Journal of Electronic Imaging 5(2), 122–128 (April 1996).

Comparison of video shot boundary detection techniques

John S. Boreczky
Lawrence A. Rowe
University of California Berkeley
Computer Science Division, EECS
Berkeley, California 94720
E-mail: johnb@cs.berkeley.edu, rowe@cs.berkeley.edu

Abstract. *Many algorithms have been proposed for detecting video shot boundaries and classifying shot and shot transition types. Few published studies compare available algorithms, and those that do have looked at limited range of test material. This paper presents a comparison of several shot boundary detection and classification techniques and their variations including histograms, discrete cosine transform, motion vector, and block matching methods. The performance and ease of selecting good thresholds for these algorithms are evaluated based on a wide variety of video sequences with a good mix of transition types. Threshold selection requires a trade-off between recall and precision that must be guided by the target application.* © *1996 SPIE and IS&T.*

1 Introduction

The increased availability and usage of on-line digital video has created a need for automated video content analysis techniques. Most research on video content involves automatically detecting the boundaries between camera shots.

A shot is an unbroken sequence of frames from one camera. Thus, a movie sequence that alternated between views of two people would consist of multiple shots. A scene is defined as a collection of one or more adjoining shots that focus on an object or objects of interest. For example, a person walking down a hallway into a room would be one scene, even though different camera angles might be shown. Three camera shots showing three different people walking down a hallway might be one scene if the important object was the hallway and not the people.

There are a number of different types of transitions or boundaries between shots. A cut is an abrupt shot change that occurs in a single frame. A fade is a slow change in brightness usually resulting in or starting with a solid black frame. A dissolve occurs when the images of the first shot get dimmer and the images of the second shot get brighter, with frames within the transition showing one image superimposed on the other. A wipe occurs when pixels from the second shot replace those of the first shot in a regular pattern such as in a line from the left edge of the frames. Of course, many other types of gradual transition are possible.

Paper MMS-01 received March 15, 1996; accepted for publication April 5, 1996.
1017-9909/96/$6.00 © 1996 SPIE and IS&T.

Our research involves detecting scene boundaries in video based on the shot boundaries and an analysis of the audio track. Providing this higher level structure is important because people perceive video as a collection of scenes, not shots. This research requires a good shot boundary detection algorithm that correctly identifies gradual shot transitions.

Unfortunately, it is difficult to compare shot boundary detection methods based on the published literature because few studies report quantitative results and those that do involve limited test sequences. Many researchers working in this area have expressed the need for an unbiased comparison of the available techniques.

This paper describes an experiment designed to compare shot boundary detection algorithms. We digitized and manually indexed a large set of test data and implemented several algorithms similar to those proposed in the literature.

The remainder of this paper is organized as follows. Section 2 describes previous work on shot boundary detection algorithms. Section 3 describes the algorithms that were tested and the video segments used for testing. Section 4 describes the experimental results, and Section 5 concludes the paper.

2 Related Work

There has been a good deal of research on using computers to do automatic content extraction of videos. Early techniques focused on cut detection, and more recent work has focused on detecting gradual transitions. The major techniques that have been used for shot boundary detection are pixel differences, statistical differences, histogram comparisons, edge differences, compression differences, and motion vectors. The only reported comparisons of these techniques[1–3] applied the tested methods to a small number of short test sequences and sometimes tuned the methods to work well on those sequences.

2.1 *Pixel Differences*

The easiest way to detect if two frames are significantly different is to count the number of pixels that change in value more than some threshold. This total is compared

against a second threshold to determine if a shot boundary has been found. This method is sensitive to camera motion. Zhang, Kankanhalli, and Smoliar[1] implemented this method with the additional step of using 3×3 averaging filter before the comparison to reduce camera motion and noise effects. They found that by selecting a threshold tailored to the input sequence good results were obtained, although the method was somewhat slow. We note that manually adjusting the threshold is unlikely to be practical.

