Video Segmentation and Indexing using Motion Estimation

Sarah Victoria Porter

A dissertation submitted to the University of Bristol in accordance with the requirements for the degree of Doctor of Philosophy in the Faculty of Engineering, Department of Computer Science.

February 2004

39,000 words

Abstract

Video indexing is a central component necessary to facilitate efficient content-based retrieval and browsing of visual information stored in large multimedia databases. This thesis presents work towards a unified framework for automated video indexing. To create an efficient index, a set of representative key frames are selected which capture and encapsulate the entire video content. This is achieved by, firstly, segmenting the video into its constituent shots and, secondly, selecting an optimal number of frames between the identified shot boundaries. The segmentation algorithm is designed to detect both abrupt shot transitions, or *cuts*, and gradual transitions, such as *dissolves* and *fades*. This is achieved by means of a two-component frame differencing metric taking both image structure and colour distributions into account. The application of hierarchical block-based normalised correlation and local colour histogram differences leads to a method which is both accurate and robust.

After the segmentation step, the key frames are selected to minimise representational redundancy whilst still portraying the content in each shot. This is achieved by employing a graph-based representation of each shot where nodes represent frames and connection weights the amount of shared content between the frames corresponding to the connected nodes. The key frames are then selected as those corresponding to nodes present on the least weight path through the graph. As a final step, the camera motion is characterised to provide an additional layer of video annotation which may prove useful for indexing.

Declaration

I declare that the work in this dissertation was carried out in accordance with the Regulations of the University of Bristol. The work is original except where indicated by special reference in the text and no part of the dissertation has been submitted for any other degree.

Any views expressed in the dissertation are those of the author and in no way represent those of the University of Bristol.

The dissertation has not been presented to any other University for examination either in the United Kingdom or overseas.

SIGNED: DATE:

Acknowledgements

The work herein was funded in part by the UK EPSRC. I would also like to acknowledge the funding and support given by Pam Fisher, Moving Image Research Ltd, without which I could not have afforded my rent. Thanks also goes to the Department of Computer Science for providing a stimulating environment in which to work.

I would also like to thank my advisers, Majid Mirmehdi and Barry Thomas for believing in me in the first place.

I've received unfloundering support from my family, John, Sue and Allison—look, mum, I *can* do it! However, the question 'when are you getting a proper job' still remains to be answered.

I'd also like to extend my thanks to many members (past and present) of the Department of Computer Science, for endless patience when faced with my oft-uttered 'but why?', in particular Andrew Calway, Peter Flach and Nigel Smart. Your contributions are much appreciated. Thanks are also due to Colin Dalton, Nigel Jewell, Fré Vercauteren and Rob Thomas for a variety of reasons: artistic talent, code contributions, (ab)use of machinery and tireless IT support.

Special thanks go to Angus Clark—a fantastic friend and flatmate. I promise never to discuss work at home again. David Hedley and Kate Devlin for constant support and entertainment, especially in the Peak District and Georgina, I'm looking forward to swimming again. To Mark Everingham and David Tweed, who may have left, but made sure they wouldn't be forgotten—your impact on my progress cannot be overestimated. My thanks also to Oli Cooper and Annie Yao, lab mates, and for providing a good laugh.

Other special friends: Nana, Chris, Cath and Dave—you've made the last few years very memorable. I couldn't have done it without your support.

I must thank the England Rugby team for giving me a good excuse for taking Saturdays off work. Two long overdue triumphs in the same year—The World Cup, and my submission.

And almost finally, to Henk Muller who lost out on the last spot by a hair's breadth. Thank you for your constant inspiration and endless faith that I could actually do this. Your advice has been invaluable. I owe you big time.

And finally, to Stefan for all the patience and understanding that I could unreasonably expect. Thank you for all your help and those endless discussions about my work. Thank goodness you had been through this before—I couldn't have done it without you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Video Indexing

Indexing and annotating large quantities of film and video material is becoming an increasing problem throughout the media industry, particularly where archived material is concerned. Manual indexing is currently the most accurate method but it is a very labourious and time consuming process [42]. To catalogue material an archivist has to view many hours of film on a sequential basis to locate shot boundaries and textually annotate each individual shot. This means that considerable amounts of archived data remain unindexed. Apart from being slow to generate, the use of textual annotations for indexing and searching large databases can obviously lead to loss of information. Unless there is a well defined grammar to follow, it can be unclear to the person providing the annotation what level of detail the description should be. The drawbacks of existing facilities suggest an interface is required for browsing and searching video content that is easier and less time consuming to produce.

For film researchers, probably the most important development of the last few years is the increasing availability of on-line catalogues [21]. Previously, film researchers physically visited many of the available libraries to conduct their research, or assigned a member of staff to do it on their behalf [30]. Increasingly, this can be done remotely, thus saving both time and money. For instance, the on-line catalogue has proved vital for catalogues held abroad. Whilst the storage and delivery capabilities of computer systems are expanding to meet these demands, on-line catalogues can only replace an actual visit to a library if they are at least as capable as the old system [21]. To be a useful resource an on-line catalogue should:

- contain details of the whole database. Limited catalogues or those that offer a 'best of' or 'most popular' collection might not be useful for all enquiries;
- offer efficient indexing and querying tools. Older card indexes with the added back-up of staff who really know their collection can sometimes provide faster access, often with cross-referencing capabilities that put many search engines to shame [21].

One of the important developments of on-line catalogues is the availability of sites such as FOOTAGE.net[1] which provides access to hundreds of major *stock footage* sources worldwide without having to access them one by one. It provides a facility to search millions of shots in the combined on-line databases by matching words in the textual description for each shot.

Stock footage is a collection of film images that have been previously shot, and are reused in commercials, TV shows, corporate productions, etc. For example, if a travel company needs a shot of a sunset on an empty beach, instead of hiring a crew, renting equipment, travelling, shooting footage, developing the footage, re-shooting in case it does not turn out right, there probably exists a shot within a stock footage collection which can be used rather than the production of new film. One drawback of re-utilising existing material is the difficulty in searching and browsing such collections to find a suitable shot. The current state of the art for an on-line catalogue is to search for the keywords in the textual annotation for each shot. Searching for "sunset" and "beach" on FOOTAGE.net, for example, returns 2785 records in 25 databases. The description of shots can vary from

> 'Honolulu palm trees, *beach* and *sunset* sky' to
>
> 'Two men galloping on *beach*/Three riders in *sunset* silhouette'.

Obviously, these descriptions are sufficient to know that the second shot may not be suitable if an empty beach is required and more words can be added to refine the search. However, there are many other shots with descriptions such as 'Two palm trees on *beach* with tropical *sunset* sky' that might be equally suitable. At this stage, a film researcher cannot view the shots and will have to pay for a custom viewing cassette which is a compilation of all shots they are interested in. Although not always cheap it is often less expensive than shooting new film. If a film researcher was on-site in a library they could view the individual shots but many libraries charge for using such facilities and it is a time consuming process to locate each shot to view. Hence, the use of multimedia information, on-line or in a library, is inhibited by the inability to browse and search the video content efficiently.

[1] http://www.footage.net

The predominant approach to automate the video indexing process is to create a video abstract. A video abstract is defined as a sequence of images extracted from a video, much shorter than the original yet preserving its essential message [59]. As well as being less time consuming to produce than a textual annotation, a visual summary to be interpreted by a human user is semantically much richer than a text. A video abstract can then be used to index video that so far has not been catalogued, as browsing a shorter sequence of images is quicker than watching a whole film. It can also be used in conjunction with an existing textual annotation to augment the searching process. In the example above, searching for a sunset on a beach, a video abstract would give the film researcher an indication of the visual content of each shot and aid in the process of deciding which shots would be suitable, resulting in fewer shots being ordered on the custom viewing cassette. In addition, once a shot has been found that appears to be suitable, the images in the video abstract can be used to retrieve similar shots (retrieval by example) [1, 5, 12, 75].

The difficulty in composing such an abstract is determining which frames best represent the video contents. The solution to this problem often depends on the context of the application in which the abstract is being used. Lienhart et al. [59] concentrated on the generation of trailers for movies resulting in abstracts that are short and designed to attract the attention of the viewer without revealing too much of the storyline. In contrast, an abstract for a documentary or a digital video library should try to represent all the video content [68]. The present work focuses on generating a video abstract for the purpose of video indexing and retrieval.

In order for a video abstract to be an efficient index, each key frame should represent a video segment where there is little or no significant change in the scene content. This enables a video abstract to encapsulate and capture the content of the sequence whilst removing the visual content redundancy amongst video frames [43, 91]. There exists a hierarchical structure within a video sequence, as illustrated in Fig. 1.1, which can be exploited in the extraction of such key frames. At the lowest level it consists of a set of frames. At the next level frames are grouped together to form *shots*, defined as a sequence of frames that were captured continuously from the same camera operation. Shots unified by a common locale or event are then grouped together into *scenes*. These scenes then complete the video sequence. Once the video sequence has been broken down into a set of meaningful and manageable segments (shots), work can be done to characterise the individual components for indexing and annotation. Therefore, temporal segmentation of a video sequence is typically the first step towards automatic annotation of digital video sequences [45, 13].

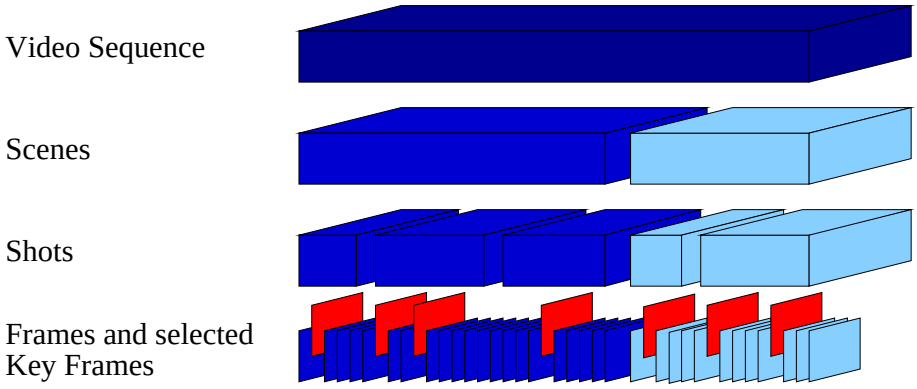


Figure 1.1: Hierarchical structure within a video sequence.

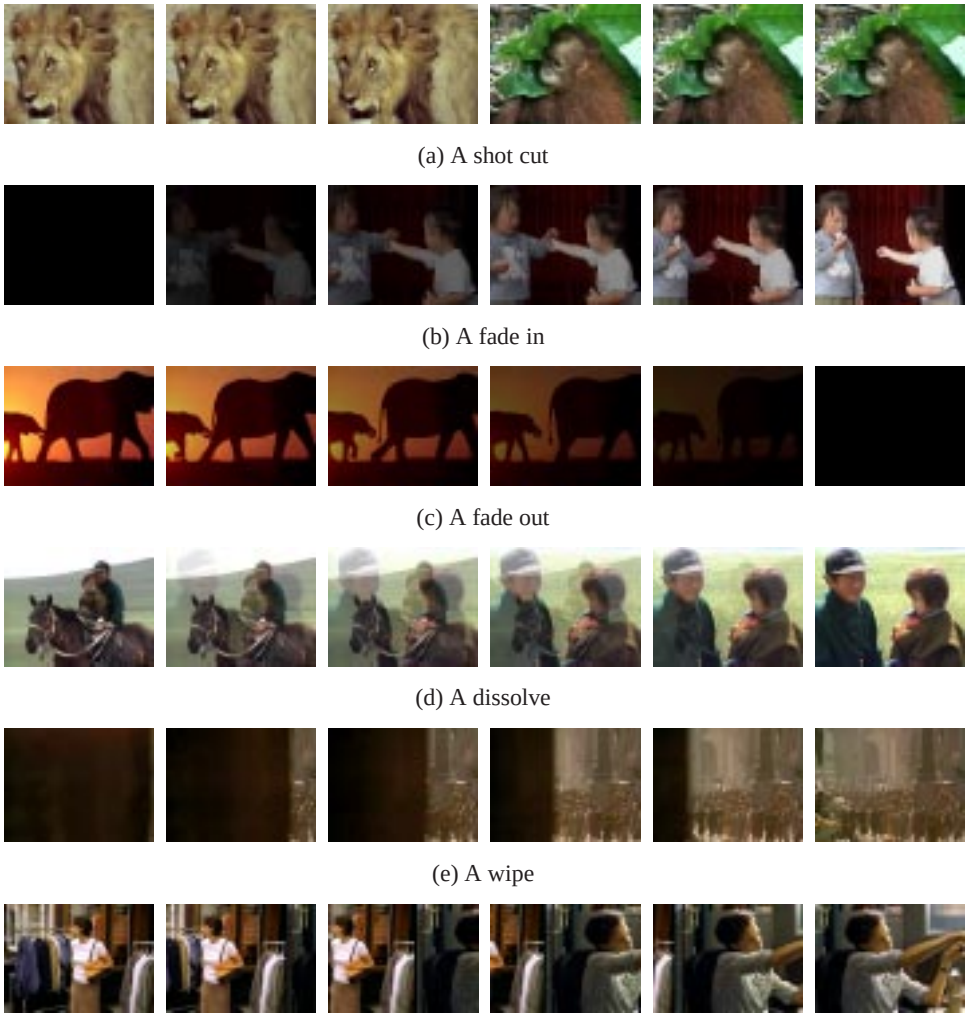
1.2 Shot Transitions

Shot transitions are time between camera shots that lead the viewer from one shot to another and are added during post-production. The connections between shots are as important as the shots themselves with all transitions showing a change of time or space [50, 47]. There are two different types of transitions that can occur between shots: abrupt (discontinuous) shot transitions, also referred to as cuts; or gradual (continuous) shot transitions, such as fades, dissolves and wipes or pushes. These transitions can be defined as follows:

- *cut*: an instantaneous change from one shot to another;
- *fade-in*: a shot gradually appears from a constant image;
- *fade-out*: a shot gradually disappears to a constant image;
- *dissolve*: the current shot fades out while the next shot fades in;
- *wipe*: the next shot is revealed by a moving boundary in the form of a line or pattern;
- *push*: the next shot pushes the previous shot off the screen to the left, right, up or down.

Examples of such shot transitions are shown in Fig. 1.2. There are hundreds of different types of pushes and wipes, and all are considered to be special effects. A couple of variations of the wipe transition are illustrated in Fig. 1.3. Detection of all the categorised transitions will segment a video sequence into its individual shots, each representing a different time or space, ready for further higher-level processing to characterise it.

Most people have watched innumerable hours of television and/or film during their lifetime and share an implicit film/video ‘grammar’, particularly when it comes to shot transitions [44]. For



(a) A shot cut

(b) A fade in

(c) A fade out

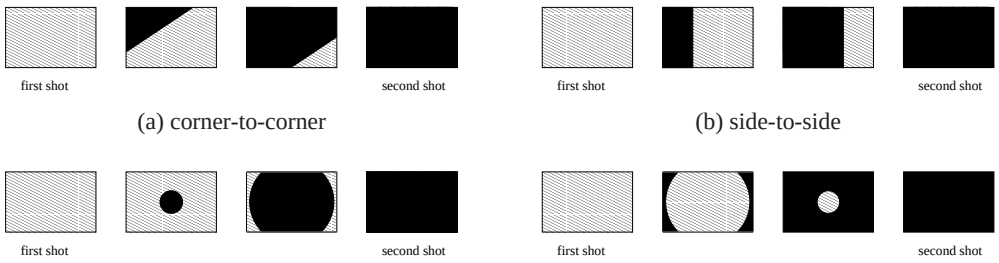
(d) A dissolve

(e) A wipe

(f) A push to the left

Figure 1.2: Examples of different types of shot transitions.

example, a dissolve from one camera shot to another usually means a relatively short amount of time has passed. Producers use this implicit grammar to help viewers understand the video and violating the grammar will frustrate the viewer's expectations. The cut is the simplest, most common way of moving from one shot to the next. It is considered a 'smooth cut' if there is continuity between the two images, for example, in an interview scene, if the cut moves the observer between the interviewee and the interviewer. The dissolve (also known as a cross dissolve) influences the audience's perception of screen time and the rhythm of events. It suggests a thematic tie between two shots. For example, a dissolve might be used to shorten a long action such as an air plane flight and to shift from the take-off location to the destination of the same flight. A fade denotes the beginning or end of a scene, episode or idea. Fades often imply a more important change of place or passage of time than a dissolve. Due to this film grammar being used consistently, the



(a) corner-to-corner (b) side-to-side

(c) centre-out (d) boundary-in

Figure 1.3: Illustration of a variety of wipe transitions.

most common edit effects found in video sequences are cuts, fades and dissolves. In fact, the data set used in this work contains video sequences from several different genres, including sport, film, documentary, drama and comedy series and in total contains 757 cuts, 94 dissolves and 104 fades over more than 94000 frames. For this reason, the majority of previous work and the work presented here focuses on detecting only these types of transitions.

1.3 Aims and Objectives

The considerable amount of video data in multimedia databases requires sophisticated indices for its effective use [13]. Manual indexing is currently the most effective method to do this, but it is slow and expensive. For this reason there is a need for automated methods to annotate video sequences and provide a content description according to the user's needs, by allowing efficient access to the video data without viewing the material in its entirety. Users can be interested in a range of video characteristics ranging from its decomposition into shots and scenes, to the most representative frames or sequences, the camera work (e.g. panning, tilting, zooming), the identity of the people that appear in it, and their movements and dialogues [13].

Based on this observation, the work presented here investigates the video indexing task. The objective is to extract a subset of representative key frames from a video sequence to enable content-based video browsing and retrieval. The work is performed in the domain of uncompressed video and aims to detect shot transitions to segment temporally a video sequence into individual shots before extracting key frames. The motivation for addressing this problem is to establish efficient authoring and querying tools to promote full exploitation of on-line and archived video data.

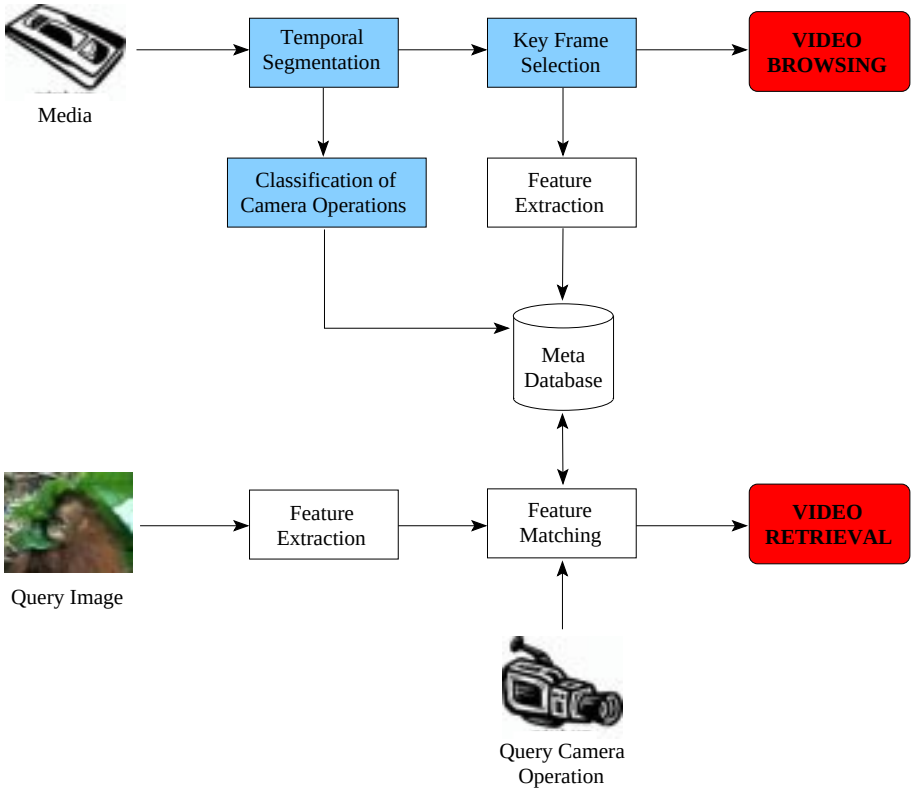


Figure 1.4: A diagram of an automated video indexing system.

1.4 Contributions

The diagram in Fig. 1.4 outlines an automated video indexing system which temporally segments a video sequence into shots and selects representative key frames. These key frames can then be used to browse the video content or extracted features can be used to match video content to a users query to enable shot retrieval. The main contributions are highlighted in blue in Fig. 1.4 and can be outlined as follows.

1. A novel method for detecting abrupt shot transitions which uses block-based correlation coefficients and histogram differences to measure the visual content similarity between frame pairs. Hierarchical motion compensation is employed to ensure the algorithm is robust in the presence of camera and object motions. It is shown to achieve a higher recall and precision compared with five commonly used techniques. This is an extension of the work first published in [69].

2. A novel method for detecting gradual shot transitions which measures the difference between the visual content of distant frames using the same features as in the shot cut detection

algorithm, first reported in [70, 72]. Motion estimates obtained from the shot cut detection algorithm are used to track regions of interest through the video sequence. This enables the distinction between content changes caused by gradual transitions and those caused by camera and object motions.

3. A method which, by taking explicit account of camera motion estimates and adopting a graph-based approach, extracts representative key frames for each shot identified by the shot transition detection algorithm. This creates an efficient index into the video data. This work builds on that originally published in [71].

4. The use of a line simplification algorithm to characterise the apparent camera motions contained within each shot to provide a further level of annotation, originally presented in [73].

1.5 Thesis Outline

The thesis is organised as follows.

Ch. 2 provides a detailed review of previous approaches to shot transition detection and video indexing. It highlights their relative merits and shortcomings which sets the context for the present work.

Ch. 3 describes the hierarchical motion and histogram based approach to detecting abrupt transitions. Correlation coefficients and histogram differences are used to generate a motion compensated measure of content similarity between two frames.

Ch. 4 describes an extension of the shot cut detection method to detect fades and dissolves.

Ch. 5 describes a method that uses a graph-based approach to extract representative key frames to create an efficient index into video data. A method is also described to classify different types of camera motions.

Ch. 6 concludes the thesis with a synopsis of the presented work and a summary of the relative merits and shortcomings of the approach. Directions are also given as to where future research effort could be spent to enhance and improve on the work.

Chapter 2

Background

The purpose of this chapter is to present a brief review of previous work relating to the problems of shot cut detection, gradual transition detection and key frame selection. The methods to be outlined have been chosen as it was felt they were representative of the main approaches that have been used to tackle these problems so far. This review also motivates and presents a platform for the current work. Any algorithms that are used for comparative purposes are presented in more detail in the relevant sections. Furthermore, a summary of the key issues faced when tackling each problem is given at the beginning of each corresponding chapter.

2.1 Shot Cut Detection

This section summarises previous approaches to shot cut detection. In the case of a shot cut, one image is immediately replaced by another. Therefore, the aim of any shot cut detection method is to select some feature related to the visual content of a video such that:

- any frames within the same shot exhibit similar properties, and
- frames belonging to different shots would have dissimilar feature characteristics.

Most of the existing methods use some inter-frame difference metric. A frame pair where this difference is greater than a predefined threshold is deemed to contain a shot cut. There have been many different features and metrics proposed for the purpose of shot cut detection and these have

been analysed in several comparative studies [8, 61, 96, 58]. These features and metrics have been subtly modified in a variety of ways. For this reason, only the main approaches are presented here.

Pixel-Level Comparisons

Arguably, the simplest way to quantify the difference between two frames is to compare the intensity values of corresponding pixels. If the mean absolute change in the intensity value of the pixels is greater than a threshold T_{cut}, a shot cut is deemed to be present between two frames [51]. Given two frames f_{n-1} and f_n, this can be defined as

$$\frac{\sum_{\boldsymbol{p}} |f_{n-1}(\boldsymbol{p}) - f_n(\boldsymbol{p})|}{w \cdot h} \begin{cases} > T_{cut} & \text{shot cut} \\ \leq T_{cut} & \text{no shot cut} \end{cases} \tag{2.1}$$

where $f_n(\boldsymbol{p})$ is the intensity value of pixel $\boldsymbol{p}$ in f_n. A potential problem with this approach is its sensitivity to camera and object motion. Using the mean absolute change it is unable to distinguish between a large change in a small area and a smaller change in a large area. Thus, the presence of large object motion can lead to the false detection of a shot cut.

An improvement to this approach was proposed by Zhang et al. to determine the percentage of pixels that have changed considerably between two frames [99]. This approach is known as the pair-wise pixel comparison. A pixel is deemed to have changed considerably if its difference is greater than a given threshold. A shot cut is then declared present if the percentage of changed pixels is greater than a second threshold. Although an improvement, this approach is still sensitive to object and camera motion. For example, a camera pan could cause the majority of pixels to appear significantly changed. To reduce further the influence of motion, it was suggested a smoothing filter should be applied to the images before the comparison.

Global-Level Comparisons

In an attempt to further overcome the problem of camera and object motion, instead of comparing individual pixels alternative approaches were proposed that compare global features of each frame. The average intensity measure takes the average value for each RGB component in the current frame and compares it with the values obtained for the previous and successive frames [39]. Although less sensitive to motion than pixel-level comparisons, two shots with different colour distributions can have similar average intensity values resulting in a missed detection.

A different approach is the comparison of global histograms. This method is based on the assumption that two frames with an unchanging background and unchanging objects will show little difference in their corresponding histograms. This approach should be less sensitive to motion than the pixel-level comparison as it ignores changes in the spatial distribution within a frame, but herein also lies its weakness. There can exist two neighbouring shots with similar histograms but entirely different content, resulting in a difference measure similar to that caused by camera and object motion. This means that it can be difficult to detect all the shot cuts without also incurring false detections. However, histogram approaches offer a reasonable trade-off between accuracy and computational efficiency and are the most commonly used methods in use today.

Nagasaka and Tanaka proposed the comparison of the grey level histograms between two frames [63]. The histogram $H_n(k)$ is obtained by counting the number of pixels in frame f_n with grey level k. The difference between two histograms is then determined using

$$DH_n = \sum_{k=1}^{K} |H_{n-1}(k) - H_n(k)| \tag{2.2}$$

where K is the number of grey levels. If DH_n is greater than a given threshold a shot cut is declared. In their experiments, Nagasaka and Tanaka used 64 different quantisation levels. However, they reported that the metric was not robust in the presence of momentary noise, such as camera flashes and large object motion [63]. A more robust measure was suggested to compare the colour histograms of two frames. The difference measure was still computed using (2.2) but with $H_n(k)$ obtained by counting the number of pixels with the colour code k. The authors proposed using a 6 bit colour code obtained by taking the two most significant bits of each RGB component resulting in 64 colour codes. To make the difference between two frames containing a shot cut be more strongly reflected they also proposed using the chi-square statistic (χ^2) which can be used to measure the difference between two binned distributions [74].

Many variations in calculating the histogram differences between two frames for the purpose of shot cut detection have been proposed. The histograms can be obtained using different colour spaces such as RGB, HSV, YIQ, Lab, Munsell and opponent colour axes, which have been used previously for image database retrieval by colour. Different comparison metrics have also been used such as the bin-to-bin difference defined by (2.2), the χ^2 statistic and the histogram intersection

$$INT_n = \frac{\sum_{k=1}^{K} \min(H_{n-1}(k), H_n(k))}{w \cdot h} \tag{2.3}$$

where $w \cdot h$ is the number of pixels in each frame [80]. The difference between two frames is then defined by

$$INTD_n = 1 - INT_n \tag{2.4}$$

An extensive comparison of different colour spaces and frame difference measures is given in [33, 61, 8, 22]. However, it has been shown that the simple comparison between RGB or YUV colour histograms with each colour component quantised to 2^b different values with usually b set to 2 or 3, is a simple but effective method for detecting hard cuts [63, 33, 56]. In fact, Lienhart suggested that there is little performance improvement to be gained by fine-tuning the colour space and difference measures.

Block-Based Comparisons

A weakness of the global-level comparisons is that they can miss changes in the spatial distribution between two different shots. Yet, pixel-level comparisons lack robustness in the presence of camera and object motion. As a trade-off between both of these approaches, Zhang et al. proposed the comparison of corresponding regions (blocks) in two successive frames [99]. The blocks were compared on the basis of second-order statistical characteristics of their intensity values using the likelihood ratio [49]. A shot cut was then detected if the number of blocks with a likelihood ratio greater than T_{diff} exceeds a given threshold T_{cut}. The number of blocks required to indicate a significant change to declare a shot cut obviously depends on how the frame has been partitioned.

Nagasaka and Tanaka also proposed dividing each frame into 4×4 regions and comparing the colour histograms of corresponding regions [63]. They also suggested that momentary noise such as camera flashes and motion usually influence less than half the frame. Based on this observation, the blocks were sorted and the 8 blocks with the largest difference values were discarded. The average of the remaining values was used to detect a shot cut. Ueda et al. proposed an alternative approach by increasing the number of blocks to 48 and determining the difference measure between two frames as the total number of blocks with a histogram difference greater than a given threshold T_{cut} [86]. This method was found to be more sensitive to detecting shot cuts than the previous approach [67]. Although removing the largest differences in the latter approach effectively removed the influence of noise it also reduced the difference between two frames from different shots. In contrast, Ueda's approach put the emphasis on the blocks that change the most from one frame to another. A combination of this and the fact that the blocks were smaller meant it also became more sensitive to camera and object motion [41]. This highlights the problem of choos-

ing an appropriate scale for the comparison between features relating to the visual content of two frames. Using a more local scale increases the susceptibility of an algorithm to object and camera motion, whilst using a more global scale decreases the sensitivity of an algorithm to changes in the spatial distribution.

Motion-Based Approaches

To overcome further the problem of object and camera motion several methods have been proposed which attempt to eliminate differences between two frames caused by such motions before performing a comparison. Methods have been suggested that incorporate a block-matching process to obtain an inter-frame similarity measure based on motion [2, 79, 61]. For each block in frame f_{n-1}, the best matching block in a neighbourhood around the corresponding block in frame f_n is sought. Block-matching is performed on the image intensity values and the best matching block is chosen to be the one that maximises the normalised correlation coefficient. The maximum correlation coefficient is then used as a measure of similarity between the two blocks.

The main distinction between these approaches is how the measures of all the blocks are combined to obtain a global match parameter. Akutsu et al. used the average of the maximum correlation coefficient for each block [2]. This had the disadvantage of combining poor matches with good ones to obtain a passable match between two frames belonging to the same shot. Shahraray used a non-linear digital order statistic filter [79]. This allowed the similarity values for each block to be weighted so more importance could be given to the blocks that have matched well. This improved its performance for cases when some of the blocks being compared have a high level of mismatch. The drawback of this approach was that there can exist good matches between two frames from different shots resulting in a less significant change indicating a shot cut. To overcome this, the authors suggested that blocks be weighted such that a number of the best matching blocks are also excluded. This suggests that the coefficients for the non-linear averaging filter must be chosen carefully when the distribution of similarity values between two frames can vary greatly.

Lupatini et al. summed the motion compensated pixel difference values for each block [61]. If this sum exceeded a given threshold between two frames a shot cut was declared. On the other hand, a novel approach was proposed by Vlachos which used phase correlation to obtain a measure of content similarity between two frames [89]. This method is insensitive to changes in the global illumination and lends itself to a computationally tractable frequency domain implementation.

Finally, Fernando et al. exploited the fact that motion vectors are random in nature during an abrupt shot cut [28]. The mean motion vector between two frames was determined and the Euclidean

distance with respect to the mean vector calculated for all the motion vectors. If there exists a shot cut, the majority of motion vectors will have a large variance due to the poor correlation between the two frames. A large increase in the Euclidean distance can then be used to declare a shot cut. The idea that the two frames on either side of a shot cut are completely uncorrelated, such that incoherent motion estimates are obtained has also been exploited by others [2, 9].

Feature-Based Approaches

Another feature used for the detection of shot cuts is edge features. Zabih et al. proposed a method to detect shot cuts by checking the spatial distribution of exiting and entering edge pixels, known as the *edge change ratio* [98]. This method exploited the fact that edges of objects in the frame before a shot cut cannot be found in the same location in the first frame after a shot cut, i.e. new edges appear far from the locations of disappearing, older edges. A registration technique was used to compensate for global motion between two frames. To compensate for small object motions, edge pixels in one frame within a small distance of edge pixels in the other were not considered to be entering or exiting edges. Thus, any difference between edge pixels should only be the result of a shot cut. Let E_n be the total number of edge pixels in frame f_n, and I_n and O_{n-1} the number of entering and exiting edge pixels in frames n and $n-1$ respectively. The edge change ratio between f_{n-1} and f_n is then defined as

$$ECR_n = \max(I_n/E_n, O_{n-1}/E_{n-1}) \tag{2.5}$$

such that $0 \leq ECR_n \leq 1$ where 0 indicates equality. Although this method illustrated the viability of edge features to detect a change in the spatial decomposition between two frames, its performance was disappointing compared with more simple metrics that are less computationally expensive [22, 61, 57].

2.2 Gradual Transition Detection

There has been much previous work related to the detection of shot cuts. However, less work has been reported on the detection of gradual transitions and accurate detection can still be considered an unsolved problem. Lienhart claimed to present the first robust and reliable dissolve detection algorithm achieving a detection rate of 69% whilst reducing the false alarm rate to 68% [57]. Although an advancement on previous approaches, this still needs to be improved upon for in-

tegration in an actual video indexing system and there is much more improvement required to achieve similar results to those obtained for shot cut detection [41].

Unlike shot cuts, the inter-frame difference during a gradual transition is small. For this reason, it can be difficult to distinguish between changes caused by camera and object motion and those caused by a gradual transition. As a result, detecting all of the gradual transitions often results in many false detections. In fact, Boreczky and Rowe concluded in their comparative study that algorithms do "a poor job of identifying gradual transitions" [8]. The purpose of this section is to review previous work related to the detection of fades and dissolves and address some of the key issues faced when tackling this problem. Useful surveys of previous approaches are also presented in [41, 58].

Histogram-Based Approach

One of the first attempts for detecting gradual transitions was the twin-comparison technique proposed by Zhang et al. which compares the histogram difference with two thresholds [99]. A lower threshold T_{low} was used to detect small differences that occur for the duration of the gradual transition, while a higher threshold T_{high} was used in the detection of shot cuts and gradual transitions. If the difference value was greater than T_{low}, the frame was considered to be the start of a gradual transition. This frame was then compared to subsequent frames, which was referred to as an accumulated difference, since during a gradual transition this will increase. If this accumulated difference was higher than T_{high} the end of the gradual transition was marked when the difference between frame pairs drops below T_{low} for more than two or three frame pairs.

The authors noted that similar changes can be caused by camera and object motion. An approach was proposed to distinguish between such motions and edit effects by detecting patterns in the image motion that are induced by camera movements such as pans, tilts and zooms. If such motions were detected the candidate gradual transition was ignored. Although this helped reduce the number of false detections, it also failed to detect transitions with camera motion before, during or after the transition. Furthermore, it did not handle false positives caused by more complex camera motions or object motion.

Feature-Based Approaches

During a dissolve, the edges of objects gradually disappear while the edges of new objects gradually become apparent. During a fade-out the edges gradually disappear, whilst during a fade-in

edge features gradually emerge. This is exploited by the edge change ratio used to detect shot cuts, which was extended to detect gradual transitions as well [98].

During the first half of the dissolve the number of exiting edge pixels dominates whilst during the second half the number of entering edge pixels is large. Similarly, during a fade-in/out the number of entering/exiting edge pixels are the most predominant. This results in an increased value in the edge change ratio for a period of time during the sequence which can be used to detect the boundaries of gradual transitions. Although, the detection rate of gradual transitions with this method was reported to be good, the false positive rate was usually unacceptably high [56, 61]. There were several reasons for this. The algorithm compensated only for translational motion, meaning that zooms are a cause of false detections. Also, the registration technique only computed the dominant motion, making multiple object motions within the frame another source of false detections. Moreover, if there are strong motions before or after a cut, the cut was misclassified as a dissolve and cuts to or from a constant image are misclassified as fades. The method also did not find the actual boundaries of the transitions well as a result of the edge change ratio values returning to normal values earlier than the real boundaries are reached [58].

Another feature that exploits the loss of contrast during a dissolve was the edge-based contrast proposed by Lienhart [56]. This feature captured and amplified the relation between stronger and weaker edges and proposed to avoid the problem of motion encountered by the edge change ratio. A Canny edge detector was used to locate the edge pixels within the frame and two thresholds employed to determine the weaker and stronger edges. The final contrast measure was high during a shot when there are few weak edges and exhibited a distinct local minima during a dissolve when the weak edges are predominant. This approach was shown to outperform the edge change ratio approach resulting in a much lower false alarm rate [58]. However, Lienhart reports that the average false alarm rate is still too high for most practical purposes [56]. The use of edge magnitudes was also used by Yu et al. who incorporated an edge count to capture the changing statistics of a fade and dissolve [94, 93].

Although not strictly a feature-based approach, another method worth mentioning is the detection of gradual transitions through temporal slice analysis [65, 66]. Here, a video sequence was represented as a 3-D volume which was viewed as a set of spatio-temporal 2-D slices. These slices were then used to extract an indicator which can be used to capture the coherency of the video. Each slice contained regions of uniform colour and texture and the boundaries of these regions were used to detect the presence of shot transitions.

Variance-Based Approach

Another method for detecting gradual transitions is to analyse the temporal behaviour of the variance of the pixel intensities in each frame. This was first proposed by Alattar [3] but has been modified by many other authors as well [29, 60, 85]. It can be shown that the variance curve of an ideal dissolve has a parabolic shape. Thus, detecting dissolves becomes a problem of detecting this pattern within the variance time series. Given an ideal dissolve the first order derivative before and after a dissolve should be zero and a positive constant during a dissolve. Alattar proposed to detect the boundaries of a dissolve by detecting two large spikes in the second-order difference of this curve.

Although these models are reported to perform well, assumptions made about the behaviour of an ideal transition do not generalise well to real video sequences [64]. The two main assumptions are: (i) the transition is linear and (ii) there is no motion during the transition. These assumptions do not always hold for real transitions and as a result of noise and motion in the video sequences the parabolic curve is not sufficiently pronounced for reliable detection. To overcome this problem, Nam and Tewfik presented a novel technique to estimate the actual transition curve by using a B-spline polynomial curve fitting technique [64]. Moreover, Truong et al. noted in their study of real dissolves that the large spikes are not always obvious and instead exploited the fact that the first derivative during a dissolve should be monotonically increasing and constrained the length a potential dissolve can have [84].

Approaches have been proposed specifically for the detection of fade transitions[56, 60, 85]. They start by locating monochrome images which are identified as frames with little or no variance of their pixel intensities. The boundaries are then detected by searching for a linear increase in the standard deviation of the pixel intensities. Lienhart reported accurate detection with this approach on a large test set [56].

2.3 Key Frame Selection

The main approach to automate the video indexing process is to select representative key frames from shots or scenes to generate a video abstract (or storyboard). The collection of key frames can then be employed for further characterisation and subsequent queries on the video data. It is worth noting that the choice of key frames is subjective and often application dependent. For effective video browsing and retrieval, the selected key frames should be able to represent the content of the entire video sequence [16]. In contrast, Dufaux proposed a technique to automatically extract

a single key from a video sequence designed for a system to search for a video on the World Wide Web [23]. There has recently been many works related to the problem of key frame selection and several surveys on the automatic indexing of video data are presented in [13, 45, 6]. In this section, a brief review of previous approaches is presented.

