management has concentrated on the parsing of video sequences into video shots.[1] In most cases, the logical parsing point is a change in the camera viewpoint or a change in the scene. Usually, the histogram of each scene is generated and a large change in the histogram from one scene to the next is used as a cutting point [19] (see also [13]). Ueda et al. [20] suggest the use of the *rate* of change of the histogram instead of the absolute change to increase the reliability of the shot parsing mechanism. Ueda et al. also consider zooming and the panning of the camera; each video frame is divided into a number of non-overlapping small regions, and in each region the optical flow of pixels belonging to that region is approximated and classified into zooming and panning of the camera. This information is then stored along with each shot.

Nagasaka and Tanaka [12] studied various measures to detect scene changes. The best measure, according to their studies, is a normalized χ^2 test to compare the distance between two histograms. Additionally, to minimize the effects of camera flashes and certain other noises, the frames are each divided into several subframes. Then, rather than comparing pairs of frames, every pair of subframes between the two frames is compared, the largest differences discarded, and the final decision is based upon the differences of the remaining subframes.

Little and Venkatesh [11] have proposed a method that detects scene changes based on the number of bytes that are used in encoding each frame. This approach works only with JPEG sequences (the coder for fixed bit rate MPEG sequences, for example, will maintain a constant number of bits per second). Our experiments show that the success rate of this approach is low and must be combined with other approaches for an effective system.

As mentioned in the introduction, DCT coefficients are used in our system to perform scene change detection and other low-level video processing. The use of DCT coefficients prior to decompression has been attempted in other applications. Hsu et al. [7] use DCT compressed images in a military target classification system to discriminate between man-made and natural objects. The Bhattacharyya distance discriminator [5] is used to measure and rank numerous statistical calculations derived from the DCT coefficients, which is in turn used in the decision-making process.

More recently, Smith and Rowe [16] extended many properties of the cosine/Fourier transform and used the DCT coefficients to perform several algebraic operations on a pair of images. Scalar addition, scalar multiplication, pixel-wise addition, and pixel-wise multiplication operations on two images were defined using the DCT coefficients; these operations may be used in video editing systems to perform such tasks as dissolving and subtitling.

3 Low-level video processing

The principal goal of the low-level video processing is to select a frame to serve as the representative for each "shot" of the video sequence. The representative frames are then used in any higher-level processing that may be necessary. Browsing video sequences, for example, may be achieved by viewing the representative frames of a video sequence. Relating video shots from various video sequences based on shape or color content, for example, may also be achieved by using only the representative frames. The computational efficiency of inferring representative frames, or detecting scene changes, is of critical importance since it is performed on each frame of the video sequence (usually on the order of tens of thousands). Therefore, classical image processing measures of similarity, such as template matching, are not suitable in this case.

In the selective decoding approach, scene-change detection is performed using the DCT coefficients in the MPEG- or JPEG-encoded video sequences. Only certain blocks of the encoded frames are monitored over time. Also, using only a few of the DCT coefficients of each selected block, a vector is formed that is used in detecting any changes in the contents of the frames. As mentioned in Sect. 2, most previous attempts to scene change detection use color histograms. Although this procedure is simple, it is a time-consuming step. For example, if each frame is 320×240, each histogram calculation would take roughly 230,000 additions ($256 \times 256 \times 3$ colors) and 76,000 increments (to scan each frame), totalling over 10^5 operations per frame and an additional 10^3 operations for comparing a pair of histograms. Instead, our approach requires several orders of magnitude fewer oprerations per frame comparison.

[1] In this paper, we define a video shot as a subset of a video sequence that begins with a scene change or a video cut and ends with the frame before the next scene change.

3.1 Frame selection in DCT domain

We use the information already encoded in the compression process prior to decompression to take advantage of several facts. First, the computational cost of decompressing every frame would not be necessary and could be saved if only a selected number of frames are chosen for further processing or browsing prior to decompression. Second, coefficients in the frequency domain are mathematically related to the spatial domain, and they may be directly used in detecting changes in the video sequence. Third, the knowledge of the block's location preserves spatial domain information to a certain extent.

