An Evaluation of Alternative Techniques for Automatic Detection of Shot Boundaries in Digital Video

A.F. Smeaton[1(*)], J. Gilvarry, G. Gormley, B. Tobin S. Marlow and N. Murphy

School of Electronic Engineering and (*)School of Computer Applications
Dublin City University
Glasnevin, Dublin 9, IRELAND
asmeaton@compapp.dcu.ie

Abstract. The application of image processing techniques to achieve substantial compression in digital video is one of the reasons why computer-supported video processing and digital TV are now becoming commonplace. The encoding formats used for video, such as the MPEG family of standards, have been developed primarily to achieve high compression rates, but now that this has been achieved, effort is being concentrated on other, content-based activities. MPEG-7, for example is a standard intended to support such developments. In the work described here, we are developing and deploying techniques to support content-based navigation and browsing through digital video (broadcast TV) archives. Fundamental to this is being able to automatically structure video into shots and scenes. In this paper we report our progress on developing a variety of approaches to automatic shot boundary detection in MPEG-1 video, and their evaluation on a large test suite of 8 hours of broadcast TV. Our work to date indicates that different techniques work well for different shot transition types and that a combination of techniques may yield the most accurate segmentation.

1. Introduction

Video, whether digital or analog, consists of a series of individual frames or frames displayed at a constant rate — the effect of which is to give the illusion of motion — along with an associated audio track. The work described here is based on MPEG-1 video and audio encoding with a video frame rate of 25 frames per second.

To allow any kind of content-based navigation of video, the material has first to be broken up into constituent elements and structured. For video, these elements are *shots* and *scenes*. A shot is defined as the video resulting from a continuous recording by a single camera. A scene is made up of multiple shots, while a television broadcast of a program consists of a collection of scenes. For studio broadcasts (for example the news transmitted live), it is fairly easy to break the program up as the boundaries between shots are "hard". However, many television programs and most films use

[1] Author for correspondence

special post-production techniques to "soften" the boundaries, thus making them easier on the human eye, but more difficult to detect automatically.

There are four major types of boundaries between shots [1]:

- *A cut.* This is a "hard" boundary and occurs when there is a complete change of shot over a span of two consecutive frames. This is commonly used in live or in-studio transmissions.
- *A fade.* There are two types of fade, a fade-out and a fade-in. A fade-out occurs when the picture gradually fades to a dot or black screen, while a fade-in occurs when the picture is gradually displayed from a black screen. Both these effects occur over a few frames, e.g. 12 frames for a half-second fade-out.
- *A dissolve.* This is the simultaneous occurrence of a fade-out and a fade-in, the two frames being superimposed on each other over a fixed duration of, say, ½ second (12 frames). This can be used in live in-studio transmissions.
- *A wipe*. This effect is like a virtual line going across the screen clearing one picture as it brings in another, again occurring over a few frames. It was particularly common in early TV such as the *Batman* series, but it still used.

Each of these post-production and live techniques makes the automatic detection of shot boundaries in video a non-trivial task. Without the prevalence of these video artifacts, shot boundary detection would default to a more straightforward comparison between adjacent frames, but because these techniques can be used, and we don't know if they are being used or not, the problem is made more difficult.

A number of techniques have been tried in shot boundary detection with varying degrees of success. These include pixel differences between adjacent frames, comparison of adjacent frames using colour histograms, detection and comparison of macroblocks from MPEG encoding, and the detection and comparison of edges in adjacent frames. Each of these techniques is known to work well for different transition types, e.g. frame comparison based on colours works well on cuts, but not on fades or dissolves, while edge detection handles wipes and dissolves quite well. Techniques such as these are used in commercial video indexing and manipulation products such as MediaKey from ISLIP and Virage's indexing and retrieval browser [2]. However, apart from our own evaluation on a large video test suite and similar work at the University of California, Berkeley, nothing has been published which compares the techniques on large data sets. (See [10] for a possible exception).

In this paper we report our on-going evaluation studies of these shot boundary detection methods. In the following section we give a brief overview of MPEG encoding and the possibility it offers for shot boundary detection. Section 3 sketches our test suite of broadcast material which we use for evaluation purposes and in section 4 we report on our implementation of shot boundary detection methods based on comparing colour histograms, on detecting edges using motion vectors, and on using macroblock encoding information. Section 5 is an overview of our plans for combining different shot boundary detection methods into one unified technique and a brief look at related work, and a concluding section finishes the paper.

