An M Tech Dissertation

titled

HISTOGRAM BASED EFFICIENT VIDEO SHOT DETECTION ALGORITHMS

Submitted in partial fulfilment towards the award of the degree of

MASTERS OF TECHNOLOGY IN COMPUTER ENGINEERING

BY

Mr. Padalkar Milind Gajanan

Supervisor

Dr. M. A. Zaveri, SVNIT, Surat

2009-2010

Department of Computer Engineering

SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY, SURAT

Declaration

I hereby declare that the work being presented in this dissertation report entitled "HISTOGRAM BASED EFFICIENT VIDEO SHOT DETECTION ALGORITHMS" by me i.e. Mr. PADALKAR MILIND GAJANAN, bearing Roll No: P08CO961 and submitted to the Computer Engineering Department at Sardar Vallabhbhai National Institute of Technology, Surat; is an authentic record of my own work carried out during the period of July 2009 to June 2010 under the supervision of Dr. M. A. ZAVERI. The matter presented in this report has not been submitted by me in any other University/Institute for any cause.

Neither the source code there in, nor the content of the project report have been copied or downloaded from any other source. I understand that my result grades would be revoked if later it is found to be so.

_____

(Padalkar Milind Gajanan)

C E R T I F I C A T E

This is to certify that the dissertation report entitled "HISTOGRAM BASED EFFICIENT VIDEO SHOT DETECTION ALGORITHMS", submitted by Mr. PADALKAR MILIND GAJANAN, bearing Roll No: P08CO961 in partial fulfillment of the requirement for the award of the degree of MASTER OF TECHNOLOGY in Computer Engineering, at Computer Engineering Department of the Sardar Vallabhbhai National Institute of Technology, Surat is a record of his own work carried out as part of the coursework for the year 2009-10. To the best of our knowledge, the matter embodied in the report has not been submitted elsewhere for the award of any degree or diploma.

Certified by

_____

(Dr. M. A. Zaveri)
Professor,
Department of Computer Engineering,
S V National Institute of Technology,
Surat – 395007
India

_____

PG Incharge,
M Tech in Computer Engineering,
SVNIT, Surat

_____

Head,
Department of Computer Engineering,
S V National Institute of Technology,
Surat – 395007, India

Department of Computer Engineering

SARDAR VALLABHBHAI NATIONAL INSTITUTE OF TECHNOLOGY, SURAT

(2009-10)

Approval Sheet

This is to state that the Dissertation Report entitled HISTOGRAM BASED EFFICIENT VIDEO SHOT DETECTION ALGORITHMS submitted by Mr. PADALKAR MILIND GAJANAN (Admission No: P08CO961) is approved for the award of the degree of Masters of Technology in Computer Engineering.

Board of Examiners

Examiners

_____

_____

Supervisor

_____

Head, Department of Computer Engineering

_____

Date:_____
Place:_____

Acknowledgements

Apart from my own efforts, the success of this report depends largely on the encouragement and guidelines of many others. I am especially grateful to my supervisor Dr. M. A. Zaveri, Professor, Computer Engineering Department, SVNIT, who has provided guidance, expertise and encouragement.

I express my heartfelt gratefulness to Dr. D. C. Jinwala and Prof. U. P. Rao, for their stimulating supervision whenever required during my dissertation work. I deeply thank Prof. Mrs. R. G. Mehta especially for providing the Video Camera to generate a few experimental videos. I am also thankful to the rest of the staff of Computer Engineering Department for their cooperation and support.

I acknowledge Mr. Suresh Limkar, Research Scholar, Computer Engineering Department, SVNIT, for his efforts to provide us the video tutorials on LaTeX. I would like to put forward my heartfelt acknowledgement to all my classmates, friends and all those who have directly or indirectly provided their overwhelming support during my project work and the development of this report.

I thank my parents and relatives in Mumbai for their support and encouragement throughout my project work. I am grateful to God for his blessings.

Padalkar Milind Gajanan

Abstract

Videos have become a popular means of entertainment over the years. With the increase in the amount of user generated videos, a large collection is readily available on popular video sharing websites. Searching for videos with desired content from such a large collection is becoming a tedious task. The viewers require better control over the video data and for this reason the video browsing and indexing applications are being developed. These applications require the videos to be available in an elementary form called shots in the initial step. Detection of shots requires extraction and comparison of various features of the video frames. In this report we present two efficient techniques for shot detection based on histogram feature.

We present the first technique which is used to detect shots joined by dissolve type transitions, where we initially declare the video frames to be either of dissolve type or non-dissolve type. Later, we iteratively combine the sequences of non-dissolve frames that constitute the different shots. To detect shots joined by both the dissolve type transitions as well as abrupt transitions, we propose a strict clustering based approach in which initially only the almost identical and consecutive frames are grouped into one cluster. We then iteratively calculate and compare the histograms of these clusters to merge the clusters having similar histograms. Finally we discard the clusters having number of frames less than a minimum number, to obtain the shots.

For experimental purpose we have used the videos downloaded from the popular video sharing website YouTube. The ground truth being unavailable for these videos, we also generated and used a few videos by manually adding the shot transitions, in order to have a firm ground truth. To evaluate the performance, we have used frame precision and frame recall as metrics. The experimental results prove that our dissolve detection based technique is able to reduce the number of misdetection to overcome the problem of over-segmentation, and, our strict clustering based technique runs significantly faster in comparison with the existing techniques.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Videos have become a popular means of entertainment over the years. Traditionally, videos were created only by a limited number of producers. But now, the commoners as well can afford and use with simplicity the video capturing devices, as a result of which there is an increase in the amount of user generated videos. A large collection of videos is readily available on various video sharing websites. Searching for videos with desired content from such a large collection has become a tedious task. Also, viewers want to have a better control over the video data. As a result, many video browsing, indexing and summarization applications are being developed [1] [2] [3] [4]. These applications require the videos to be available in an elementary form called shots. Thus, video shot detection becomes the primitive task in such applications.

Once the video shots are available, the keyframes from each shot can be extracted and used to represent the respective shot. Various features of the keyframes as well as the shot as a whole can then be used for content representation. These features can later be used by the indexing and retrieval applications for retrieval of videos based on the visual content.

Video shots are obtained by temporal segmentation of the entire video sequence, such that each segment is an uninterrupted sequence of video frames. Gargi *et al.* [5] define video shot as a contagious sequence of video frames recorded from a single camera operation, representing a continuous action in time and space. Video shot is

also defined as the longest continuous frame sequence that originates from a single camera take, i.e. the camera images in an uninterrupted run [6]. Thus, a video shot is a small video segment, which is a part of a larger video and is captured without any interruption from the time the camera is turned on to the time at which the camera is turned off.A keyframe is that frame of a shot which conveys maximum information about the visual content in the entire shot. Thus, a keyframe is the most representative frame of a shot.

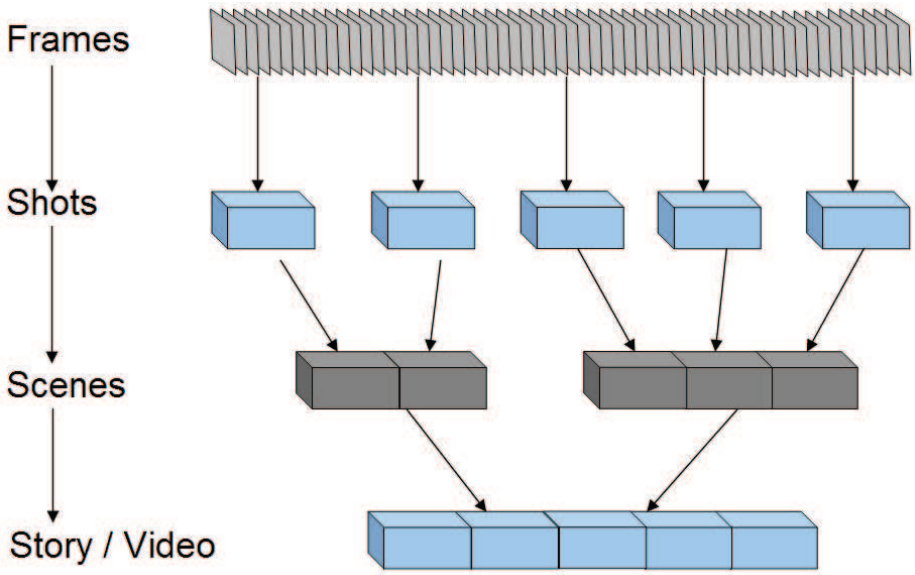
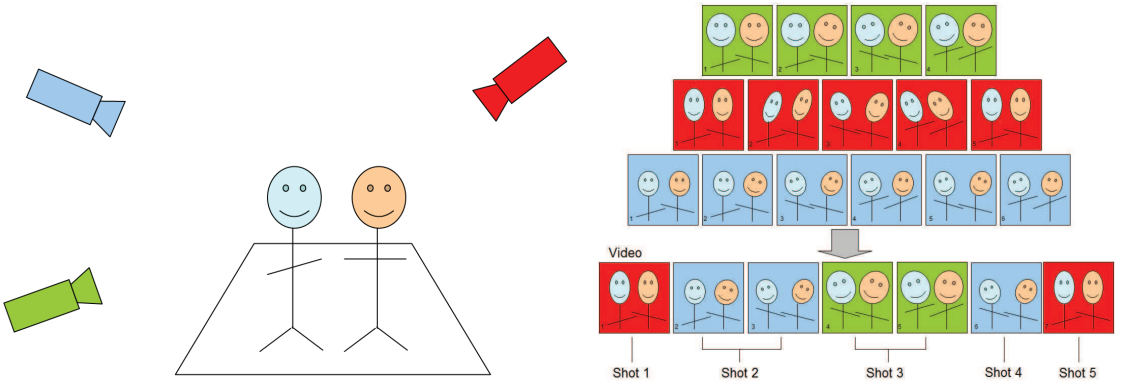


Figure 1.1: Hierarchical Structure of Video



(a) Physical scene captured by different cameras

(b) Sequences captured by different cameras (shots) combined to form the scene

Figure 1.2: A Video Scene

The hierarchical structure of a video is shown in figure 1.1. The frames are the elementary units of a video. The frames captured in a single uninterrupted camera take constitute a shot. A shot is therefore an elementary unit in the form of a shorter video. A scene is series of shots that are coherent from a narrative point of view. In other words, a scene is a collection of shots that convey different views of a single event or object and contain the same objects of interest. The illustration of a scene is shown in figure 1.2. The shots for a single event are captured by the individual cameras. These shots are then combined appropriately to form the scene. The story or the video is formed by placing the different scenes at appropriate places in the timeline.

1.1 Motivation

Video browsing and retrieval have become important activities for searching videos with desired content from a large collection. Also, with large number of user generated videos being produced everyday, the size of such collection keeps on increasing. Video browsing offers users a way to view and find relevant information from large number of video documents. On the other hand video retrieval enables users to find a particular video segment based on various attributes given as a query. These attributes may be description, keywords or even a sample picture of the required video segment.

Both video browsing and video retrieval need the original video document to be structured in some form, such that it forms a hierarchy (shown in Figure 1.1) of the basic elements of video streams. The fundamental unit of representing any video stream in the form of an independent video itself is a video shot. For any intelligent video processing system, analysis of video-shots plays an important role to know the characteristics of the video at hand. Therefore, the detection of these shots can be considered as an elementary operation for such applications. In addition, detecting shots manually is very cumbersome and highly subjective. Hence, automatic shot boundary detection is required.

Detection of a shot boundary is done by comparing various feature of the video

frames. The histograms provide a compact summarization of the data (intensity distribution) in a video frame and are also invariant to translation and rotation. Thus, the histogram of successive frames in a single shot would be having a nearly similar distribution of pixel intensities. These properties of the histogram feature and its usefulness in shot detection, along with the increasing importance of video shot detection in video indexing, retrieval and summarization applications, have motivated to carry out dissertation on video shot detection based on histogram feature.

1.2 Objective

The shot boundaries may be separated by various transitions like cuts and dissolves. If we are able to detect the dissolves, then the shot boundaries are automatically detected, as they lie on the two sides of the dissolve transitions. However, when we apply the existing approach for dissolve detection [7] to detect shot boundaries, it suffers from the problem of over-segmentation due to misdetections. The resulting shots are smaller video sequences that are parts of an originally larger shot. As a result, frames in between these smaller video sequences, which are also a part of an originally larger shot, are missed out. When we use the SVD based techniques for shot detection, the time taken by the detection algorithm increases with the increase in size of the feature matrix on which the SVD operation is to be applied. Therefore, more the number of frames in the video, more is the size of the feature matrix and more is the time taken by the detection algorithm.

Our objective is to present efficient techniques for shot detection that reduce number of misdetections and also have smaller execution time in comparison to the existing techniques. By reducing the misdetections we intend to overcome the problem of over-segmentation. We create feature matrices of smaller sizes in order to reduce the execution time.

1.3 Contribution

Our proposed techniques allow the shot boundary detection to be performed efficiently using a simple histogram feature. The use of SVD helps to compare the frame features using optimal approximations. As a result, we are able to find discontinuity in the contents of consecutive frames and thus detect the change in shot. Shot detection in videos having smooth shot transitions can be effectively performed using our dissolve detection based technique. The main contribution of this technique is that it is able to reduce the number of misdetection arising due to over-segmentation of the video, for videos having smooth shot transitions. Our strict clustering based technique is useful to detect shots in videos having both abrupt as well as smooth shot transitions. This technique has performance as good as to that of the existing techniques. The main advantage of using this technique is that it executes much faster in comparison to the existing techniques. This will facilitate the indexing, summarization and retrieval applications to respond quickly.

1.4 Outline

The organization of this report is as follows. In chapter 2 the theoretical background and literature survey are presented. In chapter 3 we discuss the proposed shot detection techniques. In chapter 4 the implementation methodology along with the tools used is presented. Performance results and analysis of the proposed algorithms is discussed in chapter 5. Conclusion and future work are given in chapter 6.

Chapter 2

Theoretical Background And Literature Survey

Frames surrounding the boundary of a shot exhibit significant variation in their content. The shot-boundary detection process is then the identification of considerable discontinuities in the visual-flow of the frame sequence. The theoretical background of the phases involved in shot detection process and the literature survey are presented in the following sections.

2.1 Theoretical Background

Before proceeding with identification of the discontinuities some preprocessing is required. This step involves noise removal, frame resizing, etc. Now that the preprocessing is done, the first step is feature extraction. A metric is then used to compare the features between frames f and $f + l$ (where l is the inter-frame distance i.e.). The discontinuity value is the measure of the variation in features from frame f to $f + l$. This value $d(f, f + l)$ is an input to the detector which is compared to a pre-defined threshold T. If $d(f, f + l) \geq T$ then a shot boundary between frames f and $f + l$ is detected [8]. This concept is shown in figure 2.1.

The features may be anything that can describe the visual content of a frame or group of frames (say Color histogram, Dominant Color Descriptor, etc.) [9]. A feature

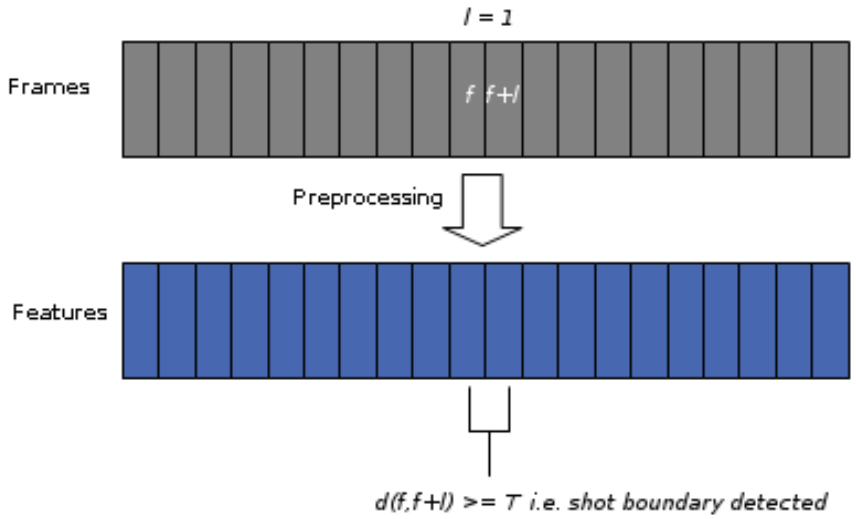


Figure 2.1: Shot Boundary Detection Concept

may also be a characteristic or property of a frame or a group of frames (say a pseudo image representing the nature of variation of pixel intensity observed over a group of frames).

