Transactions Letters

Performance Characterization of Video-Shot-Change Detection Methods

Ullas Gargi, Rangachar Kasturi, and Susan H. Strayer

***Abstract*—A number of automated shot-change detection methods for indexing a video sequence to facilitate browsing and retrieval have been proposed in recent years. Many of these methods use color histograms or features computed from block motion or compression parameters to compute frame differences. It is important to evaluate and characterize their performance so as to deliver a single set of algorithms that may be used by other researchers for indexing video databases. We present the results of a performance evaluation and characterization of a number of shot-change detection methods that use color histograms, block motion matching, or MPEG compressed data.**

***Index Terms*—Performance evaluation, shot-change detection, video databases.**

I. Introduction

A NUMBER of initiatives have been undertaken which aim to include digital video content in a digital library accessible over a data network. The interface to this content is meant to allow browsing and searching of the video. Given the temporally linear and data-intensive nature of digital video, coupled with the bandwidth constraints of the network, temporal segmentation of any video sequence stored in the video database has been generally accepted to be a necessary first step in the creation of the interface [1]. Shot-change detection is the process of identifying changes in the scene content of a video sequence so that alternate representations may be derived for the purposes of browsing and retrieval, e.g., keyframes may be extracted from a distinct shot to represent it. An example of a keyframe representation for a 10 000-frame sequence is shown in Fig. 1. Such video storyboards can be obtained by a shot-detection process running on stored or live incoming video using the methods described later in this paper. This representation can be used as a video summary, to browse the content of the whole sequence, and if desired, to quickly pick the shots of interest and view them.

The definition of a shot change is difficult to make. Pronounced object or camera motions may change the content of the view frame drastically. To be consistent, we define a *shot* to be a sequence of frames that was (or appears to be) continuously captured from the same camera. Ideally, a shot can encompass pans, tilts, or zooms; in reality, algorithms for shot-change detection may also react to significant object and camera motion unless global motion-compensation is performed. Identifying shot changes and indexing the video sequence will facilitate the fast browsing and retrieval of subsequences of interest to the user. Shot changes may occur in a variety of ways: *cuts*, where a frame from one shot is followed by a frame from a different shot, or *gradual transitions* such as cross-dissolves, fade-ins, fade-outs, and various graphical editing effects (wipes, pins), which may also be accorded varying semantic significance (e.g., a fade-out to black, followed by a fade-in, is often used by film directors or editors to indicate the passage of time or a change of location). Thus, it is important to be able to detect these events distinctly.

Previous researchers [2]–[5] have employed color histograms to temporally segment color video sequences. The difference between the histograms of consecutive frames is computed and large frame differences are marked as possible shot changes. A number of methods to compare two histograms are possible, prominently: absolute difference between corresponding bins (possibly with a color similarity matrix [6] or without [7]), histogram intersection [8], and chi-square (χ^2) comparison [2]. We investigate the efficacy of these different measures for cut detection and the effect of color space representation on the performance of histogram-based shot detection.

Since stored digital video is likely to be compressed, other researchers have proposed a number of algorithms that perform shot-change detection directly on transform coded MPEG compressed video [9]–[19]. Future versions of the MPEG standard are likely to use higher (object) level "semantic" compression, thus easing the task of indexing the video. Until these standards are implemented, the discrete cosine transform (DCT) MPEG standards are likely to dominate, and thus, performing shot detection on MPEG video acquires importance. We evaluate these algorithms on the basis of their performance on detecting cuts and gradual transitions. We also characterize their performance with respect to robustness to changes in the specific encoder used and the compressed bitrate.

Finally, a number of algorithms have been proposed that perform motion analysis or optical-flow computation to determine shot changes. Three algorithms that perform block motion matching are included in our evaluation. The MPEG techniques also use the block motion prediction information contained in an MPEG bitstream, but the MPEG encoder assigns block motion vectors based on compressive value, not on optical flow consistency.

Although each proposed method described in the literature was tested to varying degrees by the authors, there is a need to

Manuscript received December 12, 1996; revised August 27, 1999. This paper was recommended by Associate Editor S. Panchanathan.

U. Gargi and R. Kasturi are with the Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802 USA.

S. H. Strayer is with Raytheon Systems Company, State College, PA 16802 USA.

Publisher Item Identifier S 1051-8215(00)01627-X.

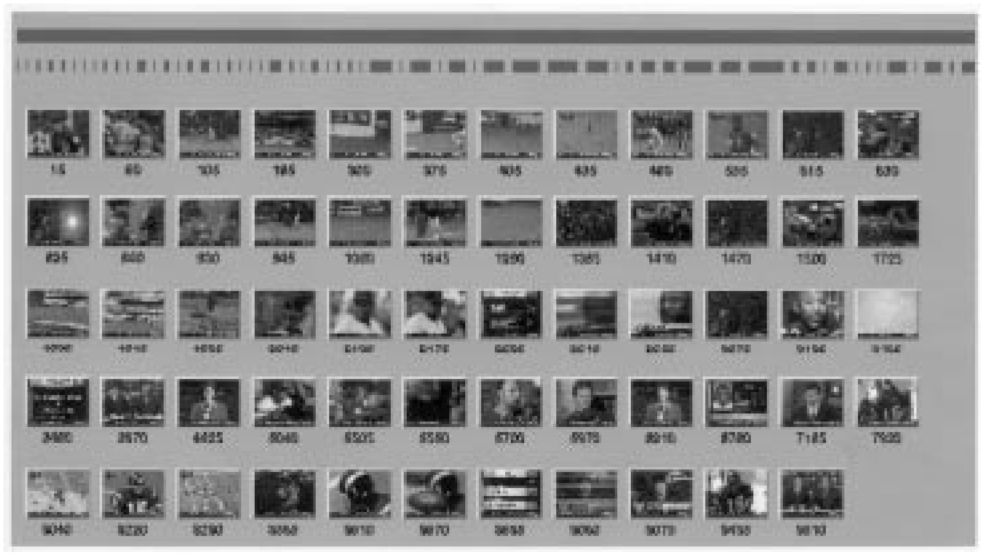


Fig. 1. An example of a keyframe-based browsing and viewing representation for a sports video sequence.

evaluate them and characterize their performance on a common data set. This is the primary motivation for the research described in this paper.

The outline of the paper is as follows. In Section II, we review previous work on performance evaluation of shot-segmentation algorithms. Section III describes our ground-truthed dataset and experimental protocol. A description of the shot-change detection algorithms that were implemented is presented in Section IV. Section V discusses the automatic thresholding methods we employed. Results of our evaluation are presented in Section VI. The results of characterizing selected MPEG algorithms in terms of their sensitivity to data, parameter, and encoder changes are presented in Section VII. Section VIII contains a summary and conclusions.

II. Previous Work

Ahanger *et al.* [20] surveyed the field of video indexing without evaluating or characterizing particular algorithms. Boreczky *et al.* [21] compared five different indexing algorithms: global histograms, region histograms, global histograms using twin-comparison thresholding, motion-compensated pixel difference (similar to [22] which we also evaluate), and DCT-coefficient difference, with respect to shot-change detection. The histograms were grayscale histograms. Their data set consisted of 233 min of motion-JPEG video of various types digitized in 320 $\times$ 240 resolution at 30 frames/s. Some manual tuning was done to find suitable ranges of thresholds to generate the operating curves.

Dailianas *et al.* [23] compared algorithms based on histogram differencing, moment invariants, pixel-value changes, and edge detection. The histogram differencing methods they studied were:

1) *simple*: bin-to-bin histogram differencing without normalization;
2) *weighted*: red, green, and blue histogram differences are assigned different weights;
3) *equalized*: simple differencing after histogram equalization;
4) intersection;
5) chi-square.

The last two methods are described later in this paper. The moment-invariant difference was computed as the L_2 distance between a 3-element vector formed of invariants computed from normalized central moments. Segmentation based on pixel-value changes used the method of Aigrain and Joly [24]. This assigns certain ranges of corresponding pixel changes as being caused by camera cuts (128–255 for 8-bit pixels) or dissolves and fades (7–40). The edge-based algorithm is that of Zabih *et al.* [25], where disappearance of old edge pixels and appearance of new edge pixels far from each other indicate a shot transition.

Their data set consisted of two news broadcasts, a training video and a feature movie. These were digitized from NTSC at 12–15 frames/s in Intel Indeo format, with a total of 2.5 h of video with 1141 cuts.

Our work differs from these previous studies in that for our histogram comparisons we used color histograms in multiple color spaces and investigated the effect of different color space representations. Further, color histograms are often computed as either three one-dimensional (1-D) histograms or as one three-dimensional (3-D) histogram. We evaluated the effect of such representations and explicitly used local thresholding and obtained operating characteristic curves. We observe that the evaluation process itself is parameterized, and explicitly state our parameters. Unlike either study, we evaluated and characterized MPEG algorithms, since compressed-domain methods have become dominant in this field. Both of these studies used human observers to determine the ground truth, and the latter study noted that this creation of ground truth involves ambiguity in the presence of shot changes and that differences between observers occur. We present some experimental results that indicate the conditions under which unbiased human ground truthing is most effective.