Two main approaches to key frame selection have been suggested: (i) with the explicit detection of shot transitions, and (ii) with out such detection. An initial approach was proposed in which the first frame of each shot was selected as a key frame [38, 17]. The ordered set of key frames is sometimes referred to as a *filmstrip* [18]. This is not always sufficient as there can exist salient changes within a shot due to camera or object motion. To increase the number of frames in a shot Ardizzone and Cascia suggested the number of frames should be related to the length of the shot [7]. If the shot is shorter than one second, the middle frame was chosen and if the shot is longer, a key frame for each second was chosen. Once the key frames have been selected they were characterised by their optical flow field for the purpose of video indexing. This approach can oversample a sequence, as a shot may be long but contain little content change.

Zhang et al. suggested extracting key frames using similarity measures similar to those used in shot cut detection [100]. Given a shot, the first frame was always chosen. Following frames in the shot were compared with the last selected key frame, based on some similarity measure such as colour histograms or moments. If a significant content change was deemed to have occurred, the current frame was chosen as an additional key frame. It was proposed that any significant action would be represented by a key frame whilst static shots resulted in only one key frame. The use of a cumulative measure between key frames was also proposed by Kim and Park [52]. Once the key frames have been selected, the similarity between different video sequences is evaluated by the modified Hausdorff distance between sets of key frames. Chang et al. proposed an interesting approach to determine the minimum set of key frames for a shot such that the distance between every frame in the shot and at least one key frame in the set was less than some threshold [16]. They used examples of colour histograms and correlation as the dissimilarity measure. The minimal cover of a proximity graph was used to search for the set of optimal key frames.

An alternative approach to find the optimal set of key frames such that the frames are maximally distinct and individually carry the most information was proposed by Vermaak et al. [88]. The input video was transformed into a sequence of representative feature vectors. Using this representation a utility function was defined and the key frame sequence that maximises this function was obtained by a non-iterative Dynamic Programming procedure.

Instead of using a distance criterion, Wolf used the optical flow to identify local minima of motion in a shot to identify key frames [90]. It was proposed key frames are identified by stillness—either

the camera stops in a new position or the characters hold gestures to emphasise their importance. The sum of the magnitudes of the components of optical flow at each pixel were computed and points of local minima in the sequence were used to select key frames.

As mentioned above, there are approaches that do not specifically target individual shots to select key frames. Typically, the frames are represented in a lower dimensional space using a representation similar to those used for shot change detection, obtained by global SVD [36] or windowed PCA [40]. Grouping of frames is typically done by thresholding [36], greedy clustering [92], or line simplification [92] in the reduced dimensionality space.

2.4 Summary

There have been many algorithms proposed for detection of shot transitions, most of which are negatively influenced by the presence of camera or object motions. This suggests an algorithm is required that can distinguish between content changes caused by transitions and those caused by such motions. A primary focus of the present work is to reduce the false detection rate, particularly where gradual transition detection is concerned. It has been suggested that the performance of such algorithms may be improved by employing an adaptive thresholding technique [95]. However, this work will focus on designing an algorithm to detect all types of shot transitions using a single technique and the same parameter set, rather than a set of dedicated methods.

The present work is also concerned with the generation of a video abstract for the purpose of indexing and retrieval. For this reason, the key frame selection algorithm will include the explicit detection of shot transitions. Inspired by previous approaches to find the optimal set of key frames, this algorithm will allow the number of key frames to vary depending on the level of content change contained within each shot. Each key frame must represent the content of each video segment whilst minimising representational redundancy.

Chapter 3

Shot Cut Detection

The decomposition of a video sequence into its constituent shots is the first step toward automatic annotation of digital video sequences. The shot cut is the simplest, most commonly occurring transition from one shot to the next, and its detection thus represents the natural starting point when faced with the problem of automated video indexing.

In this chapter a hierarchical block-based motion compensated shot cut detection method is presented. The proposed method uses correlation coefficients in combination with colour histogram differences as a measure of content similarity between two consecutive frames. Results are given, including a comparative analysis against five commonly cited methods.

3.1 Block-Based Motion Compensated Shot Cut Detection

The aim of any shot cut detection algorithm is to detect temporal discontinuities in the visual content of a video sequence. The key element of such a method is the choice of feature(s) used to represent the visual content. A feature should be selected that exhibits similar properties for images belonging to the same shot, but which shows a pronounced change when compared between images belonging to different shots. Existing algorithms differ in (i) the features used to represent the visual content of each image and (ii) the difference metrics used to compare them. By computing a difference metric for consecutive frame pairs a shot cut is deemed to be present if the difference is greater than some predefined threshold. In addition, a good shot transition detection algorithm should be able to distinguish visual changes caused by a shot transition from those caused by camera and object motions.

Global features are less sensitive to changes in the spatial layout of images than pixel-level comparisons [96]. For this reason, algorithms based on the comparison of global features have been proposed to overcome the problem of object and camera motions. However, such methods can fail to detect transitions between two shots whose global features exhibit similar properties but have different spatial layouts. To address this issue, block-based methods were proposed which compare the features of corresponding regions of different images [61].

The difficulty of dealing with camera and object motions motivates the use of a motion-based algorithm to eliminate differences caused by motion prior to computing a disparity metric between two frames. Any difference in the visual content should then only be due to a shot cut. Two shots joined by a transition can have either:

1. different colour distributions, or
2. similar colour distributions but a different spatial layout.

Therefore, a change in the structural layout of an image is usually indicative of a visual discontinuity. Edge features represent much of the intrinsic structure of an image, meaning that the distribution of edges in the last frame before a transition can be expected to be different to the layout of edge features in the first frame after a transition. This observation was, for example, exploited by Zabih et al. who proposed an edge change ratio to detect shot transitions [98]. A registration technique was used to compute the global motion between two frames before computing the edge change fraction. Whilst this method illustrated the viability of using edge features in detecting visual discontinuities, the authors also indicated two of its limitations which may give rise to false detections. Firstly, the edge detector used does not cope with rapid changes in overall scene brightness or scenes with little contrast. Secondly, the motion compensation technique employed does not handle multiple moving objects or global motions that violate the 2-D translational model.

Based on the above observations we conclude that future algorithms need to be designed with the reduction of the number of false detections in the presence of camera and object motion in mind. This implies using a feature for comparing two images which shows little change in the presence of camera and object motions, whilst still being able to detect changes in the spatial layout of an image. We propose a motion-based method which uses the correlation of edge features between corresponding regions to measure the visual content similarity between two frames, as outlined in Fig. 3.1. For each frame pair in the video sequence, the first frame is divided into a regular grid of blocks. A similarity metric for each frame pair can then be derived by comparing the edge features contained within each block. The next step is to estimate the motion for each block between the

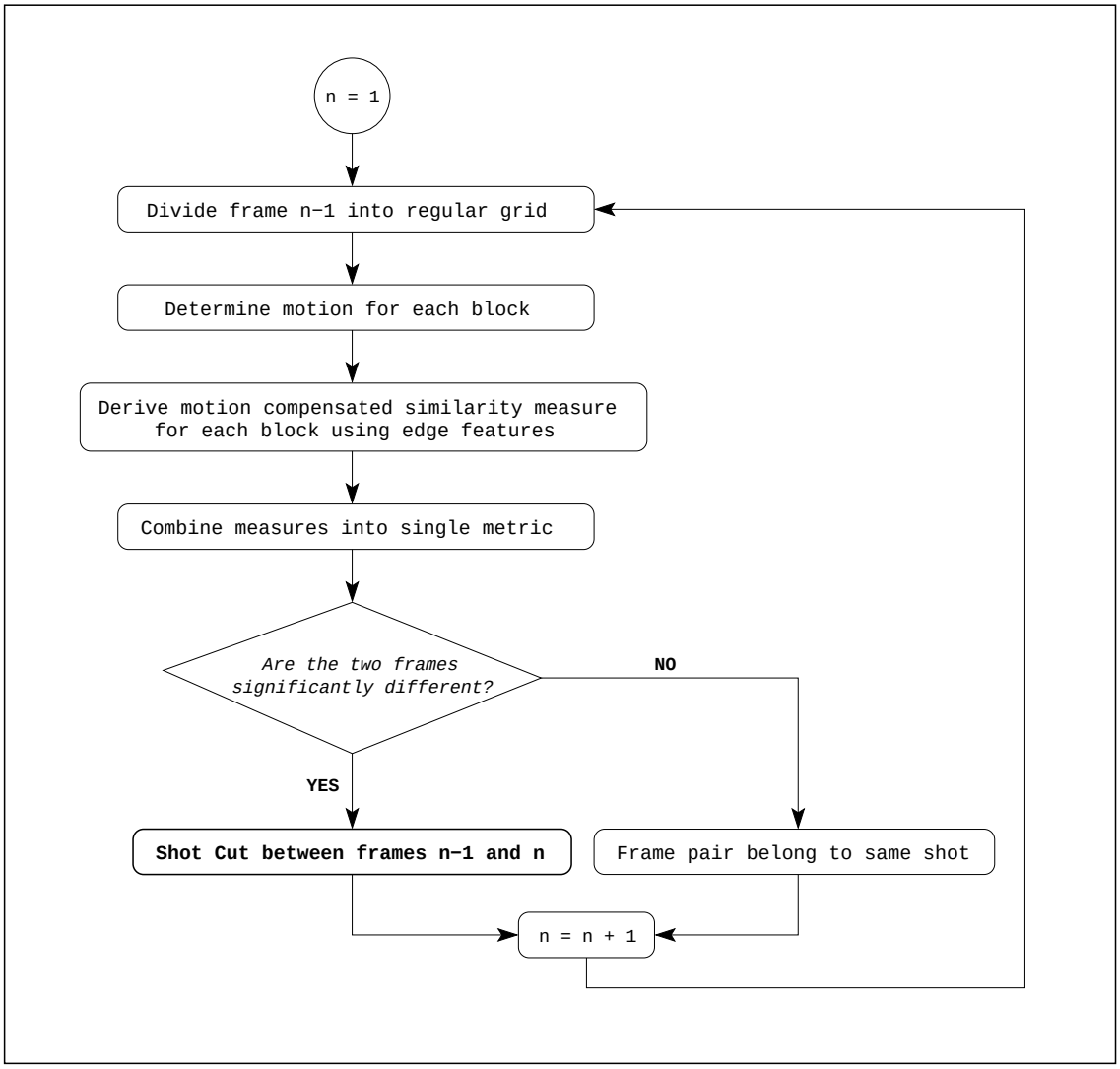


Figure 3.1: Method outline for shot cut detection.

two frames to compensate for differences caused by camera and object motions. Any remaining differences in the visual content should only be the result of a shot cut between the frame pair. For each block in the first frame, the best matching block in a neighbourhood around the corresponding block in the second frame is sought. The location of the best matching block can be used to find the offset of each block between the two frames to then compute a motion compensated similarity metric. Given that a measure of the similarity of the distribution of the edge features contained within each block is required, an obvious candidate for such a matching function is normalised correlation, as a similarity metric can be computed as an integral part of the motion estimation process [11, 53]. The value of the maximum correlation coefficient can be used as a *goodness-of-fit* measure for each block. Between two frames belonging to the same shot the spatial distribution of edge features should be similar and the goodness-of-fit for the majority of the blocks should

indicate a good match. A high number of poor matches should indicate the spatial distribution of edge features has changed, suggesting the presence of a shot cut. A similarity metric for each frame pair is derived by combining the goodness-of-fit measures of all the blocks. The final step is to use this similarity metric to detect the presence of a shot cut.

3.2 Block-Based Motion Estimation

The first step in this method is to estimate the motion for each block between the two frames to eliminate differences caused by local and global motions. Block-based motion estimation is founded on the assumption that the image is composed of moving blocks and that the motion remains unchanged over a particular block of pixels. Then the i-th block $x_i(\boldsymbol{\xi})$ in frame f_{n-1} is modelled as a shifted version of a same-size block $x_j(\boldsymbol{\xi})$ in frame f_n as

$$x_i(\boldsymbol{\xi}) = x_j(\boldsymbol{\xi} + \boldsymbol{d}) \tag{3.1}$$

where $\boldsymbol{\xi}$ is the spatial coordinate vector and $\boldsymbol{d}$ is the displacement vector determined by searching for the position of the best matching block in a neighbourhood around the corresponding block in frame f_n. Block-matching algorithms vary according to the matching criteria and search strategy used [81].

Many ways of measuring the difference or similarity between the grey-level pattern contained within $x_i(\boldsymbol{\xi})$ and a potentially best matching block in f_n have been proposed [34, 11]. The most commonly used distance measures are the sum of absolute differences and the sum of squared differences. Both measures can be modified to consider the effect of global grey-level variations by setting the average grey level difference to 0 or by locally scaling the intensity [34]. Minimising the sum of squared differences can be replaced by maximising the cross-correlation term which gives a measure of the degree of similarity between two regions in different images. Given $x_i(\boldsymbol{\xi})$ and $y_j(\boldsymbol{\xi})$, where the latter represents a neighbourhood around the corresponding block in frame f_n, the correlation of the two 2-D functions is defined by

$$\rho(\boldsymbol{\zeta}) = (x_i \star y_j)(\boldsymbol{\zeta}) = \sum_{\boldsymbol{\xi}} x_i(\boldsymbol{\xi} + \boldsymbol{\zeta}) y_j(\boldsymbol{\xi}). \tag{3.2}$$

Informally, this associates with each possible discrete offset $\boldsymbol{\zeta}$ a similarity value which rates the quality of the match obtained when applying the offset corresponding to $\boldsymbol{\zeta}$ to the region x_i. In other

words, by computing $\rho(\boldsymbol{\zeta})$ over all possible translations (all instances where x_i and y_j overlap), the correlation field has its peak value at the integer translation closest to the true translation of x_i between f_{n-1} and f_n. In other words, if (3.1) holds for the region x_i, then $\rho(\boldsymbol{\zeta})$ has its peak value at $\boldsymbol{\zeta} = \boldsymbol{d}$, enabling the translation to be determined. There are, however, several disadvantages to using (3.2) for motion estimation: the range of $\rho(\boldsymbol{\zeta})$ is dependent on the size of x_i; and it is not invariant to changes in image amplitude such as those caused by changing lighting conditions across the image sequence [55].

A related measure which overcomes these difficulties is the correlation coefficient, referred to as the normalised cross-correlation, defined as

$$\gamma(\boldsymbol{\zeta}) = \frac{\sum_{\boldsymbol{\xi}}(x_i(\boldsymbol{\xi}+\boldsymbol{\zeta}) - \bar{x}_i(\boldsymbol{\xi}))(y_j(\boldsymbol{\xi}) - \bar{y}_j(\boldsymbol{\zeta}))}{[\sum_{\boldsymbol{\xi}}(x_i(\boldsymbol{\xi}+\boldsymbol{\zeta}) - \bar{x}_i(\boldsymbol{\xi}))^2 \sum_{\boldsymbol{\xi}}(y_j(\boldsymbol{\xi}) - \bar{y}_j(\boldsymbol{\zeta}))^2]^{1/2}} \tag{3.3}$$

where $\bar{x}_i(\boldsymbol{\xi})$ is the mean of the block x_i and $\bar{y}_j(\boldsymbol{\zeta})$ is the mean of y_j in the overlap region under x_i. This statistical measure has the property that it measures correlation on an absolute scale and gives a linear indication of the similarity between blocks [11]. The value of the maximum correlation coefficient can then be used as a measure of confidence in a match. In addition, by normalising the cross-correlation function, the measure becomes invariant to changes in image amplitude [55].

Calculating the normalised correlation in the spatial domain is, however, computationally expensive unless the regions are small. Assuming x_i and y_j are square regions of size $N \times N$ and $M \times M$ respectively, where usually $N \leq M$, normalised correlation between the two can be computed more efficiently in the frequency domain as N approaches M and with larger N and M [55]. Frequency domain normalised correlation is defined as

$$\rho(\boldsymbol{\xi}) = \frac{\mathcal{F}^{-1}\{\hat{x}_i(\boldsymbol{\omega})\,\hat{y}_j^*(\boldsymbol{\omega})\}}{\sqrt{\int |\hat{x}_i(\boldsymbol{\omega})|^2\, d\boldsymbol{\omega} \;\cdot\; \int |\hat{y}_j(\boldsymbol{\omega})|^2\, d\boldsymbol{\omega}}} \tag{3.4}$$

where $\boldsymbol{\xi}$ and $\boldsymbol{\omega}$ are the spatial and spatial frequency coordinate vectors respectively, $\hat{x}_i(\boldsymbol{\omega})$ denotes the Fourier transform of block $x_i(\boldsymbol{\xi})$, $\mathcal{F}^{-1}$ denotes the inverse Fourier operator and * is the complex conjugate [15]. Although correlation in the frequency domain is computationally efficient, it is equivalent to *cyclic*, or *circular correlation*, which must be considered when using this technique in practical motion estimation. Circular correlation results in contributions due to the 'wraparound' of blocks which are physically inappropriate for motion estimation since objects disappearing at one end of the window generally do not reappear at the other end. In addition, the 2-D DFT assumes periodicity in both directions. Discontinuities from left to right boundaries,

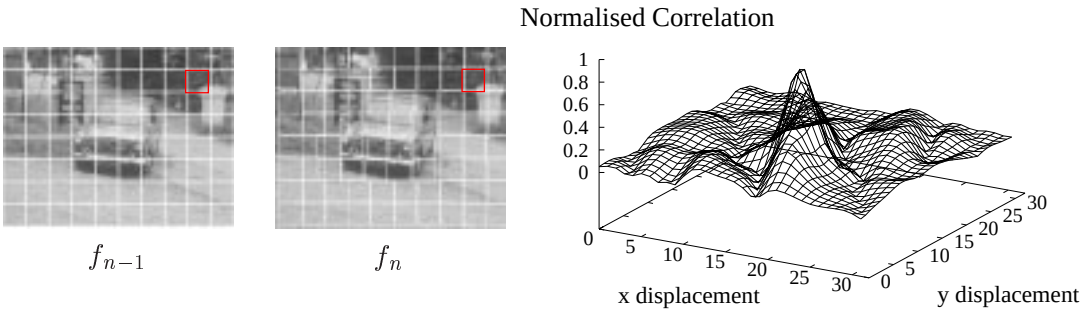
and from top to bottom, can potentially introduce spurious peaks. Both of these effects can be counteracted by using a block twice as big as the area of interest and applying a window function which tapers off outside the region of interest without introducing strong artifacts in the block. In addition, the resulting displacement estimates need to be centred to accommodate negative displacements. Therefore, for an even size block M the range of estimates is $[-M/2+1, M/2]$. However, due to the wraparound effect and the application of the window function displacement estimates are unreliable toward the edge of the correlation field. For this reason, for a block size M only motion estimates within the range $[-M/4+1, M/4]$ are considered. To estimate the motion between an area of interest x_i of size $N \times N$ from f_{n-1} with a corresponding area of interest x_j of the same size in f_n the method can be described as follows.

1. Extract blocks x_i' and x_j' both of size $M \times M$ where $M = 2N$.
2. Apply window function to x_i' and x_j' which tapers off outside the regions of interest.
3. Perform normalised correlation using (3.4).
4. Only search for the correlation peak within the displacement estimates $[-N/2+1, N/2]$.

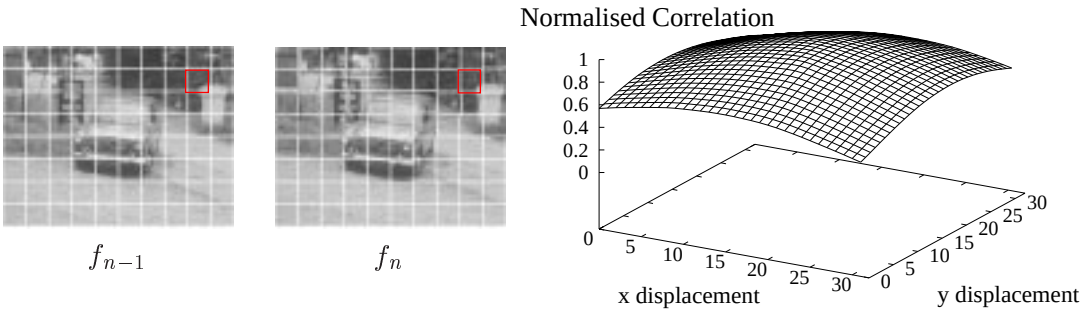
For our application, the use of the frequency domain implementation of normalised correlation represents a significant efficiency win compared with the direct implementation. A comparative complexity analysis can be found in Appendix B.

3.3 Correlating High-Pass Features

The use of normalised correlation in the block matching process is based on the assumption that the distribution of edge features will change during a shot transition. Therefore the correlation of edge features between two images must be computed and not just the correlation of the grey-level pattern contained within each block. Since edges represent high-frequency image phenomenon, a high-pass filter is applied to each image before performing the correlations. This serves to accentuate the contributions from higher spatial frequencies which normally correspond to object structure. It also suppresses the low frequency information so the correlation is not biased toward the matching of lower frequency components. A correlation field derived from high-pass regions will contain more detectable peaks, whereas correlation fields derived from low-pass regions will result in a flat correlation field leading to inaccurate peak detection as shown in Fig. 3.2. The motion is estimated more accurately between the first frame pair which have been high-pass filtered before computing the normalised correlation. A high-pass filter was not applied to the second



(a) A high-pass filter has been applied prior to correlation between the highlighted blocks.

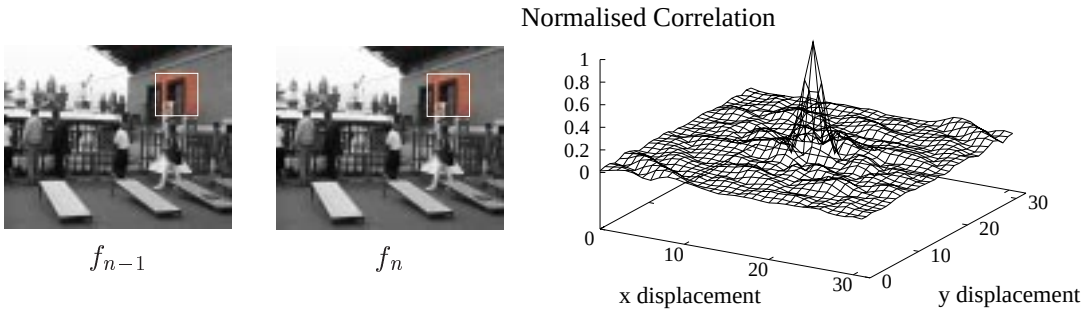


(b) No high-pass filter has been applied prior to correlation between the highlighted blocks.

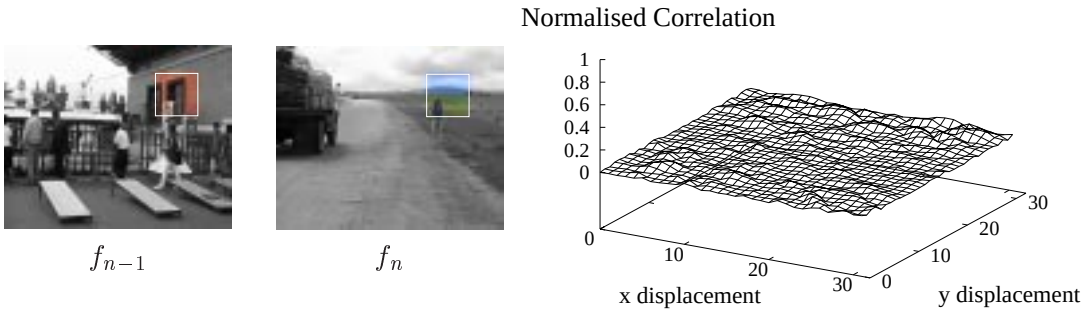
Figure 3.2: Applying a high-pass filter leads to more accurate motion estimation.

frame pair, leading to a flat correlation field and an inaccurate motion estimate. An additional consequence of applying a high-pass filter is that the local mean of the image intensities is removed, meaning that the correlation between blocks is invariant to changes in the mean intensity. By normalising the correlation, the method is insensitive to a positive scaling of the image intensities and invariant to changes in the global illumination across the image sequence.

Given the above, the proposed algorithm can be summarised as follows. For a given frame pair f_{n-1} and f_n, a high-pass filter is applied to both images. The first frame is partitioned into a regular spatial grid of non-overlapping blocks of size $N \times N$. Each block in frame f_{n-1} is then doubled in size and correlated with the corresponding block in frame f_n to estimate the motion between them. The value of the of the maximum correlation coefficient is used as a goodness-of-fit measure for each block. Fig. 3.3 shows the typical output of the normalised correlation between two blocks from temporarily adjacent frames. The first frame pair are from the same shot and the second frame pair contain a shot cut. The large, well defined peak in Fig. 3.3(a) suggests a good match between the block contents. In contrast, the low smooth correlation field in Fig. 3.3(b) implies poor correlation between the two blocks. Between two frames belonging to the same shot,



(a) Frame pair are from the same shot.



(b) Frame pair contain a shot cut.

Figure 3.3: The correlation function obtains a measure of content similarity.

the goodness-of-fit for the majority of the blocks can be expected to be large, indicating a good match.

Figure 3.4 illustrates the complete algorithm for computing a similarity metric for each block between two frames. A similarity metric E_n for each frame pair f_{n-1} and f_n is derived by combining the the goodness-of-fit measures of all the blocks. This was initially achieved by calculating the mean of the goodness-of-fit measures, given by

$$\mu = \frac{\sum_{i=1}^{B} p_i}{B} \tag{3.5}$$

where $p_i = \max(\rho(\boldsymbol{\xi}))$ for block i, and B is the total number of blocks. However, the linear combination of the goodness-of-fit measures has the disadvantage of averaging high match values with low ones to generate a passable match. This is not a good approach since mismatches can occur between blocks from two frames that do not contain a shot cut while the majority of blocks match well. Examples of this are shown by the normalised distribution of goodness-of-fit measures

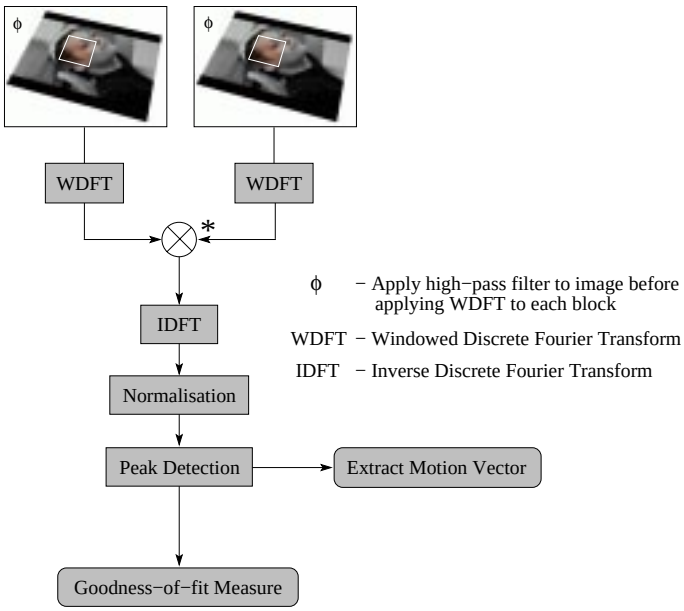


Figure 3.4: Obtaining a goodness-of-fit measure and a motion vector for a block between temporally adjacent frames.

(quantised to 0.05) for the frame pairs in Fig. 3.5. Poor match values can occur between two frames belonging to the same shot for a number of different reasons.

1. A window does not contain sufficient high-frequency information, such as edges and corners, to determine the motion accurately, which results in a low flat correlation field (Fig. 3.5(a)).

2. Multiple object motions present within a single window results in several smaller peaks.

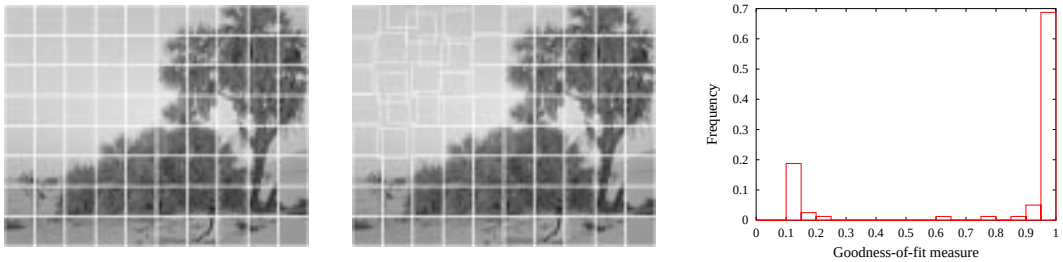
3. Occlusion, where object motion results in covered/uncovered background, as illustrated in Fig. 3.5(b).

4. Data that does not fit the translational motion model well, e.g. zooms, rotational motion and local deformations.

5. Motion blur, or out of focus (Fig. 3.5(b)).

6. Motions greater than $[-N/2+1, N/2]$.

To prevent these outliers negatively influencing the similarity metric for a frame pair, a more satisfactory measure can be obtained by using the median of the goodness-of-fit measures. For example, for the frame pair in Fig. 3.5(a) $\mu = 0.78$, whilst the median equals 0.99 and for the



(a) Several blocks do not contain sufficient high-frequency information.

(b) The foreground object is out of focus, and its motion causes occlusion.

Figure 3.5: Poor matches can exist between frames belonging to the same shot.

second frame pair in Fig. 3.5(b) $\mu = 0.79$, yet the median equals 0.94. Therefore, a similarity metric E_n between the edge distributions in the frame pair f_{n-1} and f_n is defined as

$$E_n = \text{median}\{p_i\}. \tag{3.6}$$

A shot cut is then detected if $E_n < T_E$ i.e. if the similarity between a frame pair is less than some threshold T_E. Figure 3.6 shows the similarity metric for each frame pair for a video sequence containing 6 shot cuts. It can be seen that during a shot E_n remains high. On the other hand, a shot cut manifests itself as a sudden decrease in E_n. In this example, any value for T_E in the range $[0.3, 0.7]$ would detect all the shot cuts. The most appropriate value for each sequence can vary, however a robust algorithm should give a good performance with the same value of T_E across all video sequences. The dependency of an algorithm's performance on the chosen value is discussed further in Sec. 3.7. Figure 3.7 shows two more examples of video sequences with 22 and 33 shot cuts respectively.

3.4 Optimal Block Size

The block size is an important parameter to any block-based motion estimation algorithm. A common problem encountered in motion estimation is the aperture problem, where motion can

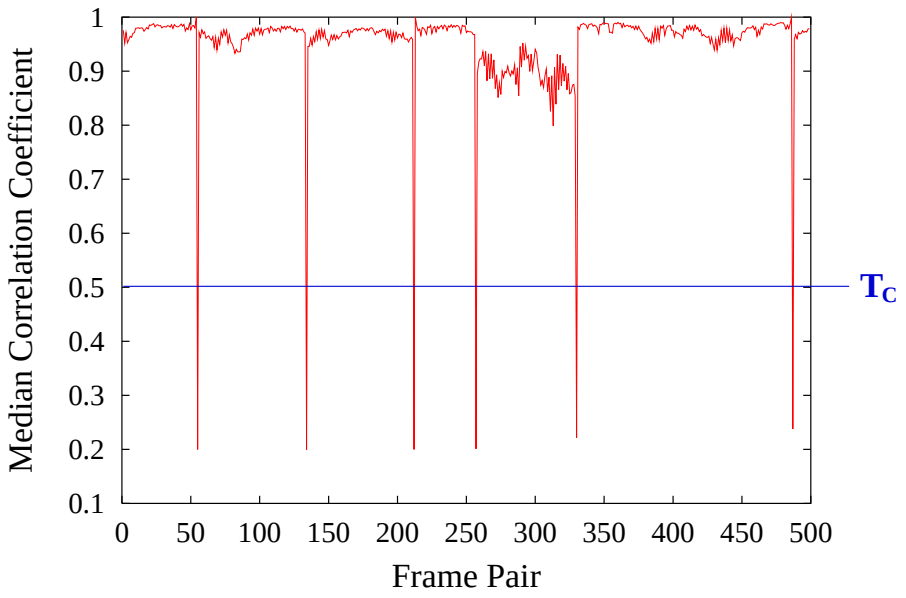


Figure 3.6: A pronounced change in the similarity metric indicates a shot cut.

only be estimated reliably in a direction parallel to a spatial image gradient. The most appropriate aperture size will vary with the data—a larger block is more likely to contain sufficient grey-level variation to estimate the correct motion, but at the same time, the single translational motion per block model is more likely to be violated. Although the data contained in smaller blocks are more likely to translate by the same motion vector, smaller blocks are also less likely to contain enough intensity variation, which makes it impossible to measure the motion accurately. Thus there are two conflicting problems influencing the block size. This problem is sometimes referred to as the "Generalised Aperture Problem" [46]. In addition, the block must be sufficiently large to estimate all sizes of displacements present in the video data.

Initially, a window size of 32×32 was chosen empirically as it appeared to give an acceptable trade-off for the sequences used (with approximate image dimensions 256×256). Although this method performed well, there were two main sources of false detections.

1. Frames with large regions containing little or no high frequency information, resulting in poor correlation for the majority of the blocks.
2. Motions outside the range $[-15, 16]$, again resulting in poor correlation.

The first item was the main contributing factor and is illustrated by several frames in Fig. 3.8. With a block size of 32 (white grid) the majority of the blocks contain little high frequency information resulting in poor correlation with their corresponding blocks in the following frame. This in turn

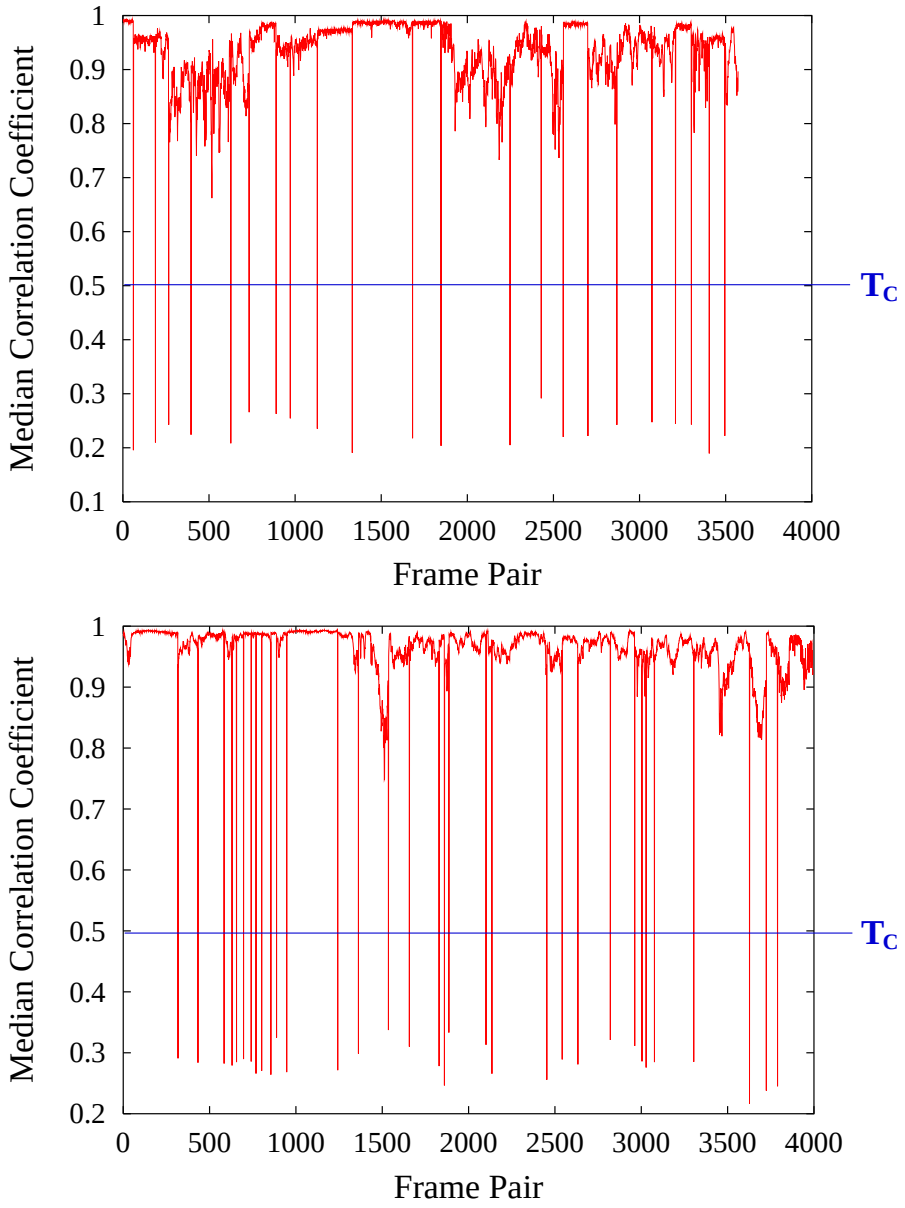


Figure 3.7: Illustration of the similarity metric for two different video sequences.

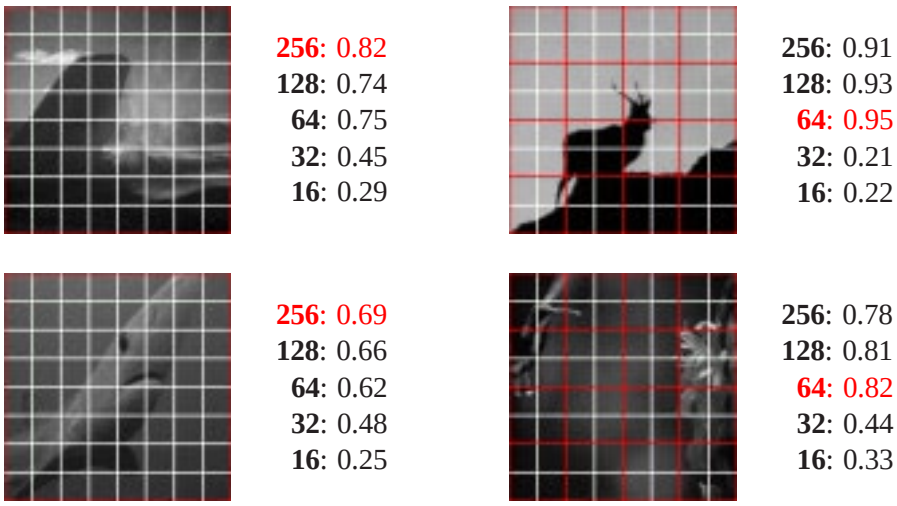


Figure 3.8: The optimal block size will vary with the video data: frames with little high frequency information require larger block sizes. The white grid illustrates the fixed block size of 32 whilst the optimal block size is illustrated in red.

results in a low similarity metric for each frame pair. Usually this problem can be overcome by increasing the block size. Next to each frame (with dimensions 256×256) are the similarity metrics obtained for each frame pair using various different block sizes ranging from 256 down to 16 (in each case the second frame is almost identical so they have not been shown). The optimal block size is shown by a red grid and the similarity metric obtained using this block size is also highlighted in red. It can be seen that in these cases a larger block size results in an improved similarity metric.

