Automatic Detection of Shot Boundaries in Digital Video

Yusseri Yusoff

Submitted for the Degree of
Doctor of Philosophy
from the
University of Surrey

UniS

Centre for Vision, Speech and Signal Processing
School of Electronics and Physical Sciences
University of Surrey
Guildford, Surrey GU2 7XH, U.K.

March 2002

© Yusseri Yusoff 2002
Abstract

This thesis describes the implementation of automatic shot boundary detection algorithms for the detection of cuts and gradual transitions in digital video sequences. The objective was to develop a fully automatic video segmentation system as a pre-processing step for video database retrieval management systems as well as other applications which have large video sequences as part of their systems.

For the detection of cuts, we begin by looking at a set of baseline algorithms that look into measuring specific features of video images and calculating the dissimilarity of the measures between frames in the video sequence. We then propose two different approaches and compare them against the set of baseline algorithms. These approaches are themselves built upon the base set of algorithms.

Observing that the baseline algorithms initially use hard thresholds to determine shot boundaries, we build Receiver Operating Characteristic (ROC) curves to plot the characteristics of the algorithms when varying the thresholds. In the first approach, we look into combining the multiple algorithms in such a way that as a collective, the detection of cuts are improved. The results of the fusion are then compared against the baseline algorithms on the ROC curve. For the second approach, we look into having adaptive thresholds for the baseline algorithms. A selection of adaptive thresholding methods were applied to the data set and compared with the baseline algorithms that are using hard thresholds.

In the case of gradual transition detection, an application of a filtering technique used to detect ramp edges in images is adapted for use in video sequences. The approach is taken by starting with the observation that shot boundaries represent edges in time, with cuts being sharp edges and gradual transitions closely approximating ramp edges.

The methods that we propose reflect our concentration on producing a reliable and efficient shot boundary detection mechanism. In each instance, be it for cuts or gradual transitions, we tested our algorithms on a comprehensive set of video sequences, containing a variety of content and obtained highly competitive results.

Key words: shot boundary detection, cuts, wipes, dissolves, gradual transitions, video database indexing and retrieval
Acknowledgements

I wish to express unending gratitude to Prof. Josef Kittler and Dr. William Christmas for their invaluable advice and unstinting support throughout the duration of this work. The vast experience of Josef and the quiet nudgings of Bill has been the greatest sources of energy for completing this thesis.

Almost the entire software developed for this research was done using the AMMA C++ Library, an extensive set of image and video software routines originally implemented by Radek Marik and further developed by the members of the AMMA team at CVSSP. This was after developing the software using pure C for the first year before bowing to the inevitable whence in Rome.

I thank my fellow students (Tony, Dan D., Dan B, Kieron, Ratna, Charles, Rups, the lot) for being around and sometimes for disrupting the peace. Finally, many thanks to my family for being family and most importantly to Noi.

The work presented in this thesis has been partially funded by the following entities: the Digital-VCE (http://www.digital-vce.com/) with partial support from EPSRC Grant GR/L61095, and the collaborative ASSAVID project, funded by the EPSRC Grant IST-13082.
5 Combining Multiple Experts

5.1 Introduction ........................................... 47
5.2 Related Work ........................................ 49
   5.2.1 Baseline Shot Detection Algorithms ......... 49
   5.2.2 Compound Shot Detection Algorithms ....... 53
5.3 Combining Multiple Experts (CME) ................. 55
5.4 Experimental Results ................................ 59
   5.4.1 Failure Analysis .................................. 63
5.5 Conclusions .......................................... 65

6 Adaptive Thresholding Methods ......................... 67

6.1 Introduction .......................................... 67
6.2 Previous work ....................................... 68
6.3 The experimental data and baseline shot cut detection algorithms .... 71
6.4 The adaptive thresholding scheme .................... 71
   6.4.1 The general method ............................. 72
   6.4.2 Computing the decision threshold .......... 73
6.5 Experimental Results ................................ 75
   6.5.1 Organization ..................................... 75
   6.5.2 Comparison with non-adaptive results ...... 76
   6.5.3 Effects of varying the window size .......... 76
   6.5.4 Determining the best performing adaptive thresholding method .... 77
   6.5.5 Other sequences ................................ 79
6.6 Conclusion .......................................... 80

7 Detection of Gradual Shot Transitions using Temporal Filtering .................. 85

7.1 Introduction .......................................... 85
7.2 The gradual shot transition detector ................. 86
   7.2.1 Canny's criteria ................................ 87
   7.2.2 The temporal filtering ....................... 88
   7.2.3 Absolute sum of pixel values as a measure .... 90
7.3 Implementation considerations ......................... 92
7.4 Experimental Results ......................................................... 95
  7.4.1 Experimental Data ......................................................... 95
  7.4.2 Effects of filter size ....................................................... 96
  7.4.3 Comparison between using luminance information only (LUM) and RGB information .................. 97
  7.4.4 Comparison between the TF method and the HC method ... 97
  7.4.5 Failure analysis ............................................................ 99
  7.4.6 Final remarks .............................................................. 100
7.5 Conclusion ............................................................................. 102

8 Summary and Concluding Remarks ........................................ 105
Chapter 1

Introduction

The increase in low-cost mass storage media, improved compression techniques and higher bandwidth has meant that the use of video in multimedia applications has seen rapid increase in recent years. The use of digital video is becoming more widespread in areas such as video conferencing, "video-on-demand", multimedia authoring systems, geographic information systems, monitoring systems, education, training and law enforcement, to name a few. For applications such as these, large video databases are created and stored. The ability to browse the stored video data or to retrieve the content of interest is an essential functional requirement of any video archiving system.

Recently, a considerable effort has been directed to finding ways to describe the information contained in video sequences. Methods for video annotation and for indexing into large video databases have been suggested to facilitate efficient search and query applications. The methods implemented are collectively described as content-based video retrieval.

It can be argued that current technology cannot effectively deal with the vast amount of data and information that video sequences carry. In order to better realise the issues raised in developing a video database retrieval system, one has to consider why traditional database retrieval techniques are insufficient for video retrieval.
Traditional database management systems are designed to promote controlled sharing of textual information [GM89]. Information was collected as discrete units (e.g., numeric and alphanumeric) and retrieved using mechanisms such as SQL (Structured Querying Language) strings of keywords which are either metadata or brief descriptions of images [SL96].

To construct a video database retrieval system, we have to understand the nature of video data, which is complex and structured in a different way to textual data. Much of the interpretation of visual data is dependent on the perception of the person viewing it. As such, there is a need to construct the representation of the visual data in such a way that the vagaries and varieties of human perception can be accommodated as much as is possible.

A number of systems have been proposed for the retrieval of multimedia data. In [AL96] these systems were broadly put under four categories: query by content, iconic query, SQL query and mixed queries. In query by content, the data is structured such that the queries are based on images, similarity retrieval or by the component features of the video such as the shape, colour, texture, or motion. With iconic query, normally after an initial query, a selection of "look alike" icons is presented to the user and the following queries follows from the user's selection of the appropriate icon(s). SQL queries follows the textual database management model where keywords are used as the basis for the queries usually assisted by Boolean operators (AND, OR) to specify the relationship between the keywords. Mixed queries use mixtures of the above methods, e.g., combining icons and keywords together.

The systems described above used different methods for indexing the video data. However, it is recognised that whichever method is used, it has to be flexible. Due to the complex nature of visual data and the impossibility of anticipating every possible kinds of querying that is desired by potential users, a retrieval system has to be able to present an approachable search and query mechanism. A flexible and approachable system would allow any user to get a positive result regardless of the fact that the system may not be tailored to the user's tastes.
To construct such a system, one must have an appreciation of the properties of video data. Digital video is by nature hierarchical. At the top is the whole video sequence which can be broken down into segments, then scenes, followed by shots and finally the individual frames. Once these components have been broken down, work can then be done to characterise the individual components for indexing and annotation.

A shot is a collection of frames from a single camera operation, a scene is a collection of shots which are related to each other, for e.g. a scene in a TV studio depicting a conversation amongst multiple participants in a chat show. The shots would change multiple times as the focus is shifted from one speaker to another, but the context of the scene is not changed, remaining in the studio. Further on, a segment describes a collection of scenes which together describe a plot point. An example of a segment is a collection of scenes in a news broadcast which together describe a segment of the news, e.g. home news, world news, business, sport, and weather.

As can be seen in Fig. 1.1, each level of the hierarchy is composed by components from its lower level. The level in which this study concentrates on is the shots. Specifically, we look into methods to achieve automatic detection of shot changes in a video sequence. In the next chapter, we present an overview of shot change de-
tection methods which we divide into cuts and gradual transitions. Following from that, we outline in more detail several shot change detection algorithms for detecting cuts. These algorithms will provide the baseline for chapters 5 and 6. Chapter 7 details the methodology and experimental work done in detecting gradual shot transitions. In the final chapter we present our conclusions and a discussion on the future direction of the research.

1.1 Applications in Shot Change Detection

Before we begin further exploration into shot boundary detection methods, in this section, we present two common uses of video shots, one for generating keyframes for facilitating the indexing and retrieval of video databases, which we had touched on a little earlier in this chapter, and the other, as a tool in assisting the segmentation of speech and audio tracks in video sequences.

1.1.1 Video Database Retrieval Systems

The more popular, and known, application for video shot detection is as a keyframe generator for video database indexing and retrieval systems. As discussed earlier in the chapter, video indexing is dissimilar to text indexing in that exact indexing and matching is extremely difficult. A more appropriate technique for searching and retrieval of video information is by using similarity matching, given a template (example) of what is required. This approach is known as query-by-example (QBE). It describes a querying method in the form of sketches, structural descriptions, colour or texture, which offers more flexibility of traditional text-based querying methods.

Video database indexing and retrieval extends the image database indexing and retrieval systems by applying the techniques used for image databases—collectively referred to as Content Based Image Retrieval—to that of video. Early image database management systems used various strategies from textual encoding and logical records to utilising relational databases [ST97, GM92]. Newer approaches
adds the flexibility mentioned above in their strategies for storage and retrieval.
For example, IBM’s QBIC system [FSN+95] approached the indexing of images by translating visual information into numerical descriptors which are then stored in a database. It can index and retrieve images based on average colour, histogram colour, texture, shape and sketches. There are various other approaches and strategies for the storage and retrieval of images and video which can be seen in [FwSZ95, Nar95, GR95].

Focusing in video databases, it is apparent that there is a need to reduce the amount of data in which to index in order to keep the searching and retrieval processes at a manageable level. As such, for the generation of metadata for use in indexing, instead of each frame in a video sequence being analysed and indexed, only a significant subset is used. This subset of the video sequence is referred to as the keyframes. The keyframes usually consists of individual shots within the video sequence and has been the focus of our research. In [LOCK+00] we show the integration of our shot boundary detection work described in the following chapters with that of an object-based video retrieval system (OVID). The OVID system integrates a keyframe browser (Fig. 1.2) which display the keyframes extracted for a particular sequence in a sequential fashion (Storyboard-like).

The OVID system implemented uses a query-by-example and a query-by-region techniques to query the video database. The user can query using a template keyframe of the colour and texture in regions of the template. Fig. 1.3 shows an example where the user queries by colour and texture on three different regions.

Once the user completes his selection, the search mechanism commences. The system currently enables the experimentation of a neural network based searching mechanism, which is trained using the query areas so that a decision boundary between the query regions and what we call a “world dataset” is evaluated. The world dataset is a random sample of points in the databases, and plays the role of “anti-query”. The calculations of feature vectors for the query regions and the training of the neural network are performed on-the-fly. As a result, less metadata needs to be stored. When several query areas are selected for a query, the search is
Chapter 1. Introduction

Figure 1.2: Keyframe Browser for OVID

Figure 1.3: Query by colour and texture on three different regions
1.1. Applications in Shot Change Detection

sequential: the system first searches using the first presented area; it then searches using the second presented area but only among the results of the first search, and so on. This process emulates an AND search. The images retrieved are the images with the highest number of matching pixels. Once the query is completed, the user is presented with the images retrieved as shown in Fig. 1.4.

![Figure 1.4: Returns from the query shown in Fig. 1.3](image)

1.1.2 Video Assisted Segmentation of Speech and Audio Track

In parallel to research in video shot segmentation, there are work currently being carried out in the segmentation of audio tracks in video sequences. In [Wol96], the author gave an overview of approaches to content-based classification, search and retrieval of audio. He used loudness, pitch, brightness, bandwidth and harmonicity as sound features. In this feature space, he adopted a Euclidean metric and a likelihood function as two distinct techniques for classification of sounds.

The Informedia Digital Video Library Project [Inf96] at Carnegie Mellon University is creating a digital library of text, images, video and audio data available for full content retrieval. Through the integration of technologies from the fields of natural language understanding, image processing, speech recognition and video compression, the Informedia System allows a user to explore multimedia data in depth as well as in breadth. The work is centred in processing news stories from TV
broadcasts. The same methods can be used to index and search other streamed multimedia data by content. Gelin [GW96] describes a method of indexing the video soundtrack using keyword spotting. The word spotter achieves indexing on open vocabularies uttered by any speaker. In [SL97], the authors describe the audio as a support to scene change detection. A scene change detection can be performed using shot cut detection together with other modules which exploit audio/video semantic correlation among shots. A split-and-merge procedure on both video and audio signals is performed so as to identify each scene. Depending on the video, a scene change may occur jointly with an audio silence segment. Therefore, information on silent audio segments can be used to make the shot cut detection more robust, especially for news and advertisements.

Some research work had been done on audio segmentation. In [WCKB94], the authors proposed an audio segmentation technique based on HMM and clustering. In [ZK99] an approach was proposed whereby the segmentation of the audio track is carried out “on-the-fly”. In this, the feature vectors of energy, zero-crossing and fundamental frequency were used. Then for each feature curve, two sliding windows are used to compute average amplitude in each window. Whenever there is a big difference between them, an abrupt change is claimed to be detected. However, this approach may drift from one speaker to another without correctly detecting speaker “changes”. Other techniques have been suggested which are based on HMM and are very sensitive to audio content and in this sense powerful. However, they require an unacceptably long segment of a single source audio for training [KW96]. Recently, Delacourt, et. al [DW99] also proposed a speaker-based segmentation for audio data indexing using second order statistics. The technique basically implements an “on-the-fly” approach, but relies on the assumption that people do not speak simultaneously, i.e. only one speaker at a time.

The present work also addresses the problem of automatic segmentation of the audio track of digital multimedia material with application to multimedia content annotation, and retrieval by content. The audio track includes speech, music and even multiple speakers talking simultaneously. In [PKYC99, PYKC99], a different approach from those cited earlier is taken, in the sense that we wish to make the
1.1. Applications in Shot Change Detection

segmentation process fully automatic and realistic. The first requirement implies that the classes of audio content and their number is not known \textit{a priori}. Moreover, their models have to be learnt without supervision. The second requirement imposes a very strict limit on the amount of audio data available for the acquisition of new audio content models. The results obtained from basing the segmentation process on audio content only proved unsatisfactory. Therefore the possibility of applying contextual post-processing of the audio segmentation result by correlating it with the video shot segmentation and clustering was implemented. In the video segmentation process, the histogram comparison method to be described in Chapter 4 was implemented. Once we have completed the shot boundary detection, the shots are clustered together. Since the correct number of end clusters is not known, a hierarchical clustering method [NS93] was employed. We have shown in [PKYC99, PYKC99] that the result of automatic audio content segmentation can be improved dramatically by correlating it with video segmentation and clustering.
Chapter 2

Overview of Shot Change Detection Methods

2.1 The Video Shot

In [DSP91], the authors described the shot as the fundamental film component. Picard [Pic95] described the shot as an unbroken sequence of frames from one camera, e.g. a zoom of a person talking. The partitioning of video sequences into shots is considered an integral part of indexing and annotating video. Once the video is partitioned, further higher-level processing can be done to characterise it.

