HDH Based Compressed Video Cut Detection

Bo Shen*

Keywords: Cut detection, Hausdorff distance histogram, MPEG domain edge detection, Shot detection

Abstract

This paper presents a video cut detection algorithm using multi-level Hausdorff distance histograms (HDH). Hausdorff distance is obtained by comparing edge points of successive frames, wherein the edge information is extracted from compressed frames directly. The use of Hausdorff distance histogram instead of the comparison of entering/exiting edge pixel counts [9] makes the algorithm more robust to complicated camera shots. The extraction of edge information in compressed domain greatly escalates the cut detection process, which is critical for indexing of large amount of video materials in large scale video databases. The performance of this algorithm has been tested on a variety of videos. The experimental results show that this algorithm can robustly tolerate rapid changes in scene brightness as well as multiple object and camera motions.

*Bo Shen is with the Visual Computing Department, HP Laboratories, 1501 Page Mill Road, Palo Alto, CA 94304, U.S.A. Email: boshen@hpl.hp.com

1 Introduction

One of the key problems in video data management is the issue of video content representation. A widely accepted method for this purpose is the use of key frames. The extraction of key frames from a video requires isolation of individual video shots by locating shot boundaries or cut point. Two types of shot transitions or boundaries are present in an edited video: (1) the straight cut, and (2) the optical cut. A straight cut is an abrupt transition from one shot to the next. Usually the straight cuts are well-defined and relatively easy to detect. An optical cut provides a visually gradual transition between two shots and takes place over a sequence of frames. Compared to the straight cuts, optical cuts are more sophisticated and generally difficult to isolate.

Driven by video databases and multimedia applications, a large number of methods for automatic cut detection have been presented in recent years. Some earlier methods for cut detection are based on measuring frame difference either at the pixel level or at the block level. However, the frame difference methods are sensitive to camera or object movement, noise, and illumination changes. In order to avoid this weakness, a number of methods based on some global or local features have been proposed by many researchers. For example, Zhang et al [11] used histogram differences to detect straight cuts and gradual transition. Ueda, Miyatake,and Yoshizawa [10], and Zhang et al [11] used motion vectors to obtain improved cut detection. Recently, Zabih et al [9] proposed a method based on edge features, which appears to be more accurate at detecting cuts than intensity histograms. Most of the methods for cut detection operate on uncompressed video, however. A number of cut detection methods for compressed video have been suggested lately as digital video is becoming common place. Not only are such methods able to take the advantage of the lower data rate of compressed video, they also avoid extra computation involved in decoding when the incoming video is in compressed form. The examples of compressed video cut detection methods include Arman, Hsu, and Chiu [1], Yeo and Liu [4], and Sethi and Patel [6]. Patel and Sethi [5] have also shown how a number of cut detection methods suggested for uncompressed video can be implemented for MPEG video. While compression domain methods are superior in respect of computing requirements, these methods usually have lower success rate compared to methods operating upon uncompressed video.

We propose in this paper a method possessing the accuracy of the feature-based method and the efficiency of the compressed domain cut detection. The feature-based side of our method is motivated by the work proposed in [9] wherein edge information has been shown to provide excellent information for decisions of shot boundary. In [9], the entering and exiting edge pixel counts were used to decide where a cut occurs in an image sequence. Since a small amount of dilation is applied on the reference frame, this method can perform well for camera shots containing small relative object motions. However, if one object in the scene has a motion much larger than anther object (as would occur in many video shots having objects at different depths), this method would generate many false cuts. To improve this, we propose the use of Hausdorff distance histogram and develop a multi-pass merging algorithm to get rid of noise at each pass. To improve computational efficiency, our method uses a compressed domain edge extraction method to obtain edge information rapidly [8]. The performance of the proposed method has been tested on a variety of videos. The experimental results indicate that this method permits fast and accurate detection of shot boundaries.

The reminder of the paper is organized as follows. Section 2 presents our cut detection algorithm via the establishment of Hausdorff distance histogram. Section 3 describes our convolution-based edge detection in the block DCT domain. The experimental results for the detection of straight cuts and optical cuts are presented in Section 4. Finally, a summary of the work and future directions are provided in Section 5.

