A Multi-Level Approach for Video Temporal Segmentation based on Adaptive Examples

Robert Babak Yeganeh

B.S., University of Kansas, 2003

Submitted to the Department of Electrical Engineering and Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science

Dr. John Gauch
(Committee Chair)

Dr. Arvin Agah
(Committee Member)

Dr. James Miller
(Committee Member)

Date Defended: June 26, 2006

© Copyrighted by Robert B. Yeganeh
All Rights Reserved

Abstract

Over the past decade, various methods for video shot boundary detection and classification have been proposed and implemented by numerous researchers. Nonetheless, many of these techniques are specific to the type of transition or they are more complicated than necessary. Our research is developed around the idea of creating a simple, real-time and general algorithm which can be used to detect both effects as well as transitions within a video-stream segment. We have implemented several novel methods which have led to such a system. Adaptive examples, use of no thresholds, parameter free algorithms, extremely sensitive change detectors, parallel analyzers and uncertainty groups are among these methods.

Keywords: Video shot boundary detection, video segmentation, temporal segmentation, and threshold.

Acknowledgements

My thanks go to all my mentors, family members, relatives, friends, colleagues, supervisors and other researchers who have directly or indirectly helped me during the years to get to the point I am today.

Even though, all these people deserve to be mentioned, neither time nor space permits the individual recognition of everybody. Hence, only those who have supported me the most are mentioned here.

First most, I would like to thank **Dr. John Gauch**, my advisor of five years, mentor and committee chair for his encouragements throughout my undergraduate and graduate years at University of Kansas (KU). His guidance and counsels have been very valuable throughout the course of this work. His knowledge and research works on video processing, computer vision and digital image processing provided the foundation for this thesis.

I would also like to thank **Dr. Arvin Agah** for serving as my thesis defense committee member and as my directed reading advisor. His suggestions and comments during the years have proved beneficial while working on my thesis.

I want to thank **Dr. James Miller**, my thesis defense committee member, for his dedications to excellence in teaching and for interest in research. The courses I took under him were very informative. His comprehensive and detailed assessments of my works have helped me to improve my programming as well as writing skills.

I would also like to thank **Dr. Nancy Kinnersley**, **Dr. Costas Tsatsoulis**, and **Dr. Jerzy Grzymala Busse** who have served as my main undergraduate instructors and have provided me with valuable feedbacks during the years.

Last but not least, I particularly like to thank my family members since without their support, I would not have reached this point. I owe all my achievements to them.

Giti and Ali Yeganeh, my parents, deserve my special thanks for giving me the necessary supports during the past twenty four years; and of course, my brothers, **Adrien and Sasan Yeganeh**, who have always been there for me.

Thank you... Thank you all!

Table of Contents

List of Figures

List of Tables

Chapter 1

Background

1. Background

1.1. Introduction

Advances in video research have benefited a wide range of industries, such as news websites, television channels, film industries, surveillance and security companies, remote sensing projects, meteorology centers, and medical imaging industries. Further advances in technologies have opened a vast market for video-related products and research. As demand rises, the need for more precise algorithms increases. Consequently, in the recent years significant efforts have been devoted to video data analysis and recognition, including video shot detection, shot grouping, scene detection, classification and retrieval, audio track analysis, video indexing, and motion detection.

Video temporal segmentation (video shot boundary detection) is at the center of many video related research topics. Hence, a lot of time and resources have been invested to improve the existing techniques and to open the door to a more efficient and accurate approaches. This thesis aims toward explaining two novel methods for uncompressed video shots boundary detection and classification in real time. The proposed methods use examples of transitions as basis for detection and classification of transitions, instead of complicated mathematical or statistical models used in other methods. Our first algorithm is based on predefined examples whereas our second method is based on adaptive examples.

1.2. Motivation

Video temporal segmentation is used in many other research areas. Hence it is important to achieve perfect or near perfect results in temporal segmentation. Although this topic has been researched for over one decade, no algorithm has been able to perfectly detect all the transitions. The motivations behind this research was to introduce a distinct algorithm which results in better outcomes for all the three primary transitions (cuts, fades and dissolves) which can easily be expanded to be used against other transitions and effects.

1.3. Goals

Consider a video processing system that detects the common types of transitions such as cuts, dissolves and fades. If in this system, a mathematical model is defined for every transition type then with the introduction of each new transition or effect a new model becomes necessary. Hence, a downfall of these methods is that they tend to be very complex. This has a negative effect on performance. As noted by *Boreczky and Rowe* [16], the simpler algorithms typically outperform the more complicated ones. The second problem with mathematical model approach is that it is not general enough, meaning it cannot be expanded to other types of transitions easily.

Hence, the motivations behind this research was to develop an algorithm that is as simple as possible but not simpler. An example based technique will achieve this goal and at the same time it will be general enough to detect not only all types of transitions but also camera and graphical effects within the video.

With simplicity comes a faster execution speed which is an important fact for many higher level research works which require the output of temporal segmentation algorithms.

1.4. Overview

Many temporal segmentation algorithms can be divided into three main stages (representation, detection and classification) and two optional stages (false prevention and detection). These steps are often merged into a pipeline (see figure 1.1) and occasionally two or more stages are combined for ease of implementation. Our two temporal segmentation algorithms follow this general design. Implementation details are described on the subsequent chapters.

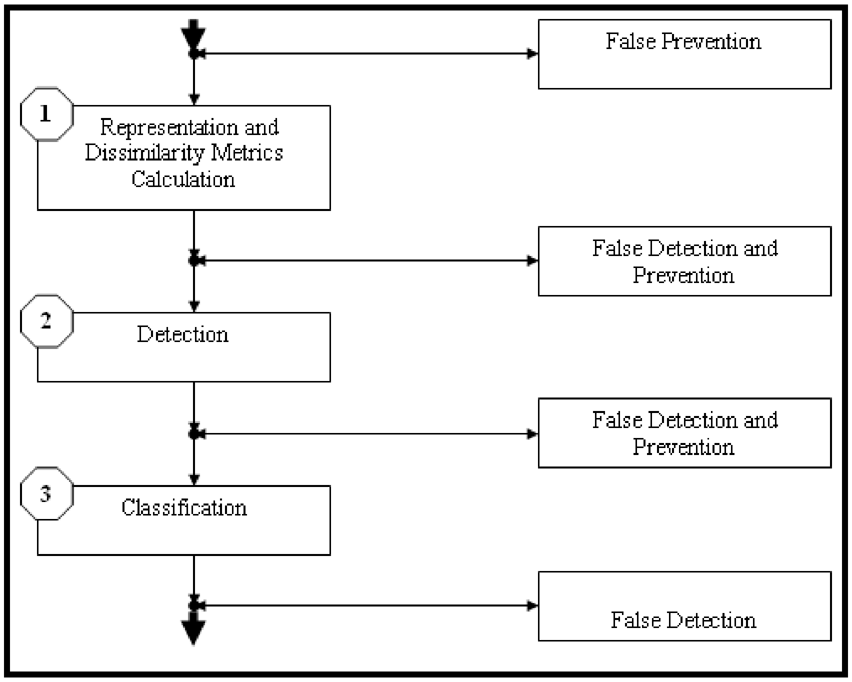

False Prevention
1
Representation and Dissimilarity Metrics Calculation
False Detection and Prevention
2
Detection
False Detection and Prevention
3
Classification
False Detection

Fig. 1.1. Illustrates a general process flow for temporal video segmentation algorithms.

1.5. Thesis Organization

This thesis is organized as follow:

- **Chapter 1** provides the reader with the motivations and goals behind the work and also with an overview of the algorithms used.
- **Chapter 2** reviews existing literature and related work in video temporal segmentation. It also covers topics such as classification algorithms as well as labeling, thresholding, and false positive detection and prevention methods in video shot boundary detection.
- **Chapter 3** discusses the design and implementation details of the first technique, *temporal segmentation based on predefined examples*.
- **Chapter 4** discusses the design and implementation details of the second technique, *temporal segmentation based on adaptive examples*.
- **Chapter 5** presents and discusses the experimental results for both of the algorithms which are discussed in chapter 3 and 4.
- **Chapter 6** contains the summaries, and discussion of future work.
- **Bibliography** contains the list of references used in this document.
- **Appendix A** contains the derivations for some of the equations.
- **Appendix B** contains the glossary of some of the commonly used terms in the field of temporal segmentation as well as terms introduced in this document.

Chapter 2

Literature Review

2. Literature Review

Video related research is a fairly new field as the computing power did not meet researchers' needs previously. Though in the past few years, tremendous growth in video research and technologies has taken place, especially in the area of video temporal segmentation (shot boundary detection).

2.1. Introduction

The research efforts in the last decade have provided the community with great collection of literature. Thus it is essential and worthy to review these works and aim to improve rather than reinvent the wheel. This chapter presents the literature review for different stages of temporal video segmentation as illustrated in figure 1.1.

2.2. Basic Camera Operations

To be able to perform thorough research on temporal video segmentation, it is important to first learn about the basic camera operations used in video capture. Having this knowledge is necessary in order to effectively distinguish between these operations and transitions or effects in question later on. Basic camera operations are divided into three groups: *Linear Movements*, *Rotational Movements*, and *Lens Movements*. Figure 2.1 demonstrates all these movements.

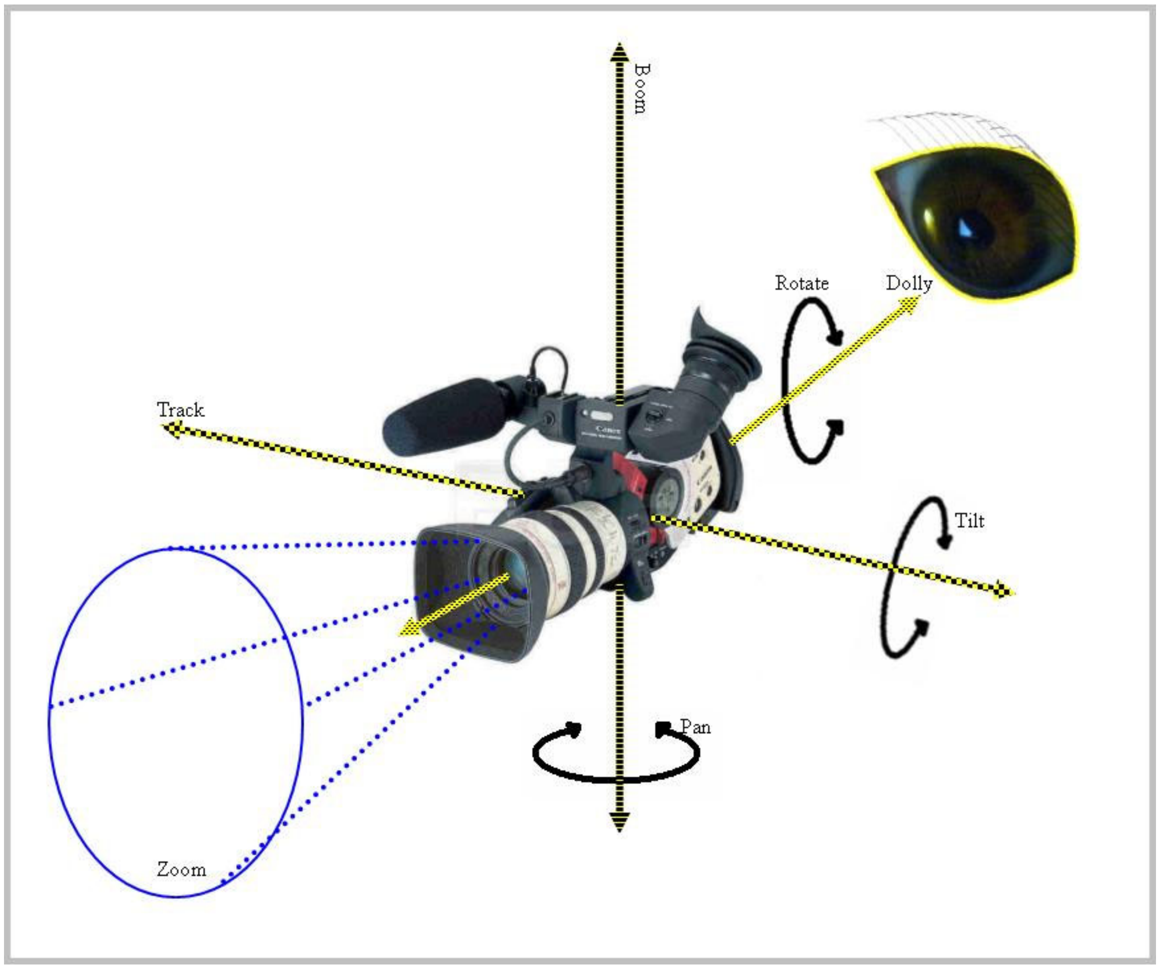


Fig. 2.1. Demonstrates the major camera operations (movements).

Linear Movements are divided into three groups and each is described below:

- **Boom** refers to the movement along the boom axis. In other words, camera moves up and down.
- **Dolly** refers to the movement along the dolly axis. In other words, camera moves back and forth.
- **Track** refers to the movement along the track axis. In other words, camera moves left and right.

Rotational Movements are caused by camera rotation along one of the main three axes.

- **Tilt** is the camera rotation along the track axis.
- **Rotate** is the camera rotation along the dolly axis.
- **Pan** is the camera rotation along the boom axis.

Lens Movements or zooms are due to the movements of lens (back and forth).

- **Zoom in** results in objects appearing closer than they really are or closer than the previous state (in previous frame).
- **Zoom out** results in objects appearing farther than they really are or farther than the previous state (in previous frame).

Camera movements are one of the main reasons behind the lower detection quality in many algorithms. They are mentioned further in this chapter, especially in false detection and prevention related sections.

The next step in producing video programs (such as movies, news, and sport shows) is the editing stage during which the different shots are joined through the use of transitions to create the final product. These topics are discussed in the following section.

2.3. Transitions Overview

Transitions are used to merge different shots into a final product in the editing stage. Many well-known video editing software such as Adobe Premiere and Ulead Media Studio are used by the directors and other people to edit and merge the video shots. On the other hand, our task in this research is to reverse this process. In other words, rather than creating transitions, we have to identify them and extract the surrounding shots.

Many different transition types exist which can be used in the editing process, however as Lienhart claims [63], 99% of all transitions (edits) fall under one of the following three categories:

- cuts,
- fades, or
- dissolves.

Hence, this section mainly focuses on these transitions. These transitions types fall under temporal transition category, meaning they cause a gradual frame by frame space-wise global change in pixel intensities of one shot which will eventually lead to a frame in the next shot.

Although having the knowledge of other types of transitions, is also of a great interest since many of them can create confusion while detecting one of the main transitions. Therefore also discussed in this section, are the following transition types:

- wipe, and
- graphical transitions.

One major point of distinction among different transition types is the length. According to this property they are divided into two groups. First are those transition types which lack any actual lengths (i.e. the transitions which can be defined as a sudden change in video stream while moving from one frame to the next). The second category can be defined as those transitions which are presented as a frame by frame modification of one shot which will eventually lead to the next. These categories are labeled as abrupt and gradual transitions respectively.

2.3.1. Abrupt Transitions (Cuts)

Almost all abrupt transitions are cuts. In fact they are common enough to be frequently used in place of abrupt transitions in the existing literature. This transition type has also been labeled as breaks and hard cuts by the previous researchers of the field.

As previously mentioned cuts lack any actual length. In other words, they are defined as a sudden change in the video stream while moving from on frame to the next. These transitions are easier to detect than gradual transitions since they are presented as an abrupt change between two frames of no correlation [6]. Figure 2.2 displays a frame sequence of two shots merged together by a cut.

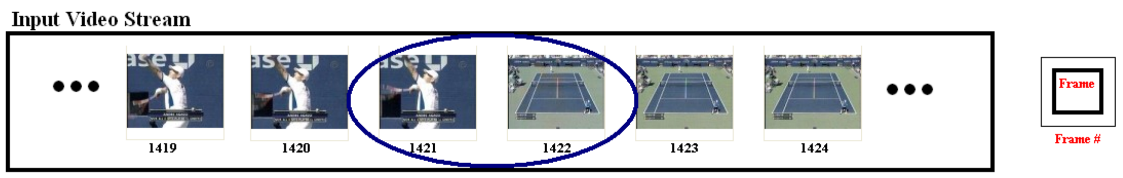


Fig. 2.2. Demonstrates an example for an abrupt transition (cut).

Figure 2.3 demonstrates the work performed by Miadowicz [73] to represent the video stream in a numerical format. As this figure demonstrates, the cut transition between two frames is clearly distinguishable from the rest of the video sequence.

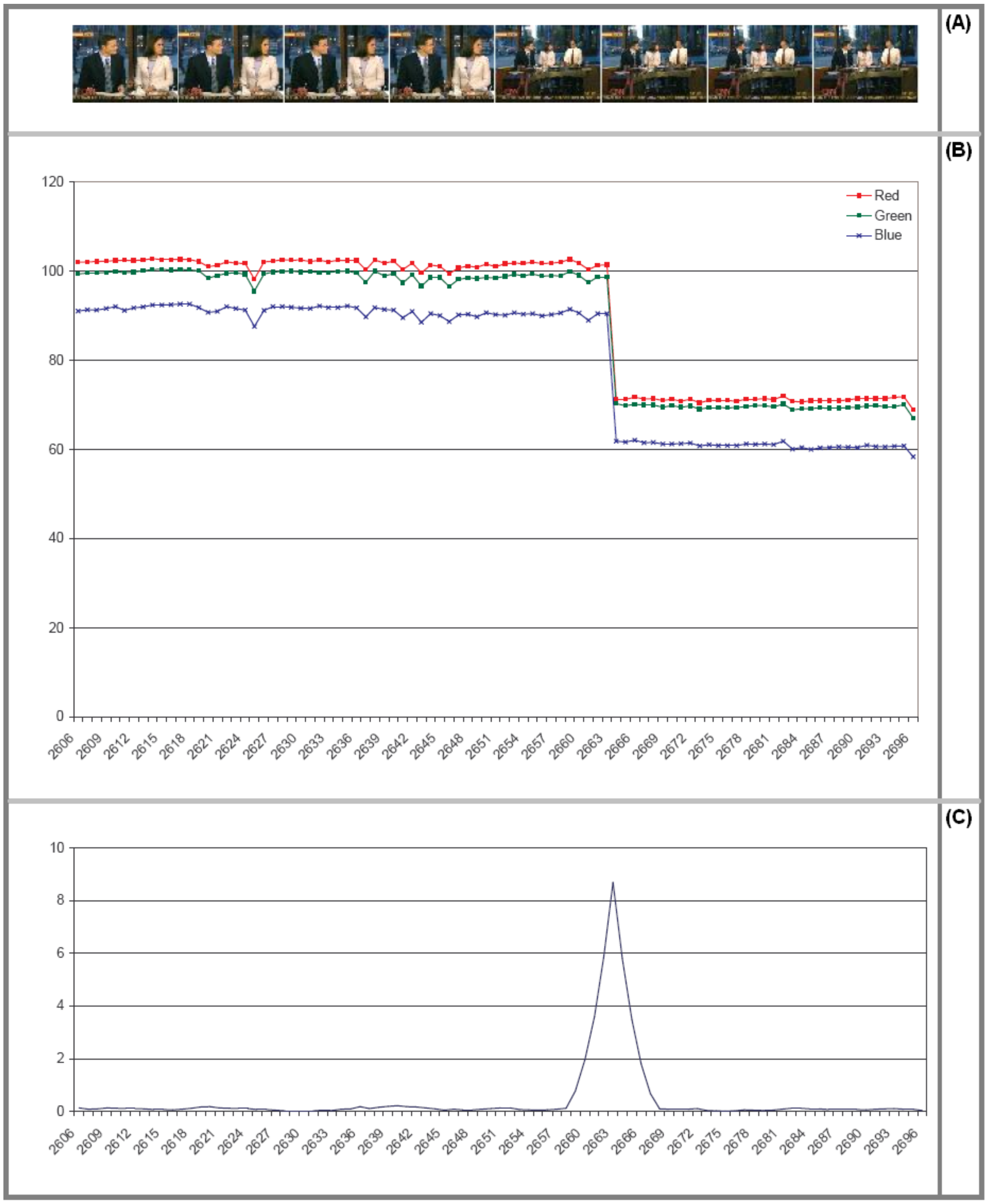


Fig. 2.3. Demonstrates an abrupt transition (cut) in a numerical representation format of a video stream. (A) Presents the video sequence. (B) Presents the numerical representation of the video sequence and (C) Presents the first order derivative or cross differences curve for the same cut transition.

Even though, cut transitions can be easily detected in an usual case (such as figure 2.2), due to complications such as camera movements, graphical effects, and other transitions, no one has been able to develop a system that detects all types of transitions within all types of video with a 100% accuracy. Therefore, considerable amount of research has been dedicated to cut detection, such as [34], [36], [63], [74], [75], [88], [100], and [103]. These works are reviewed later on in this chapter.

Most of these methods result in an acceptable detection output for example, Lienhart in comparison of different techniques [63], states that hard cuts detection was very reliable in most cases and 95% hit rate at 5% false hits are attainable. He adds that the false positives are caused by dark or very dynamic scenes with strong object motion, blasts or fast camera pans. Due to the high accuracy of cut detection algorithms, many researchers have shifted their focus from cut recognition to improving the gradual transitions related techniques.

2.3.2. Gradual Transitions

As previously mentioned, gradual transitions are defined as a frame by frame modification of one shot which will eventually lead to the next. Depending on these modifications, gradual transitions are categorized into various groups. Fades, dissolves, wipes and graphical transitions are among those discussed in this section.

Fade

During editing, fades are produced by use of monochrome frames and usually linear scaling of the pixel intensities or their statistical representation [16]. A complete fade is consisted of two parts, fade out and fade in. During a fade out, the first shot gradually transforms into a monochrome frame whereas during a fade in, the monochrome frame transforms into the frames of the next shot. Figure 2.4 demonstrates a complete fade transition where frame 428 represents the monochrome frame.

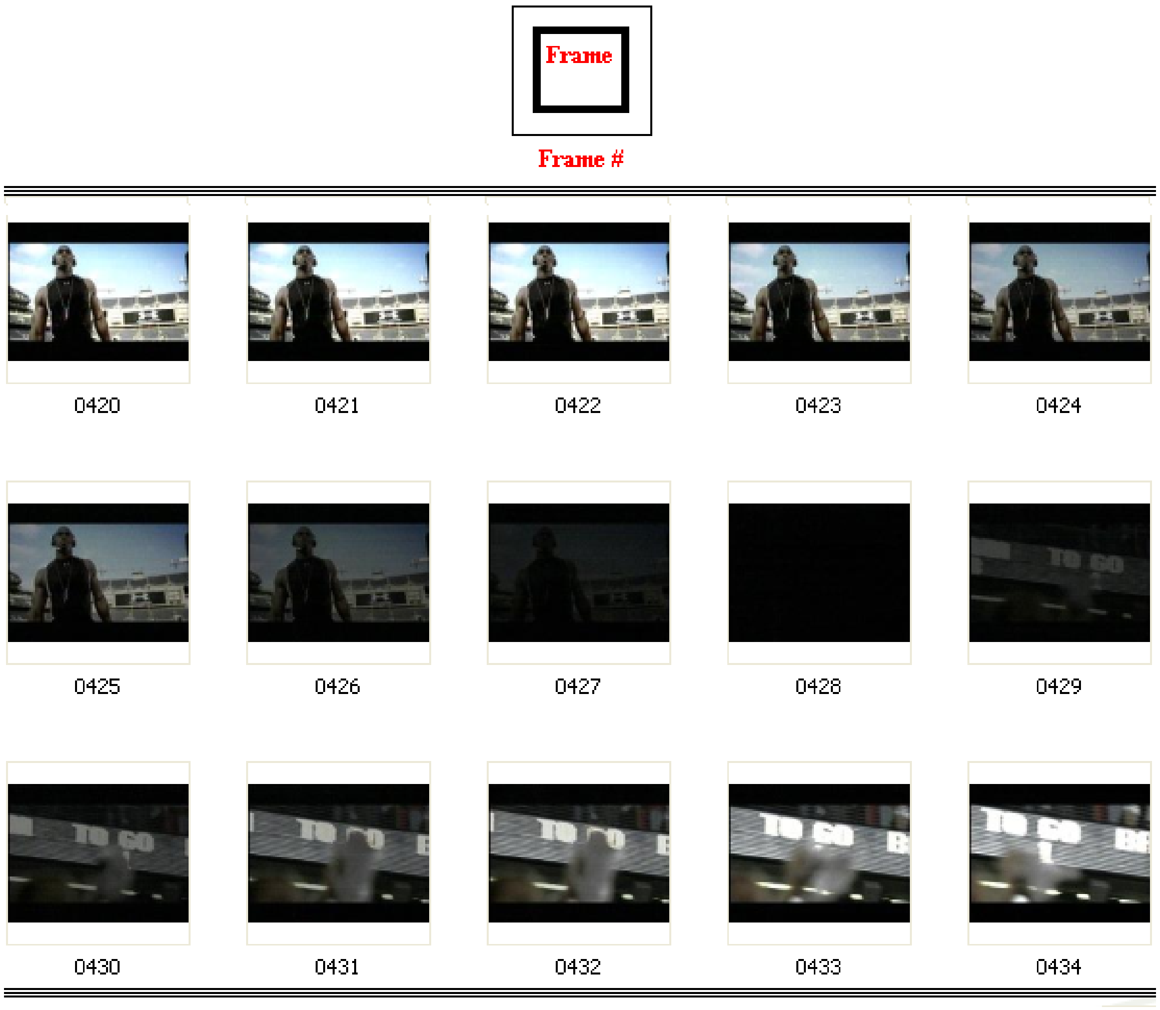


Fig. 2.4. Demonstrates a complete fairly short fade to black.

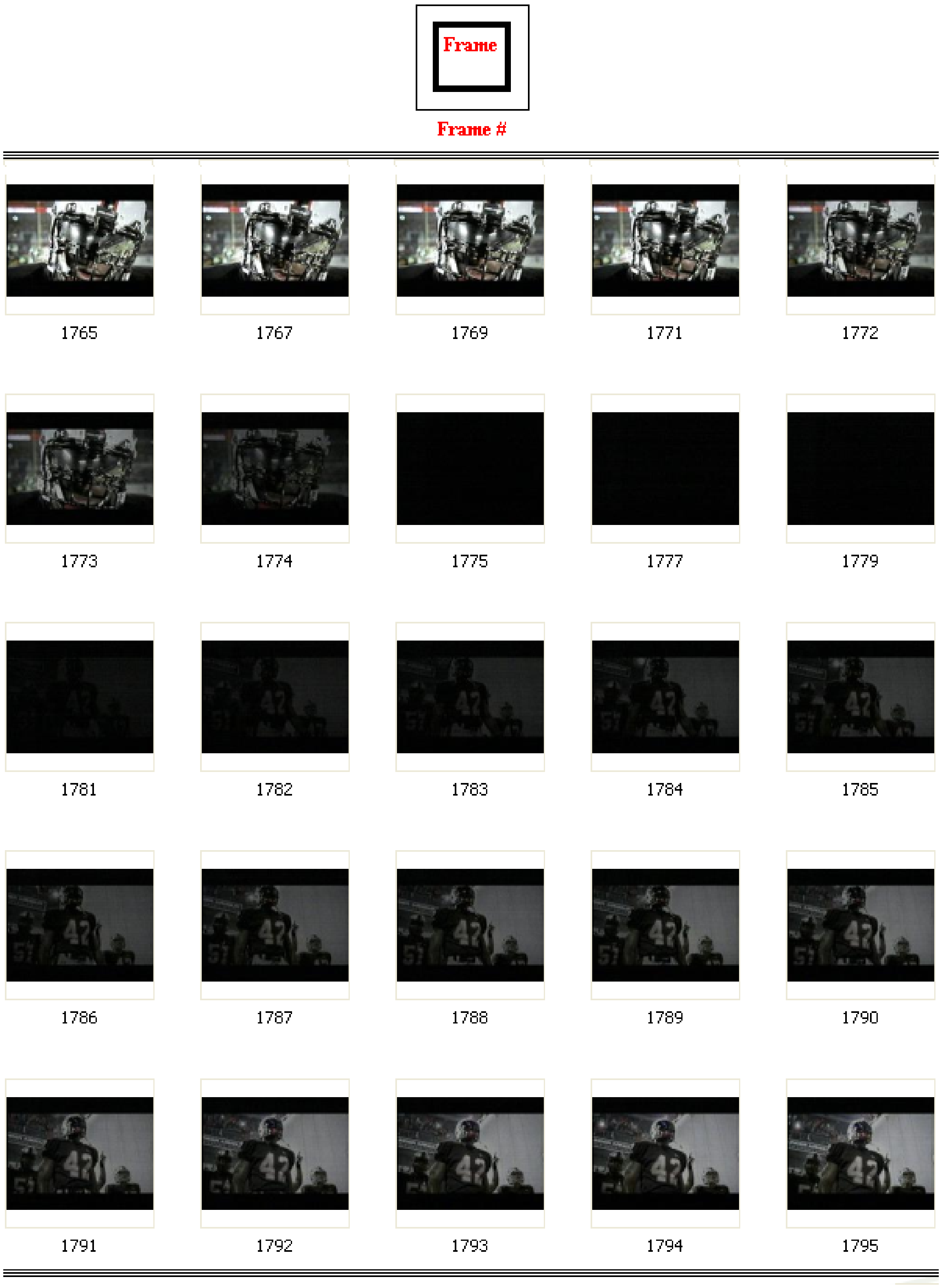


Fig. 2.5. Demonstrates a complete fade to black with several monochrome frames in the middle.

A fade transition can have one monochrome frame between fade out and fade in (such as the fade in figure 2.4) or it can be consisted of several such frames. Example of latter fade, is presented by figure 2.5. No specific limit exists on number of monochrome frames or the length of fades in and fades out. Hence, depending on the assumptions of segmentation algorithm, a very long sequence of monochrome frames can be counted as a separate shot if desired.

Truong, *et. al*. [103] aims at improving the fade detection algorithms by considering different properties of fades. These properties and authors' other observations are as follow:

1. All fades have one or more monochrome frames.
2. There are large negative spikes that appear near beginning of a fade-out and ending of a fade-in on the second derivative curve of luminance variance.
3. Depending on whether frames sequence is fading-in or fading-out, the variance of fading frames will increase or decrease rapidly.
4. To avoid false positives caused by dark scenes, the variance or the starting frame of a fade-out and the ending frame of a fade-in should be limited to be above a threshold.

Truong has also improved the existing techniques by proposing many enhancements through analysis of variance and mean curves. Similar techniques are used by Miadowicz [73] for fade detection. Figure 2.6 demonstrates a fade sequence along with its numerical representation. It represents a special case where there is a cut combined by a fade in. As will be discussed in more details later this scenario cannot be detected by many transitions.

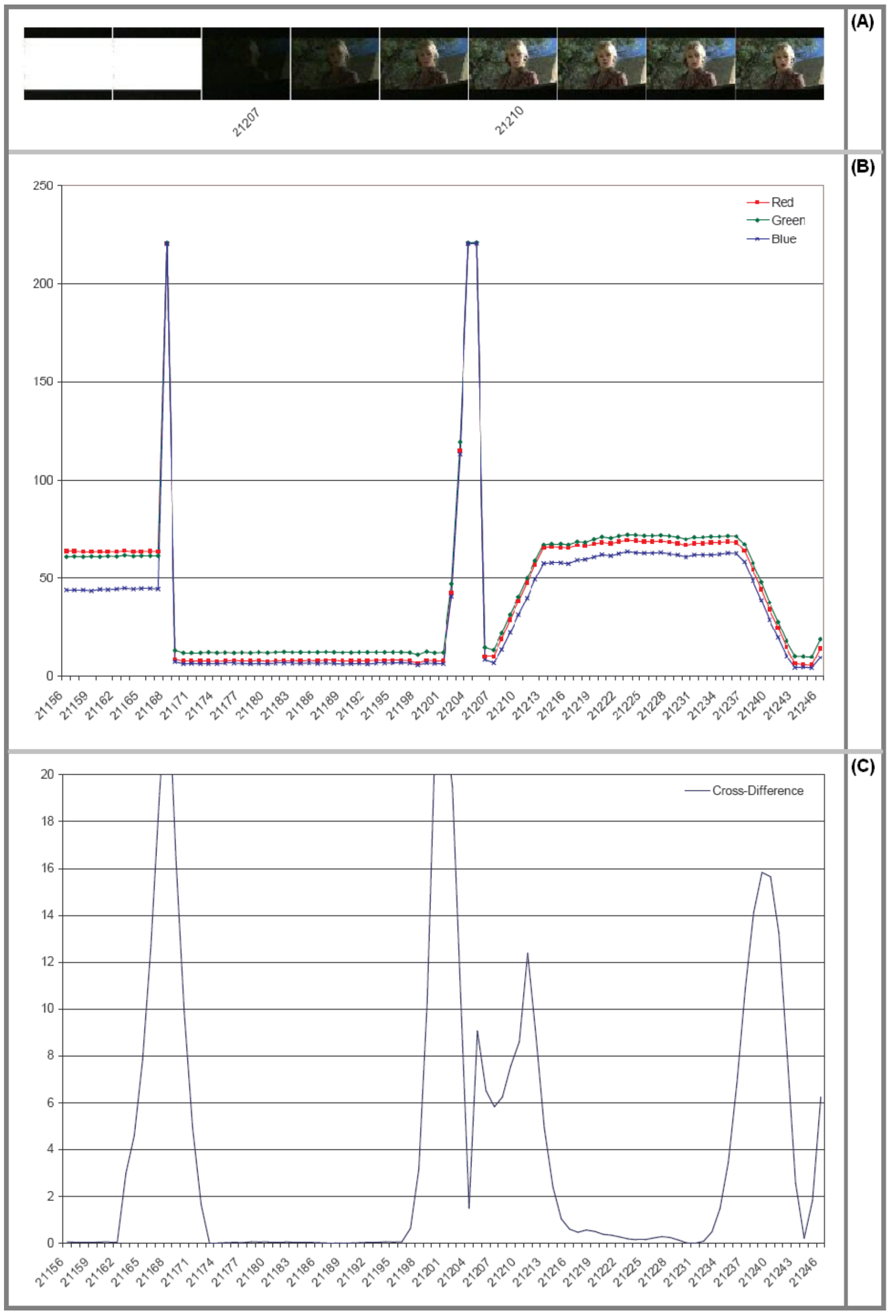


Fig. 2.6. Demonstrates a cut transition immediately followed by a fade transition in a numerical representation format of a video stream. (A) Presents the video sequence. (B) Presents the numerical representation of the video sequence and (C) Presents the first order derivative or cross differences curve for the same transitions.

Hampapur *et. al.* represents a mathematical model for fade out transitions in their paper [50]. This model is presented in equation 2.1. In this equation, *E(x, y, t)* represent the generated pixel intensity value for each pixel located at (x, y) coordinate of frame *t*, d^i represents the length of the transition, $S_{i-1}(x, y, t + t_e^{i-1})$ represents the current pixel intensity at location (x, y) for the segment of the shot S_{i-1} starting at time t_e^{i-1} which is used to generate the transition. i sub- and superscripts represent the item index number. Finally, $(1 - \frac{t}{d^i})$ is used as the scaling function. Fade in follows a similar model and further details on both equations are available in [50].

$$E(x, y, t) = S_{i-1}(x, y, t + t_e^{i-1}) \cdot (1 - \frac{t}{d^i}), \qquad t \in [0, d^i] \tag{2.1}$$

Improper use of scaling function can lead to an effect such as the one presented in figure 2.7. In such a case, the fades in or fades out do not start or end (respectively) with a monochrome frame. Hence depending on the specifications the detection algorithms can label such cases as fade or simply as change in brightness.

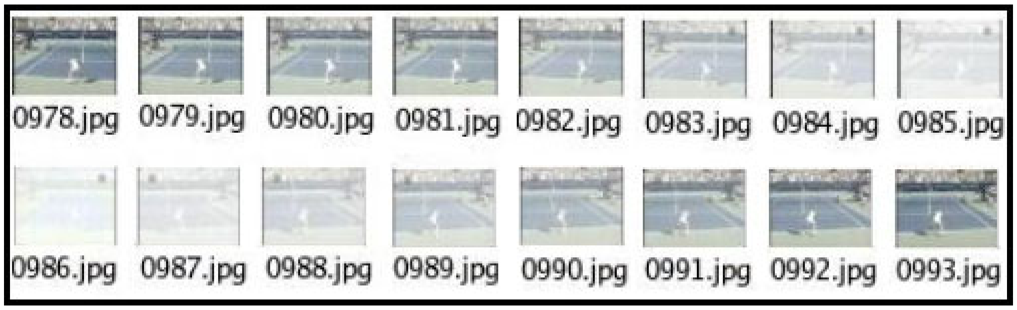


Fig. 2.7. Demonstrates an incomplete fade to white.

Fades in and fades out are specific cases of dissolve transition type where the first or second shot respectively is a monochrome shot. Hence the equations presented in dissolve section can also be applied to fades.

Dissolve

A Dissolve is generated through gradually decreasing the effect of the pixel intensities of the ending frames in the outgoing shot and increasing the effect of the pixel intensities of the beginning frames in the incoming shot. According to [3] dissolves can be grouped into two main categories (cross-dissolve and additive dissolve) depending on the scaling function for incoming and outgoing shots. Perry [87] takes this one step farther by naming many different types of dissolve. According to [87], types of dissolves include additive, cross, dip to color, dither, fade in/fade out, non-additive, random invert, and ripple.

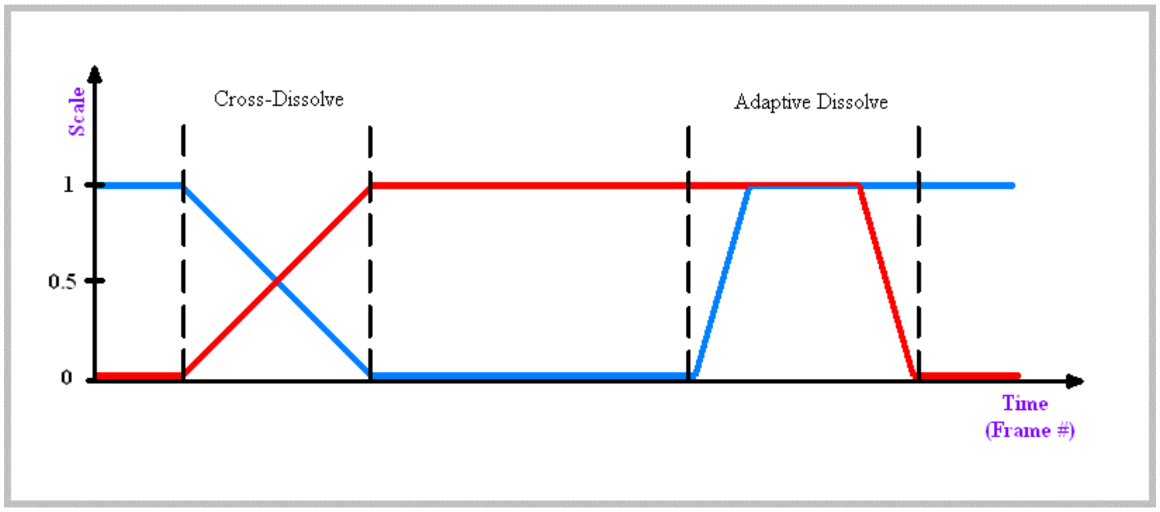


Fig. 2.8.Illustrates variations of intensity scaling functions used to produce dissolves transitions.

Figure 2.8 illustrates these different scaling functions. In both cross and additive dissolve types, intensity scaling function of the outgoing shot decreases with the same rate as the intensity scaling function of the incoming shot increases. The difference between the two is in additive dissolve scaling functions; during the transition, both scaling functions for incoming and outgoing shots of adaptive dissolve, are at their highest values.

Cross-dissolves are more common than additive dissolves. Hence many techniques do not distinguish between the two. Recall that a dissolve transition, *$S_n(i, j)$*, starting from frame L_1 and ending at frame *(L_1+F)* is modeled by [6] (equation 2.2).

$$S_n(i,j) = \begin{cases} f_n(i,j) & 0 \le n \le L_1, \\ \left[1-(\frac{n-L_1}{F})\right]\cdot f_n(i,j) + (\frac{n-L_1}{F})\cdot g_n(i,j) & L_1 \le n \le (L_1+F), \\ g_n(i,j) & (L_1+F) \le n \le L_{2.} \end{cases} \tag{2.2}$$

This equation represents a cross dissolve. The model for additive dissolve can be obtained by simply modifying the intensity scaling factor $(\frac{n-L_1}{F})$.

Similar to cuts and fades, various mathematical and statistical models based techniques as well as learning based methods exist for dissolve detection (such as [6], [30], [63], [64], [78] and [103]). Lienhart [64] developed and evaluated many learning techniques for video shot boundary detection including neural networks, support vector machines, and linear vector quantization whereas Ngo [78] focused on detection based on support vector machines. Truong *et. al.* improved the existing mathematical model based techniques for dissolve detection by proposing many enhancements through analysis of variance and mean curves of main transition types [103].

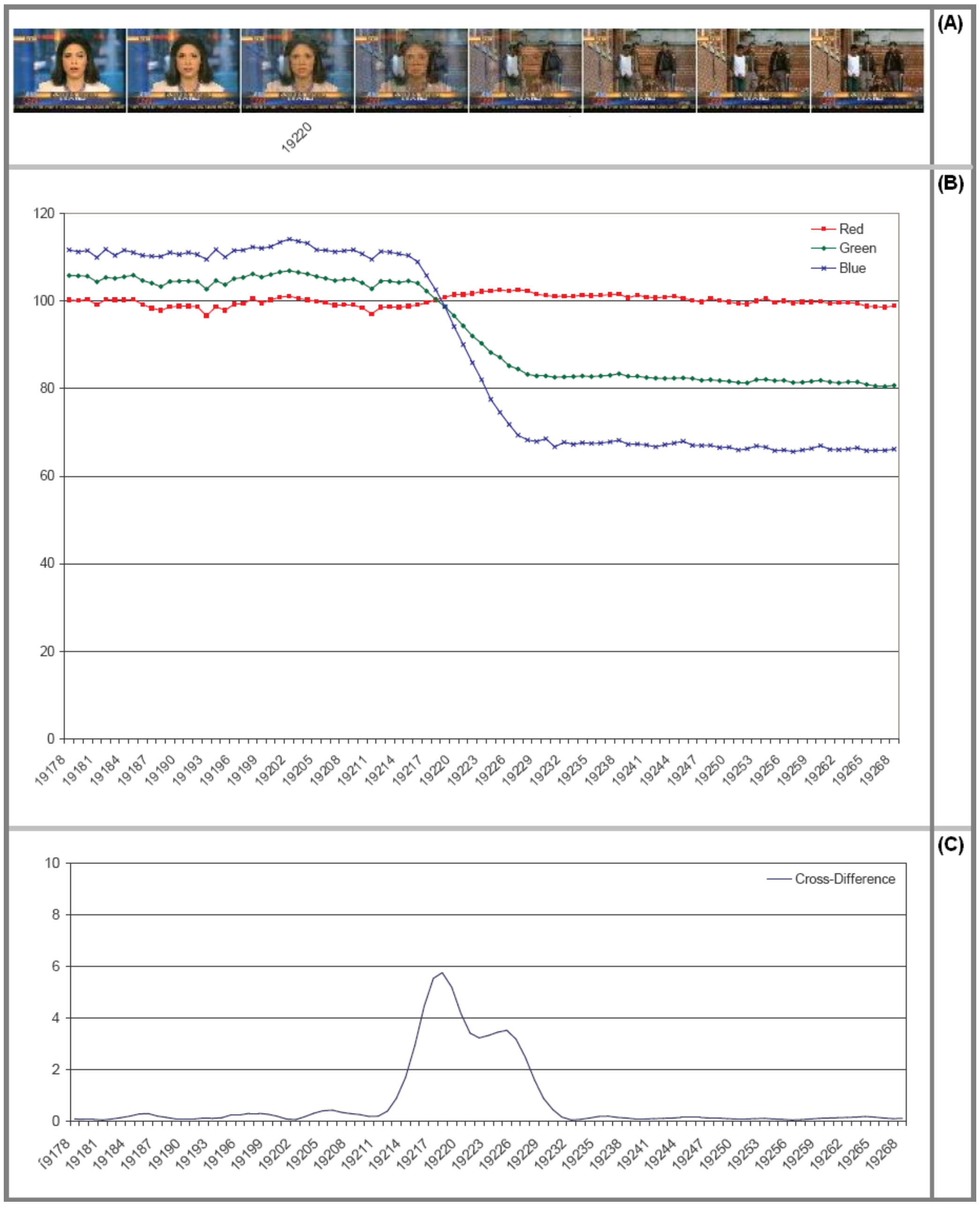


Fig. 2.9. Demonstrates a dissolve transition in a numerical representation format of a video stream. (A) Presents the video sequence. (B) Presents the numerical representation of the video sequence and (C) Presents the first order derivative or cross differences curve for the same transitions.

Figure 2.9 demonstrates a dissolve sequence along with its numerical representation. Similar charts and sequence are presented in figure 2.10 for a different dissolve. The difference between the two is in their numerical representation. The latter is less common and it is meant to demonstrate the reasons behind the difficulties with dissolve detection due to variations in its representation.

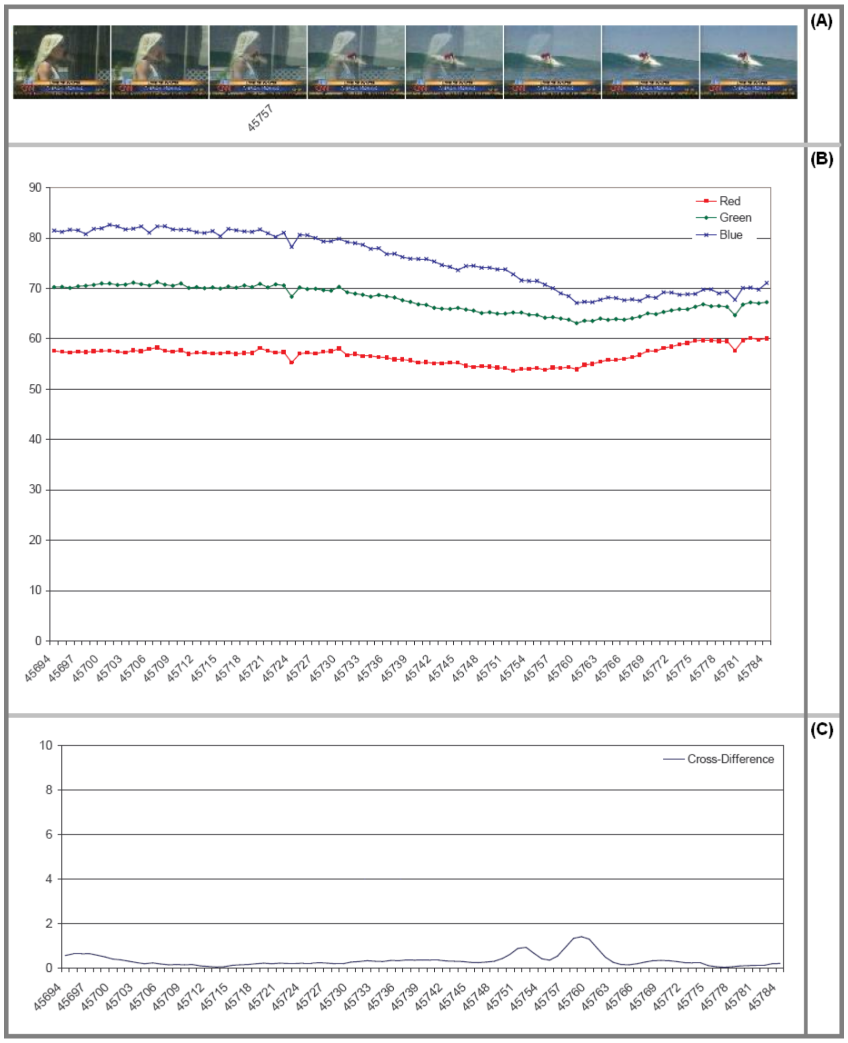


Fig. 2.10. Demonstrates a dissolve transition in numerical representation format of a video stream. This dissolve causes a completely different input stream behavior than the one in figure 2.9. (A) Presents the video sequence. (B) Presents the numerical representation of the video sequence and (C) Presents the first order derivative or cross differences curve for the same transitions.

Truong *et. al.* have algebraically shown that the first order difference of the variance curve changes linearly from a negative value to a positive value. Hence, the zero crossing sequences whose starting values are below and their endings values are above a specified threshold are points of dissolve transitions.

They also performed smoothing to cancel the effect of noise and motion while detecting the dissolves. The smoothing process causes the position of negative and positive peaks of the curve not to be coincident. To fix this problem the algorithm looks back and forth with a specific threshold in mind to find the correct starting and ending positions of dissolve transitions. As the last step, the algorithm looks at the variance curves while considering the fact that it will have a parabolic shape during a dissolve.

Wipe and Slide

Although wipe transition type is less common than cut, fade and dissolve transition types, there exist more varieties of wipes than other three transitions due to the fact that it is a specific type of spatial transition.

A spatial transition is a gradual pixel by pixel space-wise localized change in pixel intensities of one shot during which the pixels in the ending frames of that shot give their place to the corresponding pixels in the corresponding frames of the upcoming shot which will eventually lead to a frame within the second shot. Wipes are specific type of spatial transition since there should exist a specific order in which pixels of a frame in the preceding shot give their place to the pixels in a frame of the upcoming shot. This order yields a pattern in the video sequence which is known as wipe (in short, Wipe and slide transitions work by sliding a new shot into the existing shot [87]).

Depending on the order and generated pattern, wipes are divided into many categories some of which are listed and described in this section.

- **Vertical Wipe** is the most common type of wipe during which the transformation either initiates on the left hand side of transition starting frame and terminates on the right hand side of transition ending frame or vice versa. Figure 2.11 presents an example of such transition.
- **Horizontal Wipe** follows exactly the same pattern as vertical type except it initiates by changing the top pixels of transition starting frames and terminates by changing the bottom pixels of transition ending frames or vise versa.
- **Diagonal Wipe** follows exactly the same pattern as horizontal and vertical wipes with the exception that the transformation starts from one of the four corners and ends on the opposite side.
- **In-Out Wipe** is based on the same idea as previous wipes with exception that the change initiates or terminates at the center pixels of transition starting or ending frame respectively. If it initiates at the center pixels, it is called to be an in-out wipe and an out-in wipe otherwise. These two types also enjoy a diverse variety, some of which are presented in figures 2.12 and 2.13. Figure 2.12 presents a vertical in-out wipe combined with the graphical effect of an opening door. In this case the door wipe splits the existing shot in two and slides the pieces to each side, revealing the next shot behind the door. Figure 2.13 presents a circular in-out wipe.

- **Iris** is a type of wipe very similar to the one presented in figure 2.13 with exception that usually either the first or the second shot is a monochrome shot (it has the same relationship with wipe transition type as fade transition type has with dissolve transition type). Use of the iris transition was popular in silent films, but it is not used as often in modern filmmaking. This transition appears as a shape closing in on a scene, or opening outward to the screen's edges [87].

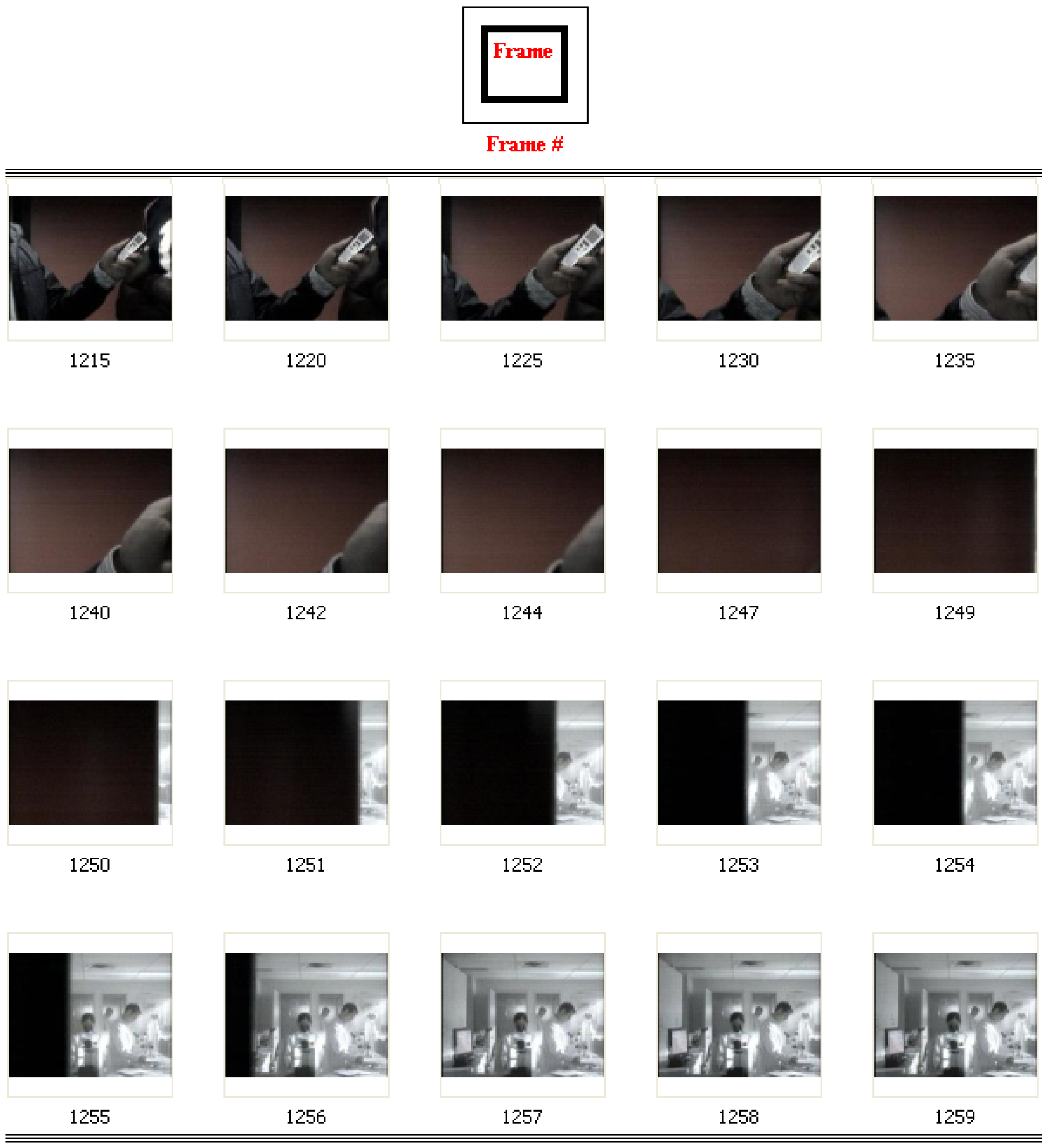


Fig. 2.11. Presents a vertical wipe.

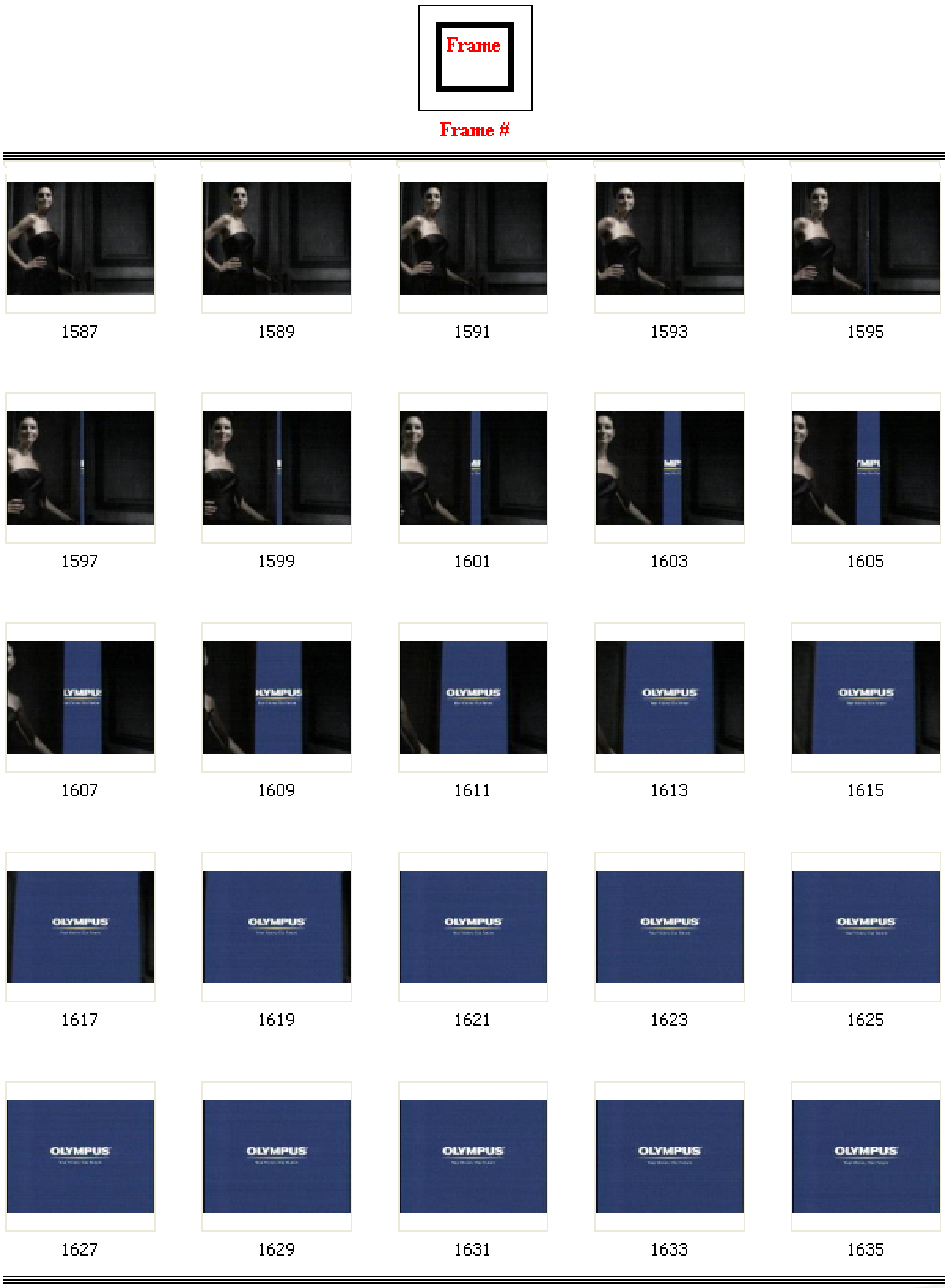


Fig. 2.12. Presents a special effect wipe.