In the previous chapter, we touched on the hierarchical nature of video sequences (Fig. 1.1). Apart from the individual frames which make up the video sequence, the shot is the lowest denominator within the hierarchical structure. The moment of change from one shot to another – the shot boundary – can be done in several ways. The simplest of these is the camera cut. Figure 2.1 show an example of this. Frames (a) and (b) belong to the same shot and frames (c) and (d) to another. There are significant changes in the contents of the frames between (b) and (c).

There are other effects used for the demarkation of a shot boundary which are collectively referred to as gradual transitions. Whereas in the cut, the change takes place between two frames, gradual transitions takes place over a number of frames.
The plethora of editing effects available to the video editor today leads to various ingenious and creative methods by which this is achieved. Generally though, they can be categorised as dissolves, fades and wipes. Fig. 2.2 and Fig. 2.3 gives two examples of dissolves which are visually different.

The lengths of the gradual transitions are not set and can vary greatly even within the same video sequence, as can be seen in the two examples where one is 15 frames long and the other 6.

Another approach to classifying shot transitions is detailed in [HJW95], where the authors classified shot transitions into four classes based on the 2D image transformations applied during transition production:

1. **Identity class**: Neither of the two shots involved are modified, and no additional edit frames are added. Only hard cuts qualify for this class.

2. **Spatial Class**: Some spatial transformations are applied to the two shots involved. Examples are wipe, page turn, slide, and iris effects.

3. **Chromatic Class**: Some color space transformations are applied to the two shots involved. Examples are fade and dissolve effects.

4. **Spatio-Chromatic Class**: Some spatial as well as some color space transformations are applied to the two shots involved. All morphing effects fall into this category. Note that in practice often all effects in the spatial class in principle fall into the spatio-chromatic class since some chromatic transformations are
2.1. The Video Shot

Figure 2.2: A dissolve sequence that spans across 15 frames

Figure 2.3: Example of a zoom out
always applied at the boundary between the pixels of the first and second shot such as anti-aliasing, smoothing or shading operations.

For our purposes we have grouped the Spatial, Chromatic and Spatio-Chromatic classes together as gradual transitions and the Identity class as cuts.

2.2 Shot Change Detection Methods

2.2.1 Cuts

There has been much research done into shot cut detection, and as such, pointers are not hard to come by. In general, cut detection methods can be split into two groups - spatial domain methods and frequency domain methods. In the spatial domain, there are pixel, block and frame level comparisons and histogram comparison methods. In the frequency domain, there are methods relying on transform coefficients as well as sub-band feature comparison methods.

Video shot detection has been described as a temporal segmentation [Pic95]. Therefore, the aim of any shot change detector is to identify some property of the video, such that any frames that are within the same shot would have the same contiguous properties, and frames that don't belong to the same shot would have feature characteristics that are not contiguous.

Spatial domain methods in general uses a difference metric to distinguish one shot from another. This could be viewed as analogous to an edge detection in time. The simplest of these methods is the pair-wise pixel comparison [ZKS93, HJW94] where each pixel in the frame is compared to the corresponding pixel in the following frame. In the case of greylevel images, we take the difference in the intensity values of the two pixels and decide that the pixel has changed if the difference exceeds a certain threshold, $T_p$. The equation for this calculation as given in [ZKS93, AL96] is as follows:
2.2. Shot Change Detection Methods

\[ D(P_i(k, l)) = \begin{cases} 
1 & \text{if } |P_i(k, l) - P_{i+1}(k, l)| > T_p \\
0 & \text{otherwise}
\end{cases} \]  

(2.1)

where \( P_i(k, l) \) denotes the intensity value of the pixel at position \((k, l)\) in frame \(i\). To construct the algorithm, a summation of the pixel contributions calculated as in Eq. 2.1 across the entire frame of dimensions \(M \times N\) is carried out. If this summation exceeds a prespecified threshold, \(T\), the current frame is deemed to represent a shot cut, i.e.:

\[
\frac{\sum_{k=0}^{M-1} \sum_{l=0}^{N-1} D(P_i(k, l))}{M \times N} \times 100 \begin{cases} 
> T & \text{shot cut} \\
\leq T & \text{contiguous shot}
\end{cases}
\]  

(2.2)

The intuitive impression that one would get from this algorithm is that it would be very sensitive to motion. Any translation of the objects in the frames would create high difference values for many pixel-pairs. Camera motion such as zooming and panning would also create this effect. For this reason, Zhang et al. suggest a more robust method for shot change detection by dividing the frames into regions and comparing them with the corresponding regions in the next frame. The likelihood ratio [KJ91] was used as the metric for the comparisons (see Section 4.1). By using regions rather than individual pixels, the level of tolerance to motion is reduced. An added bonus for using regions is that the shape and composition of the regions under consideration could also be constructed. The most basic way of partitioning a frame is to divide it into \(m \times m\) blocks across the entire frame. However, with appropriate spatial segmentation, regions in the shape of the objects in the frame could also be used. This could increase the accuracy of the likelihood ratio algorithm.

The average intensity measurement [HJW94] seeks to further reduce the sensitivity to object and camera motion. In this method, instead of considering individual pixels or blocks of pixels, the whole frame is taken into consideration. Implementation of the algorithm is simply taking an average of the intensity values for each component (YUV, RGB, etc.) in the frame and comparing them with successive frames.
A further refinement to the approach is the histogram comparison. An intensity histogram of an image describes the *distribution* of the intensities within the image. For frames belonging to the same shot, one can assume that the intensities would be very similar even in the presence of object or camera motion. Unchanging background and unchanging objects (irrespective of where they move to across the span of the frames) would still give a fairly similar intensity histogram. The histograms in Fig. 2.4 demonstrates this. The histograms are a concatenation of the Y, U, and V components of two successive frames belonging to the same shot in a video sequence.

This histogram comparison method is described in the paper by Nagasaka and Tanaka [NT92]. They also suggest a more robust approach by using the colour code values derived from the three primary colours: red, green and blue (RGB codes). The colour codes are 24 bit code words, so creating a histogram from this would need $2^{24}$ bins. Creating histograms of that size across thousands of frames might prove to be infeasible as well as introducing additional noise by having a bin for
2.2. Shot Change Detection Methods

each colour code. Consequently, a method was implemented where only the top two bits of each colour were used to compose the colour code. This resulted in a 6 bit code word producing $2^6$ bins. To further improve the cut detection, it was suggested that instead of using the absolute difference between histograms, the rate of change of the histogram difference is used [OT93, UMY91].

As mentioned, the histogram comparison method seeks, as much as possible, to reduce sensitivity to camera and object motion. Conversely, there are methods that would take motion into account in applying a shot detection algorithm. Shahraray [Sha95] noted that in algorithms that employ partitioning of frames into regions or blocks, as in e.g. the likelihood ratio method, the comparisons are done on blocks in the same locations between the frames. It is suggested that a more accurate measure can be obtained by first performing a block-matching algorithm such that, for each of the selected blocks in the first image, the "best" fitting block in the second image is used as the corresponding block to be compared with. The use of motion estimation methods is also convenient when working with compressed video, using for example the MPEG [Gal91] compression standard, since motion estimation is an integral component of the system.

Compression of digital still and video images has been a topic of interest to many researchers due to the savings compression accords when it comes to storage. Even with the decrease in the cost of mass storage it still makes sense to first compress video sequences before adding them to a database. In view of this, there has been some research done on the processing of compressed data in large video databases.

The most prevalent compression techniques use the Discrete Cosine Transform (DCT) [RY90]. Amongst the DCT-based compression standards are JPEG [Wal91], MPEG [Gal91], and H.261 [Lio91]. Deardoff et al [DLM+94] used the size of the frame when it is compressed using the JPEG standard to detect shot changes. This is based on the premise that the complexity of the frames within the same shot is fairly similar and that frames of different shots are not.

In most compression applications, the DCT is applied on blocks of pixels within a frame. The DC coefficient of the DCT is directly proportional to the average value
of the pixels within the block in which the DCT is applied. Vellaikal and Kuo, in their work on indexing images [VK96], pointed out that a Euclidean distance measure can be used to calculate the similarities between two images by comparing the mean of the DC values for all the blocks in the frame.

In [AHC93b, AHC93a], the authors performed shot change detection using DCT coefficients in a subset of the blocks in MPEG or JPEG encoded video sequences. Processing on video sequences already encoded in the compression process means that the computational costs involved in decompressing the sequence before applying a shot change algorithm can be eliminated. This method works by choosing a set of $n$-connected regions that are constant throughout the sequence. A selection of the DCT coefficients computed for the blocks contained in the region is used for the calculation. The method gives the flexibility to modify the size and location of the regions and number of DCT coefficients, to balance speed and computational cost.

Another method which only uses a subset of the DCT coefficients is that proposed by Ariki and Saito [AS96]. Here a clustering method was suggested where similar consecutive frames are gathered together. The method seeks to eliminate any sensitivity to abrupt intensity changes such as camera flashes. The difference between this method and most other methods is the way in which a shot change is determined. The clustering method allows for a certain number of consecutive frames that are relatively dissimilar before flagging a shot change. In this manner, in the case of camera flashes, since a camera flash usually takes only one or two frames, false detection of a shot change can be eliminated.

In [FLM96], the authors contend that a shot cut will result in a significant variation in both the DC and AC information in the I-frames of an MPEG sequence. This is due to the different bitrates required to encode each macroblock in the frame. The metric used to decide if a couple of I-frames belong to different shots is called the \textit{normalised bit-rate difference}. Boccignone et al [BSP00] noted that the method used in [FLM96] underestimated the role of the DC component of the picture, accounting mainly for residual information (details), thus neglecting the average background
2.2. Shot Change Detection Methods

information. To which end, they proposed an improvement to the normalised bit-rate method which corrected for this. As an extension to analysing only I-frames, the authors also implemented difference algorithms for analysing the P- and B-frames before using a step-wise refinement strategy to detect a shot cut.

2.2.2 Gradual Transitions

The detection of gradual transitions present a different set of problems to that of detecting cuts, and remains a vexed subject in shot boundary detection. A reason for the difficulty in detecting gradual transitions is due to the variety of transition effects. With this in mind, there has been some research done in detecting certain types of gradual transitions only, e.g. concentrating on wipes [NPC99, KPK99, YW98] or dissolves [Ala93, YL95, HZ99] exclusively.

A particular reason for restricting the research into particular kinds of gradual transitions is that it allows the work to be reduced to a single problem, similar to that of detecting shot boundaries delineated by cuts. However, in reducing the complexity, the detection of other subsets of gradual transitions were not tackled.

In a paper on performance characterisation of shot change detection methods [GKS00], the authors noted that gradual transition detection performance of the algorithms they evaluated tend to be poor due to the fact that these algorithms expect some sort of ideal curve; a plateau for a dissolve, a sharp peak for a cut. However, the actual frame differences are noisy and do not often follow this ideal pattern. They also pointed out that localisation of the gradual transitions' beginning and end points are very poor.

One of the earliest gradual transition detection method proposed was that by Zhang, et al. in [ZKS93] where a histogram comparison method was applied with twin thresholds. If the histogram difference was between the two thresholds then it was tentatively marked as a gradual transition. Succeeding frames were compared against the first marked frame and if the running difference exceeded the larger threshold, a gradual transition was marked as detected.
In [NPC99], the authors presented a method for detecting wipes by constructing 2-D images from horizontal and vertical slices of the frames. They detected cuts and wipes by analysing the resulting reconstructed slices. This work is mirrored in [KPKS99] where the authors also constructed slices for analysis but preferring to use slices cut diagonally across the frame. Another wipe transition method proposed a multi-resolution approach using wavelet transformation for the decomposition [YW98].

In [HZ99], the authors computed dissimilarity measures based on motion compensation using block matching. Measurements for cut detection was carried out by comparing consecutive frames and for gradual transitions by skipping frames. The authors also incorporated studies which involve statistical measurements of shot lengths for motion pictures to construct an *a priori* probability for a shot boundary to happen after a certain shot length. Cut and gradual transition detection was then carried out using a combination of an adaptive thresholding method first proposed in [YL95] on the dissimilarity measures and the *a priori* probability model.

Gradual transitions like dissolves and fades can have a characteristic trace depending on the algorithm applied. This is one of the assumptions made in the method described in [HZ99] above and in [YL95] where a rising edge followed by a plateau and a falling edge in the dissimilarity measure trace is tracked.

An early approach taken in detecting dissolves was based on the observation that the frame-based intensity variance curve of an ideal dissolve has a parabolic shape [Ala93]. Essentially, this means that the first order derivative before and after a dissolve is zero and a positive constant during a dissolve. By approximating the second derivative by means of the second-order difference, two large negative spikes are introduced at the boundaries of the dissolves. In [Ala93], the author used these negative spikes to detect dissolve candidates. In [TDV00a, TDV00b], the authors noted that in their observation, the negative spikes are not easily obvious during a dissolve, due to noise and motion. Therefore, they proposed to exploit the facts that

1. the first-order derivative of the variance curve changes linearly from a nega-
2.2. Shot Change Detection Methods

tive value to a positive value,

2. the intensity variances of the two bordering shots should be larger than a minimal threshold,

3. and the actual dissolve duration falls between two well defined thresholds.

In order to minimise the effect of noise and motion, the curves are smoothed. Since the first-order derivative of the variance curve changes from a negative to a positive value, they proposed to trigger the detection of dissolves at the zero-crossing incidents.

A different approach was taken in [ZMM95, ZMM99] whereby they calculated the edge change ratio of two frames. This consisted of doing a spatial edge filtering, after which they calculated the number of entering and exiting edge pixels in the frames. The edge change ratio is then defined as [Lie99]:

\[ E_n = \max\left(\frac{X_{n}^{\text{in}}}{\sigma_n}, \frac{X_{n-1}^{\text{out}}}{\sigma_{n-1}}\right) \] (2.3)

where \( \sigma_n \) is the number of edge pixels in frame \( n \), \( X_{n}^{\text{in}} \) and \( X_{n-1}^{\text{out}} \) are the number of entering and exiting edge pixels in frames \( n \) and \( n - 1 \), respectively.