Increasing the block size can result in poor correlation for other frame pairs if they contain multiple motions or data that violates the translational only motion model as shown in Fig. 3.9. The first frame pair contain two objects moving to the right as the background moves to the left and between the second frame pair the camera zooms out slightly. A smaller block size is preferable in the presence of such motions. These examples illustrate that the ideal block size varies with the video data. It is clear from the performed experiments that using a fixed window size is not flexible enough to cope with real-world data. The method was therefore extended to use hierarchical motion estimation, described in the next section. As well as improving the motion estimates of the smaller block sizes, it also allows an adaptive window size to be used to compute the similarity metric for each frame pair. The window size which is the most appropriate for the underlying data is then chosen.

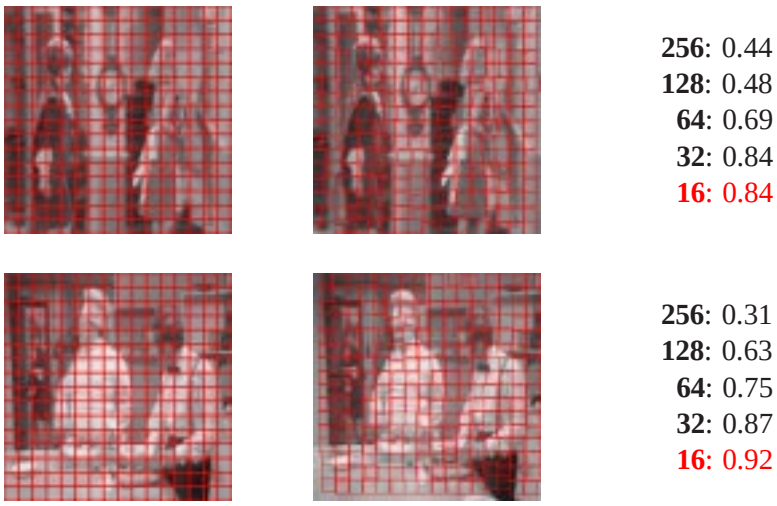


Figure 3.9: The optimal block size will vary with the video data: frames containing multiple motions or motion that violates the 2-D translational model require smaller block sizes. The red grid illustrates the optimal block size.

3.5 Hierarchical Motion-Based Shot Cut Detection

The basic idea of hierarchical block matching is to perform motion estimation at some coarse scale and then use these estimates as initial guesses for estimating the motion at finer scales. The coarse scales give broad estimates of the visual motion whilst the finer scales serve to fine-tune the displacement vector estimate. Hierarchical motion estimation techniques come in two slightly different flavours: *multiresolution* and *multigrid*. A multiresolution algorithm typically constructs a pyramid representation of each frame where the full resolution image is at the bottom and images at the upper levels are obtained by appropriate low-pass filtering and sub-sampling [14]. Motion estimation is performed at each level successively, starting with the lowest resolution image. The displacement vector estimated at a lower resolution level is used as a'rough guess' at the next resolution level. In contrast, the multigrid approach is to start at some large window size in the image, and gradually move down to smaller analysis windows using the ancestral estimates as a starting point. In this work we adopt a multigrid approach as the multiresolution family can introduce unwanted spatial aliasing due to the sub-sampling step, and by blurring the image the structural detail is gradually removed—the very structure that must be correlated between different images to identify any change in its distribution. Indeed, the same structure must be correlated at different spatial resolutions.

The technique can be outlined as follows. Starting at some large block size, the displacement between each corresponding block between frames f_{n-1} and f_n is estimated. Each block is then

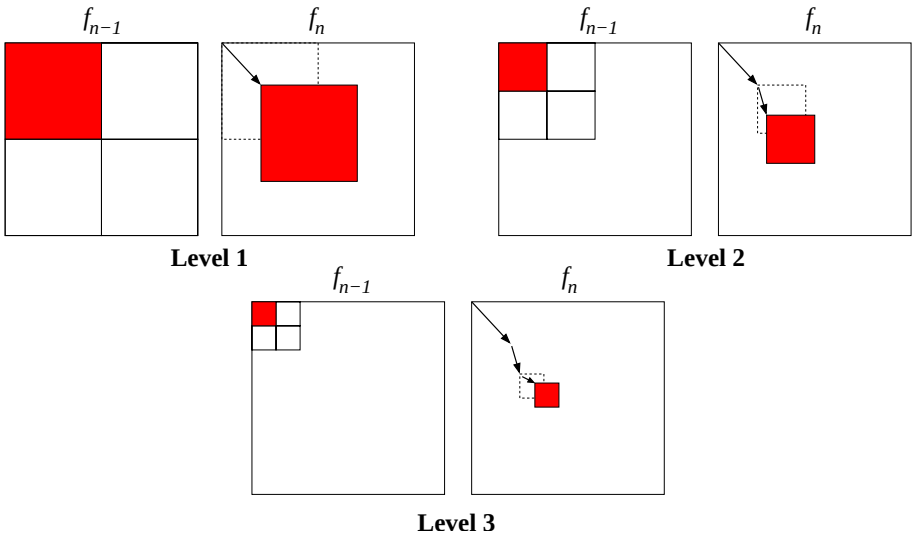


Figure 3.10: Hierarchical block matching.

subdivided into four smaller 'child' blocks and these estimates are used to determine which block in f_n each child in f_{n-1} is correlated with, i.e. if $\boldsymbol{p}$ is the position of the child block, then it is correlated with the block at $\boldsymbol{p}+\boldsymbol{d}$ where $\boldsymbol{d}$ is the displacement vector estimated at the parent level. The search procedure continues until a small block size is reached resulting in a 'vector train' describing the displacement of the small size block between f_{n-1} and f_n. This search procedure using a single parent quad-tree is illustrated in Fig. 3.10.

One problem of a coarse-to-fine approach is that if the coarse-scale estimate is incorrect by a substantial amount, then the finer scales may be unable to recover from these errors. A region containing a motion boundary will track one of the two motions which may mean that the other motion cannot be recovered at smaller region sizes. For example, in Fig. 3.11 assuming block 3 tracks the motion of the red object, if block A inherits its parents motion it is unlikely its real motion can be recovered. It is worth noting that a given region can be better off tracking one of its parent's neighbours rather than its direct parent. In this case block A would be better off tracking the motion of the larger block 2. For this reason, it may be beneficial to use a four parent quad-tree, where each child block can be correlated with up to four possible corresponding blocks in the next image. The correlation field with the highest peak value is used to compute the displacement vector for the child block. This can help to overcome discontinuities in the motion field e.g. at motion boundaries. However, a four parent quad-tree incurs a significant increase in computational time. After viewing the test data it is apparent that when a block contains multiple motions, the most common occurrence is that it contains only two motions. For example, an object can be moving against a stationary background, or an object can be tracked so it appears stationary against a moving background. To overcome such situations, each block is correlated

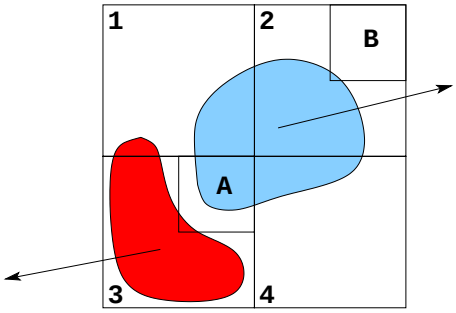


Figure 3.11: One level of hierarchical block matching using a four parent quad-tree.

twice—once using its parent's displacement estimate and once without. In Fig. 3.11, assuming the content of block B corresponds to stationary background, it would be better for this block not to track the motion of its parent. In addition, correlating block A without its parent's motion increases the likelihood of recovering its real motion. Although this approach doubles the number of correlations required, the FFT of each block in f_{n-1} is only computed once and all the blocks in the top level without parents are only correlated with one corresponding block. The increase in the computational time is approximately 1.5 times that required for the single parent quad-tree. Although neither algorithm has been implemented optimally, Table 3.1 shows a comparison of the time required for each algorithm running on a 1.53 GHz AMD Athlon XP 1800+, for a number of different levels. Employing hierarchical motion estimation in place of the fixed 32×32 window size with the smallest size blocks in the quad-tree equal to this size, improved the motion estimates in the presence of motions greater than $[-15, 16]$ between two frames. As a result the maximum correlation coefficient for each block increased in value and therefore so did the overall similarity metric E_n computed for the frame pair. This reduced the number of false detections of shot cuts for the fixed block size algorithm in the presence of large object and camera motions.

The hierarchical method allows the use of an adaptive single window size to obtain a measure of the content similarity. A similarity metric is computed at each level of the hierarchical search procedure and the highest value is chosen as the similarity for the current frame pair. This way, the most appropriate block size is chosen to represent the visual content of each frame pair which will vary according to the video data. For the frames in Figs 3.8 and 3.9 the selected block sizes are highlighted in red and even for this small example set this varies from 16 to 256. For each frame pair, the first frame is divided into a regular grid of blocks equal to some large block size. Hierarchical motion estimation continues, splitting each block into 4, until a designated smaller block size is reached which results in K levels of blocks. The median correlation coefficient M_k is computed across all the blocks at each level k where $1 \geq k \geq K$. That is, M_k represents the similarity metric for the frame pair using the block size at level k. The largest of these values is

Number of levels : [Block sizes]	Single parent	Double correlation
2 : [32,64]	1.822s	2.609s
3 : [32,64,128]	2.608s	3.975s
4 : [32,64,128,256]	4.145s	5.856s

Table 3.1: An approximate comparison of the computational time required for two different hierarchical block matching schemes running on an AMD Athlon XP 1800+ processor.

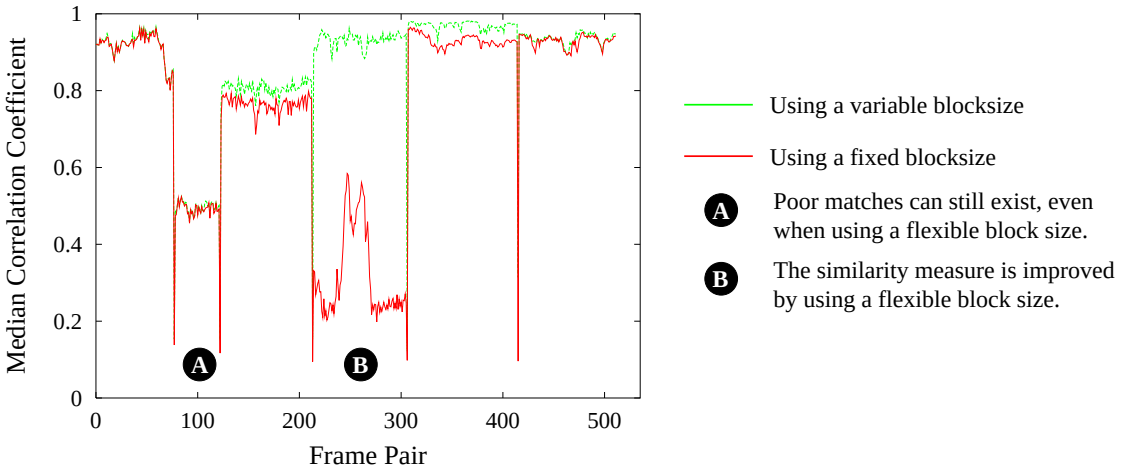
then used to represent the similarity metric E_n for a frame pair f_{n-1} and f_n

$$E_n = \max\{M_k\}. \tag{3.7}$$

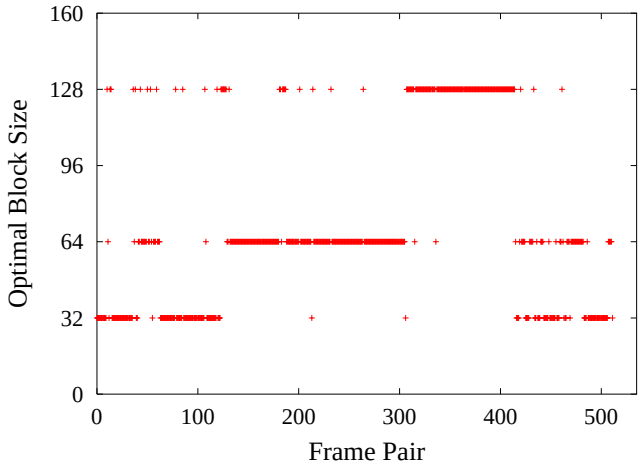
Figure 3.12(a) shows a comparison of the value obtained for E_n using a fixed block size of 32×32 (shown in red) with that obtained using 3 levels of hierarchical block matching and using the optimal block size to determine the value of E_n (shown in green). In addition, Fig. 3.12(b) shows the variation of the optimal block size used during this sequence varying between 128 and 32. It can be seen that allowing an adaptive window size can drastically improve the similarity measure obtained during a shot. However, there are still some cases where the most appropriate block size results in a relatively poor similarity metric. If the similarity metric is sufficiently low it can result in a false shot cut detection. In the next section, we discuss how the correlation measure can be combined with a colour histogram difference to distinguish between frames belonging to different shots.

3.6 Block-Based Colour Comparison

Even using the adaptive window size there are cases where correlation is of limited use as a similarity metric. In some video data there is such little structure in the image that correlating with a large analysis window still results in a poor match as illustrated by the frame pairs in Fig. 3.13. Another case where correlation proves unreliable is if there are many objects moving between two frames such as the frame pairs in Fig. 3.14. A small analysis window may be the most appropriate but it can still result in poor correlation due to the fact significant local motion causes the contents of many blocks to change. Other causes of poor correlation, irrespective of the block size used, are shown in Fig. 3.15. Motion blur and content change at the image boundaries causes poor correlation between the first frame pair. Between the second frame pair, a large block size results in poor correlation because of occlusion and multiple motions. Whilst the smaller blocks help to overcome this, the similarity metric at the lower levels is negatively influenced by many of the



(a) A comparison of E_n obtained using a fixed and an adaptive block size.



(b) The optimal block size used varies greatly during a sequence.

Figure 3.12: Using an adaptive window size can improve the similarity measure obtained between two frames but there can still exist relatively poor matches.

Figure 3.13: Frame pairs containing little structure still result in relatively poor correlation even with a large optimal block size.

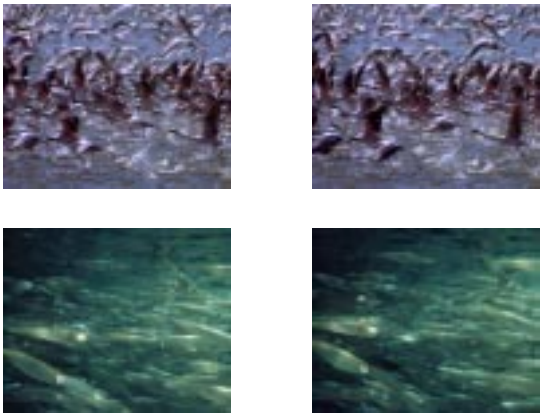


Figure 3.14: Frame pairs containing significant local motion still result in relatively poor correlation even with a small optimal block size.

256: 0.19
128: 0.26
64: 0.27
32: 0.42
16: 0.51

256: 0.21
128: 0.23
64: 0.27
32: 0.35
16: 0.47

Figure 3.15: Motion blur and occlusion between frame pairs can still result in a poor similarity metric.

blocks containing little or no high frequency information. It is notable that there is little change in the colour distributions of the frame pairs in the examples given. It seems pertinent to exploit this in order to improve the discriminatory power of the algorithm.

Many studies have highlighted the comparison of colour distributions to detect shot cuts. Histogram-based methods represent the most common approach to shot cut detection in use today, since they offer a good trade-off between accuracy and computational efficiency [8, 61]. However, frames from different shots can have similar colour distributions such that the difference in the global histograms are of the same small magnitude as differences caused by motion or changes in the global illumination. For example, using a method to compare global colour histograms which results in a difference measure of 0 if the distributions are similar and 2 if they are different to compare the frame pairs in Fig. 3.16, results in a difference measure of only 0.032 and 0.061. Consequently, histogram methods may need to set a low threshold to detect all of the shot cuts, which in turn can result in a high number of false detections.

Methods based on the comparison of histograms of corresponding local regions are more sensitive to changes in the spatial distribution of the colours. For example, for the first frame pair in Fig. 3.16, using the median to combine the difference metrics of 4 corresponding blocks results in an overall difference measure of 0.21, while using 16 blocks equals 0.41 and 64 blocks results in a difference measure of 0.63. For the second frame pair, using 4, 16 and 64 blocks results in a difference measure of 0.16, 0.26 and 0.55 respectively. Whilst methods based on the comparison of histograms of corresponding regions are more sensitive to changes in the spatial distribution of the colours they also become more sensitive to object and camera motion than methods based on global comparisons. Changes in the global illumination negatively influence any algorithm based on histogram comparisons, although methods to quantise the colour component have been

Global comparison: 0.03
4 blocks: 0.21
16 blocks: 0.41
64 blocks: 0.63

Global comparison: 0.06
4 blocks: 0.16
16 blocks: 0.26
64 blocks: 0.55

Figure 3.16: Histogram methods based on the comparison of corresponding regions are more sensitive to changes in the spatial distribution than using a global comparison.

proposed to reduce the effect [63, 61].

Yusoff et al. [97] presented a shot cut detection technique using a combination of multiple "experts", where the experts themselves were stand-alone methods to detect shot cuts. They illustrated that the combination method gives significantly better results than these experts alone by exploiting the fact that the various methods calculate different features of the video sequence. If a shot cut is undetected by comparing one feature, it may be detected by another. Conversely, a false shot cut detected by one method might correctly be left undetected by another. This work highlighted that using a single feature for comparison is unlikely to be sufficient to detect all the shot cuts without incurring a number of false detections. In fact, the performance of different features can actually complement each other.

Two shots can have different colour distributions or similar colour distributions but a different spatial structure (such as those in Fig. 3.16). This suggests that two important features for comparison in detecting shot cuts are (i) colour distributions and (ii) the intrinsic structure. For this reason, our shot cut detection algorithm was extended to incorporate the comparison of colour distributions, in order to combat some of the shortcomings indicated earlier. Comparing the colour distributions of corresponding regions is more sensitive to changes in the spatial distribution of colours, compared with global colour histograms. Armed with this knowledge, we conclude the shot cut detection algorithm should incorporate a block-based histogram comparison.

In our shot cut detection algorithm, each block is motion compensated before comparing the histograms of corresponding regions. In other words, the colour histogram of each block in frame f_{n-1} is compared with the histogram of the best matching block in frame f_n. Incorporating a colour histogram comparison with the comparison of edge features results in a similarity vector

for each frame pair where the two components are:

1. a motion compensated similarity measure of the distribution of edge features in each frame, and
2. a motion compensated difference measure of the colour distributions of each frame.

This combination is based on the assumption that, ideally, during the same shot consecutive frames will have the same spatial structure and colour distributions. On the other hand, two frames separated by a shot cut should have different edge and colour distributions. Indeed, in an ideal case either metric would be sufficient to distinguish between frame pairs belonging to the same shot and frame pairs containing a shot cut. The benefit of using both metrics is that when one measure indicates a potential shot cut, the other measure can be used to verify if this is, in fact, the case.

If two frames contain little high frequency phenomena such that the similarity of edge features is relatively low (as shown in Fig. 3.13), then the two frames can be expected to have similar colour distributions. In addition, if there are multiple motions, or motions that invalidate the motion model even using the smallest block size, resulting in a relatively low median correlation coefficient (as shown in Fig. 3.14), corresponding blocks will still have similar colour distributions. In other words, if the similarity measure based on the edge features is relatively low during a shot such that a frame pair could be mistaken to contain a shot cut, the difference between the colour distributions should be smaller than that which occurs during a shot cut to prevent a false detection.

If there is a change in the global illumination during a shot such that the colour distributions between two frames are as different as when there is a shot cut, such as the frame pair in Fig. 3.17 which contains a camera flash, the edge distributions can still be expected to be similar, meaning that it is not falsely detected as a shot cut. The final situation to consider is if the difference between the colour histograms of two frames containing a shot cut is of the same small magnitude as differences caused by motion and illumination changes. Using the histogram difference on its own could make it difficult to differentiate between the two. However, if there is a shot cut the correlation will be poor, thus making a correct detection. For example, for the first frame pair in Fig. 3.16 the correlation measure equals 0.30 and for the second frame pair equals 0.40. If there is not a shot cut and the difference is caused by motion or an illumination change the correlation measure would be higher than it is during a shot cut preventing a false detection. Furthermore, this final situation is unlikely to occur because the histogram difference uses a block-based comparison not a global comparison thus increasing the histogram difference across a shot cut and each block is motion compensated prior to comparison therefore reducing any differences caused by motion. In other words, a shot cut will only be detected when the correlation is poor *and* there is a difference in the colour distributions as shown in Table 3.2.

Colour comparison:
Global: 0.51
4 blocks: 0.61
16 blocks: 0.78
64 blocks: 1.03
Optimal correlation measure:
Block size 32: 0.73

Figure 3.17: If there is a global illumination change between a frame pair in the same shot the similarity of edge features will remain high.

There have been many different methods to calculate the difference between colour and intensity histograms proposed for shot cut detection. These vary from using bin-wise differences, the chi-square statistic or histogram intersections combined with different colour spaces such as RGB, HSV, YIQ, Lab, Luv, Munsell and opponent colours [58]. There has also been many comparative studies characterising the various histogram-based shot cut detection algorithms, although Lienhart suggests that the performance improvements gained by choosing the right algorithm to classify the discontinuity far exceeds any improvement that can be gained by fine-tuning the colour space and difference metric [58]. For this reason, we have not focused our efforts on characterising the performance of different colour spaces and metrics ourselves but have chosen a 6-bit colour code to represent each RGB component and the chi-square statistic to compare the two binned data sets [74]. Let $H_i(k)$ be the number of events in bin k for block x_i in the first frame f_{n-1} and $H_j(k)$ the number of events in the same bin k for the corresponding block x_j in frame f_n. Then the chi-square statistic is

$$\chi_i^2 = \sum_k \frac{(H_i(k) - H_j(k))^2}{H_i(k) + H_j(k)} \tag{3.8}$$

where both distributions are normalised and $0 \leq \chi_i^2 \leq 2$ where 0 indicates equality. Nagasaka and Tanaka [63] used the chi-square statistic to reflect more strongly the difference between two colour distributions. Experiments performed by Zhang et al. [99] showed that while this metric enhances the difference between a frame pair containing a shot cut, is also increases the difference between frames representing small changes due to camera or object movements. Therefore, the authors concluded the overall performance was not necessarily better than that achieved by using the sum of absolute differences. However, Sethi and Patel [78] found this measure to perform best compared with two other tests. In addition, in our shot cut detection method differences caused by motion should be eliminated. We therefore found the performance of the chi-square statistic to be preferable.

Edge Features	Colour Distributions	Shot Cut?
Similar	Smaller Difference	No
Similar	Larger Difference	No
Poor	Smaller Difference	No
Poor	**Larger Difference**	**Yes**

Table 3.2: A shot cut will only be detected when the correlation is poor and there is a difference in the colour distributions.

For each block x_i in f_{n-1} the motion estimate determined from the normalised correlation is used to find its corresponding block in f_n. Then χ_i^2 is computed between the colour distributions of each block. Colour histograms are obtained by representing each pixel by a colour code. The colour code is determined by merging the two most significant bits of each RGB colour component resulting in 64 bins. The colour code is used to reduce the effect of luminance changes. A difference measure based on the colour distributions of all the blocks is then defined as

$$C_n = \text{median}\{\chi_i^2\}. \tag{3.9}$$

Thus, we obtain two values, E_n and C_n, which can then be used to detect any change in the visual content. A shot cut is detected if the correlation between edge features is poor and there is a difference between the colour distributions. To achieve this we use two thresholds T_E and T_C to detect shot cuts. If $E_n < T_E$ and $C_n > T_C$ a shot cut is detected. We can visualise the classification of shot cuts in this 2-D metric space as shown in Fig. 3.18. It can be seen that frame pairs from the same shot with a large histogram difference, but a high similarity value or with a poor similarity value and a small histogram difference (in the blue shaded areas), will not be detected as shot cuts. It is worthy of note that in both these cases, the frame pairs would be considered to contain a shot cut if only a single metric was used.

For each frame pair f_{n-1} and f_n, the optimal block size for the similarity measure E_n between the edge distributions of each frame is determined by the block size that returns the highest value for E_n i.e. the block size that returns the highest overall correlation measure. It is clear that a block-based comparison is more sensitive to changes in the spatial distribution of colours than a global histogram comparison. The examples shown so far suggest that the smaller the block size used, the more sensitive to shot cuts the method becomes. Figure 3.19 shows the histogram difference computed during a sequence using a block size of 128×128 (4 blocks) and 32×32 (64 blocks) with the latter shifted to the right by 5 frames to enable a comparison. It can be seen that as the number of blocks is increased the difference during a shot cut is also increased. In fact, a method based on region histogram differences was proposed by Nagasaka and Tanaka which used 16 blocks and an alternative approach was proposed by Ueda et al. which increased

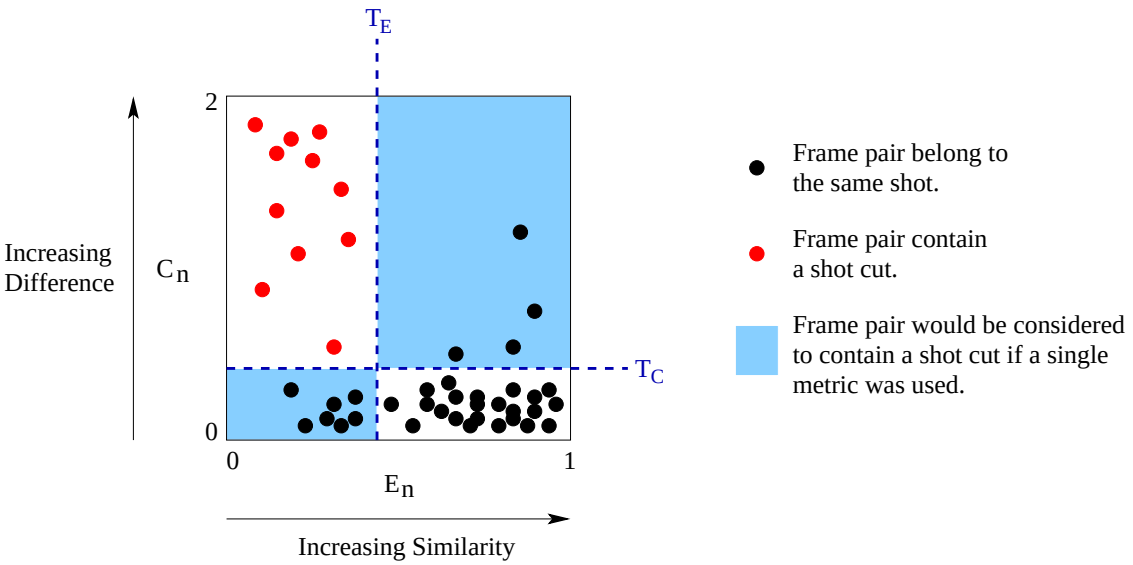


Figure 3.18: Classification of shot cuts using two metrics.

the number of blocks to 48 [63, 86]. This latter approach was found to be more sensitive to shot cuts [67]. Since emphasis was put on the blocks which changed most from one frame to the next it was also reported to be highly sensitive to motion [41]. In our approach, motion compensation is employed prior to each comparison thus reducing the sensitivity of smaller blocks to object and camera motion. Therefore, during a shot the histogram differences obtained at each block size are approximately equivalent.

The above observations imply that always comparing the histograms of the blocks at the lowest level of the hierarchical block matching will help improve the discriminatory power of the algorithm. However, it is possible that two shots can have similar colour distributions and for the spatial distribution of the colours to be similar even though the visual content is different. In such cases, using a larger block size can return a larger difference than a small block size. For each frame pair in Fig. 3.20, using a small block size results in the majority of the blocks having similar colour distributions. The difference is almost doubled by using a larger block size.

If there is a large difference between the colour distributions of two frames belonging to the same shot caused by a global illumination change, this is reflected in the difference metric obtained at every level. Therefore, the preferable block size for determining the histogram difference is actually that which returns the largest difference between two frames containing a shot cut and, as we have shown, this can vary. For this reason, we propose that for each frame pair the histogram difference at each block size is computed and the block size that results in the largest difference should be chosen. In reality, this is indeed the smallest block size for the majority of cases.

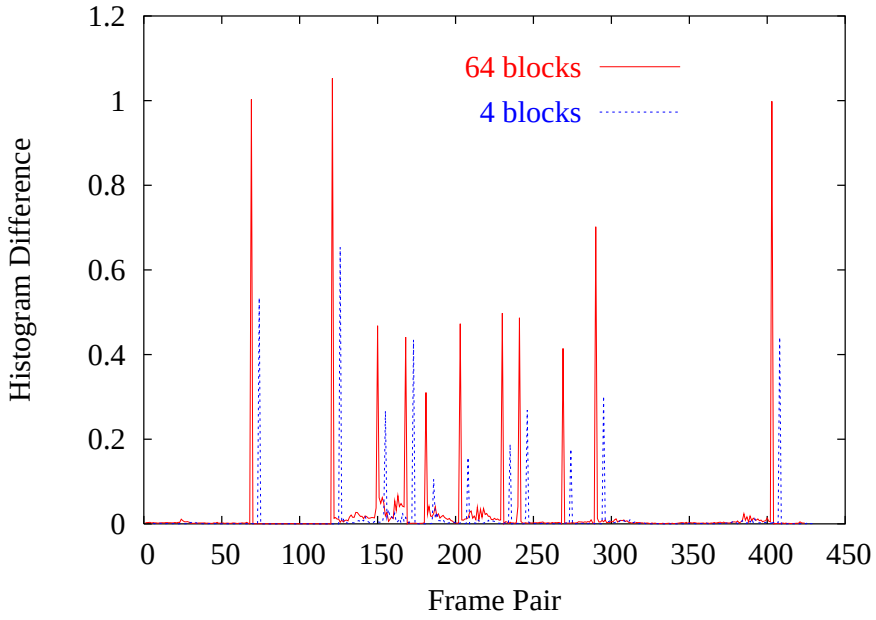


Figure 3.19: Using a smaller block size in the region-based histogram comparison can increase the colour difference between frame pairs containing a shot cut.

Global comparison: 0.23
4 blocks: 0.24
16 blocks: 0.08
64 blocks: 0.06

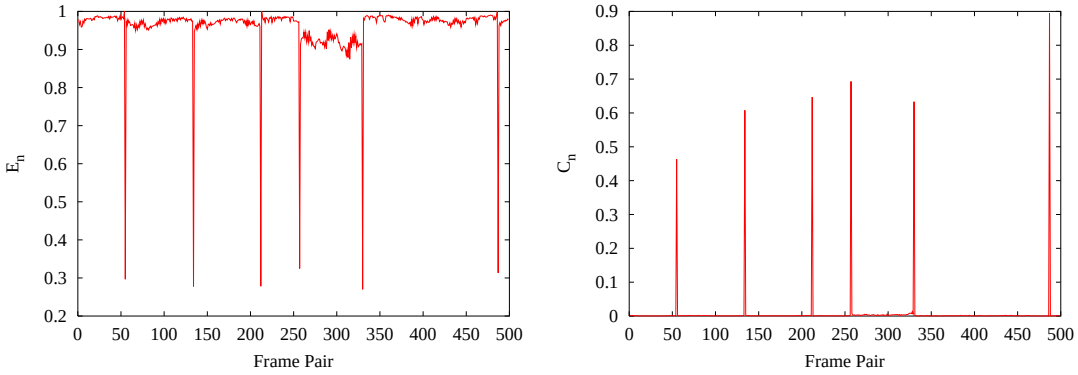
Global comparison: 0.31
4 blocks: 0.38
16 blocks: 0.35
64 blocks: 0.17

Figure 3.20: Histogram methods based on the comparison of small block sizes can result in the majority of the blocks having similar colour distributions if the spatial distribution of the colours is similar.

To summarise, hierarchical block-based motion estimation is employed between two frames to obtain a motion compensated similarity measure between the edge features contained in each frame. This measure can then be used to declare if there exists a shot cut between the two frames or whether the the frame pair belong to the same shot. The main drawback of this metric is that a relatively low measure can be obtained between two frames belonging to the same shot if an unsuitable block size is used due to the generalised aperture problem. For this reason, a similarity measure is acquired at each level of the hierarchical motion estimation and the block size which returns the largest similarity measure is used. The use of a flexible aperture size can substantially increase the similarity measure obtained during the same shot, thus helping to reduce the number of false shot cut detections. During a shot cut the similarity of edge features E_n between two frames will be poor therefore if E_n is less than a threshold T_E the frame pair is marked as a candidate shot cut.

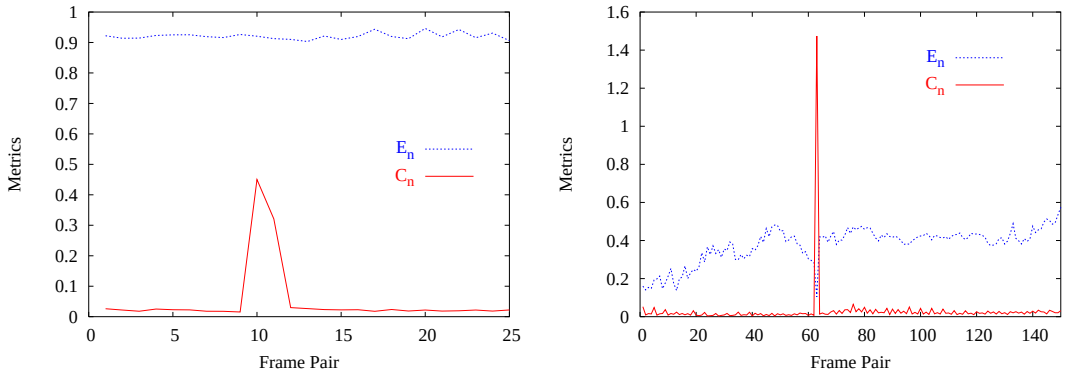
The use of an adaptive block size helps to overcome the generalised aperture problem. However, there can still exist relatively poor measures between two frames belonging to the same shot. Therefore, given a candidate shot cut, a measure of the histogram difference C_n between two frames is used to verify if this is a shot cut or not. The main drawback of using a histogram comparison is that two frames from different shots can have similar colour distributions. For this reason, a block-wise histogram difference at each level of the motion compensation is computed and the block size which returns the largest difference is used to confirm whether the candidate shot cut is a correct or not. Using the largest histogram difference helps reduce the number of missed shot cuts between two frames belonging to shots with similar colour distributions. Thus, if C_n is greater than a threshold T_C the candidate shot cut is declared to be correct.

Figure 3.21 shows a plot of E_n and C_n for a video sequence that contains 6 shot cuts. It can be seen that there is a large decrease in E_n and a large increase in C_n for each shot cut. In fact, for this sequence either measure alone would be sufficient to detect all of the shot cuts correctly. Figure 3.22(a) illustrates both metrics for 25 frame pairs during which there is an illumination change between two frame pairs. This would result in 2 false detections if only the histogram difference was being used. However, it can be seen that E_n remains high so they are correctly not detected. Conversely, Fig. 3.22(b) illustrates the metrics for the first 150 frames of an underwater sequence with 1 shot cut. Six evenly distributed frames from this sequence are shown in Fig. 3.23. It can be seen that in some cases there is little high frequency image phenomena and in addition there are shoals of fish which move very swiftly between frame pairs causing significant local motion and content change. The correlation of edge features is an unsuitable metric for measuring changes in the visual content of this sequence and using it alone would result in many false detections. Combining this metric with the histogram difference results in a correct detection of the shot cut and no false detections.



(a) Similarity of Edge Features (b) Difference of Colour Distributions

Figure 3.21: Metrics for a sequence containing 6 shot cuts.



(a) Using only C_n would result in 2 false detections. (b) Using only E_n would result in many false detections.

Figure 3.22: Combining the two metrics results in no false detections.

Figure 3.23: The similarity of edge features is an unsuitable metric for shots containing either little high frequency phenomena or significant local motion.

3.7 Results and Comparative Study

In this section, the performance of the proposed shot cut detection method is evaluated and compared with five alternative methods. These methods vary according to the scale of comparison used i.e. pixel-level, region-based or global-level and the features being compared. Firstly, we present in more detail the previous algorithms used for comparisons.

Pair-Wise Pixel Comparison (PC)

Arguably the simplest of the shot cut detection methods is the pair-wise pixel comparison of pixel intensities [99]. This method compares the intensity of each pixel in frame f_{n-1} with its corresponding pixel in frame f_n. If the difference in the intensity values of two pixels $DPC_n(\boldsymbol{p})$ is greater than some threshold T_{diff} the pixel is considered to have changed. This is defined as

$$DPC_n(\boldsymbol{p}) = \begin{cases} 1 & \text{if } |f_{n-1}(\boldsymbol{p}) - f_n(\boldsymbol{p})| > T_{diff} \\ 0 & \text{otherwise} \end{cases} \tag{3.10}$$

where $f_{n-1}(\boldsymbol{p})$ and $f_n(\boldsymbol{p})$ are the pixel intensities at point $\boldsymbol{p}$ in frames f_{n-1} and f_n respectively. Then, if the percentage of changed pixels is greater than some threshold T_{cut} the frame pair are considered to contain a shot cut. The detection of a shot cut can be represented by:

$$\frac{\sum_{\boldsymbol{p}} DPC_n(\boldsymbol{p})}{w \cdot h} \cdot 100 \begin{cases} > T_{cut} & \text{shot cut} \\ \leq T_{cut} & \text{no shot cut} \end{cases} \tag{3.11}$$

where $w \cdot h$ is the total number of pixels in each frame.

Block-Based Histogram Comparison (BH)

Instead of comparing individual pixels the block-based histogram method compares the colour distributions of corresponding regions [63, 86]. For this comparison we have used Ueda's approach which was found to be more sensitive to shot cuts and can be described as follows [86, 67, 41]. First, a frame is divided into 48 blocks. For each block x_i in frame f_{n-1}, the chi-square statistic χ_i^2 is computed between the colour histograms of x_i and its corresponding block x_j in frame f_n as defined in (3.8). Similarly, a 6 bit colour code is used to represent each RGB component, resulting

in 64 binned histograms. The discontinuity value is then defined as the number of blocks with a histogram difference greater than a threshold T_{diff}, that is

$$\sum_{i=1}^{48} DBH_n(i) \begin{cases} > T_{cut} & \text{shot cut} \\ \leq T_{cut} & \text{no shot cut} \end{cases} \tag{3.12}$$

with

$$DBH_n(i) = \begin{cases} 1 & \text{if } \chi_i^2 > T_{diff} \\ 0 & \text{otherwise} \end{cases} \tag{3.13}$$

Likelihood Ratio (LR)

Instead of comparing the colour distributions of corresponding blocks, the block-wise comparison used in the LR approach is based on the second-order statistical characteristics of their intensity values [99, 49]. Given a block x_i in the frame f_{n-1} and its corresponding block x_j at the same position in frame f_n, let μ_i and μ_j represent the mean intensity of each block respectively and let σ_i and σ_j denote the corresponding variances. The likelihood ratio between x_i and x_j is then defined as

$$lhr_i = \frac{[\frac{\sigma_i + \sigma_j}{2} + (\frac{\mu_i - \mu_j}{2})^2]^2}{\sigma_i \cdot \sigma_j} \tag{3.14}$$

A shot cut is declared if the number of blocks whose likelihood ratio exceeds a threshold T_{diff} is greater than some predefined threshold T_{cut}. The decision whether a shot cut has occurred is determined by

$$\sum_{i=1}^{48} DLR_n(i) \begin{cases} > T_{cut} & \text{shot cut} \\ \leq T_{cut} & \text{no shot cut} \end{cases} \tag{3.15}$$

where

$$DLR_n(i) = \begin{cases} 1 & \text{if } lhr_i > T_{diff} \\ 0 & \text{otherwise} \end{cases} \tag{3.16}$$

and each frame divided into a regular grid of 48 blocks. For each block, if its corresponding block is similar the likelihood ratio is approximately equal to 1. If they are different, the likelihood ratio is greater than 1.