2.1.1 Feature Comparison Techniques

To find similarity for the frame features, the singular value decomposition [10] concept has been widely used due to its ability of comparing entities based on their conceptual content. Therefore, it is important to understand the singular value decomposition of a matrix and its use in feature comparison. The advantage of SVD over other feature comparison metrics makes it suitable for shot detection.

Singular Value Decomposition (SVD)

Consider a $M \times N$ matrix A. Then, singular value decomposition (SVD) of this matrix is given by, $A = U\Sigma V^T$, such that the matrices U and V are $M \times M$ and $N \times N$ singular matrices respectively. Σ is a diagonal matrix of singular values and is of the same dimensions as that of A. The singular values of matrix Σ are in decreasing order. This allows us to discard the insignificant singular values, so that the size of both the matrices U and V are reduced, and still the reconstruction of matrix

A remains unaffected. This concept is shown later in figure 3.5. The comparison using only significant singular values in SVD allows us a simple strategy for optimal approximate fit using reduced matrices.

The example for document retrieval using SVD is given in [10]. In the example, the key terms used in the various documents are arranged as the rows of matrix A, while the various documents, according to their category, are arranged as the columns of matrix A. The number of times a term appears in a document is the number of the corresponding cell of matrix A. Using SVD on such a matrix and then reducing to only the significant singular valued representation, the rows of matrix $V\Sigma$ represent the coordinates of the documents. These coordinates can then be used to measure the cosine of their angle, which gives the measure of similarity between the documents.

Now, if the documents are replaced by the various frames in the video and the terms are replaced by the features, then a similar method can be used to find similarity in between the frames. The obtained discontinuity value can be used to detect the shots in the video. We do not want an exact match of the frames, but we require high similarity of frames based on the content. By using SVD, we are able to achieve this very effectively, because, it does an optimal approximate comparison to give the most appropriate values of discontinuity.

L_n norm

The discontinuity value in the frames can be measured using the L_n norm. If the K^{th} feature F of the frames are represented by F_k, then the L_n norm [6] to find the discontinuity value in frames f_i and f_j is,

$$D_{L_n}(f_i, f_j) = \sqrt[n]{\sum_{k=1}^{K} \left| F_{f_i}(k) - F_{f_j}(k) \right|^n} \tag{2.1}$$

By varying the value of n, we get different distance measures. The advantage of using this measure is that it is very simple to compute. But it has disadvantage is that it is highly sensitive to small changes or noise. Therefore, it cannot be used for

approximate comparison which is required for frame comparison in shot detection.

Chi-square

For the histogram feature, the commonly used metric of similarity measure is the chi-square. If the K^{th} feature F of the frames are represented by F_k, then the L_n norm [6] to find the discontinuity value in frames f_i and f_j is,

$$D_{\chi^2}(f_i, f_j) = \sum_{k=1}^{K} \frac{\Big(F_{f_i}(k) - F_{f_j}(k)\Big)^2}{F_{f_i}(k)} \tag{2.2}$$

Chi Square is employed to test the difference between an actual sample and another hypothetical or previously established distribution such as that which may be expected due to chance or probability. For calculating the discontinuity value, Chi Square can been used to test differences between two actual samples or features.

2.1.2 Frames Features

It is generally observed that the frames surrounding the boundary of a shot display a significant change in their content. If the change is drastic then it is called a *hard-cut* (or simply *cut*). When the change is gradual and spread over a group of frames, it can be classified as a *fade*, *wipe* or *dissolve* transition. Out of these, the *dissolve* transitions are the most difficult to detect, because, the change in visual content of the frames in transition is very smooth. If the dissolves are detected, the shot boundaries lie on the two sides of the dissolve transitions.

The change in contents can be detected by comparing various features of the frames. These features need to be insensitive to object / camera motion and lighting changes. Table 2.1 summarizes the features used by the existing techniques for shot boundary detection.

Features Used / Can Detect	Cuts	Fades	Dissolves
Color Histogram	Yes	-	-
Edge Change Ratio	Yes	Yes	Yes
Edge Based Contrast	-	Yes	-
Standard Deviation of Pixel Intensity	-	Yes	Yes

Table 2.1: Features used to detect various transitions

Color Histogram

Color content does not change rapidly within a shot, but across shots. Therefore color histogram can be generated for every frame in the video and the comparison can be done between them. If the difference in color histogram between two frames exceeds a certain user specified threshold then a shot change can be marked. This makes color histogram a suitable feature for abrupt change detection and has therefore been used for cut detection.

Edge Change Ratio

For a given video if the edge information of the frames is used, then, for consecutive frames we observe that some edges that were not present in the 1^{st} frame appear into position (in) in the 2^{nd} frame, while some edges that are in the 1^{st} frame are missing or disappear (out) in the 2^{nd} frame. The number of such *in* and *out* edges can be counted. If both the frames belong to the same shot then the number of *in* and *out* edges is more or less same. On the other hand if this difference is sufficiently large then a shot change is detected.

If there are n pixels in the frame and the *in* edges are X_{in} and the *out* edges are X_{out} for frames f and $f-1$ respectively, then the edge change ratio (ECR) as define in [11] is,

$$ECR_f = max\left(\frac{X_{in}}{n}, \frac{X_{out}}{n}\right) \tag{2.3}$$

Thus, whenever we obtain peaks in the frame-ECR plot, the peak indicates shot change. Whenever an isolated peak is obtained, it signifies a hard-cut, whereas a

peaks spread over a number of frames indicates dissolve transitions.

Edge Based Contrast

The dissolve transition leads to change in contrast of the edge pixels in an image [12]. This fact is exploited to detect dissolves using the Edge Based Contrast feature. Consider two shots joined by a dissolve transition. The edge contrast of the 1^{st} shot slowly goes on decreasing, while that of the 2^{nd} shot goes on increasing. Thus, the contrast is the least at the midpoint of a dissolve. This helps in detecting dissolves and thus the associated shots.

Standard Deviation of Pixel Intensity

Fade and Dissolve transitions are composed of monotonous change in pixel intensities. Using sliding window one can monitor frames for their intensity change. If sufficient numbers of pixels show monotonous change in their intensity within the window, we can assign current frame to be frame in dissolve transition [7].

2.2 Literature Survey

The widespread availability of a large number of videos and the need of the users to have a good control over the video data has led to the growing demand of tools for efficient indexing, browsing and retrieval of the videos. Structural analysis of the video in the form of shots is a prerequisite step in these application and therefore shot boundary detection has been an active area of research for the past decade. The transition between the shots can be either abrupt (cuts) or gradual (dissolve, fade). Based on these transitions, many detection techniques have been proposed in the literature. A comparative study of the early attempts for shot detection can be found in [12] [13] [14] [15].

The abrupt change in the video content is easier to detect as compared to detecting smooth change. Because of this reason, in the early days, the techniques for detection

of shots separated by cuts have been successful to a large extent. Cuts have been successfully detected in [16] [17] [18] [19] [20] [21]. A simple method for the detection of hard cuts using only interframe differences has been proposed in [16]. They used the concept that a meaningful event is defined to be having a large deviation from the expected background process. Therefore, such events would be the ones having little probability of occurrence given a probabilistic background model. They defined a hard cut when the interframe differences have little probability to be produced by a given model of interframe differences of non-cut frames. This techniques is simple, because, only the interframe differences are used and there is no need to perform motion estimation or any other type of processing.

Hard-cut detection for archive film is addressed in [17] which is mainly for black-and-white videos. This hard-cut detection system is based on modified phase correlation such that hard-cut detection is carried out using spatially sub-sampled video frames, and a candidate hard-cut is indicated in the case of low correlation. For uniformly colored video frames the phase correlation is extremely sensitive to noise and visual effects. Their work provides a through theoretical analysis to show the usefulness of spatial sub-sampling.

The technique in [18] makes use of the intuition that frames preceding a cut are similar to each other, and dissimilar to those following the cut. First, for each frame in a video, they extract from video footage a set or window of consecutive, ordered frames centered on the current frame. Second, that order the frames in the window by decreasing similarity to the current frame. Last, they inspect the ranking of the frames, and record the number of frames preceding the current frame in the original video that are ranked in the first half of the list; calling this the pre-frame count. They repeat this process for each frame. Cuts are then detected by identifying significant changes in the pre-frame count between consecutive frames.

Porter *et al.* [19] have proposed a technique for detection of cuts in the frequency domain. They perform normalized correlation in the frequency domain on small blocks, and determine an overall correlation coefficient for each frame based on the most similar blocks and by rejecting the more dissimilar ones. They use a

fixed threshold making it ideal for automatic shot detection. Since for videos having different contents, using a fixed threshold may not always work for all the variety of videos. Keeping this in mind, an adaptive threshold based techniques is proposed in [20]. They proposed three models namely 'Covariance model', 'Proportional model' and 'Dugad model' [22] for calculating the thresholds adaptively for every test video. They showed that adaptive thresholding considerably improves the rate of detection for shot cuts regardless of the method used.

Video cut detection technique using Gabor filter is proposed in [21]. Their feature extraction technique that uses 2D Gabor filtering, computing tridimensional image feature vectors for the video frames. The identification approach used in their proposed technique is using an automatic unsupervised distance classification procedure instead of using the traditional way of thresholding. They compute 3D image feature vectors to provide frame content characterization.

A new approach for shot boundary detection in the uncompressed image domain, based on the mutual information and the joint entropy between consecutive frames is proposed in [23]. The mutual information is a measure of transported information from one frame to another. Mutual information is used for detecting abrupt cuts, where the image intensity or color is abruptly changed. In the case of a fades and dissolve, where visual intensity is usually decreasing or increasing to a black image or frame in the consecutive shot, the decreasing or increasing inter-frame joint entropy respectively is used as a metric. Their approach combines two shot boundary detection schemes based on color frame differences and color vector histogram differences between successive frames.

Cooper *et al.* [24] [25] use self similarity of the video across time for shot detection where the similarity of every frame with every other frame in the video is tested. They analyze the inter-frame similarity matrices to find the shot boundaries. The approach is flexible to the choice of both feature parametrization and similarity measure and it is robust because the data is used to model itself. They detect shot boundaries by considering the self-similarity of the video across time. For each instant in the video, the self-similarity for past and future regions is computed, as well as the cross-

similarity between the past and future. A significantly novel point in the video, i. e. a shot boundary, will have high self-similarity in the past and future and low cross-similarity between them. The color information for region based segmentation in given in [26].

Texture feature representation found in [27] can be taken up as a feature for detecting shots. Texture granularity refers to size of basic primitives of a given texture. Small granularity means that the given texture consists of large primitives and vice versa—large granularity means that the texture consists of small primitives. In other words, small granularity implies a finer texture while large granularity implies a coarser texture. A texture with large primitives is fairly uniform over large areas compared to a texture with small primitives which exhibits more frequent changes in pattern over the image space. This leads to the idea of measuring granularity as the spatial rate of change of the image intensity. Whenever the shot change occurs, both abrupt and smooth, the texture of changes. Thus, the granularity of the texture also changes and therefore can be used for shot change detection. Shots having similar value of texture granularity, although the texture is not similar to each other cannot be distinguished using this feature.

Zhao *et al.* [28] use a minmax optimization method to improve color based shot detection. They proposed a new learning criterion which keeps the detection rate fixed while at the same time reduces the false alarm rate. Shot boundary detection based on Singular Value Decomposition (SVD) using the histogram feature is described in [29] and [30]. In both these techniques a feature matrix is created which is subjected to SVD so that the frame similarity can be determined. Similar frames can then be grouped together to obtain shots.

In [29] the feature matrix consists of the frame histograms arranged as the columns. The complete matrix is then subjected to SVD to find the cosine of angle between consecutive frames. The frames for which the cosine of the in between angle is below a user defined threshold are included in one cluster and those for which the cosine exceeds the user defined threshold are put into different clusters. These clusters form the various shots in the video. The advantage of using SVD in this case is that the

approximate comparison of the frame histograms is possible using reduced singular matrices. This technique is able to detect the shots separated by both the abrupt as well as smooth transitions. As the number of frames increase in the video, the size of the feature matrix will also increase. This leads to increase in time to compute the the SVD of the feature matrix. Therefore, although this technique is suitable to be used for videos having abrupt as well as smooth transitions, the time taken will be very large as the video size increases. Also, for computing the SVD of a large matrix, more resources will be required, making this technique expensive in terms of both time and cost.

In the technique used by [30], the feature matrix is constructed for a small number of frames (i.e. frames in a fixed sized observation window). The rows of the feature matrix represent the color histogram of the frames in the observation window and this matrix is updated as the new frames enter for shot detection. The advantage of this technique is that the complete video is not required at hand before starting the detection process, unlike all the other techniques. This makes the technique suitable for real time shot detection as and when the new frames become available. Once the feature matrix is available, it is subjected to SVD to obtain the rank of the frame under consideration. This is followed by rank tracing, which results in detection of the shot boundaries. This technique is suitable for sports videos.

Su *et al.* [7] have presented an algorithm to detect the *dissolve* type transitions using change in pixel intensity over time. If the smooth shot transitions are detected using this technique, then it can be used to declare shot boundaries which lie on the two sides of such smooth shot transitions. The detection of dissolve transitions is done by observing the change in pixel intensity over a group of frames. If for the video frames, a certain minimum proportion of pixels show a monotonous change in pixel intensity over the observation window, then such frames are declared to be in dissolve type transitions. Once the dissolve type transitions are detected, shot boundaries are the ends of these smooth transitions.

Chapter 3

Design And Analysis

Shot detection is the splitting of a video into temporal units, such that each unit is a sequence of consecutive frames, taken by a single camera, representing continuous action in time and space. As mentioned in chapter 2, various features of the frames are extracted and compared in order to find a discontinuity value, which is compared against a pre-specified threshold to declare a shot boundary. Our proposed shot detection techniques are discussed in detail in the following sections. Both our proposed shot detection techniques use the histogram feature along with SVD for shot identification.

3.1 Dissolve Detection Based Shot Identification

Shot boundary detection for shots separated by abrupt changes has been successful to a large extent, but detecting shot boundaries with gradual transitions in between, has been very challenging. The gradual change spread over a group of frames can be classified as a *fade*, *wipe* or *dissolve* transition. These transitions are the most difficult to detect, because, the change in visual content of the frames in transition is very smooth. If these transitions are detected, the shot boundaries lie on the two sides of the dissolve transitions. These transitions are discussed below.

- *Fade-out*: The frames in this transition go on fading out until the original content

of the shot completely vanishes. What remains is a totally black frame. This effect is usually used in movies to end a scene very smoothly. A sample of such a transition is shown in figure 3.1(a).