The procedure is as follows: for JPEG movie files, entropy and dequantization steps are performed to arrive at the DCT coefficients for each frame. Similarly, for MPEG sequences the frame types are identified, and the entropy and dequantization steps are performed for I frames. The remaining data are discarded in B and P type frames. Then, given the series of 8×8 DCT blocks β_i, $0 < i < \mu$ (where μ is the total number of blocks), in a single DCT-based encoded video frame f, a subset of blocks $\tilde{\beta}_j$, $0 < j < \varrho$, $\varrho \ll \mu$, is chosen a priori. The subset of blocks is chosen such that they correspond to n connected regions in each frame of the video sequence. The regions are the same in both frames under examination. Next, of the 64 coefficients in each block, α are chosen: $\Omega_j = \{c_z, c_y, c_x, \ldots\}$ where c_z is the z^{th} coefficient and the cardinality of Ω_j is α. Again, the members of the set Ω are the same in both frames under examination. The α coefficients may or may not be randomly distributed among the AC terms of the blocks. Using the coefficients from each frame f a vector $\boldsymbol{V}_f = \{c_1, c_2, c_3, \ldots, c_k\}$ is formed, where $k = \alpha\varrho$. Only the blocks in the set $\tilde{\beta}_j$ contribute to $\boldsymbol{V}_f$. This vector then represents frame f of the video sequence in the DCT space. The determination on whether two scenes φ frames apart are dissimilar is achieved in this space. The representations of each two consecutive frames $\boldsymbol{V}_f$ and $\boldsymbol{V}_{f+\varphi}$ are compared using the inner product:

$$\psi(f, f+\varphi) = \frac{\boldsymbol{V}_f \cdot \boldsymbol{V}_{f+\varphi}}{|\boldsymbol{V}_f||\boldsymbol{V}_{f+\varphi}|} .$$

We can assume a change of scene in the video sequence from frame $\boldsymbol{V}_f$ to frame $\boldsymbol{V}_{f+\varphi}$ if $1 - |\psi| > \tau_1$, where $0 \ll \tau_1 < 1$.

The above procedure introduced several parameters; their effects on the performance of the scene detection module are outlined here. First, ϱ determines the number of blocks that are used in the calculations. In general, as ϱ increases more accurate detections may result since larger areas are monitored; however, the higher accuracy is at the expense of more computing time.

Second, n determines how the ϱ blocks are connected in forming the connected regions. It is possible to have $n = \varrho$ resulting in many regions; however, this approach is very sensitive to minor camera motion and the minor movements of the subjects. On the other extreme, n could be set to 1, i.e., the ϱ blocks are selected such that they correspond to one connected region. This approach would not take full advantage of the encoded information since most areas of each frame will not be under observation (unless $\varrho = \mu$ covering the entire frame). A better choice would be to have the connected regions each cover different parts of the frame. Systems using this approach may choose to use any a priori knowledge of the particular domain they may have to place the n regions. For example, process-control monitoring systems in the manufacturing environment may be interested only in observing product flow portions of each video frame.

Third, α is the number of coefficients from each block; if $\alpha = 64$, then the entire frequency spectrum would be used, resulting in high computational expense, but providing an accurate representation of the spatial domain features. More importantly, however, are the choices of coefficients as well as the number of coefficients. Choosing high-frequency components alone assumes such components would exist in the n regions and in all scenes. Therefore, the best approach is to distribute the coefficients among the high- and low-frequency components.

The last factor is φ; it dictates the resolution in the temporal domain. By letting $\varphi = 1$, every frame of the video sequence is processed or, alternatively, higher and higher values of φ could be used to cut computational costs in exchange for lesser resolution in the temporal domain for detecting scene changes. Note that these parameters are the same for the two frames under consideration at any time.

