The ViBE Video Database System: An Update and Further Studies

Cuneyt M. Taskiran, Charles A. Bouman, and Edward J. Delp

Video and Image Processing Laboratory (*VIPER*)
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285

ABSTRACT

In this paper we extend the shot transition detection component of the *ViBE* video database system to include gradual scene changes. *ViBE* (Video Indexing and Browsing Environment), a browseable/searchable paradigm for organizing video data containing a large number of sequences, is being developed at Purdue as a testbed to explore ideas and concepts in video databases.

We also present results on the performance of our cut detection algorithm using a large test set. The performance of two other techniques are compared against our method.

Keywords: video database, shot transition detection, regression tree

1. INTRODUCTION

In recent years there has been a great interest in designing and building systems that organize and search video data based on its content.[1] Previously we have described an integrated video database system known as *ViBE* (video indexing and browsing environment) for managing large amounts of video.[2–5] In *ViBE* a variety of algorithms and techniques for processing, representing, and managing video are tightly integrated into a single system which can be scaled to large database sizes and extended to a wide variety of functionalities. The major components of the *ViBE* system are illustrated in Figure 1. The system first segments video sequences into shots by using information obtained from the DC-sequence of the MPEG compressed data stream. Each video shot is then represented by a hierarchical tree structure of key frames, and the shots are automatically classified into predetermined pseudo-semantic classes.[6,2,4] Finally, the results are presented to the user in an active browsing environment using a similarity pyramid data structure. This paper is an update on the shot transition detection used in the *ViBE* system.[7,4]

Although the frame is the smallest physical unit of video, the shot is the smallest possible semantic video unit. Hence the first task in processing video data is segmenting shots by detecting shot boundaries, thereby breaking the video sequence into distinct "semantic chunks." We shall use the following working definition: a *shot* is a group of frames from a video sequence that has continuity in some general conceptual or visual sense. Often, a shot is composed of frames which depict the same physical scene, signify a single camera operation, or contain a distinct event or action. We shall use the terms "shot boundary detection" or "shot transition detection" interchangeably to refer to the task of temporal segmentation of video. Once the shots are identified, they may be clustered to obtain scenes which form the next semantic level for video.

Although a large number of techniques have been proposed to solve the shot boundary detection problem, finding a detection algorithm which would work for common types of shot transitions for a wide variety of video genres with high accuracy is still an open problem. One problem is that the performance of the shot transition algorithm may change a great deal from video genre to genre, and it may be even quite different for programs in the same genre. Therefore, the performance of the detection algorithm must be estimated using as many different sequences from different programs and from different genres as possible. Unfortunately, in the literature the performance of a proposed algorithm is usually measured using a small number of sequences and/or sequences obtained from a small number of different programs in different genres.

Address all correspondence to E. J. Delp, `ace@ecn.purdue.edu`, `http://www.ece.purdue.edu/∼ace` or telephone: +1 765 494 1740.

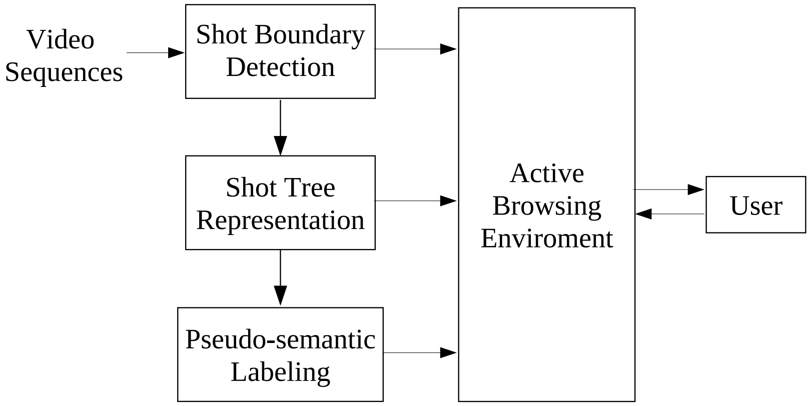

Figure 1. Major components of the *ViBE* system.