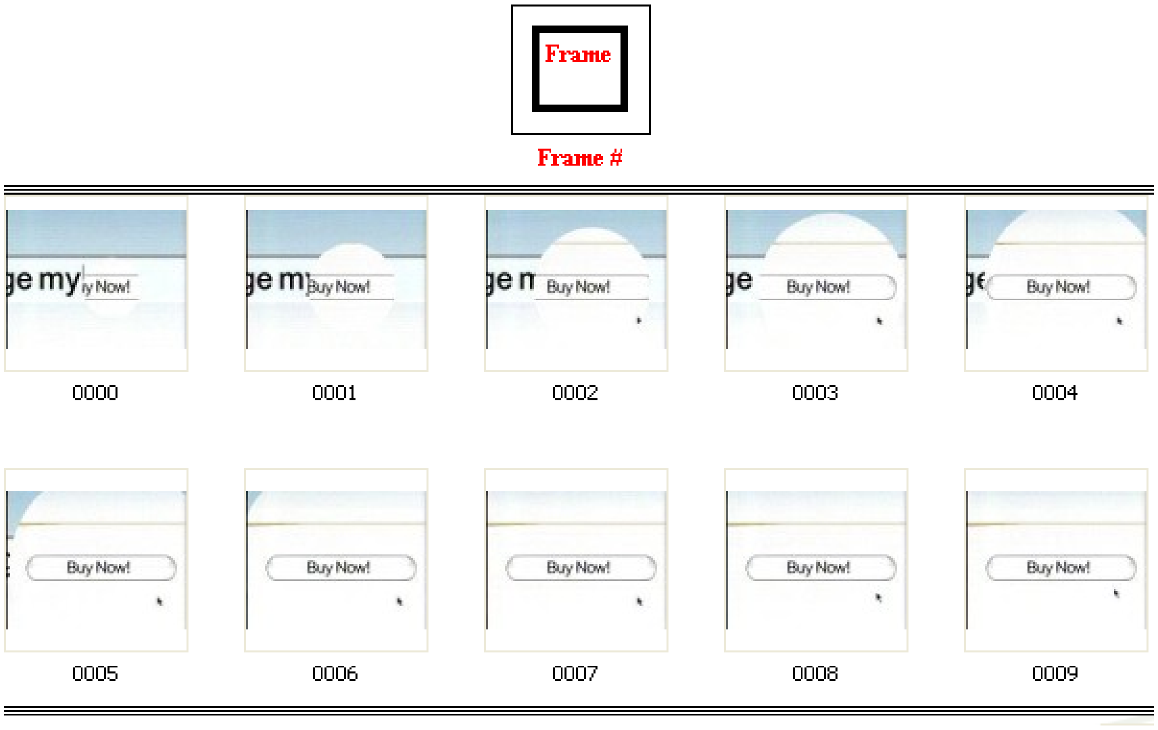


Fig. 2.13. Presents a circular wipe.

Due to the existence of numerous variations and the spatial characteristic of wipes, they cannot be presented through a specific mathematical model. Thus, other methods in video and image processing are utilized to identify these transitions. The most popular wipe detection techniques are edge and motion detection methods. Many researchers have used these two methods or have proposed other specialized and distinct algorithms for conducting this task. Among these research work are [30], [49], [74], [81], and [114].

Many wipes with complicated patterns can be labeled as graphical transitions depending on the specifications against which the detection process is carried out. Figure 2.14 presents an example of such wipe transition.

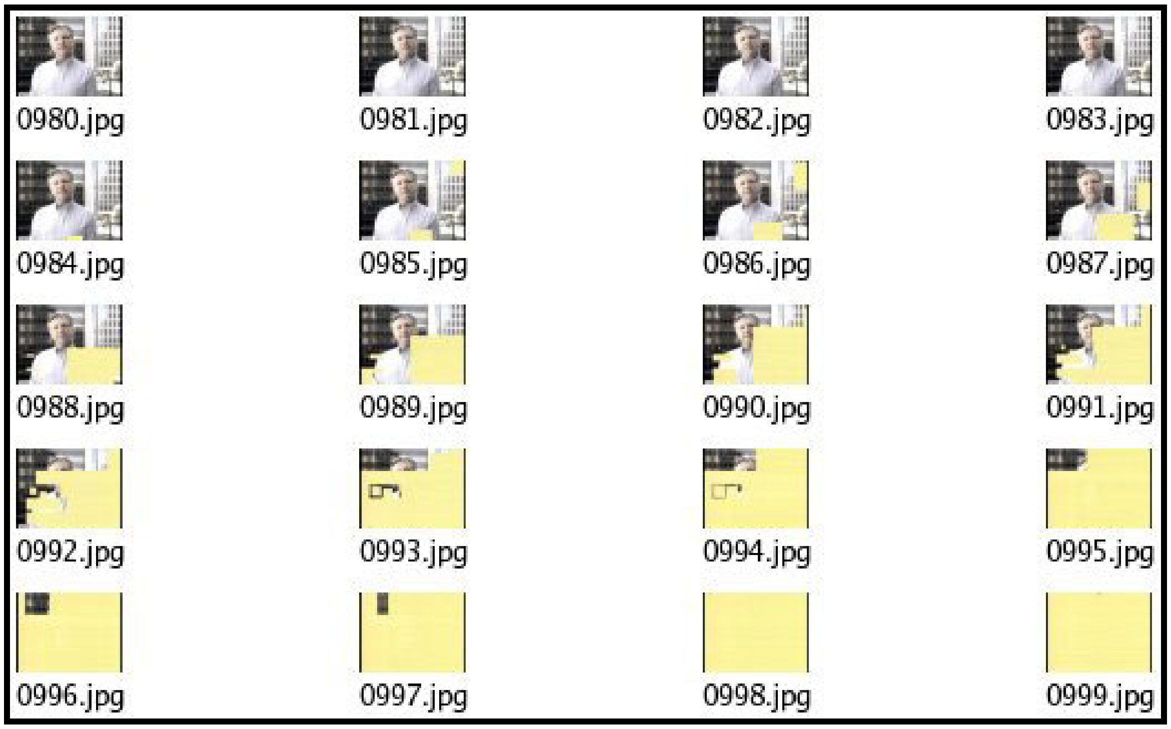


Fig. 2.14. Demonstrates a graphical transition example.

Graphical Transition

Similar but to a higher extent than wipes, a diverse and large set of transitions can be labeled as graphical transitions. This transition type can be defined as a type of transition which is compiled through the use of effects, computer graphics as well as other transitions in combination with the frames from the surrounding shots leading to a frame by frame transformation solution for converting one shot to the next.

The recent improvements of editing software in the past decade and attractiveness of these transitions have increased their use in many types of video especially in sport games, sport shows, commercials and some movies. As a result, their further study is necessary.

Mathematical model based techniques might prove very accurate for some variations, fail for others, or cannot be executed due to the lack of such model. Hence it is recommended that a learning algorithm ([63]) or a direct comparison method (chapter 3 and 4) to be used. Though, regardless of the technique, these transitions can prove difficult to detect and therefore to obtain the best detection quality, they should be considered in a case by case basis. Hence some of the main types of these graphical transitions are listed and described at this point.

- **Blinds** transitions are based on the real life blinds. Just as vertical and horizontal window blinds expose the outside world (or a different room) when their individual parts are twisted upward or sideways [87], transition blind will expose the frames in a new shot.
- **Morphing** is usually used in movies to transform one image (object) into another. Morphing or tweening is an animation technique that based on starting and ending shapes the algorithm creates the in-between frames while using mathematical equations to control the movement of key points in the parent shapes during the in-between frames. If this technique is used globally on all pixels of bordering frames of two adjacent shot, then it is a video shot transition. However, it is rarely used for this purpose.
- **Special Effect** is a diverse group of transitions. For example if the last frame on the first shot burn or shatter away into the next shot [87]. Other effects in this category include but are not limited to swirling, ripple effect, and rolling fog. These transitions also are not limited to 2D. Having 3D transition adds excitement to the video and therefore used in many types of videos such as documentary shows for kids to keep

them interested. For example the last frames of the first shot can fold or bend into a cube or sphere and role, bounce or fly off the screen, revealing the frames of the adjacent shot.

Graphical transitions are only limited to ones' imagination, making them the most difficult type of transition to detect. Besides existence of numerous variations, similarities of some of these transitions to object motions or other transitions and effects, make the detection task even more complicated to not only detect them but also to avoid detecting them as other transitions mistakenly. Figures 2.15 to 2.18 present some examples of graphical transitions.

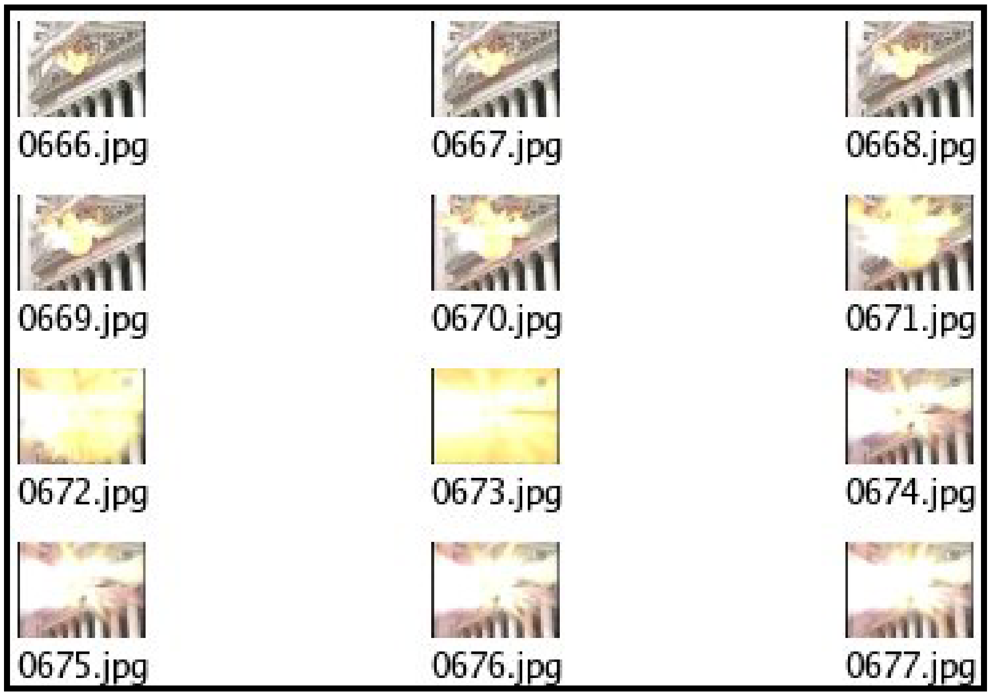


Fig. 2.15. Demonstrates a graphical transition example.

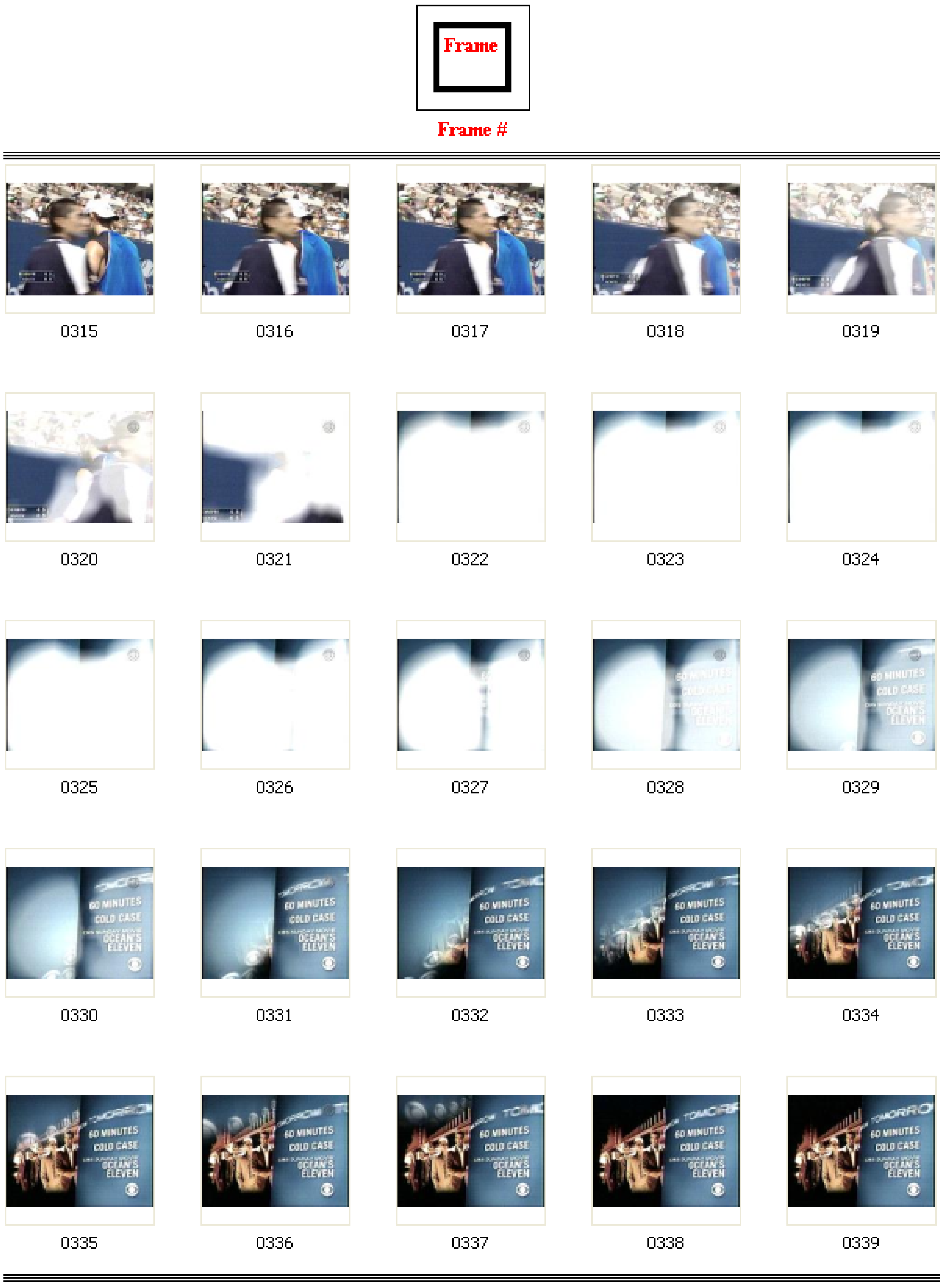


Fig. 2.16. Demonstrates a graphical transitions example.

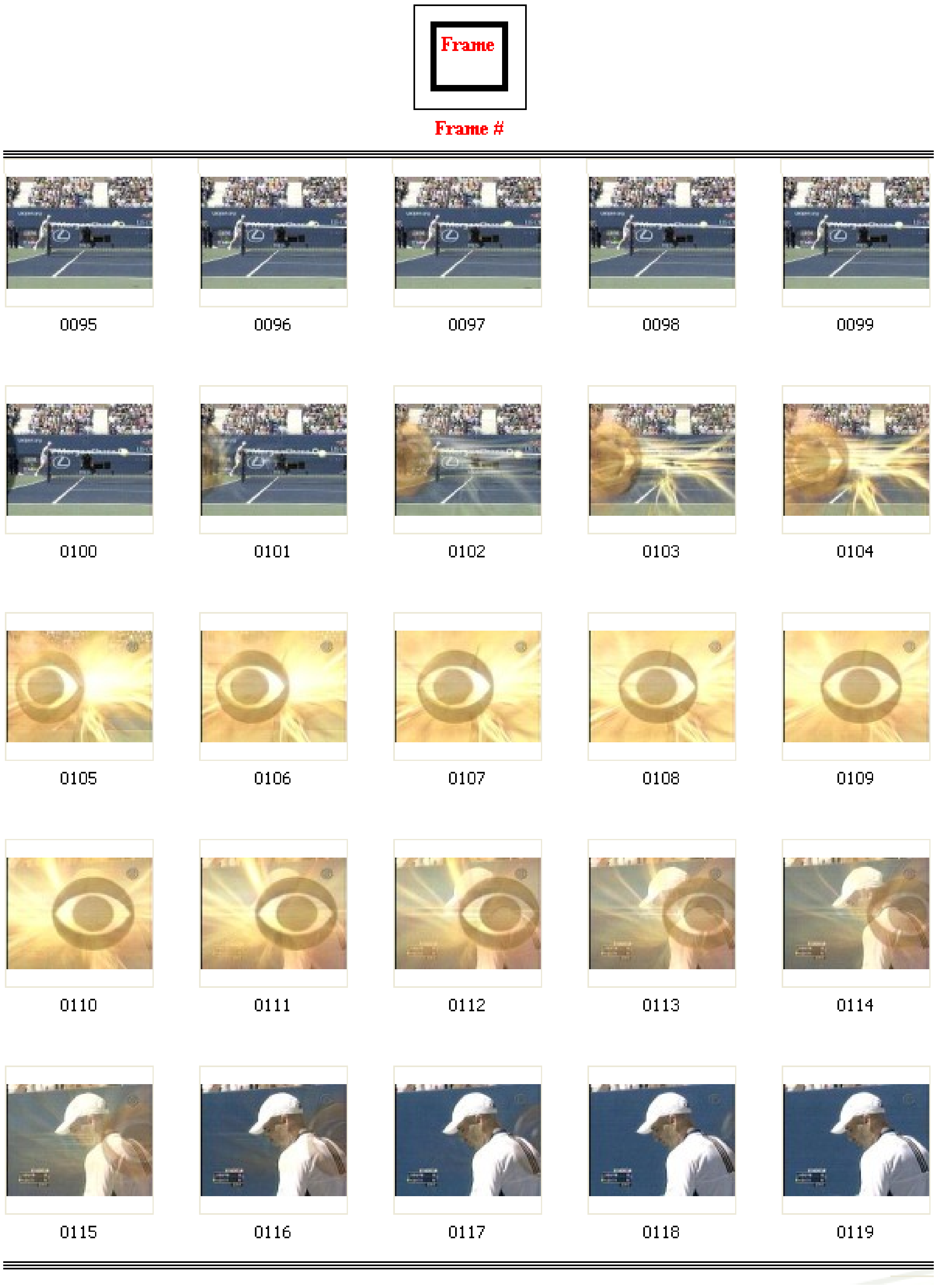


Fig. 2.17. Demonstrates a graphical transition example.

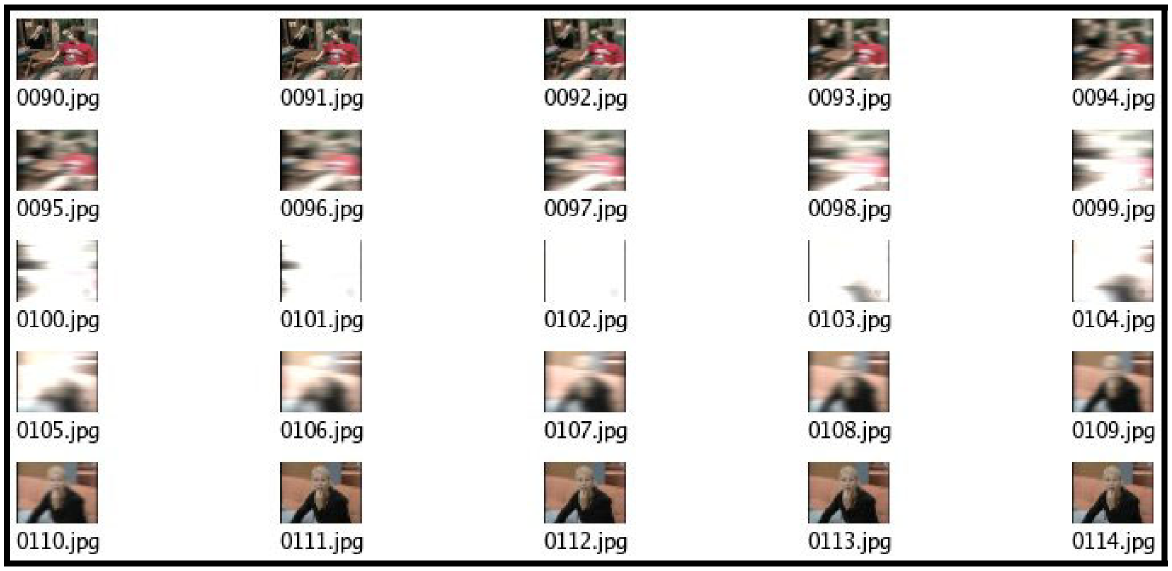


Fig. 2.18. Demonstrates a fade like graphical transition which is combined with fast camera movement.

2.3.3. Conclusion

In this section, we reviewed the different types of transitions providing the reader with mathematical models, examples (figures), list of subcategories, and detailed description of each when appropriate. It is important to learn about all these different types of transitions even if one does not attempt to detect all of them. This is due to the fact that some of these transitions such as graphical transitions can cause confusion for the temporal video segmentation algorithms which are meant to detect other types of transitions and eventually leading to a poor performance.

The next topic of interest is the way video frames are converted into numerical and statistical stream which is the most desirable format for processing through use of computers.

2.4. Representation

The first step in all temporal video segmentation algorithms is to extract different features of video, representing it in a numerical or statistical format. Another technique is also required to represent the measure of difference or the similarity metric between to image sequences. Some of the attributes and features used in this stage include but are not limited to color, shape, texture, luminance, edges, motions, and DCT coefficients. The future *detection* and *classification* stages are directly influenced by the *representation* stage and therefore extraction of appropriate features is of great importance.

2.4.1. Segmentation based on Pixel (Spatial) Differences

Pixel differences technique is one of the simplest schemes in video shot boundary detection research. The idea behind it is that the differences of pixel values in consecutive frames are low unless those frames are located in or on the boundaries of a transition such as cut, fade, or dissolve. These algorithms basically count number of the pixels that have a difference in value above a threshold. If this total is above a second threshold then a shot boundary is detected.

[49], [94], and [116] are among pixel differences based research work. Hampapur, *et. al.* [49] computed chromatic images. These images are obtained by dividing the difference pixel values in gray level of two frames by the pixel values of the second frames. Then they show that during dissolves and fades, the chromatic image takes on a reasonably constant value. They also used similar technique for detecting wipes. Regrettably, this technique is very sensitive to camera and object motion.

Zhang, *et. al.* [116] used similar technique for representation purposes. However as the first step they reduced the camera motion and other noise in the data by applying a 3x3 averaging filter. The algorithm was slow and hence they used a threshold tailored to the input sequence.

Shahraray [94] approached the problem by dividing the images into twelve regions and using a matching process similar to the one used to extract the motion vectors from an image pairs. Basically, for each region in the first image, the best match was found in the neighborhood of the same region in the next image. Then the weighted sum of the region differences provided the image difference measure. As the next step, a cumulative difference measure for consecutive image differences was measured which eventually was used to detect the gradual transitions in the video stream.

Even though pixel difference based techniques are the easiest detection methods to implement, they are not used widely since they are too sensitive to object and camera motion and can require a long processing time.

2.4.2. Segmentation based on Statistical Differences

The idea behind statistical differences schemes is very similar of that in pixels differences methods. In statistical differences the algorithm compares the statistical measures of pixels in different image regions. [16], [22], [51], [48], [58], [68], [73], [83], [100], and [109] are among the literature which include statistical based techniques.

Kasturi *et. al.* [58] developed an algorithm based on the mean and standard deviation of the gray levels in regions of the image. This method is claimed to be slow due to complexity of statistical formulas. It also introduces too many false positives in the end-results, perhaps due to the use of gray levels.

Miadowicz [73], use "color moments" in their analysis. Color moments are basically the mean, standard deviation, skew, and center of gravities for the three primary color channels, red, green and blue.

Similar method as [73] was used by Tahaghoghi, *et. al.* [100]. They used a moving window to analyze the statistical features of frames in each possible window. A cut was detected if there was a big change in data while moving from one frame to the next. Their cut classification method can be summarized as follow.

I. At each time instance, *t*, the difference between f(t) and f(x) for $\forall x \in [t-w, t+w]$ is calculated:

$$d(t,x) = |f(t) - f(x)| \quad \text{where} \quad x \in [t-w, t) \cup (t, t+w] \tag{2.3}$$

II. The values of *d(t, x)* are sorted in a decreasing order.

III. If there were no frames from the first half of the window in the first $\frac{(2 \cdot w + 1)}{2}$ of the sorted distance values (the top-ranked frames) then a cut had occurred at time instance, t.

Unlike Tahaghoghi's method, Liu *et. al.* [68] used eigenspace (introduced in [22]) method along with temporal statistics modeling. Eigenspace method has also been used in many other fields and applications such as data compression [48], feature extraction [109], and object recognition [83]. Liu detected shot boundaries by comparing the difference between the current frame and a model trained from multiple previous frames. Their final results showed improvements compared to so-called *direct differencing method*s (methods which use first order derivatives of video stream frames). The eigenspace used in this paper can also be used to model many other representation streams such histogram differences.

2.4.3. Segmentation based on Histogram Differences

Histograms are the most common method used for shot boundary detection research. Many variations of histogram based algorithms exist such as gray level and color histograms. These techniques use the images from two consecutive frames in a video stream; if the bin-wise difference between the generated histograms for each image is above a threshold, a shot boundary is assumed [16].

Many researchers have employed histograms in their analysis and experiments, [16], [63], [68], [75], [84], [99], and [105]. O'Toole, *et. al.* [84] used a "cosine similarity measure" for histogram comparison whereas Liu, *et. al.* [68] used histograms as the features for generating an eigenspace model of previous frames which was later used in video boundary shot detection. Swanberg, *et. al.* [99] used gray level histogram differences in regions. Similarly, Boreczky, *et. al.* [16] implemented three different variations of histogram-based techniques, simple histograms, region histograms, and running histograms based techniques. In simple histograms technique, a 64-bin gray-scale histogram over the entire frame was computed and the difference measure is the sum of the absolute bin-wise histogram differences. If the histogram difference between consecutive frames exceeds a pre-defined threshold then a shot boundary was declared.

The region histogram based technique is very similar to the simple histogram. The different is in use of two thresholds and different regions. In [16] each frame was divided into 16 blocks in a 4x4 pattern. Then they calculated a 64-bin gray-scale histogram for each region. Similar to the previous technique the histogram differences were computed for each region between consecutive frames. A shot boundary was declared if the number of region differences that exceeded the difference threshold was greater than the count threshold.

The final histogram method in [16] was the running histogram based technique. Similar algorithm was earlier used by Zhang, Kankanhalli, and Smoliar [116]. Similar to region histogram technique, two thresholds (high and low) are used here. First a 64-bin gray-scale histogram over each image is computed. If the difference between consecutive frames exceeded the high threshold, a cut was declared. On the other hand, start of a gradual transition was marked if the histogram difference exceeds the low threshold. From this point on if the running different exceed the high threshold then the end of gradual transitions are marked. Otherwise if it drops below the low threshold for more than two frames, they stopped computing running differences.

Unlike Boreczky's work in which gray scale histograms were used, Lienhart [63] used a color histogram based technique. Lienhart let $p_i(r,g,b)$ be the number of pixels of color red (*r*), green (*g*) and blue (*b*) in frame I_i which contain *N* pixels. Then each color component was discretized to 2^B different values, resulting in $r,g,b \in [0,2^B-1]$. Note usually *B* is set to 2 or 3 in order to reduce sensitivity to noise and slight light, object as well as view changes.

$$CHD_i = \frac{1}{N} \cdot \sum_{r=0}^{2^B-1} \sum_{g=0}^{2^B-1} \sum_{b=0}^{2^B-1} \left| p_i(r,g,b) - p_{i-1}(r,g,b) \right| \tag{2.4}$$

The equation 2.4 is the color histogram difference CHD_i between two color frame I_{i-1} and I_i.

Ueda, Miyatake, and Yoshizawa [105] also used a color histogram change rate to find shot boundaries. In [75] Nagasaka and Tanaka used compared various methods including statistics based on gray scale and color histograms in regions.

The IBM research TRECVID-2001 video retrieval system [98] is based on IBM *CueVideo* program. This program extracts sampled three-dimensional RGB color histograms from video frames. They also use an adaptive threshold and a state machine to detect transitions.

Histogram-based algorithms are easy to implement and have proved to be reliable method for detecting video shot boundary detection. Hence they are the most common method used. Though, over the years many new techniques have been developed which have superior detection rates than histogram based techniques in different circumstances.

2.4.4. Segmentation based on Edge Tracking

Using the edges of the objects and tracking them in a sequence of consecutive frames to detect transitions in those sequences is a common method, especially for detecting dissolves. [6], [63], [64], and [113] are among these works.

Edge Change Ratio

Lienhart [63] defines edge change ratio (ECR) as follow. Let σ_n be the number of edge pixels in frame *n*, X_n^{in} and X_{n-1}^{out} the number of entering and exiting edge pixels in frames n and n-1, respectively. Then equation 2.5 gives the edge change ratio ECR_n between frames *n-1* and n and it ranges from 0 to 1.

$$ECR_n = \max(\frac{X_n^{in}}{\sigma_n}, \frac{X_{n-1}^{out}}{\sigma_{n-1}}) \tag{2.5}$$

Zabih, *et. al.* [113] approached video shot boundary detection by constructing an edge detection technique and later they compared this algorithm with histogram and chromatic scaling techniques. Their algorithm compared the number and position of edges in the edge detected images. Then the percentage of entering and exiting edges from one frame to another was computed.

A big change in these percentage values indicated a shot boundary. Hard cuts are identified as isolated peaks. Dissolve and fades were recognized by looking at the relative values of the entering and exiting edge percentages; during fades in or fades out, the number of incoming or outgoing edges respectively pre-dominates; and during dissolve, initially the outgoing edges of the first shot protrude before the incoming edges of the second shot start to dominate the second half of a dissolve [63] (see figure 2.19). Finally they conclude by stating that their algorithm precision is higher than histogram-based techniques and it is less sensitive to motion than chromatic scaling.

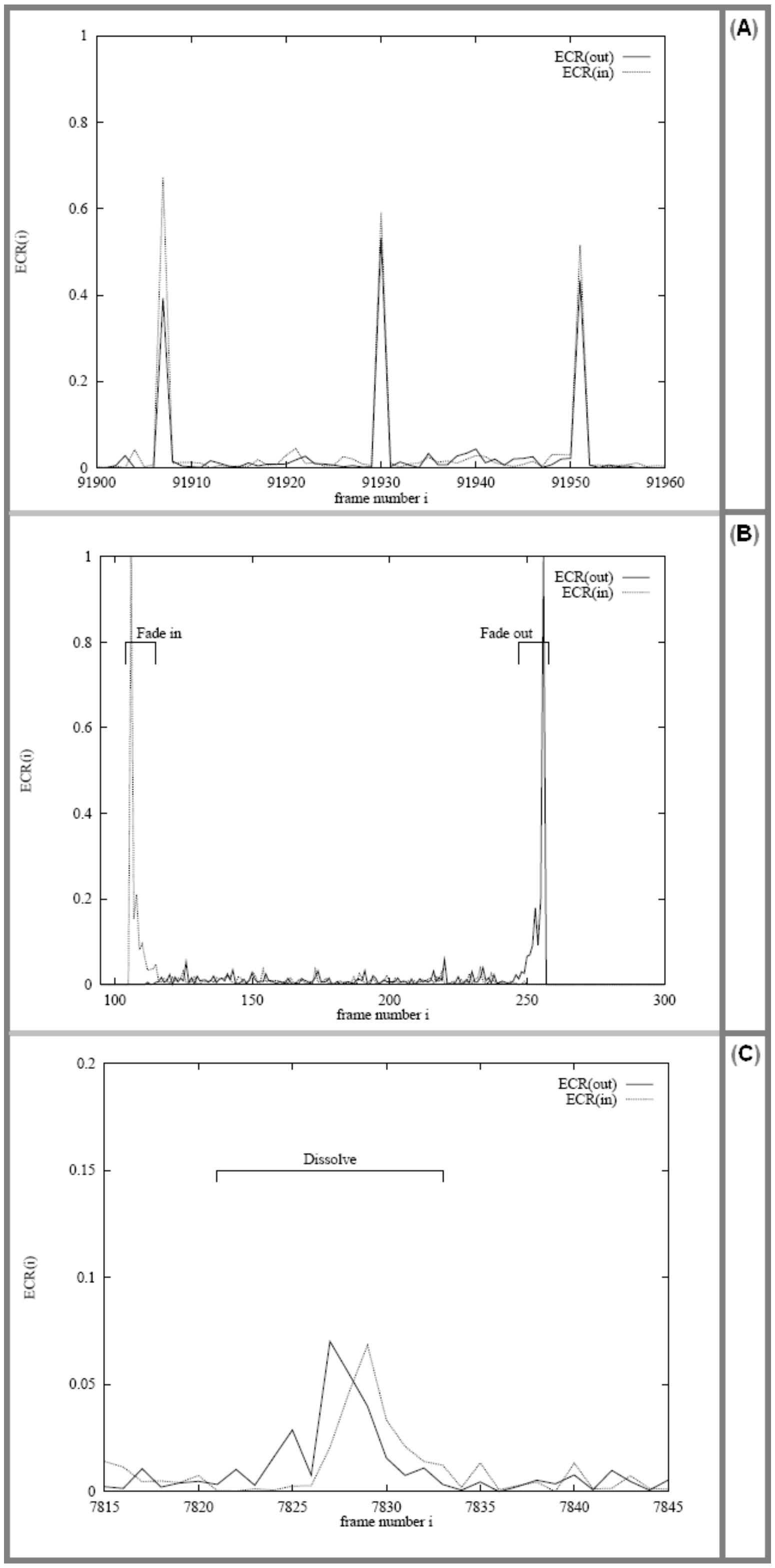


Fig. 2.19. A Typical *ECR* patterns for (A) hard cuts, (B) fades and (C) dissolves. (See Lienhart [63]).

Lienhart [63] built his algorithm on top of the technique used by Zabih et al. [63]. He used Canny edge detector [20] to calculate the edges. He added many details to the algorithm presented in [63] and hence, his end-results are of higher precision and recall. In a pre-processing step he smoothes the ECR time series, however, his algorithm ignores the points in ECR time series which exceeded a pre-defined threshold. Then by further analysis of time series he distinguishes between various transitions. Lastly, he ran post-process false hits detection algorithms.

Edge-based Contrast

Edge-based contrast is mainly used for dissolve detection. The idea behind edge-based contrast is to capture and emphasize the loss in contrast and/or sharpness to enable dissolve detection [63].

Lienhart implemented this technique in [63], and it was later used in the reliable dissolve detection system [64] and [66]. Similar to their edge change ratio algorithm, Canny edge detector [20] was used for calculating the edges. The proposed edge-based contrast algorithm had better hits and false hits rate for dissolve detection than the edge change ration algorithm of previous section.

Similar Techniques and Other Edge Tracking Methods

Rather than using edges in images, the next algorithm uses *Information Path* as the means for image comparison. Albanese, *et. al.* [6], introduced another method similar to the edge detection technique. Their algorithm focuses on identifying dissolve edits. The concept of *Animate Vision* (the visual biologic system capacity of quickly detecting interesting region of visual stimulus) [14], *scanPoth* [82], as well as Koch-Ullman algorithm [60], based on Itti-Koch model [56] and [59] were used in [6] to generate multiple connected saliency points in each frame (image), or as they call it *Information-Path* (IP).

In [57] the authors developed an *Information Path Matching algorithm* which returns the similarity measure between two images. This measure of similarity along side of a thresholding technique was used to detect various transitions boundaries.

The authors explain the dissolve transitions are harder to detect because the change in a dissolve is far more gradual than in an abrupt transition such as a cut. Hence *Information–Path* Matching algorithm can easily be adapted for cuts detection but not for dissolves detections.

2.4.5. Segmentation based on Motion Analysis

As mentioned before edge detection algorithm have proven reliable for detection of many types of transitions, especially dissolves. Motion detection algorithm can also be used for temporal segmentation of video.

Motion research can be used in scene and shot boundary detection algorithms, for grouping shots that are taken in the same site [79], for video objects clustering and retrieval [80], and for video indexing [37]. [116], [16], [63], [18], [19], [105], [85], and [77] employ motion characteristics of video to detect shot boundaries. Other works in this area includes visual motion model [2], [96], camera work analysis [57], texture modeling [69], video tomography [5], [102], epipolar plane image analysis [15], and periodicity analysis [70].

Video motion research has been an active area for the past decades. The research effort in this topic can be divided into two major groups, temporal motion segmentation [18], [77], and spatial motion segmentation [77]. Previous works demonstrate that motion analysis can improve the results of shot boundary detection algorithms considerably. For example, Lienhart *et. al.* [64] used the motion estimation algorithm that was suggested by Dufaux *et. al.* [31] to eliminate false positives that were caused by camera operations such as pan and zoom. Hence, in this section, besides temporal motion segmentation, other motion related topics are also briefly discussed.

Spatial Motion Segmentation

As explained in [77] temporal segmentation of image sequences expeditiously facilitates the motion annotation and content representation of a video, while the spatial decomposition of image sequences leads to a prominent way of reconstructing background panoramic images and computing foreground objects.

The work on latter group can be subcategorized to sequential motion estimation [13], [55] and simultaneous motion estimation [28], [92], [106]. In the former subcategory, images are analyzed and after computation of a dominant motion, the pixels related to that motion are removed. This process is repeated iteratively until a terminal condition is met. On the other hand, in simultaneous motion estimation, as its name applies, multiple motion models compete for the support of pixels and these pixels in turn influence the estimation of model parameters [77].

Yoon, DeMenthon, and Doermann [111] proposed two techniques for detecting critical events. Both methods work on compressed domain of MPEG video. To obtain the portions of video where important events were occurring, a “curve simplification” method was used. The second method was a “blob tracking” technique in which the trajectories of moving blobs along frames were obtained.

Temporal Motion Segmentation

Thus far, past spatial motion research was briefly discussed. This section reviews the earlier work on temporal motion segmentation, and how temporal motion analysis and other similar techniques used to detect video shots boundaries.

Bouthemy, *et. al.* [18] worked on a temporal segmentation method. They employed the affine motion parameters to describe dominant camera motions. By using these information, their algorithm detected the shot boundaries.

Ueda, *et. al.* [105] calculated motion vectors from block matching and used this to detect whether or not a shot was a zoom or a pan and they stored these information as a description of the cut. Similar work as [105] was performed by Zhang, *et. al.* [116].

Boreczky *et. al.* [16] developed a motion compensated pixel differences algorithm. In this algorithm three thresholds were used. They divided each frame into twelve blocks in a 4x3 pattern. Block matching with a 24 by 18 search window was used to generate a set of motion vectors and a set of block match values They used those values to drive a match value. A cut was defined if the value exceeded a threshold, and they use the low and high thresholds to detect the gradual transitions. The algorithm in [16] is based on the work done by Shahraray [94].

Bruno and Pellerin's approach [19] is based on linear prediction of motion wavelet coefficients which is calculated directly from two successive frames. The idea behind this concept is as follow. When transitions occur, the *brightness constancy assumption* (which is used to predict motion) fails (the *brightness constancy assumption* states that the image brightness is a simple deformation of the image at any given time).

Joly, *et. al.* [57] construct what they call *X-ray* image (derived by projecting the intensities along given lines which result in xt- or yt-plane in the image sequence) then they use Hough transform to search for a particular patterns in them.

The MPEG-1 bit stream may also be directly exploited for camera motion characterization. For more information refer to the *Segmentation using Compressed Data* section or to [85] and [110].

2.4.6. Segmentation based on Transforms and Frequency Domains

This section presents the literature review on temporal video segmentation research based on different transforms and frequency domains. Depending on transforms or frequency domains used these works are divided into individual sections.

- **Fast Fourier Transform (FFT) Coefficients.** Fourier was first published in 1822 by Jeane Baptiste Joseph Fourier in France [11]. FFT uses both complex and real numbers as well as both sin and cosine waves to represent an image. Miene *et al.* [74] use two metrics in spatial domains and one in frequency domain by using FFT coefficients calculated from a grayscale version of the frames (images) for the basis of their comparisons. They calculated the sum of the real-part of lower frequencies and the sum of the appropriate imaginary parts. The final value used was the sum of absolute differences of the real- and the imaginary- part for each consecutive frames pair.

- **Discrete Cosine Transforms (DCT) Coefficients.** Similar to FFT, in DCT both complex and real numbers can be used. Cooper *et. al* [27] calculated the low-order DCT coefficients for each frame. Then they evaluated the similarity of each frame to the surrounding frames. A cut was identified if the frames before and after the current time instance had a high similarity to past and future frames respectively but low similarity across the boundary.
- **Other Work in Frequency Domain.** Porter *et al.* [88] developed a system for BBC Wildvision; as they mentioned two predominant features of wildlife films are that they contain potentially significant object motion and that a new shot is often taken of the same scene but from a different angle. They argue that these types of scenes have similar intensity distribution. Hence, histogram based methods are not a good choice even though that they can be robust in presence of motion. As a result they pursued a motion-based algorithm. Their technique can be summed up in two steps: they first, calculate the normalized correlation between blocks of 32 X 32 in two adjacent frames. Then they locate the correlation coefficient with the largest magnitude. The first step could not be done in spatial domain since it is prohibitively expensive. Hence, they performed it in *frequency domain*.

2.4.7. Segmentation based on Compressed Data

The methods discussed so far are mainly based on uncompressed data. Similar techniques can be used on compressed data. These topics are discussed in more details later on in this section.

MPEG is a defined international standard for a compressed video bit-rate. It includes many powerful and efficient compression algorithms, such as different motion compensation modes. The compression task is perform through elimination of temporal and spatial redundancies within the image [72].

The use of compressed data is advantageous since video databases are often available in compressed formats. Hence direct processing of the MPEG bit-stream is a major stake [18]. On the other hand, the disadvantage is that using encoded features directly can result in lower accuracy. Hence, many other published papers such as [33] aimed at reducing error rates in detections while using MPEG video. Among other research work in this field are [25], [27], [34], [10], [72], and [95].

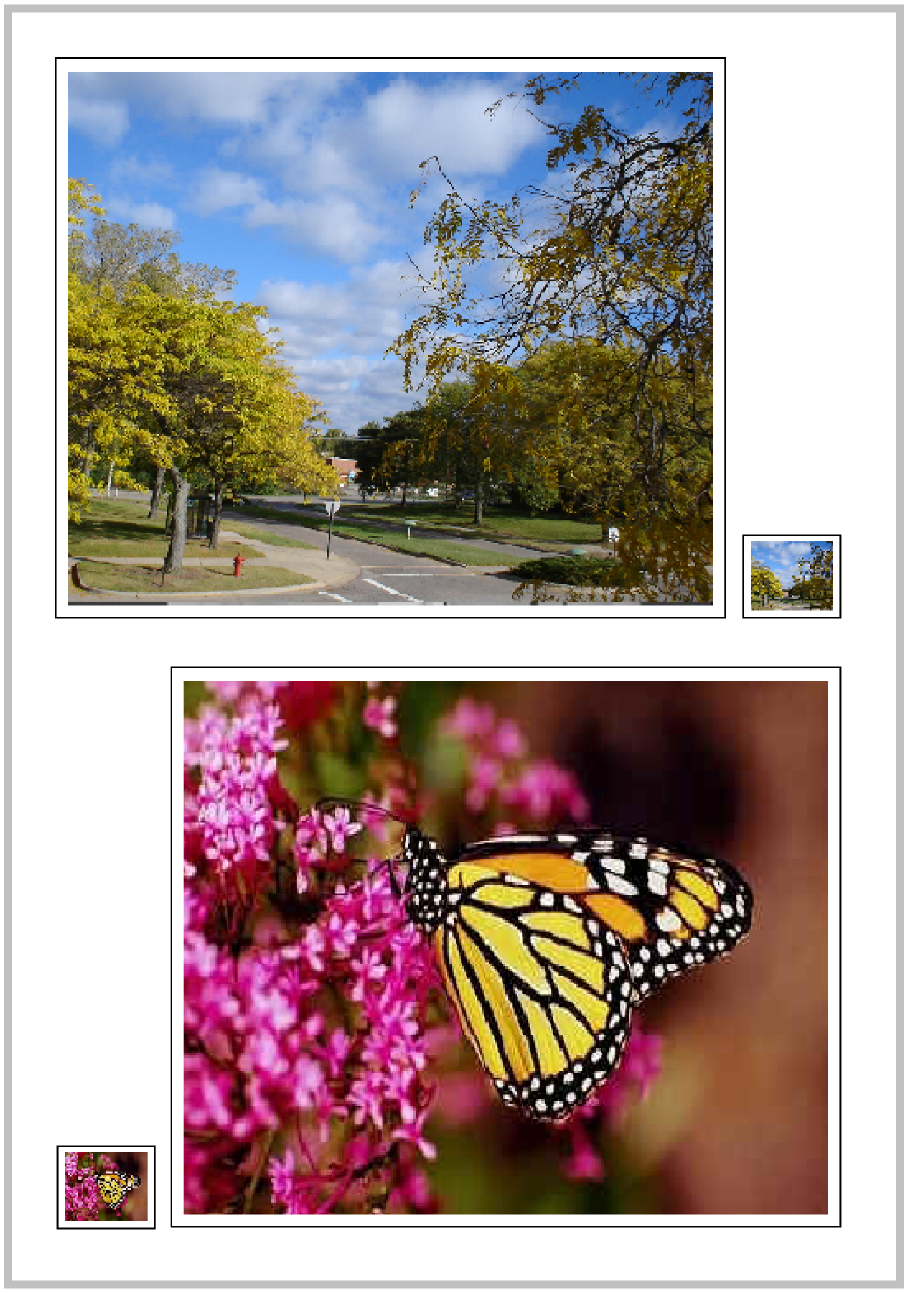


Fig. 2.20. Illustrates the original frames (large images) and their respective DC frames.

These methods are based on the features that are available in the compressed format such as DCT coefficients, macro blocks, or motion vectors. Figure 2.20 shows two examples of regular frames and respective DC frames. Since in MPEG format DCTs are generated by dividing the images (frames) into an 8×8 grid, the DC image is $\frac{1}{8}$ of the original image.

The histogram based frame difference measurements in MPEG videos can vary systematically even for identical frame transition types depending on their relative position within a group of pictures. Ewerth, *et. al.* [34] proposed a method for solving this problem. They analyzed the characteristic of histogram based difference measurements of MPEG DC frames.

Meng *et. al* [72], proposed an algorithm mainly for scene change detection. Among boundary shots they mainly focused on dissolves. The detection is carried out with a partial decoding of the compressed bit-stream.

In [85], Patel and Sethi computed the image intensity histograms using the DC component of DCT related to I-frames. Similar work was performed by Yeo and Liu in [110] with exception that they used P- and B-frames to compute the image intensity histogram from DCT. In these methods the MPEG-1 bit stream was used directly for camera motion characterization, using motion vectors related to P- and B-frames.

The problem with these methods is that they are not resilient to presence of mobile objects of significant size. The problems with these technique is that as is stated in [16], the block matching performed as part of MPEG encoding selects vectors based on compression efficiency and thus often selects inappropriate vectors for image processing. Hence, these approaches are not resilient to the presence of mobile objects of significant size.

2.4.8. Segmentation based on Audio Track

Audio track provides another valuable source for input data. [91] used the audio track to categorize speech, silence, music and noise. [76] used audio track for speakers identification which are very interesting. These research can be used for temporal segmentation of video. [17], [44] and [107] are among research which have used audio track in one way or another. Audio track was not used in our research and hence it is left to the readers to investigate this topic further.

2.4.9. Other Methods

So far in this section the more common video representation techniques were discussed. This section presents the less common and more specialized representation techniques used in previous temporal segmentation research.

- **Segmentation based on Hausdorff Distance.** Zabih *et. al* [114] uses a method based on Hausdroff distance and edge detection algorithm results. Further investigation is left to the reader since this topic is too specific and is not used in our research.
- **Segmentation based on Self- and Cross-Similarity.** Cooper *et. al.* [26] introduced a new method for scene boundary detection. Unlike many existing approaches they avoid using histograms and rather they use self-similarity and cross-similarity among various frames of the video to detect the transitions between videos. Authors argue since they do not use histogram information their system makes minimal assumptions which is an essential requirement for numerous applications.

  The work in [26] can be summarized in a few sentences as follow: First they compute cosine similarity matrix to be used to compute correlation along diagonal of similarity matrix with Gaussian checkerboard kernel. Then they locate peaks via analysis of the first and second differences of the output signal and finally label peaks as scene boundaries.

2.4.10. Conclusion

In this section a diverse set of techniques used for video representation was discussed. Video representation is the initial stage for any video temporal segmentation algorithm and it directly influences the methods used in the future stages, namely detection and classification stages. These stages are described in more details in the subsequent sections.

2.5. Detection

Detection is the next major step in temporal video segmentation algorithms. Detection can be described as the method for discovering boundary shots – usually regions with significantly distinctive data – in the *video representation* or in the *measure of difference* curve of two image sequences. The most common detection method is through applying different types of thresholds. As it is mentioned in [84], two conflicting points exist for accurately segmenting videos using thresholds and they are as follow:

- The need to prevent detection of false shot boundaries, by setting a sufficiently high or low (depending on the data) threshold level so as to insulate the detector from noise.
- The need to detect subtle shot transitions such as dissolves, by making the detector sensitive enough to recognize gradual transformations.

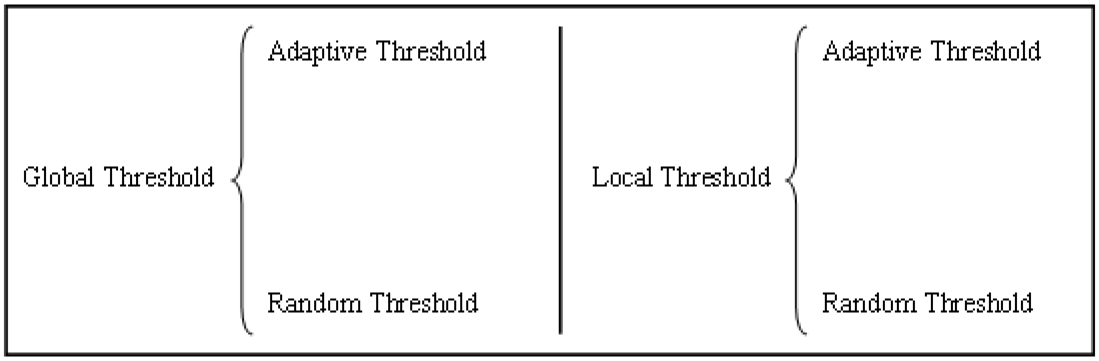


Fig. 2.21. Illustrates the four possible types of thresholds.

Threshold can be categorized into four groups. As figure 2.21 illustrates, thresholds are either global or local, and each of these two groups can be divided to adaptive or static thresholds.

- **Global Threshold** refers to the limitation applied to the input stream as a whole. If these limitations are applied through consideration of statistics extracted from the input data, then it is said to be *adapted* to the input data. Otherwise it is said to be a *random* threshold. Random thresholds should be avoided if possible. Their use becomes necessary only if a threshold must be used and if a prior knowledge of these statistics is not in hand. Random thresholds are also called *static* thresholds and global thresholds are also called *fix* threshold.
- **Local Threshold** refers to the limitation applied to a smaller segment of input stream. If these localized limitations are applied through consideration of statistics extracted from the input data for the corresponding segments, then it is said to be *adapted* to that segment of input data. Otherwise it is said to be a *random* threshold. And as mentioned before, random thresholds should be avoided if possible.

Local and global adaptive thresholds are discussed in further detail in the subsequent sections.

2.5.1. Global Adaptive Threshold

Global thresholding is among the easiest method for distinguishing between the real transitions and potential false positives. Yet since it is an overall limit forced over the entire video data (or a very large portion of video), it is error prone.

Earlier in this chapter, past literature was reviewed from *representation* perspective. The video representation or measure of difference between to image sequences is used to detect the shot boundaries. Figure 2.22 (b) is the graph of dissimilarity measure against time. The spikes represent the shot boundaries (mainly cuts) while the other part of the video stream represents normal frame pairs or regions of low activity.

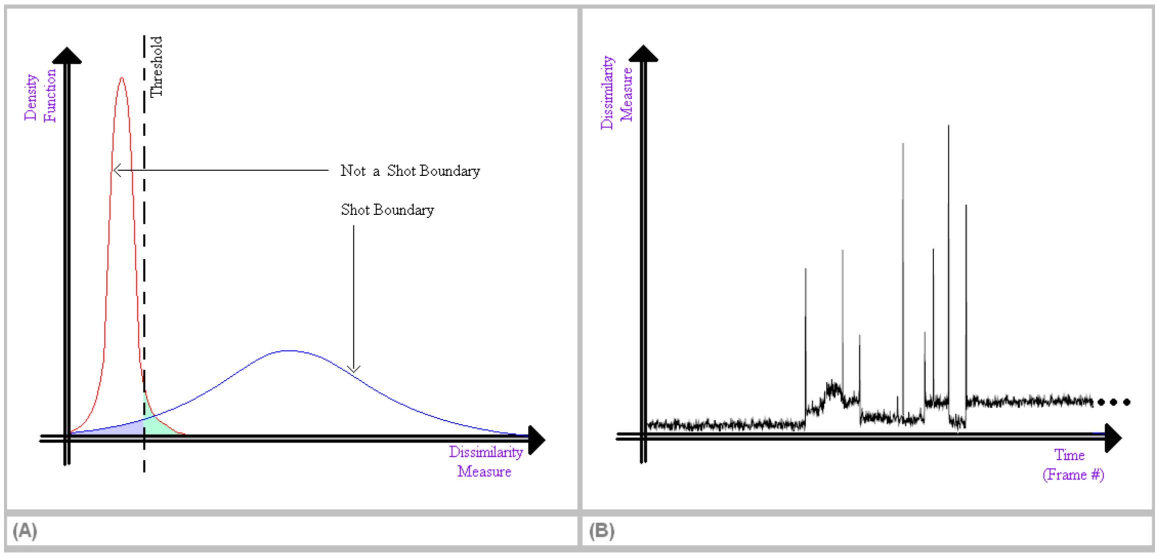


Fig. 2.22. (A) Presents the distribution curves for shot boundary and not shot boundary sets. (B) Presents the graph of dissimilarity measure against time.

Figure 2.22 (a) illustrates the density function against dissimilarity measure. As can be seen in this chart, *not a shot boundary* curve is narrow (occupying the low dissimilarity measures section) and stretched vertically (occurs more often), whereas the *shot boundary* curve is wider and stretched horizontally. As this points out it is impossible to divide *shot boundaries* and *not shot boundaries* into two distinct well defined groups using a global threshold. However, the difference between a well defined and slightly off threshold can be drastic. This is mainly caused because the mean of *not a shot boundary* curve is very close to the threshold.

Global thresholds are mainly used for abrupt transitions detection, though a single fix threshold was used in [84], [114] to detect all types of transitions. Some other algorithms use combination of thresholds to detect gradual transitions in addition to abrupt ones. A global threshold was also used in by O'Toole and *et al.* [84] to detect shot boundaries; their experimental results show that the detection rate can vary by up to 20% even for the same type of video content if such threshold is used.

The global thresholding technique cannot be improved too much since it is a rather simple method, however prior studies of data and other modifications to the algorithm can prove effective. The first technique is to perform prior analysis of the video content and based on that utilize multiple thresholds in our experimentation. This technique was also used by OToole, *et. al.* [84]. Another method for improving adaptive global thresholds, is using more than one dissimilarity metric (measure of difference). The two metrics should not be too similar to each other or this method will not add too much value to the detection algorithm and it will rather cause unnecessary complications. Figure 2.23 demonstrate use of two distinct features.

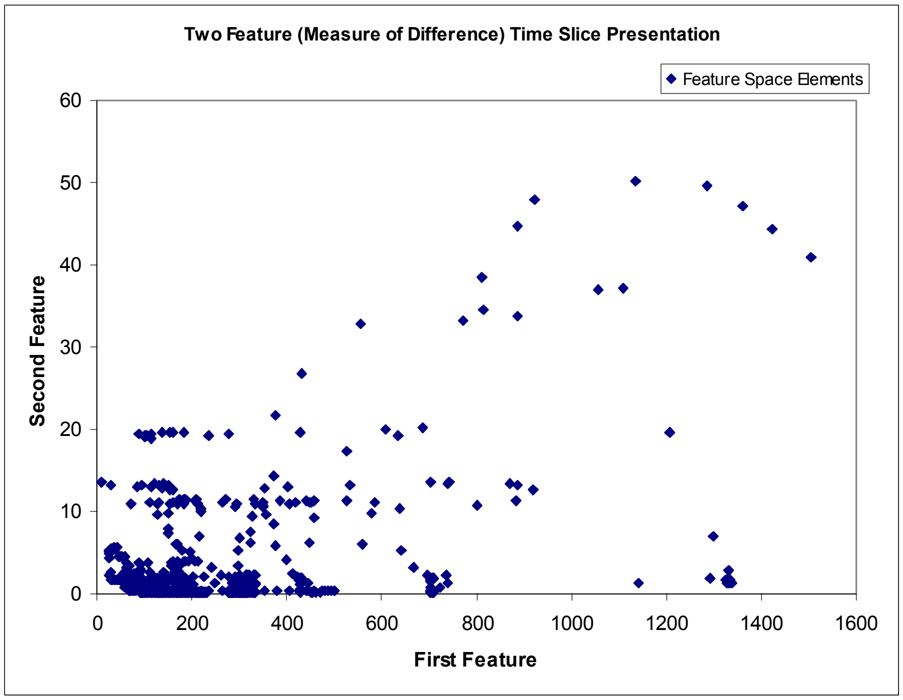


Fig. 2.23. Demonstrates features space for two features extracted from the video.

Various experiments point out the fact that having a general upper or lower limit on data is one of the main reasons behind higher than expected false positive and miss rates. Yusoff et al. [112] point out that global or so-called "single decision" threshold can be consistently grossly over- or underestimated when applied to video material with distinctive characteristics, such as sports events or cartoons. Lienhart takes this one step farther by stating that it is impossible to find a single global threshold that works with all kinds of video material, henceforth, global thresholds should be avoided [65]. Consequently, much research has focused on usage of adaptive local thresholds.

2.5.2. Local Adaptive Threshold

Use of a local adaptive threshold will help to overcome the inaccuracies of the global threshold. These thresholds analyze a smaller data range (data within a sliding window). Hence better results are attainable since each proposed threshold is built around and adapted to the local data. Figure 2.24 demonstrate a very localized adaptive threshold.

Most variations of local adaptive thresholds are based on the following methods. At each time instance, *t*, the local neighborhood is marked by a temporal sliding window of size *2w+1* centered on *t*. Only the data related to the current window is analyzed at each time instance. An abrupt transition, and starting or ending of a gradual transition are identified if the statistics at points in question do not follow a certain set of conditions. [74], [98], [100], [103], [110], and [112] follow this logic.