2 Edge-based cut detection

Based on the edge feature extracted directly from the I frames of an input MEPG video sequence, our scheme of using Hausdorff distance histogram (HDH) to locate cuts is presented in this section. This approach is based on the following considerations. The distribution of edges in current frame will be quite different from that in past frames if a cut occurs. On the other hand, if there is no cut between two frames, then the corresponding edge points from these two frames within a small region will have very similar movement. In most cases, the edge point motion in adjacent regions will be similar to some extent. Therefore, we will establish the Hausdorff distance histograms based on each small region. From computation complexity point of view, it is same as obtaining the histogram based on the whole edge map.

This approach consists of the following three steps:

1. The edge map of a frame is decomposed 8 times in both horizontal and vertical direction to generate 64 regions. The Hausdorff distance histograms are obtained for each region by comparing the edge points extracted from successive I frames.
2. The histogram of the whole frame is obtained by merging the histograms of subregions in multiple passes. The merging algorithm is designed to increase SNR of true motion during each pass while suppressing mismatch information introduced by noise.
3. The shot breaks can be accurately detected by using the peak value of the Hausdorff distance histogram at the frame level.

Fig. 1 shows the flow chart of the algorithm. On the top row of the chart, the frames with solid lines are treated as base frames for which the histograms will be established. The frames with doted lines are the reference frames. Since both the current frame and the last frame can be treated as base frames, two sets of histograms (using either last frame or current frame as the base frame) are obtained and used for cut detection.

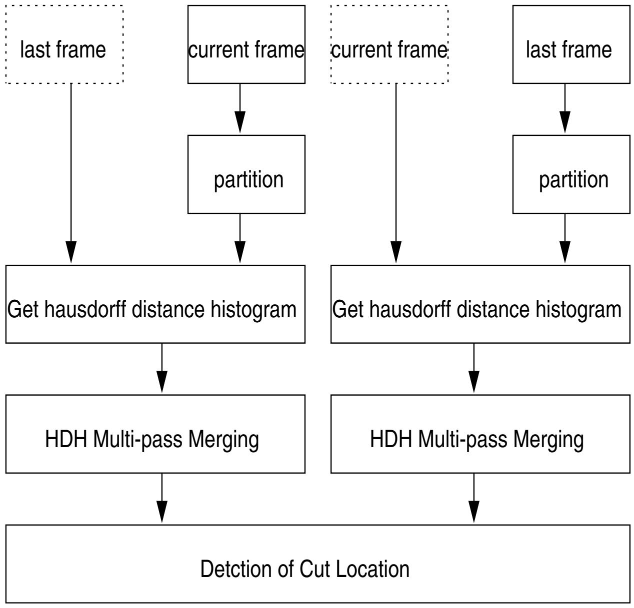


Figure 1: **Processing flow**

2.1 Establishment of HDH in subregions

In this step, we divide the base frame into 64 regions and then establish the Hausdorff distance histogram for each region. As we have mentioned above, when no cuts occur, the edge points within a small region will have very similar movement, that is, these edge points will have a distribution similar to the reference frame. If there is a cut, the distribution of the edges will be quite different between the base frame and the reference frame. Thus, we can detect the cuts by simply measuring the similarity of edge distribution between these two frames. The Hausdorff distance, a very common tool for measuring the degree of resemblance between two point-sets, is employed to measure this similarity.

As defined in [3], given two finite point sets A and B, the Hausdorff distance is computed as:

$$H(A,B) = \max(h(A,B), h(B,A)) \tag{1}$$

where

$$h(A,B) = \max_{a \epsilon A} \min_{b \epsilon B} ||a - b||,$$

and $||\cdot||$ denotes a normalization on the points of A and B.

The function $h(A,B)$ is called the directed Hausdorff distance from A to B. Most of the applications use the best partial distance, a generalization of the Hausdorff distance. It is expressed as :

$$h_K(A,B) = K^{th}_{a\epsilon A} \min_{b\epsilon B} ||a-b||. \tag{2}$$

In our approach, we fix the $h_K(A,B)$ as a threshold h, and calculate the value K to measure the similarity. The quantity K denotes the number of points in model set A whose minimum distance from the reference set B is less than threshold h. By calculating the K value for each possible translation of the region, we can obtain a histogram $\{K_{i,j}\}$ for each region in the base frame, where i and j correspond to displacements in X and Y direction, respectively. The threshold h is mainly related to the tolerance of the relative movement within a region and the edge position error caused by edge detector. A fast implementation of this algorithm is as follows.