2. Digital Video, MPEG and Shot Boundaries

MPEG-1 (or ISO/IEC 11172 as it is formally known) was the first standard to be released by the Moving Picture Experts Group (MPEG). It supports the coding of video and audio for digital storage media at a bit rate of up to 1.5 Mbps. The quality of MPEG-1 is reckoned to be good enough to process, preview and analyse video content, but not for consumer viewing, the targeted application of MPEG-2.

MPEG-1 achieves it its high compression rate by the use of motion estimation and compensation between frames. It takes advantage of the fact that from frame to frame there is very little change in the picture, usually only small movements of objects, apart from changes in shot or scene. For this reason MPEG uses macroblocks, which are areas within a frame which can be compared across frames. If a macroblock is deemed to occur in two adjacent frames (either because it represents a stationary area which has not moved in the time between frames, or has moved in some direction by only a small amount as represented by a motion vector) then instead of encoding that entire macroblock in a following frame, the difference between the two macroblocks and their motion vector is encoded and transmitted.

There are three main types of frame in MPEG-1, namely I (Intra), P (Predicted) and B (Bi-directionally predicted), corresponding to 3 ways that a frame can be encoded.

I frames – are *intra-coded* frames: they are coded independently of the context of adjacent frames. This does not give very efficient compression, but it does mean that I-frames can be used as random access points in the video bitstream. This is important for stored video where searching through a sequence is common. I-frames can be viewed as starting points in a sequence: they act as reference frames for prediction of other non-I frames. MPEG-1 requires at least two of them in every second of video.

P frames – are forward-predicted frames, which take information from previous I or P frames. This is used along with the P-frame's own data to reconstruct the frame, using *motion compensation* (see below). As a result of this, P-frames offer better compression than I-frames. Like I-frames, P-frames can also be used as reference frames, but they are classed as *inter-coded* frames. Figure 1 shows an example of forward prediction: the first P-picture is taking information from the previous I-picture and the second P-picture is taking information from the first P-picture.

B frames – are bi-directionally predicted frames, in that they can take information from previous and/or future reference frames. B-frames themselves are not used as reference frames (i.e. not used as prediction for previous/future frames). B-frames offer even greater compression than P-frames. Bi-directional prediction for B-frames is shown in Figure 2. It can be seen that each of the four B-frames takes information from the closest I and P-frames. The data taken from I or P-frames is "motion compensated". A brief description of how MPEG-1 uses motion compensation is given below.

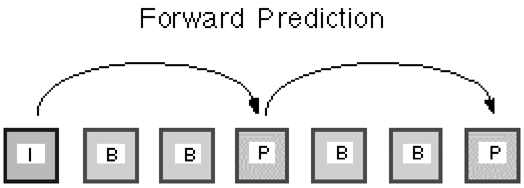


Fig. 1. A P-frame is encoded using information from a previous I or P frame.

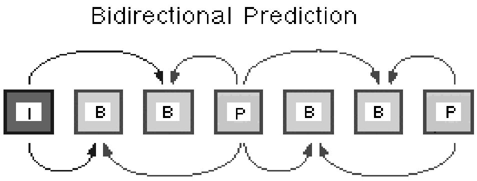


Fig. 2. A B-frame is encoded using information from both a previous and future I or P-Frame

Consider two frames, A and B as shown in Figure 3. Each frame is divided up into macroblocks. Macroblock "y" in B is the one we wish to encode, and macroblock "x" is its counterpart in the reference frame, A. A search is done around "x" in an area represented by the dotted box to find the best match for "y". This search is limited to a finite area, and even if there is a perfectly matching macroblock outside the search area, it will not be found. The displacement between the two macroblocks gives the *motion vector* associated with "y".

There are many search algorithms which can be used to find the best matching macroblock. A full search during encoding gives the best match but it is computationally expensive. Alternatives to this are the logarithmic search, one-at-a-time search, three-step search and the hierarchical search [3]. The choice of search is decided by the encoder, with the usual trade-off between time and accuracy.

Once video has been encoded, the data stream can be analysed in order to determine its logical structure and in particular to detect shot boundaries. In order to decide whether a shot boundary has occurred, it is necessary to set a threshold, or thresholds for comparison between adjacent frames. In our work we have found that as a result of the huge diversity of broadcast video, different threshold levels are required for different types of broadcast and in related work we are developing a dynamic or adaptive threshold that can change to suit each broadcast type.