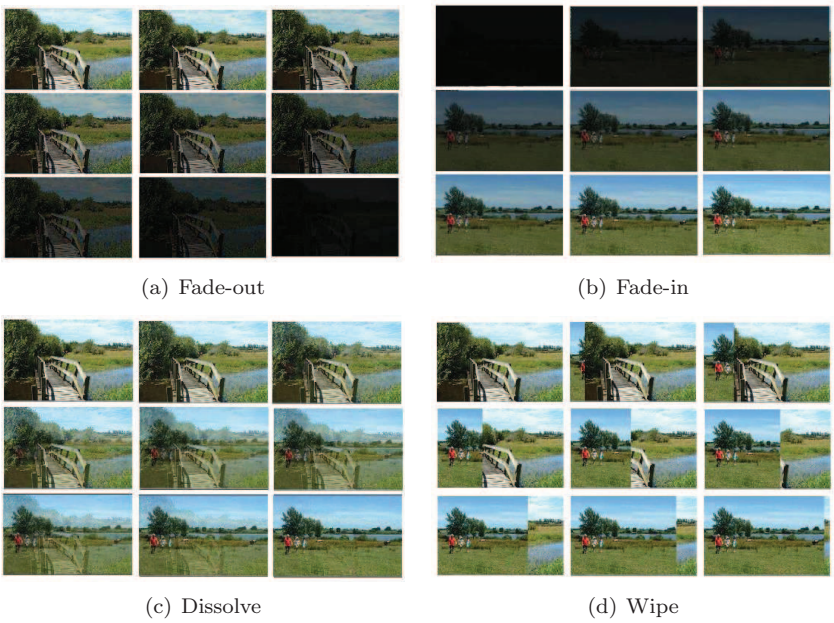
- *Fade-in*: The frames in this transition appear into position from a completely dark sequence. The each successive frame becomes brighter till the shot starts. This effect is popularly used in movies to start a scene smoothly. Figure 3.1(b) shows a fade-in transition.

- *Dissolve*: A dissolve transition is typically a combination of a fade-out transition and fade-in transition where the dark sequence is replaced by another shot. The ending frames of the first shot go on fading-out slowly, and at the same time the frames in the second shot appear into position (fading-in). Thus, the transition from one shot to another is very smooth. This effect is popularly used in the video industry for smooth scene and shot changes. An example of this effect is shown in figure 3.1(c).

- *Wipe*: This transition is not a very smooth transition. The frames of the first shot in this transition are replaced by frames in the second shot in steps. Initially no frames in the second shot are visible. Gradually in steps, the parts of frames in the second shot replace the respective parts of the frames in the first shot. Such a horizontally replacing wipe transition is shown in figure 3.1(d).

In wipes, the difference in consecutive frames is small but significant such that the discontinuity in the spatial domain is obvious and can be used for purpose of detection. In dissolve, fade-out and fade-in transitions (together addressed as dissolve type transitions from here on), it is difficult to find a clear distinction between two consecutive frames, thus often becoming hard to detect. Another problem that makes dissolve detection difficult is that it is often confused with motion.

In the proposed approach, we start with feature extraction, followed by detecting dissolve type transitions. This enables us to divide entire video into two categories viz. dissolve frames (which are part of dissolve type transitions) and non-dissolve



(a) Fade-out (b) Fade-in

(c) Dissolve (d) Wipe

Figure 3.1: Video Transitions

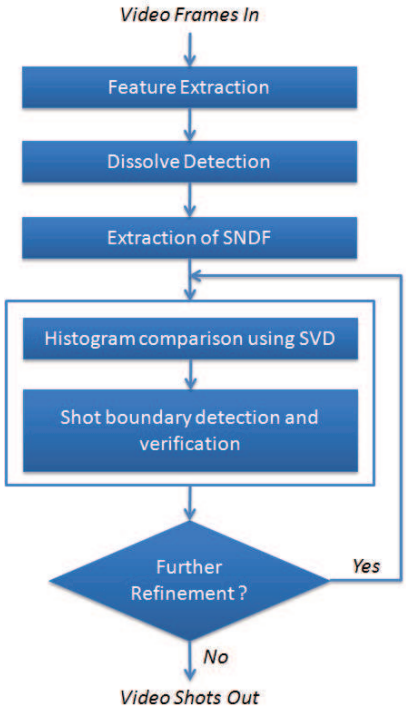


Figure 3.2: Proposed Approach Using Dissolve Detection For Shot Identification

frames. Simply combining the consecutive non-dissolve frames and assuming these as shots suffers from the problem of over-segmentation due to misdetections. The resulting shots are smaller video sequences that are parts of an originally larger shot. As a result, the frames in between these smaller video sequences, which also are a part of the originally larger shot, are missed out. These smaller video sequences called Sequence of Non-Dissolve Frames (SNDF) give a rough estimation of shots present in the video. A generalized histogram for every SNDF is calculated using the histograms of the frames present in the respective SNDF. We then iteratively compare these generalized histograms using Singular Value Decomposition (SVD) to merge similar SNDF. Figure 3.2 shows the flow of our approach. We discuss in detail, the extraction of the required features, dissolve detection and histogram comparison using SVD in the following subsections.

3.1.1 Feature Extraction

Features depict the visual aspects of the content of the video. In general, the shot boundary is detected by comparing the global features of the frames which are separated by a certain fixed distance and then assigning a discontinuity value to these pair of frames. Whenever the discontinuity value crosses a certain threshold, a shot boundary is said to be detected between these pair of frames.

In our approach, we extract two global features for every frame in the video. The first feature is the frame histogram. Histograms represent the distribution of the pixel intensities in frame. Whenever there is a drastic change in the content of a pair of frames, the pixel intensity distribution changes significantly. This property of histogram feature is exploited for the detection of shot boundaries. The second feature is the ratio of number of pixels changing intensity monotonously in an observation window to the number of pixels changing intensity at least once in the same window. This feature is used to declare a frame to be a part of a dissolve type transition, and has been used for dissolve detection in [7].

A dissolve transition consists of a certain number of final frames of the first shot

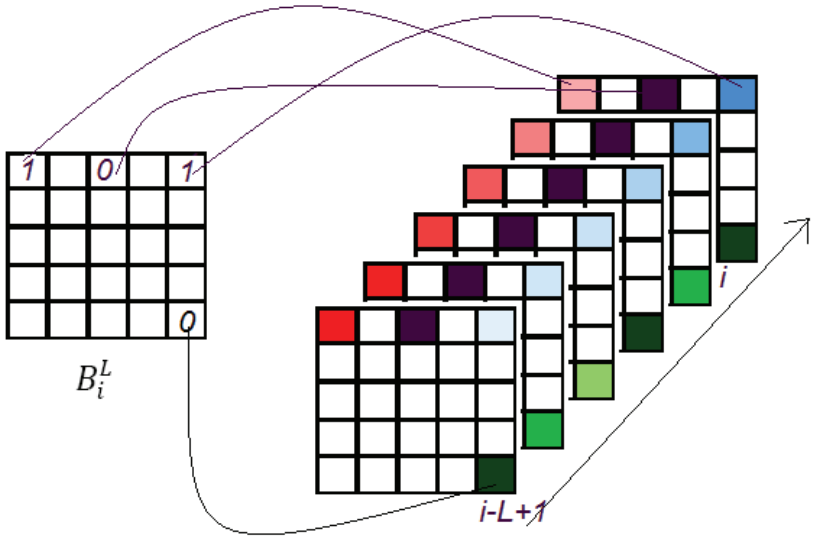


Figure 3.3: Pseudo Image B_i^L for i^{th} frame

and initial frames of the second shot. Fade transition can also be considered as a special case of the dissolve transition where either of the shots is a dark sequence. We have therefore referred together the dissolve transitions and fade transitions as dissolve type transitions throughout this report. It is observed in a dissolve type transition that the intensity of the pixels in successive frames either increases or decreases. With this observation, the change in pixel intensity is noted over a group of frames in a sliding window fashion. Wherever the change is seen to be monotonous, the count of such pixels is associated with the current frame. Also the count of pixels changing intensity at least once in the same observation window is also associated to the current frame.

Assume that the observation window consists of $L+1$ frames and let f_k denote the k^{th} frame in the video. For any i^{th} frame in the video, the observation window will span $[i-L+1, i]$ frames. A pseudo binary image B_i^L will then be generated for the i^{th} frame such that a pixel (x, y) in this pseudo binary image will have a value 1 if monotonous change is observed in the observation window (proponent pixels), 0 otherwise. Figure 3.3 shows a sample of pseudo image B_i^L for the i^{th} frame.

$$B_i^L(x,y) = \begin{cases} 1, & \text{if,} \quad f_k(x,y) - f_{k-1}(x,y) < 0 \quad \forall k \in [i-L+1, i] \\ 1, & \text{if,} \quad f_k(x,y) - f_{k-1}(x,y) > 0 \quad \forall k \in [i-L+1, i] \\ 0, & \text{Otherwise} \end{cases} \tag{3.1}$$

Similarly, the number of pixels changing intensity at least once in the observation window can be calculated as follows.

$$N_i^L(x,y) = \begin{cases} 0, & \text{if,} \quad f_k(x,y) = f_{k-1}(x,y) \quad \forall k \in [i-L+1, i] \\ 1, & \text{Otherwise} \end{cases} \tag{3.2}$$

$$S(N,L,i) = \frac{\sum\limits_{x,y} B_i^L(x,y)}{\sum\limits_{x,y} N_i^L(x,y)} \tag{3.3}$$

$S(N,L,i)$ in equation 3.3 is the ratio of number of pixels changing intensity monotonously in an observation window to the number of pixels changing intensity at least once in the same window.

3.1.2 Dissolve Detection

For dissolve detection, we use the method proposed in [7] as follows. The $S(N,L,i)$ value of every i^{th} frame is compared with a certain threshold. Frames for which $S(N,L,i)$ value is above the threshold are declared to be in a dissolve type transition. The probability of monotonous change in pixel intensity over a group of $L+1$ frames is taken to be $P(B^L(x,y)=1) = (1/2)^{L-1}$. Therefore, $P(B^L(x,y)=0) = 1-(1/2)^{L-1}$. Using the binomial expression for these events the following CDF is calculated.

$$CDF(N_s, L, \delta^{'}) = \sum_{K=0}^{\delta^{'}} \binom{N_s}{K} (P(B^L(x,y) = 1))^K \times (P(B^L(x,y) = 0))^{N_s - K} \tag{3.4}$$

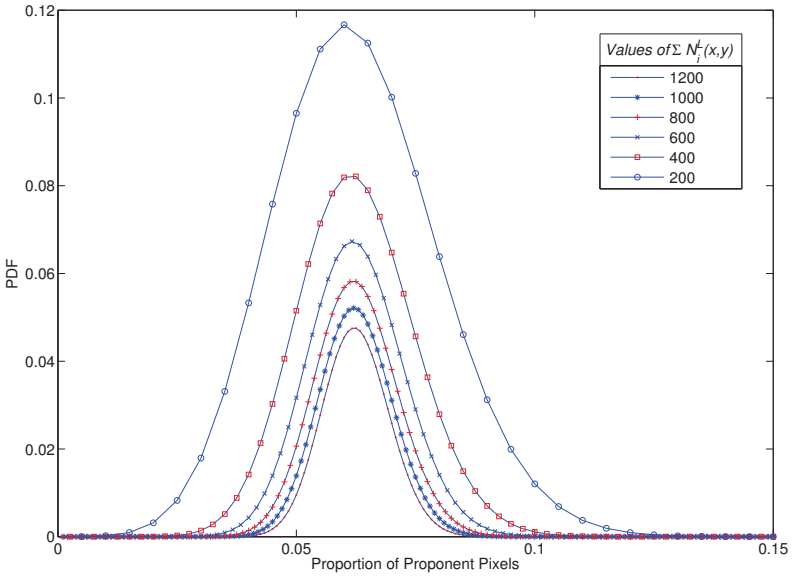
where N_s is the minimum value of $\sum N_i^L(x,y)$. The monotonous change in pixel intensity in the window can be accounted to two reasons viz. dissolve type transition and change in background environment. In equation 3.4, $\delta^{'}$ is the number of pixels changing intensity monotonously in the observation window due to change in the background environment.

The value of N_s is taken as the minimum value of $\sum N_i^L(x,y)$ because, as the value of $\sum N_i^L(x,y)$ goes on increasing, the curve of the *CDF* in equation 3.4 becomes steeper. As the curve becomes steeper, the *CDF* takes less time to saturate. On the other hand, if the value of $\sum N_i^L(x,y)$ goes on decreasing, the curve becomes smoother, thus taking longer time to saturate. The smallest value of $\sum N_i^L(x,y)$ will assure that it take longest time for the curve to saturate. Now if we take the very point at which the *CDF* saturates to 1 as the cut-off, it will correspond to the maximum number of active pixels due to background environment. If the number of active pixels exceeds this number, then the exceeding active pixels are due to dissolve type transition. The value of $\delta^{'}$ corresponding to saturation point can then be used to calculate the threshold.

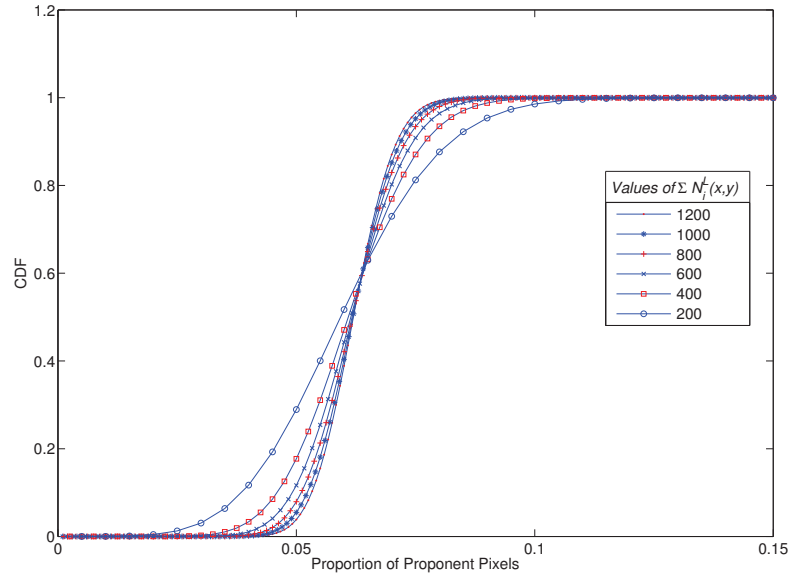
$$\delta_1 = \frac{\delta^{'}}{N_s} \tag{3.5}$$

where δ_1 is the threshold. The frames for which $S(N, L, i) \geq \delta_1$ are declared to be the frames in dissolve type transition. The remaining frames are declared as non-dissolve frames. Frames with no dissolve frames in between, are grouped together as SNDF.

We have detected dissolves by observing the change in pixel intensity over a group



(a)



(b)

Figure 3.4: (a) PDF and (b) CDF of binomial distribution using different values of $\Sigma N_i^L(x, y)$. The CDFs for smaller values of $\Sigma N_i^L(x, y)$ take longer time to saturate.

of frames. Pixel intensity can change many times within a shot. Also, within a clip, the variation in background causes change in pixel intensity from frame to frame. Thus, using only pixel intensity as a cue for shot detection causes over-segmentation of the video. We obtain smaller video sequences that are parts of an originally larger shot. These sequences (SNDF) therefore cannot be considered as shots. To extract shots from SNDFs, we combine similar SNDFs using the process explained in the following subsection.