III. Database Generation and Experimental Protocol

To evaluate the performance of the shot-change detection algorithms, we captured a database of video sequences and generated the corresponding ground-truth information. We captured video at 30 frames/s at 640 $\times$ 480 resolution and stored in M-JPEG format. The video sequences were ground truthed, with respect to temporal segmentation by human volunteers using a software program that allowed them to view the sequences and mark the frames at which they considered a shot change to have occurred. We have developed a tool which can automatically generate ground-truthed sequences of arbitrary length and complexity by splicing such manually ground-truthed sequences at known shot-change points.

Although for this research all ground-truth data was obtained by experienced humans using frame-by-frame inspection, it is interesting to ask whether it is possible for lay volunteers to provide ground truth simply by watching video. This would allow large amounts of video to be ground truthed relatively cheaply. But there are a number of issues to consider. Creation of the ground-truth data is a subjective process. Different human subjects may mark the same sequence very differently. A subject may mark the same sequence differently on successive viewings. Also, having access to semantic content, humans will tend to mark many more shot changes on a given sequence than will be detected by any algorithm using purely visual information. Our goal was to produce a data set and corresponding ground truth which would be usable for evaluating various algorithms and would be as consistent as possible. In order to establish the optimum experimental conditions under which human volunteers perform their best, a preliminary study of human ground truthing was carried out.

A. Experimental Study of Human Ground-Truthing

The purpose of this study was to determine the best way to allow human subjects to establish a ground truth for temporally segmenting video sequences. The parameters of the experiment and the associated questions to be answered were the following.

1) Playback Speed: Was full frame rate or some slower speed the best at which the subject could be most accurate?

2) Number of Previewings: Many subjects might find it useful to view the sequence one or more times before actually performing the segmentation, especially for very active or fast moving sequences. How many previewings would be best?

3) Manner of Evaluation: Was it feasible to ask people to rank frame differences in order of significance to compare with the ranking that an algorithm might produce?

4) Response Time: What response delay do humans exhibit? This would be useful in mapping the marked ground-truth shot changes to the shot changes found by an algorithm. Our training set sequences (see Section III-B) were shown to each of nine volunteers and they were asked to mark cuts in them. Each volunteer saw each sequence at a number of combinations of playback speed and number of previewings. The playback frame rate was either 1.0 (30 frames/s display rate), 0.5 (half frame rate) or 0.25 (quarter frame rate). The number of previewings was either 0, 1, or 2 (the previewings themselves were at full speed). The interface for the volunteers consisted of a program running on a workstation that played the sequence at the desired speed in a window on the monitor and allowed mouse clicks to record shot changes. The results of this experiment clearly showed that we achieved the best quality and most consistent results when subjects were allowed to mark the scene changes at half speed after viewing the sequence once at full speed. Frame-difference ranking was too subjective and too time consuming. The volunteers produced highly inconsistent results. For half-speed marking, the average delay time was about 10 frames. It ranged from about 7–20 and was fairly consistent for a given subject. This delay would need to be factored into the evaluation process as a parameter to f_e (see Section III-C).

These preliminary results tentatively indicate the conditions under which lay volunteers may be used to generate ground-truth data for video segmentation.

TABLE I
Description of the Sequences in the Dataset

Sequence	Length (s)	No. of cuts	No. of gradual transitions
training	928	172	38
news	1200	180	138
sports news	1200	291	73
sitcom	1200	316	13

B. Dataset

Our database consisted of the sequences listed in Table I. A total of 959 cuts and 262 gradual transitions was present in these approximately 76 min of video. The sequences were converted to MPEG-1 sequences at 320 $\times$ 240 resolution using software encoders. The MPEG encoding process often split a two-frame gradual transition into a single cut; such gradual transitions were labeled as cuts in the ground truth.

The sequences were complex with extensive motion and graphical effects. All the sequences, with the exception of the sitcom, had commercials, some with rapid changes. It is interesting to note that the sitcom had the most cuts, despite having no commercials. This was caused by a large number of back and forth camera angle changes, though the number of distinct scenes was less than in the other types of sequence. There are instances of three cuts in four consecutive frames, cuts occurring in the middle of a gradual transition, long cross-dissolve with object motion, and cuts in only portions of the frame, etc. An example of a complex transition is shown in Fig. 2 The transition shows a 2-frame dissolve followed by a zoom in, computer-generated graphical effects, and a dissolve followed immediately by a 2-frame dissolve. Using a software program for frame accurate video display and event (cut or gradual transition begin or end) recording on a workstation, it took an experienced person roughly 5–6 hours to ground truth 20 min of video. Designing the ground-truthing protocol, building tools for ground truthing, and determining appropriate conditions for ground truthing accounted for a significant fraction of our research efforts over the past few years.

The training sequence contained diverse types of content from news to rock music videos and was used to optimize parameters and for learning the parameters for local thresholding. The rest of the sequences acted as a testing set.

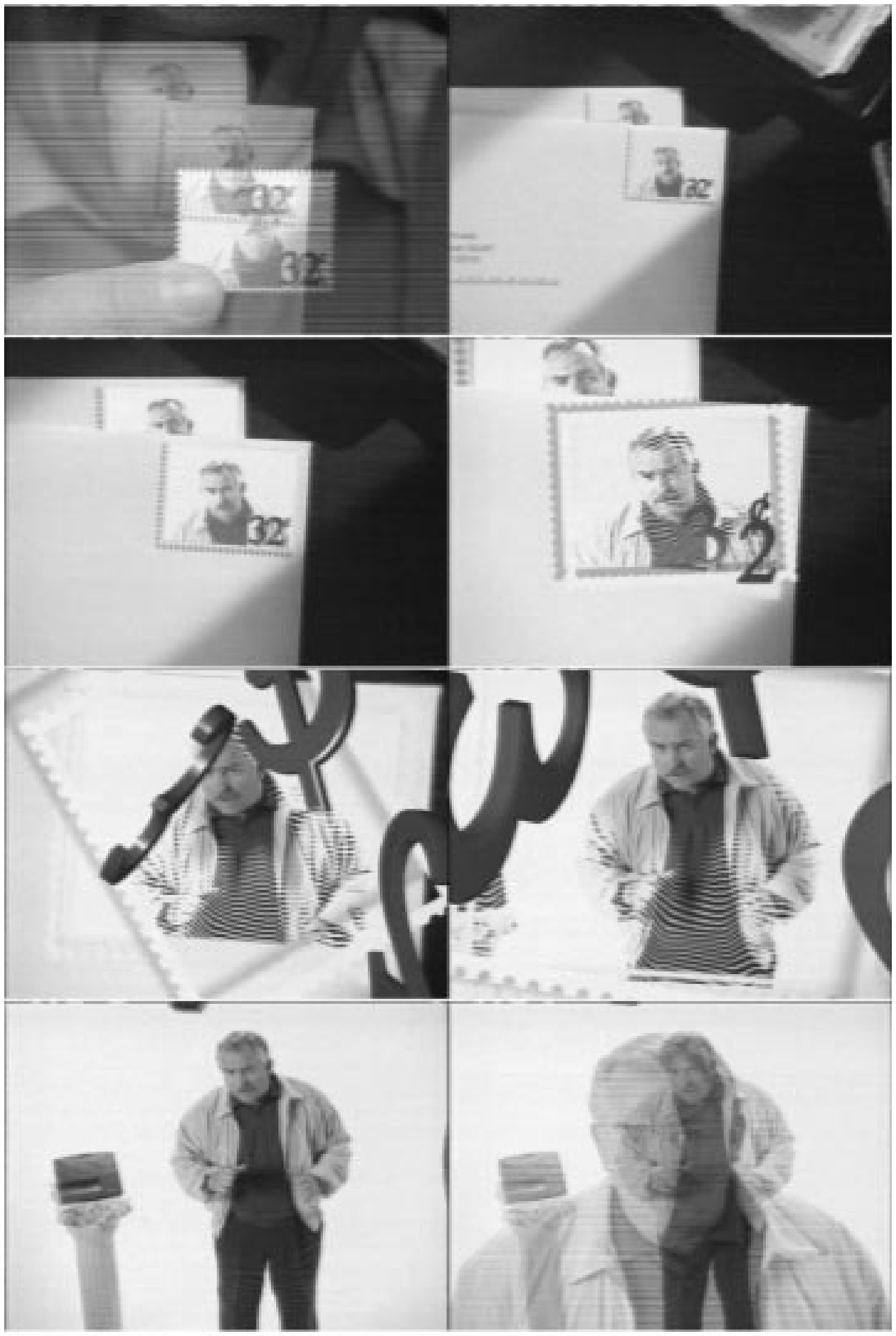


Fig. 2. Sample transition: frames 11635, 11688, 11725, 11761, 11787, 11800, 11823, and 11825 of a sequence.

C. Evaluation Process

The comparison between an algorithm's output and the ground truth is based on the numbers of missed detections (MD's) and false alarms (FA's), expressed as recall and precision

$$\text{Recall} = \frac{\text{Detects}}{\text{Detects} + MD\text{'s}}$$

$$\text{Precision} = \frac{\text{Detects}}{\text{Detects} + FA.\text{'s}}$$

Sethi and Patel [26] suggest that false-alarm errors be ignored entirely. In our view, this is not desirable; under such an evaluation scheme, an algorithm that reported events at every single frame would outperform a more conservative one. Since shot detection is needed for semantic compression of video, too many shot changes will lower the efficiency of the representation. Even with perfect precision, some types of video sequence have shot changes every few seconds; increasing this with false alarms would render the resultant video summary useless for the end user.