A recently published study [Han02] investigated the shot boundary detection problem in detail and identified major issues that needs to be considered. Then, the author presented a statistical shot boundary detector based on the minimisation of the average detection error probability. The statistical functions were modeled using a robust metric for visual content discontinuities using motion compensation and took into account all \textit{a priori} knowledge that he found relevant to shot boundary detection (based on the study of the major issues noted above). The shot boundary detector was evaluated against two criteria, which are

1. excellent detection for all types of shot boundaries, and

2. constant quality of the detection performance for any arbitrary sequence.
Chapter 2. Overview of Shot Change Detection Methods

The statistical detector makes a decision on a shot boundary after taking into account the input from the various components within the detector which are evaluated and combined. As mentioned, one of the criteria in constructing the shot boundary detector is excellent detection for all types of shot boundaries. So this represents an attempt at a unified shot boundary detector that detects both cuts and gradual transitions. At the moment, the author has tested the detector on cuts and dissolves only over a wide range of video sequences and exhibited very good results.

Interestingly, in the comparison studies that have been published [BR96, Lie99], which included all the above methods we discussed apart from the statistical shot boundary detector proposed in [Han02], the results of the gradual transition algorithms the authors reviewed had not been encouraging. For example, in [BR96], the author noted that in their tests, the tested algorithms all did a poor job in detecting gradual transitions. In [Lie99], the author produced some measurement results and again, the tested gradual transition algorithms did poorly - in one case for the edge change ratio, for a true positive detection rate of 66.67%, the false positive rate recorded 37100% higher than the number of actual transitions in the test data. With this in mind, we concentrated on developing a method that would produce a fairly effective gradual transition detector whilst not compromising the precision too much.
Chapter 3

The Data Set

In order to conduct our experiments, we constructed our data set from a variety of sources. The object was to have as much variety as possible such that we could test our work satisfactorily.

Our initial test data was constructed synthetically from a collection of single shot video sequences. This was helpful in the beginning during the evaluation of our baseline shot cut detection algorithms (Chapter 4). But there was an obvious need for greater variety for further experimental purposes. To this end, we started out to construct a larger database of video sequences, taking in data from terrestrial and satellite broadcast television. We used two different kinds of video grabbing hardware for the data collection, the first being based on SGI hardware and the second, PC hardware.

All our video sequences uses the YUV 4:2:0 QCIF format. We choose this format since it is the format used by a large number of researchers working on video processing. Using the SGI's direct to disk capabilities, we grabbed raw composite video from our broadcast source and save the results as an RGB file in PAL format. By applying appropriate filters – which are similar to those used for MPEG codecs – we then downsample and convert the incoming video source to YUV 4:2:0 QCIF.

With the PC, we initially record our video data into DV tapes and then downsample
Chapter 3. The Data Set

and convert the source directly onto YUV 4:2:0 QCIF.

Our data set consists of several off air sequences consisting of news programs, documentaries, children's shows, daytime soaps, etc. Basically we aimed at capturing as much variety as possible. We also have two other sequences. One is a collection of various cartoons and the other, a sequence featuring a rugby league match. These two sequences represent two different extremes in terms of content. Table 3.1 details the composition of our test data.

<table>
<thead>
<tr>
<th>Name</th>
<th>No. of frames</th>
<th>Time (mins)</th>
<th>No. of shot cuts</th>
</tr>
</thead>
<tbody>
<tr>
<td>GENERAL</td>
<td>161928</td>
<td>108</td>
<td>1160</td>
</tr>
<tr>
<td>CARTOON</td>
<td>41750</td>
<td>27.8</td>
<td>256</td>
</tr>
<tr>
<td>RUGBY</td>
<td>40490</td>
<td>27</td>
<td>257</td>
</tr>
</tbody>
</table>

The GENERAL sequence consists of four separate broadcast sequences which are shown in Tab. 3.2.

<table>
<thead>
<tr>
<th>Name</th>
<th>No. of frames</th>
<th>Time (mins)</th>
<th>No. of shot cuts</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHF</td>
<td>36007</td>
<td>24</td>
<td>199</td>
</tr>
<tr>
<td>DV</td>
<td>36000</td>
<td>24</td>
<td>268</td>
</tr>
<tr>
<td>SKY</td>
<td>45000</td>
<td>30</td>
<td>289</td>
</tr>
<tr>
<td>SUPERMAN</td>
<td>44921</td>
<td>29.9</td>
<td>404</td>
</tr>
</tbody>
</table>

Once the data was collected, we proceeded to manually segment the sequences to
generate our ground truth for evaluating the performance of our algorithms. The process of classifying the video content presented an interesting situation.

At the most fundamental level, there are two classes in a video sequence, which we labelled as $S$ (shot boundary) and $N$ (not a shot boundary). The $S$ class is divided into two, cuts, $S_c$ and gradual transitions, $S_g$. Further, $S_g$ could be broadly subdivided into wipes, fades and dissolves. Into the $N$ class, all frames that are within a shot should be grouped together.

False positives are what happens when one of the $N$ frames are misclassified as a shot boundary. If this rule is strictly enforced, then we found that one of the most common false positive occurrences are those due to sudden changes in intensity, such as lightning strikes or camera flashes. However, we could further sub-divide the $N$ class to provide for features such as flashes, $N_f$ and localised changes within a shot, $N_l$, which are typically exemplified by captions. This represents a more relaxed rule set by which to classify our video sequences.

In the proceeding chapters, we have adopted this relaxed classification strategy for our ground truth and experimental results. We do not show results of our algorithms in detecting features other than shot boundaries since these features are typically less common and are not what we aim to detect.
Chapter 4

Baseline Shot Cut Detection Algorithms

After evaluating several algorithms that were described in the Chapter 2, a selection of these were implemented. The following sections discuss the formulation of the algorithms in more detail as well as any modifications that were made.

At the end of the chapter, we present some preliminary experimental results of using these algorithms which will serve as the baseline with which we compare our own methods and improvements against in later chapters.

4.1 Likelihood Ratio (LH)

As mentioned earlier in the overview (Chapter 2), the simplest shot cut detection method to implement is the pair-wise comparison of the pixel intensities. However, it has been shown to be highly inefficient in the presence of camera or object motion. In order to overcome this, it was proposed that instead of comparing individual pixels, regions (or blocks) of pixels were used. Zhang et al [ZKS93] proposed using a likelihood ratio [KJ91] for the metric to compare the corresponding regions. The comparison of the regions in two successive frames is based on the second-order statistical characteristics of their intensity values.
Chapter 4. Baseline Shot Cut Detection Algorithms

Let $\mu_i$ and $\mu_{i+1}$ denote the mean intensity values for a given block in two successive frames and let $\sigma_i$ and $\sigma_{i+1}$ denote the corresponding variances. The likelihood ratio, $lh_b$ for the block $b$ can then be calculated by the following formula:

$$lh_b = \frac{\left(\frac{\sigma_i + \sigma_{i+1}}{2} + \left(\frac{\mu_i - \mu_{i+1}}{2}\right)^2\right)}{\sigma_i \times \sigma_{i+1}}$$

(4.1)

In the implementation that was carried out for our experiments, the luminance component of the frames were divided into $16 \times 16$ blocks and compared using the equation above (Eq. 4.1). The results were then summed up and averaged over the total number of blocks in the frame to create the likelihood ratio for the whole frame:

$$\overline{lh} = \frac{\sum_{b=0}^{B-1} lh_b}{B}$$

(4.2)

where $B$ denotes the total number of blocks.

For each of the blocks, if the corresponding block in the next frame is similar, then the result of the likelihood ratio would be $\approx 1$. If, on the other hand, the blocks are dissimilar, then the result of the likelihood ratio would be $>> 1$. Based on this information, a suitable threshold can be selected such that a shot cut is judged to have occurred if the likelihood ratio exceeds it.

$$\overline{lh} \begin{cases} > T & \text{shot cut} \\ \leq T & \text{contiguous shot} \end{cases}$$

(4.3)

4.2 Average Intensity Measurement (AIM)

Following the likelihood ratio method, a method where the average intensity of the whole image was taken and then used to detect the shot cut can be used. Where the pair-wise method is a local operation at the pixel level, the likelihood ratio is also a local operation with the exception of using groups of pixels. By taking the average
4.2. Average Intensity Measurement (AIM)

Intensity across the entire frame, the operation has become a global one instead. This is termed as the Average Intensity Measurement.

For the Average Intensity Measurement, the average intensities over the entire image in each of the Y, U and V components are calculated. Let us denote the average of each component \( c \) of a frame, \( i \), of size \( M \times N \) as \( A(V_i(c)) \), i.e.:

\[
A(V_i(c)) = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} V(x, y, i, c)}{M \times N} \quad c \in \{Y, U, V\} \tag{4.4}
\]

After the averaging, the absolute difference between the current frame and the new frame is divided by the absolute difference between the previous frame and the current frame, as shown in Eq. 4.5. To obtain the final result, the average intensity difference for each component is summed (Eq. 4.6).

\[
F_i(c) = \frac{|A(V_i(c)) - A(V_{i+1}(c))|}{|A(V_{i-1}(c)) - A(V_i(c))|} \tag{4.5}
\]

\[
F_{ai}(c) = (F_i(Y) + F_i(U) + F_i(V)) \tag{4.6}
\]

By taking the average intensities over the entire image, the problem of sensitivity to camera and object motion is reduced compared to simple pair-wise pixel difference. Using entire frames increases the tolerance to slow and small object motion from frame to frame.

The algorithm in Eq. 4.5 depends on the order of magnitude for both the numerator and denominator to be equal for frames that are within the same shot. Our experiments have shown that this is not always the case. As a result, there are cases where the algorithm becomes unstable and false positives were flagged. This instability impacted the performance of the algorithm to quite a degree. In view of this, we implemented the algorithm without the denominator as so:

\[
F_i(c) = |A(V_i(c)) - A(V_{i+1}(c))| \tag{4.7}
\]
4.3 Histogram Comparison (HC)

An alternative to taking the average of intensities over the entire frame is to construct the histogram of the intensities. The difference between the intensity histogram of successive frames is then taken and used as the metric for which shot cuts are decided on. The principle behind this algorithm is that two frames having an unchanging background, and unchanging objects will show little difference in their respective histograms.

In [ZKS93], the histogram of grey-level values is created. In our implementation, the idea is extended to include the colour components as well. The histogram created is still a one dimensional histogram with the colour components arranged in bins following those of the grey-level bins (Fig. 4.1).

![Histogram of Y, U, and V components](image)

**Figure 4.1**: Arrangement of the bins for the Y, U and V components in the histogram

Let $B$ be the number of bins in the histogram and $H_i(j)$ denote the histogram value for the $i$th frame, where $j$ is one of the possible bin values. To perform the his-
4.4 Motion Estimation / Prediction Error (ME)

Motion estimation is the opposite approach to the above methods where the motion activity itself is used as a metric for detecting shot changes. For the motion estimation approach, we use motion estimation prediction similar to that used in current video coding standards, e.g. MPEG-1, H.261 and H.263. The prediction error between the reconstructed frame and the original frame is used as the metric.

To estimate the motion, the block-based n-step search algorithm for a $\pm 2^n$ search window as described in [Tek95] is used. The search area is $\pm 16$ pixels, i.e. $n = 4$. After the motion estimation is done, the motion vectors are used to construct the next frame in the sequence. To obtain the prediction error, the absolute difference between the reconstructed frame and the original frame is taken and summed. Eq. 4.9 demonstrates this, where $F_i(x, y)$ is the intensity of the pixel at point $(x, y)$ in frame $i$ and $F'_i(x, y)$ is the intensity of the pixel at point $(x, y)$ in the reconstructed frame based on the block-matching applied to the previous frame.

$$E(F_i) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} |F_i(x, y) - F'_i(x, y)|$$

where $E(F_i)$ is the prediction error of the reconstructed frame.
4.5 DCT based methods

The Euclidean distance method and the clustering method described in this section both utilise the Discrete Cosine Transform. Here, we have moved from working in the spatial domain to the frequency domain. We start by defining the Discrete Cosine Transform. The forward 2-dimensional Discrete Cosine Transform on an $N \times N$ block can be written as [Pit93]:

$$F(u, v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \cos \left\{ \frac{(2x + 1)\pi x}{2N} \right\} \cos \left\{ \frac{(2y + 1)\pi y}{2N} \right\}$$

In the implementation developed for this investigation, the frames are divided into $16 \times 16$ blocks for the luminance and $8 \times 8$ macroblocks for the chrominance for QCIF sequences\(^1\). The DCT is applied to each of these blocks separately.

The DCT coefficients are obtained using the one-dimensional Fast-DCT algorithm developed by Chen, Smith and Fralick [CSF77]. The separable nature of multidimensional DCT algorithms is exploited such that the horizontal and vertical directions could be calculated in succession using the same one-dimensional algorithm.

4.5.1 Euclidean Distance (ED)

To measure the Euclidean distance, only the first coefficient, i.e. the DC coefficient, of the DCT calculation is used. These coefficients for each block in the frame (luminance and chrominance components) was extracted and averaged as shown in Eq. 4.11.

$$F(0, 0) = \frac{1}{B} \sum_{b=0}^{B-1} F^b(0, 0)$$

where $B$ denotes the total number of blocks.

These values are the measure of the average colour/intensity, or energy, in each frame. We denote the average for each band of $Y$, $U$ and $V$ as $\overline{Y}$, $\overline{U}$ and $\overline{V}$ respectively.

\(^1\)For CIF sized sequences, the block sizes were $32 \times 32$ for luminance and $16 \times 16$ for chrominance
4.5. DCT based methods

The Euclidean distance between two consecutive frames $i$ and $i + 1$ with respect to the average (mean) DC value of the DCT coefficients in each block can now be expressed as [VK96]:

$$ S(i, i + 1) = (Y_{t+1} - Y_i)^2 + (U_{t+1} - U_i)^2 + (V_{t+1} - V_i)^2 $$

(4.12)

It is clear that within the same shot, the energy in each of the frames would be fairly similar and as such the result of the Euclidean distance calculation would be very small. On the other hand, we would expect that the difference between the average energy in frames from two different shots would be significantly large. Therefore, if the Euclidean distance is above a certain threshold, $T$, then a shot change is noted as to have occurred.

Strictly speaking, the DCT need not be implemented if we were only to use the DC coefficients since it is only a measure of the mean pixel intensity within a block. Therefore, Eq. 4.11 is numerically equal to Eq. 4.4. But since we need to use the frequency coefficients as well (to be described in the next sub-section) we felt that we could use the data that is made available here to mirror the work first carried out by Vellaikal and Kuo [VK96].

4.5.2 DCT Clustering (DCT-C)

In the clustering method [AS96], only the luminance component of the YUV sequences is used. The division of the frames into $16 \times 16$ blocks (for the QCIF sequences used as our experimental video input) gives rise to $11 \times 9$ blocks. For each block, the DC, first horizontal and first vertical coefficients are extracted and put into a vector of dimension $11 \times 9 \times 3$. In other words, there are 297 components in the vector representing each frame.

An initial cluster is constructed, on the assumption that the first 30 frames of the video is within the same shot. We make two assumptions for the calculation of the cluster:

1. The DCT coefficients could be modelled by a Gaussian Distribution
Chapter 4. Baseline Shot Cut Detection Algorithms

2. The components of the sample DCT vector are statistically independent.

We calculate the mean, \( \mu \), and standard deviation, \( \sigma \), for the initial cluster. For each subsequent frame, the DCT component, \( x_c \) in the sample vector of the new frame is checked against the corresponding mean value \( \mu_c \) from the cluster. If \( x_c \) is more than \( k\sigma_c \) from the mean value, the component is considered to be outside the current cluster (see Eq. 4.13).

\[
|x_c - \mu_c| > k\sigma_c
\]

(4.13)

where \( k \) is a user-defined constant.