1. The edge points in the reference frame are dilated by a disk of radius h.

2. For each region in the base frame, the histogram $K_{i,j}$ is calculated as

$$K_{i,j} = \sum_{a_{x,y}\epsilon A} f_{i,j}(a_{x,y}) \qquad |i| \leq D_x \; and \; |j| \leq D_y \tag{3}$$

   where

$$f_{i,j}(a_{x,y}) = \begin{cases} 1, & if\ a_{x,y}\epsilon A\ and\ b_{x+i,y+j}\epsilon B^+ \\ 0, & Otherwise \end{cases} .$$

   Here Dx and Dy are the maximum possible movement for a large object in x and y direction, respectively. A is the set of edge points within a region of base frame. B^+ is the dilated edge point set of the reference frame. Thus, we get a motion distance histogram for each region within the base frame.

2.2 Multi-pass merging of HDH

The histograms based on small regions can robustly tolerate object and camera motion but are sensitive to noise and contain mismatch information. We use a multi-pass merging process to obtain the final HDH which has much less noise and mismatching in order to be used in the final cut detection stage.

Since the actual movement of pixels within a region usually corresponds to a relatively high value in the HDH, we can eliminate most of the noise and mismatches by simply using a threshold and setting all those values below the threshold to zero:

$$K'_{i,j} = \begin{cases} K_{i,j}, & if\ K_{i,j} > Th \\ 0, & Otherwise \end{cases} \tag{4}$$

where

$$Th = \max\{\alpha \times n, \beta \times (p - m) + m\}.$$

In the above equation, n is the total number of edge points in the region, p is the peak value of the histogram, and m is the mean value of the histogram. α and β are two constants ranged from 0 to 1. In our experiments, we choose 0.36 for α and 0.2 for β.

Since the neighboring regions usually have similar motion, we can further improve the signal-to-noise ratio by combining the HDHs of neighboring regions to obtain new histograms at a higher level. In order to make the new histograms still robustly tolerate object and camera motion, we apply an overlapped sampling on the old histograms to lower their motion resolution before we merge them. The sampling process for each HDH is defined as the selection of the peak values within overlapped sampling windows. It can be expressed as:

$$\begin{aligned} K_{m,n} = \max(K'_{i,j}) \quad & \lambda(m-1)+1 \leq i \leq \lambda(m+1)-1 \\ & and\ \lambda(n-1)+1 \leq j \leq \lambda(n+1)-1 \end{aligned}$$

where λ is an integer denoting the sampling rate. $K'_{i,j}$ is obtained from Eq. (4). After this sampling, the new histogram will have λ times fewer bins in both x and y direction. The histogram for next higher level can be obtained by adding up the corresponding bins of the neighboring four sampled histograms. This merge process is defined as:

$$K^{+}_{m,n} = \sum^{4} K_{m,n}.$$

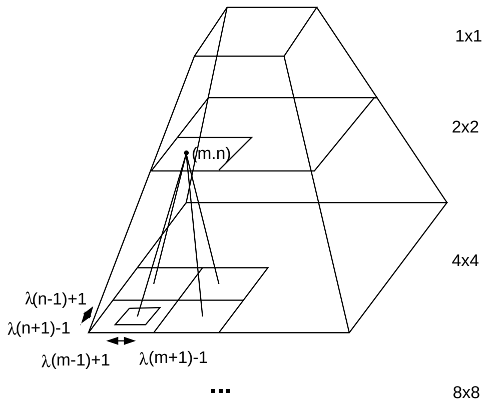


Figure 2: **Multi-level HDH sampling and merging process**

Since we decomposed the original frame 8 times along both directions and obtained HDHs for each 64 regions, we now repeat the sampling and merging processes discussed above. As shown in Fig. 2, four neighboring regional HDHs are first sampled and then merged to generate one HDH at each pass. Repeating the above process 4 times as shown in Fig. 2, we can finally obtain a single HDH representing the whole base frame. We call it the frame level HDH. Although this HDH has lower motion resolution, it can robustly measure the similarly between the edges of two successive frames. If there is no cut between them, the peak value of the HDH will be very high, otherwise, it will be very low.