The evaluation process is also parameterized, in this case by how far from a ground-truth point we allow an algorithm detection to be and still be counted as a detection. Defining τ to be the set of ground-truth events or features in the data, and η to be the events or features marked by a detection algorithm, the evaluation process can be defined as the process of mapping f_e: $\eta \rightarrow \tau$. This mapping process will also, in general, be parameterized. The parameter for our evaluation consisted of the mapping range R_M, which is the temporal interval within which a ground truth and a detected event are matched. This mapping needs to take place because the detected event frame number will sometimes not match the ground-truth event frame number exactly. This occurs either because of the difference in interpretation of when a video event occurs when multiple choices over a (small) range are possible, or because of the temporal resolution used in processing sequences or because of the subjective placement of a ground-truth event. This is especially true of gradual transitions. In addition, human response time would further affect the location of a ground-truth cut if ground truth were generated by viewing a sequence instead of by frame-by-frame inspection.

IV. Alogorthms Evaluated

A. *Color-Histogram Algorithms*

1) Color Spaces: Histograms were computed in the color spaces listed below. Equations for color space conversions are available in [27].

1) **RGB**: The NTSC red, green, blue space.
2) **HSV**: The hue, saturation, value space.
3) **YIQ**: The NTSC transmission standard for encoding television pictures.
4) **XYZ**: A CIE artificial primary system in which all three tristimulus values are always positive. Y represents the luminance of the color and is the same as the "Y" in the YIQ model.
5) **L*a*b*** (LAB): This is a perceptually uniform color space developed by the C.I.E. to provide a computationally simple measure of color in agreement with the Munsell color system. L* approximately represents the luminance component while a* correlates with redness-greenness and b* correlates with yellow-blueness.
6) **L*u*v*** (LUV): Another uniform color space that evolved from the L*a*b* system and became a CIE standard in 1976.
7) **Munsell** (MTM): The Munsell renotation system also uses a conceptual breakdown of color into hue, value, and chroma components. Hue can be expressed not only as a number, but also as a name. The Munsell Book of Color realizes this space as a set of color chips laid out in rows and columns for different hue values. For a given constant hue, the chips are intended to be perceived with constant brightness in one dimension and constant chroma in another dimension. This conversion was approximated using the method given by Miyahara and Yoshida [28].
8) **Opponent** color axes (OPP): The RGB space may be converted into an opponent color axis space that approximately separates the luminance and chrominance components with simple integer computations. This space was presented in [29] as a useful space for performing video indexing.

As a baseline comparison, we also used the luminance alone (YYY) to verify whether color information was useful in shot-change detection.

2) Frame Difference Measures: Four measures of difference between histograms were studied.

1) Bin-to-bin difference (B2B). Given two histograms h_1 and h_2

$$fd_{b2b}(h_1, h_2) = \frac{1}{2N} \sum_i |h_1[i] - h_2[i]| \tag{1}$$

where N is the number of pixels in a frame.

2) Chi-square test histogram difference (CHI). The histogram bin difference values are normalized to sharpen the frame differences being computed. The authors in [2] justify this to "make the evaluation value more strongly reflect the difference of two frames"

$$fd_{chi} = \frac{1}{N^2} \sum_i \frac{(h_1[i] - h_2[i])^2}{h_2[i]}, \qquad h_2[i] \neq 0 \tag{2}$$

$$= \frac{1}{N^2} \sum_i \frac{(h_1[i] - h_2[i])^2}{h_1[i]}, \quad h_2[i] = 0. \tag{3}$$

The normalization by N^2 is necessary for singular cases as when one histogram consists of exactly one bin and the other histogram has exactly one pixel in the same bin.

3) Histogram intersection (INT). As described in [8], the intersection and corresponding frame difference between two color histograms are

$$\text{Intersection}(h_1, h_2) = \frac{\sum_i \min(h_1[i], h_2[i])}{N} \tag{4}$$

$$fd_{int}(h_1, h_2) = 1 - \text{Intersection}(h_1, h_2). \tag{5}$$

4) Average color (AVG). In [30] it is shown that the difference of average colors of a histogram defined as the 3 $\times$1 vector

$$h_{\text{avg}}[j] = \sum_i h[i]c[j, i] \quad j = 0, 1, 2 \tag{6}$$

where j is the color axis (e.g., R, G, or B) and $c[j, i]$ is the value of color tristimulus j at bin i (for RGB colors, $c[j, i] = i$), is a lower bound on the quadratic form histogram distance using a color-similarity matrix. The frame difference was computed as the square of the average frame-color differences $h^1_{\text{avg}} - h^2_{\text{avg}}$.

3) Dimensionality: Each of the above histogram difference measures (except for AVG which is applied only to 1-D histograms) can be applied to 1- , 2- , or 3-D histograms. That is, the color distribution in a frame may be represented as either three independent 1-D distributions in each of the primaries, or a distribution in two dimensions over the two axes that are not correlated with luminance, or as a distribution over all three dimensions simultaneously. A 2-D representation neglects the luminance component (if one exists) and uses only the chrominance components simultaneously (the RGB color space has no 2-D representation). The idea behind a 2-D histogram is that changes in the brightness of the frame may not be correlated with real content change. For example, flash photography may momentarily increase the brightness of the recording of a news conference. The quantization levels of color space were as follows: for 1-D histograms, 256 bins; for 2-D histograms, 16×16 bins; for 3-D histograms, $8 \times 8 \times 8$ bins. We abbreviate the method and histogram dimensionality into a single code such as B2B1D, INT3D, etc.

B. MPEG Algorithms

1) Brief Introduction to MPEG Video: A very brief overview of the MPEG-1 video format [9] is presented here. There are three types of temporally interleaved frames in an MPEG-1 bitstream: 1) I frames, which are encoded using lossy DCT and lossless entropy coding of the pixel data; 2) P frames, which are motion compensated in the forward direction from I and other P frames (i.e., vectors point from a future frame to a past frame); and 3) B frames, which are motion compensated in both temporal directions from I and P frames. A group-of-pictures (GOP) refers to the frames between two I frames. Motion compensation is carried out at the resolution of macroblocks which are 16×16 pixel blocks, and induces compression by allowing a target macroblock to be represented as a vector pointing to a source macroblock in a reference (I or P) frame and a residual error (which is itself DCT-encoded). A zero prediction vector results if the best matching macroblock in the reference frame that the encoder can find occupies the same position as that of the target macroblock. If the residual error after motion compensation for a particular macroblock in a B or P frame is still too high, the encoder can choose to intracode that macroblock, i.e., to DCT-encode the pixel values directly. A macroblock in a B frame can be forward predicted (the prediction vector points to a macroblock in a past frame), backward predicted (the prediction vector points to a future frame), or bidirectionally predicted (the best matching source macroblocks in the previous and the next reference frame are averaged). DCT coding is carried on units of 8×8 pixel blocks, giving rise to 64 DCT coefficients per block and 4 blocks/macroblock. Pixel values are represented in the YCrCb space and the chrominance components may be spatially subsampled relative to the Y component (the 4 : 2 : 0 format).

2) Description of MPEG Algorithms:

Algorithm MPEG-A: Correlation of DCT-coefficient vectors and prediction vector count statistics. Correlation of DCT-coefficient vectors was originally defined for JPEG images [31] and subsequently modified for the I frames of MPEG sequences [32]. Some *a priori* subset of 8×8 blocks in the frame is chosen. As specified in [31], we chose a small portion of the frame: three connected regions on the sides and in the center of the frame, covering 15% of the frame, were used. Some *a priori* subset of the 64 DCT-coefficients for each block is chosen; we chose the 16 low-frequency components. A vector is formed from these coefficients of each chosen block. The inner product of two consecutive I frames gives the frame similarity.

Additionally, breaks in B frames are detected as follows: if N_{forw} and N_{back} are the numbers of nonzero forward and backward macroblock prediction vectors used in for a particular B frame, then a cut is declared if $\min(N_{\mathrm{forw}}, N_{\mathrm{back}}) < T$, i.e., if the bidirectional frame prediction favors a particular direction. A twin-threshold comparison is used to detect gradual transitions. The total number of parameters needed to specify this algorithm is eight.

Algorthim MPEG-B: Variance and prediction statistics [15]. For cut detection, this approach uses statistics on the numbers and types of prediction vectors used to encode P and B frames. For P frames, if N_{intra} is the number of intracoded macroblocks, and N_{inter} is the number of intercoded macroblocks (those for which motion-prediction vectors are available), then a high value of $N_{\mathrm{intra}}/N_{\mathrm{inter}}$ indicates a shot change. For B frames, the ratio $N_{\mathrm{back}}/N_{\mathrm{forw}}$ vectors is used. I frame cuts are detected by finding peaks in the difference of frame intensity variance where the variance is computed for the discrete cosine (DC) coefficients in I and P frames. An adaptive thresholding method is used for peak detection: the ratio of peak values to average values within a 2–4 GOP size interval is used. Linear dissolves are also recognized by using the difference of frame variances. For dissolves, the ideal variance curve is shown to be parabolic. Peaks in intensity variance separated by a deep trough indicate a dissolve. This algorithm uses I and P frames fully, and prediction vector statistics from B frames. The total number of parameters needed to implement this algorithm is seven. This algorithm is used in the VideoQ system [33].