The algorithm applied in our experiments adjusts several of the above parameters dynamically as a function of the input video. Initially, φ is set to a time interval much greater than $1/30^{\text{th}}$ of a second (in our case φ is initialized to 64 frames or about 2 s). Then, if a scene change is not detected it may be assumed that in the past 64 frames no change has occurred; otherwise, φ is divided into two and ψ is re-calculated between the two new, and now closer in time, frames. The operation is continued until either no scene changes are detected or $\varphi = 1$, whichever occurs first. Under ideal conditions, i.e., periods in video sequence with little or no change, such as still graphics, this method allows for rapid analysis of the time period. The worst case scenario, if $\varphi = 2^x$, then at most $2x+1$ comparisons are made – a potential for substantial saving in computational expense. For example, if $\varphi = 64$ ($x = 6$), then rather than making 64 comparisons, at most 13 are made. In addition, the block count parameter, ϱ, and the number of coefficients per block, α, could increase as φ decreases to allow for an even more accurate representation of the frames. In our system $\varrho = 100$ and $\alpha = 6$ at $\varphi > 1$, when $\varphi = 1$ we set $\varrho = \mu$ (all the blocks in the frame are considered) and $\alpha = 15$ to minimize the effects of motion in the scene. Also, in order to minimize the effects of τ_1 on the outcome of the scene change detection, the threshold is initially set low to over-estimate the number of shots. Then, each video cut is examined more closely for the scene changes where $1 - |\psi(f_i, f_j)| < \tau_2$, and $0 < \tau_1 < \tau_2 < 1$. In other words, if $1 - |\psi(f_i, f_j)| < \tau_1$, then there are no changes between frame f_i and f_j, if $1 - |\psi(f_i, f_j)| > \tau_2$ there is definitely a scene change between frame f_i and f_j, and finally if $\tau_1 < 1 - |\psi(f_i, f_j)| < \tau_2$, frames f_i and f_j are examined by decompressing the two frames and performing color histogram analysis (Sect. 3.2, see also [1]).

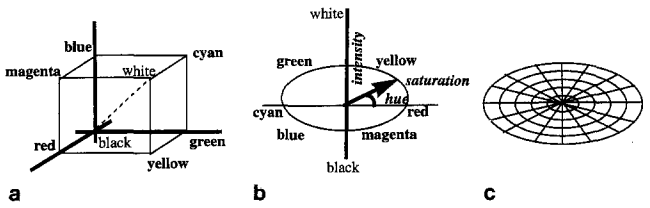


Fig. 2a–c. **a** The RGB color space, **b** the HSI color, **c** color categorization: each bin is one category

3.2 Frame-selection refinement using color histograms

The goal of color histogram analysis is to detect a scene change using the color contents of the given frames. The input to this module is a very small subset of the actual video sequence chosen using the above DCT-based procedure. Noting that similar frames have similar color contents, the color contents of each frame may be measured using the color histogram and compared.

Prior to deriving the color histogram for each frame, one must choose an appropriate color space, which is used to describe the color in each frame. The color space used in the system is the hue, saturation, and intensity (HSI), where only the hue and saturation information are used (see Fig. 2). The transformation from the input RGB color space into the HSI color space is "similar" to the cartesian-to-cylindrical coordinate transformation [17]; however, the transformation is non-linear and ill-conditioned [8]. For example, black (R = G = B = 0) must be handled as a special case. Nevertheless, the HSI color space has less of the artifacts that exist in the RGB color system [9]. Furthermore, past studies have shown that the hue of object surfaces remains the same under various lighting conditions [4], and it is possible to quantize the possible number of colors in a video frame into a smaller, finite number, a process referred to as *categorization* [4]. In this process, the entire spectrum of computer recordable colors is quantized into 720 bins – 5° resolution in hue and 10 levels of quantization of saturation (see Fig. 2). Fathima's studies [4] show that about 200 different color categories are sufficient to describe the color space. Using the HSI color space and the categorization process, a color histogram, which may be assumed to be a surface in the 3D space, is desired for each frame of the video sequence. To detect scene changes, or changes in the color histograms from one frame to another, the two given color histograms are subtracted, forming a color-difference histogram. Assuming color histogram h is defined as

$$h_{f_i}(\mathrm{H},\mathrm{S}) = \sum_{\mathrm{frame}_i} \mathrm{pixels}\ (\mathrm{H},\mathrm{S})$$

the difference histogram is then calculated as

$$\delta_{f_i,f_j}(\mathrm{H},\mathrm{S}) = h_{f_i}(\mathrm{H},\mathrm{S}) - h_{f_j}(\mathrm{H},\mathrm{S})$$

Similar to color histogram, h, the difference histogram is also a surface in the 3D space. To characterize the changes evident in the difference histogram, the volume under the absolute value of the difference histogram surface is calculated as follows:

$$\Lambda_{f_i,f_j}(\mathrm{H},\mathrm{S}) = \sum |\delta_{f_i,f_j}(\mathrm{H},\mathrm{S})| \cdot \Delta H \cdot \Delta S$$

where ΔH and ΔS are the resolutions of the hue and saturation values during the categorization process. Then, once $1 - |\psi(f_i, f_j)|$ is found to be greater than τ_1 but less than τ_2, frames f_i and f_j are decompressed and δ_{f_i,f_j} is calculated. Only if Λ_{f_i,f_j} is greater than τ_3, is the transition between frames f_i and f_j flagged as a scene change; otherwise, no scene changes are detected.