Average Intensity Measure (AIM)

This method uses the average intensities over the entire image in each RGB colour channel to detect shot cuts [39]. Consider a colour frame $\boldsymbol{f}_n$ where each pixel has 3 colour components such that $\boldsymbol{f}_n(\boldsymbol{p}) = (R, G, B)^T$. Let the average of each RGB component of a frame $\boldsymbol{f}_n$ with width w and height h be defined as follows

$$\boldsymbol{A}_n = \frac{\sum_{\boldsymbol{p}} \boldsymbol{f}_n(\boldsymbol{p})}{w \cdot h} \tag{3.17}$$

In the implementation proposed by Hampapur et al. [39], after the averaging the absolute difference between the current frame and the next frame is divided by the absolute difference between the current frame and the previous frame, defined by

$$\boldsymbol{D}_n = \frac{|\boldsymbol{A}_n - \boldsymbol{A}_{n+1}|}{|\boldsymbol{A}_{n-1} - \boldsymbol{A}_n|} \tag{3.18}$$

To obtain the discontinuity value the difference in the average intensity of each colour channel is summed. A shot cut is then detected using

$$\sum \boldsymbol{D}_n \begin{cases} > T_{cut} & \text{shot cut} \\ \leq T_{cut} & \text{no shot cut} \end{cases} \tag{3.19}$$

In other words, if the sum of absolute differences is greater than a threshold a shot cut is declared. We initially used this implementation. However, if during a shot there was a relatively small change between the current frame the next frame but little or no change between the current frame and the previous frame (i.e. the denominator in (3.18) ≈ 0), then $\sum \boldsymbol{D}_n$ would frequently be large. This resulted in many false detections. Conversely, if there was a difference between the frame pair before a shot cut the discontinuity value indicating a shot cut could be relatively small. For

this reason, we redefined (3.18) as simply

$$\boldsymbol{D}_n = |\boldsymbol{A}_{n-1} - \boldsymbol{A}_n| \tag{3.20}$$

After comparing the performance of both implementations we found that using (3.20) improved the precision of the algorithm.

Global Histogram Comparison (GH)

Like the AIM the GH approach uses a global comparison and compares the colour histograms of successive frames [99, 63]. Again, this method uses the chi-square statistic to compare the difference between the global colour histograms of frames f_{n-1} and f_n. It is defined as

$$\chi_n^2 = \sum_k \frac{(H_{n-1}(k) - H_n(k))^2}{H_{n-1}(k) - H_n(k)} \tag{3.21}$$

where $H_{n-1}(k)$ is the number of events in bin k for frame f_{n-1}, $H_n(k)$ the number of events in bin k for frame f_n, both distributions are normalised and $0 \leq \chi_n^2 \leq 2$. Each RGB component is represented using a 6 bit colour code resulting in 64 binned histograms. If this difference is greater than a threshold T_{cut} a shot cut is detected i.e.

$$\chi_n^2 \begin{cases} > T_{cut} & \text{shot cut} \\ \leq T_{cut} & \text{no shot cut} \end{cases} \tag{3.22}$$

Motion-Based Method (MB)

This is the shot cut detection method proposed in this thesis. So far, we have shown examples using 5 levels of hierarchical motion compensation. In this method, there is a trade-off between the precision gained by using many levels of motion compensation and the computational efficiency. In most cases, we have found 3 levels sufficient and present results in this thesis using a top block size of 128 and a bottom block size of 32. For this implementation each frame is scaled to have dimensions 256×256.

Sequence	No. of frames	No. of shot cuts	Genre
1	17558	143	Film
2	9038	80	Documentary
3	4877	57	Documentary
4	3086	51	Sport
5	2633	34	Comedy series
6	3995	33	Comedy series
7	3573	22	Documentary
8	5986	19	Drama series
9	214	14	Advert
10	500	12	Cartoon
11	6463	12	Promotional Video
12	425	11	Film
13	500	6	Film
14	561	5	Documentary
15	500	5	Chat show
16	625	4	Documentary
17	478	4	News
18	500	2	Sport
ALL	61512	514	Various

Table 3.3: Test data used to compare shot cut algorithms.

Test Data

To test these methods we used 18 different video sequences totalling 61512 frames. All sequences contained only shot cuts amounting to 514 transitions. The work in this thesis has not targeted an application for any particular category of media such as a digital video library containing only wildlife footage or historic news footage. For this reason, the sequences were chosen to include different genres, varying from film, documentary, drama and comedy series, cartoon, sport and news. The number of frames, shot cuts and the category of each sequence is summarised in Table 3.3. The locations of these transitions were hand labelled to obtain a ground truth to evaluate the performance of each algorithm. Two parameters often used to evaluate and compare the performance of shot cut detection algorithms are recall and precision [61, 8, 33]. These evaluation criteria are commonly used in the field of information retrieval [87]. When searching and retrieving documents from a large collection, intuitively a good information retrieval system should try to maximise the retrieval of relevant documents, and minimise the retrieval of irrelevant documents. Likewise, a good shot cut detection algorithm should try to maximise the number of actual shot cuts detected, whilst minimising the number of false detections. When detecting shot cuts in a sequence, there are four types of detections an algorithm can make, summarised in Table 3.4. Recall

	Shot Cut	Non Shot Cut
Detected as Shot Cut	C	F
Not Detected	M	N

Table 3.4: Four types of detection an algorithm can make.

and Precision can then be defined by

$$\textit{Recall} = \frac{C}{C+M} \tag{3.23}$$

$$\textit{Precision} = \frac{C}{C+F} \tag{3.24}$$

In other words, recall is the percentage of true transitions detected and precision is the percentage of detected transitions that are actually correct (precision defines the level of "noise" in the transitions detected by the algorithm). In an ideal world, recall and precision would both be equal to 1.0. However, it is usually difficult to achieve a high level of recall without sacrificing precision. For example, it would be trivial to achieve a recall equal to 1.0 by detecting all frame pairs as shot cuts, although the precision would be poor. Conversely, a precision equal to 1.0 could be achieved by detecting just one shot cut, resulting in a poor recall. In reality, an algorithm attempts to maximise both recall and precision simultaneously. It is worth noting, that the lowest precision obtainable is defined by the percentage of total frame pairs that are shot cuts. A common property amongst all of these algorithms is that they involve at least one thresholding operation to declare a shot cut. Consequently, the performance of these algorithms is dependent on the values chosen for any parameters used. With this in mind, a recall/precision curve (RP curve) can be constructed for each sequence by varying the value of one or more thresholds between its extremes, which then illustrates the performance trade-offs available for a given algorithm on that data. RP curves for each of the six algorithms over sequence 1 and 2 are shown in Fig. 3.24 and Fig. 3.25 respectively. RP curves can also be used to obtain the parameter set which resulted in a particular operating point. For example, to guarantee a recall of 0.9 on sequence 1 the RP curves for each algorithm can be compared to decide which algorithm resulted in a higher precision at that recall value. Then, for the chosen algorithm the parameter set used to obtain that performance can be determined.

We are not concerned with the performance of an algorithm on one particular genre of media. More importantly, the performance evaluation of these methods should reflect how well they could potentially perform on unseen data. In other words, it must be evaluated if the performance of a parameter set for an algorithm generalises well to other sequences. For example, the parameter set that resulted in a recall and precision equal to 1.0 on sequence 2 for the MB algorithm may have returned a similar recall for sequence 1 but with a much lower precision, i.e. if the performance of

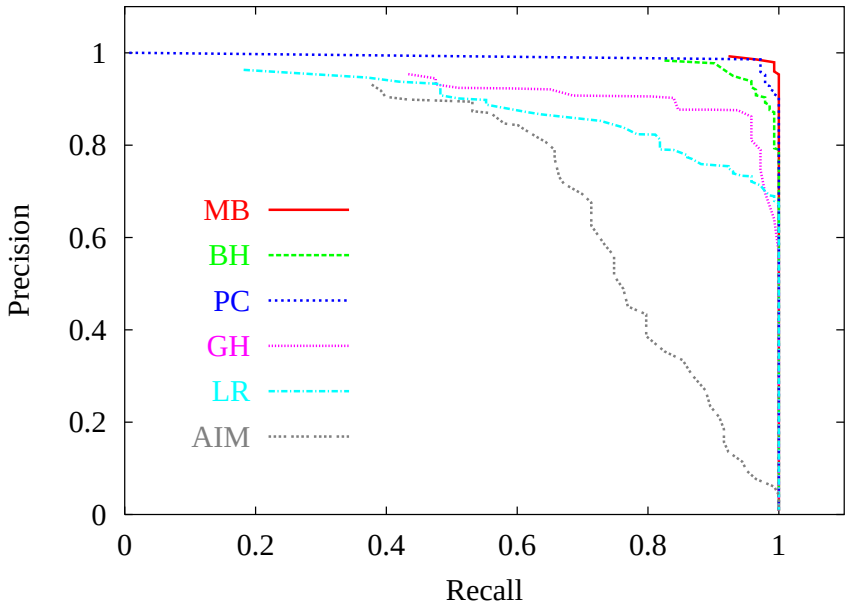


Figure 3.24: RP curves for each algorithm on sequence 1.

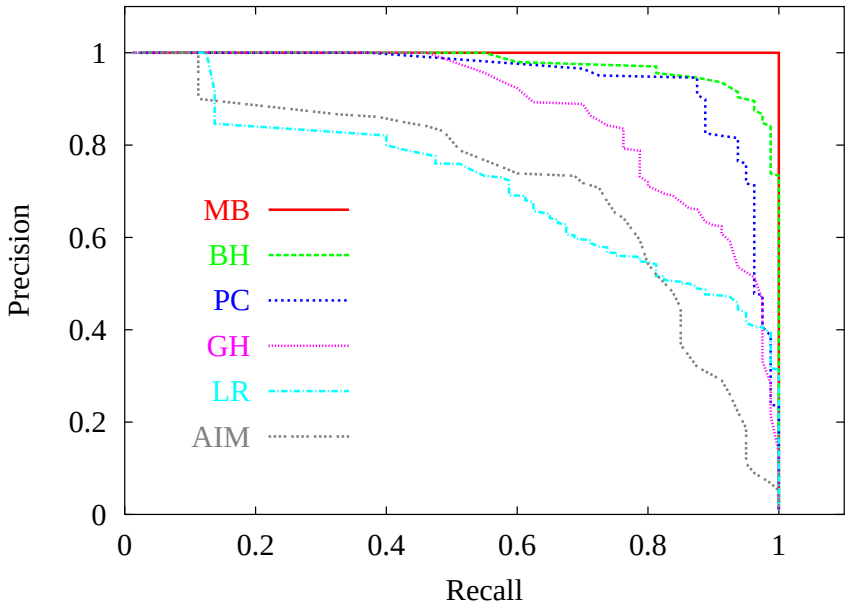


Figure 3.25: RP curves for each algorithm on sequence 2.

a parameter set generalises well, then its corresponding recall-precision pair should always appear in a similar position on the recall/precision graph.

Based on this observation, we do not want to evaluate the performance of each algorithm on one particular sequence, but rather the performance over the complete data set. To achieve this, for each different parameter set for an algorithm, the total number of correctly detected, falsely detected and missed shot cuts over all the sequences are determined. These values can then be used to produce a RP curve showing the performance trade-off available with each algorithm on the complete data set. Fig. 3.26 shows the RP curve for each of the six algorithms over all the sequences. It can be seen that for any recall value the proposed method (MB) results in a higher precision value and dominates over the entire performance space. Consequently, on this data set the proposed method resulted in a better performance than the alternative algorithms.

It is worth noting that an alternative approach to evaluate the performance of each algorithm over all the sequences could be to compute the average recall and precision value obtained by each parameter set (this is known as micro-evaluation when evaluating information retrieval systems [87]). For example, given 3 sequences with the number of shot cuts shown in Table 3.5, two thresholds T_A and T_B detect the number of shot cuts shown in columns 3 and 5 respectively. The corresponding recall values are shown in columns 4 and 6. Using the average recall value to evaluate the performance of each threshold would result in a recall value of $(0.5 + 0.9 + 0.9)/3 = 0.76$ for T_A and $(0.9 + 0.5 + 0.5)/3 = 0.63$ for T_B. Conversely, using the total number of correctly detected and missed shot cuts to compute a recall value for each threshold would result in $518/(518 + 502) = 0.51$ for T_A and $910/(910 + 110) = 0.89$ for T_B. Assuming this resulted in no false detections for each threshold, the optimal performance would be achieved by the threshold which detects the most shot cuts i.e. the threshold which results in the maximum recall value. Using the average recall value this would be T_A and using the total number of correct detections would indicate T_B resulted in the best performance.

If the performance of an algorithm on each sequence is considered equally important, then the average recall should be used. In other words, if the sequences contain a different number of shot cuts and are of different lengths but every shot cut detected is equally important then threshold T_A would be chosen as it returned a higher recall. However, if the performance over the whole data set is important and the performance on each sequence should be weighted then computing the recall over all the sequences should be used as it is weighted by the number of shots cuts in each sequence. For example, detecting 90% of the shot cuts contained in sequence 1 would be preferable to detecting 90% of those in sequences 2 and 3. Thus, threshold T_B would result in the optimal performance. In this work, the total number of correctly detected, falsely detected and missed shot cuts were used to determine the recall and precision values to evaluate the performance

Sequence	Total no. of shot cuts	No. of shot cuts detected using T_A	Recall for T_A	No. of shot cuts detected using T_B	Recall for T_B
S_1	1000	500	0.5	900	0.9
S_2	10	9	0.9	5	0.5
S_3	10	9	0.9	5	0.5

Table 3.5: Assumed number of detected shot cuts for an algorithm using two different threshold values.

of the shot cut detection methods. If there is not one algorithm that can be declared better than the others, RP graphs can be used to aid in the choice of algorithm. The algorithm chosen as the best often depends upon the performance requirements. For example, considering the two methods LR and AIM between the recall values 0.0 and approximately 0.7, AIM returns the highest precision. However, if an application required a recall greater than 0.7 then LR results in a higher precision value and would be most preferable. One measure of performance that takes into account both recall and precision is the F-measure, defined as

$$F_\alpha = \frac{P \cdot R}{(1-\alpha)P + \alpha R} \tag{3.25}$$

where P equals precision, R equals recall and α defines the importance of recall and precision in the performance [87]. When $\alpha \to 0$, $F_\alpha \to R$ which corresponds to no importance being attached to precision. Alternatively, when $\alpha \to 1$, $F_\alpha \to P$ which corresponds to no importance being attached to recall. Finally, when $\alpha = 0.5$, F_α corresponds to equal importance being attached to recall and precision. This measure, $F_{0.5}$ is most commonly used and is known as the harmonic mean of recall and precision

$$F_{0.5} = \frac{2PR}{P+R} \tag{3.26}$$

Both recall and precision need to be high for the harmonic mean to be high. Consequently, if the harmonic mean is computed for each operating point on a RP curve, the maximum value is a good indication of the best recall/precision compromise. Table 3.6 ranks the performance of each algorithm with respect to the maximum harmonic mean value. It can be seen that the proposed algorithm achieves the optimal performance with 99.61% recall and 98.65% precision. If there were no specified performance requirements for an algorithm, i.e. recall and precision are considered equally important, the algorithm which returns the maximum compromise between recall and precision could be chosen as the "best" algorithm.

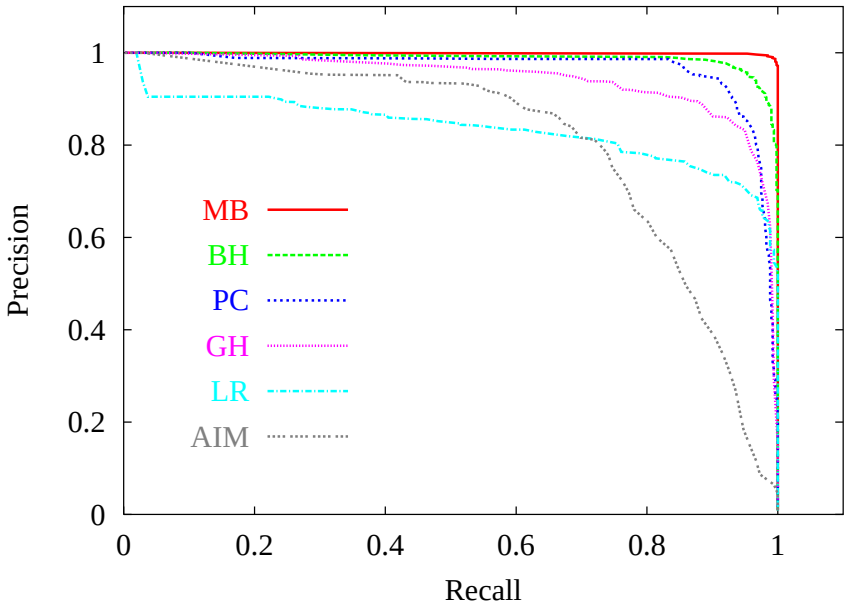


Figure 3.26: RP curves for each algorithm on the complete data set.

However, in real world applications this is rarely the case and specific performance criteria are required. For example, shot cut detection algorithms could be used to assist a user in labelling the shot cut boundaries. In such a scenario, it may be preferable to detect all of the shot cuts at the cost of detecting false transitions. It would be quicker and easier for the user to browse potential shot cuts and label the correct transitions than to observe an entire video sequence to detect missed shot cuts i.e. the consequence of missing a shot cut is more severe than returning a false detection. Based on this observation the performance of each algorithm can be compared with respect to the maximum precision value obtained when the recall value equals 1.0. Table 3.7 presents the performances of each algorithm represented as precision with recall $= 1$. It can be seen that the proposed algorithm can detect all of the shot cuts whilst detecting only 15 false positives—a better performance than the other algorithms. Finally, for any shot cut detection algorithm that uses a global threshold to work well it is important that the difference (or similarity) values obtained for cut and non-cut frame pairs are clearly separated for all video sequences. The distribution of E_n and C_n values used in the MB method for cut and non-cut frame pairs over all the sequences are shown in Fig. 3.27. It can be seen that there is little overlap between the two distributions and that the threshold values used to maximise the harmonic mean separate the two distributions well resulting in a high recall and precision value.

Algorithm	Threshold values		C	M	F	R	P	$F_{0.5}$
MB	$T_E = 0.42$	$T_C = 0.22$	512	2	7	0.9961	0.9865	0.9913
BHXS	$T_{diff} = 0.36$	$T_{cut} = 30$	490	24	22	0.9533	0.9570	0.9552
PC	$T_{diff} = 15$	$T_{cut} = 0.70$	468	46	28	0.9105	0.9435	0.9267
GH		$T_{cut} = 0.12$	474	40	77	0.9222	0.8603	0.8901
LR	$T_{diff} = 1.512$	$T_{cut} = 30$	472	42	170	0.9183	0.7352	0.8166
AIM		$T_{cut} = 27$	370	144	86	0.7198	0.8114	0.7629

Table 3.6: Evaluation based on the harmonic mean—equal importance is attached to recall and precision.

Algorithm	Threshold values		C	M	F	R	P
MB	$T_E = 0.4$	$T_C = 0.08$	514	0	15	1.0000	0.9716
BH	$T_{diff} = 0.25$	$T_{cut} = 20$	514	0	218	1.0000	0.7022
LR	$T_{diff} = 1.064$	$T_{cut} = 37$	514	0	479	1.0000	0.5176
PC	$T_{diff} = 5$	$T_{cut} = 0.73$	514	0	1870	1.0000	0.2156
GH		$T_{cut} = 0.01$	514	0	2818	1.0000	0.1543
AIM		$T_{cut} = 3$	514	0	10883	1.0000	0.0451

Table 3.7: Performance evaluation based on the detection of all the shot cuts—maximum precision when $R = 1$.

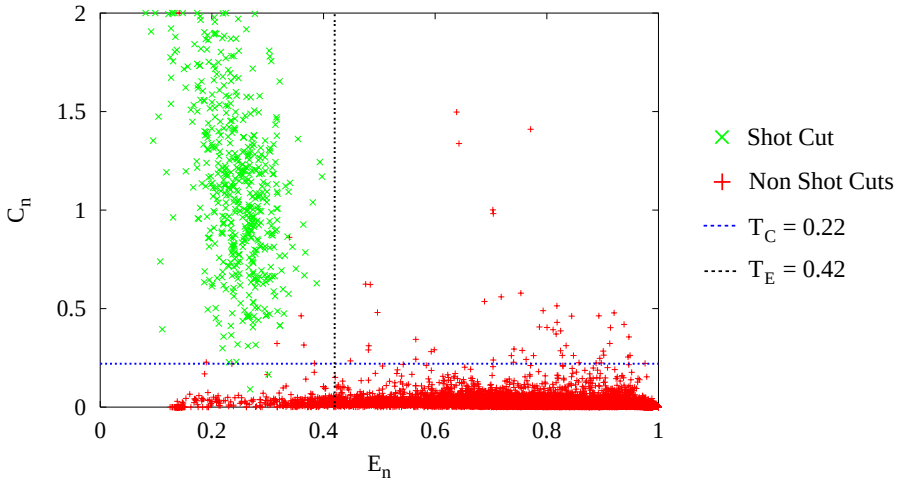


Figure 3.27: There is little overlap between the distributions for cut and non-cut frame pairs.

3.8 Summary

A method for detecting shot cuts was presented, employing hierarchical motion compensation using a normalised correlation measure in conjunction with local colour histogram differences. The correlation metric matches image features corresponding to high-frequency phenomena, such as edges, corners and certain types of textures. The use of this measure was motivated by the observation that the distribution of edges differ between frames separated by a shot cut. However, although effective in the majority of cases, correlation alone is not suitable in every situation. Occasionally, images may lack the necessary level of high frequency information in order to enable reliable correlations to be obtained. This may be due to a general lack of edges, soft focus, motion blur or significant local motions. In order to deal with such situations a local colour histogram difference metric was introduced, based on the observation that frames within a shot typically have similar local colour distributions. These two metrics were found to complement each other. The chapter concluded with a comparative study of the performance of the proposed algorithm and a number of existing techniques. The proposed algorithm was found to perform the best over the data used for the experiments.

Chapter 4

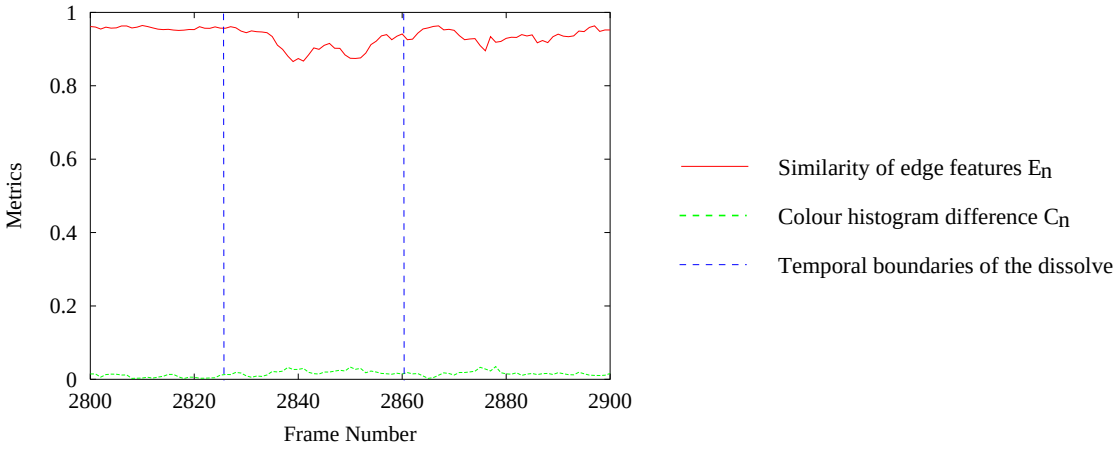
Gradual Shot Transition Detection

Shots unified by a common event, locale and time are grouped together into scenes. Gradual transitions are often used at scene boundaries to emphasise the change in content of the video sequence [8]. Hence, detecting gradual transitions is particularly important for the identification of key frames to provide an efficient index. A comparison of recent algorithms shows that the false positive rate when detecting dissolves is usually unacceptably high, indicating that reliable dissolve detection is still an unsolved problem [56]. In this chapter we extend our shot cut detection method for the detection and classification of the most commonly used gradual transitions: fades and dissolves.

4.1 Motion-Based Gradual Transition Detection

In the case of shot cuts, the content change occurs between two consecutive frames and can be detected by comparing the difference between image features that represent the visual content, as described in the previous chapter. During a shot cut the inter-frame difference is usually large and a shot cut is detected when the difference exceeds a predefined threshold. However, during a gradual transition the inter-frame difference is much smaller than that which occurs during a shot cut. For this reason, it is difficult to detect gradual transitions by comparing consecutive frame pairs alone. The 4 frames shown in Fig. 4.1(a) come from a dissolve which has been used to indicate a passage of time between the tennis player winning the match and later receiving the trophy. A plot of the similarity of edge features E_n and the colour histogram difference C_n during this dissolve is shown in Fig. 4.1(b). The dissolve occurs between frames 2826 and 2860 and it can be seen that there is little indication of any change. In addition, the inter-frame difference during a gradual

(a) Four frames illustrating a dissolve.



(b) The inter-frame difference during a gradual transition is usually small.

(c) Comparing the the visual content between the two frames at the start and end of the transition results in a difference comparable to a shot cut.

Figure 4.1: It is difficult to detect gradual transitions by comparing consecutive frame pairs.

transition can be of the same magnitude as differences caused by camera and object motion. This makes it difficult to distinguish between changes caused by a gradual transition from those caused by such motions. Simply lowering the shot cut threshold to detect these small changes would result in a large number of false detections.

Nevertheless, it can be seen in Fig. 4.1(c) that if the visual content of each shot connected by the dissolve was compared directly the difference would be large; comparable to that which occurs during a shot cut. This suggests that gradual transitions can be detected by measuring the difference between the visual content of temporally distant frames using metrics similar to those used in shot cut detection. If the difference between two frames far apart is greater than some threshold there could potentially be a gradual transition between them. This concept was first proposed by

Zhang et al. who used a "twin comparison" technique which compares the histogram difference with two thresholds, a high and a low one [99]. Whenever the histogram difference between two consecutive frames is greater than the higher threshold, a shot cut is detected. If the difference lies between the two thresholds the frame is marked as a potential start of a gradual transition. Successive frames are then compared with the first frame of the transition and if the difference exceeds the high threshold, a gradual transition is detected. The end of the gradual transition is marked once the difference between frame pairs drops below the low threshold for two frame pairs.

Zhang et al.'s method addressed the fact that if there exists a gradual transition between two shots then the difference between the first frame of the transition and the last should be of the same magnitude as if a shot cut existed between them. However, this approach can fail when camera operations, such as pans, generate a change in the colour distribution similar to that caused by a gradual transition. To overcome this, they suggested analysing the motion between frames to identify camera operations such as pans, tilts and zooms. Where this type of motion is identified, the gradual transition is assumed to be false to reduce the number of false positives. However, this means that gradual transitions containing camera motion will not be detected. In addition, this method depends on a small change between consecutive frames to trigger the comparison between frame pairs that are further apart. As shown in Fig. 4.1(b), sometimes the difference is so small it never exceeds the lower threshold and the transition is missed.

In summary, methods that detect gradual transitions by locating a sustained increase in the difference metric used for shot cut detection are susceptible to two different factors.

1. Small changes caused by camera and object motion can result in false detections.
2. The difference during a gradual transition is not sufficient for the change to be detected.

To address these issues, motion-based algorithms have been proposed which aim to eliminate any differences between frame pairs caused by camera and object motion so that any change must be the result of an edit effect. Shahraray used a weighted sum of the motion-compensated pixel differences as the disparity metric to detect shot cuts and detected gradual transitions if there existed a sustained small increase [79]. The method proposed by Zabih et al. to detect shot cuts was also extended to detect gradual transitions by examining the relative values of entering and exiting edge percentages [98]. However, despite the motion compensation employed in both these methods, changes caused by a gradual transition were still difficult to distinguish from those caused by motion resulting in a large number of false detections [56, 58].

In an attempt to overcome the problem of motion, other methods detect fades and dissolves by examining the temporal behaviour of the variance of the pixel intensities. This approach was first

proposed by Alattar [3, 4] but has been used and modified by other authors as well [29, 84, 60, 62, 64]. Such methods model a dissolve between two sequences g and h as

$$f_t(\boldsymbol{p}) = \alpha_t \cdot g_t(\boldsymbol{p}) + (1 - \alpha_t) \cdot h_t(\boldsymbol{p}) \qquad \text{for } 0 \leq t \leq T \tag{4.1}$$

where T is the length of the transition and α_t is a decreasing function from $\alpha_t = 1$ at the start of the transition to $\alpha_t = 0$ at the end. A fade transition or 'dip to colour' can be defined as a special case when either g or h contains only solid colour frames. Fades and dissolves can, therefore, be represented by the same model. The majority of these methods also depend on two assumptions [64].

1. That α_t decreases linearly.
2. There is little or no motion in the sequences g and h, i.e. g and h are ergodic processes.

Given this ideal model it can be shown that the variance of the pixel intensities in each frame follows a parabolic pattern during a dissolve [41]. Detecting a dissolve then becomes a problem of detecting the parabolic pattern in the variance sequence. Although these assumptions in the model result in a computationally efficient detection algorithm, actual dissolves are not that simple [64]. During many gradual transitions, α_t does not decrease linearly [64, 60]. Particularly during artistic transitions, there may be a pause, a long lead-in time or some other non-linearity in α_t [48]. Creating transitions is an art form in itself. Indeed, the connections between shots are often as important as the shots themselves as they give rhythm and style to the film [50]. In addition, if α_t does decrease linearly, the linearity is often distorted by a wide variety of post-processing operations such as compression coding and filtering [64].

There can also exist camera and object motion before, during and after a gradual transition. It is most important that any gradual transition appears smooth. However, if for example two different shots contain similar camera motion a dissolve can be used to connect them thus resulting in a continuous motion throughout the transition. A further problem is that large object and camera motion during a shot can introduce similar parabola shape in the variance sequence [60]. As a result the parabolic pattern is not sufficiently pronounced and many algorithms lack robustness when detecting gradual transitions in real video sequences [41, 64]. For example, Fig. 4.2 shows a plot of the variance of the pixel intensities for each frame over a sequence which contains one dissolve and a fade-out. The temporal bounds of each transition are shown and even by human eye the parabolic pattern relating to the dissolve is difficult to extricate.

It is noticeable that the end of the fade-out is marked by at least one frame with variance approxi-

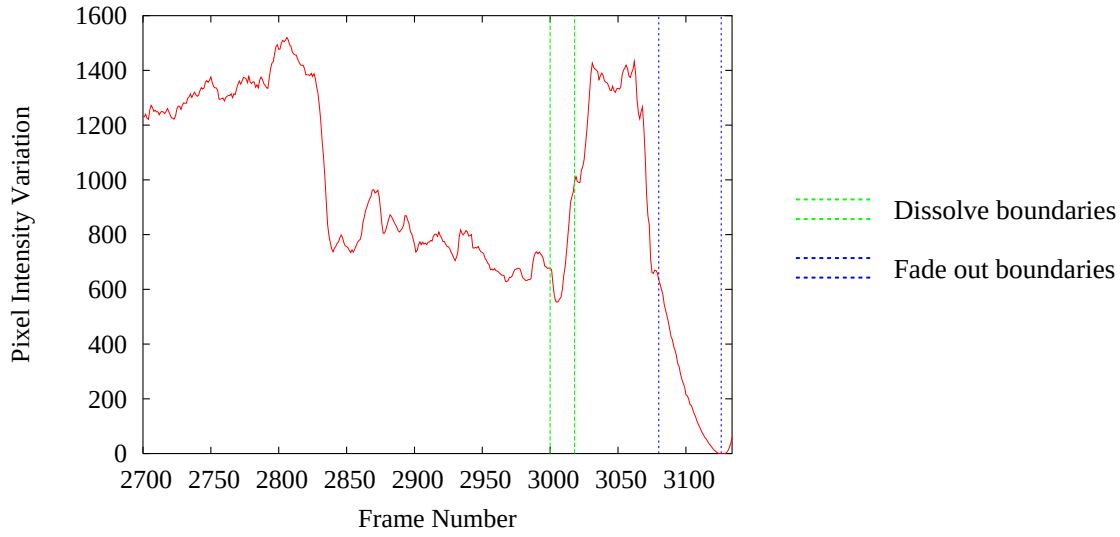


Figure 4.2: The parabolic pattern relating to a dissolve can be difficult to extricate due to noise and motion.

mately equal to zero. Lu et al. noted that this is an important feature that can be used to detect the start of a fade-in and the end of a fade-out [60]. Moreover, the frame variance distinctly decreases during the fade-out and increases during a fade-in, i.e. the transition is monotonic. We describe an approach to detect fade transitions that starts by locating frames with pixel intensity variation close to zero. The shot cut detection method is then used to distinguish between fades and cuts to/from a constant frame. If the frames are declared to mark the start of a fade-in, the end of the transition is detected by analysing the mean and standard deviation of the pixel intensities in the following frames. A similar comparison is used to detect the start of a fade-out.

The majority of the existing methods for detecting gradual transitions, particularly dissolves, are weakened by camera and object motion. Based on the above observations it seems clear that it is not sufficient to consider only the difference between consecutive frame pairs to detect dissolves. Frames that are further apart must be compared to detect a change in the visual content analogous to that of a shot cut. In addition, this comparison must be performed continuously not only when there exists small local changes. However, the visual content between two frames that are far apart can potentially be very different if there exists camera and object motion between them. For example, Fig. 4.3 illustrates four frames from a shot during which the camera pans right and zooms-in and another character enters the setting. It can be seen the contents in the first frame are incomparable with those in the last. Therefore, we conclude that an algorithm to detect dissolves must compare frames that are temporally distant whilst still eliminating differences between them caused by camera and object motion.

Figure 4.3: Camera and object motion can cause the visual content of a shot to change significantly.

Ideally, the contents of a frame (potentially the first frame in the shot) should be compared with every successive frame to detect a dissolve. If there exists no camera or object motion until the occurrence of a gradual transition, the contents between the first frame and the current frame will start to differ as the transition starts and will be significantly different by the end. However, in reality there does exist camera and object motion during a shot. Therefore, an algorithm must be able to adapt which content from previous frames is being compared with the current frame. In the presence of camera or object motion only content from a previous frame that is still present in the current frame should be compared. Hence the content will not appear to have changed significantly.

To achieve this we extend our shot cut detection method to track blocks through the video sequence. It monitors which blocks from previous frames are still present in the current frame and have not been removed due to camera or object motion. It then compares the difference between each block's contents in the frame it was selected from and the current frame at its new position. If the difference is significantly large for the majority of the blocks being tracked a gradual transition is detected. Moreover, this method does not rely on an ideal model to detect dissolve transitions. Therefore, it can also detect the less common type of dissolve known as an additive dissolve. Many previous approaches are restricted to detecting the most common dissolve type which are cross-dissolves [58]. This method to detect dissolves is outlined in Fig. 4.4.

The proposed method for detecting gradual transitions has 3 distinct stages:

1. Detect fade transitions.
   (a) Detect frames with zero variance to mark the start of a fade-in or the end of a fade-out.
   (b) Use the shot cut detection method to differentiate between fades and shot cuts to/from blank images.
   (c) If declared a fade, determine the start of a fade-out or the end of a fade-in.
2. Detect shot cuts.
   (a) Apply the shot cut detection algorithm between fade boundaries to detect shot cuts.

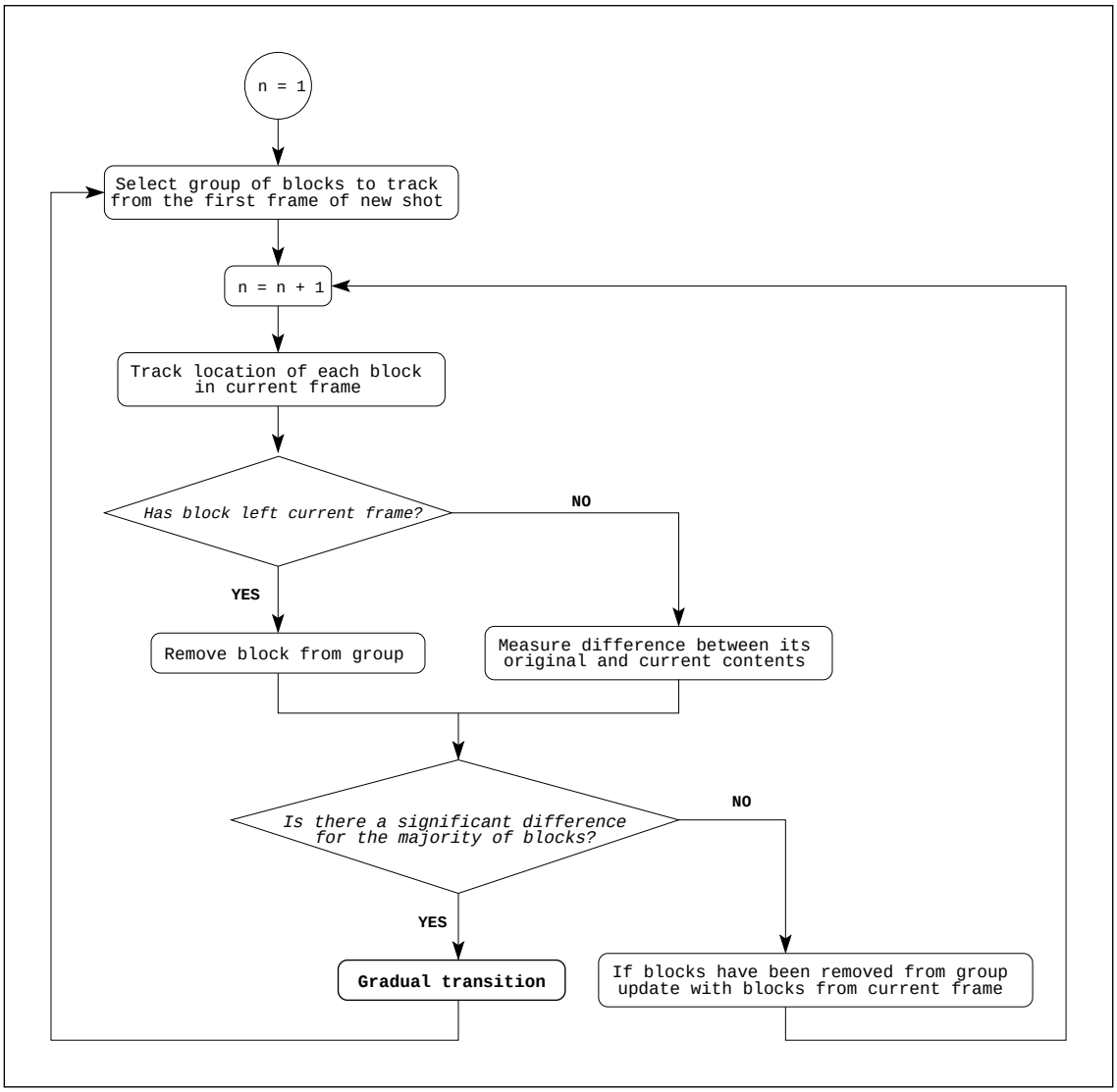


Figure 4.4: Method outline to detect dissolves.