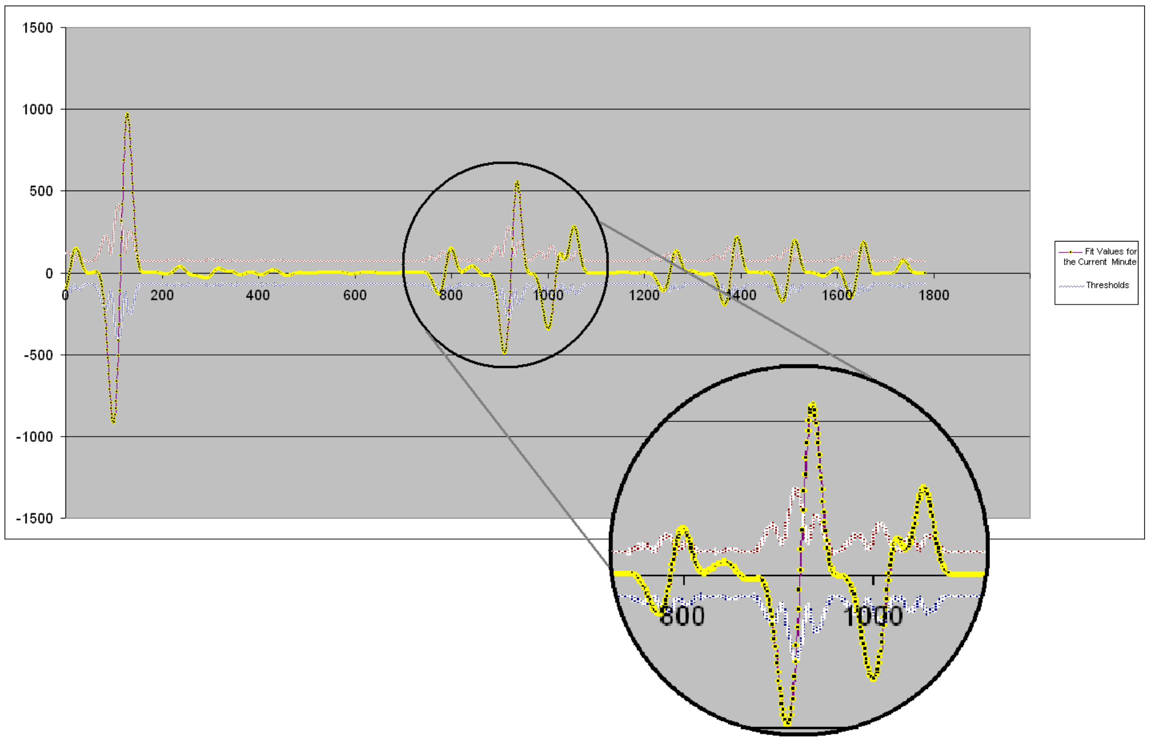


Fig. 2.24. Illustrates use of two localized adaptive threshold (white curves) against input stream (yellow curve).

Miene et al. [74] proposed a simple adaptive threshold in which they find the maximum value of frames dissimilarity measurements and at each stage they define the threshold to be a certain percentage of maximum number.

$$\text{Threshold} = \text{Max} \cdot \text{K}\,\% \tag{2.6}$$

They use dissimilarity measurement and threshold along side of a Fast Fourier Transform (FFT) and YUV features to detect cuts and wipes.

In [110] a local thresholding method was proposed. The authors identify the transitions in a sliding window if all the following conditions are met:

I. The value at time instant t is the maximum value inside the window.

$$f(t) \geq f(x) \quad \text{where} \quad t, \forall x \in [t-w, t+w] \tag{2.7}$$

II. The maximum value is greater than the second largest value in the window by a factor of *threshold$_2$*.

$$f(t) \geq threshold_1 \cdot f(t_2) \geq f(x) \quad \text{where} \quad t, t_2, \forall x \in [t-w, t+w] \tag{2.8}$$

Truong et al. [103] uses a method similar to [110] with additional conditions in the second half. In their algorithm a transition is declared if the following conditions hold.

I. The value at time instant t is the maximum value inside the window.

$$f(t) \geq f(x) \quad \text{where} \quad t, \forall x \in [t-w, t+w] \tag{2.9}$$

II. The maximum value is greater than the second largest value in the window, *f(t$_2$)* by a factor of *threshold$_2$*.

$$f(t) \geq threshold_1 \times f(t_2) \geq f(x) \qquad (2.10)$$

$$(\text{i.e. } ratio_1 = \frac{f(t)}{f(t_2)} \geq threshold_1)$$

$$\text{where} \quad t, t_2, \forall x \in [t-w, t+w]$$

$$ratio_2 = \frac{f(t)+c}{g(t)+c} \geq threshold_2 \qquad (2.11)$$

where *g(t)* is the mean of all the other frames of the window, not considering the frame in the time instance t. If *ratio$_2$* exceeds the *threshold$_2$* in value then a cut is detected, whereby the condition II (a) will also be true.

The constant c is added to the ratio calculation in order to deal with freeze frames. These frames make the determination of a good adaptive threshold difficult since they cause the discontinuity to be near zero.

Tahaghoghi et al. [100] proposed a different method for adaptive thresholding. Their algorithm is as follow.

I. At each time instance, *t*, the difference between *f(t)* and *f(x)* for $\forall x \in [t-w, t+w]$ is calculated:

$$d(t, x) = | f(t) - f(x) | \quad \text{where} \quad x \in [t-w, t) \cup (t, t+w] \qquad (2.12)$$

II. The values of *d(t, x)* are sorted in a decreasing order.

III. If there were no frames from the first half of the window in the first $\frac{(2 \cdot w+1)}{2}$ of the sorted distance values (the top-ranked frames) then a cut had occurred at time instance, *t*.

Yusoff et al. [112] described three models for setting a threshold with regard to dissimilarity-density chart in the previous, global threshold, section. In their paper, they suggest a method that dynamically changes the threshold based on dissimilarity measures from the previous and next few frames. Since the threshold will be near the mean of *not a shot boundary* curve, it is important to use static information related to that curve in calculation of the threshold. The mean of the curve, M_{nsb}, and its variance, V_{nsb}, used in [112] to adaptively set the detection threshold. The *shot boundary* curve is assumed to be stationary.

Similar to the previous adaptive thresholds, they use a sliding window as well. To estimate the threshold and the curves dynamically at each time instance, t, they use only the statistics derived from the $window_t$. This method along side various assumptions lead to various models for defining the threshold.

- **Constant variance model** assumes the distributions are unimodal, *shot boundary* curve values in the intersection of two curves density functions can be ignored (since mean of *not a shot boundary* curve varies over a small enough range and the *shot boundary* curve is sufficiently wide), and finally, the distributions are all stationary (except for *not a shot boundary* curve mean).

In other words, they use the mean of *not a shot boundary* curve and add an offset, k_1, to it to obtain the value for the threshold.

$$\textit{Threshold} = M_{nsb} + k_1 \tag{2.13}$$

They determine k_1 value through experimentation of training set (video material and truth data).

- **Proportional variance model** is similar to constant variance model except V_{nsb} is assumed to vary with M_{nsb}^2.

$$\textit{Threshold} = k_2 . (M_{nsb}) \tag{2.14}$$

Similar to k_1, k_2 is also determined through experiment.

- **The Dugad model** is the last method they used. It is based on the work done in [32] which is described next.

The authors also performed experiments to discover the effects of window size on different algorithms used. Although the trend is not universal, increasing window size tends to increase the accuracy of the shot detection, and then will eventually decreases after a specific point [112]. This experiment showed that histogram comparison and likelihood ratio work best with a small window size while motion estimation method had unpredictable performance but it improved for larger window sizes.

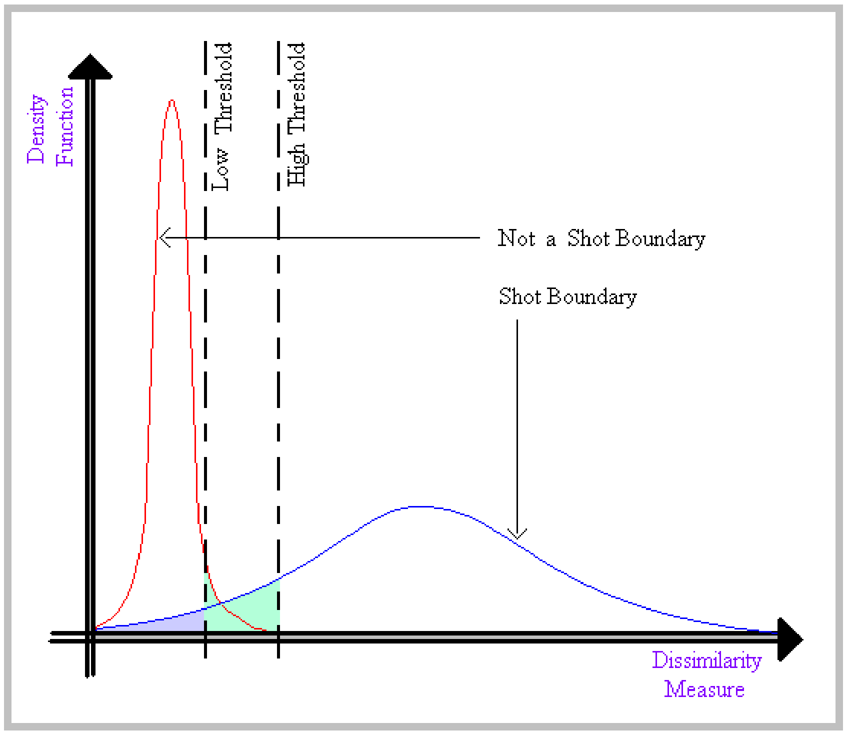


Fig. 2.25. Demonstrates desirable values for high and low thresholds.

Dugad et al. [32] method also can be explained by using the dissimilarity-density chart (figure 2.25). To dynamically estimate a threshold they use a sliding window with time instance, *t*, in the middle of the window. The means and standard deviation on the left and right of *t* are calculated.

$$Threshold = M_{nsb} + k_3 \cdot \sqrt{V_{nsb}} \tag{2.15}$$

During their experiment, they estimated value of 3 for a *low threshold* and 5 for a *high threshold* to be used as k3. If the dissimilarity measurement for *t* was greater than *high threshold* then a shot change was declared, else if it was greater than *low threshold* then they used a likelihood ratio to investigate the matter further. If dissimilarity measure was less than *low threshold* then shot boundary was not declared.

Past experiments shown, transition detection and classification accuracy improves if adaptive localized thresholds are used instead of previously explained global thresholds. However, adaptive thresholds also introduce limits that cannot be used universally for all types of input video stream and transitions.

2.5.3. No Thresholds

Based on statistical behavioral studies of frame differences, [47] points out that a threshold that is appropriate for one type of video data may not yield acceptable results for another type. Hence, even employing adaptive thresholds can result in an undesirable margin of error. Therefore, over the years, new detection and classification techniques which are threshold-free, parameter-less or model-free have been proposed. Most of them do not have detection stage and they only focus on representation and classification stages. The use of threshold becomes unnecessary if a learning, clustering, or similar methods are used in *classification* stage. These techniques are discussed in *Classification* section.

2.5.4. Conclusion

As discussed, identifying the right thresholds is very crucial issue. Algorithms with well-behaved threshold definitions or related algorithms will have a much greater hit rate over miss rate ratio and their false positive count is much lower than the other algorithms. Hence, in this section many detection methods were discussed in detail. The next major step in temporal video segmentation is *classification* which is also the topic of next section.

2.6. Classification

Classification can be described as the method for labeling or categorizing the boundary shots (the regions with significantly varying data) that were previously detected in *video representation* or *measure of difference* curve. Many types of classification exist and they are discussed in detail in this section.

2.6.1. Classification based on Direct Analysis

Abrupt transitions (namely cuts) are easier to detect and classify than gradual transitions since they are a sudden change between two consecutive frames. Hence, it represents an isolated and sharp pulse in the *video representation* or *measure of difference* curve while the gradual transitions will only represent a mild change in the representation curves (see figure 2.6). Many papers and research work such as [114] have used this fact to classify the cuts.

More common gradual transitions such as fades and dissolves, on the other hand, are identified as a gradually increasing picks. Zabih *et al.* [114] used edge detection and edge pixel analysis to distinguish between fades and dissolves. They pointed out that during a fade in number of entering shot edge pixels is a lot higher than the number of edge pixels in the existing shots, and vice versa for fade out. On the other hand, during a dissolve the frames which have same number of entering edge pixel as their exiting edge pixels. During the first half of dissolve the number of entering (appearing) edge pixels surpass the number of exiting (disappearing) edge pixels, and vice versa for the second half.

Classifying wipes has proven to be a more challenging task than either fades or dissolves. Unlike latter transitions which cause a gradual change in pixel values over time, wipes cause a sudden change in pixels values and a sequential change in pixels spatial distribution over time (see wipe section under transitions overview for more details).

Hence, wipes can be classified through analysis of *video representation* or *measure of difference* as well as spatial distribution of entering (incoming) and exiting (outgoing) shots. The following works have aimed at detecting and classifying wipes: [30], [49], [74], [81], and [114].

Zabih *et al.* [114] have also attempted to classify simple wipes (horizontal and vertical wipes) by looking at the spatial distribution of entering and exiting edge pixels. They do this by recording the edge pixels and analyzing their spatial distribution while computing the edge change ratio fraction. To detect the wipes they divide the images (frames) into two vertical and two horizontal halves.

During a left-to-right wipe, the change in pixels values take place in the left hand side of frames during first half of the wipe. Similarly, the changes take place in the right hand side of the frames during the second half of the wipe.

During a top-to-bottom wipe, the change in pixels values take place in the top half of frames during first half of the wipe. Similarly, the changes take place in the bottom half of the frames during the second half of the wipe.

During wipes, there are no patterns between the number of entering and exiting edge pixels as there was in case of fades and dissolves. Hence, the relative differences between the number of entering and exiting edge pixels is small. This is because the changing pixels only occur in a limited strip in the image.

Motion is the other *representation* technique that is commonly used as basis of comparison for wipe classification.

2.6.2. Classification based on Clustering

Classification step can be replaced by a clustering technique. Clustering does not require pre-defined classes whereas classification methods do require the classes to be predefined. In clustering techniques, the items are divided into different groups and classes one by one as they are analyzed. Many papers suggested the use of clustering methods to avoid the lower performance caused by introduction of thresholds. This way, they overcome the need for a training stage and the problem of parameter estimation [36]. The work in clustering include but is not limited to [36], [42], [46], and [80].

Gao *et. al.* [42] used a fuzzy clustering technique in combination with spatial differences and histogram differences for representation, to avoid use of any thresholds. They argue that a clear distinction between the two classes cannot be made and hence they introduce a fuzzy c-means algorithm.

Unlike [42], Ewerth, *et. al.* [36] avoid use of any fuzzy variations for clustering. They argue that fuzzy clustering does not prevent thresholding completely and rather it shifts it to the defuzzyifying stage. Hence they suggest a different clustering method to perform temporal video segmentation without the need for any thresholds and as they claim without use of any parameters. They use a c-means algorithm to divide the items into cuts and non-cuts groups and after a class membership optimization stage present their final results.

2.6.3. Classification based on Learning

Over the years, much research has been carried out in the field of machine learning and many related techniques are the results of these works. This section provides a brief overview for most of different types of learning techniques and a more details overview for a few of them. These techniques can be used for video boundary shot detection and as another method for preventing thresholds. The details are not discussed since learning algorithms are not the topic of this research. Below is a list of these learning methods:

- Case Base Reasoning (CBR) – is a context based method. It is based on syntax rather than numbers. Hence it is not used in video processing. Refer to [104] for more information.
- Support Vector Machines (SVM) – is discussed later on in this section.
- Neural Networks (NN) – more information can be found [40], [66] and other related papers.
- Linear Vector Quantization (LVQ) – [66] used this technique as a learning algorithm for detecting and classifying dissolves. However the author of this document was unable to find out more about this technique.
- Bayes Classifier – is a classification method used in many different areas. More information regarding this technique can be found [41], [66]
- Hidden Markov Models (HMM) – is discussed later on in this section.

Support Vector Machines (SVM)

Support vector machines are usually used when each item (class) in the data set contains many attributes. For example in [39], each compound (class) in the data set had thousands of features (attributes) which were reduced to two hundreds to reduce the algorithm run time. [39] and many other papers have used [23] provided SVM algorithm. [23] offers a library for support vector machine. The algorithm is not the most advanced algorithm; however it has proven to work great for many different data sets as is shown in [39].

Hsu, *et. al.* [52] have put together a library for Support Vector Machine (SVM) methods and in their paper they describe the process that will give the best possible results if followed. They mentioned that SVM works very well for cases where there are not many attributes presenting each class. In such a case, there needs to be a pre-processing feature selection algorithm where it identifies a subset of attributes and uses those in SVM algorithm. Figure 2.26 represents the screen shots for the web interface of a simple version of SVM, provided by Hsu, *et. al.* [52] for classification of 2D feature spaces using SVM. It demonstrates the results from running the SVM algorithm for three distinguished groups (figure 2.26 (b)), and the results from running the SVM for three more realistic groups where there cannot be a clear border between each (figure 2.26 (c)).

Chua *et al.* [24] proposed a unified approach to detect cuts and gradual transitions by using temporal multi-resolution approach through applying wavelet transform to frame dissimilarity measures. They use histogram differences and coarse representation of MPEG motion vectors. As the first step, they detect candidates from the set of local maxima and then they use an adaptive thresholding technique. As the last step, they use SVM via active learning to find the active hyper-plane that separates cuts and non-cuts.

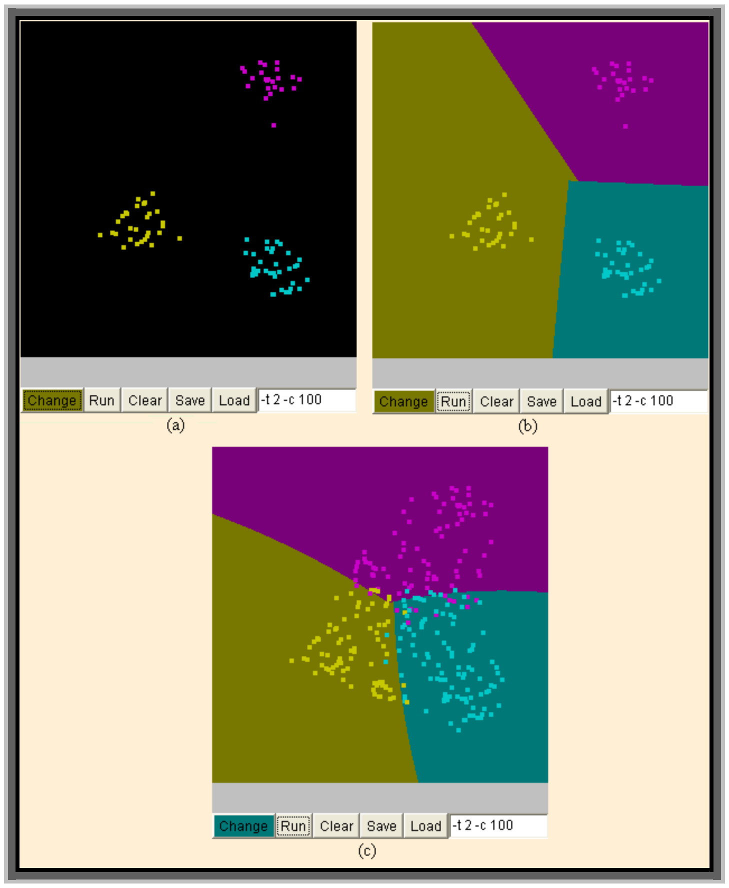


Fig. 2.26. (a) Represents initial three input data sets (b) SVM result for well distinguishable sets (c) SVM result for a more realistic case with sets overlapping. In such a case, not all the items will be located in their SVM determined regions.

[78] presents yet another research which aims at detecting dissolves by using support vector machines.

Hidden Markov Model (HMM)

By using HMM there will no longer exist a need for any types of thresholds, which is a plus since threshold based techniques usually have higher numbers of false negatives in the final outcome. HMM framework allows any number of features to be included in a feature vector.

Boreczky, *et. al.* [17] use hidden markov models (HMM) for temporal video segmentation. Their HMM is consisted of the following states: shot, fade, dissolve, cut1, cut2, zoom, and pan. Two states for cuts are used to avoid mislabeling them as short gradual transitions. Each state in the model was connected to the other states and a probability was assigned to each connection. From the shot state it is possible to go to all the other states other than cut2, however, from the transition states and global motion states it is not possible to go to any other state rather than back to shot state. All states except cut1 and cut2 can loop to themselves (number of self-loops represent the length of shot, transition, or motion depending on the state). Cut2 can be accessed only by cut1 state and cut1 can only access shot with a probability of one.

The authors calculate three features (measure of difference) and use them as the basis for their model. First feature they use is the gray-level difference between two adjacent frames. Second feature is the audio distance based on the acoustic difference in intervals before and after the frames. And finally, an estimate of object motion between the two frames. They calculated the values for transition probabilities by using a standard algorithm for training hidden Markov model parameters, namely Baum-Welch algorithm which was presented in [89]. As the final step they performed segementation by using the Viterbi algorithm (a standard technique for segmentation and recognition using HMMs) which was also presented in [89].

2.6.4. Conclusion

In this section, *classification* and labeling stages of various video shot boundary detection algorithms were discussed. Most of the reviewed methods are based on various modeling techniques. Using model based techniques have been improved over the years; however, for each type of transition, they require a new model and consequently a new scheme. Hence, it is very difficult to use above techniques for edits such as wipes and graphical transitions which have many different variations. Hence considerable amount of research has been dedicated to use of learning or clustering methods which were described earlier in this section. The next section covers the last stage in temporal video segmentation process flow.

2.7. Pre- and Post-Refinement Methods

Video shot boundary detection is a problem that has been extensively researched, but achieving highly accurate results continues to be a challenge [97]. Lienhart concludes his paper, [63], by claiming that all detection algorithms are influenced negatively by global and local motion in the video. He goes on by suggesting future approaches should concentrate particularly on identification of local and global motion.

Even detecting the simplest transitions (namely cuts) can prove difficult in a noisy video stream comprised of the effects pointed out in the introduction section. Lienhart [63] argues that his algorithm 5% false hits rate is caused by dark or very dynamic scenes with strong object motion, blasts or fast camera pans.

Most videos contain one or more of the false positive causing effects. For example, Porter, *et. al.* [88] system for BBC Wildversion was produced while having object motion in mind. As they argue, wildlife films have significant object motions. Hence, many researchers have developed pre and post detection techniques to prevent or detect false alarms and missed items. Many post-refinement methods have been proposed over the years such as the ones mentioned in [33], [29], [63], [64], [71], [103], and [113].

2.7.1. False Positive (False Alarms)

Based on [116], [113], [63], [88], [71], [33], [29], [84] and our research main sources of false positives are as follow:

- Object motion, especially
    - fast moving objects (figure 2.27),
    - close to camera object movements (figure 2.28).
- Global camera motion (see *Basic Camera Operations* section),
    - zoom ins and outs,
    - panning, and
    - fast camera motion.
- Sudden change in pixel intensities and image luminance due to
    - camera flash (figure 2.29),
    - shinny objects (figure 2.30),
    - lightning,
    - blasts,
    - poor video quality, or
    - simply change in brightness (figure 2.31).
- Existence of MPEG various frame types (I, P and B).
- Multiple identification of a single gradual transition.
- Graphical transitions or computer generated effects such as morphing.

- Captions and overlays (see figure 2.32) such as
    - movies overlaying opening credits on top of a scene, and
    - news overlaying location of reporter in the beginning of story.
- Split-screen technique such as
    - ticker-tape, and
    - interviews.
- False positive causing effects for fade such as
    - Wide screen black bands (figure 2.33), and
    - Channel Logo (figure 2.34).
- 3D motion of 2D object (figure 2.35 and 2.36).
- Sudden Appearance of objects (such as magic shows).
- Monochrome objects or views (such as sky). See figure 2.37.
- Wipe false positives causing effects (such as shot of a door opening). See figure 2.38.
- High Level of activity (i.e. occurrence of too many transitions and effects in close proximity).
- Overlapped shots (i.e. shot generated through an overlapping process similar to dissolve generating process). See figure 2.39.
- Andy Sequence

An example of split-screen interview is when the anchor-person and background stays fairly static while a small window displays another event which switches among different reporters. The events in the small window trigger a large change in features (measures of difference) and consequently result in a false positive since in reality the main window has not changed.

Above effects are discussed in many literature such as [37], [71] and [114]. [71] proposed a method based on average shot length and other statistics to deal with these effects whereas [114] discussed the difference between overlays and transitions.

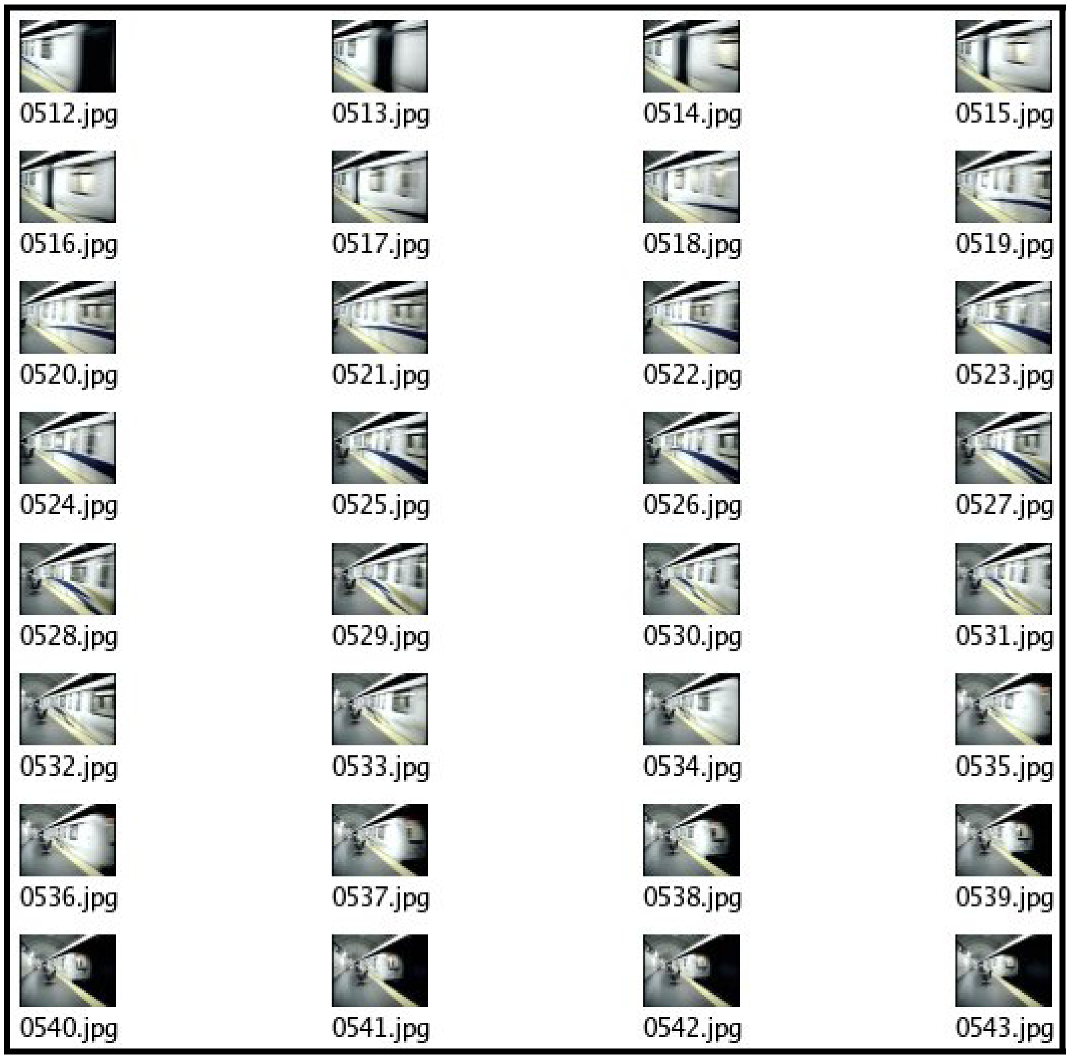


Fig. 2.27. Presents an example of a false positive causing effect: Fast moving object.

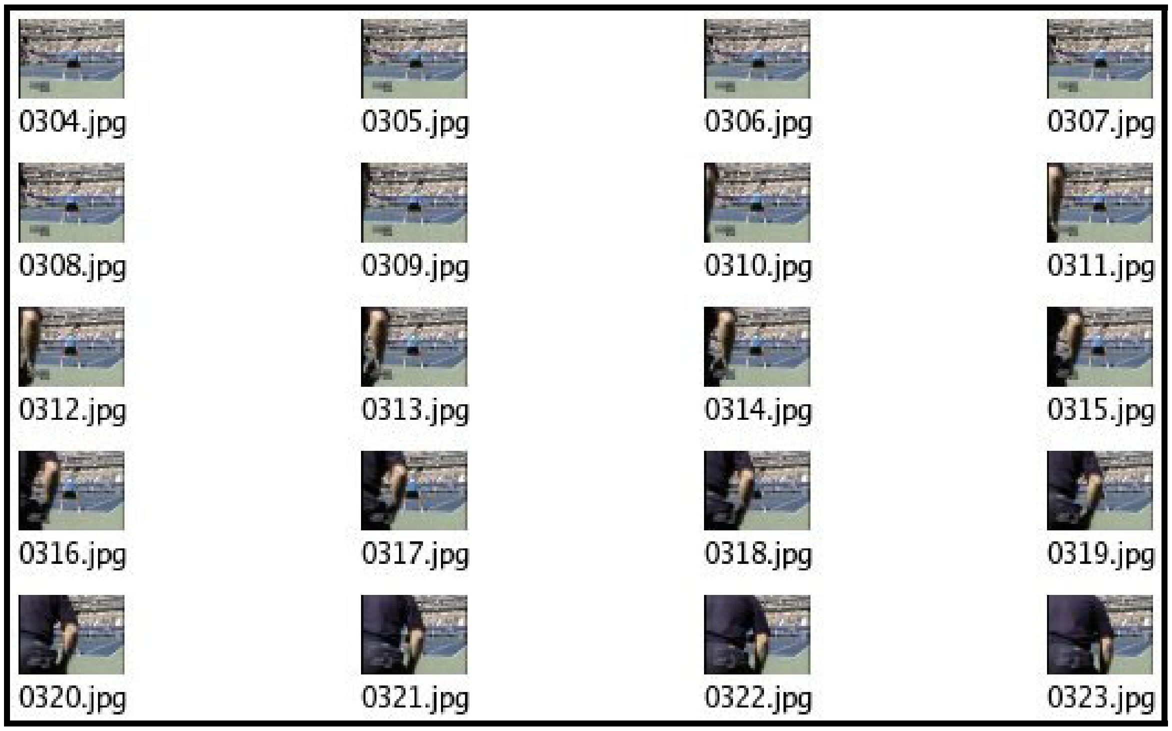


Fig. 2.28. Presents an example of a false positive causing effect: Close to camera object motion.

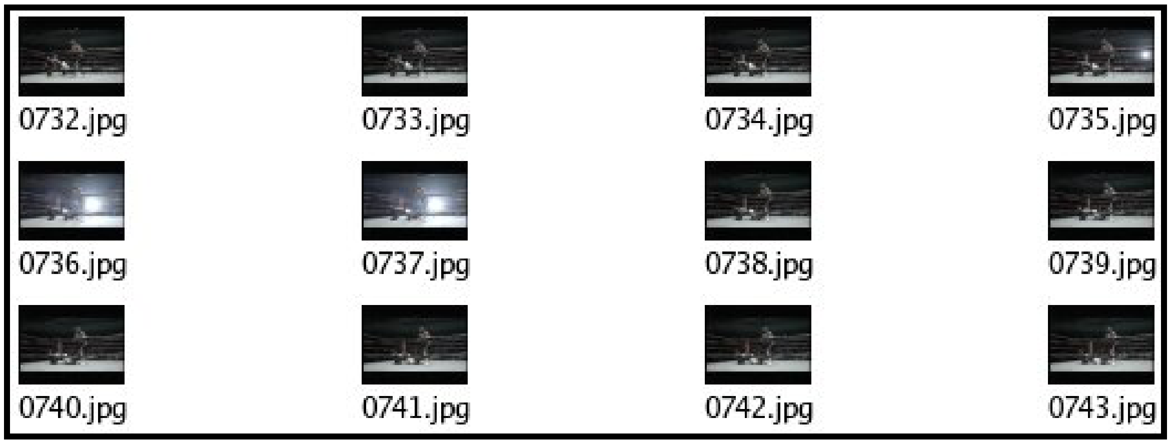


Fig. 2.29. Presents an example of a false positive: Photography camera flash.

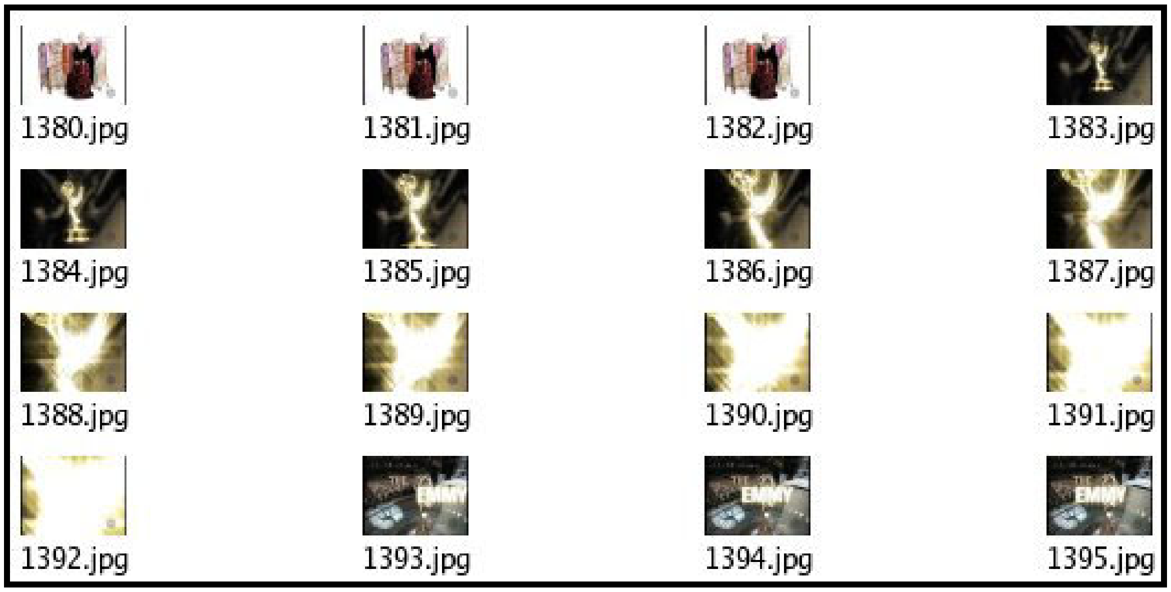


Fig. 2.30. Presents an example of a false positive causing effect: Shinny object.

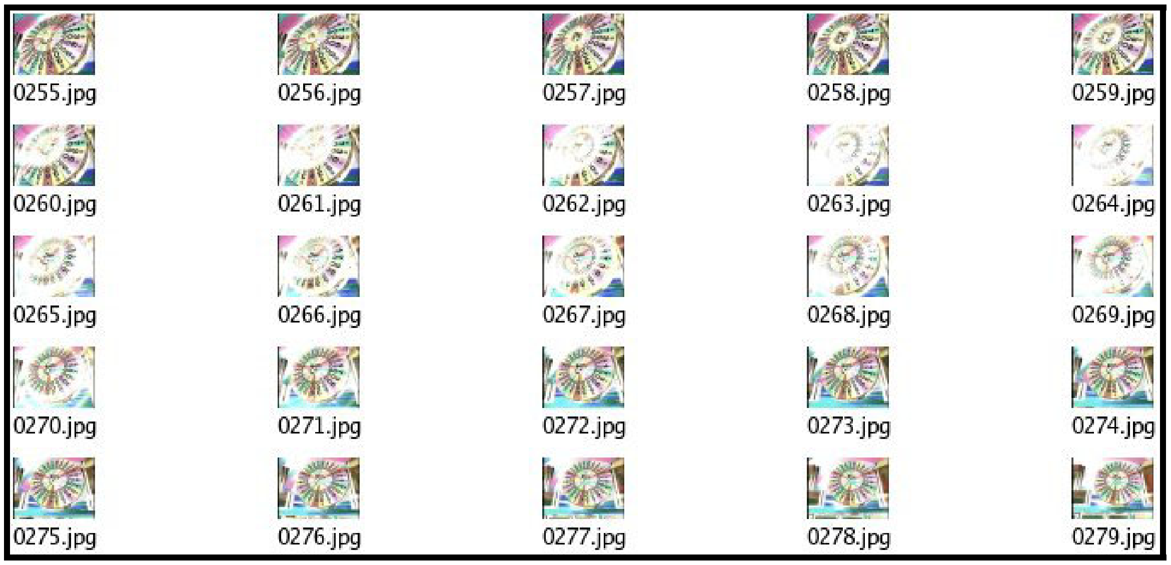


Fig. 2.31. Presents an example of a false positive causing effect: Sudden change in brightness.

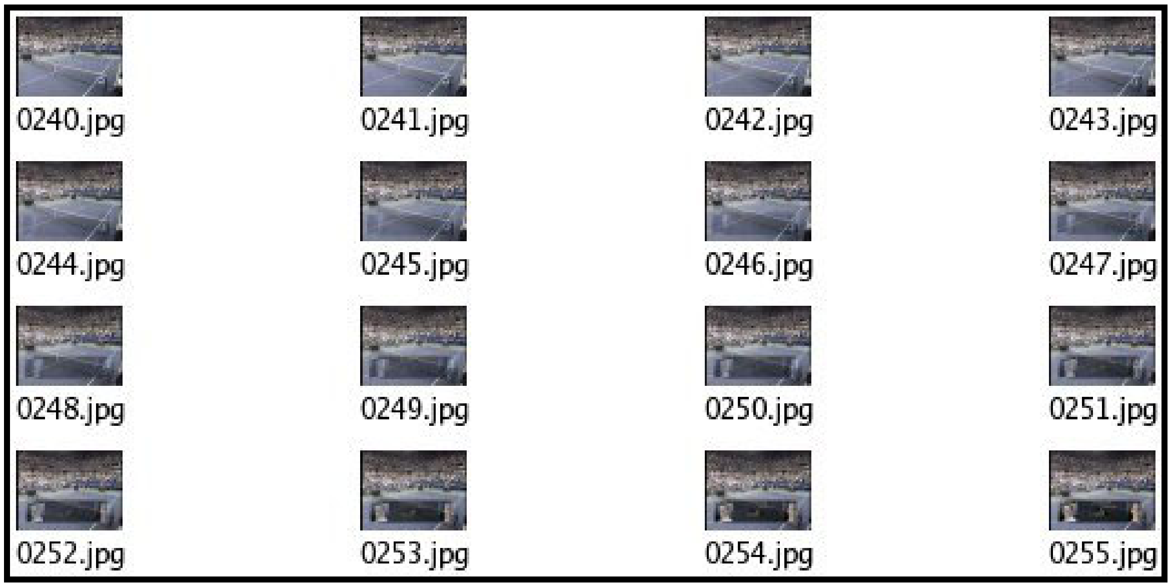


Fig. 2.32. Presents an example of a false positive causing effect: Overlay dissolve.

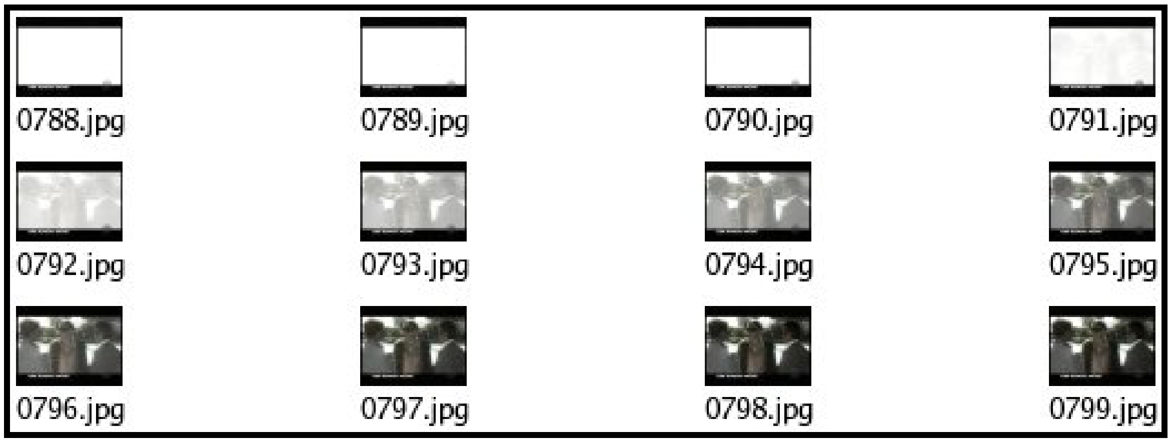


Fig. 2.33. Presents an example of a false positive causing effect: Existence of wide screen bands.

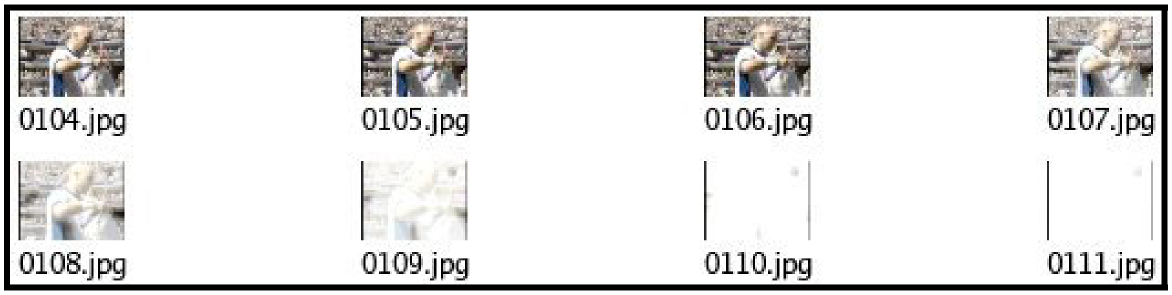


Fig. 2.34. Presents an example of a false positive causing effect: Channel logo.

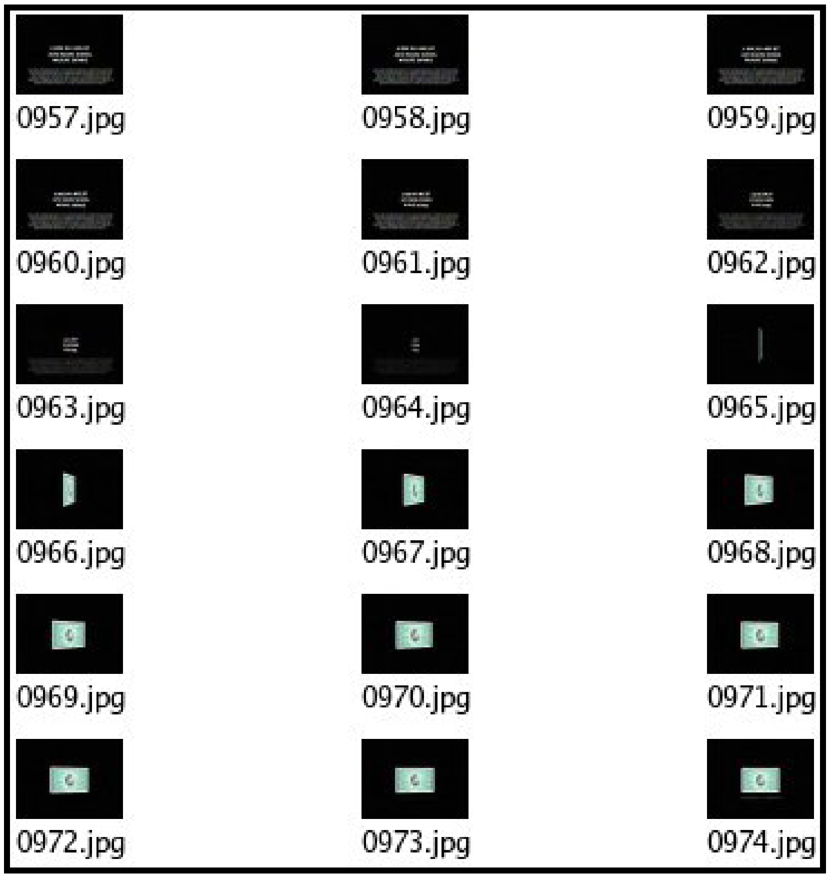


Fig. 2.35. Presents an example of a false positive causing effect: 3D motion of 2D object.

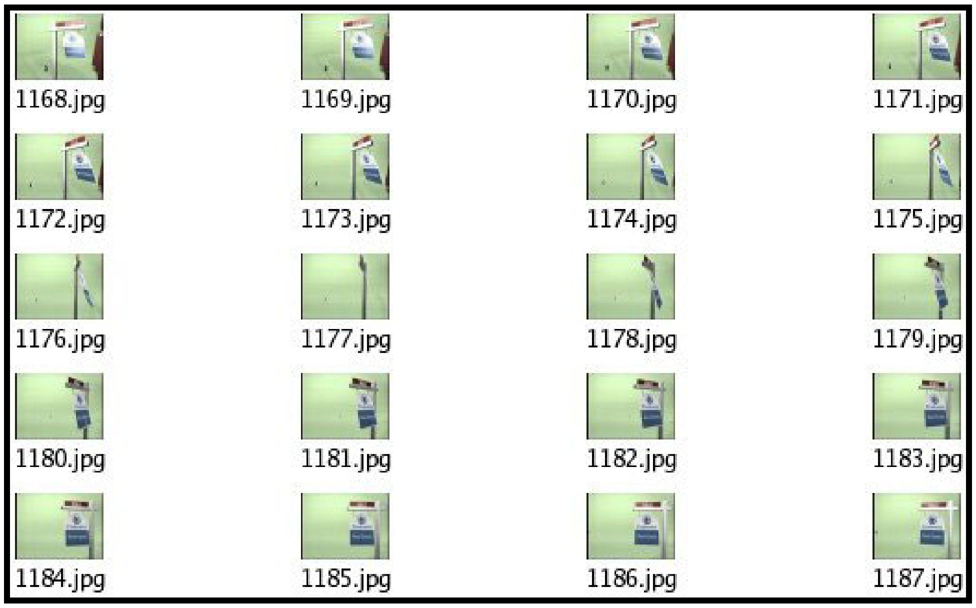


Fig. 2.36. Presents an example of a false positive causing effect: 3D motion of 2D object.

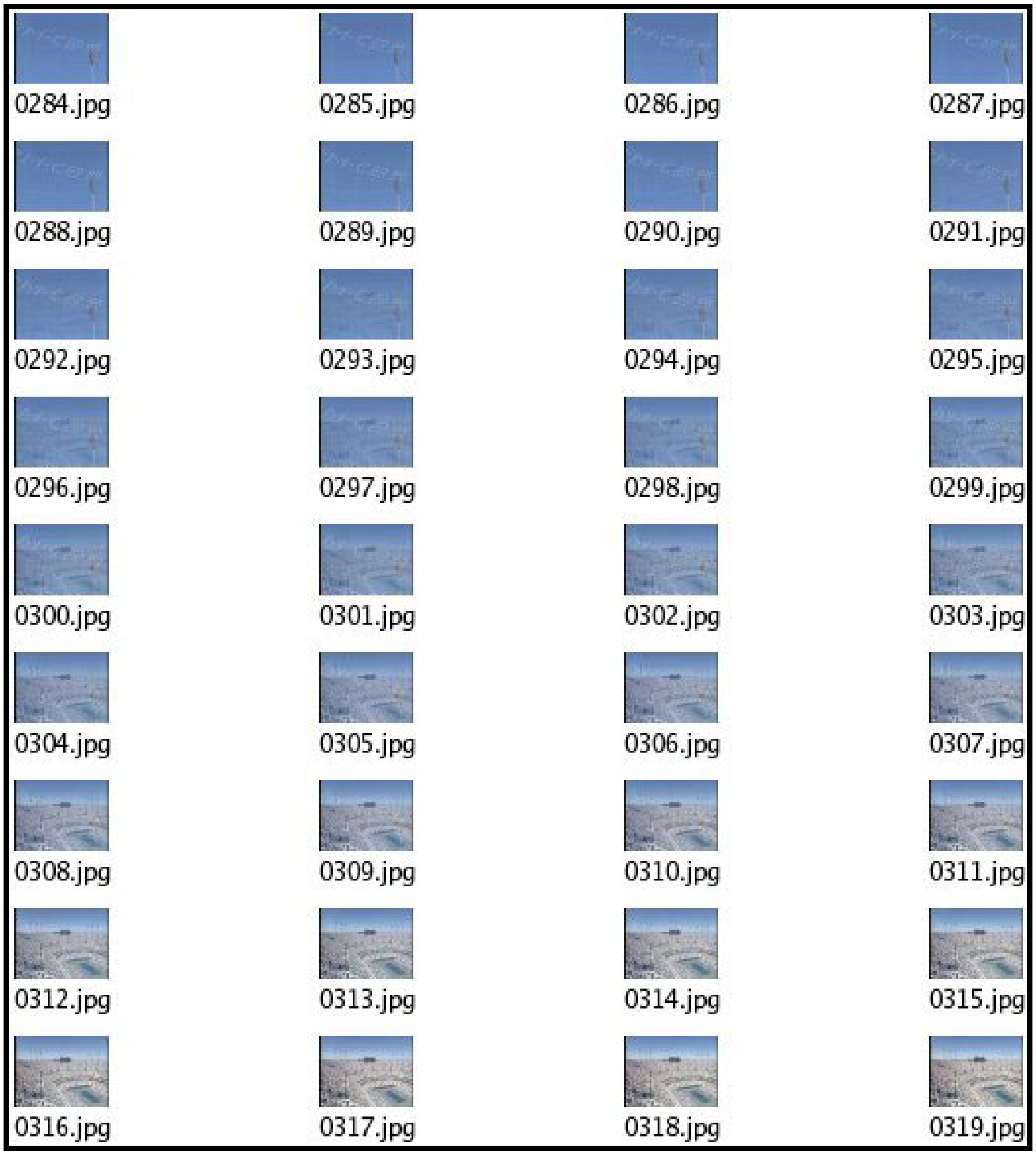


Fig. 2.37. Presents an example of a false positive causing effect: Frames including the sky resemble blue monochrome frames.

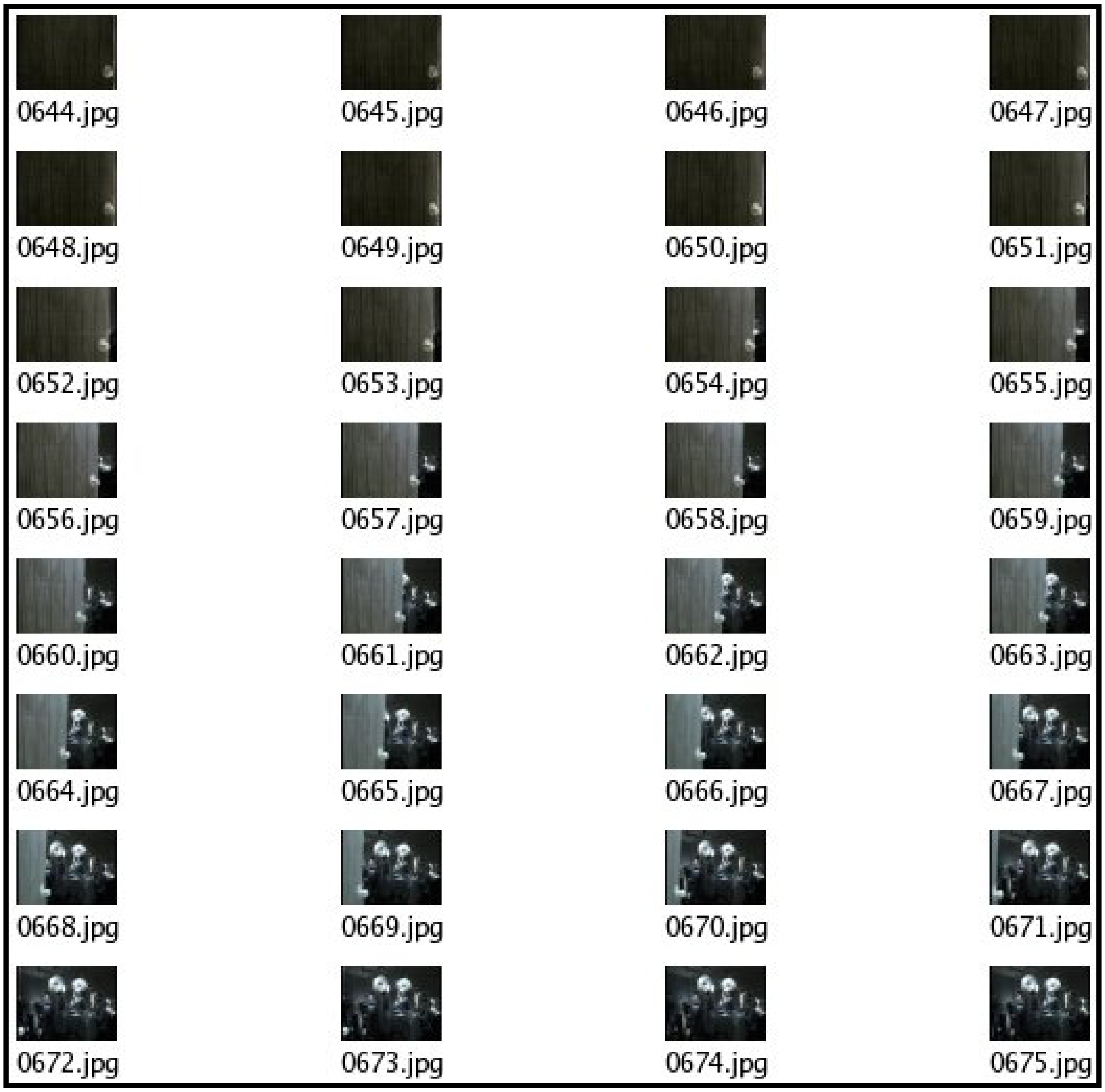


Fig. 2.38. Presents an example of a false positive causing effect: Shot of an opening door.

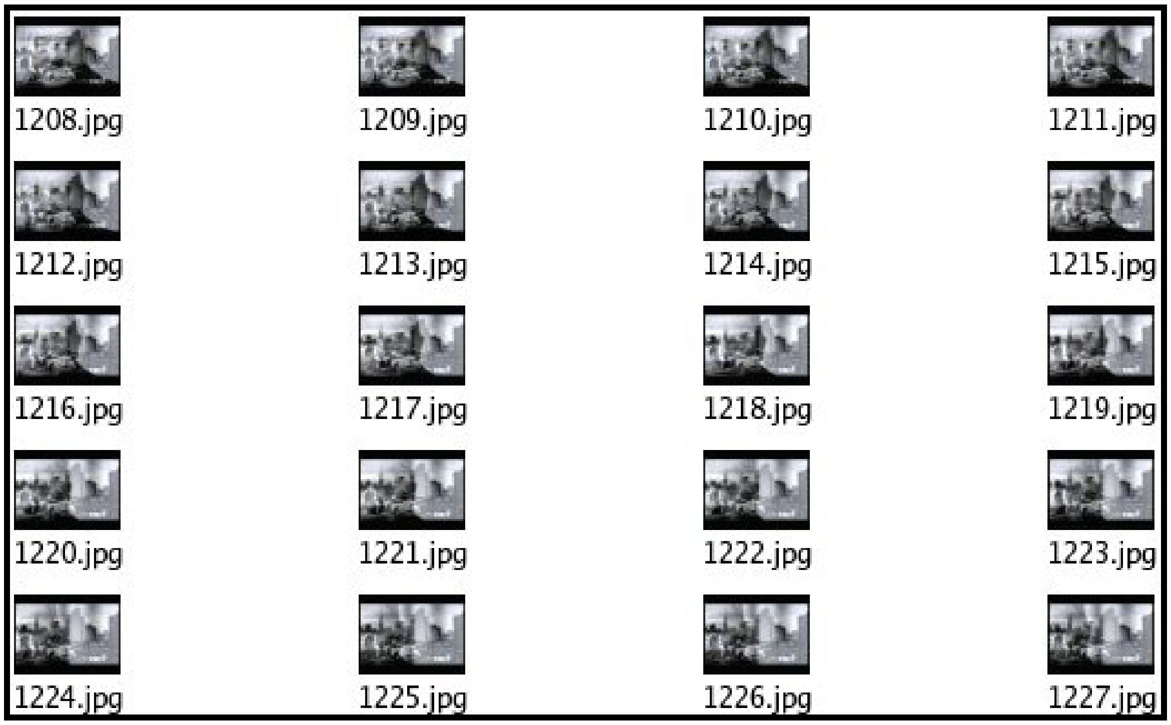


Fig. 2.39. Presents an example of a false positive causing effect: Overlapped shots.

Prevention

False positive prevention refers to the methods used before each major stage of temporal video segmentation algorithms and specifically intended to reduce the number of false positives. Most of these methods directly deal with input data manipulation.

One method which can be used is *smoothing*. Smoothing technique similar to those used in digital image and signal processing algorithms can be applied to video input stream to reduce amount noise in the *video representation* or the *measure of differences* curves.

Normalization is another method which can be used for this purpose. Similar to smoothing methods normalizations directly affects the values in the input stream.

Normalization stage insures that the different data used will have similar range and follow the same distribution curve.

Zabih, *et. al.* [114] suggest that avoiding methods such as sub-sampling will help to prevent some of the false positives. In algorithms based on compressed data, similar problems exist. In those algorithms, the false positives can be avoided without a need for major modifications to the algorithm through introduction of specialized methods for false positive prevention or detections. Further investigation of this topic is left to the reader since compressed data are not used in our research.

Other methods for false positive preventions include but are not limited to analysis of video at multiple resolutions [24], prior detection of significant camera and object motions [94], and use of motion vectors [16].

Detection

False positive detection refers to the methods used after temporal video segmentation algorithm execution and specifically intended to reduce the number of false positives by identifying them in the final output and eliminating them.

Lienhart [64] used the motion estimation algorithm that was suggested by Dufaux *et. al.* [31] to eliminate false positives that were caused by camera operations such as pan and zoom.

Truong *el al.* [103] suggest a simple histogram based algorithm for detecting false positive. They basically take two arbitrary frames, one from the preceding and one from the following shots of the transition in question. If these frames difference is less than a empirically determined threshold then the previously proposed boundary is marked a false positive.

Lienhart [63] proposes a technique for dissolves false positives. As a post-processing step, his algorithm iterate through all the dissolve candidates. The internal frames of each potential dissolve are removed, leaving only the boundaries of preceding and succeeding shots. Then the result is processed through a cut detection algorithm and if it was detected as a cut then it is marked as a true positive. His method has improved the final results significantly (especially in one experiment false positives rate was reduced from 8500% to 400%).