3.3 Experimental results

The above procedure has been fully implemented and tested on JPEG- and MPEG-encoded video sequences. MPEG sequences are treated differently in that the B and P frames are parsed off and not used; only the I frames are analyzed using the above procedure.

The video sequence types used in the experiments varied in subject and in settings. They included documentaries on wildlife (some underwater), graphics, animation, and news broadcasts, which contained an anchor person, outdoor and indoor scenes. Various transitions, such as simple cuts, wipes, fades, and animated graphics, were included as well.

Figure 3 exhibits the results of calculating $1 - |\psi|$ for a few minutes of a JPEG encoded video of a news type broadcast (see also [2]). In Fig. 3a, the time step, φ, is set to 64. With relatively few calculations, several of the time periods in which no transitions occur are identified – most notably from frame number 1250 to 1750. As mentioned earlier, φ is divided by 2 when $1 - |\psi| > \tau_1$ (set to 0.55), and ψ is re-calculated. Figure 3b shows the results after several iterations, $\varphi = 8$, where many of the transitions have been localized. Figure 3c shows the results when $\varphi = 1$ where almost all of the cuts have been identified. However, as mentioned earlier, the system only accepts video cuts when $1 - |\psi| > \tau_2$ and rejects the cuts where $1 - |\psi| < \tau_1$. When $\varphi = 1$, τ_1 is set to 0.35 and τ_2 is set to 0.65; video cuts that fall in-between the two thresholds are decoded for a color histogram analysis (Fig. 3d), as described in Sect. 3.2. Video cuts detected from $1 - |\psi| > \tau_2$ measure are combined with video cuts detected for frames $\Lambda > \tau_3$ – set to 10^5 (see Fig. 3e and [1]). In the last step, video shots, which last less than 1 s (or 15 frames in the example of Fig. 3), are merged.

In the example of Fig. 3, 32 scene changes actually exist – counting the graphics transition from frames 160 to 310 as *one* (Fig. 4). Thirty-one scene changes are detected, as shown in Fig. 3e. Of the 31, 28 are correct, 3 came from the graphics transition of frames 160 to 310, and 1 actual scene change was missed because the assumption that each shot is 2 s or longer ($\varphi = 64$) was violated.

Detecting complex transitions or special effects graphics remains a problem. The main reason is the fact that detection is assumed to occur from one frame to the next, and this assumption is violated. For this reason, the time window for