If the number of components which satisfy Eq. 4.13 exceeds half the total number of components in the vector, the frame is considered as outside the cluster. When the number of frames outside the cluster reaches a certain threshold (say, \( R \) frames), a new cluster is created. Otherwise, the \( \mu \) and \( \sigma \) are recalculated and updated by adding the samples from the new frames. In the experiments described in the next section, the values of \( k \) and \( R \) were 3 and 12 respectively.
The DCT clustering (DCT-C) is employed in two different ways. In the first implementation, if the Euclidean method (ED) detected a shot change and the DCT-C did not in the first pass, a second pass by the DCT-C algorithm is made by doing a "roll-back" of 30 frames. The cluster is reinitialised using the 30 frames up to but not including the frame in which a new shot is detected by the ED method. The clustering algorithm then proceeds as previously described. In the second implementation, no "roll-back" is employed.

The reasoning for having to reinitialise the clustering in the event of a disagreement with the Euclidean distancing is explained in the following paragraphs.

The problem can be divided into two cases:

1. There is an actual shot cut, therefore ED is correct, or
2. There is no actual shot cut, i.e. ED is wrong

As previously explained, for the DCT-C, if the frame currently under consideration is judged to be within the same shot, the samples in the vector is added into the cluster and the mean and standard deviation are recalculated correspondingly. However, if the frames within a shot contain substantial motion, the standard deviation, $\sigma$, when calculated over all the frames in the shot may be large.

In the case where the next frame is from a new shot (as in case 1), the number of components in its sample vector that are at a distance of $k\sigma$ from the cluster mean may be less than half of the total vector dimension. This could be attributed to the large value of the standard deviation, $\sigma$.

Since the DCT-C would fail to detect a shot cut in that frame, the samples in the vector is again added to the cluster and the $\mu$ and $\sigma$ are recalculated. As more data from the subsequent frames are added, the $\sigma$ would continue to expand. The experiments have shown that once an actual shot cut failed to be detected by the DCT-C, every other following shot cuts were also undetected. Therefore, to overcome this, the roll-back method described above is implemented.
4.6 Experimental results and Discussion

In our experiments, two types of video sequences were used. In the first, a montage of separate sequences were concatenated together to simulate camera breaks. There are seven sequences of this type. We use this set of sequences mainly as training for the algorithms. In the second set, the sequences are taken from broadcast television as described in Chapter 3, Table 3.1. The training sequences consisted of camera breaks only. The broadcast sequences contained both camera breaks and a whole range of gradual transitions, as well as camera flashes.

In the tables and following analyses the algorithms have been assigned “keycodes” as shown in Table 4.1. Table 4.2 gives the number of actual shot cuts included in these sequences. The sequences prefixed by “T” are the training sequences and the broadcast sequences are prefixed by a “B”. Each of the algorithms were run in turn for each of these sequences. The analysis of the results follows.

<table>
<thead>
<tr>
<th>Algorithm Name</th>
<th>Keycode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Likelihood Ratio</td>
<td>LH</td>
</tr>
<tr>
<td>Average Intensity Measurement</td>
<td>AIM</td>
</tr>
<tr>
<td>Histogram Comparison</td>
<td>HC</td>
</tr>
<tr>
<td>Motion Estimation</td>
<td>ME</td>
</tr>
<tr>
<td>Euclidean Distance</td>
<td>ED</td>
</tr>
<tr>
<td>DCT Clustering</td>
<td>DCT-C</td>
</tr>
</tbody>
</table>

Table 4.1: Keycodes for the algorithms.

All the algorithms have two common properties:

1. Some measure of the image intensity is carried out by all the algorithms.

   The likelihood ratio compares the intensity of the frames within regions. The average intensity difference measurement is just that, a measure of the intensity difference averaged across the entire frame. For the histogram comparison, the difference between the intensity distributions of successive frames
4.6. Experimental results and Discussion

<table>
<thead>
<tr>
<th>Sequence</th>
<th>no. of frames</th>
<th>no. of shot cuts</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-1</td>
<td>600</td>
<td>6</td>
</tr>
<tr>
<td>T-2</td>
<td>590</td>
<td>6</td>
</tr>
<tr>
<td>T-3</td>
<td>655</td>
<td>7</td>
</tr>
<tr>
<td>T-4</td>
<td>950</td>
<td>9</td>
</tr>
<tr>
<td>T-5</td>
<td>2195</td>
<td>24</td>
</tr>
<tr>
<td>T-6</td>
<td>6645</td>
<td>21</td>
</tr>
<tr>
<td>T-7</td>
<td>9615</td>
<td>71</td>
</tr>
<tr>
<td>B-1 (DV)</td>
<td>36000</td>
<td>268</td>
</tr>
<tr>
<td>B-2 (SKY)</td>
<td>45000</td>
<td>289</td>
</tr>
</tbody>
</table>

Table 4.2: Experimental video sequences with no. of actual shot changes.

was computed. The block matching in the motion estimation method calculates the minimum mean squared error of the intensity of the block being currently compared to the previous frame. For the DCT-based methods, the DCT transform was performed on the pixel intensities.

2. A thresholding operation is applied to make a decision.

Except for the case of the DCT clustering, a thresholding was applied in the final stage of processing. The clustering, on the other hand, had three thresholding stages. The first is related to the distance which defined whether a vector component to be classified was “outside” the cluster. The second was the number of “outside” vector components allowed before the frame was considered significantly different from its predecessor. Finally, the number of consecutive frames outside the current cluster requires another threshold.

The results presented here represent our preliminary experiments and findings on the algorithms discussed above. The training set was mainly used to find a good operating threshold for the algorithms. This was done by manual inspection of the results for the training set as the threshold was varied. Once we have determined the best threshold value for a particular algorithm, we used that value on a subset
of the GENERAL broadcast set, i.e. DV and SKY. For a small data set, this empirical method was sufficient. However, as the data set becomes larger, a more robust and efficient method was needed. In Chapter 5 we will describe a method for finding the operating threshold using Receiver Operating Characteristic (ROC) curves which fulfill this requirement.

4.7 Analysis of the results

In the following discussion of the experimental results, we confine our analysis to sequences 6 and 7 only from the training set. We then follow through with comments on the two broadcast sequences tested. The analysis of the results apply equally to the other sequences tested as well.

4.7.1 The Training Set

Tables 4.3 and 4.4 give the results of the algorithms on the sequences. In the tables, $S_t$ denotes true shot changes detected, $S_f$ denotes false detection of shot cuts and $S_u$ denotes undetected true shot cuts, respectively.

For sequence 6, except for the DCT-C, all the algorithms detected several false shot cuts. The false detections occurred mainly within 50 frames of a sequence that contained an explosion as shown in Fig. 4.3.

As previously noted, the ED and AIM method are similar. The results of the algorithms on the test sequences also verify this with the exception of sequence 6. ED detected 5 more false positives compared to AIM. This can be attributed to the selection of thresholds for the two algorithms with ED being more critically lower compared to AIM. Also, all the 5 false positives in ED occurred within the 50 frames that contained the explosion.

As was previously mentioned, all the algorithms compute some measure of pixel intensity. As demonstrated in Fig. 4.3, the increase in intensity as the explosion grew was very rapid. The DCT-C managed to avoid this false shot detection pitfall
## 4.7. Analysis of the results

### Table 4.3: Results of the algorithms in detecting shot changes: Sequences T1–T4.

<table>
<thead>
<tr>
<th></th>
<th>Sequence</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$S_l$</td>
<td>$S_f$</td>
<td>$S_u$</td>
<td>$S_l$</td>
<td>$S_f$</td>
<td>$S_u$</td>
</tr>
<tr>
<td>LH</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>AIM</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HC</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ME</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ED</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DCT-C</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

### Table 4.4: Results of the algorithms in detecting shot changes: Sequences T5–T7.

<table>
<thead>
<tr>
<th></th>
<th>Sequence</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$S_l$</td>
<td>$S_f$</td>
<td>$S_u$</td>
<td>$S_l$</td>
<td>$S_f$</td>
<td>$S_u$</td>
</tr>
<tr>
<td>LH</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>13</td>
<td>1</td>
</tr>
<tr>
<td>AIM</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>HC</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>21</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>ME</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>ED</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>DCT-C</td>
<td>22</td>
<td>0</td>
<td>2</td>
<td>20</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 4.3: Ten consecutive frames of an explosion shot.
since the algorithm was designed to be more robust when faced with such occurrences. Fig. 4.4 shows the response graphs for the other algorithms within this 50 frames.

![Response graphs for the explosion sequence](image)

(a) Likelihood Ratio  (b) Average Intensity Measurement  (c) Histogram Comparison

(d) Motion Estimation  (e) Euclidean distance

Figure 4.4: Response graphs for the explosion sequence in Fig. 4.3.

Another interesting point to note is that all the algorithms, except for HC, had one undetected shot cut. Figure 4.5 shows the two frames where the shot cut occurred. However, the consequence of HC being able to detect the shot cut is that it recorded the highest number of false detections for the explosion shot as well. The failure of the other algorithms to detect the shot cut could be attributed to two factors. First, the intensity levels of the two shots are fairly similar. Second, the threshold values used by the algorithm are difficult to choose. We note that, if the nature of the video to be parsed is known, the threshold could be *tuned* to achieve the best results for it. However, this would reduce the suitability of the algorithms to a more general case. For our experiments, we have decided to keep to the same thresholds for all the sequences.
4.7. Analysis of the results

For sequence 7, all the algorithms performed satisfactorily. The HC and the ME methods obtained a perfect score. This can be attributed to a carefully chosen threshold for the decision making. In the case of the LH, one false detection was made. For the ED and AIM, four false detections were made. The DCT-C algorithm, whilst robust to sudden changes to intensity, also tended to have a higher degree of failure for detecting actual shot cuts. This was demonstrated by its failure to detect four actual shot cuts in sequence 7. This is a side effect of the clustering method which has a reduced sensitivity to intensity changes due to the expansion of the standard deviation in certain cases.

A closer examination of the two algorithms, AIM and ED, would show that they are both basically the same. As mentioned in Section 4.5, the DC coefficient of the DCT on a block is a measure of the average value of the pixel intensities within the block which is the average intensity measurement of the block. The sum of the average intensity measurement of each block results in the average intensity measurement of the whole image. Fig. 4.6 shows the response graph of the algorithms. Peak no. 6 in both graphs correspond to a false detection between two frames similar to that shown in Fig. 4.7.

The difference between the two algorithms lie only in the way the final calculation is made. As a matter of fact, the calculations for the Euclidean distance could have been made without the use of the Discrete Cosine Transform. However, as mentioned previously in connection with the clustering method, another shot de-
Chapter 4. Baseline Shot Cut Detection Algorithms

![Figure 4.6: Response graphs of the likelihood ratio (left) and the Euclidean distance (right) algorithms on Sequence 1. The peaks in the graphs are the shot cuts.](image)

ection method was needed to give feedback to the clustering algorithm so that a re-initialisation of the vectors could be instigated to avoid undetected shot cuts (Section 4.5.2). The DCT coefficients were already calculated for the DCT-C. Therefore, implementing the ED algorithm was computationally cheap in the sense that it also makes use of the DCT coefficients.

For the LH, AIM and ED, the false detections are not as clear cut as it first appears. Fig. 4.7 show two consecutive frames in sequence 7 where the false detection in LH, AIM and ED occurs. The shot is of a news program where there are two newsreaders with a screen in the background. Whilst it can be argued that the two frames are still in the same shot, the TV screen in the background was clearly having a shot cut of its own. This problem can be looked at in two ways. In the first, a decision can be made such that the TV screen in the background is only part of the frame and as such, a shot cut is not classified. The opposite of this is to classify a shot cut since there clearly was a shot cut even if it was only within part of the frames. A decision to choose one definition or the other depends on the application. For the purposes of this chapter, instances such as shown in Fig. 4.7 were classified as a false detection to highlight this particular issue.

It would be interesting to study shot cuts in sequences such as those in Fig. 4.7. Since the LH and ED algorithms are block based methods, a dual classification system could be implemented. Since block based methods can be looked at as local operations that calculate properties within each block, a local threshold that
4.7. Analysis of the results

Figure 4.7: Two consecutive frames with a localised shot change.

compares the corresponding blocks only can be utilised. If a certain number of connected blocks exceeds the threshold then it could be noted as a localised shot cut. Fig. 4.8 demonstrates this idea.

Figure 4.8: The division of the frame into blocks. The background TV screen consists of approximately 20 connected blocks.

The best way to carry out this implementation is to have a priori information regarding the video sequence. The sequence in which a frame was shown in Fig. 4.8 was a news sequence with a TV screen in the centre. As such, the method could be adapted to also keep track of the 20 blocks occupied by the TV screen. Another example would be a caption bar at the bottom of the screen.
4.7.2 The Broadcast Set

The results of the algorithms on the two large sequences are shown in Table 4.5. We could see from the results that there is a higher variation in performance between the algorithms when faced with more difficult data. The number of shot cuts to be detected has also greatly increased and gives us a better evaluation of the algorithms.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>( S_t )</th>
<th>( S_f )</th>
<th>( S_u )</th>
<th>( S_t )</th>
<th>( S_f )</th>
<th>( S_u )</th>
</tr>
</thead>
<tbody>
<tr>
<td>LH</td>
<td>223</td>
<td>49</td>
<td>45</td>
<td>258</td>
<td>28</td>
<td>21</td>
</tr>
<tr>
<td>AIM</td>
<td>213</td>
<td>51</td>
<td>55</td>
<td>257</td>
<td>88</td>
<td>22</td>
</tr>
<tr>
<td>HC</td>
<td>252</td>
<td>53</td>
<td>16</td>
<td>274</td>
<td>61</td>
<td>5</td>
</tr>
<tr>
<td>ME</td>
<td>203</td>
<td>11</td>
<td>65</td>
<td>247</td>
<td>58</td>
<td>32</td>
</tr>
<tr>
<td>ED</td>
<td>205</td>
<td>57</td>
<td>63</td>
<td>251</td>
<td>75</td>
<td>28</td>
</tr>
<tr>
<td>DCT-C</td>
<td>183</td>
<td>4</td>
<td>85</td>
<td>223</td>
<td>6</td>
<td>56</td>
</tr>
</tbody>
</table>

Table 4.5: Results of the algorithms in detecting shot changes.

The HC algorithm demonstrated the best performance in detecting actual shot cuts. The LH and AIM algorithms also did creditably with the ME and ED algorithms trailing behind. However, all these algorithms tend to produce a sizeable number of false detections. The DCT-C algorithm was the worst performing in terms of detecting actual shot cuts. There are certain cases where the DCT-C is certain to fail. An example would be the case where there are less than \( R \) frames before a shot cut. However, it also recorded the lowest amount of false detections. The conclusion that can be derived from this is that the DCT-C algorithm is generally less sensitive to content change in video sequences.

