2004

Adaptive video segmentation

Nagamani Banda
West Virginia University

Adaptive Video Segmentation

Nagamani Banda

Thesis submitted to the

College of Engineering and Mineral Resources

at West Virginia University

in partial fulfillment of the requirements

for the degree of

Master of Science

in

Computer Science

Donald A. Adjeroh, Ph.D, Assistant Professor

Mark A. Jerabek, Ph.D, Associate Professor

Vasudevan Jagannathan, Ph.D, Associate Professor

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia

2004

Keywords: Video indexing, Edge-based features, Multi-resolution framework, Adaptive features, Adaptive thresholds

ABSTRACT

Adaptive Video Segmentation

Nagamani Banda

With the explosive growth of video data in digital libraries and other repositories, there is an increasing need for robust indexing techniques which can provide faster access to the desired information content. The efficiency of a video indexing technique depends on the efficiency of the video segmentation algorithm which is a fundamental step in video indexing. Video segmentation is a process of splitting up a video sequence into its constituent scenes. This work focuses on the problem of video segmentation. A content-based approach has been used which segments a video based on the information extracted from the video itself. The main emphasis is on using structural information in the video such as edges as they are largely invariant to illumination and motion changes. The edge-based features have been used in conjunction with the intensity-based features in a multi-resolution framework to improve the performance of the segmentation algorithm. Higher resolutions provide information about the localized structures in the video frames and hence capture the minute variations from one frame to the other frame. The multi-resolution edge-based features produced an average of 90% performance, in terms of precision and recall.

To further improve the performance and to reduce the problem of automated choice of parameters, we introduce adaptation in the video segmentation process. The motivation for using adaptive analysis of video comes from the fact that video data contains wide variety of content. Thus, adjusting the analysis parameters according to the video content should lead to a better result. We consider adaptation at three levels: at the feature extraction stage, at the video sequence level, and at the individual scene level. Adaptation at video sequence level produced 99% performance; however, characterizing an entire video into one type is difficult. Scene-level adaptation characterizes the scenes in a video based on activity and motion measures, and processes each scene with a different set of parameters based on the scene characteristics. Scene level adaptation produced an average of 91% performance.

DEDICATION

This thesis is dedicated to my parents Mr. and Mrs. Banda Subba Rayudu and my sisters Kavitha Banda and Lalitha Banda, without whose support and prayers this wouldn't have been possible. Thanks.

ACKNOWLEDGEMENT

I would like to express my sincere thanks to Dr. Donald A. Adjeroh, for giving me the opportunity to work under him and finish my thesis under his guidance. I would also like to thank Dr. Mark A. Jerabek and Dr. Vasudevan Jagannathan for serving on my committee.

Thanks to Umasankar Kandaswamy for helping me in the initial stages of the project.

I would like to thank all my friends, especially my roommates for being there when I needed them. Thanks a million.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

With rapid advances in storage and communication technologies, video databases have become very popular. Digital libraries and video-on-demand systems store thousands of hours of video data. These huge video databases need to be carefully organized in order for the users to browse through them and access the relevant information efficiently. Hence arises the problem of providing an index for the video database which provides rapid access to the desired locations in the videos. Video indices save a user the trouble of fast-forwarding or rewinding a video to reach the desired scene or content in the video.

Since video data comes in large volumes, it is not possible to index it manually. Analyzing and annotating a video manually is a very labor-intensive task and almost impossible for very long videos. Thus arises the need for automating the process of indexing a video sequence.

1.2 CURRENT APPROACHES FOR VIDEO INDEXING

Video indexing can be done in compressed domain or uncompressed domain. In compressed domain (also called transform domain), the DCT or a similar transformation is applied to the original image which produces a set of AC and DC coefficients. Since these coefficients are mathematically related to the spatial domain, they can be used to detect scene changes in a video. Hence, video indexing in the transform domain is done using these AC and DC coefficients.

In uncompressed domain (also called spatial domain), video indexing can be done using the pixel values directly or by using features extracted from the pixel values. Pixel

values represent the color or intensity values at the pixel locations. Features are the quantitative measures computed from the pixel values, which give useful information about the image.

In this thesis, video indexing has been done in an uncompressed domain using color and edge features. Indexing in either domain can further be classified as content-based or context-based. Content-based video indexing techniques index a video based on its visual similarity. In the case of context-based indexing, the video is indexed based on context or events in the video. This thesis focuses on content-based video indexing.

Video indexing in general includes two major stages: (i) Video segmentation or partitioning, and (ii) Video abstractions. Video segmentation is usually done by computing a difference metric, also called as distance metric, between successive frames in a video. The distance metric gives the amount of dissimilarity between two frames, and it is calculated using a set of features extracted from each frame. In a video, the successive frames in the same scene do not differ much. But the difference between two frames is high when there is a scene change between the two frames, i.e., one of the frames is the last frame of a scene and the next frame is the first frame of the next scene. The next stage in video indexing is generating video abstractions. There are several ways in which a video can be abstracted. Examples of video abstractions include: key-frames, video mosaics, super-resolution frames etc.

The entire process of video indexing can be represented using the following flowchart (Fig. 1.1):

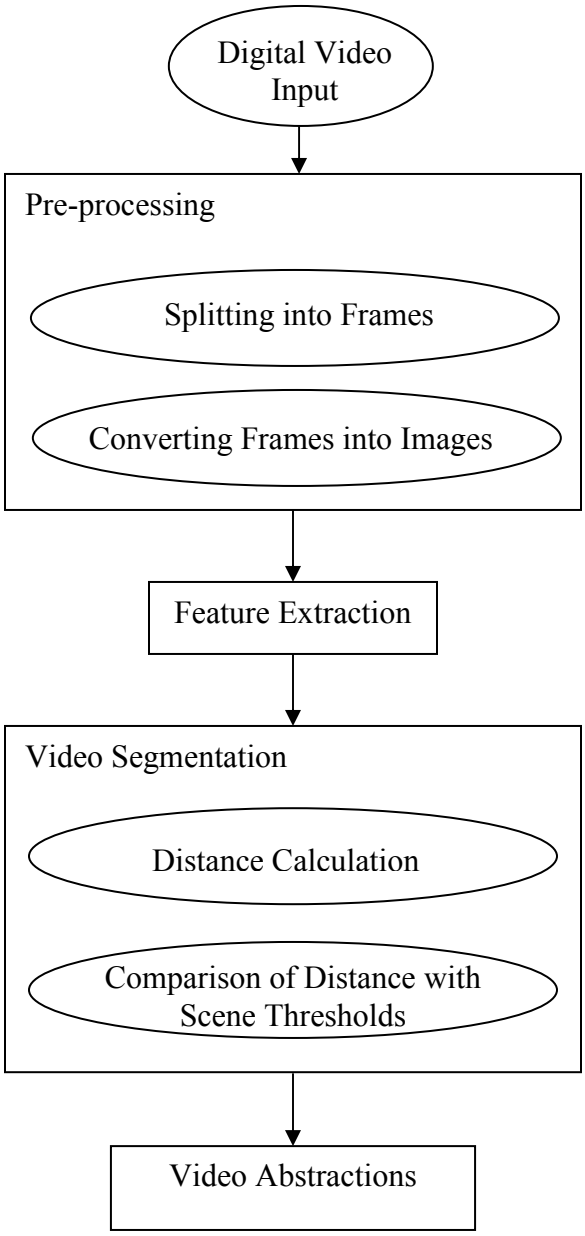

Figure 1.1: Block diagram showing the entire process of video indexing

In this thesis, we consider the problem of video segmentation. The performance of the video indexing system depends on the accuracy with which the system can detect all the scene cuts in a video without any human intervention, and the amount of computation required in order to generate the index for the entire video. In particular, we focus on accuracy rather than speed because indexing is done offline and only once for

each video. Also the indexing has been done in the uncompressed domain in order to get accurate results.