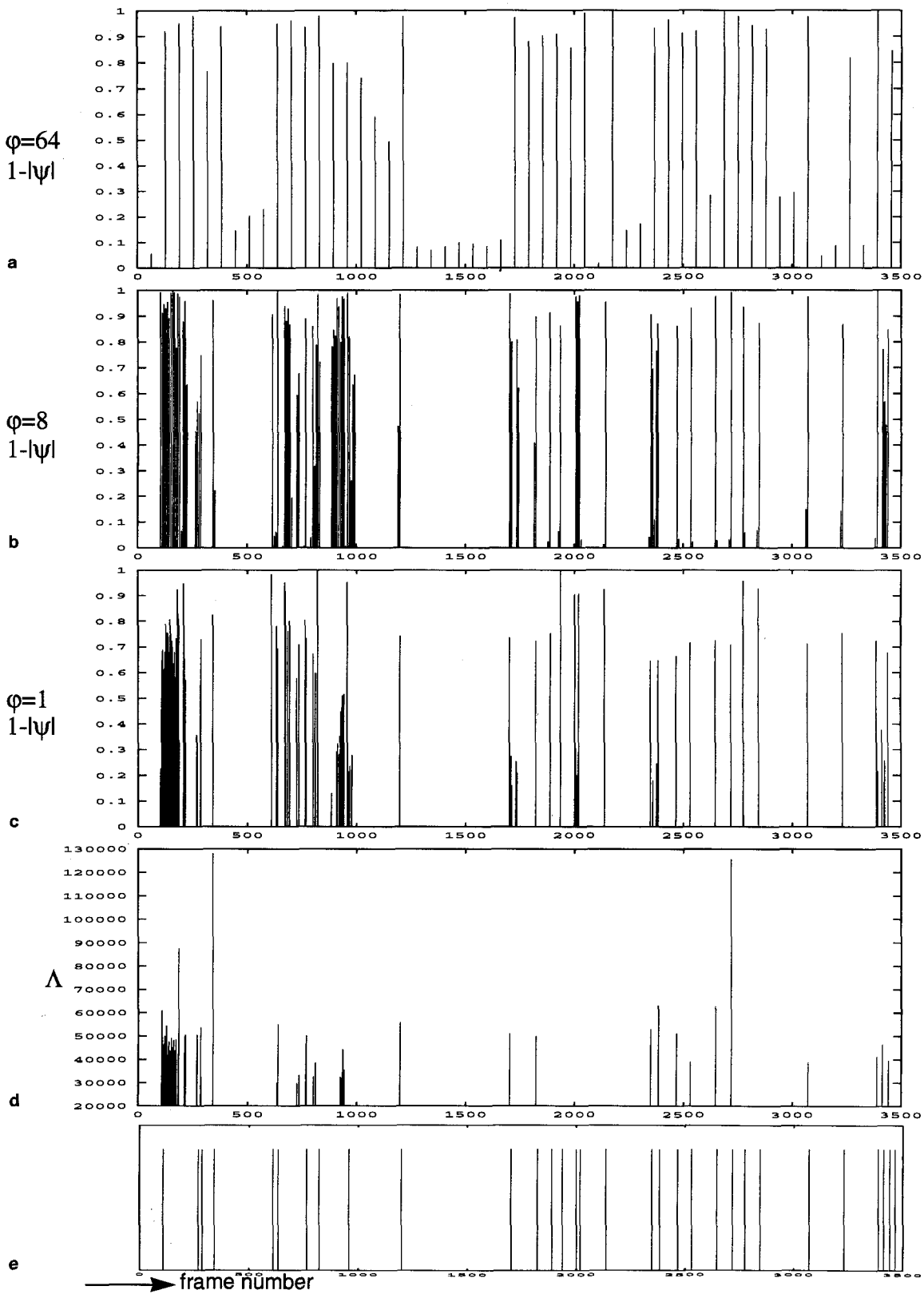


Fig. 3. a–c The plot of $1 - |\psi|$ at three of the seven temporal resolutions in increasing resolution from top-to-bottom. **d** Use of color histogram on a subset of frames. **e** Final video cuts after merging **c** and **d**

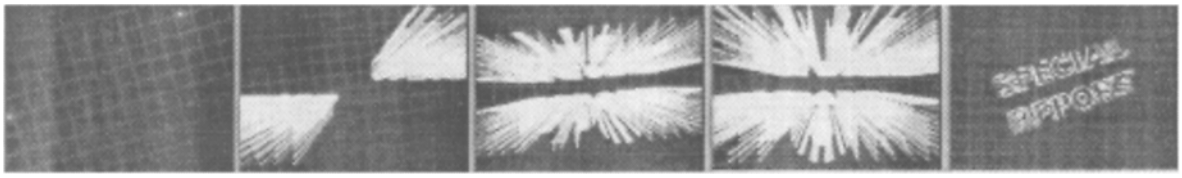


Fig. 4. Five frames of the animation sequence of example of Fig. 3 (frames 160, 165, 170, 175 and 180). Frame 180 reads "Special Report"

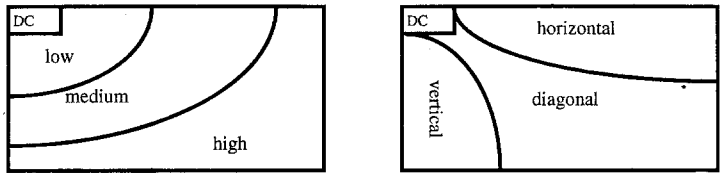


Fig. 5. Frequency distribution (**a**) and block features (**b**) of DCT coefficients [14] within a block