2.3 Detection of cut location

The detection of shot breaks is based on the peak values of the histograms on frame level. From the above discussion, for each regional HDH, lower peak value indicates the movement of edge pixels is uniformly distributed, therefore, it could not be the result of an object movement. It could be introduced by a scene cut, where incoming edges are largely different than the out-going edges. However, it also could be simply because there is not so many changes in edge pixel locations. Since the situation is excluded out by selection of threshold in Eq. (4), lower peak value of HDH generally gives as indication of larger difference in respect of distribution of edge points within that region between two frames. Through the multi-pass merging of HDH from each region, our algorithm makes sure that the peak value decreases if the difference is introduced due to the occurrence of a real cut. On the other hand, the peak value is brought up during this multiple-pass merging if the difference is introduced due to noise or mismatching. Therefore, the peak value of the HDH obtained at the frame level denotes more correctly whether a cut occurs or not.

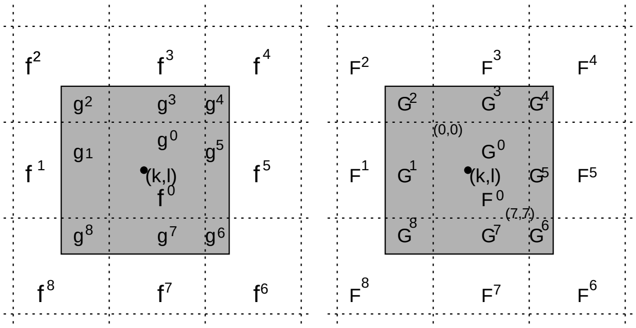
As we can see from Eq. (2), the HDH uses the number of edge points to measure the similarity. Since the frames in a video sequence usually do not have uniform total number of edge points, we need to normalize the HDHs. Letting P denote the total number of edge points in the base frame, we express the normalized HDH as:

$$\overline{K_{m,n}} = \frac{K_{m,n}}{P}$$

Since both the current frame and the last frame can be used as the base frame, we will get two peak values for each frame. In our experiments, only the smaller peak values are used to locate cuts. In order to localize shot breaks that occur over multiple frames, we restrict the cuts to occur only when the lower peak value is a local minimum within a number of consecutive frames that have HDH peak values below the threshold.

3 Edge extraction in compressed domain

Since compressed video offers a lower data rate, our cut detection method performs edge detection directly on I frames of MPEG [2] video without full decompression. The details of our edge detection scheme can be found in [8]; we provide here only a brief sketch of the compressed domain edge detector.



(a) f^n is the overlapping area regarding g^n (b) F^n is DCT block

Figure 3: **Block-DCT convolution**

The main difficulty in performing edge detection in compressed domain is the presence of blocks which are individually transformed during the compression process. To illustrate this difficulty, consider Fig. 3. Since edge detection can be viewed as a convolution operation, the result at pixel location (k, l) can be expressed as

$$h_{k,l} = \sum_n \Big(\sum_i \sum_j f^n_{i,j} g^{n,k,l}_{i,j} \Big). \tag{5}$$

As we intend to use block DCT data as direct input, pixel location (k, l) is considered with one of the blocks in the block-DCT domain. The doted lines in Fig. 3 show the 8×8 block grid. These lines decomposes the convolution kernel into 9 regions, g^0 to g^8. The nine neighborhood blocks associated with them are marked as f^0 to f^8.

Generally, the kernel does not have to be square, we assume square kernel of size N for simplification. For N less than or equal to 17, n is no larger than 9. It means there are at most 9 neighborhood blocks that are involved in the computation of the convolution result for pixel location (k, l).

Detail derivation can be found in [8], here we show the final equation for computing of the convolution result directly from DCT blocks:

$$h_{k,l} = \sum_n < F^n, G^{n,k,l} > . \tag{6}$$

where

$$G^{n,k,l} = \sum_i \sum_j g^{n,k,l}_{i,j} T^{i,j}. \tag{7}$$

$T^{i,j}$ is one of the 8×8 (indexed by x, y) DCT matrix and defined as:

$$T^{i,j}_{x,y} = \frac{C_x C_y}{4} \cos \frac{(2i+1)x\pi}{16} \cos \frac{(2j+1)y\pi}{16}. \tag{8}$$

From above, G^n is a sum of tensor blocks scaled with corresponding mask entry. It is an 8×8 matrix or a vector of size 64 and can be calculated before hand and kept in memory. We call it convolution tensor. Therefore, the convolution result at location (k, l) is nothing but the sum of several inner products of two vectors of size 64. F^n is the DCT block in compressed domain, typically only 10% of its coefficients are nonzero which means approximately 7 coefficients in F^n is nonzero. The computation of Eq. (6) only needs $7n$ multiplications and $6n$ additions.