3. Detect dissolves.

   (a) Assume the only possible transitions in the sub-sequences between shot cut and fade transition boundaries are dissolves.

   (b) Track blocks through sub-sequence.

   (c) If the content of the majority of the blocks changes significantly, a dissolve is detected.

In the following sections, we discuss in more detail how the boundaries of fade and dissolve transitions are detected to complete the segmentation of a video sequence into its individual shots.

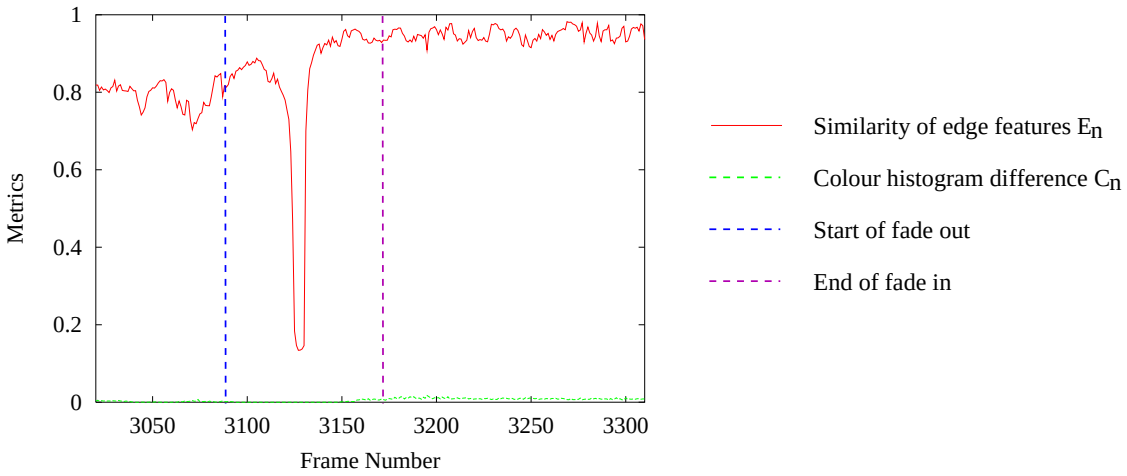


Figure 4.5: During a fade transition the colour histogram indicates little change.

4.2 Fade Detection

The first step in this method is to detect fade transitions. The end of a fade-out and the start of a fade-in is marked by a blank image. However, simply detecting such images is not sufficient to declare a fade transition as there can also exist shot cuts to or from a constant image. If a shot transition detection algorithm is to be used as the initial step in creating a video index it is important that it can detect the boundaries of gradual transitions accurately. A key frame selected from the middle of a gradual transition will not convey the contents of either of the adjoining shots properly. Therefore, the algorithm must be able to distinguish between fades and shot cuts to/from a constant image for the boundaries of the fade transition to be detected properly.

Let σ_n define the standard deviation of the pixel intensities in frame f_n. If f_n is a constant colour $\sigma_n = 0$. In practise $\sigma_n \approx 0$ due to the presence of noise in the image. Hence a frame is marked to be a blank image if $\sigma_n < T_{sd}$. In our approach T_{sd} was constant and equal to 1. Once a blank frame has been detected there must have either been a fade-out or a cut to a blank frame. Let f_n be a blank frame and f_{n-1} be the previous frame such that $\sigma_{n-1} \geq T_{sd}$. To determine if f_n marks the end of a fade-out or if the frame pair contain a shot cut, the metrics E_n and C_n can be computed. The similarity of edge features E_n will be poor. However, if f_n marks the end of a fade-out which is a gradual transition the difference between the colour distributions should be small and will not indicate any change. This is illustrated in Fig. 4.5 which presents the metrics E_n and C_n during a sub-sequence of video that contains a fade-out followed by a fade in. It can be seen that there does not exist any change in C_n. On the other hand, if the frame pair contain a shot cut then both these measures will indicate a significant change. This is shown in Fig. 4.6 which shows a plot of E_n and C_n for a section of video that contains a shot cut to black followed by a fade-in.

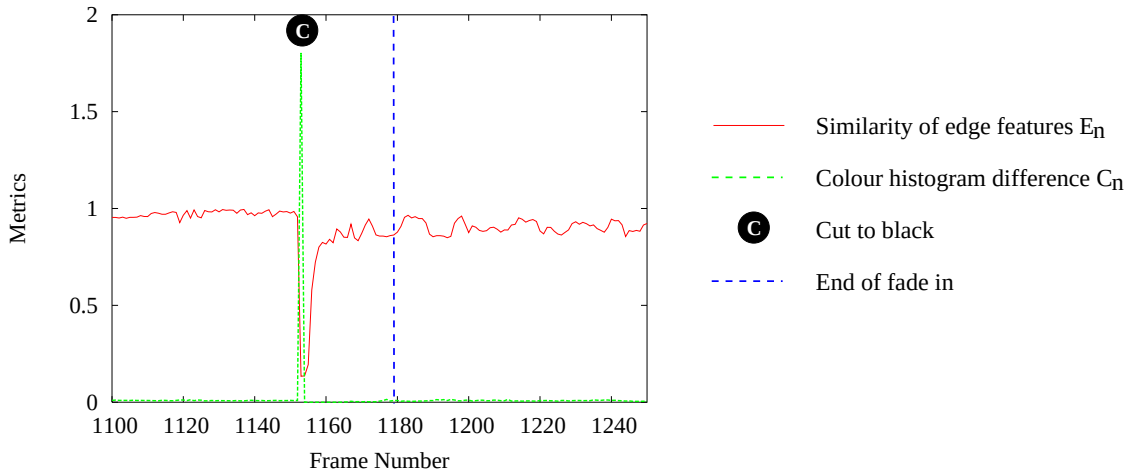


Figure 4.6: During a shot cut to a constant frame both metrics indicate a significant change.

If the constant frame is declared to mark the start/end of a fade transition then the opposite bound must be ascertained. A fade is a scaling of the pixel intensities over time which can be be observed by a parabolic pattern in the variance of the pixel intensities. Furthermore, this is reflected by an approximately linear scaling in the standard deviation of the pixel intensities σ_n as shown in Fig. 4.7 which illustrates σ_n during a fade-in. Reliable fade detection can not be achieved by assuming the increasing rate of change is constant during the transition. However, it is noticeable that the end of the fade-in is marked by a considerable change in the rate of increase in σ_n. This suggests that the end of a fade-in can be detected by locating the frame pair where this large change occurs (a similar comparison can be used to detect the start of the fade-out by analysing the rate of increase backwards from the end of the fade-out). It is worth noting that the standard deviation continues to increase slightly after the end of the fade-in so it is not sufficient to detect the frame pair where the rate of change is less than or equal to zero as the end of the fade-in.

However, if a fade is to/from a relatively low contrast scene then using the standard deviation alone is not sufficient to detect the boundaries accurately. Figure 4.8 shows 8 frames from a shot during which the camera pans up to the sky and then ends with a fade-out. It can be seen by the red line in Fig. 4.9 that there is little change in the standard deviation of the pixel intensities during the fade-out. The largest decrease in this time series corresponds to the section of the shot where the camera pans up to the low contrast sky. In fact, the largest indication of this fade transition is found in the time series of the mean of the pixel intensities as shown by the green dashed line in Fig. 4.9.

This approach to detect fade transitions also includes the detection of ‘dips to colour’. Although uncommon, it can occur that there is little change in the mean of the pixel intensities and that the

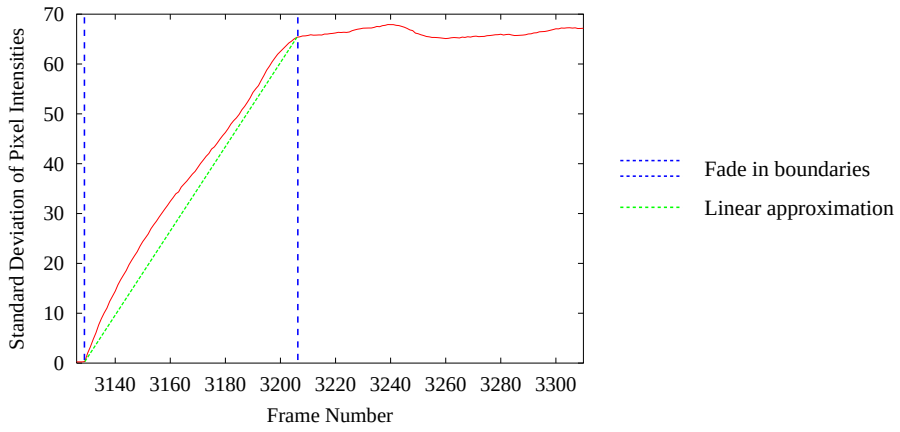


Figure 4.7: A fade is a scaling of the pixel intensities over time which can be observed in the standard deviation of the pixel intensities.

Figure 4.8: Frames illustrating a fade-out from a low contrast image.

largest indication is given by the standard deviation. As a result neither measure alone is sufficient to reliably detect the boundaries of a fade. Therefore, the linear combination of both measures is used as shown by the blue dashed line in Fig. 4.9. If we only consider a fade-in, σ_n will always be close to zero for the first frame irrespective of its colour. However, the mean could be anywhere in the range $[0, 255]$. Indeed, the mean could decrease during a fade-in. For this reason, the mean value of the first frame of the fade-in and each subsequent frame is transformed by

$$\mu'_n = |\mu_n - \mu_s| \qquad \text{for } n \geq s \tag{4.2}$$

where μ_n is the mean of frame f_n and frame f_s is the first frame of the fade-in. Detecting the end of the fade-in now becomes a problem of locating the frame where there is a considerable

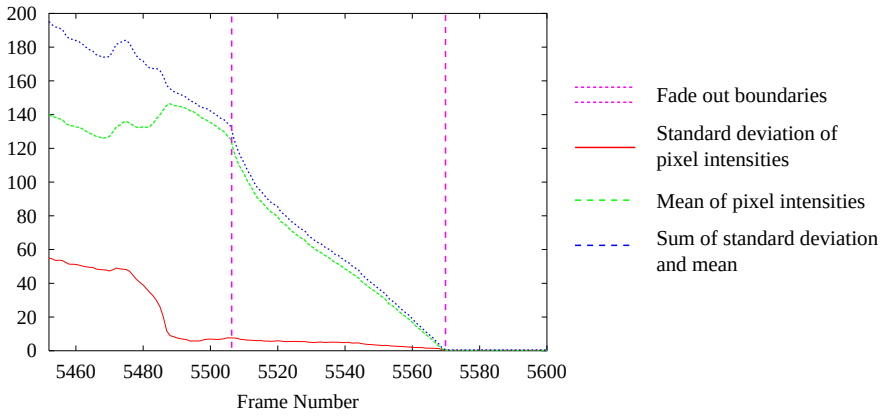


Figure 4.9: If a fade is to/from a low contrast scene there can be little change in the standard deviation. A larger change can be observed in the mean.

reduction in the rate of change of the time series representing the linear combination

$$l_n = \mu'_n + \sigma_n \tag{4.3}$$

The detection of this turning point is achieved by using the average rate of change of l_n between the starting frame and a current frame to predict the value of l_n for the next frame. If the actual value for the next frame is less than its predicted value for several consecutive frames, the rate of increase is deemed to have changed considerably and the end of the fade-in is declared.

More formally, for each frame f_n following the first frame f_s such that $n > s$, the average rate of change of l_n is computed where $\overline{l}_n = l_n/(s-n)$. Then, the value of l_n is predicted for the next frame f_{n+1} using $l'_{n+1} = l_n + \overline{l}_n$. If the actual value of l_{n+1} is greater than or equal to the predicted value i.e. $l_{n+1} \geq l'_{n+1}$ then the fade-in is deemed to still be continuing. On the other hand, if $l_{n+1} < l'_{n+1}$, the current frame f_n is marked as the potential end of the fade-in. If the actual value of l_{n+1} is less than its predicted value for several successive frames the fade-in is determined to have finished. If the number of successive frames N_{succ} is equal to 1 this would be equivalent to assuming the rate of increase must be linear and the end of the fade-in will frequently be detected too early. However, computing the average rate of change between the starting frame and each current frame and using $N_{succ} > 1$ allows for a non-linear increasing function. In fact, if N_{succ} is set too high, the average rate of change will eventually become close to zero and potentially l_{n+1} will be greater than or equal to its predicted value for many successive frames. In this case, the detected fade-in will be declared too long. We found $N_{succ} = 5$ resulted in the start/end of each transition to be determined accurately and was kept constant in our approach.

4.3 Dissolve Detection

The final extension to the segmentation algorithm is to incorporate the detection of dissolve transitions. This stage of the algorithm is only applied to shorter sections of the video sequence between any detected shot cuts and fade transitions. For this reason, this approach assumes that the only transitions that may exist in the shorter sequence are dissolves. During a dissolve the contents of the current shot gradually disappear while the contents of the next shot gradually become apparent. If a sub-sequence does not contain any camera or object motion or any dissolve transitions then the visual content of the first frame should be similar to that in the last. In contrast, in the presence of a dissolve a similar comparison should indicate the visual content has significantly changed. Comparing the content of frames that are far apart is the main idea behind the detection of dissolve transitions. However, a sequence may contain camera and object motion which result in a similar change in the visual content. To compensate for such motions, the first frame of a shot is subdivided into a number of blocks and each block is tracked through the video sequence. Then, for selected blocks, the original content of each block is compared with its content in subsequent frames. If a block becomes occluded or leaves the shot its content is no longer compared. Therefore, any significant change should only be as a result of a dissolve transition.

The comparison between the contents of a block in distant frames is achieved using the same features and metrics used in the current work to detect shot cuts. However, the shot cut detection algorithm uses a variable block size to represent the visual content as a fixed block size is not appropriate for all video data. Tracking variable sized blocks through a sequence is a complex problem in its own right, and we therefore opt for using a fixed size blocks in the detection of dissolves. Although this means a slight sacrifice in generality, we are not interested in the tracking of actual objects, suggesting that a fixed block size will be adequate for our needs. This leaves the problem of choosing the most appropriate block size to use to track blocks through the video sequence and compare the visual content of frames that are far apart. If a larger block size is chosen, a block may contain multiple motions, i.e. it must be subdivided over time to track its content accurately. If a smaller block size is chosen, a block is less likely to contain multiple motions. However, it may be impossible to track its content accurately in the presence of large motions and it may not contain sufficient high-frequency phenomena to track its content reliably. Moreover, if the majority of blocks contain little or no high-pass information then the similarity metric based on the comparison of edge features will be poor between consecutive frames and frames that are further apart leading to the false detection of dissolves. Nevertheless, the problems encountered using a smaller block size are easier to overcome in this approach than using a block size too large. Therefore, a smaller block size of 32×32 was chosen.

To overcome the problem of tracking each block accurately in the presence of motions larger than

$[-15, 16]$, motion estimates resulting from the detection of shot cuts can be used as "parent" estimates to be inherited by each block. Hierarchical motion estimation was employed in the detection of shot cuts to find the optimal block size to represent the visual content of each frame pair. The chosen block size is such that it provides the best overall correlation between the two frames. It can, therefore, be assumed that the optimal block size also estimates the motion sufficiently accurately between them. These motion estimates corresponding to the optimal block size can then be used as initial guesses for estimating the motion of the fixed sized blocks to be tracked between the two frames.

Given two frames f_{n-1} and f_n, the blocks corresponding to the optimal block size y_i are in a regular spatial grid in f_{n-1}. However, the fixed sized blocks x_j being tracked through the sequence can be at any position. We define that each block x_j inherits a motion estimate from the block y_i that it overlaps with the most, i.e. the parent of x_j is defined to be the block y_i that contains the centre point of x_j. Given a block x_j with its centre at position $\boldsymbol{p}$ in frame f_{n-1}, it is correlated with the block at the position $\boldsymbol{p} + \boldsymbol{d}$ in f_n where $\boldsymbol{d}$ is the displacement vector estimated for y_i containing the point $\boldsymbol{p}$.

The second problem that may be encountered is if a block does not contain sufficient grey-level variation. If the chosen block size is inappropriate, the similarity measure based on the comparison of edge features will be poor, indicating the visual content has changed even though a dissolve has not occurred. The inclusion of the comparison of colour distributions helps to overcome this problem. However, if a block is not being tracked reliably or there is a change in the global illumination, its colour distribution may change. If this occurs for the majority of blocks then the difference between the visual content of two distant frames will appear to be large leading to a false detection. Based on this observation, only blocks that resulted in a high similarity metric E_n between the first frame pair are chosen to compare their original content with the content they contain in subsequent frames. These selected blocks are referred to as regions of interest (ROI). To select the ROI the blocks are grouped into 2 clusters based in their corresponding similarity metric E_n. This is achieved by using the k-means clustering algorithm with $K = 2$. The blocks belonging to the cluster with the highest mean value are chosen to be the ROI. Figure 4.10 illustrates the first frame from four different shots. The blocks chosen to be a ROI are shown in white and the mean value of E_n for each cluster is also presented. It can be seen that the blocks that were not chosen correspond to areas in the image with little or no high-pass phenomena or contain multiple motions.

Motion estimation between each frame pair is now used to track all of the blocks over time in the video sequence. In addition, each ROI is compared with the corresponding block at its new location in frame $n+1$, $n+2$, etc. as shown in Fig. 4.11(a-c), using the same features and metrics

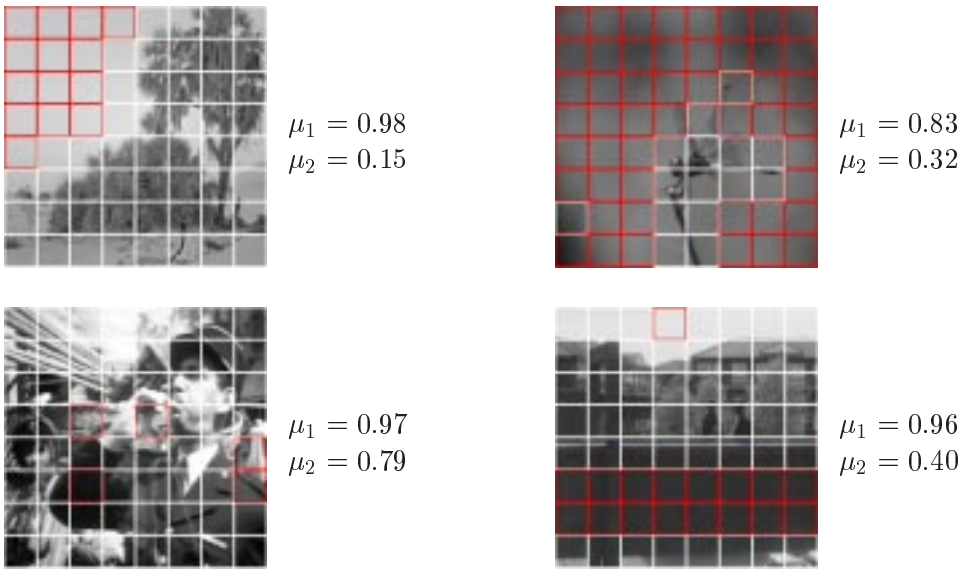


Figure 4.10: Blocks are selected to regions of interest (ROI) in the first frame of each shot.

used to detect shot cuts. This results in two measures, $ROIE_n$ corresponding to the similarity of edge features for the ROI and $ROIC_n$ corresponding to the difference between the colour distributions of the ROI. During a shot, $ROIE_n$ should remain high and $ROIC_n$ should be small indicating that the contents of each ROI has not changed significantly. During a dissolve, the content of each ROI will gradually change. This means that $ROIE_n$ will decrease and $ROIC_n$ will increase until they reach values comparable to those obtained for a shot cut.

Whilst tracking, object or camera motion or inaccurate motion estimation may cause blocks to become overlapped or move out of the scene as shown in Fig. 4.11(c). Once this occurs the block tracking is no longer reliable because block matching cannot easily resolve occlusion. This may cause the content of a block to change. Blocks that are overlapping or have started to move outside the image are removed as shown in Fig. 4.11(d). If any of the removed blocks were a ROI they are also removed from the current set of ROI. A block is removed if the proportion of its area overlapping with other blocks is greater than T_A. A block is also removed if it is more than T_A outside of the frame. In this approach, T_A was kept constant and equal to 0.5. The removal of blocks will leave areas of the image uncovered, the contents of which still need to be tracked. For this reason, new blocks are re-introduced in the uncovered areas. This is achieved by comparing the current positions of the remaining blocks to a regular spatial grid. Any blocks in this regular grid that are not covered more than T_A by the current set of blocks are added as shown by the white blocks in Fig. 4.11(d). This helps maintain a regular grid of blocks to track the video content.

Initially, the set of ROI are selected from the first frame pair of the sub-sequence. As blocks are gradually removed, the set will become empty. As well as adding blocks from the regular grid to

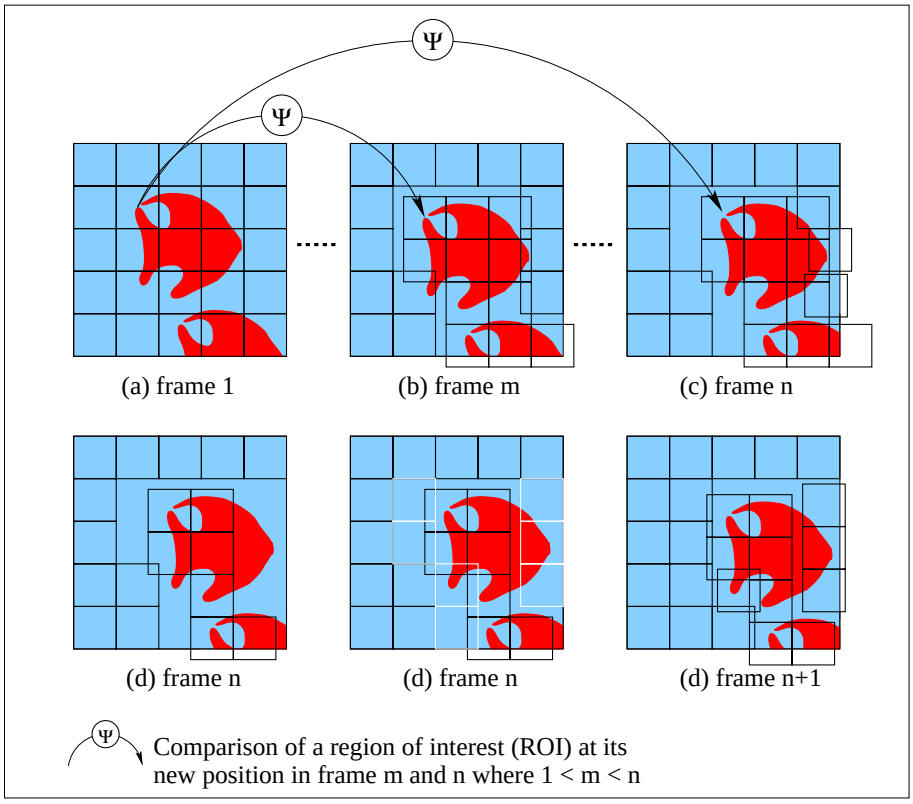
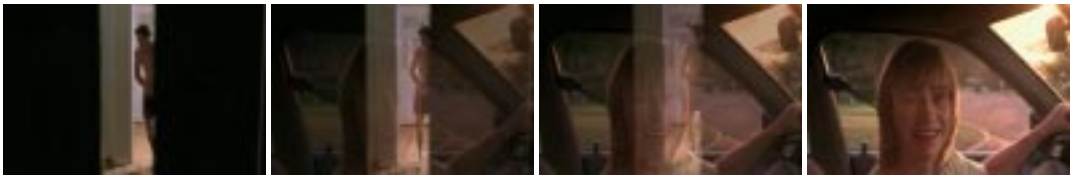


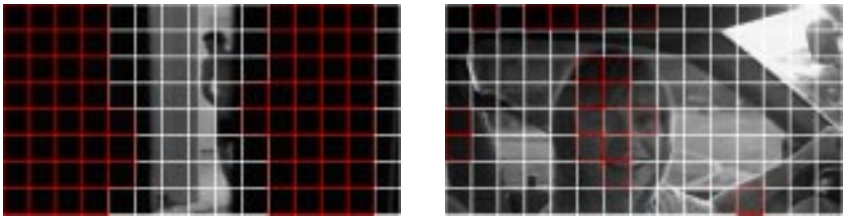
Figure 4.11: (a-c) Blocks are tracked over time and the content of each ROI is compared. Blocks may become overlapped, (d) overlapped blocks removed, (e) blocks added in uncovered area, (f) blocks continue to be tracked.

track the video content, blocks must be added to conserve the current set of ROI. Once the addition of blocks in frame f_n is complete it is possible to continue to track the blocks into the next frame f_{n+1} (Fig. 4.11(f)). A similarity metric E_{n+1} is computed for all the blocks between f_n and f_{n+1} and the blocks are grouped into 2 clusters in the same manner as between the first frame pair. If a block has just been added in f_n and is in the cluster with the highest mean value it is added to the current set of ROI. The addition and the removal of blocks allows the set of ROI to be updated for changes due to camera and object motion.

Comparing nothing but the content of the ROI obviously restricts the amount of image content that is being compared between frames that are far apart. In some cases, this can result in only a small amount of the frame content being compared to detect a dissolve transition. For example, Fig. 4.12(a) shows 4 frames during a dissolve. If the ROI are chosen from the first frame as shown by white in Fig. 4.12(b), then less than half of the frame content is actually being compared in subsequent frames. Moreover, it can be seen that by the end of the dissolve the content of all the blocks has changed significantly not only the content of the ROI. Indeed, the content change in the rest of the blocks is also a good indication of the presence of a dissolve and should not be



(a) Four frames representing a dissolve



(b) Comparing nothing but the content of the ROI limits the amount of change in the image content that will be detected

Figure 4.12: The content of blocks that are not ROI also changes significantly and should not be overlooked.

overlooked. It can be seen in Fig. 4.12(b) that if the dissolve had been from the second shot to the first, nearly all of the blocks would have been chosen as ROI.

Blocks with little high-pass information or multiple motions are not included in the set of ROI because their similarity metric based on the comparison of edge features will be poor and negatively influence the value of $ROIE_n$. If there is not a dissolve in the sub-sequence, then these blocks should not be included in the set of ROI. However, if there is a dissolve such that their content significantly changes it would be preferable to also compare their content between distant frames. A good indication that the content of such blocks has changed is if their content starts to result in a high value for E_n between consecutive frame pairs, i.e. their content now includes high-pass phenomena. Therefore, between each frame pair all of the blocks are grouped into 2 clusters. If any block is in the cluster with the highest mean value for E_n that is not already a ROI it is added to the current set so its original content can be compared with its current content.

A dissolve is detected using the same thresholds used in the shot cut detection. If $ROIE_n < T_E$ and $ROIC_n > T_C$ a dissolve is deemed to be present. Usually, this occurs towards the end of the dissolve. However, the actual boundaries of the dissolve must be determined. If there exists motion during the dissolve this may be sufficient to cause the set of ROI to be updated before the end of the dissolve. The difference between the content of two frames will, therefore, appear less different before the actual end of the dissolve. Consequently, if $ROIE_n < T_E$ and $ROIC_n > T_C$ at frame f_n, the blocks are no longer tracked through the sequence and the set of ROI is not updated. Each ROI is simply compared in the same location in the following frames. The difference between the visual content will continue to increase until the content becomes maximally different at the end

of the dissolve. Once the end of the dissolve is detected, the first frame of the next shot is divided into a regular grid of blocks and a new set of ROI are selected to be tracked for the detection of the next dissolve. This continues until the end of the sub-sequence is reached.

The start of the dissolve is detected by locating the frame pair where $ROIE_n$ started to decrease and $ROIC_n$ started to increase i.e. the frame pair where the visual content started to change. The end is marked by the frame where $ROIE_n$ and $ROIC_n$ reached their minimum and maximum respectively. A dissolve can be of any length lasting anywhere from approximately $1/2$ second ($10-12$ frames) to extremely long lengths of over a minute. Although they can be as short as only $1-10$ frames known as a soft-cut [50]. For this reason, the detection of the boundaries does not make any assumptions about the typical length of a dissolve as in some previous approaches [84, 37]. Fig. 4.13 illustrates $ROIE_n$ and $ROIC_n$ during a dissolve transition. The blue vertical line shows the frame where $ROIE_n < T_E$ and $ROIC_n > T_C$ and it can be seen that this happens towards the end of the dissolve. Once this occurs, the ROI are compared with the block in the same position in the subsequent frames until the end of the shot is reached and then the boundaries of the dissolve are detected by analysing $ROIE_n$ and $ROIC_n$. In practise, after $ROIE_n < T_E$ and $ROIC_n > T_C$ each block is only correlated with N subsequent frames where is kept constant and equal to 50 or until the end of the shorter sequence. The boundaries of the dissolve are then detected by analysing the combination of $ROIE_n$ and $ROIC_n$ defined by

$$M_n = 2(1 - ROIE_n) + ROIC_n \tag{4.4}$$

If $ROIE_n < T_E$ and $ROIC_n > T_C$ at frame f_n, the average rate of change of M_n from f_n to the end of the sequence is computed. The point with the largest difference between its actual value for M_n and a predicted value for M_n using the average rate of change is selected as the end of the dissolve as shown in Fig. 4.14. A similar comparison is used between the end of the last transition and f_n.

Finally, Fig. 4.15 illustrates $ROIE_n$ and $ROIC_n$ during two consecutive dissolves in a video sequence. It can be seen that $ROIE_n$ decreases and $ROIC_n$ increases during a dissolve and they reach their minimum and maximum respectively at the end. The two dissolves are, therefore, easily detected.

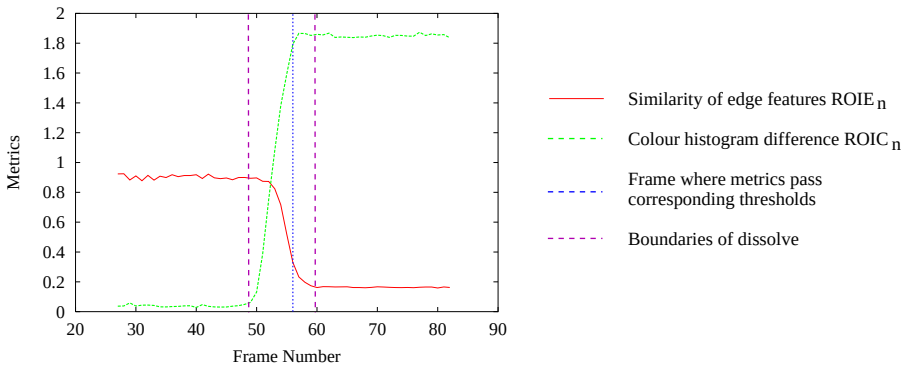


Figure 4.13: During a dissolve $ROIE_n$ decreases and $ROIC_n$ increases.

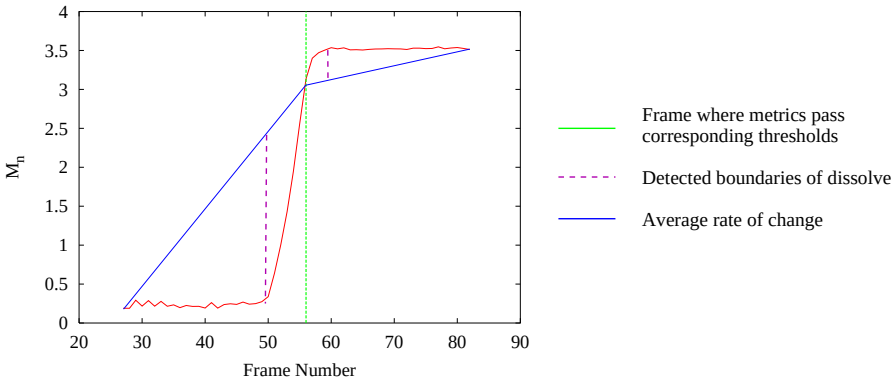


Figure 4.14: A combination of $ROIC_n$ and $ROIE_n$ can be used to find the boundaries of the dissolve.

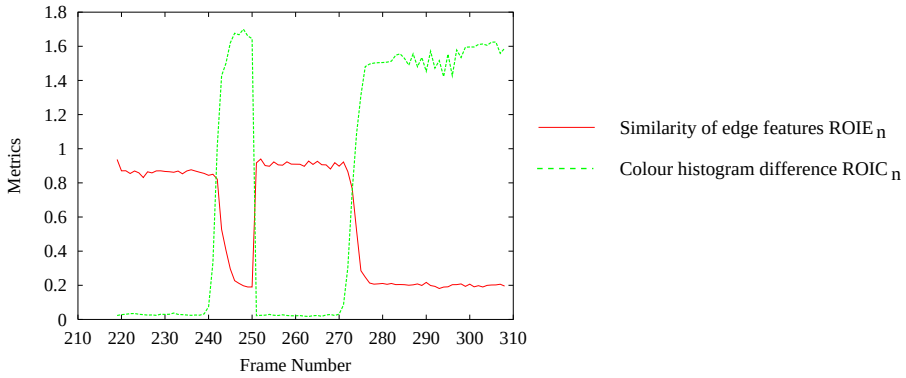


Figure 4.15: The metrics can be used to identify consecutive dissolves.

4.4 Comparative Results

In this section, the performance of the proposed gradual transition detection method is evaluated and compared with one other method, the twin comparison technique. The histogram-based method was chosen since it is a well established technique and has been shown to perform well in detecting edit effects [61]. Other methods, such as the feature-based methods were not included in the comparison as their performance has been shown to be disappointing [70]. Moreover, techniques such as the edge change ratio and the edge based contrast require several thresholds, their performance can be sensitive to the values chosen and require 'hidden' parameters not specified in the literature [8].

The histogram-based method we used is similar to the method with the best performance in the comparative investigation by Lupatini et al. [61]. This approach uses the χ^2 statistic to define the difference between two global colour histograms which is compared against two thresholds, T_H and T_L. Whenever the histogram difference between two consecutive frames is greater than T_H, a shot cut is detected. If the difference lies between the two thresholds the frame is marked as the potential start of a gradual transition. Successive frames are then compared with the first frame of the transition and if the difference exceeds T_H, a gradual transition is detected. The end of the gradual transition is marked once the difference between frame pairs drops below T_L for two frame pairs.

Test Data

To test these methods we used 10 different video sequences totalling 33248 frames. All sequences contained a combination of shot cuts, fades and dissolves and were chosen to include different genres. The number of frames, transitions and the category of each sequence is summarised in Table 4.1. The locations of these transitions were hand labelled to obtain a ground truth to evaluate the performance of each algorithm.

The real boundaries of a gradual transition can be difficult to ascertain when hand labelling the transitions. This means that it is difficult to evaluate the accuracy of the detected boundaries. However, if a shot cut detection algorithm is to be used in the first step of a video indexing process it is important that the transition boundaries are detected precisely. To represent properly the content of a shot, key frames should not be selected partway through a gradual transition. Therefore, for these experiments an algorithm must detect and classify correctly the type of the transition to the best of its ability for the transition to be considered a correct detection. In other words, if an algorithm can distinguish between cuts and gradual transitions it must detect a cut as a cut

Sequence	Frames	Cuts	Fade-ins	Fade-outs	Dissolves	Genre
1	478	7	5	4	3	Cartoon
2	422	8	3	4	2	Film
3	462	4	0	1	2	Film
4	3326	12	6	5	8	Documentary
5	3843	20	7	9	8	Comedy series
6	9037	80	8	7	19	Documentary
7	4376	29	14	9	23	Documentary
8	5736	11	2	4	7	Drama series
9	2483	21	6	10	11	Sit-com
10	3085	51	0	0	11	Sport
ALL	33248	243	51	53	94	Various

Table 4.1: Test data used to compare gradual transition detection algorithms.

and fades and dissolves as gradual transitions. In these experiments, if an edit effect was detected but classified incorrectly it was considered a false detection and the actual transition was labelled undetected. For example, an algorithm that detected a long dissolve by marking two frames as a shot cut would not segment the video properly. The proposed algorithm must classify all of the transitions correctly into cuts, fade-ins, fade-outs and dissolves. Although there exists some error on the labelled boundaries of a gradual transition, a gradual transition is only considered correct if it overlaps with an actual gradual transition and the detected boundaries are within N frames of the real boundaries. In our experiments, $N = 50$.

This work aims to detect all types of shot transitions using a single technique and the same parameter set. It is proposed that the gradual transitions can be detected using the same thresholds used in the detection of shot cuts. For this reason, the threshold values for T_E and T_C used to maximise the recall and precision for the shot cut detection algorithm were used in these experiments.

Initially, the threshold value that maximised the harmonic mean for the global-histogram comparison shot cut detection method was to be used for T_H and the value for T_L would be varied to find the most appropriate threshold. However, it soon became apparent that $T_H = 0.12$ was not a suitable value to be used in the detection of gradual transitions. During many gradual transitions (particularly short effects) the difference between frame pairs was greater than T_H. In this case, all of the shot cuts would be considered false and the gradual transition undetected. As a result, T_H needed to be increased to prevent the misclassification of gradual transitions. Various possible values for T_H and T_L were tried between 0 and 2, and the results presented here for the twin comparison technique are always using the threshold set that maximises both recall and precision i.e. the threshold pair that maximised the harmonic mean.

The performance of the motion-based approach (MB) compared with that of the twin-comparison

Algorithm	Threshold values		C	M	F	R	P
MB	$T_E = 0.42$	$T_C = 0.22$	241	2	0	0.99	1.00
TC	$T_L = 0.14$	$T_H = 0.29$	190	53	92	0.78	0.67

Table 4.2: Comparative performance for shot cut detection only.

Algorithm	Threshold values		C	M	F	R	P
MB	$T_E = 0.42$	$T_C = 0.22$	195	3	5	0.98	0.98
TC	$T_L = 0.01$	$T_H = 0.19$	123	75	36	0.62	0.77

Table 4.3: Comparative performance for gradual transition detection only.

Algorithm	Threshold values		C	M	F	R	P
MB	$T_E = 0.42$	$T_C = 0.22$	436	5	5	0.99	0.99
TC	$T_L = 0.01$	$T_H = 0.27$	309	132	130	0.70	0.70

Table 4.4: Comparative performance for the detection of all the transitions.

technique (TC) for shot cut detection only can be seen in Table 4.2. A comparison of the performance of the algorithms for the detection of gradual transitions can be seen in Table 4.3. It should be noted that while the MB method classifies gradual transitions into fade-ins, fade-outs, and dissolves, TC does not make a distinction at all. From these tables it is simple to notice the better performance of our method compared with the TC technique. It should also be mentioned that the missed and false gradual transitions were all dissolves and the detection of fade transitions actually had recall and precision equal to 1.0.