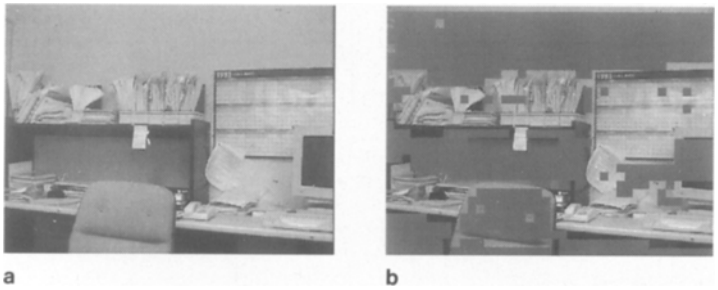


Fig. 6. Example of selecting subregions containing edges using the DCT coefficients. **a** The original frame. **b** The subregions found to contain no edges are shown in *solid*; the remaining regions may be decompressed for edge detection

detecting complex scene changes must be expanded to include numerous frames and the rate of change must be considered. In this case, the scene-change detection algorithm must return a time span rather than an instance. Modules using the scene change detection may then use this information as needed.

3.4 Region selection in the DCT domain

Other applications of selective decoding may be found in tasks such as "matching" video sequences and extracting relevant "structural" information that correlate strongly with the semantics of the video content. DCT coefficients are used to select subregions of the frames for decompression and subsequent image processing and image analysis steps. This is in contrast to conventional approaches where, only after many computationally expensive image processing steps, a subset of the resulting features is chosen for further processing [18]. In such cases, the computational bottleneck has already been performed and choosing a subset of features, although necessary, is no longer a time-saving step.

In choosing subregions from each representative frame, we examine the coefficients of each 8×8 DCT block of the frame. By noting the correlation of frequency distribution in each block and the corresponding spatial features (see Fig. 5a), we may simply choose to process further only the blocks that meet certain criteria dictated by the higher-level processes. For example, to detect edges, only the medium- and high-frequency components are needed [15]; therefore, only the blocks containing coefficients in that range are considered for decompression. In addition, we may use the feature distribution patterns of DCT coefficients to choose subregions of a representative video frame (see Fig. 5b). One specific goal in our system is to detect straight edges. Therefore, blocks containing high and medium frequencies are selected, and only the set of blocks that correspond to a "large" region in the spatial domain is selected for decompression and subsequent edge detection (see Fig. 6).

The computational savings that result from this simple step are manifold. First, only a percentage of the pixels is decoded, resulting in less decompression time. Second, the edge detection algorithm need not be applied to the entire image, resulting in additional savings in time.

Third, since a smaller image area has been analyzed for edge detection, all subsequent steps, such as detecting straight edges or detecting long edges, can be completed more efficiently.

The example of Fig. 6 presents a 320×240 frame. Of the 1200 blocks that form the frame, 735 were determined not to contain any edges. This determination was made by simply counting the number of coefficients in each block that belonged to the high frequency region of the frequency distribution (Fig. 5). The blocks with only a few high-frequency components were not decompressed. In this example, the pixel area was cut into more than half; in other examples, an even higher percentage of regions could be eliminated for further processing.

Fig. 7. Example of a simple video browser. Each icon is the representative frame for a video shot. Users may scroll to see all the icons and may choose one to view the corresponding video shot

4 Browsing video sequences

One possible use of detected scene changes is in browsing video sequences (Fig. 7). In this case, an icon formed from a frame early in the shot is used to represent that shot.[2] By scanning all the icons, one can rapidly find the particular segment of the video sequence for detailed viewing.