In *ViBE*[3] we extract a large number of features from the video sequence and use a binary regression tree to estimate the conditional probability of a shot transition for each frame. Previously we had reported some initial results which showed that this approach was quite effective in detecting cuts in video sequences.[4] In this paper we extend the technique to detect and identify gradual shot tranitions. We also describe the performance of the cut detection algorithm on a large number of video sequences chosen from six different genres containing diverse content.

2. SHOT BOUNDARY DETECTION AND IDENTIFICATION

Although shot transitions may take a wide varity of forms, for our purposes we have divided all transitions into four classes: cuts, dissolves, fades, and all other transitions. We can define these transitions using the following video edit model:

$$f_i = \begin{cases} g_i & i < b_k \\ (1-\alpha_k)g_i + \alpha_k h_i & b_k \le i \le e_k \\ h_i & i > e_k \end{cases} \qquad (1)$$

where f_i, g_i, and h_i are frames in the video sequence, b_k and e_k are the beginning and ending frame numbers of the transition, and α_k is a linearly increasing function of k with $\alpha_{b_k} = 0$ and $\alpha_{e_k} = 1$. For dissolves we have $g_i \in$ previous shot and $h_i \in$ next shot. Cuts are defined similarly with the additional constraint that $e_k = b_k + 1$. For a fade-out we will have $h_i \in \mathcal{N}$ where $\mathcal{N}$ is a shot with all black frames. For a fade-in, we have the reverse true, i.e., $g_i \in \mathcal{N}$.

2.1. Previous Work

A large number of techniques have been reported in literature for the temporal segmentation problem. The general approach to detect shot transitions is to derive various low-level features from frames, derive a dissimilarity value between frames using these, and flag a shot transition whenever this value shows some nontypical behavior. Among the features that have been used are:

- **Pixel-wise frame differences.** Shahraray[8] divides frames into blocks and finds the "best" matching blocks between frames for comparison, similar to the block matching technique of MPEG. Yeo and Liu[9] use the pixel differences of the luminance component of DC frames in an MPEG sequence. Ardizzone and La Cascia[10] process the pixel differences from two frames using a multi-layer perceptron to decide if they belong to the same shot.

- **MPEG motion vector features.** Kobla, Doerman, and Lin[11] perform cut detection using a feature value derived from the numbers of each type of macroblocks in the compressed frames. Meng, Juan, and Chang[12] define various ratios of the number of macroblocks with forward, backward, and no motion compensation to perform cut detection for P and B frames.

- **Edge-based features.** Zabih, Miller, and Mai[13] detect cuts using entering and exiting edge pixel number change fractions for frames. Shen, Li, and Sethi[14] have applied this technique to MPEG sequences using multi-level Hausdorff distance histograms.

- **Color histogram-based features.** Patel and Sethi[15] use intensity histograms derived from DC frames of an MPEG sequence. Ferman and Tekalp[16] use the sum of histogram differences for the Y, U, and V components.
- **Other features** In their *VideoTrail* system Idris, Kobla and Doerman[17] form a feature vector from DC coefficients of each frame and use dimensionality reduction on this vector. Han and Tewfik[18] use a similar technique in uncompressed domain where groups of subsampled pixel values from the frames within a time window is used as the feature vector.

Various strategies are employed in processing these features. In many cases a simple thresholding using a global threshold is used as the significance test. There are also model-based approaches that detect transitions by modeling shot characteristics.[19,20]

2.2. The Generalized Trace

Detecting and identifying shot transitions by processing a single frame dissimilarity feature has a number of problems: First, it is difficult to determine a single feature that could be used to accurately detect shot boundaries in a wide variety of situations. Second, for techniques where a global threshold is used, there is the problem of determining the optimal value of the threshold to be used, since this may vary considerably from sequence to sequence.

In *ViBE* shot boundary detection is performed by first extracting a set of features from each frame in the video sequence which are placed in a feature vector. We call this sequence of feature vectors the *generalized trace* (GT) of the video sequence.[7] The GT is then processed using a binary regression tree to determine the probability that each frame belongs to a shot transition. Finally, these probabilities are post-processed to determine the locations of shot transitions.