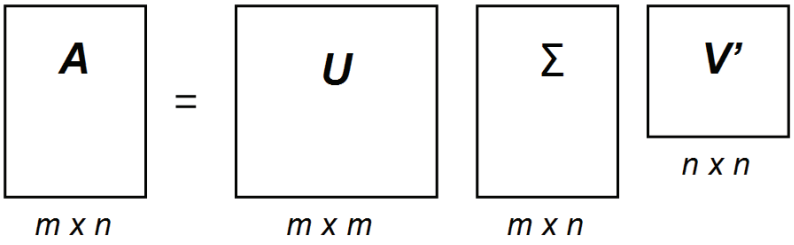
3.1.3 Histogram Comparison using SVD

In this step, we compare histograms of consecutive SNDFs using SVD and merge them into one SNDF if found similar. The process starts with obtaining a histogram representing each SNDF. Consider h_i to be the histogram of frame f_i in the SNDF S_j. The SNDF histogram is calculated as,

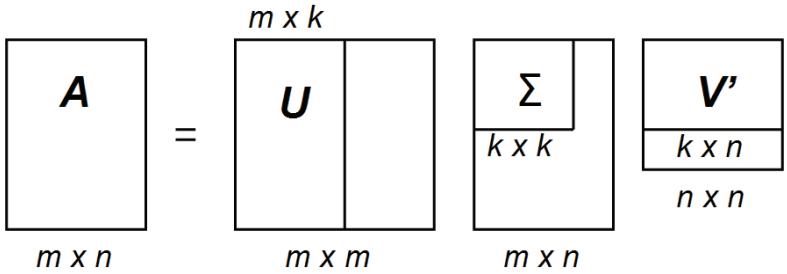
$$H(S_j) = \begin{cases} h_i & \text{if } i = 1 \quad \forall f_i \in S_j \\ \\ \dfrac{H(S_j) + h_i}{2} & \text{if } i \neq 1 \quad \forall f_i \in S_j \end{cases} \tag{3.6}$$

We then form a matrix A, columns of which are the SNDF histograms. A is factorized to $A = U\Sigma V^T$ using Singular Value Decomposition (SVD). SVD is a powerful linear algebra technique which exposes the geometric structure of a matrix. A matrix can be seen as a transformation from one vector space to another. The rank of a matrix, singular values and orthogonal matrices of a given matrix can be found using SVD. This operation can be applied to any real matrix A. SVD has been used successfully for image compression for a few years [31] and has also found its application in Video Shot Segmentation. Following is the mathematical explanation of SVD.

If A is an $M \times N$ matrix, then U is an $M \times N$ matrix of left singular vectors, V is an $N \times N$ matrix of right singular vectors, and Σ is a matrix of singular values such that $\Sigma = diag(\sigma_i, \sigma_2, ..., \sigma_R)$, where R is rank of A and $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_R \geq 0$



(a)



(b)

Figure 3.5: (a) Singular Value Decomposition of matrix A. (b) Singular Value Decomposition of matrix A considering only k singular values.

[29]. Thus, σ_1 and σ_R are the most significant and the least significant singular values respectively. If we decide to retain only the first K significant singular values then we can approximately reconstruct A with dimensions $M \times N$ using $A = U\Sigma V^T$, where U is $M \times K$, Σ is $K \times K$ and V^T is $K \times N$. In fact, if $K = R$ then we can exactly reconstruct A.

Now, if the columns of A are considered as documents and the rows of A (i.e. gray levels) are considered as terms, then we can view A as a term-document matrix. Deerwester *et al.* [10] have presented a technique to compare two documents using the Latent Semantic Analysis (LSA). They considered rows of matrix $V\Sigma$ as coordinates of the documents. Let us represent these coordinates by $\widetilde{v}_i$ for every i^{th} document i.e. let the i^{th} row of $V\Sigma$ be represented by $\widetilde{v}_i$ which corresponds to SNDF S_i. The cosine of angle between coordinates of two documents denotes the measure of their similarity. If $\widetilde{v}_i$ and $\widetilde{v}_j$ are similar, then the cosine of angle between them is closer to 1, else it is closer to 0. Thus, to measure similarity between SNDFs S_i and S_j we use the following metric.

$$\Phi(S_i, S_j) = cos(\widetilde{v}_i, \widetilde{v}_j) = \frac{(\widetilde{v}_i.\widetilde{v}_j^T)}{\|\widetilde{v}_i\| \, \|\widetilde{v}_j\|} \tag{3.7}$$

Based on a threshold (δ_2) which we derived by experimental observations, we decide whether or not to combine the two consecutive SNDFs S_i and S_j. Thus, SNDFs are combined to dynamically form N clusters. These clusters are nothing but the modified SNDFs. We use the method proposed in [29] with minor modifications to suit our technique for combining similar SNDFs.

- Initially the first two SNDFs satisfying the condition $\Phi(S_i, S_{i+1}) \geq \delta_2$ are combined into one cluster C_1. The mean of this cluster is then calculated as,

$$\widetilde{m_1} = \frac{\widetilde{v}_i + \widetilde{v}_{i+1}}{2} \tag{3.8}$$

- The next SNDF S_i is tested for addition in the same cluster.

$$cos(\widetilde{m}_1, \widetilde{v}_i) \geq \delta_2 \tag{3.9}$$

  if this condition is satisfied, then $\widetilde{m}_1$ is updated, else S_i becomes the seed of a new cluster with mean $\widetilde{m}_j = \widetilde{v}_i$. These steps are repeated till all the SNDFs are processed.

If the number of these clusters is the same as number of SNDFs in the previous step, then we declare the SNDFs obtained in the previous step as the final shots. If this is not the case, then the modified SNDFs S_i for corresponding cluster C_i are calculated as follows.

- Start of SNDF S_i = Start of 1^{st} SNDF in cluster C_i
- End of SNDF S_i = End of *last* SNDF in cluster C_i

The new SNDF histograms need to be calculated to proceed with the next iteration. The SNDF histograms for the modified SNDF are calculated as follows.

$$H(S_i) = mean(H_{iD}, h_{iM}) \tag{3.10}$$

where H_{iD} represents SNDF histograms of the SNDF in cluster C_i, and h_{iM} represents the histogram of the frames in between the SNDFs in cluster C_i which were detected as dissolve frames previously. Once we have the modified SNDF histograms, we start the next iteration of factorizing the SNDF histogram matrix using SVD. Thus, combining the similar SNDFs will assure that the frames in-between the similar SNDFs, that were delcared to be frames in dissolve type transition previously, will not be misdetected. This will improve the performance of the detection algorithm.

3.2 Strict Clustering Based Shot Detection

In this approach, we propose a divide-and-merge technique for shot detection. The video frames can be initially considered as independent clusters, similar to that of leaf nodes in a tree structure. Based on histogram similarity, the frames can be combined to form modified clusters. This process of comparison and merging can be performed iteratively until no more merging is possible. The final clusters having number of frames less than a certain minimum number are declared to be frames in shot transition and are therefore discarded. The remaining clusters are the finally obtained shots.

We start with extraction of 3D histogram feature for every frame. The initial frames stand as independent clusters at this stage. We then compare every group of N_{step} number of consecutive frames for similarity. Only the consecutive similar frames in every group are combined into a single cluster. The 3D histogram of each cluster is then modified accordingly. This initial clustering is done very strictly so that only the almost identical frames remain in one cluster. We then iteratively combine similar clusters. Finally, we discard the clusters having frames less than a certain minimum number to obtain the final shots. We discuss in detail every step in the following

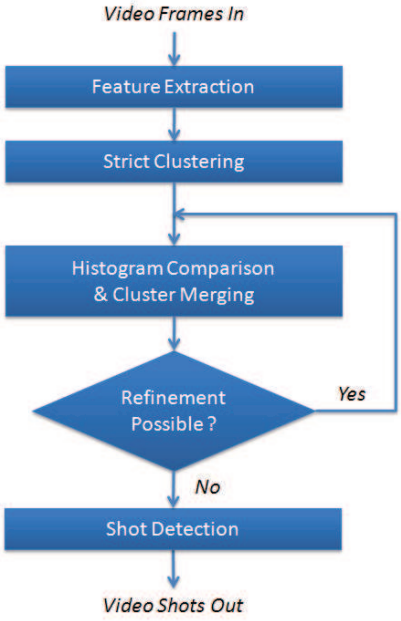


Figure 3.6: Proposed Approach for Shot Detection Using Strict Clustering

subsections.

3.2.1 Feature Extraction

Features convey the prominent attributes of a frame. As mentioned earlier, for shot detection, the frame features are compared with a pre-defined threshold to find the visual discontinuity. In this approach we have used the three-dimensional histogram feature in the RGB color space with 16 bins. Thus, the dimensionality of the feature vector is $16^3 = 4096$. The calculation of this three-dimensional histogram is done as follows.

Since the range from 0 to 255 (assuming that the frame is 4-bit image) is to be divided into 16 bins of equal sizes, the values of R_c, G_c and B_c are,

$$R_c/G_c/B_c = \begin{cases} 0 & if \quad r/g/b \in [0,15] \\ 1 & if \quad r/g/b \in [16,31] \\ 2 & if \quad r/g/b \in [32,47] \\ 3 & if \quad r/g/b \in [48,63] \\ 4 & if \quad r/g/b \in [64,79] \\ 5 & if \quad r/g/b \in [80,95] \\ 6 & if \quad r/g/b \in [96,111] \\ 7 & if \quad r/g/b \in [112,127] \\ 8 & if \quad r/g/b \in [128,143] \\ 9 & if \quad r/g/b \in [144,159] \\ 10 & if \quad r/g/b \in [160,175] \\ 11 & if \quad r/g/b \in [176,191] \\ 12 & if \quad r/g/b \in [192,207] \\ 13 & if \quad r/g/b \in [208,223] \\ 14 & if \quad r/g/b \in [224,239] \\ 15 & if \quad r/g/b \in [240,255] \end{cases} \tag{3.11}$$

where R_c, G_c and B_c will correspond to the respective bin numbers in the RGB color space for a pixel (x, y) having intensities [r,g,b] in the respective color channels. Number of intensities in every bin is $(2^8)/16 = 16$. Therefore, out of the 4096 intensities, the intensity of a pixel (x, y) with RGB color [r,g,b] will be $I(x, y)$ as follows.

$$I(x,y) = R_c(x,y) * 16^2 + G_c(x,y) * 16^1 + B_c(x,y) * 16^0 \tag{3.12}$$

All the pixel in the frame can be processed in this manner to obtain the three-dimensional histogram. Once the three-dimensional histogram feature is extracted for every frame, we can proceed with the initial strict clustering.

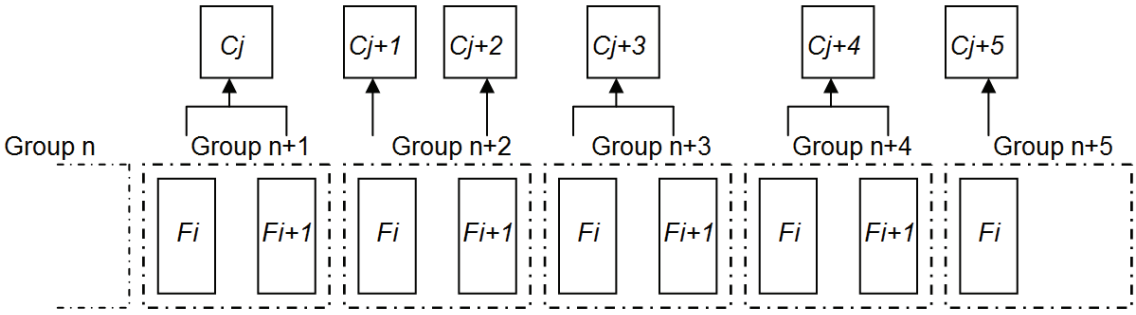


Figure 3.7: Strict Clustering using $N_{step} = 2$. F_is are the frames, C_js are clusters obtained using δ_1 as cut-off. Frames in one group for which $cos(\widetilde{v}_i, \widetilde{v}_{i+1}) \geq \delta_1$ are combined into one cluster (groups $n+1$, $n+3$, $n+4$), else are placed in different clusters (group $n+2$).

3.2.2 Strict Clustering

We consider the frames to be in distinct groups such at each group consists of at most N_{step} consecutive frames. The histograms of frames in one group are then arranged as columns of matrix A. Matrix A is subjected to singular value decomposition (SVD) to obtain $A = U\Sigma V^T$, where U and V are singular matrices and Σ a matrix of singular values (similar to that in section 3.1.3).

The rows of matrix $V\Sigma$ correspond to coordinates of the video frames in a M dimensional space, where $M = 4096$. Let us represent these coordinates by $\widetilde{v}_i$ for every i^{th} frame in the group such that $1 \leq i \leq N_{step}$. Cosine of the angle between the coordinates of similar frames is closer to 1 and that for dissimilar frames is closer to 0 [29]. In order to strictly group only the almost identical frames in one cluster we use a threshold δ_1 such that $\delta_1 \approx 1$. The cosine of the angle between coordinates of frames i and $i+1$ is given by,

$$cos(\widetilde{v}_i, \widetilde{v}_{i+1}) = \frac{(\widetilde{v}_i . \widetilde{v}_{i+1}^T)}{\|\widetilde{v}_i\| \, \|\widetilde{v}_{i+1}\|} \tag{3.13}$$

Only if $cos(\widetilde{v}_i, \widetilde{v}_{i+1}) \geq \delta_1$ is satisfied, i^{th} and $i + 1^{th}$ frames are merged into one cluster, else they are in different clusters. If the frames are merged into one cluster C_j then the cluster histogram $H(C_j)$ is calculated as follows.

$$H(C_j) = \begin{cases} h_i & \text{if } i = 1 \quad \forall f_i \in C_j \\ \\ \dfrac{H(C_j) + h_i}{2} & \text{if } i \neq 1 \quad \forall f_i \in C_j \end{cases} \tag{3.14}$$

where f_i is the i^{th} frame in cluster C_j. Three-dimensional histogram of frame f_i is represented by h_i. Now that the almost identical consecutive frames are in the same cluster, we can compare the cluster histograms for similarity. This will allow us to combine non-identical yet very similar frames into one cluster.

If we again use δ_1 to decide the cluster histogram similarity, only the nearly identical consecutive frames that were in different consecutive groups of N_{step} number of frames can be merged into one cluster. But, the frames that were judged to be non-identical (although they are similar as far as the shot is concerned) will never be merged into one cluster. On the other hand, if we use a different threshold δ_2 such that $\delta_2 < \delta_1$, both the almost identical frames as well as the similar consecutive frames can be combined into a single cluster. Also, if we directly use δ_2 for strict clustering instead of δ_1, there are chances that the non-similar frames are also combined into a single cluster, which would lead to false detection. Thus, the use of two different thresholds is justified. In the following subsection, we discuss the comparison and merging of similar clusters.

3.2.3 Histogram Comparison and Cluster Merging

Once we have the cluster histograms, we can iteratively compare and merge clusters in one group having similar histograms. The procedure for histogram comparison is the same as that used for strict clustering, with an exception that the threshold used is δ_2 (obtained experimentally) instead of δ_1 such that $\delta_2 < \delta_1$. The process of iterative comparison and merging of clusters in carried out until no further refinement is possible.

The histograms of clusters in one group are arranged as columns of matrix A.

Matrix A is subjected to SVD such that we again obtain $A = U\Sigma V^T$. The rows of matrix $V\Sigma$ now correspond to the coordinated of each cluster histogram. Let these rows be denoted by $\widetilde{c_j}$ for every cluster C_j in the respective group such that $1 \leq j \leq N_{step}$. The cosine of angle between the coordinates of cluster histograms will decide the similarity. The cosine is calculated similar to that in equation 3.13 as follows.

$$cos(\widetilde{c}_j, \widetilde{c}_{j+1}) = \frac{(\widetilde{c}_j . \widetilde{c}_{j+1}^T)}{\|\widetilde{c}_j\| \, \|\widetilde{c}_{j+1}\|} \tag{3.15}$$

The condition for merging clusters C_j and C_{j+1} is as follows.

$$cos(\widetilde{c}_j, \widetilde{c}_{j+1}) \geq \delta_2 \tag{3.16}$$

If equation 3.16 is not satisfied then the respective clusters are not merged. The process of comparing and merging clusters is iterative. If the number of clusters obtained in the current iteration is the same as that obtained in the previous iteration, then we deduce that no clusters have been combined in the current iteration. Therefore, any further iteration of comparison and merging of clusters will not cause any refinement. We then proceed to the shot detection step. If any refinement is possible, then the histograms of the modified clusters (C_{Mod}^k) are calculated for the next iteration as follows.

$$H(C_{Mod}^k) = \begin{cases} H(C_j) & \text{if } j = 1 \quad \forall C_j \in C_{Mod}^k \\ \\ \dfrac{H(C_{Mod}^k) + H(C_j)}{2} & \text{if } j \neq 1 \quad \forall C_j \in C_{Mod}^k \end{cases} \tag{3.17}$$

where k is the number of modified clusters. This iterative process will continue until no further refinement is possible i.e. k is equal to the number of clusters obtained in the previous iteration.