Many different types of shot breaks appear in broadcast video. As mentioned already, to detect shot boundaries, we need to identify cuts, fades, dissolves and wipes.

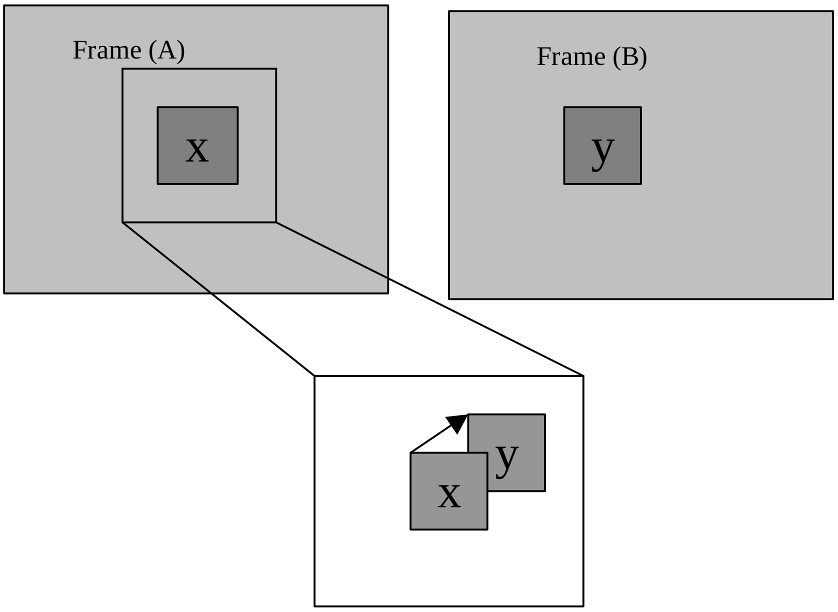


Fig. 3. A search is done around "x" to find the best match for "y".

Modern television productions make extensive use of effects such as:

- Computer generated effects: these can be commonly seen in advertisements where one shot is "morphed" into another, or at the beginning or end of soaps or dramas, where there is a constant background and images, objects or credits are faded or dissolved in and out.
- Split-screen techniques: these are commonly seen on the news where a ticker tape may be used to display the latest stock-exchange details, or in an interview where two or more screens appear in the TV picture simultaneously.
- Global camera motion: this is used in almost all types of production. It consists of the camera zooming, panning, tilting, booming, dollying or rotating. [2]

Initially we used a colour histogram based method for computing the similarity between adjacent frames to detecting shot boundaries. This method is good at detecting cuts, but needs dynamic thresholding to work on other effects. Like others, we have found that no single method investigated works well on detecting shot boundaries throughout a whole video sequence containing different video types. For this reason and because our target application is generic broadcast TV, we are looking at other methods and in this paper we present results of some implementation and evaluation. We hope to amalgamate a number of different methods for shot boundary detection, which when pooled together should be able to handle a wide range of video types.

3. The Evaluation Test Suite

Most of the published work on evaluating the effectiveness shot boundary detection does so on small video segments of the order of some minutes. In our work, to scale up to realistic quantities, we recorded 8 hours of broadcast TV (RTE1, 12 June 1998) in MPEG-1 format. We also developed a mark-up tool which allowed us to manually determine and log the boundaries between different shots, and to mark each type of shot boundary (fade, dissolve, cut), being used. This work yielded a ground truth against which our automatic shot boundary detection methods can be tested. More details of this test suite, including consistency checking on the manual mark-up, can be found at http://lorca.compapp.dcu.ie/Video/.

Some characteristics of our test suite are given in Table 1. The quantity of data involved means that as we develop and implement shot boundary detection methods and evaluate them on this suite, we are not "over-fitting" a technique to a particular data set. Our data set includes news programs, weather forecasts, soaps, cooking, magazine/chat show, a quiz show, a documentary, some comedy/drama and many, many commercials. Nevertheless, there are still some important omissions and recently it has been decided to include some sports programs and some music videos.

Table 1. Characteristics of Video Evaluation Test Suite

Size (Mbytes) (MPEG-1 encoding)	5.4 Gbytes
Duration (hh:mm:ss)	8:00:00
Number of frames	720,000
Resolution (size)	352 x 288 pixels (SIF)
Shot Boundary Types:	
Cuts	5380
Fades & Dissolves	779
Total:	6159
Motion vector range	-64 to +63.5 pixels

As we implement a shot boundary detection method, we compare the shot boundaries detected on this 8 hours of MPEG-1 to the ground truth. Effectiveness is measured in terms of recall and precision where recall is the proportion of the overall true shot bounds which have been detected while precision is the proportion of the detected shot bounds that are true shot boundaries.