Shahraray[4] divided the images into 12 regions and found the best match for each region in a neighborhood around the region in the other image. This matching process duplicates the process used to extract motion vectors from an image pair. The pixel differences for each region were sorted, and the weighted sum of the sorted region differences provided the image difference measure. Gradual transitions were detected by generating a cumulative difference measure from consecutive values of the image differences.

Hampapur, Jain, and Weymouth[5] computed what they call chromatic images by dividing the change in gray level of each pixel between two images by the gray level of that pixel in the second image. During dissolves and fades, this chromatic image assumes a reasonably constant value. They also computed a similar image that detects wipes. Unfortunately, this technique is very sensitive to camera and object motion.

2.2 *Statistical Differences*

Statistical methods expand on the idea of pixel differences by breaking the images into regions and comparing statistical measures of the pixels in those regions. For example, Kasturi and Jain[6] compute a measure based on the mean and standard deviation of the gray levels in regions of the images. This method is reasonably tolerant of noise, but is slow due the complexity of the statistical formulas. It also generates many false positives (i.e., changes not caused by a shot boundary).

2.3 *Histograms*

Histograms are the most common method used to detect shot boundaries. The simplest histogram method computes gray level or color histograms of the two images. If the bin-wise difference between the two histograms is above a threshold, a shot boundary is assumed. Ueda, Miyatake, and Yoshizawa[7] used the color histogram change rate to find shot boundaries.

Nagasaka and Tanaka[2] compared several simple statistics based on gray level and color histograms. They found the best results by breaking the images into 16 regions, using a χ^2 test on color histograms of those regions, and discarding the eight largest differences to reduce the effects of object motion and noise.

Swanberg, Shu, and Jain[8] used gray level histogram differences in regions, weighted by how likely the region was to change in the video sequence. This worked well because their test video (CNN Headline News) had a very regular spatial structure. They did some simple shot categorization by comparing shots with the known types (e.g., anchor person shot) in a database. They were also able to group shots into higher level objects such as scenes and segments by matching the shot types with the known temporal structure.

Zhang, Kankanhalli, and Smoliar[1] compared pixel differences, statistical differences, and several different histogram methods and found that the histogram methods were a good trade-off between accuracy and speed. In order to properly detect gradual transitions such as wipes and dissolves, they used two thresholds. If the histogram difference fell between the thresholds, they tentatively marked it as the beginning of a gradual transition sequence, and succeeding frames were compared against the first frame in the sequence. If the running difference exceeded the larger threshold, the sequence was marked as a gradual transition. To reduce the amount of processing needed, they compared nonadjacent frames and did finer level comparisons if a possible break was detected.

2.4 *Compression Differences*

Little et al.[9] used differences in the size of JPEG compressed frames to detect shot boundaries as a supplement to a manual indexing system. Arman, Hsu, and Chiu[10] found shot boundaries by comparing a small number of connected regions. They used differences in the discrete cosine transform (DCT) coefficients of JPEG compressed frames as their measure of frame similarity, thus avoiding the need to decompress the frames. A further speedup was obtained by sampling the frames temporally and using a form of binary search to find the actual boundary. Potential boundaries were checked using a color histogram difference method.

2.5 *Edge Tracking*

Zabih, Miller, and Mai[3] compared color histograms, chromatic scaling, and their own algorithm based on edge detection. They aligned consecutive frames to reduce the effects of camera motion and compared the number and position of edges in the edge detected images. The percentage of edges that enter and exit between the two frames was computed. Shot boundaries were detected by looking for large edge change percentages. Dissolves and fades were identified by looking at the relative values of the entering and exiting edge percentages. They determined that their method was more accurate at detecting cuts than histograms and much less sensitive to motion than chromatic scaling.