Note that from the results of these two sequences, we can see that there is a discernible variation in the performance. For example, the ME algorithm detected very few false positives in sequence 1 but did not perform so well for sequence 2. Also, all the sequences were better at detecting true positives for sequence 2 than
In sequence 1 there was an instance where there were shot cuts in three consecutive frames as shown in Fig. 4.9. This represents an anomaly which could be attributed to production error prior to broadcasting. Nevertheless, all the algorithms bar the DCT-C detected the shot cuts. In this case, the DCT-C only detected the first shot cut.

4.8 Summary

The above experiments have shown the capabilities of the individual algorithms. The histogram comparison (HC) method showed the best overall performance with the LH and AIM methods coming close behind. The DCT-C algorithm proved most insensitive to false positives but was not as capable as the other algorithms in detecting true cuts.

In terms of processing cost, the HC and AIM methods were the fastest to compute. On the other hand, these two methods are global measurement methods, and so would not lend themselves well to applications that would also want to detect local changes.
Chapter 4. Baseline Shot Cut Detection Algorithms

The global measurement methods are the most insensitive to motion within shots whereas the motion estimation approach (ME) is based on analysing the motion across frames. The ME, ED and DCT-C methods are useful when working with video sequences that are compressed using one of the video coding standards like MPEG-1, H.263, etc.

<table>
<thead>
<tr>
<th>Local operations</th>
<th>Global operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>LH</td>
<td>AIM</td>
</tr>
<tr>
<td>ME</td>
<td>HC</td>
</tr>
<tr>
<td>ED</td>
<td></td>
</tr>
<tr>
<td>DCT-C</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.6: Local and global algorithms for automatic shot detection.

The selection of the operating threshold was arrived at empirically by varying the thresholds on the training set and observing the results. The value was then used for the broadcast sequences. Whilst this approach is viable for small datasets, it becomes an unwieldy method once exposed to larger datasets. As shown in the results for the broadcast sequences, there was a larger variety in performance for each of the algorithms. This demands a greater analysis of the results and a more robust method for determining the operating threshold.
Chapter 5

Combining Multiple Experts

5.1 Introduction

In this chapter we present a shot cut detection technique using a combination of multiple experts. The experts themselves are the stand-alone methods to detect shot cuts detailed in Chapter 4. We propose a combination method that gives significantly better results compared to these experts on their own. We achieve this by exploiting the complementarity of expert knowledge, i.e. the fact that the various experts calculate different features of the video sequences.

For a shot detection method to be successful, it needs to be as accurate as possible. This accuracy is normally measured in terms of the percentage of true shot changes it is able to detect, as well as the number of false positives. Needless to say, the choice of a threshold directly affects these performance measures.

In a previous work [YCK98] as described in Chapter 4, we studied a selection of these methods and evaluated their individual strengths and weaknesses. Whilst each method had the capability of performing quite well, there was still scope for improvement. In particular, it became clear that different methods performed well in diverse circumstances, which has also been pointed out in various comparison studies published recently [YCK98, BR96, AL96, DAE95, OSMM99]. In other words, no simple approach outperformed the other in all situations. Rather, the su-

47
priority of a particular method was data dependent. This immediately suggested the potential benefit of using these various methods together; to fuse them in such a way that the strengths of each method are consolidated and the weaknesses muted. This was also the approach suggested in [Sme98, SGG+99] in discussing disparate shot cut detection algorithms and their performances.

Multiple expert fusion has recently become popular in many application domains. A large variety of techniques have been suggested for combining hard and soft decision outputs, scores and probabilities [Kit98]. Different approaches to generating multiple opinions have also been proposed, including boosting [SFBL97], bagging [Bre96], and multiple designs based on different initialisations and training conditions [RF97a]. A few attempts to develop a theory underpinning multiple expert fusion have also been reported [RF97b, KHDM98].

In the shot cut detection problem considered here the individual experts output hard decisions. This restricts the available fusion strategies to schemes such as voting and behaviour knowledge classification [HS95, SFBL97]. These are invariably applied to experts making their decisions at some chosen operating point such as equal error rates, i.e. equal probability of true shot cuts undetected and false shot cuts detected, or where the sum of these probabilities is minimum. Interestingly, beneficial effects of fusion can be achieved by setting the operating points of individual experts at suboptimal performance levels. This has been demonstrated, e.g. by Scott, et al. [SNP98] in the context of maximising the area under the receiver operating characteristic curve of a multiple expert decision making system. However, in their approach, expert fusion system performance is limited by the convex hull of the ROC curves of the individual experts. In this chapter we show that one can improve the performance significantly beyond this level by exploring more fully the multidimensional space of all the possible joint operating points of the cooperating experts. In fact, in some cases we have achieved error reduction by a factor of two as compared with the ROC convex hull limit.

This chapter is organised as follows. In the next section, we give a short description of the methods we have chosen as the experts in our combination process. We also
discuss other work related to using multiple experts in classification systems. Following that section we give details of our approach to combining multiple experts in shot cut detection. We then present our experimental results and the final section gives our conclusions and future work plans.

5.2 Related Work

5.2.1 Baseline Shot Detection Algorithms

From our earlier experiments, we implemented six algorithms for detecting shot cuts, which are:

1. Average Intensity Measurement [AIM]
2. Euclidean Distancing [ED]
3. Histogram Comparison [HC]
4. Likelihood Ratio [LH]
5. Motion Estimation / Prediction Error [ME]
6. DCT-coefficient clustering

Apart from the last method, these methods calculate a dissimilarity measure from which we could use to make shot cut decisions. The dissimilarity measure is the response of the system when given two consecutive frames as input. We could then plot a response graph for the set of values generated by each method over a whole video sequence (Fig. 5.1). The rationale behind using a dissimilarity measure is that we expect two consecutive frames belonging to the same shot to have a low value and vice versa for two consecutive frames of different shots. Fig. 5.1 shows an example of this where the peaks in the graph are points where the dissimilarity measure values are high, and this would correspond to a high probability that these points are shot change boundaries.
In the previous chapter, we employed a global threshold for each of the algorithm at the final stage of the processing to make the shot boundary decision. We used a small training set to empirically choose the thresholds. Upon testing the thresholds, it became clear that a more robust method for detecting the thresholds is needed. In view of this, we constructed a receiver operating characteristic curve for each method, obtained by setting the thresholds to various possible values.

To construct the ROC curves, we calculate the proportion of undetected true shot boundaries, $p_u$ against the proportion of incorrectly identified shot boundaries, $p_f$.

\[ p_u = \frac{S_u}{S_a} \quad (5.1) \]

\[ p_f = \frac{S_f}{S_a} \quad (5.2) \]

where $S_u$ is the number of undetected true shot boundaries, $S_f$ is the number of falsely identified ones and $S_a$ is the number of actual shot boundaries.

We then set the thresholds to different values and plot $p_u$ against $p_f$. Note that none of the experts require any parameters to be set (apart from the threshold) and therefore none need any training. We plot the ROC curves for each of the five algorithms on the video sequences (Figs. 5.2 and 5.3). With the help of the ROC curves, we can decide on a particular operating point and obtain the corresponding
Figure 5.2: The ROC curves for the video sequences
Figure 5.3: The ROC curves for the video sequences
threshold for each expert. For example, in terms of equal error performance, we would choose the threshold that corresponds to the point on the curve nearest to the origin from both axes, since we would ideally want to minimise both $p_u$ and $p_f$.

We can observe that there is a variation in the performances of the algorithms over the different sequences. We can expose these variations further by showing the points on which the algorithms are plotted given the same threshold value, as demonstrated in Fig. 5.4.

### 5.2.2 Compound Shot Detection Algorithms

For the four “real-world” sequences, the best individual expert would be either the HC method or the LH method. The observation of this inconsistent behaviour led to other authors exploiting multiple experts in tandem. For example, in [TD98], two methods were used to create a generalised sequence trace of the video sequence. The two features used were the luminance histogram difference and standard deviation difference. This trace is defined as the sum of the square root of the difference between two frames for each feature. Then, using a technique based on mathematical morphology, the authors constructed what they termed a morphological laplacian graph for the sequence. The morphological laplacian algorithm essentially calculates the difference between the gradient of dilation and the gradient of erosion for the generalised sequence trace, which corresponds to an approximation to the second derivative of the aforementioned trace. The zero crossings on the graph indicated shot boundaries. To distinguish between zero crossings due to true shot boundaries and noise, a threshold was applied.

In [DRA98], the authors proposed a two step shot detection strategy whereby a histogram comparison method was used in the first step and a likelihood ratio method was selectively used as the second step. In their implementation, the histogram comparison results were subjected to two thresholds, $T_H$ and $T_L$, with $T_H$ being the higher threshold. If the comparison result is higher than $T_H$, then a cut is declared immediately. If, on the other hand, it is lower than $T_H$ but higher than $T_L$, a likelihood ratio operation is carried out. If the results of the likelihood ratio is
Figure 5.4: Plots of threshold values for the different sequences.
above a set threshold, \( T_R \), then a cut is declared.

In another work [NMF+98], a shot detection scheme employing two algorithms were described. Using the histogram comparison and a pixel-wise differencing algorithm (similar to AIM), the authors then employed a K-means clustering algorithm to classify the results into two clusters. After this, an elimination step based on a heuristic observation is employed to reduce the number of false positives.

The three works mentioned above [TD98, DRA98, NMF+98] presented some quantitative results though the possibility of applying more experts was not explored.

5.3 Combining Multiple Experts (CME)

An interesting decision-level fusion algorithm is advocated in [SNP98]. This algorithm was originally devised to minimise the area under the ROC curve by randomising the decisions of an expert corresponding to selected pairs of operating points on the ROC. A judicial choice of such operating points results in an improved performance defined by a convex hull of the original ROC curve. This idea can be extended to the problem of combining several experts producing a fusion and hence exhibiting the performance corresponding to a ROC curve which is the convex hull of all the individual experts’ ROCs.

In order to achieve even greater performance gains, we approach our particular problem from a different angle. We base our method on the hypothesis that different algorithms calculate different features or properties of the video data. Therefore, we contend that the true positives undetected by one method may be detected by another. Conversely, a false positive detected by one method might be correctly rejected by another. This is supported by analysis of the response graphs for each method. We find that each method would exhibit a peak at the respective shot boundaries. However, the strength of the peaks would be different.

The graphs in Figs. 5.5 and Fig. 5.6 show the responses of the LH and ME algorithms over different sections of the same video sequence. The frames (a) and (b), (c) and (d), (e) and (f) correspond to shot boundary peaks A, B and C on the graphs.
Chapter 5. Combining Multiple Experts

Figure 5.5: Example of response graphs to the LH and ME algorithms I.

Figure 5.6: Example of response graphs to the LH and ME algorithms II.
respectively. The horizontal line shows our optimum equal error threshold for each algorithm. In Fig. 5.5 we can see that there are two peaks (B and C) above the threshold in LH that are below the threshold in ME. Correspondingly, in Fig. 5.6 the opposite holds true where there are two peaks (again B and C) above the threshold in ME that are under the threshold in LH.

Also, it is worth noting that, in both examples, there are peaks (marked as A) that are below the thresholds of both algorithms. Our manual inspection shows that these peaks are at bona fide shot boundaries. These figures that we have just discussed gives us the examples of situations which we would like to improve — by using the experts in a cooperative manner to detect the shot boundaries like those of B and C, where not all of the experts are in agreement, and using suboptimal threshold positions to detect shot boundaries like A.

Our approach is based on the hypothesis that the expert fusion system performance can be substantially improved by setting the operating points of the individual experts at different points. Let us denote the operating point \((p_u, p_f)\) of expert \(i\), \(i = 1, ..., n\), by \((p_u^i, p_f^i)\) where \(p_u^i\) and \(p_f^i\) are the probabilities of a false rejection and false acceptance respectively. The corresponding threshold will then be \(T(p_u^i, p_f^i)\).

For every pair of frames, expert \(i\) computes the dissimilarity measure \(S_i\) and makes a decision:

\[
d(S_i, p_u^i, p_f^i) = \begin{cases} 
\text{cut} & \text{if } S_i \geq T(p_u^i, p_f^i) \\
\text{no cut} & \text{otherwise}
\end{cases} \quad (5.3)
\]

Let the number of experts be \(n\) and the number of experts of the opinion that there is a cut between the two frames is \(n_c\). The complement \(n_n = n - n_c\). Then the consensus decision \(D\) of the multiple expert fusion system (CME) is as follows:

\[
D = \begin{cases} 
\text{cut} & \text{if } n_c \geq n_n \\
\text{no cut} & \text{otherwise}
\end{cases} \quad (5.4)
\]

The performance of CME is dependent on the choice of the operating points, \((p_u^i, p_f^i)\). Our aim is to find the values of \((p_u, p_f)\) for each \(i\) which gives the best per-
Table 5.1: Threshold values used in the CME algorithm.

<table>
<thead>
<tr>
<th>$n$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIM</td>
<td>1.5</td>
<td>1.9</td>
<td>2.8</td>
<td>5.15</td>
<td>13.2</td>
</tr>
<tr>
<td>ED</td>
<td>0.78</td>
<td>0.88</td>
<td>1.31</td>
<td>2.11</td>
<td>5.91</td>
</tr>
<tr>
<td>HC</td>
<td>0.095</td>
<td>0.111</td>
<td>0.170</td>
<td>0.285</td>
<td>0.452</td>
</tr>
<tr>
<td>LH</td>
<td>1.105</td>
<td>1.160</td>
<td>1.240</td>
<td>1.560</td>
<td>2.220</td>
</tr>
<tr>
<td>ME</td>
<td>25.5</td>
<td>29.5</td>
<td>33.0</td>
<td>39.5</td>
<td>55.0</td>
</tr>
</tbody>
</table>

formance. Since the space of all the operating points is $n$-dimensional and therefore difficult to explore exhaustively, we optimise CME by sampling it quite coarsely. To this end we select the threshold values at five points on the ROC curve for each of the algorithms. These threshold values are taken at the points $p_0$, $p_1$, $p_2$, $p_3$ and $p_4$ as shown in Fig. 5.7.

![Threshold values at the points specified at the lines $p_n$ where $n = 0..4$. DV sequence.](image)

We use a base-$N$ codeword designation to identify our thresholds. Since there are $n$ experts, each with $N$ possible thresholds, this leaves us with $N^n$ possible combinations for which the thresholds could be used. For $N = 5$ and $n = 5$, this gives us 3125 possible combinations. For example, from Table 5.1, the value 04322 would mean that we use the results from the AIM method using a threshold value of 1.5, ED using 5.91, HC using using 0.285, LH using 1.240 and ME using 33.0.
Each of the individual methods would signal a shot change at values above the given threshold. The CME method would signal a shot change only when 3 or more of the algorithms signal a shot change.

5.4 Experimental Results

For this chapter, we selected the equal error rate as our performance criterion. However, it should be noted that this is chosen as a means to quantify our results compared to the individual experts. We could, for example, choose other measures that would be application dependent as the performance criterion—a maximum allowable $p_u$, for instance.