He improves his dissolve false positive detection technique even further in [64] by using the motion estimation method that was previously proposed by Dufaux and Konrad in [31]. Lienhart compares the dissolve candidates with frames containing significant camera motions. If they had $\frac{2}{3}$ of their frames in common then the dissolve was marked as a false positive. Keep in mind that even this algorithm itself is not error-prone. To prevent some of the problems, Lienhart suggest integrating the camera motion parameters with the detection algorithm instead of using it in a post-processing stage.

Ewerth, *et. al.* [33] and [34] suggest a method for detecting false positives in their MPEG based cut detection algorithm without reducing the recall rate. The main reason for a false alarm is existence of many frames in MPEG format (I, B, and P) – for more information about MPEG refer to *Segmentation based on Compressed Data* section earlier in this chapter. They suggest using a *normfactor* which will take into account the different types of frames in MPEG.

Dailianas *et al.* [29] suggest two algorithms for dealing with false positives. The first compares the current frame with the *k* preceding and *k* succeeding frames and replacing the current value with the local minimum if some condition was not met. The second algorithm employs a simple moving average window technique.

The first step in Lu and Tan's post-refinement method [71] is to detect "small shots" by choosing a low threshold in the detection algorithm. These shots are basically falsely detected shots or the shots that are caused by false positive causing effects that were mentioned earlier. After detecting these shots, they merge them together or to the adjacent shots.

As their second step, they look at the histogram pattern of shot boundary candidates. If the patterns of preceding and succeeding shots of transition candidate follow the correct distribution pattern then a shot boundary is declared; else if the succeeding continued the same pattern as the previous one then it was declared as false positive.

As Zabih *et. al* [114] mention the overlays can be similar to a cut in a sense that they suddenly appear and disappear, or they can emerge and fade away in a similar fashion as fades or dissolves, however they do contain more data than transitions. Zabih *et al.* use this fact to detect captions using edge detection techniques as *measure of differences* and *Haudorff distance* similar to the one used in [54] to detect captions as well as scene breaks. Huttenlocher, *et. al.* [54] incorporates the probability of a false match to better locate the true false positives. Also there has been specific works by researchers such as Lu *et. al.* [71] for post-refinement of temporal segmentation results.

2.7.2. False Negative (Missed Items)

Based on [71], [29] and our research main sources of false negatives are as follow:

- Very long gradual transitions.
- Very short gradual transitions.
- Close proximity to other transitions or effects.
- Resemblance of bordering frames of two adjacent shots in terms of
    - luminance (and brightness level),
    - colors used, and
    - color distribution.
- Black and white video input streams.
- Camera on/off effect (figure 2.40).

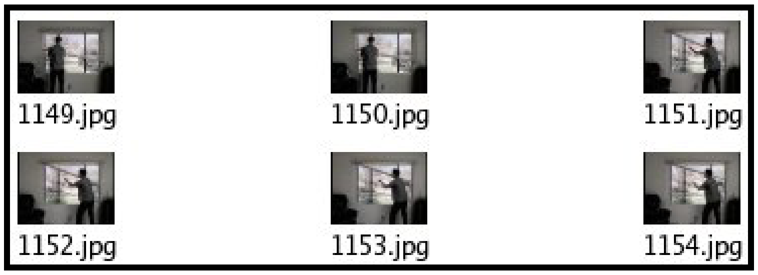


Fig. 2.40. Presents an example of a false negative causing effect: Camera on/off effect.

Prevention

False negative prevention refers to the methods used before each major stage of temporal video segmentation algorithms and specifically intended to reduce the number of false negatives. Most of these methods directly deal with input data manipulation.

The major technique for preventing false negative is avoidance of any threshold applications. Thresholds, especially the global and random thresholds, are the main source of false negatives in the algorithms which utilize them. Beside thresholds, similar techniques as described in false positive prevention section can be used to eliminate completely or decrease number of false negatives.

Earlier works such as Dailianas *et al.* [29] concluded that use of a local threshold (rather than global) approach to reduce the number of false negatives, however nowadays this is a fact.

Detection

False negative detection refers to the methods used after temporal video segmentation algorithm execution and specifically intended to reduce the number of false negatives by identifying them in the original input stream while considering the final output.

Lu, *et. al.* [71] proposed to detect false negatives through a sequential detection on all the frames within a shot. They first obtain the color histogram for each shot and partitioning it into two equal segments. Then two feature distributions are approximately estimated using each segment. To detect false negatives, they examine the first- and second-order finite differences (as was proposed in [1] and [86]) of the log-likelihood ratios.

Although in the recent years new ideas are proposed for dealing with false negatives, less works are done for dealing with false-negatives during a post-processing step than false positives. This is because false negative elimination is achieved through main algorithm enhancements.

2.7.3. Conclusion

To this end, no algorithm exists for perfectly detecting all types of transitions in an arbitrary input video stream. Hence, over the years many pre- and post-procedures were developed to decrease the consequence of the false causing effects. These methods discussed in this section have significantly improved the end results of video boundary detection algorithms.

2.8. Surveys and Other Resources

Numerous research and papers, [4], [11], [16], [24], [29], [61], [63], [64], [65], [75], [103], [113], and [116] have surveyed, compared and contrasted, or used various methods for video boundary shot detection.

Lienhart [63] has compared four different methods, Color Histogram Differences, Edge Change Ratio, Standard Deviation of Pixel Intensities and last but not least Edge-based Contrast.

Although many published methods of detecting shot boundaries exist, it is difficult to compare and contrast the available techniques. This is due to several reasons. Firstly, full system implementation details are not always published and this can make recreation of the systems difficult. Secondly, most systems are evaluated on small, homogeneous sequences of video. These results give little indication how such systems would perform on a broader range of video content types, or indeed how differing content types can affect system performance [84].

As a result, *National Institute of Standards and Technology* (*NIST*) annually holds a conference called *TrecVid* during which different participants' algorithms are measured against the same data and in the same circumstances. The conference then ranks different algorithms distinguishing the superior ones in each field.

2.9. Conclusion

Study of existing techniques should always be the first step for any research. This way the researchers will be improving the previous works of others rather than reinventing the wheel all over again. In this chapter, existing research and related previous work on basic camera operations, transitions, video representation, detection, classification, and false detection and prevention techniques were discussed.

Many temporal video segmentation algorithms fail to accurately detect all the transitions in a general video stream even after use of false detection and prevention methods. Hence, temporal video segmentation is still an open topic. Next chapters discuss our research performed to solve some of the existing problems in this area.

Chapter 3

Direct Comparison based on Predefined Examples

3. Direct Comparison based on Predefined Examples

The literature review in the field of video temporal segmentation and its related fields was presented to the reader in the preceding chapter. In this chapter we will discuss the first algorithm suggested by the author.

Two different algorithms were implemented and tested for temporal segmentation and they are as follow:

1. Direct Comparison based on Examples Collections
2. Direct Comparison based on Adaptive Examples

In this chapter, the first method is discussed in detail. We describe our second method in the following chapter. The subsequent chapter provides the reader with our results and evaluations.

3.1. Introduction

Although video shot boundary detection has been a known problem for over a decade, it has become a very active research area in recent years. This is mainly due to the fact that video shot boundary detection acts as the basis for all other video processing research such as scene detection, video indexing and retrieval, and commercial detection.

Hence, much research has focused on improving the final detection outcome through usage of different techniques. The reasons for lower than expected outcome were reviewed in the literature review chapter.

This chapter presents a detailed explanation of the design and implementation of our new temporal segmentation algorithm. The final results of experimentation and testing as well as information about the input files discussed in the futures chapters.

3.2. Suggested Algorithm

In order to detect video transitions such as cuts, fades and dissolves, we compare predefined examples of these transitions to the video stream to locate new cuts, fades and dissolves. Thus, no explicit mathematical models are used in our novel example-based approach.

The goal in this project is to implement an algorithm which is independent of all transition models. In other word a unified algorithm that is used to detect all types of transition as well as any other false causing effects (see chapter 2 for a list of false causing effects, namely false positives) or camera operations.

This chapter includes design and implementation of the proposed video shot boundary detection algorithm.

3.2.1. Design

In this section, the system design requirements are discussed. As mentioned in the last chapter, there are three main stages in a temporal video segmentation system. Figure 1.1

represents a general process flow for all the temporal segmentation algorithms; although this process flow can differ in details from one algorithm to the next.

Like many algorithms our technique follows a slightly different process flow. Figure 3.1 represents the detailed process flow diagram for predefined example based algorithm.

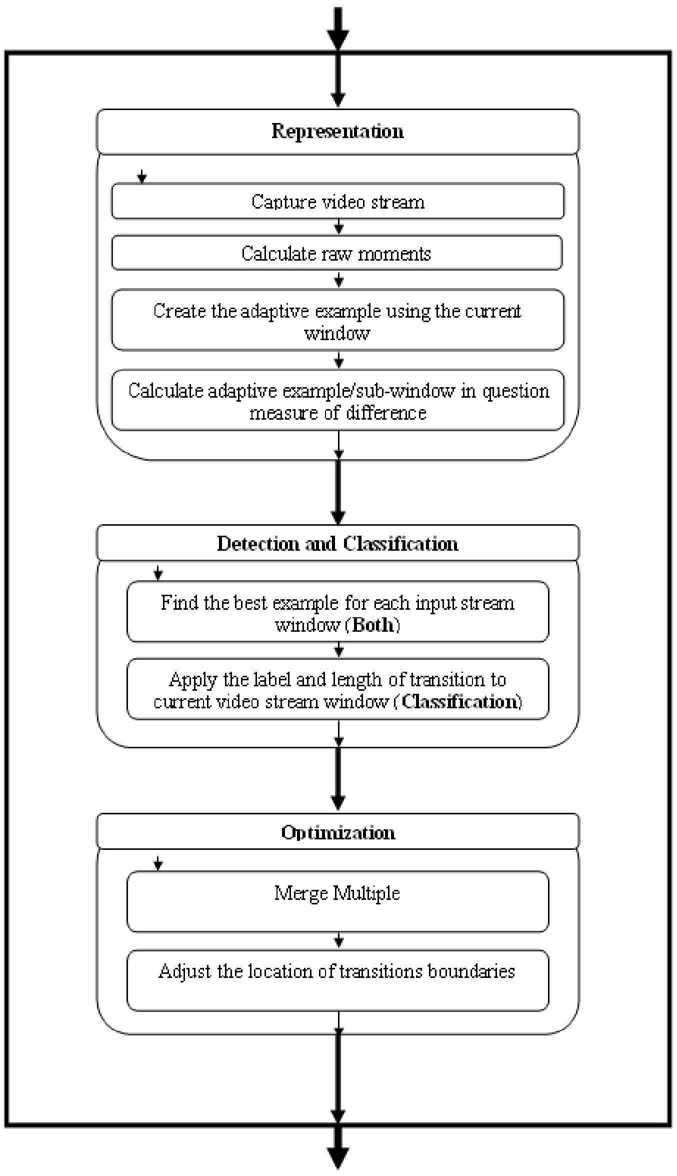

Representation
Capture video stream
Calculate raw moments
Create the adaptive example using the current window
Calculate adaptive example/sub-window in question measure of difference
Detection and Classification
Find the best example for each input stream window (Both)
Apply the label and length of transition to current video stream window (Classification)
Optimization
Merge Multiple
Adjust the location of transitions boundaries

Fig. 3.1. Visualizes the detailed process flow for method based on predefined examples

In the *representation* stage we begin by extracting numerical features that characterize the video input stream and our predefined examples. Then a method is required to compare each time instance of the input video stream with the predefined examples. This step is known as *measure of difference calculation.*

After calculating the *measures of difference* for a series of time instances, the data are sent to *detection* algorithm for further analysis. In this stage, the regions of interest or the potential candidates for shot boundaries are identified. Only these data are used during *classification* and future stages. This way we avoid unnecessary calculations on the fit values that contain no transitions or effects of interest. In our algorithm, *classification* and *detection* stages are combined. In other words, the label of the best matched example is assigned to the current window.

In the final stage the detected and classified results are *optimized.* This section will improve the quality of detection by adjusting the boundaries of detected transitions and by identifying the duplicate detections and removing one of them.

These steps are discussed in details in the succeeding *implementation* section.

3.2.2. Implementation

In this section, the system implementation requirements are discussed in detail for temporal segmentation based on predefined examples. The implementation stage is divided into the following sections:

- Representation
- Detection

- Classification
- Optimization
- False Detection and Prevention

Representation

Video representation and *measure of difference calculation* are the initial and essential component of any video shot boundary detection techniques. This section includes information regarding example sets, as well as these two components.

Examples Set

Examples are the center piece of this algorithm. If proper numerical features are extracted from the video then these characteristics will most likely form groups such as those represented by figure 3.1 when plotted. An example set can be formed by taking some number of examples from each of these groups which are of interest to the research and using them as the representatives of those groups.

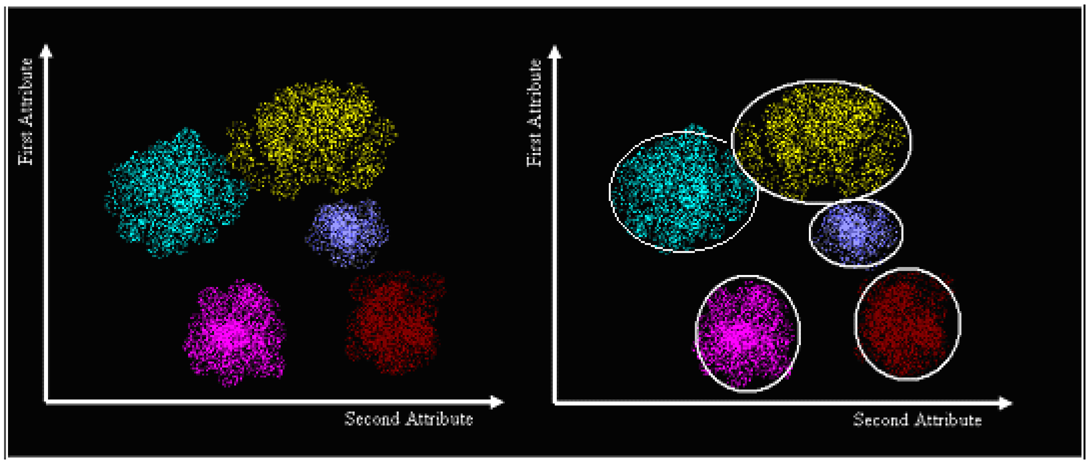


Fig. 3.2. Demonstrates the basic idea behind classification and clustering techniques.

The example set includes not only examples of transitions (such as cuts, dissolves, fade ins, fade outs, morphing, and graphical transitions), but also includes effects that cause false positives (such as high motion sequences or camera pan and zoom effects). See figures 3.3 and 3.4 for more details.

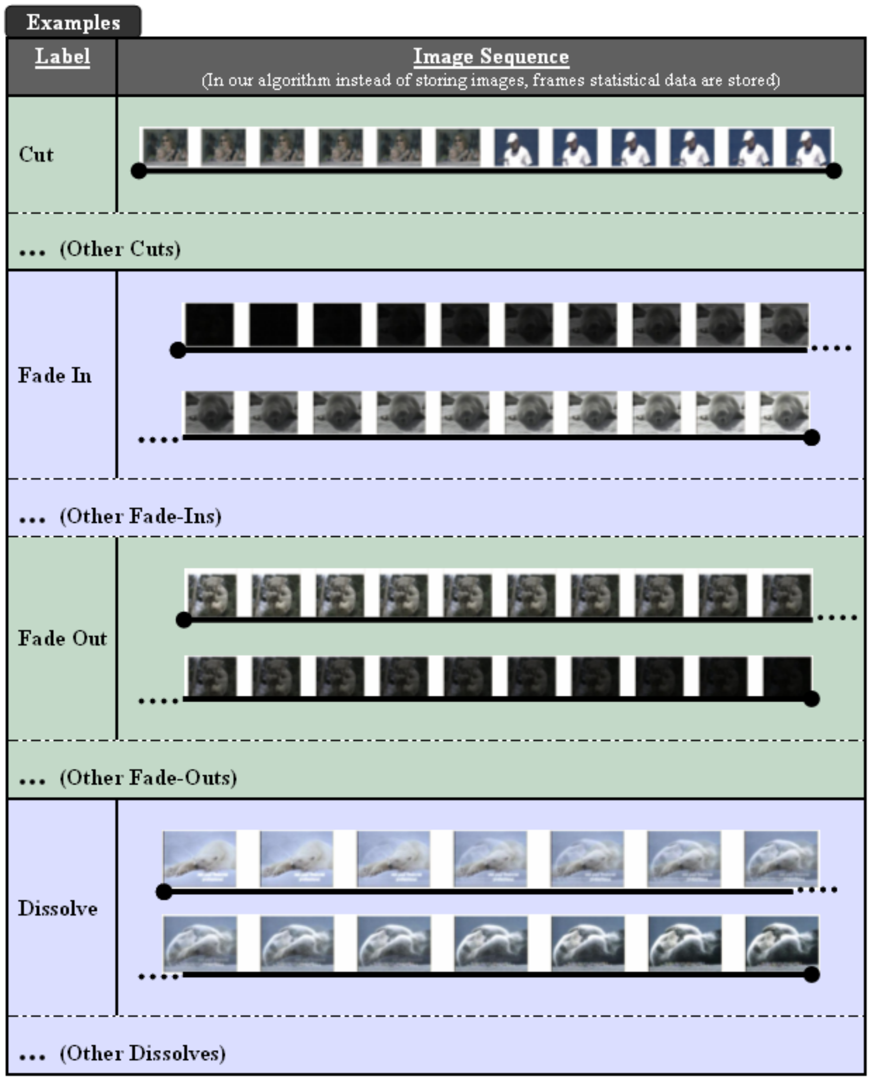


Fig. 3.3. Sample cut, fade in, fade out, and dissolve sequences.

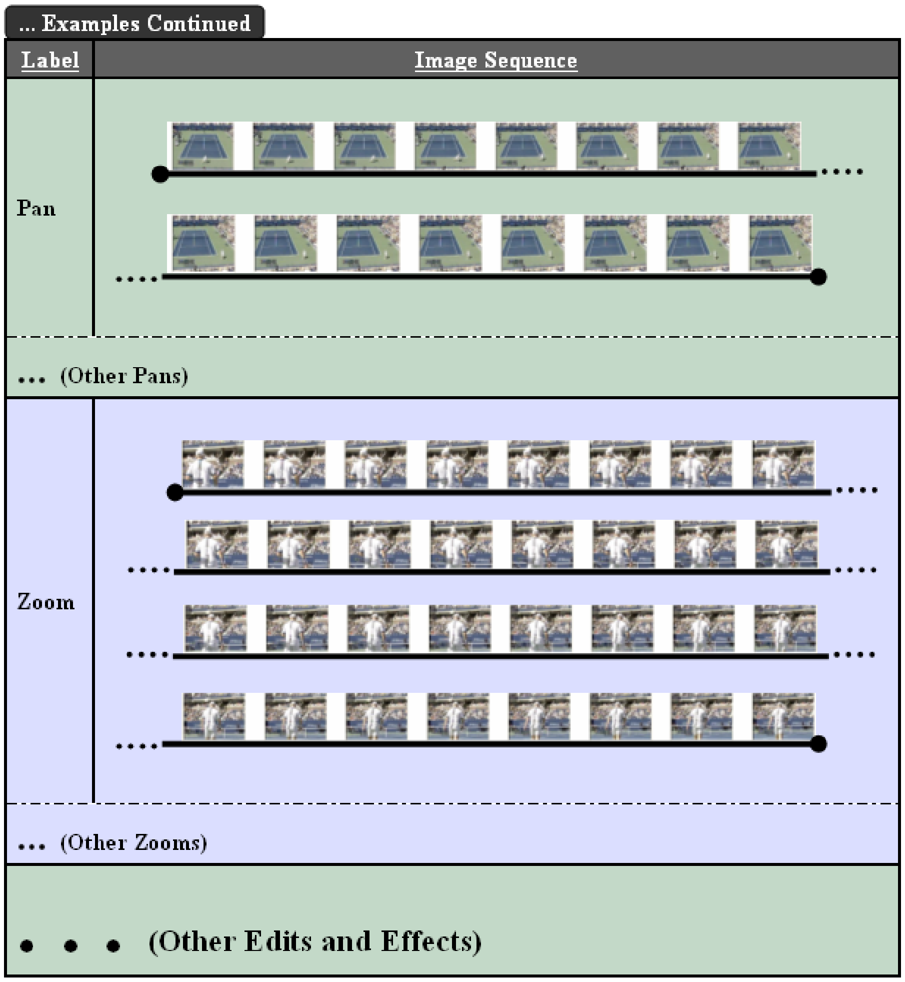


Fig. 3.4. Sample camera pan and zoom sequences.

Example set is the center piece of our algorithm. Nevertheless, our research is based on direct analysis of examples. Hence, employing high-quality example sets is of high importance in this research. What is high-quality example set? In this section this question will be answered by explaining the characteristics of a high-quality example set for video shot boundary detection.

There are two main properties which come to mind when thinking about transitions/effects example set, quantity as well as quality of examples.

Quantity of Examples

Quantity of examples directly affects the performance and detection results. Having more examples require more space and processing time. On the other hand, having too few examples results in insufficient number of examples which leads to undesirable results. Quantity leads to a more desirable outcome if and only if the quality of the examples increases.

Quality of Examples

What are the qualitative characteristics of a high-quality example set? To answer this question, different characteristics of transitions and effects of interest have to be studied. Below we discuss some of these characteristics:

- **Example Duration (length):** Transitions and other effects in a typical video stream have a wide variety of different lengths. Thus, including examples of different length in our example set is important.
- **Example Type:** Video streams use different combinations of transitions. Hence it is important to a wide variety of examples such as: cuts, dissolves, fade-ins, fade-outs, morphing, pan and zoom.
- **Video Type:** Past experiment has revealed that different type of video (such as sport events, news, cartoon, commercials, and so on) contain different type of effects and transitions. Hence, examples of different type of videos have to be considered.
- **Balance:** This factor directs the bias among transitions due to unevenly distributed number of examples (in other words, the quality decreases if there

exists considerably more number of examples for one transition or effect than there is for others).

- **Color Variety:** Our input video stream may contain a wide variety of colors. Hence our examples should represent this diversity. If during a transition example red color intensity drops from high to low then there should be another example for which the red color intensity rises from low to high (similar examples are needed for all possible combinations of red, blue and green color intensities).
- **Combined Transitions:** Besides having examples of single transitions, it is also necessary to have examples of combined transitions such as fade out followed by cut. This also requires the detection and classification algorithm to be adapted to support such examples.

Video Representation

Instead of using sequence of images to represent video we use series of statistical properties which are extracted directly from video for each frame in the image sequence. To represent each image in statistically, the first step is to extract the primary color bands (red, green and blue) for each frame. Figure 3.5 demonstrates this fact.

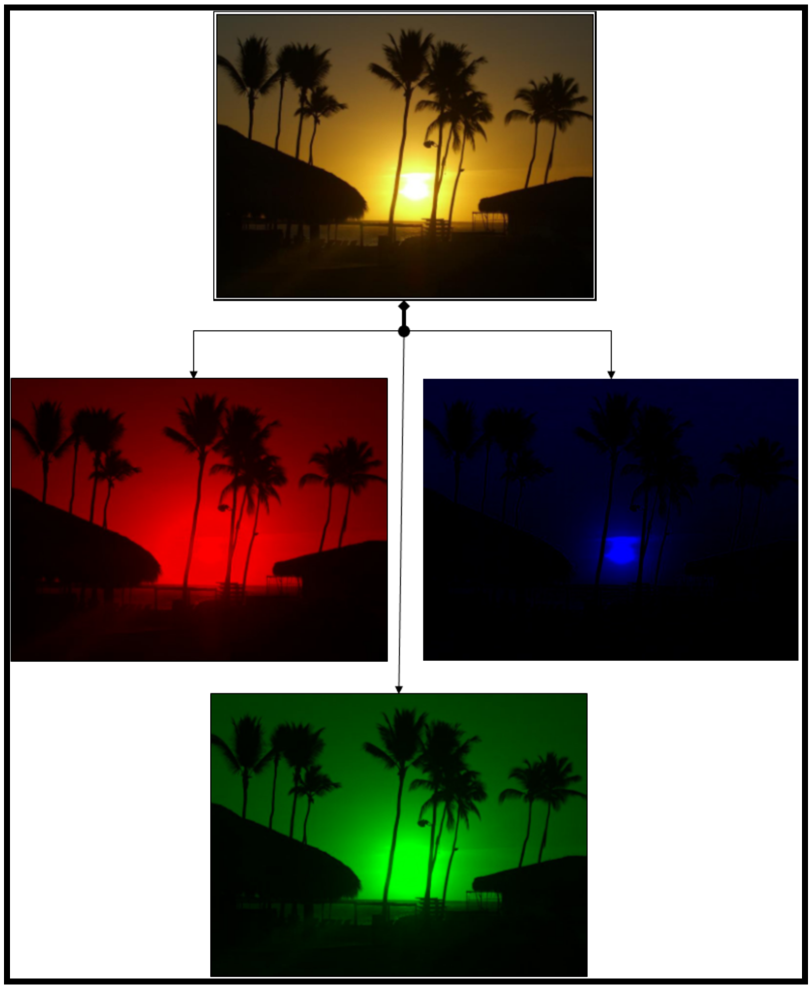
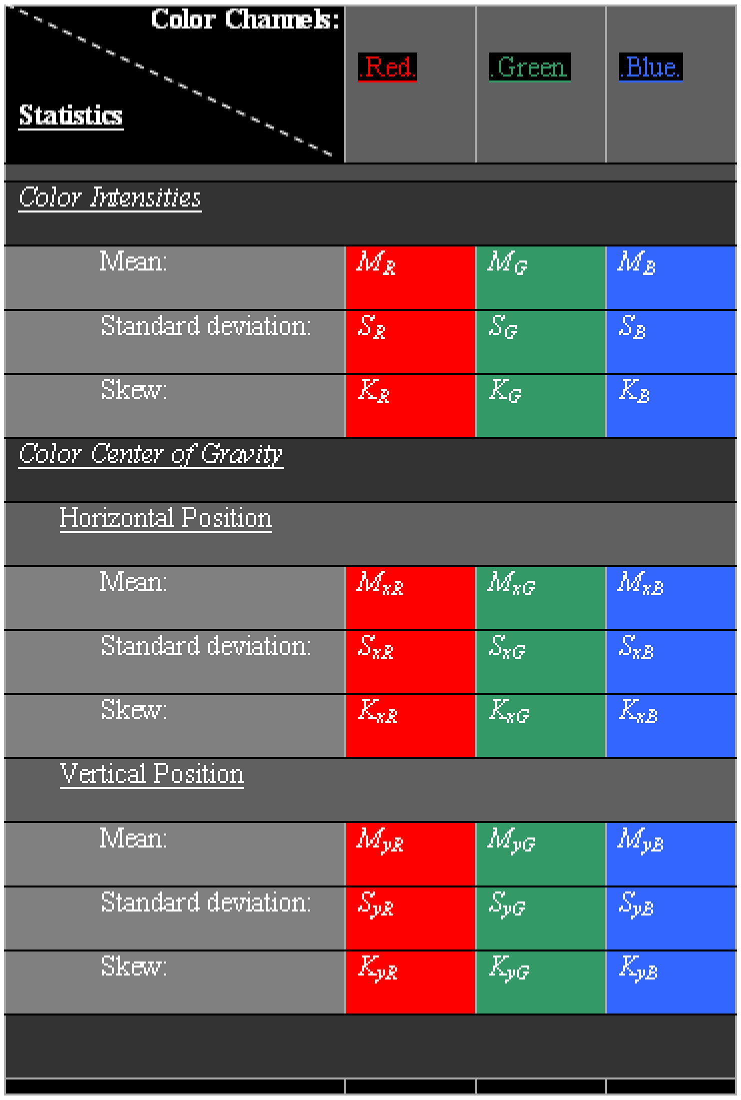


Fig. 3.5. An original image and the extracted three primary color channels images

After extracting the RGB components from original image, the statistical information such as mean (M_R, M_G, M_B) standard deviation (S_R, S_G, S_B), skew (K_R, K_G, K_B), center of gravity (M_{xR}, M_{xG}, M_{xB}, M_{yR}, M_{yG}, M_{yB}) and its related statistical data (S_{xR}, S_{xG}, S_{xB}, S_{yR}, S_{yG}, $S_{yB,}$ K_{xR}, K_{xG}, K_{xB}, K_{yR}, K_{yG}, K_{yB}) for each color component can be calculated. These statistical properties are referred to as color moments and they are summarized in Table 3.1.



Color Channels: / Statistics	Red	Green	Blue
Color Intensities			
Mean:	M_R	M_G	M_B
Standard deviation:	S_R	S_G	S_B
Skew:	K_R	K_G	K_B
Color Center of Gravity			
Horizontal Position			
Mean:	M_{xR}	M_{xG}	M_{xB}
Standard deviation:	S_{xR}	S_{xG}	S_{xB}
Skew:	K_{xR}	K_{xG}	K_{xB}
Vertical Position			
Mean:	M_{yR}	M_{yG}	M_{yB}
Standard deviation:	S_{yR}	S_{yG}	S_{yB}
Skew:	K_{yR}	K_{yG}	K_{yB}

Table 3.1. Organizes the twenty seven moments in an easy to understand fashion

Statistics: Color Intensities

Color Intensities used as the basis for generating the nine basic statistics: the mean, *M*, standard deviation, *S*, and skew, *K*, of the three primary color intensities are calculated for each image (frame). Following this paragraph are the equations and descriptions of these statistics:

- **Mean** – tells us the degree of brightness for average color intensity of the image.

$$M(t,c) = \frac{1}{N}\sum_{xy} I(x,y,t,c) \tag{3.1}$$

- **Standard Deviation** – tells us how wide the object the color distribution curve is (see figure 3.6 a).

$$S(t,c) = \sqrt{\frac{1}{N}\sum_{xy}[I(x,y,t,c) - M(t,c)]^2} \tag{3.2}$$

- **Skew** – tells us how lop sided the color distribution curve is (see figure 3.6 b).

$$K(t,c) = \sqrt[3]{\frac{1}{N}\sum_{xy}[I(x,y,t,c) - M(t,c)]^3} \tag{3.3}$$

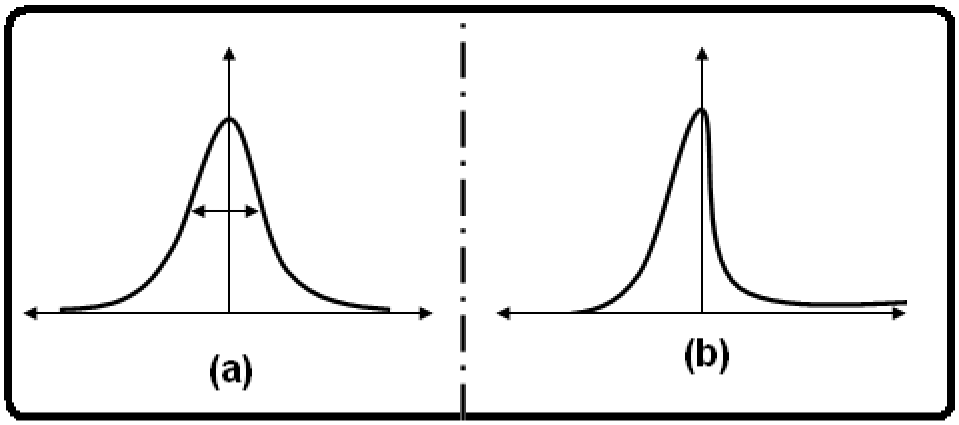


Fig. 3.6. (a) Demonstrates the distribution curve and the fact that standard deviation affects the width of distribution curve. (b) Demonstrates a lop sided distribution curve; the lop sidedness can be calculated using skew.

Statistics: Center of Gravity

If the image has a uniform distribution of colors then the center of gravity simply becomes the center of the image. On the other hand, if the pixel intensities for each component are separately used as weights for *x* and *y* positions during mean calculation, then the final result will be the center of gravity of that color component. This fact can be presented through the following equations:

$$M_x(t,c) = \frac{1}{N}\sum_{xy}\frac{I(x,y,t,c)\cdot x}{M(t,c)} \tag{3.4}$$

$$M_y(t,c) = \frac{1}{N}\sum_{xy}\frac{I(x,y,t,c)\cdot y}{M(t,c)} \tag{3.5}$$

$$S_x(t,c) = \sqrt{\frac{1}{N\cdot M_x(t,c)}\sum_{xy}\left[I(x,y,t,c)\cdot(x-M_x(t,c))^2\right]} \tag{3.6}$$

$$S_y(t,c) = \sqrt{\frac{1}{N\cdot M_y(t,c)}\sum_{xy}\left[I(x,y,t,c)\cdot(y-M_y(t,c))^2\right]} \tag{3.7}$$

$$K_x(t,c) = \sqrt[3]{\frac{1}{N\cdot M_x(t,c)}\sum_{xy}\left[I(x,y,t,c)\cdot(x-M_x(t,c)^3\right]} \tag{3.8}$$

$$K_y(t,c) = \sqrt[3]{\frac{1}{N\cdot M_y(t,c)}\sum_{xy}\left[I(x,y,t,c)\cdot(y-M_y(t,c)^3\right]} \tag{3.9}$$

As is pointed out by the above equations, the center of gravity is calculated for both x- and y- components of Cartesian coordinate system or image. Hence, together with the statistics from color intensities, there are twenty seven numbers which represent each image. The figures 3.7, 3.8, and 3.9 provide a visual for center of gravity calculation process.

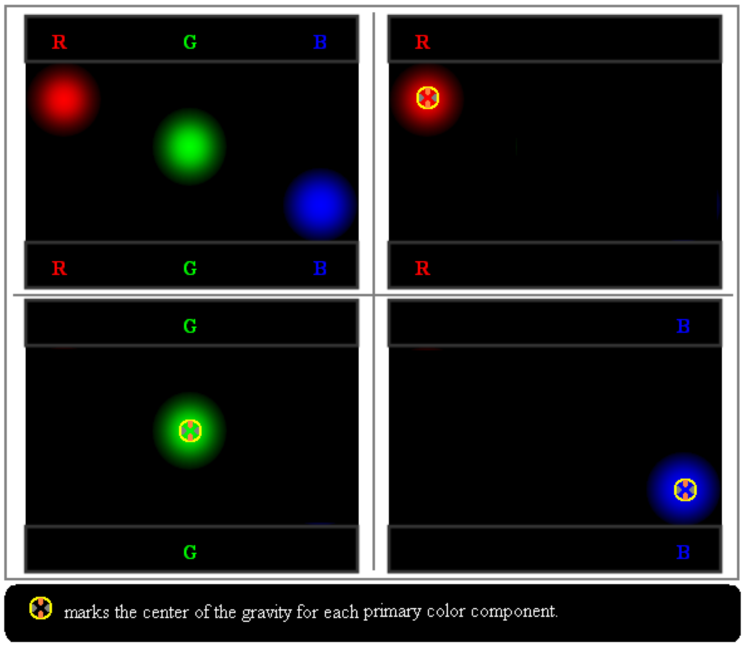

R
G
B
marks the center of the gravity for each primary color component.

Fig. 3.7. Represents the center of gravity, or weighted mean of position, for each of the three primary color components.

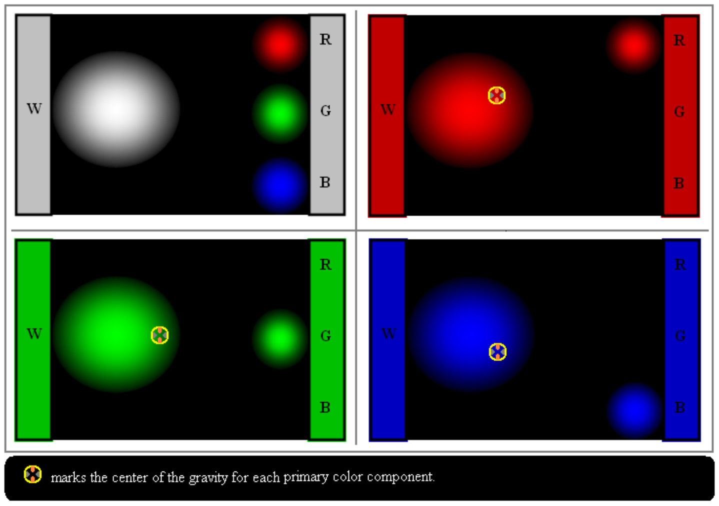

W
R
G
B
marks the center of the gravity for each primary color component.

Fig. 3.8. Points out that pure color white consists of equal amount of red, green and blue, and also the fact that center of gravity for each color component does not necessarily have to lie within the area with the highest intensity.

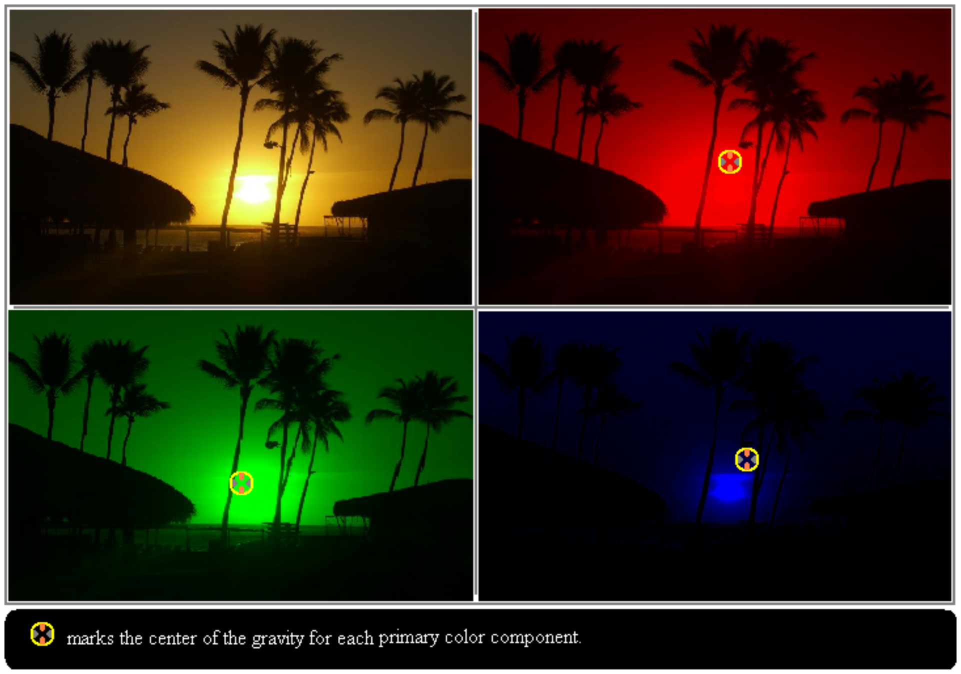

marks the center of the gravity for each primary color component.

Fig. 3.9. Illustrates the center of gravities for each of the three color components in real life picture.

Temporal First Order Difference of Statistical Data

The twenty seven values as described in the last section are referred to as *raw color moments*. These statistical values are dependent on color components. In other words, as can be seen in figure 3.10 if the first cut (during which values of red and green components decrease and value of blue component increases while moving from first shot to the second shot) is in examples set and the second cut (during which values of red and green components increase and value of blue component decreases while moving from second shot to the third shot) is in the input video stream the resulting fit value or *measure of difference* for these two cases will be lower than expected.

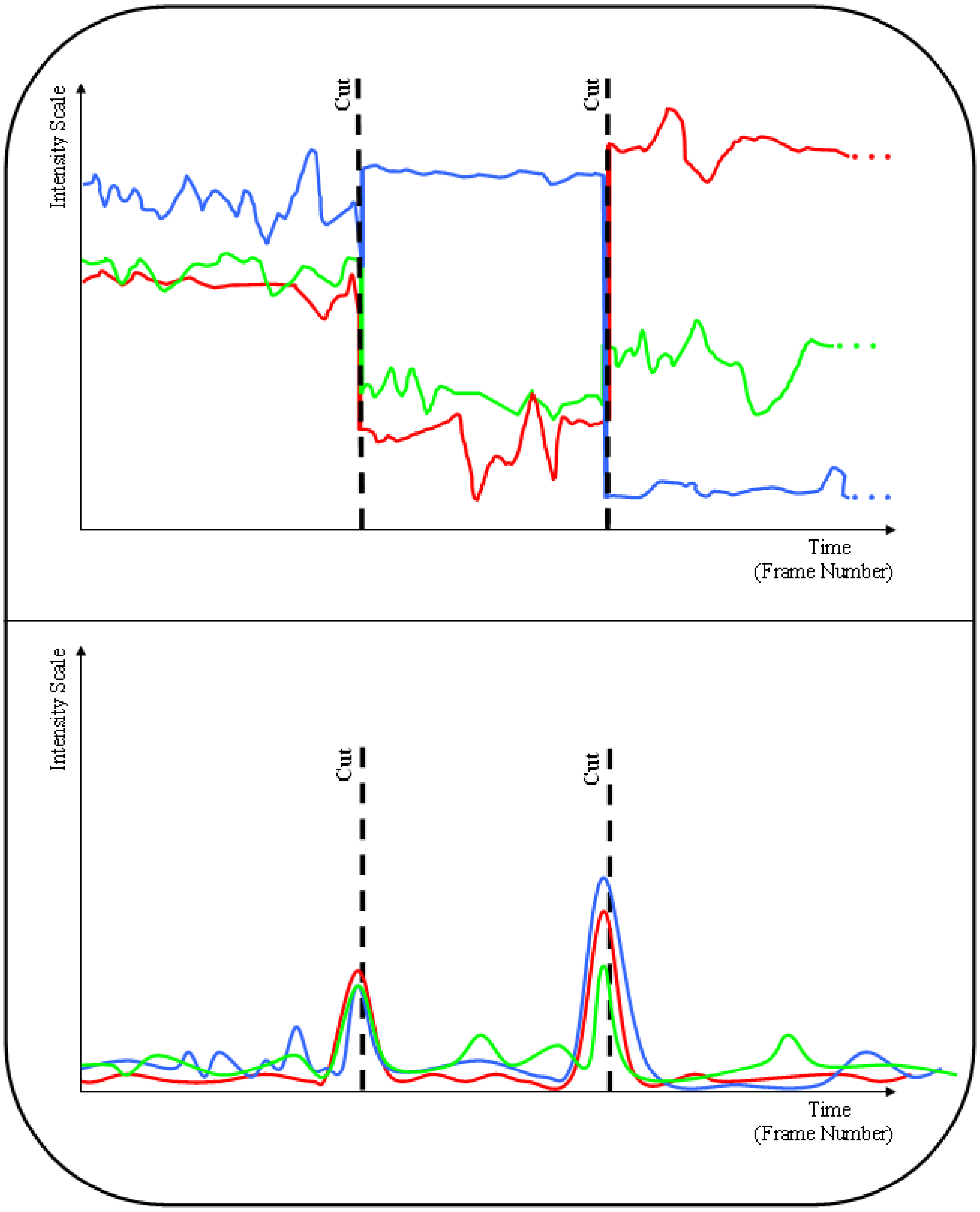


Fig. 3.10. (Top) demonstrate raw moments presentations for two variations of cut (during the first cut values of red and green components decrease and value of blue component increases while moving from first shot to the second shot. During the second cut values of red and green components increase and value of blue component decreases while moving from second shot to the third shot). (Bottom) demonstrates the first order derivative of raw moments of the top section.

To overcome this problem we suggest two solutions. The first solution is to assure the examples set contain all possible examples with all the possible cases of RGB band behavior (with each component increasing or decreasing). This approach improves the algorithm but more examples mean slower algorithm. Hence, the second method is more desirable.

The second solution is to use the temporal first-order difference (derivative) of statistical information. In our algorithm, raw moments are used along with the absolute value of the temporal first order frame by frame differences (or their first derivative) as basis for *measure of difference* calculation. First order derivative of *raw moments* are calculated by subtracting the statistical values of the current time instance, *t* from the values of the next time instance, *t+1* (Figure 3.11).

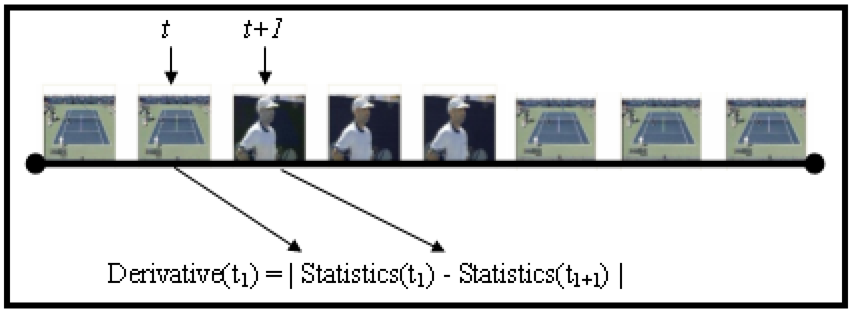


Fig. 3.11. Visualizes the process of calculating the first order derivative of raw moments.

Measure of Difference

As discussed in the last chapter, twenty seven values per frame (image) are used to represent the video in a numerical format. The next step is to measure the difference between transition examples and video input stream. These differences are then used as basis for *detection* and *classification*.

The following formula was used to calculate the *measure of difference* between the video input stream and each of the transition examples.

$$Diff = \frac{\alpha \cdot \left[\sum_{f=0}^{F} \left[\frac{\sum_{i=0}^{I} \left| m'_{f,i} - M'_{f,i} \right|^{p}}{I} \right] \right]^{1/p}}{F} + \frac{\beta \cdot \left[\sum_{f=0}^{F} \left[\frac{\sum_{i=0}^{I} \left| d'_{f,i} - D'_{f,i} \right|^{p}}{I} \right] \right]^{1/p}}{F} \tag{3.10}$$

where

$m_{f,i} =$ *moment i of the current frame from the input stream*
$d_{f,i} =$ *derivative of* $m_{f,i}$
$M_{f,i} =$ *moment i of current frame from the current example*
$D_{f,i} =$ *derivative of* $M_{f,i}$

$m'_{f,i} = m_{f,i} \cdot w_i$
$w_i =$ *weight used for moment i*
$d'_{f,i} = d_{f,i} \cdot w_i =$ *derivative of* $m'_{f,i}$
$M'_{f,i} = M_{f,i} \cdot w_i$
$D'_{f,i} = D_{f,i} \cdot w_i =$ *derivative of* $M'_{f,i}$

p is the power attribute; if a power of one is used then the minimum absolute error (MAE) is calculated. If a power of two is used then a minimum square error (MSE) is calculated. α and β can be used to control the effect of derivatives versus raw moments. *I* represents number of moments used in measure of difference calculation and *F* represents number of frames in the current example (window).

Unlike [64] which uses a transition synthesizer system to generate examples of the same size, our set of examples are of varied length. Hence, at each time instance, *t*, we analyze the image sequence in a dynamic-size sliding window of length *n* which starts from position *t* where *n* is the length of the current transition example in figure 3.12 the squares with dotted lines represent this sliding window.

During analysis the windows image sequence at each time instance *t* are compared against all examples in the examples set. For each window-example pair, a fit value is generated using the statistical features. For each time instance *t* the example with the best fit value is saved and then sent to our detection and classification systems.

This process results in about N X M matrix of fit values where N represents the number of frames per minute (normally 1800) and M represents the number examples. See the figure 3.13 for visual of these values. Figure 3.14 demonstrates the same fit values but in that figure the values are sorted for each window (frame). Figure 3.16 demonstrates the same fit values as figure 3.13. It also shows the best fit values graph and labels for some of the transitions.

Figure 3.15 displays only the best fit values for each window in the same minute as figures 3.12 and 3.13 In this case derivatives are used to calculate the measure of difference between examples and video streams and that is the reason behind high values near transitions. This fact becomes clearer when we explain why cuts go up, down and then up again.

Figure 3.17 explains why cuts follow a specific pattern in best fit value graph (figure 3.15). While reviewing figure 3.17, keep in mind that fit values are calculated by subtracting the derivative values of all the frames in the current example from the frames in the current video stream window.

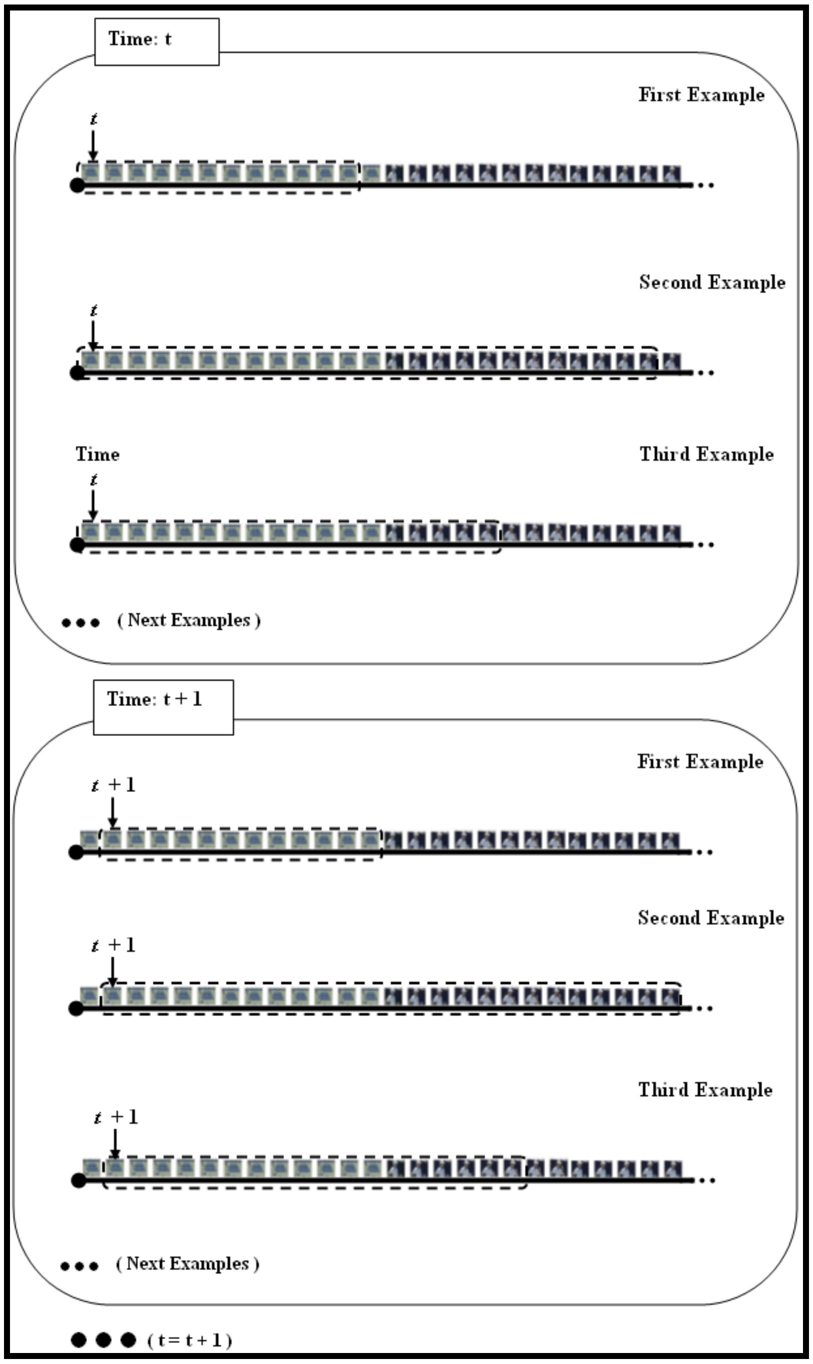


Fig. 3.12. Provides a visual for the process of calculating fit values by using variable length sliding windows.

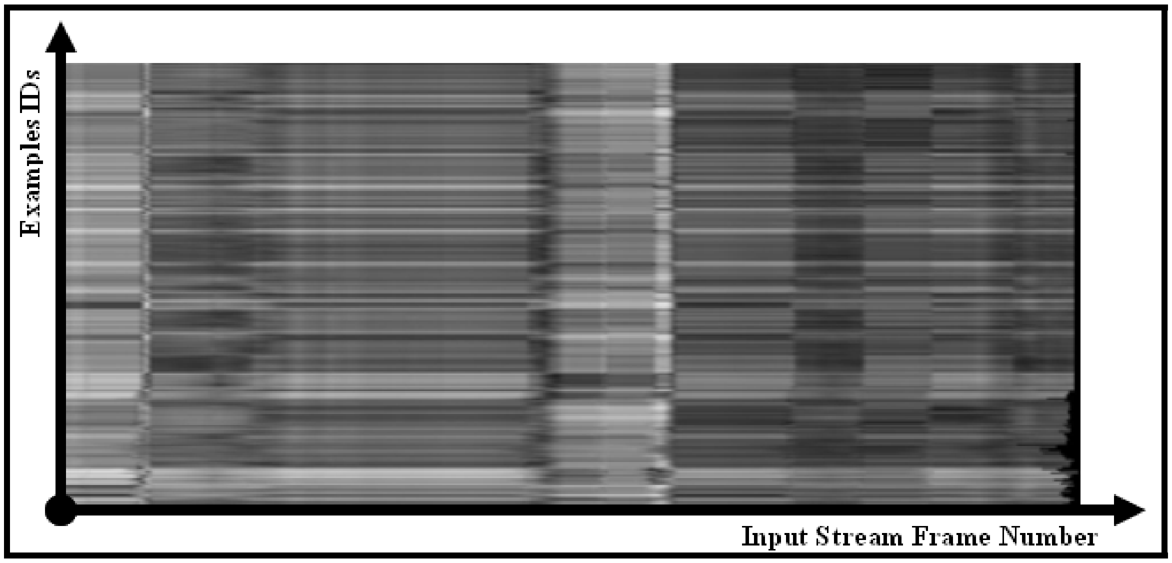


Fig. 3.13. The following settings were used in equation 3.10 to generate the above fit values matrix: Derivatives (raw moments weight was set to zero and derivatives weight to one), Original data (versus normalized data), Power of one, Moment weights of one, Unsorted

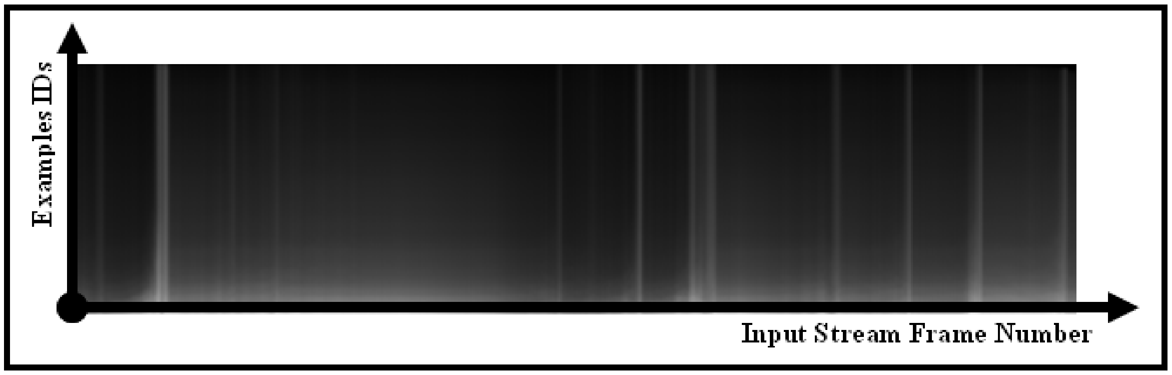


Fig. 3.14. The following settings were used for equation 3.10 to generate the above fit values matrix: Derivatives (raw moments weight was set to zero and derivatives weight to one), Original data (versus normalized data), Power of one, Moment weights of one, Sorted (for each input stream frame the example fit values were sorted).

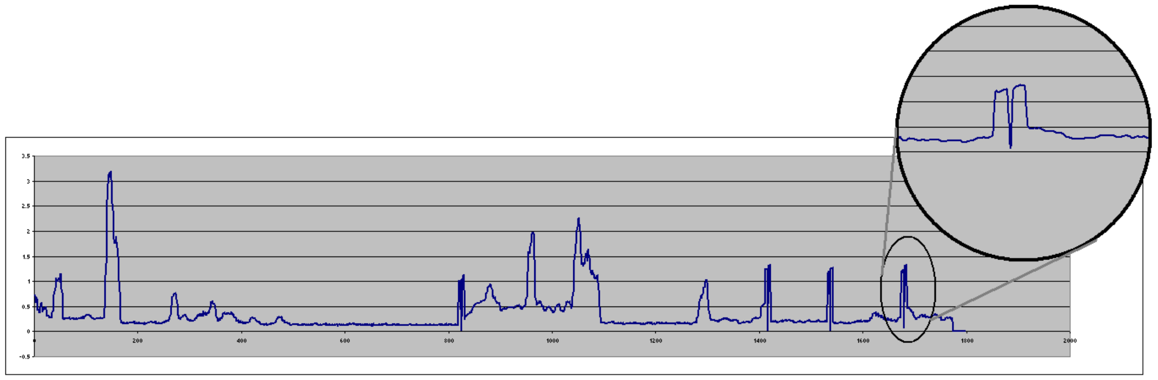


Fig. 3.15. Represents the best fit values for each window for one minute of input data

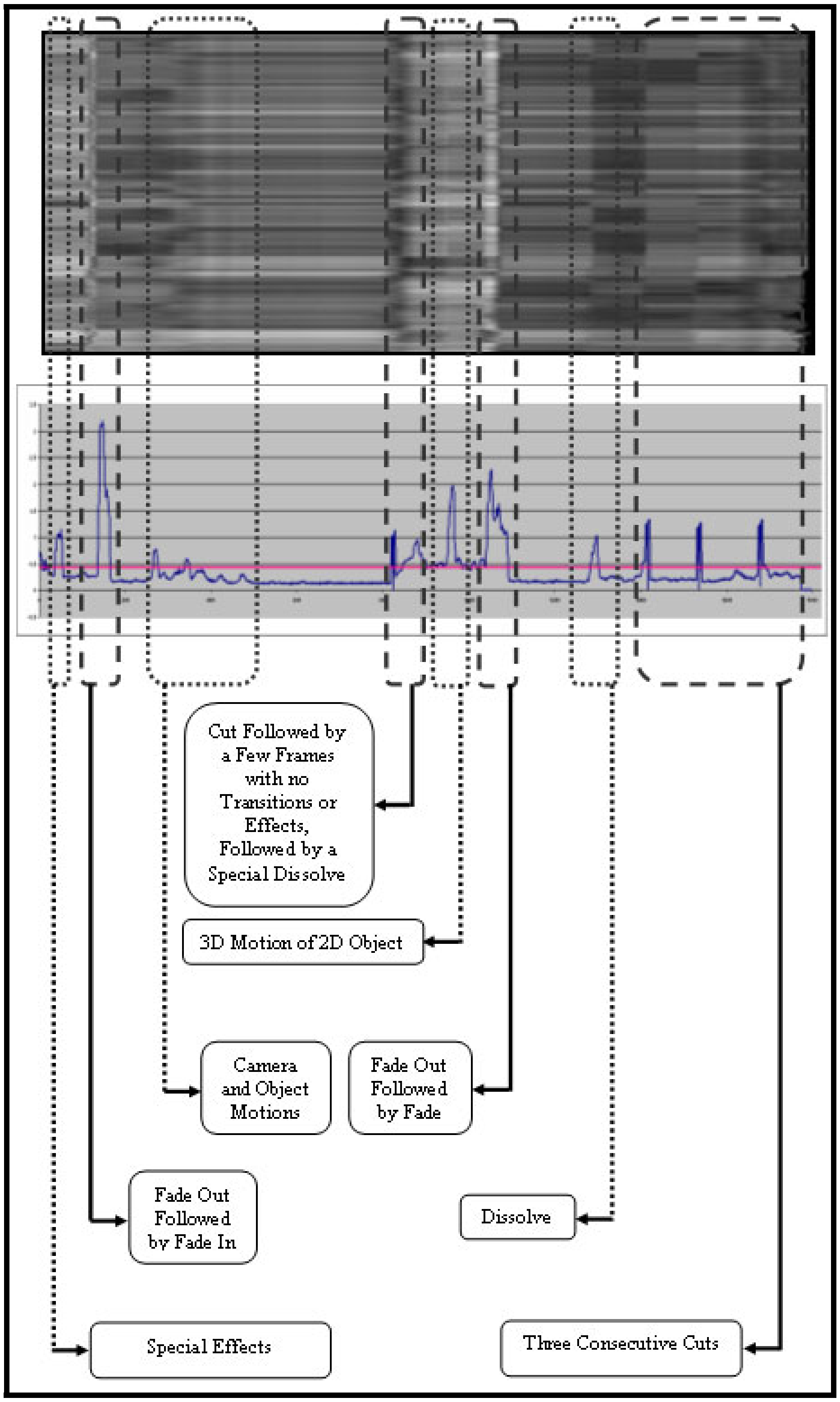


Fig. 3.16. The fit values image and the best fit values graph with labels for the transitions and effects.