2.6 *Motion Vectors*

Ueda, Miyatake, and Yoshizawa[7] and Zhang, Kankanhalli, and Smoliar[1] used motion vectors determined from block matching to detect whether or not a shot was a zoom or a pan. Shahraray[4] used the motion vectors extracted as part of the region-based pixel difference computation described above to decide if there is a large amount of camera or object motion in a shot. Because shots with camera motion can be incorrectly classified as gradual transitions, detecting zooms and pans increases the accuracy of a shot boundary detection algorithm.

Motion vector information can also be obtained from MPEG compressed video sequences. However, the block matching performed as part of MPEG encoding selects vectors based on compression efficiency and thus often selects inappropriate vectors for image processing purposes.

3 Implementation and Test Data

This section describes the implementation of the algorithms selected for testing and the test data used.

We decided to test a subset of the described algorithms based on ease of implementation, expected performance, and the presence of interesting features. The source code was not available for all of these algorithms, so we created our own implementations in order to provide a consistent test. For each of the algorithms, there were many design details that were unspecified in the literature. Where possible, we tried to be consistent among the algorithms, and let our choices be biased toward simplicity. This means that although the tested algorithms resemble those proposed in various papers, many details are different, and this could have a large effect on performance. All of the algorithms were designed to examine every frame of the test data rather than perform temporal sampling.

We selected the following five algorithms for our test:

1. *Histograms:* One threshold is used. We compute a 64-bin gray-scale histogram over the entire frame. The difference measure is the sum of the absolute bin-wise histogram differences. A shot boundary is declared if the histogram difference between consecutive frames exceeds a threshold.
2. *Region histograms:* Two thresholds are used. Each frame is divided into 16 blocks in a 4×4 pattern. A 64-bin gray-scale histogram is computed for each region. Histogram differences are computed for each region between consecutive frames. If the number of region differences that exceed the difference threshold is greater than the count threshold, a shot boundary is declared.
3. *Running histograms:* This algorithm closely resembles the algorithm described by Zhang, Kankanhalli, and Smoliar.[1] Two thresholds are used. We compute a 64-bin gray-scale histograms over each image. If the histogram difference between consecutive frames exceeds the high threshold, a cut is declared. If the histogram difference exceeds the low threshold we assume that we are starting a gradual shot transition, so we start computing differences from the start of the gradual transition. If this running difference exceeds the high threshold, we will declare a gradual transition once the run ends. If the difference drops below the low threshold for more than two frames, we stop computing running differences and decide that the gradual transition, if there was one, must be over. To reduce false positives due to camera motion or the motion of large objects, we compute a set of motion vectors based on block matching in a 4×3 grid. If the motion vectors for a frame indicate this type of excessive motion, we assume that a pending gradual transition is false, and we stop computing running differences until the motion ends.
4. *Motion compensated pixel differences:* This algorithm resembles the algorithm described by Shahraray,[4] although many details are not specified there. There are three threshold values: cut, high, and low. Each frame is divided into 12 blocks in a 4×3 pattern. Block matching with a 24×18 search window is used to generate a set of motion vectors and a set of block match values. The two highest and two lowest match values are discarded and the remaining values are averaged to produce the match value. If the match value exceeds the cut threshold, then a cut is declared. We keep a cumulative total of the amount that the match value goes above or below the low threshold, with the idea the match values above the low threshold indicate that we might be in a gradual transition. If the cumulative total exceeds the high threshold, we declare a gradual transition once the match value drops below the low threshold. To guard against false positives due to motion, the motion vectors are examined, just as in the running histogram algorithm, to determine if there is a lot of uniform motion. If there is sufficient motion, the cumulative total is reset to a low value.
5. *DCT coefficient differences:* This algorithm closely resembles the algorithm described by Arman, Hsu, and Chiu.[10] One threshold is used. We take the same 15 DCT coefficients from each block of frame and concatenate them to produce a vector. The difference measure is computed by subtracting the inner product of the vectors of consecutive frames from one. If this difference exceeds the threshold, declare a possible shot boundary. Arman, Hsu, and Chiu used two thresholds and used color histograms to decide the in-between cases. We chose to use only one threshold to evaluate the effectiveness of using this algorithm as a filter for other algorithms.