Our method has a number of advantages. First, the GT feature vector allows a multitude of different features to be collectively used to detect shot transitions. This is important since different features may be useful in detecting different types of shot transitions. Second, the output of the regression tree is normalized in the range [0-1] and approximates the probability that the frame under question belongs to a shot transition, which allows consistent and meaningful thresholding. Moreover, the method is highly extensible. New features can easily be incorporated into the existing system.

Our method uses the "DC sequence" extracted from the compressed video sequence. The DC sequence is formed from the DC coefficients of the DCT used in MPEG. While the DC coefficients are directly available for I frames, they must be estimated for P and B frames. We have used the method described in[21] for estimating the DC coefficients.

Among the features that were proposed in the literature, we have chosen features that are easy to derive from the MPEG stream and that have good transition detection performance to be included in the GT feature vector. The GT for frame i of the sequence is denoted by $\vec{g}_i$ and its j^{th} component by $g_{i,j}$. For the experiments described in this paper, the GT consists of the following features:

$g_{i,1-3}$ - Histogram intersection[22] of frames i and $i+1$ for the Y, U, and V color components.

$g_{i,4-6}$ - Pixel-wise standard deviations of the Y, U, and V color components for frame i.

$g_{i,7}$ - Number of intracoded macroblocks in frame i.

$g_{i,8}$ - Number of forward predicted macroblocks in frame i.

$g_{i,9}$ - Number of backward predicted macroblocks in frame i.

$g_{i,10-12}$ - Flags which identify the frame type $\{I$, P, or $B\}$ for frame i.

The frame type flags are needed because the information contained in the GT must be interpreted differently for different types of frames in the MPEG sequence. For example, P frames contain no backward predicted macroblocks so $g_{i,9}$ will identically be zero. Similarly, the number of intracoded macroblocks per frame, $g_{i,7}$, should be interpreted differently for I, P, and B frames.

2.3. Binary Regression Tree

An intuitively appealing methodology in decision making is to proceed in multiple stages, making partial decisions along the way, thereby breaking up a complex decision boundary into a union of several simpler boundaries. Trees are one of the possible approaches to multistage decision making which are hierarchical in nature. The tree methodology is known to be very effective in a wide variety of domains.[23,24]

The regression tree used in *ViBE* is a variation of the technique proposed in.[25] The difference is that the training and pruning step is used only once. The training process uses two sequences with known shot transition locations, which we refer to as *ground truth sequences.* One ground truth sequence is used to build a large tree and the other sequence is then used to prune the resulting tree. The growing stage starts with a single terminal node and at each step the terminal node that yields the greatest reduction in the classification error is split. In this way, we produce a tree that overfits the data.[25] The tree is then pruned in a bottom-up fashion using the second ground truth sequence where we remove nodes whose deletion decreases the classification error.

The tree-based approach has a number of advantages when compared to more traditional nonparametric methods such as nearest neighbor or kernel estimator approaches. The regression tree has a simple form which can be compactly stored, and it efficiently classifies data. It also does automatic feature selection and complexity reduction.[25]

Due to their dissimilar nature, we detect cuts and gradual shot transitions separately.

2.3.1. Detection of Cuts

To train the first stage regression tree, we use the known cut locations in the ground truth sequences and ignore gradual transitions. After the tree has been trained using two groundtruth sequences, it is used to process the GT from the sequence whose cuts are to be detected. A schematic diagram of the steps in cut detection is shown in Figure 2. To detect if a cut has occurred between frames i and $i+1$ we place a window of length $2W_1+1$ centered around frame i and all the GT vectors in this window are concatenated into one large feature vector. These agglomerate vectors are then used by the regression tree which provides a piecewise linear approximation to the conditional mean, i.e.,

$$y_i \approx E[\alpha_i | \vec{g}_{i-W_1} \cdots \vec{g}_i \cdots \vec{g}_{i+W_1}] \tag{2}$$

where α_i is the cut indicator function

$$\alpha_i = \begin{cases} 1 & \text{if a cut occurs between frames } i \text{ and } i+1 \\ 0 & \text{if no cut occurs} \end{cases}$$

and where y_i is the output of the regression tree for frame i. The output y_i can be interpreted to be the probability of a cut occurring at frame i.[26] It should be noted that in general the classifier uses more than just $\vec{g}_i$ to determine if a shot boundary has occurred. Our experiments have shown that the use of $\vec{g}_{i-1}$, $\vec{g}_i$ and $\vec{g}_{i+1}$, that is, taking $W_1 = 1$ provides a reasonable balance between complexity and performance.