One problem in video partitioning is the detection of gradual transitions. Gradual transitions are the scene transitions which occur within a duration of several frames rather than between two frames. Examples of gradual transitions are dissolves, wipes and fades. Hence the gradual transitions result in missed detection if special care is not taken to handle them. Another problem in video partitioning is the detection of false alarms. False alarms are generally due to camera motion such as panning, zooming etc. False alarms can also be generated due to fast moving objects in a scene. Hence special care should be taken when a scene has high motion; this motion may be due to camera movement or object movement.

The performance of a video segmentation system is evaluated using two quantitative measures which compute the proportion of false and missed detections. These measures are called precision and recall. Precision is the proportion of correct detections out of the total scenes detected by the system. Recall is the proportion of correct detections out of the total true scenes in the video. True scenes were detected for the test data set by human observation.

1.3 ADAPTIVE VIDEO INDEXING

Adaptive video indexing increases the performance of the indexing technique in terms of effectiveness and efficiency. The indexing technique adapts itself to the wide variety of content in the video. The variety may be in terms of motion and activity in the video scenes. Ideally, with adaptive indexing, each video scene is treated in such a way that will produce the best performance. Adaptivity can be implemented at various levels:

video-level, scene-level and resolution-level. There could be several parameters for adaptation, such as features, scene thresholds, skip factors etc. Adaptation can be done in temporal, motion or spatial domain. Temporal adaptation introduces a skip factor, φ which is the number of frames that can be skipped or excluded from being processed, without degrading the results of the video indexing algorithm. This can reduce the amount of processing and hence speed up the process of indexing. Similarly, in the case of spatial adaptation, only a significant part of a frame can be used in the computation rather than considering the entire frame.

We see that video partitioning is a major step in video indexing, and the efficiency of a video indexing technique depends on the efficiency of the video partitioning algorithm. Hence, we study the problem of video partitioning using an edge-oriented adaptive framework in order to gain improvements in efficiency and effectiveness of the system. In the next chapter, we discuss the related work done in this direction. Chapter 3 describes our approach to video segmentation using edge-based features. Chapter 4 presents the multi-resolution framework and introduces adaptivity in our algorithm. Chapter 5 presents experimental results on real video sequences. We conclude the thesis in chapter 6.

CHAPTER 2: LITERATURE REVIEW

Earliest methods of video indexing were based on manually entered keywords. But manual indexing is not feasible for large video databases. Also the keywords do not represent the exact content of the video. Hence further research was done on techniques to index a video based on its content. Automatic video indexing techniques were developed which were based on the information contained in the video. These automatic techniques are sometimes called content-based video indexing techniques since they depend only on the contents of the video. One limitation with content-based video indexing was its inability to capture the contextual information. Contextual information represents the meaning of the objects or the events in a video. To overcome this limitation, researchers came up with several approaches for context-based indexing. This thesis focuses on content-based video indexing and hence in the rest of the chapter, our discussion will be mainly on content-based video indexing techniques.

Various approaches for content-based video indexing have been proposed. These approaches can be broadly grouped into 3 categories: pixel-comparisons, histogram-comparisons, and feature-based segmentation. In pixel-comparisons, the pixel values in one frame are directly compared against the corresponding pixel values in the successive frame. This approach suffered with illumination and motion changes in the video. The histogram-comparisons compared the distribution of the intensity values in the two frames being compared. But this approach will not produce correct results when two quite different frames have similar intensity distribution. In feature-based video segmentation, the comparison between two frames is made using the features extracted from the frames. This approach also suffers from illumination and motion changes based

on the features used. Hence, the challenge is to generate the features that are robust to illumination and motion changes. The effectiveness of a feature-based segmentation algorithm depends on the effectiveness of the features.

2.1 PIXEL-BASED COMPARISONS

In 1993, Zhang et. al. [19] developed a difference metric for video segmentation based on pair-wise pixel comparisons. The pair-wise segmentation algorithm compares the corresponding pixels in the two successive frames and counts the number of pixels that changed from one frame to the other. A scene cut is detected if the total number of pixels that changed exceeds a given percentage of the total number of pixels in a frame. The difference metric is given as a binary function $DP_i(k,l)$, where (k,l) is the pixel position being compared and subscript *i* denotes the index of the frame, and $I_i(k,l)$ is the intensity value at pixel (k,l). The difference metric $DP_i(k,l)$ is defined as:

$$DP_i(k,l) = \begin{cases} 1\ , & \text{if } \left| I_i(k,l) - I_{i+1}(k,l) \right| > t \\ 0\ , & \text{otherwise} \end{cases} \tag{2.1}$$

The criteria for identifying a scene cut is then given as:

$$\frac{\sum_{k=1}^{M}\sum_{l=1}^{N} DP_i(k,l)}{M * N} * 100 > T \tag{2.2}$$

where *T* is the scene threshold, and *M* and *N* are the frame dimensions. This metric was sensitive to camera movement, and hence the likelihood ratio was proposed.

2.2 LIKELIHOOD RATIOS

In the likelihood ratio approach [19], the corresponding regions (blocks) in two successive frames were compared using a likelihood ratio. The likelihood ratio was computed as follows:

$$LR_i = \frac{\left[\frac{\sigma_i^{\,2} + \sigma_{i+1}^{\,2}}{2} + \left(\frac{\mu_i - \mu_{i+1}}{2} \right)^2 \right]^2}{\sigma_i^{\,2} * \sigma_{i+1}^{\,2}} \tag{2.3}$$

where μ_i and μ_{i+1} denote the mean intensity values for a given region in frame i and frame i+1, and $\sigma_i^{\,2}$ and $\sigma_{i+1}^{\,2}$ denote the corresponding variances. The likelihood ratio of all the regions in a frame are compared against a threshold and a camera break is detected if the total number of regions whose likelihood ratio exceeds the threshold is sufficiently large. One major limitation of the likelihood ratio is that if two sample areas to be compared have the same mean and variance, but completely different probability density functions, no change will be detected.

2.3 HISTOGRAM-BASED COMPARISONS

An alternative approach to pixel-comparisons and likelihood approach is the histogram approach [19], which is based on the principle that two frames having an unchanging background and unchanging objects will show little difference in their respective histograms. The histogram difference between two successive frames is given as:

$$SD_i = \sum_{j=1}^{G} \left| H_i(j) - H_{i+1}(j) \right| \tag{2.4}$$

where $H_i(j)$ is the histogram value for the *i*-th frame and *j* is one of the *G* possible grey levels. A scene cut is detected when the overall difference SD_i is larger than a given threshold *T*. To select a suitable threshold, SD_i can be normalized by dividing it by 2**M* * *N*.