3.2.4 Shot Identification

The final clusters obtained after iterative comparison of clusters will consists of frames having similar intensity distribution with respect to adjacent frames in the same cluster. Frames in one shot exhibit a very small amount of variation in content with respect to the neighboring frames. Therefore, the intensity distribution will also vary in very small amount. In other words, the intensity distribution of these frames will be similar. With this observation, the clusters we obtained can be considered as final shots. But, we need to keep in mind that frames in transitions (like dissolve and fade transitions) are also having similar content as those of the neighboring frames. Since the content similarity in these transitional frames is not as much as that of frames in one shot, the transitional frames will form large number of clusters, each cluster having small number of frames. Therefore, what we obtained are not the final shots, but candidates for the actual shots present in the video. One can then identify the clusters having frames less than a certain minimum number and discard them from the set of final cluster. The remaining clusters are the different shots present in the video sequence.

By observing a large number of videos, one can find an average of smallest durations of a video shot transition. By knowing the frame rate (i.e. number of frames shown per second), the number of frames covered by the smallest transitions can be calculated. We set this number (δ_3) as the cut-off, such that, clusters having number of frames less than or equal to δ_3 are the frames in shot transition and therefore can be discarded. The remaining clusters give the different shots present in the video. Thus, only the clusters satisfying the following equation are considered as shots.

$$n(C_j) \geq \delta_3 \tag{3.18}$$

where C_j is the j^{th} cluster and $n(C_j)$ is the number of frames in the cluster C_j.

Discarding the clusters that do not satisfy the equation 3.18 takes care of elimi-

nating smooth transitions so as to avoid false detection. The abrupt transitions (like cuts) are taken care of initially by strict clustering followed by iterative comparison and merging of similar clusters (i.e. clusters having similar contents). Thus, this technique for shot detection can effectively detect the shot boundaries.

Summary

In this chapter we discussed our proposed techniques for video shot detection. In the dissolve detection based technique, we first extract the sequences of consecutive frames not in dissolve type transitions (SNDFs). Later, we iteratively merge the similar consecutive SNDFs, including the in between frames, until no further refinement is possible. By this, the number of misdetections are reduced. This techniques works with videos involving dissolve type shot transitions. For videos with both smooth as well as abrupt shot transitions, our strict clustering based technique starts with forming initial clusters such that each cluster includes only the almost identical consecutive frames. Later, we iteratively merge the similar consecutive clusters until no more merging is possible. Finally, we discard the clusters having number of frames less than a threshold value and consider all other clusters as the final shots.

Chapter 4

Implementation Methodology

The two proposed techniques for shot detection have been implemented according to the steps discussed in sections 3.1 and 3.2. The implementation has been done in Matlab [32]. To read the video sequence into Matlab and export the video shots, we used the videoIO toolbox [33]. For implementation of the two proposed techniques we started with feature extraction. This was followed by feature comparison to find the visual discontinuity for shot for shot boundary identification. The metrics used for performance evaluation of these algorithms are presented in the following section.

4.1 Methodology of Evaluation & Metrics

The basic measures for performance evaluation in shot detection (or simply detection in general) are the *recall* and *precision* values [5] [6] [34]. Recall is the measure of how much proportion of the correct entries are detected. Precision is the measure of how much of the detected entries are correct.

For calculating recall and precision, we need to have the set of actual correct and incorrect entries. Let D denote the correct detections (i.e. intersection of the set of entries detected as correct by the algorithm and the set of actual correct entries), MD denote the misdetections (i.e. intersection of the set of entries detected as incorrect by the algorithm and the set of actual correct entries) and FD denote the false detections (i.e.intersection of the set of entries detected as correct by the algorithm and the set

of actual incorrect entries). Recall and precision [5] [6] [34] [7] are then calculated as follows.

$$Recall = \frac{D}{D + MD} \tag{4.1}$$

$$Precision = \frac{D}{D + FD} \tag{4.2}$$

In the context of classification tasks, the terms true positives (TP), true negatives (TP), false positives (fTP) and false negatives (FN) are used to compare the given classification of an item (the class label assigned to the item by a classifier) with the desired correct classification (the class the item actually belongs to) such that,

- TN / True Negative: case was negative and predicted negative
- TP / True Positive: case was positive and predicted positive
- FN / False Negative: case was positive but predicted negative
- FP / False Positive: case was negative but predicted positive

then, recall and precision [35] are calculated as follows.

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4.4}$$

Precision and recall are set based measures. That is, they evaluate the quality of an unordered set of retrieved documents. The precision can be plotted against recall to visualize the performance of the retrieval technique. If we try to improve the

performance of a technique such that the number of correct detection increases, it may happen that some incorrect entries may also be detected as correct. Thus, if we try to improve the recall, the precision may drop. Similarly, if we try to reduce the number of false detections, some correct entries may as well be detected as incorrect entries, leading to increase in misdetections. Therefore, increasing the precision may reduce recall. Therefore, the retrieval algorithm should have performance such that both the recall and precision values are well balanced.

Since the evaluation is done at frame level, we used the following definitions of frame recall and frame precision.

$$frame\ recall = \frac{\#frames\ shared\ between\ detected\ and\ reference\ shots}{\#frames\ of\ reference\ shots} \tag{4.5}$$

$$frame\ precision = \frac{\#frames\ shared\ between\ detected\ and\ reference\ shots}{\#frames\ of\ detected\ shots} \tag{4.6}$$

For evaluating the performance of the proposed techniques we used *frame recall* and *frame precision* as metrics.

4.2 Experimental Setup

The experiments for the proposed techniques are done in Matlab using the videoIO toolbox for reading the video frames. The experimental setup for these techniques is as follows. The experiments have been performed on an Intel Core 2 Duo machine with 4GB RAM.

4.2.1 Dissolve Detection Based Shot Identification

This technique of shot detection is based on dissolve detection. Therefore, videos were carefully chosen for experiments such that most of the shot transitions were of dissolve type (viz. fade-in, fade-out and dissolve). These videos have been downloaded from

the popular video sharing website YouTube [36]. None of these videos were edited for the experimental purpose, except the fact that they had to be scaled to 240×352 resolution to maintain uniformity.

Ground truth entries are required for the calculation of recall and precision. As the ground truth for these videos was not available, we sought help of human volunteers to determine the locations of the shot boundaries. The shot boundaries determined in this manner are subjective as the observations might vary depending on each volunteer. We have therefore considered the average of the observations made by different volunteers as the final locations of the shot boundaries and have used them as the ground truth.

The various parameters used for the experiments are (a) Observation Window Size (b) δ_1 and (c) δ_2. To detect the dissolve type transitions we set the size of the observation window to 6 frames assuming the smallest dissolve duration in a 30 fps video is of 0.2 sec. The value of δ_1 is calculated dynamically according to the equation 3.5. We set the values of δ_2 in the range [0.75, 0.99] to experimentally observe the change in recall-precision values.

4.2.2 Strict Clustering Based Shot Detection

Most of the videos chosen for experimenting this technique have been downloaded from the popular video sharing website YouTube [36]. None of these downloaded videos were edited for experimental purpose except the fact that they had to be scaled to 240×320 resolution to maintain uniformity. The ground truth entries for these videos was not available, so sought help of human volunteers to determine the locations of the shot boundaries. Human observations being subjective, we considered the average of the observations made by different volunteers as the final locations of the shot boundaries and have used them as the ground truth.

Apart from the downloaded videos, we shot a few scenes and generated videos that can be classified broadly into (a) Indoor video (b) Outdoor and (c) High Action and (d) Moving Camera. We included both abrupt and smooth shot transitions manually

at appropriate locations in these videos. For this video editing task, we used the Kino Video Editor (an open source video editing software on Linux platform) [37]. Therefore, the ground truth for shots in these videos became available.

The various parameters used for the experiments are (a) N_{step} (b) δ_1 and (c) δ_2. The value of number of frames in a group (N_{step}) has been taken to be 2. This is because more the value of N_{step}, more will be the number of columns in the matrix to be decomposed, and more will be the time taken for performing SVD. The minimum value that can be chosen for N_{step} is 2. Since δ_1 is the threshold used in the strict clustering step, where we intend to include only the almost identical frames into one cluster, the value of δ_1 needs to be very near to 1. Therefore, we chose $\delta_1 = 0.99$. Finally, we set the values of δ_2 in the range [0.30, 0.90] to experimentally observe the change in recall-precision values.

4.3 Test Applications

The two proposed techniques have been implemented in Matlab. We created a library of the commonly used functions by the proposed techniques. The videos for which ground truth was not available and we considered the average of volunteers' observations as ground truth, a frame-by-frame inspection of the videos was done by the volunteers. To facilitate the frame-by-frame inspection of a video, we developed a video-player like application in which the user can control the frame rate, jump to a specific frame in the video etc which are the common operations required for a detailed inspection of the frames in the video.

4.3.1 Proposed Techniques

Following are the screen shots of the proposed techniques in execution.

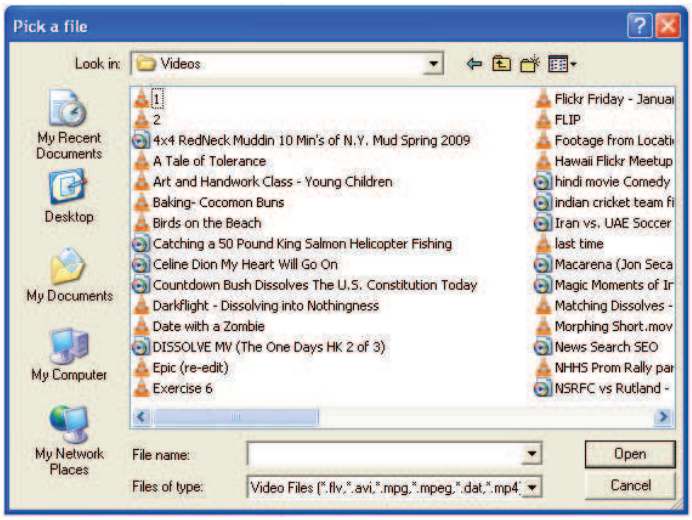


Figure 4.1: Video File Selector

```
1     %Dissolve Detection Based Shot Identification
2 -   clear all;
3 -   clc;
4 -   path=Fbrowser();
5 -   if (strcmp(path,'NULL')==0)
6
7         %-----------------------------------------------------------------------
8         %get video parameters
9 -       obj1 = videoReader(path);
10 -      numFrames=get(obj1,'numFrames');
11    %     numFrames=500;
12 -      fps=get(obj1,'fps');
13 -      seek(obj1,0);
14 -      [m n o] = size(getframe(obj1));
15 -      if(m>n)
16 -          m=352;n=240;
17 -          ResScale=[352 240];
18 -      else
19 -          m=240;n=352;
20 -          ResScale=[240 352];
21 -      end
22        %-----------------------------------------------------------------------
23
24        %-----------------------------------------------------------------------
25        %Set Initial Parameters
26 -      WinSz = 6; %Window size
27 -      L=WinSz-1;
28 -      sframe=0; %A Video always starts from 0 frame
29 -      Nf=[]; %This will store number of pixels changing atleast once in window f
30 -      Prop=[]; %This will store number of Proponent pixels for all windows
31 -      Diso=zeros(numFrames-L,1); %This will store 1 if a dissolve is detected at
32 -      minN=m*n;
33 -      HistMat=sparse(256,numFrames);
34 -      Delta1=0.90;
35        %-----------------------------------------------------------------------
```