Candidate cut locations are then determined by thresholding the output of the regression tree; if $y_i \geq \tau$, then we decide that there might be a cut between frames i and $i+1$. The detected candidate cut locations are then post-processed to remove cuts that are too close together. We have used the rule that if two candidate cut locations are closer than 10 frames, the candidate cut with a smaller value of y_i is deleted.

2.3.2. Detection of Gradual Transitions

For the detection of dissolves and fades a windowing approach similar to the one above is used. However, since gradual transitions are distributed over a large number of frames, one would ideally want have a window size that is close to the average length of the gradual transitions. This would require the use of a large value, e.g., 20-50 frames, for the window length. The problem with this approach is that the agglomerate feature vectors defined in Equation 2 derived from such a large window would have high dimensionality. One could try to reduce dimensionality of these vectors using techniques similar to ones used in.[17,18]

We have taken a different approach to solve this problem by adopting a two stage method for the detection of gradual shot transitions. The method is illustrated in Figure 3. In this technique, we first use windowing on the GT using a small value for window length, W_1, to avoid dimensionality problems, the result of which is used by the first regression tree. The output of this regression tree is then windowed with a window length of $2W_2+1$ using

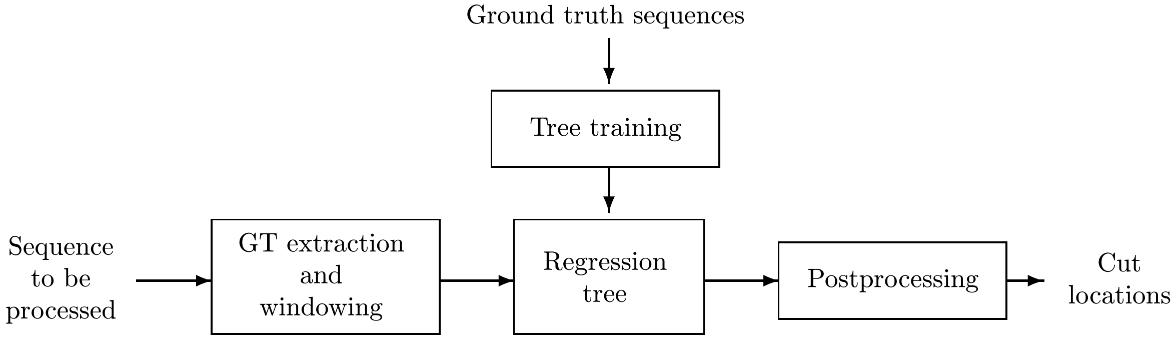

Figure 2. Schematic diagram for cut detection.

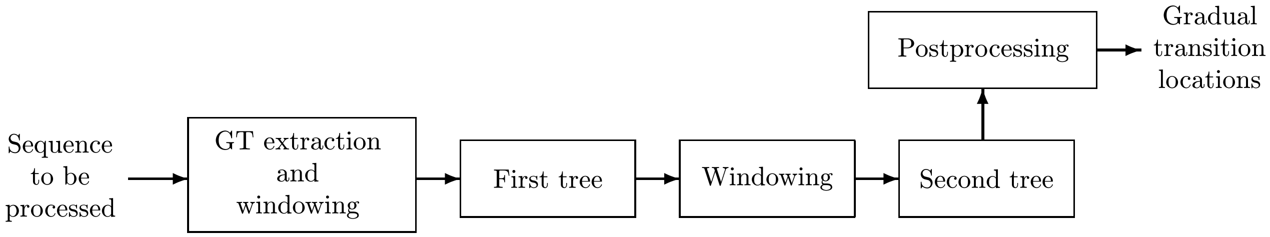

Figure 3. Two stage approach to gradual transition detection.

