VIDEO SHOT BOUNDARY DETECTION BY GRAPH THEORETIC APPROACHES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EMRAH AŞAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2008

Approval of the thesis:

VIDEO SHOT BOUNDARY DETECTION BY GRAPH THEORETIC APPROACHES

Submitted by **EMRAH AŞAN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Assoc. Prof. Dr. A. Aydın Alatan _____
Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Gözde Bozdağı Akar _____
Electrical and Electronics Engineering Dept., METU

Assoc. Prof. Dr. A. Aydın Alatan _____
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. Cüneyt F. Bazlamaçcı _____
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. İlkay Ulusoy _____
Electrical and Electronics Engineering Dept., METU

Ufuk Sakarya (M.S) _____
TÜBİTAK UZAY

Date: 03 September 2008

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name :

Signature :

ABSTRACT

VIDEO SHOT BOUNDARY DETECTION BY GRAPH THEORETIC APPROACHES

Aşan, Emrah

M.S, Department of Electrical and Electronics Engineering
Supervisor: Assoc. Prof. Dr. A. Aydın Alatan

September 2008, 109 pages

This thesis aims comparative analysis of the state of the art shot boundary detection algorithms. The major methods that have been used for shot boundary detection such as pixel intensity based, histogram-based, edge-based, and motion vectors based, are implemented and analyzed. A recent method which utilizes “graph partition model” together with the support vector machine classifier as a shot boundary detection algorithm is also implemented and analyzed.

Moreover, a novel graph theoretic concept, “dominant sets”, is also successfully applied to the shot boundary detection problem as a contribution to the solution domain.

Keywords: Shot Boundary Detection, Graph Partition Model, Dominant Sets.

ÖZ

ÇİZGE TEMELLİ YAKLAŞIMLAR İLE VİDEO ÇEKİM SINIRI SEZME

Aşan, Emrah

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. A. Aydın Alatan

Eylül 2008, 109 sayfa

Bu tezde gelişmiş çekim sınırı sezme metodlarının karşılaştırmalı bir analizi hedeflenmektedir. Başlıca çekim sınırı sezme yöntemlerinden piksel temelli, histogram temelli, kenar temelli ve hareket temelli yöntemler gerçekleştirilmiş ve analiz edilmiştir. Son dönemde geliştirilmiş olan ve çekim sınırı sezme problemine çizge bölütleme modelini destek vectör makinaları sınıflandırıcısı ile birlikte uygulayan yöntem de gerçekleştirimiş ve analiz edilmiştir.

Bunun yanında, yeni bir çizge temelli kavram olan “baskın kümeler” konusu çalışılmış ve bu yeni yöntem çekim sınırı sezme problemine başarıyla uygulanmıştır.

Anahtar Kelimeler: Çekim Sınırı Sezme, Çizge Bölütleme Modeli, Baskın Kümeler.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Assoc. Prof. Dr. A. Aydın Alatan for his valuable supervision and insightful comments throughout this thesis. I am grateful to him for his guidance, encouragement, and support which guided me in the development of this study.

I would like to express my sincere thanks to my wife Nuran Aşan, my parents, Nesime Aşan and Türker Aşan and my brother Fırat Aşan for their love, understanding and supports.

The technical assistance of Ufuk Sakarya is gratefully acknowledged.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1. Motivation

Recent developments in video compression technology, the widespread use of digital cameras, high capacity digital systems, coupled with the significant increase in computer performance and the growth of Internet and broadband communication, have increased the usage and availability of digital video. Applications such as multimedia information systems, distance learning, video-on-demand produce and use huge amount of video data. This situation created a need for tools that can effectively categorize, search and retrieve the relevant video material.

In general, management of such activities over large collections of video requires knowledge of the “content” of the video. In particular, digital video data can be processed with the objective of extracting the information about the content conveyed with this data. The algorithms developed for this purpose, referred as “video content analysis” algorithms and serve as the basis for developing tools that would enable us to understand the events and objects within the scene of a video, or generate summary of large video material or even to derive semantically meaningful information from the video [1].

The definition of “content” is highly application dependent but there are a number of commonalities in the applications of content analysis. Among others, *shot boundary detection* (SBD), also known as *temporal video segmentation* is one of the important aspects.

Parsing a video into its basic temporal units -shots- is considered as the initial step in the process of video content analysis. A shot is a series of video frames taken by a single camera, such as, for instance, by zooming into a person or an object, or simply by panning along a landscape [1]. The content is similar in

shot regions. The regions where the significant content change occurs are, therefore, called shot boundaries.

Since the SBD is a prerequisite step for most of the video applications involving the understanding, parsing, indexing, characterization, or categorization of video, temporal video segmentation has been an active topic of research in the area of content based video analysis.

1.2. Fundamental Problems of SBD

Shot boundary detection (SBD) is not a new problem anymore. It has been studied more than a decade and resulting algorithms have reached some maturity. However, challenges still exist and are summarized in the upcoming sections:

1.2.1. Detection of Gradual Transitions

During the video production process, first step is capturing the shots by using a single camera. Two consecutive shots are then attached together by a shot boundary that can either be abrupt or gradual.

Abrupt shot boundaries are created by simply attaching a shot to another. While there is no modification in the consequent shots in an abrupt shot boundary, gradual transitions result from editing effects applied to the shots during attachment operation. According to the editing effect gradual transitions can be further divided into different subtypes. The number of possible transitions due to editing effect is quite high but most of the transitions fall into the three main categories: dissolve, fades (fade in, fade out), and wipes [2]. Different types of transitions are demonstrated in the following figures:

Figure 1-1 Dissolve Effect

Figure 1-2 Wipe Effect

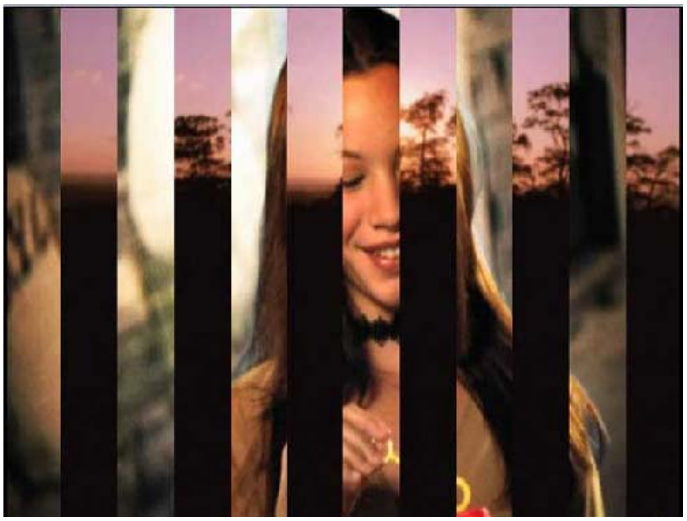

Figure 1-3 Random Bars Wipe Effect

Figure 1-4 Dissolve Effect

Detection of abrupt changes has been studied for a long time and not a difficult problem anymore [3]. On the other hand, gradual transitions pose a much more difficult problem. This situation is mainly due to the amount of available video editing effects. The problem gets harder when multiple effects are composed in the case of a lot of object or camera motion.

Another reason is that the gradual transitions spread over time. Each editing effect has a different temporal pattern than the others and the temporal duration changes from three frames to hundred frames.

Finally, the temporal patterns, as a result of editing effects to create a gradual transition, are very similar to the patterns due to camera/object motion. Therefore, gradual transitions remain to be one of the most challenging problems in SBD.

Figure 1-5 Fade In/Fade Out Effect

1.2.2. Flashlights

Color is the primary element of video content. Most of the video content representations employ color as a feature. Continuity signals based on color feature exhibit significant changes under abrupt illumination changes, such as flashlights. Such a significant change might be identified as a content change (i.e. a shot boundary) by most of the shot boundary detection tools. Several algorithms propose using illumination invariant features, but these algorithms always face with a trade off between using an illumination invariant feature and loosing the most significant feature in characterizing the variation of the visual content [4]. Therefore, flashlight detection is one of the major challenges in SBD algorithms.

1.2.3. Object/Camera Motion

Visual content of the video changes significantly with the extreme object/camera motion and screenplay effects (e.g. one turns on the light in a dark room) very similar to the typical shot changes. Sometimes, slow motion cause content change similar to gradual transitions, whereas extremely fast camera/object movements cause content change similar to cuts. Therefore, it is difficult to differentiate shot changes from the object/camera motion [2].

1.2.4. Thresholding

Shot boundaries are identified based on the visual content change. Therefore, the most critical activity in the SBD process is the selection of the thresholds in any shot boundary detection step. The performance of the algorithm mainly remains in the thresholding phase. However, using a single threshold can not perform equally well for all video sequences. Using a dynamic global threshold by extracting the overall sequence characteristic can not solve this problem. Dynamic local thresholds are considered as a better alternative but thresholding still remains as a major problem in this area [5].

1.2.5. Complexity of the Detector

Shot boundary detection is considered as a preprocessing step in most of the video content analysis applications. There are high level algorithms which perform more complex content analysis. Shot boundary detection results are used by these high level analysis algorithms. Since the video content applications takes most of the available computational power and time, it is necessary to keep the computational complexity of the shot boundary detector low. Such a need challenges for algorithms which are sufficiently precise but also computationally inexpensive [2].

As the shot boundary detection problem evolved, in order to increase the performance of the detection, proposed algorithms started to use more than one feature for content representation. On the other hand, such a strategy brings a computational burden on the detector, since each feature requires a separate processing step.

1.3. Scope of the Thesis

This thesis aims comparative analysis of the state of the art SBD algorithms. The major methods that have been used for SBD such as pixel intensity based, histogram based, edge based, and motion vectors based, are analyzed and implemented. A recent method, which utilizes “graph partition model” together with the support vector machine classifier as a SBD algorithm, is also

implemented and analyzed. Finally, a novel graph theoretic concept, namely *dominant sets*, is also successfully applied to the SBD problem as a novel contribution to the solution domain.

1.4. Organization of the Thesis

This thesis is organized as follows:

In Chapter 2, fundamental SBD algorithms in the literature are overviewed.

In Chapter 3, state of the art SBD algorithms are analyzed and experimental results are presented.

In Chapter 4, a novel graph theoretic concept referred as “dominant sets” is introduced and application of this concept into the SBD problem is discussed by the help of some experimental results.

In Chapter 5, some concluding remarks and future considerations are stated.

CHAPTER 2

RELATED WORK

Since SBD is an early step for most of the video applications involving the understanding, indexing, characterization, or categorization of video, temporal video segmentation has been an active topic of research in the area of content based video analysis. Such research efforts have resulted in a variety of algorithms. In this chapter, we start with a general framework of the SBD problem. The framework is based primarily on three steps [4]: the representation of visual content, construction of the continuity signal and the classification of continuity values. In the second section, we review the work related to SBD. The methods discussed in this section are categorized according to their roles in the formal frame framework. Finally, TRECVID organization, which posses an important role in the development of the SBD algorithms, are also briefly introduced.

2.1. General SBD Framework

Although SBD research resulted in a great variety of algorithms, very few of them studied the formal definition of the SBD problem. A Bayesian formulation of the problem was proposed by Vasconcelos et al. [6]. Hanjalic develops a theoretical framework for SBD by unraveling the problem and identifying the critical factors that need to be considered for robust detection performance [2]. In [7] Albanese et al. presents a formal model of the video shot segmentation process based on mathematical models. Recently, Yuan et al. has conducted a formal study of the SBD problem, which includes the latest developments in the area [4].

The basic assumption in SBD is that the frames within the same shots have similar visual content and the visual content changes through the shot boundary. Therefore, common idea in the SBD methods is finding the

discontinuities of visual content. Based on this commonality, no matter what kind of detection technique is used, a SBD process consists of three major steps: the representation of visual content, construction of the continuity signal and the classification of continuity values [4].

2.1.1. Representation of the Visual Content

The image itself contains various information details about the visual content. Alternatively, it is possible to obtain more valuable information by processing the image content in order to extract visual features such as histograms, edges, motion vectors, etc. Therefore, this step can also be called as *feature extraction*.

Problem of visual content representation can be considered as a mapping from image space, Q to the feature space, F. Let $V_t \in F$ denote the feature of $I_t \in Q$, where I_t represents the t^{th} frame. The problem is now converted into finding the most proper feature for the SBD, and can be formalized as a mapping Φ:

$$\begin{aligned} \Phi : Q &\rightarrow F \\ I_t &\rightarrow V_t \end{aligned} \tag{1}$$

A good feature to be used for SBD should have two characteristics: *invariance* and *sensitivity* [2]. Firstly, the feature should be stable against the content changes within the shot (e.g. rotation or translation of the picture) and should change significantly when there is a shot boundary. This is denoted as the invariance requirement [2].

On the other hand, the feature should sense the visual content changes, which is referred as the sensitivity. If a feature is sensitive, it is expected that the feature is aware of the details of the visual content.

By satisfying these two requirements, a proper feature remains stable within a shot (invariance) and demonstrates significant changes at shot boundaries (sensitivity).

2.1.2. Construction of Continuity Signal

Most common approach to detecting shot boundaries is to search for large discontinuities in the visual content flow of a video. In order to achieve this aim, a continuity (similarity) signal needs to be calculated for the frame sequence to determine the temporal variations of the extracted features. The constructed signal provides us with an idea about how similar the images in the video sequence are. Obviously, the continuity signal constructed by such a way is expected to demonstrate high values within a shot, while drops significantly at the transition regions.

S denoting the space of continuity values and s_t being the content continuity between V_t and V_{t+1}, continuity signal calculation can be formalized as a mapping (Θ) from the cartesian product of feature space to the continuity value space:

$$\begin{aligned} \Theta : F^{2\times d} &\rightarrow S \\ A_t^d &\rightarrow s_t \end{aligned} \tag{2}$$

where $A_t^d = (V_{t-d+1},...,V_t,V_{t+1},...,V_{t+d})$ and d denotes the radius of the involved neighborhood when calculating the content continuity between V_t and V_{t+1} [4].

In the earlier work, d is usually set to 1, which corresponds to pair-wise comparison of the adjacent frames. Unfortunately, the continuity signal obtained using pair-wise similarities does not perform well in the presence of object/camera motion, abrupt illumination changes, etc. Therefore, a more robust way of constructing a continuity signal is considering a neighborhood instead of just comparing adjacent frames. Recent studies set $d > 1$, which utilizes the neighborhood for the continuity signal construction. Using such contextual information improves the performance against disturbances [4].

2.1.3. Classification of Continuity Values

Final step of the framework is the decision module. All we need to do is to find a mapping Ψ from the continuity signal values to the decision space W. Decision space includes the decisions of whether the signal values correspond to shot boundaries or not. If $w \in W$ denotes the decision (shot boundary or not, or type of the transition), classification problem can be formulized as:

$$\begin{aligned} \Psi : S^{2\times r+1} &\rightarrow W \\ B_t^r &\rightarrow w \end{aligned} \tag{3}$$

where $B_t^r = (s_{t-r}, ..., s_t, s_{t+1}, ..., s_{t+r})$ and r is the radius of the neighboring continuity values required by the classifier [4].

Simple thresholding with a single continuity value (i.e. $r = 0$) is the most popular classifier being used in the literature. Recent algorithms treat the shot boundary detection problem into a pattern classification problem and employ machine learning techniques by training the classifiers with the temporal pattern of the continuity values.

In [2], Hanjalic states that in order to reach the optimal SBD performance “prior information” should also be used in the decision module. Prior information does not contain any information that is obtained from the image content by a kind of measurement. Prior information is based on the knowledge of the structure of the video. For example, one can intuitively assume that the probability of observing a shot boundary immediately after the last detected boundary is negligible. The general SBD framework including the “prior information” is summarized in the following figure:

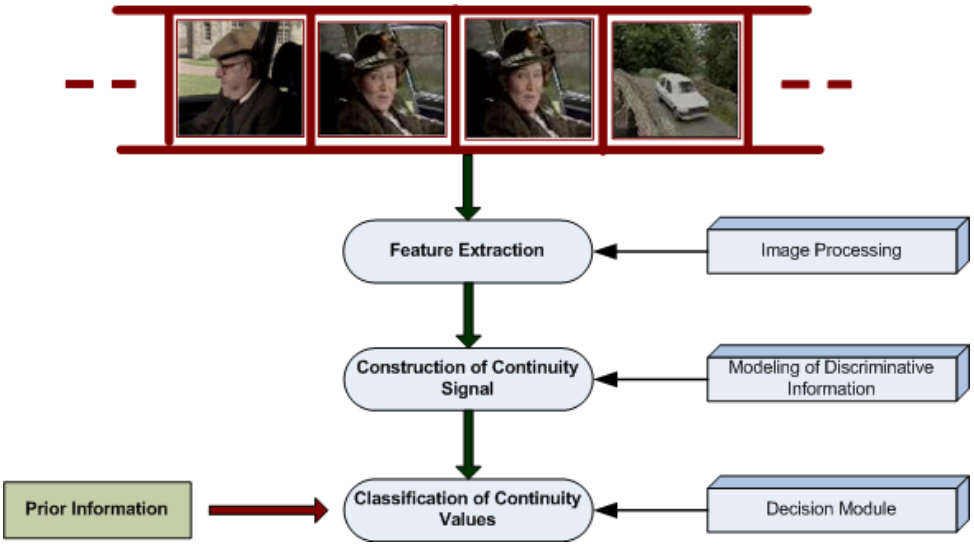

Figure 2-1 Framework for SBD

2.2. Taxonomy

SBD is a popular area in the video content analysis and has been studied for a long time. Research has resulted in a variety of algorithms. In this section, we briefly review the SBD work in the literature. Following the formal framework of the SBD problem, we categorize the literature according to their roles in the formal framework presented in the previous section.

2.2.1. Methods of Visual Content Representation

Feature selection is the crucial step in the SBD process. The algorithms in this step can be summarized under two classes: algorithms run on the compressed domain and the algorithms run on the uncompressed domain [10].

2.2.1.1. Uncompressed Domain

Algorithms in this group directly work on the full pixel domain. The data is assumed to be decompressed before SBD, if it is available as an encoded bit-stream.

2.2.1.1.1. Pixel Based Methods

Pixel based methods are the first and the most simple algorithms in the SBD literature. The basic idea behind pixel-based methods is that the intensity values of the pixels at the same locations of the sequential frames do not change significantly unless there is a shot boundary.

The initial pixel based algorithms investigates the sum of absolute pixel intensity differences and if the difference is above a certain value a shot boundary is assigned [8]. Even very small changes in the illumination or very small vibration in the camera can result in significant changes in total value of the pixel differences. Therefore, later algorithms count only the pixels that have changed significantly from one frame to another. If the total number of pixels that have changed is above a threshold, it is decided that there is a shot boundary between two frames [9] [10].

Even with this improvement pixel based algorithms are still very sensitive to object/camera motion and illumination changes. More robust techniques use block based motion compensation and then apply the above algorithms [11].

Instead of pixel-wise differences, some of the literature proposes using statistics of pixel intensities between two frames. An example to this is comparing the blocks using a metric called *likelihood ratio,* which is simply based on statistical properties mean and variance [2].

Another method proposed by Zhang et al. against disturbances is to smooth the images by a 3x3 filter before performing pixel wise comparison [12].

Pixel-based methods are simple algorithms and do not require high computational power. The problem with pixel-based methods are high sensitivity to camera/object motion and disturbances [2].

2.2.1.1.2. Histogram Based Methods

Another example of a feature that is from the full pixel domain is the histogram. The reasoning is that the frames within the same shot should have similar color histograms, while frames of different shots should have significantly different

color histograms. Earlier approaches compare gray level histograms [12] and recent methods utilize color histogram information [13].

Several histogram comparison metrics are proposed in the literature. The most common techniques are: histogram difference, histogram intersection, cosine measure, Kolmogorov-Smirrov test and Chi-Square test. Research shows that histogram intersection formula performs best in the SBD area [2].

Zhang et al. proposed a method called *twin-comparison* to detect gradual transitions using the color histogram difference [12]. This method requires two thresholds. Abrupt transitions are detected using the higher threshold. A lower threshold is used on the remaining frames. A frame that differs from the previous frame by an amount above this threshold is declared as a potential start of a gradual transition. This frame is then compared to the subsequent frames to get the accumulated difference. During a gradual transition, this accumulated value will gradually increase. The end frame of a gradual transition is detected when the difference between consecutive frames drops below the lower threshold and the accumulated value has increased to a value that exceeds the higher threshold. If the difference between consecutive frames drops below the lower threshold before the accumulated difference exceeds the higher one, then the starting point is dropped and the search process is applied for other gradual transition candidates. Otherwise, a gradual transition is assigned [12].

As the histograms do not change with the spatial changes within a frame, histogram differences are more robust against the object motion with a constant background. However, histogram differences are also sensitive to camera motion, such as panning, tilting or zooming [2].

One can note that two images, which have completely different visual content, might still have similar histograms. However, research has shown that the probability of such events is low enough [10] [1] [12].

Similar to the pixel based methods, block based techniques can be utilized in order to improve the performance of the histogram based SBD algorithms [14].

Histogram-based algorithms are less sensitive to object motion than the pixel-based algorithms. Histogram-based algorithms are robust against global motion. Histogram calculations require more computational power compared to pixel-wise calculations [12].

2.2.1.1.3. Edge-Based Methods

Another feature that is proved to be useful in shot boundary detection is edges. Three edge-based features are mostly referred in the literature: Edge Change Ratio (ECR), Edge Contrast (EC) and Edge Energy.

Zabih et al. [15] propose an edge-based technique based on the idea that during a shot transition new intensity edges are observed far from the locations of the old intensity edges. Similarly, old edges disappear far from the location of new edges. Moreover, the patterns in the appearance of new edges and disappearance of old edges are different for different types of transitions [15].