Figure 4.2: Matlab file for Dissolve Detection Based Technique

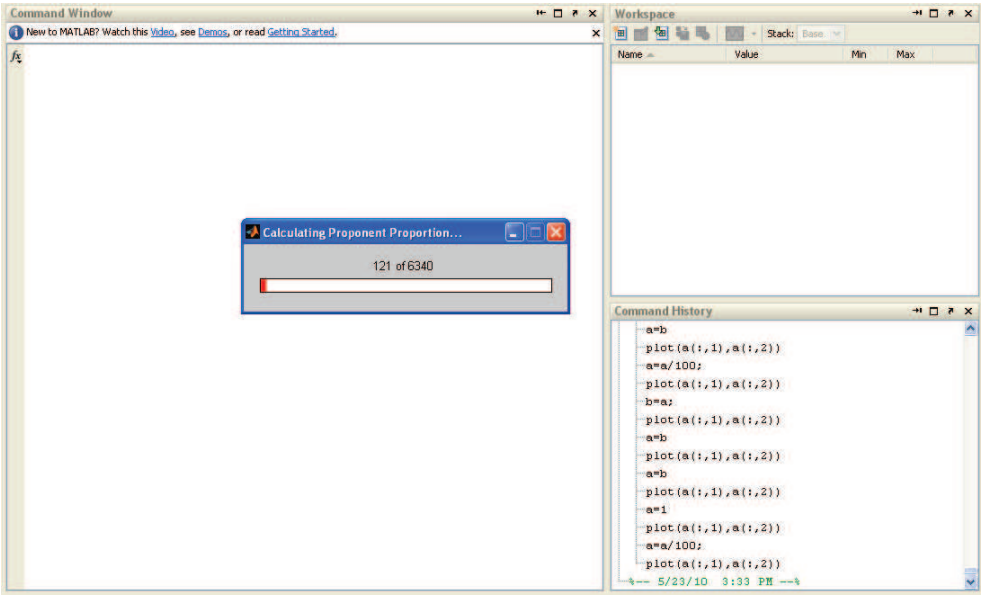


Figure 4.3: Calculation of Proponent Pixels

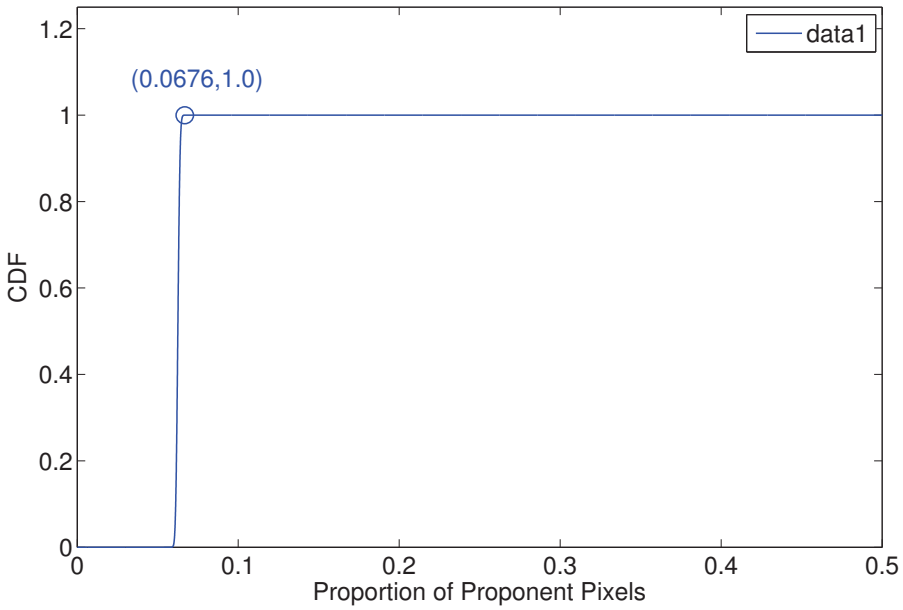


Figure 4.4: CDF for calculation of δ_1 in the dissolve detection based technique

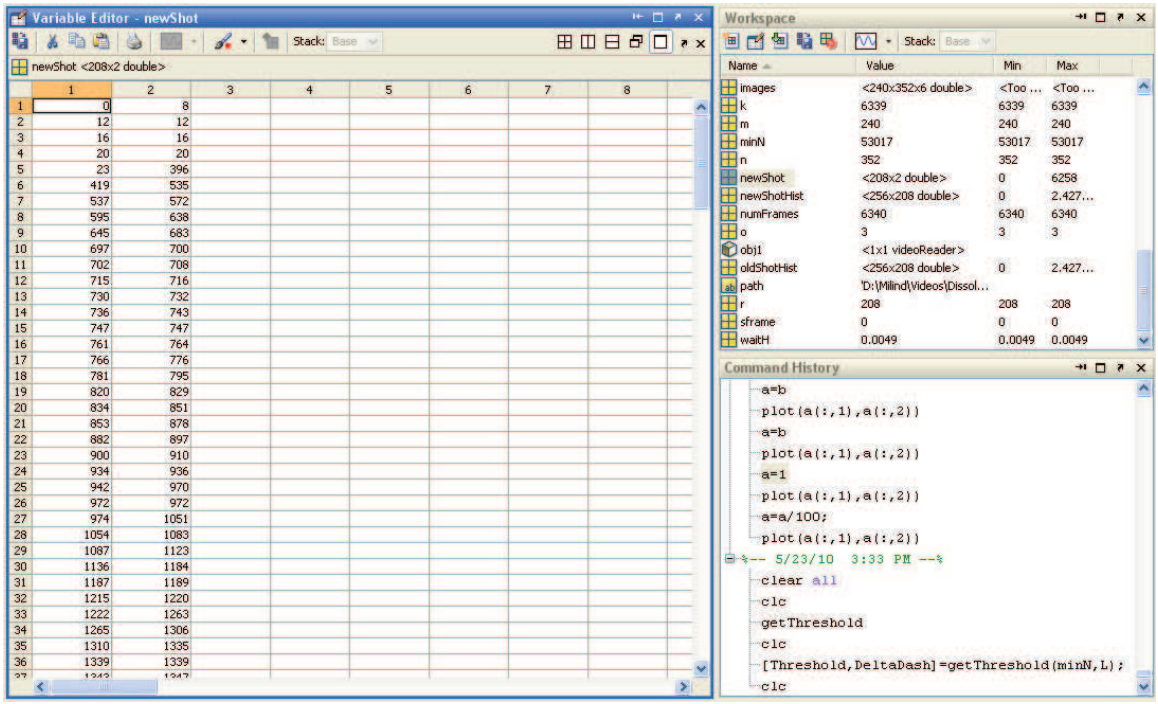


Figure 4.5: Resulting Shot Locations (frame numbers) using Dissolve Detection Based Technique. Column 1 denotes shot start and column 2 denotes shot end

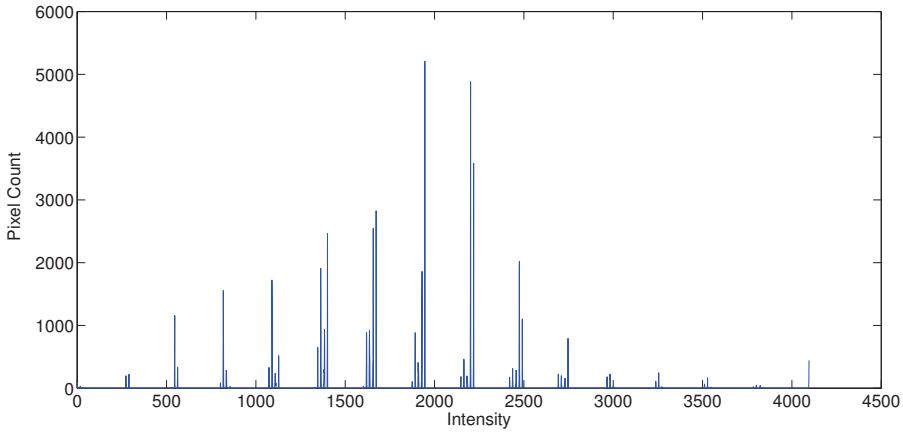


Figure 4.6: Three-dimensional histogram (16 bins) of a frame (8-bit)

```
%Code for Strict Clustering Based Technique
clear all;
close all;
clc;
path=Fbrowser();
if(strcmp(path,'NULL')==0)
    %-------------------------------------------------------------------------
    %Get the video and set initial parameters
    vidobj=videoReader(path);
    numFrames=get(vidobj,'numFrames');
%      numFrames=500;
    seek(vidobj,0);
    img1=getframe(vidobj);
    [m,n,o]=size(img1);
    if(m>n)
        m=320;
        n=240;
    else
        m=240;
        n=320;
    end
    reScale=[m n];
    Step=2;
    oldnumFrames=numFrames;
    rem=mod(numFrames,Step);
    HistMat=[];
    Clust=[];
    FrmHist=sparse([]);
    delta1=0.99;
    delta2=0.60;
    %-------------------------------------------------------------------------

    %-------------------------------------------------------------------------
    %Get Histogram of all the frames
    waitH = waitbar(0,'1','Name','Calculating 3D Histogram of frames');
    for i=1:numFrames
```

Figure 4.7: Matlab file for Strict Clustering Based Technique

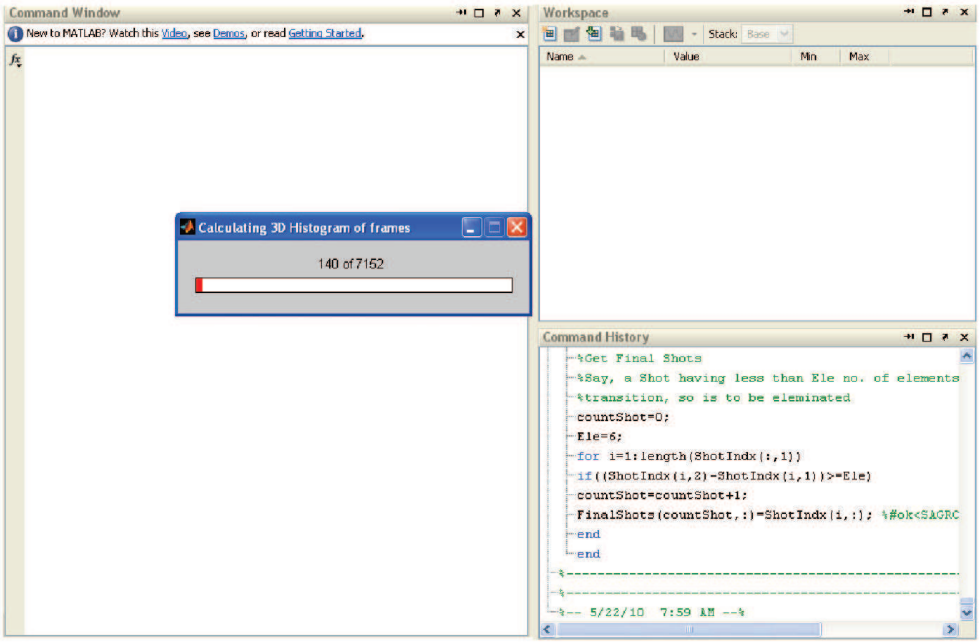


Figure 4.8: Three-dimensional histogram calculation

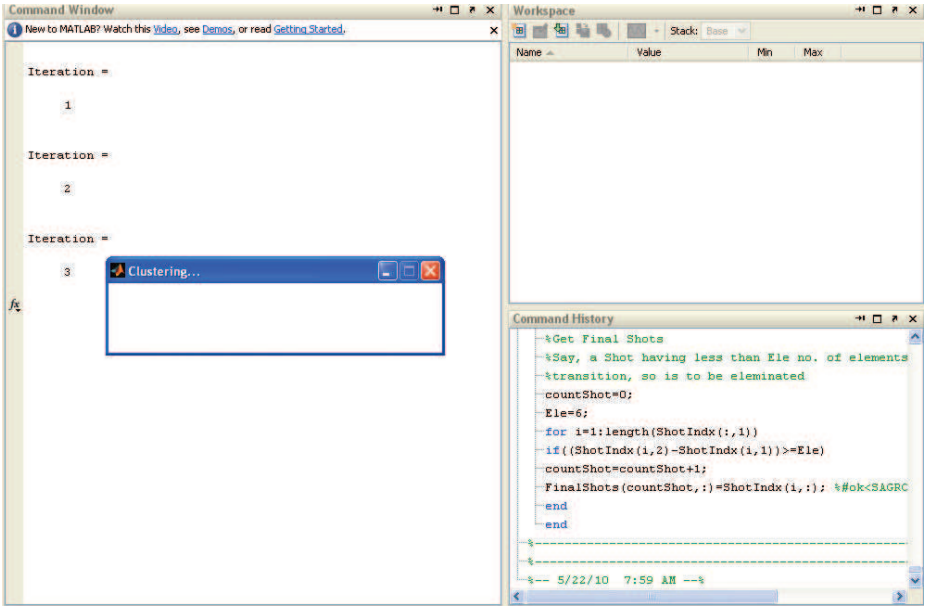


Figure 4.9: Iterative Clustering

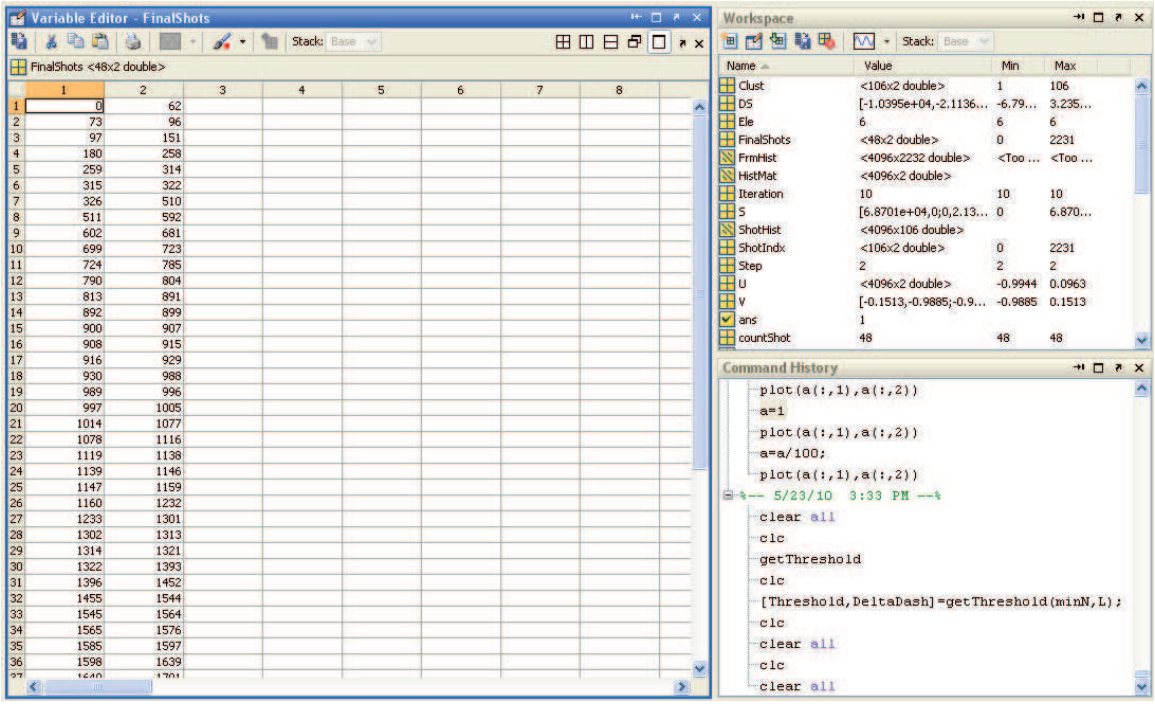
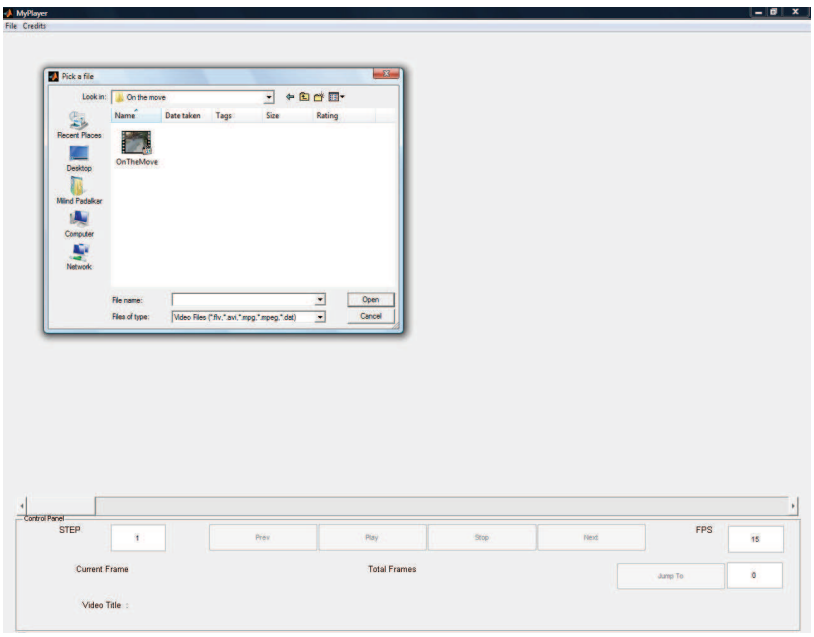


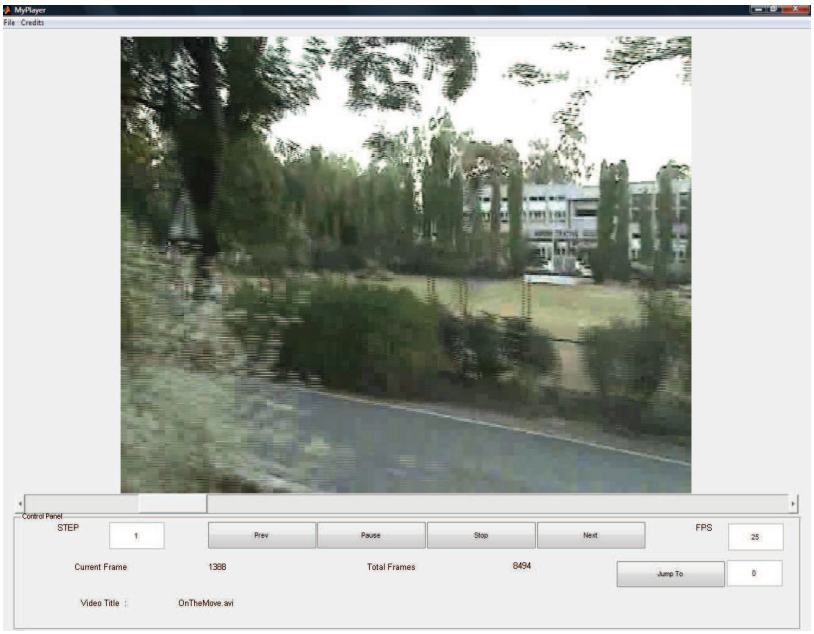
Figure 4.10: Resulting Shot Locations (frame numbers) using Strict Clustering Based Technique. Column 1 denotes shot start and column 2 denotes shot end

4.3.2 Video Inspection Tool

The screen shots of the video inspection tool for frame-by-frame inspection are shown in figure 4.11(a). Using this tool one can do a frame by frame inspection of the video. Both forwarding and reversing the video, setting frame-rate, jumping to a desired frame can be done using this tool. The current frame number, total number of frames in the video and the file name are visible in the Control Panel.