Algorithm MPEG-C: Motion-prediction statistics [14]. This paper suggests the use of the average error power of macroblock prediction residual error in P-frames. This measure would only detect cuts that occur between a P-frame and its past reference frame. However, the method finally proposed uses the ratios of forward, backward and bidirectional motion prediction vector numbers for B frames. It defines the following frame difference measure:

$$\frac{1}{\min\left(\frac{(N_{\mathrm{forw}} + N_{\mathrm{bidir}})}{N_{\mathrm{total}}}, \frac{(N_{\mathrm{back}} + N_{\mathrm{bidir}})}{N_{\mathrm{total}}}\right)}$$

where N_{bidir} is the number of bidirectionally predicted blocks in a B frame [9] and N_{total} is the total number of macroblocks. A median filter algorithm is used for thresholding. This algorithm detects only cuts, uses only B frames, and requires two parameters.

Algorithm MPEG-D: DC-frame differences [16]. DC coefficient values for I, P, and B frames are extracted (DC coefficient values for P and B frames are reconstructed as in [34]). These values are used to construct a DC-frame sequence, where a DC-frame is a frame consisting of the average intensities of 8×8-blocks of the original frames. Differences between these DC-frames are then computed. Results are presented for two metrics: the sum of the absolute DC-frame pixel-to-pixel differences, and the bin-to-bin difference between histograms of the DC-frame pixel luminances. Automatic thresholding is achieved by a sliding window technique—a peak is declared if it is greater than the second largest difference within the window by some factor. Gradual transitions are detected by looking for a gradually increasing multiframe difference followed by a plateau followed by a decreasing frame difference. This algorithm uses I, P, and B frames. There are 11 distinct parameters that need to be specified for using this algorithm.

Algorithm MPEG-E: Statistical tests on DC coefficients [26]. This computes the histograms of two consecutive DC-frames, applies the χ^2 statistical test to these two distributions to determine the probability that they were drawn from different source distributions and thresholds using a fixed value. An improved version of this algorithm [11] computes row and column histograms, in addition to the overall frame histogram, and uses decision logic to combine the three event decisions into one. These algorithms operate only on I frames. Our implementation allowed the choice of using only I frames, I and P frames, or I, P, and B frames. The latter two were found to give better results, especially in terms of event localization. Four parameters were needed to implement it.

Algorithm MPEG-F: DC-coefficient histogram differencing [19]. Unlike the other methods described here, this method uses the color information present in the bitstream. 1-D histograms of the luminance Y, and chrominance Cb and Cr DC components of blocks are computed and bin-to-bin differencing is applied to these histograms. Both static and locally adaptive thresholds are used for peak finding. Median filtered frame-difference values are used to detect gradual transitions by looking for a series of medium-high difference values, a majority of which need to be above a soft threshold. The algorithm needs seven parameters to be specified.

3) Parameter Optimization for MPEG Methods: The values of parameters needed for implementation of the algorithms were not always specified in their published descriptions, or only a range was specified. Further, in some cases, even those that were specified, resulted in poor performance on our dataset. We also found that we had to add some parameter variables to implement an algorithm. A complete simultaneous optimization with respect to all the parameters of an algorithm was infeasible. We, therefore, used the given values or reasonable values for those parameters that were thought to have a range of equally valid values (e.g., histogram resolution) and for the parameters that the algorithm(s) proved more sensitive to, we chose optimal values by empirical optimization maximizing a figure of merit over the training set. The figure of merit used was the product of recall and precision.

C. Block Matching Methods

Three algorithms which used block matching on uncompressed video data were evaluated. These were the following.

Algorithm Block-A: This algorithm [35] detects shot changes by using a motion smoothness measure. Each chosen frame is divided into 8 $\times$ 8 blocks. Each block is matched to a block in the next chosen frame within a 30 $\times$ 30 pixel neighborhood. The value of the correlation coefficient for the best matching block is computed. The average of these correlations represents an interframe similarity measure.

Algorithm Block-B: This algorithm also uses a form of block matching and motion estimation to detect scene changes [22]. The corresponding motion vector and best correlation value are computed as before. The correlation values are sorted into ascending order. A similarity measure is computed by taking the average of the first s values from the sorted list where s is the number of blocks scaled by a user-specified matching percentage.

Algorithm Block-C: This algorithm does not do motion compensated matching. Instead, as described in [36], the differences between corresponding blocks of the two frames are computed

$$\lambda = \frac{\left[\frac{\sigma_1^2+\sigma_2^2}{2} + \left(\frac{\mu_1-\mu_2}{2}\right)^2\right]^2}{\sigma_1^2\sigma_2^2} \tag{7}$$

where μ denotes the mean gray value and σ denotes the square root of the variance from the frames for the sample areas. The likelihood ratio frame difference is the normalized sum of the individual block differences. A block size of 40 $\times$ 40 was used to capture large-scale frame content changes.

V. THRESHOLDING

The MPEG algorithms had associated thresholding methods described by the authors. For the histogram and block matching methods, an automatic thresholding framework was required. The first question to ask was whether a global threshold would work. If the cut and noncut frame difference values were clearly separated for all video sequences, then a single global threshold would work. Or this threshold could be set differently depending on the type of video (commercials, sitcom, news, sports, etc.). The distribution of cut and noncut frame difference values for algorithms RGB INT1D and LAB B2B2D over the sitcom sequence are shown in Fig. 3. For both cases, the two distributions overlap such that no single threshold can be found that will separate these two distributions. Because the two distributions are normalized by the total number of points and noncuts outnumber cuts by a large factor, the threshold would need to be set well into the noncut distribution tail before reasonable values of precision are obtained. For example, for the RGB INT1D case, a threshold of 0.1 gives a recall of 88% and a precision of 50%. This performance may be sufficient for some applications but further improvement is desirable. The distribution overlap was true for all the algorithms. Therefore, a global threshold might not work well even for video of a single homogeneous type and in fact a global thresholding method based on k-means clustering did not perform as well as local thresholding. Dailianas *et al.* use a different approach of a single global threshold in concert with a simple filtering algorithm to suppress any difference values that were not local maxima [23]. The authors suggest that local thresholding might be an alternative.

We therefore investigate local thresholding to see if it can do better. There are many different ways of choosing a locally adaptive threshold. We would like to be able to evaluate different frame-differencing methods on their ability to separate cuts and noncuts locally, independent of the thresholding method used. We use the following local separability metric

$$M_{\text{sep}} = \frac{\overline{C}fd - N\overline{C}fd}{\sigma} \tag{8}$$

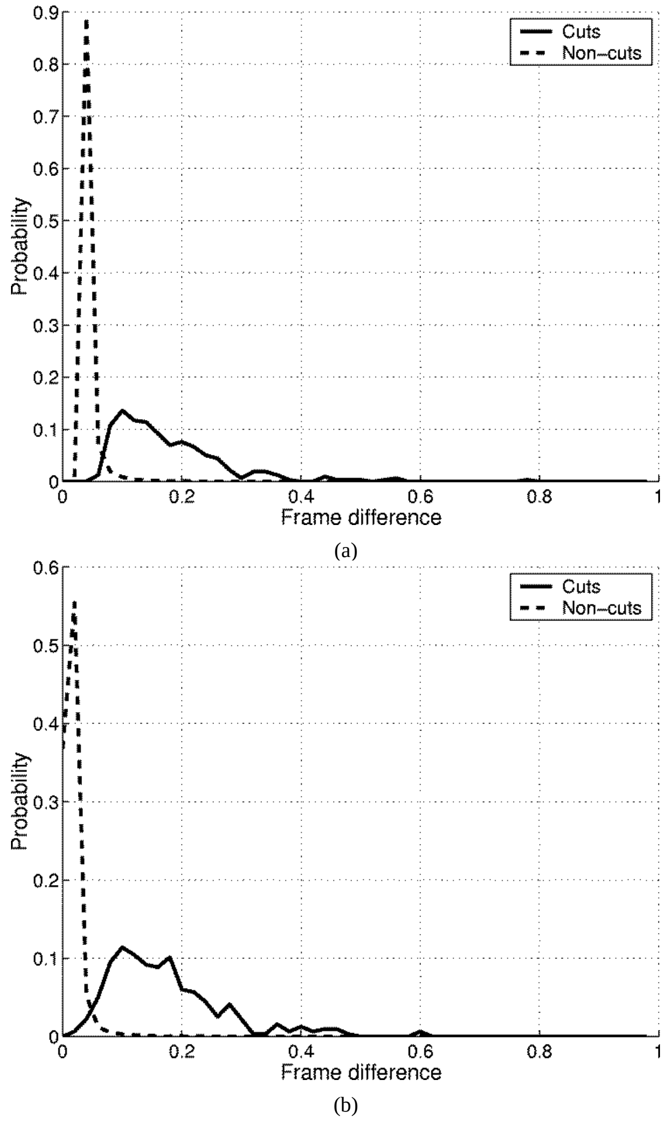


Fig. 3. Cut and noncut frame difference distributions for (a) RGB INT1D and (b) LAB B2B2D for sitcom sequence.

where $N\overline{C}fd$ is the average noncut frame, $\overline{C}fd$ is the average cut frame difference, and σ is the mean standard deviation of both cut and noncut frame differences within a local window. This is computed at every cut frame and the mean over all cut frames represents the local separability of the frame differences for that algorithm. The window size corresponds to a local temporal neighborhood for video. We empirically choose a small window of 12 frames so that we can compare a point with very similar points and not let large-temporal-scale camera motion-affect cut detection. Computing $M_{\rm sep}$ over the testing set, we find that it varies between 2.21 and 2.23 quite uniformly for all color spaces and methods, except for the YYY space and the CHI and AVG methods, which had lower values from 1.99–2.16. Block-A, Block-C, and Block-B had values of 2.07, 1.94, and 1.97. These lower values indicate that they do not separate cuts from noncuts as well as the color-histogram methods.