ECR algorithm employs motion compensation techniques prior to edge comparison. Therefore, this feature is robust against motion. On the other hand literature [16] shows that ECR algorithm does not outperform histogram based algorithms in abrupt transition detection. The advantage of the ECR is that it can be used for detecting different types of transitions (i.e. cut, dissolve, fade, wipe).

Lienhart [16] proposed the *Edge Contrast* method as an alternative for detecting dissolves. Since during a dissolve, visual content is a composition of two shots, the frames within a dissolve loose their contrast and sharpness. EC method captures and amplifies the relation between stronger and weaker edges.

Song and Ra, and Petersohn uses *Edge Energy* in order to find gradual transitions [17] [18]. Algorithm is based on the similar observation that Lienhart proposed. Image contrast and sharpness decrease during a gradual transition. Therefore, it is expected that the edges gets weaker up to some point and than, as the second image starts to appear edges get stronger. As a consequence,

Edge Energy, which reflects the total number of strong edges, is expected to start decreasing during a GT and local minimum appears at the center of GT [17]. Petersohn finds the GTs by locating the U-curves in the Edge Energy diagram.

Edge-based methods can be used both for abrupt and gradual transition detection. Edge-based methods require significant computational power. Methods adopting edges as a feature are relatively more robust against motion but in general does not outperform histogram-based or pixel-based algorithms [2].

2.2.1.1.4. Motion-Based Methods

Motion-based algorithms rely on the observation that while motion within a shot is smooth, motion between the frames that are surrounding a shot boundary tends to be abrupt. This assumption makes sense, because the motion of the objects or camera is generally smooth and continuous within the shot, which results in a continuous motion field. In contrast, abrupt changes are expected for the motion field at the shot boundary.

Motion-based algorithms in the uncompressed domain are computationally very expensive. Therefore, there are very few SBD algorithms based on motion vectors.

Shahraray [11] uses block matching and motion estimation to detect shot boundaries. Recently, Kawai [19] proposes a very effective SBD algorithm based on block matching motion estimation which produces very good results in TRECVID 2007. For each block a best matching block is found in the previous frame. Blocks are compared based on histogram difference. For the best matching block, if the histogram difference is above a certain value, it is decided that this difference is due to a shot change, not due to motion. If the total number of blocks that are marked as such is above a certain percentage of the total block number, it is evaluated that there is a shot boundary [19].

Motion-based algorithms are computationally quite expensive algorithms but perform well in abrupt transition detection. In general, motion-based algorithms can not differentiate illumination changes and motion [20].

2.2.1.2. Compressed Domain

Noting that most of the visual content is stored in a compressed form, processing and analysis of such an encoded bit-stream is quite advantageous. The features explained for the uncompressed domain are also available for the compressed domain. In this section, we will focus on the literature which only uses the compressed domain features of the video.

2.2.1.2.1. DC Images

In compressed domain, the zero frequency term of the DCT coefficient series is known as DC term. DC term is a scaled version of the block's average value. The set of all DC terms in an *I-frame* (i.e. a frame encoded with no usage of prior frames) forms a DC image. Since the DC term is the average of the luminance/chrominance of all pixels within the 8x8 block, the DC image can also be accepted as a spatially reduced version of the original image [21]. The DC image retains most of the information about the original image, and therefore, most of the features in the original image can be approximated by its corresponding DC image. In case of abrupt shot detection, two frames belonging to the same shot should have similar DC images, whereas frames from different shots should have significantly different DC images.

Yeo and Liu uses the DC images directly from the compressed video and then employ pixel and histogram differences of DC images as video content representation [20]. Song and Ra first obtain the DC images from the MPEG compressed videos and then extract the edge images for SBD [17].

Fernando et al. [22] use statistical features of the DC images (i.e. mean and standard deviation) for SBD.

DC images can be easily extracted from the compressed video and be utilized for SBD. Since the DC images are a kind of spatially reduced versions of the

original images, using DC images increases the speed of the algorithms. Moreover, DC images are more robust against disturbances and noise.

2.2.1.2.2. DCT Coefficients

In *I-frames*, the DCT coefficients in each 8x8 DCT block are related to the luminance/chrominance of 8x8 pixels in spatial domain. Therefore, DCT coefficients can be used to detect the difference between the luminance/chrominance signal of all pixels within the 8x8 block, and therefore, they can be used to detect hard cuts [21]. The main idea behind the algorithms based on the DCT coefficients is that the pixel blocks similar to each other should have similar DCT coefficients. Similarly, if the blocks are significantly different, then they should have significantly different DCT coefficients [21].

Arman et al. [23] uses the inner products of the DCT coefficients for cut detection. After selecting a subset of the blocks, a subset of coefficients for each selected block is chosen. They set up a vector consisting of these coefficients and this vector represents the frame. The normalized inner product of these representing vectors is used to find the differences between two frames.

Using DCT coefficients, which are readily available in the compressed videos, increases the speed of the SBD algorithms.

2.2.1.2.3. Bit-Rate Information

Bit-rate information is obtained from the video stream by using the size of the block, macro-block or slice of each frame. Frames with similar content should have similar bit rates.

For the *I-blocks*, if the frame content changes, bit rate changes. This is due to the fact that *I-frames* are independent of the other frames and the bit rate required for the *I-frame* is strictly dependent on the content of this frame. Deardorff et al. uses this observation to detect cuts for the videos that contain only *I-frames* [24].

Another algorithm, which can be applied to *I-frames* only, sets up a bit rate vector using the available bit rate information from the macro blocks. Using these vectors the algorithm compares two frames based on a selected metric, and decides whether there is a shot boundary or not [21].

2.2.1.2.4. Macro Block Information

In most of the MPEG video compression standards *P-* and *B-frames* are encoded according to the predicted motion. Consequently, each macro-block in the inter-frames (i.e. *P-* and *B-frames*) has a motion vector type. In an MPEG coded video sequence, P frames have references to previous I/P frames, and B frames have references to both previous and following I/P frames. Furthermore, the level of referencing depends on the similarity between the referencing frame and referenced frame. Therefore, a shot change would cause an abrupt change in the referencing pattern of B/P frames [21].

The main idea for P and B frames' motion vector behavior is; if a frame is inside of a shot, then the macro blocks should be predicted well from previous or next frames. However, when the frames are on the shot boundary, the frames cannot be predicted from the related macro blocks, and a high prediction error occurs. This causes most of the macro blocks of the P frames to be intra coded instead of motion compensated [25].

The problem with the algorithms that are using motion-compensation information is that during the gradual transition the motion-compensation information is not reliable [20].

2.2.2. Methods of Constructing Continuity Signal

In this section, instead of focusing on the algorithms constructing the continuity signal, we concentrate on the contextual vs. pair-wise comparison methods. It should be noted that continuity signal is defined as the set of measurements from a sequence frames that denote the similarity between a frame and its neighbors.

2.2.2.1. Pair-wise Comparison Scheme

In the previous research, neighborhood is set to 1, which corresponds to pair-wise comparison of the adjacent frames. The continuity signal is simply consists of pair-wise feature differences of the successive frames [9] [8] [15] [19] [17] [18]. Most of the algorithms search for the large discontinuity values, especially the peaks in the 1D continuity signal.

2.2.2.2. Contextual Information Scheme

Unfortunately, the continuity signal obtained by using pair-wise similarities does not perform well in the presence of object/camera motion, abrupt illumination changes, etc. Therefore, a more robust way of constructing a continuity signal is considering a neighborhood, instead of just comparing adjacent frames. Recent studies utilize the neighborhood for the continuity signal construction. Utilizing such contextual information improves the performance against disturbance [1] [4].

One of the mostly used techniques is to use the features of all of the frames from a temporal window [1]. Easterbrook applies this method as an adaptive thresholding scheme [26].

Recently graph theoretic approaches are getting popular in the SBD area. In the graph based segmentation algorithms, all the pair-wise similarities are calculated within the neighborhood and a score signal is calculated based on these contextual information [14].

2.2.3. Classification Methods

At the decision level we can group the classifiers under two groups, as classifiers with simple thresholding and statistical pattern recognition techniques; these methods are examined in the upcoming two sections.

2.2.3.1. Classifiers with Simple Thresholding

In the basic thresholding scheme, continuity signal at the shot boundary candidate position is compared with a constant threshold [2] [8] [9] [15]. Final decision depends on whether the output is larger than the threshold or not.

Such a constant thresholding method can be successful if the video content is stable.

Various adaptive thresholding methods are proposed in order to deal with the changing video content [20] [26]. However, the decision is still based on a comparison with a single threshold.

2.2.3.2. Statistical Machine Learning

Recent research define SBD problem as a pattern recognition problem and applies machine learning methods. Various discriminative approaches, including K-means [18] [17], and support vector machines (SVMs) [4] [29] [30] [31], have been employed to perform SBD.

2.3. TRECVID History

For the purpose of promoting progress in content-based retrieval from digital video via open, metrics-based evaluation the TREC Video Retrieval Evaluation (TRECVID) meetings are organized by the National Institute of Standards and Technology (NIST) since 2003.

The TRECVID evaluation meetings are on-going series of workshops focusing on a list of different information retrieval (IR) research areas in content based retrieval of video. It is co-sponsored by the NIST and the Intelligence Advanced Projects Activity (IARPA) of the United States Office of the Director of National Intelligence (ODNI). The goal of the workshop is to encourage research in information retrieval by providing a large test collection, uniform scoring procedures, and a forum for organizations interested in comparing their results.

One of the primary research areas in TRECVID organizations is the SBD problem. The SBD results in TRECVID 2007 are summarized in Figure 2-2 and Figure 2-3.

The results are obviously showing that the algorithms are quite acceptable levels for practical applications. Especially for abrupt shot boundary detection, most of the state-of-the-art algorithms that attend TRECVID perform very well.

TRECVID organization stated that “it is time to declare victory for SBD problem” and SBD problem will no longer be tested in the TRECVID organization [3].

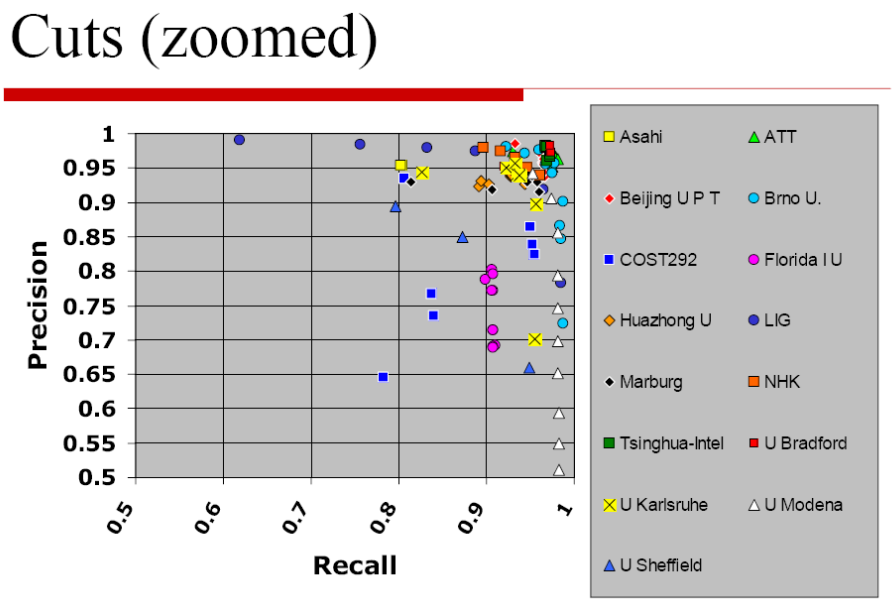

Figure 2-2 TRECVID 2007 CUT Detection Results [3]

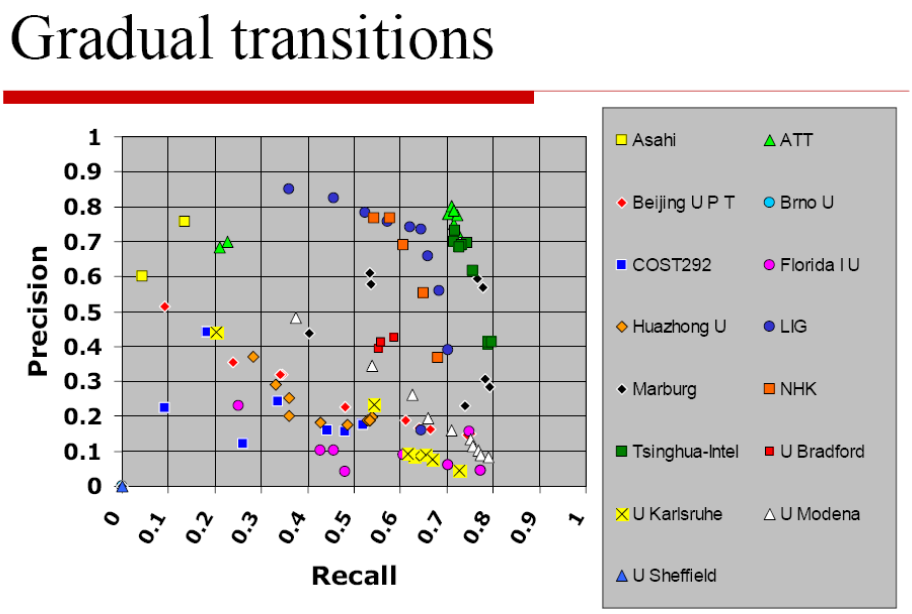

Figure 2-3 TRECVID 2007 Gradual Transition Detection Results [3]

CHAPTER 3

STATE OF THE ART SHOT BOUNDARY DETECTION

In this chapter, 6 state-of-the-art SBD algorithms from the literature are implemented and analyzed; these are the following algorithms

- Pixel based algorithm with adaptive thresholding [26],
- Histogram difference algorithm [2],
- Zabih's algorithm based on the edge change ratio [15] [16],
- Petersohn's shot boundary detection system, which incorporates pixel, edge and histogram difference statistics and employs k-means clustering [17] [18],
- Yuan's algorithm based on the graph partition model with support vector machine as the classifier [4], and,
- The algorithm by NHK, which utilizes the motion vectors based on dual cross search block matching method [19].

In the following section, the evaluation criteria and the test data used during the experiments are introduced. A brief summary of the algorithms and the experimental results are presented in the remainder of this chapter.

3.1. Test Data and Evaluation Criteria

TRECVID 2003 SBD test collections are adopted for the experiments. Totally there are 8 videos in the data set. The ground truth data for this test data is also distributed by TRECVID organization. The test collection includes 3581 transitions, of which 2488 (%70) are cuts, 749 (%20) are dissolves, and 345

(%10) are other type of transitions. Properties of the test data is summarized in Table 3-1.

In the experiments, the transition types are identified similar to TRECVID evaluation criteria: cuts, dissolves and others.

Table 3-1 TRECVID 2003 Test Data

#	Video ID	Duration	Total Frames	Total Transitions	Transition Types		
					CUT	DIS	Other
1	19980203_CNN	00:31:32	56717	451	280	124	47
2	19980222_CNN	00:29:19	52736	411	309	69	33
3	19980224_ABC	00:28:28	51204	428	296	91	41
4	19980412_ABC	00:28:50	51877	483	345	93	45
5	19980425_ABC	00:28:48	51814	476	295	141	40
6	19980515_CNN	00:27:56	50254	415	283	70	62
7	19980531_CNN	00:27:58	50275	468	359	71	38
8	19980619_ABC	00:28:29	51244	449	321	90	38
Total	**All**	**04:01:41**	**416121**	**3581**	**2488 (%70)**	**749 (%20)**	**345 (%10)**

The performances of the implemented algorithms are evaluated based on the *recall* and *precision* criteria. *Recall* is defined as the percentage of desired items that are retrieved. *Precision* is defined as the percentage of retrieved items that are desired items [32]:

$$Recall = \frac{Correct}{Correct + Missed} \tag{4}$$

$$Pecision = \frac{Correct}{Correct + FalsePositive} \tag{5}$$

In order compare the overall performance of the algorithms, F_1 measure, which combines recall and precision results with equal weight, is adopted [3]:

$$F_1(recall, precision) = \frac{2 \times recall \times precision}{recall + precision} \quad (6)$$

During the experiments, while all the algorithms are tested against abrupt changes, only edge change ratio algorithm, Petersohn's SBD system and the algorithm based on graph partition method are tested against gradual transitions.

In the Simulation Results sections, the results are presented in an accumulated manner so that we will be able to compare the results of the SBD algorithm under test with the results of the previously tested algorithms.

3.2. Pixel-wise Difference with Adaptive Thresholding

3.2.1. Algorithm

Pair-wise comparison evaluates the differences in intensity or color values of corresponding pixels in two successive frames. The simplest way is to calculate the absolute sum of pixel differences and compare it against a threshold [8]:

$$D(i) = \frac{1}{X * Y} \sum_{x=1}^{X} \sum_{y=1}^{Y} \left| f_{i+1}(x, y) - f_i(x, y) \right| \quad (7)$$

where X and Y are the frame width and height respectively, and $f_i(x,y)$ denotes the intensity value of the pixel at (x,y).

A simple improvement is to count the number of pixels that change in value more than some threshold and compare the total against a second threshold [12] [2].

$$DP(i, x, y) = \begin{cases} 1 & if \quad \left| P_i(x, y) - P_{i+1}(x, y) \right| > T_1 \\ 0 & otherwise \end{cases} \quad (8)$$

$$D(i) = \frac{\sum_{x=1}^{X} \sum_{y=1}^{Y} DP(i, x, y)}{X \cdot Y}$$

If the percentage of changed pixels *D(i)* is greater than a threshold T_2, a cut is declared. The algorithm is implemented basically based on the above formulation except for instead of simple thresholding, we adopted the adaptive thresholding proposed by the BBC research group [26]:

$$\begin{aligned} MAX(i) &= \max(D(i-2), D(i-1), D(i), D(i+1), D(i+2)) \\ MIN(i) &= \min(D(i-2), D(i-1), D(i), D(i+1), D(i+2)) \\ THR(i) &= MAX(i) + 0.8(MAX(i) - MIN(i)) \end{aligned} \tag{9}$$

The factor of 0.8 has been determined experimentally. In order to detect cuts, the difference signal is compared with the average of the threshold values of material before and after the current frame:

$$C(i) = D(i) - \frac{THR(i+3) + THR(i-3)}{2} \tag{10}$$

Finally, cuts are identified by searching for the peaks of the difference signal in (10).

3.2.2. Simulation Results

Figure 3-1 shows the continuity signal, *D(i)*, that is computed based on (8) for a test video. Spikes caused by cuts at frames 333, 388, 445 and 483 are clearly visible, but the cut at frame 189 is easily missed if the threshold cannot be selected carefully. Even if the threshold was selected so that we can detect the cut at frame 189, then with this threshold the frame at 398 will also be detected as a cut, but in fact it is not. The high level of activity in the images around frame number 189 produce a larger difference signal than does the cut itself.

Although the magnitude of the difference signal at frame 189 is not very high, it is clearly larger than that of the surrounding frames.

Figure 3-2 shows the resulting *C(i)* signal calculated for the *D(i)* values shown in Figure 3-1. In this figure, it is clear that there is no cut at frame 398. The signal *C(i)* is near or below zero, with well defined peaks at cuts. Comparing

this signal with a fixed threshold yields cut detection. A threshold value of 3.25 has been found to be appropriate for a range of picture material.

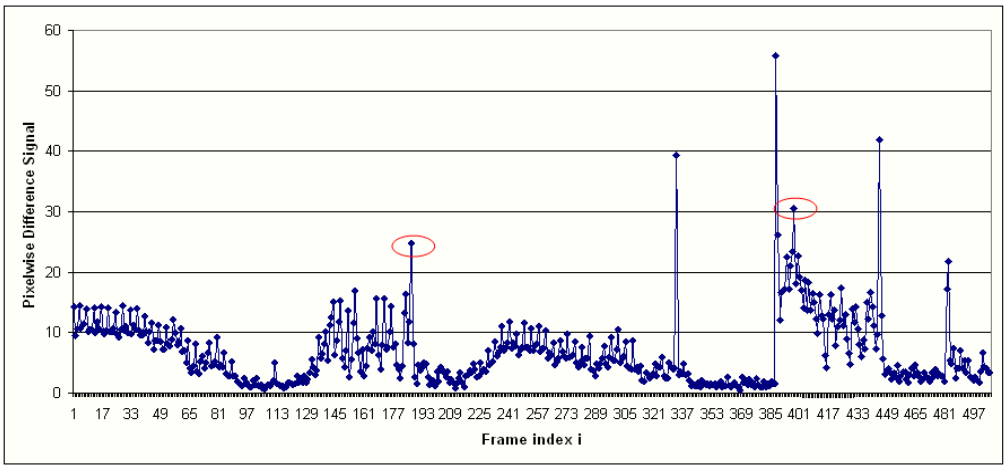

Figure 3-1 Thresholding Problem

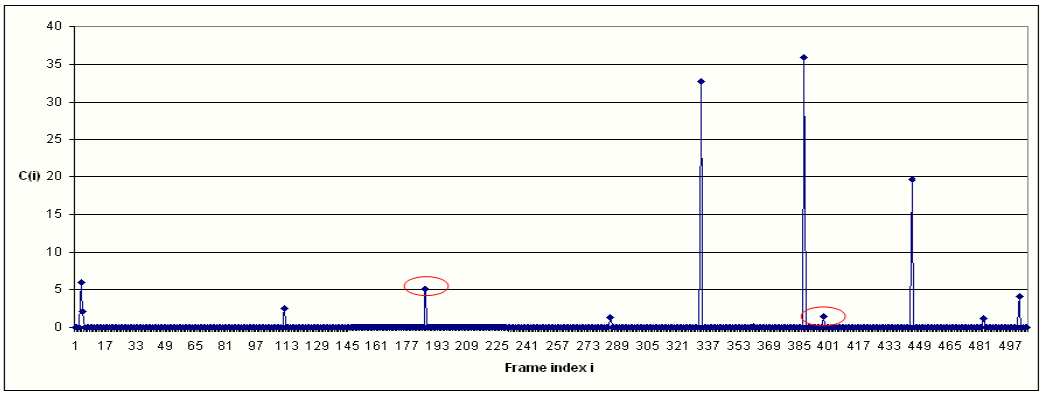

Figure 3-2 C(i) signal calculated for D(i)

Figure 3-3 shows a false positive detected by the pixel-wise difference algorithm. There is a significant brightness change caused by the man moving so that the sun behind him appears. Similar false alarms occur when a big object right in front of the camera rapidly moves out of the scene and the background appears.