(a)



(b)

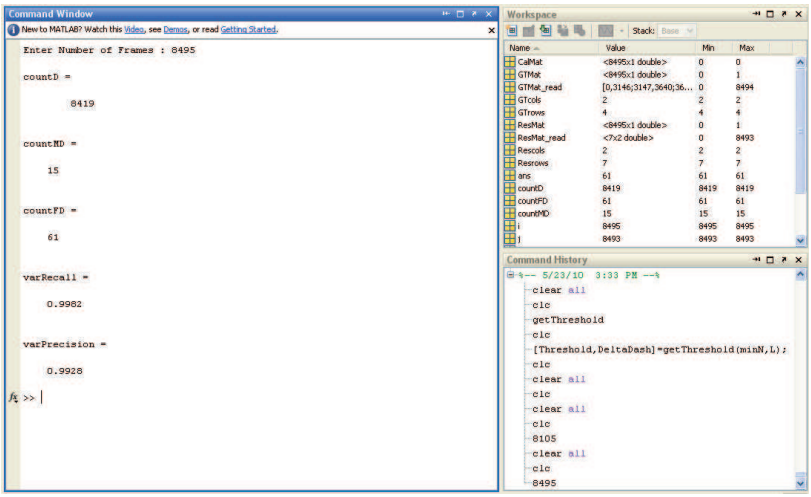
Figure 4.11: Video Inspection Tool

4.3.3 Precision-Recall Calculator

Following are the screen shots of the precision-recall calculator.

```
%Program for Comparison (To calculate Precision & Recalll calulation)
close all;
clear all;
clc;
path1=Fbrowser('Select Ground Truth File');
path2=Fbrowser('Select Implementation Result File');

if((isequal(path1,'NULL')==0) && (isequal(path2,'NULL')==0))

    %-------------------------------------------------------------------------
    %Read Shot durations from the input file
    GTMat_read=xlsread(path1, -1);
    ResMat_read=xlsread(path2, -1);
    %-------------------------------------------------------------------------

    %-------------------------------------------------------------------------
    %Initialize Matrices
    numFrames=str2double(input('Enter Number of Frames : ','s'));
    GTMat=zeros(numFrames,1);
    ResMat=zeros(numFrames,1);
    CalMat=zeros(numFrames,1);
    [GTrows,GTcols]=size(GTMat_read);
    [Resrows,Rescols]=size(ResMat_read);
    %-------------------------------------------------------------------------

    %-------------------------------------------------------------------------
    %Assign a value 1 to all the frames which are part of some shot
    for i=1:GTrows
        for j=GTMat_read(i,1):GTMat_read(i,2)
            GTMat((j+1),1)=1;
        end
    end
    %-------------------------------------------------------------------------
```

(a)



(b)

Figure 4.12: Precision-Recall Calculator. (a) Matlab file of the calculator. (b) Metric values for an input video

Chapter 5

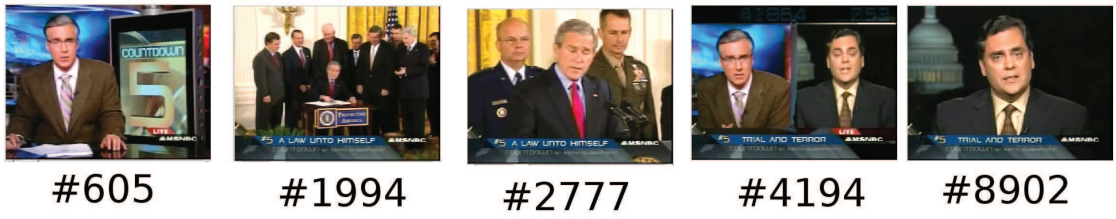
Performance Results And Analysis

The performance of the proposed techniques is measured in terms of recall and precision values discussed in section 4.1. The performance of the strict clustering based technique is compared with the existing histogram based techniques for shot detection proposed in [29] and [30]. For the dissolve detection based shot identification using SVD technique, we compare the performance with method of shot identification using only dissolve detection [7]. In the following sections we discuss the performance results of the proposed techniques.

5.1 Dissolve Detection Based Technique

Video shots are separated by various transitions. Therefore, by detecting the transitions one can identify the shot boundaries. We therefore used the dissolve detection technique proposed in [7] and tried to detect the shot boundaries. Observation was, this method was causing over-segmentation. Pixel intensity can change many times within a shot. Also, within a clip, the variation in background causes change in pixel intensity from frame to frame. Thus, using only pixel intensity variation in an observation window as a cue for shot detection proved insufficient.

The experimental setup used for this technique is discussed earlier in section 4.2.1. The videos we chose for the experiments exhibit various characteristics. The video Mud Race involves speedily moving objects and also consists of zooming effects. The



(a) Bush TV

(b) Art Hand

(c) Mud Race

(d) Iran-UAE

Figure 5.1: Sample videos used for experiments

videos Zombie Date, Dissolve MV and Last Time are short films involving focus switching from one person to another at regular time intervals. The video Hawaii Flickr is having relatively longer shot durations. Chittorgarh is a presentation involving still images and very short videos with moving camera. The video Bush TV is a news report.

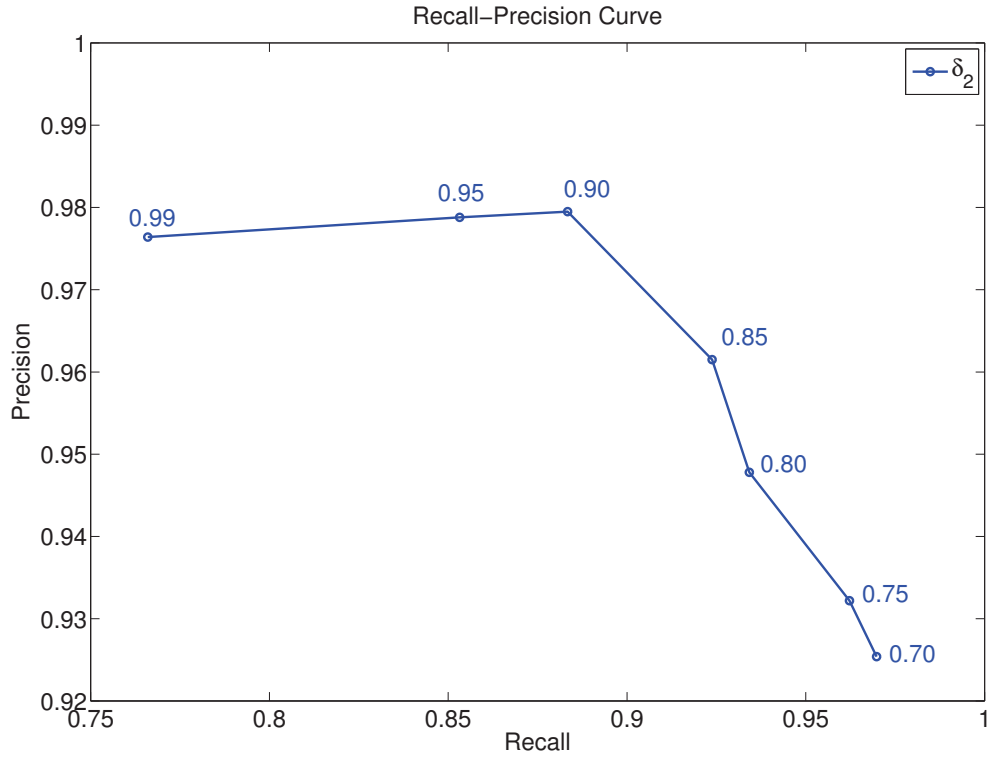


Figure 5.2: Recall-Precision Curve by varying the value of δ_2

For SNDF-merging, we varied the threshold δ_2 in the range $[0.70, 0.99]$ and noted the results to evaluate the performance of our algorithm. The obtained recall-precision curve is shown in Fig. 5.2. Using a low threshold value (close to 0.70) allowed some frames, which were not actually a part of shot but at the boundary of the shot, to be declared as a part of the shot. Therefore, along with frames actually present in the shot, a few frames not belonging to the shot were also declared to be a part of shot, causing high recall value but low precision value. Similarly, if a high threshold value (close to 0.99) is used, the merging SNDFs having similar intensity distribution

becomes very strict, causing some frames which are actually a part of the shot to be missed out from being declared as a part of the shot. Thus, the precision value increases, but at the same time, the recall value drops. We observed that for most of the videos, using $\delta_2 = 0.9$ had a fair trade-off between the recall and precision values. Using other values for δ_2, either recall or precision rises, but at the same time the other drops to an unacceptable level. Hence, the value of threshold δ_2 has been set to 0.9. Using this value we summarize the performance of our technique in comparison with using only the dissolve detection method (MTDD) for shot detection in table 5.1.

This technique is designed to detect shot boundaries separated by smooth dissolve type transitions. If it is used to detect only the abrupt transitions like cuts, then, since no dissolve type transitions are present, the complete video will be detected as single SNDF. Therefore, the there will be no refinement by the SNDF histogram comparison and merging process. Hence, the complete video will be detected as a single shot.

Videos	**#Frames**	**MTDD**		**Proposed1**	
		Recall (%)	**Precision** (%)	**Recall** (%)	**Precision** (%)
Mud Race	15866	75.66	97.15	96.09	97.36
Chittorgarh	7152	88.59	98.97	91.99	98.51
Bush TV	12820	89.91	99.12	96.42	99.01
Zombie Date	9967	96.34	92.02	98.62	91.87
Dissolve MV	7236	88.15	83.49	94.84	83.46
Hawaii Flickr	6430	71.38	97.48	88.34	97.95
Last Time	2050	99.56	80.23	99.81	80.23

Table 5.1: Performance of Dissolve Detection Based proposed technique in terms of Recall and Precision

For videos having dissolve type transitions, this technique is able to reduce the misdetection caused due to the original shots being detected as a series of smaller video sequences (for example, figure 5.3), when using only the dissolve detection method without SNDF merging (MTDD). Thus, the proposed technique is able to overcome the problem of over-segmentation. Our experimental results presented in Table 5.1 show that our algorithm is indeed able to reduce the misdetections as compared to

Figure 5.3: Experimental video sequence (Hawaii Flickr) with respective frame numbers. Proposed algorithm detects all these frames to be in a single video shot. Shot boundary detection using only dissolve detection, identifies the frames 419, 450, 537 & 570 to be in distinct shots, whereas frames 490 & 515 are misdetected.

the technique MTDD. We are able to obtain higher values of recall while maintaining the precision. The experimental results prove that our proposed technique is indeed effective.

5.2 Strict Clustering Based Technique

Video shots separated by dissolve type transitions can be effectively detected using our dissolve detection based proposed technique. For videos having both abrupt and smooth shot transitions, the shot boundary detector should be able to identify both, the drastic as well as smooth changes in the visual content. The techniques proposed in [29] and [30] work well for videos with both, the smooth as well as hard shot transitions.

In our proposed Strict Clustering based technique we have adopted the use of Singular Value Decomposition (SVD) for finding content similarity in consecutive frame, which has been used in both [29] and [30]. The experimental setup is discussed earlier in section 4.2.2. The videos chosen for experiments have various characteristics. The video Last Time is a short movie involving focus switching from one person to another at regular time intervals. The videos Snowboarding, Magic Moments, Iran vs UAE are sports videos involving high action and moving camera. Sr-71 is a short

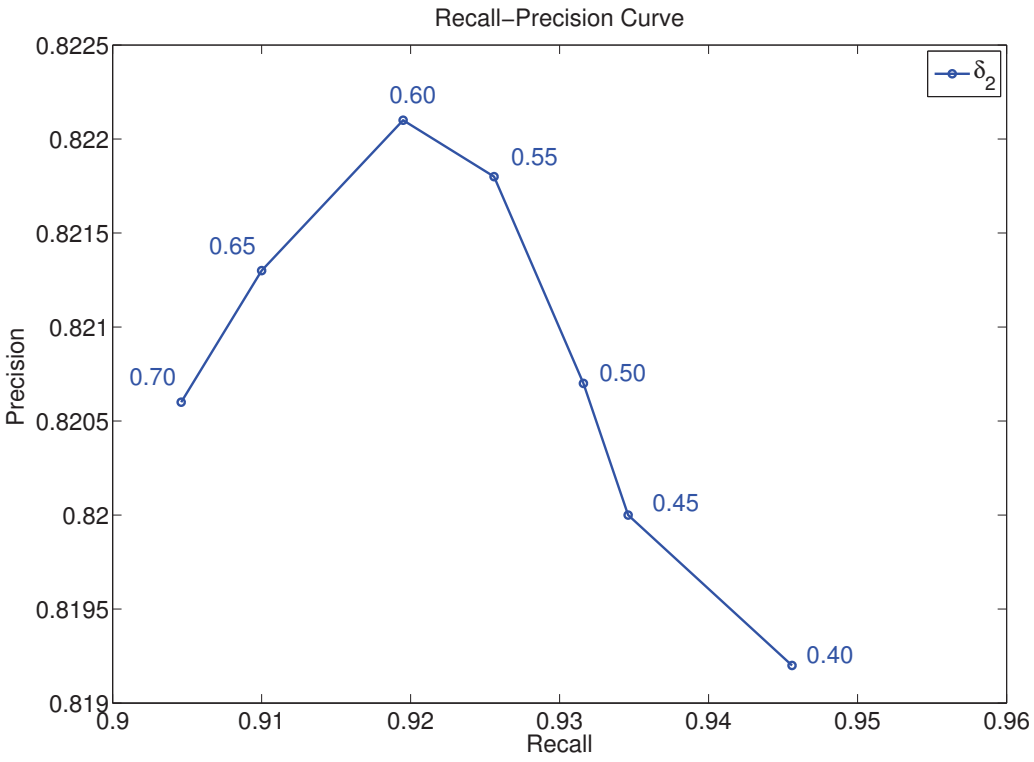


Figure 5.4: Recall-Precision Curve by varying the value of δ_2 (low action videos)

documentary involving moving camera. The videos Tale of Tolerance, Art & Handwork, Chittorgarh involve less camera movement and many dissolve type transitions. The videos On The Move and Indoor were shot and edited for experimental purpose. These involve both abrupt and smooth shot transitions.

As mentioned earlier, we varied the value of δ_2 in the range $[0.30, 0.90]$ and observed the results. For videos with low action, the obtained recall-precision curve is shown in figure 5.4. Using lower value of threshold (close to 0.30) proved to be a very loose criteria to allow merging of clusters having similar intensity distribution. As a result, clusters having a very small amount of similarity in the intensity distribution were merged, contributing to higher recall value. This also caused the adjacent clusters having very small number of frames to be combined into a single cluster having enough number of frames. These clusters were not discarded in the final step, causing large

number of false detections and in turn contributing to smaller value of precision. Similarly, using a higher value of threshold (close to 0.90) became a too strict criteria for merging clusters having similar intensity distribution. This caused many similar clusters (that were actually a part of a single shot) not to be merged together, leading to lower value of recall. Here, the precision value initially goes on increasing because the detected locations of the shots become more and more closer to the actual locations of the shots. Later, the value of threshold drops such that false detections occur, causing the drop in the value of precision. For most of these videos, the value of $\delta_2 = 0.60$ showed a fair compromise of the recall and precision values. We therefore set the value of $\delta_2 = 0.60$ for such videos.