2.4 FEATURE-BASED COMPARISONS

2.4.1 COLOR RATIOS

In 1997, Adjeroh and Lee [1] proposed a robust and fast video indexing technique using neighborhood-based color ratios. These color ratio features are invariant to illumination and motion changes. The neighborhood color ratios were defined as:

$$\Phi(x,y) = \frac{\frac{1}{m}\sum_{i=1}^{i=m} h_i(x,y)}{h(x,y)} \tag{2.5}$$

where *m* is the size of the neighborhood, $h_i(x,y)$ is the pixel value at the *i*-th neighbor.

A color ratio histogram was computed using the obtained color ratios. The color ratio histogram indicates the number of color changes in an image. The color ratios appeared to be congested and mostly close to 1, except in a region where there was a distinctive color boundary. The histogram matching was done in a stepwise incremental fashion, by choosing a step size at each matching step. The matching process was terminated (without necessarily comparing all the bins) when it became clear that further comparisons would not make any significant difference to the match results already obtained. This was called premature termination. Hence the histogram matching process involves the determination of formal thresholds, optimal match step size and the optimal

termination step. The threshold in this case is the minimum proportion of the bins that must be matched before deciding whether two images were similar or not.

The color-ratio model is stable under linear combinations making it suitable for DCT, KLT and other linear transforms. Hence it is applied in transform domain video indexing. In the transform domain, the color ratio features are obtained using the neighboring blocks. Rather than using ratio histograms to compare two frames, a new measure for block-wise color ratio comparison was formulated using the transform coefficients and the block-based neighborhoods as shown in equation (2.6).

$$\Phi(u,v) = \frac{\frac{1}{m}\sum_{i=1}^{i=m} H_i(u,v)}{H(u,v)} \tag{2.6}$$

where m is the number of neighboring blocks, $H(u,v)$ is the transform coefficient at location (u,v) for the block under consideration, and $H_i(u,v)$ is the coefficient at the corresponding (u,v) location in the i-th neighboring block.

The optimal parameters for the color-ratio model were obtained using the transform domain coefficients. For detecting special effects, DC components were used directly without calculating their neighborhood-based ratios.

2.4.2 EDGE-BASED

In 1999, Zabih et. al. [18] proposed a feature-based algorithm for detecting and classifying production effects like scene cuts, fades, wipes and dissolves. They extracted edges from the images using Canny's edge detector [6]. The dissimilarity measure was computed as the proportion of edge pixels in a frame that appear far (i.e., more than a distance r) from the edge pixels in the previous frame. The dissimilarity metric was

called Edge change fraction and represented as ρ. Scene cuts were detected when the ρ assumed high values between consecutive frames. One limitation of Zabih and Miller's algorithm was its sensitivity to the illumination changes and to rapid motion of multiple objects.

2.4.3 HIDDEN MARKOV MODEL

In 2003, Park et. al. [13] proposed a video scene change detection technique using hierarchical Hidden Markov Models (HMMs). Two types of features, histogram-based and moment-based, were used to train the HMMs. First, a wavelet transform was applied to each frame of a video, and then the frame was decomposed into frequency-localized sub-bands. Histogram-based features were extracted from a low frequency sub-band which preserves the video information in the form of smoothed and compressed patterns. Moment-based features were extracted from the double wavelet difference of wavelet coefficients in the high-frequency sub-bands of wavelet-transformed images. Since the wavelet coefficients that lie in high frequency sub-bands of a wavelet transformed frame represent the edge information with the intrinsic direction, they can be efficiently used to segment the scene of a video that transforms gradually.

This method was tested on a video database containing news, movies and music videos. The two HMMs were trained using histogram difference and the moment-based features. After training the models, the Viterbi Algorithm was used to find the optimal state sequence with maximum posterior probability and the segmentation was carried out by simply mapping the state sequence onto the given model. The histogram-based HMM was first used to segment the input video sequence into three categories: shot, cut and gradual scene changes. Then the moment-based HMM was used to further segment the

gradual changes into fades, dissolves and wipes. The experimental results were compared against those obtained from threshold-based methods, and the HMM-based method was shown to be more effective in segmenting the videos than the threshold-based methods, with 94.3% recall and 97.5% precision.

2.5 MOTION-BASED COMPARISONS

In 2001, Porter et. al. [14] used the average inter-frame correlation coefficient and block-based motion estimation to distinguish changes caused by shot transitions from those caused by camera and object motion. Each frame was divided into blocks and the normalized correlation between blocks was calculated in the frequency domain because calculating the normalized correlation in the spatial domain is very expensive. A high-pass filter was applied to each image before performing the correlations as the correlation field derived from high-pass regions contained more detectable peaks. As a consequence of applying the high-pass filter, the mean of the image was removed and hence the correlation was invariant to changes in the mean intensity. Normalization of correlation was done in order to make the method insensitive to the positive scaling of the image intensities. Hence the method was robust to the changes in the global illumination.

The value of the maximum correlation coefficient was used as a *goodness-of-fit* measure for each block. The value of the *goodness-of-fit* measure lies between 0 and 1, where a value of 0 indicates a complete mismatch and a value of 1 indicates a perfect match. A similarity metric for each frame pair was derived by taking the mean of the *goodness-of-fit* measure of all the blocks.

Given M as the average of the previous similarity measures since the last shot cut, and M_n be the similarity metric between frames n and $n+1$, then a shot cut is detected if

$M - M_n > T_c$, i.e., if the rate of change from the average similarity measure is greater than some threshold T_c. Other motion-based approaches have been described in [5, 8].

2.6 SPECIAL-EFFECT TRANSITIONS

Zhang et. al. [19] used Twin comparison approach to detect special effects such as dissolves, fade-ins and fade-outs. Twin comparison required the use of two cutoff thresholds: T_b for camera break detection and T_s for special effect detection, where $T_s < T_b$. Twin comparison detects the difference values that exceed T_b and declares them as camera breaks. It also detects the difference values that are less than T_b but greater than T_s, and marks them as the potential start (F_s) of a gradual transition. This start frame (F_s) is then compared to subsequent frames and the end of transition is detected when this comparison gives the difference value that exceeds T_b. The changes due to camera movements like panning and zooming also produce the same kind of behavior and they may be detected as gradual transitions. In order to distinguish camera movements from special effect transitions, motion vectors were detected and analyzed.

Zabih et. al. [18] computed a proportion of entering (ρ_{in}) and exiting (ρ_{out}) edge pixels between frames in order to detect gradual transitions. Fade-outs are detected when ρ_{in} assumes higher values, and fade-ins are detected when ρ_{out} assumes higher values. During a dissolve, one scene fades-out and the next scene fades-in. Hence a dissolve can be detected in an interval where ρ_{out} is high for the first half and then ρ_{in} is high for the next half of the interval. Wipes can be distinguished from dissolves and fades by analyzing the spatial distribution of the entering and exiting pixels (which are also called changing pixels). A percentage of changing pixels in the top half and left half of the frame was computed. For a left-to-right wipe, the majority of changing pixels occur in

the left-half of the image during the first-half of the wipe, then in the right-half of the image during the second-half of the wipe. Similarly for top-to-bottom wipe, majority of changing pixels occurs in top half, then in the bottom half. The edge change fraction in case of instantaneous cuts is computed as:

$$\rho = \max(\rho_{in}, \rho_{out}) \tag{2.7}$$

Porter et. al. [14] detected fades based on the value of M_n. The correlation of an image with a constant image results in $M_n = 0$. Hence, fades were detected when $M_n = 0$. A fade is a scaling of the pixel intensities over time which can be observed in the standard deviation of the pixel intensities. If M_n falls to 0 and the standard deviation of the pixel intensities decreased prior to this, the frame where the standard deviation started to decrease is marked as the first frame of the fade-out. If the standard deviation of the pixel intensities increases after the similarity metric increases from 0, the frame where the standard deviation becomes constant is marked as the end of the fade-in. In order to detect dissolves, the first frame of the sequence is divided into a regular grid of blocks of size 32x32, and a selection of these blocks is used to represent regions of interest (ROI) in the image. A block is selected as a ROI if

$$\sigma_b^2 = \frac{\sigma_I^2}{\ln(\sigma_I^2)} \tag{2.8}$$

where σ_b is the variance of a block b and σ_I is the variance of the image I. Between each frame pair n and $n+1$, maximum correlation between each ROI is used as a *goodness-of-fit* measure, and a single similarity metric F_n for the set of ROI is calculated as the average of the *goodness-of-fit* measures of the set of ROI. F_n remains high during a shot indicating that the contents of each ROI has not changed significantly. During a

dissolve, the content of each ROI gradually changes and F_n will decrease until it reaches its lowest value at the end of the dissolve. During a shot M_n and F_n should be approximately equivalent. F_n is compared against M_n and the change in F_n with respect to M_n is calculated as the ratio R_n,

$$R_n = \frac{M_n}{F_n} \tag{2.9}$$

The start of a dissolve is marked when R_n starts to increase, and the end of dissolve is marked once R_n reaches its maximum.

2.7 ADAPTIVE METHODS

In 2000, Yusoff et. al. [17] proposed the use of adaptive thresholds in order to improve the performance of shot detection methods. They experimented with 3 methods of adaptive thresholding on 4 shot boundary detection algorithms. The 4 shot boundary detection algorithms used were:

- Average Intensity Measurement (AIM)
- Histogram Comparison (HC)
- Likelihood Ratio (LH)
- Motion Estimation (ME)

The AIM algorithm computes the average of the intensity values in the current frame and compares it with that for the following frame. The HC and LH algorithms [1] have been explained earlier in this chapter. In ME approach, the next frame in a video sequence is estimated and reconstructed based on the motion information in the current frame. The dissimilarity measure is given by the mean absolute difference between the reconstructed frame and the original frame.

In each of the shot boundary detection algorithms, a single statistic m was generated for each pair of frames to quantify the degree of dissimilarity between the two frames. Their scheme for determining the decision threshold m_T was based on the assumption that the dissimilarity measure $\{m\}$ comes from one of the two distributions: one for shot boundaries (S) and one for "not-a-shot-boundary" (N). In general, S has a considerably larger mean and standard deviation than N as shown in Fig. 2.1. Assuming the cost of false positives and undetected true positives to be same, the standard classification methods would indicate that the decision threshold m_T should be fixed so that the tails of the two density functions p_S and p_N have an equal area as shown in the figure. We can see that the decision threshold m_T is fairly close to the mean μ_N of N, and hence it is important that the position and width of N be accurately determined.

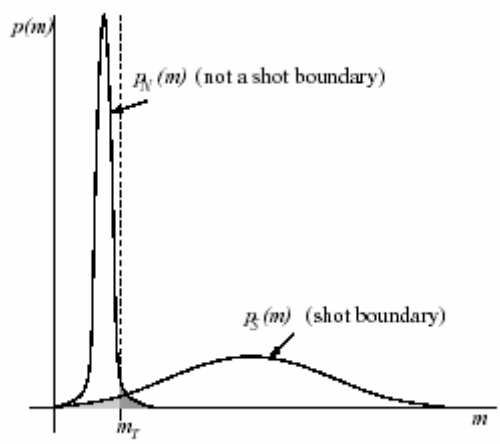

Figure 2.1: Distribution of Dissimilarity measure *m*

Experiments demonstrated that a single decision threshold can be consistently grossly over- or underestimated when applied to video material with distinctive characteristics, such as sports events or cartoons. And hence, p_N was estimated dynamically using the dissimilarity measures from the previous and next few frames.

Then, the mean, μ_N and the variance σ_N were estimated and used in determining the decision threshold m_T as some function of these two statistics. The 3 models used for computing decision thresholds were:

- Constant variance model (A): The decision threshold was set at some fixed positive offset from μ_N.

$$m_T = \mu_N + T_c \tag{2.10}$$

where T_c was determined by experimenting with a range of values on a training set of video data.

- Proportional variance model (M): The decision threshold was set at some multiple of μ_N.

$$m_T = T_p \mu_N \tag{2.11}$$

where T_p was determined from experimentation.

- The Dugad model (D): The decision threshold was set at some multiple of $\sqrt{\sigma_N}$ from μ_N.

$$m_T = \mu_N + T_d \sqrt{\sigma_N} \tag{2.12}$$

where T_d was obtained by experimenting with a range of values on the training data set.

The adaptive thresholds were determined based on the statistics of the dissimilarity measure within a sliding window. For each of the above 3 models, two different strategies: single window and dual windows, were adopted leading to six different adaptive thresholding methods. The single window strategy considered all the samples,

including the central sample for the calculation. In the dual windows strategy, the window was split into two halves on either side of the center sample. All the shot boundary detection algorithms showed considerable improvement in their performance when adaptive thresholding was used.

In 2004, Adjeroh and Lee [2] proposed scene-adaptive video partitioning in compressed domain using transform domain coefficients. They analyzed the scenes in a video sequence and classified them into eight scene classes based on the scene complexity. Scene complexity was calculated using scene activity and motion measures. They made use of AC and DC coefficients and the neighborhood differences in order to obtain scene activity and motion. The neighborhood approach is relatively robust to camera operations such as panning and zooming, and is invariant to the illumination changes in the image. The parameters of the video partitioning algorithm were adapted according to the scene classes. Five video analysis parameters were used: number of DCT coefficients, temporal skip factor, weights on AC and DC coefficients, minimum frame proportions to be involved f_{min}, and scene threshold. The generalized color ratio (GCR) model [1] was used for video partitioning. The scenes were also classified into 6 temporal classes based on scene duration. This temporal classification was useful in selecting local thresholds for scene partitioning. Video sequences were also classified into video classes based on the video quality which is useful in determining f_{min}, and the measures used for such a classification were peak signal to noise ratio (PSNR) and mean square error (MSE). The temporal skip factor was determined using motion class. The number of coefficients to be used was determined by the scene class. Adaptive skip factor improves efficiency, whereas scene-adaptive thresholds and weights improve the

robustness of the system. The parameters (no. of coefficients) and f_{min} have impact on both efficiency and robustness. The scene thresholds were determined from the mean and standard deviation of the similarity evaluation function. Thresholds were defined for each scene, using the scene class and the temporal class (local and global thresholds). The experimental results showed an improvement in detection performance.

In this thesis, we propose an adaptive approach for video segmentation in an uncompressed domain. The adaptation is not only in terms of scene thresholds, but also in terms of the features used in the computation of the dissimilarity measure. It is unique in its approach, as it uses motion and illumination invariant edge-based features in a multi-resolution framework.