Figure 3-3 False Positive due to Object Hidden behind Another

The algorithm is tested for detecting only abrupt changes. The simulation results for both single threshold and adaptive thresholding are summarized in the following table:

Table 3-2 Cut Detection Results for Pixel-wise Difference Algorithm

#	Pixel-wise			Pixel-wise with Adaptive Thresholding		
	Recall (R)	Precision (P)	F1	Recall (R)	Precision (P)	F1
1	0.62	0.8	0.7	0.74	0.82	0.83
2	0.68	0.84	0.75	0.79	0.9	0.86
3	0.74	0.89	0.81	0.89	0.88	0.92
4	0.84	0.9	0.87	0.9	0.92	0.92
5	0.69	0.79	0.74	0.85	0.81	0.88
6	0.67	0.88	0.76	0.78	0.84	0.85
7	0.73	0.81	0.77	0.9	0.89	0.92
8	0.77	0.9	0.83	0.81	0.87	0.95
Avg	**0.72**	**0.85**	**0.78**	**0.83**	**0.87**	**0.85**

Even with the adaptive thresholding, the algorithm produces false alarms, if the shot before/after the shot boundary includes high motion activity.

Figure 3-4 High Motion Activity Results in Missed Shot Boundaries

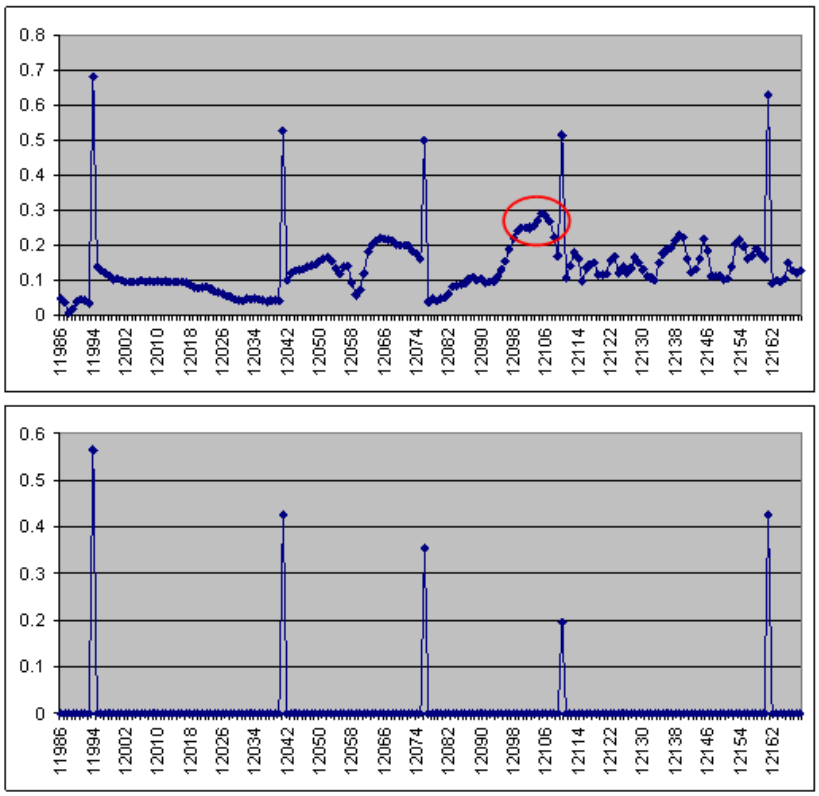

Figure 3-5 Top: Pixel Difference Signal,
Bottom: Signal Produced as a Result of Adaptive Thresholding

Figure 3-4 presents a sample frame sequence of this kind. Figure 3-5 shows the pixel difference signal and the corresponding *C(i)* signal after adaptive thresholding. High motion activity before the shot boundary produces relatively high values in the difference signal. Consequently, the threshold values found for the shot boundary frames are high, which results in a small peak value in the *C(i)* signal. Therefore, the algorithm misses the shot boundary.

3.2.3. Conclusion

The results indicate that the pixel-wise difference algorithm gives quite acceptable results with adaptive thresholding. Simulation results indicate that considering the difference between the difference signal values of adjacent frames is a worthwhile approach. In practice, we have observed that it is useful to reduce the effects of scenes containing a lot of movement by comparing the difference signal with a threshold derived from the maximum and minimum difference signals over a small aperture.

Even with the adaptive thresholding, the algorithm produces false alarms, if the shot before/after the shot boundary includes high motion activity. The reason can be explained as follows: The weakness of the pixel based features is the high sensitivity to the video content. It is difficult for this algorithm to understand whether the change in the continuity signal is due to shot boundary or due to disturbances/motion. In order to enhance the algorithm, we preferred adaptive thresholding. However, the high level of activity in the images around shot boundary produces a larger difference signal than expected. As a result adaptively obtained threshold is larger. A threshold that is larger than expected results in missed shot boundary.

The main disadvantage of this method is its inability to distinguish between a large change in a small area and a small change in a large area. We have observed that cuts are falsely detected when a small part of the frame undergoes a large, rapid change. For the same reason, the algorithm is not able to detect most of the flashlights.

3.3. Histogram Difference with Adaptive Thresholding

3.3.1. Algorithm

While the pixel-wise approach focuses on local intensity (color) comparison between individual pixels, this method is interested with the global percentage of colors that an image contains. The method works by calculating percentages from the bin totals and comparing them with those of the adjacent frame giving a difference value. A difference above the threshold value will be classed as a shot change [2].

If one tries to compute the overall number of possible colors, the calculations would be unnecessarily hard due to large number of bins (2^{24} bins). Due to the limited response of human visual system, we are not able to distinguish the whole levels of possible colors (Figure 3-6).

Figure 3-6 Difference bw. 24 bits/pixel and 12 bits/pixel Images [13]

A simple solution is considering only the most significant bits of each component RGB [13]. In 12-bit histograms, all possible colors are grouped into 2^{12} different color levels in RGB space (Figure 3-7), which corresponds to 4096 colors. Similarly in 6-bit histograms 64 color levels are used.

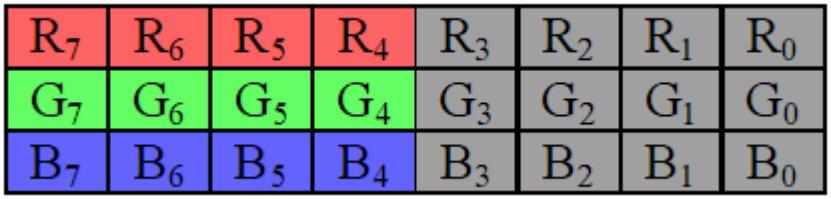

Figure 3-7 Only 4 MSB Color Bits are Used for 12 Bit Histograms [13]

In our implementation, we have adopted the 12-bit histograms. The difference between the histograms is calculated according to the following formula:

$$D_{RGB}(i,i+1)=\sum_{j=1}^{M}\left|H_i(j)-H_{i+1}(j)\right| \qquad (11)$$

where $H_i(j)$ is the histogram value for the color level j in the frame i, j is the color level value and M is the total number of color levels. $H_i(j)$ is the number of pixels from frame i with the color level j. A cut is declared if the absolute sum of histogram differences between two successive frames is greater than a threshold T.

3.3.2. Simulation Results

Typical histogram difference patterns for CUTs are shown in Figure 3-8.

Despite the fact that during a gradual transition the frame to frame differences are usually higher than those within a shot, they are much smaller than the differences in the case of a cut and can not be detected with the same threshold. On the other hand, the increase in the frame to frame differences due to object and camera motions might be larger than the gradual transitions. In Figure 3-9, there are three areas in the difference signal that are not cuts, but they also have higher difference values than the rest. The first (from frame 70 to frame 95) and third (from frame 220 to frame 237) intervals are due to dissolves. On the other hand, the second interval is due to large motion. Obviously, it is not possible to differentiate the transitions from the

disturbances. Therefore, this method is not tested for finding gradual transitions.

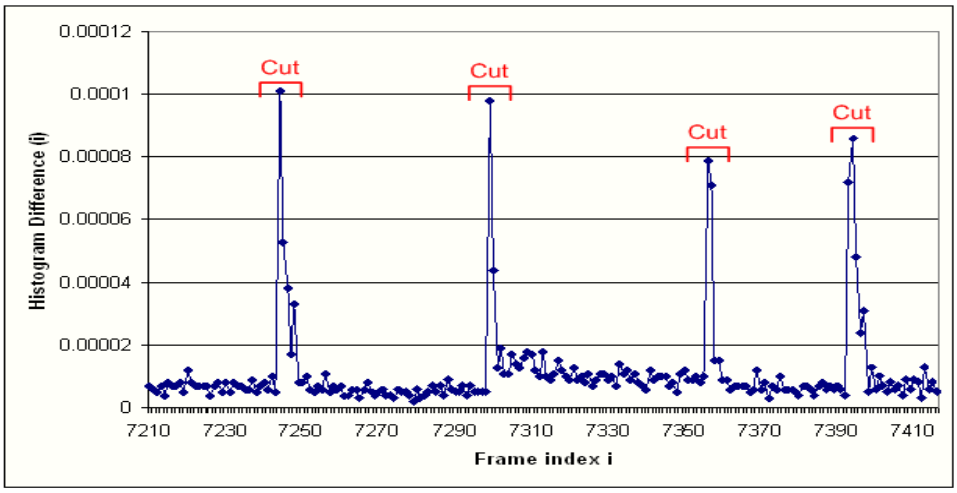

Figure 3-8 Typical Histogram Difference Patterns for CUTs

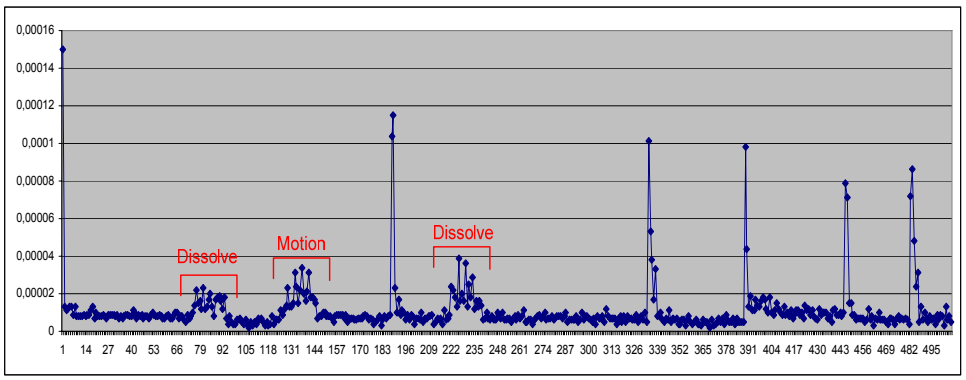

Figure 3-9 Gradual Transitions vs. Camera/Object Motion

Figure 3-10 shows a false detection due to the sliding commercial text. The background for the text changes the color histogram values significantly (by contributing with large red values), which is considered as a scene change by the algorithm.

Figure 3-10 False Positive due to Sliding Commercial Text

Figure 3-11 presents a frame sequence in which fast zooming is applied. As a result of zooming the frame content significantly changes and the algorithm presents another false alarm.

Figure 3-11 False Positive due to Fast Zooming

As we expect, most of the missed cuts are due to very similar color content of the frames surrounding the shot boundary. A typical example is shown Figure 3-12.

Figure 3-12 Missed Cut due to Similar Color Content

Another type of shot transition that most of the algorithms easily misses is the ones with very dark video content (Figure 3-13).

Figure 3-13 Frames with Dark Content is Difficult to Detect

Directors sometimes use a video effect which is called *video-in-video*. The outer frame remains constant whereas the central part of the frame includes a moving video. Such video effects make the shot boundaries difficult to detect, especially for the histogram based algorithms. Since the outer frame, which remains constant, decreases the change ratio of the histogram bins significantly, histogram based algorithms easily miss such shot boundaries.

Figure 3-14 shows an example shot boundary that our implementation missed during the experiments.

Figure 3-14 Video in Video Effects

Table 3-3 Cut Detection Results for Histogram Difference

#	Pixel-wise (Adaptive Thresh.)			Histogram Difference		
	R	P	F1	R	P	F1
1	0.74	0.82	0.83	0.83	0.83	0.83
2	0.79	0.9	0.86	0.94	0.9	0.92
3	0.89	0.88	0.92	0.96	0.85	0.9
4	0.9	0.92	0.92	0.95	0.86	0.9
5	0.85	0.81	0.88	0.91	0.84	0.87
6	0.78	0.84	0.85	0.88	0.86	0.87
7	0.9	0.89	0.92	0.95	0.89	0.92
8	0.81	0.87	0.95	0.94	0.86	0.9
Avg	**0.83**	**0.87**	**0.85**	**0.92**	**0.86**	**0.89**

Table 3-3**Error! Reference source not found.** summarizes the simulation results of the histogram difference method and compares the results with the pixel-wise difference with adaptive thresholding algorithm. The results of pixel-

wise difference algorithm are also provided for comparison. During the experiments, we have also observed that as we increase the threshold the precision results improve but the recall performance gets worse.

3.3.3. Conclusion

Simulation results indicate that histogram difference method obviously performs better than the pixel-wise method. Major reason for this is histogram method is not sensitive to local motion and local illumination changes. In the case of slight illumination changes or small camera/object motion, histogram difference method provides robust performance and better results compared to pixel-wise difference algorithm.

On the other hand, we have observed that global changes in the video frames, such as large brightness change, zooming or fading effects (especially fast zooming), result in false alarms. This is an expected result, since histogram feature is sensitive to the overall (or global) content of the video. Therefore, any effect resulting in a global change in the video content (e.g. fast zooming, large object movement) can be erroneously interpreted by the histogram algorithm.

Another conclusion from the simulation results is the algorithm cannot detect shot boundaries if there is a video-in-video effect. Since the outer frame, which remains constant, decreases the change ratio of the histogram bins significantly, amount of histogram difference is small. Consequently the transition is missed.

On the other hand, since the dark frames do not provide enough color information, histogram method cannot produce good results with dark video content as well.

3.4. Edge Change Ratio

3.4.1. Algorithm

During a cut or a dissolve, new edges appear far from the locations of old edges. In addition, old edges disappear far from the location of new edges. This

observation was applied to digital video segmentation by Zabih et. al. [15] who identified two new types of edge pixels:

- Entering pixel: One that appears far from an existing edge pixel.
- Exiting pixel: One that disappears far from an existing edge pixel.

According to Zabih et. al. it is possible to detect CUTs and GTs by counting the entering and exiting pixels. A large number of entering pixels ρ_{in} should exist during a fade in, start of dissolve and during a cut, whereas a large number of exiting pixels ρ_{out} should occur during a fade out, end of dissolve and during a cut [15].

For the implementation, two consecutive color images are converted to gray-scale and the edges are detected by using the Canny's method resulting in two binary images E and E'. Entering pixels ρ_{in} denotes the fraction of edge pixels in E' which are more than a fixed distance r from the closest edge pixel in E. Similarly, exiting pixels ρ_{out} is the fraction of edge pixels in E which are farther than r away from the closest edge pixel in E'.

Visual content discontinuity is defined as:

$$\rho = \max(\rho_{in}, \rho_{out}) \tag{12}$$

This measure is defined as *edge change ratio* (ECR) and represents the fraction of the changed edges; this fraction of the edges have entered or exited. Scene breaks can be detected by looking for the peaks in ρ.

In [15], registration techniques are offered in order to handle global motion. In [16], in order to make the algorithm robust against small object motions, edge pixels in one image which have edge pixels nearby in the other image (e.g. within 7 pixels' distance) are not regarded as entering or exiting edge pixels.

In this thesis, motion compensation is achieved by dilating each edge pixel with a diamond shape of pixels, and registration techniques are not applied. An un-

dilated frame is represented by E and its adjacent frame by E', dilated adjacent frames are represented by $\overline{E}$ and $\overline{E}'$. Edge change ratio (ECR) is computed by comparing every pixel in the first un-dilated frame, E, against the corresponding pixels in the second dilated frame E'.

There are two possibilities for this comparison:

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

2. If a pixel is found in the first frame, E, and not in the second frame $\overline{E}'$, this implies that a pixel has exited from the first frame, E.

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

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

2. If a pixel is found in the second frame, E', and not in the first frame $\overline{E}$, this implies that a pixel has entered the second frame, E'.

Taking both scenarios into account, ρ_{in} and ρ_{out} are calculated as follows [15]:

$$\rho_{in} = 1 - \frac{\sum_{x,y} \overline{E}[x,y]E'[x,y]}{\sum_{x,y} E[x,y]} \tag{13}$$

$$\rho_{out} = 1 - \frac{\sum_{x,y} E[x,y]\bar{E}'[x,y]}{\sum_{x,y} E[x,y]} \tag{14}$$

The edge change ratio ρ is the maximum value of ρ_{in} and ρ_{out} in each frame. Typical ECR patterns for both CUTs and GTs are shown in Figure 3-15.

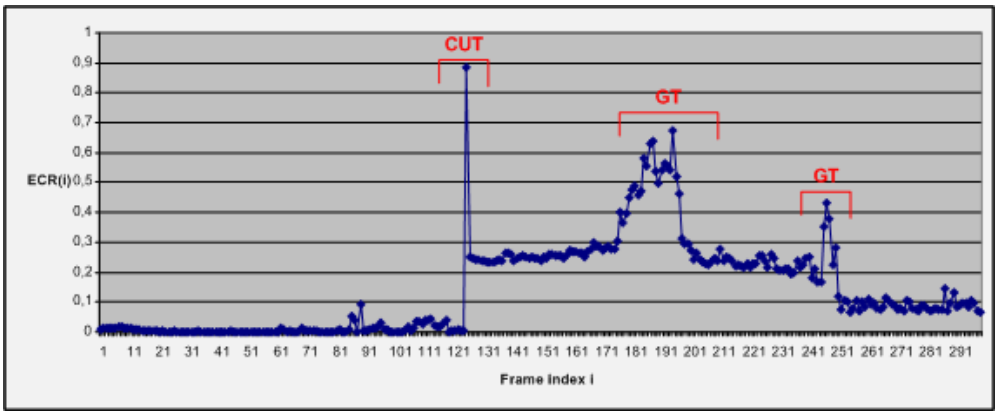

Figure 3-15 Typical ECR Patterns for CUTs and GTs

Although the cuts appear as a single peak, it is obvious that the gradual transition patterns are not in a good shape. In order to obtain a more solid pattern, we smooth the ECR signal with a sliding mean value filter with radius 4. The resulting signal is shown in Figure 3-16. The gradual transition regions appear similar to an upside down *'U–pattern'*. We identify the cut positions from the initial signal by looking for single peaks. Gradual transitions are determined by searching for the upside down *U-shape* in the smoothed ECR signal.

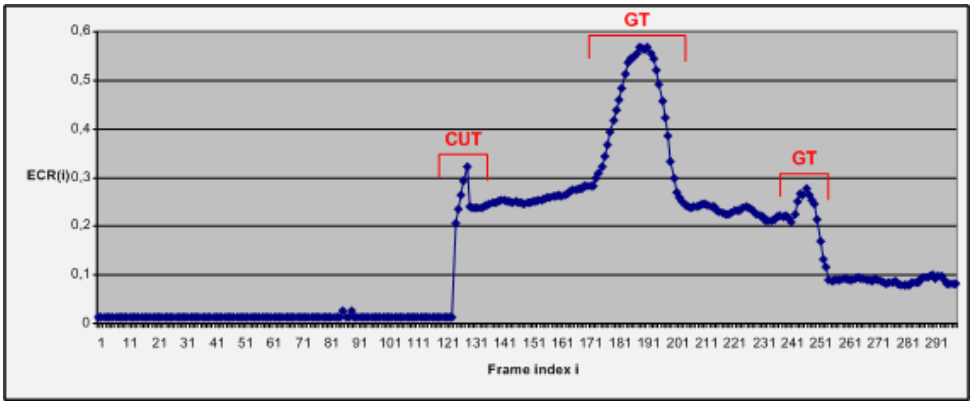

Figure 3-16 Typical ECR Patterns for CUTs and GTs after Smoothing

3.4.2. Simulation Results

Table 3-4 shows the cut detection results of ECR together with the results of pixel-wise difference and histogram difference algorithms. Results indicate that although the ECR algorithm is more complex and consumes more computational power and time, the cut detection performance of the ECR algorithm is not better than the simpler histogram difference algorithm. However, it is better than the pixel-wise difference algorithm.

Table 3-4 Cut Detection Result for ECR

#	Pixel-wise (Adaptive Thresh.)			Histogram Difference			ECR		
	R	P	F1	R	P	F1	R	P	F1
1	0.74	0.82	0.83	0.83	0.83	0.83	0.82	0.81	0.81
2	0.79	0.9	0.86	0.94	0.9	0.92	0.91	0.88	0.89
3	0.89	0.88	0.92	0.96	0.85	0.9	0.95	0.79	0.86
4	0.9	0.92	0.92	0.95	0.86	0.9	0.94	0.89	0.91
5	0.85	0.81	0.88	0.91	0.84	0.87	0.89	0.84	0.86
6	0.78	0.84	0.85	0.88	0.86	0.87	0.84	0.86	0.85
7	0.9	0.89	0.92	0.95	0.89	0.92	0.91	0.84	0.87
8	0.81	0.87	0.95	0.94	0.86	0.9	0.94	0.85	0.89
Avg	**0.83**	**0.87**	**0.85**	**0.92**	**0.86**	**0.89**	**0.90**	**0.85**	**0.87**

Quick camera/object motion is the main weakness of this algorithm. Especially large objects entering or leaving the scene causes false alarms for both cuts and GTs.