We found window average thresholding to be effective and robust. The threshold at any frame is formed as a multiple of the local window average and a constant factor r.

VI. Performance Comparison

The histogram methods and Block-C were run on every third frame; Block-A and Block-B were run on every fifth frame, due to their much greater computational cost. The MPEG algorithms operated on either every frame or on only frames of a certain type (I, P, or B). Any optimization or training was done on portions of the training set, while results are presented for the whole data set. Some of the algorithms we implemented detected only cuts, while some detected gradual transitions as well. We present results for these video events separately.

A. Cut Detection

The evaluation parameter R_M was set to three frames for this evaluation. Fig. 4 plots performance curves for the LAB-based color-histogram methods. Curves for the other color spaces are very similar. The figure shows that the color-histogram-based methods did very well on cut detection with desirable performance curves trading off recall against precision. A good operating point on this curve would be a 90%–95% recall with 70%–80% precision.

Table II presents the performances of the color-histogram algorithms represented as the precision at recall = 95% (one point on the performance curves). From these results and the curves, we can observe the following.

Among the color-histogram-based methods, the histogram intersection method is the best. The CHI method did significantly worse than the others. It thus does not appear to be suitable for coarse histogram differencing for shot-change detection. The average color of a frame was also insufficient to capture shot information, being completely inadequate in this application. The 2-D methods did worse than the combined 1-D or 3-D methods, indicating that luminance is an important feature in shot separation. There does not appear to be much difference between using 3-D histograms and three 1-D histograms. The latter are attractive because of their lower memory and computational cost.

While the choice of color space has less of an impact than the choice of histogram differencing method, the MTM method was clearly the best. It was also the computationally most intensive, involving significantly more floating point computation than the other color conversions. The CIE uniform color spaces (LAB and LUV) also did well. To optimize both performance and computational cost, the LAB space is the best compromise. The OPP space performs close to that of the aforementioned three and has the advantage of needing only integer computations. All color spaces were better than the luminance alone (YYY), indicating that the color content of the video frame does characterize the frame for shot-change detection.

Table III presents the performances of the MPEG algorithms. The MPEG algorithms had a number of different parameters which could be tuned to trade off their recall against their precision. We present performance curves on the two best algorithms for the training set. Fig. 5 plots the operating characteristic curve of algorithms MPEG-D and MPEG-F computed on sequences encoded by the SGI encoder at 4.15 Mb/s. The tuning parameter in each case was a thresholding ratio parameter.

From the results of Table III, we see that among the MPEG algorithms, MPEG-D is clearly the best for cut detection,

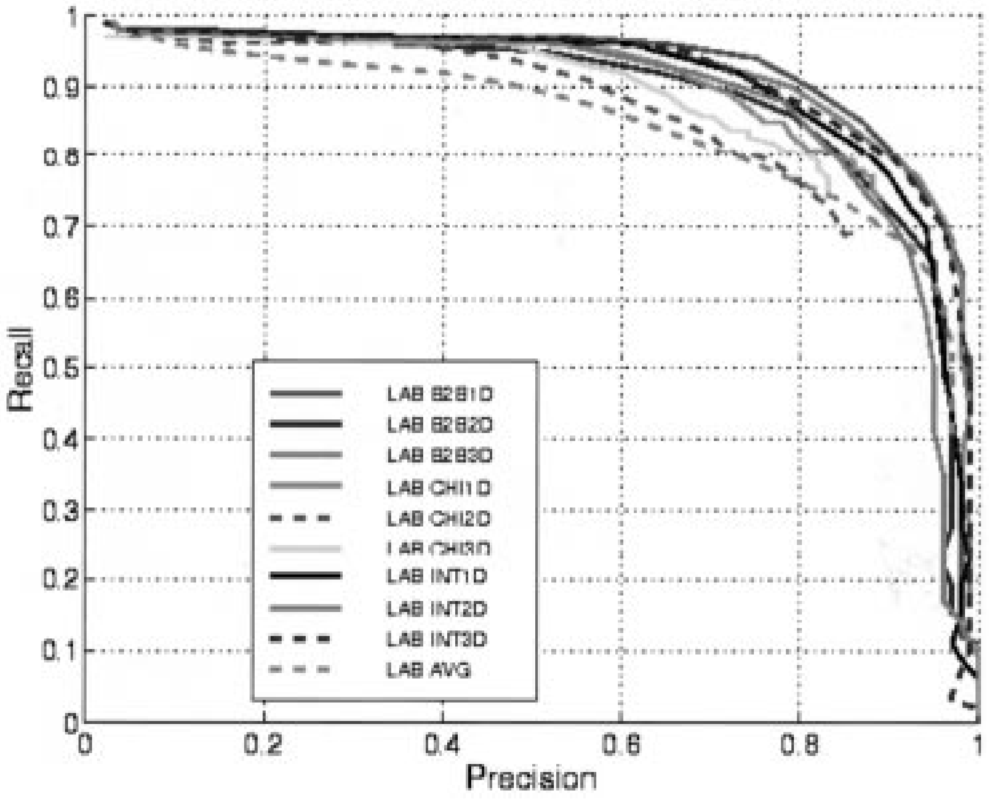


Fig. 4. Operating curves for LAB color-histogram algorithms as the local threshold is varied.

TABLE II
CUT DETECTION PERFORMANCE OF HISTOGRAM DIFFERENCING METHODS: RECALL AT PRECISION = 95%

Color Space	Differencing Method									
	B2B1D	B2B2D	B2B3D	CHI1D	CHI2D	CHI3D	INT1D	INT2D	INT3D	AVG
RGB	60 %		68 %	45 %		45 %	60 %		65 %	10 %
HSV	65 %	63 %	68 %	63 %	53 %	54 %	61 %	61 %	63 %	15 %
YIQ	68 %	37 %	62 %	62 %	23 %	49 %	64 %	50 %	60 %	14 %
LAB	70 %	51 %	65 %	59 %	42 %	50 %	64 %	57 %	63 %	15 %
LUV	68 %	37 %	64 %	59 %	29 %	47 %	67 %	49 %	67 %	14 %
MTM	69 %	65 %	68 %	59 %	54 %	55 %	67 %	63 %	70 %	13 %
XYZ	60 %	65 %	64 %	47 %	57 %	35 %	68 %	68 %	57 %	8 %
OPP	65 %	34 %	57 %	60 %	34 %	45 %	64 %	50 %	67 %	14 %
YYY	55 %			46 %			45 %			6 %

TABLE III
CUT DETECTION PERFORMANCE OF MPEG ALGORITHMS ON TESTING SET

Method	Detects	MDs	FAs	Recall	Precision
MPEG-A	932	27	14820	97 %	6 %
MPEG-B	473	486	3059	49 %	13 %
MPEG-C	286	673	795	30 %	26 %
MPEG-D	754	205	105	79 %	88 %
MPEG-E	862	97	4904	90 %	15 %
MPEG-F	792	167	663	83 %	54 %

with both high recall and precision. The other algorithms have markedly lower precision. Algorithm MPEG-A achieves high recall at the cost of markedly lower precision. Method MPEG-F also does well.

Comparing the color-histogram methods to the MPEG methods, we see that they have comparable precision, but the color-histogram methods have better recall. A 90% recall rate is probably just about sufficient for a video indexing system. The block-matching methods do not do well compared to the other two classes of algorithms. They also had the disadvantage of being computationally intensive. Fig. 6 plots performance curves for the block-motion methods.

B. Gradual Transition Detection

Only the MPEG algorithms were used for detection of gradual transitions. The mapping parameter R_M was set to ten frames for this evaluation. Table IV presents the performances of those algorithms that detected gradual transitions. Algorithms MPEG-D and MPEG-F again have the best performance, though none of them does particularly well. Algorithm MPEG-A could not detect any gradual transitions because it only uses I frames, which in our sequences were 12 frames apart.