All of these algorithms were implemented in C and run on Unix workstations. The programs generate large output files of image statistics so that the video frames are accessed only once per algorithm. These output files are used as input to Perl scripts that generate lists of shot boundaries based on input thresholds.

All input video was digitized at a size of 320×240 pixels at a frame rate of 30 frames per second using a DEC Alpha equipped with a J300 video board. The digitized video was stored as motion JPEG with a compression ratio of about 25 to 1, requiring about 1 G byte of space to store 1 h of video.

The algorithms were tuned on some small movie clips and a short computer-generated animation sequence. These clips contained 44 cuts, two dissolves, and some pans and zooms. Gradual transitions were also tested on a longer sequence from a movie that contained 51 cuts and 46 fades, wipes, and dissolves. This tuning involved making sure that the algorithms produced reasonable output and getting a sense of the useful ranges for the thresholds.

In order to conduct a comprehensive test of the implemented algorithms, we selected an assortment of video clips as test data. The four types of video we selected were:

1. *Television programs:* This material included a full half-hour episode of ‘‘Friends,’’ the second half-hour of an episode of ‘‘Homicide’’ (lots of camera motion and subtle cuts), and the second half-hour of an epi-

Table 1 Video test data.

Video Type	# of Frames	Cuts	Gradual Transitions
Television	133,204	831	42
News	81,595	293	99
Movies	142,507	564	95
Commercials	51,733	755	254
Miscellaneous	10,706	64	16
Total	419,745	2507	506

sode of ''Babylon 5'' (several computer graphic sequences). The commercials were separated out, leaving 74 min of video.

2. *News programs:* This material included one half-hour of CNN Headline News and a half-hour local news broadcast. Again, the commercials were separated out, leaving more than 45 min of video.
3. *Movies:* This material included the last half-hour of ''Citizen Kane'' (black and white, often dark, some unusual transitions), the first half-hour of ''Raiders of the Lost Ark,'' the first 10 min of ''The Sting'' (several unusual transitions), and the first 9 min of ''The Player'' (one 8-min shot with lots of camera and object motion). This test set was 79 min long.
4. *Television commercials:* The commercials from the television programs and news shows were combined into one test set of almost 28 min.

We also digitized a 6-min Bugs Bunny cartoon that was excluded from the four smaller test sets, but was included in the full data set of more than 233 min of video. The characteristics of the test sets are presented in Table 1. The locations and types of shot boundaries within the test videos were determined by a painstaking manual analysis. The locations of gradual transitions were listed as a range from the frame where the transition was first apparent to the frame where the transition was last apparent.

4 Experimental Results

This section describes the results of applying the implemented algorithms to the test data.

For each of the algorithms a wide range of thresholds was tested: 35 thresholds for histogram, 48 threshold pairs for region, 46 threshold pairs for running, 28 threshold tuples for motion, and 14 thresholds for DCT. For each combination of algorithm, test sequence, and threshold set, we measured the number of shot boundaries that were correctly detected, the number of false positives, and the number of missed boundaries. A gradual transition was correctly detected if any of the frames of the transition was marked as a shot boundary. Algorithms were not penalized for reporting multiple consecutive frames as shot boundaries. This reduced the number of false positives during gradual transitions. However, if for example, an algorithm marked every other frame of a gradual transition as a shot boundary, the first would be a correct detection and the remainder would be false positives.