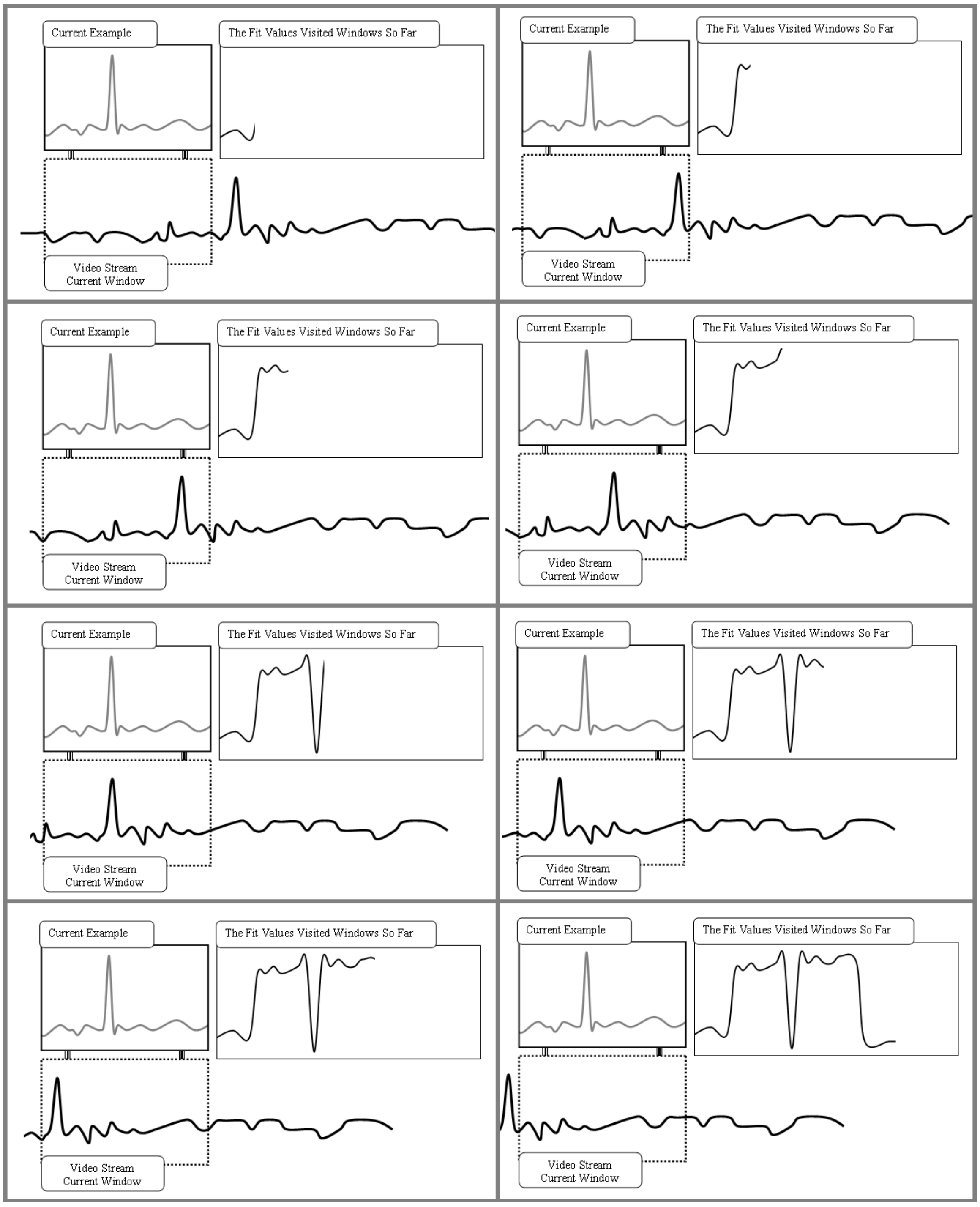

Current Example
The Fit Values Visited Windows So Far
Video Stream Current Window

Fig. 3.17. Illustrates why cuts follow a specific pattern in best fit value graph of figure 3.15. This figure presents a step by step visual for calculating the fit values (measure of difference) for a cut example and the sliding window as it moves across a cut.

As the last step for data representation, a common method is used to normalize the data. This method is described in details in the succeeding section.

Data Normalization

The image sequences in the examples and video streams may have a very different range of feature values (i.e. different means, standard deviation and skew). This can cause calculation problems or difficulties later on. To correct for this, we normalize raw moments (derivatives do not need to be normalized) by linearly scaling both set of features by using equation 3.11.

$$m'_{f,i} = \overline{m}'_{f,i} + \frac{\sigma'_{f,i} \cdot \left(m_{f,i} - \overline{m}_i\right)}{\sigma_i} \tag{3.11}$$

where

- $m'_{f,i}$ = The moment *i* after normalization for frame number *f*
- $m_{f,i}$ = The moment *i* before normalization for frame number *f*
- $\overline{m}'_{f,i}$ = The desired mean for moment *i* and frame number *f*
- $\overline{m}_i$ = The overall mean for moment *i*
- $\sigma'_{f,i}$ = The desired standard deviation for moment *i* and frame *f*
- σ_i = The overall standard deviation for moment *i*

As pointed out by equations 3.11 to linearly normalize the distribution curves first the mean needs to be moved (which moves the distribution curve) and then the standard deviation is changed (which modifies the range of data). In this equation, values of $\overline{m}'_{f,i}$ and $\sigma'_{f,i}$ are defined by the researchers. In our algorithm, the mean was moved to zero and the standard deviation was moved to ten. $\overline{m}_i$ and σ_i were calculated using as much input video stream as possible. These four values preferably should be the same for both examples and video stream normalizations.

This process gives both sets of features in examples and video stream the same mean and standard deviation values. Figure 3.18 shows the distribution curves for an arbitrary data set before and after normalization.

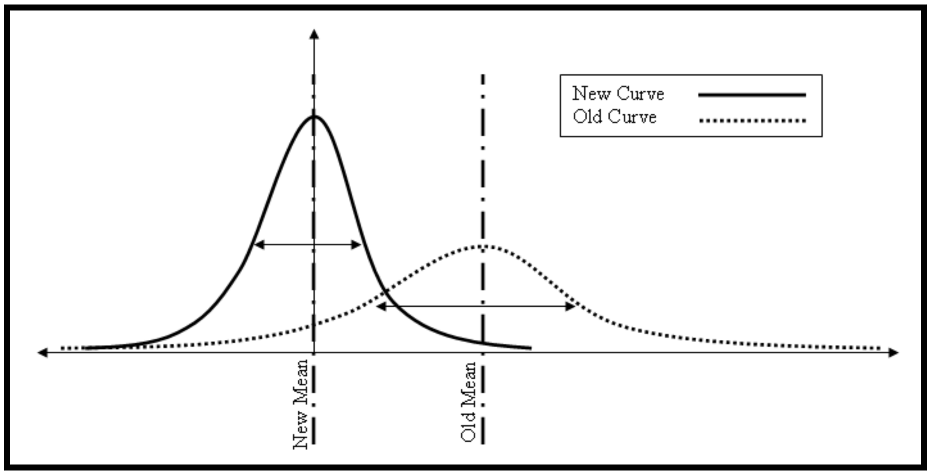


Fig. 3.18. Illustrates the distribution curves for arbitrary data set before and after normalization

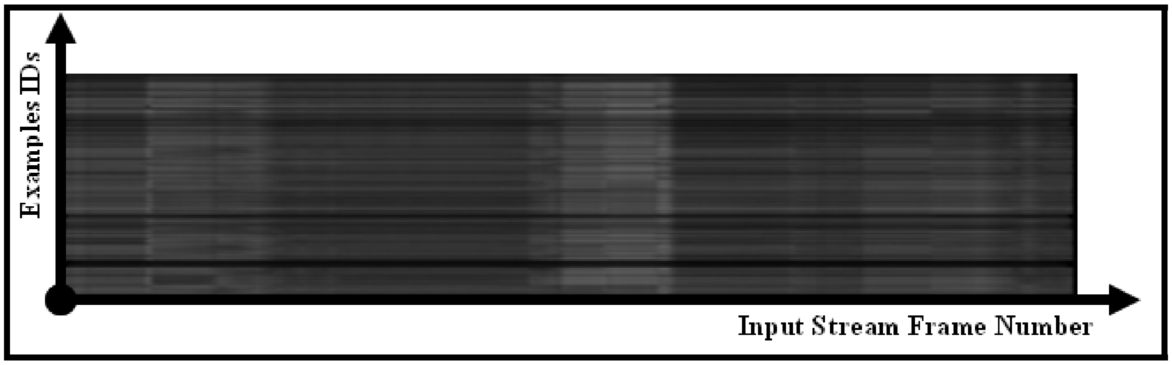


Fig. 3.19. Demonstrates the fit values image before normalization

Figure 3.19 represents the fit values matrix (image) before normalization using power of one and derivatives to calculate the fit values. Figure 3.20 on the other hand is the same representation after normalization.

Figure 3.20 represents the data after normalization. These data produced a brighter image since the fit values have a smaller range.

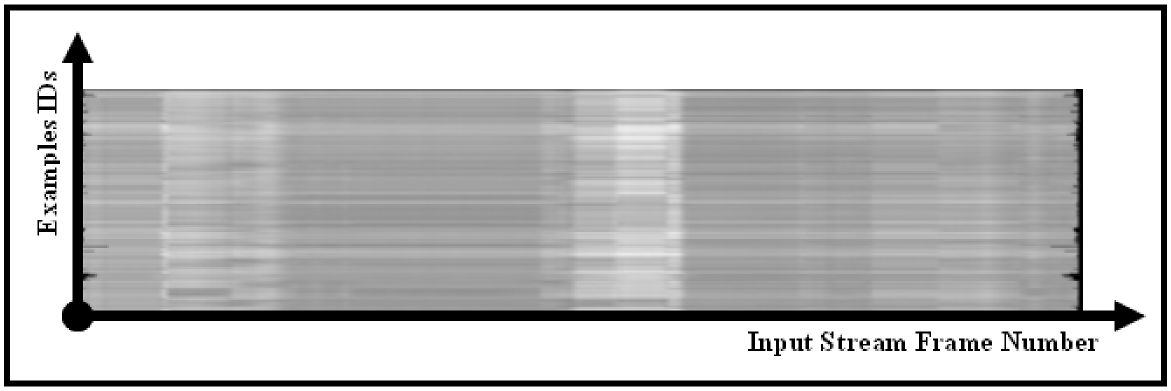


Fig. 3.20. Demonstrates the fit values image after normalization

This section as described the normalization technique which was utilized in this research. Although many other normalization techniques do exist, they are not used here and hence are not discussed at this time.

Detection and Classification

As described previously the detection and classification steps are combined. In this section, we discuss these two steps in details.

Detection

Detection refers to the process of data analysis leading to the discovery of regions of interest corresponding to the potential candidates for video shot boundaries.

Many *detection* techniques were reviewed in chapter two. In this approach, we decided to use a local adaptive threshold to find the regions of interest. This threshold is simply is based on the mean and standard deviation of the current window as well as the mean of the current minute. See equation 3.12.

The *detection* consists of two stages. In the first stage, for each window in the video stream the best example (the one with the smallest fit value depending on the data) is selected. Then in the second stage, regions with minimal activities are discarded by using an adaptive localized threshold.

$$threshold = \overline{m}_{f,i} + K \cdot \sigma_{f,i} \tag{3.12}$$

where i is the moment index (0 to 27) , K is the factor used to modify the threshold and

- $\overline{m}_{f,i}$ = the mean of fit values for the frames in each window with starting frame f
- $\sigma_{f,i}$ = the standard deviation of fit values for the frames in each window with starting frame f

The goal was to have an adaptive threshold that discards majority of uninteresting frames which does not introduce any false negatives and/or a high number of false positives at all the same time. However even usage of a localized adaptive threshold proved to introduce false negatives and also required adjustments for different types of video (depending on the degree of the noise and number of transitions/effects per twenty frames). Figure 3.21 shows an example of a cut that is missed even if an adaptive localized threshold is used. Even though the graph shows a distinguishable jump from the first shot to the second, the magnitude of the change is still low enough to lead to confusion and mislabeling of this cut as motion or as in our case being discarded all together by the threshold.

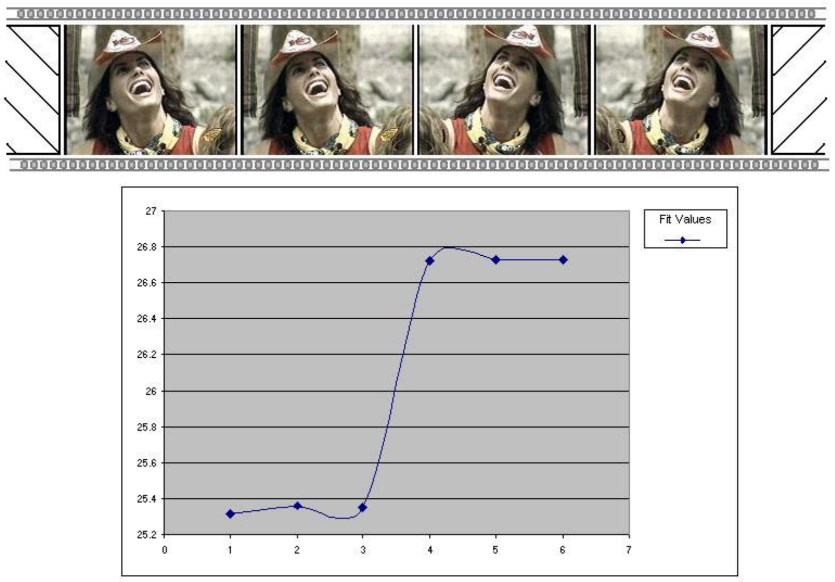


Fig. 3.21. Demonstrates the raw moments graph and the image sequences for a cut that was missed during detection process.

Classification

In our algorithm classification is part of the detection process. When we find the best example at time t, the transition/effect type (label) of this example is assigned to the current window. Thus, no additional work is necessary to classify the potential transition/effect in question. This is a major benefit of this approach.

After the detection and classification stages many problems will still stay unsolved. For example the size of transitions assumed to be the same as the same as the length of the best example match which is not necessary true. The second problem arises since a threshold technique is used. Because of threshold there can exist multiple detections for one transition. To avoid these problems another step is proposed, optimization.

Optimization

Optimization is the last step in temporal video segmentation based on direct predefined examples. In this step the location of the transition, length of the best fit example and other gathered data is used to calculate a good estimate for the boundaries of the transitions.

Optimization is also used to clean up the final results. For example, as you can see in figure 3.22 it is possible a single transition to result in multiple detections. This can be detected and fixed with a simple algorithm which considers the distance of each transition with the neighboring transitions and also considers the activities in the surrounding environment.

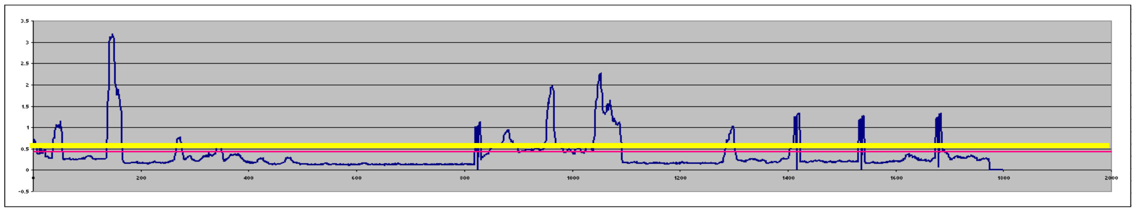


Fig. 3.22. Demonstrates an adaptive threshold localized for one minute of data.

The duplicate detections can be prevented by finding the local maximum for each group of frames which are labeled as regions of interest. Figure 3.22 shows the regions of interest above the global threshold (red line).

3.3 Conclusion

In this section, we discussed the first proposed novel algorithm for temporal video segmentation based on predefined examples. This approach combines *detection* and *classification* stages. The results and further discussions for this algorithm can be find in the experimental results chapter of this document. The next chapter, discusses another novel algorithm which is based on adaptive examples rather than predefined examples and it addresses the shortcomings of the algorithm discussed in this chapter.

Chapter 4

Direct Comparison based on Adaptive Examples

4. Direct Comparison based on Adaptive Examples

Previous chapter introduced the first approach for temporal segmentation of video based on examples. This chapter introduces another novel method which is designed to compensate for the shortcomings of the previous method.

4.1. Introduction

The first algorithm had many weaknesses which could not be easily directed without avoiding complexity. In the next chapter the issues which lead to these shortcomings are discussed. Due to these problems a new method is designed and implemented which has a higher performance than the previous method as well as many other algorithms which were discussed in Chapter 2.

This chapter reviews the design and implementation stages of the new technique, however the readers are recommended to review the design and implementation sections of the technique discussed in the previous chapter since the two algorithms share many similarities (specifically in *representation* section). Hence, some of the topics will no longer be discussed in this chapter.

4.2. Assumptions

All the definitions provided in this document such as sections on cuts, fades and dissolves of chapter 2 act as assumptions around which this research has been implemented. This section provides the list of assumptions used while implementing this method.

While dealing with dissolves and other gradual transitions, pinpointing the exact boundaries is difficult since even human eyes cannot distinguish between beginning and ending frames belonging to transitions and the adjacent frames of adjacent shots. Therefore in our algorithm and many algorithms described in chapter 2, as long as the detected transition overlaps with the actual transition, a match (true positive) is declared.

The first assumption is that fades are longer than three frames long. One frame long fades do not exist. Two and three frame long fades have either a sudden change (figure 4.1) similar to cuts or are incomplete fades (change in brightness). Hence three is the smallest window that is used for fades.

The first assumption on cuts is that if a cut is immediately succeeding or preceding another gradual transition (namely fade) then the two transitions are counted as one. The goal should be to detect either one of the two (more desirably the fade). This is being treated as a special case of fade (figure 4.2).

Another interesting and fairly common transition is a cut followed by a series of monochrome frames which is also followed by another cut. If there are less than five monochrome frames in between the two cuts then this sequence is considered as one transition (i.e. at least one of cuts have to be detected). If the monochrome shot is longer then it is treated as a normal shot (figure 4.3). Similar logic is followed if there exist two cuts in very close proximity which are separated by a short shot rather than monochrome frames.

In many cases there exists a sudden zoom such as the one in figure 4.4. This effect is considered as a cut since the camera has made a sudden transition and it looks as intermediate frames have been moved.

If a graphical transition such as the one in figure 4.5 (frames 1644 to 1652) is detected as a fade depending on how well it resembles an actual fade then it can be marked as a false positive. The one in that figure will not be labeled as such since it is very similar to an actual fade.

In many shots there exist situations such that the text on the screen changes. For example the text fade, dissolve or suddenly changes to another text (figure 4.6). If the text is part of a bigger picture then it is not suppose to be detected as a transition. However if it is the main object in the shot without any other object (for instance with white background) then since it follows the definition of corresponding transitions, it will not be labeled as false positive if detected as a transition.

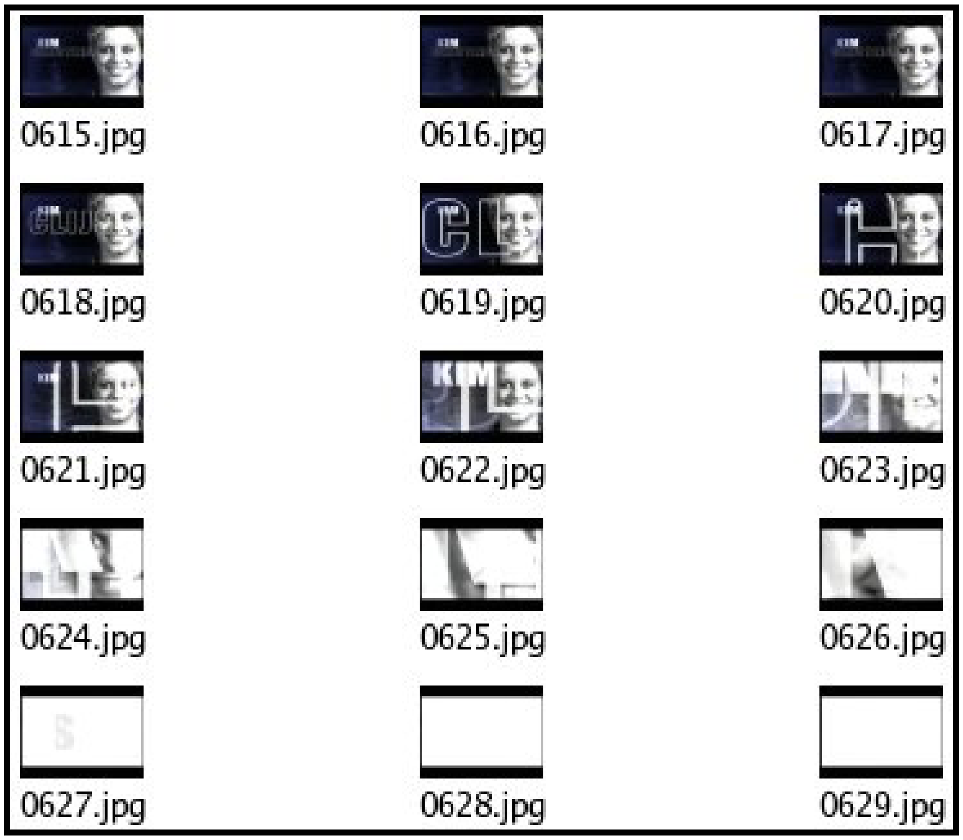


Fig. 4.1. Illustrates an unacceptably short fade which is succeeding a special effect.

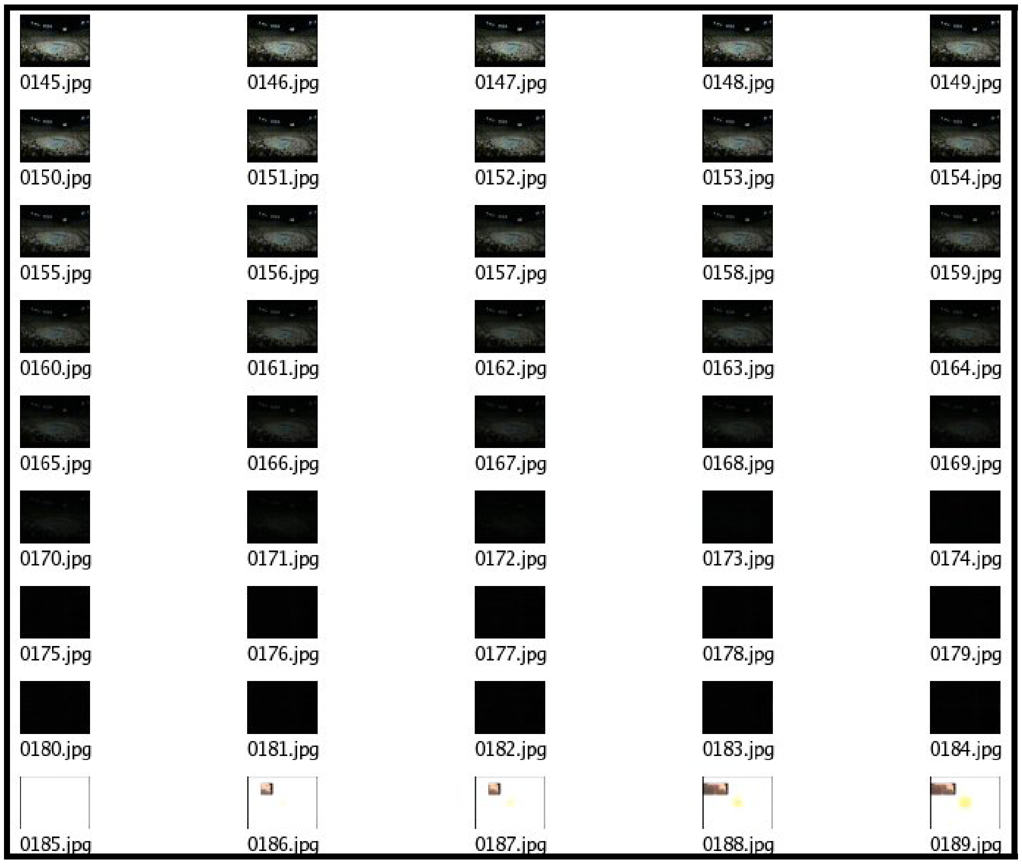

0145.jpg 0146.jpg 0147.jpg 0148.jpg 0149.jpg
0150.jpg 0151.jpg 0152.jpg 0153.jpg 0154.jpg
0155.jpg 0156.jpg 0157.jpg 0158.jpg 0159.jpg
0160.jpg 0161.jpg 0162.jpg 0163.jpg 0164.jpg
0165.jpg 0166.jpg 0167.jpg 0168.jpg 0169.jpg
0170.jpg 0171.jpg 0172.jpg 0173.jpg 0174.jpg
0175.jpg 0176.jpg 0177.jpg 0178.jpg 0179.jpg
0180.jpg 0181.jpg 0182.jpg 0183.jpg 0184.jpg
0185.jpg 0186.jpg 0187.jpg 0188.jpg 0189.jpg

Fig. 4.2. Illustrates a fade followed by a cut.

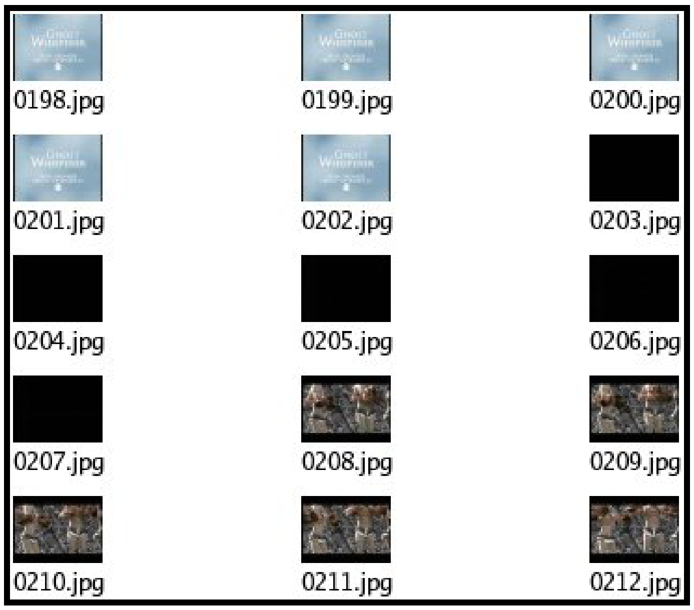

0198.jpg 0199.jpg 0200.jpg
0201.jpg 0202.jpg 0203.jpg
0204.jpg 0205.jpg 0206.jpg
0207.jpg 0208.jpg 0209.jpg
0210.jpg 0211.jpg 0212.jpg

Fig. 4.3. Illustrates a cut followed by monochrome frames followed by another shot.

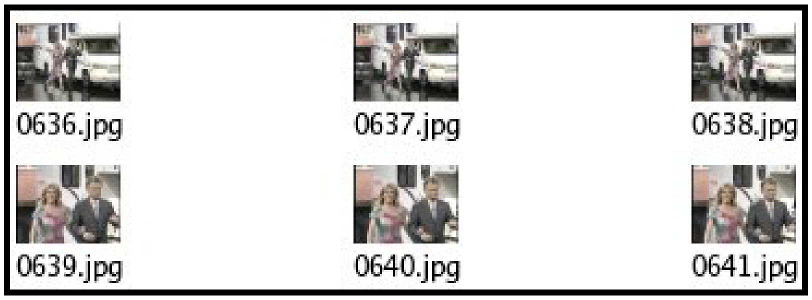


Fig. 4.4. Illustrates a shot which contains a sudden zoom.

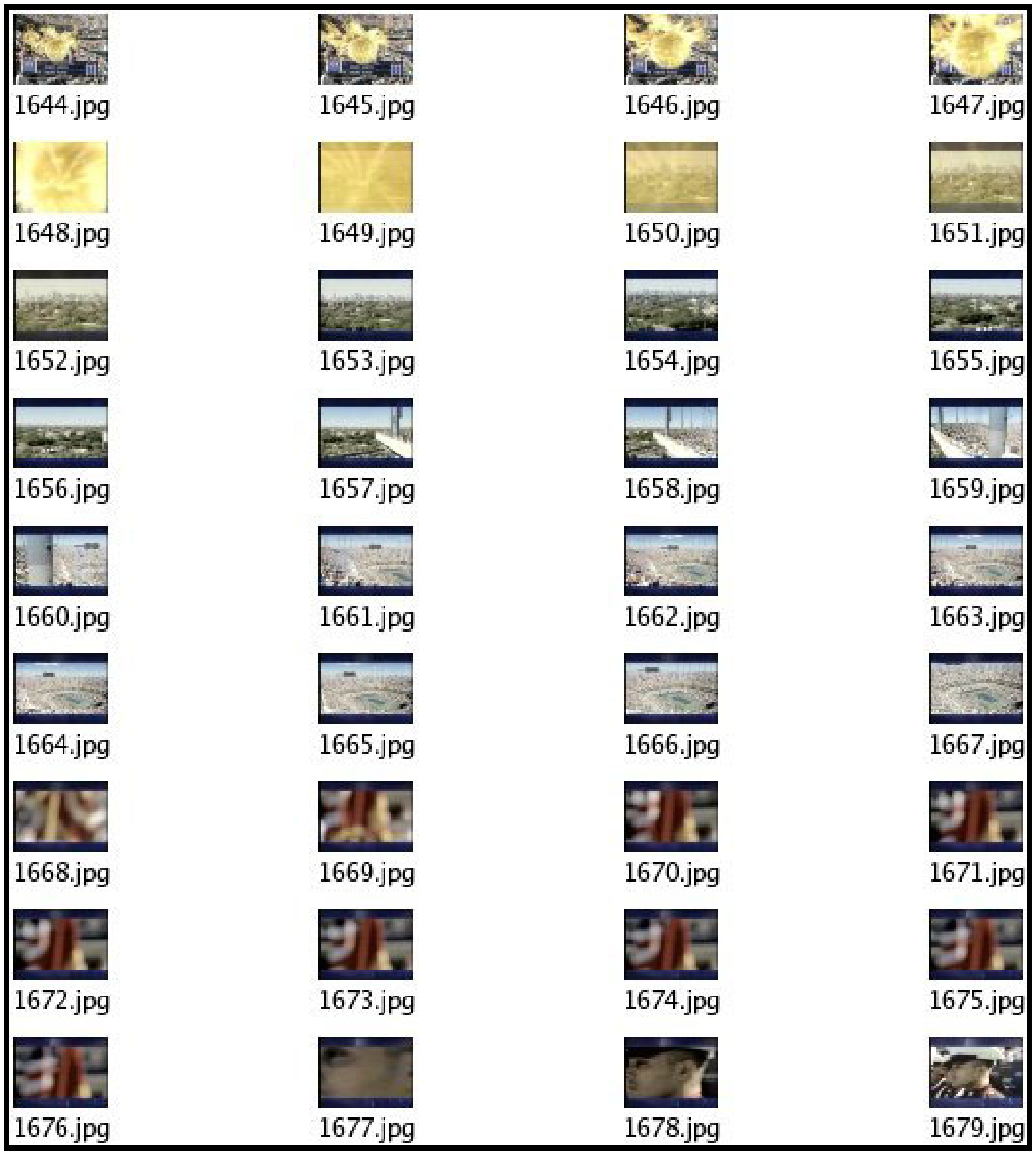


Fig. 4.5. Illustrates a region of high activity as well as a graphical transition which is a potential false positive for fade detectors.

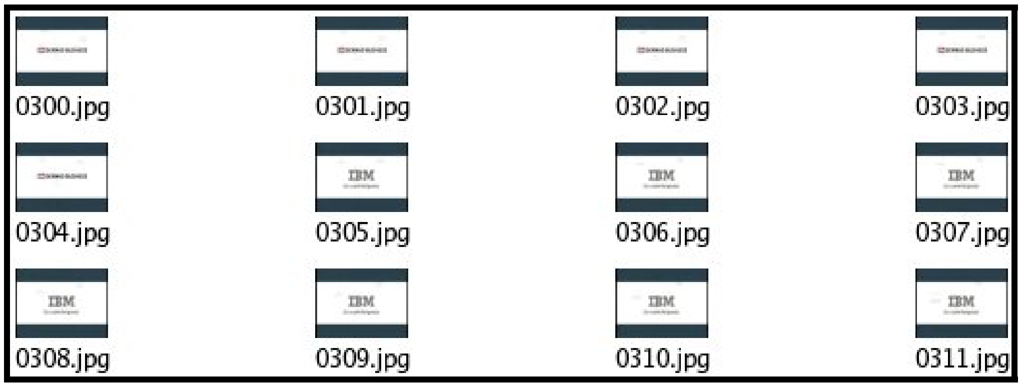


Fig. 4.6. Illustrates a sudden change in text.

4.3. Goals

The goals behind the different algorithms in this research are to achieve the following properties:

- **Real Time** – This program is designed in a way to minimize the execution time, so data can be processed as quickly as it is generated. In other words, when analyzing one minute of data the execution time does not exceed one minute.
- **Generality** – One of the main goals of this research was to create a general technique free of any specific models or algorithms which can be easily extended to other transitions or effects.
- **No Threshold –** The idea of no threshold is one of the center pieces of this research. Not having a threshold will remove the problem of threshold selection and should decrease the number of false negatives by a great number.
- **Multilevel Property** – This property means a hierarchical structure is used during design and implementation stages. In a multilevel system, each stage takes the

input of previous stage and after analysis it forwards its own output along with its inputs to the next stage. Multilevel property allows to easily incorporating other data representation or methods into different stages of algorithm. Also as algorithm progresses there will be more data for analysis available to the future algorithm.

4.4. Design

The first step in design is to identify the problems with the previous method. In the first method, the idea was to use predefined examples set containing a large number of examples. The problem is that having a large enough examples set which contains all the possible variations of transitions and effects can drastically effect the performance of algorithm (mainly in terms of time). The main advantage for having large number of examples is that it removes majority of false negatives. However having many examples of transitions will also introduce many false positives since the probability of examples matching with motions or areas of higher activity (with no transition) rises.

Hence, rather than using predefined examples, a new method based on adaptive examples is introduced. Adaptive examples are based on the context in which transition has occurred. This fact eliminates the need for a large number of predefined examples, quality dilemma and the trade off between quality and performance. Figure 4.7 illustrates the process flow diagram for this algorithm.

At the first sight the second algorithm is very similar to the previous method however closer look reveals otherwise. The representation stage in the new method does not need any predefined examples preparation and instead before calculating the measure of difference it will generates adaptive examples.

Since the new method only works on raw moments, the need for calculating first order derivatives is eliminated. Also, normalization is not necessary for the new method since adaptive examples are used.

Another change in the new process flow is that the detection process is shorter since the new method does not use any thresholds (a great advantage over other techniques). The detection is basically done by introduction a new group or *normal* group. This group represents the frames within the shots in contrast to frames within transitions (neighboring a cut or within gradual transitions). The classification and optimization stages remain the same as the previous method. The next section discusses above topics in more details.

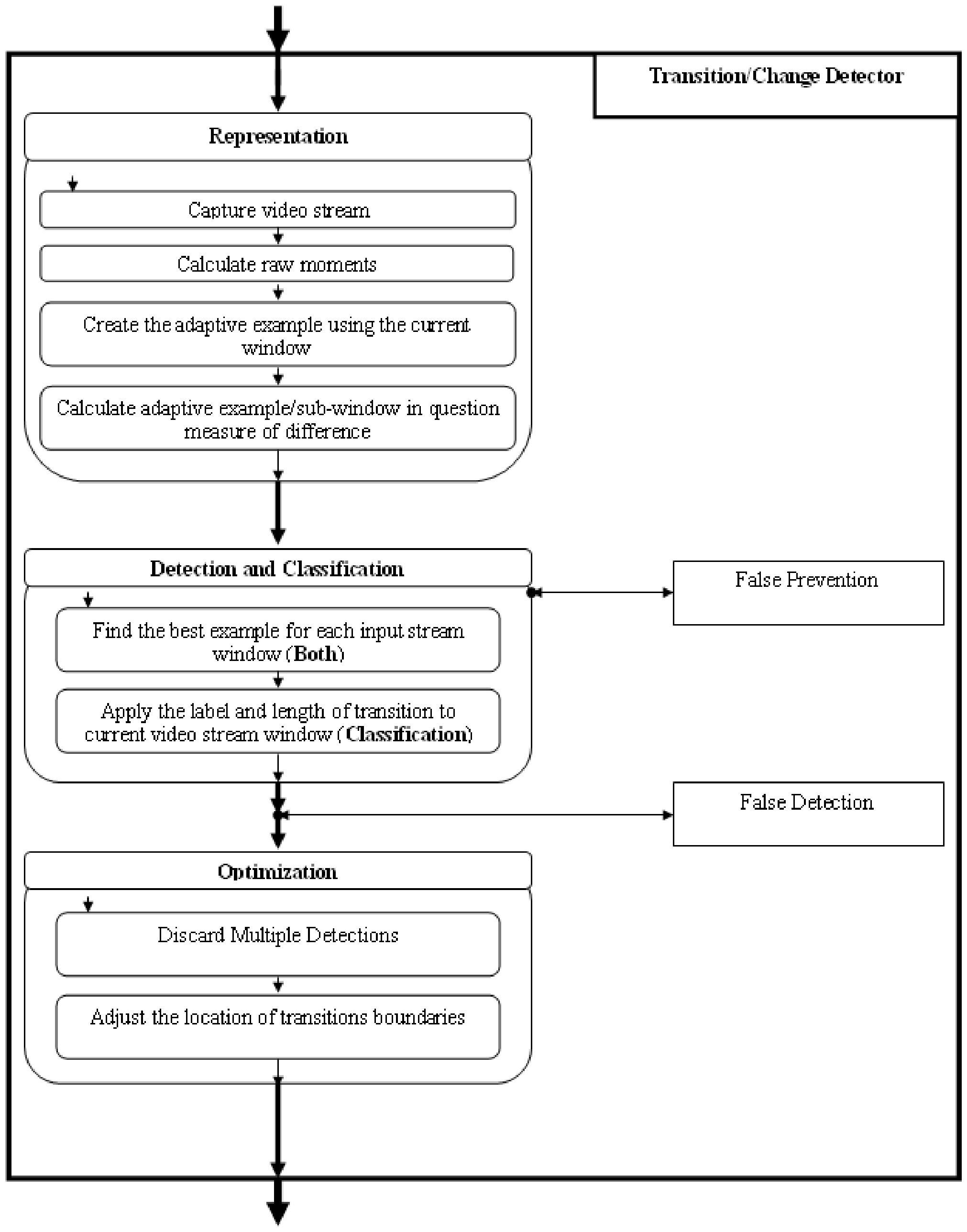

Transition/Change Detector
Representation
Capture video stream
Calculate raw moments
Create the adaptive example using the current window
Calculate adaptive example/sub-window in question measure of difference
Detection and Classification
Find the best example for each input stream window (Both)
Apply the label and length of transition to current video stream window (Classification)
False Prevention
False Detection
Optimization
Discard Multiple Detections
Adjust the location of transitions boundaries

Fig. 4.7. Illustrates the process flow for main component of the algorithm based on adaptive example or transition/change detector.

4.5. Detectors Implementation Details

4.5.1. Representation

Representation step is consisted of capturing the video stream, calculating the statistical information (raw moments), calculating the adaptive examples and finally calculating the measure of difference between each of the adaptive examples and each of the video streams windows. The first two topics are not discussed in this chapter since they have been discussed in details in the previous chapter representation section. The remaining topics are discussed later on in this section.

In the previous algorithm, the examples required to be predefined. Doing so introduces problems which can be partially addressed through normalization, addition of many new examples and using many other algorithms in parallel but not only these solutions will not help to fix the problem fully rather they also introduce other problems such as slowing down the algorithm drastically and resulting in an unacceptable performance level (unacceptable number of false positives). Hence, a novel solution is needed and adaptive examples are the major players in the suggested solution.

The previous algorithm requires covering all the possible variety of transitions and effects in order to avoid introducing large number of false negatives. On the other hand, a large number of examples introduce many false positives and will drastically increase the execution time (in the new algorithm the only variation needed is examples of different size while detecting gradual transitions).

The solution to this dilemma is adaptive examples. They, not only provide an adapted example for each instance of sliding window, but also remove the need for having tens of thousands of examples. This way, both quality of detection (lower number of false negative and positives) and the execution time improve considerably.

Adaptive Examples

Similar to the previous method, the new algorithm is based on sliding window idea. For each frame (except the starting frames) there is one fit value per window. Depending on the type of transition different sizes of sliding window are used and partitioned differently. Some of the partitions are used to create the adaptive example and the rest are used as potential candidates for transitions. In the future sections we discuss how adaptive examples are created for each of the primary transitions and for normal (or examples of no activity or transition) groups. Figure 4.8 demonstrates a potential candidate and its corresponding generated adaptive example.

One main factor in the new algorithm is that either varied length windows are examined or one fix length is required for all type of transition. This fact eliminates the window size as a parameter. Hence, this algorithm not only has no thresholds it is also parameter free.

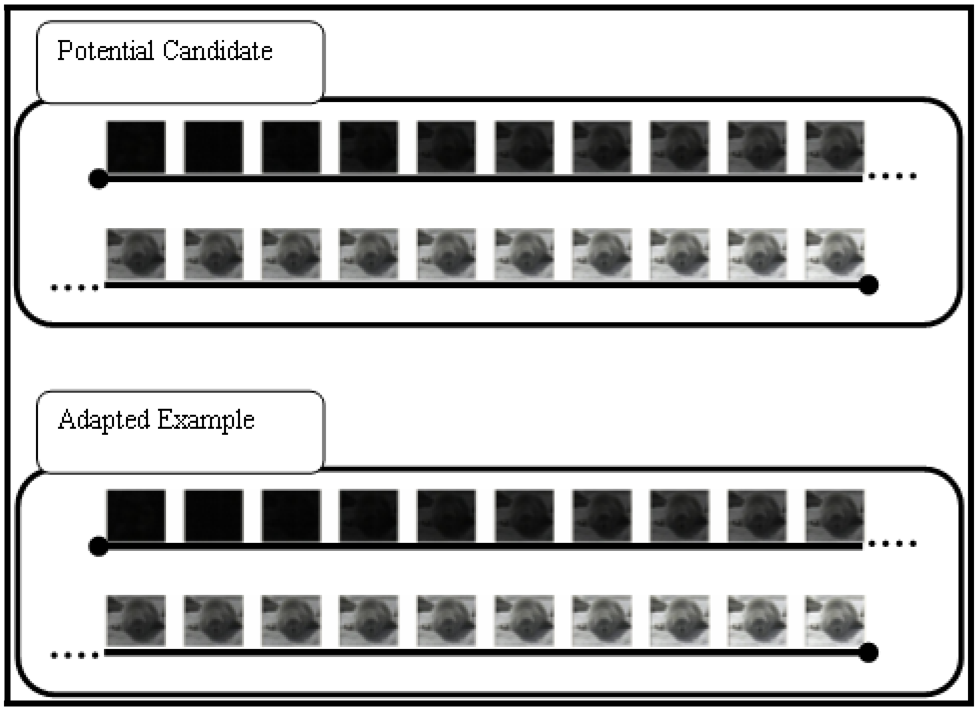


Fig. 4.8. Illustrates a selected potential candidate and its corresponding generated fade adapted example.

Cut

As is demonstrated in figure 4.9 the window is divided into three partitions which are labeled, A, T, and B while generating the adaptive example and calculating the measure of difference for cuts. Note that A and B have to be half of T in size since they are merged to create the adaptive example and then compared against T frame by frame to calculate the measure of difference.

The adaptive example was first produced by simply merging partitions A and B. Since the adapted example is generated using the surrounding context of T, it matches the potential candidate very closely for all twenty seven moments.

In cuts all twenty seven moments are used since the adaptive examples can easily be generated by merging the moment value stream from A to the value stream from B partition.

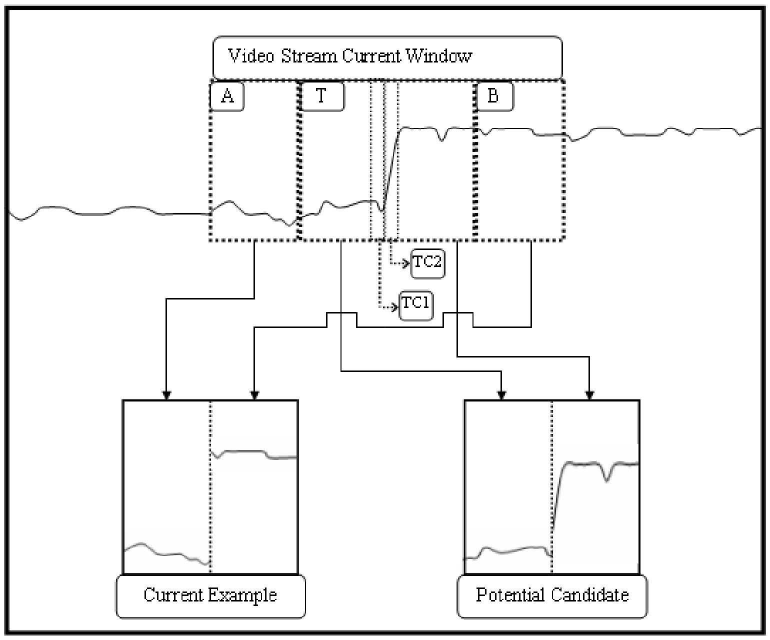


Fig. 4.9. Illustrates the process of extracting potential candidate and generating a cut adaptive example while T partition of the window is centered on a cut transition.

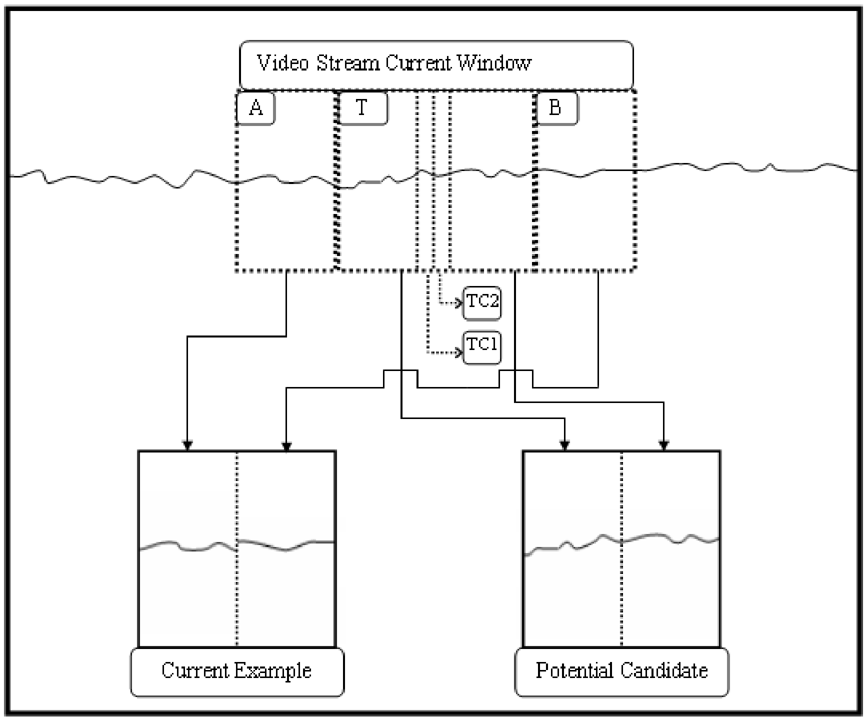


Fig. 4.10. Illustrates the process of extracting potential candidate and generating a cut adaptive example while T partition of the window is over a region of no activity (regions containing minor object motions).

Figure 4.10 represents the same settings as figure 4.9 with the exception of sliding window been located over a normal group rather than a cut. These figures are discussed further in the upcoming *measure of difference* section. Also discussed in that section is the complications cut transitions have compared to gradual transitions due to not having any actual length. To address that problem a specialized technique is developed to distinguish cuts from normal frames.

Dissolve

As pointed out in chapter 2, dissolves are the most difficult transitions for detection (among the three primary transition types). This fact is mainly due to dissolve detectors mistakenly detect cuts, fades as well as other transitions and effects. Hence, the new algorithm is designed to first identify all the cuts and fades with a high degree of recall and precision and then it uses these results to distinguish dissolves.).

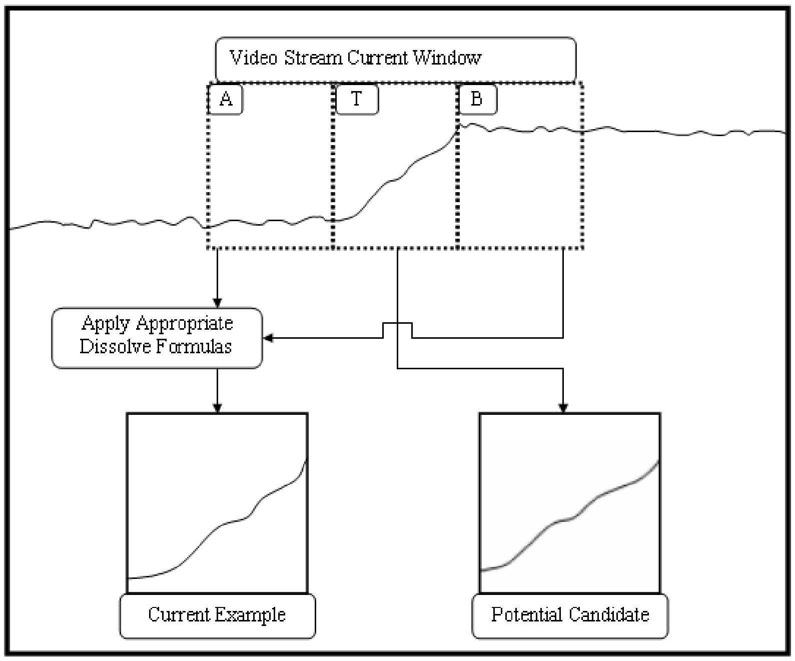

Video Stream Current Window
A
T
B
Apply Appropriate Dissolve Formulas
Current Example
Potential Candidate

Fig. 4.11. Illustrates the process of extracting potential candidate and generating a dissolve adaptive example while T partition of the window is centered on a dissolve transitions.

In dissolve detector, only the first six moments are used whereas all twenty seven are used for cuts. This is due to the fact that dissolve adaptive examples cannot be simply extracted by merging two partitions. New equations need to be derived for each type of statistical data (mean, standard deviation, and skew) which calculates the moment values of adaptive examples by using already calculated statistical data of A and B partitions.

This method is much faster than first creating synthetic dissolve examples and then extracting all the twenty seven moments. On the other hand, due to the complexity of mathematical formulas the skew cannot be derived and hence is not used in dissolve or fade detectors. The statistics from center of gravity are also not used since they are data dependent and hence cannot be defined.

Figure 4.11 illustrate the process of generating adaptive examples for dissolve detector. In this process, the window is divided into three equally sized partitions, A, T and B. A and B are used in equations 4.1 to 4.6 to generate the new values for adaptive examples. The equations 4.1 to 4.3 are used for calculating the first three moments (means for each of the primary colors). The second set of equations (4.4 to 4.6) are used for calculating the moments three through six (standard deviations of the primary colors.

$$M_{t,i,R} = \alpha \cdot \overline{A}_{t,i,R} + (1-\alpha) \cdot \overline{B}_{t,i,R} \tag{4.1}$$

$$M_{t,i,G} = \alpha \cdot \overline{A}_{t,i,G} + (1-\alpha) \cdot \overline{B}_{t,i,G} \tag{4.2}$$

$$M_{t,i,B} = \alpha \cdot \overline{A}_{t,i,B} + (1-\alpha) \cdot \overline{B}_{t,i,B} \tag{4.3}$$

$$\sigma_{t,i,R} \approx \sqrt{\alpha^2 \cdot \sigma^2_{A_{t,i,R}} + (1-\alpha)^2 \cdot \sigma^2_{B_{t,i,R}}} \tag{4.4}$$

$$\sigma_{t,i,G} \approx \sqrt{\alpha^2 \cdot \sigma^2_{A_{t,i,G}} + (1-\alpha)^2 \cdot \sigma^2_{B_{t,i,G}}} \tag{4.5}$$

$$\sigma_{t,i,B} \approx \sqrt{\alpha^2 \cdot \sigma^2_{A_{t,i,B}} + (1-\alpha)^2 \cdot \sigma^2_{B_{t,i,B}}} \tag{4.6}$$

where $M_{t,i,*}$ and $\sigma_{t,i,*}$ represents the generated mean and standard deviation for adaptive examples and *R*, *G*, *B* subscripts represent the three primary color channels, *t* represents the frame number (time) and *i* represents the current moment.

As reader can notice the derived equations (see Appendix A) for standard deviation are slightly differ from the equation used. The last expression is ignored since in a long term it will add up to zero since it can take both positive and negative values.

Fade

Fade is a specific variation of dissolve. Hence the dissolves equations (equations 4.3 to 4.6) are also used in fade detectors. Fades are generated by using a normal shot and a monochrome shot. If the monochrome frames are leading the frames from the normal shot then the transition is a fade out. Otherwise the transition is called to be a fade in. A complete fade is consisted of both of these sections as can be seen in figure 4.12.

Fade Out

Figure 4.12 demonstrates the process of producing a fade out adapted example. The window is divided to two parts, A and T. A is used to create the adapted example. So it has to be the same size as T.

To use the dissolve equations another shot is necessary. The other shot is created by using previously stored monochrome frames. It is generated by repeating the monochrome frames to create a shot which has the same length as partition A. The set of monochrome frames should contain frames of different colors (in our algorithm different varieties of white and black monochrome frames were used).

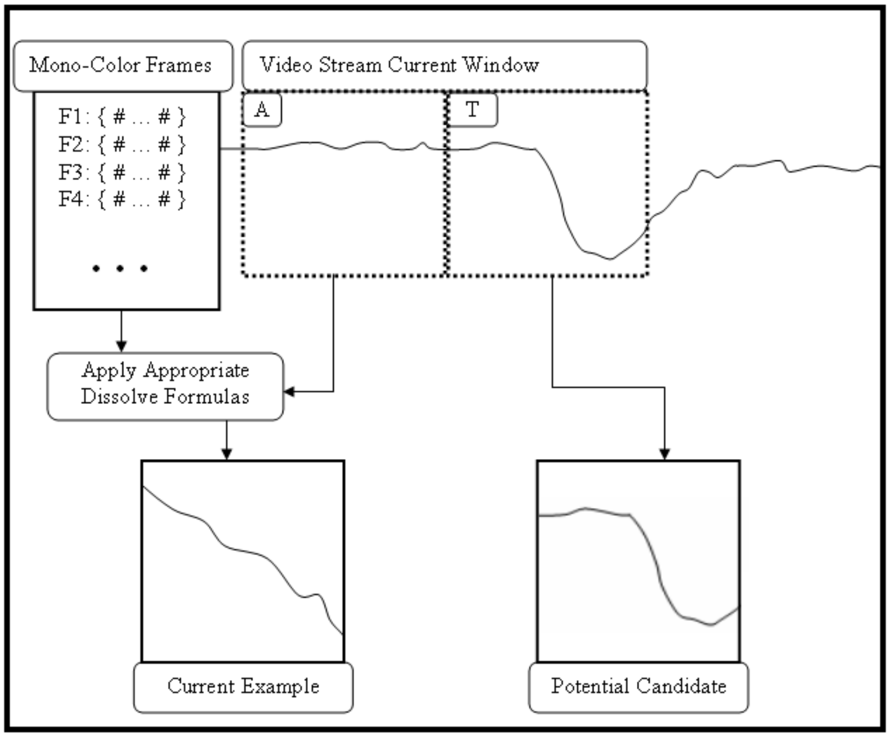


Fig. 4.12. Illustrates the process of extracting potential candidate and generating a fade out adaptive example while T partition of the window is centered on a fade out transition.

Fade In

The only difference between fade out and fade in is the order in which the monochrome frames appear. In fade in the monochrome shot leads the normal shot. Hence two modification to fade out algorithm make it possible to achieve desirable results for fade in using the same algorithm.

The first modification is to switch the order of windows. As can be observed in figure 4.13 instead of partition A, partition B is used (i.e. the partition used in example generation and partition T are swapped). This modification is needed to make sure the algorithm conform to the definition of fade in. Although to completely follow the definition of fade in the order of alpha had to also be changed. The variable alpha will be set so the effect of monochrome shot increases in time.