The reason for the poor gradual transition-detection performance of all the algorithms is that the algorithms expect some

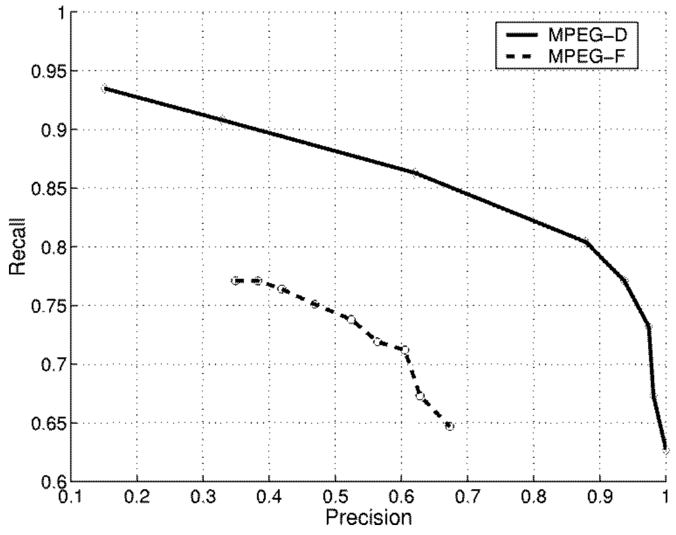


Fig. 5. Operating curve for algorithms MPEG-D and MPEG-F as threshold ratio parameter is varied.

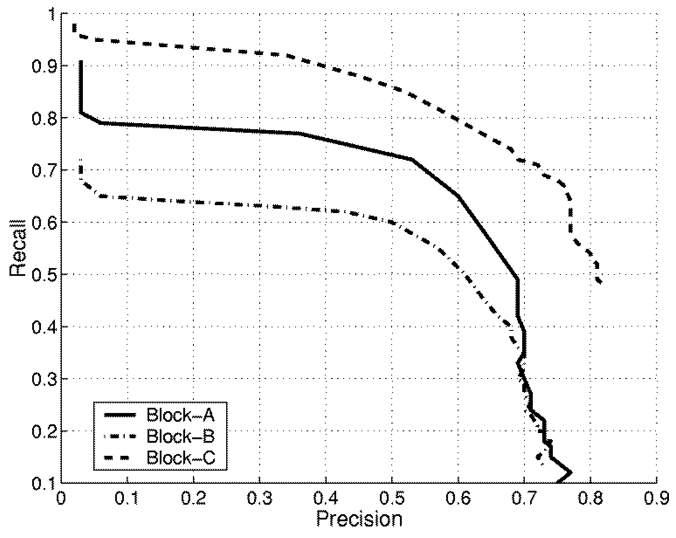


Fig. 6. Operating curves for block-matching algorithms as the local threshold is varied.

TABLE IV
GRADUAL TRANSITION DETECTION PERFORMANCE OF MPEG ALGORITHMS

Method	Detects	MDs	FAs	Recall	Precision
MPEG-A	0	289	21	0 %	-
MPEG-B	75	214	922	26 %	8 %
MPEG-D	83	182	952	31 %	8 %
MPEG-E	32	233	218	12 %	13 %
MPEG-F	18	247	153	7 %	11 %

sort of ideal curve (a plateau or a parabola) for a gradual transition, but the actual frame differences are noisy and do not follow this ideal pattern, or do not follow it smoothly for the entire dissolve. This causes the localization of the transition to be incorrect (beyond the mapping range R_M of our evaluation program), as a single transition is broken into multiple transition detections. Also, the transitions in our dataset often had other effects occurring simultaneously, which caused the actual frame differences to deviate from the theoretical curve for a gradual transition. The measured performance of these algorithms also depends on the parameter R_M of the evaluation mapping f_e: large mapping ranges lead to better measured performance. A mapping range parameter of $R_M = 10$ was used. Smaller values caused performance to drop drastically, illustrating the relatively poor localization of gradual transition beginning and end points. The sequences used had many complex gradual transitions, that varied in length from a few frames to hundreds of frames in length, making accurate detection a difficult task. However, these kinds of transitions are typical of those used in general video.

VII. CHARACTERIZATION OF MPEG ALGORITHMS

Given their performance and the fact that they operate directly on international-standard compressed video with consequent benefits in speed and universality, the MPEG algorithms appear to be the most promising ones for further development. We, therefore, carried out a further characterization presented below.

A. *Full Data Use*

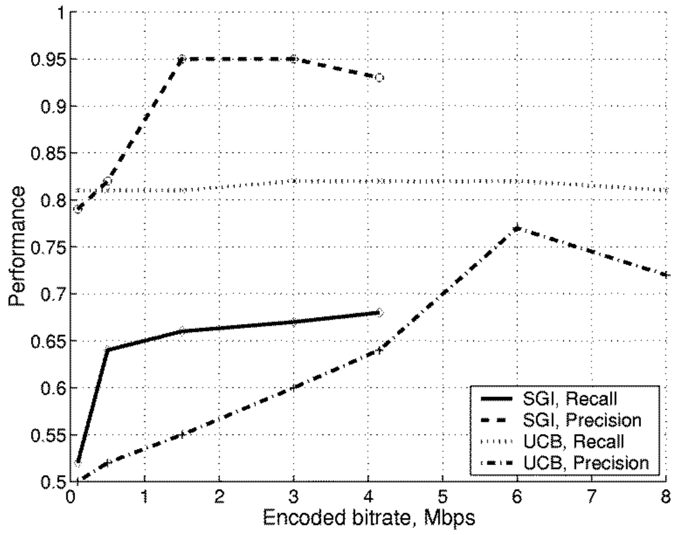
Some of the algorithms (MPEG-A, MPEG-C, MPEG-E) do not process all I, P, and B frames that are present in the input stream. An interesting question is whether this significantly decreases their performance. From Table III, the algorithms that used more data did better. The modification to algorithm MPEG-E to process all frame types improved its performance significantly over the original. Also, from Table IV and as mentioned earlier, algorithm MPEG-A is unable to detect any gradual transitions at all because it uses only I frames. In addition, the algorithms that processed all frames localized the event locations better. Thus, use of all frame types does improve performance significantly.

Also, algorithms used different measures on the different frame types. These different measures had different performances. For example, algorithm MPEG-B's P frame differences were much more reliable than its B frame differences.

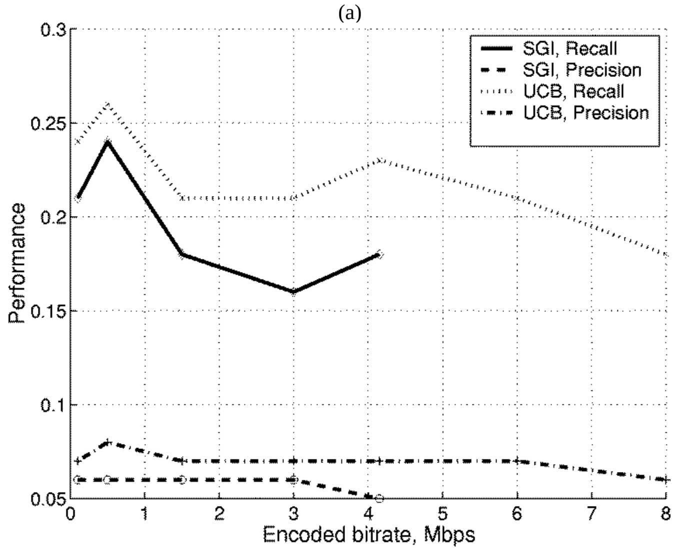
B. *Source Effects*

The performance of a video-indexing algorithm operating on an MPEG stream should, ideally, be independent of the encoder used and the encoding bitrate. We investigated the dependence of the algorithms variations in encoder and bitrate.

We investigated the dependence of the algorithms on two different software encoder implementations. One was the SGI software encoder. The second was the University of California at Berkeley `mpeg_encode` software encoder (UCB). Both used the same original M-JPEG data, and as far as possible, the same encoding parameters: IPB pattern, motion-prediction vector resolution, and search window for motion prediction. The IPB pattern was IBBPBBPBBPBB, the vector resolution was half-pel, the search window was 32 in all directions. The quantization scale factors (IQSCALE, PQSCALE, BQSCALE) varied to achieve the specified bitrate. Fig. 7 shows the variation of cut and gradual transition-detection performance of algorithm MPEG-D with bitrate for the two encoders for bitrates of 100, 500 kb/s, and 1.5, 3, and 4.15 Mb/s on the training set. The UCB encoder also allowed encoding at 6 and 8 Mb/s. As

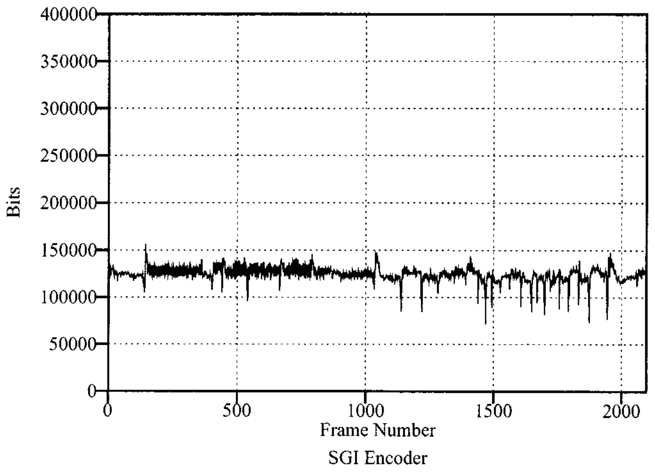


(a)

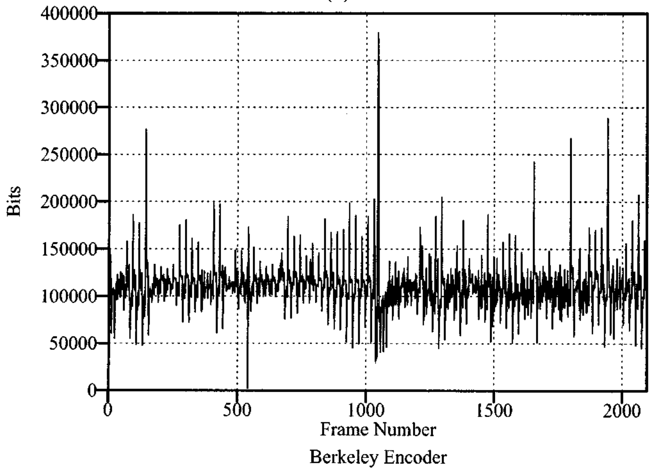


(b)

Fig. 7. Effect of encoder on (a) cut and (b) gradual transition detection performance of algorithm MPEG-D.