Figure 3-17 Zoom out Effect with Object Entering the Scene Causes False Alarm

Figure 3-17 shows a difficult situation for an edge-based algorithm. As the camera zooms out from the scene, arm of a man rapidly enters the scene, which results in a false detection. During the zoom out, the edges produced by the fence changes the positions. In parallel with this the arm of the men enters the scene quickly. The arm introduces lots of strong edges. As a result, the algorithm considers the high activity in the edge positions as a shot boundary.

For the gradual transitions, ECR algorithm often fails, when there is motion in both sides of the shot boundary. Especially, if the scenes, that contain strong edges, also have significant motion, ECR might produce false alarms.

Table 3-5 presents the gradual transition results for the ECR algorithm. It is obvious that the algorithm detects the gradual transitions but the performance

of the algorithm is not acceptable. False hit rate is quite high resulting in very low precision values.

The frame sequences, which contain motion of objects with significant edges, produce false GT alarms. Since significant amount of edge pixels enter or leave the scene with the motion of the objects, the algorithm identifies these intervals as GTs. Figure 3-18 shows an example of such a case. In this figure, a man in the scene moves his hands and arms quickly from the top to the bottom. At the end, the hands (especially the fingers) leave the scene which results in a significant amount of out going edges.

Table 3-5 Gradual Transition Results for ECR

#	ECR		
	Recall	Precision	F1
1	0.55	0.45	0.50
2	0.34	0.46	0.39
3	0.48	0.12	0.19
4	0.62	0.52	0.57
5	0.45	0.49	0.47
6	0.84	0.1	0.18
7	0.41	0.12	0.19
8	0.55	0.2	0.29
Avg	**0.53**	**0.31**	**0.35**

Another case that ECR algorithm fails is the long dissolves. As the edges appear and disappear very slowly the algorithm can not detect such a shot change. Figure 3-19 shows an example of a long dissolve that ECR algorithm cannot detect.

Figure 3-18 False GT Detection due to Motion of Objects with Significant Edges

Figure 3-19 ECR fails to Detect Long Dissolves

3.4.3. Conclusion

During the experiments, we have observed that if the image sequence has few colors and indistinct edges, it is hard for the ECR algorithm to detect shot boundaries. The reason for this is closely related to the bright colors. If the video content has brighter colors the edges are clear. Therefore, the results are better. Especially the cut transitions in the dark scenes result in false alarms.

The main drawback for this algorithm is its execution time. Calculating the edges and the dilation operation consumes significant computing power and in parallel takes more time to process the frames. Cut detection results can be considered acceptable, if the algorithm could be faster. However, gradual transition results are not promising.

3.5. Petersohn's Algorithm with 2-Means Clustering

3.5.1. Algorithm

In [18], Petersohn proposes a system which uses pixel, edge and histogram difference statistics for detecting CUTs and GTs. The system uses the luminance information only and down-samples all the frames by a factor of 8 in *x* and *y* directions before processing.

For CUT detection, the system utilizes histogram differences and pixel differences:

$$d^{pix}(i,i-1) = \sum_{x}\sum_{y}\left|p_i(x,y) - p_{i-1}(x,y)\right|$$
$$d^{hist}(i,i-1) = \sum_{k}\left|h_i(k) - h_{i-1}(k)\right| \quad (15)$$

where $p_i(x,y)$ denotes the intensity value of (x,y)th pixel in the ith DC frame, and $h_i(k)$ denotes the kth bin value of normalized histogram of the ith DC frame.

In order to deal with the previously mentioned weaknesses of the histogram and pixel difference features, the system jointly uses these two features and

employs 2-means clustering based decision module. In order to use two features together they need to be normalized between 0 and 1. Before constructing the 2 dimensional feature vector *unsharp masking technique* [17] is applied as a low pass filtering in the following form:

$$d_f(i,i-1) = \begin{cases} d(i,i-1) - \tilde{d}(i,i-1), \; if \;\; d(i,i-1) > \tilde{d}(i,i-1), \\ 0, \quad otherwise. \end{cases} \tag{16}$$

The difference signal $d(i,i-1)$ can be either histogram difference signal or the pixel difference signal. $\tilde{d}(i,i-1)$ denotes the median filtered $d(i,i-1)$ signal. $d_f(i,i-1)$ is the unsharp masking output. Sample pixel difference signal and the corresponding signal after unsharp masking is shown in the following figures:

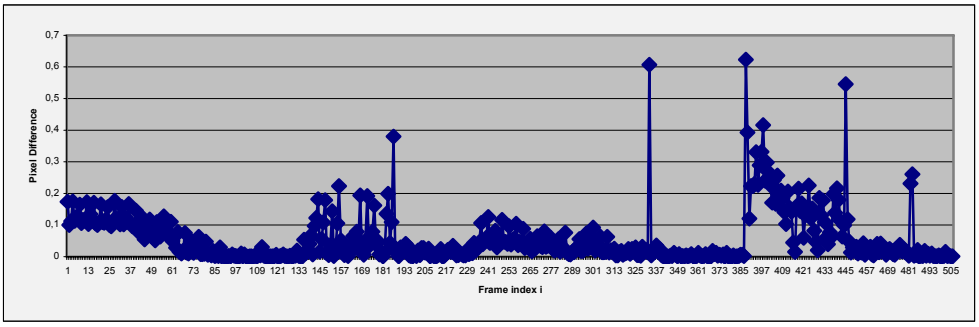

Figure 3-20 Pixel Difference Signal

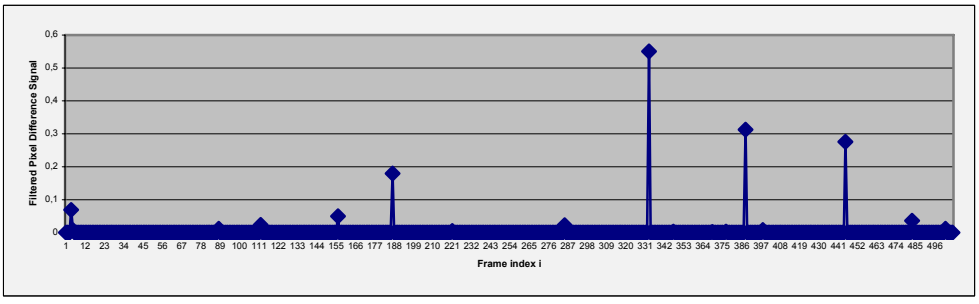

Figure 3-21 Pixel Difference Signal after Unsharp masking

Histogram and pixel difference signals obtained as a result of unsharp masking are fed to K-Means clustering algorithm, where K=2. A final processing is applied in order to eliminate the false alarms due to flashlights. The pixel and histogram differences for pairs of frames with different temporal distances around the hard cut candidate are examined and if the difference is below a certain threshold, then the candidate is marked as false alarm. Cut detection algorithm is summarized in Figure 3-22.

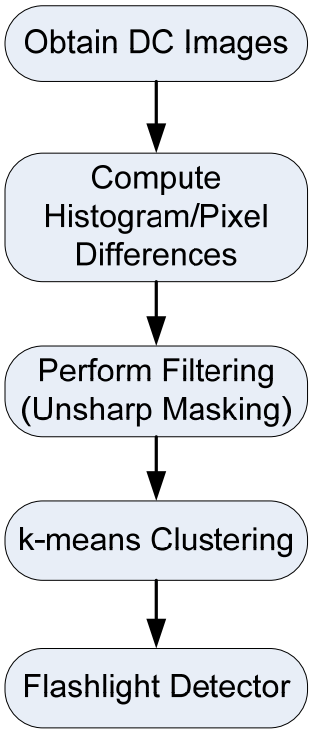

Figure 3-22 Petersohn's Cut Detection Algorithm [17]

GT detection is based on the Edge Energy. Sobel edge operator is used for detecting edges. Using the edge images of the frames, Edge Energy is calculated as the sum of the intensity values of the pixels which are marked as edges.

During a gradual transition a spectator observes a loss of contrast and sharpness of the images. Therefore, it is expected that the edges gets weaker up to some point and than, as the second image starts to appear edges get

stronger. As a consequence, Edge Energy is expected to start decreasing during a GT and local minimum appears at the center of GT [17]. Petersohn finds the GTs by locating the U-curves in the Edge Energy diagram. If the candidate is already detected as a CUT the system discards it.

The U-curves are located by calculating the Least Squares estimates of the slopes of left (m_L) and right (m_R) lines at each frame using previous and next 6 frames. If $m_L<0$, $m_R>0$ and $m_R - m_L>1$, the center of the candidate gradual cut is located. The start and end frames of the transitions are determined by searching the frames where slopes diminishes [33].

False-positives are eliminated by analyzing the histogram and edge differences for the start and end frames of the GT intervals. For correct GT intervals the histogram and edge differences are greater than specific thresholds.

3.5.2. Simulation Results

Cut detection results of the algorithm are shown in Table 3-6. The results are quite good. Both recall and precision results are successful and better than the previous algorithms.

Table 3-6 Cut Detection Results for Petersohn's SBD System

#	Pixel-wise (Adaptive Thresh.)			Histogram Difference			ECR			Petersohn		
	R	P	F1	R	P	F1	R	P	F1	R	P	F1
1	0.74	0.82	0.83	0.83	0.83	0.83	0.82	0.81	0.81	0.91	0.86	0.88
2	0.79	0.9	0.86	0.94	0.9	0.92	0.91	0.88	0.89	0.95	0.87	0.91
3	0.89	0.88	0.92	0.96	0.85	0.9	0.95	0.79	0.86	0.97	0.85	0.91
4	0.9	0.92	0.92	0.95	0.86	0.9	0.94	0.89	0.91	0.99	0.83	0.9
5	0.85	0.81	0.88	0.91	0.84	0.87	0.89	0.84	0.86	0.95	0.85	0.9
6	0.78	0.84	0.85	0.88	0.86	0.87	0.84	0.86	0.85	0.91	0.85	0.88
7	0.9	0.89	0.92	0.95	0.89	0.92	0.91	0.84	0.87	0.97	0.84	0.9
8	0.81	0.87	0.95	0.94	0.86	0.9	0.94	0.85	0.89	0.98	0.93	0.95
Avg	**0.83**	**0.87**	**0.85**	**0.92**	**0.86**	**0.89**	**0.90**	**0.85**	**0.87**	**0.95**	**0.86**	**0.90**

Gradual transition detection performance of the algorithm is not as promising as the cut detection results (Table 3-7). The recall results are very similar to the ECR results. However, the precision results are obviously better compared to the ECR.

Table 3-7 Gradual Transition Results for Petersohn's SBD System

#	Video ID	ECR			Petersohn		
		Recall	Precision	F1	Recall	Precision	F1
1	19980203_CNN	0.55	0.45	0.50	0.64	0.55	0.59
2	19980222_CNN	0.34	0.46	0.39	0.45	0.34	0.39
3	19980224_ABC	0.48	0.12	0.19	0.57	0.55	0.56
4	19980412_ABC	0.62	0.52	0.57	0.6	0.45	0.51
5	19980425_ABC	0.45	0.49	0.47	0.67	0.6	0.63
6	19980515_CNN	0.84	0.1	0.18	0.64	0.42	0.51
7	19980531_CNN	0.41	0.12	0.19	0.4	0.37	0.38
8	19980619_ABC	0.55	0.2	0.29	0.56	0.57	0.56
Avg	**All**	**0.53**	**0.31**	**0.35**	**0.57**	**0.48**	**0.52**

Since the GT detection algorithm is based on the edge difference, the algorithm fails when the scene contains lots of small objects, thereby contains lots of edges, in both side of the shot transition:

Figure 3-23 Scenes with Lots of Edges is Difficult to Detect

Similar to the ECR algorithm, when rapid object motion occurs, especially when an object with significant edges enters to scene or leaves the scene the algorithm produces false GT alarms. In Figure 3-24, the camera follows the victim of an accident, but at the same time, in the back side, a fireman walks (a large object with significant edges) in the opposite direction with the victim. Since the camera motion and the object (fireman) motion is in the opposite directions, the resulting motion is twice as much and produces great amount of re-positioning edges. Consequently the algorithm falsely identifies the situation as a shot change.

Figure 3-25 shows a false alarm due to object motion together with the illumination change. The motion of a man produces changes in the position of the edges. In parallel with this, as the camera follows the moving man, due to the position of the sun, shadows appear and disappear. The edges due to the shadows contribute significantly to the change in energy because the edges produced by shadows are quite strong.

We have also observed that zooming (especially fast zooming) results in false alarms.

Figure 3-24 Camera and Object Motion Together Results in a False GT Alarm

Figure 3-25 Object Motion and Illumination Change

3.5.3. Conclusion

During the experiments, we have observed that the algorithm is quite fast. Main reason behind this improvement in the speed of the shot boundary detection is that the system uses the luminance information only and down-samples all the frames by a factor of 8 in *x* and *y* directions.

Experiments indicate that although the system utilizes the same features (i.e. histogram, edge, pixel values), it is obviously better than all the aforementioned algorithms. Our justification for this improvement is again bases on the preference of using DC images. Using the down-sampled images (i.e. DC images) makes the system more robust to small changes. DC images are less sensitive to small camera/object motion. Therefore, algorithm senses only significant changes in the video content. In addition to using DC images, the algorithm employs pixel and histogram features together. These two together is considered as the major factor that Petersohn's system performs better than pixel difference and histogram difference methods.

In the beginning, one argues that down-sampling an image causes a significant amount of content details to be lost. However, simulation results showed that for the purpose of SBD, the overall image content is important than the details of the image.

Both ECR and Petersohn use edge information for detecting gradual transitions. Although recall results are close to each other, precision results of Petersohn are obviously better than ECR algorithm. It is evaluated that the final step of the Petersohn's algorithm for detecting false alarms by comparing the pixel and histogram difference of the start and end frames is the reason for this difference.

Dark video content makes the algorithm misses the gradual transitions due to the fact that dark image content does not produce significant edges. Similarly, rapid movement of large objects results in false gradual transition alarms, since, especially, objects with significant edges, changes the ECR and making the algorithm produce erroneous results.

Illumination change is also a strong factor of changes in the position of the edges and therefore, a significant difficulty for gradual transition detection based on edges.

3.6. Graph Partition Model with Support Vector Machine

3.6.1. Algorithm

Graph theoretic segmentation algorithms are widely used in the fields of computer vision and pattern recognition [51]. Segmentation with graph partition model [14] is one of the graph theoretic segmentation algorithms, which offers data clustering by using a graph model. Pair-wise similarities between all data objects are used to construct a weighted graph as an *adjacency matrix* (*weight matrix* or *similarity matrix*) that contains all necessary information for clustering. Representing the data set in the form of an edge-weighted graph converts the data clustering problem into a graph partitioning problem [14]. In Chapter 4, graph theoretic clustering is explained in more detail.

The graph theoretic SBD algorithm implemented during this study is based on the algorithm explained in [4].

3.6.1.1. Segmentation with Graph Cuts

Given a weighted graph G with node set V, edge set E and weight matrix W, the problem is to partition the graph into two sub-graphs A and B using an objective function. In graph theory, clustering algorithms mainly differ based on the selected objective function. There are several objective functions mostly used in graph partitioning such as min-max cut [34], normalized cut [35], ratio cut [36] [37] and minimum cut [36]. Literature shows that min-max cut algorithm performs the best among others [34]. Therefore, in this thesis, min-max cut algorithm is adopted for the objective function.

Min-max cut principle aims to minimize the similarity between clusters and maximize the similarity within a cluster. The similarity between nodes i and j is denoted by $w_{ij} \in [0,1]$. The larger the w_{ij}, the stronger the connectivity is between the nodes i and j.

The cut which divides the graph G into two sub-graphs A and B is defined as [34]:

$$cut(A,B) = \sum_{i \in A, j \in B} w_{ij} \tag{17}$$

The association of sub-graph A is defined as [34]:

$$assoc(A) = \sum_{i, j \in A} w_{ij} \tag{18}$$

A graph can be constructed by treating each sample (i.e. each frame in our concept) within a data set as a node and linking an edge between each pair of the nodes. By defining the weight of the edge as the similarity of the samples, clustering can be formulated as a graph partition problem [14]. From the shot boundary detection point of view, the objective function M_{cut}, which tries to minimize the association between the two sub-graphs while maximizing the association within each sub-graph, can be defined as [34]:

$$M_{cut}(A,B) = \frac{cut(A,B)}{assoc(A)} + \frac{cut(A,B)}{assoc(B)} \tag{19}$$

3.6.1.2. Temporal Constraints

The drawback with this objective function is it is not applicable to large data sets. For every possible sub-graph, this objective function must be calculated. The number of sub-graphs is in the exponential degree. However, in the problem domain of shot boundary detection, we can apply temporal constraints to the problem and decrease the number of candidate sub-graphs remarkably.

In a video sequence, a cut can occur at a position between any two adjacent nodes (Figure 3-26). Given that there are *N* nodes in graph *G*, there are *N-1* possible positions for the cuts. For example, if the graph includes 4 nodes, there are 3 possible positions for dividing the graph into two sub-graphs:

1. Node 1 ∈ sub-graph A, Node 2,3,4 ∈ sub-graph B. (cut between node 1 & 2),
2. Node 1,2 ∈ sub-graph A, Node 3,4 ∈ sub-graph B. (cut between node 2 & 3),
3. Node 1,2,3 ∈ sub-graph A, Node 4 ∈ sub-graph B. (cut between node 3 & 4)

Therefore, the objective function M_{cut} must be calculated for all *N-1* possible positions (instead of 2^N) and the minimum value will indicate the optimum position of a cut for partitioning the graph.

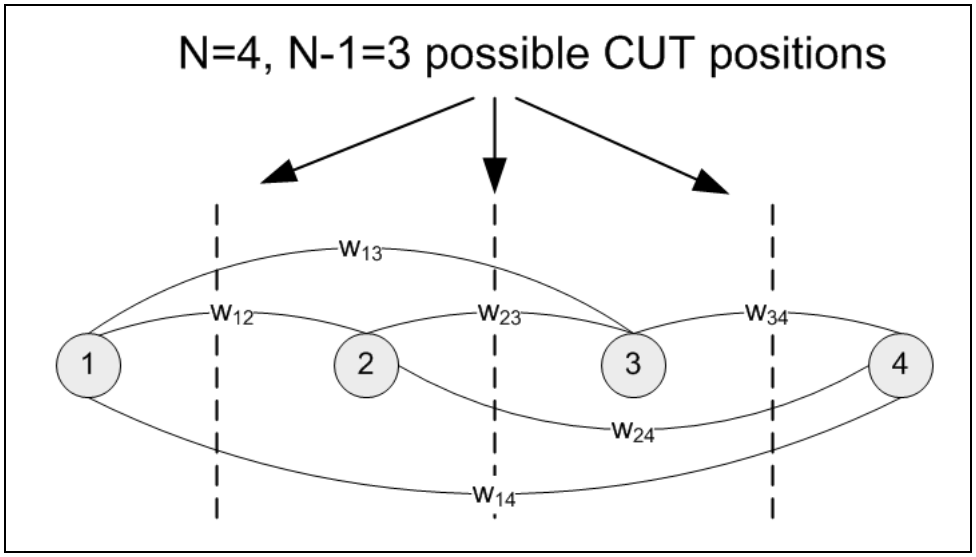

Figure 3-26 Sample Graph with 4 Nodes

The continuity signal can be defined as [4]:

$$score(t) = M_{cut} = \left(\{1,2,...,t\},\{t+1,t+2,...,N\}\right) \quad (20)$$

Since the *score* values are calculated based on the weights, definition of the w_{ij} is the most critical part of the algorithm. The weight w_{ij} is usually defined as [4]:

$$w_{ij} = sim(i,j) \times \begin{cases} e^{\frac{-\|i-j\|^2}{\sigma^2}}, & if\ |i-j| < r \\ 0, & otherwise \end{cases} \quad (21)$$

The weight value between frame *i* and frame *j* reflects the likelihood (i.e. similarity) that two frames belong to the same shot. Therefore, *sim(i,j)* in (21) denotes the similarity function. In [4] histogram intersection method is adopted as the similarity measure:

$$w_{ij} = \sum_k \frac{\min(H_k^i, H_k^j)}{H_k^i} \times \begin{cases} e^{\frac{-\|i-j\|^2}{\sigma^2}}, & if\ |i-j| < r \\ 0, & otherwise \end{cases} \quad (22)$$

Based on this measure, for more similar frames *i* and *j*, w_{ij} should be much higher. On the other hand, as the distance between frames *i* and *j* increases, the probability that these two frames belong to the same shot decreases. Therefore, σ in (21) and (22) is a factor reflecting the similarity decaying with the temporal interval increasing, and, *r* denotes the maximum range in which the frames are considered to influence each other. As a result the calculations will be restricted in a $r \times r$ sub-matrix which is called *active matrix*. Consequently, the continuity signal can be redefined as [4]:

$$score(t) = M_{cut} = \left(\{t-r, t-r+1..., t\}, \{t+1, t+2, ..., t+r\}\right) \quad (23)$$

3.6.1.3. Cut Detection

Note the difference between (20) and (23), the continuity signal is now being calculated by using the information neither from all of the frames, nor just from the consecutive frames, but according to the content variation within an interval of range $2 \times r$. Therefore, over the curve of continuity signal, a valley-shape is expected for a cut transition instead of an isolated peak (Figure 3-27). Based on this fact the cut positions can be determined by seeking for the sharp valleys in the continuity signal.