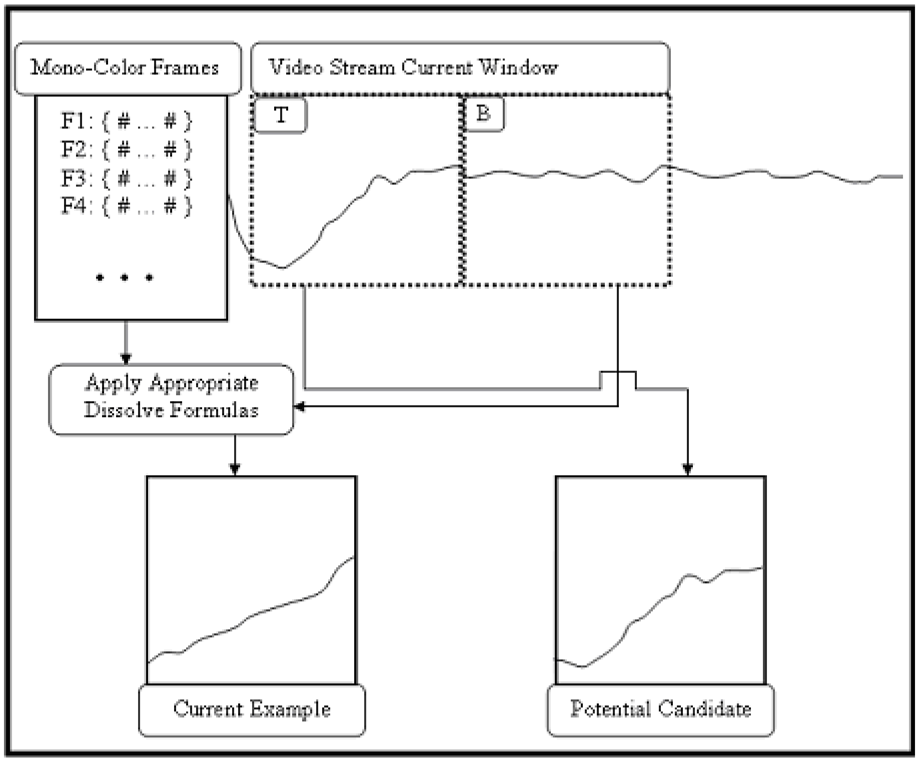


Fig. 4.13. Illustrates the process of extracting potential candidate and generating a fade in adaptive example while T partition of the window is centered on a fade in transition.

Normal

Introduction of normal groups is another center piece within the adaptive example itself. Having normal groups allows the algorithm to avoid use of thresholds and to implementation of a novel method which achieves high quality results.

The window is divided into A and T partitions. A is directly used for generating the example and T is used as the potential candidate. While generating adaptive examples for normal groups, large windows have to be avoided since the window has to be localized;

otherwise, it is possible for each window to contain more than one transition, leading to a higher number of false detections. At the same time, the window has to be large enough to detect any major changes in the image sequence of partition T. Also, A partition has to be the same size as T partition.

Generating the normal groups is rather simple. The task basically ends by dividing the window to A and T partitions (see figure 4.14). Partitions A and T are used directly to generate examples and to present the potential candidate correspondingly.

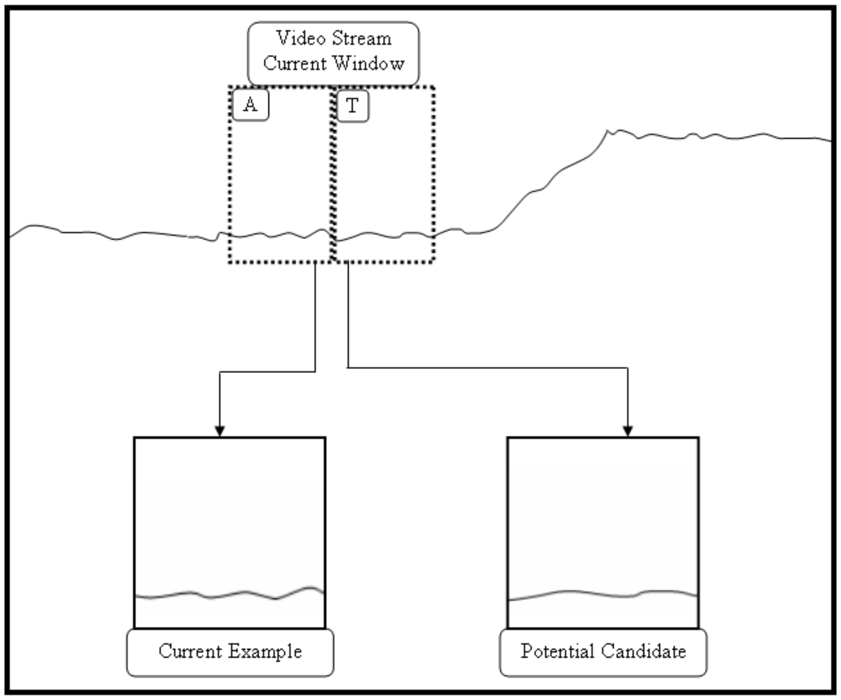


Fig. 4.14. Illustrates the process of extracting potential candidate and generating a normal adaptive example for gradual transitions detector while T partition of the window is over a region of no activity (regions containing minor object motions).

Conclusion

In this section, the various sliding windows were discussed for different possible transitions which were of interest to this research. In the next section, calculating measure of difference (fit values) is looked at.

Measure of Difference

After having adaptive examples and potential candidates in hand, the task of calculating the measure of difference becomes straightforward. This section is divided into two parts; the first part discusses the task of calculating measure of difference for gradual transitions and the second part talks about the measure of difference for abrupt transitions.

Gradual Transitions and Their Normal Groups

In gradual transitions, calculating the measure of difference for the potential candidate and adaptive example is as simple as finding the sum of the differences of the moments used to represent these two image sequences for all the frames. Then the sum is used to calculate the average with respect to total number of moments as well as total number of frames in the T partition of the window.

This average value is also called the fit value which is a representation for how well the current window matches the adaptive example (i.e. how closely the frames in the window resemble the frames in a specific type of transition such as cut, fade or dissolve).

Abrupt Transitions and Their Normal Groups

As mentioned before, in some cases the process of generating fit values need to be specialized to direct some of the complications introduced by abrupt transitions. In this section, both the problem and the proposed solution are discussed.

Problems

The problem is introduced by the fact that the abrupt transitions lack an actual size. If the gradual transitions adaptive examples (figure 4.14) are also used in detecting cuts then some of the underlying relations among different possible fit values of normal groups (regions of low activity) and cuts (represented as a sudden change between two frames) will not hold. These fit values relationships are discussed later on in this section. As demonstrated in figures 4.8, 4.9, 4.14 and 4.15, there are four possibilities and they are as follow:

1. If the window is on a region of activity (namely on a cut) and the algorithm is generating the normal group adaptive examples (Cn).
2. If the window is on a region of activity (namely on a cut) and the algorithm is generating the cut adaptive examples (Cc).
3. If the window is not on a region of activity (namely on a cut) and the algorithm is generating the normal group adaptive examples (Nn).
4. If the window is not on a region of activity (namely on a cut) and the algorithm is generating the cut adaptive examples (Nc).

Each of these four situations will generate a fit value. These four fit values are labeled Cn, Cc, Nn and Nc respectively. When the window is on a cut then Cc should be less than Cn since the minimum value is defined to be the best matched. On the other hand, when the window is on a normal (no activity) region Nn should be less than Nc.

These two conditions are sufficient for gradual transitions, however since cuts do not have an actual size the second condition (Nn < Nc) does not hold true most of the time. This fact is due to value of Nc being very similar to Nn. The first condition (Cc < Cn) was met since only A partition is used while generating adaptive examples for normal groups.

As shown in figure 4.15 the adaptive examples for situation 3 (Nn) and situation 4 (Nc) both lack any great change; henceforth in many cases fit value for situation 4, Nc, drops below the fit value for situation 3, Nn causing the second condition to break. This fact results in an excessive number of false positives and it cannot be solved by simply adjusting the window size. Therefore a specialized method is required which is discussed in the subsequent section.

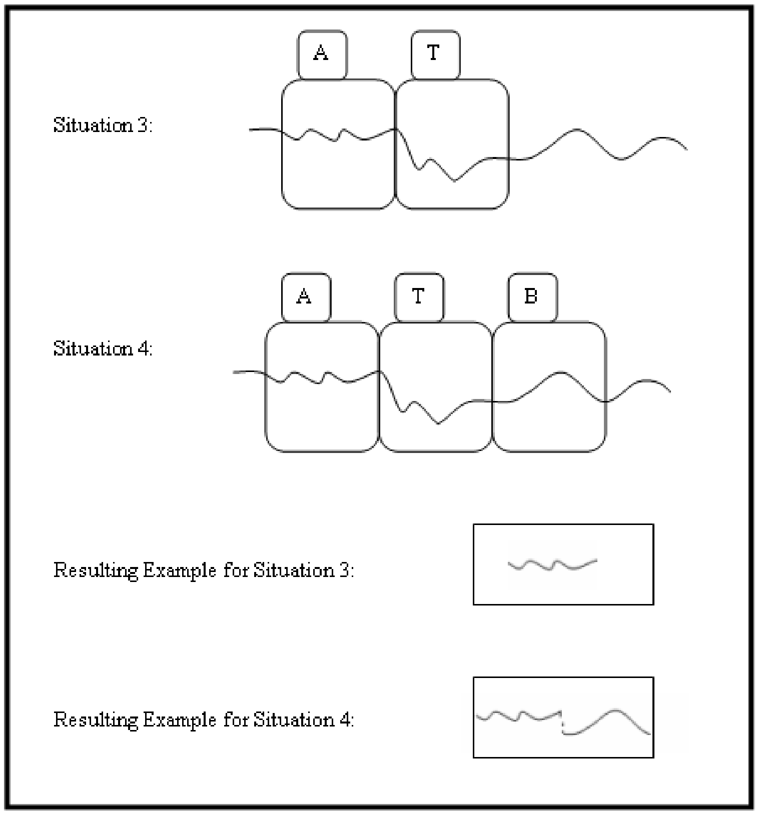

A
T
Situation 3:
A
T
B
Situation 4:
Resulting Example for Situation 3:
Resulting Example for Situation 4:

Fig. 4.15. Illustrates the similarity between situation 3 and 4 (i.e. lack of any great change in situation 4).

Proposed Solution

The first suggested solution is to increase the window partition size lower limit while generating cut adaptive examples. Doing so will cause the partitions A and B to span a more diverse set of frames. Hence the average scores or fit values will increase. However this method will not work since doing so will increase number of false positives by an unacceptable amount. Similarly decreasing the lower limit on normal groups will result in unacceptable number of false negatives. Hence another solution becomes necessary.

To design a universal solution to this problem, not only Cc-Cn and Nc-Nn relationships have to be analyzed but also Cc-Nc and Cn-Nn relationships. Cc has to be less than Nc

(Cc < Nc) and Nn has to be less than Cn (Nn < Cn). The relationship between Cn and Nc is meaningless and of no interest to this research. Figure 4.16 demonstrates the relationship between the four possible pairs of fit values discussed above.

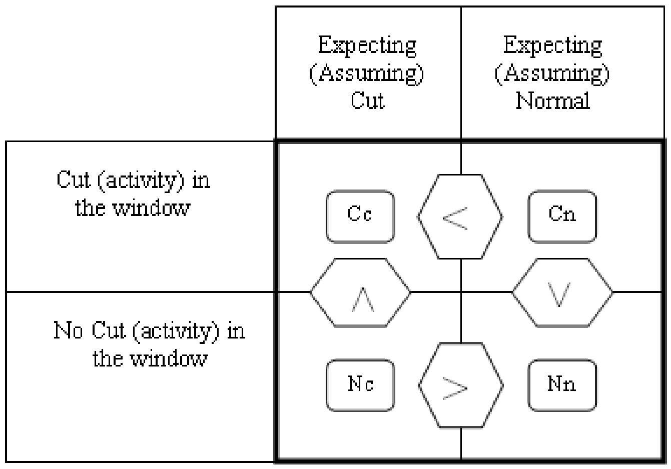

Expecting (Assuming) Cut
Expecting (Assuming) Normal
Cut (activity) in the window
No Cut (activity) in the window
Cc
<
Cn
∧
∨
Nc
>
Nn

Fig. 4.16. Illustrates the main relationships which must hold in order for any suggested solution to work universally (in all circumstances).

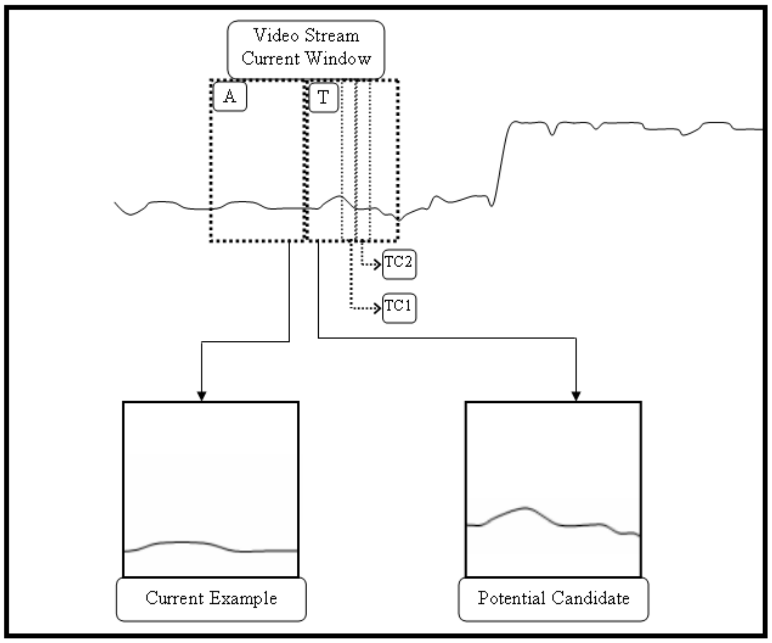


Fig. 4.17. Illustrates the process of extracting potential candidate and generating a normal adaptive example for abrupt transitions detector while T partition of the window is over a region of no activity (regions containing minor object motions).

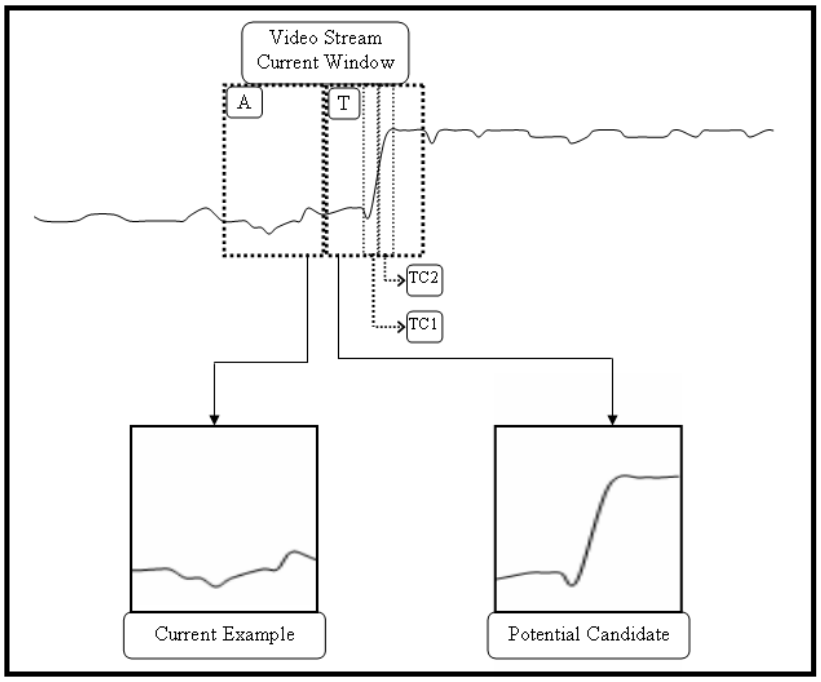


Fig. 4.18. Illustrates the process of extracting potential candidate and generating a normal adaptive example for abrupt transitions detector while T partition of the window is centered on a cut.

The proposed universal solution (a solution which works for all circumstances) is to use the center of partition T (potential candidate). As illustrated in figures 4.8, 4.9, 4.14 and 4.15 the center of T is repartitioned and the center partitions are labeled as *TC1* and *TC2* respectively. Each of these partitions is one frame long.

Figure 4.18 illustrates the normal groups sliding window for cut whereas figure 4.18 illustrates the same settings with exception of sliding window being on a transition (namely cut) rather than over a region of low activity.

The difference between average of the moments of *TC1* and *TC2* is noted by label *Dc*. *Dc* is very large if there exist a cut (or a large jump due to another type of transition or effect) which is centered in the middle of sliding window partition T. *Dc* is used along with the original fit value (as described in gradual transition) to achieve all four main relationships of figure 4.16. Equations 4.7 and 4.8 are used while generating the cut and normal group adaptive examples (*Fv* represent the original fit value similar to the one discussed in gradual transition adaptive examples sections).

- When expecting regions of low activity (i.e. generating adaptive examples for normal groups)

$$NewFitValue = \frac{Fv}{Dc + 0.01} \tag{4.7}$$

- When expecting cuts (i.e. generating adaptive examples for cuts)

$$NewFitValue = \frac{Fv}{Dc + 1} \tag{4.8}$$

These equations are discussed in detail under the *further discussions* section.

Extremely Sensitive Change Detector (ESCD)

In previous sections, we discuss why equations 4.7 and 4.8 actually work. Usage of the mentioned equations is named *Extremely Sensitive Change Detector* (ESCD). The goal of this section is to explain ESCD technique and convince the reader that it guarantees the four main relationships which are presented in figure 4.16 to hold in majority of situations.

ESCD is the reason why adaptive examples generation process as well as no threshold technique work for abrupt transitions.

The division operation is at the center of ESCD algorithm. Specifically the properties of division that state: if the denominator is smaller than one then the results will be larger than numerator and vice versa (if denominator is equal to one the result will be equal to the numerator). These properties can be used to raise the value of Cn and Nc considerably so all the four main relationships to hold (see figure 4.16).

Before discussing the reasons why the four main relationships hold, the equations 4.7 and 4.8 need to be discussed further. As mentioned in the previous section, *Dc* is very large when there exists a cut centered in the middle of partition T (in most cases many times larger than 1) and very small when T partition is over a region of low activities (less than 1).

Incorporating the properties of divide and the topics of previous paragraph helps us to achieve our goal. In the case when nothing is happening (when window is over a region of low activities) *Dc* will be less than 1 which will cause the original fit value, *Fv*, to increase in value noticeably if divided by *Dc.*

Obviously the increase in *Fv* value is not an acceptable and therefore the denominator is incremented only by 0.01 when expecting a region of low activities (0.01 is just to ensure the value does not go to zero and it is small enough to not effect the four main relationships in anyway and it is fixed, meaning it does not have to be adjusted for different types of video or environments). Also *Fv* for normal is smaller than the *Fv* value for cuts. All these lead to new fit values for Nn and Nc such that Nn is less than Nc (Nn < Nc).

In case when something is happening (when window is over a region containing transitions, effects or even distinct motion), large value of *Dc* will be plugged into cut equation (4.8) causing the original fit value, *Fv*, to drop noticeably (in this case, *Fv* is also large in value but still smaller than *Dc* due to the fact that more context is used while calculating *Fv*).

Keep in mind that the denominator of cut equation is incremented by one. On the other, hand since only A partition is used in normal groups calculation, *Fv* (for when expecting group of low activity) is very small. This fact is compensated for by a large *Dc* value which leads to a very small final fit value. Therefore the new equations will not effect the relationship between fit value for cut and normal in this case (Cc < Cn).

The previous paragraphs explained the technique used to compensate for the complications of cut when generating adaptive examples. They also explained why the relationship Cc-Cn still holds even after applying the new equations. The upcoming paragraph will explain why the secondary relationships (Cc-Nc and Cn-Nn) hold as well.

Cut equation will result in a higher value when there is no activity compare to when the window is over a region of high activity (Cc < Nc). This is because original *Fv* in when there is activity is less than the *Fv* when there is no activity (the denominator is ignored here since it the same in both cases). Similar logic leads to the following relationship: Nn < Cn (i.e. the secondary relationships were also true before the new equations were applied; the relationships also preserved afterward since for both relationships the equations share the same denominator).

4.5.2. Detection and Classification

The detection and classification stages are very simple since majority of work has been done in the representation stage.

Using adaptive example generators and ESCD eliminate the need of any thresholds. Hence the *detection* step is only consisted of one step which is to finding the minimum fit value (the best match adapted example). If the minimum value is calculated using a normal adaptive example then the frames in T is labeled as normal and be ignored in future steps.

Classification on the other hand is consisted of two sections. The first part is the same as detection stage. The second part is basically taking the label (and other necessary information) from the best matched example and applying it to all the frames in the sliding window T partition.

More details can be found under *Overlapping Windows Frame Scoring System* section later on in this chapter.

4.6. Implementation Details – Second Level Algorithms

At this point the reader should have a good understanding of low level details which are the underlying techniques in this research. This section contains the implementation details for higher level techniques, providing a bigger picture of the system.

4.6.1. Detectors

Figure 4.20 represents a high level implementation diagram for this algorithm. The first step is to capture the video stream and extract the twenty seven moments for each frame within the video stream. These moments then are received as input by individual detectors.

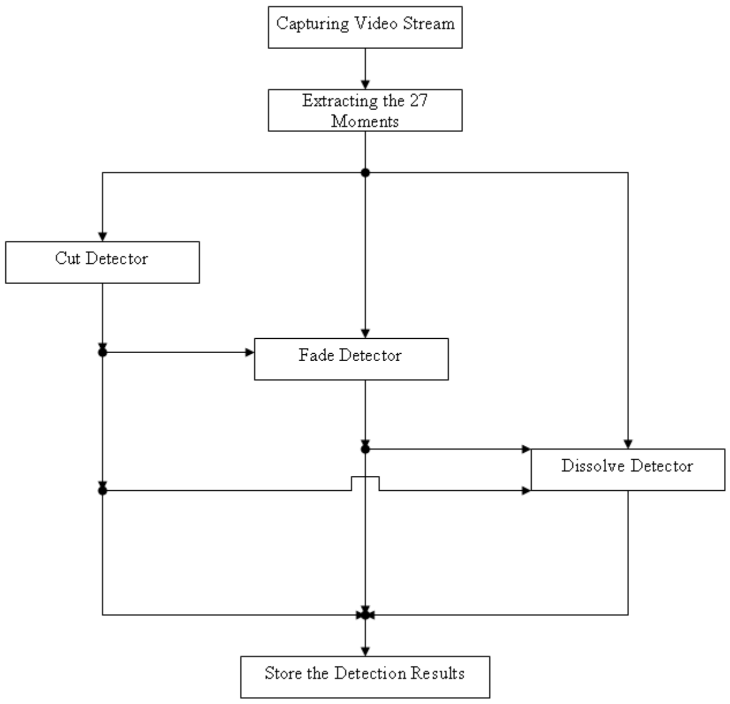


Fig. 4.19. Illustrates the high level process flow for the second algorithm.

Notice that the output of cut is sent to both fade and dissolve and output of fade is sent to dissolve. These inputs result in a lower detector complexity for fade and dissolve since they are used for false positive elimination task in each of these detectors.

At the end the results are stored and analyzed separately. This does not cause any problems since the fade and cut algorithms are used in the succeeding detectors and since no dissolve will be marked as fade by mistake due to the clear distinction between their definitions.

Cut

Cut detector is the most complicated among the three detectors since it is the first detector to be executed. Hence no finalized detections results exist to be used as input. Cut detector is executed first since in general they are the easiest transition to detect compare to dissolves and fades.

Since frames moments are the only inputs, cut detector uses ESCD techniques for cuts of length four and eight as well as a dissolve gradual transition detector (GTDD) algorithm.

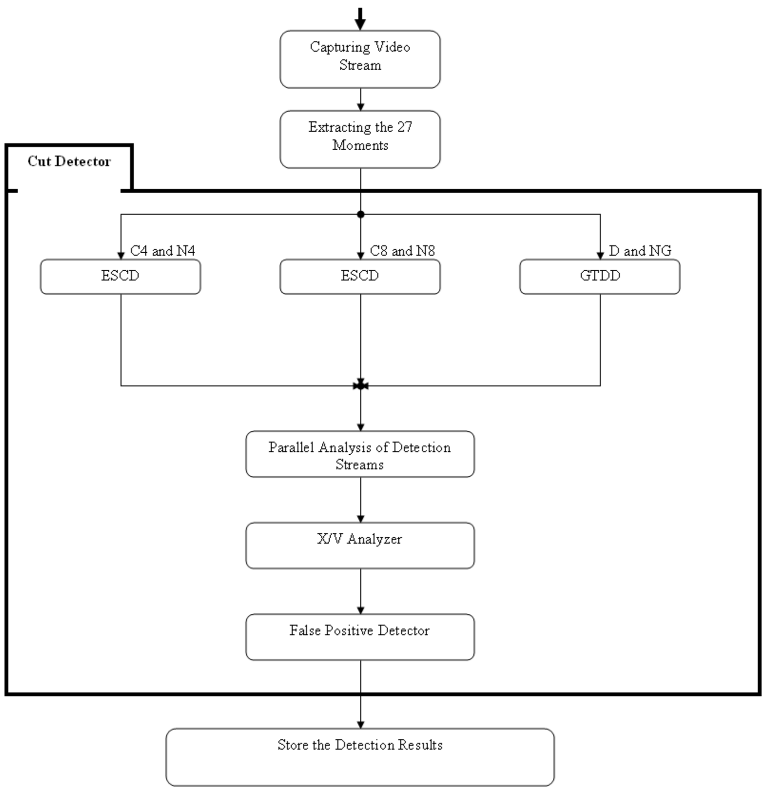


Fig. 4.20. Illustrates the high level process flow for cut detector.

In figure 4.20, C4 and N4 indicate that ESCD is designed for detecting cuts of length four. C8 and N8 indicate that ESCD is designed for detecting cuts of length eight. Similarly, D and NG indicate that the dissolve and normal groups for gradual transitions are used in GTDD.

The results from all three detection scheme is forwarded to *parallel analyzer* and then parallel analyzer output is forwarded to *X/V analyzer* and its output to *false positive detector*. These stages are discussed later on in this chapter.

Fade

Fades are executed second (see figure 4.19) since they are harder than cuts and easier than dissolves to detect. Also because fade detector does not detect the dissolves and the fact that dissolve detector can detect fades by mistake.

Figure 4.21 is very similar to cut detector process flow of figure 4.20. The first step in fade detector is fades gradual transition detection (GTDF) during which the potential frames are marked as fade.

The second stage is monochrome frame detection. The detector designed for this task detects and labels all the monochrome frames within the video stream by using a set of previously collected statistics for different varieties of monochrome frames.

The detection stream from each of these detectors is sent to parallel analyzer. These streams are analyzed along with finalized detection results from cut detector. Parallel analyzer outputs fades detection stream which then sent to fades positive detector for further analysis. These sections are discussed in details later on in this chapter.

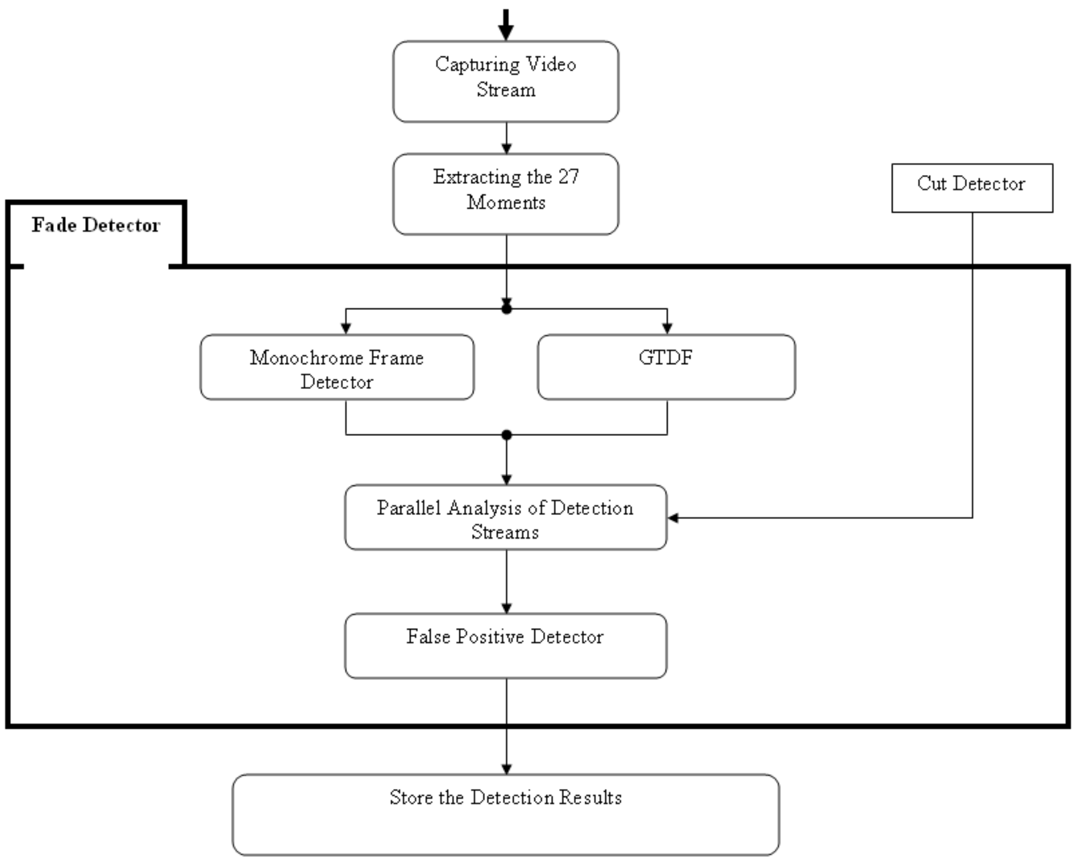

Capturing Video Stream
Extracting the 27 Moments
Cut Detector
Fade Detector
Monochrome Frame Detector
GTDF
Parallel Analysis of Detection Streams
False Positive Detector
Store the Detection Results

Fig. 4.21. Illustrates the high level process flow for fade detector.

Dissolve

Dissolve follows the same process flow as fade with exception of running GTDD instead of GTDF and does not require any monochrome frame detector (see figure 4.22). It also uses the finalized detection results of both cut as well as fade detectors.

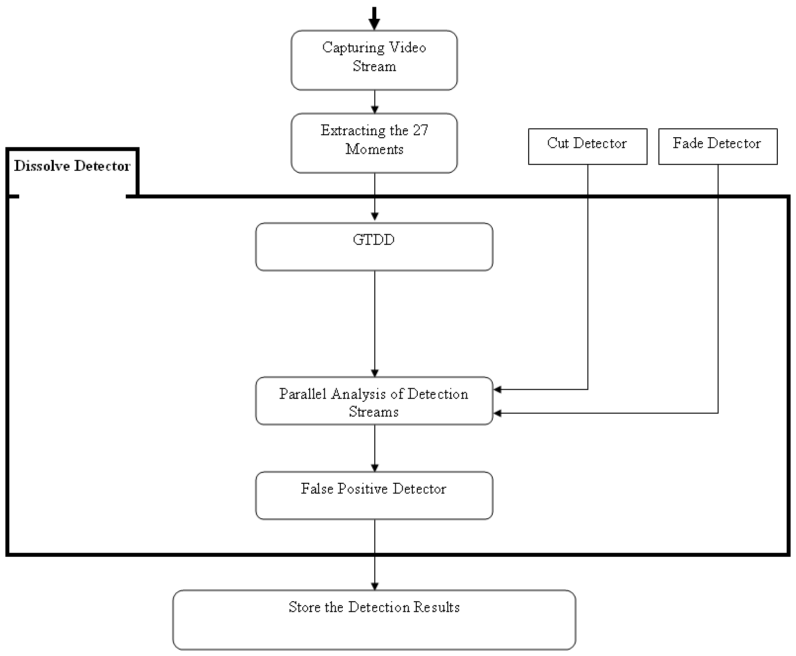


Fig. 4.22. Illustrates the high level process flow for dissolve detector.

4.6.2. Techniques

To achieve the goals discussed earlier in this chapter, many novel techniques had to be proposed and implemented. Some of these techniques, namely *Adaptive Examples* and *Extremely Sensitive Change Detector* have been discussed in the earlier sections of this chapter. The section focuses on the remaining of these techniques such as parallel analyzer, X/V analyzer and false positive detector.

Overlapping Windows Frame Scoring System

This section will describe the techniques used to find the best fit example for each of the frames in the video stream. Figure 4.23 illustrates the execution process at time *t*.

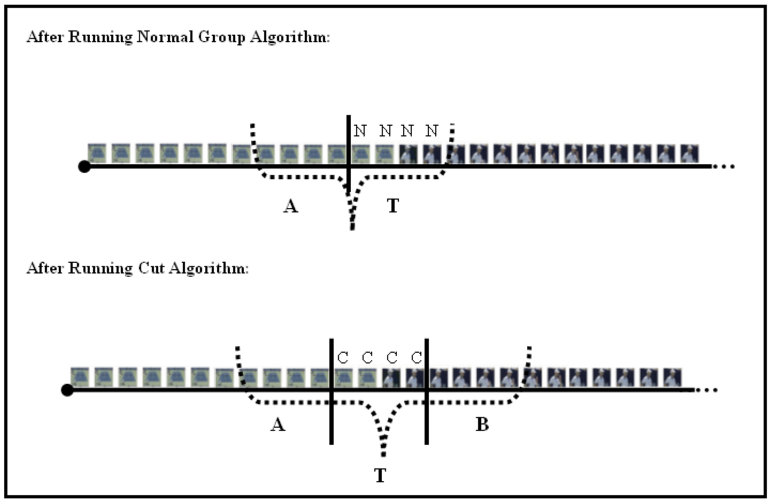

After Running Normal Group Algorithm:
N N N N
A
T
After Running Cut Algorithm:
C C C C
A
B
T

Fig. 4.23. Illustrates the high level process flow for cut detector.

The top section (of figure 4.23) represents the results after execution of normal group algorithm. Since normal groups are the first algorithm executed, the label *N* (normal) is assigned to all the frames in T partition of the window.

On the second part the cut algorithm is executed. The cut algorithm results in a smaller fit value since the T partition of the window is located over an actual cut; hence all the previously labeled frames in the current T partition are relabeled as *C*.

At this point, the first iteration of cut detector ends and the second iteration starts. In other words, both windows (for normal group and cut) will move forward by one frame and the same procedure will be repeated for time instance *t+1*. Keep in mind that at time *t+1* the label of the first frame of T partition in time *t* cannot be changed any more (i.e. the first frame in each window will not be changed, once the algorithm moves to the next frame).

In gradual transitions this scoring technique is used with various window sizes since gradual transitions exist in different sizes. Conversely for normal groups and cuts only a specific size window is used. This size is fixed and does not have to be changed for different type of video.

In case of cuts, the size of partitions needs to be as small as possible because windows which have bigger context (more frames) introduce more noise. A partition size of two or less is not appropriate since cut is defined as a sudden change between two frames. Hence the window size has to be bigger than two. Size three also introduces a problem since three is an odd number the partition will have twice as much information about one shot as it will have for the other one which in returns leads to poor performance. As a result the cut will cause bias in calculation for one of the shots. Henceforth the cut algorithm will use a partition size of four.

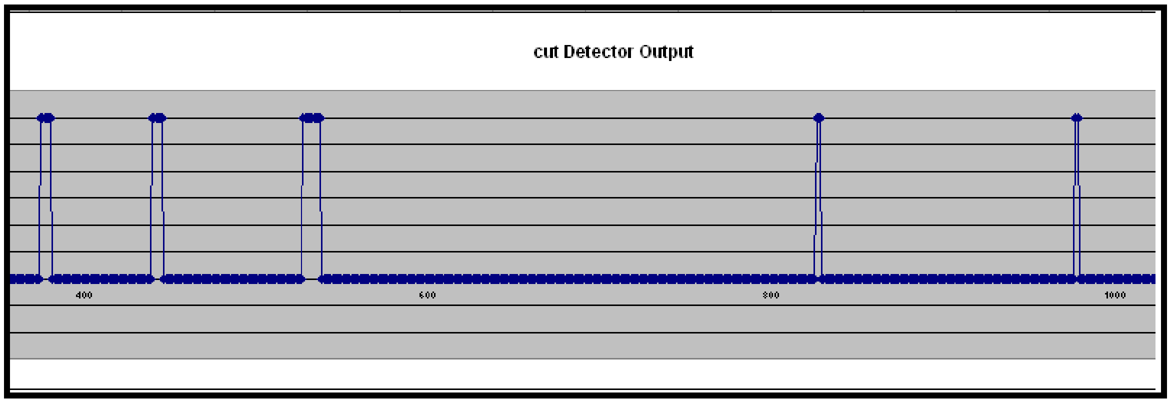


Fig. 4.24. Visual for detection streams of cut (with partition size of eight).

However, in cut detector ESCD is executed twice. During the first run, it uses partition size of four. The second time, it uses the partition size of eight which introduces twice as much context to each partition (the detection streams are illustrated in figures 4.24 and

4.25). More contexts results in a slightly different detection stream which is used in *parallel analyzer* to better detect cuts (see figures 4.25 and 4.26).

Parallel Analyzer

Parallel analyzer, as the name suggests is stage in which the data from different detectors are all analyzed in parallel (at the same time). Main advantage of this technique is that it does not use any complicated pattern recognition algorithms.

Parallel analyzer simply iterates through the video stream (all the labels) and groups the consecutive repetitions of labels of interest other than normal frames (label *N*). Figure 4.25 provides the visuals for the cut and dissolve detection streams which are used in parallel analyzer of cut detector.

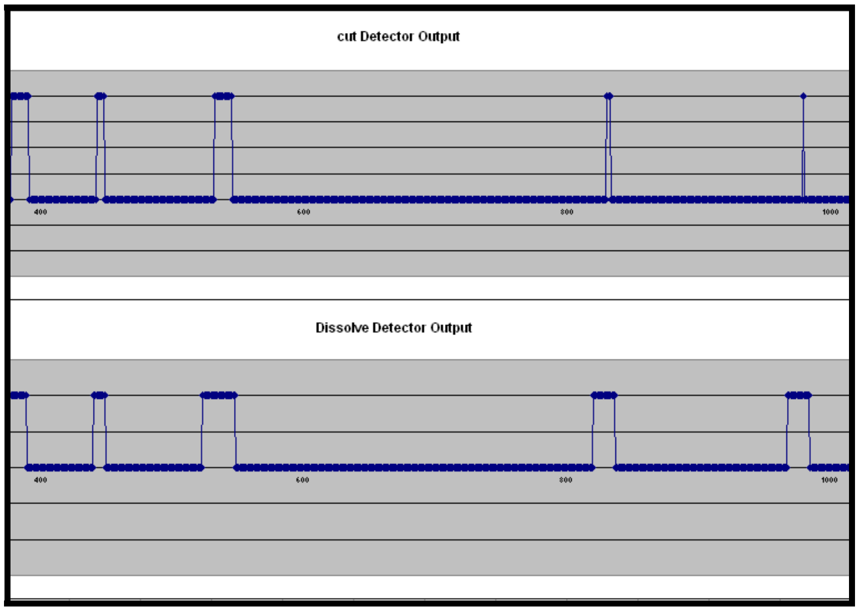


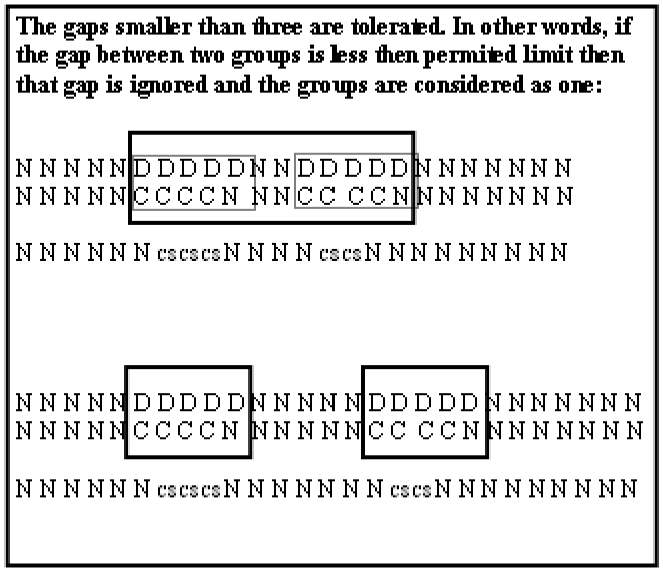
Fig. 4.25. Visual for detection streams of cut (with partition size of four) and dissolve detectors.

In the rest of this section, the parallel analyzer for cut, fade and dissolve detectors are discussed.

Cuts

The parallel analyzer for cuts receives three different detection input streams which were generated in cut detector (see figure 4.20). These streams compensate for the lack of any finalized detection results. Three detection streams inputs are as follow:

1. GTDD output stream
2. C4 (cut with partition size of four) output stream
3. C8 (cut with partition size of eight) output stream



The gaps smaller than three are tolerated. In other words, if the gap between two groups is less then permited limit then that gap is ignored and the groups are considered as one:

N N N N N D D D D D N N D D D D D N N N N N N N
N N N N N C C C C N N N C C C C N N N N N N N N
N N N N N N cscscs N N N N cscs N N N N N N N N
N N N N N D D D D D N N N N N D D D D D N N N N N N N
N N N N N C C C C N N N N N N C C C C N N N N N N N N
N N N N N N cscscs N N N N N N N cscs N N N N N N N N N

Fig. 4.26. Visual for detection streams of cut (with partition size of four) and dissolve detectors.

The number of cuts and dissolves in figure 4.26 and all the similar future figures are not the exact numbers used in the algorithm. Different sizes are used to visualize the process and aimed to help the reader to gain a better understanding of the process.

In figure 4.26, the first two streams (dissolve and C4) are used to group the frames related to each potential transition (region of interest). C8 stream is used as another measure while deciding the type of transitions for each group. Next paragraph explains why each of these streams is used.

Dissolve detection stream (GTDD output stream) provides another basis for comparison in *parallel analyzer*. It helps to distinguish between abrupt and gradual transitions. Cut (of size eight) provides another detection stream which was generated by using twice as much context as cuts of size four (leading to a similar output stream, another basis for comparison in *parallel analyzer*).

If there is a gap (frames labeled *N*) between two groups and it is smaller than the permitted limit (three frames) then that gap is ignored and the groups are merged into one group).

Table 4.1 contains the if-statements used to distinguish between various possible groups (Cs), (Ds), (Vs), (Xs) and (Ns). Ds identify the regions of the video which include transitions, effects and motions and they are ignored by labeling them as *N*. Xs and Vs represents the *uncertainty groups* discussed later on. Ns present the regions of low activity in the video stream (the regions which formed a group due to motion or effects).

Names of most variables used in Table 4.1. are self-explanatory except a few which are explained here:

- *cutWinSize* is the size of the partition (it is set to four during the first iteration and to eight during the second iteration).
- *lastIndexInsideIf* is the index of the last frame in the current group as the algorithm iterates through the groups.

SOURCE CODE

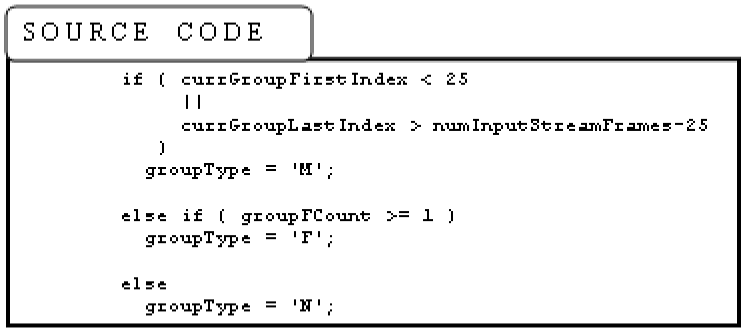
```
if ( (cutWinSize*2-2) < groupC4Count  &&  groupC4Count < (cutWinSize*2+2)
     &&
     groupDAndCCount < 4
     &&
     (
       ( groupC8Count > 3  &&  groupC4Count == cutWinSize*2+1 )  ||
       ( groupC8Count == groupC4Count )  // for rest of possible groupC4Sizes
     )
  )
  groupType = 'C';

else if ( groupDCount > 4
          &&
          // This will get rid of some motions (19:55-227to236 N detected as D)
          groupDCount - groupC4Count - groupC8Count > 10
          &&
          groupDCount >= groupLength-2
          &&
          groupC8Count < 5
          &&
          ( lastIndexInsideIf > cutWinSize*4  &&  lastIndexInsideIf  < (numInputStreamFrames - cutWinSize*4) )
        )
  groupType = 'D';

else if ( ( 3 < groupC4Count )
          &&
          (
            (
              ( groupDAndCCount < 5 )
              &&
              ( groupC4Count-groupDCount > 5  ||  groupDCount < 2 )
            )
            ||
            ( lastIndexInsideIf < cutWinSize*4  ||  lastIndexInsideIf  > (numInputStreamFrames - cutWinSize*4) )
          )
        )
  groupType = 'V';

else if ( groupDCount > 3  ||  ( groupLength > 1  &&  groupDCount == groupLength ) )
  groupType = 'X';

else
  groupType = 'N';
```

Table 4.1. Presents the if-statements used in parallel analyzer to label the different groups.

The labeling section (table 4.1) is followed by boundary determination for both detected transitions (cuts) and other groups (X, V, D and N). This process simply takes place through adjusting the group boundaries slightly after brief analysis of fit values for the frames within the group.

Fades

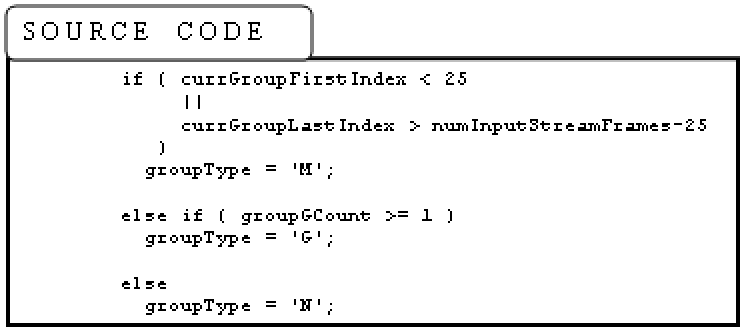
Fades benefit from a simpler process flow since they use finalized results of cuts. The parallel analyzer of fade takes two detection streams as inputs (see figure). but the algorithm is divided into two sections. These sections are used to label groups as fade outs and fade ins. Table 4.2 represents the conditional statements used to identify and label groups as fade ins and table 4.3 present the conditional statements used to label groups as fade outs.



SOURCE CODE

```
if ( currGroupFirstIndex < 25
     ||
     currGroupLastIndex > numInputStreamFrames-25
  )
  groupType = 'M';

else if ( groupFCount >= 1 )
  groupType = 'F';

else
  groupType = 'N';
```

Table 4.2. The conditional statements used for identification and labeling of fade ins.



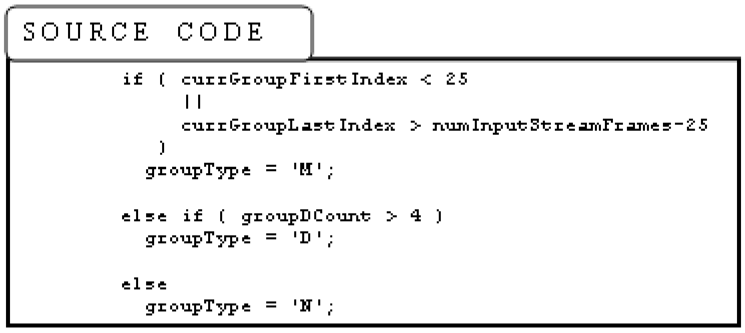
```
SOURCE CODE
        if ( currGroupFirstIndex < 25
             ||
             currGroupLastIndex > numInputStreamFrames-25
           )
          groupType = 'M';

        else if ( groupGCount >= 1 )
          groupType = 'G';

        else
          groupType = 'N';
```

Table 4.3. The conditional statements used for identification and labeling of fade outs.

The groups which were identified and labeled as fade are compared against cuts finalized detection results. If a fade is in a close proximity of a previously detected cut then it is ignored (marked as N).

Dissolves

Dissolves use the cuts finalized detection results and unlike fades, there is only one type of dissolve. Hence dissolve algorithm is simpler than fade algorithm. Table 4.4 provides the if-statements used for identification and labeling of groups as dissolve.



```
SOURCE CODE
        if ( currGroupFirstIndex < 25
             ||
             currGroupLastIndex > numInputStreamFrames-25
           )
          groupType = 'M';

        else if ( groupDCount > 4 )
          groupType = 'D';

        else
          groupType = 'N';
```

Table 4.4. Presents the conditional statements used in identification and labeling of dissolves.

After the potential groups are identified and labeled as dissolve, they compared against the previously finalized detection results of cuts and fades. If a detected fade is in close proximity of a previously detected cut or fade then it is ignored (marked as N).

Uncertainty Groups

The parallel analyzer in cut detector labels the groups that it is uncertain about as V and X. The V groups share more similarities with cuts whereas the X groups share more similarities with dissolves (although an X or V group can be a cut, a dissolve or neither of the two). As a result the *uncertainty group analyzer* is also called *X/V analyzer*.

Using uncertainty groups will allow us to distinguish among the groups which are most likely true transitions and potential false positives. Hence in the future steps the algorithms do not need to analyze the detections (groups) which we are certain about.

Fade and dissolve detectors do not require uncertainty groups since they have access to finalized results of the previously executed detectors. In these cases, if the new detection is near a previously detected transition (fade or cut) then it will be ignored.

Variations of Uncertainty Groups

In parallel analyzer the groups which do not follow the conditions that define a cut group, are labeled as X or V (uncertain). Figure 4.27 presents different possible causes for uncertainty groups. They are explained further below:

A. **The group size was shrunk**. This problem occurs due to lack of sufficient amount of information in video representation stage. For example, the cuts which are located between two frames with similar intensities and color distributions (see figure 4.28) will not clearly be presented by using the twenty seven moments and therefore the algorithm confuses them with object motions, hence *Frame Scoring System* described previously will not work as desired when dealing with the shrunk groups.

B. **The group size was enlarged**. This problem can occur due to two reasons:

    1) *Close proximity to other gradual transitions or Effects*. If some effects such as transitions of overlays, change in brightness, or transitions in frames occur near a cut then the cut and the effect are merged.

    2) *Two or more cut groups were merged*. This situation occurs due to the following:

        a) The cut groups were too close to one another hence they were merged.

        b) The cut groups were separated but the group was merged due to GTDD output stream.

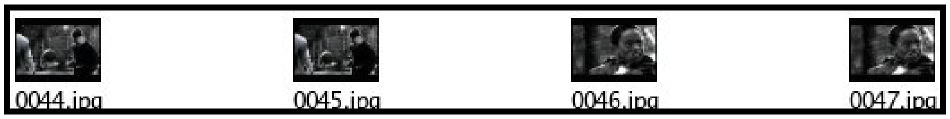


Fig. 4.27. Example of a cut between two frames with similar intensities and color distributions.

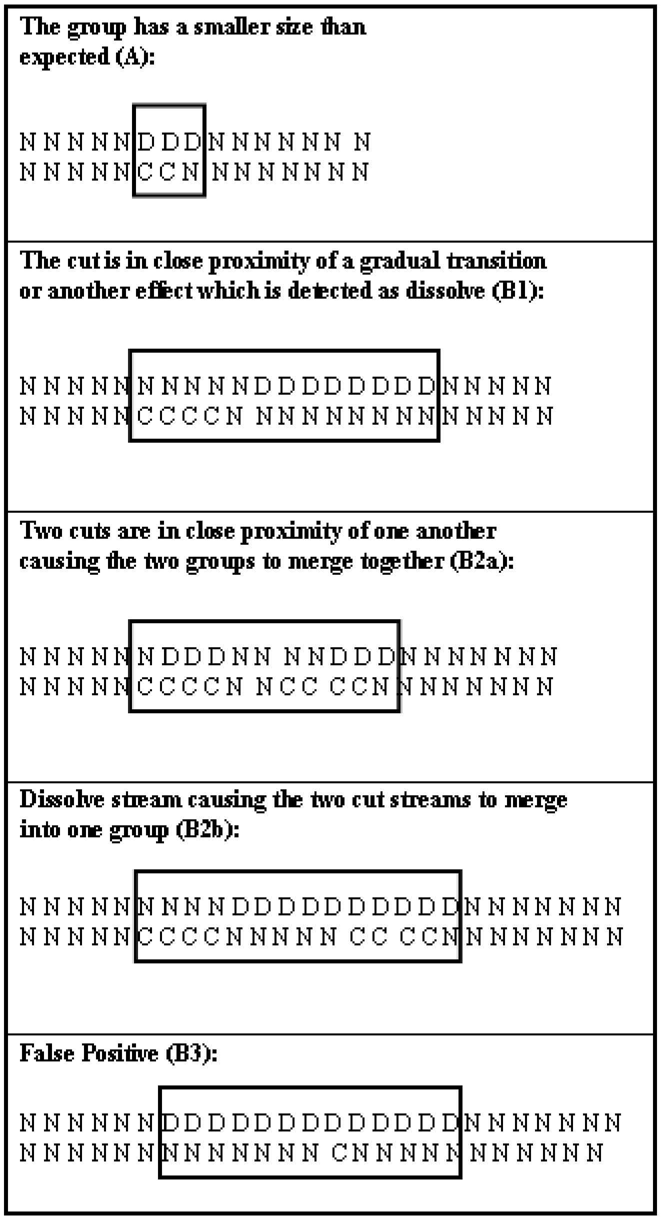

The group has a smaller size than expected (A):
N N N N N D D D N N N N N N N N
N N N N N C C N N N N N N N N
The cut is in close proximity of a gradual transition or another effect which is detected as dissolve (B1):
N N N N N N N N N N D D D D D D D D N N N N N
N N N N N C C C C N N N N N N N N N N N N N N
Two cuts are in close proximity of one another causing the two groups to merge together (B2a):
N N N N N N D D D N N N N D D D N N N N N N N
N N N N N C C C C N N C C C C N N N N N N N N
Dissolve stream causing the two cut streams to merge into one group (B2b):
N N N N N N N N N D D D D D D D D D D D N N N N N N N
N N N N N C C C C N N N N N C C C C N N N N N N N N
False Positive (B3):
N N N N N N D D D D D D D D D D D D D N N N N N N N
N N N N N N N N N N N N N N C N N N N N N N N N N N

Fig. 4.28. Presents different factors behind identification of a group as an uncertainty group.

Classification Solution for Uncertainty Groups

The goal of *X/V analyzer* is to deicide if an uncertainty group should be labeled as a cut (C) or as a normal group (N). It is structured in similar fashion as the parallel analyzer algorithm. The *X/V analyzer* and *parallel analyzer* can be merged into one step however doing so will add to the complexity of the algorithm and hence was avoided in this research. As the first step, each of the uncertainty groups is divided into subgroups by using the C4 frames as the only grouping criteria (see figure 4.29).

As can be seen in figure 4.29 two techniques are used for grouping. The first uses only the labels of C4 whereas the second one uses the fit values of C4 output stream. Fit values are used to distinguish between C4 labels of one cut compared to the other in cases where the labels are merged and are not distinguishable from one another at the first sight.

Keep in mind that the size of uncertainty groups does not play any rule in how these groups are divided. Similar to larger groups, the smaller ones are either separated into smaller groups or preserve their original length.

After dividing each of the possible uncertainty groups into subgroups, a new label has to be assigned to each of these subgroups. The rest of this section focuses on the labeling task for each of the variations of uncertainty groups.

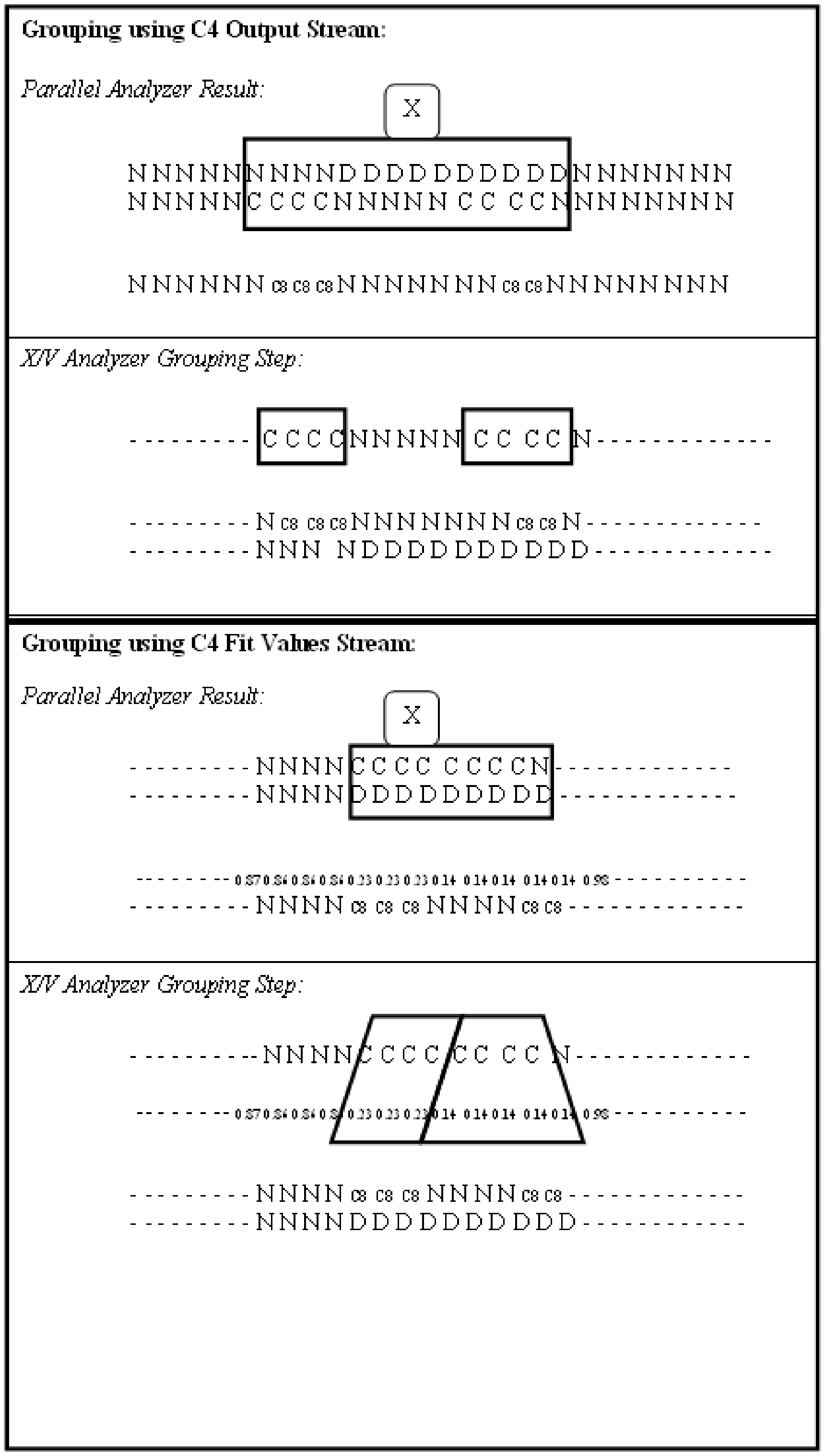

Grouping using C4 Output Stream:
Parallel Analyzer Result:
X
N N N N N N N N N D D D D D D D D D D N N N N N N N N
N N N N N C C C C N N N N N C C C C N N N N N N N N N
N N N N N N C8 C8 C8 N N N N N N N N C8 C8 N N N N N N N N
X/V Analyzer Grouping Step:
- - - - - - - - - C C C C N N N N N C C C C N - - - - - - - - - - - - -
- - - - - - - - - N C8 C8 C8 N N N N N N N N C8 C8 N - - - - - - - - - - - - -
- - - - - - - - - N N N N D D D D D D D D D D D - - - - - - - - - - - - -
Grouping using C4 Fit Values Stream:
Parallel Analyzer Result:
X
- - - - - - - - - N N N N C C C C C C C C N - - - - - - - - - - - - -
- - - - - - - - - N N N N D D D D D D D D D - - - - - - - - - - - - -
- - - - - - - - 0.87 0.84 0.84 0.84 0.23 0.23 0.23 0.14 0.14 0.14 0.14 0.14 0.98 - - - - - - - - - - -
- - - - - - - - - N N N N C8 C8 C8 N N N N C8 C8 - - - - - - - - - - - - -
X/V Analyzer Grouping Step:
- - - - - - - - - - N N N N C C C C C C C C N - - - - - - - - - - - - -
- - - - - - - - 0.87 0.84 0.84 0.84 0.23 0.23 0.23 0.14 0.14 0.14 0.14 0.14 0.98 - - - - - - - - - - -
- - - - - - - - - N N N N C8 C8 C8 N N N N C8 C8 - - - - - - - - - - - - -
- - - - - - - - - N N N N D D D D D D D D D D D D - - - - - - - - - - - -

Fig. 4.29. Illustrates the regrouping techniques used in X/V analyzer.