We chose recall and precision as the appropriate evaluation criteria. Recall and precision are commonly used in the field of information retrieval. Recall is defined as the percentage of desired items that are retrieved. Precision is defined as the percentage of retrieved items that are desired items. From the test results we compute:

$$\text{Recall} = \frac{\text{Correct}}{\text{Correct} + \text{Missed}}, \tag{1}$$

$$\text{Precision} = \frac{\text{Correct}}{\text{Correct} + \text{FalsePositive}}. \tag{2}$$

It is difficult to make comparisons between algorithms based on recall and precision values. For example, an automated video indexing system that uses a human operator to screen the results requires a high recall. A system that summarizes video by selecting a key frame for each minute of video places higher emphasis on precision. In any application a trade-off must be made between recall and precision. It may or may not be acceptable to retrieve one extra shot boundary that would otherwise be missed at the expense of retrieving 100 nonboundaries incorrectly.

This experiment produced a great deal of data that can only be summarized here. This data will be described in further detail in a forthcoming PhD dissertation.[11]

The following data plots do not show recall values below 0.5. In general, the graphs are uninteresting for low recall values because the precision values are roughly constant and even decline slightly as recall approaches zero. In each of the plots, we discarded data points that fell well below the curve. These points were generated due to the interaction of the multiple thresholds. In some cases many points were discarded, especially for the motion algorithm, which often generated four or five different precision values for a given recall value.

Figure 1 shows a plot of recall versus precision for the television programs data set. Overall the algorithms performed best on this data set. That is, for a given recall value, the precision values were higher than for the other data sets. The primary reason for this result is the small number of gradual transitions.

Figure 2 shows of plot of recall versus precision for the news program data set. The set of threshold tuples used for the motion algorithm did not produce high values of recall for this data set.

Figure 3 shows a plot of recall versus precision for the movie data set. For a given recall, precision values are lower than for the other test sets, except for the region algorithm. This data set caused a large number of false positives due to large amounts of camera motion and a large number of missed cuts due to similar backgrounds in adjacent shots.

Figure 4 shows a plot of recall versus precision for the television commercial data set. Performance was poor here due to the large number of gradual transitions and a number of light flashes.

Figure 5 shows a plot of recall versus precision for the full data set and Figure 6 shows a plot of recall versus precision for the full data set for cuts only. The cut-only data was obtained by eliminating from the results all correctly detected and missed gradual transitions and all false

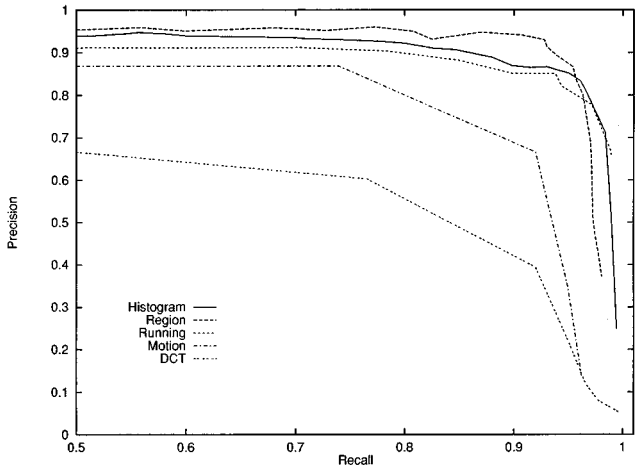


Fig. 1 Recall versus precision for the television program data set.

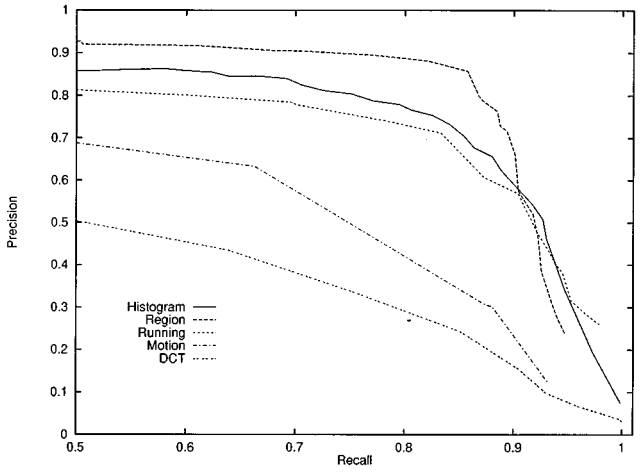


Fig. 3 Recall versus precision for the movie data set.

positives that occurred in shots that ended with a gradual transition. As expected, the cut-only plot is shifted to the upper right.