Finally, Table 4.4 summarises the performance of both algorithms for the detection of all the transitions. Although the TC method obtained results similar to those reported in the literature, the performance of the MB method was considerably better. Hanjalic noted that robust algorithms for detecting various types of transitions have not been found yet [41]. The author defined an algorithm to be “robust” if it gives excellent performance for all types of shot transitions (cuts and gradual) and provides constant quality of the detection performance for any arbitrary sequence. In other words, there is no need for manual fine-tuning of parameters for different sequences. Having shown the performance obtained on these sequences using the same parameter values used in the detection of shot cuts we feel that the proposed algorithm is accurate and robust.

4.5 Summary

We have presented a novel, unified approach that classifies shot boundaries with a better resolution into cuts, fade-ins, fade-outs and dissolves. The recall and precision values show either a signifi-

cant improvement on other approaches or are easily comparable given that all shot transitions are separately resolved.

Dissolves were detected by tracking regions over time, gradually removing blocks which no longer can be relied upon, and adding blocks where gaps appear, or where new objects may enter the scene. Blocks from previous frames still present in the current frame, and the differences in each such block's contents is monitored. If the majority of the difference metrics for such blocks exceed a threshold, a dissolve is detected.

Fade transitions were detected by firstly locating constant images, and then using the shot cut detection algorithm to determine if this frame was either a shot cut to a blank image, or the potential start or end point of a fade. If the blank frame was determined to be the boundary of a fade, the opposing boundary was detected by examining the pixel mean and variance time series, looking for a significant increase in the rate of change. The chapter concluded with the results of experiments confirming the viability of the approach.

Chapter 5

Video Indexing

Extracting a small number of key frames that can summarise the content of a video is important for efficient browsing and retrieval in multimedia databases. In this chapter a novel approach is presented to select multiple key frames within an isolated video shot where there is camera motion causing significant content change. This is achieved by determining the dominant motion between frame pairs and computing the similarity between their contents. A directed weighted graph is formed for each shot and the shortest path through the graph is used to designate key frames. The extracted set of key frames portray both the video content and camera motions, both of which are useful features for video indexing and retrieval. In addition, the perceived camera motions contained within each shot are also annotated to provide a supplementary video index.

5.1 Video Summarisation

A set of key frames that summarise the video content can be used in conjunction with existing textual annotations to augment the indexing process, to enable non-sequential browsing or to create a visual index into video that has not been previously textually annotated—as the saying goes “a picture is worth a thousand words”. Selected key frames arranged as a storyboard can be used to quickly peruse the contents of a selected program. However, more importantly, if key frames are properly selected, many higher level content searches can be performed on the key frames rather than the complete video, thus reducing the computational requirements involved. The difficulty in composing a visual summary is determining which frames best represent the video contents and correctly portray the storyline.

Figure 5.1: Equally distributed frames representing a single shot - a single key frame would not represent all the shot's content.

To allow efficient indexing, a summary for a digital video library must represent the entire video content with as little redundancy as possible [59, 45]. Each key frame should represent a video segment which exhibits consistency in content. A common approach to this problem is to segment temporally a sequence into shots and then select a single representative key frame for each shot [82, 26]. The resulting ordered set of key frames is often referred to as a filmstrip [18]. Although this method illustrates the practical use of segmenting a video into its constituent shots, one key frame may not always be sufficient to represent each shot. A shot can contain events such as camera or object motions that may drastically change its content. Fig. 5.1 shows 8 equally distributed frames from a single shot. At the beginning of the shot there is little activity, then two women appear from a building entrance and the camera pans right to track them walking down the street. This would be difficult to represent using a single key frame, which leads to the idea of motion-based key frame detection.

Motion analysis has been used previously to extract key frames. Wolf proposed a method that selects key frames at local minima of motion activity within a shot based on the assumption significant pauses are used to emphasise video content [90]. The author conjectures that in many shots key frames are identified by stillness—either the camera stops on a new position or the characters hold gestures to emphasise their importance [90]. This method selects key frames which correspond to pauses in the video sequence between motion activity within a shot. Consequently, the content change between key frames can be expected to be the result of camera or object motion. Although this method aims to select key frames corresponding to stillness, it illustrates how motion analysis can be used to select more than one key frame to represent the change in a shot's content. However, the selected key frames may still fail to represent adequately the entire video content. For example, during a pan it may be the content which the camera is panning over is

important which would not be represented by a key frame using this method. In addition, there is no measure of the significance of the motion between key frames. Hence, the content change between key frames may be small. The two main contributions of Wolf's work were the novel use of motion analysis to identify content change, and the assumption that the number of key frames per shot need not be restricted to one, selecting instead the number appropriate to the composition of the shot.

Several methods have been proposed for key frame selection that require no content-based analysis within a shot. The most straightforward technique is to select a preset number, n of key frames for every shot. Although this approach is simple, if $n = 1$ it does not guarantee that the chosen frame is a faithful representative of the shot and if $n > 1$, selected key frames may be redundant. An alternative method is to progressively compare each frame in the shot against the last key frame selected using difference metrics similar to those used in shot cut detection. A frame is selected as an additional key frame if the difference between itself and the last key frame is greater than some threshold [100]. Using such approaches, the reason why the difference metric exceeded the threshold can not be determined, and the visual summary may include redundant, or an insufficient number of frames. Again, this suggests that content-based analysis must be used to determine which frames best represent a shot's content.

Based on the above observations we conclude that an algorithm used to extract key frames to summarise and index data in a digital video library must represent all the video content whilst focusing on minimising the similarity between key frames. At least one key frame must be used to represent each shot in a video sequence. More than one key frame may be required if there has been significant content change during the shot. The problem is determining when there has been sufficient content change to warrant more than one key frame. The content of a shot can be changed by camera or object motions. Detection of generic objects and understanding semantic changes in the video content caused by their motion is still beyond the capability of current algorithms. Methods have been proposed to summarise the semantic content of the video based on object motion and extracting key frames that represent 'events of interest', e.g. appearance/disappearance, entrance/exit and the deposit/removal of objects. However, the application of such techniques are limited to constrained problems such as the indexing of surveillance videos where usually the camera is static [76, 20]. During a shot there can be 4 different scenarios involving camera and object motion which are illustrated in Table 5.1. If a shot contains no camera or object motion, a single key frame would be sufficient to summarise the shot. If a shot contains camera motion but no object motion, then key frames correctly selected to represent the camera motion will also illustrate any objects within the shot. Additionally, we conjecture that in the presence of camera and object motion, it is the camera motion that causes the most significant content change. That is to say, if key frames are selected to describe the camera motion adequately then any content change

Camera Motion	Object Motion
No	No
No	Yes
Yes	No
Yes	Yes

Table 5.1: Different combinations of camera and object motion that can occur during a shot.

caused by object motion will also be depicted by the visual summary. For example, in Fig. 5.1, if frames 1, 6, 7 and 8 were selected to illustrate the pan right then the two women walking down the street would also be shown. We propose that if camera motion-based analysis is used to select key frames for each shot, then 3 out of the 4 situations will be represented satisfactorily. For this reason, we focus on the detection of content change caused by camera motion only and propose that this will provide a sufficient summarisation in the majority of cases. Once a shot has been identified to contain no camera motion, further analysis can be done to identify object motions that may have caused the content to change.

We propose an algorithm based on the assumption that a video sequence has already been temporally segmented into individual shots. Estimates of the dominant motion between each frame pair are used to decide when there has been sufficient camera motion to require another key frame. If there has been no motion between two frames we assume their contents are equal. As the amount of camera motion between two frames increases, the overlap between their contents will decrease until the contents of the two frames become disparate. For this reason, if a shot contains

- *little or no camera motion,* only a single key frame is required.
- *significant camera motion,* then more than one key frame must be extracted.

Thus, the number of key frames selected will vary for each shot depending on the amount of camera motion contained within it. For example, Fig. 5.2(a) shows 4 equally distributed frames from a 49 frame shot containing no camera motion. In this example, one key frame would be sufficient to represent its content. In contrast, Fig. 5.2(b) shows 4 equally distributed frames from a 214 frame shot during which the camera tilts up. Representing all the content in this shot would require more than one key frame. In addition, to selecting a sufficient number of key frames, the similarity between them must be minimised.

If a shot contains a small amount of camera motion, the first and last frame of the shot may be sufficient to represent the entire content. However, in Fig. 5.2(b) the contents of the first and last frame do not overlap and it would be difficult to determine what had occurred during the shot. At least one intermediate key frame would be needed to illustrate the tilt up and summarise all of the

(a) No camera motion - a single key frame would be sufficient to represent this shot.

(b) A tilt up - more than one key frame is required to represent this shot.

Figure 5.2: The number of key frames required to adequately represent a shot's content will vary according to the composition of the camera motion within it.

visual content. If there is no overlap between the first and last frame we must choose intermediate frames that provide a path through the shot that connects the first and last frame. Indeed, this problem lends itself well to a graph based representation and the problem of finding a path from one vertex to another. Each shot can be represented by a graph where the vertices correspond to frames in the shot and edge weights are a measure of similarity between the corresponding frames. If there is no similarity between two frames such as the first and last frame in Fig. 5.2(b) then the path can not pass directly between them and another path must be sought. Obviously, one possible path could include every frame, however, we also want to minimise the similarity between key frames that represent the path. Given a directed weighted graph, the shortest path between two vertices is the path of minimum total weight. We want to minimise the similarity between key frames. Therefore, we formulate the selection of the optimal number of key frames as a shortest path problem. The method can be described as follows.

1. **Estimate the dominant motion between each consecutive frame pair.**

   Given an isolated shot in a video sequence, motion estimates obtained from the block-based motion compensation employed in the edit effect detection algorithm are used to estimate the dominant motion between each consecutive frame pair.

2. **Compute a similarity metric between all combinations of frame pairs in the shot.**

   For each frame in the shot, the dominant motion estimates are accumulated between it and every successive frame in the shot. A similarity metric between each frame pair is then computed based on the total amount of motion between them. If there has been no motion, the two frames will contain the same content resulting in a high similarity metric. As the amount of motion increases the similarity metric will decrease.

3. **Form a weighted directed graph for each shot.**

A weighted directed graph is formed where the vertices represent the frames in the shot and the weight on each edge is equivalent to the similarity metric between each frame pair.

4. **Find the shortest path through the graph to select key frames.**

   To minimise the similarity between key frames, the frames corresponding to vertices forming the shortest path through the graph are used as representative key frames for each shot.

5.2 Estimating the Dominant Motion

The first step in this method is to estimate the dominant motion between each consecutive frame pair. Given the 2D motion vector field obtained from the block-based motion compensation employed in the edit effect detection algorithm, a robust estimator is used to estimate the parameters of a simple motion model. Assuming the dominant motion between a frame pair is caused by camera motion, these estimates can then be used to identify shots containing significant camera motion that may require more than one key frame to represent their content.

The video segmentation algorithm employs hierarchical motion estimation to find the optimal block size to represent the visual content of each frame pair. The chosen block size is such that it provides the best overall correlation between the given frame pair. If there is little high frequency information and/or little motion between two frames a larger block size tended to be chosen. If there exists multiple motions or motion that violates the 2D translational model between two frames a smaller block size was chosen. For this reason, we can assume the optimal block size to represent the visual contents also estimates the motion sufficiently accurately to estimate the dominant motion between a consecutive frame pair.

Common camera operations used in video production can be grouped into two broad classes: (i) tripod motion and (ii) free motion. If a camera is fixed to a tripod it can only exhibit three types of motion as shown in red in Fig. 5.3:

1. **Pan** - a rotational movement of the camera about the vertical axis.
2. **Tilt** - a rotational movement of the camera about the horizontal axis.
3. **Zoom** - convergent or divergent.

If there is free motion of the camera it can exhibit three additional motions shown in black in Fig. 5.3:

1. **Boom** - upward/downward motion of the camera along the vertical axis.

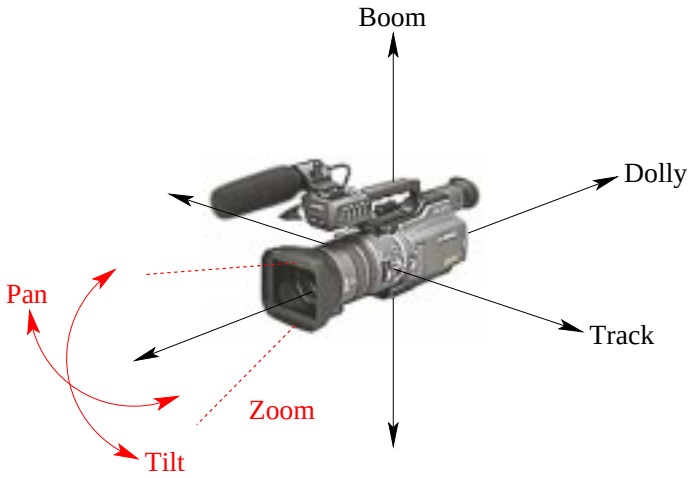


Figure 5.3: Common camera operations used in video production.

2. **Track** - right/left motion of the camera along the horizontal axis.
3. **Dolly** - forward/backward motion of the camera along the optical axis.

The effect of a pan on the change of contents in a shot and the perceived image motion is very similar to that of a track. For example, assuming a static scene, if a camera pans or tracks right, the background and objects appear to move to the left and gradually leave the shot while new background and objects may appear on the right. Such similarities can also be drawn between "tilt and boom" and "zoom and dolly". For this reason, we only consider pan, tilt and zoom and use a simple motion model which only represents the scale and translation in x and y between two frames. The point $\boldsymbol{p}$ in the frame f_{n-1} is transformed to the point $\boldsymbol{p}'$ in frame f_n, with respect to a reference point $\boldsymbol{r}$ according to

$$\boldsymbol{p}' = s_n(\boldsymbol{p} - \boldsymbol{r}) + \boldsymbol{d}_n \tag{5.1}$$

where s_n corresponds to the scale and $\boldsymbol{d}_n$ is the translation vector between the frames f_{n-1} and f_n. In practice, the frame centre is taken as the reference point. The model parameters are estimated using the robust estimator MSAC [83] which provides good estimates in the presence of outliers. Outliers could possibly be present in the data where the motion equation is invalidated or where points correspond to secondary motions.

Robust estimators such as MSAC [83] are commonly used when least squares estimators can not handle significant numbers of outliers. Using the MSAC algorithm we can select two points in frame f_{n-1} at random and estimate the model parameters s_n and $\boldsymbol{d}_n$ using motion vectors corresponding to each point. The algorithm then finds how many of the remaining data items fit

the model using the estimated parameters within a given tolerance, T. Each remaining point $\boldsymbol{p}$ in frame f_{n-1} is transformed to point $\boldsymbol{p}'$ in frame f_n according to s_n and $\boldsymbol{d}_n$, and to point $\boldsymbol{p}''$ using its corresponding motion vector. The error e between the model and the actual measurements for point $\boldsymbol{p}$ is then the Euclidean distance $e = |\boldsymbol{p}' - \boldsymbol{p}''|$. We allow the error e to be a maximum of one pixel in x and y, giving $T = \sqrt{2}$. MSAC then proceeds in the same manner seeking to minimise the cost function $C = \sum_{\boldsymbol{p}} \rho(e^2)$ where

$$\rho(e^2) = \begin{cases} e^2 & e^2 < T^2 \\ T^2 & e^2 \geq T^2. \end{cases} \tag{5.2}$$

In other words, the estimated parameters which result in the minimum error for all points $\boldsymbol{p}$ are chosen to model the camera motion between frames f_{n-1} and f_n.

5.3 Determining Shared Content

If a shot contains sufficient camera motion to warrant more than one key frame to represent all of its content, then at least the first and last frame is required and potentially several intermediate frames. Using the 3-component motion model defined in (5.1), we initially attempted to use the accumulated amount of each individual component to determine when to select each key frame. If the magnitude of either the scale, translation in x or translation in y exceeded its own threshold since the last key frame, a new key frame was selected. Thresholding each individual motion component, however, can allow the overall amount of motion to be more significant between one pair of key frames than another depending on whether there is only a single or multiple motions between key frames. All three motion components could potentially reach their threshold before a new key frame is selected. This situation is illustrated in Fig. 5.4, for a translation-only case. The motion illustrated by the red arrow, representing a combination of translation in x and y, is more significant than that shown by the black arrow representing a single motion.

The Euclidean distance could be used to determine the overall amount of translational motion. Then, if this distance exceeds a threshold, as shown by the blue ellipse in Fig. 5.4, a new key frame is selected. In this example, a frame would be chosen before the motion shown by the red arrow was completed. However, the Euclidean distance could not readily be combined with scale to determine the overall amount of camera motion.

Determining when to extract intermediate key frames when more than one motion component is non-zero, lead us to use the amount of shared content between two frames. Each successive frame

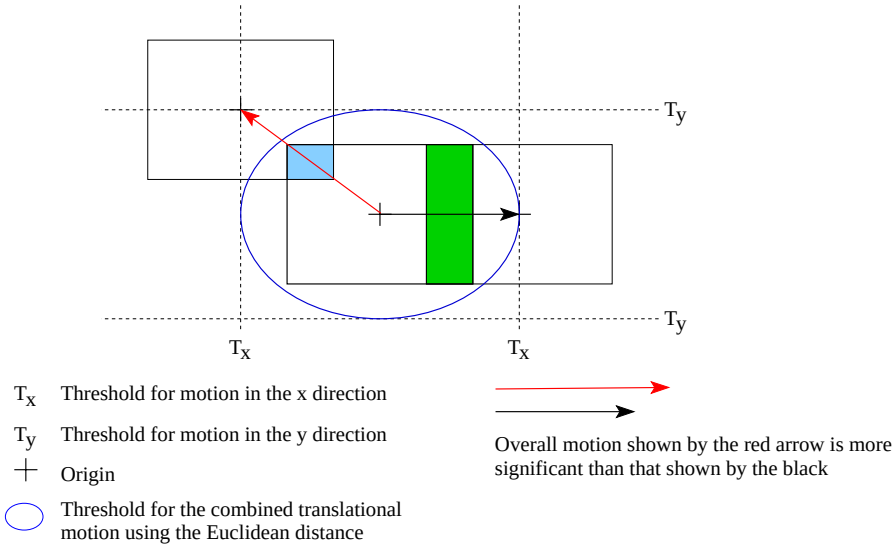


Figure 5.4: Thresholding each individual motion to extract key frames can potentially lead to more overall motion between one frame pair than another.

is compared with the last key frame and if the amount of content overlap between them is less than some threshold, the previous frame is selected as a new key frame. Thus, the amount of shared content between key frames will always be greater than or equal to a predefined amount. It can be seen in Fig. 5.4, that the overlap shown by the light blue shading is less than that shown by the green shading and again, a new key frame would be selected before the motion illustrated by the red arrow was accomplished. The measure of shared content can also be determined in the presence of scale to ascertain the effect of multiple motions.

Consider a synthetic motion consisting of a pan right followed by a tilt down shown by the blue track in Fig. 5.5(a). Fig. 5.5(b) illustrates the positions of the intermediate key frames selected by comparing the amount of shared content between each frame and the last key frame with a predefined threshold (in this case 20%). It shows two intermediate key frames are required to maintain the minimum amount of overlap between the key frames that will be used in the visual summary to adequately portray this camera motion. Fig. 5.5(b) also illustrates the first problem with the approach of selecting key frames using a threshold, which is that it can often result in a large overlap between the last two key frames. To overcome this, a new set of key frames was selected using a new threshold equal to the average amount of overlap in the current set of key frames. In this example, the new minimum amount of overlap to be maintained is 30%. This approach to redistribute frames more evenly could be used in the presence of a single motion. However, Fig. 5.5(c) shows the new set of key frames chosen for this example and it can be seen that in the presence of more than one motion it can introduce an additional key frame resulting in representational redundancy.

The second problem with this method and the key frames it selects in Fig. 5.5(b) is that it does not minimise the overlap between them. Fig. 5.5(d) shows that by selecting the third key frame slightly earlier the overall amount of overlap is decreased, reducing the representational redundancy contained in the visual summary. An algorithm is required that determines the optimal distribution of the key frames during the camera motion to minimise the shared content between them. This can be achieved by employing a shortest path algorithm to select representative key frames, described in the rest of this chapter. This algorithm:

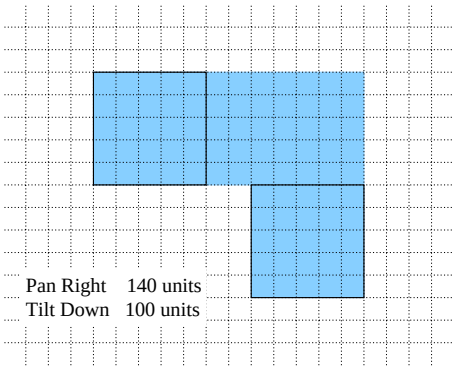
- minimises the number of key frames required to represent a shot's content;
- minimises the shared content between key frames and consequently minimises representational redundancy;
- can be used to evenly distribute frames throughout a shot; and
- selects the optimal number of key frames according to the composition of the camera motion in the shot.

5.4 Measuring Key Frame Similarity

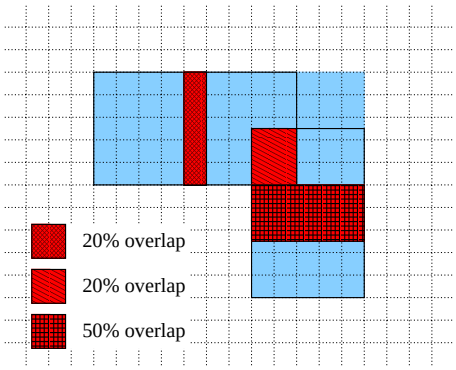
The first step is to represent a shot as a graph where vertices correspond to frames in the shot and edge weights are a measure of similarity between two frames. The frames corresponding to vertices forming the shortest path through the graph are then used as representative key frames for each shot.

As the amount of camera motion increases, the similarity between the contents of a frame pair will decrease. Therefore, we define the similarity metric $\psi(p, q)$ as the amount of overlap between the contents of any two frames f_p and f_q, potentially far apart, where $0 \leq \psi(p, q) \leq 1$. If there has been no camera motion between the frame pair, then $\psi(p, q) = 1$. As the amount of camera motion increases the amount of overlap will decrease until the contents of each frame are disparate and $\psi(p, q) = 0$.

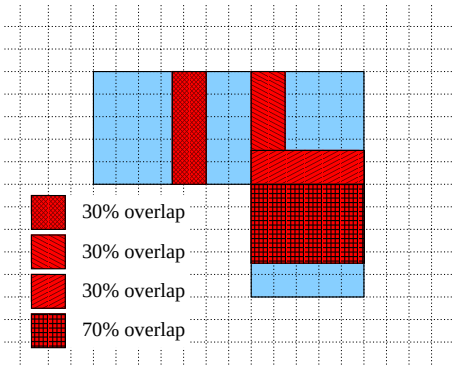
For each shot we have an estimate of the dominant motion parameters s_n and $\boldsymbol{d}_n$ for each consecutive frame pair f_{n-1} and f_n. Given any two frames f_p and f_q where $p < q$, we accumulate the



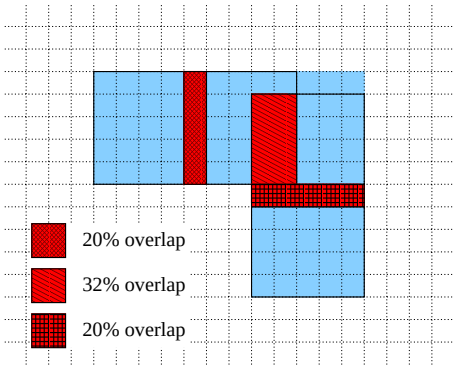
(a) A synthetic camera motion: pan right followed by a tilt down.



(b) Key frames selected once the overlap between two frames is less than T_{min}

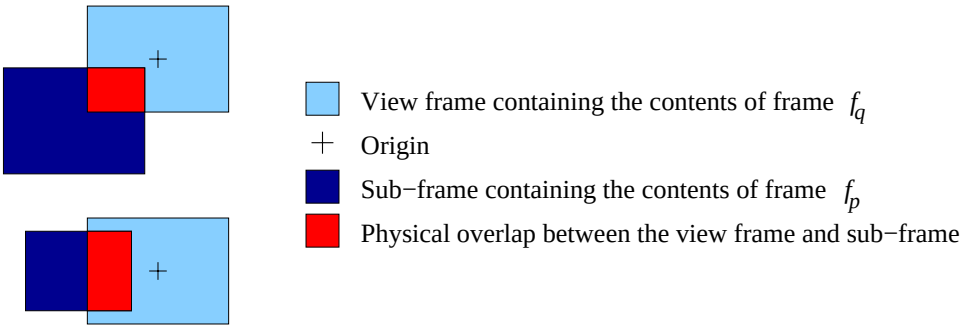


(c) Redistributing key frames whilst trying to maintain the average amount of overlap.

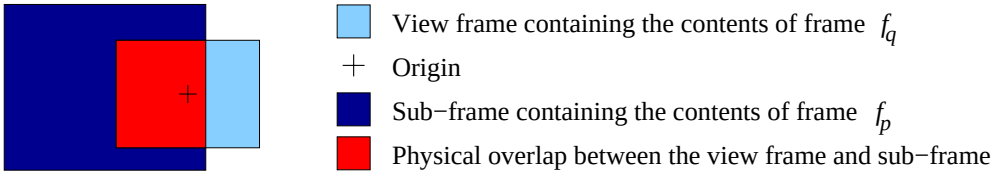


(d) Key frames can be selected to minimise the overlap between them.

Figure 5.5: Problems extracting key frames that are overcome by applying the shortest path algorithm.



(a) $\theta_{pq} \leq 1$. If there has been no scale or a scale down, compute what proportion of the view frame still contains the contents of frame f_p



(b) $\theta_{pq} > 1$. If there has been a scale up, compute what proportion of the contents of f_p are still present in the view frame.

Figure 5.6: Computing the similarity metric between frames f_p and f_q.

motion parameters between them to obtain θ_{pq} and $\mathbf{\Delta}_{pq}$ where

$$\theta_{pq} = \prod_{n=p+1}^{q} s_n \tag{5.3}$$

$$\mathbf{\Delta}_{pq} = \boldsymbol{d}_q + s_q \cdot \mathbf{\Delta}_{p(q-1)} \tag{5.4}$$

and $\mathbf{\Delta}_{p(q-1)} = \mathbf{0}$ when $p = q - 1$. Hence, θ_{pq} and $\mathbf{\Delta}_{pq}$ are the total amount of scale and translation respectively, between f_p and f_q. These accumulated motion parameters can then be used to compute the amount of overlap between the contents of the two frames.

There are two cases to be considered depending on the scale parameter θ_{pq}: (i) $\theta_{pq} \leq 1$ and (ii) $\theta_{pq} > 1$. In each case, assume there is a fixed view frame which has constant width w, height h and area $A = w \cdot h$ with its centre at the position $(0, 0)$. Given two frames f_p and f_q with the contents of f_p initially contained within the view frame, we define that by the frame f_q in the shot, the contents of f_q are now within the view frame and the contents of f_p have moved from the view frame according to the motion transformation defined by θ_{pq} and $\mathbf{\Delta}_{pq}$. We apply the accumulated scale and translation to the view frame to obtain a sub-frame z_{pq} with a new width ${}^{w}z_{pq} = w \cdot \theta_{pq}$, height ${}^{h}z_{pq} = h \cdot \theta_{pq}$ and area ${}^{A}z_{pq} = {}^{w}z_{pq} \cdot {}^{h}z_{pq}$ with its centre at the position $\mathbf{\Delta}_{pq}$. In other words, the sub-frame conveys the size and position of the contents of frame f_p relative to the

view frame by frame f_q. If the contents of f_q are now within the view frame we can compute the physical overlap between the sub-frame and the view frame, defined by $\phi(p,q)$, to determine the amount of overlap between the contents of frame f_p and frame f_q.

If $\theta_{pq} \leq 1$, there has either been no scaling or a scale down plus possibly translation in x and y applied to the contents of f_p. In this case, we must compute what proportion of the view frame still contains the contents of f_p to determine the overlap between the contents of f_p and f_q and subsequently the similarity between the frame pair. Hence, $\psi(p,q) = \phi(p,q)/A$, as illustrated in Fig. 5.6(a). In case (ii) when $\theta_{pq} > 1$, there has been a scale up plus possibly translation in x and y so we must compute what proportion of the contents of f_p are still in the view frame, thus $\psi(p,q) = \phi(p,q)/{}^{A}z_{pq}$ as shown in Fig. 5.6(b).

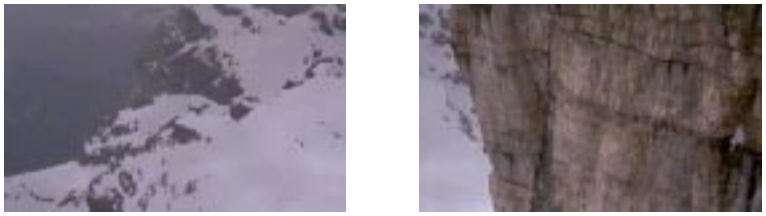
5.5 Representing a Shot as a Graph

We now use the similarity metric $\psi(p,q)$ described above to represent each individual shot as a graph. Let us define a graph $G = \{V, E\}$ comprised of a set V of N vertices, $\{v_1, .., v_N\}$, and a set $E \subseteq V \times V$ of directed weighted edges connecting vertices in V. In a directed graph, each edge also has a direction, so edges (v_p, v_q) and (v_q, v_p), where $i \neq j$, are distinct. The weight of an edge connecting two vertices v_p and v_q is defined by $\omega(v_p, v_q)$. A path from vertex v_p to vertex v_m is a set of connected edges $\{(v_p, v_q), (v_q, v_k), ..., (v_l, v_m)\}$ from E. The weight of path $p = \langle v_0, v_1, ..., v_k \rangle$ is the sum of the weights of its constituent edges:

$$\Omega(p) = \sum_{i=1}^{k} \omega(v_{i-1}, v_p). \tag{5.5}$$

If one or more paths exist from v_0 to v_k the shortest path is defined as the path p with the minimum total weight, $min\{\Omega(p)\}$ [19].

The connectivity of a graph can be represented as an adjacency matrix M in which each element (p,q) represents the edge between vertices v_p and v_q. If there exists an edge (v_p, v_q) then $M_{pq} = \omega(v_p, v_q)$ otherwise $M_{pq} = 0$. Our goal is to form an adjacency matrix for each shot where vertices correspond to individual frames and edge weights are a measure of similarity between each frame pair i.e. $\omega(v_p, v_q) = \psi(p,q)$. Hence, the shortest path from the first to the last frame will minimise the amount of overlap between the contents of the representative key frames. If there is no overlap between two frames f_p and f_q then by definition, $M_{pq} = \psi(p,q) = 0$. This implies that the key frames corresponding to the vertices in the shortest path must always have some overlap of



(a) Two distinct key frames.

(b) Two key frames with overlap.

Figure 5.7: Overlap between key frames makes it easier to determine the storyline during a shot.

their contents. Fig. 5.7(a) shows two distinct key frames representing a single shot and without any overlap it is difficult to ascertain in a glance how the two key frames are related. However, Fig. 5.7(b) shows a different pair of key frames from the same shot with some degree of overlap. It is easier to determine that the camera has panned right during this shot. In fact, we define a threshold T_{min} to specify the minimum amount of overlap there must be between key frames. Thus an edge (v_p, v_q) only exists if $\omega(p, q) \geq T_{min}$. Additionally, to preserve temporal coherence in the video index, a directed edge (v_p, v_q) can only exist if f_q succeeds f_p in the video sequence.

Fig. 5.8 shows a visualisation of the adjacency matrix representing a shot where the camera pans continuously to the right with $T_{min} = 0.2$. The shortest path from the first to the final vertex is $p = \langle 0, 38, 74 \rangle$, with the corresponding key frames shown in Fig. 5.9(a). The weight of the shortest path edges are $\omega(0, 38) = 0.432$ and $\omega(38, 74) = 0.439$ and the total weight is $\Omega(p) = 0.871$. For comparison, the frames representing the second shortest path $p' = \langle 0, 29, 55, 74 \rangle$ are shown in Fig. 5.9(b) with $\Omega(p') = 1.850$. It can be seen that applying the shortest path algorithm results in less representational redundancy.

Fig. 5.10(a) shows the adjacency matrix representing a shot of 289 frames where the camera pans to the right followed by a pan left returning just past the origin. It can be seen where the latter frames start to overlap again with those earlier in the sequence. The shortest path in this graph is $p = \langle 0, 288 \rangle$ and $\Omega(p) = 0.564$. To be an efficient index into a video, the key frames must depict all of the content and convey the temporal order of events in the shot i.e. the camera motion. The key frames here (i.e. 0 and 288) can be seen in Fig. 5.11. In this example, the two selected key frames would not represent any of the video content when the camera pans to the right. We therefore add a final constraint to forming an adjacency matrix such that an edge

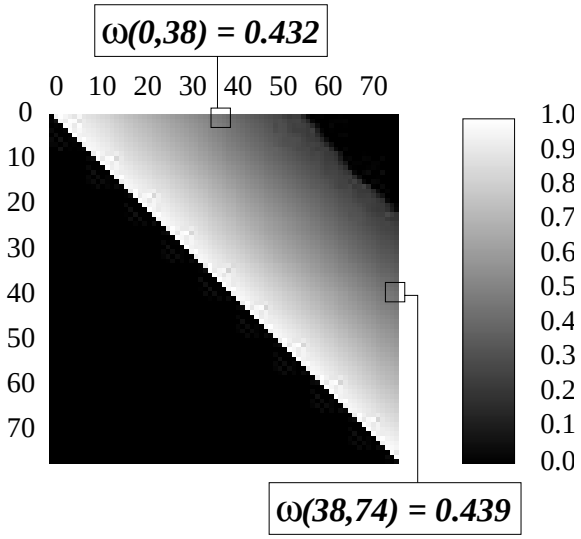
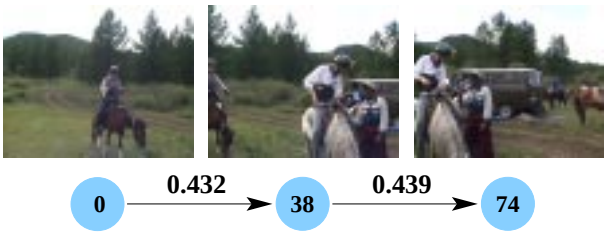
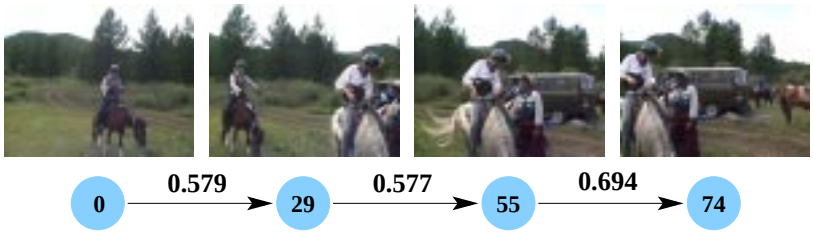


Figure 5.8: Adjacency matrix representing a panning shot with $T_{min} = 0.2$.

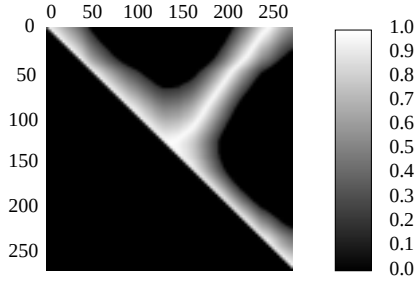


(a) Key frames corresponding to the shortest path.

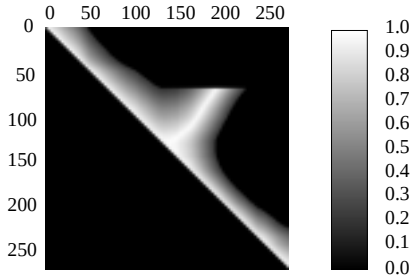


(b) Key frames corresponding to the second shortest path.

Figure 5.9: Applying the shortest path algorithm minimises representational redundancy between key frames.



(a) Any edge exists if $\omega(v_p, v_k) \geq T_{min}$



(b) An edge only exists if $\omega(v_p, v_k) \geq T_{min}, \forall i < k < j$.

Figure 5.10: Two adjacency matrices under different constraints for a shot with a significant pan right followed by a pan left back past the origin.

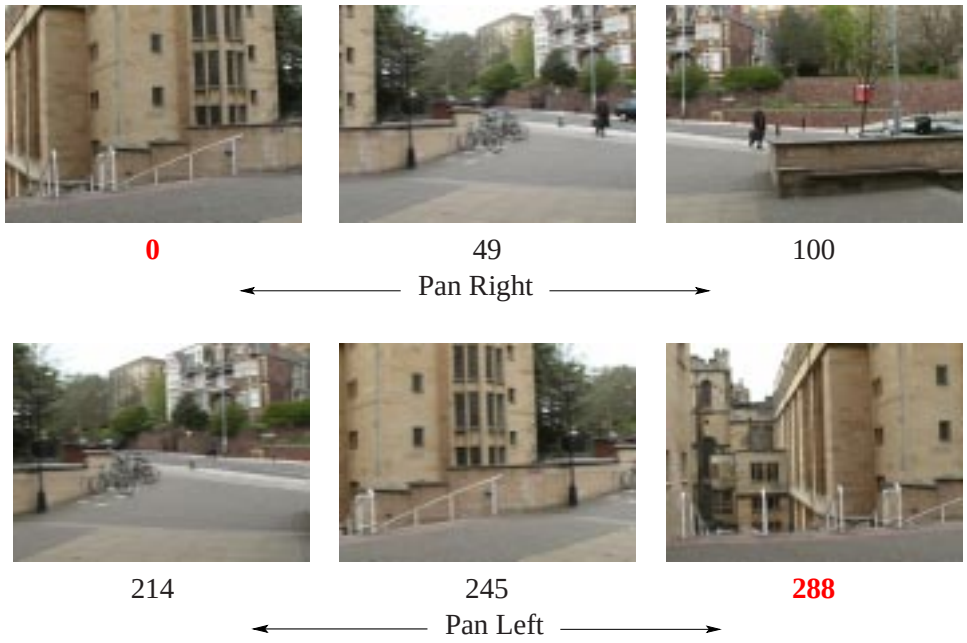


Figure 5.11: Shortest path key frames for the adjacency matrix in Fig. 5.10(b). The two highlighted frames denote the shortest path for the adjacency matrix in Fig. 5.10(a).

(v_p, v_q) only exists if there is more than T_{min} overlap between the corresponding frames f_p and f_q and more than T_{min} overlap between the frame f_p and every frame in between f_p and f_q. That is to say, in addition to the earlier condition that an edge only exists if $\omega(v_p, v_q) \geq T_{min}$, the edge (v_p, v_q) only exists if $\omega(v_p, v_k) \geq T_{min}$ for all $i \leq k \leq j$. Fig. 5.10(b) shows the adjacency matrix representing this shot after this constraint has been added and the shortest path is now $p = \langle 0, 49, 100, 214, 245, 288 \rangle$ with $\Omega(p) = 1.363$ which represent all of the shot content, as shown in Fig. 5.11.