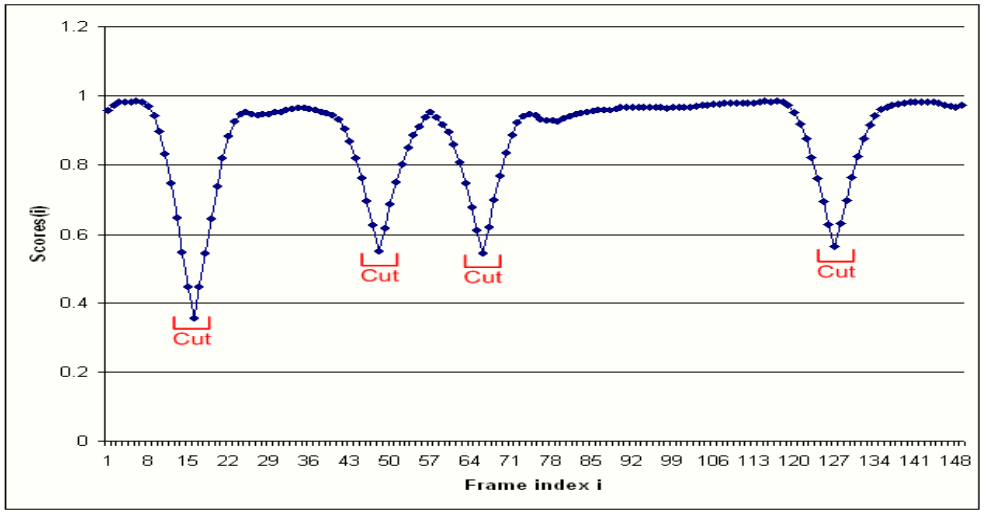

Figure 3-27 Typical Cut Patterns for Scores Signal

The segmentation algorithm can be summarized as follows [4]:

1. Given a video file, construct a weighted graph G. Treat each frame as a node and link each other by an edge.
2. Compute w_{ij}, the weight of each edge, to obtain the similarity matrix W.
3. Calculate scores of the feasible cuts according to (23).
4. Find the *V-patterns* in the scores signal and select the local minima as the candidate cut positions.
5. Declare the candidates whose score values are below a pre-defined threshold as cuts.

3.6.2. GT Detection

Since the abrupt changes occur between two adjacent frames, the similarity matrix exhibits some kind of chessboard pattern for the cuts (Figure 3-28) [4]. This behavior makes it possible to identify the cut positions by using a single r value in (23).

On the other hand, GTs spread over an interval and are difficult to find with a single *r* value. The similarity matrix exhibits a blurry pattern at the GT intervals as shown in the middle plot in Figure 3-28. In [4] and [14], it is shown that for

different values for the length of the GT, there will always be a clear "chessboard" pattern at a lower resolution (right figure in Figure 3-28).

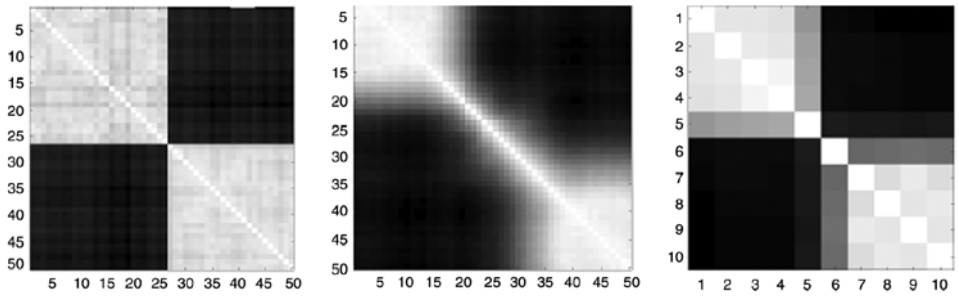

Figure 3-28 Similarity Matrix Patterns for Cuts and GTs (Left: Cut, Middle: GT, Right: GT pattern at a lower resolution) [4]

In order to detect GTs, the continuity signal can be re-defined as [4]:

$$score(t,\delta) = M_{cut}(\{t-(r-1)\times\delta,\ldots,t-\delta,t\},\{t+\delta,\ldots,t+r\times\delta\}) \qquad (24)$$

where $\delta \in \{1,2,\ldots\}$ denoting the sampling rate of frames. Equation (24) means that when calculating the continuity signal for a candidate transition position, instead of involving all the frames in a neighborhood, the algorithm only samples every δ frames in a broader range. In this manner, it is possible to construct multiple temporal resolution graphs, while δ varies.

3.6.2.1. Support Vector Machine

Identifying the shot boundaries by thresholding the obtained scores signal does not provide satisfactory results. In [4], Yuan et. al. propose employing a Support Vector Machine (SVM) with active learning strategy in order to identify the shot boundaries according to the shapes of local minima.

Each shot boundary corresponds to a valley-shaped pattern in the scores signal. However, the reverse is not true; not every local minimum is a shot boundary. Only using a thresholding method does not help us in classifying

between shot boundaries and non-boundaries. Observation of the typical *scores* signal indicates that the valley patterns corresponding to the boundaries are different from the local minima which correspond to non-boundaries. Based on this observation, by constructing feature vectors, which characterizes the shape of the valleys, and feeding this information to the SVM classifier, it could be possible to discriminate the boundaries and non-boundaries.

Let s_t denote the score value at cut candidate t and the shape of the valley centering at s_t can be characterized by the feature vector B_t^r [4]:

$$B_t^r = (s_{t-r}, \ldots, s_t, s_{t+1}, \ldots, s_{t+r}) \tag{25}$$

where r is the same as in (24).

Due to the multi-resolution analysis, the feature vectors for GTs will be different than the cuts. Instead of constructing multi-resolution score signal for GTs, changing the sampling rate of the scores signal should provide us with the similar results. Let δ denote the sampling rate of the continuity signal, the shape of the valley centering at a GT candidate t can be characterized by the feature vector B_t^δ:

$$B_t^\delta = (s_{t-r\times\delta}, s_{t-(r-1)\times\delta}, \ldots, s_t, s_{t+\delta}, \ldots, s_{t+r\times\delta}) \tag{26}$$

Note that the length of the feature vector does not change with δ, but the length of the neighborhood changes.

By concatenating the feature vectors with different resolutions (i.e. with different δ), it is possible to represent a GT candidate with a single feature vector. With $\delta \in \{1,3,5\}$ the candidate centering at t can be described by the following multi-resolution representation:

$$B_t = (B_t^1, B_t^3, B_t^5) \tag{27}$$

Instead of manually labeling all local minima, an active learning strategy can be employed. All the valleys which are under a specified threshold are identified and the SVM classifier is trained with the feature vectors in the form of (25) and (27) for cuts and GTs respectively.

3.6.3. Simulation Results

The television programs from national broadcast and some of the TRECVID 2002 SBD test videos are used as the training set for the SVM. The SVM code from [38] is used during the experiments.

The training videos include news programs, commercials and movies. Totally 18114, including 4707 positive and 7880 negative samples are used to train SVM of a Gauss kernel function. 11 and 60 dimensional samples are used for cuts and GTs respectively. Table 3-8 summarizes the training set.

Table 3-8 Training Set for SVM

#	Video ID	CUT		DIS	
		Positive	Negative	Positive	Negative
1	ATV	548	2052	404	1313
2	Kanalturk	20	217	111	93
3	NTV2006	512	1183	269	955
4	NTV2007	726	653	142	1012
5	SHOWTV	586	585	134	806
6	STAR	82	417	12	330
7	TRT1	621	1104	93	1079
8	USG1	0	340	86	168
9	USG4	151	201	63	227
10	LivingSt	0	98	10	60
11	USG9	0	381	137	133
Total	**All**	**3246**	**7231**	**1461**	**6176**

Table 3-9 gives the cut detections results of the graph partition method. The results show that cut detection performance of the graph partition method is very similar by the histogram difference method.

Gradual transition detection results of graph partition algorithm are summarized in Table 3-10. Although cut detection performance of the graph partition algorithm does not outperform the others, it is obvious that GT transition performance yields the best results so far. The recall results of this algorithm are quite close to that of the Petersohn's SBD system, whereas precision results are far better than Petersohn's.

The score values of some GTs result in a V-shape, which are very similar to the shapes that occur as a result of hard cuts. Such erroneous decisions occur generally in two situations. If the dissolve spreads over a long interval so that frame differences are small, the resulting similarities between the frames belonging to different shots are akin to the ones within the same shot with disturbances. Figure 3-29 shows a dissolve sequence which is identified as a cut by the algorithm.

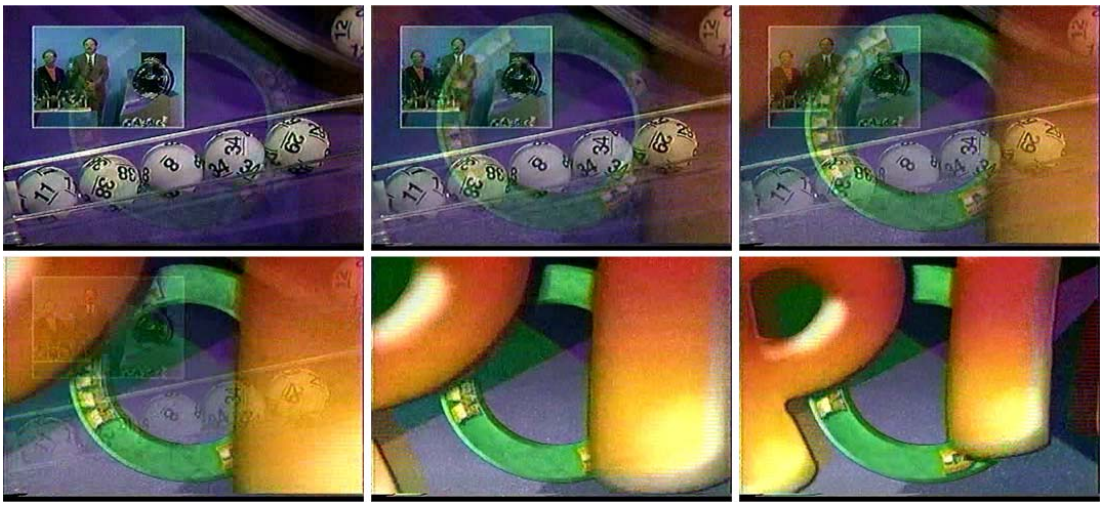

Figure 3-29 A Dissolve Identified as a Cut (sample 1)

Another situation is due to the case where two shots surrounding the dissolve area have similar color histograms. Figure 3-30 illustrates this situation.

Figure 3-30 A Dissolve Identified as a Cut (sample 2)

Table 3-9 Cut Detection Results for Graph Partition with SVM Algorithm

#	Pixel-wise (Adaptive Thresh.)			Histogram Difference			ECR			Petersohn			Graph Partition with SVM		
	R	P	F1	R	P	F1	R	P	F1	R	P	F1	R	P	F1
1	0.74	0.82	0.83	0.83	0.83	0.83	0.82	0.81	0.81	0.91	0.86	0.88	0.67	0.89	0.76
2	0.79	0.9	0.86	0.94	0.9	0.92	0.91	0.88	0.89	0.95	0.87	0.91	0.91	0.93	0.92
3	0.89	0.88	0.92	0.96	0.85	0.9	0.95	0.79	0.86	0.97	0.85	0.91	0.9	0.92	0.91
4	0.9	0.92	0.92	0.95	0.86	0.9	0.94	0.89	0.91	0.99	0.83	0.9	0.94	0.94	0.94
5	0.85	0.81	0.88	0.91	0.84	0.87	0.89	0.84	0.86	0.95	0.85	0.9	0.89	0.87	0.88
6	0.78	0.84	0.85	0.88	0.86	0.87	0.84	0.86	0.85	0.91	0.85	0.88	0.76	0.92	0.83
7	0.9	0.89	0.92	0.95	0.89	0.92	0.91	0.84	0.87	0.97	0.84	0.9	0.92	0.94	0.93
8	0.81	0.87	0.95	0.94	0.86	0.9	0.94	0.85	0.89	0.98	0.93	0.95	0.88	0.93	0.9
Avg	**0.83**	**0.87**	**0.85**	**0.92**	**0.86**	**0.89**	**0.90**	**0.85**	**0.87**	**0.95**	**0.86**	**0.90**	**0.86**	**0.92**	**0.88**

Table 3-10 GT Detection Results for Graph Partition with SVM Algorithm

#	Video ID	ECR			Petersohn			Graph Partition with SVM		
		Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
1	19980203_CNN	0.55	0.45	0.50	0.64	0.55	0.59	0.66	0.79	0.72
2	19980222_CNN	0.34	0.46	0.39	0.45	0.34	0.39	0.63	0.83	0.72
3	19980224_ABC	0.48	0.12	0.19	0.57	0.55	0.56	0.58	0.83	0.68
4	19980412_ABC	0.62	0.52	0.57	0.6	0.45	0.51	0.62	0.94	0.75
5	19980425_ABC	0.45	0.49	0.47	0.67	0.6	0.63	0.67	0.86	0.75
6	19980515_CNN	0.84	0.1	0.18	0.64	0.42	0.51	0.5	0.83	0.62
7	19980531_CNN	0.41	0.12	0.19	0.4	0.37	0.38	0.61	0.8	0.69
8	19980619_ABC	0.55	0.2	0.29	0.56	0.57	0.56	0.67	0.78	0.72
Avg	**All**	**0.53**	**0.31**	**0.35**	**0.57**	**0.48**	**0.52**	**0.62**	**0.83**	**0.71**

3.6.4. Conclusion

The first observation about graph partition algorithm is that it consumes significant computing power. Processing the histograms for each frame and then constructing the weight matrix (even for the active matrix) is quite expensive.

Graph partition method is the only aforementioned algorithm that shows a good performance on detecting the flashlights. This is expected, since continuity signal is not calculated by using the information just from the consecutive frames, but according to the content variation within an interval of range $2 \times r$. The frames before and after the flashlight position exhibit high similarity. Therefore, the flashlights are not identified as transitions during graph partitioning. This kind of contextual processing provides a more robust method for also against other sources of noises and disturbances.

One of the remarkable improvements of this algorithm is its high precision results. For both cuts and GTs, precision results are significantly better than the other algorithms. This property makes this algorithm the most robust and main reason for robustness is due to the machine learning algorithm applied at the decision module.

We have observed that some of the dissolves exhibits V-shape patterns very similar to the ones that are produced by cuts. As a result the algorithm falsely identifies the dissolve center as cut. Long dissolves are the situations that such false detections mostly occur.

The performance of this algorithm depends highly on the performance of the SVM. Performance of the SVM increases with the size of the training set with good samples. In the early stages of the experiments with only a relatively small set of training data, the experimental results were quite poor. As we performed the training on a larger training set, it is observed that the performance gets significantly better. It is difficult to find training data including enough number of GTs. Therefore, it could be concluded that the GT detection performance of the

algorithm should be better, if training data is as much as one could find for the hard cuts.

3.7. Motion-Based Algorithm

3.7.1. Algorithm

The motion based SBD algorithm proposed by Kawai et al. is one of the best cut detection algorithms in TRECVID 2007 [19] [3]. In this thesis, as a motion–based SBD algorithm, we have adopted this method with slight modifications.

Similar to the Petersohn's algorithm, frames are down-sampled by a factor of 2 in both *x*- and *y*-directions. The proposed algorithm further performs a preprocessing step for filtering out the obvious non-boundary frames. With this achievement, it is allowed to spend more processing power on the areas that are likely to be shot boundaries.

In this early processing step, sum of absolute differences between the R, G and B values are calculated for candidate shot boundary frames:

$$d_{SAD}(i,i-1) = \frac{1}{|X \times Y|} \sum_{x} \sum_{y} |p_i(x,y) - p_{i-1}(x,y)| \tag{28}$$

where $p_i(x,y)$ denotes the R, G or B value of $(x,y)^{th}$ pixel in the iith down-sampled frame, whereas *X* and *Y* denote the dimensions of the down-sampled frame. The difference is calculated for all color spaces and the sum gives the SAD value. If the SAD value is below a threshold, we justify that this frame pair cannot be a shot boundary and we skip the remainder of the process.

If the SAD value is above a threshold, a detailed frame difference is calculated based on block matching. For the block matching algorithm, one divides the frames into 24 blocks (6x4) and for each block in the current frame searches the previous frame in order to find the best match with minimum cost:

$$\lambda_n(f_{i-1}, f_i) = \min_{v \in S_n} \{ d_{HIST}(f_{i-1}(r+v), f_i(r)), r \in B_n \} \tag{29}$$

λ_n represents the minimum cost between the n[th] block of the current frame and its best match in the previous frame. $f_i(r)$ represents the pixels of the n[th] block in the current frame. S_n is the search range, while v is the estimated motion vector. $f_{i-1}(r+v)$ indicates the sliding block in the previous frame searching for the best match (Figure 3-31).

As explained in [19], calculation of the motion vector v is the part that consumes most of the computational power, as well as execution time. In order to increase the speed of the algorithm, Dual Cross Search (DCS) algorithm is preferred as the block matching algorithm [39]. DCS algorithm improves the speed of block matching task by three effective steps: initial search center prediction, early search termination and dual cross search pattern.

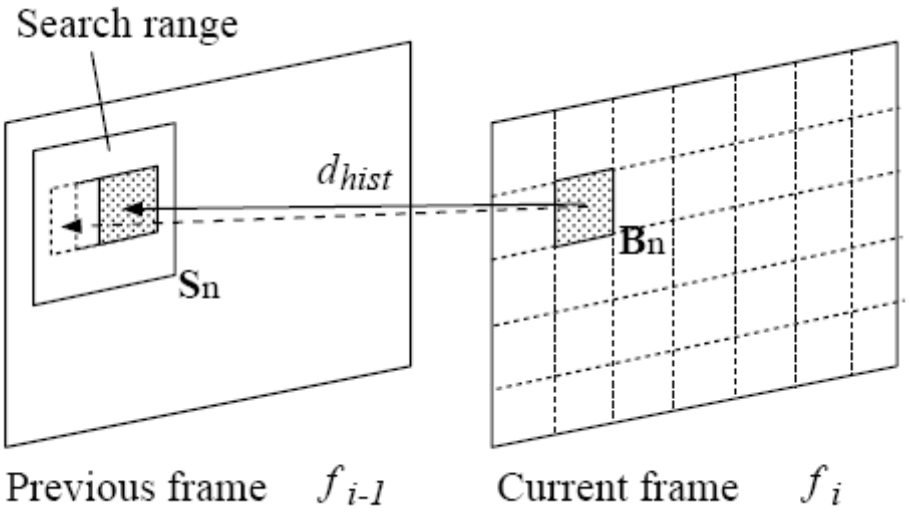

Figure 3-31 Block Matching Scheme

For the cost calculations, instead of sum of absolute pixel differences, we adopted the histogram intersection method, since the histogram based methods are more proper for SBD detection task:

$$d_{HIST} = \sum_{k} \frac{\min(H_k^i, H_k^{i-1})}{H_k^i} \tag{30}$$

For each block, if the resulting cost is above a certain threshold, it is evaluated that the block is changed more than a regular motion:

$$V(n) = \begin{cases} 1 & if \quad \lambda_n(f_{i-1}, f_i) > T_\lambda \\ 0 & else \end{cases} \tag{31}$$

$V(n)$ is set to 1, if the calculated cost value for the current block indicates a motion while being larger than a specific threshold. Therefore, we sum up such blocks in a particular frame in order to find out the percentage of the blocks producing a large motion vector (or a large minimum cost), which is most probably resulting from a shot change, not a regular motion:

$$d_{bm}(f_{i-1}, f_i) = \sum_{n=1}^{N} V(n) \tag{32}$$

As a consequence, frame difference is calculated based on the block matching. At the end, if the number of blocks that are marked as having extraordinary motion is above a certain value, one decides for an abrupt shot transition. Therefore, shot boundaries can be identified by comparing the d_{bm} in (32) by a threshold. However, during the intervals of the video with significant motion, this method may not present good results. A more robust way of finding the cuts is to search for an increase in d_{bm}:

$$d_{bm}(f_{i-1}, f_i) - d_{bm}(f_{i-2}, f_{i-1}) > T_{CUT} \tag{33}$$

Finally, a flashlight detector is also used for the false alarms. The proposed method finds the minimum intensity image for the 3 frames neighborhood of the current SBD candidate:

$$f_i^{'}(r) = \min(f_i(r), f_{i+1}(r), f_{i+2}(r), f_{i+3}(r)) \tag{34}$$

If the change in d_{bm} is due to a flashlight at f_i, then the sum of absolute difference between f_{i-1} and $f_i^{'}$ will be very small. Therefore, if the so-called difference is below a threshold we skip the candidate and identify it as a flashlight.

3.7.2. Simulation Results

Table 3-11 summarizes the cut detection results of the algorithm. The recall results are the best with an average value of 0.97, which is quite noteworthy. On the other hand, the precision results are not as good as its recall results.

The flashlight detector of the algorithm does not perform well. Especially, if the scene has a strong illumination, or the flashlight spreads to more than one frame, the proposed algorithm usually fails (Figure 3-32).

Figure 3-32 Flashlight Detector Fails

The examined algorithm also fails to detect shot changes for dark scenes (see Figure 3-33).