Equation 3. We choose W_2 such that $W_2 \gg W_1$ is satisfied. These agglomerate vectors are then used by a second regression tree, with output w_i, which approximates

$$w_i \approx E[\beta_i | \vec{y}_{i-W_2} \cdots \vec{y}_i \cdots \vec{y}_{i+W_2}] \tag{3}$$

where β_i is the gradual transition indicator function

$$\beta_i = \begin{cases} 1 & \text{if frame } i \text{ belongs to a gradual transition} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

We then perform two postprocessing operations on the output w_i. The first operation is similar to the one done for cuts and removes gradual transitions that are closer than 30 frames. The second one deletes gradual transitions that shorter than 3 frames.

3. RESULTS

3.1. The Experimental Data Set

At this time *ViBE* contains more than 90 MPEG-1 sequences, each 10 minutes long, which were recorded from miscallenous television programs. The sequences have been digitized at a rate of 1.5 Mb/sec in CIF format (352 x 240).

We have selected 29 sequences from our database which we believe represent a number of different program genres. Commercials, if they exist, were edited out. The locations of all the shot transitions in these sequences were recorded by a human operator. These 29 sequences are classified into six program genres as follows:

- Soap operas (5 sequences). Two episodes from *The Young And The Restless*, one episode from *The Bold And The Beautiful*, *Guiding Light*, and *As The World Turns* each. These consist mostly of dialogue with cut transitions between them and have very little camera motion.

Sequence Class	# frames	# cuts	# dissolves	# fades	# others	avg. shot length
soap opera	67582	337	2	0	0	196 ± 17
talk show	107150	331	108	1	6	239 ± 97
sports	78051	173	45	0	29	299 ± 40
news	58219	297	7	0	6	182 ± 10
movies	54160	262	15	6	1	760 ± 270
cspan	90269	95	19	0	0	206 ± 81
TOTAL	455431	1495	196	7	42	

Table 1. The number of different types of shot boundaries for different sequence classes used. The mean shot length and standard deviation in frames is also given.

- Talk shows (7 sequences). One sequence from *The Rosie O'Donnell Show*, two sequences from *Oprah Winfrey Show*, *Late Night With David Letterman*, and *Regis and Kathie Lee* each. These contain a lot of dissolves and a lot of camera motion in the beginnings of the programs.
- Sports (5 sequences). One college football sequence, two sequences from the NCAA Tournament, and two sequences with car races, one from NASCAR and one from the Texas 500 Car Race. This is the hardest genre to process since it has a lot of special effects shot transitions, rapidly moving objects, and a high degree of camera motion.
- News (4 sequences). Three sequences from WLFI, a local television station, and one CBS News.
- Movies (3 sequences). One sequence from *Before He Wakes* and two from *Lawrence of Arabia*.
- Sequences from CSPAN (5 sequences). Obtained from CSPAN-I and CSPAN-II. Consists of long shots with very little camera or object motion.

Note that, with the exception of the sequences from *Lawrence of Arabia* we never use more than one sequence from a given airing of a particular program, in order to achieve maximum content variation. Statistical information about these classes is given in Table 1.

3.2. Cut Detection Experiments

To get an honest estimate of the performance of our cut detection system, we have used the following procedure which is similar to a cross-validation procedure

```
for each genre $G \in \{soap, talk, sports, news, movies, cspan\}$
    for $i = 1$ to 4
        randomly choose two sequences, $S_1$ and $S_2$, both not in $G$
        train the regression tree using $S_1$ and $S_2$
        process all the sequences in $G$ using this tree
        average the cut detection performance of the tree over the sequences in $G$
    average the four sets of values to get the performance for genre $G$
```

In the results presented here, we have used a window size of $W_1 = 1$ which means that the regression tree uses 36 features. For all results of our method, we have used a threshold of $\tau = 0.35$ as our detection criteria.