(a)



(b)

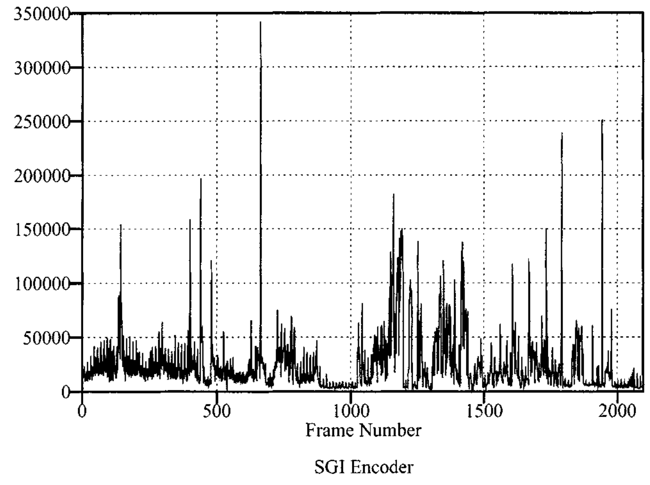
Fig. 8. Sizes of B frames for a sequence encoded by (a) SGI and (b) UCB encoders.

can be seen, there is a significant difference in the performance of the algorithm on the data from the two encoders. Also, this difference is consistent across bitrates and was not attributable to a simple thresholding parameter change. The reason for the difference lies in the differing characteristics of the encoders. For B frames, the Berkeley encoder appears to use intracoding very sparingly and uses forward prediction much more than backward or bidirectional prediction. This imbalance causes the bursty nature of B frame sizes compared to the SGI encoder (Fig. 8). The effect is to delay the coding of frame changes in B frames until the next reference frame, leading to a larger eventual frame difference value (Fig. 9 for algorithm MPEG-D). For the same sequence, the frame difference mean and variance for B frames were 24 446 and 27 925 for the SGI encoder and 33 107 and 32 773 for the UCB encoder. Since B frames constitute 67% of all frames in our sequences (a typical fraction), this has the observed effect on performance. For algorithms that use the statistics of the predicted frames (algorithms MPEG-A, MPEG-B, and MPEG-C), the effect is likely to be even greater. The MPEG standard does not specify an encoding method, only the syntax of the encoded bitstream. Different encoders may use different error measures when performing motion compensation (thus leading to different motion vectors) and also different quantization tables for the DCT coefficient compression. They may also choose to prefer one direction of predictive encoding (forward or backward) over another while still adhering to the standard. Since a number of the algorithms we evaluated use heuristics on these values to detect video events, their performance is thus encoder dependent.

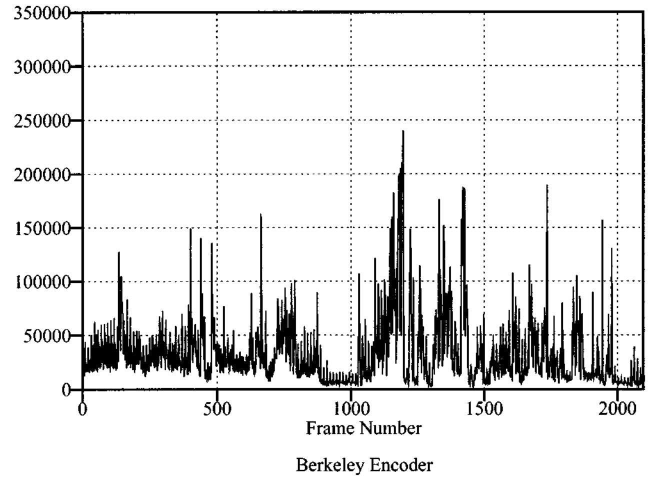
Further results on encoder effects are shown in Table V which shows the recall and precision obtained by the two MPEG algorithms running on the testing set encoded at 4.15 Mb/s. Algorithm MPEG-F appears to be more robust to encoder changes than MPEG-D.

The variation of performance with bitrate using the SGI encoder is shown in Fig. 10 . The algorithms appear to be somewhat affected by the bitrate of the encoding stream, especially at lower bitrates, which may be an important consideration for low bitrate coding applications.

We do not present a quantitative comparison of algorithms on the basis of computational cost because the speed of a algorithm depends on the frame size (which varied between the



(a)



(b)

Fig. 9. Frame difference values at B frames computed by algorithm MPEG-D.

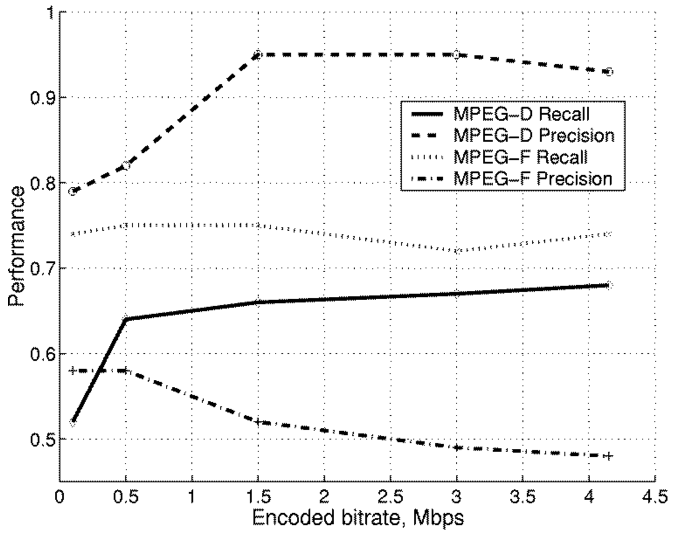
TABLE V
CUT DETECTION PERFORMANCE OF MPEG ALGORITHMS ON TESTING SET USING TWO DIFFERENT ENCODERS

Method	SGI Encoder		UCB Encoder	
	Recall	Precision	Recall	Precision
MPEG-D	79 %	88%	87%	79 %
MPEG-F	83 %	54 %	84 %	52 %

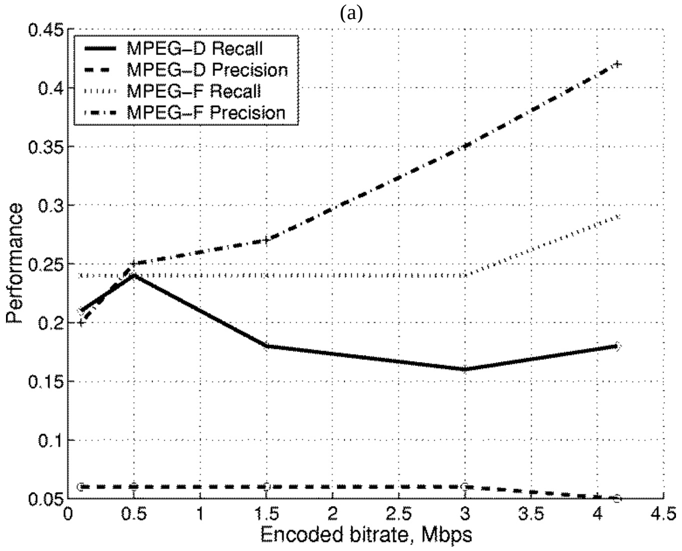
uncompressed and the compressed data) and the data access method. Qualitatively, the block-matching algorithms were the most computationally intensive by a large factor, followed by the color-histogram methods. The color space conversion was the largest factor here, especially those conversions that required heavy floating point computations such as LAB, LUV, and MTM (which was the most intensive). The MPEG methods were the fastest, running at 15 frames/s on a 200-MHz SGI workstation.

VIII. SUMMARY AND CONCLUSION

We have evaluated and characterized the performance of a number of shot-change detection methods using color histograms, MPEG compression parameter information, and image block-motion matching, on a sufficiently varied



(a)



(b)

Fig. 10. (a) Cut and (b) gradual transition performance of algorithms MPEG-D and MPEG-F using SGI encoder with varying bitrate.

ground-truthed test set. Some conclusions to be drawn from our research follow.