4. Techniques for Shot Boundary Detection

In this section we describe the techniques for shot boundary detection that we have implemented and evaluated on our test suite.

4.1 shot boundary detection Based on Colour Histograms

The first way in which we measure shot bounds is to compute frame-to-frame similarities based on the colours which appear within them. While image retrieval based on computing image-image similarities can be accomplished by dividing a frame into blocks and comparing colours appearing in corresponding blocks [3], we use a simpler approach with a colour histogram for an entire image, irrespective of the relative positions of those colours in the frame. A disadvantage of this is that we are not able to detect high similarity between frames which have, say, a large red blob in the top right corner and a large red blob in the bottom right corner. However, in comparing the similarity between *adjacent* video frames, such a scenario is likely not to occur, except to represent very fast motion in the shot.

Once inter-frame similarities have been computed, a threshold can be used to indicate shot boundaries. A detailed description of our shot boundary detection based on colour histograms is given in [2]. As with much work in this area, we have found that a colour based similarity between frames does not have a single best threshold, as the following performance table indicates. A higher value for the threshold gives more missed cuts and fewer correctly identified, but fewer false positives. What this simple summary table hides is the fact that different thresholds are more appropriate for different broadcast types, varying from visually “noise” commercials and MTV-type broadcasts, to the visually “quiet” soaps and news programmes.

Table 2. Total figures for the entire test set of 720000 frames.

Threshold	Total # of shot boundaries	# correctly identified	# falsely identified	# missed	Recall	Precision
1 (0.010)	6159	5689	3775	470	92	60
2 (0.020)		5472	1504	687	89	78
3 (0.035)		5163	731	996	85	88
4 (0.060)		4508	431	1651	74	92
5 (0.15)		2789	195	3370	45	94

4.2 Edge Detection

The shot boundary detection technique which is the subject of this paper is based on detecting edges in two adjacent images and comparing them. By detecting the appearance of intensity edges in a frame that are far away from the intensity edges in the previous frame, it should be possible to detect and classify the four different types of shot breaks. The colour images are converted to gray-scale and the edges detected using the Sobel method.[5] Each edge pixel is then “dilated”, by surrounding it with a diamond shape of pixels to allow for movement which may occur between frames when one frame is being compared against the adjacent one. An undilated frame is represented by *E* and its adjacent frame by *E'*, dilated adjacent frames are represented by $\overline{E}$ and $\overline{E'}$. An edge pixel in the second frame which is outside the dilation range in the first frame is defined as an “entering” edge pixel. An edge pixel in the first frame that disappears far from an existing pixel in the second frame is defined as an “exiting” edge pixel. By calculating the sum of values of entering and exiting edge pixels, the edge change fraction can be calculated for every frame. Then, by analysing the results of the edge change fraction, it is possible to detect and classify cuts, fades and dissolves. Furthermore, by analysing the spatial distribution of entering and exiting pixels, wipes can be detected.

When performing the edge detection, the frames can be re-scaled to various resolutions. A set of three image resolutions, 352*288, 176*144 or 88*72 pixels, more commonly known as SIF, QSIF and 16CIF, were chosen for test purposes. The trade-off in using a higher resolution size against a lower one is the accuracy of results in the edge change fraction versus the increased execution time of the computation. The method we use to compute the edge change fraction, *p*, is based on that used in [4] to compare every pixel in the first undilated frame, *E*, against the corresponding pixels in the second dilated frame, $\overline{E'}$. There are two possibilities for this comparison:

- if a pixel is found in location *(x,y)* in frame E, and a matching pixel is found in its dilated area *(x+dx,y+dy)* in the second frame, $\overline{E'}$, then this implies that no change has occurred, so this is not an entering or exiting pixel;
- if a pixel is found in the first frame, E, and not in the second frame $\overline{E'}$, this implies that a pixel has exited from the first frame, E.

A repeat of this procedure is carried out, where the second undilated frame, E', is compared against the first dilated frame, $\overline{E}$, again with two scenarios:

- if a pixel is found in location *(x,y)* in frame E', and a matching pixel is found in its dilated area *(x+dx,y+dy)* in the first frame, $\overline{E}$, then this implies that no change has occurred, so this is not an entering or exiting pixel.
- if a pixel is found in the second frame, E', and not in the first frame $\overline{E}$, this implies that a pixel has entered the second frame, E'.