CHAPTER 3: MULTI-RESOLUTION EDGE RESPONSE VECTORS

3.1 INTRODUCTION

Edges in an image typically manifest as sudden changes in intensity. They can be thought of as locations of abrupt grey-level change. They form the boundary between two dissimilar regions in an image. Edges are very important features in an image as they provide information about structure in the image (see Fig. 3.1).

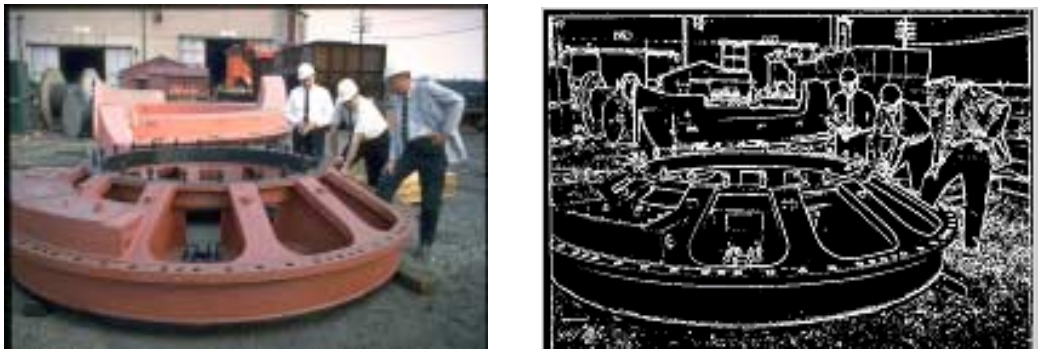

Fig 3.1 (a) Original Image (b) Edges in the image

Edges in an image can be detected using high pass filters like Sobel, Canny, Prewitts, Roberts etc. The high pass filters retain only the high frequencies, that is, the sharp details like edges in the image. Edge detection using the above filters can be performed by convolving the horizontal and vertical kernels over the entire image and then adding up the results obtained by the two convolutions. Convolving an image with a horizontal kernel gives the horizontal edges, whereas convolving the image with a vertical kernel gives the vertical edges. Combining these two results would give us both the horizontal and vertical edges. We have used Sobel filters as the edge detector in our method. The reason for such a selection is that the Sobel filter is simple and easy to implement, and at the same time produces a good result.

3.2 FEATURE EXTRACTION

In order to analyze a video, we need to extract some quantitative measures from the video content. These quantitative measures are called the features. Color, edges, texture and shape are simple examples of image features. In this work, we use the color feature along with edge-based features. The edge-based features are derived from edge information in the images. We use the mean and standard deviation of these features at various resolution levels as the quantitative measures for video analysis.

We use a multi-resolution approach in the processing of the images. That is, an image is divided into smaller blocks and these smaller blocks are then used for comparison between images. We process the image at multiple resolutions, with the lowest resolution corresponding to the original image, and the highest resolution being the smallest blocks into which the image is being divided (see Fig. 3.2). Based on our experiments, we considered only 3 levels of resolution, since higher levels do not yield better results but involve more of computation. We number the resolution levels as $k = 0$, 1 and 2. Level 0 is the lowest resolution, that is, the original image. At Level 1, the image is divided into 4 blocks, and at level 2, the image is divided into 16 blocks. Hence, at a given level k, an image is divided into 2^{2k} number of blocks. The comparisons at all the levels are taken into consideration while measuring the similarity or dissimilarity between the images.

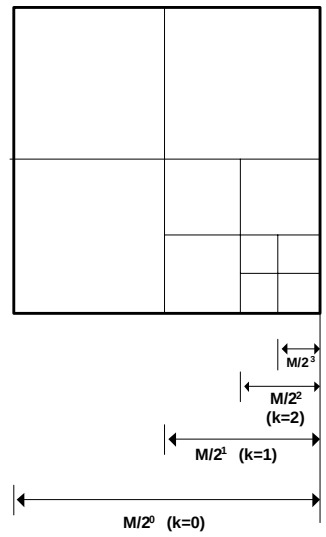

Figure 3.2: Multi-resolution Decomposition of a frame

3.2.1 EDGE DETECTION

Edges can be detected by calculating the gradient of the image in the *x*- and *y*-directions. The horizontal and vertical gradients detect the horizontal and vertical edges respectively.

Let $I(x,y)$ be an image, and H_x and H_y be the horizontal and vertical Sobel masks respectively, then the gradients $G_x(x,y)$ and $G_y(x,y)$ are obtained by convolving H_x and H_y over the image $I(x,y)$. That is, $H_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$, $H_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

$$G_x(x,y) = I(x,y) * H_x(x,y) \tag{3.1}$$

$$G_y(x,y) = I(x,y) * H_y(x,y) \tag{3.2}$$

* represents the convolution operation.

The gradient amplitude $G_A(x,y)$ is given by:

$$G_A(x,y) = \sqrt{G_x^2(x,y) + G_y^2(x,y)} \tag{3.3}$$

The gradient magnitude can also be calculated approximately using the simple absolute sum:

$$G_A(x,y) = |G_x(x,y)| + |G_y(x,y)| \tag{3.4}$$

The gradient magnitude gives the edge response at each pixel position in the image. The edge pixels can be determined by comparing the edge response of each pixel with an edge threshold, τ_e. If the pixel's edge response exceeds the edge threshold, then the pixel is said to be an edge pixel or edge point. The edge threshold is calculated as:

$$\tau_e = \frac{\alpha}{(M)^2} \sum_{x=1}^{M} \sum_{y=1}^{M} G_A(x,y) \tag{3.5}$$

where α is a constant. (Based on empirical results, we chose $\alpha = \frac{2}{3}$ or $\alpha = \frac{5}{6}$).

The edge direction or the phase angle at pixel $I(x,y)$ can be calculated as:

$$G_\phi(x,y) = \tan^{-1}\left(\frac{G_y(x,y)}{G_x(x,y)}\right) \tag{3.6}$$

3.2.2 EDGE-BASED FEATURES

One of the advantages of using edge-based features is that, since they represent the structural information in the image, they are invariant to illumination changes, and also relatively robust under changes due to motion. The edge-based features used in our research work are:

- Edge Response
- Edge Response at edge-points
- Edge Direction
- Edge Density

At each resolution level *k,* and for each given block *r*, the mean and standard deviation of each of the edge-based features are calculated as follows:

(i) Edge Response

$$\mu_{e,r}^{k} = \frac{\alpha}{\left(m_r^k\right)^2} \sum_{x=1}^{m_r^k} \sum_{y=1}^{m_r^k} G_{A,r}^{k}(x,y) \tag{3.7}$$

$$\sigma_{e,r}^{k} = \left[\frac{\alpha}{\left(m_r^k\right)^2 - 1} \sum_{x=1}^{m_r^k} \sum_{y=1}^{m_r^k} \left(G_{A,r}^{k}(x,y) - \mu_r^k \right)^2 \right]^{\frac{1}{2}} \tag{3.8}$$

(ii) Edge Response at points

It is same as the edge response calculated in Eqn (3.7) & (3.8), except that instead of considering all the pixels in a given block *r*, we consider only the edge pixels in the given block *r* for the calculation of $\mu_{ep,r}^{k}$ and $\sigma_{ep,r}^{k}$.