As we wish to find an optimal setting of expert operating points to get the best CME performance we now need training data. We used the DV sequence as our training set; i.e., we took the value of the thresholds from the ROC curves generated from this set. The CME algorithm was then applied to the training set as well as the data sets.

For each of the combinations, we calculate the values for $p_u$ and $p_f$. Fig. 5.8 is a plot of the CME algorithm against the ROC curves that we had constructed earlier showing the region near the origin. Each point on the graph corresponds to a different CME combination.

From Fig. 5.4, noting that the operating points for a given threshold varies from one sequence to another, we would expect the performance of the CME algorithm to display the same behaviour. This is demonstrated in Fig. 5.8(b), for e.g., in the CHF sequence, where the CME cluster is shifted up and to the left of the graph compared to DV and SKY. For the CARTOON and RUGBY sequence, the variation is even more pronounced.

We can see in Fig. 5.8, in the case of the real-world sequences, there are a significant number of points from the CME algorithm that are nearer to the origin than those of any of the individual experts. Since the aim is to reduce the equal error rate by
Figure 5.8: Close-up of the CME algorithm against the ROC curves
5.4. Experimental Results

multiple expert fusion, it is clear that the proposed scheme achieves that objective for these sequences.

Looking closer at the CARTOON sequence, it is observed that the threshold values for a given operating point for each algorithm is higher compared to the other sequences. This is further shown in Fig. 5.9. As such, the CME algorithm is less efficient, with only a few points gaining any advantage over the experts. In the case of the RUGBY sequence, the performance of the HC algorithm far outstripped the other experts. Also, the LH and ME algorithms performed especially badly for this particular sequence. Consequently, our experiments demonstrated that the CME algorithm is unable to improve on the performance displayed by HC.

![Figure 5.9: Operating points](image)

In view of this, we can conclude that the CME algorithm is capable of achieving a greater performance gain in detecting shot changes compared to individual experts when applied to “real-world” sequences.

Since each operating point in the CME algorithm represents a combination of operating points from the individual experts, we need to find an optimal combination. To do this, we combined the results of the CME algorithms and the individual experts for the four sequences. We then plot the CME against the ROC curves again (Fig. 5.10).

Having found the optimal operating point, we apply the combination on the indi-
Chapter 5. Combining Multiple Experts

Figure 5.10: Combined results of the “real-world” sequences.

Figure 5.11: Optimal CME operating point.
5.4. Experimental Results

vidual sequences again. From Fig. 5.11, we could see that all the points were better than any of the individual algorithms. Even though the improvement is not as dramatic for all the sequences (Fig. 5.11(d) being a case in point), we demonstrated that no "tuning" is required to get a good performance. Thus, it is shown that the CME algorithm is also robust towards slight shifts in operating points.

5.4.1 Failure Analysis

The algorithm is shown to produce better results compared to individual experts for the real-world sequences but failed to make an impact with CARTOON and RUGBY.

For the RUGBY sequence, the performance of the LH and ME algorithms were especially abysmal and this influenced the results of CME. The AIM and ED algorithms were also not too sterling in their performances. Note that these four algorithms were motion sensitive. HC, which is motion insensitive, on the other hand demonstrated very good results, far outstripping the rest, including CME. We plan to continue our experiments with ME by extending the search window in the hope of achieving an improvement. Also, in the case of LH and ED, since they are block based algorithms, a suggestion would be to have the frames motion compensated before applying the algorithms.

In the following paragraphs, we discuss instances where false positives are detected and occasions where we are unable to detect true positives.

False Positives

Fig. 5.12 gives an example of false positives detected by our method. The frames shown in the figure are frames 12406–12413, denoted within the vertical bars on the response graphs. The horizontal lines on each of the response graphs are the thresholds currently applied to each individual algorithm. Then, we can observe that, using the majority decision, only one of the false positives would be detected by the CME method. If we stipulate that a shot change could not possibly occur
Figure 5.12: False positive example.
within a small number of frames, \( N \), from another shot change, then we can eliminate these types of false positives. However, if a false positive is detected, followed by a true positive within \( N \) frames, then we have doubled our error rate.

**Undetected True Positives**

Undetected true positives present a more undesirable failure than false positives. Detecting false positives would generally result in redundancy where frames from the same shot are detected. Undetected shots, on the other hand, represent loss of information. In most cases, the failures that we experienced are due to two shots containing similar information. Fig. 5.13 shows three shot boundaries that were not detected using the optimal equal error CME combination on the DV sequence.

Here, we base our performance evaluation on the equal error criterion. If we are willing to accept higher rates of false positives, we can then, of course, achieve higher accuracy in detecting true positives.

Another possibility in tackling this problem is by adding a weighting mechanism on the individual experts. In the current version of the CME method, all the experts were given equal weighting. But, from the ROC curves that we have constructed, we can see that not all the methods are equally efficient. This is one avenue which we will be looking into in the future.

### 5.5 Conclusions

We have presented a method of combining multiple experts to achieve significantly better results in detecting shot cut boundaries, by exploiting the fact that the experts carry out shot detection operations on different features of video. We show that we can achieve these results by using suboptimal operating points of the individual experts.

The sampling of the operating points for optimising the CME was done quite coarsely. Even so, we have shown that the error rate could be halved (Fig. 5.10).
The video sequences that we used for our experiments were a combination of news programs, daytime soap operas, documentaries and advertisements. We also tested the algorithm on two other kinds of sequences and have seen that the results were not so encouraging.

Future work would include investigation into improving the performances of the individual algorithms for high speed and animated sequences. There is also scope for employing more experts as well as the improved versions of the current ones. We shall also be exploring other decision making options, one of which would be the application of weighting factors on the experts.

Figure 5.13: Examples of undetected true positives.
Chapter 6

Adaptive Thresholding Methods

6.1 Introduction

In the two previous chapters, we described algorithms for detecting cuts where the decision making depended on a hard threshold which is selected based on experimentation. This generally consists of trying out various values of the thresholds until an optimal value is arrived at. This optimal value depends on the requirements of the application and will be a trade-off between the number of false positives detected and the number of undetected true positives. We also constructed ROC curves to aid in the selection of thresholds.

There have also been several works where the decision making process is carried out within a small window to better increase the accuracy of the algorithm [YL95, DRA98, ZMM99]. In effect, this approach makes the decision making step a local one, as opposed to a global thresholding approach utilised in the previous chapters. These works did not give any indication of how much better the results are compared with conventional (global) decision making. This prompted an exploration into the benefits of localised adaptive thresholding which we set out to quantify in this chapter.
6.2 Previous work

In Chapter 4, we described a set of algorithms for detecting shot cuts. In each of these methods, a single statistic \( m \) is generated for each pair of frames to quantify the degree of dissimilarity between the two frames. We make the assumption that these dissimilarity measures \{\( m \)\} come from one of two distributions: one for shot boundaries (\( S \)) and one for "not-a-shot-boundary" (\( N \)). In general, \( S \) has a considerably larger mean and standard deviation than \( N \) (Fig. 6.1). If, for example, we assume that the costs of false positives and undetected true positives are the same, and that the distribution statistics are stationary, the standard classification methods would indicate that the decision threshold \( m_T \) should be fixed so that the tails of the two density functions \( p_S \) and \( p_N \) (shown shaded in Fig. 6.1) have an equal area. Because of the difference between the widths of the two distributions, we can see that this threshold is fairly close to the mean \( \mu_N \) of \( N \), and that it is therefore important that the position and width of \( N \) are accurately determined.

![Figure 6.1: Distributions of dissimilarity measure \( m \)](image)

In order to determine whether our assumption, as shown in Fig. 6.1 is valid, we plot the dissimilarity measures for each of the baseline algorithms for \( p_S \) and \( p_N \) as shown in Fig. 6.2 for the general sequence. The general sequence and the baseline algorithms are as described in Chapter 3 and Chapter 4, respectively. As can be seen, the tails of \( p_S \) and \( p_N \) overlap for each of the algorithms.
In Chapter 5, it was implicitly assumed that the distributions were indeed stationary, and thus that a single decision threshold could be used. To find this threshold, we experimented with a range of thresholds until the best value was found. In practice however we found, not surprisingly, that the stationarity assumption for $N$ does not hold up well. In particular, we realised that $p_N$ often varies gradually within a shot, and abruptly at shot boundaries. This can be seen from Fig. 6.3, which shows an example of the dissimilarity measure (in this case the mean absolute pixel difference) as a function of frame number, for a sequence of off-air news material. (The sharp peaks correspond to shot boundaries.) In this sequence we can see that the mean level in particular appears to change gradually if at all within
a shot, but jumps significantly in value at the shot boundaries. Furthermore, experiments demonstrated that this single decision threshold can be consistently grossly over- or underestimated when applied to video material with distinctive characteristics, such as sports events or cartoons.

This suggests that it may be possible to improve the detection performance by estimating $p_N$ dynamically, using the dissimilarity measures from the previous and next few frames, and using the result to adaptively set the detection threshold. In practice, we estimate the mean $\mu_N$ and possibly the variance $\sigma_N$ in this way. We then set the threshold $m_T$ to be some function of these statistics, e.g. some fixed distance from $\mu_N$, or some multiple of $\sqrt{\sigma_N}$ from $\mu_N$. The method therefore uses a sliding window of a predetermined size where only the samples within this window are considered for estimating $p_N$. Since frame pairs including a shot boundary are relatively rare occurrences compared with pairs that don’t, we have to assume that $p_S$ is roughly stationary, and slowly varying compared with $p_N$ (Fig. 6.1).

![Graph](image.png)

**Figure 6.3:** Plot of dissimilarity measure against frame number

Investigation of other work in this area has produced some examples which did not fully exploit this possibility. In [OSMM99], the authors implemented a semi-automatic selection of thresholds depending on the program type as take from a television schedule. As such, they had a range of thresholds for various program types, such as news, drama or documentary. The results of their experiments was inconclusive and prompted them to suggest a better adaptive thresholding
method to be developed. In [ZMM99], the authors used a sliding window but a fixed threshold. In [YL95], the threshold is represented as a multiple of the second highest sample within the window. It is in [DRA98] that a structured method to adaptively set the detection threshold is mooted. This is described further in Section 6.4. However, no quantifiable results were presented in these works. Also, there are other possibilities in which the adaptation can be employed. Thus we set out to compare the performance of adapted versus non-adapted algorithms and contrast between several methods which we have developed for the adaptation process and that proposed by Dugad, et. al in [DRA98].

6.3 The experimental data and baseline shot cut detection algorithms

We construct our experiments using the same experimental data that was used in previous chapters. Further, we group the sequences CHF, DV, SKY and SUPERMAN into a general "real-world" sequence since these sequences represent the major part of our data set and has a very good variety of content. The CARTOON and RUGBY sequences represent two different extremes in terms of content and as such, we feel requires a separate treatment from the general sequence.

In terms of the baseline algorithms, we will be using four of the algorithms, namely:

1. Average Intensity Measurement [AIM]
2. Histogram Comparison [HC]
3. Likelihood Ratio [LH]
4. Motion Estimation / Prediction Error [ME]

6.4 The adaptive thresholding scheme

In [YL95], the authors applied a local thresholding method whereby the frame differences of successive $m$ frames is examined. They then declare a shot change when
two conditions are simultaneously satisfied:

1. the difference is the maximum within a symmetric sliding windows of size 
   \(2m - 1\) and

2. the difference is \(n\) times the second largest maximum in the sliding window.

Expanding from this work, a method was proposed in which the means and standard deviations from either side of the middle sample in the window is calculated [DRA98]. The middle sample represents a shot change if the conditions below are simultaneously satisfied:

1. The middle sample is the maximum in the window and

2. The middle sample is greater than \(\max(\mu_{\text{left}} + T_d\sigma_{\text{left}}, \mu_{\text{right}} + T_d\sigma_{\text{right}})\)

where \(T_d\) is taken as 5.

In our work, we experimented with three different methods of estimating the decision threshold. However they all follow the same scheme. We describe the overall scheme next, and then describe the individual thresholding methods in Section 6.4.2.

### 6.4.1 The general method

The general method is as follows. We estimate the mean \(\mu_N\) (and variance \(\sigma_N\) if required) of \(N\) dynamically as part of the shot detection process, from a buffer containing similarity measures \(m\) from the previous \(M\) frames. The value of the decision threshold \(m_T\) is recalculated for each new frame, using one of the methods discussed in Section 6.4.2, and a decision made. However, after a shot cut is detected (and also at the beginning of the video sequence), no new decisions are made until \(M\) frames have elapsed. The following algorithm summarises the method:

```
repeat:
```
empty buffer

for next $M$ frames do:

compute $m$ from previous and current frames
add $m$ to buffer
repeat
get next frame
compute $m$ from previous and current frames
estimate $\mu_N$ from values in buffer
compute new decision threshold $m_T$
replace oldest item in buffer with $m$
until $m > m_T$
record that shot boundary has been detected

until no more frames

6.4.2 Computing the decision threshold

We next describe the three different strategies for setting the threshold. For each of the strategies, we make no assumption about the precise shape of the distributions.

Constant variance model

Here we assume that:

(a) they are unimodal,

(b) $\mu_N$ varies over a small enough range and $p_S$ is sufficiently broad that changes in the value of $p_S$ in the region of the intersection of the density functions can be ignored, and

(c) apart from $\mu_N$, the distributions are stationary.
These assumptions suggest that the threshold could be set at some fixed positive offset from $\mu_N$:

$$m_T = \mu_N + T_c$$  \hspace{1cm} (6.1)

Thus $T_c$ reflects the width of $N$ in some way. The best value of $T_c$ is determined by experimenting with a range of values on a training set of video material for which ground truth information is available.

**Proportional variance model**

If on the other hand the variance, $\sigma_N$, is assumed to vary with $\mu_N^2$, we should set the threshold at some multiple of $\mu_N$:

$$m_T = T_p \mu_N$$  \hspace{1cm} (6.2)

In this case, the width of $N$ is reflected in the value of $T_p \mu_N$. The value of $T_p$ is also determined from experimentation.

**The Dugad model**

As we described earlier, this is an implementation of the model proposed by Dugad, et. al [DRA98], as given below:

$$m_T = \mu_N + T_d \sqrt{\sigma_N}$$  \hspace{1cm} (6.3)

As explained earlier, the authors calculate the means and standard deviations on the left and right of the centre sample and use the maximum of the two. They applied this method on a histogram comparison algorithm and their experimentation gave them an optimum value for $T_d$ as 5. Since we have four base methods, and our histogram comparison methods may not be exactly equivalent, we carried out our experiments, as with the other models, over a range of values to get an optimum $T_d$. 
6.5 Experimental Results

6.5.1 Organization

The three models we have from Section 6.4.2 are:

1. Constant Variance Model – the threshold is added to the mean of the samples. We label this as A.
2. Proportional Variance Model – the threshold is multiplied by the mean of the samples. We label this M.
3. The Dugad, et. al Model – the threshold is multiplied by the standard deviation of the samples and added to the mean. We label this D.

Also, for each model, we adopted two different strategies as follows, leading to six different methods:

1. In the method shown in [DRA98], the window is split into two halves on either side of the centre sample. This method is prefixed with the letter D (Dual windows). Thus a split window using the Dugad model would be labelled DD.
2. Another strategy is to include all the samples, including the centre sample, into the calculation. This method is prefixed with the letter S (Single window). Thus a single window strategy using the Multiplicative method would be labelled SM.