By accounting for both scenarios, P_{out}, which measures the fraction of exiting edge pixels is calculated for the first frame, and $P_{in,}$, the fraction of entering edge pixels is calculated for the second frame, as follows:

$$P_{out} = 1 - \frac{\sum_{x,y} E[x,y]\overline{E'}[x,y]}{\sum_{x,y} E[x,y]} \qquad (1)$$

$$P_{in} = 1 - \frac{\sum_{x,y} \overline{E}[x,y]E'[x,y]}{\sum_{x,y} E[x,y]} \qquad (2)$$

The edge change fraction, *p,* is the maximum value of P_{in} and P_{out} in each frame.

$$p = \max(P_{in}, P_{out}) \qquad \textbf{(3)}$$

Scene breaks can be determined by looking for peaks in *p*, the edge change fraction. However, a more effective and easier method for analysis is to calculate difference values from the array of edge change fraction values.

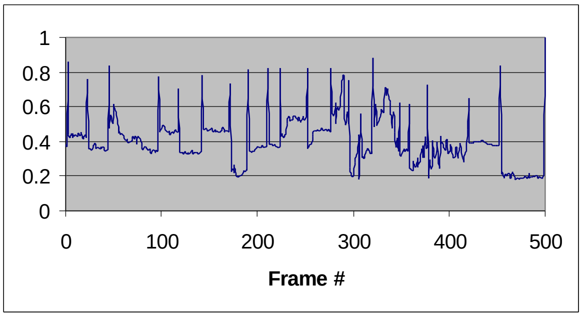
In dilating the images we used a choice of three different structure element (dilation) sizes, each with a similar dilation structure, 3*3, 7*7 and 13*13. Once again, the trade-off in using a higher structuring element is the accuracy of the edge change fraction over the execution time. A plot of the edge change fraction values over 500 frames is shown in figure 4(a). Figure 4(b) shows the difference values over the same 500 frames. The difference values are calculated by subtracting the value of the edge change fraction in one frame from its value in the previous frame. Each of the high peaks in both plots represent a cut. The Edge Detection Program can detect, miss or falsely detect shot changes. By using the difference values as opposed to the maximum edge change fraction values to detect shot changes, the number of falsely detected shot changes can be greatly reduced. This can be seen in the figure 4(a) where there is extra video content between frame 280 and 380, which leads to false cuts being detected. However, this extra content does not produce such high peaks in the difference plot. The maximum edge change fraction values hold quite clear characteristics for fades and dissolves, and so by analysing the data in figure 4(a), these types of shot breaks can be detected.

In figure 5 we show the edge change fraction over a 2000 frame segment of video. Frame 0-1073 is taken from the soap “Home & Away”, while frames 1073-2000 are taken from an advert for a PlayStation which is visually very active. The extra activity seen around frame 500 is due to camera panning, whereas the extra activity seen between frames 1600-1700 is a result of a combination of camera motion and object motion. Clearly, a set threshold would detect false cuts, also known as false positives, and miss other real cuts.

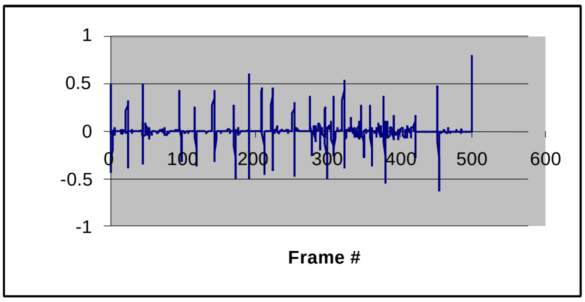
We now look at the results of this method of shot boundary detection in order to compare how well it works on different types of TV broadcast. The technique was tested over 2 hours and 40 minutes of the baseline video with a set resolution size of 88*72 pixels and a set structure element size of 7*7 pixels. With these parameters, it takes approximately 3 times real time running on a SUN Exterprise 3000 server. The results of testing are shown below in the Table 3.

The most common reasons for missing cuts using edge change are transitions from:

- a blurred image, where the intensity edges are not clearly defined,
- an image with similar background outline or intensity edges to the next image,
- a dark or very bright image as the intensity edges are not well defined,
- a cut from a blank screen to a dark image,
- a cut from an image, to an image in the same scene only from a different angle.