Condition 1	$p < q$
Condition 2	$\omega(v_p, v_q) \geq T_{min}$
Condition 3	$\omega(v_p, v_k) \geq T_{min}$ for all $p < k < q$

Table 5.2: Conditions for which a directed edge (v_p, v_q) exists.

In summary, we form an adjacency matrix for each shot where the vertices represent each frame of the sequence and $\omega(v_p, v_q) = \psi(p, q)$. Table 5.2 outlines the conditions for which a directed edge (v_p, v_q) exists.

5.6 Finding the Shortest Path

To perform an exhaustive search to find the shortest path from a starting vertex to a final vertex, can be computationally expensive. For this reason, we employ the A^* search algorithm which is an informed search strategy [32].

Central to the A^* algorithm is the use of an evaluation function for ordering the vertices in the search space, defined as

$$e(v_p) = g(v_p) + h(v_p) \tag{5.6}$$

where $g(v_p)$ is the actual cost of reaching v_p from the starting vertex and $h(v_p)$ is a heuristic estimate of reaching the final vertex from vertex v_p which must always be an underestimate of the actual cost. It can be shown that an optimistic heuristic h always results in an optimal solution [32]. In our algorithm, $h(v_p)$ is the minimum weight of all existing edges from v_p. Hence, the actual cost from v_p to reach the final vertex will always be greater than or equal to this estimate.

In a graph there may exist several paths of equal length which may turn out to be the shortest. Given two possible shortest paths p and p' with $\Omega(p) = \Omega(p')$, each path corresponds to a different set of key frames. If the total amount of overlap between the selected key frames is the same, is one set of key frames more preferable than the other? Fig. 5.12 illustrates two possible distributions of key frames during a synthetic pan right. The first set of key frames in Fig. 5.12(a) minimises the overlap between the first frame pair (assuming $T_{min} = 20\%$) but results in a large overlap between the second frame pair (60%). In the second set of key frames in Fig. 5.12(b), the overlap between both frame pairs is equivalent (40%) and the key frames are evenly distributed through the camera motion. However, the total amount of overlap between the key frames in both sets is the same (80%) and consequently would result in two paths of equal length. We propose that the second set

(a) Key frames are unevenly distributed.

(b) Key frames are evenly distributed.

Figure 5.12: Two possible distributions of key frames during a synthetic pan right with equal amounts of overlap.

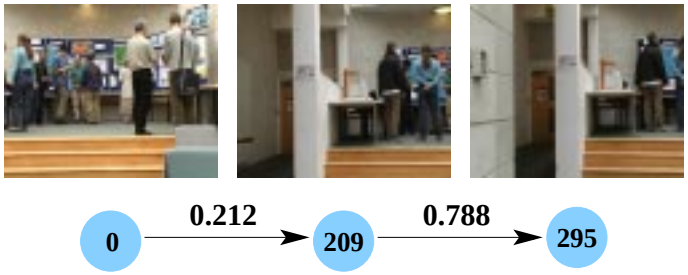
of key frames is more preferable and that given two paths of equal length, ideally the one with the smallest spread of edge weights should be chosen as the shortest path, i.e. the path corresponding to the most evenly distributed key frames. Hence, we compute the standard deviation σ of the constituent edge weights on each path and select the path with the smallest σ. In practice, given a shortest path p there rarely exists another path p' with $\Omega(p) = \Omega(p')$ because of floating point resolution. Even so, there may exist another path p' with $\Omega(p') \approx \Omega(p)$ and $\sigma(p') < \sigma(p)$ that would be preferred as the shortest path if $\mid \Omega(p) - \Omega(p') \mid \leq \varepsilon$ and ε is small. Given a path p with $\Omega(p)$ and $\sigma(p)$, we define a new weight for p as

$$\Upsilon(p) = \Omega(p) + \delta \cdot \sigma(p) \tag{5.7}$$

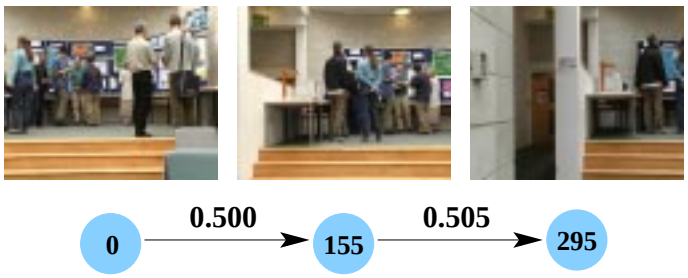
where δ is a weighting. We now choose p if $\Upsilon(p) \leq \Upsilon(p')$, otherwise we select path p'. In practice, in our algorithm, $\delta = 1$ because the standard deviations are small. However, comparing every possible path with the shortest path is not feasible, due to combinatorial explosion. Thus, we need to bias the search for the shortest path towards a path with a small σ of its constituent edge weights. Using the A^* algorithm, the vertices are ordered in the search space using $e(v_p)$. If there are several vertices for which $e(v_p)$ are approximately equal, we want the search to favour the path through the vertex where the standard deviation of the edge weights is the smallest. Therefore, we define a new heuristic

$$d(v_p) = e(v_p) + \delta \cdot \sigma(e(v_p)) \tag{5.8}$$

to order the vertices in the search space. Given several possible paths of approximately equal length the search algorithm is biased towards finding the path with the smallest spread of edge weights. Fig. 5.13(a) and Fig. 5.13(b) show comparative key frames representing the shortest paths found using the heuristics defined in (5.6) and (5.8) respectively, for the same panning shot.



(a) Key frames representing the shortest path p with $\Omega(p) = 1.000$ and $\sigma(p) = 0.288$.



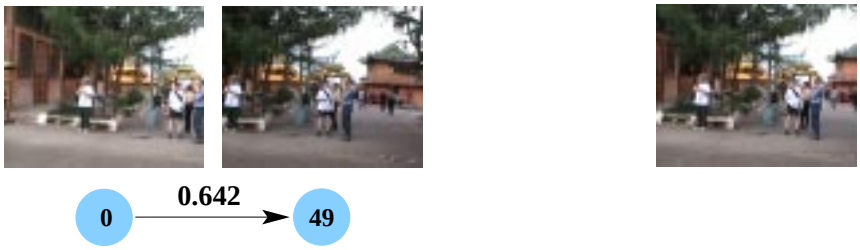
(b) Key frames representing the path p' with $\Omega(p') = 1.005$ and $\sigma(p') = 0.0025$.

Figure 5.13: Two paths with approximately the same weight: the one with the smallest spread of its constituent edge weights is selected.

It can be seen that, at the cost of a slightly longer path, the latter key frames are more evenly distributed through the shot.

5.7 Selecting Key Frames

So far we have used the frames corresponding to the vertices in the shortest path to represent the video content. It follows that there will always be a minimum of two key frames to represent each shot—the first and last frames. However, when there is little or no camera motion a single key frame could potentially be sufficient. To detect this, we introduce a second threshold T_{max} which defines the maximum amount of overlap between two key frames. If there are more than two vertices in the shortest path or there are only two vertices with $\Omega(p) < T_{max}$, then the path accurately summarises the video. However, if there are only two vertices v_p, v_q in p and $\Omega(p) \geq T_{max}$, the single vertex that best represents the edge (v_p, v_q) is selected. This is defined as the vertex v_k corresponding to the frame with the most overlap with all the other frames between and including the first and last frame. That is the vertex v_k with the maximum sum of edge weights $\sum_{l=i}^{q} \omega(v_k, v_l)$, where $\omega(v_k, v_l) = \omega(v_l, v_k)$ if $l < k$ and $\omega(v_k, v_l) = 1$ if $k = l$.



(a) If $\Omega(p) < T_{max}$ the path accurately summarises the video.

(b) If $\Omega(p) \geq T_{max}$ the frame with the most overlap with other frames in the shot is chosen.

Figure 5.14: If there is little of no camera motion a single key frame could potentially be sufficient.

For example, Fig. 5.14(a) shows the first and last frame from a shot during which the camera pans right slightly. If $T_{max} = 0.8$ then the overlap between these two frames is not enough and both frames would be used in the visual summary. However, if $T_{max} = 0.5$ the overlap between these two frames is sufficient and a single frame is selected. In this case, the frame is chosen from the middle of the pan which is shown in Fig. 5.14(b). The values $T_{min} = 0.2$ and $T_{max} = 0.8$ were fixed in all our experiments and to generate the following example video abstracts. These parameters can be set according to user preference.

There is no absolute measure for the quality of an abstraction. Ultimately, the effectiveness of an approach can only be evaluated by users of a video library in which the algorithm is implemented. Here, we shall show some example video abstracts to demonstrate our proposed method. For each shot, we shall show every nth frame which will be deliberately oversampled to portray all the contents of the shot. Alongside we shall show the key frames extracted using the shortest path algorithm and those selected using another method for comparison.

The comparative method used is a mathematical approach rather than a content-based approach and is similar to those outlined in [35, 77, 40, 92]. It transforms each frame of a video sequence into an eigenspace and then groups frames in this space using a line simplification algorithm. Principal Components Analysis (PCA) is applied to perform dimensionality reduction so that the high dimensional image space can be represented in a lower dimensional space whilst retaining the significant variations of the original data [35]. The algorithm assumes the video is segmented into individual shots and each image is scaled to have dimensions 45×36 ($1/16$th of DV PAL image dimensions). Given N frames of a shot, each of which contains M pixels and $N << M$, eigenvectors for the raw image data are computed. The reduction of the dimensionality is achieved by selecting the first d principal components that capture 95% of the variance in the data where d is limited to a maximum of 6. That is, each frame is projected onto the d eigenvectors corresponding to the d largest eigenvalues into its eigen-image representation. A binary line splitting algorithm

is then used to group the frames in the eigenspace. The line simplification algorithm starts with a straight line segment approximating the polyline in the eigenspace from the first to the last point. The remaining points are tested for closeness to that segment and if there are any vertices further than a specified tolerance from that line, the point furthest away is added to the line approximation. The algorithm continues until the distance of each point from the simplification is less than a threshold or until the number of points in the simplification is a maximum of 12. The threshold is equal to 0.5% of the distance from the point in the eigenspace with the maximum variation to the origin. As a result the chosen threshold is relative to the amount of variation within each shot.

The first shot which lasts for 182 frames and contains a substantial zoom-in is summarised in Fig. 5.15 with every 16th frame shown in Fig. 5.15(a). It can be seen that the object with the light blue top moves left across the image slightly but then a zoom-in starts between the 3rd and 4th key frame and continues until the end of the shot. Fig. 5.15(b) illustrates the 7 key frames extracted using the PCA approach and Fig. 5.15(c) shows the key frames selected using our approach. We consider the zoom-in to be the main cause of content change during this shot. It is too large to be represented by only the first and last key frame so we would expect at least one intermediate frame to be selected and the shortest path algorithm does, in fact, result in just 3 key frames.

For the remainder of the examples, the content of each shot is portrayed sufficiently by the key frames selected using either the PCA approach or the shortest path algorithm. For this reason, the shot will no longer be illustrated by evenly distributed frames. The second example is a shot which lasts for 131 frames during which the camera pans continuously to the right. In this example the content in the first frame is completely different to that in the last so again at least one intermediate frame is required to show all of the shot's content. Fig. 5.16(a) illustrates the 8 key frames selected using the PCA approach. The 3 key frames chosen by the shortest path algorithm are shown in Fig. 5.16(b).

The third visual summary represents a shot from a wildlife documentary during which there is no camera motion and very little object motion. In such a case, a single key frame would be sufficient to represent its content which is returned by the shortest path algorithm, as shown in Fig. 5.17(b). However, the PCA approach returns 4 key frames, shown in Fig. 5.17(a), illustrating that the shortest path algorithm can be used to minimise representational redundancy.

As with any method that does not use content-based analysis to select key frames, e.g. simply thresholding a difference metric, the means of operation of the PCA method can only be treated as a 'black box'. That is to say, it is not always obvious what triggers it to select key frames, making its behaviour hard to predict. For this reason, it can be difficult to determine the correlation between a given threshold value and the number of key frames it will return. For example, Fig. 5.18

illustrates the visual summaries for two different shots that both contain a zoom-out. The shortest path algorithm selects three key frames to represent each shot but the PCA approach selects 3 for the first shot and 7 for the second. It is not immediately clear why several more key frames were chosen to represent the contents of the second shot. After viewing the shot itself it can be seen that the camera jitters slightly during the zoom and we assume this is the reason. In addition, the key frames chosen by the shortest path algorithm for the first shot are more evenly distributed through the zoom than those chosen by the PCA approach.

A visual summary should enable a researcher to efficiently browse a large quantity of film material and locate shots containing the visual content they are interested in. Minimising the number of key frames is, therefore, important whilst still representing all of the visual content. We proposed that using estimates of the dominant motion to select key frames would represent the contents and portray the order of events sufficiently for the majority of cases. However, what this method does not address is representing any content change caused by object motion in the absence of any camera motion. For example, Fig. 5.19(a) shows 3 frames chosen by the PCA method from a shot during which a flower blossoms and Fig. 5.20(a) shows 4 frames from a shot during which the object leaves the scene. In both cases, the shortest path algorithm only selects one key frame because there has been no camera motion. However, the content between the first and last frames is different. To overcome this problem, there are several possible solutions. When shots with no camera motion are identified either:

- Always choose the first and last frame.
- Use a method such as the PCA algorithm to select key frames.
- Apply the difference metric used in the shot cut detection method between the first and last frame. If the difference exceeds some predefined threshold the first and last frame are chosen otherwise a single frame is used.
- Perform further content-based analysis to detect if there has been any significant events such as an object entering or leaving.

Ideally, the use of one or a combination of these approaches could be used to distinguish between the examples shown in Fig. 5.19 and Fig. 5.20 which would preferably be represented using two key frames and those in Fig. 5.17 and Fig. 5.21 where one key frame would be sufficient.

(a) Every 16th frame.

(b) Key frames selected using the PCA approach.

(c) Key frames selected using the shortest path algorithm.

Figure 5.15: Three different visual summaries for a shot containing a zoom in.

(a) Key frames selected using the PCA approach.

(b) Key frames selected using the shortest path algorithm.

Figure 5.16: Two different visual summaries for a shot containing a pan right.

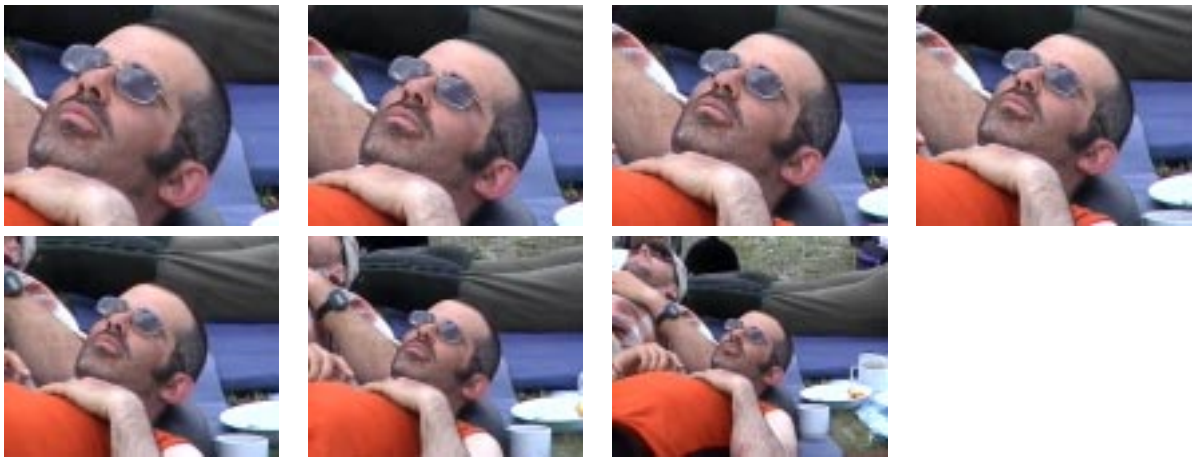
(a) Key frames selected using the PCA approach.

(b) Key frames selected using the shortest path algorithm.

Figure 5.17: Two different visual summaries for a shot containing no camera motion.

(a) Key frames selected using the PCA approach.

(b) Key frames selected using the shortest path algorithm.



(c) Key frames selected using the PCA approach.

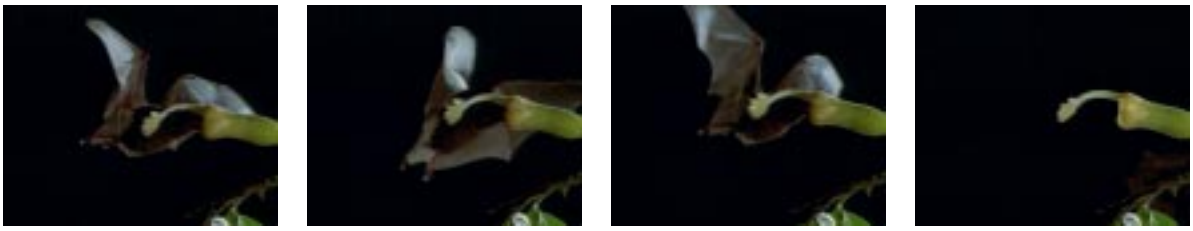
(d) Key frames selected using the shortest path algorithm.

Figure 5.18: Different visual summaries for two different shot containing a zoom in. Using the PCA approach it can be difficult to determine what caused the selection of additional key frames.

(a) Key frames selected using the PCA approach.

(b) Key frame selected using the shortest path algorithm.

Figure 5.19: If there is no camera motion the shortest path algorithm only selects one key frame.



(a) Key frames selected using the PCA approach.

(b) Key frame selected using the shortest path algorithm.

Figure 5.20: If there is no camera motion the shortest path algorithm only selects one key frame. Further processing could be performed to identify any object motion.

(a) Key frames selected using the PCA approach.

(b) Key frame selected using the shortest path algorithm.

Figure 5.21: If there is no camera motion and little object motion, one key frame is sufficient.

5.8 Motion Characterisation

For each shot, the extracted key frames give a graphic, sequential depiction of the narrative; analogous to a storyboard. However, as well as browsing the visual content a user may also wish to perform a specific search. A search query could, for example:

- be for a particular object e.g. ‘find all shots containing a lion’ or ‘return all shots in the Fawlty Towers footage that contain the character Manuel’;
- be scene specific e.g. ‘find a panning shot of the Chicago horizon’ or ‘return all shots of the desert’
- be dependent on specific attributes of the shot, e.g.
  - camera motion: pan, tilt, zoom etc.
  - camera angle: high, low, aerial etc.
  - camera position: close up, long shot, mid shot etc.
  - lighting: artificial, daylight, dark etc.

If a textual annotation exists for a shot it can contain details of such information, for example, ‘A low angle shot pans right following flight of two white terns among trees’ or ‘Mid-length shot of penguin perched on rocks’. Then, a search query can be performed by matching words of interest in the textual description for each shot. However, this work focuses on automated video indexing for data that has not previously been annotated. Key frames provide a visual index into the video content and, in addition, can be used to retrieve similar shots using different image features if an exemplar is provided. However, we propose that future work should investigate the automatic production of metadata that provides descriptive information about the context and characteristics of each shot. The metadata for each shot would then be inclusive of the textual annotations and the visual summary which could then provide a collection of indices to search the video content. The automated production of metadata should aim to provide as much as possible of the information provided by the manual descriptions.

The problem of describing and recognising objects which can then be used to provide meta data such as ‘shot contains a lion’ or ‘shot contains character A’ is an active area of research in itself [25, 31]. However, there are some attributes which can be labelled using current image processing techniques, such as the camera motion. Given that we have estimates of the camera motion as a result of the video summarisation algorithm, this approach is extended to characterise and textually annotate the apparent camera motion contained within each shot. As mentioned above, as well as searching for a shot containing a specific object or scene, a user may also wish to search for a shot which appears to contain a particular type of camera motion. For example, a producer of a wildlife documentary may search an archive for a “fill in” shot which must appear to pan across a particular background (but be less concerned with how the shot was filmed i.e. the actual motion of the camera). If the perceived camera motion has been annotated then only visual summaries of shots containing a pan should be returned as the result of such a query.

To annotate the camera motion contained within a shot we do not just want to label the motion between each consecutive frame pair but to classify the motion for different segments of the shot that contain the same overall camera motion. One approach could be to classify the camera motion between each frame pair and then group successive frames together that have been labelled with the same motion. However, such a bottom-up approach would be sensitive to noise such as momentary changes during a camera motion or camera jitter. For example, Fig. 5.22(a) shows 4 frames from a 66 frame shot which contains camera jitter but no predominant camera motion (the jitter can be seen by looking at the appearance and disappearance of the bush/tree on the left). Fig. 5.22(b) shows the translation in x (red line) and y (green dashed line) between each frame pair. The result of classifying the motion between each frame pair as a pan or tilt, if the motion in the x or y direction respectively is greater than 1, and then grouping similar motions together is also shown. It can be seen that this approach would result in several different motions being

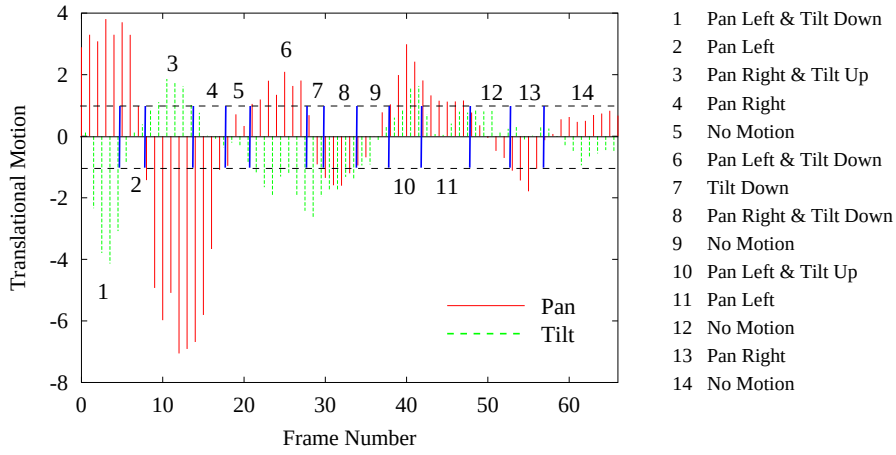
annotated for this shot. Determining whether each cluster of frames actually results in a significant motion that should be annotated would necessitate additional constraints e.g. does the translation exceed a distance threshold, does the motion last more than n frames etc.

However, Fig. 5.22(c) shows the accumulated motion in the x and y direction and it can be seen that the overall translation by the end of the shot in the x and y direction is approximately 5 and 23 pixels respectively. Given that the dimensions of the frames in this sequence are 336×272 the absolute translational motion seems insignificant. In addition, it can be seen that the accumulated motion never varies more than 30 pixels from the point of origin. Both of these factors indicate there was no significant camera motion during this shot. If either motion had deviated by a large distance it could indicate a significant camera motion that had returned to the origin.

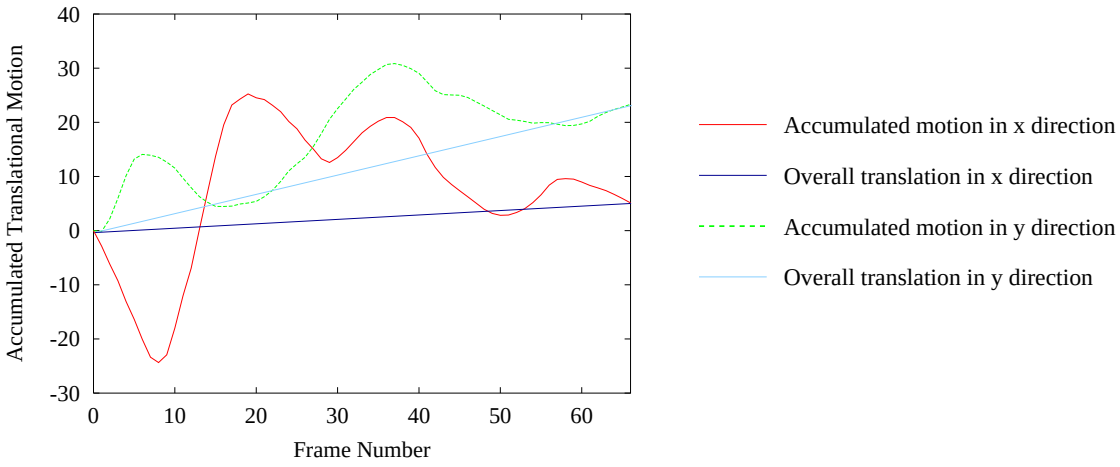
Rather than just measure the deviation of the motion from the origin, this approach can be extended to use a model of the overall motion in the shot and measure how well the actual camera motion fits this model. If it fits well the model can be used to classify the motion within the shot. If the actual motion deviates significantly, then additional motion models can be used to represent smaller segments of the shot. For example, in Fig. 5.22(c) if we assume the models for the motion in both directions fit well, we can then use the average motion, e.g. 5 pixels$/66$ frames $=$ 0.08 pixels per frame in the x direction, to characterise the motion. In this case, we would classify this shot to contain no motion which would be correct. Given the above observations, we use a top-down approach to characterise the motion contained within each shot. We start with a crude initial model of the camera motion between the first and last frame of each shot which is then recursively refined. Fig. 5.22(c) illustrates that if we use the motion estimates between frame pairs to plot a polyline through space then a line simplification algorithm would in fact lend itself well for this refinement process. The resulting line approximation can then be used to classify the camera motion contained in different segments of the shot which results in less sensitivity in the presence of noise, such as camera jitter.

Several different algorithms exist for reducing the points in a polyline to produce a simplified polyline that approximates the original within a specified tolerance. Arguably the least complex is vertex reduction. In this method, vertices that are clustered too closely are reduced to a single vertex. Whereas vertex reduction uses the closeness of vertices as a rejection criterion, the Douglas-Peucker (DP) algorithm uses the closeness of a vertex to an edge segment. Given that we want to measure the closeness of the camera motion to a model which can be represented by a straight line, the DP algorithm best suites this purpose. The result of applying the DP line simplification algorithm is the division of each shot into segments where the rate of change of the motion that appears in the image is constant. Although we use the principle of the DP algorithm to approximate a polyline the conventional distance measure is not suitable for our application.

(a) Frames from a shot containing camera jitter.



(b) Using a bottom-up approach to classify camera motion is sensitive to noise.



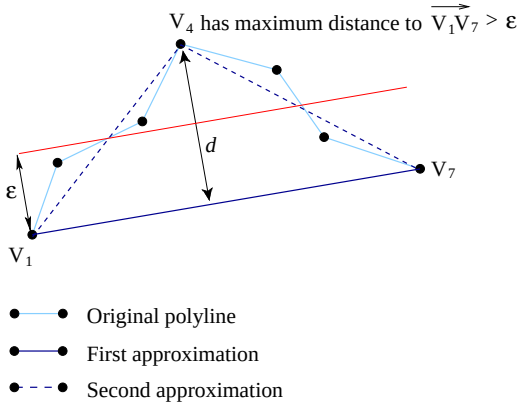
(c) The overall motion contained in this shot is insignificant.

Figure 5.22: A top-down approach results in less sensitivity in the presence of noise.

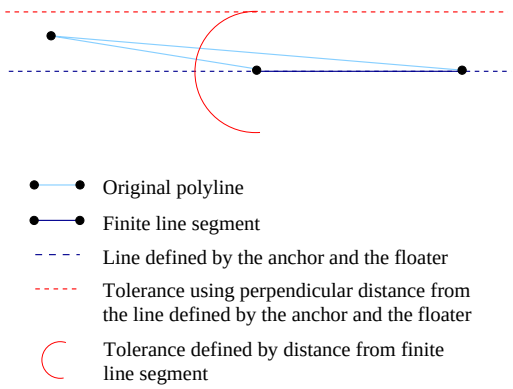
In this section, we start with a brief outline of the DP algorithm and explain how and why we modify the distance metric used to test the closeness of points in the original polyline to its current simplification. Our method is firstly illustrated by classifying the motion in the x direction and it is then extended to classify all three types of motion.

The DP approximation algorithm is used extensively in both Computer Graphics and Geographic Information Systems to reduce the vertices and edges of a polyline. The DP algorithm starts with a straight line between the two endpoints of the polyline as an initial rough approximation. Then, how well it approximates the whole polyline is determined by computing the distances from all intermediate vertices to that line segment. If all these distances are less than a specified tolerance the approximation is good, the endpoints are retained and the other vertices are eliminated. However, if any of these distances exceed the tolerance, the point that is furthest away is taken as a new vertex subdividing the original approximation into two shorter lines, as illustrated in Fig. 5.23(a). This procedure is repeated recursively until all possible points have been eliminated. What can often be ambiguous in a description of the DP algorithm is the definition of the distance criterion used for the selection of an intermediate point. This problem has been addressed by Ebisch who reports that the most widely used measure is to select the point with the greatest perpendicular distance between it and the straight line defined by the anchor and the floater [24]. Here, the anchor and floater are the start and endpoint of a straight line segment respectively. This definition overcomes the problem when the perpendicular distance between the point and the finite segment is undefined. However, the mistake of using this criterion is shown by the polyline with three vertices in Fig. 5.23(b). If this line is simplified with the tolerance band shown by the red dashed line the middle point will be eliminated. Ebisch uses such an example to demonstrate that it is in fact the minimum distance from the point to the segment that is needed, not the distance from the line nor the perpendicular distance from the segment. That is to say, if the perpendicular to the line segment is undefined then the distance to each end point is computed and the minimum distance is used. If this distance is greater than a threshold the intermediate point is included in the simplification. It can be seen in Fig. 5.23(b) that using the tolerance shown by the red line would result in the middle point being included in the simplification.

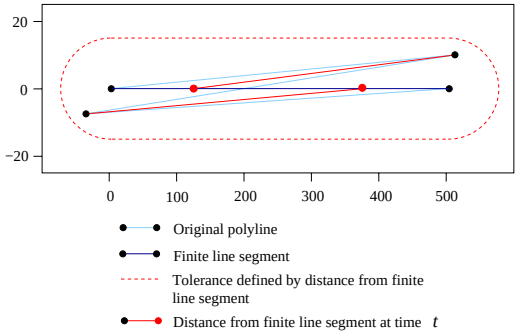
Nevertheless, even with this correction in place the polyline shown in Fig. 5.23(c) would still be simplified to a single straight line. If this simplification was for such purposes as displaying the polyline on a screen then such an approximation might be satisfactory. However, considering this polyline to represent a plot of the accumulated camera motion in the x and y direction at successive time intervals and with the motion in the y direction being negligible, it would correspond to a pan right, followed by a pan left, followed by another pan right. Simplifying this line to enable the correct classification of the different motions contained within the video sequence would require three separate line segments in the approximation.



(a) The point with the maximum distance greater than some tolerance is used to subdivide the approximation.



(b) The middle vertex would be eliminated using a tolerance band determined by the perpendicular distance from the straight line defined by the anchor and the floater.



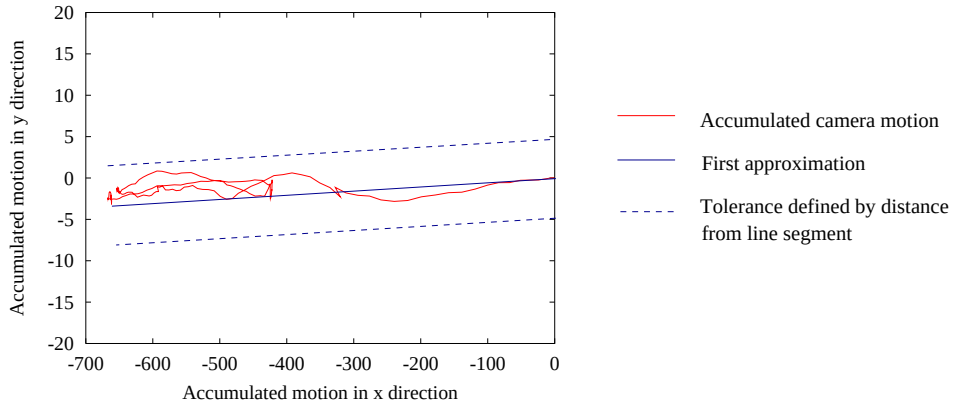
(c) The distance of a point from its expected position at time t is used as the distance criterion for selecting intermediate points.

Figure 5.23: The Douglas-Peucker line simplification algorithm.

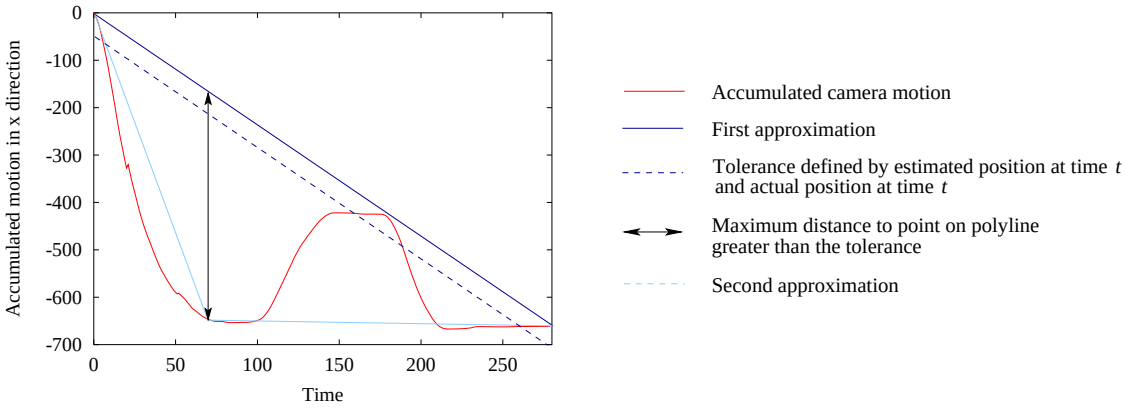
Given that the polyline represents a discrete time series describing the state of the accumulated camera motion at constant time intervals, the actual distance measure required to test the closeness of a point on the polyline to the approximation is the distance between itself at time t and the point on the finite line segment at time t. That is to say, the distance between the actual position at time t and the estimated position at time t using the current approximation. If a polyline contains N vertices and $N-1$ edges representing $N-1$ time steps, the finite line segment between the start and end vertices is subdivided into $N-1$ edges of equal length. The closeness of a vertex on the polyline to the simplification is now equal to the distance between itself and its corresponding vertex. For the polyline in Fig. 5.23(c) the distances of the intermediate vertices from the simplification are shown by the solid red lines which are now greater than the tolerance value, hence these points are included in the approximation.

Although this may appear to be a very specific example, such repetitive motions can occur within a video sequence for several different reasons. For example, during a dialogue between two characters the video can either cut to and from each person or the camera can pan left and right between them within the same shot. Fig. 5.24(a) shows several frames from the same shot which starts with a pan left following one character as he moves across the room (shown by the first 4 frames), the camera then pans back right to focus on the second character that he is talking to (between the 4th and 5th frame) followed by another pan left to focus on the first character whilst he continues to talk (between the 5th and 6th frame). A plot of the accumulated translational motion in the x and y direction is shown in Fig. 5.24(b) which illustrates that using the conventional distance criterion would result in an approximation containing a single straight line segment. However, assuming the motion in the y direction is insignificant, a plot of the accumulated motion in the x direction against time in Fig. 5.24(c) shows that using the distance between the approximated position at time t and the actual position at time t would result in further points being included in the simplification. Another common occurrence of such repetitive motion is during sports footage, such as a football match as a consequence of the camera panning left and right across the field following the play of the ball. The modified distance criterion also addresses the problem of identifying segments of a video sequence with little or no camera motion. This can be illustrated by the example in Fig. 5.25(a) where the blue dotted line is the sum of the translation in x against time. It can be seen that the amount of translation in the negative direction increases, then the camera remains stationary for a period of time followed by a translation in the positive direction until it finally becomes constant for the remainder of the shot. In applying the DP algorithm to characterise the motion, we start with a straight line segment between the two endpoints of the original line as shown by the red line in Fig. 5.25(a). This line segment is an initial rough approximation which describes the overall motion contained within a shot. How well it approximates the camera motion contained within the shot is determined by testing the closeness of all intermediate polyline vertices to that straight line.

(a) Frames from a shot with repetitive motion following a dialogue between two characters.



(b) A plot of the accumulated translational motion in the x and y direction. Using the conventional distance criterion would result in a simplification with a single line segment.



(c) A plot of the accumulated motion in the x direction against time.

Figure 5.24: A modified distance criterion.

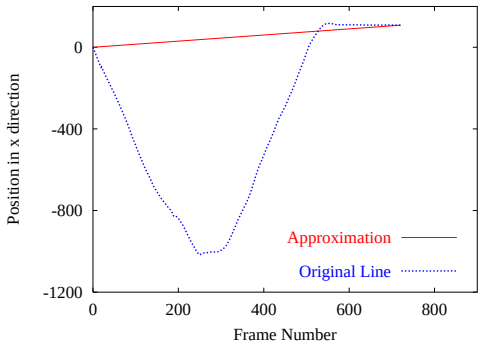
As outlined above we compute the Euclidean distance between the estimated amount of motion and the actual amount of motion at time t. If all these distances are less than a specified tolerance the approximation is good. The endpoints are then retained and the other vertices are eliminated. However, if any of these distances exceed the tolerance the point that is the furthest away is taken as a new vertex, sub-dividing the original approximation into two shorter lines as shown in Fig. 5.25(b). The second line segment now approximates a segment of video where the camera is stationary, followed by a pan left until it becomes stationary again for the remainder of the shot. If the original distance criterion was used, the vertices corresponding to no motion would be at the same position as the start and end points of the line segment resulting in them being very close to the simplification and this line segment being evaluated as a good approximation. However, this line segment represents a constant pan and if this was the true motion in the shot then the accumulated motion would not have increased in the positive direction as much as it actually has by approximately frame 525. This is still a poor approximation. Using the modified distance measure results in the second line segment being subdivided further as shown in Fig. 5.25(d).

This procedure is repeated recursively until all points are within the specified tolerance and a final approximation is reached as shown in Fig. 5.25(e). The result of the line simplification algorithm is an approximation of the original line where the average rate of change of the motion along each line segment is constant within a specified tolerance. We then use the average rate of change to classify the motion as shown in Fig. 5.25(e). We use the term pan to characterise image translation in the x direction, tilt to characterise image translation in the y direction and zoom to characterise image scale. Each line segment represents a different type of motion or a similar motion but with a different rate of change. For example, in Fig. 5.25(e) the camera pans left over 1.5 times faster for the first pan compared with the second. If we want to identify the changes in speed of the apparent motion, the final approximation in Fig. 5.25(e) may be used. However, once the motion has been characterised for each line segment we prefer to merge similar motions together to obtain the final approximation shown in Fig. 5.25(f).