Specifically, for isotropic kernel, which is often used in many edge detection algorithms, we can take advantage of the symmetric property of DCT itself to derive even more efficient algorithms. For instance, the convolution tensors at symmetric locations within a block can be represented by one tensor with some sign-reversal processes. The convolution tensor at location $(k, 7-l), (7-k, l)$ and $(7-k, 7-l)$ can be computed easily from the convolution tensor at (k, l), that is:

$$\begin{aligned} G^{0,7-k,l} &= [S][G^{0,k,l}] \\ G^{0,k,7-l} &= [G^{0,k,l}][S] \\ G^{0,7-k,7-l} &= [S][G^{0,k,l}][S] \end{aligned} \tag{9}$$

where the entries of [S] is defined as $S_{i,j} = (-1)^i \delta(i, j)$, $(i, j) = (0, 0)...(7, 7)$.

Therefore, the contribution of region 0 to the four symmetric locations can be expressed as

$$\begin{aligned} h^0_{k,l} &=< F^0, G^{0,k,l} > \\ h^0_{7-k,l} &=< F^0, SG^{0,k,l} > \\ h^0_{k,7-l} &=< F^0, G^{0,k,l}S > \\ h^0_{7-k,7-l} &=< F^0, SG^{0,k,l}S > . \end{aligned} \tag{10}$$

The above four equations only need one set of multiplications – the multiplications required by inner product of the first equation of Eqs. (10). The only difference for the rest of equations in Eqs. (10) is to accumulate the products from the same set of multiplications with different signs.

Some algorithmic tricks have to be applied on the algorithm to perform convolution on the whole image [8]. Mathematically, it has been proved that the speedup is proportional to the sparseness of the DCT blocks. Denote N the kernel size and P the average percentage of non-zero coefficients in DCT blocks, the number of multiplications required in our method is $P(\lfloor \frac{N}{2} \rfloor + 4)^2$. In general, we experience a speedup of 3 to 11 for each I frame of some QCIF MPEG videos. Fig. 4 shows the edge detection results from a MPEG stream ("Erika1"). An 11×11 LoG operator is used with zero-crossing to extract out the row edge pixels. No additional edge linking or individual edge points exclusion processes are performed.

Edges in P or B frames can be derived from the edge map of reference frames using motion vectors. It should be noted that interframes may contain intra macroblocks. Thus, edge