(a)



(b)

Fig. 4. (a) plot of max edge change fraction values over 500 frames, **(b)** plot of difference values over the same 500 frames

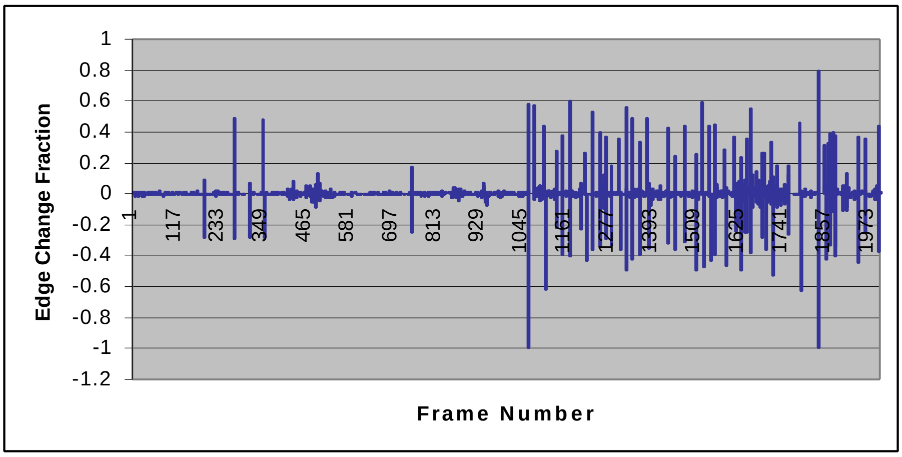


Fig. 5. Plot of difference values: Frame 0 - 1073 from a soap, Frame 1074 - 2000 from a high action content advert.

Table 3. results on 8 hours

	Total # of shot boundaries	# correctly identified	# falsely identified	# missed	Recall	Precision
Threshold 1 (0.010)	6159	5010	972	1149	81	84

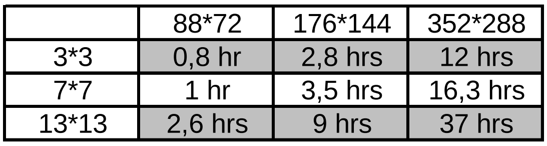
The detection of false cuts are caused by:

- fast moving or high action scenes, with greater edge differences between frames,
- a camera flash,
- close-up moving scenes,
- an object not involved in the scene, moving in front of the camera lens,
- a zoom in or out, camera pan, or other camera motion,
- animated scenes,
- computer generated scenes,
- interference from the broadcast or recording of the program,
- an object cut from an image.

Common problems across all video types are missed cuts during dark scenes and the detection of false cuts during the credits at the end of programs.

In the tests to date we used the lowest resolution size of 88*72 and a structure element size of 7*7. The edge detection is required to run in approximately real time when complete. By increasing the resolution from 88*72 to 176*144, the execution time of the program is increased by a factor of 3. If it is increased again to 352*288, the execution time is further increased by a factor of 4. This implies that the difference in execution time between a resolution of 88*72 and 352*288 is a factor of 12. If the structure element size is increased from 3*3 to 7*7, the execution time of the program is increased by a factor of 0.5. If it is increased again to 13*13, the execution time increases by a further factor of 2.5. This implies the difference in execution time between a structure element size of 3*3 and 13*13 is approximately a factor of 3. The table below shows the execution times for each test case. A large increase of 36 hours can be clearly seen in the execution time between the test case with parameters [3*3, 88*72], and that of [13*13, 352*288].

Table 4. Execution times for varying resolutions & structure element sizes



	88*72	176*144	352*288
3*3	0,8 hr	2,8 hrs	12 hrs
7*7	1 hr	3,5 hrs	16,3 hrs
13*13	2,6 hrs	9 hrs	37 hrs

Rather than do such runs on all combinations, we tried edge detection based shot boundary detection on a sample of 20 minutes (30,000 frames) and the results are shown in Table 5.

Table 5. Performance of edge detection on a sample 20 minute segment.

		88*72	176*144	352*288
3*3	Correct	184	192	201
	False	53	76	121
	Missed	110	109	105
7*7	Correct	173	189	202
	False	50	77	110
	Missed	120	112	106
13*13	Correct	154	180	206
	False	58	82	123
	Missed	142	120	99

The differences between these results are marginal in comparison to the difference in execution time for each set. Also, the number of false shot changes has increased dramatically as the resolution and structure element sizes increase. In analysing the results we found that quite a few false positives detected in all the tests were due to camera panning or zooming, i.e. camera motion. It is possible that MPEG motion vectors can be used to compensate for these movements. For example, if the camera is panning the motion vectors for each macro block will tend to point in the opposite direction and their magnitude will indicate the speed of the pan. The images could then be aligned before the process of edge detection is initiated. We have developed a technique to extract such vectors from an MPEG stream and plan to incorporate these vector values into the edge detection based technique.