We have compared our method with two other methods. The first method uses a global threshold on the sum of the histogram intersections of the Y, U, and V components, i.e, it uses $g_{i,1} + g_{i,2} + g_{i,3}$ as the frame similarity feature. This is the simplest possible method and is included here as a baseline, to serve as an indication of the

	tree classifier				*sliding window*				*simple thresholding*			
Sequence Class	Detect	Miss	FA	MC	Detect	Miss	FA	MC	Detect	Miss	FA	MC
soap opera	0.941	0.059	13.3	0	0.916	0.084	99	0	0.852	0.145	24	0
talk show	0.942	0.058	32.3	7.5	0.950	0.050	45	1	0.968	0.032	171	15
sports	0.939	0.051	82.5	34.8	0.785	0.215	59	1	0.925	0.075	251	73
news	0.958	0.042	38.0	0.75	0.886	0.114	61	0	0.926	0.074	212	1
movies	0.821	0.179	43.3	2	0.856	0.144	25	0	0.816	0.184	25	3
cspan	0.915	0.085	54.3	8.5	0.994	0.006	40	0	0.943	0.057	3	20

Table 2. Results for cut detection using the GT/tree classifier, the sliding window method, and simple thresholding. *Detect* and *Miss* indicate the average detection rate and missed detection, respectively, for each class. *FA* and *MC* are the total number of false alarms and misclassifies, respectively.

relative difficulty in detecting the cuts in various video genres. A global threshold value of 0.45 was found to give best results and this value was used for all of our experiments. Again, we remove the cut with a lower feature value if two cuts are closer than 10 frames.

We have also implemented a sliding window technique, similar to the one proposed by Yeo and Liu,[9] using the same feature used for the global thresholding method described above. In this technique, a symmetric window of size $2m + 1$ is placed around the i^{th} frame and a cut is declared from frame i to $i + 1$ if

1. the value of the feature value for i is the maximum within the window, and
2. it is also n times the value of the second maximum in the window.

In our experiments we have used $m = 7$ and $n = 2$ because these values gave the best overall performance. Note that this windowing places an inherent limitation on the resolution for cut detection, e.g. for this value of m no two cuts can be closer than 21 frames or else one cut will be missed.

The results of these experiments are shown in Table 2. From this table we can make the following observations: The GT/regression tree method gives a very consistent performance for video sequences with diverse content whereas the sliding window method runs into problems with some types of sequences. This is most clear with the *sports* and *news* genres, where the the detection rate of the sliding window detector is low.

Our experiments have shown that the genres of the sequences used to train the regression tree do not affect the performance of the system. The only criterion important in choosing the sequences to be used in the training step is the number of cuts the sequences contain. One should use sequences that contains as many cuts as possible, while not having a lot of gradual scene transitions.

4. CONCLUSION

In this paper we have introduced a new method for shot transition detection in the compressed domain for MPEG video sequences. We have extensively tested the performance of our algorithm on sequences containing diverse program content, such as soap operas and sports programs for cut detection. We have also compared the performance of our algorithm with two other algorithms, one using simple global thresholding, the other using a sliding window approach. We have found that our algorithm gives very consistent performances independent of the type of program being analyzed. The performance of the algorithm is also quite independent of the training sequences.