Performance evaluation of algorithms requires a common dataset, evaluation criteria, and evaluation process. Ground-truthing video sequences containing complex transitions with respect to video events to single frame accuracy is a challenging task requiring customized software tools. Faster ground-truthing may be obtained at some cost to event location accuracy by using the consensus of multiple humans viewing the video at 15 frames/s with one previewing at 30 frames/s.

Any performance-evaluation process will be parameterized by the method one uses to compare algorithm output to ground truth, and these parameters should be stated explicitly. The number of parameters needed for an algorithm's implementation may be greater than the number of published parameters.

Color-histogram-based shot detection performs sufficiently well to be used in a video database application at a moderate computational cost. Histogram intersection in the Munsell (MTM) space had the best performance. Faster computation at the cost of decreased performance can be obtained by choosing the LAB, LUV, or OPP color spaces. Luminance is an important feature to characterize video shots; 2-D histograms did not work well.

Threshold selection is critical and local window average thresholding works well. Block-motion matching algorithms do not perform as well as color histogram or MPEG-based methods, in addition to being very computationally intensive. MPEG-based shot-detection algorithms are fast, running in real time, but do not perform as well as the color-histogram methods. They are sensitive to changes in the MPEG encoder used and the encoded bitrate. There is a need to make them robust to these variations, as well as increase their recall-precision performance toward the 90%–70% point.

Complex gradual transitions are difficult to detect and accurately locate. The algorithms we tested did not perform satisfactorily on complex transitions. Some researchers have proposed using shot detection to aid in characterizing the semantics of a video (e.g., a feature film). A general-purpose video database system thus needs better performance in this regard.

ACKNOWLEDGMENT

The authors would like to thank S. Antani and D. Crandall for their assistance.

REFERENCES

[1] P. Santos, S. Hudson, M. Guzdial, and A. Badre, "Video temporal compression techniques to facilitate usability evaluation," Graphics, Visualisation and Usability Center, Georgia Tech., Atlanta, GA, Tech. Rep. 95-17, 1995.

[2] A. Nagasaka and Y. Tanaka, "Automatic video indexing and full-video search for object appearances," in *Proc. IFIP 2nd Working Conf. Visual Database Systems*, 1992, pp. 113–127.

[3] C. M. Lee and M. C. Ip, "A robust approach for camera break detection in color video sequence," in *Proc. IAPR Workshop Machine Vision Applications*, Kawasaki, Japan, 1994, pp. 502–505.

[4] J. Wei, M. S. Drew, and Z.-N. Li, "Illumination invariant video segmentation by hierarchical robust thresholding," in *Proc. IS&T/SPIE Conf. Storage and Retrieval for Image and Video Databases VI*, 1997, pp. 188–201.

[5] A. Hampapur, R. Jain, and T. Weymouth, "Production model based digital video segmentation," *J. Multimedia Tools Applicat.*, vol. 1, no. 1, pp. 9–46, 1995.

[6] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin, "The QBIC project: query images by content using color, texture, and shape," *Proc. SPIE*, vol. 1908, pp. 173–181, 1993.

[7] B. M. Mehtre, M. S. Kankanhalli, A. D. Narasimhalu, and G. C. Man, "Color matching for image retrieval," *Pattern Recognit. Lett.*, vol. 16, pp. 325–331, 1994.

[8] M. J. Swain and D. H. Ballard, "Color indexing," *Int. J. Comput. Vis.*, vol. 26, no. 4, pp. 461–470, 1993.

[9] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg, and D. J. LeGall, "MPEG video compression standard," in *Digital Multimedia Standards Series*. London, U.K.: Chapman & Hall, 1997.

[10] N. V. Patel and I. K. Sethi, "Compressed video processing for video segmentation," in *IEEE Proc. Vision, Image, and Signal Processing*, 1996.

[11] ——, "Video shot detection and characterization for video databases," *Pattern Recognit.*, vol. 30, pp. 583–592, 1997.

[12] K. Tse, J. Wei, and S. Panchanathan, "A scene change detection algorithm for MPEG compressed video sequences," in *Proc. Canadian Conf. Electrical and Computer Engineering (CCECE '95)*, vol. 2, 1995, pp. 827–830.

[13] H. C. Liu and G. L. Zick, "Scene decomposition of MPEG compressed video," in *Proc. SPIE/IS&T Symp. Electronic Imaging Science and Technology: Digital Video Compression: Algorithms and Technologies*, vol. 2419, 1995.

[14] H. C. Liu and G. L. Zick, "Automatic determination of scene changes in MPEG compressed video," in *Proc. ISCAS-IEEE Int. Symp. Circuits and Systems*, 1995, pp. 764–767.

[15] J. Meng, Y. Juan, and S. F. Chang, "Scene change detection in a MPEG compressed video sequence," in *Proc. SPIE/IS&T Symp. Electronic Imaging Science and Technology: Digital Video Compression: Algorithms and Technologies*, vol. 2419, 1995.

[16] B.-L. Yeo and B. Liu, "A unified approach to temporal segmentation of motion JPEG and MPEG compressed video," in *Proc. 2nd Int. Conf. Multimedia Computing and Systems*, 1995, pp. 81–83.

[17] B. Yeo and B. Liu, "Rapid scene analysis on compressed video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 5, 1995.

[18] H. J. Zhang, C. Y. Low, and S. W. Smoliar, "Video parsing and browsing using compressed data," *Multimedia Tools and Applicat.*, vol. 1, no. 1, pp. 91–113, 1995.

[19] K. Shen and E. J. Delp, "A fast algorithm for video parsing using MPEG compressed sequences," *IEEE Int. Conf. Image Processing*, pp. 252–255, Oct. 1995.

[20] G. Ahanger and T. D. C. Little, "A survey of technologies for parsing and indexing digital video," *J. Vis. Commun. and Image Rep.*, vol. 7, no. 1, pp. 28–43, 1996.

[21] J. S. Boreczky and L. A. Rowe, "Comparison of video shot boundary detection techniques," in *Proc. IS&T/SPIE Conf. Storage and Retrieval for Image and Video Databases IV*, vol. SPIE 2670, 1996, pp. 170–179.

[22] B. Shahraray, "Scene change detection and content-based sampling of video sequences," in *Proc. SPIE/IS&T Symp. Electronic Imaging Science and Technology: Digital Video Compression, Algorithms and Technologies*, vol. 2419, 1995, pp. 2–13.

[23] A. Dailianas, R. B. Allen, and P. England, "Comparisons of automatic video segmentation algorithms," in *Proc. Integration Issues in Large Commercial Media Delivery Systems*, vol. SPIE 2615, Oct. 1995, pp. 2–16.

[24] P. Aigrain and P. Joly, "The Automatic real--time analysis of film editing and transition effects and its applications," *Comput. Graph.*, vol. 18, no. 1, pp. 93–103, Jan./Feb. 1994.

[25] R. Zabih, J. Miller, and K. Mai, "A feature-based algorithm for detecting and classifying scene breaks," in *Proc. ACM Int. Conf. Multimedia*, 1995, pp. 189–200.

[26] I. K. Sethi and N. Patel, "A statistical approach to scene change detection," in *Proc. IS&T/SPIE Conf. Storage and Retrieval for Image and Video Databases III*, vol. SPIE 2420, 1995, pp. 329–338.

[27] U. Gargi, R. Kasturi, S. Strayer, and S. Antani, "An evaluation of color histogram based methods in video indexing," Dept. Comput. Sci. Eng., Penn State University, University Park, PA, Tech. Rep. CSE-96-053, 1996.

[28] M. Miyahara and Y. Yoshida, "Mathematical transform of (R, G, B) color data to Munsell (H, V, C) color data," in *Proc. SPIE Visual Communications and Image Processing*, vol. 1001, 1988, vol. SPIE 1818, pp. 650–657.

[29] B. Furht, S. W. Smoliar, and H. J. Zhang, *Video and Image Processing in Multimedia Systems*. Norwell, MA: Kluwer, 1995.

[30] J. Hafner, H. S. Sawhney, W. Equitz, M. Flickner, and W. Niblack, "Efficient color histogram indexing for quadratic form distance functions," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 17, 1995.

[31] F. Arman, A. Hsu, and M.-Y. Chiu, "Feature management for large video databases," in *Proc. IS&T/SPIE Conf. Storage and Retrieval for Image and Video Databases I*, vol. SPIE 1908, 1993, pp. 2–12.

[32] H. J. Zhang, "Video parsing using compressed data," in *Proc. SPIE Symp. Electronic Imaging Science and Technology: Image and Video Processing II*, 1994, pp. 142–149.

[33] S.-F. Chang, W. Chen, H. J. Meng, H. Sundaram, and D. Zhong, "A fully automated content-based video search engine supporting spatiotemporal queries," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 602–615, Oct. 1998.

[34] S. F. Chang and D. G. Messerschmitt, "Manipulation and compositing of MC-DCT compressed video," *IEEE J. Select. Areas Commun.*, vol. 13, pp. 1–11, Jan. 1995.

[35] A. Akutsu, "Video indexing using motion vectors," in *Proc. SPIE Visual Communications and Image Processing*, vol. SPIE 1818, 1992, pp. 1522–1530.

[36] R. Kasturi and R. C. Jain, *Computer Vision: Principles*. New York: IEEE Press, 1991, pp. 469–480.

[37] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin, , IBM Tech. Rep. RJ9203 (81511), 1993.