(iii) Edge Direction

$$\mu_{\varphi,r}^{k} = \frac{\alpha}{\left(m_r^k\right)^2} \sum_{x=1}^{m_r^k} \sum_{y=1}^{m_r^k} G_{\varphi,r}^{k}(x,y) \tag{3.9}$$

$$\sigma_{\varphi,r}^{k} = \left[\frac{\alpha}{\left(m_r^k\right)^2 - 1} \sum_{x=1}^{m_r^k} \sum_{y=1}^{m_r^k} \left(G_{\varphi,r}^{k}(x,y) - \mu_r^k \right)^2 \right]^{\frac{1}{2}} \tag{3.10}$$

(iv) Edge Density

Edge density is computed using an edge map. An edge map is defined using the edge threshold τ_e as follows:

$$E_r^k(x,y) = \begin{cases} 1\ , & G_{A,r}^{k}(x,y) > \tau_e \\ 0\ , & \text{otherwise} \end{cases} \tag{3.11}$$

where $\tau_e = \frac{\alpha}{(M)^2} \sum_{x=1}^{M} \sum_{y=1}^{M} G_A(x, y)$

where α is a constant (as described earlier).

Edge density has no standard deviation, and hence only the mean is calculated as follows:

$$\mu_{\lambda,r}^{k} = \frac{\alpha}{\left(m_r^k\right)^2} \sum_{x=1}^{m_r^k} \sum_{y=1}^{m_r^k} E_r^k(x, y) \tag{3.12}$$

Essentially, the edge density represents the average number of edge points per pixel in the image block *r*.

3.2.3 COLOR FEATURE

The color feature is given by the intensity value or the gray-level value of the pixels in the image. Though the color feature is sensitive to illumination changes, it provides an important measure of information content in the image, when combined with the illumination and motion invariant edge-based features. At each resolution *k*, the mean and standard deviation of the color feature are calculated in a manner similar to the edge-based features:

$$\mu_{c,r}^{k} = \frac{\alpha}{\left(m_r^k\right)^2} \sum_{x=1}^{m_r^k} \sum_{y=1}^{m_r^k} I_r^k(x, y) \tag{3.13}$$

$$\sigma_{c,r}^{k} = \left[\frac{\alpha}{\left(m_r^k\right)^2 - 1} \sum_{x=1}^{m_r^k} \sum_{y=1}^{m_r^k} \left(I_r^k(x, y) - \mu_{c,r}^{k} \right)^2 \right]^{\frac{1}{2}} \tag{3.14}$$

3.3 DISTANCE METRIC

Having computed the features, we need a method to use these features to determine similar frames in the video. The distance metric gives the amount of dissimilarity between two frames. The distance can be calculated using the extracted

multi-resolution edge response vectors. The distance calculation using edge densities involves only the mean, since edge density has no standard deviation.

$$d_{\lambda}(I_1, I_2) = \sum_k \sum_r \left| \mu_{\lambda,r}^{k,1} - \mu_{\lambda,r}^{k,2} \right| \tag{3.15}$$

The distance calculation for the rest of the features involves both mean and standard deviation.

$$d_{e}(I_1, I_2) = \sum_k \sum_r \left(\left| \mu_{e,r}^{k,1} - \mu_{e,r}^{k,2} \right| + \left| \sigma_{e,r}^{k,1} - \sigma_{e,r}^{k,2} \right| \right) \tag{3.16}$$

To prevent the distance value from being dominated by one component (typically, the mean), the distances are normalized to restrict their values to the range of 0 to 1. This gives a better picture of the variation in the distance values.

$$d_{\lambda}(I_1, I_2) = \sum_k \sum_r \left(\frac{\left| \mu_{\lambda,r}^{k,1} - \mu_{\lambda,r}^{k,2} \right|}{1 + \mu_{\lambda,r}^{k,1} + \mu_{\lambda,r}^{k,2}} \right) \tag{3.17}$$

$$d_{e}(I_1, I_2) = \sum_k \sum_r \left(\frac{\left| \mu_{e,r}^{k,1} - \mu_{e,r}^{k,2} \right|}{1 + \mu_{e,r}^{k,1} + \mu_{e,r}^{k,2}} + \frac{\left| \sigma_{e,r}^{k,1} - \sigma_{e,r}^{k,2} \right|}{1 + \sigma_{e,r}^{k,1} + \sigma_{e,r}^{k,2}} \right) \tag{3.18}$$

Similarly, we obtain normalized *distances* $d_c(I_1, I_2)$, $d_{ep}(I_1, I_2)$ and $d_{\varphi}(I_1, I_2)$, for color, edge response at edge points and the edge direction respectively.

The overall distance between the two frames is determined as a simple average of the individual distances from all the features:

$$D(I_1, I_2) = \tfrac{1}{5} * [d_c(I_1, I_2) + d_e(I_1, I_2) + d_{ep}(I_1, I_2) + d_{\varphi}(I_1, I_2) + d_{\lambda}(I_1, I_2)] \tag{3.19}$$

3.4 DETECTING SCENE CUTS

A scene is a collection of all the video frames generated in a single camera operation. It is also called as a shot. The shot boundaries or the scene changes are

detected based on the fact that the last frame of a scene and the first frame of the next scene should differ quite a lot. This difference leads to a large *distance* between these two frames when compared to any two successive frames within a scene. Hence, such peaks in the *distance* measure can be captured as the scene cuts using a scene threshold. The effectiveness of the system is based on the determination of an appropriate scene threshold that can pick up all the scene cuts, and at the same time it should not pick up any false scene cuts. Fig. 3.2 shows a distance plot for "Canyon" video sequence. The 'x' symbols in the figure show the true scene cuts.

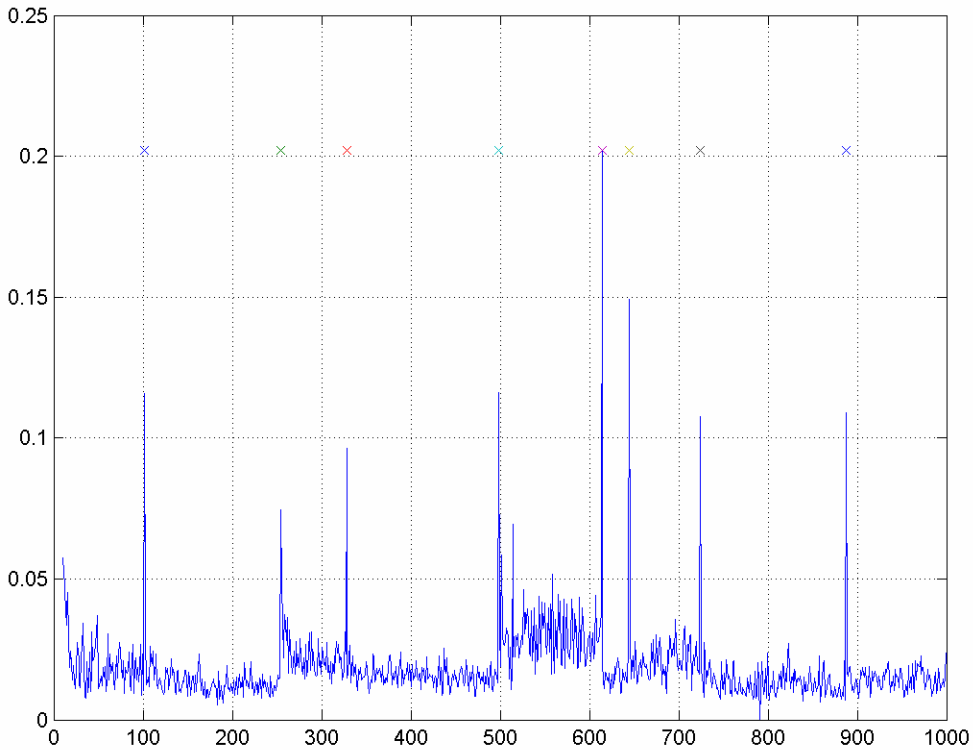

Fig 3.2 Distance plot of "Canyon" video sequence

The scene threshold is determined as:

$$t_s = t * \max\{D(I_1, I_2)\} \tag{3.20}$$

where t is the threshold parameter. (we use $t = 0.4$ in our tests).