A further refinement we are planning is to change from Sobel edge detection in order to define the edges more clearly. This would give a better recognition of background changes in the case of missed cuts, and less dilution of edges in the case of very bright and very dark scenes.[5]. There are several different filtering operations that could be tested, one of which is Canny filtering [6]. The difficulty is that its complexity would cause a large increase in execution time.

The edge change fraction, *p*, represents the intensity of edges in a particular frame. This value for *p* tends to remain relatively constant throughout an entire shot, giving each shot a characteristic value. It has been suggested that by matching shots which have the same characteristic value, it could be possible to detect similar shots or to group together different shots taken from the same logical scene [5]. The example in figure 6 is a sequence taken from a Play Station advert. The first *p* level corresponds to a scene with a man being interrogated at a table in a dark room surrounded by detectives. The clip then cuts to an outdoor action scene of a car blowing up, then switches back to the interrogation scene. This sequence can be seen in the plot, but as yet, has not been examined to see if it is reliable as a stand alone method in grouping together similar shots. [5]

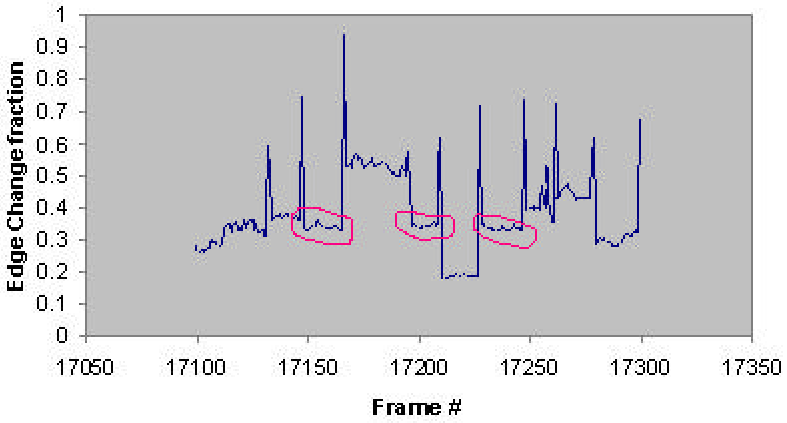


Fig. 6. 250 frames of Play station add. The circled areas represent the same logical scene

4.3 Shot Boundary Detection Using Macroblocks

Different MPEG picture types have different attributes corresponding to the type of macroblocks contained in them and it is these attributes that could hold the key to a third technique for detecting shot cuts. The different macroblock types use either forward prediction, backward prediction, or no prediction at all. If a frame type that normally contains backward predicted blocks doesn’t have any, then it could be suggested that the future frame has changed dramatically and this could indicate a shot cut. There are problems with this though: if there is a shot change and the frame in the new shot contains similar information to the old shot, then a lot of prediction blocks could be used and the macroblock information in the frame would look similar to any normal macroblock. At points like this in a sequence the detection of shot cuts is more complicated.

The classification of the different macroblock types happens at the encoder, based on the reliability of the motion estimation and encoding efficiency. There are three types of macroblock, and while they have the same names (I, P and B) as the frame types, each macroblock type doesn’t only occur in the corresponding frame type.

I-macroblocks – are *intra*-blocks and like I-frames are coded by themselves. I-frames are made up *only* of I-blocks because of the fact that they take no information from other frames. I-blocks can also occur in P and B frames. The occurrence of I-blocks in P and B-frames could be due to the fact that the motion prediction from a previous or future frame is not effective enough, and the whole block is then coded on its own.

P-macroblocks – are predicted-blocks, and can occur in both P and B-frames. Each P-block has a motion vector, which indicates the difference between the current block to be decoded and a similar block in a previous frame. P-blocks use forward prediction, which involves searching an area in a previous frame for a similar block. Figure 7 illustrates the motion compensated forward prediction for a P-block in a P or B-Frame with reference to an I-block in a previous I or P-Frame.