The effectiveness of this approach, as compared to the traditional fast-forward and rewind, must be examined. The measure is the time it takes to locate a particular shot. First, we consider the case when the video sequence is viewed by fast-forward and rewind. Assuming the length of the video sequence (L) is 2 h ($L = 216\,000$ frames) and that the sequence may be viewed at an average of 20 times the normal 30 frames/s speed ($\lambda = 20 \times 30 = 600$ frames/s), then the time to view the entire sequence is 360 s or 6 min (L/λ). Further, if we assume that this operation is performed numerous times, on average the desired shot is found half way through viewing the sequence; therefore, on average the time to find a particular scene is $\bar{\omega} = L/2\lambda$, or in our example about $\bar{\omega} = 3$ min.[3] In addition, in most cases some additional time may be required to get to the exact frame on the sequence; we represent that time by the constant κ. Then, the *average* required time to locate the beginning of a desired shot by using fast-forward and rewind is:

$$\bar{\omega} = \frac{1}{2}\left(\frac{L}{\lambda}\right) + \kappa$$

Notice that this computation remains the same regardless of the medium on which the video is stored, i.e., video tape, laser disc, or digitally on hard disk.

On the other hand, locating a shot using the simple browser presented above is much more efficient in time. Assuming that there are δ scene changes per second (for example $\delta = 0.15$, or one every 7 s), then the total number of scene changes per video sequence is $\delta(L/30)$ or in our example a total of 1080 scene changes, requiring 1080 icons to represent the entire video sequence. In the simple browser presented above all the icons are presented in one row that the user may then scroll through until reaching the desired shot. Assuming that ν icons can be shown across the display screen (in our browser 12 icons are presented at one time[4]), the number of screenfuls that are necessary to view all the icons is $\delta L/30\nu$. Then, the *average* time required to locate a particular shot using the simple browser is:

$$\bar{\omega}' = \frac{1}{2}\left(\frac{a}{\nu}\right)\left(\frac{1}{\delta}\right)\left(\frac{L}{30}\right) + \kappa'$$

where a is the number of seconds required to view the screenful (set to one second), and κ' is the time it takes to load the icons into memory. Given the sample figures for our experiments, $\bar{\omega}'$ is equal to 45 s, which is four times faster than the traditional approach.

5 Conclusions

The paper presented a novel approach to scene-change detection using DCT coefficients. Although the DCT coefficients are accessible only after entropy decoding and dequantization steps, the proposed approach avoids the IDCT process, which is by far the most computationally expensive step, requiring numerous multiplications and summations per block. The selective-decoding approach takes advantage of the information contained in the DCT coefficients of encoded video sequences, such as MPEG, JPEG, or H.261, and avoids processing every frame of the video sequence. The system has

[2] Choosing the proper frame is important in some shots, such as the animated sequence of Fig. 4; the last frame is the most appropriate.

[3] This calculation further assumes that the probability of occurrence of the desired scene is equally distributed within the sequence.

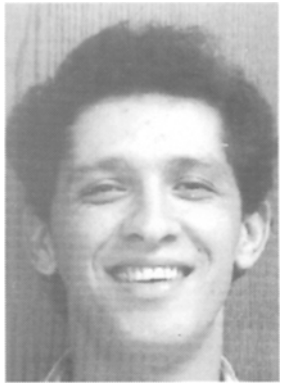
[4] Although more icons can be shown, the browser presented here aims to preserve the temporal continuity of the shots by displaying the icons in one row. Other approaches are certainly possible.

been tested successfully on video sequences of various subjects, including meetings, presentations, personal interviews, documentaries, and others. Furthermore, the DCT coefficients are used to detect only necessary regions of interest for subsequent image-processing steps. This results in a significant reduction of pixels that need to be processed, translating into more efficient processing of video sequences.

Acknowledgements. Scene-change detection on MPEG sequences incorporates the MPEG decoder from The University of California at Berkeley. The user interface of the video browser was implemented by R. Depommier. Siemens Stromberg-Carlson provided the video that was used in the video browser example. The authors would also like to thank D. Benson and P. Baca for their comments when we were preparing this manuscript.