3.5 HANDLING SPECIAL EFFECTS

A video consists of several production effects like dissolves, wipes and fades which mark the transition from one scene to another in a sequence of successive frames rather than just two successive frames. These are called the special effects in a video and are difficult to capture because the scene transition is gradual.

Fade: A fade is a gradual transition from a scene to a constant image (*fade-out*), or from a constant image to a scene (*fade-in*).

Dissolve: A dissolve is a gradual transition from one scene to another, in which the first scene fades out and the second scene fades in.

Wipe: A wipe is a gradual transition from one scene to another, in which a line moves across the screen, with the new scene appearing behind the line.

Since the scene change in case of special effects involves more than two frames and is gradual, the distance metric does not show any peaks during these transitions (see Fig. 3.3).

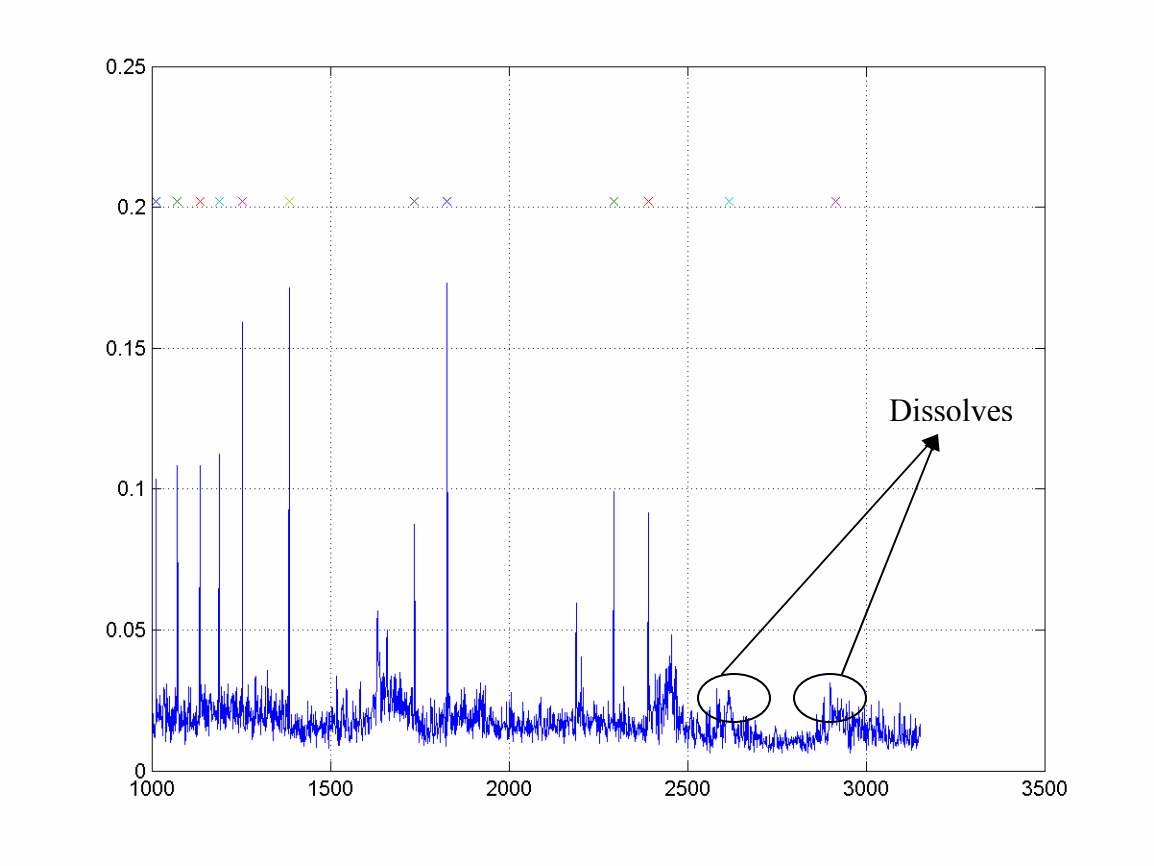

Fig 3.3 Distance Plot showing the variation in distance during the dissolves at frames 2610–2625 and 2905-2920.

From the figure, we can clearly see that the distance values during the dissolve do not show large variation, and hence they cannot be captured with the simple distance metric. Hence we follow a different approach for distance calculation in the case of special effects. The distance is calculated between every I_i and I_{i+n} th frame, where n is the number of frames involved in the scene transition. The new distance obtained by comparing two frames separated by n frames is shown in Fig. 3.4.

