CUT DETECTION VIA COMPRESSED DOMAIN EDGE EXTRACTION

Bo Shen, Donnge Li and Ishwar K. Sethi

Vision and Neural Networks Laboratory.
Department of Computer Science
Wayne State University
Detroit, Michigan 48202, USA
{bos,dil,sethi}@cs.wayne.edu

ABSTRACT

This paper presents a video cut detection algorithm using multi-level Hausdorff distance histograms. 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 [7] makes the algorithm more robust to complicated camera shots. The experimental results show that this algorithm can robustly tolerate rapid changes in scene brightness as well as multiple object and camera motions.

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 [8, 9]. Recently, Zabih et. al [7] 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 the compressed form. Examples of such algorithms can be found in [1, 4, 5]. While compressed domain methods are superior in terms 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 [7] wherein the edge feature has been shown to provide excellent information for decisions of shot boundary. In [7], 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 [6].

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 basic 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 basic frames, two sets of histograms are obtained and used for cut detection.

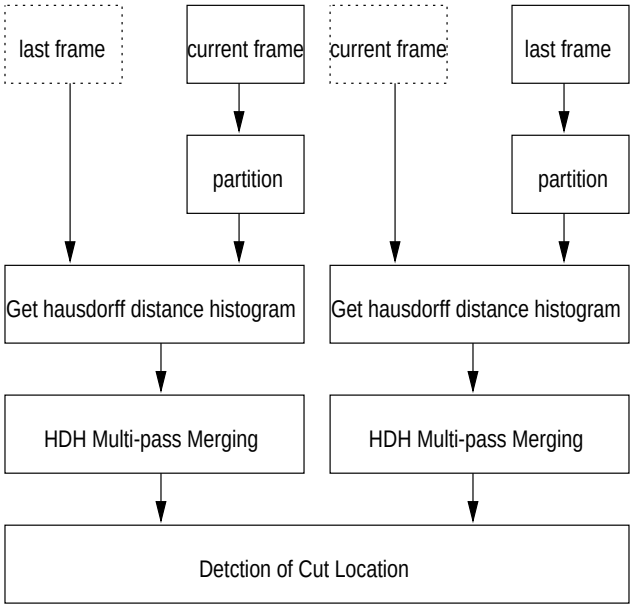

Figure 1: **Processing flow**

2.1. Establishment of HDH in subregions

In this step, we divide the basic frame into 64 regions and then establish the Hausdorff distance histogram for each of the 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 basic 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 points sets, is used 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 basic 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 basic 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 basic frame. B^+ is the dilated edge points set of reference frame. Thus, we get a motion distance histogram for each of the regions within the basic frame.

2.2. Multi-pass merging of HDH

The histograms based on small regions can robustly tolerate moving object and camera motion but are sensitive to noise and contain mismatching 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 mismatching by simply using a threshold and setting all those value 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.

Because 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

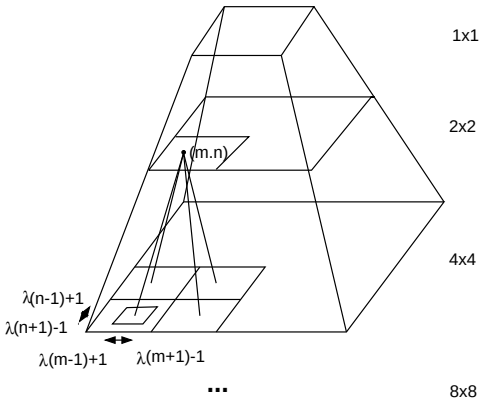

Figure 2: **Multi-pass merging of HDH**

for each HDH is defined as the selection of the peak values within overlapped sampling windows. It can be expressed as:

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

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:

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

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 region HDHs are first sampled and then merged to generate one HDH at each pass. By repeating the above process 4 times as shown in Fig. 2, we can finally obtain a single HDH based on the whole basic frame. 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 layer. From the above discussion, for HDH of each region, 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. Or it could be simply because there is not so much change in edge pixels. 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 terms of distribution of edge points within that region between two frames. Through the multiple-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. Hence, the peak value of the HDH obtained at the whole 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 basic frame, we express the normalized HDH as:

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

Because both the current frame and the last frame can be used as the basic frame, we will get two peak values for each frame of the video sequence. 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 which have HDH peak values below the threshold.

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 [6]. Mathematically, it has been proved that the speedup is proportional to the sparseness of the DCT blocks. In general, we experience a speedup of 3 to 11 for each I frame of some QCIF MPEG videos.