Figure 3-33 A Missed Cut due to Dark Scene

Video-in-video kind of shot boundaries are also difficult to detect by such an algorithm. Since the blocks of the outer video frame do not change, the number of blocks changed in the center of the video does not produce enough increase in the d_{bm} signal. In addition to this, if the inner frame includes a dark content, it gets even more difficult for this algorithm to detect the transition:

Figure 3-34 Failure of Detection due to Video Effects

Since this method adopts histogram based block matching algorithm, we observe the weaknesses of histogram based methods in this algorithm as well. For example, the overall illumination changes cause the algorithm to produce false alarms (see Figure 3-35):

Figure 3-35 False Alarm due to Overall Illumination Change

Table 3-11 Cut Detection Results for Motion Based Algorithm

#	Pixel-wise (Adaptive Thresh.)			Histogram Difference			ECR			Petersohn			Graph Partition with SVM			Motion with Dual Cross Search		
	R	P	F1	R	P	F1	R	P	F1	R	P	F1	R	P	F1	R	P	F1
1	0.74	0.82	0.83	0.83	0.83	0.83	0.82	0.81	0.81	0.91	0.86	0.88	0.67	0.89	0.76	0.94	0.73	0.82
2	0.79	0.9	0.86	0.94	0.9	0.92	0.91	0.88	0.89	0.95	0.87	0.91	0.91	0.93	0.92	0.98	0.80	0.88
3	0.89	0.88	0.92	0.96	0.85	0.9	0.95	0.79	0.86	0.97	0.85	0.91	0.9	0.92	0.91	0.96	0.80	0.87
4	0.9	0.92	0.92	0.95	0.86	0.9	0.94	0.89	0.91	0.99	0.83	0.9	0.94	0.94	0.94	0.99	0.76	0.86
5	0.85	0.81	0.88	0.91	0.84	0.87	0.89	0.84	0.86	0.95	0.85	0.9	0.89	0.87	0.88	0.97	0.68	0.80
6	0.78	0.84	0.85	0.88	0.86	0.87	0.84	0.86	0.85	0.91	0.85	0.88	0.76	0.92	0.83	0.92	0.73	0.81
7	0.9	0.89	0.92	0.95	0.89	0.92	0.91	0.84	0.87	0.97	0.84	0.9	0.92	0.94	0.93	0.98	0.75	0.85
8	0.81	0.87	0.95	0.94	0.86	0.9	0.94	0.85	0.89	0.98	0.93	0.95	0.88	0.93	0.90	0.99	0.85	0.91
Avg	**0.83**	**0.87**	**0.85**	**0.92**	**0.86**	**0.89**	**0.90**	**0.85**	**0.87**	**0.95**	**0.86**	**0.90**	**0.86**	**0.92**	**0.89**	**0.97**	**0.76**	**0.85**

3.7.3. Conclusion

The algorithm is one of the best cut detection algorithms reported in TRECVID 2007 [3]. Our experimental results also support this performance with the best recall results. However, we are not able to observe a remarkable precision result for this algorithm.

Generally, motion-based algorithms are not preferred in the uncompressed domain, since estimation of motion vectors consumes significant computational power and time. In contrary, this algorithm is the fastest among the methods that have been tested in this thesis. This observation is mainly due to the preprocessing step for skipping the frames which have very low probability of being a shot boundary. Secondly, utilizing down-sampled images together with the fastest block matching algorithm (i.e. Dual Cross Search) increases the speed of the algorithm significantly.

During the experiments, we have observed that motion is the major factor that makes the cut detection harder. Therefore, it is expected that a motion based SBD algorithm could produce good detection performance. However, precision results should be improved.

3.8. Discussion

In this thesis, we have implemented six popular state-of-the-art SBD algorithms. In order to gain an integral idea about SBD problem, we have selected these algorithms adopting different content representation, various continuity signal construction and different classifiers.

We have observed that although pixel-based algorithm is known to be the primitive one in the field, with appropriate improvements such as adaptive thresholding, it is possible to obtain quite promising cut detection results.

Experimental results indicated that histogram-based algorithms are very applicable to SBD. They provide global information about the video content and are less sensitive to local changes. We have also observed that algorithms can

be further improved by quantization in the color spaces. The resulting algorithms are observed to be faster without any performance degradations.

We have also observed that the algorithms adopting complex features, such as edges, do not outperform the performance of the simpler algorithms using pixel or histogram information. Complex features additionally increase the necessity for more computational power, which makes them less preferable.

On the other hand, our simulation results show that for the purpose of SBD, the overall image content is important than the details of the image. Therefore, down-sampled images can be used to both increase the speed of the algorithm, as well as to enhance the robustness against motion and disturbances. Increasing the speed of the algorithm let us use more than one feature, which further increases the overall performance of an algorithm.

During the experiments, we have realized that motion is the major factor that makes the cut detection precise. The simulation results showed that a good motion based SBD algorithm could produce a remarkable detection performance.

Although graph-theoretic approaches with machine learning are computationally complex, they are very promising algorithms with very trustable results. They perform best flashlight detection and are less sensitive to disturbances.

Gradual transition detection is a very difficult problem due to the fact that there are various types of editing effects, while each editing effect has a different temporal pattern, which makes it very difficult to detect with a single algorithm. Due to similar reasons, in 2007, TRECVID organization decreased the amount of gradual transitions in the test videos significantly and completely concentrated on cut detection.

CHAPTER 4

DOMINANT SETS

Cluster analysis aims to understand the internal structure within a given dataset. In [40] Pavan applies cluster analysis to the domain of computer vision, specifically to image segmentation and image database organization problems. A framework is developed for image segmentation problem based on a novel graph theoretic formulation of clustering, namely *dominant sets*. In this thesis, we have applied this new concept to the SBD problem. The experimental results show that this new concept can be used in detecting abrupt scene transitions.

In this chapter, we will start with an introduction of the fundamentals of graph theory and its relevant definitions. The novel combinatorial concept, *dominant sets,* is explained based on these definitions. Finally, we propose a SBD algorithm based on the dominant sets concept. The performance of the proposed algorithm is to be evaluated against the TRECVID 2003 SBD test set.

4.1. Graph Theoretic Clustering

Most of the pattern recognition problems are quite difficult that it is not possible to guess the optimum classification decision in advance. Therefore, one spends most of the time during learning. Learning refers to some form of algorithm for reducing the error on a set of training data [41]. Unsupervised learning is a type of machine learning, in which manual labels of inputs (i.e. training set) are not used.

Clustering is considered to be major unsupervised learning problem. One can define clustering as “the process of organizing objects into groups whose members are similar in some way”. Therefore, the objects within a cluster are expected to be similar between them and dissimilar to the objects from other clusters [42].

Data representation can be considered as the first step in solving a clustering problem. Different data representation types can be employed for different clustering problems. Among these two types of representations are widely used for clustering problems in computer vision fields, while one of them is called geometric representation, in which data items are mapped to some real normed vector space (i.e. feature space). The other type, denoted as the graph representation, maps the data items to the nodes of a graph [43]. Representation of the data in the form of a graph brings the problem into the graph theory domain

4.1.1. Fundamentals of Graph Theory

Some key terminology, which will help us in understanding the graph theoretic clustering, is presented below.

A *graph* is a set of objects called nodes or vertices connected by links, namely lines or *edges*. Each *edge* has a set of one or two vertices associated to it, which are denoted as its *endpoints.* An *edge* is said to join its endpoints. Edges are generally represented by the pair of vertices that they are connecting.

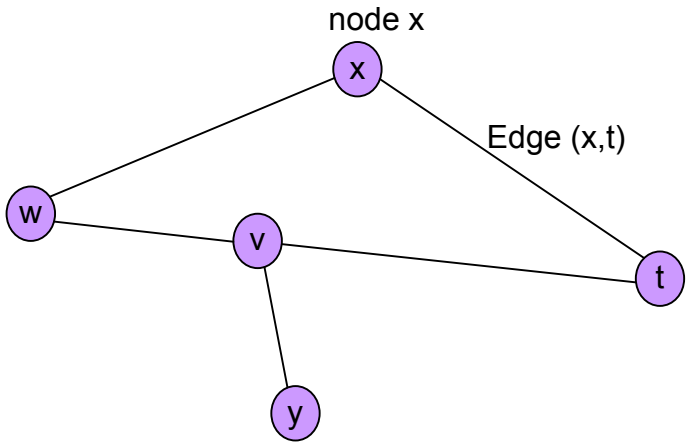

Figure 4-1 A Simple Undirected Graph

A node *u* is *adjacent* to node *v* if they are joined by an edge. Two adjacent nodes might also be called *neighbors*.

In a proper graph, which is by default *undirected*, a line from point A to point B is considered to be the same thing as a line from point B to point A. Formally, an *undirected graph* is one in which no distinction is drawn between edges (a,b) and (b,a) [44].

A *weighted graph* is one in which a weight is associated with each edge.

A *clique* of a graph *G* is a subset of nodes *V* such that all vertices are pair-wise adjacent. A *maximal clique* is a clique that is not contained in any other clique. A *maximum clique* is a maximal clique of maximum size. In Figure 4-2, graph G consists of 5 nodes. The sub-graphs S_1 and S_2 are both cliques of graph G because all nodes are pair-wise adjacent. They are both maximal cliques since they are not included in any other sub-graph of G. Sub-graph S_1 is the maximum clique of G because it has more nodes than sub-graph S_2.

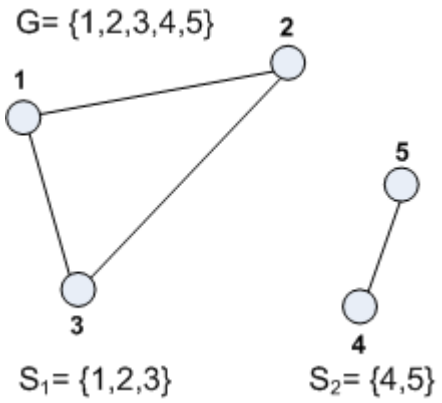

Figure 4-2 Maximal and Maximum Cliques

4.1.2. General Clustering Approach

In graph theory, clustering is considered as dividing the graph into “good pieces”, which is called graph partitioning. Firstly, each data element in the data set to be clustered is mapped to a node in the graph. The next, and maybe the most critical, step is to determine the similarity metric. All the data items in the set will be compared in a pair-wise manner with all the other data items according to this similarity metric. Whether two nodes are alike and therefore be in the same cluster is mostly determined by this similarity metric. In other words, the criteria to determine “good pieces” are the similarity metric [45].

Following the identification of the similarity metric, a weighted similarity matrix (or affinity matrix) is formed by using the pair-wise similarities between all data items. There is a row and column for each data item in the similarity matrix. The $(i,j)^{th}$ element of the matrix represents the similarity that is calculated for data item *i* and data item *j* based on the similarity metric [4]. All the elements of the similarity matrix is calculated in this manner. The motivation behind graph theoretic clustering algorithms is the idea that weighted similarity matrix contains all the information necessary for clustering [34].

In order to achieve the weighted graph representation of the data set, an edge is constructed from every data item to every other, and corresponding similarity value from the previously formed weighted similarity matrix is assigned to this edge as a weight. In general, if two nodes are not similar (i.e. zero similarity) the edge between these two nodes is not shown in the graph (Figure 4-3).

At this point, since the graph representation is ready, we have already transferred the data clustering problem into a graph partitioning problem. Therefore, the final step is to obtain an appropriate algorithm to cut the graph into sub-graphs which have relatively large interior weights.

Figure 4-4 from [45] summarizes the partitioning process. On the top left, graph representation of the problem in the form of an undirected weighted graph is given. On the top right is a common visualization of the similarity matrix of this graph. Larger similarity values are indicated with lighter color.

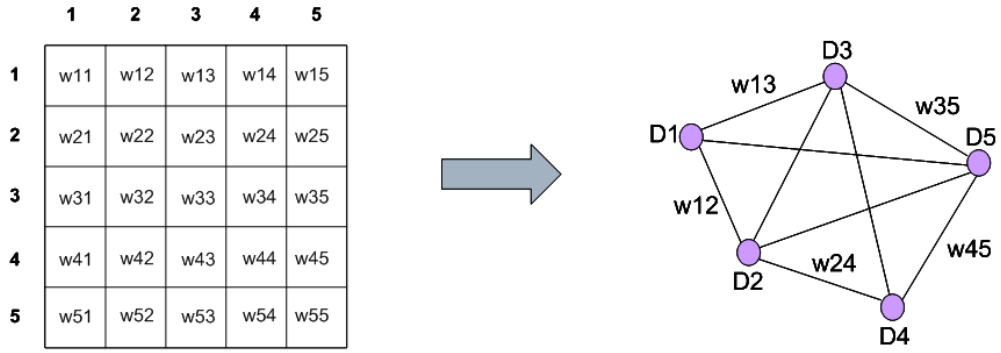

Figure 4-3 Graph Representation from Similarity Matrix

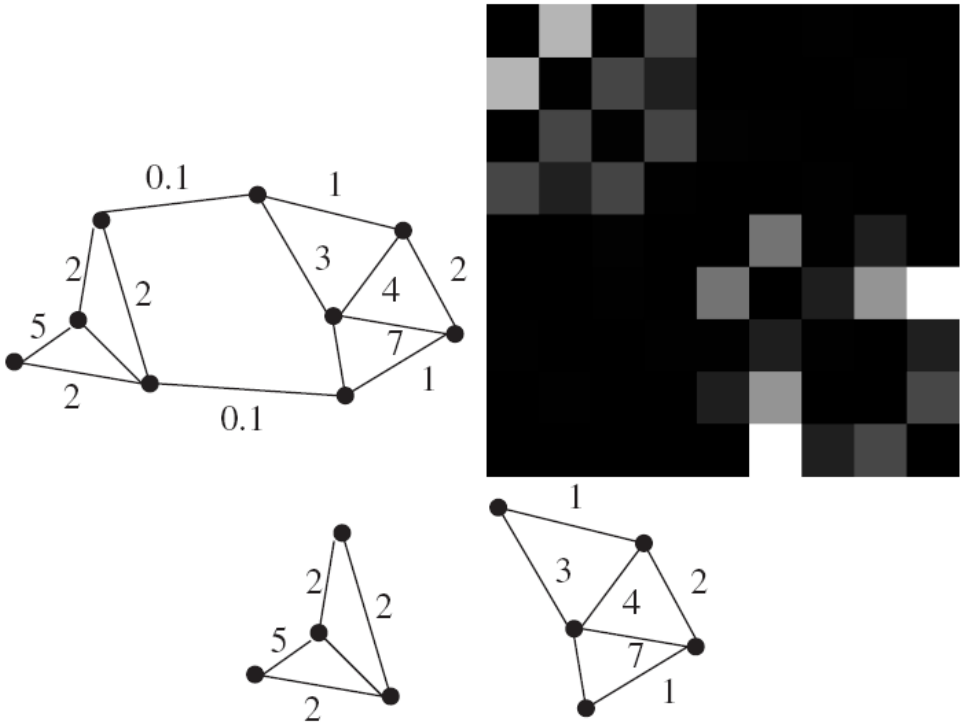

Figure 4-4 Graph Partition Example

"By associating the vertices with rows (and columns) in a different order, the matrix can be shuffled. The ordering is chosen to show the matrix in a form that

emphasizes the fact that it is very largely block-diagonal. The figure on the bottom shows a cut of that graph that decomposes the graph into two tightly linked components. This cut decomposes the graph's matrix into the two main blocks on the diagonal" [45].

4.2. Maximum Clique Problem and Motzkin-Straus Theorem

There are different approaches for dividing a graph into good pieces. Among these methods, a classic approach formulates the clustering problem as "maximum clique problem". The maximum clique problem, which searches for the maximum cliques in a graph, is the strictest definition of a cluster [46].

Motzkin-Straus theorem is a remarkable contribution to the solution of the maximum clique problem by establishing a connection between the maximizer of the Lagrangian of a graph and its maximum cliques. The basic idea can be summarized as follows [47]:

A good cluster is one where elements that are strongly associated with the cluster also have large values in the similarity matrix. Let vector **x** (generally referred as characteristic vector) represents the association of each node with the cluster, and ***A*** be the binary similarity matrix. A proper objective function for clustering can be [45]:

$$\mathbf{f(x) = x^T A\ x} \tag{35}$$

This notation is a sum of the terms of the form [45]:

$$\begin{aligned} &\{\text{association of element i with cluster}\} \times \\ &\quad \{\text{similarity between i and j}\} \times \\ &\{\text{association of element j with cluster}\} \end{aligned} \tag{36}$$

A sub-graph with elements maximizing this objective function gives us a cluster. However, the objective function without normalization cannot be used, since scaling the associations of the nodes to the cluster changes the results.

Therefore, we can use the normalization $\mathbf{x}^T\mathbf{x} = \mathbf{1}$ as a constraint and obtain the Lagrangian [45]:

$$\begin{aligned} &\textit{maximize} && \mathbf{x}^T\mathbf{A}\,\mathbf{x} \\ &\textit{subject to} && \mathbf{x}^T\mathbf{x} = \mathbf{1} \end{aligned} \tag{37}$$

In [48], Motzkin-Straus proved that a subset of vertices S is a maximum clique of graph G, if and only if its *unweighted characteristic vector* $\mathbf{x}$ is a global maximizer of the Motzkin-Straus quadratic program defined in (37). Motzkin-Straus theorem is an important result from graph theory and has been applied to various computer vision and pattern recognition problems [47].

Unfortunately, while the other graph theoretic clustering algorithms (e.g. minimum spanning tree, minimum cut, etc.) work with the weighted similarity graph; the concept of *maximum clique* is defined on unweighted graphs. The general approach is not different than the one that is explained in the previous section. The unweighted similarity matrix could be derived from the weighted similarity matrix with any thresholding operation. If the similarity between two nodes is above a certain threshold it is assumed that two nodes are similar and the corresponding entry in the binary unweighted similarity graph is set to “1”. Otherwise it is set to “0” and the edge between these two nodes is removed from the graph [40].

Although Motzkin-Straus theorem is a remarkably important contribution to the maximum clique problem, working with an unweighted similarity matrix is not acceptable for most of the computer vision problems. The unweighted similarity matrix and unweighted characteristic vector do not provide any discriminative information about the participation of the nodes to the cluster. If a node is associated with the cluster with the value 0.4 and the other is with 0.9, and if the selected threshold is 0.3, both nodes are associated with this cluster. There is no clue about the degree of the association (the node with association value of 0.9 is obviously more strongly connected to the cluster). Therefore, unweighted similarities can be used to generate a hierarchy of clusters that can be presented to the user, but it is not useable and feasible in most of the

pattern recognition and computer vision problems due to the large number of data to be clustered.

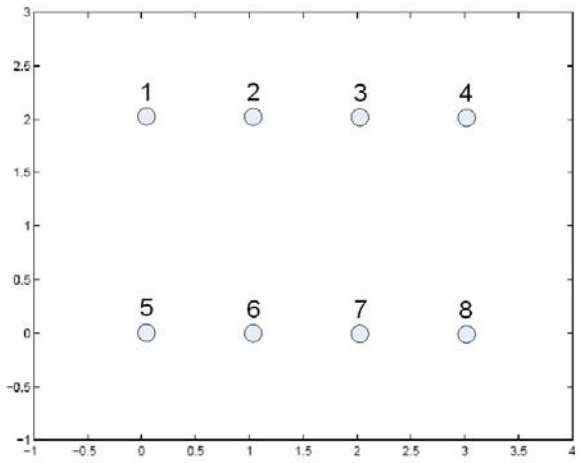

Figure 4-5 Clustering Problem due to Unweighted Similarities

For example, let's assume that we have eight data points in the 2D space with a feature vector consisting of their position in the *x* and *y* coordinates (Figure 4-5). The similarity measure is the distance between their positions in the space. It is obvious that the nodes 1, 2, 3 and 4 form a cluster, and the nodes 5, 6, 7 and 8 form another one. However, using the unweighted similarities, it is not possible to find a threshold value that may be used to separate the nodes 1, 2, 3 and 4 from the nodes 5, 6, 7, and 8. The algorithm will check the similarity (the distance) between node 1 and node 3 and will observe that it is the same as the similarity between the node 1 and node 5. As a result, by any kind of thresholding, it is not possible for the algorithm to segment these three nodes in different clusters. In another way, it is not possible to find a threshold value such that the resulting unweighted graph contains exactly two strictly maximal cliques corresponding to the two clusters in Figure 4-5.

Therefore, it is important to generalize the notion of the maximum clique to edge weighted graphs, thereby allowing the development of a new partitional

(not hierarchical) clustering approach. This is exactly what Pavan and Pelillo propose a new method for pair-wise clustering based on the novel cluster concept *dominant sets* [49].

4.3. A Novel Graph Theoretic Definition of a Cluster: Dominant Sets

Pavan and Pelillo has followed the traditional graph theoretic approaches and represented the data to be clustered as an undirected weighted graph with no self loop [49]. The graph is represented with the weighted similarity matrix $\mathrm{A} = (a_{ij})$, where a_{ij} denotes the similarity between node-*i* and node-*j*. Due to the fact that the graph does not contain any self loops, diagonal entries in the corresponding similarity matrix are set to zero.

One should recall from the previous sections that a good cluster has two important features: internal similarity within a cluster should be high and the similarities between an element within the cluster and the one from outside should be low. In the graph domain, this corresponds to large edge weights within a cluster and low weights on the edges connecting the cluster nodes to the external ones.

Pavan and Pelillo [49] starts from the analysis of the intuitive idea that assignment of weights to the edges, is one way of assigning weights to the nodes of a graph. For example, considering the graph *G* in Figure 4-6, the edges incident on node-1 has the similarity values 3 and 4, which are the lowest weights according to the weights on the graph. Similarly, edges incident on node-3 has the similarity values 4 and 5, which are the top weights in the overall ranking. This example may intuitively tell us about the natural *ranking* of participation of the nodes in the cluster, i.e. the weight of node-3 is larger than the weight of node-1. By weight of a node, the *degree* of the participation is being implied [49].