The histogram algorithm was very consistent. It usually produced the first or second best precision for a given recall value. The simplicity of the algorithm and the straightforward threshold selection make this algorithm a reasonable choice for many applications. At the highest recall level, histogram detected 96% of the gradual transitions.

The region algorithm seemed to be the best algorithm for applications where recall is not the highest priority. In general, for recall values below 0.93, this algorithm had the highest precision values. At the highest recall level, region detected 82% of the gradual transitions.

The running algorithm seems to be the best algorithm for applications where recall is important. In general, for recall values above 0.93, this algorithm had the first or second best precision values. At the highest recall level, running detected 88% of the gradual transitions. However, only 16% of the gradual transitions were correctly identified as gradual transitions at any of the threshold settings. The motion vector analysis did help to reduce the number of false positives.

The motion algorithm did not perform as well as the histogram-based algorithms in general. We expected better results based on the good performance of this algorithm on the tuning data. Most gradual transitions did not reach the high threshold. There were a number of false positives due to camera motion that was greater than the motion vector search window and due to object motion that was not detected well by the block matching. Smaller blocks might work better or perhaps histograms work better than pixel differences for block matching. At the highest recall level, motion detected 75% of the gradual transitions. Only 14% of the gradual transitions were correctly identified as gradual transitions at any of the threshold settings. This algorithm did a good job of deciding that gradual transitions were not cuts, but did not do a good job of deciding that they were shot boundaries.

The DCT algorithm performed as expected, giving low precision values for a given recall. This algorithm generated a large number of false positives in black frames be-

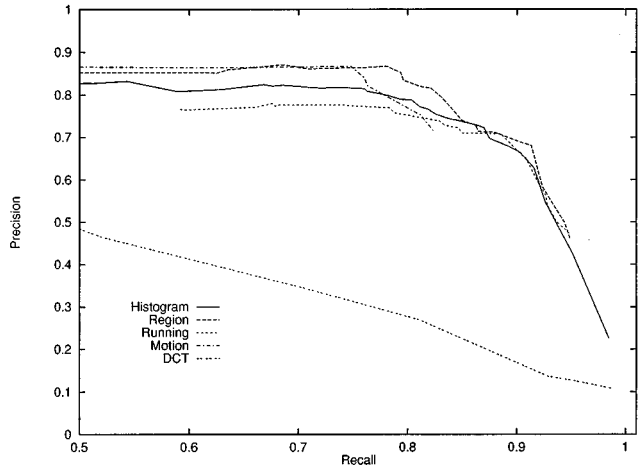


Fig. 2 Recall versus precision for the news program data set.

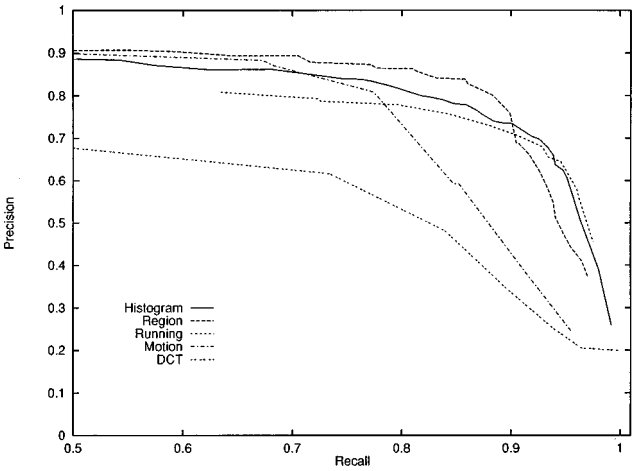


Fig. 4 Recall versus precision for the television commercial data set.