Table 6.1 indicates the optimal window sizes for each method, and for each of the baseline algorithms mentioned above, as explained in Section 6.5.3.

We then apply a range of thresholds for each of the models and construct a Receiver Operating Curve (ROC). In the ROC graphs, the x-axis denote the proportion of false positives calculated as:

\[ p_u = \frac{S_u}{S_o} \]  

(6.4)
Table 6.1: Optimal window sizes

<table>
<thead>
<tr>
<th>Method</th>
<th>AIM</th>
<th>HC</th>
<th>LH</th>
<th>ME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA</td>
<td>25</td>
<td>17</td>
<td>9</td>
<td>21</td>
</tr>
<tr>
<td>DA</td>
<td>25</td>
<td>9</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>SM</td>
<td>15</td>
<td>9</td>
<td>9</td>
<td>19</td>
</tr>
<tr>
<td>DM</td>
<td>15</td>
<td>17</td>
<td>11</td>
<td>21</td>
</tr>
<tr>
<td>SD</td>
<td>21</td>
<td>21</td>
<td>9</td>
<td>21</td>
</tr>
<tr>
<td>DD</td>
<td>21</td>
<td>21</td>
<td>21</td>
<td>21</td>
</tr>
</tbody>
</table>

and the y-axis denote the proportion of undetected true positives:

\[
p_f = \frac{S_f}{S_a}
\]  

(6.5)

where \( S_u \) is the number of undetected true shot boundaries, \( S_f \) is the number of falsely identified ones and \( S_a \) is the number of actual shot boundaries.

For comparison purposes, we consider the equal error rate as our performance criteria. Regardless of this, some applications may have different requirements such as the minimal number of undetected true positives, at the expense of a higher false positive rate if need be.

6.5.2 Comparison with non-adaptive results

We have discovered that the adaptive thresholding methods have shown significantly better results for all the algorithms concerned. This is very much evident for all the methods that we employed. This is demonstrated in Fig. 6.4 where the results for the AIM and ME algorithms and a selection of the methods are shown.

6.5.3 Effects of varying the window size

We initially used window sizes of 9, 11, 15, 21, 25 and 29. If the results show a simple trend, we test all of the window sizes between the two best performing ones
6.5. Experimental Results

Figure 6.4: Comparison of non-adaptive measures and a selection of adaptive methods.

and take the best of these as the optimum. Otherwise, we test all of the window sizes starting from 3 to 31.

For the most part, increasing the window size would increase the accuracy of the shot detection, after which the performance will decrease. This is not a universal behaviour, however. The LH method, for example, is happiest with a small window size, for almost all the methods; generally getting worse as the size increases. This is also demonstrated by HC for certain methods. Another case would be for LH using the DD method where the results are fairly constant until the window size is large, where it gets worse. ME using DD and SF, in the other hand, shows erratic behaviour, again stabilising once the window size gets larger.

6.5.4 Determining the best performing adaptive thresholding method

After we had obtained the optimal window sizes for all the possibilities, we proceeded to comparing the results. First, we plot the 6 adaptive methods for each algorithm and select the best one as shown in Fig. 6.5. Then we plot these “best of method” results against each other to come up with the best overall (Fig. 6.6).

As shown in Fig. 6.6, if we base the winning criteria to be the equal error rate, then the best performing method would be between MESA21 and HCSM9. Upon closer
inspection, MESA21 is the better of the two. This came about due to the higher false detection rate of HCSM9. However, if we want to minimise the undetected true positive rate, then HCSM9 would be better. MESA21 suffers from having a large window size which causes it to miss cuts which are within the window. Our experiments have shown that there are a few of these cuts occurring within our experimental data, mainly during commercials.

In Figs. 6.7 and 6.8, we give examples of instances where a true positive was undetected by MESA21 and HCSM9 respectively. In the former case, we note that there is some blurring due to excessive camera zoom on the shot left of the cut. This gave a high error rate for our motion prediction as shown in Fig. 6.9(a). For comparison,
6.5. Experimental Results

Figure 6.6: ROC curves of best performing algorithm and adaptive thresholding methods

we show the transfer response for the HCSM9 case in Fig. 6.9(b).

In the latter case (Fig. 6.8), the histogram distribution response graph shows a very small peak during the shot cut. This is demonstrated in Fig. 6.10(b). This is mainly because the luminance and colour distribution between the two shots are very similar. MESA21, on the other hand, detected this cut.

6.5.5 Other sequences

We tested our earlier methods on the animated sequence and the sports program and discovered that the results were very poor as shown in Fig. 6.11. It was then interesting to see if the adaptive thresholding models that we have would improve the results such that they would be of the same level as our main data set, the General Sequence.

We went through the same procedure as for the main data set and determined the best performing models for each algorithm. We compare these models with the best performing non-adaptive method for each of these sequences. In the case of the animated sequence, Cartoon, the best non-adapted method is ME and for the sport sequence, Rugby, HC. As shown in Fig. 6.12, there has been considerable improvement in the detection rate for both these sequences.
These two sequences are different from our main data set. In the case of the animated sequence, the content is synthetic. Also, in the sequence that we had, the maximum number of distinct colours was only 256, which we believe to be the norm for television cartoons. We have also noticed that on occasions where there is relatively little action going on over the course of a shot, the frames were repeated once over every frame before returning to the normal 25 frames per second rate. Such is the nature of animated characters.

For the sports sequence, the main feature was the presence of busy camera movements and high speed action. Also, in this sequence, there were numerous shots of objects moving across the camera whilst it (the camera) was focused at goings on further ahead in the distance as demonstrated in Fig. 6.13.

For both these sequences, the threshold $T$ which produced the best results turned out to be higher than that used for the General Sequence. This is mainly due to these sequences generally being more active, by its very nature.

### 6.6 Conclusion

We have shown that adaptive thresholding considerably improves the rate of detection for shot cuts regardless of the method used. In some cases, the improvements
It is found that the adaptively thresholded versions of ME and HC are the best performing algorithms. The ME based model achieved a marginally better equal error rate for the general sequence but is limited in its true positive detection rate due to the large window size required to achieve this. In this respect, the HC based model is better.

Interestingly, for the Cartoon and Rugby sequences, there was one algorithm which was clearly better than the others, which was ME for Cartoon and HC for Rugby. This meant that the adaptive thresholding versions of these algorithms were also the best performers for the respective sequences. It also indicates that certain algorithms are better suited for some sequences than others.
Chapter 6. Adaptive Thresholding Methods

Figure 6.9: Response graphs for MESA21 and HCSM9. On 6.9(a), the first trace, ME, is the algorithm results; the second trace is the results of Eq 6.1. In the case of 6.9(b), the first trace is again the algorithm results; the second trace is the results of Eq. 6.2.

Figure 6.10: Response graphs for MESA21 and HCSM9. The traces are the same calculations as those in Fig. 6.9
6.6. Conclusion

![Figure 6.11: ROC curves for the non-adapted algorithms.](image)

![Figure 6.12: Comparison between the best non-adapted method against the adaptively thresholded methods.](image)
Figure 6.13: Occlusion effects
Chapter 7

Detection of Gradual Shot Transitions using Temporal Filtering

7.1 Introduction

In the previous chapters, we presented methods for detecting cuts using first, a combination of multiple experts and second, localised adaptive thresholding for the individual experts. In this chapter, we set upon tackling the problem of detecting gradual transitions.

As noted in Chapter 2, the detection of gradual transitions represent a different set of issues compared to the detection of cuts. Foremost of these is the fact that there are a variety of effects which would fall under the heading of gradual transitions. Some researchers have tackled this problem by only considering a subset of the editing effect used to produce gradual transitions, namely wipes, fades and dissolves.

Also in Chapter 2, we mentioned that in previous comparison studies that have been published, the results of the gradual transition algorithms reviewed had not been encouraging. We noted that in [Lie99], the author produced some measure-
ment results and the tested gradual transition algorithms did poorly – in one case for the edge change ratio, for a true positive detection rate of 66.67%, the false positive rate recorded 37100% higher than the number of actual transitions in the test data. With this in mind, we concentrated on developing a method that would produce a fairly effective gradual transition detector whilst not compromising the precision too much.

In this chapter, we present a framework upon which we construct a gradual transition method that attempts to identify gradual transitions regardless of which subset it belongs to.

In Section 7.2, we describe our application of an existing temporal filtering technique to gradual transition detection, based on the observation that a shot boundary represents a temporal edge, and that this edge can be approximated by an optimal ramp edge model. In Section 7.3 we discuss the implementation of this technique. We present our experimental results in Section 7.4, and our conclusions in Section 7.5.

7.2 The gradual shot transition detector

We approach the problem of detecting gradual shot transitions by making certain observations on the nature of shot boundaries. A shot boundary can be considered as an edge in the temporal domain. A cut is an abrupt, sharp edge whereas a gradual transition can be looked at as a slowly varying edge, e.g. a ramp edge. In [YL95], the authors defined a gradual transition from shot \( X \) to shot \( Y \) as:

\[
(1 - \frac{t}{T})X + \frac{t}{T}Y, \quad 0 \leq t \leq T. \tag{7.1}
\]

with fade-in being a special case of \( X = 0 \) and a fade-out, \( Y = 0 \). This is a linear model describing an ideal transition from one pixel intensity value to another. A wipe is a different case where the transition from one shot to another is along a line across the frame with the shots being on either side of the line. This line itself can be of any shape and lead in from any point in the frame.
With this in mind, we propose the use of an edge filter in the temporal direction to detect the shot boundaries. We first apply a temporal filtering method on the input data and then analyse the filtered results. The following subsections describe our approach to temporal filtering and the analytical method used to evaluate the results.

### 7.2.1 Canny's criteria

In [Can86], Canny formulated three separate criteria for good extrema detection for a 1D signal \( e(t) \), namely the good detection criterion (maximum SNR):

\[
S_c = \frac{\left| \int_{-\omega}^{\omega} g(t) e(-t) dt \right|}{E(\eta^2) \sqrt{\int_{-\omega}^{\omega} |g(t)|^2 dt}} \tag{7.2}
\]

the localisation criterion:

\[
L_c = \frac{s^2 \int_{-\omega}^{\omega} g(t) e'(-t) dt}{E(\eta^2) \sqrt{\int_{-\omega}^{\omega} |g(t)|^2 dt}} \tag{7.3}
\]

and the suppression of false response:

\[
C = \frac{1}{\omega} \sqrt{\frac{\int_{-\omega}^{\omega} |g'(t)|^2 dt}{\sqrt{\int_{-\omega}^{\omega} |g''(t)|^2 dt}}} \tag{7.4}
\]

where \( \omega \) is the filter width and \( s \) is a scaling factor. In Canny's original paper [Can86], and subsequent development [Der87], the function \( e(t) \) is assumed to be a step edge. However, these criteria are suitable for optimising the detection of any arbitrary feature \( e(t) \) with a fixed convolution kernel \( g(t) \). Given this observation, in [PK91] an optimal edge detector for ramp edges was proposed. In their work, the authors assume that the profile of the ramp edge can be modelled by the function:

\[
e(t) = \begin{cases} 
1 - e^{-st}/2 & \text{for } t \geq 0 \\
e^{st}/2 & \text{for } t \leq 0
\end{cases} \tag{7.5}
\]

where \( s \) is some positive constant, describing the slope of the ramp which has dimensions \([\text{length}]^{-1}\).
7.2.2 The temporal filtering

Based on the work developed in [Can86], [Der87] and [PK91], we can now propose a temporal filtering approach to shot boundary detection.

In [PK91], the authors went through the derivation of the filter given certain preset parameters, one of which is the slope, $s$, of the ramp that we mentioned above. We note that it is not possible to have a prediction of what the slopes would be for every pixel position in the frames, considering the variety of video content and editing methods available. Furthermore, in practice a gradual transition is not necessarily linear over time, nor is it necessarily uniform for all the pixels in the frame. However, as a baseline we make the initial assumption that:

1. All the pixels are uniformly changing from one intensity to another according to a ramp profile and
2. The ramp profile fits the profile modelled in Eq. 7.5.

Based on these assumptions, we apply the filter to every pixel in the frame in the temporal direction, as illustrated in Fig. 7.1. To generate the filter coefficients, we use the suggested default parameters as laid out in [PK91] and calculated our coefficients for various tap sizes (Table 7.1 and Fig. 7.2).

![Figure 7.1: Temporal filtering filter](image)
Table 7.1: Filter coefficients

<table>
<thead>
<tr>
<th>Size</th>
<th>Coefficients $c_0, c_1, \ldots, c_{n-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>$-0.9611, 0.0, 0.9611$</td>
</tr>
<tr>
<td>5</td>
<td>$-0.6457, -0.9656, 0.0, 0.9656, 0.6457$</td>
</tr>
<tr>
<td>7</td>
<td>$-0.4287, -0.9411, -0.8524, 0.0, 0.08524, \ldots$</td>
</tr>
<tr>
<td>9</td>
<td>$-0.2913, -0.7626, -0.9994, -0.7538, 0.0, \ldots$</td>
</tr>
<tr>
<td>11</td>
<td>$-0.2098, -0.6029, -0.9203, -0.9838, \ldots$</td>
</tr>
</tbody>
</table>

Figure 7.2: Plot of the filter coefficients
Chapter 7. Detection of Gradual Shot Transitions using Temporal Filtering

The output of the 5-tap filter is shown in Figs. 7.3 (no shot change) and 7.4 (gradual transition). If the frames are within the same shot, the output of the filter would be largely zero (Fig. 7.3(f)). For a gradual transition, this depends on the rate of change of the video content. A fairly rapid dissolve would produce a more visible output in filtered frame, a slow dissolve, much less. In the example we give in Fig. 7.4, the dissolve takes place within 11 frames. By extension, we could consider a cut as a special class of gradual transitions, with a duration of 1, and correspondingly, the output of the filter when applied to a cut would be even more visible.

(a) (b) (c) (d) (e)

(f)

Figure 7.3: Original frames and filtered frame – for frames within the same shot

7.2.3 Absolute sum of pixel values as a measure

Once the filtered images are obtained, we need a method by which to analyse them. As shown in Fig. 7.5 a wipe would show as a stripe in the shape of the wipe’s border. Either side of the border, the intensities are close to zero. For a dissolve, there would be a gradual increase in the filtered images’ intensity until the approximate centre of the dissolve and followed by a reduction.
7.2. The gradual shot transition detector

Figure 7.4: Original frames and filtered frame – dissolve

Figure 7.5: Original frames (a) and filtered frames (b) – wipe
The examples in Figs. 7.3 to 7.5, suggest that a measure of the change in video content could be obtained by taking the absolute values of each pixel in the frame. To further refine our measurements, we apply an adaptive thresholding method, similar to the cut detection method in [YCK00]. Fig. 7.6 compares the absolute sum of the pixel values for the filtered frames with the adaptive threshold levels at each frame. We show three shot boundaries, labelled A, B and C. Shot boundary A is the wipe shown in Fig. 7.5 and B and C are two following cuts. Note that the magnitude of the wipes and cuts are very different, but the use of adaptive thresholding compensates for this.