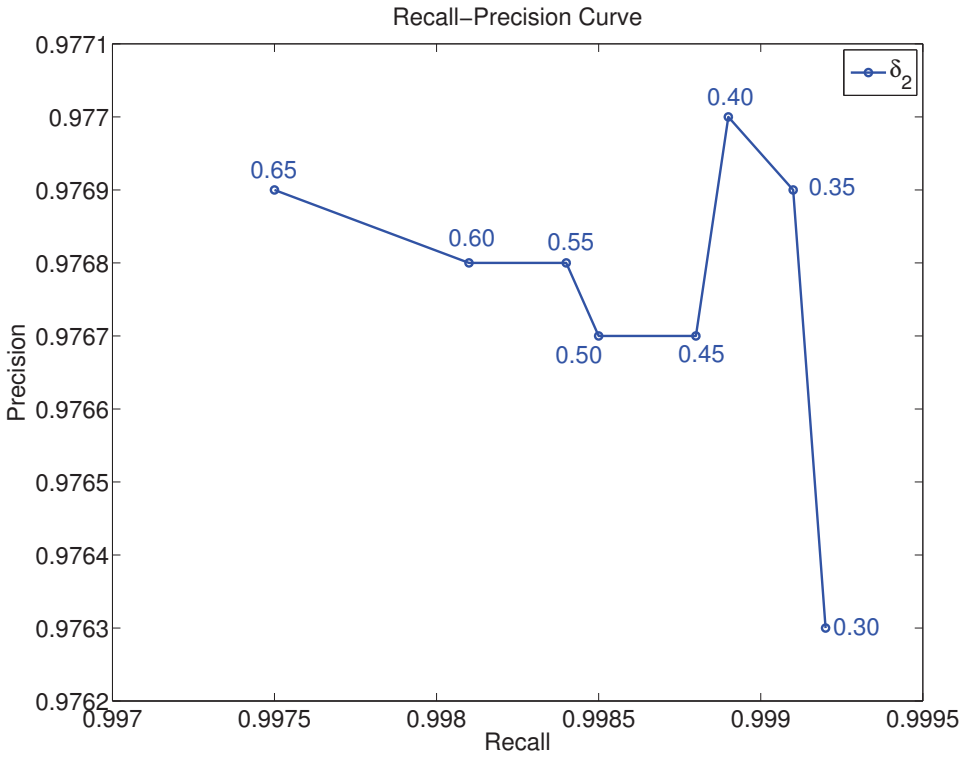


Figure 5.5: Recall-Precision Curve by varying the value of δ_2 (high action videos)

However, for videos involving high action, a lower value of δ_2 was required, because, the change in the intensity distribution for such videos would be large in consecutive

frames, and setting higher threshold for comparison would lead to misdetections. Here, in the lower range of values $[0.30, 0.65]$, we observe that as the threshold goes on decreasing, the value of recall goes on increasing allowing merger of similar clusters. We observed the variation in recall-precision values for a number of videos by varying the threshold δ_2. The value $\delta_2 = 0.4$ was found to be optimal to maximize both recall and precision. The obtained curve using various values of δ_2 for such a video is shown in figure 5.5.

(a)

Videos	**#Frames**	**Proposed2**		**OSBD**		**VSS-SVD**	
		Recall (%)	**Precision** (%)	**Recall** (%)	**Recall** (%)	**Precision** (%)	**Precision** (%)
Tolerance	11685	100	75.75	100	75.59	99.97	75.21
Chittorgarh	7152	100	94.03	100	94.01	100	93.81
Art-Hand	2232	100	81.68	99.32	81.35	100	80.42
Last Time	2050	100	78.43	100	77.93	100	77.32
Indoor	1244	91.91	92.52	93.27	94.01	92.52	93.81

(b)

Videos	**#Frames**	**Proposed2**		**OSBD**		**VSS-SVD**	
		Recall (%)	**Precision** (%)	**Recall** (%)	**Recall** (%)	**Precision** (%)	**Precision** (%)
Magic Mts	8105	94.56	81.92	91.59	82.52	92.23	82.52
Sr-71	13842	99.89	97.70	99.22	97.70	99.72	99.74
Iran-UAE	8586	99.92	97.52	99.57	97.51	99.67	97.52
On move	8495	99.82	99.28	99.63	99.26	99.22	99.28
Snowboard	18736	99.88	97.80	99.63	97.97	99.01	97.78

Table 5.2: Performance of Strict Clustering Based proposed technique in terms of Recall and Precision. (a) Low action videos ($\delta_2 = 0.60$), (b) High action videos ($\delta_2 = 0.40$)

Our proposed technique is based on singular value decomposition (SVD) of the matrix formed using the three-dimensional histogram feature. The computation time for an SVD of a $m \times n$ matrix A assuming that $m >> n$ as given in [38] is,

- Computation of U, V and D: $4m^2n + 8mn^2 + 9n^3$
- Computation of V and D: $4mn^2 + 8n^3$

The technique used in [29] considers all the video frames to form the matrix A. Therefore, as the number of frames in the video increase, the computation time of

Videos	#Frames	Avg. approx. time taken (min)		
		Proposed2	OSBD	VSS-SVD
Snowboard	18736	32	45	94
Tolerance	11685	19	26	46
On move	8495	18	37	145
Chittorgarh	7152	8	10	10
Art-Hand	2232	3	4	7
Last Time	2050	3	4	7
Indoor	1244	2.5	5.5	21.5
Sr-71	13842	13	15	10
Iran-UAE	8586	9	11	7
Magic Mts	8105	10	9.5	5

Table 5.3: Execution time of different techniques for various videos

the SVD increases exponentially. The size of matrix A in our proposed algorithm is controlled by N_{step}. By setting $N_{step} = 2$, the dimension of matrix A is reduced to $m \times 2$ from $m \times n$, where n is the number of frames in the video. Thus, the computation time of calculating the SVD is reduced to,

- Computation of U, V and D: $8m^2 + 32m + 72 \equiv O(m^2)$
- Computation of V and D: $16m + 64 \equiv O(m)$

We then need to compute the SVD (n/N_{step}) times i.e. $(n/2)$ times. Thus the complexity of clustering the frames becomes $O(mn)$, which will reduce the computation time significantly.

From table 5.2 it is clear that our proposed technique is able to perform as effectively as the techniques OSBD [30] and VSS-SVD [29], with slightly higher recall and precision values in some cases. Moreover, from table 5.3 we can clearly see that our technique runs significantly faster than OSBD and VSS-SVD which justifies the computational time calculated above. Thus, our proposed technique in indeed efficient.

Chapter 6

Conclusion And Future Work

Video shot detection is in important step in the video indexing, retrieval and summarization applications. This report presents two novel techniques for video shot detection based on (a) dissolve detection and (b) strict initial clustering of the video frames, both using singular value decomposition. The conclusion and future scope of our work is as follows.

6.1 Conclusion

Video shot detection can be done by modelling and detecting the shot transitions. The abrupt transitions are easy to detect as the visual discontinuity can be clearly observed. Smooth transitions are however difficult to detect as the change in visual contents of the frames in such transitions is very small. Identifying shot boundaries by detecting the dissolve type shot transitions only results in over-segmentation of the video causing large number of smaller video sequences (most of these belonging to the same shot) to be detected as individual shots. Our first approach is able to overcome the problem of over-segmentation by comparing the histograms and merging the respective sequences of non dissolve frames. By this we are able to achieve improved performance in terms of recall values while maintaining the precision.

This technique works well with videos having most shot transitions of dissolve type. For videos having many abrupt shot transitions, this technique will not have

any dissolve transitions to detect. Therefore, such videos will be detected as single shot. Our second proposed technique based on strict initial clustering is able to detect the shots having both abrupt as well as smooth transitions, where, we strictly combined only almost identical and consecutive frames in a single cluster, so that the abrupt changes in visual flow get detected initially. Later, we iteratively calculate and compare the three-dimensional histograms of these clusters so as to merge the similar clusters. By controlling the size of each cluster, we are able to control the size of the feature matrix which helps in significantly reducing the execution time when the video consists of large number of frames. The results show our technique is efficient and runs faster in comparison with other techniques.

6.2 Future Work

Videos can be classified into various categories based on the environment and context. Developing a common shot detection system for all the categories is very difficult as each will be having a distinct property. In the dissolve detection based technique we narrowed down to only videos having the dissolve type shot transitions. Our strict clustering based technique works well for videos having both abrupt as well as smooth shot transitions. In the dissolve detection based technique, applying strict clustering on the initially detected sequences of non-dissolve frames (SNDFs), we will be able to separate the consecutive frames having abrupt change in content. In future, this can be adopted to make our dissolve detection based technique more robust for videos also having abrupt shot transitions along with dissolve type transitions. We have used the histogram feature for frame content comparison. Other global features of the frames or group of frames can be used depending on the application under consideration. The features extracted for the MPEG-7 descriptions can be also used as these will also be used by the indexing, summarization and retrieval applications.

Our Publications

1. M. G. Padalkar and M. A. Zaveri, "Dissolve Detection Based Shot Identification Using Singular Value Decomposition," in *2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation (AMS2010)*, Kota Kinabalu (Malaysia), May 2010, pp. 312-316. (Presented)

Bibliography

[1] H. Zhang and J. H. Smoliar, Stephen W.and Wu, "Content-based video browsing tools," in *Proc. SPIE, Multimedia Computing and Networking*, vol. 2417, 1995, pp. 389–398.

[2] P. Bouthemy, C. Garcia, R. Ronfard, G. Tziritas, E. Veneau, and D. Zugaj, "Scene segmentation and image feature extraction for video indexing and retrieval," in *VISUAL '99: Proceedings of the Third International Conference on Visual Information and Information Systems*, 1999, pp. 245–252.

[3] A. Doulamis, N. Doulamis, and S. Kollias, "A fuzzy video content representation for video summarization and content-based retrieval," *SP*, vol. 80, no. 6, pp. 1049–1067, June 2000.

[4] X. Mu, "A content-based video browsing system based on visual neighbor similarity," in *JCDL '06: Proceedings of the 6th ACM/IEEE-CS joint conference on Digital libraries*, 2006, pp. 373–373.

[5] U. Gargi, R. Kasturi, and S. Strayer, "Performance characterization of video-shot-change detection methods," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 10, no. 1, pp. 1 –13, feb 2000.

[6] C. Cotsaces, N. Nikolaidis, and I. Pitas, "Video shot detection and condensed representation. a review," *Signal Processing Magazine, IEEE*, vol. 23, no. 2, pp. 28 –37, march 2006.

[7] C.-W. Su, H.-Y. Liao, H.-R. Tyan, K.-C. Fan, and L.-H. Chen, "A motion-tolerant dissolve detection algorithm," *Multimedia, IEEE Transactions on*, vol. 7, no. 6, pp. 1106 – 1113, dec. 2005.

[8] A. Hanjalic, "Shot-boundary detection: unraveled and resolved?" *CirSysVideo*, vol. 12, no. 2, pp. 90–105, February 2002.

[9] T. Sikora, "The mpeg-7 visual standard for content description-an overview," *CirSysVideo*, vol. 11, no. 6, pp. 696–702, June 2001.

[10] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society of Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[11] A. Hanjalic, *Content-Based Analysis of Digital Video*, 1st ed. Springer, July 2004.

[12] R. Lienhart, "Comparison of automatic shot boundary detection algorithms," in *Storage and Retrieval for Image and Video Databases*, no. SPIE 3656, January 1999, pp. 290–301.

[13] J. Boreczky and L. Rowe, "Comparison of video shot boundary detection techniques," *JEI*, vol. 5, no. 2, pp. 122–128, April 1996.

[14] H. Jiang, A. Helal, A. K. Elmagarmid, and A. Joshi, "Scene change detection techniques for video database systems," *Multimedia Syst.*, vol. 6, no. 3, pp. 186–195, 1998.

[15] C. O'Toole, A. F. Smeaton, N. Murphy, and S. Marlow., "Evaluation of automatic shot boundary detection on a large video test suite," in *CIR'99 - The Challenge of Image Retrieval: 2nd UK Conference on Image Retrieval*, 1999, pp. 1–12.

[16] A. Pardo, "Simple and robust hard cut detection using interframe differences," in *CIARP*, 2005, pp. 409–419.

[17] O. Urhan, M. K. Güllü, and S. Ertürk, "Modified phase-correlation based robust hard-cut detection with application to archive film," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 16, no. 6, pp. 753–770, 2006.

[18] S. M. M. Tahaghoghi, H. E. Williams, J. A. Thom, and T. Volkmer, "Video cut detection using frame windows," in *ACSC '05: Proceedings of the Twenty-eighth Australasian conference on Computer Science*, 2005, pp. 193–199.

[19] S. Porter, M. Mirmehdi, and B. Thomas, "Video cut detection using frequency domain correlation," *Pattern Recognition, International Conference on*, vol. 3, p. 3413, 2000.

[20] O. D. Robles, P. Toharia, A. Rodrguez, and L. Pastor, "Automatic video cut detection using adaptive thresholds," in *Proceedings of the Fourth IASTED International Conference on Visualization, Imaging and Image Processing*, sep 2004, pp. 517–522.

[21] T. Barbu, "Novel automatic video cut detection technique using gabor filtering," *Comput. Electr. Eng.*, vol. 35, no. 5, pp. 712–721, 2009.

[22] R. Dugad, K. Ratakonda, and N. Ahuja, "Robust video shot change detection," in *Multimedia Signal Processing, 1998 IEEE Second Workshop on*, 7-9 1998, pp. 376 –381.

[23] Z. Cernekova, C. Nikou, and I. Pitas, "Shot detection in video sequences using entropy based metrics," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 3, june 2002, pp. III–421 – III–424 vol.3.

[24] M. L. Cooper and J. Foote, "Scene boundary detection via video self-similarity analysis," in *ICIP (3)*, 2001, pp. 378–381.

[25] M. Cooper, J. Foote, J. Adcock, and S. Casi, "Shot boundary detection via similarity analysis," in *in Proceedings of the TRECVID 2003 Workshop*, 2003, pp. 79–84.

[26] Y. Ohta, T. Kanade, and T. Sakai, "Color information for region segmentation," *Computer Graphics and Image Processing*, vol. 13, no. 1, pp. 222 – 241, July 1980.

[27] I. Radev, G. Paschos, N. Pissinou, and K. Makki, "Video content representation based on texture and lighting," in *VISUAL '00: Proceedings of the 4th International Conference on Advances in Visual Information Systems*, 2000, pp. 457–466.

[28] W. Zhao, J. Wang, D. Bhat, K. Sakiewicz, N. Nandhakumar, and W. Chang, "Improving color based video shot detection," in *ICMCS '99: Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, 1999, p. 752.

[29] Z. Cernekova, C. Kotropoulos, and I. Pitas, "Video shot segmentation using singular value decomposition," in *Multimedia and Expo, 2003. ICME '03. Proceedings. 2003 International Conference on*, vol. 2, july 2003, pp. II – 301–4 vol.2.

[30] W. Abd-Almageed, "Online, simultaneous shot boundary detection and key frame extraction for sports videos using rank tracing," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, oct. 2008, pp. 3200 –3203.

[31] M. Tian, S.-W. Luo, and L.-Z. Liao, "An investigation into using singular value decomposition as a method of image compression," in *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, vol. 8, 18-21 2005, pp. 5200 –5204.

[32] "MathWorks Matlab." [Online]. Available: http://www.mathworks.com/

[33] "videoIO Toolbox for Matlab," 2010. [Online]. Available: http://sourceforge.net/projects/videoio/

[34] N. Manickam, A. Parnami, and S. Chandran, "Reducing false positives in video shot detection using learning techniques," in *Indian Conference on Computer Vision, Graphics and Image Processing*, vol. 4338, 2006, pp. 421–432.

[35] "Precision and Recall," 2010. [Online]. Available: http://en.wikipedia.org/wiki/Precision_and_recall

[36] "YouTube - Broadcast Yourself," 2009. [Online]. Available: http://www.youtube.com/

[37] "Kino Video Editor," 2009. [Online]. Available: http://www.kinodv.org/

[38] "Singular Value Decomposition," 2008. [Online]. Available: http://campar.in.tum.de/twiki/pub/Chair/TeachingWs05ComputerVision/3DCV_svd_000.pdf