3. 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 (M3), two musical videos (M4-5), two movie videos (M6-7) and two small TV video sequences (M1-2). The two small TV video sequences involve fast object and camera motions. These materials can be accessed at: http://www.cs.wayne.edu/~dil/.

	GOP Size	Frame Num.	True Cut	Detected	Missed	False
M1	1	48	2:0	2:0	0:0	0
M2	1	50	4:1	4:1	0:0	0
M3	12	150	2:0	2:0	0:0	0
M4	1	150	4:0	4:0	0:0	0
M5	4	3150	60:3	60:1	0:2	0
M6	3	7600	39:29	36:27	3:2	6
M7	3	17627	76:65	75:61	1:4	5
Sum		28775	187:98	183:90	4:8	11

Table 1: **Cut detection result**

MPEG File	Frame Size	GOP Size	Speed (fps)
M2	160x128	1	1.62
M3	352x240	12	0.53
M4	160x128	1	1.88

Table 2: **Cut detection speed**

The results of our cut detection scheme are shown in Table 1. The number of cuts is represented in the form of "straight-cut:optical-cut". From Table 1, we can notice that the recall and precision (95.8% and 96.1% on average, respectively) for these videos are very high although they contain a large number of complicated shots with multiple moving objects, fast camera motions and rapid brightness changes.

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. The experiments show that a reasonably high threshold for the edge detector will give an impressive speedup without affecting the accuracy of the 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 [7]. Its result on the "M2" sequence is shown in Fig. 3b. Fig. 3a shows our method

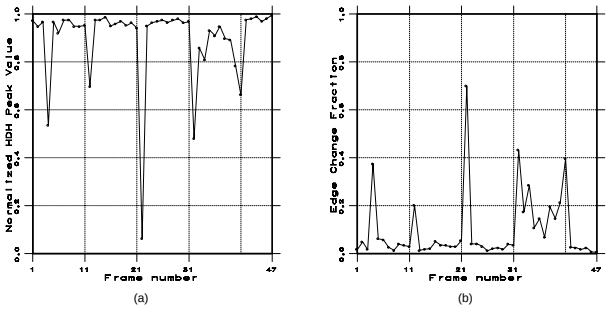

Figure 3: **Result comparison of two methods**

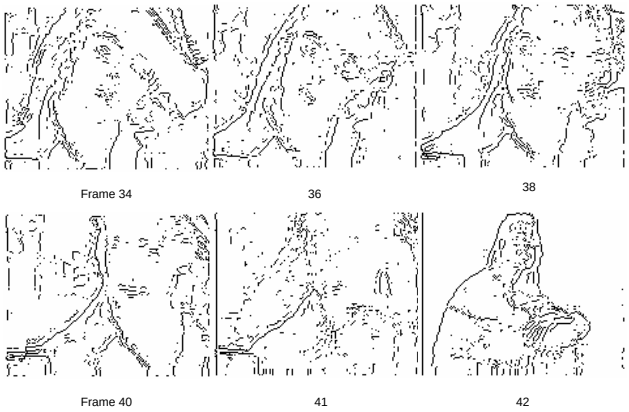

Figure 4: **Edge extracted from M2 sequence**

using the normalized peak value of HDH obtained from each frame of the same video.

Fig. 3 shows that 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. 4, there is a complicated camera shot. It contains object occlusion (a hand moves through part of the head) while the head is rotating (in 3D space), in the mean time, the camera is panning right. Also shown in the bottom row of Fig. 4 are the raw edge maps of frames 40 to 42 where an optical (cross-dissolve) cut occurs. In both cases, our method can provide better discrimination between true-cut and non-cut frames. As can be noticed from Fig. 3b, 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 38. Of course, these false detections can be avoided in this case by using a window technique [7]. However, the selection of the window size would pose a limitation on the system.

4. 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. By using the edge features, this method can effectively decrease the influence of rapid changes in scene brightness. By using the Hausdroff distance histogram on subregions and merging them up through multiple passes, this algorithm can robustly tolerate multiple object moving and camera motions. The scheme of performing the detection directly in the compressed domain leads to a significant computation 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.

5. 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] Bo Shen and Ishwar K. Sethi, "Convolution-based edge detection for image/video in block DCT domain," Journal of Visual Communications and Image Representation, vol. 7, no. 4, pp. 441-423, Dec. 1996.

[7] 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.

[8] 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.

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