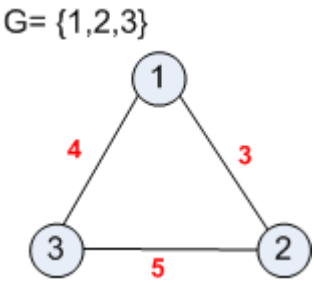

Figure 4-6 Sample Weighted Graph

In order to analyze this intuitive idea of node weights, some preliminary definitions should be stated. The *average weighted degree* of *i* with respect to a non-empty sub-graph *S* is defined as follows, where a_{ij} denotes the similarity between node-*i* and node-*j* [49]:

$$aw\deg_S(i) = \frac{1}{|S|}\sum_{j\in S} a_{ij} \tag{38}$$

As demonstrated in Figure 4-7, average weighted degree is a relative weight calculation with respect to a sub-graph and gives us an idea about the degree of participation of a node to the given sub-graph [49].

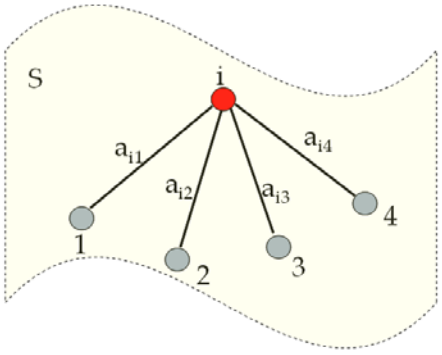

Figure 4-7 Average Weighted Degree Calculation

For *j* is not an element of *S*, one can define the following similarity measure between nodes *i* and *j* [49]:

$$\phi_S(i, j) = a_{ij} - aw\deg_S(i) \tag{39}$$

Intuitively, $\phi_S(i, j)$ measures the similarity between nodes *j* and *i* with respect to the average similarity between node-*i* and its neighbors in *S* [49]. Note that $\phi_S(i, j)$ can be either positive or negative.

One can interpret this definition by the help of Figure 4-8. Let's assume that we are considering enlarging the cluster (i.e. sub-graph S) by including node *j* to this cluster. If $\phi_S(i, j)$ is positive, this means that the similarity (or connectivity) between the nodes *i* and *j* are stronger than the average similarity of node *i* to the other nodes of the cluster. This derivation intuitively increases the probability of including the node-*j* to the cluster, since its connectivity to the node-*i* is stronger than the others. Therefore, if node-*i* is in this cluster, node-*j* should be in this cluster as well.

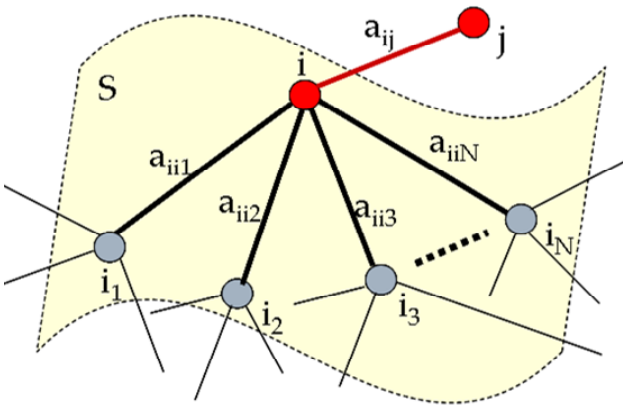

Figure 4-8 Similarity wrt. Average Similarity

Using these definitions, one can define the *node weights*, $w_S(i)$. Let *S* be a nonempty sub-graph and $i \in S$. The weight of *i* with respect to *S* is:

$$w_S(i) = \begin{cases} 1, & if\ |S| = 1 \\ \sum_{j \in S \setminus \{i\}} \phi_{S \setminus \{i\}}(j,i) w_{S \setminus \{i\}}(j), & otherwise \end{cases} \quad (40)$$

Moreover, the total weight of *S* is defined to be:

$$W(S) = \sum_{i \in S} w_S(i) \quad (41)$$

Intuitively, $w_S(i)$ gives us a measure of the overall similarity between node-*i* and the other nodes in $S \setminus \{i\}$ with respect to the overall similarity among the nodes in $S \setminus \{i\}$. Let's interpret this definition as follows: In order to calculate the weight of node-*i*, one should remove the node from the cluster (i.e. from sub-graph S), and for the rest of the cluster, we compare the connectivity of node-*i* to each node-*j*'s average connectivity ($j \in S \setminus \{i\}$) to the cluster. Moreover, we multiply this comparison with the weight of node-*j* (i.e. its degree of association to cluster), because even if the relative connectivity of node-*i* may be stronger than the average connectivity of node-*j*, the degree of association of node-*j* to the cluster may be low. Therefore, the weight of node-*j* must be a weighting factor for this comparison.

Referring back to Figure 4-6, the weights calculated using the definition provided above are: $w_{\{1,2,3\}}(1) = 10$, $w_{\{1,2,3\}}(2) = 16$, and $w_{\{1,2,3\}}(3) = 18$. These results are in accordance with the intuitive idea that was presented in the beginning of this section.

We are now in a position to define dominant set concept:

Definition 1: A nonempty subset *S* of a graph *G* is said to be *dominant* if [49]:

$$\begin{aligned} &1.\quad w_S(i) > 0, \qquad \textit{for all } i \in S, \\ &2.\quad w_{S\cup\{i\}}(i) < 0, \qquad \textit{for all } i \notin S, \end{aligned} \tag{42}$$

One should note that the properties of a *dominant set* as stated in (42) implicitly correspond to the two main properties of a good cluster, which are high internal connectivity and low connectivity between the cluster and the external nodes. This fact is the main reason behind considering the dominant sets as a new definition for a cluster of nodes [49].

Definition 2: *Weighted characteristic vector* $\boldsymbol{x^S}$ *for a non-empty subset S is* [49]:

$$x_i^S = \begin{cases} \dfrac{w_S(i)}{W(S)}, & \textit{if } i \in S \\ 0 \quad , & \textit{otw} \end{cases} \tag{43}$$

Weighted characteristic vector calculated in this way indicates a subset. If there are *n* data nodes in the graph, the vector is *n*-dimensional. Each element in the vector indicates whether the node corresponding to the element index is included in the cluster (i.e. the sub-graph) or not. Moreover, the value of the vector element also provides information about the degree of association to the cluster for that node. If the value is zero the node is not included in the cluster. Remember that the values in the vector are normalized so that all the elements in the vector sum up to 1.

For example, for a graph with 5 nodes, the following characteristic vector is given:

$$x^S = \begin{bmatrix} 0.3 \\ 0.4 \\ 0 \\ 0 \\ 0.3 \end{bmatrix} \tag{44}$$

This weighted characteristic vector indicates to a subset *S* (a sub-graph) which is identified by the non-zero entries of the vector. Hence, the subset *S* is equal to {1, 2, 5}. Moreover, the weighted characteristic vector tell us that while the nodes 1 and 5 are connected to this cluster with a degree of 0.3, the participation of node 2 to this cluster is 0.4 and stronger than both node 1 and node 5.

Pavan and Pelillo generalize the Motzkin-Straus Theorem to the edge-weighted graphs by using the dominant set definition [49]:

Theorem: *If S is a dominant subset of graph G, then its weighted characteristic vector* $\mathbf{x}^S$ *is a strict local solution of Motzkin-Straus quadratic program with weighted similarity matrix. Conversely, if* $\mathbf{x}^*$ *is a strict local solution of Motzkin-Straus quadratic program with weighted similarity matrix, then the subset S indicated by* $\mathbf{x}^*$ *is a dominant subset of G* [49].

Following this theorem, Pavan provides a lengthy proof [40] in which the definition of dominant set is proven to be equivalent to that of a strictly maximal clique (i.e. maximum clique) when applied to unweighted graphs.

4.4. Finding Dominant Sets by Replicator Dynamics

The characteristic vector $\mathbf{x}^S$ indicates a subset *S*, but it does not provide us with the information whether the subset *S* is a dominant set, thereby a cluster, or not. Therefore, in order to find the dominant sets of a graph, we need to solve the weighted Motzkin-Straus quadratic program with weighted similarity matrix and weighted characteristic vector.

Solution to this quadratic program can be determined by using the *replicator dynamics* from evolutionary game theory [50]. Pavan uses the following model, which corresponds to the discrete-time version of the first order replicator equations:

$$x_i(t+1) = x_i(t)\frac{(\mathbf{Ax})_i}{\mathbf{x}(t)^T\mathbf{Ax}(t)} \tag{45}$$

where x_i is the i[th] element of the weighted characteristic vector **x,** and **A** is the weighted similarity matrix. The solution is iteratively found based on this model. The iterations start from an initial point and stops when the new iterations do not update the vector.

At the starting point, since we do not know which nodes are in a cluster, it is better to start with a weighted characteristic vector which has equal association values for each data node. For example, if the graph has 5 data nodes to be clustered, the starting weighted characteristic vector **x(t=0)** would be:

$$x^S = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \end{bmatrix} \tag{46}$$

The characteristic vector may also be called as the *state vector* [40]. The initial state vector given above will be updated according to the first order replicator equations at each iteration. When there are no updates in the *state vector* the algorithm stops. The entries, which have significant large values in the final state vector, represent the dominant set.

Let's assume that we have 6 points in the 2D space to be clustered (Figure 4-9**Error! Reference source not found.**). The points in the upper left corner represents the nodes 1, 2, and 3 of the graph, and the points in the bottom right corner represents the nodes 4, 5 and 6 of graph. The similarity metric is the distance between the points in space. The starting state vector **x(t=0)** would be:

$$x^S_{t=0} = \begin{bmatrix} 1/6 \\ 1/6 \\ 1/6 \\ 1/6 \\ 1/6 \\ 1/6 \end{bmatrix} \tag{47}$$

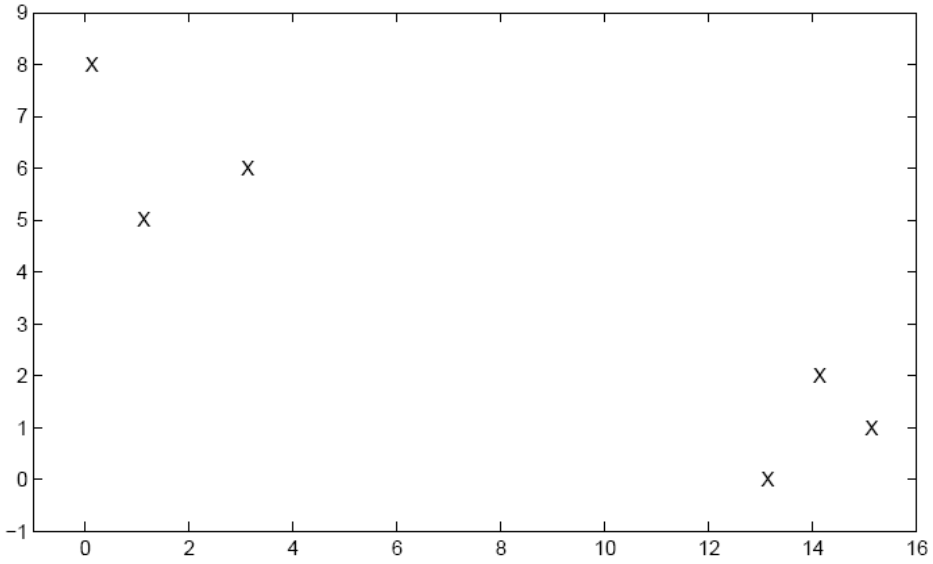

Figure 4-9 Simple Data Set (6 points) [40]

Figure 4-10 shows the evolution of the state vector using the first order discrete time replicator model. The x coordinates in the figure show the index values of the state vector corresponding to the data items, and the y coordinate indicates the values of the state vector entries. Note that the values for the nodes 1, 2 and 3 get smaller and smaller, while the values for the nodes 4, 5 and 6 get larger. Finally the state vector values do not change between iteration 4 and iteration 5, and the algorithm stops. Therefore, final state presents us with the dominant set consisting of nodes 4, 5 and 6.

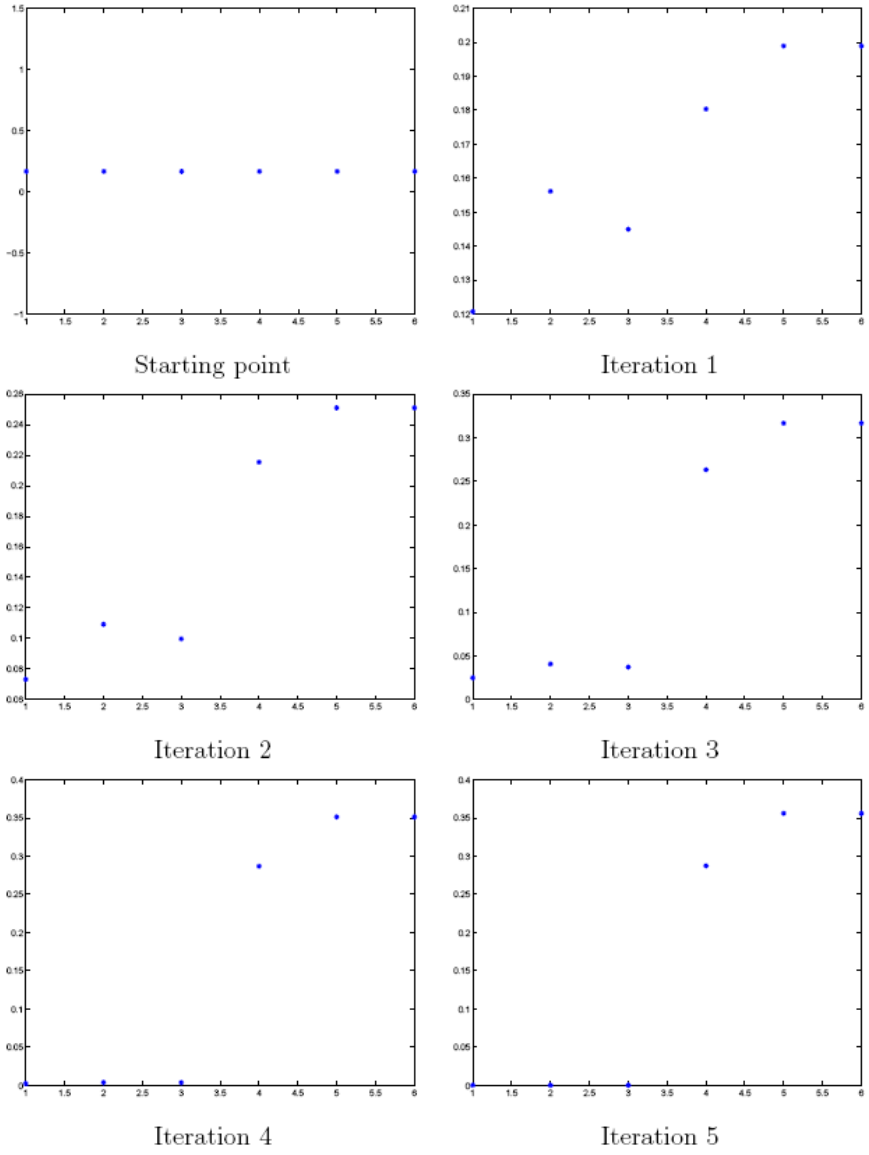

Figure 4-10 Evolution of the state vector x(t) [40]

4.5. SBD Algorithm Based on Dominant Sets

Graph theoretic segmentation algorithms are getting more popular in the field of pattern recognition and computer vision [51]. *Dominant sets*, which is a very novel concept, is a remarkably important contribution to the graph theory domain and has already found applications in image segmentation [49].

Furthermore, Sakarya and Telatar applied this novel concept to video scene detection problem [52].

In this thesis, we propose a SBD algorithm for detecting abrupt changes based on dominant sets concept. In designing the algorithm, we have used the experience that we have acquired during the simulations of the state-of-the-art SBD algorithms in Chapter 3.

First of all, during the experiments, we have observed that graph theoretic approaches are computationally complex due to weight matrix calculations and contextual information usage. Therefore, we need to design a relatively fast algorithm, which compensates the time one looses in graph related calculations. Although motion-based algorithm presented a very good recall performance, since motion vector calculation is a complex process we decided not to use motion information together with a graph theoretic approach.

Secondly, we have learnt that the algorithms, which use the computationally complex features, such as edges, do not outperform the performance of the simpler algorithms, which uses pixel or histogram information. These two observations together led us utilization of the histogram and pixel-wise difference as the similarity metric.

In order to further increase the speed of the algorithm, we decide to use the DC images and the preprocessing step proposed in [19].

During the experiments, we have noted that one of the main reasons for false alarms for all kinds of algorithms is due to video-in-video type of effects or the sliding text at the bottom of the video. In addition to this, considering that the center of the video includes the more important content, we have adopted a Region-of–Interest (RoI) idea proposed in [30] and decided to perform the calculations in this region. Figure 4-11 shows a typical RoI for a sample frame.

Figure 4-11 Region of Interest (RoI)

The proposed algorithm can be summarized as follows:

1. Find the cut candidates based on the sum of absolute differences of the pixel intensity values over the RoI. If the difference is below a certain threshold skip the rest of the algorithm. If the difference is above a certain threshold continue with the next step.

2. For the cut candidates, construct the graph representation by taking 4 frames before the shot boundary and taking 2 frames after the shot boundary. For the sample candidate cut position shown in Figure 4-12, graph representation is shown in Figure 4-13. Each node in the graph represents the frames in the sequence.

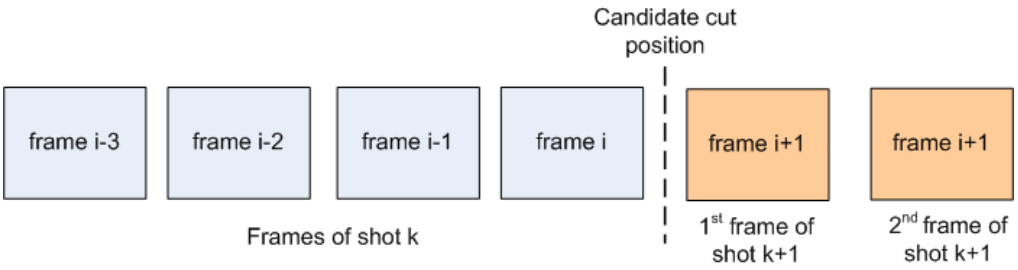

Figure 4-12 Candidate Cut Position

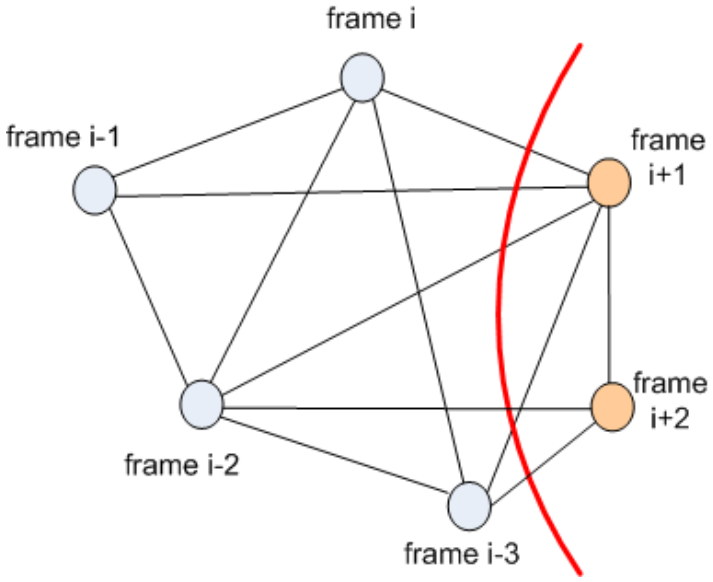

Figure 4-13 Graph Representation of the Candidate Cut position

3. Select features for video content representation and construct the weighted similarity matrix **A** based on the selected feature(s).

4. Start with an initial state vector of $x_{t=0}^{S} = [1/6, 1/6, 1/6, 1/6, 1/6, 1/6]$, and find the dominant set according to the first order replicator equation in (46). Iteratively update the state vector until there is no update or the iteration number reaches the maximum number of iterations (selected as 15).

5. For the true cut positions, one expects the resulting dominant set to include first 4 frames. Therefore, we want the values of first four elements of the state vector to be above the starting value (i.e. 1/6) and we want the 5th and 6th elements of the state vector to be less than a threshold (selected as 0.14).

4.6. Simulation Results

During the experiments, we have started with the 1D feature vectors. We have tested the algorithm with pixel difference, histogram difference and motion

difference separately. Finally, we have tested the algorithm with the 2D feature vector including both pixel and histogram difference. Table 4-1 presents the results of this experiment. It is obvious that using a 2D feature produces better results for both recall and precision values compared to using 1D feature. Results for motion based algorithm are slightly better than the one using 2D feature vector. However, during the experiments we have observed that using motion difference as a feature results in a very slow algorithm.

During the experiments, we have also tested the algorithm with an alternative structure in which we have used 4 frames from one shot and only 1 frame from the next shot (Figure 4-14). Simulation results for both cases are presented in Table 4-2.

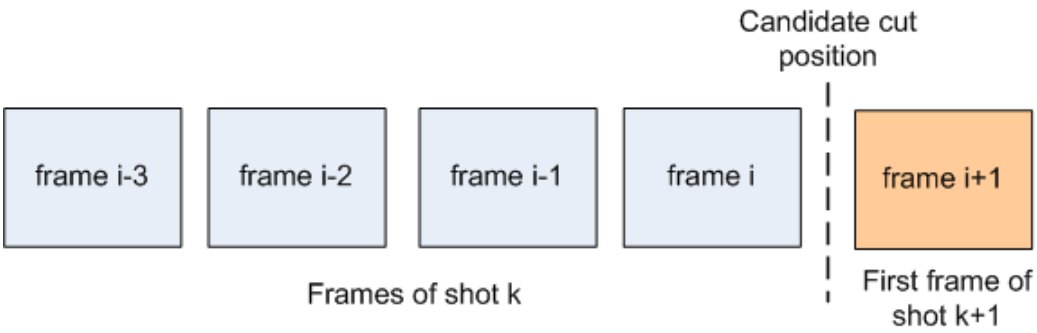

Figure 4-14 Alternative Test Structure

Adopting the 1D motion feature as a similarity measure for graph representation, overall algorithm is tested with the TRECVID 2003 SBD test videos. Table 4-3 presents the simulation results of the SBD algorithm based on dominant set concept together with all the algorithms tested in Chapter 3. Results show that the algorithm produces quite good recall results. Precision results need to be improved.