References

1. Arman F, Hsu A, Chiu M-Y (1993) Feature management for large video databases. In: Niblack W (ed) Storage and retrieval for image and video databases, SPIE, San Jose, pp 2–12, SPIE 1908
2. Arman F, Hsu A, Chiu M-Y (1993) Image processing on compressed data for large video databases. In: Proceeding of first ACM International Conference on Multimedia. SPIE, Anaheim, pp 267–272
3. Chang S-K, Hsu A (1992) Image information systems: where do we go from here? IEEE Trans Knowl Data Eng 4:431–442
4. Fathima ST (1992) Data and model-driven selection using color regions. In: Proceedings of the Image Understanding Workshop. Morgan Kaufmann, San Diego, CA, pp 705–716
5. Fu KS (1968) Sequential methods in pattern recognition and machine learning. Academic Press, New York
6. Le Gall D, (1991) MPEG: a video compression standard for multimedia applications. Commun ACM 34:46–58
7. Hsu YS, Prum S, Kagel JH, Anrews HC (1983) Pattern recognition experiments in mandala/cosine domain. IEEE Trans Patt Anal Mach Intell 5:512–520
8. Kender JR (1977) Instabilities in color transformations. In: Proceedings of the Conference on Pattern Reognition and Image Processing. IEEE, RPI, Troy, NY
9. Ledley RS, Baus M, Golab TJ (1980) Fundamentals of true color image processing. In: Proceedings of the International Conference on Pattern Reognition. IEEE, Miami Beach, FL, pp 791–795
10. Liou M (1991) Overview of the $p \times 64$ kbits/s video coding standard. Commun ACM 34:59–63
11. Little TDC, Venkatesh D (1994) Video scene decomposition with the motion picture parser. SPIE Conf. on Digital Video Compression and Processing on PCs: Algorithms and Technologies. SPIE, San Jose, CA (to appear)
12. Nagasaka A, Tanaka Y (1991) Automatic video indexing and full-video search for object appearences. In: Knuth E, Wegner LM (eds) Proceeding of the IFIP TC2/WG2.6 Second Working Conference on Visual Database Systems. North-Holland, Amsterdam, pp 113–127
13. Otsuji K, Tonomura Y (1993) Projection detecting filter for video cut detection. In: Proceedings of First ACM International Conference on Multimedia. ACM Press, Anaheim, pp 251–257
14. Rao KR, Yip P (1990) Discrete cosine transform – algorithms, advantages, applications. Academic Press, New York
15. Rosenfeld A, Kak AC (1987) Digital picture processing. Academic Press, Orlando
16. Smith BC, Rowe A (1993) Algorithms for manipulating compressed images. IEEE Comput Graphics Applic 13, 34–42
17. Strickland RN, Kim C-S, McDonnell WF (1986) Luminance, hue, and saturation processing of digital color images. In: SPIE Conference on Applications of Digital Image Processing, vol 697. SPIE, San Diego, CA, pp 286–292
18. Suetens P, Fua P, Hanson AJ (1992) Computational strategies for object recognition. ACM Comput Surv 24:5–61
19. Tonomura Y, Abe S (1990) Content oriented visual interface using video icons for visual database systems. J Vis Lang Comput 1:183–198
20. Ueda H, Miyatake T, Yoshizawa S (1991) Impact: an interactive natural-motion-picture dedicated multimedia authoring system. In: Robertson SP, Olson GM, Olson JS (eds) Proceedings of Human Factors in Computing Systems (CHI 91). ACM, New Orleans, LA, pp 343–350
21. Wallace GK (1991) The JPEG still picture compression standard. Commun ACM 34:30–44



FARSHID ARMAN received a B.E. degree from the University of New Mexico, and an M.S and Ph.D. in Electrical Engineering from The University of Texas at Austin. His research in graduate school was on 2D and 3D image understanding and segmentation. He has been a member of the technical staff at Siemens Corporate Research, in Princeton, NJ, since 1992 where his current research focus is on digital video processing and video databases. He is a member of Tau Beta Pi and Eta Kappa Nu.

ARDING HSU received a BS degree in Computer Science from Tankang University, Taiwan, an M.S. degree in Computer Science from the University of Waterloo, and a Ph.D. degree in Computer Science from Rutgers University. In 1983, he joined Siemens Corporate Research, Princeton, NJ, where he is currently manager of the Image and Video Management group in the imaging department. His research interests include image/video information management, knowledge-based systems, semantic data modeling, and multimedia systems.

MING-YEE CHIU received a BS degree in Physics from National Taiwan University and a Ph.D. degree in Optical Science from the University of Arizona. He joined Siemens Corporate Research, Inc., Princeton, NJ, in 1980. Currently, he is the head of the Imaging Department at SCR, as well as the manager of Image Processing Core Technology for the Corporate Research and Development Division of Siemens, AG. His research interests include image processing, pattern recognition, digital video, exploratory data analysis, and multimedia technology.