Smaller Groups (A)

The goal behind analyzing smaller groups is to distinguish the groups which were shrunk due to existence of motion or false positive effects from the real cuts such as the one in figure 4.27. *X/V analyzer* uses the *C8* output stream for this task. A group can be labeled as a cut if there is sufficient number of *C8* frames within that group.

Larger Groups (B)

The larger groups will be divided into smaller subgroups through the process illustrated in figure 4.29. If the problem is due to the close proximity of a cut to a gradual transition (B1) then it is directed through use of GTDD output stream.

If the problem is due to the close proximity to other cuts (B2a) then it is easily solved through dividing the large groups by using the dissolve fit values (see figure 4.29). Otherwise if it is caused due to the complications with GTDD output stream (B2b), it is solved by using only cuts of length four as the grouping criteria. Each of these subgroups is discussed here.

Smaller subgroups are handled using the same algorithm as smaller groups (see previous section). Similar to *parallel analyzer*, the *X/V analyzer* labels the perfect or near perfect size subgroups as cuts whereas the larger subgroups are distinguished from effects and transitions such as short gradual transitions through analysis of cuts (C4 and C8), and dissolve detection stream (also see assumptions section earlier in this chapter).

If none of the conditions presented in the previous sections hold true then the subgroups and groups are labeled as *M* or *L* which are just another notation for normal groups. In *false positive detector* label *K* serves the same purpose. Different labels are used to identify which step labeled the frame as normal.

False Positive Detector

The general causes of false positives were discussed in chapter 2. On the other hand, the more specific causes of false positives and the suggested solutions are topics of this section.

Cut

The *False Positive Detector* of cut detector performs two tasks:

1. Removes the cut label of potential false positives without raising the false negative count above a permitted limit.
2. Introduces a threshold which controls the tradeoff between FPs and FNs.

These two steps are described in the rest of this section.

FP Detection

The main task of false positive detector, as the name implies, is to detect and eliminate false positives. In this detector, only the groups previously labeled as cuts are considered. After analysis the groups which have high probability of being false positive are re-labeled as *K*s (just another notation for normal groups).

During the analysis stage of false positive detector, for each cut group, the average of the twenty seven moments is calculated for each frame. Then the frames are sorted with respect to the average values and the following value is calculated:

$$Val = \frac{\text{tDiff[tDiffIndex - 1]}}{\text{otherLargest}} \tag{4.9}$$

where

- tDiff[tDiffIndex-1] represents the frame with the largest average value.
- otherLargest is the average of the average values of the second and third frames with the largest average values.

As described in the next, section this value is used along with a threshold to eliminate most of the false positives.

Threshold as a Controller

A threshold is applied to the values generated by equation 4.9 to provide a controller for adjusting the tradeoff between FPs and FNs. This directly effects the recalls and precisions values. Introduction of this threshold does not in any ways undermine the importance of not utilizing a threshold in the previous stages. The results of different thresholds are presented in the next chapter.

Fades and Dissolves

The gradual transition adaptive examples for when expecting a region of low activity, has a slightly larger T partition than the one used in cut adaptive examples for when expecting a region of low activity (see figures 4.13 and 4.14). This compensates for the large size of gradual transitions window size (since more context results in a larger *Fv* value).

Due to the simplicity of fade and dissolve detectors, many fades or cuts can be detected as dissolves, or cuts can be detected as fades mistakenly. The goal is to avoid these false positives without adding to the complexity of the algorithm.

Therefore, instead of introducing new ESCDs or gradual transitions detectors (GTDs) similar to those in cut detector (figure 4.20), the finalized detection results are compared against the previously detected transitions (cuts or fades). If the new detection is within a specific predefined range from the previously detected transition then the new detection is relabeled as normal (i.e. it is ignored).

This step is performed in parallel analyzer. Additional false positive detectors (such as the one in cut detector) can be easily added for each of the primary gradual transition detectors due to the multilevel property of our algorithm.

Boundaries Determination

If a group with perfect or near perfect size is labeled as a cut then the transition is most likely located in the center of that group (i.e. the two frames in the center of the group are used as boundaries of the detected cut).

In case of larger and smaller cut groups, same techniques is utilized which obviously will account for a small amount of error. To decrease this amount, the fit values are used to adjust the boundaries of detected transitions.

As for gradual transitions, transition boundaries are set to the boundaries of the representative groups. In some cases adjustments will help to locate the boundaries more precisely.

4.7. Conclusion

The implementation and design details of the second algorithm, direct comparison based on predefined examples, were discussed in this chapter. Also discussed were the novel techniques used to meet the initial goals of this research. Then the next chapter includes the experimental results for different test cases and scenarios followed by the discussion and analysis of the results.

Chapter 5

Experiments, Results and Discussions

5. Experiments, Results and Discussions

This chapter describes the evaluation techniques used in our research and the experiments we conducted to compare our two temporal segmentation algorithms.

5.1. Introduction

Recall and precision are the common techniques used to evaluate the results in the field of temporal segmentation. Thus, these techniques are used in our evaluation programs, providing a basis for comparison against other research works.

Direct comparison based on predefined examples and *direct comparison based on adaptive examples* are the two techniques discussed in this chapter. These two algorithms share similar data preparation and evaluation techniques; hence only results and results discussions sections distinguish between the two.

5.2. Input Data

The algorithms were tested on one hour of video sequence obtained from a typical twenty four hours broadcast of CBS channel. The one hour is picked in the way to assure it contains video from the morning, afternoon, evening and night shows (different mix of video types). These video segments are picked in random and the commercials are <u>not</u> being ignored (commercials introduce high level activities which throws off many existing algorithms – see figure 5.1). Figure 5.1 illustrates 43 frames (from 726 to 769) during which there exists three cuts (frames 726, 745, and 762) and four effects which resemble cuts (763,766, 767, and 768).

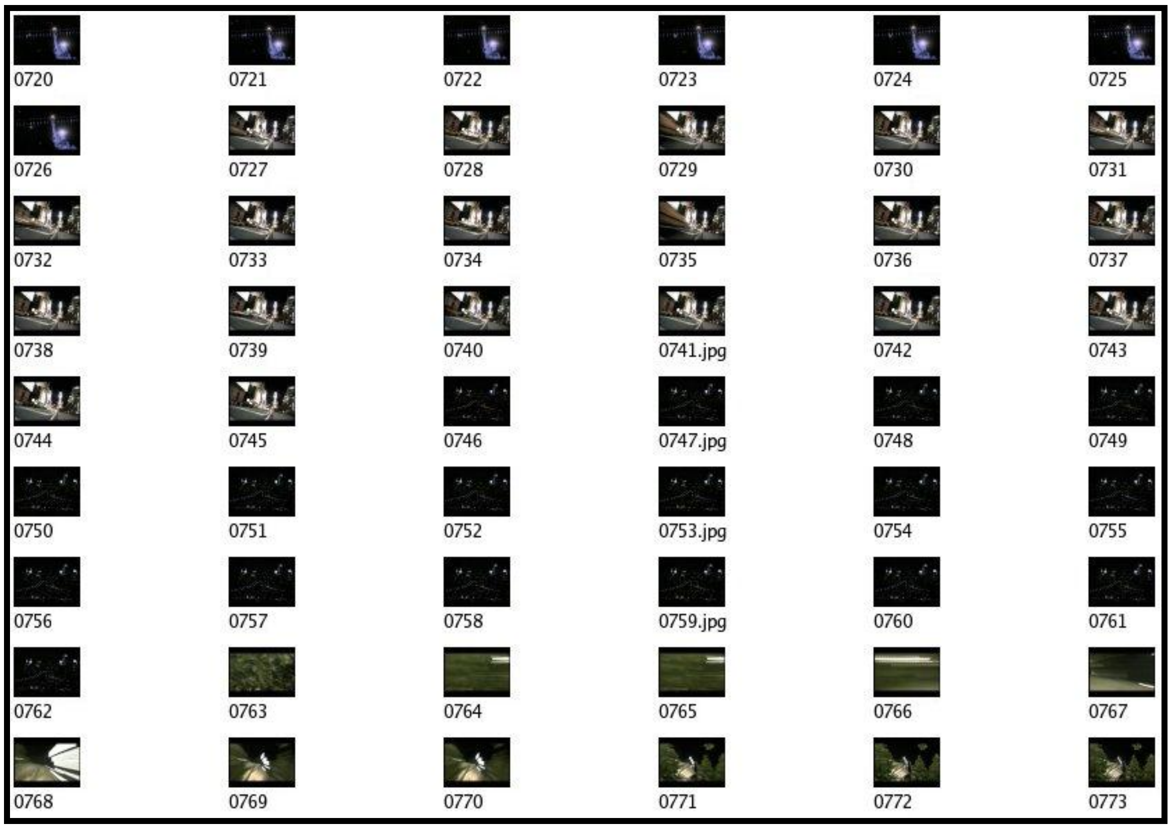


Fig. 5.1. Illustrates 43 frames (from 726 to 769) during which there exists three cuts (frames 726, 745, and 762) and four effects which resemble cuts (763,766, 767, and 768).

The video is captured by using digitizer hardware which stores the frames in JPEG formats of 160 X 120 resolutions as demonstrated in figure 5.1. The frames for each minute are grouped together and then equations 3.1 to 3.9 are used to extract the statistical information from each frame. These data are used to represent both the input video stream as well as examples. A detailed discussion on data preparation is presented to the reader in *Representation* section of chapter 3.

5.3. Truth Data and Truth Grabber Program

The input data was viewed and manually segmented to locate the true positives of all cuts, fades, dissolves, and other transitions. This information is stored in truth files.

The truth grabber program is used to prepare the inputs for the video segmentation based on predefined examples algorithm. It requires folders containing the video moments and optionally video files, as well as the truths files. Truth files contain the labels and exact boundaries for the actual transitions. The *TruthGrabber* program reads the truth file and depending on the truth data and user's preferences it copies the statistical information into a main moments file (examples set) and copies the video files into specified folders.

5.4. Evaluation Techniques

Before discussing any evaluation methodologies, some of the common terminologies are described below:

- **True Positives (TP or** $N_{correct}^{x}$**)** – are the items (transitions in this case) which exist in both truth data as well as the finalized detection results (i.e. the transitions which were detected correctly).
- **False Negatives (FN or** N_{missed}^{x}**)** – are the items which exist in truth data but not in the finalized detection results (i.e. the transitions which are not detected).
- **False Positives (FP or** N_{false}^{x}**)** – are the items which do not exist in the truth data but are introduced in the finalized detection results (i.e. the false alarms).

- **True Negatives (TN)** – are the items which do not exist in the truth data and also are absent in the finalized detection results (i.e. all the frames that are not within any transition boundaries).

When deciding if a TP occurs, we look at the beginning and the ending boundaries. If they are off by less than a specific number of frames (depending on the size and type of the transition in question) then a match is declared (the detection is labeled as a true positive).

In reporting the experimental results, the common recall and precision measures of performance are exploited. These two measures as well as the utility are discussed in the future sections and used to evaluate system performance.

To calculate the measures of performance the results of the automated detection (detection finalized results) were compared to those of manual segmentation (truth data) in order to find matching pairs. As mentioned before, the two boundaries as well as the labels for the truth data and automated detection results should match exactly in order to have a perfect detection. However in our evaluation program if the detected boundaries are off by a specific number frames (depending on length of transition) the detection results will still be labeled as true positive (i.e. as long as the detected transition overlaps with the manual detection of the same transition or is very close to the manual transition then it is labeled as a true positive).

In the recall and precision definition, the superscript x represent the type of transition in question. For example if cuts are being analyzed *x* takes the value of ‘cut’. The subscript *m* stands for manually detected transitions (truth data) whereas subscript *a* stands for automatically detected transitions.

Below is the mathematical description of these measures (the recall and precision definitions are based on [84] and formulas based on [73]):

- **Recall** – is the proportion of shot boundaries correctly identified by the system to the total number of shot boundaries presented.

$$\mathrm{Re}call = R^x = \frac{N^x_{correct}}{N^x_{correct} + N^x_{missed}} \times 100\% \tag{5.1}$$

  where

  - $TP = N^x_{correct} = |\Theta|$, where

    $\Theta = \{S^x_i, i \in \{1,...,k^x_a\} \mid \exists j \in \{1,...,k^x_m\} \text{ and } S^x_i \cap S^x_j \neq \phi\}$

  - $FN = N^x_{missed} = |\Theta|$, where

    $\Theta = \{S^x_i, i \in \{1,...,k^x_a\} \mid \forall j \in \{1,...,k^x_m\} \text{ and } S^x_i \cap S^x_j = \phi\}$

- **Precision** – is the proportion of correct shot boundaries identified by the system to the total number of shot boundaries identified by the system.

$$\mathrm{Pr}ecision = P^x = \frac{N^x_{correct}}{N^x_{correct} + N^x_{false}} \times 100\% \tag{5.2}$$

  where

  - $N^x_{correct}$ (TP) has the same definition as in equation 5.1 and

- $FP = N^{x}_{false} = |\Theta|$, where

  $\Theta = \{S^{x}_{j}, j \in \{1,...,k^{x}_{m}\} | \forall i \in \{1,...,k^{x}_{a}\} \ and \ S^{x}_{i} \cap S^{x}_{j} = \phi\}$

- **Utility** – For comparison purposes many algorithms aggregate the recall and precision values. In this research, utility serves the same purpose. Utility is often referred to as *f-measure* in the information retrieval literature and it allows the researchers to objectively determine what set of parameters lead to the best performance possible. Equation 5.3 is how this aggregated value is calculated.

$$Utility = \alpha \cdot \mathrm{Re}call + (1 - \alpha) \cdot \mathrm{Pr}ecision \qquad (5.3)$$

  In equation 5.3 α is used to control the degree of influence of either of recall or precisions. In our case the value of ½ is chosen for alpha which means the recall and precisions are considered equally. This in return, results in a more specific variation of equation 5.3 which is also known the mean (average) equation (equation 5.4).

$$Utility = \frac{(\mathrm{Re}call + \mathrm{Pr}ecision)}{2} \qquad (5.4)$$

The recall and precision are both equal to one in an ideal case (or 100% if expressed in percentage). A recall of one indicates that all the existing shot boundaries were identified correctly (both boundaries and the label were identified correctly). A Precision of one indicates that no false boundaries or labels (false alarms) will exist in the final results. If both are equal to one then the finalized detection results should be almost exactly the same as the truth data.

5.4.1. Performance Evaluation Program

Performance evaluation program takes the finalized detection and truth data as input and after analysis it outputs the number of true positives, false positives and false negatives as well as recall, precision, and utility values. These data are used later on to perform a comparison on different methods used.

In temporal segmentation the definitions for true positive, true negative, false positive and false negative which were provided earlier in this chapter need to be expanded upon.

The size of abrupt transitions the size is two whereas in gradual transitions exist in different sizes. Hence, the false positive category can be divided into the following subcategories:

- *False Positives*
    - **Type A** – are those items which are labeled as false positive because they are in correct range but have the wrong type.
    - **Type B** – are those items which are labeled as false positive because they did not exist in the truth data.
    - **Type C** – are those items which are labeled as false positive because they were detected already or a better match was found later on.

The performance evaluation program is very flexible. This program requires minimal modifications to perform the following tasks:

- It allows for one specific label to be ignored completely in either truth data or finalized detection data.

    - For example if the finalized detection results contain all primary transitions (namely cuts, dissolves and fades) but the truth data only was gathered for cuts then dissolves and fades can be easily ignored by adding letters d, f and g into the to *ToBeIgnore list* (g represents fade in whereas f represents fade out).
- It also allows for another type of ignore list, or false positive ignore list. If a label from truth data is added to this list then if a false positive occurs in close proximity from the transition/effect and if its label was added to this list then that false positive will not be counted as a false positive.
    - For example if an especial effect such as the one in figure 4.3 (frame numbers 1644 to 1652) occurs too often within the test data and every time it is detected as fade then it can be ignored.
- It allows one type to be counted as another type.
    - For example depending on the specifications one might need to consider both fade in and fade out as fade.
- It allows Type C false positives to be ignored for all labels or for a specific label.

5.5. Results

In this section, the experimentation results from both algorithms, *Direct Comparison based on Predefined Examples* and *Direct Comparison based on Adapted Examples* are presented to the reader. As we will show our technique based on predefined examples, was not as successful as our technique based on adaptive examples.

5.5.1. Direct Comparison based on Predefined Examples

In our experiment, we used 30 and 45 minutes of test data to calculate recall, precision and utility for all possible combinations of program options. First we considered raw moments versus derivatives. The next criterion of interest was amount of context taken into considerations (number of extra frames that were added to each example). Two cases were considered: having no frames and having five frames on each side. In this algorithm the extra frames are necessary since the examples are predefined and therefore it is necessary to have a few extra frames on each side providing information about the surroundings of each example. Finally we consider the effects of data normalization.

Normalization is not needed for derivatives since derivatives are the difference values and hence they are near zero almost all the time except were there is a big change from one frame to another.

Figure 5.3 present the experimentation results for 30 minutes of data. In this experimentation only the raw moments and no frame on each side is used. In the thirty minutes there were 494 transitions. In case of *with normalization* 247 transitions were detected correctly (true positives), 247 transitions were false negatives and 337 detections were false positives. On the other hand, in *without normalization* there were 215 true positives, 279 false negatives and 368 false positives.

As the next step, 45 minutes of data was used in a comprehensive experimentation, the results of which are presented to the reader in figure 5.2. In this case, the best results were obtained when raw moments are used with 5 side frames and with normalization. The use of derivative did not yield desirable results.

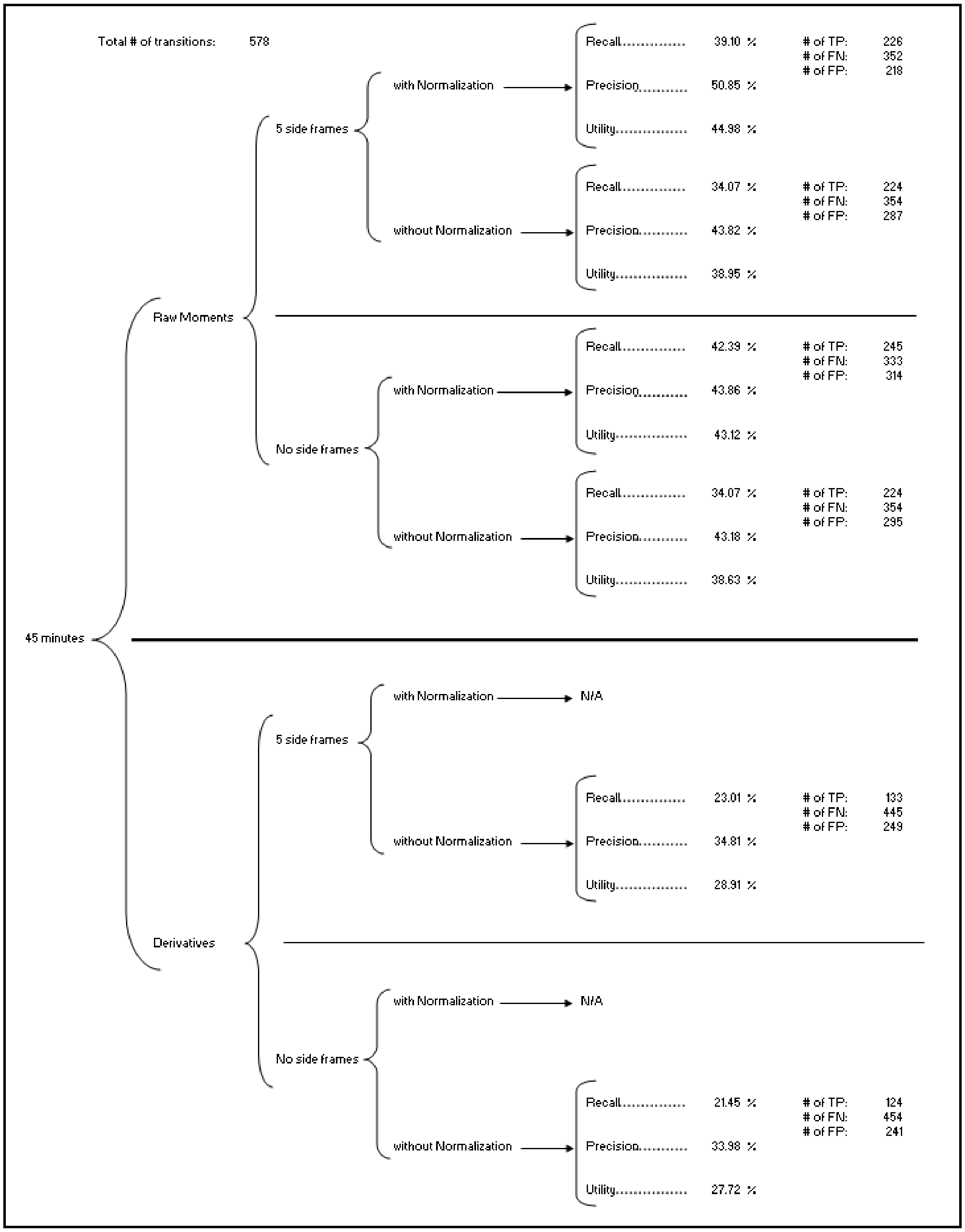

Total # of transitions: 578
45 minutes
Raw Moments
5 side frames
with Normalization
Recall.............. 39.10 %
Precision........... 50.85 %
Utility................ 44.98 %
# of TP: 226
# of FN: 352
# of FP: 218
without Normalization
Recall.............. 34.07 %
Precision........... 43.82 %
Utility................ 38.95 %
# of TP: 224
# of FN: 354
# of FP: 287
No side frames
with Normalization
Recall.............. 42.39 %
Precision........... 43.86 %
Utility................ 43.12 %
# of TP: 245
# of FN: 333
# of FP: 314
without Normalization
Recall.............. 34.07 %
Precision........... 43.18 %
Utility................ 38.63 %
# of TP: 224
# of FN: 354
# of FP: 295
Derivatives
5 side frames
with Normalization
N/A
without Normalization
Recall.............. 23.01 %
Precision........... 34.81 %
Utility................ 28.91 %
# of TP: 133
# of FN: 445
# of FP: 249
No side frames
with Normalization
N/A
without Normalization
Recall.............. 21.45 %
Precision........... 33.98 %
Utility................ 27.72 %
# of TP: 124
# of FN: 454
# of FP: 241

Fig. 5.2. Presents the experimentation results for 45 minutes of data.

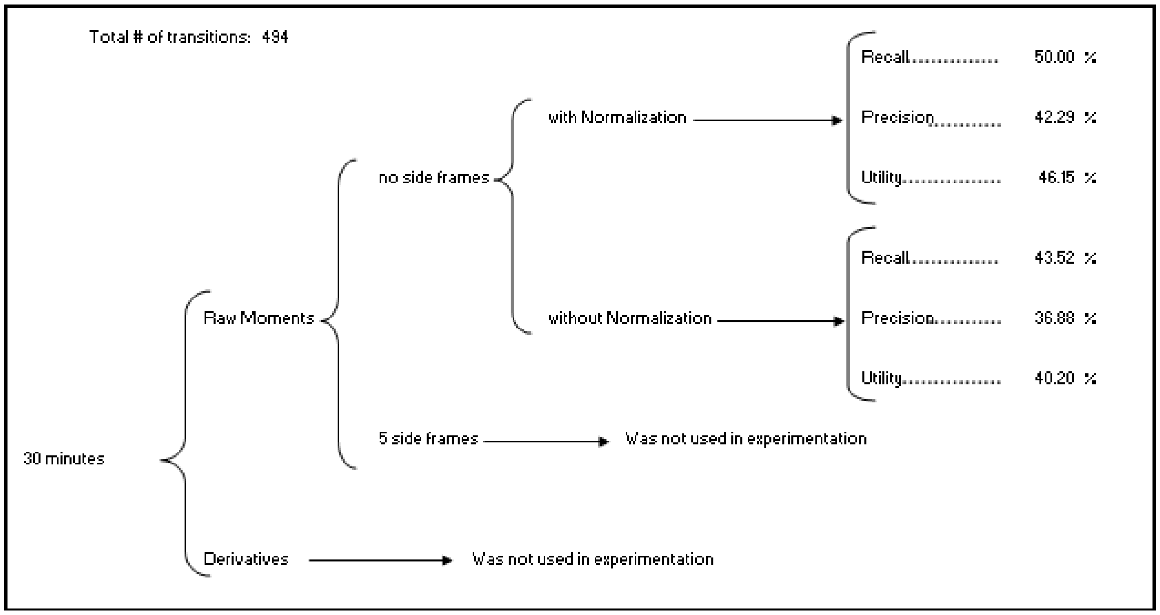

Total # of transitions: 494
Recall.............. 50.00 %
with Normalization
Precision........... 42.29 %
no side frames
Utility................ 46.15 %
Recall.............. 43.52 %
Raw Moments
without Normalization
Precision........... 36.88 %
Utility................ 40.20 %
5 side frames
Was not used in experimentation
30 minutes
Derivatives
Was not used in experimentation

Fig. 5.3. Presents the experimentation results for 30 minutes of data.

5.5.2. Direct Comparison based on Adaptive Examples

In the direct comparison based on adaptive examples algorithm, each transition is calculated separately and then the results from all detectors are combined in a final stage (see figure 4.19) before being evaluated. The results from each of the detectors are evaluated and presented to the reader at this point.

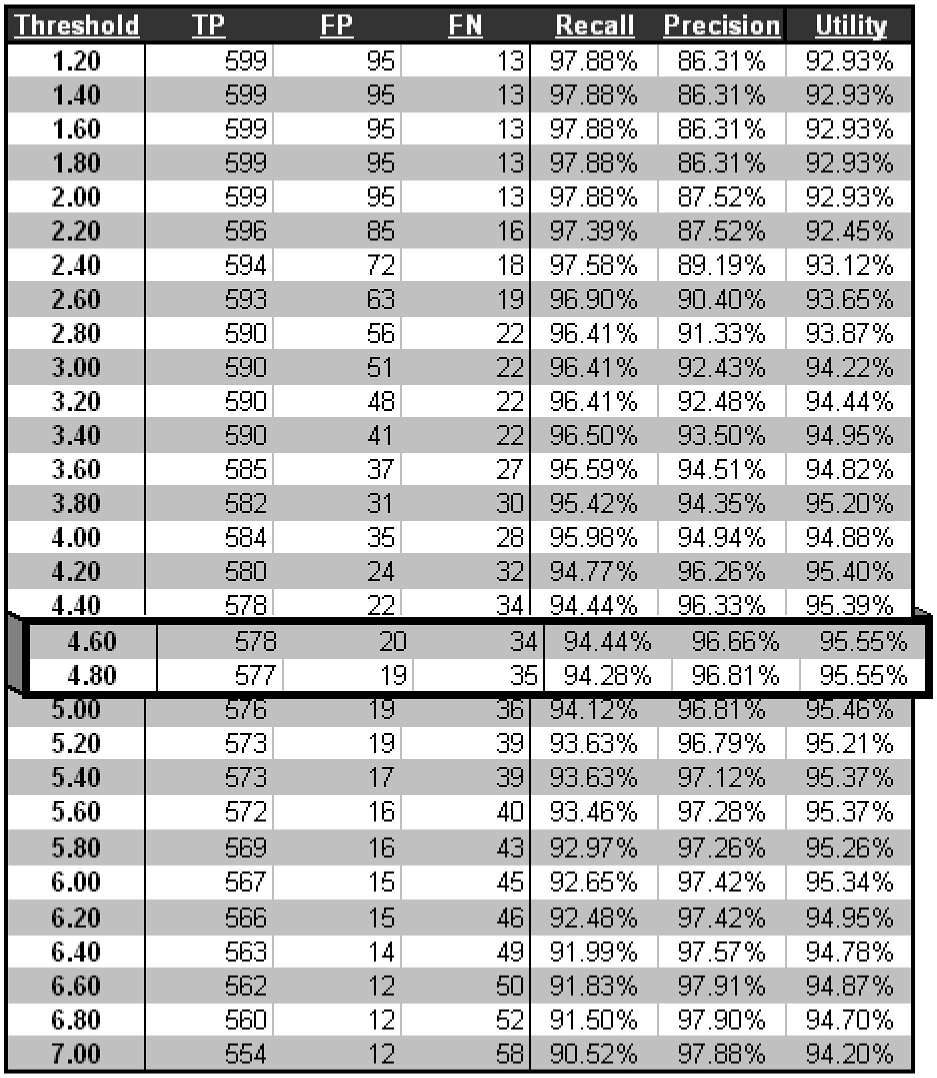
	Match (True Positives)	**False Alarm** (False Positives)	**Missed** (False Negative)	**Recall**	**Precision**	**Utility**
Cuts	578	20	34	94.44%	96.66%	95.55%
Fades	41	3	0	100.00%	93.18%	96.59%
Dissolves	57	40	3	95.00%	58.76%	76.88%
Total	676	63	37	94.81%	91.47%	93.14%

Table 5.1. Presents the final results of the second algorithm.

The table 5.1 represents the final results over an hour of data. Our results for both cuts and fades are excellent (over 95% utility), but our dissolve detection algorithm was not as precise and therefore the utility fell to 76%. The detailed experimentation results for each of the three primary transitions are discussed bellow.

Cuts

Our cut detection algorithm has only one parameter, a threshold that is used in the false positive detector and it acts as a controller for the trade off between false positives and false negatives. Table 5.2 contains all the TP, FP, FN, recall, precision and utility values. Figure 5.5 is the graph of utility values as a function of thresholds whereas figure 5.4 represents the ROC chart which is used to visualize the recall and precision values for each of the different possible thresholds. The best results are obtained through selection of a threshold between 4.6 and 4.8. Threshold 4.6 was used to obtain the results presented in Table 5.1.



Threshold	TP	FP	FN	Recall	Precision	Utility
1.20	599	95	13	97.88%	86.31%	92.93%
1.40	599	95	13	97.88%	86.31%	92.93%
1.60	599	95	13	97.88%	86.31%	92.93%
1.80	599	95	13	97.88%	86.31%	92.93%
2.00	599	95	13	97.88%	87.52%	92.93%
2.20	596	85	16	97.39%	87.52%	92.45%
2.40	594	72	18	97.58%	89.19%	93.12%
2.60	593	63	19	96.90%	90.40%	93.65%
2.80	590	56	22	96.41%	91.33%	93.87%
3.00	590	51	22	96.41%	92.43%	94.22%
3.20	590	48	22	96.41%	92.48%	94.44%
3.40	590	41	22	96.50%	93.50%	94.95%
3.60	585	37	27	95.59%	94.51%	94.82%
3.80	582	31	30	95.42%	94.35%	95.20%
4.00	584	35	28	95.98%	94.94%	94.88%
4.20	580	24	32	94.77%	96.26%	95.40%
4.40	578	22	34	94.44%	96.33%	95.39%
4.60	578	20	34	94.44%	96.66%	95.55%
4.80	577	19	35	94.28%	96.81%	95.55%
5.00	576	19	36	94.12%	96.81%	95.46%
5.20	573	19	39	93.63%	96.79%	95.21%
5.40	573	17	39	93.63%	97.12%	95.37%
5.60	572	16	40	93.46%	97.28%	95.37%
5.80	569	16	43	92.97%	97.26%	95.26%
6.00	567	15	45	92.65%	97.42%	95.34%
6.20	566	15	46	92.48%	97.42%	94.95%
6.40	563	14	49	91.99%	97.57%	94.78%
6.60	562	12	50	91.83%	97.91%	94.87%
6.80	560	12	52	91.50%	97.90%	94.70%
7.00	554	12	58	90.52%	97.88%	94.20%

Table 5.2. Presents number of true positives, false negatives, false positives, as well as recall, precision and utility for different thresholds used in false positive detector of cut detector.

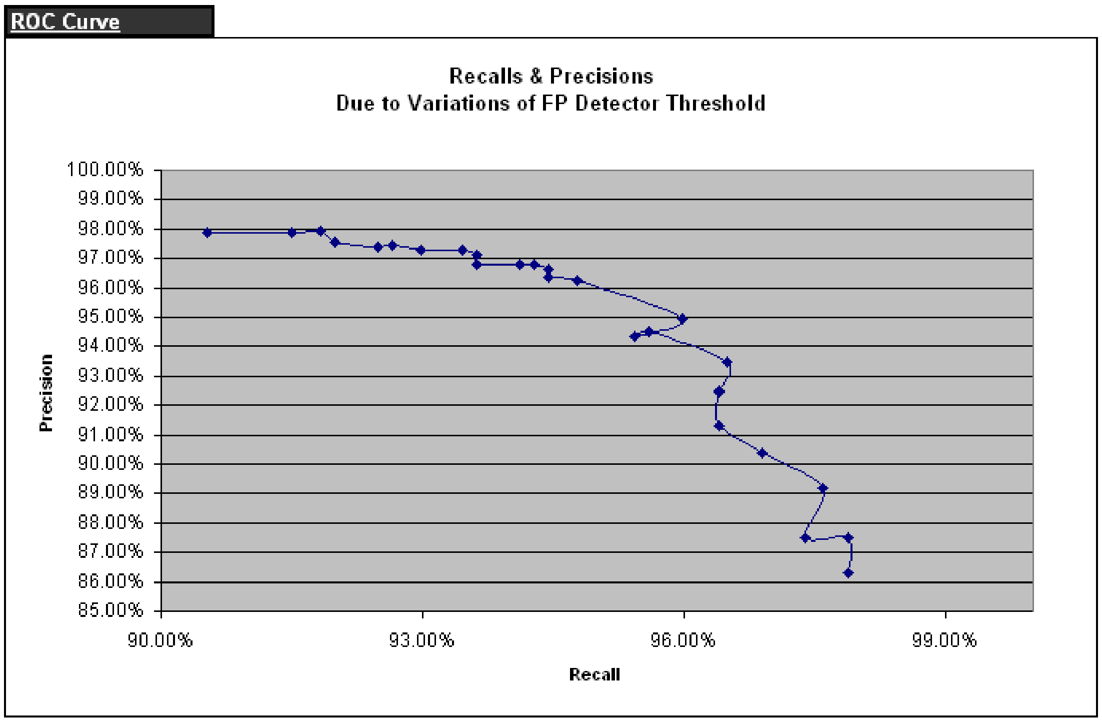


Fig. 5.4. Presents the recall and precision values for different thresholds used in false positive detector of cut detector as well as the ROC curve for the second algorithm.

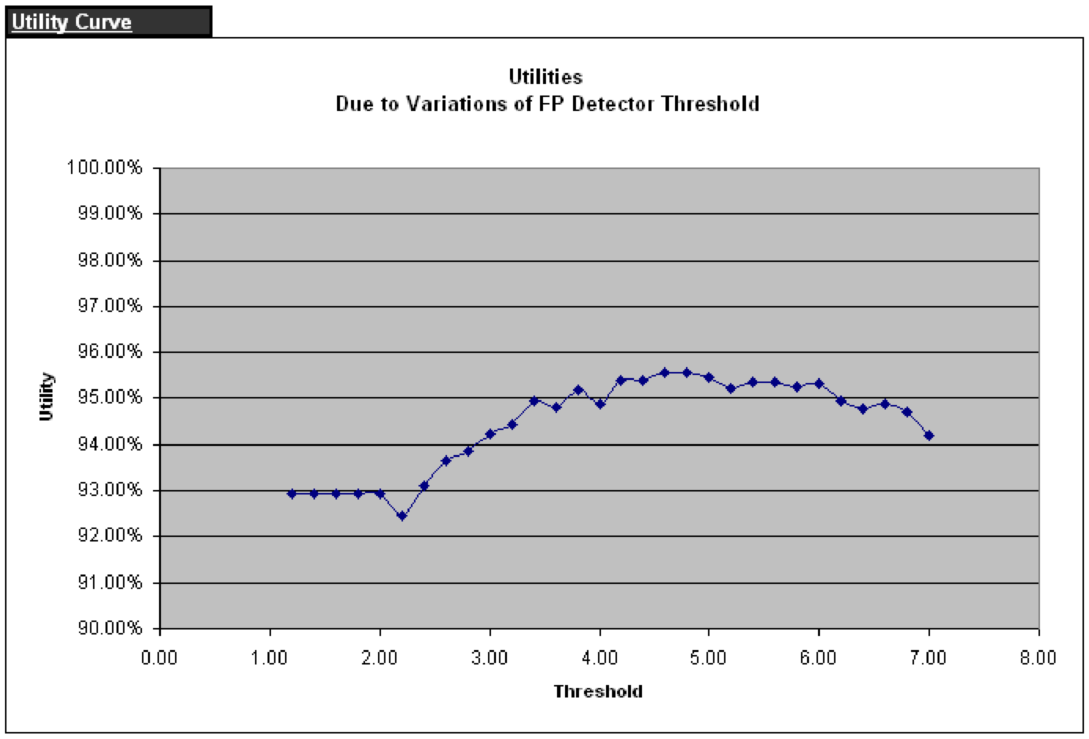


Fig. 5.5. Presents the utility values for different thresholds used in false positive detector of cut detector as well as the utility curve for the second algorithm.

Fades

The experimentations results for fades are presented in Table 5.1. Similar to Cuts it is possible to introduce a threshold for cuts to control the trade off between recalls and precisions without affecting the utility if possible, but we did not explore this option.

Dissolves

The experimentations results for dissolves are presented in table 5.1. Similar to Cuts it is possible to introduce a threshold for dissolve to control the trade off between recalls and precisions without affecting the utility if possible, but we did not explore this option.

5.6. Results Discussions

This section contains the discussion and analysis of the results presented in the preceding section for both algorithms.

5.6.1. Direct Comparison based on Predefined Examples

The derivatives were expected to result in a more desirable outcome than raw moments. However that was not the case in our experiment. The outcomes were due to the fact that this algorithm was designed for raw moments and hence did not perform as well with derivatives. However further investigation is necessary.

Normalization and amount of context used were the other criteria in our experimentation. As the results suggest, normalizing the data considerably improves the performance whereas using more context slightly raises the utility value.

The main issue is the number of examples used. If the number of examples is low (especially if raw moments are used) then it is impossible to find a close match for all the existing transitions. Hence a large number of examples are required to eliminate false negatives.

However as results demonstrate, having more examples does not solve the problem completely. This fact is due to increase in possibility of detecting a transition when there is actually no transition (i.e. the number FP will increase). Hence the utility score will remain about the same. To avoid this problem, a lot of examples have to be used (i.e. example set should contain thousands or even more examples).

Again having a lot of examples will also raise the execution time above an unacceptable limit, making it impossible for the algorithm to run in real time. Since the current algorithm takes a long time to execute, having a lot more examples is unacceptable. Therefore a completely different algorithm based on similar ideas is proposed.

5.6.2. Direct Comparison based on Adaptive Examples

As mentioned before the algorithm is parameter free because the values such as window size (such as cut window size) are either fixed for all types of videos or different possible sizes are used in the experimentation (such as dissolve window size).

We have not used any threshold during the detection stage and rather introduced a single threshold on our false positive detection. This makes it possible to reach very good or decent (in case of dissolve) precision values while maintaining exceptionally high recall values. Review of other programs results, reveals that our algorithm's performance has superiority over many other algorithms discussed in chapter 2 (However, this fact cannot be proven conclusively since different data sets were used for evaluation purposes.

After careful analysis of results presented in the *Results* section, one can divide the FPs and FNs into two groups: the false items which are preventable by simple adjustments (such as changing FP/FN tradeoff threshold) and the ones which require major changes in the algorithm (such as introduction of new data representation or completely new techniques which will add to the number of inputs used in parallel analyzers). Figures 5.6 to 5.11 demonstrate some of these cases. Most FPs in the finalized dissolve detection results were due to the fact that they were too close to the frames containing camera motion, high level of object motion, and high level of zoom.

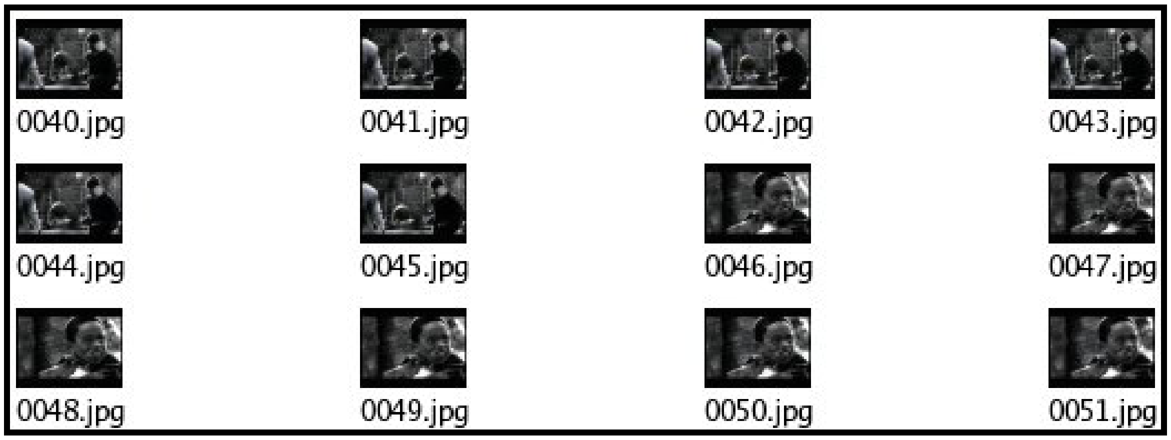


Fig. 5.6. Illustrates a scenario where the cut is between two frames with similar color intensities and distribution.

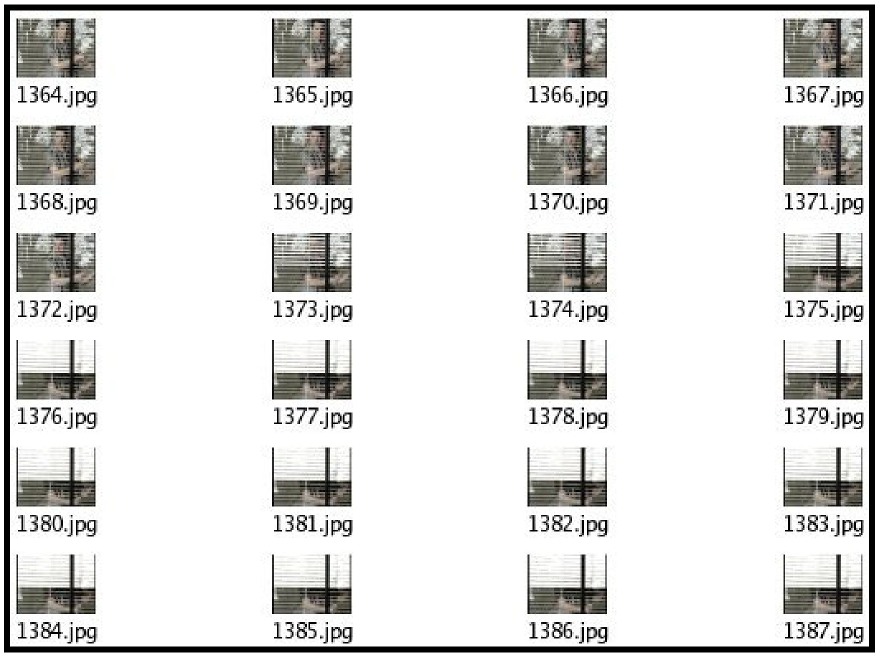


Fig. 5.7. Illustrates a scenario in which a person is closing the blinds, to be detected as a fade.

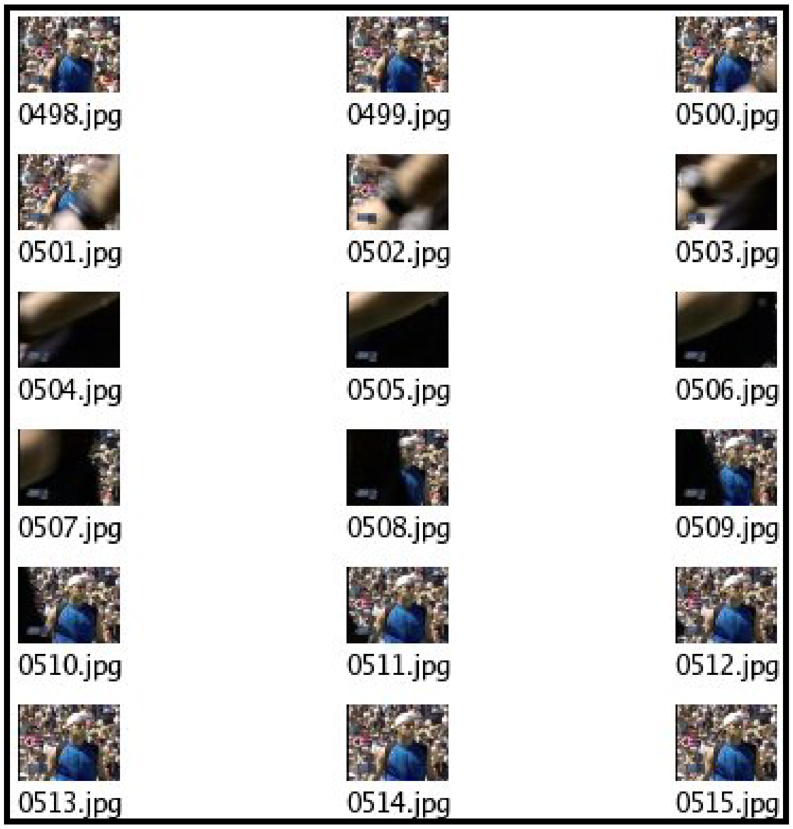


Fig. 5.8. Illustrates a close to the camera object motion that was detected as a fade.

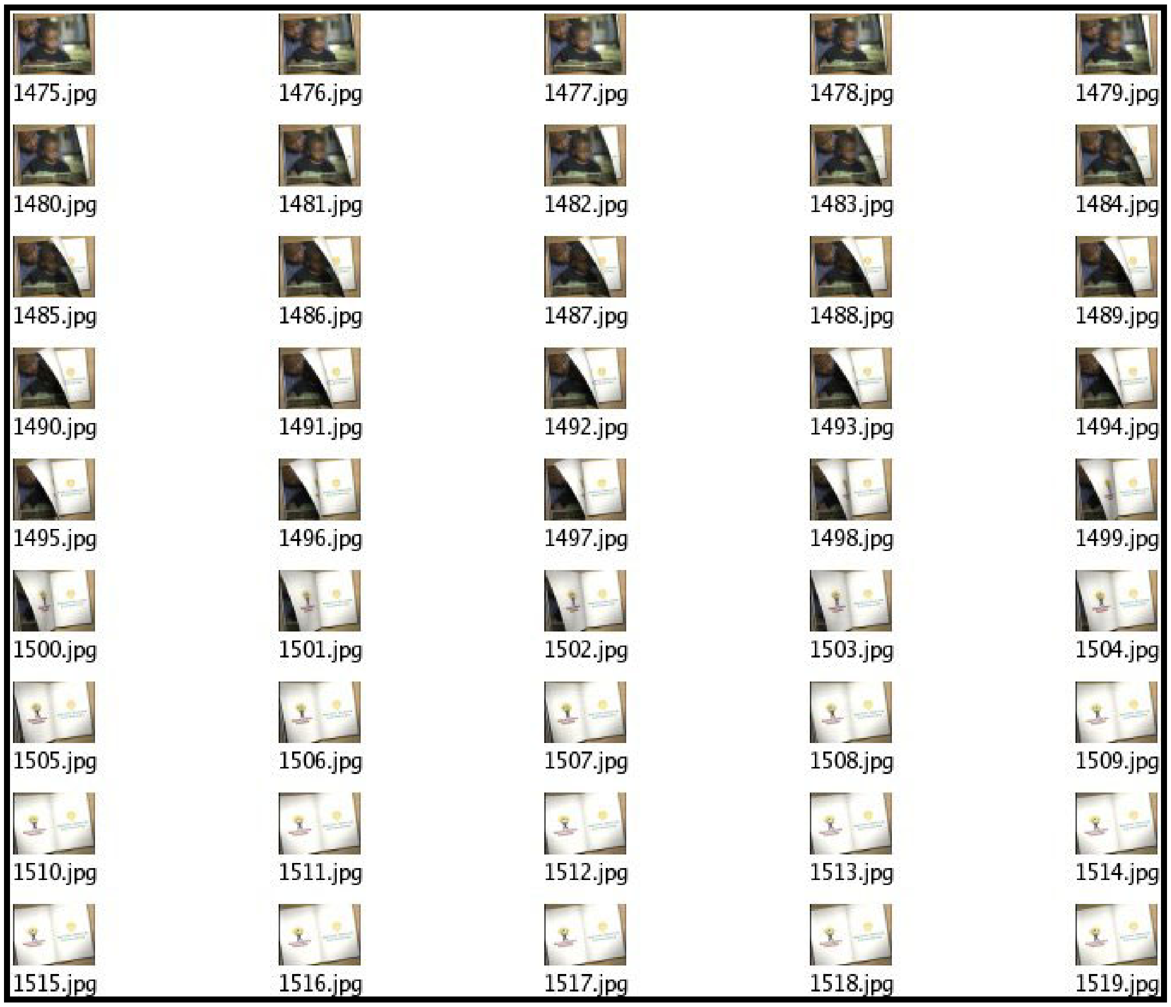


Fig. 5.9. Illustrates a graphical transition that was mistakenly labeled as dissolve.

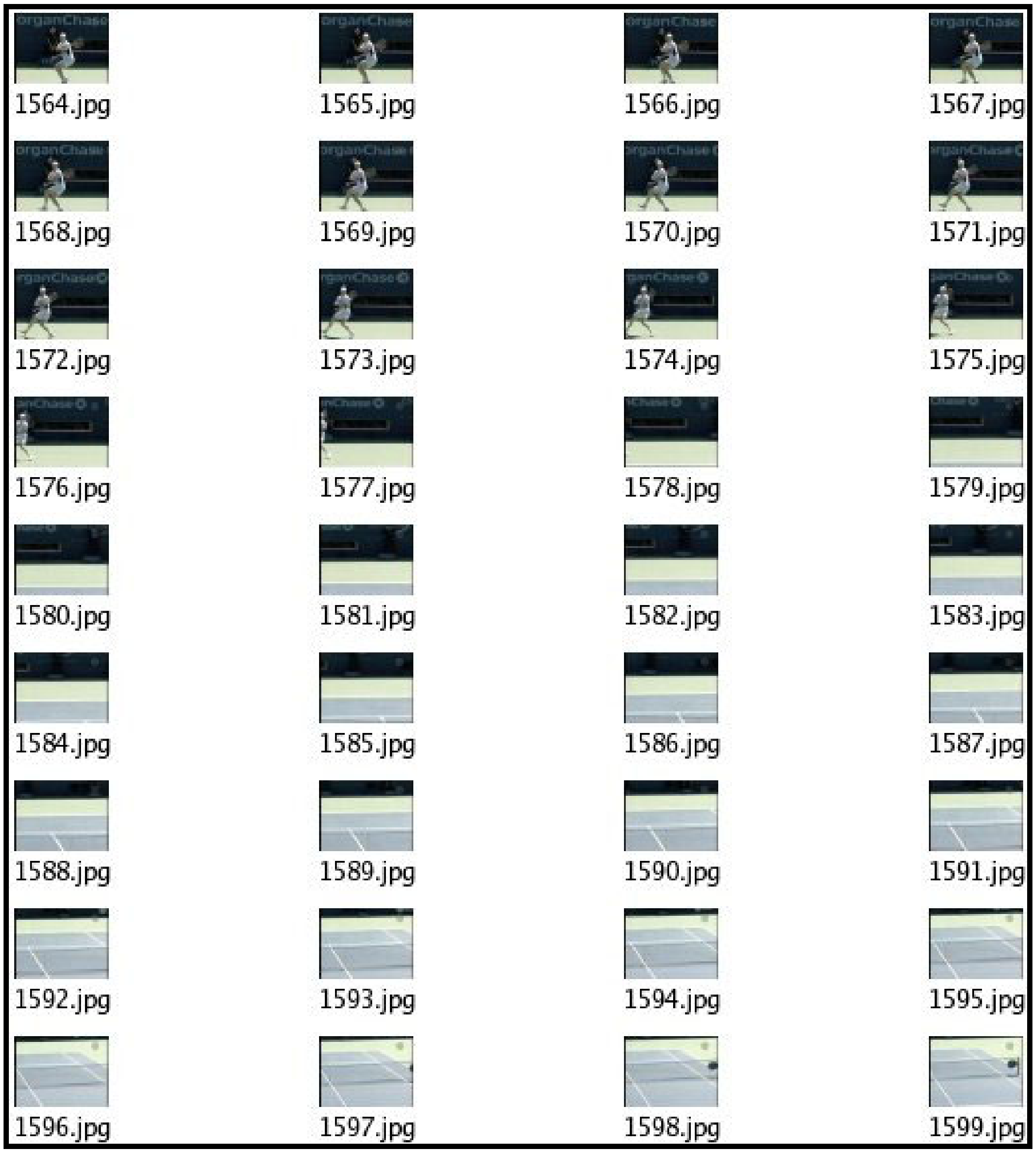


Fig. 5.10. Illustrates a camera motion with zoom activity that was mistakenly labeled as dissolve.

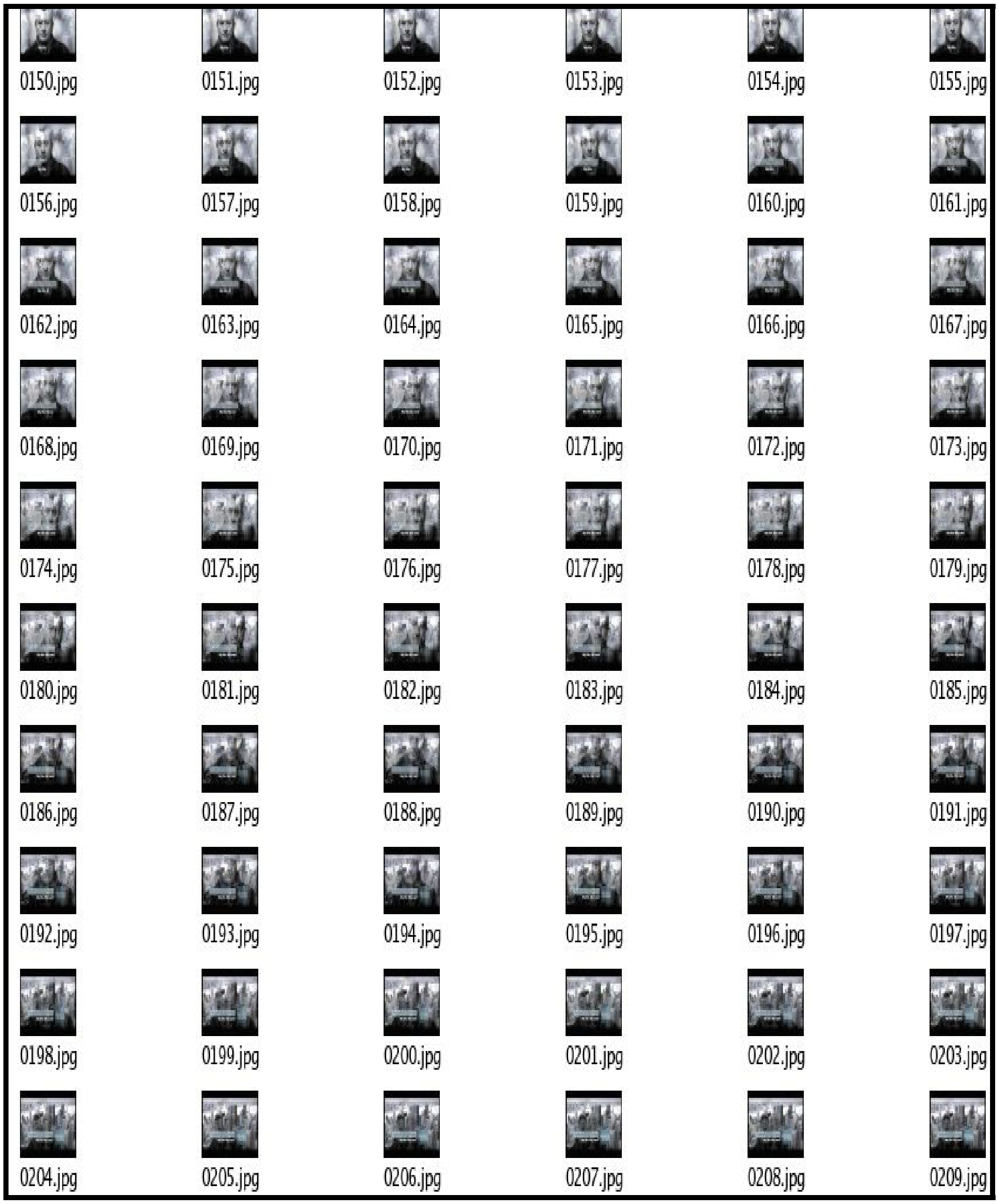


Fig. 5.11. Illustrates a scenario which was missed by fade detector. It is caused due to the very lengthy and uncommon fade and the fact that the shots are in black and white.