Experiments showed that the types of shot boundaries that this algorithm produces wrong results are not different than the pixel-wise difference and histogram difference algorithms. Main situations that result in wrong decisions are video in video effects, dark scenes and large object movements.

Table 4-1 Dominant Sets Results for for Single and Joint Features

#	Video ID	Dominant Sets											
		Pixel-wise Feature			Histogram Feature			Motion Feature			Histogram & Pixel Feature		
		R	P	F1	R	P	F1	R	P	F1	R	P	F1
1	19980203_CNN	0,80	0,81	0,80	0,78	0,54	0,64	0,87	0,81	0,84	0,87	0,72	0,79
2	19980222_CNN	0,92	0,87	0,89	0,87	0,64	0,74	0,96	0,84	0,90	0,95	0,83	0,89
3	19980224_ABC	0,87	0,84	0,85	0,88	0,69	0,77	0,95	0,84	0,89	0,93	0,82	0,87
4	19980412_ABC	0,94	0,85	0,89	0,88	0,65	0,75	0,99	0,81	0,89	0,97	0,79	0,87
5	19980425_ABC	0,87	0,82	0,84	0,85	0,53	0,65	0,94	0,72	0,82	0,94	0,71	0,81
6	19980515_CNN	0,86	0,75	0,80	0,81	0,60	0,69	0,91	0,72	0,80	0,90	0,73	0,81
7	19980531_CNN	0,91	0,85	0,88	0,89	0,68	0,77	0,97	0,78	0,86	0,97	0,77	0,86
8	19980619_ABC	0,96	0,90	0,93	0,90	0,76	0,82	0,99	0,87	0,93	0,99	0,83	0,90
Avg	**All**	**0,89**	**0,84**	**0,86**	**0,86**	**0,64**	**0,73**	**0,95**	**0,80**	**0,87**	**0,94**	**0,78**	**0,85**

Table 4-2 Results for 4+1 Structure vs. 4+2 Structure

#	Video ID	Histogram & Pixel Feature					
		4+1 Structure			4+2 Structure		
		R	P	F1	R	P	F1
1	19980203_CNN	0,85	0,73	0,79	0,87	0,72	0,79
2	19980222_CNN	0,94	0,80	0,86	0,95	0,83	0,89
3	19980224_ABC	0,92	0,80	0,86	0,93	0,82	0,87
4	19980412_ABC	0,96	0,80	0,87	0,97	0,79	0,87
5	19980425_ABC	0,93	0,56	0,70	0,94	0,71	0,81
6	19980515_CNN	0,90	0,70	0,79	0,90	0,73	0,81
7	19980531_CNN	0,95	0,78	0,86	0,97	0,77	0,86
8	19980619_ABC	0,98	0,86	0,92	0,99	0,83	0,90
Avg	**All**	**0,93**	**0,75**	**0,83**	**0,94**	**0,78**	**0,85**

Table 4-3 CUT Detection Results for Dominant Sets with Motion as the Similarity Metric

#	Pixel-wise (Adaptive Thresh.)			Histogram Difference			ECR			Petersohn			Graph Partition with SVM			Motion with Dual Cross Search			Dominant Sets		
	R	P	F1	R	P	F1	R	P	F1	R	P	F1	R	P	F1	R	P	F1	R	P	F1
1	0.74	0.82	0.83	0.83	0.83	0.83	0.82	0.81	0.81	0.91	0.86	0.88	0.67	0.89	0.76	0.94	0.73	0.82	0,87	0,81	0,84
2	0.79	0.9	0.86	0.94	0.9	0.92	0.91	0.88	0.89	0.95	0.87	0.91	0.91	0.93	0.92	0.98	0.80	0.88	0,96	0,84	0,90
3	0.89	0.88	0.92	0.96	0.85	0.9	0.95	0.79	0.86	0.97	0.85	0.91	0.9	0.92	0.91	0.96	0.80	0.87	0,95	0,84	0,89
4	0.9	0.92	0.92	0.95	0.86	0.9	0.94	0.89	0.91	0.99	0.83	0.9	0.94	0.94	0.94	0.99	0.76	0.86	0,99	0,81	0,89
5	0.85	0.81	0.88	0.91	0.84	0.87	0.89	0.84	0.86	0.95	0.85	0.9	0.89	0.87	0.88	0.97	0.68	0.80	0,94	0,72	0,82
6	0.78	0.84	0.85	0.88	0.86	0.87	0.84	0.86	0.85	0.91	0.85	0.88	0.76	0.92	0.83	0.92	0.73	0.81	0,91	0,72	0,80
7	0.9	0.89	0.92	0.95	0.89	0.92	0.91	0.84	0.87	0.97	0.84	0.9	0.92	0.94	0.93	0.98	0.75	0.85	0,97	0,78	0,86
8	0.81	0.87	0.95	0.94	0.86	0.9	0.94	0.85	0.89	0.98	0.93	0.95	0.88	0.93	0.90	0.99	0.85	0.91	0,99	0,87	0,93
Avg	**0.83**	**0.87**	**0.85**	**0.92**	**0.86**	**0.89**	**0.90**	**0.85**	**0.87**	**0.95**	**0.86**	**0.90**	**0.86**	**0.92**	**0.89**	**0.97**	**0.76**	**0.85**	**0,95**	**0,80**	**0,87**

During the experiments, we have also incorporated this algorithm with the aforementioned motion based algorithm as a post processing step. One should recall that the motion based algorithm produced quite good recall results during the tests in Chapter 3. However, the precision values obtained were not as well as recall results. Therefore, we have improved the algorithm by using the algorithm based on the dominant set concept as a post processing step on the results obtained by motion based algorithm. The proposed algorithm double checks the cuts detected by the motion based algorithm by using the dominant sets algorithm with 2D feature vector as the similarity measure. The results are summarized in Table 4-4. Results show that although the recall values decrease a little, precision values increase significantly by applying dominant sets as a post processing step.

Table 4-4 Motion Based Algorithm improved by DS Algorithm

#	Video ID	Motion with Dual Cross Search			Motion with DS		
		R	P	F1	R	P	F1
1	19980203_CNN	0,94	0,73	0,82	0,87	0,82	0,84
2	19980222_CNN	0,98	0,80	0,88	0,96	0,88	0,92
3	19980224_ABC	0,96	0,80	0,87	0,94	0,88	0,91
4	19980412_ABC	0,99	0,76	0,86	0,98	0,85	0,91
5	19980425_ABC	0,97	0,68	0,80	0,95	0,80	0,87
6	19980515_CNN	0,92	0,73	0,81	0,91	0,82	0,86
7	19980531_CNN	0,98	0,75	0,85	0,95	0,83	0,89
8	19980619_ABC	0,99	0,85	0,91	0,99	0,89	0,94
Avg	**All**	**0,97**	**0,76**	**0,85**	**0,94**	**0,85**	**0,89**

4.7. Conclusion

The resulting precision values for the proposed algorithm are not higher than some of the aforementioned SBD algorithms. The main reason for this result is due to motion and disturbances. If the visual similarity between 6 frames from the same shot is somehow not at the same degree, the algorithm chooses the frames with the strongest connectivity during the iterations of the state vector.

Consequently, an abrupt change in the state vector values appears between the 4^{th} and the 5^{th} elements. Therefore, the algorithm produces false alarms.

This is one of the consequences of the idea that maximal/maximum cliques are the strictest definition of a cluster. The algorithms based on the maximal/maximum clique definition search for the strongest cluster in a graph. Therefore, if due to disturbances or significant motion, the 5^{th} and 6^{th} frames are slightly different than the other four, the algorithm leaves the 5^{th} and 6^{th} frames out of the dominant set.

Another observation is that the algorithm is not robust against flashlights, as expected. This result is due to the inclusion of only 2 frames from the next shot. Therefore, the algorithm does not have enough information to check whether the frame next to flashlight is similar to the previous frames or not. We have considered taking more frames from the next shot; however, we have observed that such an approach, unfortunately, decreases the recall results significantly. In order to make use of the concept of dominant sets, we need to allocate as much weight as possible to the dominant set. At the final step, we identify the dominant set boundary by using a constant threshold. Therefore, in order to keep the separation easily detectable by a threshold, we need to keep first 4 frames from the earlier shot and only a single frame or at most 2 frames from the next shot. Otherwise, iterations result in similar values in the state vector and we miss significant amount of shot boundaries.

We have observed that if we adopt 2D feature with pixel and histogram differences as the similarity metric, the algorithm is faster than the graph partition algorithm with SVM. This result proves that graph theoretic approaches can be used in real-time SBD applications with proper enhancements.

On the other hand, algorithm with motion difference as the similarity metric is the slowest one among the algorithms tested during this thesis. This is an expected result since the graph based algorithms are computationally expensive algorithms due to similarity matrix calculations. For the algorithm that we have proposed for dominant sets, for each cut candidate, a 6x6 similarity

matrix is calculated. Although the similarity matrix is symmetric and the elements on the diagonal are set to 0 (no self loops in the graph), one still needs to run the motion based algorithm for 15 cells in the similarity matrix. This requirement makes the algorithm 15 times slower than the aforementioned motion based algorithm. This observation supports the idea that motion based features together with the graph theoretic SBD algorithms requires significantly high computational power.

During the experiments we have observed that dominant sets algorithm can be used to improve the precision performance of the motion based algorithms. Since both algorithms are computationally expensive, instead of 1D motion feature, one should adopt 2D feature vector with pixel and histogram differences as a similarity measure, while using dominant sets algorithm with a motion based algorithm.

CHAPTER 5

CONCLUSION & FUTURE WORK

5.1. Conclusions

In this thesis, we have implemented 6 of the well known state-of-the-art algorithms from the uncompressed domain. The algorithms are tested against TRECVID 2003 test video and compared with the recall, precision and F1 measures.

Pixel based algorithm is the simplest tested SBD method. It is very sensitive to video content. Therefore, comparing the continuity signal with a constant threshold did not produce good results. Following this observation, we have reduced the effects of scenes containing a lot of movement by comparing the difference signal with a threshold derived from the maximum and minimum difference signals over a small aperture. Simulation results showed that, although the pixel based method is known to be the most primitive one in the literature, it is possible to obtain very good results with such an adaptive thresholding scheme.

Color histograms are the features that are mostly used in SBD task. Experiments have shown that histogram-based algorithms are very appropriate for SBD. They provide global information about the video content and are less sensitive to local changes. We have also observed that algorithms can be further improved by quantization in the color spaces. The resulting algorithms are observed to be faster without any performance degradations.

We have also observed that the algorithms adopting complex features, such as edges, do not outperform the performance of the simpler algorithms that utilize pixel or histogram information. In addition to this fact, complex features increase the requirement for more computational power.

Examining Petersohn's SBD system showed us that, by processing down-sampled images, it is possible to both adopt feature vectors with more than one feature and still has a faster shot boundary detection system. Furthermore, we have observed that DC images include global information about the video content, therefore more robust against motion and disturbances.

Investigating the error set of the algorithms showed us that motion is the major factor that makes the shot boundary detection harder. Motion-based algorithm with the highest recall results proved that a good motion based SBD algorithm can produce quite good detection performance.

Although the algorithm based on graph partition model with machine learning is computationally complex, the observed high precision results show that they are quite promising algorithms with very robust results. They especially perform best for flashlight detection and are also less sensitive to disturbances.

Gradual transition detection is a difficult job. There are various types of editing effects and each editing effect has a different temporal pattern, which makes it very difficult to detect with a single algorithm. Due to similar reasons, in 2007, TRECVID organization decreased the amount of gradual transitions in the test videos significantly and completely concentrated on cut detection.

Although different algorithms have different strong and weak characteristics, we have observed that all algorithms have difficulty in detecting the shot boundaries under specific circumstances such as dark video content, rapid zooming, video in video effects and large motion activity.

We have also studied the *dominant sets* concept from graph theory and proposed a novel shot boundary detection algorithm based on dominant sets. The results are promising and showed us that with proper improvement dominant sets can be used in the shot boundary detection area.

5.2. Future Work

Recent research approaches the shot boundary problem as a pattern recognition problem. As a result machine learning algorithms are being adopted

at the decision level. In TRECVID 2007, five out of the fifteen proposed SBD algorithms uses SVM as the classifier. Therefore, as a future work, instead of finding the dominant set from the final state vector by a simple thresholding scheme, SVM can be used.

REFERENCES

[1] A. Hanjalic, “Shot Boundary Detection: Unraveled and Resolved?,” *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 12, no. 2, pp. 90-105, February 2002.

[2] A. Hanjalic, *Content-based Analysis of Digital Video*, Boston: Kluwer Academic Publishers, 2004.

[3] A. F. Smeaton, W. Kraaij, and P. Over, "TRECVID 2007 - Overview," 2008, http://www-nlpir.nist.gov/projects/tvpubs/tv.pubs.org.html, visited on 10 May 2008.

[4] J. Yuan, H. Wang, L. Xiao *et al.*, “A Formal Study of Shot Boundary Detection,” *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 168-186, 2007.

[5] H. Koumaras, and G. Gardikis, “Shot Boundary Detection without Threshold Parameters,” *J. Electron. Imaging,* vol. 15, no. 020503, 2006.

[6] N. Vasconcelos, and A. Lippman, “Statistical models of video structure for content analysis and characterization,” *IEEE Transactions on Image Processing,* vol. 9, no. 1, pp. 3-19, 2000.

[7] M. Albanese, and V. Moscato, “A Formal Model for Video Shot Segmentation and its Application via Animate Vision,” *Multimedia Tools and Applications* vol. 24, no. 3, pp. 253-272, December 2004.

[8] T. Kikukawa, and S. Kawafuchi, “Development of an automatic summary editing system for the audio-visual resources,” *Transactions on Electronics and Information J75-A* pp. 204-212, 1992.

[9] K. Otsuji, Y. Tonomura, and Y. Ohba, “Video browsing using brightness data,” *in Proc. SPIE VCIP'91,* vol. 1606, pp. 980-989, 1991.

[10] I. Koprinska, and S. Carrato, "Temporal Video Segmentation: A Survey," *Signal Processing: Image Communication,* 16, 2001.

[11] B. Shahraray, “Scene change detection and content-based sampling of video sequences,” *Proc. IS&T/SPIE 2419*, pp. 2-13, 1995.

[12] H. J. Zhang, A. Kankanhalli, and S. Smoliar, “Automatic Partitioning of Full-Motion Video,” *Multimedia Systems,* vol. 1, pp. 10-28, 1993.

[13] J. Mas, and G. Fernandez, "Video Shot Boundary Detection based on Color Histogram," *Notebook Papers TRECVID2003*, 2003.

[14] J. Yuan, B. Zhang, and F. Lin, “Graph partition model for robust temporal data segmentation,” *Proc. of PAKDD*, pp. 539-542, 2005.

[15] R. Zabih, J. Miller, and K. Mai, “A Feature-Based Algorithm for Detecting and Classifying Scene Breaks,” *Proc. ACM Multimedia 95*, pp. 189-200, 1995.

[16] R. Lienhart, “Comparison of Automatic Shot Boundary Detection Algorithms,” *Image and Video Processing VII*, 1999.

[17] B. C. Song, and J. B. Ra, “Automatic Shot Change Detection Algorithm Using Multi-stage Clustering for MPEG-Compressed Videos,” *Journal of Visual Communication and Image Representation,* vol. 12, no. 3, pp. 364-385, 2002.

[18] C. Petersohn, "Fraunhofer HHI at Trecvid 2004: Shot Boundary Detection System," *TREC Video Retrieval Evaluation Online Proceedings*, 2004.

[19] Y. Kawai, H. Sumiyosi, and N. Yagi, "Shot Boundary Detection at TRECVID 2007," 2007.

[20] B. L. Yeo, and B. Liu, “Rapid Scene Analysis on Compressed Videos Dec,” *IEEE Trans. Circuits and Systems for Video Technology,* vol. 5, pp. 533-544, December 1995.

[21] B. T. Truong, “Shot Transition Detection and Genre Identification for Video Indexing and Retrieval,” Honours, School of Computing, Curtin University of Technology, 1999.

[22] W. A. C. Fernando, C. N. Canagarajah, and D. R. Bull, “A Unified Approach To Scene Change Detection In Uncompressed And Compressed Video,” *IEEE Transactions on Consumer Electronics,* vol. 46, no. 3, pp. 769-779, 2000.

[23] F. Arman, A. Hsu, and M. Y. Chiu, “Image processing on compressed data for large video databases,” *Proc. ACM Multimedia*, pp. 267-272, 1993.

[24] E. Deardorff, T. Little, J. Marshall *et al.*, “Video scene decomposition with the motion picture parser,” *IS&T/SPIE,* vol. 2187, pp. 44-45, 1994.

[25] I. B. Kayaalp, “Video Segmentation Using Partially Decoded MPEG Bitstream,” METU, Ankara, 2003.

[26] J. Easterbrook. "A simple cut detector for video sequences," http://sourceforge.net/, visited on 24 August 2007.

[27] A. Hanjalic, and R. L. Lagendijk, “Automated high-level movie segmentation for advanced video-retrieval systems,” *IEEE Transaction*

on Circuits and Systems for Video Technology, vol. 9, no. 4, pp. 580-588, 1999.

[28] Z. Rasheed, and M. Shah, “A Graph Theoretic Approach for Scene Detection in Produced Videos,” *Multimedia Information Retrieval Workshop*, 2003.

[29] C. Ngo, “A robust dissolve detector by support vector machine,” *Proc. ACM Multimedia*, pp. 283-286, 2003.

[30] Z. Liu, E. Zavesky, D. Gibbon *et al.*, “AT&T Research at 2007,” 2008.

[31] S. Wei, Y. Zhao, Z. Zhu *et al.*, "BJTU TRECVID 2007 Video Search," Institute of Information Science, Beijing Jiaotong University, 2008.

[32] J. S. Boreczky, and L. A. Rowe, “Comparison of Video Shot Boundary Detection Techniques,” *Storage and Retrieval for Still Image and Video Databases IV, Proc. SPIE 2664*, pp. 170-179, 1996.

[33] E. Esen, and A. Alatan, "The COST292 experimental framework for TRECVID 2007," http://www-nlpir.nist.gov/projects/tvpubs/tv.pubs.org.html, visited on 11 June 2008.

[34] C. Ding, X. He, H. Zha *et al.*, “A min-max cut algorithm for graph partitioning and data clustering,” *Proc. IEEE International Conference on Data Mining*, 2001.

[35] J. Shi, and J. Malik, “Normalized cuts and image segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[36] C.-K. Cheng, and Y. A. Wei, “An improved two-way partitioning algorithm with stable performance,” *IEEE Trans. on Computer Aided Design,* vol. 10, pp. 1502-1511, 1991.

[37] L. Hagen, and A. B. Kahng, “New spectral methods for ratio cut partitioning and clustering,” *IEEE Trans. on Computer Aided Design,* vol. 11, pp. 1074-1085, 1992.

[38] C.-C. C. C.-J. Lin, "LIBSVM: a library for support vector machines," http://www.csie.ntu.edu.tw/~cjlin/libsvm, 2001, visited on 13 June 2007.

[39] X. Banh, and Y. Tan, “Adaptive Dual-Cross Search Algorithm for Block-Matching Motion Estimation,” *IEEE Trans. Consumer Electronics,* vol. 50, no. 2, pp. 766-775, 2004.

[40] M. Pavan, “A New Graph-Theoretic Approach to Clustering, with Applications to Computer Vision,” Università Ca' Foscari di Venezia, 2004.

[41] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, Second ed.: J. Wiley & Sons, 2000.

[42] M. Matteucci. "A tutorial on clustering algorithms,", http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/, visited on 26 August 2008.

[43] Y. Chen, "An unsupervised learning approach to content-based image retrieval," *IEEE Proc. Inter. Symposium on Signal Processing and Its Applications* vol. 1, pp. 197-200, 2003.

[44] "Graph Theory," ; http://en.wikipedia.org/wiki/Graph_theory, visited on 28 June 2008.

[45] D. A. F. J. Ponce, *Computer Vision: A Modern Approach*: Prentice Hall, 2002.

[46] J. G. Auguston, and J. Minker, "An analysis of some graph theoretical clustering techniques," *J. ACM,* vol. 17, no. 4, pp. 571-588, 1970.

[47] M. P. M. Pelillo, "Generalizing the Motzkin-Straus theorem to edge-weighted graphs, with applications to image segmentation," *Lecture notes in computer science*, pp. 485-500, 2003.

[48] T. S. Motzkin, and E. G. Straus, "Maxima for Graphs and a New Proof of a Theorem of Turán," *Canadian J. Math,* vol. 17, pp. 533-540, 1965.

[49] M. Pavan, and M. Pelillo, "Dominant Sets and Pairwise Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 29, no. 1, pp. 167-172, 2007.

[50] M. P. M. Pelillo, "A New Graph-Theoretic Approach to Clustering and Segmentation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 145-152, 2003.

[51] U. Sakarya, Z. Telatar, "Graph partition based scene boundary detection," In: M. Petrou, T. Saramäki, A. Erçil, S. Lončarić (Eds.), Proc. The 5th International Symposium on Image and Signal Processing and Analysis (ISPA 2007), Istanbul, Turkey, 2007, pp. 544-549.

[52] U. Sakarya, Z. Telatar, "Video sahne sezme için çizge temelli bir yaklaşım," in IEEE 16. Sinyal İşleme, İletişim ve Uygulamaları Kurultayı (SİU 2008), Didim, Aydın, 2008.