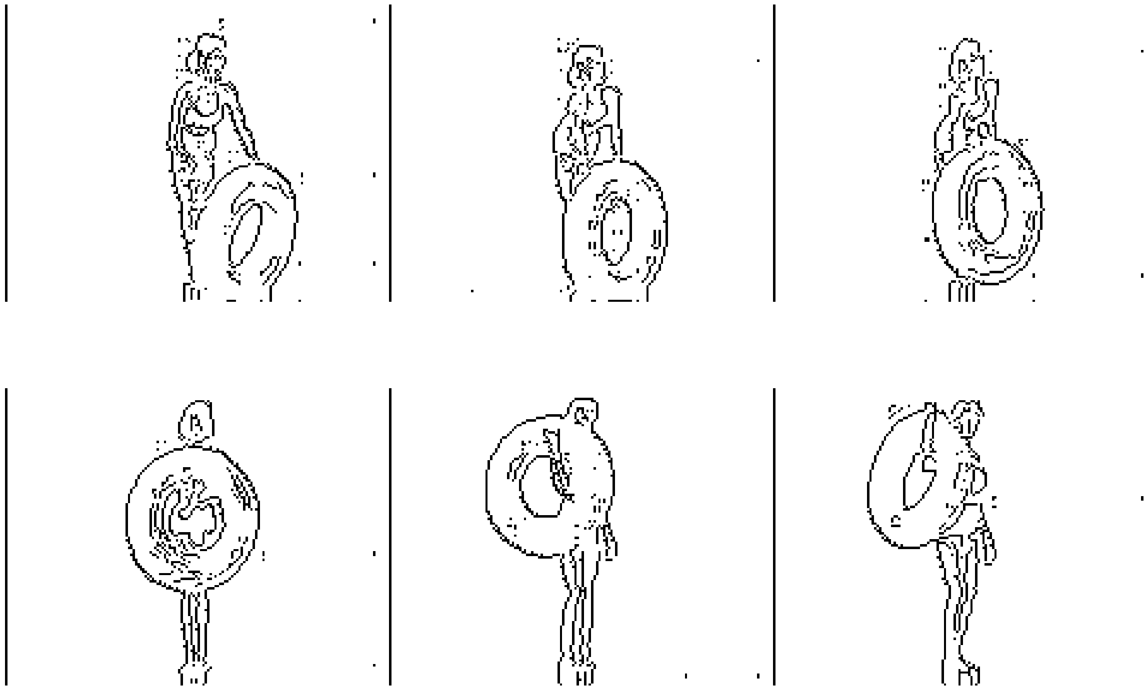


Figure 4: **Raw edge maps from compressed domain edge detection**

extraction discussed above can be applied on them directly. In fact, the percentage of intra macroblocks contained in an interframe may give some clue on whether a cut occurs. Since we concentrate on using edge information, we shall not discuss it here.

4 Experimental results

To evaluate the performance of this suggested cut detection scheme, we examine the behavior of this method on a variety of videos. These videos consist of one MPEG benchmark sequence – table-tennis video, two musical videos, two movie videos and two small TV video sequences. The two musical videos are the "Madonna" video and a small sequence from a Machael Jackson music video (Danger). The "Madonna" video has fast motion in close-ups with special effects and very smooth optical transitions. The two movie videos, the "Few Good Man" video (Movie1)and the "Body Guard" video (Movie2), contain a large number of shots with rapid brightness change, fast camera motions and multiple moving objects. The two small TV video sequences (Erika1 and Erika2) involve fast object and camera motions.

The results of our cut detection scheme are shown in Table 1. The number of cuts are represented in the form of straight cut:optical cut. The performance in terms of recall and precision is also given. Recall is defined as the percentage of desired cuts that are retrieved, Precision is defined as the percentage of retrieved cuts that are desired. They are computed by the following expression:

$$Recall = \frac{n_c}{n_c + n_m}$$

and

$$Precision = \frac{n_c}{n_c + n_f}$$

where n_c, n_m and n_f are the number of correct missed and false cuts, respectively. From Table 1, we can notice that recall and precision are very high although these videos contain a large number of complicate shots with multiple moving objects, fast camera motions, or rapid scene brightness changes.

MPEG	GOP size	Frame #	Actual Cut	Detected	Missed	False	Recall	Precision
Erika1	1	48	2:0	2:0	0:0	0:0	100	100
Erika2	1	50	4:1	4:1	0:0	0:0	100	100
Tennis	12	150	2:0	2:0	0:0	0:0	100	100
Danger	1	150	4:0	4:0	0:0	0:0	100	100
Madana	4	3150	60:3	60:1	0:2	0:0	96.8	100
Movie1	3	7600	39:29	36:27	3:2	6	92.6	91.3
Movie2	3	17627	76:65	75:61	1:4	5	96.5	96.5
Total		28775	187:98	183:90	4:8	11	95.8	96.1

Table 1: **Cut detection result**

The algorithm is implemented on a Sparc workstation with a 70-MHz RISC processor. The running time of performing the edge detection and cut detection on I frames of MPEG sequence is given in Table 2. The running time mainly depends on the number of edge points and histogram bins. Experiments show that a reasonable high threshold for the edge detector will give an impressive speedup without affecting the accuracy of cut detection. Although this algorithm has already shown a reasonably high speed, many other methods can be derived to further improve the performance of the Hausdorff distance search. For example, we can use Hausdorff distance information in the previous frame and motion information. This is especially useful when we have the video sequence in MPEG form.

To further evaluate our approach, we reconstructed the method proposed in [9]. Its result on the "Erika2" sequence is shown in Fig. 5b. Fig. 5a shows our method using the normalized peak value of HDH obtained from each frame of the same video.