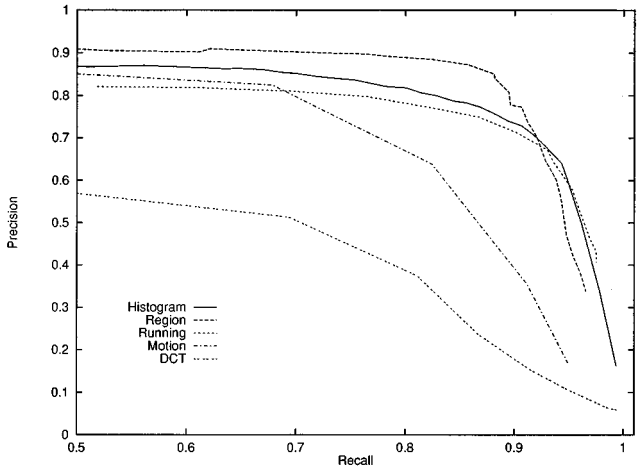


Fig. 5 Recall versus precision for the full data set.

tween television commercials. Because the inner product is normalized for the Euclidean length of the coefficient vectors, small random noise in very dark frames can produce a high frame difference measure. At the highest recall level, DCT detected 99% of the gradual transitions. At this recall level (0.997), DCT generated more than 50,000 false positives. This result means that a secondary algorithm is needed to examine up to one quarter of the video frames, making DCT an ineffective filter.

The effect of changing the thresholds that detect cuts is straightforward for all algorithms. It is much more difficult to evaluate the effect of changes in the thresholds that are used to detect gradual transitions in the running and motion algorithms.

5 Conclusions

The algorithm features that seemed to produce good results were region-based comparisons, running differences, and motion vector analysis. A combination of these three features might produce better results than either the region histogram or the running histogram algorithms. In general, the simpler algorithms outperformed the more complicated algorithms. These complicated algorithms were sensitive to the threshold settings and ''hidden'' parameters not specified in the literature.

Because the tested algorithms did a poor job of identifying gradual transitions, we will implement and test an edge tracking algorithm,[3] which is reported to correctly identify gradual transitions in limited tests.

Ignoring commercials, there were 2004 shot boundaries and more than 200 scene boundaries. More than 30% of those scene boundaries were marked by gradual transitions, whereas just more than 10% of nonscene shot boundaries were marked by gradual transitions. This difference is large enough to make finding gradual transitions an important part of any scene boundary detection algorithm.

The implemented shot boundary detection algorithms are being integrated into the Berkeley Video-on-Demand System.[12] Our video database entry tool will allow people to select an algorithm and thresholds that they want to use for shot boundary detection and allow the user to evaluate and modify the results before entry into the database.

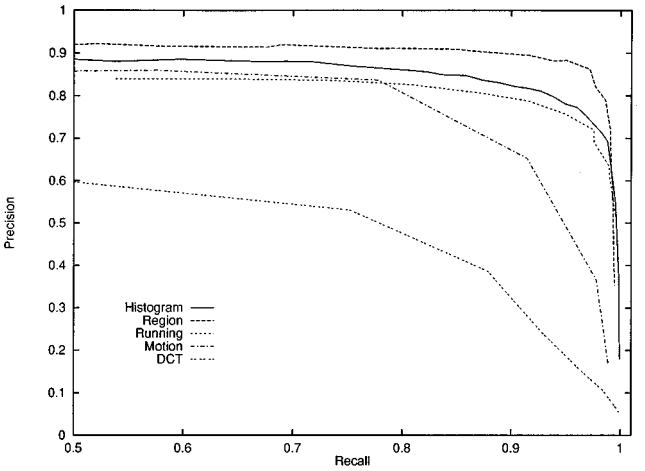
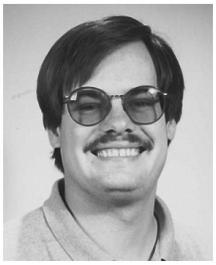


Fig. 6 Recall versus precision for the full data set, cuts only.