Figures 5.12 presents example of a dissolve transition which was detected successfully whereas figure 5.13 present such example for a fade transition.

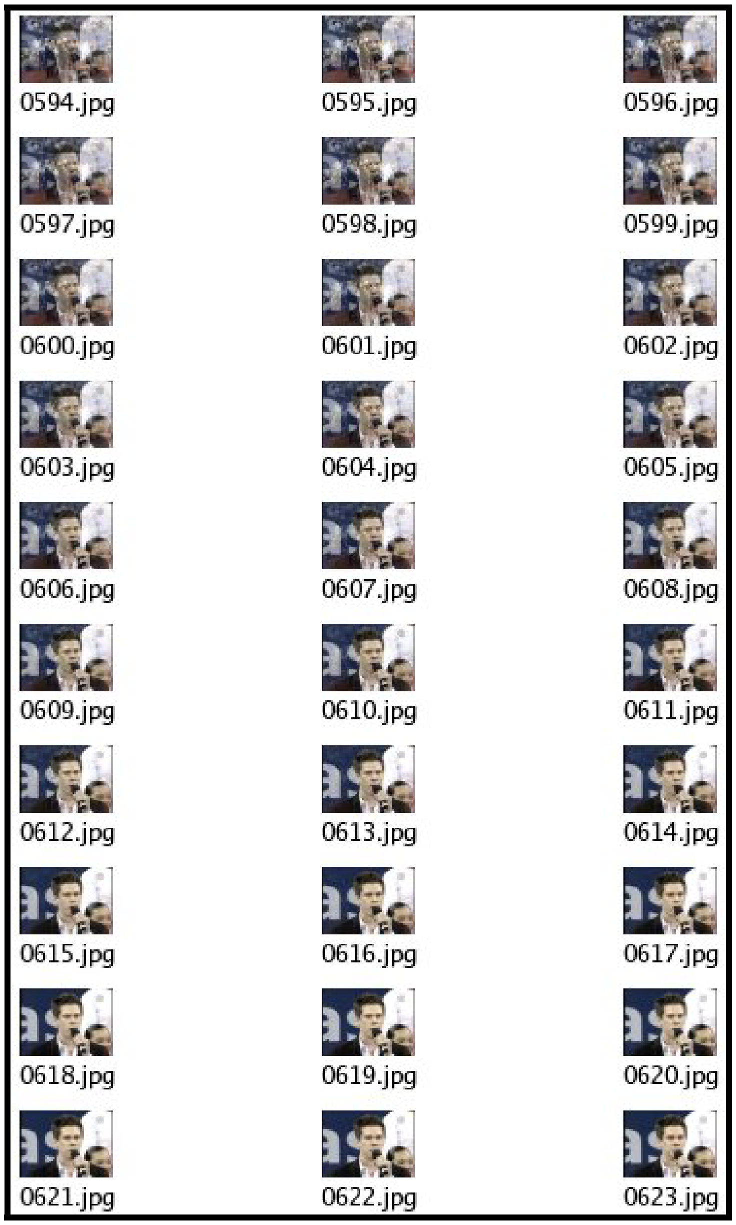


Fig. 5.12. Illustrates a dissolve which was successfully detected.

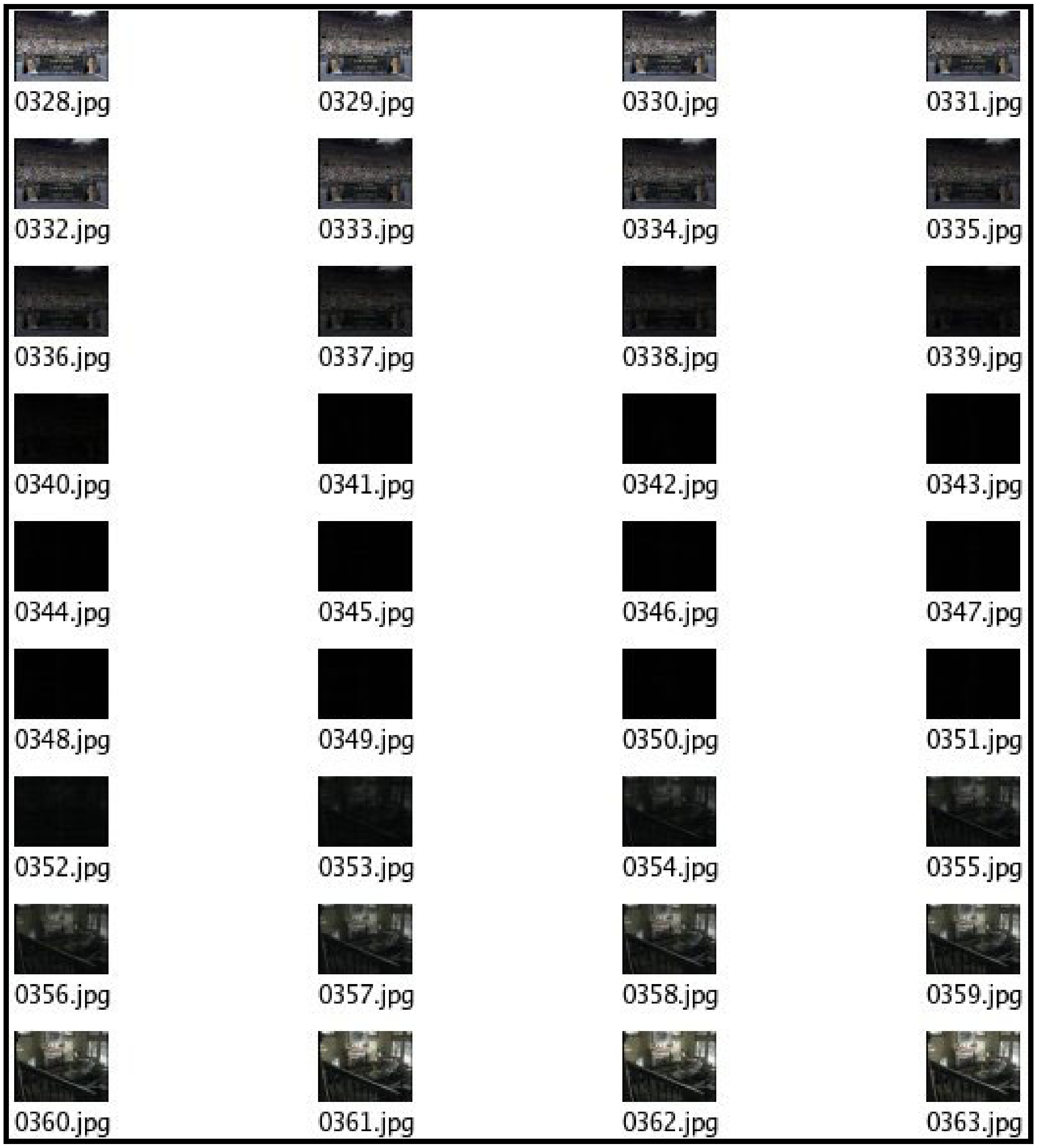


Fig. 5.13. Illustrates a fade which was successfully detected.

Figures 5.14 and 5.15 present examples of false positives which were detected and eliminated. They were eliminated in the *parallel analyzer* by comparing the inner detector detection streams with the previously detected transitions.

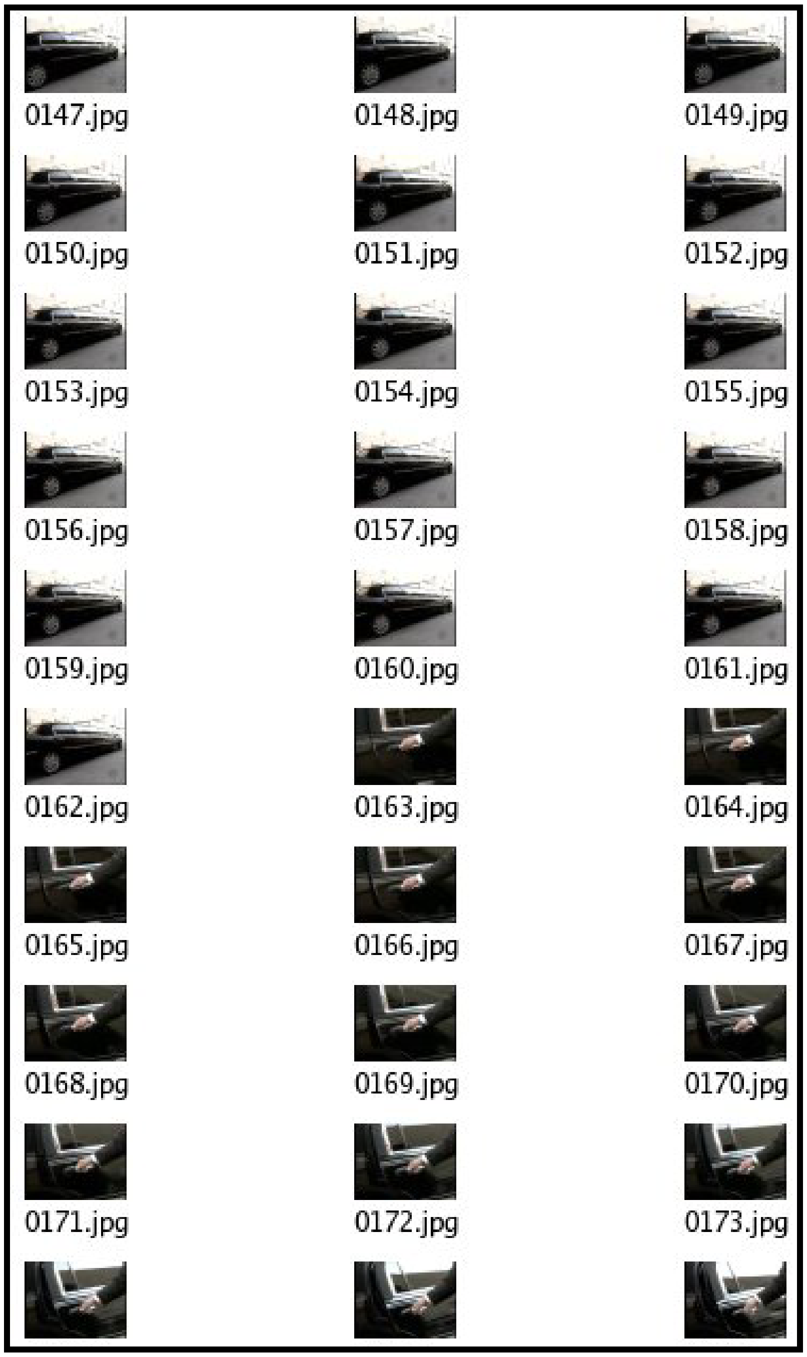


Fig. 5.14. Illustrates a scenario which was detected as a cut and also mistakenly as a dissolve but was corrected by the algorithm.

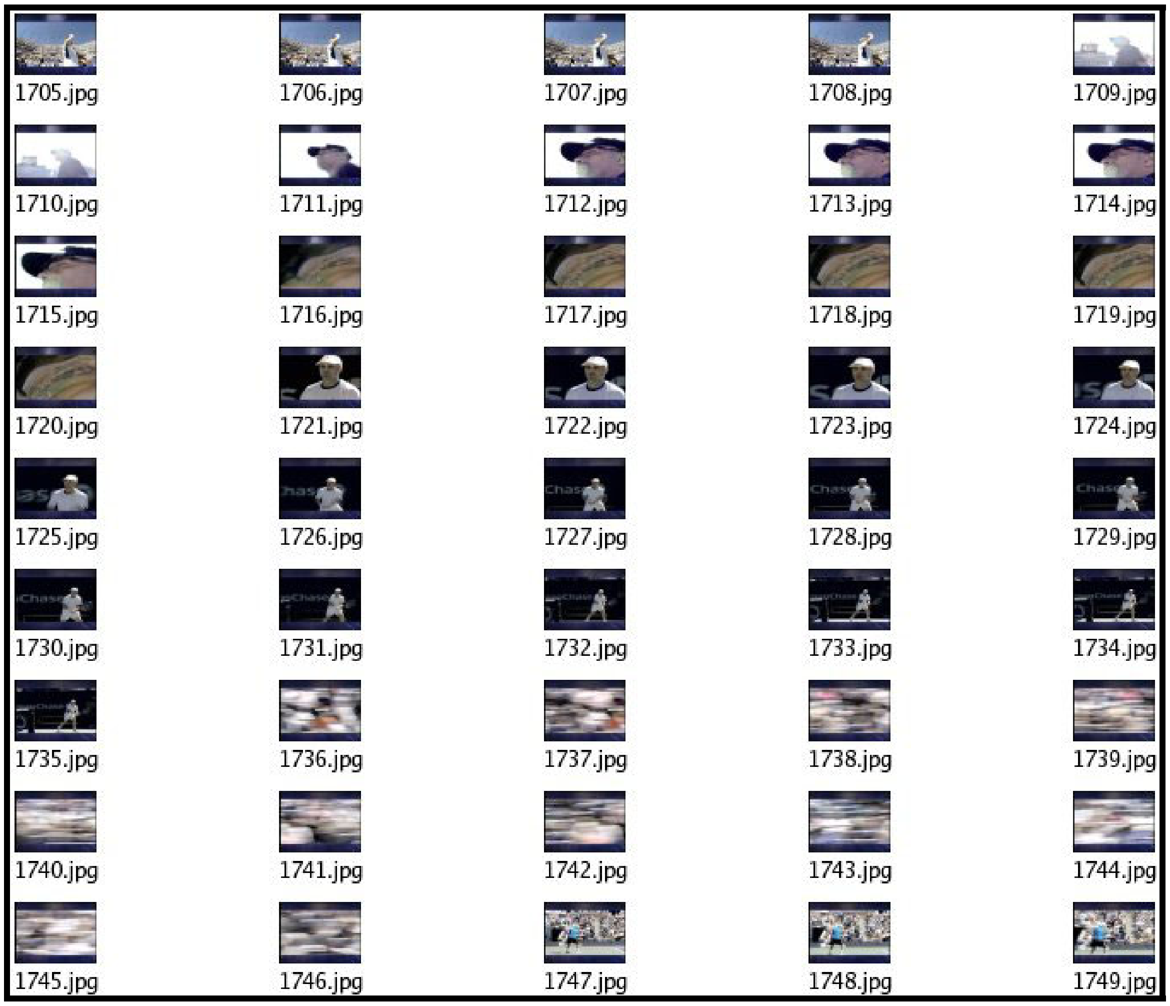


Fig. 5.15. Illustrates a scenario of high activity in which existence of too many cuts, change in brightness, motion and effects resulted in a FP (around frame 1715) in final dissolve detection results which was later fixed since it was too close to detected cuts.

In utility graph (figure 5.5), the thresholds values between 1 and 2 will not affect the number of FPs and FNs in any ways. This is one way to detect FPs and FNs which can only be solved through introduction of new representation and/or a new technique rather through simple techniques.

As mentioned before cuts and normal groups used a fixed window size while different sizes where used for gradual transitions. The reason is that the cuts do not have any actual length since they are an abrupt change between two frames. Therefore at any given time (frame number) the upcoming cut will be detected by the larger windows first. The cut will also exit the larger sliding windows last (see figure 5.16). Hence, having different window sizes for cuts become irrelevant.

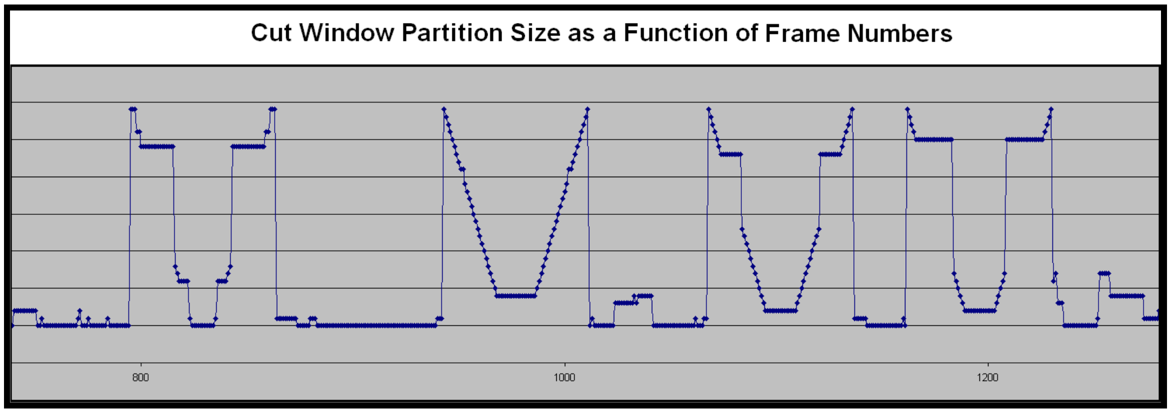


Fig. 5.16. Illustrates usage of different window sizes for cuts.

Another downside to using various lengths for cuts is that it causes the cuts of close proximity to merge and therefore be missed. It will also slow down the algorithm slightly. Therefore only two sizes are used in this algorithm and they are analyzed in parallel (window with partition size of four and eight – refer to previous section for more details).

Another main advantage of the new algorithm is its low execution time which makes it possible to run in real time. Table 5.3 presents the execution times needed at each step for preparing and analyzing one minute of data.

As can be seen in execution time table this algorithm requires less than a quarter of the permissible execution time for a real time method. This will allow for future expansions which is a big advantage compare to other techniques.

	Execution Time
Statistical Data Preparation	1.0 seconds
Cuts	2.1 seconds
Fades	10.8 seconds
Dissolves	0.3 seconds
Total	14.2 seconds

Table 5.3. Presents the time performance of the second algorithm.

5.7. Conclusion

This chapter provided the details on performance measurement and evaluation methodologies, some of the terminologies related to temporal segmentation research, the results of our experimentations and finally the discussion and analysis of the results.

Next chapter presents a brief overview of major topics discussed in this document, suggestions and explanations for future works, the glossary of commonly used terms and finally the bibliography information.

Chapter 6

Conclusions

6. Conclusions

6.1. Summary

This document presented a comprehensive survey of various works in the field of temporal segmentation of video (chapter 2). It also discussed the design and implementation details for two algorithms, one based on predefined examples (chapter 3) and the second based on adaptive examples (chapter 4). The latter algorithm was the centerpiece of this research and introduced many novel techniques and ideas such as adaptive examples, no threshold, parallel analysis, extremely sensitive change detector, uncertainty groups, layered architecture, and frame scoring system with overlapping window. The algorithms of chapter 3 and 4 were thoroughly tested and the experimentation results and the discussions of the results were presented in chapter 5.

This chapter contains brief conclusion for *Temporal Segmentation*, and *Future Work* section in which future related research areas are emphasized and also future possible improvements to both algorithms are suggested. This section is followed by *Glossary* where some of the definitions unique to this document as well as some of the more commonly used definitions in the area of video segmentation are defined in one place for easy access.

6.2. Temporal Segmentation

Temporal segmentation is the basis for many other video related research topics. After a comprehensive literature review and implementation of some simple shot detection methods, the need for a new technique became more obvious. In this document many advanced and novel techniques were presented all of which are described in chapter 3 and 4. Our algorithms were tested on one hour of video obtained from a typical twenty four hours broadcast of CBS channel. It used the three primary color moments mean, standard deviation and skew to represent frames within a video. *Direct Comparison based on Adaptive Examples* method out performed many other algorithms for all three primary transitions (cuts, fades, and dissolves).

6.3. Future Work

This section provides ideas for expansion and improvements of the algorithms discussed in this document as well as suggestions for future approaches.

6.3.1. Enhancements and Improvements

This section provides suggestions for future enhancements and improvements of the two algorithms discussed in this document.

Direct Comparison based on Predefined Examples

Making future enhancements and improvements to this algorithm is difficult. One critical issue is the trade off between number of examples and execution time. Having too many examples leads to many unnecessary false positives whereas too few examples lead to many unnecessary false negatives. It is also difficult to modify the algorithm and still guarantee simplicity and generality.

Use of supplementary video representation techniques, is the main suggested improvement. This topic and other suggestions are discussed further in *Other Suggestions* section.

Direct Comparison based on Adaptive Examples

This section contains only the suggested ideas and improvements for the second algorithm, direct comparison based on adaptive examples.

The fade detector can be improved on by finding the actual monochrome frames rather than predefining them. In other words, the monochrome frames detector has to be improved and tested thoroughly to ensure the detection of all monochrome frames. Use of the actual monochrome will result in more reliable fade adaptive examples, which leads to higher detection quality.

The detection quality can also be improved by finding the exact location of the transitions. Currently the algorithm does not find the transitions boundaries as precisely as possible. For different situations, specific conditions have to be met to detect these boundaries as accurate as possible.

The time performance of the second algorithm is very promising. Hence, it must be expanded upon so it allows analysis of real time input stream. As it is, the algorithm by itself only supports the analysis of one minute of data during each run.

The last suggestion for the second algorithm is to provide a solution for the unusually long (longer than 40 frames) fades and dissolves. This algorithm also needs to precisely detect the transition boundaries.

Other Suggestions

This section contains the suggested ideas and improvements which can be applied to both algorithms.

One way to improve the detection quality is through introduction of new moments. In other words, more features have to be extracted from the video and provided as inputs to the algorithms. Two sets of numerical data are recommended: edge detection data and motion analysis results. Both of these techniques are reviewed in detail in chapter 2. Another option to consider is to combine different existing video representation techniques such as histogram, intensity and spatial differences [42].

Another useful resource, besides video image representation, is audio track. Audio track was not discussed in depth; however it deserves further considerations.

Besides use of supplementary video representation techniques, higher detection quality can be achieved through further enhancements of false positive detector. These detectors can be enhanced via introduction of several specific algorithms for each of the various types of false positive causing effects. Some of the notable effects include overlays transitions, fast camera or object motion, close to camera object motion and change in brightness.

Number of false positives can also be reduced through introduction of other transitions and effects detectors. For example many graphical effects and transitions are mistakenly labeled as cuts, fades and dissolves. This problem can be prevented by introduction of a graphical transitions detector.

6.3.2 Next Generation Algorithm

One of the main goals of this thesis was to implement a general algorithm which detects all three types of transitions at the same time, an algorithm that can be expanded to other transitions and effects without the need for too many modifications. The first algorithm was designed to meet this goal, however due to various problems (mainly the tradeoff between number of predefined examples and execution time), it had to be discarded.

Even though the second algorithm less specific than those based on mathematical models, it still requires each transition or effect in question to be analyzed separately. Hence, this section proposes a new method based on the second algorithm which meets the generality property.

As suggested earlier the new algorithm has to consider all types transitions and effects at the same time. In other words, similar to the algorithm based on adaptive examples, many detectors have to be used to generate detection streams for each individual transition or effect in question.

The distinction from the previous method is in fit values. The fit values have to be uniform, meaning they should be in the same range and unlike previous approach the same algorithm should be used for generating normal groups for gradual and abrupt transitions.

Another issue arises while comparing fit values of cuts and shorter gradual transitions with fit values of longer gradual transitions and effects. Due to the length and object motions, shorter transitions will have more desirable fit values, causing bias in the algorithm and therefore result in a lower performance level than expected.

After fit values uniformity is achieved, a clustering technique, similar to those used in [42] and [46] should be applied to create different groups of transitions and identify the types of any future transitions.

6.4. Contact Information

Any questions, suggestions, and/or comments can be sent to the author via the following email address:

rbyeganeh@gmail.com

Any type of feedback is greatly appreciated and welcomed.

Bibliography

Bibliography

[1] Abramowitz, M. and Stegun, I. A., Eds., “Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.” New York: Dover, 1972.

[2] Adelson, E. H. and Bergen, J. “Spatiotemporal energy models for the perception of motion,” in *J. Opt. Soc. Amer.*, vol. 2, no. 2, pp. 284–299, February 1985.

[3] *Adobe Premiere 4.0 Handbuch*, Adobe Systems, San Jose, CA, USA, 1995.

[4] Ahanger, G. and Little, T. “A Survey of Technologies for Parsing and Indexing Digital Video,” in *J. Visual Communication Image Represent.*, vol. 7, no. 1, pp. 28–43, March 1996.

[5] Akutsu, A., Tonomura, Y., and H. Hamada, “Videostyler: Multi-dimensional video computing for eloquent media interface,” in *Proc. Int. Conf. on Image Processing*, vol. 1, pp. 330–3, 1995.

[6] Albanese, M., Chianese, A., Moscato, and V., Sansone, L., “Dissolve Detection in a Video Sequence Based on Animate Vision,” in *Proceedings of the 9th International Workshop on Multimedia Information Systems* (MIS 2003), pp. 115-122, Ischia, Italy, May 26-28, 2003.

[7] Albiol, A., Fulla, M. J., Albiol, A., Torres, L., "Detection of TV Commercials," In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, volume 3, Montreal, Canada, pp. 541-544, May 2004.

[8] Amir, A., Hsu, W., Iyengar, G., Lin, C.-Y., Naphade, M., Natsev, A., Neti, C., Nock, H. J., Smith, J. R., Tseng, B., Wu, Y., and Zhang, D., "IBM Research TRECVID-2003 video Retrieval system," in *NIST Text Retrieval Conference (TREC)*, Gaithersburg, Maryland, November 2003.

[9] Amir, A., Chang, S.-F., Franz, M., Iyengar, G., Kender, J. R., Lin, C.-Y., Naphade, M. R., Natsev, A., Smith, J. R., and Teˇsi´c, J.. "IBM Research TRECVID-2004 video retrieval system," in *NIST Text Retrieval Conference (TREC)*, Gaithersburg, Maryland, November 2004.

[10] Amir, A., Argillander, J., Campbell, M., Haubold, A., Iyengar, G., Ebadollahi, S., Kang, F., Naphade, M. R., Natsev, A., Smith, J. R., Tesic, J., and Volkmer, T., "IBM Research TRECVID-2005 Video Retrieval System," in *NIST Text Retrieval Conference (TREC)*, Gaithersburg, Maryland, November 2005.

[11] Araújo, A. A., Bouthemy, P., Chávez, G. C., Cord, M., Dahyot, R., Lehmann, A., Licsár, A., Szirányi, T., and Yao, J.-F., "Progress on Applications of Machine Learning Techniques," Ed. Rozenn Dahyot, in *Muscle Consortium*, September 2005.

[12] Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y., "An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions," in *J. ACM: Journal of the ACM*, 45(6):891-923, November 1998.

[13] Ayer, S., Schroeter, P., and Bigun, J., "Segmentation of moving objects by robust motion parameter estimation over multiple frames," in *Third European Conf. on Computer Vision*, pp. 316-327, Vol II, Stockholm, Sweden, May 1994.

[14] Ballard, D. H., "Animate Vision," in *Artificial Intelligence*, vol. 48, 57-86. February 1991.

[15] Bolles R. C., and Baker, H. H. "Epipolar plane image analysis: An approach to determining structure from motion," in *Int. J. Comp. Vis*., vol. 1, no. 1, pp. 7–55, 1987.

[16] Boreczky, J. S., and Rowe, L., "Comparison of Video Shot Boundary Detection Techniques," in *Storage and Retrieval for Image and Video Databases (SPIE)*, pp. 170-179, 1996.

[17] Boreczky, J. S., and Wilcox, L. D., "A Hidden Markov Model Framework for Video Segmentation using Audio and Image Features," in *Proc. Int. Conf. Acoustics, Speech, and Signal Proc.*, 6, Seattle, 1998, pp. 3741-3744.

[18] Bouthemy, P., Gelgon, M., and Ganansia, F., "A Unified Approach to Shot Change Detection and Classification and Camera Motion Characterization," in *Tech Rep. 1148,* Institute National de Recherche en Informatique et en Automatique, November 1997.

[19] Bruno, E., and Pellerin, D., "Video Shot Detection Based on Linear Prediction of Motion," in *ICME*, 2002.

[20] Canny, J. "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Retrieval for Still Image and Video Databases IV*, Proc. SPIE 2664, pp. 170-179, January 1996.

[21] Cheng, W.-H., Chu, W.-T., and Wu, J.-L., "A Visual Focus Detection Framework for Video Sequences," in *The 2004 Workshop on Consumer Electronics and Signal Processing* (*WCEsp2004*), Hsinchu, Taiwan, November 17, 2004.

[22] Chien, Y. T., and Fu, K.S. "On the generalized Karhunen-Loeve expansion," in *IEEE Transaction on Information Theory*, vol. 13, no.3, pp.518-520, July 1967.

[23] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: a library for support vector machines", 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[24] Chua, T.-S., Feng, H. M. and Anantharamu, C., "An Unified Framework for Shot Boundary Detection via Active Learning," in *Proc. Of ICASSP*, vol. II, pp. 845-848, Hong Kong, 2003.

[25] Chua, T.-S., Kankanhalli, M., and Lin Y., "A General Framework for Video Segmentation Based on Temporal Multi-Resolution Analysis," in *Proc. Of Int'l Workshop on Advanced Image Technology*, pp. 119-124, Fujisawa, Japan, 2000.

[26] Cooper, M., and Foote, J., "Scene Boundary Detection via Video Self-Similarity Analysis," in *Proc. IEEE Int. Conf. on Image Processing*, pp. 378-381, 2001.

[27] Cooper, M., Foote, J., Adcock, J., and Casi, S., "Shot Boundary Detection via similarity analysis," in *Proceedings of the TRECVID 2003 Work-shop*, pp. 79-84, Gaitherburg, Maryland, USA, 2003.

[28] Darrel, T., and Pentland, A. P., "Cooperative Robust Estimation using Layers of Support," in *IEEE Trans. Pattern Recognit. Machine Intell.*, vol. 17, no. 5, pp. 474-487, 1995.

[29] Dailianas, A., Allen, R. B., and England, P., "Comparison of Automatic Video Segmentation Algorithms," in *Proc. SPIE*, vol. 2615, pp. 2–16, 1996.

[30] Drew, M. S., Li, Z.-N., and Zhong, Xiang, "Video dissolve and wipe detection via spatio-temporal images of chromatic histogram differences," in *ICIP'00*, 2000.

[31] Dufaux, F., and Konrad, J., "Efficient, Robust and Fast Global Motion Estimation for Video Coding," in *IEEE Trans. on Image Processing*, vol. 9, no. 3, pp. 497-501, March 2000.

[32] Dugad, R., Ratakonda, K., and Ahuja, N., "Robust Video Shot Change Detection," in *IEEE Workshop on Multimedia Signal Processing*, December 1998.

[33] Ewerth, R., and Freisleben, B., "Frame Difference Normalization: An Approach to Reduce Error Rates of Cut Detection Algorithms for MPEG Videos," in *Proc. Of IEEE International Conference on Image Processing*, vol. II, Barcelona, pp. 1009-1012, 2003.

[34] Ewerth, R., and Freisleben, B., "Improving Cut Detection in MPEG Video by GOP-Oriented Frame Difference Normalization, " in *Proc. of the 17th Int'l Conf. on Pattern Recognition* (*ICPR 2004*), vol. 2, pp. 807-810, Cambridge, United Kingdom, August 2004.

[35] Ewerth, R., Friese, T., Grube, M., and Freisleben, B., "Grid Services for Distributed Video Cut Detection," in *Proc. Of the IEEE Sixth Int. Symp. on Mult. Software Engineering* (*ISMSE'04*), vol 00, pp. 164-168, Washington, DC, 2004.

[36] Ewerth, R., and Freisleben, B., "Video Cut Detection without Thresholds," in *Proc. of 11th Workshop on Signals, Systems and Image Processing* (*PTETiS*), pp. 227–230, Poznan, Poland, 2004.

[37] Ewerth, R., Schwalb, M., Tessmann, P., and Freisleben, B., "Estimation of Arbitrary Camera Motion in MPEG Videos," in *Proc. of the 17th Int'l Conf. on Pattern Recognition*, vol. 1, pp. 512-515, Cambridge, United Kingdom, 2004.

[38] Fablet, R., Bouthemy, P., and Perez, P., "Non-parametric Motion Characterization using Causal Probabilistic Models for Video Indexing and Retrieval," in *IEEE Trans. on Image Processing*, vol. 11(4), pp. 393-407, 2002.

[39] Fang, J., Dong, Y., Lushington, G. H., Ye, Qi-Zhuang, and Georg, G. I., "Support Vector Machines in HTS Data Mining: Type I MetAPs Inhibition Study," in *J. Biomol Screen*, pp. 134-144, March 2006.

[40] Fehske, A., Gaeddert, J., and Reed, J. H., "A New Approach to Signal Classification using Spectral Correlation and Neural-Networks," in *Proc. IEEE Int'l Symposium. New frontiers Dynamic Spectr. Access Networks*, vol. I, pp. 144-150, Baltimore, MD, November 2005.

[41] Fukunaga, K., "Introduction to Statistical Pattern Recognition," 2nd ed. Academic Press Prof., Inc., San Diego, CA, 1990.

[42] Gao, X., and Tang, X., "Unsupervised Video-Shot Segmentation and Model-Free Anchorperson Detection for News Video Story Parsing," in *IEEE Trans.on Circuits and Systems for Video Technology*, vol. 12, no. 9, pp. 765-776, 2002.

[43] Gargi, U., Kasturi, R., and Strayer, S. H., "Performance Characterization of Video-Shot-Change Detection Methods," in *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 1, pp.1-13, February 2000.

[44] Gish, H., Siu, M., Rohlicek, R. "Segmentation of Speakers for Speech Recognition and Speaker Identification," in *Proc. of the IEEE Int'l Conf. Acoustics, Speech, and Signal Processing*, vol. 2, pp. 873-876, Toronto, Canada, May 1991.

[45] Gonzalez, R. C., and Woods, R. E., "Digital Image Processing," Addison-Wesley, Reading, Mass., 2002.

[46] Günsel, B., Ferman, A. M., and Tekalp, A. M., "Temporal Video Segmentation using Unsupervised Clustering and Semantic Object Tracking," in *J. of Elec. Imaging*. 7 (3), pp. 592-604, 1998.

[47] Günsel, B., Ferman, A. M., and Tekalp, A. M., "Video Indexing through integration of Syntactic and Semantic Features," in *Proc. Workshop Applications of Computer Vision*, pp.90-95, Sarasota, FL, 1996.

[48] Habibi, A., and Wintz, P.A., "Image Coding by Linear Transformation and Block Quantization Techniques," *IEEE Transaction on Communication and Technology*. vol. 19, pp. 948-956, 1971.

[49] Hampapur, A., Jain, R. C., and Weymouth, T., "Digital Video Segmentation," in *Proc. ACM Multimedia 94*, pp. 357-364, San Francisco, CA, October, 1994.

[50] Hampapur, A., Jain, R. C., and Weymouth, T., "Production Model Based Digital Video Segmentation," in *Multimedia Tools and* Applications, vol. 1, no. 1, pp. 9-46, March 1995.

[51] Hanjalic, A., "Shot Boundary Detection: Unraveled and Resolved?" in *IEEE Transactions on Circuits and System for Video Technology*, Vol. 12, pp. 533-544, 2002.

[52] Hsu, C., Chang, C., and Lin, C., "A Practical Guide to Support Vector Classification," 2003, available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html

[53] Huttenlocher, D., Klanderman G, Rucklidge W., "Comparing images using the Hausdorff distance," *IEEE Trans. Pattern Anal Mach Intell*, 15(9):850–863, 1993.

[54] Huttenlocher, D., Jaquith E., "Computing visual correspondence: Incorporating the probability of a false match," in *5th International Conference on Computer Vision*, pp. 515–522, 1995.

[55] Irani, M., Rousso, B., and Peleg, S., "Computing Occluding and Transparent Motions," *Int. J.Comput. Vis.*, vol. 12, no. 1, pp. 5-16, February 1994.

[56] Itti, L., and Koch, C., "Computational Modeling of Visual Attention," in *Natire Reviews-Neuroscience*, vol.2, pp. 1-11, 2001.

[57] Joly, P., and Kim, H. K., "Efficient Automatic Analysis of Camera Work and Microsegmentation of Video using Spatiotemporal images," in *Signal Processing: Image Commun*, vol. 8, pp. 295-307, 1996.

[58] Kasturi, R., and Jain R., "Dynamic Vision," in *Computer Vision: Principles*, Kasturi R., Jain R., Editors, IEEE Computer Society Press, Washington, 1991.

[59] Koch, C., Itti, L. and Niebur, E., "A Model of Saliency Based Visual Attention for Rapid Scene Analysis," in *IEEE Transactions on PAMI*, vol.20, pp. 1254-1259, 1998.

[60] Koch, C., and Ullman, S., "Shifts in Selective Visual Attention: Towards the Underlying Neural Circuitry," in *Hum Neurobiol 4*, pp. 219-227, 1985.

[61] Koprinska, I. and Carrato, S., “Temporal Video Segmentation: A Survey,” in *Signal Processing: Image Communication* 16(5), pp. 447-500, 2001.

[62] Leake, D. B., and Sooriamurthi R., “Automatically Selecting Strategies for Multi-Case-Base Reasoning,” in *ECCBR 2002*, eds S. Craw & A. Preece, LNAI 2416, pp. 204-233, Springer Verlag, 2002.

[63] Lienhart, R., “Comparison of Automatic Shot Boundary Detection Algorithms,” in *IEEE Signal Processing Magazine*, vol. 17, pp. 12-36, November, 2000.

[64] Lienhart, R., “Reliable Dissolve Detection,” in *Proc. SPIE: Storage and Retrieval for Media Databases*, pp. 219-230, 2001.

[65] Lienhart, R., “Reliable Transition Detection in Videos: A Survey and Practitioner’s Guide,” in *Int’l J. of Image and Graphics*, August 2001.

[66] Lienhart, R., and Zaccarin, A., “Reliable Dissolve Detection,” in *Proc. 2001 Int’l Conf. on Image Processing*, vol. 3, pp. 406-409, October 2001.

[67] Lin, T., and Zhang H., “Automatic Video Scene Extraction by Shot Grouping,” in *ICPR’00*, 2000.

[68] Liu, X., and Chen, T., “Shot Boundary Detection using Temporal Statistics Modeling,” in *ICASSP 2002*, May 2002.

[69] Liu, F., “Modeling Spatial and Temporal Texture,” Ph.D. Dissertation, MIT, Cambridge, MA, 1997.

[70] Liu, F., and Picard, R. W., “Finding periodicity in space and time,” in *Proc. IEEE Int’l. Conf. on Computer Vision*, pp. 376–383, 1998.

[71] Lu, H., and Tan, T., "An Effective Post-Refinement Method for Shot Boundary Detection," in *IEEE Trans. on Circuits and Systems for Video Tech.*, vol. 15, pp. 1407-1421, November 2005.

[72] Meng, J., Juan, Y., and Chang, S. F., "Scene Change Detection in a MPEG Compressed Video Sequence," in *SPIE Symp. on Electronic Imaging: Science and Technology – Digital Video Compression: Algorithms and Techonologies*, SPIE vol. 2419, San Jose, CA, February 1995.

[73] Miadowicz, J., "Story Tracking in Video News Broadcasts," Ph.D. Dissertation, University of Kansas (KU), Lawrence, KS, June 2004.

[74] Miene, A., Hermes, T., Ioannidis, G. T., and Herzog O., "Automatic Shot Boundary detection using adaptive thresholds," in *Proceedings of the TRECVID 2003 Workshop*, pp. 275-278, Gaitherburg, Maryland, USA, 2003.

[75] Nagasaka, A. and Tanaka, Y., "Automatic Video Indexing and Full-Video Search for Object Appearances," in *Visual Database Systems II*, Knuth, E., Wegner, L. Editors, pp. 113-127, 1993.

[76] Nam, J., Cetin, E., and Tewfik, A "Speaker Identification and Video Analysis for Hierarchical Video Shot Classification," in *Proc. Int. Conf. Image Processing*, Santa Barbara, CA, October, 1997.

[77] Ngo, C.-W., Pong, T.-C. and Zhang, H.-J., "Motion Analysis and Segmentation through Spatio-Temporal Slices Processing," in *IEEE Trans. Image Processing*, vol. 12, no. 3, pp. 341-355, March 2003.

[78] Ngo, Chong-Wah, "A Robust Dissolve Detector by Support Vector Machine" in *ACM Conf. on Multimedia* (*MM*), 2003.

[79] Ngo, C. W., Pong, T. C., and Zhang, H. J., "Motion-based video representation for Scene Change Detection," in *Int'l. J. Comput. Vis.*, vol. 50, no. 2, November 2002.

[80] Ngo, C. W., Pong, T. C., and Zhang, H. J., "On Clustering and Retrieval of Video Shots," in *Proc. ACM Conf. on Multimedia*, 2001.

[81] Ngo, C. W., Pong, T. C., and Chin, R. T., "A Robust Wipe Detection Algorithm," in *Asian Conf. Computer Vision*, vol. 1, pp. 246-251, 2000.

[82] Norton, D., and Stark, L., "Scanpaths in the saccadic eye movements during pattern perception," in *Visual Research*, no. 11, pp. 929-942, 1990.

[83] Oja, E., "Subspace Methods or Pattern Recognition," Letchworth, Hertfordshire, England. New York: Wiley, 1983.

[84] O'Toole, C., Smeaton, A., Murphy, N., and Marlow, S., "Evaluation of Automatic Shot Boundary Detection on a Large Video Test Suite," presented at *the 2nd U.K. Conf. Image Retrieval: The Challenge of Image Retrieval*, Newcastle, U.K., 1999.

[85] Patel, N. V., and Sethi, I. K., "Video Shot Detection and Characterization for Video Databases," in *Pattern Recognition*, vol. 30, no. 4, pp. 607-625, Apr. 1997.

[86] Pearson, C.E., Ed., "Handbook of Applied Mathematics: Selected Results and Methods," 2nd ed. New York: Van Nostrand Reinhold Co., 1983.

[87] Perry, C., "Trouble-Free Transitions," in *Video Capture & Editing*, vol. 8, pp. 80-83, August 2002.

[88] Porter, S. V., Mirmehdi, and M., Thomas, B. T., "Video Cut Detection using Frequency Domain Correlation," in *IAPR Int'l Conf. on Pattern Recognition*, vol 3, pp. 413-416, Barcelona, Spain, September 2000.

[89] Rabiner, L, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," in *Proc IEEE*, vol. 77, no. 2, pp. 257-285, February 1989.

[90] Sabharwal, C. L., and Bhatia, S. K. "Image Databases and Nearest Perfect Hash Table," in *Pattern Recognition*, 1997.

[91] Saraceno, C. and Leonardi, R., "Audio as a Support to Scene Change Detection and Characterization of Video Sequences," in *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing*, pp. 2597-2600, Munich, Germany, April 1998.

[92] Sawhney, H. S., and Ayer, S., "Compact Representations of Videos through Dominant and Multiple Motion Estimation," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 18, no. 8, pp. 814-830, 1996.

[93] Sethi, K., and Patel, N., "A Statistical Approach to Scene Change Detection," in *Proceedings of SPIE*, vol. 2420, pp. 329-337, 1995.

[94] Shahraray, B., "Scene Change Detection and Content-Based Sampling of Video Sequences," in *Digital Video Compression: Algorithms and Technologies*, Arturo Rodriguez, Robert Safranek, Edward Delp, Editors, SPIE vol. 2419, pp. 2-13, February, 1995.

[95] Shen, K., and Delp, E. J., "A Fast Algorithm for Video Parsing Using MPEG Compressed Sequences," in *Proc. Of IEEE International Conference Image Processing*, pp.252-255, Washington DC., 1995.

[96] Simoncelli, E. P. "Distributed Representation and Analysis of Visual Motion," Ph.D. Dissertation, MIT, 1993.

[97] Smeaton, A. F., Kraaij, W., and Over, P. (2003), "TRECVID-2003 – An Introduction," in *Proceedings of the TRECVID 2003 Workshop*, pp. 1-10, Gaitherburg, Maryland, USA, 2003.

[98] Smith, J. R., Srinivasan, S., Amir, A. Basu, S., Iyengar, G., Lin, C. Y., Naphade, M. R., Ponceleon, D. B., and Tseng, B. L., "Integrating features, models, and semantics for TREC video retrieval," in *NIST Special Publication 500-250: Proceedings of the Tenth Text Retrieval Conference (TREC 2001)*, pp. 240-249, Gaithersburg, Maryland, USA, 2001.

[99] Swanberg, D., Shu, C. F., and Jain, R., "Knowledge Guided Parsing and Retrieval in Video Databases," in *Storage and Retrieval for Image and Video Databases*, Wayne Niblack, Editor, SPIE vol. 1908, pp. 173-187, February 1993.

[100] Tahaghoghi, S. M. M., Williams, H. E., Thom, J. A., and Volkmer, T., "Video Cut Detection using Frame Windows," in *V. Estivill-Castro*, editor, *Proc. of the Twenty-Eight Australian Computer Science Conference* (*ACSC 2005*), vol. 38, New Castle, NSW, Australia, 31 January – 3 February 2005.

[101] Taskiran, C., Delp, E. J., "Video Scene Change Detection Using the Generalized Sequence Trace," in *Conf. in Research and Practice in Information Technology Series*, vol. 102, New Castle, Australia, 2005.

[102] Tonomura, Y., Akutsu, A., Otsuji, K. and Sadakata, T. "Videomap and video spaceicon: Tools for anatomizing video content," in *Proc. INTERCHI*, pp. 131–136., 1993.

[103] Truong, B. T., Dorai, C., Venkatesh, S., "New Enhancements to Cut, Fade, and Dissolve Detection Processes in Video Segmentation," in *ACM Int'l Conf. on Multimedia*, pp. 219-227, Los Angeles, CA, October 2000.

[104] Tsatsoulis, C. and Williams, A., "Case-Based Reasoning," in *Knowledge-Based Systems - Techniques and Applications*, vol. 3 (Computer Techniques), Ed. C.T. Leondes, Academic Press, pp. 807-837, 2000.

[105] Ueda, H., Miyatake, T., and Yoshizawa, S., "IMPACT: An Interactive Natural-motion-picture Dedicated Multimedia Authoring System," in *Proc. of CHI*, pp.343-350, New Orleans, Louisiana, Apr.-Mar. 1991.

[106] Wang, J., and Adelson, E., "Layer Representation for Motion Analysis," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 361-366, 1993.

[107] Wilcox,- L., Ember, D., Chen, F., "Audio Indexing Using Speaker Identification," in *Proc. SPIE Conference on Automatic Systems for the Inspection and Identification of Humans*, pp. 149-157, San Diego, CA, July 1994.

[108] Wilcox, L, and Boreczky, J., "Annotation and Segmentation in Multimedia Indexing and Retrieval", in *HICSS*, January 1998.

[109] Wong, R. J., and Wintz P. A., "Information Extraction, SNR Improvement, and Data Compression in Multi-spectral Imagery," in *IEEE Transaction on Communication and Technology,* vol. CoOM-21, pp.1123-1131, 1973.

[110] Yeo, B.-L., and Liu, B., "Rapid Scene Analysis on Compressed Video," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5 no. 6, December 1995.

[111] Yoon, K., DeMenthon, D., and Doermann, D., “Event Detection from MPEG Video in the Compressed Domain,” in *Int’l Conf. on Pattern Recognition,* Barcelona, Spain, 2000.

[112] Yusoff, Y., Christmas, W., Kittler, J., Centre for Vision, Speech and Signal Processing, “Video Shot Cut Detection Using Adaptive Thresholding,” in *Proc. of the British Machine Vision Conf.*, Bristol, United Kingdom, 2000.

[113] Zabih, R., Miller, J., and Mai, K., “A Feature Based Algorithm for Detecting and Classifying Scene Breaks,” Proc. *ACM Multimedia 95*, pp. 189-200, San Francisco, CA, November 1995.

[114] Zabih, R., Miller, J., and Mai, K., “A Feature Based Algorithm for Detecting and Classifying Production Effects,” in *Proc. ACM Multimedia Systems*, 7(2): 119-128, 1999.

[115] Zhao, R., and Grosky, W. I., "Video shot detection using color anglogram and latent semantic indexing: From contents to semantics,” in *Handbook of Video Databases: Design and Applications*, chapter 15, CRC Press, September 2003.

[116] Zhang, H. J., Kankanhalli, A., and Smoliar, S.W., “Automatic Partitioning of Full-motion Video,” in *Multimedia Systems*, vol. 1, no. 1, pp. 10-28, 1993.

[117] Zhou, J., and Zhang, X., “Video Shot Boundary Detection Using Independent Component Analysis,” in *ICASSP’05*, Philadelphia, PA, 2005.

Appendix A

Equations Derivations

Appendix A

Equations Derivations

This section contains the derivation of the equations used in dissolve and fade detectors to generate adaptive examples. Consider equation bellow:

$$I_{t,i} = \alpha \cdot A_{t,i} + (1-\alpha) \cdot B_{t,i} \qquad \alpha \in [1...0]$$

where I is the generated ith moment for adaptive examples for frame number t. A and B are ith moment of frame number f of A and B partitions of the sliding window when detecting dissolves. In case of fades, A and B represent one of A or B partitions as well as the synthetic monochrome shot. α is the weight controlling the degree of influence for each of the two shots. $(1-\alpha)$ is used so as the influence of the first shot decreases the influence of the second shot will increase. In the derivation bellow the following subscripts are used:

- i represents the moment index (0 to 27).
- t represents the frame number (time)
- x and y represent the pixel location within the current image (frame)
- R indicates that the following equations are for primary color channel red.

The derivation for mean of the new intensities (of the adaptive example) is demonstrated bellow:

$$M_{t,i,R} = \frac{\sum_{x\&y} I_{t,i,x,y,R}}{N} \Rightarrow M_{t,i,R} = \frac{\sum_{x\&y} \left(\alpha \cdot A_{t,i,x,y,R} + (1-\alpha) \cdot B_{t,i,x,y,R}\right)}{N}$$

$$\Rightarrow M_{t,i,R} = \alpha \cdot \overline{A}_{t,i,R} + (1-\alpha) \cdot \overline{B}_{t,i,R}$$

The derivation for standard deviation of the new intensities (of the adaptive example) is demonstrated bellow:

$$\sigma_{t,i,R}^2 = \frac{\sum_{x\&y} \left(I_{t,i,x,y,R} - M_{t,i,R}\right)^2}{N} \Rightarrow \sigma_{t,i,R}^2 = \frac{\sum_{x\&y} \left(\alpha \cdot A_{t,i,x,y,R} - \alpha \cdot \overline{A}_{t,i,R} + (1-\alpha) \cdot B_{t,i,x,y,R} - (1-\alpha) \cdot \overline{B}_{t,i,R}\right)^2}{N}$$

$$\Rightarrow \sigma_{t,i,R}^2 = \frac{\sum_{x\&y} \left(\alpha \cdot (A_{t,i,x,y,R} - \overline{A}_{t,i,R}) + (1-\alpha) \cdot (B_{t,i,x,y,R} - \overline{B}_{t,i,R})\right)^2}{N}$$

$$\Rightarrow \sigma_{t,i,R}^2 = \frac{\sum_{x\&y} \alpha^2 \cdot (A_{t,i,x,y,R} - \overline{A}_{t,i,R})^2}{N} + \frac{\sum_{x\&y} (1-\alpha)^2 \cdot (B_{t,i,x,y,R} - \overline{B}_{t,i,R})^2}{N} + \frac{\alpha \cdot (1-\alpha) \cdot (A_{t,i,x,y,R} - \overline{A}_{t,i,R}) \cdot (B_{t,i,x,y,R} - \overline{B}_{t,i,R})}{N}$$

$$\Rightarrow \sigma_{t,i,R}^2 = \alpha^2 \cdot \sigma_{A_{t,i,R}}^2 + (1-\alpha)^2 \cdot \sigma_{B_{t,i,R}}^2 + \frac{\alpha \cdot (1-\alpha) \cdot (A_{t,i,x,y,R} - \overline{A}_{t,i,R}) \cdot (B_{t,i,x,y,R} - \overline{B}_{t,i,R})}{N}$$

$$\Rightarrow \sigma_{t,i,R}^2 \approx \alpha^2 \cdot \sigma_{A_{t,i,R}}^2 + (1-\alpha)^2 \cdot \sigma_{B_{t,i,R}}^2$$

$$\Rightarrow \sigma_{t,i,R} \approx \sqrt{\alpha^2 \cdot \sigma_{A_{t,i,R}}^2 + (1-\alpha)^2 \cdot \sigma_{B_{t,i,R}}^2}$$

Similar equations are used for blue and green primary color channels. As can be seen in the derivation above the last term in the standard deviation equation is ignored. This is because in a long run, the negative and positive values of that term add up to zero.

Appendix B

Glossary

Appendix B

Glossary

-A-

Adaptive Examples are examples which are produced using the localized data; localized meaning the data in the close proximity of data in question.

Adaptive Threshold refers to those thresholds which are defined through use of statistics and features extracted from the input stream prior to execution of the main algorithm.

Abrupt Transition are defined as a sudden change in the numerical representation of video stream; They are due to the discrete linkage of two adjacent shots.

-B-

-C-

Classification refers to the process during which the potential candidates for a specific task, are labeled (assigned to different groups).

Color Moments

<*See:* Raw Color Moments>

Complete Fade refers to a fade consisted of a fade in followed by a fade out with variable number of monochrome frames in the middle.

Cut

<*See*: *abrupt transition*>

-D-

Detection is one of the major steps in most of the video boundary shot detection and it refers to the process of input data (*video representation* as well as *measure of difference* data) analysis leading to regions of interest (potential candidates for video shot boundaries) discovery.

Dissolve is a common variation of gradual transitions during which the transition from one shot to the next takes place by decreasing the effect of the first while increasing the effect of the second shot frame by frame.

-E-

Edit Frames are the set of images generated during the editing process. Most common edit frames are transitions.

<*Also See*: Transition>

-F-

Fade is a common variation of dissolve in which one of the two shots is only consisted of monochrome frames.

<*also see*: Fade In,
Fade Out and
Complete Fade>

Fade In is a fade transition from a usual shot to a monochrome shot.

Fade Out is a fade transition from a monochrome shot to a usual shot.

False Negative (FN) is the label used for missed transitions, the actual transitions which were not detected.

False Positive (FP) is the label used for the false alarms, the detected transitions that are due to false positive causing effects.

Feature is a measurement or set of measurements made from an image sequence. A feature can be a function of individual images in the sequence or some subset of images from sequence.

-G-

Generality means the same algorithm can be used for different purposes. In case of temporal segmentation, an algorithm that can detect various types of transitions such cuts, fades and dissolves.

Global Threshold refers to those thresholds which are defined for the input stream as a whole or for a relatively large portion of input stream.

Gradual Transition is a frames sequence constructed by usage of the frames from first shot, second shot and/or supplementary frames or effects which provide a steady and smooth conversion of one shot to its adjacent shot.

Graphical Transition is a type of transition which is compiled through the use of effects, computer graphics as well as other transitions in combination with the frames from the surrounding shots leading to a frame by frame transformation solution for converting one shot to the next.

-H-

-I-

Image is a digitized representation of a picture. An image has a number of discrete pixel locations and is represented by $I(x, y) = (r, g, b)$ where $x \in [1...M]$, $y \in [1...N]$. (x, y) represents the location of a pixel within and image, M x N represents the size of the image and (r, g, b) represents the brightness values in the red, green and blue bands respectively.

Image Sequence is a set of images that are indexed by time. An image sequence is represented by E(x, y, t) = (r, g, b) where *t* represent the temporal index.

-J-

-K-

-L-

Local Threshold refers to those thresholds which are defined for a relatively smaller segment of the input stream.

<*also See*: Global Threshold

Adaptive Threshold

Random Threshold>

-M-

Measure of Difference refers to the degree of dissimilarity between two test subject (in this case two frames or two video sequences).

Monochrome Frame is a frame consisted of pixels of the same intensities.

Monochrome Shot is a monochrome frames sequence with the same pixel intensities through out the shot.

-N-

Normal Group refers to the set of frames which belong to shots rather than transitions, effects or motions under question.

-O-

-P-

Precision is the proportion of correct shot boundaries identified by the system to the total number of shot boundaries identified by the system.

-Q-

-R-

Random Threshold refers to those thresholds defined without any prior knowledge of data set.

Raw Color Moments refer to the statistical (numerical) *features* that are directly extracted from image sequence.

Recall is the proportion of shot boundaries correctly identified by the system to the total number of shot boundaries presented.

-S-

Shot is an image sequence which represents continuous action and appears to be from a single operation of the camera.

Sliding Window is a technique used to analyze a sequence of data. It is a window which traverses the sequence in a specific order.

<*also See*: Window>

Spatial Transition is a gradual pixel by pixel space-wise localized change in pixel intensities of one shot during which the pixels in the ending frames of that shot give their place to the corresponding pixels in the corresponding frames of the upcoming shot which will eventually lead to a frame within the second shot. Examples are wipes and some of the gradual transitions.

<*also See*: Temporal Transition>

-T-

Temporal Transition is a gradual frame by frame space-wise global change in pixel intensities of one shot which will eventually lead to a frame in the upcoming shot. Examples are cuts, fades and dissolves.

<*also See*: Spatial Transition>

Thresholding refers to the process of applying one or more threshold on a given data stream.

Transition is a sequence of frames (in case of gradual transition) or a sudden change (in case of abrupt transitions) which transforms or changes one frame to the next.

True Negative (TN) are the transitions which were not detected as one and actually are not a transition. Basically TN can be calculated by subtracting number of all transitions from sum of TP, FP and FN numbers.

True Positive (TP) is the label used for the detected transitions which were also marked as such in the truth data.

-U-

Uncertainty Group while classifying different groups or transitions, some of them cannot be clearly identified as a specific group due to the lack of proper video representation. Henceforth they are given a certain label and addressed later on within the algorithm.

Utility is the weighted sum of recall and precision and it is used to evaluate programs performance.

-V-

Video is an image sequence which is generated by computing several shots by a process called editing, also referred to as the final cut.

Video Representation is the process of extracting different features of the images (frames) within a video.

-W-

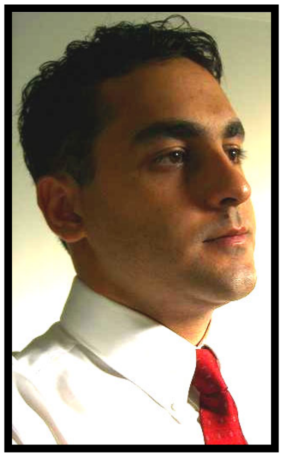
Window is a subset of data sequence in question (in this case it is a subset of frames from the video sequence).

Wipe is a type of gradual transition during which the first shot will spatially (pixel by pixel) transform into the second shot. It is a specific type of spatial transition since there should exist a specific order in which pixels of a frame in the preceding shot give their place to the pixels in a frame of the upcoming shot. This order yields a pattern in the video sequence which is known as wipe.

-X-

-Y-

-Z-



Robert B. Yeganeh received his Master of Science on August 2006. During his college years, he gained diverse professional and leadership experiences through leadership positions in student organizations, school teaching and research positions, internships and co-ops. Kansas Secretary of State Office, the United States Geological Survey (USGS), Cerner and IBM Corporations are among the off-campus organizations listed in his vitae. He will be joining IBM Corporation as a permanent application developer consultant in August 2006.