REFERENCES

1. E. J. Delp, "Video and image databases: Who cares?," *Proceedings of the SPIE/IS&T Conference on Storage and Retrieval for Image and Video Databases VII*, January 23-29 1999, San Jose, CA, pp. 274–277.
2. J.-Y. Chen, C. Taskiran, A. Albiol, E. J. Delp, and C. A. Bouman, "Vibe: A compressed video database structured for active browsing and search," *IEEE Transactions on Image Processing*, submitted to.
3. J.-Y. Chen, C. Taskiran, A. Albiol, C. A. Bouman, and E. J. Delp, "Vibe: A video indexing and browsing environment," *Proceedings of the SPIE Conference on Multimedia Storage and Archiving Systems IV*, vol. 3846, September 1999, Boston, MA, pp. 148–164.
4. C. Taskiran, J.-Y. Chen, A. Albiol, C. A. Bouman, and E. J. Delp, "A compressed video database structured for active browsing and search," *Proceedings of the IEEE International Conference on Image Processing*, October 1998, Chicago, IL.
5. J.-Y. Chen, C. Taskiran, C. A. Bouman, and E. J. Delp, "Vibe: A new paradigm for video database browsing and search," *Proceedings of the 1998 IEEE Workshop on Content-Based Access of Image and Video Databases*, June 21 1998, Santa Barbara, CA.
6. A. Albiol, C. A. Bouman, and E. J. Delp, "Face detection for pseudo-semantic labeling in video databases," *Proceedings of the IEEE International Conference on Image Processing*, October 25-28 1999, Kobe, Japan.
7. C. Taskiran and E. J. Delp, "Video scene change detection using the generalized trace," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 12-15 1998, Seattle, WA.
8. B. Shahraray, "Scene change detection and content-based sampling of video sequences," *Proceedings of SPIE Conference on Digital Video Compression: Algorithms and Technologies*, vol. 2419, February 1995, San Jose, CA, pp. 2–13.
9. B.-L. Yeo and B. Liu, "Rapid scene analysis on compressed video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 6, pp. 533–544, December 1995.
10. E. Ardizzone and M. L. Cascia, "Multifeature image and video content-based storage and retrieval," *Proceedings of the SPIE Conference on Multimedia Storage and Archiving Systems*, vol. 2916, November 18-19 1996, Boston, MA, pp. 265–276.
11. V. Kobla, D. Doerman, and K.-I. Lin, "Archiving, indexing, and retrieval of video in the compressed domain," *Proceedings of SPIE Conference on Multimedia Storage and Archiving Systems*, vol. 2916, November 1996, Boston, MA, pp. 78–89.
12. J. Meng, Y. Juan, and S.-F. Chang, "Scene change detection in a MPEG compressed video sequence," *Proceedings of SPIE Conference on Multimedia Computing and Networking*, vol. 2417, February 1995, San Jose, CA, pp. 180–191.
13. R. Zabih, J. Miller, and K. Mai, "A feature-based algorithm for detecting and classifying scene breaks," *Proceedings of the ACM International Conference on Multimedia*, November 5-9 1995, San Francisco, CA.
14. B. Shen, D. Li, and I. K. Sethi, "Cut detection via compressed domain edge extraction," *Proceedings of the IEEE Workshop on Nonlinear Signal and Image Processing*, September 1997, Mackinac Island, MI.
15. N. V. Patel and I. K. Sethi, "Video shot detection and characterization for video databases," *Pattern Recognition*, vol. 30, no. 4, pp. 583–592, April 1997.
16. A. M. Ferman and A. M. Tekalp, "Multiscale content extraction and representation for video indexing," *Proceedings of SPIE Conference on Multimedia Storage and Archiving Systems II*, November, 3-4 1997, Dallas, TX, pp. 23–31.
17. V. Kobla and D. Doermann, "Videotrails: Representing and visualizing structure in video sequences," *Proceedings of the ACM Multimedia'97: The Fifth ACM International Multimedia Conference*, November 9-13 1997, Seattle, WA, pp. 335–346.
18. K. J. Han and A. Tewfik, "Eigen-image based video segmentation and indexing," *Proceedings of the IEEE International Conference on Image Processing*, October 1997, Santa Barbara, CA, pp. 538–541.
19. P. Aigrain and P. Joly, "The automatic real-time analysis of film editing and transition effects and its applications," *Computation and Graphics*, vol. 18, no. 1, pp. 93–103, 1994.
20. N. Vasconcelos and A. Lippman, "A bayesian video modeling framework for shot segmentation and content characterization," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, October 4-7 1997, San Juan, Puerto Rico.

21. K. Shen and E. J. Delp, "A fast algorithm for video parsing using MPEG compressed sequences," *Proceedings of the IEEE International Conference on Image Processing*, October 26-29 1995, Washington, D.C., pp. 252–255.
22. M. Swain and D. H. Ballard, "Color indexing," *International Journal of Computer Vision*, vol. 7, no. 1, pp. 11–32, 1991.
23. S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Transactions on Systems Man and Cybernetics*, vol. 21, no. 3, pp. 660–674, May 1991.
24. S. K. Murthy, "Automatic construction of decision trees from data: A multi-disciplinary survey," *Data Mining and Knowledge Discovery Journal*, vol. 2, no. 4, 1998.
25. S. Gelfand, C. Ravishankar, and E. Delp, "An iterative growing and pruning algorithm for classification tree design," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 2, pp. 163–174, February 1991.
26. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Belmont, CA: Wadsworth International Group, 1984.