Fig. 5 shows our method has superior performance. The first four true cuts can be detected by both methods, except that our method gives more discrimination on the dynamic range. From frame 32 to 39, as shown in the top row of Fig. 6, there is a complex camera shot. It

MPEG File	Frame Size	GOP Size	Frame/Sec.
Tennis	352x240	12	0.53
Erika2	160x128	1	1.62
Danger	160x128	1	1.88

Table 2: **Cut detection speed**

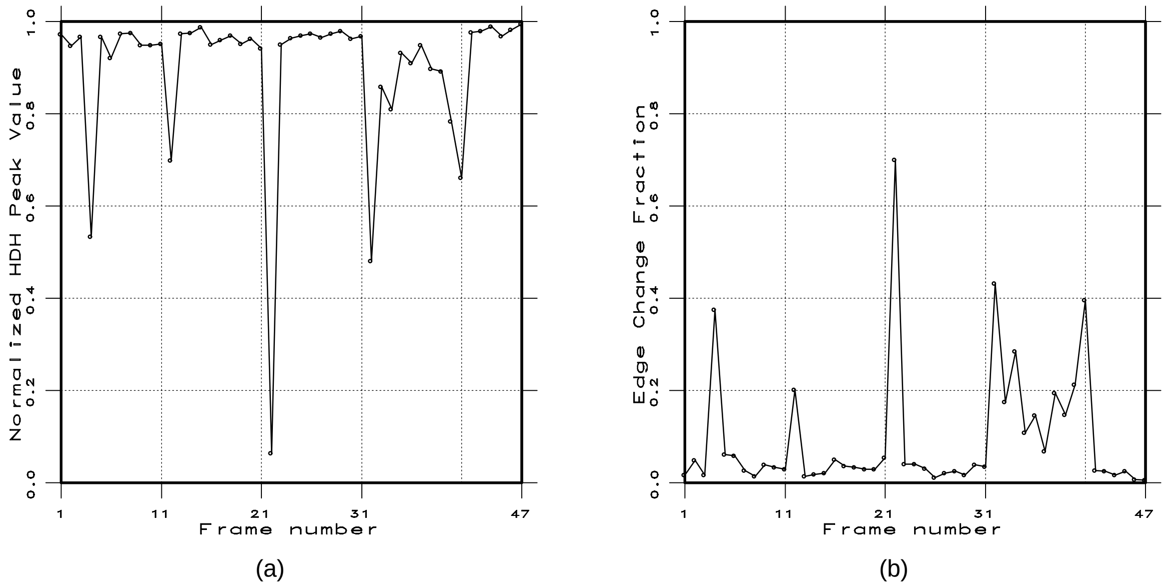


Figure 5: **Comparison of two methods.**

contains object occlusion (a hand moves through part of the head) while the head is rotating (actually in 3D space), and in the mean time, the camera is panning right. Also shown in bottom row of Fig. 6 is the edge maps for frames 40 to 42 where an optical (cross-dissolve) cut occurs. In both two cases, our method can provide better discrimination between true-cut and non-cut frames. As can be noticed from Fig. 5b, if a global threshold has to be chosen for the cut detection of the whole sequence, Zabih's method will get false detections at frame 34 and frame 38 in order not to miss the true cut at frame 12. Of course, these false detections can be avoided in this case by using a window technique [9]. However, the selection of the window size would pose a limitation on the system.