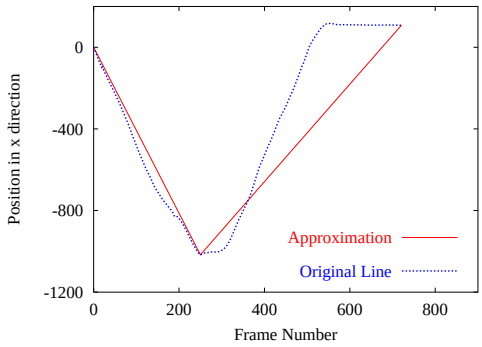
To characterise the combination of all three types of motion we must plot and simplify a line in 4D (scale, translation in x, translation in y, time). For this, the rate of change of each motion between each frame pair must be in the same unit of measure. To achieve this we use the amount of area removed by each motion. Hence, given a consecutive frame pair f_{n-1} and f_n, we redefine the motion parameters s_n and $\boldsymbol{d}_n = ({}^1d_n, {}^2d_n)$ as β_n and $\boldsymbol{\lambda}_n = ({}^1\lambda_n, {}^2\lambda_n)$ respectively where

$$\beta_n = \begin{cases} A(1 - s_n^2) & \text{if } s_n \leq 1 \\ A(1 - 1/s_n^2) & \text{otherwise} \end{cases} \tag{5.9}$$

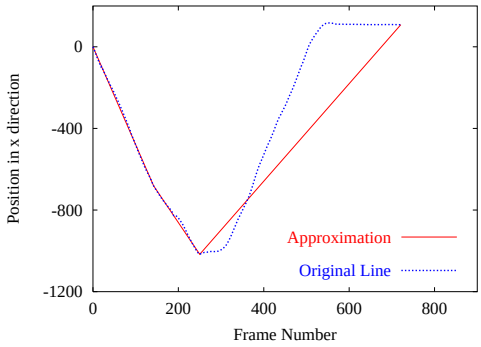
$${}^1\lambda_n = h \cdot {}^1d_n \tag{5.10}$$



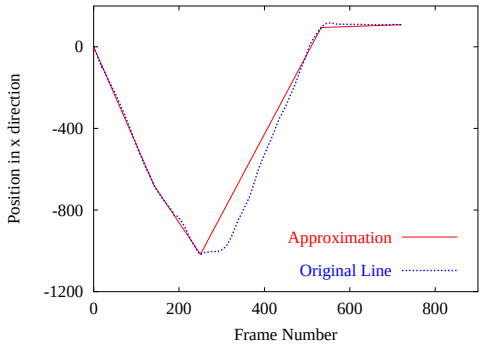
(a) 1st Approximation



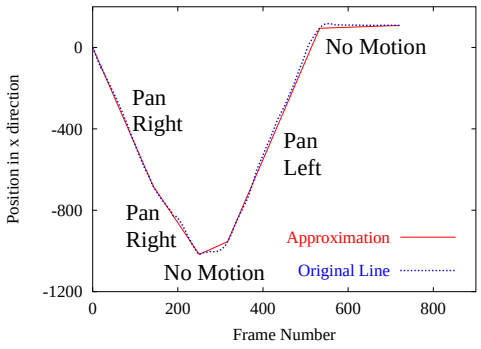
(b) 2nd Approximation



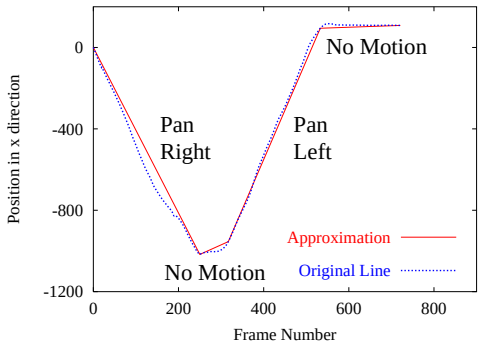
(c) 3rd Approximation



(d) 4th Approximation



(e) Final Approximation using the DP algorithm



(f) Final Approximation after merging similar camera motions

Figure 5.25: Motion characterisation using the Douglas-Peucker algorithm.

Rules	Motion Classification
If $s_{avg} \leq T_{scale}$	Zoom out
If $s_{avg} \geq 1/T_{scale}$	Zoom in
If $\neg\exists$ zoom and $\mid {}^1d_{avg} \mid \geq 1$	Pan
If $\neg\exists$ zoom and $\mid {}^2d_{avg} \mid \geq 1$	Tilt
If $\exists$ a zoom out and $\mid {}^1d_{avg} \mid \geq (w - (w \cdot s_{avg}))/2$	Zoom out and Pan
If $\exists$ a zoom out and $\mid {}^2d_{avg} \mid \geq (h - (h \cdot s_{avg}))/2$	Zoom out and Tilt
If $\exists$ a zoom in and $\mid {}^1d_{avg} \mid \geq ((w \cdot s_{avg}) - w)/2$	Zoom in and Pan
If $\exists$ a zoom in and $\mid {}^2d_{avg} \mid \geq ((h \cdot s_{avg}) - h)/2$	Zoom in and Tilt

Table 5.3: Rules for Motion Characterisation.

$$ {}^2\lambda_n \quad = \quad w \cdot {}^2d_n \tag{5.11}$$

with A, w, and h as defined earlier. We now plot the sum of β_n, ${}^1\lambda_n$ and ${}^2\lambda_n$ against time to obtain a line in 4D to be simplified. Once the original line has been simplified we use the average area removed by each motion to compute the average motion parameters, s_{avg} and $\boldsymbol{d}_{avg}$ for each line segment and use these parameters to classify the type of motion using a rule-based approach. The rules for this classification are shown in Table 5.3 where $T_{scale} = (max\{w, h\} - 2)/max\{w, h\}$. If more than one rule is satisfied then there has been a combination of motions. There are different rules for classifying a pan and tilt depending on whether there exists any zoom. If there is a zoom in then it must appear to zoom in on an object or scene not contained in the original content for there to be any pan or tilt. Likewise, if there exists a zoom out then none or only part of the original content can be present by the end of the zoom out for there to exist any pan or tilt. The direction of a pan and tilt is assigned simply by examining the sign of the parameter. Once the motions have been characterised, the key frame extraction algorithm described previously may be applied between motion boundaries rather than shot boundaries and the motions can be textually annotated. Alternatively, key frames can be extracted at the start and end of each different motion. If we are more interested in searching for a particular type of motion contained within a shot then the textual annotations can be used to present key frames from shots containing this type of motion and nothing else. Fig. 5.26 shows the camera motion annotations for a single shot contain multiple different motions. A key frame has been selected at each transition point between different motions.

5.9 Summary

A novel method for the selection of representative key frames was presented. It was based on the idea of using the amount of shared content between frames as a measure to be minimised in order

Figure 5.26: Camera motion annotations for a single shot containing multiple different motions.

to keep representational redundancy to a minimum. This was achieved by representing each shot as a weighted, fully connected directional graph, where each node corresponds to a frame and the edge weights are the similarity metrics based on the level of shared content. Key frames were then selected by finding the lowest cost path through this graph, and using the frames corresponding to the nodes. It was pointed out that the quality of key frame selection is subjective, and thus difficult to quantify. The chapter presented a comparison with a radically different technique based on PCA. The new technique appeared to result in lower representational redundancy, in that fewer key frames were selected where shot contents varied little.

The chapter concluded with the introduction of a technique for classifying the dominant motion present in a shot. It was based on a top-down refinement of an initial crude estimate of the motion from end frame to end frame of a shot. This initial estimate was then successively improved on using a line simplification algorithm, and was shown to be capable of successfully describing a number of commonly occurring camera operations.

Chapter 6

Conclusions and Further Work

The principal motivation underpinning the presented work was to enable content-based retrieval and efficient browsing of visual information stored in large multimedia databases. This can aid in the reuse of archived footage which can represent a significant saving compared with the cost of re-shooting film. This thesis presented work in the areas of video segmentation, key frame selection and motion characterisation for the purpose of generating video abstracts intended for efficient indexing into long video sequences.

6.1 Thesis Summary

It was argued in Chapter 1 that in order for a video abstract to be useful it needs to fulfill the following two criteria.

1. It should represent the entire content.

2. Representational redundancy should be kept to a minimum.

To represent the entire content the abstract should contain at least one frame from each shot present in the sequence. Moreover, it should be able to represent significant changes in content within shot boundaries, which can result in the selection of multiple key frames per shot. Minimising representational redundancy means that the difference between the selected key frames should be maximised, whilst still capturing the entire content of the video.

Following a review of some recent works in Chapter 2, Chapter 3 introduced a novel abrupt transition detection algorithm which takes into account both image structure and colour information. It used a hierarchical motion compensation scheme utilising a normalised correlation-based technique to estimate local motions of elements corresponding to high-frequencies. The motivation for this approach was that image structure, such as edges, corners and certain textures, are primarily high frequency phenomena. Motion compensation enabled the algorithm to differentiate between content changes caused by camera operations, such as pans, tilts and zooms, and those caused by shot transitions. Using a hierarchical block decomposition enabled the method to adapt its analysis scale to the underlying data to counter the generalised aperture problem, and also allowing for larger disparities to be estimated. It was shown that the method performed well, under the provision that there is enough structure present in the images.

To enable the method to deal with situations where the correlation measure is unreliable, such as images containing motion blur, soft focus, significant local motions, or other cases where there is not enough high-frequency energy present, it was extended to also employ a localised colour histogram difference metric. Local colour histogram differences represent an efficient way of measuring the similarity of images that proved an ideal complement to the correlation metric. On its own, the histogram measure has its own weaknesses, but in conjunction with the correlation metric it proved remarkably effective in detecting shot cuts in real video data. This assertion was backed up by a comparative study against a set of representative methods.

In Chapter 4, the shot-cut detection algorithm was extended to also detect the most commonly occurring gradual transitions, fades and dissolves. Detecting gradual transitions is a harder problem than the detection of cuts, and especially reliable dissolve detection can be considered an unsolved problem today. Many published techniques are based on the assumption that the transition function is linear, which is rarely the case in real video data. Another questionable assumption frequently seen is that gradual transitions will not coincide with camera or object motions.

The nature of a gradual transition means that the difference between frames is small, regardless of the metric employed, suggesting that detection needs to involve multiple frames—it can be assumed that the frame difference between the first and last frame of a gradual transition should be of the same magnitude of that which exists between two frames separated by a cut. A method for detecting dissolve transitions based on the tracking of local regions was presented. Blocks which can no longer be relied upon are removed, and new blocks are introduced where previously covered areas are uncovered, for example when an object leaves the scene. Blocks from previous frames still present in the current frame are monitored, and the differences between each block's contents in the frame it was selected from and the current frame is compared. If the majority of the blocks' differences are sufficiently large, then a dissolve transition was flagged.

The method was also extended to detect fade transitions. It used blank frames to detect the start or end of a potential fade-in or fade-out. The shot cut detection algorithm was then used to distinguish between shot cuts to/from constant images and fade transitions. Once a blank frame was determined to mark the start or end of a fade transition, the opposite boundary was detected by analysing the mean and variance time series. The boundary was detected where there was a significant change in the rate of increase in these sequences. The gradual transition detection algorithm was shown to perform favourably compared with another technique.

The next component required for automatic video abstract generation is the selection of relevant key frames, which was introduced in Chapter 5. The quality of the selection is fundamentally a subjective matter that depends on the application for which the abstract is intended. The approach taken in this work is that a sufficient number of key frames should be presented to convey any significant content change occurring within each shot. In order to achieve this, a metric based on the amount of shared content between the intra-shot frames was devised. A shot was represented as a fully connected, weighted, directional graph where each node corresponds to a frame and the edge weights correspond to the amount of shared content between the frames. The set of key frames was then defined to be those represented by the nodes present on the lowest cost path through the graph. By design, this technique has the property that if a shot contains significant motion that alters the contents, more key frames will be selected. The presented key frame selection algorithm was contrasted against a fundamentally different approach based on Principal Component Analysis. Although difficult to compare performance-wise without large-scale user preference experiments in application context, it was shown that the new method results in lower representational redundancy when inspecting the selected key frame sets from the respective approaches.

Chapter 5 also introduced a technique for characterising the dominant motion present in a shot, which may be used for automatically adding a further layer of descriptive annotation. Textual labels describing the main camera operation may provide another way of indexing and retrieving shots. The classification algorithm developed here used a top-down approach starting with a crude estimate of the motion in a shot which was then recursively refined by means of a line simplification algorithm. It was shown to be capable of describing successfully a number of commonly occurring camera operations: zoom-in, zoom-out, pan, tilt and various combinations thereof.

6.2 Principal Contributions

The work presented in this thesis was concerned with the video indexing task. This was divided into four main components.

1. Detecting shot cuts.
2. Detecting gradual transitions.
3. Selecting the appropriate number of key frames to summarise the video.
4. Categorising the apparent camera motion.

The principal contributions are:

- A novel method for abrupt shot transition detection combining hierarchical normalised correlation with local colour histogram differences. This method was shown to perform well on real video data, requiring minimal parameter tuning as shown on the data used in these experiments.
- A technique for detecting the two most commonly occurring gradual shot transitions, fades and dissolves. This was achieved without resorting to the frequently violated assumptions of linear blending functions and zero motion.
- A novel key frame selector based on the amount of shared content between frames. This was expressed in terms of a shortest path problem, and experiments suggested that the frames selected had a lower degree of representational redundancy compared with a different method.
- A camera motion classifier capable of classifying the dominant motion based on a line simplification technique.

The algorithms developed were demonstrated to perform well, and were all implemented within a unified framework which could be incorporated in an automated video indexing system.

6.3 Further Work

Although the transition detection algorithm and the key frame selector were shown to perform well, several outstanding issues remain to be explored. Some of these are outlined below, together with ideas on how research effort could be spent in order to resolve them.

Detecting Gradual Transitions

The two main causes of false detections are the presence of non-translational motions such as a camera zoom, and the choice of an inappropriate block size of the tracking. Although the incorporation of colour histogram differences helps to overcome these problems, the algorithm could benefit from the investigation of higher-order motion models, for example by allowing the tracked regions to deform. Another possible way of tackling these problems may be to employ a hierarchical block tracking scheme where the size of the analysis window can be adapted to the underlying data. It was judged that both these approaches represent research areas in their own right.

A different area that could be investigated is to try to resolve occlusion issues to improve the updating of the regions of interest. Currently, overlapping blocks are removed, so the content of occluded regions are not compared between distant frames. This can cause the set of regions of interest to be updated too frequently, resulting in a missed detection. Perhaps a depth-ordered representation could be used to resolve this, by allowing moving regions to pass over each other.

Although uncommon in real world video, there exists a number of other gradual transitions, apart from fades and dissolves, collectively known as wipes and pushes. The current work could be extended to detect those.

Selecting Key Frames

One of the fundamental assumptions made here was that the change of content in the face of camera motion was the significant factor when determining whether to select a key frame or not. This means that if a shot contains little or no camera motion, typically a single key frame will be selected. This may not be appropriate for all applications, and the selection of key frames based on object motion and event detection could be investigated further.

The major area that could benefit from further investigation is one of scene-adaptive video encoding [27, 54]. Many video codecs employ some form of inter-frame encoding punctuated by so-called *reference frames*, or temporal decorrelation structures, from which the next set of inter-frame encodings are derived. Work in this area suggest that making use of some data-specific adaption may provide for a better bit rate performance from the codec. It is entirely possible that the presented scheme to detect shot transitions and select key frames could be a useful basis for a video codec.

Although this suggests doing such analysis prior to the coding step, the flip-side of this coin is that

if the video data is already present as an encoded stream it may be possible to exploit the motion compensation information which is typically present to do the transition detection and key frame selection.

Motion Characterisation

The motion characterisation scheme could be extended to recognise more kinds of motions. For example, it currently makes no distinction between a camera pan and a tracking operation. It also makes the assumption that the camera motion is always the dominant one, which, although true for the majority of cases, represents a simplification. As it uses MSAC to fit a motion model to the available data it may be useful to investigate the meaning of the points discarded as outliers. It may be possible to glean information about other motions present, and potentially segment the video into moving objects.

6.4 Concluding Remarks

One of the central issues in this work has been one of data generality. If a video indexing system is to be useful in the wider context, it is essential that it does not depend on myriads of thresholds needing hand-tuning prior to each new video sequence encountered. Many techniques exist that can be tuned to work well for constrained data, but it was decided from the outset that the target for the current work was generality. The author believes that the results presented go some way towards reaching that target.

The question if this work has been successful or not can only be fully answered after the techniques and ideas presented are incorporated into a complete automated video indexing system akin to that outlined in the introduction of this thesis. Saying that, it is the author's opinion that the tests performed were done using a large set of real-world data from a disparate set of genres and sources. Furthermore, the techniques build upon significant amounts of established work, such as normalised correlation and histogram differencing, which should testify to the solidity of the underlying principles. It is the author's hope that the techniques developed here will eventually be deployed in a commercial application for video indexing that would prove useful for film and video researchers doing real production work.

Bibliography

[1] P. Aigrain, H. Zhang, and D. Petkovic. Content-based representation and retrieval of visual media: A state-of-the-art review. *Multimedia Tools and Applications*, 3(3):179–202, 1996.

[2] A. Akutsu, Y. Tonomura, H. Hashimoto, and Y. Ohba. Video indexing using motion vectors. In *SPIE Visual Communication and Image Processing*, volume 1818, pages 1522–1530, 1992.

[3] A. M. Alattar. Detecting and compressing dissolve regions in video sequences with a dvi multimedia image compression algorithm. In *IEEE International Symposium on Circuits and Systems*, volume 1, pages 13–16, May 1993.

[4] A. M. Alattar. Detecting fade regions in uncompressed video sequences. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3025–3028, 1997.

[5] S. Antani, R. Kasturi, and R. Jain. A survey on the use of pattern recognition methods for abstraction, indexing and retrieval of images and video. *Pattern Recognition*, 35(4):945–965, April 2002.

[6] S. Antani, R. Kasturi, and R. Jain. A survey on the use of pattern recognition methods for abstraction, indexing and retrieval of images and video. *Pattern Recognition*, 35:945–965, 2002.

[7] E. Ardizzone and M. L. Cascia. Video indexing using optical flow field. In *IEEE International Conference on Image Processing*, volume 3, pages 831–834, 1996.

[8] J. Boreczky and L. Rowe. Comparison of video shot boundary detection techniques. In *Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 2670, pages 170–179, 1996.

[9] P. Bouthemy, M. Gelgon, and F. Ganansia. A unified approach to shot change detection and camera motion characterization. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(7):1030–1044, October 1999.

[10] R. N. Bracewell. *The Fourier Transform and its Applications*. McGraw-Hill, Inc., 2nd edition, 1986.

[11] L. G. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, December 1992.

[12] R. Brunelli and O. Mich. Image retrieval by examples. *IEEE Transactions on Multimedia*, 2(3):164–171, 200.

[13] R. Brunelli, O. Mich, and C. M. Modena. A survey on the automatic indexing of video data. *Journal of Visual Communication and Image Representation*, 10(2):78–112, June 1999.

[14] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, COM-31(4):532–540, April 1983.

[15] A. D. Calway, H. Knutsson, and R. Wilson. Multiresolution estimation of 2-d disparity using a frequency domain approach. In *British Machine Vision Conference*, pages 227–236, September 1992.

[16] H. S. Chang, S. Sull, and S. U. Lee. Efficient video indexing scheme for content based retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1269–1279, December 1999.

[17] H.-Y. Chen and J.-L. Wu. A multi-layer video browsing system. *IEEE Transactions on Consumer Electronics*, 41(3):842–850, August 1995.

[18] M. Christel, A. Hauptmann, A. Warmack, and S. Crosby. Adjustable filmstrips and skims as abstractions for a digital video library. In *IEEE Advances in Digital Libraries Conference*, pages 98–104, 1999.

[19] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[20] J. D. Courtney. Automatic video indexing via object motion analysis. *Pattern Rocognition*, 30(4):607–625, April 1997.

[21] A. Cowan. Researching film and video on the internet. In *The Researcher's Guide: Film, Television, Radio and Related Documentation Collections in the UK*, pages 16–19. BUFVC, 6th edition, 2001.

[22] A. Dailianas, R. B. Allen, and P. England. Comparison of automatic video segmentation algorithms. In *Proceedings of the SPIE Conference on Integration Issues in Large Commercial Media Delivery Systems*, volume 2615, pages 2–16, 1995.

[23] F. Dufaux. Key frame selection to represent a video. In *International Conference on Image Processing*, pages 275–278, 2000.

[24] K. Ebisch. A correction to the Douglas-Peucker line generalization algorithm. *Computers and Geosciences*, 28(8):995–997, October 2002.

[25] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume II, pages 264–271, June 2003.

[26] A. M. Ferman and A. M. Takalp. Multiscale content extraction and representation for video indexing. In *Proceedings of SPIE Multimedia Storage and Archival Systems II*, volume SPIE-3229, pages 23–31, November 1997.

[27] W. A. C. Fernando and C. N. Canagarajah. Scene adaptive video encoding for MPEG and H.263+ video. *IEEE Transactions on Consumer Electronics*, 47:760–769, November 2001.

[28] W. A. C. Fernando, C. N. Canagarajah, and D. R. Bull. Video segmentation and classification for content based storage and retrieval using motion vectors. In *Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 687–698, 1999.

[29] W. A. C. Fernando, C. N. Canagarajah, and D. R. Bull. A unified approach to scene change detection in uncompressed and compressed video. *IEEE Transactions on Consumer Electronics*, 46(3):769–779, August 2000.

[30] B. U. Film and V. Council. *The Researcher's Guide: Film, Television, Radio and Related Documentation Collections in the UK*. BUFVC, 6th edition, 2001.

[31] A. Fitzgibbon and A. Zisserman. On affine invariant clustering and automatic cast listing in movies. In *Proceedings of the 7th European Conference on Computer Vision*, volume 3, pages 304–320, 2002.

[32] P. Flach. *Simply Logical:Intelligent Reasoning by Example*. John Wiley and Sons Ltd, 1994.

[33] U. Gargi, R. Kasturi, and S. Strayer. Performance characterization of video-shot-change detection methods. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(1):1–13, February 2000.

[34] A. Giachetti. Matching techniques to compute image motion. *Image and Vision Computing*, 18(3):247–260, February 2000.

[35] D. Gibson, N. Campbell, and B. Thomas. Visual abstraction of wildlife footage using Gaussian mixture models and the minimum description length criterion. In *International Conference on Pattern Recognition*, pages 814–817, August 2002.

[36] Y. Gong and X. Liu. Video summarization using singular value decomposition. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 174–180, 2000.

[37] L. Gu, K. Tsui, and D. Keightley. Dissolve detection in MPEG compressed video. In *IEEE International Conference on Intelligent Processing Systems*, volume 2, pages 1692–1696, 1997.

[38] B. Gunsel and A. M. Tekalp. Content-based video abstraction. In *IEEE International Conference on Image Processing*, volume 3, pages 128–132, 1998.

[39] A. Hampapur, R. Jain, and T. Weymouth. Digital video segmentation. In *ACM Multimedia '94 Proceedings*, pages 357–364, 1994. Reference for average intensity over entire frame.

[40] K. J. Han and A. H. Tewfik. Eigen-image based video segmentation and indexing. In *IEEE International Conference on Image Processing*, volume 2, pages 538–541, 1997.

[41] A. Hanjalic. Shot-boundary detection: Unraveled and resolved? *IEEE Transactions on Circuits and Systems for Video Technology*, 12(2):90–105, February 2002.

[42] A. Hanjalic, M. Ceccarelli, R. L. Lagendijk, and J. Biemond. Automation of systems enabling search on stored video data. In *Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases V*, volume 3022, pages 427–438, 1997.

[43] A. Hanjalic and H. J. Zhang. An integrated scheme for automated video abstraction based on unsupervised cluster-validity analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(8):1280–1289, December 1999.

[44] B. Hoffman. Phase 3: Scripting an educational video. In *Educational Video Workshop*. San Diego State University, 1999.

[45] F. Idris and S. Panchanathan. Review of image and video indexing techniques. *Journal of Visual Communication and Image Representation*, 8(2):146–166, 1997.

[46] A. Jepson and M. Black. Mixture models for optical flow computation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 760–761, June 1993.

[47] C. Jones. Transitions in video editing. In *The Encyclopedia of Educational Technology*. San Diego State University, 1994–2003.

[48] R. A. Joyce and B. Liu. Temporal segmentation of video using frame and histogram-space. In *IEEE International Conference on Image Processing*, pages 941–944, September 2000.

[49] R. Kasturi and R. Jain, editors. *Computer Vision: Principles*. IEEE Computer Society Press, 1991.

[50] S. D. Katz. *Film directing shot by shot: visualizing from concept to screen*. Michael Wiese Productions, 1991.

[51] T. Kikukawa and S. Kawafuchi. Development of an automatic summary editing system for the audio-visual resources. *Transactions on Electronics and Information*, J75-A(2):204–212, 1992.

[52] S. H. Kim and R. H. Park. A novel approach to video indexing using luminance projection. In *Proceedings of the IASTED International Conference on Signal and Image Processing*, pages 359–362, 2002.

[53] S. Krüger and A. Calway. Image registration using multiresolution frequency domain correlation. In *British Machine Vision Conference*, pages 316–325, September 1998.

[54] A. Y. Lan, A. G. Nguyen, and J.-N. Hwang. Scene-context-dependent reference-frame placement for MPEG video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 9:478–489, April 1999.

[55] J. P. Lewis. Fast normalized cross-correlation. *Vision Interface*, pages 120–123, 1995. http://www.idiom.com/ zilla/Work/nvisionInterface/nip.html.

[56] R. Lienhart. Comparison of automatic shot boundary detection algorithms. In *SPIE Conf. on Storage and Retrieval for Image & Video Databases VII*, volume 3656, pages 290–301, 1999.

[57] R. Lienhart. Reliable dissolve detection. In *Proceedings of the SPIE Conference on Storage and Retrieval for Media Databases*, volume 4315, pages 219–230, January 2001.

[58] R. Lienhart. Reliable transition detection in videos: A survey and a practitioner's guide. *International Journal of Image and Graphics*, 1(3):469–486, 2001.

[59] R. Lienhart, S. Pfeiffer, and W. Effelsberg. Video abstracting. *Communications of the ACM*, 40(12):54–62, December 1997.

[60] H. B. Lu, Y. J. Zhang, and Y. R. Yao. Robust gradual scene change detection. In *IEEE International Conference on Image Processing*, volume 3, pages 304–308, 1999.

[61] G. Lupatini, C. Saraceno, and R. Leonardi. Scene break detection: a comparison. In *8th International Workshop on Research Issues in Data Engineering*, pages 34–41, 1998.

[62] J. Meng, Y. Juan, and S.-F. Chang. Scene change detection in a MPEG compressed video sequence. In *IS&T/SPIE Symposium Proceedings on Electronic Imaging: Science & Technology*, volume 2419, pages 14–25, February 1995.

[63] A. Nagasaka and Y. Tanaka. Automatic video indexing and full-search for video appearances. In *Visual database Systems*, volume II, pages 113–127, Amsterdam, 1992.

[64] J. Nam and A. H. Tewfik. Dissolve transition detection using b-splines interpolation. In *IEEE International Conference on Multimedia and Expo*, volume 3, pages 1349–1352, 2000.

[65] C. W. Ngo, T. C. Pong, and R. T. Chin. Detection of gradual transitions through temporal slice analysis. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 36–41, 1999.

[66] C. W. Ngo, T. C. Pong, and R. T. Chin. Video partitioning by temporal slice coherency. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(8):941–953, August 2001.

[67] K. Otsuji and Y. Tonomura. Projection detecting filter for video cut detection. In *ACM Multimedia '93 Proceedings*, pages 251–257, 1993.

[68] S. Pfeiffer, R. Lienhart, S. Fischer, and W. Effelsberg. Abstracting digital movies automatically. *Journal of Visual Communication and Image Representation*, 7(4):345–353, 1996.

[69] S. Porter, M. Mirmehdi, and B. Thomas. Video cut detection using frequency domain correlation. In *Proceedings of the 15th International Conference on Pattern Recognition*, pages 413–416, September 2000.

[70] S. Porter, M. Mirmehdi, and B. Thomas. Detection and classification of shot transitions. In *Proceedings of the 12th British Machine Vision Conference*, pages 73–82, September 2001.

[71] S. Porter, M. Mirmehdi, and B. Thomas. A shortest path representation for video summarisation. In *Proceedings of the 12th International Conference on Image Analysis and Processing*, pages 460–465, September 2003.

[72] S. Porter, M. Mirmehdi, and B. Thomas. Temporal video segmentation and classification of edit effects. *Image and Vision Computing*, 21:1097–1106, December 2003.

[73] S. Porter, M. Mirmehdi, and B. Thomas. Video indexing using motion estimation. In *Proceedings of the British Machine Vision Conference 2003*, volume 2, pages 659–668, September 2003.

[74] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1988-1992.

[75] Y. Rui, T. Huang, and S. Chang. Image retrieval: current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10(4):39–62, 1999.

[76] E. Sahouria. Video indexing based on object motion. Master's thesis, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, May 1997.

[77] E. Sahouria and A. Zakhor. Content analysis of video using principal components. In *IEEE International Conference on Image Processing*, volume 3, pages 541–545, 1998.

[78] I. K. Sethi and N. V. Patel. Statistical approach to scene change detection. In *SPIE Proceedings on Storage and Retrieval for Image and Video Databases III*, volume 2420, pages 329–338, San Jose, California, February 1995.

[79] B. Shahraray. Scene change detection and content-based sampling of video sequences. In *SPIE Conference on Digital Video Compression: Algorithms and Technologies*, volume 2419, pages 2–13, February 1995.

[80] M. J. Swain and D. H. Ballard. Colour indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[81] A. M. Tekalp. *Digital Video Processing*. Prentice-Hall, Inc., 1995.

[82] Y. Tonomura, A. Akutsu, K. Otsuji, and T. Sadakata. VideoMAP and VideoSpaceIcon: Tools for anatomizing video content. In *Proceedings of ACM INTERCHI'93, Conference on Human Factors in Computing Systems*, pages 131–136, 1993.

[83] P. H. S. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *CVIU*, 78(1):138–156, 2000.

[84] B. T. Truong, C. Dorai, and S. Venkatesh. Improved fade and dissolve detection for reliable video segmentation. In *IEEE International Conference on Image Processing*, volume III, pages 961–964, September 2000.

[85] B. T. Truong, C. Dorai, and S. Venkatesh. New enhancements to cut, fade, and dissolve detection processes in video segmentation. In *Proceedings of the 8th ACM International Conference on Multimedia*, pages 219–227, 2000.

[86] H. Ueda, T. Miyatake, and S. Yoshizawa. Impact: An interactive natural-motion picture dedicated multimedia authoring system. In *CHI '91*, pages 343–350, 1991.

[87] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[88] J. Vermaak, P. Peraz, M. Gangnet, and A. Blake. Rapid summarisation and browsing of video sequences. In *British Machine Vision Conference*, volume 1, pages 424–433, 2002.

[89] T. Vlachos. Cut detection in video sequences using phase correlation. *IEEE Signal Processing Letters*, 7(7):173–175, July 2000.

[90] W. Wolf. Key frame selection by motion analysis. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 1228–1231, May 1996.

[91] W. Xiong, R. Ma, and J. C. Lee. A novel technique for automatic key frame computing. In *Proceedings of the SPIE Conference on Storage and Retrieval for Image and Video Databases V*, volume 3022, pages 166–174, 1997.

[92] K. Yoon, D. DeMenthon, and D. Doermann. Event detection from MPEG video in the compressed domain. In *International Conference on Pattern Recognition*, volume 1, pages 819–822, 2000.

[93] H. Yu, G. Bozdagi, and S. Harrington. Feature-based hierarchical video segmentation. In *IEEE International Conference on Image Processing*, volume 2, pages 498–501, 1997.

[94] H. H. Yu and W. Wolf. A hierarchical multiresolution video shot transition detection scheme. *Computer Vision and Image Understanding*, 75(1/2):196–213, July/August 1999.

[95] Y. Yusof, W. Christmas, and J. Kittler. Video shot cut detection using adaptive thresholding. In *Proceedings of the 11th British Machine Vision Conference*, pages 362–371, September 2000.

[96] Y. Yusoff, W. Christmas, and J. Kittler. A study on automatic shot change detection. In *Proceedings of the 3rd European Conference on Multimedia Applications, Services and Techniques (ECMAST)*, pages 177–189, May 1998.

[97] Y. Yusoff, J. Kittler, and W. Christmas. Combining multiple experts for classifying shot changes in video sequences. In *IEEE International Conference on Multimedia Computing and Systems*, volume 2, pages 700–704, June 1999.

[98] R. Zabih, J. Miller, and K. Mai. A feature-based algorithm for detecting and classifying production effects. *Multimedia Systems*, 7(2):119–128, 1999.

[99] H. Zhang, A. Kankanhalli, and S. W. Smoliar. Automatic partitioning of full-motion video. *Multimedia Systems*, 1(1):10–28, June 1993.

[100] H. J. Zhang, J. Wu, D. Zhong, and S. W. Smoliar. An integrated system for content-based video retrieval and browsing. *Pattern Recognition*, 30(4):643–658, 1997.

Appendix A

Publications

The work described in this thesis has been presented in the following publications:

1. S. V. Porter, M. Mirmehdi and B. T. Thomas
   Video cut detection using frequency domain correlation.
   In *Proceedings of the 15th International Conference on Pattern Recognition,* pages 413–416, Barcelona, September 2000.

2. S. V. Porter, M. Mirmehdi and B. T. Thomas
   Detection and classification of shot transitions.
   In *Proceedings of the 12th British Machine Vision Conference*, pages 73–82, Manchester, September 2001.

3. S. V. Porter, M. Mirmehdi and B. T. Thomas
   Temporal video segmentation and classification of edit effects.
   In *Image and Vision Computing*, vol. 21, pages 1097–1106, December 2003.

4. S. V. Porter, M. Mirmehdi and B. T. Thomas
   Video indexing using motion estimation.
   In *Proceedings of the British Machine Vision Conference 2003*, pages 659–668, vol. 2, Norwich, September 2003.

5. S. V. Porter, M. Mirmehdi and B. T. Thomas
   A shortest path representation for video summarisation.
   In *Proceedings of the 12th International Conference on Image Analysis and Processing,* pages 460–465, Montova, September 2003.

Appendix B

Computational Efficiency of Normalised Correlation

We wish to establish the point at which the frequency domain implementation of normalised correlation becomes more efficient than the direct implementation. Considering only the numerators in (3.3) and (3.4) and assuming the mean values in (3.3) have already been removed, then (3.3) requires $N^2(M+N-1)^2$ additions and $N^2(M+N-1)^2$ multiplications. That is $\gamma(\boldsymbol{\zeta})$ will have potentially $(M+N-1)^2$ non-zero points (all instances where x_i and y_j overlap) and requires N^2 additions and multiplications at each point. Equation (3.4) requires two Discrete Fourier Transforms (DFT), a conjugate multiplication and an Inverse Discrete Fourier Transform (IDFT). Computing normalised correlation via the frequency domain requires the two square analysis blocks x_i and x_j are of the same size. Assuming they are both of size $M \times M$ where M is a power of 2, using a popular implementation of the FFT algorithm to compute the DFTs, they can be implemented using $M^2 \log_2 M^2/2$ complex multiplications and $M^2 \log_2 M^2$ complex additions and the conjugate multiplication requires M^2 complex multiplications [10]. Therefore, this method requires $12M^2 log_2 M + 4M^2$ real multiplications and $18M^2 log_2 M + 2M^2$ real additions/subtractions. Fig. B.1 shows a comparison of the number of multiplications and the number of additions/subtractions required for each method respectively as the block size M increases assuming $N = M$ for the direct method. Only considering block sizes of a power of two, for a block size of 8 and greater, the transform method of correlating two 2-D sequences is faster than the direct method.

Circular correlation is not used when implementing correlation using the direct method. The 2-D function y_j is embedded within a larger image whose values can be used when x_i overlaps the edges of y_j. For this reason, for an area of interest of size $N \times N$, to estimate equivalent displace-

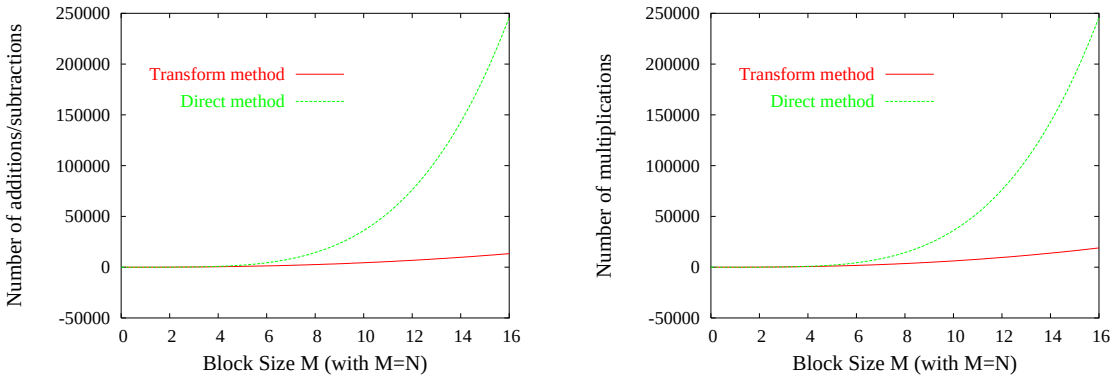


Figure B.1: Comparison of the efficiency of the direct and Fourier computation of normalised correlation.

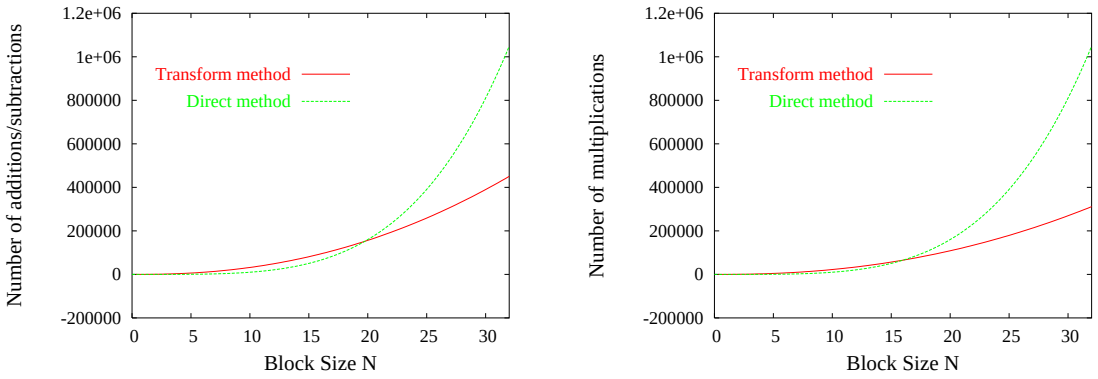


Figure B.2: Comparison of the efficiency of the direct and Fourier computation to estimate displacements $[-N/2+1, N/2]$ for a block size N.

ments $[-N/2+1, N/2]$ requires only N^2 multiplications and additions at N^2 positions. Figure B.2 compares the efficiency of the implementation of each method required to obtain $[-N/2+1, N/2]$ displacement estimates for an area of interest x_i of size $N \times N$. It illustrates that both methods are approximately equivalent for $N = 16$ particularly if only considering multiplication operations. However, the Fourier method is clearly more efficient for a block size 32 and greater.