References

1. H. J. Zhang, A. Kankanhalli, and S. W. Smoliar, ''Automatic partitioning of full-motion video,'' *Multimedia Systems* **1**(1), 10–28 (1993).
2. A. Nagasaka and Y. Tanaka, ''Automatic video indexing and full-video search for object appearances,'' in *Visual Database Systems II*, E. Knuth and L. Wegner, Eds., pp. 113–127, Elsevier Science Publishers (1992).
3. R. Zabih, J. Miller, and K. Mai, ''A feature-based algorithm for detecting and classifying scene breaks,'' *Proc. ACM Multimedia 95*, pp. 189–200, San Francisco, CA (1995).
4. B. Shahraray, ''Scene change detection and content-based sampling of video sequences,'' in *Digital Video Compression: Algorithms and Technologies, Proc. SPIE* **2419**, 2–13 (1995).
5. A. Hampapur, R. Jain, and T. Weymouth, ''Digital video segmentation,'' *Proc. ACM Multimedia 94*, pp. 357–364, San Francisco, CA (1994).
6. R. Kasturi and R. Jain, ''Dynamic vision,'' in *Computer Vision: Principles*, R. Kasturi and R. Jain, Eds., IEEE Computer Society Press, Washington (1991).
7. H. Ueda, T. Miyatake, and S. Yoshizawa, ''IMPACT: an interactive natural-motion-picture dedicated multimedia authoring system,'' *Proc. CHI. 1991*, pp. 343–350 ACM, New York (1991).
8. D. Swanberg, C. F. Shu, and R. Jain, ''Knowledge guided parsing and retrieval in video databases,'' in *Storage and Retrieval for Image and Video Databases, Proc. SPIE* **1908**, 173–187 (1993).
9. T. D. C. Little, G. Ahanger, R. J. Folz, J. F. Gibbon, F. W. Reeve, D. H. Schelleng, and D. Venkatesh, ''A digital on-demand video service supporting content-based queries,'' *Proc. ACM Multimedia 93*, pp. 427–436, Anaheim, CA (1993).
10. F. Arman, A. Hsu, and M-Y. Chiu, ''Image processing on encoded video sequences,'' *Multimedia Systems* **1**(5), 211–219 (1994).
11. J. Boreczky, ''Using video and audio data for content extraction and shot and scene boundary determination,'' PhD dissertation, University of California Berkeley, to appear (1996).
12. L. Rowe, J. Boreczky, and C. Eads, ''Indexes for user access to large video databases,'' in *Storage and Retrieval for Image and Video Databases II, Proc. SPIE* **2185**, 150–161 (1994).



John S. Boreczky received BS and MS degrees in computer science from the University of Michigan in 1987 and 1989, respectively. He is currently working toward a PhD degree at the University of California at Berkeley. His research interests include video-on-demand systems, multimedia databases, and video content extraction.

Lawrence A. Rowe received a BA in mathematics and a PhD in information and computer science from the University of California at Irvine in 1970 and 1976, respectively. Since 1976 he has been on the faculty at the University of California at Berkeley, where he is now a professor of electrical engineering and computer science. He is also the founding director of the Berkeley Multimedia Research Center. Professor Rowe's research interests are multimedia systems and applications. He heads the research group that developed the Berkeley MPEG1 video tools (i.e., software decoder, parallel encoder, and utilities), the Berkeley Continuous Media Toolkit, algorithms to compute special effects on compressed images, and the Berkeley Distributed Video-on-Demand System. He has published more than 50 papers on multimedia systems and applications, programming systems, and database systems, and he is a member of the editorial board of the *ACM Multimedia Systems Journal*.