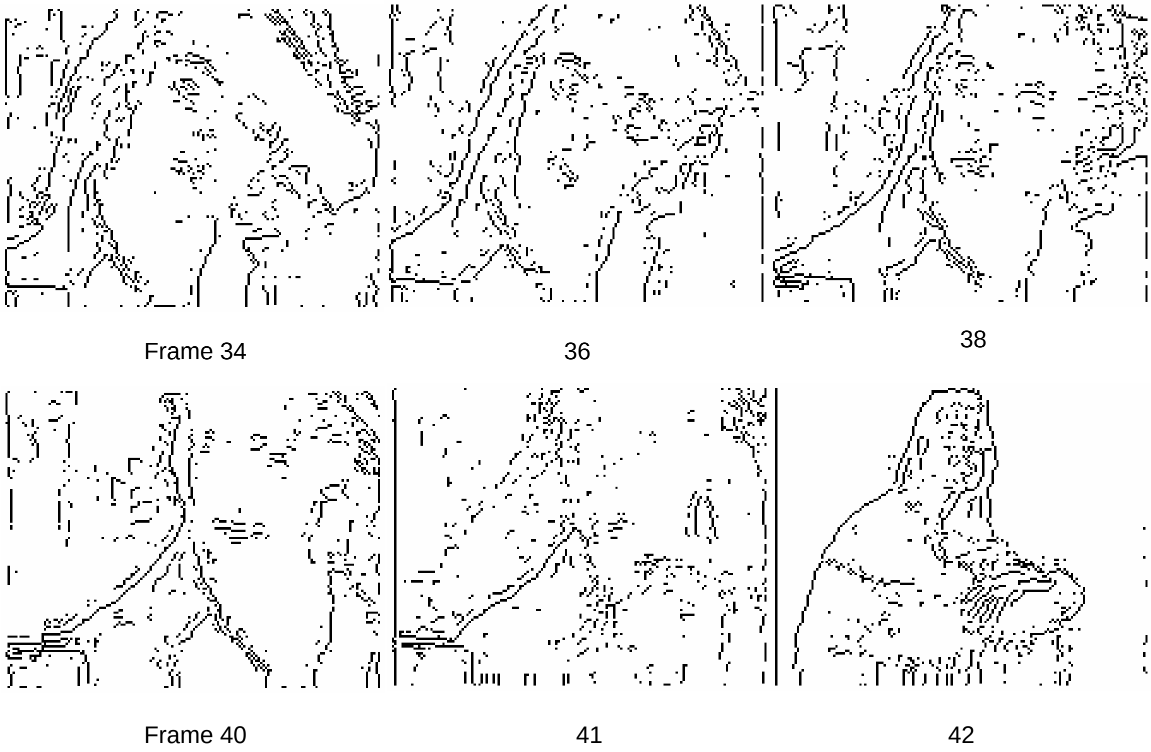


Figure 6: **Row edge extracted from Erika2 sequence**

5 Conclusion and future work

We have presented a method performing cut detection via compressed domain edge extraction. As we can notice from the experimental results, this method can detect shot breaks accurately and efficiently. Using the edge features, this method can effectively decrease the influence of rapid changes in scene brightness. Using the Hausdroff distance histograms on subregions and merging them up through multiple passes, this algorithm can robustly tolerate multiple object and camera motions. The scheme of performing the detection directly on the compressed domain leads to a significant computational speedup. We are currently investigating the possibility of using some simple features of P and B frames to further improve the performance of our cut detection method. We are also exploring the issue of performing shot break classification based on the features of Hausdorff distance histogram. It is expected that such a classification scheme will further aid in shot boundary detection.

6 References

[1] F. Arman, et al, "Image Processing on Compressed Data for Large Video Databases,"

Proc. ACM Intl. Conf. Multimedia, June 1993.

[2] D. L. Gall, "MPEG: A Video Compression Standard for Multimedia Applications," Communications of the ACM, vol. 34, no. 4, pp.47-58, Apr. 1991.

[3] Daniel P. Huttenlocher, Gregory A. Klanderman and William J. Rucklidge, "Comparing Images Using the Hausdorff Distance," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 15, no. 9, pp. 850-863, Sept. 1993.

[4] Boon-Lock Yeo and Bede Liu, "Rapid scene analysis on compressed video," IEEE Trans. on Circuits and Systems for Video Technology, vol. 5, no. 6, pp. 533-544, Dec. 1995.

[5] N. V. Patel and I. K. Sethi, "Compressed Video Processing for Cut Detection," IEE Proceedings - Vision, Image and Signal Processing vol. 143, no. 5, pp. 315-323, Oct. 1996.

[6] I. K. Sethi and N. V. Patel, "A statistical approach to scene change detection," SPIE Proc. Storage and Retrieval for Image and Video Databases III, vol. 2420, pp. 329-339, San Jose, February 1995.

[7] Bo Shen and Ishwar K. Sethi, "Inner-Block Operations On Compressed Images," Proc. ACM Intl. Conf. Multimedia'95, pp. 490-499, San Francisco, Nov. 1995.

[8] Bo Shen and Ishwar K. Sethi, "Convolution-based edge detection for image/video in block DCT domain," to appear in Journal of Visual Communications and Image Representation.

[9] R. Zabih, J. Miller and K. Mai, "A Feature-Based Algorithm for Detecting and Classifying Scene Breaks," Proc. ACM Intl. Conf. Multimedia'95, pp. 189-200, San Francisco, Nov. 1995.

[10] H. Ueda, T. Miyatake and S. Yoshizawa, "IMPACT: An interactive Natural-motion-picture dedicated multimedia authoring system," Proc. of CHP91, New Orleans, Louisiana, Apr. 1991.

[11] Hongjian Zhang, A. Kankanhalli and S. Smoliar, "Automatic Partitioning of Full-Motion Video," Multimedia Systems, vol. 1, no. 1 pp. 10-28, 1993.