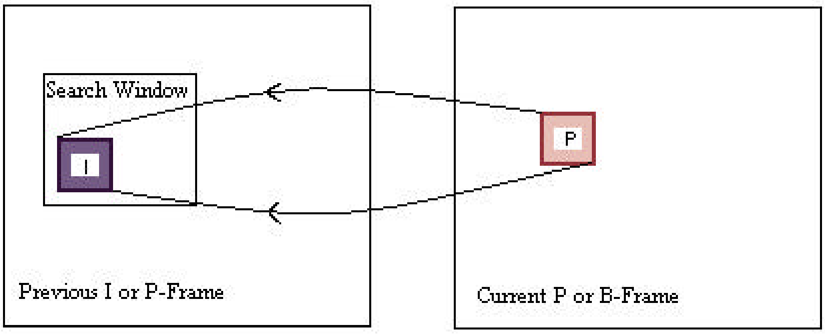


Fig. 7. Forward prediction for a P-macroblock

B-macroblocks – these are bi-directionally predicted blocks. They only occur in B-frames. They, along with P-blocks, are also known as *inter*-blocks. A B-block can have forward and backward prediction or just backward prediction. A B-picture will try to use all B-blocks, but for blocks that only use forward prediction, they code a P-block and for blocks where there is a lot of new information or the prediction is not sufficient and gives too much error, an I-block is coded.

Our work on macroblock-based techniques for shot boundary detection is ongoing.

5. Shot Boundary Detection Techniques

While the techniques we are using for evaluating shot boundary detection are not necessarily novel, our implementation of them is, and in particular our evaluation on such a large test suite has not been reported elsewhere in the literature to date. The exception is the work by Boreckzy and Rowe at University of California, Berkeley, who also use an 8-hour test collection but do not evaluate to the detail we do, nor have they evaluated different shot boundary detection techniques [7].

One of the most interesting possibilities for our work is the combination of different shot boundary detection techniques into one. In areas such as information retrieval it has been demonstrated consistently that the output or result from 2 or more independent techniques for retrieving information, can be combined using simple data fusion into a result which is more effective than any of the input components [8].

6. Conclusions

In this paper we have presented a variety of results on shot boundary detection as applied to a large video test suite. Our results are being incorporated into a demonstration system which will record and analyse broadcast TV in order to support user browsing and navigation. Fundamental to any kind of content-based access is the task of structuring video into shots and scenes which is why shot boundary detection, although a primitive operation, is so important.

We believe that our contribution to the field, evaluation and comparison of techniques on a large test suite may lead to more effective though costly shot boundary detection methods which can be used as a basis for other processing of video such as selecting representative frames for browsing or automatically abstracting a video into a trailer.

References

[1] H. J. Zhang et al, “Automatic Partitioning of Full Motion Video”, *Multimedia Systems,* Vol 1, pp 10-28, 1993.

[2] C. O'Toole et al. “Evaluation of Shot Boundary Detection on a Large Video Test Suite” in *Proceedings of Challenges in Image Retrieval*, Newcastle (UK), February 1999.

[3] Y. Gong et al. “Image Indexing and Retrieval Based on Colour Histograms”, *Multimedia Tools and Applications*, Vol 2, pp 133-156, 1996.

[4] R. Zabih et al. “A Feature Based Algorithm for detecting and Classifying Production Effects”, *Multimedia Systems,* Vol 7, pp119-128, 1999.

[5] A. Totterdell, “An Algorithm for Detecting and Classifying Scene Breaks in MPEG1 Video Bit Streams”, *Dublin City University*, available at http://lorca.compapp.dcu.ie/Video.

[6] J. Canny, “A computational approach to edge detection”, IEEE Transactions on pattern analysis and Machine Intelligence, 8(6):679-698,1986

[7] J S Boreczky and L A Rowe, “A Comparison of Video Shot Boundary Detection Techniques”, *Journal of Electronic Imaging*, 5(2) pp122-128, 1996.

[8] A.F. Smeaton, “Independence of Contributing Retrieval Strategies in Data Fusion for Effective Information Retrieval”, in Proceedings of the 20th BCS-IRSG Colloquium, Grenoble, France, Springer-Verlag Workshops in Computing, April 1998.

[9] W K Pratt, “Developing Visual Applications(XIL)”, *Prentice Hall*, Ca (USA), 1997.

[10] T Endo *et al*, “Mutual Spotting Retrieval between Speech and Video Image using Self-organized Network Databases”, in 1st International Conference on Advanced Multimedia Content Processing (AMCP'98) Osaka University Convention Center, Suita, Osaka, Japan, November 9-11, 1998.