![Image](image_url)

Figure 7.6: Plot of absolute sum of the pixel values for the filtered frames and an adaptive threshold. In the plot, A is a wipe, B and C are cuts.

### 7.3 Implementation considerations

Since a gradual transition detector has to be more sensitive to slowly changing content, it also needs to be able to discern between a true gradual transition and that due to other effects, mainly motion. We implemented a global motion estimation process and compensated for motion for all the frames forwards and backwards of the centre frame within our filter width. The obvious limitation of this is its inability to compensate local motion. To this end, we also implemented a block-based motion estimation algorithm. We used the \(n\)-step motion estimation method detailed in [Tek95], and our results have shown an improvement on the uncompensated...
7.3. Implementation considerations

Motion compensation added a considerable computational complexity to the processing, considering the fact that for each time step, the compensation has to be done $W - 1$ times for a tap size of $W$. Also, in our implementation, we used the centre frame as the reference frame and compensated forwards and backwards. Further improvements could be made in this aspect to achieve better estimation, but the investigation of more robust motion estimation/compensation methods are beyond the scope of this work and therefore we only present the results based on the available method here.

Since gradual transitions involve both a change in intensity and colour, another consideration is the use of colour information in our experiments. To this end, we constructed our experiments using both the luminance information only and the RGB colour channels. In the case of RGB colour, the absolute sum of all three channels for each pixel is added together.

In Chapter 5, we generated Receiver Operating Characteristics (ROC) curves for the purpose of visualising our results. In that process, the horizontal axis of the graph represented the proportion of false positives $S_f$ over the number of true shot cuts $S_t$, denoted as $P_f$. On the vertical axis, it was the proportion of undetected true positives $S_u$ over the number of true shot cuts $S_t$, denoted as $P_u$. The advantage of representing our results in this way is that it presents us with a range of operating points which we could apply for our applications. Furthermore, on the ROC curve we can observe that as a cut detector approaches perfection (i.e. 100% detection rate and 0% false positive rate) it would have a point approaching zero for both axes on the curve.

In detecting shot cuts, we ignored any gradual transitions that were detected and in effect treated the test data as containing only cuts. Our previous approach in calculating the ROC curves are modified for the inclusion of gradual transitions in our analysis. As previously stated, the temporal filtering method also detects cuts. Therefore, the proportion of false positives detected by the algorithm can now be
expressed as:

\[ P_f = \frac{S_f}{S_{tc} + S_{tg}} \]  \hspace{1cm} (7.6)

and the proportion of undetected true positives as:

\[ P_u = \frac{S_{uc} + S_{ug}}{S_{tc} + S_{tg}} \]  \hspace{1cm} (7.7)

where \( S_{uc} \) and \( S_{ug} \) are the number of undetected true positives for cuts and gradual transitions respectively, \( S_{tc} \) and \( S_{tg} \) are the number of true cuts and true gradual transitions.

As we shall explain later, using a combined measure for both the cuts and the gradual transitions masks the performance of the algorithm in detecting gradual transitions since

- The detection of cuts is easier
- The number of cuts in a video sequence is considerably higher than gradual transitions

If the performance of the algorithm as a unified shot change detector is used and we are concentrating on its gradual transition performance then, noting that the number of false positives is the same whether we are detecting cuts or gradual transitions, our analysis of the performance of various configurations of our temporal filtering method would have the horizontal axis as in Eq. 7.6 and the vertical axis as:

\[ P_u = \frac{S_{ug}}{S_{tg}} \]  \hspace{1cm} (7.8)

On the other hand, if it is only to be considered as a gradual transition detector, then \( P_f \) should be calculated as:

\[ P_f = \frac{S_f}{S_{tg}} \]  \hspace{1cm} (7.9)

For the analysis of the results that we present in Section 7.4 below, our ROC curves will be using Eq. 7.8 for the vertical axis and Eq. 7.9 for the horizontal axis except where noted.
As a comparison, in [Lie99], the author defines the hit rate $h$ as the ratio of correctly detected shot boundaries and the miss rate $m$ as the ratio of missed shot boundaries to the actual number of shot boundaries, i.e. $1 - h$. The false hit rate $f$ is the ratio of falsely detected shot boundaries to the actual number of shot boundaries. In the context of our performance measures, then $P_u = m$ and $P_f = f$.

7.4 Experimental Results

We arrange our experiments as follows: we start first by evaluating the results for a variety of filter sizes. Then, we compare using the luminance information only with RGB colour. Next, we consider how they performed compared with the adaptively thresholded histogram comparison method. We then conduct a failure analysis to examine the false positives that are detected and the undetected true positives. Finally we present our results.

In the following sub-sections, for brevity, we denote the temporal filtering method as TF, the measurement of the absolute sum of pixel values as ASPV and the histogram comparison method as HC.

7.4.1 Experimental Data

Our data set consists of several off air sequences consisting of news programs, documentaries, children’s shows, daytime soaps, etc., we aimed at capturing a broad variety of material. The breakdown of the individual sequences are given in Table 3.2 in Chapter 3. Table 7.2 details the composition of our test data.

The gradual transitions consisted of dissolves, fades, wipes and the odd special effects that defy categorisation. The lengths of the transitions range from 2 to 65 frames. We feel that our test data is fairly representative of what is broadcast today.
Table 7.2: Characteristics of the Video Sequences used in the experiments.

<table>
<thead>
<tr>
<th>Format</th>
<th>QCIF - 176 x 144 pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>YUV 4:2:0</td>
<td></td>
</tr>
<tr>
<td>Frame rate</td>
<td>25fps</td>
</tr>
<tr>
<td>No. of frames</td>
<td>Time (mins) No. of shot cuts No. of grad. trans</td>
</tr>
<tr>
<td>161928</td>
<td>108</td>
</tr>
</tbody>
</table>

7.4.2 Effects of filter size

We used the filter coefficients generated using the parameters determined in [PK91] for \(w = 2, 3, 4, 5\) and 6, thus giving us filters with 3, 5, 7, 9 and 11 taps respectively, where no. of taps = \(2w - 1\). Fig. 7.7 show the performance in detecting gradual transitions for the various filter sizes for TF using the RGB channels. The results of varying the filter sizes when applied to using the luminance information only display the same behaviour.

![Figure 7.7: Plots of detection rates as a function of the number of taps](image)

We have found that apart from the smallest filter sizes we used (3 and 5), the performance of the algorithms are fairly similar. The thresholds on equivalent operating points for each filter size are different with the thresholds being higher as the filter size is increased. Closer inspection reveals that higher tap sizes generate marginally better results though a considerable amount of cross-over amongst the curves is ev-
ident. However, the corresponding increase in computational load for higher filter
tap sizes may not justify this.

7.4.3 Comparison between using luminance information only (LUM) and RGB information

As mentioned previously, we conducted the experiments on the luminance informa­tion and the RGB channels. Fig. 7.8 shows the ROC curves for the RGB and LUM experiments with filter sizes of 7 and 11. Further analysis of the results revealed a pattern where the RGB method is consistently more insensitive to false positives than the LUM method with a higher detection rate. This applies for all the filter sizes that we tried. As a conclusion, we find that using colour information is better than just the luminance intensity information. Therefore, further discussion would concern the RGB experiments only.

![Figure 7.8: Comparison between LUM and RGB methods for filter sizes of 7 and 11](image)

7.4.4 Comparison between the TF method and the HC method

The cut detector that we use (HC) also occasionally detects gradual transitions; therefore we compare it with the temporal filtering method. Fig. 7.9 demonstrates the results of our experiments.

As shown in Fig. 7.9, the HC method is noticeably weaker at detecting gradual transitions compared with the TF method.
Figure 7.9: Comparison between the gradual transition detector and a cut detector using HC

![Graph comparing gradual transition detector and cut detector using HC](image)

Figure 7.9: Comparison between the gradual transition detector and a cut detector using HC

Figure 7.10: Plots of the methods at a section of the video where a dissolve occurs

(a) TF

(b) HC

Figure 7.10: Plots of the methods at a section of the video where a dissolve occurs

In Fig. 7.10 we show an example of the measurement values where a dissolve occurred and in Fig. 7.11, a wipe (A) followed by a cut (B). This wipe segment is the one shown in Fig. 7.5. In the case of the dissolve, both methods' measurements show a prominent peak. However, the adaptive thresholding that we employed for the HC method meant that this dissolve was not detected by the algorithm. For the wipe, we see that the HC measurement show a barely noticeable “hump”, especially in contrast with that exhibited by the TF method. The cut was detected by both algorithms.

As noted, the TF method also detects cuts. Performance wise, the TF method is
7.4. Experimental Results

Figure 7.11: Plots of the methods at a section of the video where a wipe (A) and a cut (B) occurs

fairly close to the HC method. This is demonstrated in Fig. 7.12.

Figure 7.12: ROC curves for TF and HC – cuts only

7.4.5 Failure analysis

Most of the false positives are due to excessive motion effects and illumination changes due to noise. This is covered in Section 7.3 above.

Another major contributory factor is the appearance of captions or credits. However, these false positives could be added as another detection class for shot boundary detectors and text recognition could be employed, for example, to specifically...
search for these instances.

As for the undetected true positives, they are due to long gradual transitions which vary in content too slowly or when the shots in either side of the boundary are too similar (this is a common problem for all shot boundary detectors, be it cuts or gradual transitions).

![Image](image.png)

Figure 7.13: False positive example. (a) – (e) are the original unfiltered frames and (f) – (j) are the filtered frames with a 7 tap filter

Fig. 7.13 give an example of a false positive. In this sequence, there was a person moving in front of the camera occluding the main focus of the shot. This is similar in effect to a wipe and we encounter false positives at similar events.

### 7.4.6 Final remarks

We used the histogram comparison technique as a comparison to temporal filtering because it is insensitive to motion, relying on the distribution of intensities in the frame. Furthermore, it is an example of a dissimilarity measure used for detecting changes between two consecutive frames. The temporal filtering method, on the other hand, attempts to detect the rate of change over a number of frames. Also,
by implementing our adaptive thresholding method, we improved the histogram comparison technique's insensitivity to false positives, and made it more robust to gradual changes as shown in Figs. 7.10(b) and 7.11(b). This makes it ideal for detecting cuts.

The increased sensitivity afforded by the temporal filtering method comes at a price, one of which is greater susceptibility at detecting motion-influenced activity as shot changes. By applying motion estimation and compensation, we have alleviated this problem to a certain extent. However, motion which exceeds our search window still causes some problems. We have found that by implementing an adaptive thresholding method as well on the temporal filtering, it can cope with sustained periods of high motion. Unfortunately, gradual transitions that are exceptionally slow or exhibiting very little change in intensity and colour are also missed.

As with any shot change method, we have to have a trade-off between efficiency (the capability of detecting true shot changes) and precision (the ability to detect only true positives).

In Table 7.3, we show a selection of the results of our experiments using the RGB channels with filter sizes of 7 and 11 along with HC. For the cuts and gradual transitions, the hit rate is defined as \((1 - P_u) \times 100\) and the false positive hit rate, \(P_f \times 100\).

<table>
<thead>
<tr>
<th>Method</th>
<th>Cuts</th>
<th>False Pos</th>
<th>Grad Trans</th>
<th>False Pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>RGB Taps = 7, (T = 1.99)</td>
<td>95.09%</td>
<td>5.45%</td>
<td>64.70%</td>
<td>32.62%</td>
</tr>
<tr>
<td>RGB Taps = 11, (T = 2.07)</td>
<td>92.95%</td>
<td>5.09%</td>
<td>64.17%</td>
<td>30.48%</td>
</tr>
<tr>
<td>HC</td>
<td>96.08%</td>
<td>1.15%</td>
<td>40.40%</td>
<td>8.53%</td>
</tr>
</tbody>
</table>

As we can see from Fig. 7.12, whilst the cut detection performance of the TF methods are almost as good as the HC method, they were unable to match the false positive rate of the latter. In the calculations we made for Table 7.3, the number of false positives is the same since the detection of shot boundaries was done on
the same test sequence. However, as we noted earlier, since the number of cuts is greater than the number of gradual transitions, the false positive hit rate would be greater when compared to the number of gradual transitions.

In view of this, a possible strategy for constructing a complete shot boundary detection system would be to use the HC method to first extract the cuts; then once these are extracted, the TF method is then employed. Our experimentation has shown that the false positives detected by the HC method are also detected by the TF method so we would not be left with a higher false positive count than those shown in Table 7.3.

7.5 Conclusion

We have described a new technique for detecting gradual transitions in video sequences. This technique shows a considerable improvement compared to a previous optimised histogram comparison technique at the expense of a higher false detection rate.

While the detection rate for gradual transitions may not be as high as that for cuts, it compares very well with previous published results and we also show a much reduced false positive rate.

We have found in our experiments that various motion effects tend to display the same characteristics of gradual transitions and this constitutes the majority of the false detections that we had. Further work into more robust motion estimation and compensation would be carried out to overcome this.

The use of the absolute sum of the pixels is perhaps not the optimal way of generating a measurement to detect shot boundaries. More research into spatial analysis methods would be undertaken to investigate other possibilities.

Further research into determining the parameters for deriving the filter coefficients should be carried out as well as the possibility of using different filters, e.g. wavelet based. Furthermore, an investigation into methods to characterise possible edge characteristics in the frames could lead to the application of adaptive filtering.
Finally, this work represents a preliminary research into gradual transition detection and we believe that there is yet further scope for optimisation, both in the detection rate and error rate.
Chapter 7. Detection of Gradual Shot Transitions using Temporal Filtering
In Chapter 4 we showed some baseline shot cut detection algorithms which are used as the basis of our proceeding experiments on shot boundary detection. We implemented these algorithms initially on a set of short video sequences before generating experimental results on longer durations of video sequences. We identified the shortcomings of these standalone algorithms and explored a method whereby a combination of these algorithms could be used to provide better results. This was presented in Chapter 5.

Noting that the shot cut algorithms thus far discussed utilised single hard thresholds upon which the boundary decision was made, we looked into implementing an adaptive thresholding method on the baseline algorithms. In Chapter 6, we experimented with several adaptive thresholding methods on our extensive data set and presented our results and conclusions as to the best performing ones.

Next we turned our attentions to gradual transitions in video sequences in Chapter 7. We postulate that a shot boundary represents a temporal edge, and that this edge can be approximated by an optimal ramp edge model. We proposed the use of an edge filter in the temporal direction to detect the shot boundaries. We presented our experimental results and showed that the performance of the detector was a
major improvement on previously published methods.

Future work would involve greater investigation into clustering methods for grouping together cuts for creating coherent snapshots of scenes within a video sequence. Further, investigation into shot boundary detectors which are specific to different types of video sequences (drama, news, sports) would be useful. In the current study that we have done, we have created two separate entities by which to detect cuts and gradual transitions. Further exploration into the integration of the two systems is merited, taking into account the work that was presented in [Han02]. In that work, the author has constructed a shot boundary detector that attempts to unify the two separate classes of shot boundaries together. He concentrated on cuts and dissolves only so a further extensive work into its performance for other gradual transition effects should be looked into.
Bibliography


Bibliography