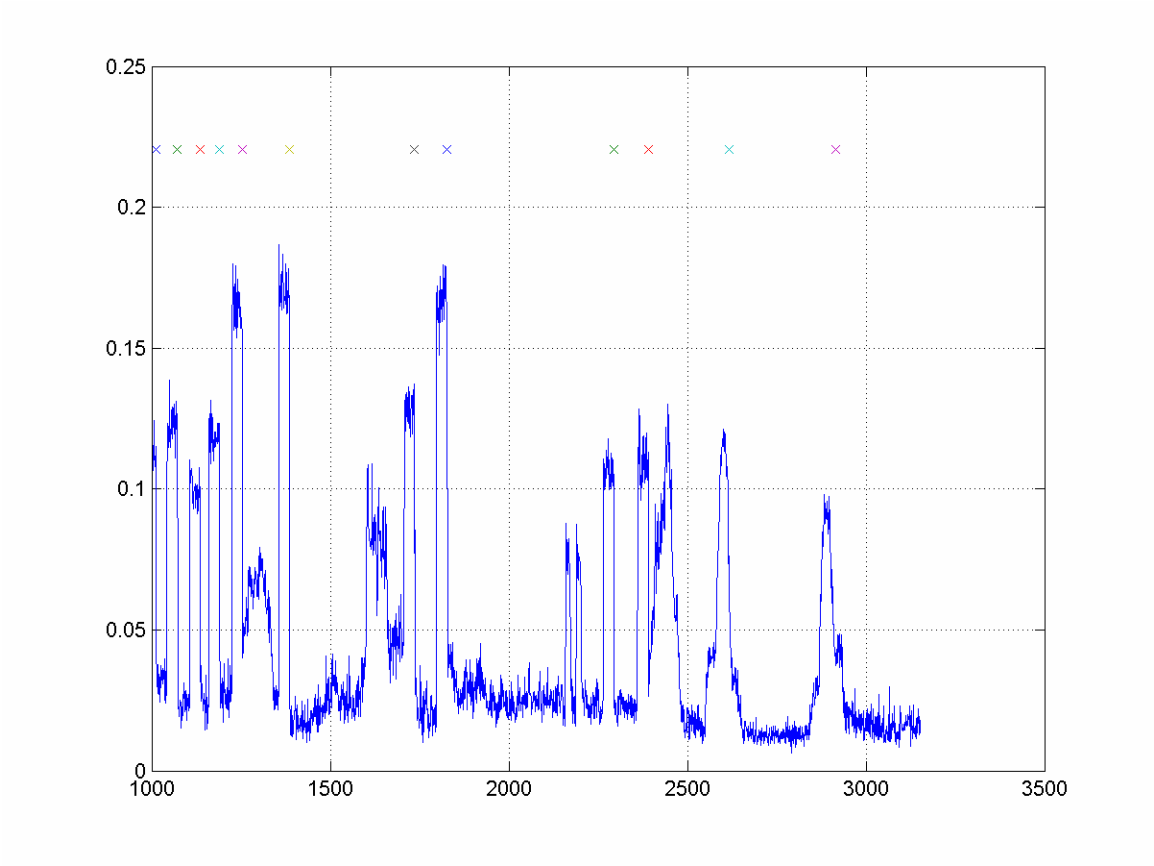

Figure 3.4 Plot showing the new distance calculated between every I_i and I_{i+n} frame.

From the above figure, we can clearly see the peaks in the new distance values at the dissolves, and hence a dissolve can be detected. One problem with this approach is the detection of false alarms. In the figure, we can see peaks in the distance values at frames 1600-1650 and 2150-2200. These false alarms are a result of high motion in the scene. Such false alarms could also be due to camera or object movement. Since we assume the cost of missed detection to be more than the cost of false alarms, we are interested in detecting the dissolves at the expense of introducing some false alarms. But we still need techniques to distinguish between a gradual transition and a camera or object movement.

CHAPTER 4: ADAPTIVE VIDEO INDEXING USING MERVS

4.1 INTRODUCTION TO ADAPTIVE VIDEO INDEXING

In chapter 3, the video was segmented based on a distance metric calculated using all the 5 features and using a fixed scene threshold which was obtained from the distance metric. All the features contributed equally in the distance calculation. But since we know that videos differ widely in their content, treating all the videos in a similar way will not yield the best results. Some videos may have more of edges in them and very little color information whereas others may have much of color information and very few edges. In such cases, equal contribution from all the features may produce sub-optimal results. The contributions of the color and the edge features should be tuned according to the content of the video. For the case of edge-oriented videos, larger weights should be given to the edge-features whereas in case of color-oriented videos, larger weights should be given to the color feature. Moreover, in case of high motion videos, a specific combination of color and edge features may produce the best overall results. This combination of features may differ from one video to the other depending upon the amount of motion in the video. Hence the features and their weights can be adapted with respect to the video content. In addition to the features, the scene thresholds can also be adapted to the video content. Further, sub-adaptive analysis can be performed at different conceptual levels in the video sequence.

4.2 ADAPTIVITY AT THE VIDEO SEQUENCE LEVEL

First, we consider adapting the video analysis algorithm based on the entire video sequence. That is, for each video sequence, we try to determine the analysis parameters

that will produce the best indexing results. This set of parameters are then used to analyze all the frames or scenes in the video.

In section 3.3, the distance metric was calculated using all the 5 features (edge-based features and 1 color feature). Rather than considering all the features for the distance calculation, we could consider only those features that are relevant to the video being analyzed. Thus, the selection of the features should depend on the type of video. Hence the distance metric defined in Equation (1) can be modified as:

$$D(I_1, I_2) = w_c d_c(I_1, I_2) + w_e d_e(I_1, I_2) + w_{ep} d_{ep}(I_1, I_2) + w_\varphi d_\varphi(I_1, I_2) + w_\lambda d_\lambda(I_1, I_2) \qquad (4.1)$$

where $w_c + w_e + w_{ep} + w_\varphi + w_\lambda = 1$. The parameters w_c, w_e, w_{ep}, w_φ and w_λ are the respective weights for the color, edge response, edge response at edge points, edge direction and edge length features. A feature can be totally discarded from the distance calculation by setting its weight equal to 0. In chapter 3, we assumed equal contribution for each feature (i.e., $w_c = w_e = w_{ep} = w_\varphi = w_\lambda = \frac{1}{5}$). We use the parameter w as an index to a particular set of weights (see Table 4.1).

To check the effect of each feature on the different video sequences, we used a combination of these weights. The 32 combinations used are shown in Table 4.1.

Table 4.1: Table of weights for the MERVs

w	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
c	0	0	0	0	1	0	0	0	0	0	0	0.5	0.5	0.5	0.5	0
e_r	0	0	0	1	0	0	0	0	0.5	0.5	0.5	0	0	0	0.5	0
e_{rp}	0	0	1	0	0	0	0.5	0.5	0	0	0.5	0	0	0.5	0	0.33
φ	0	1	0	0	0	0.5	0	0.5	0	0.5	0	0	0.5	0	0	0.33
λ	1	0	0	0	0	0.5	0.5	0	0.5	0	0	0.5	0	0	0	0.33

w	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**
c	0	0	0	0.33	0.33	0.33	0.33	0.33	0.33	0	0.25	0.25	0.25	0.25	0.2	0.25
e_r	0.33	0.33	0.33	0	0	0	0.33	0.33	0.33	0.25	0	0.25	0.25	0.25	0.2	0.13
e_{rp}	0	0.33	0.33	0	0.33	0.33	0	0	0.33	0.25	0.25	0	0.25	0.25	0.2	0.25

φ	0.33	0	0.33	0.33	0	0.33	0	0.33	0	0.25	0.25	0.25	0	0.25	0.2	0.25
λ	0.33	0.33	0	0.33	0.33	0	0.33	0	0	0.25	0.25	0.25	0.25	0	0.2	0.13

The scene threshold was determined as:

$$t_s = t * \max\{D(I_1, I_2)\} \tag{4.2}$$

Experiments were conducted with 9 values for the threshold, *t*. The scene cuts detected by the system at different scene thresholds are recorded (see Table 4.2). In the table, *PR* is used as the performance measure. Its value ranges between 0 and 1. Higher values of *PR* indicate good performance. Our goal is to achieve PR close to 1, that is, 100% performance. *PR* is explained in detail in Chapter 5.

Table 4.2: Performance of video sequences at different thresholds

t	***PR***			
	ANTELOPE (*w*=12)	CANYON (*w*=15)	DIARY (*w*=15)	JOURNAL (*w*=23)
0.15	0.88	0.89	0.63	0.72
0.18	0.88	0.93	0.77	0.81
0.22	0.92	0.95	**0.99**	0.88
0.26	**0.96**	**1.00**	0.81	0.92
0.3	0.92	**1.00**	0.78	**0.97**
0.33	0.88	**1.00**	0.77	0.91
0.37	0.84	**1.00**	0.7	0.91
0.41	0.83	0.94	0.65	0.92
0.45	0.79	0.92	0.57	0.89

We observe that, different videos may require different contributions for each feature (i.e., different *w*-value[**]) for the best performance. We also observe that, at a given *w*, different thresholds produce different results. In general, video sequences differ widely in their content. For instance, a sports video usually have a higher motion content when compared to an ordinary scientific documentary. Instead of using equal weights for all the features, we may vary the weights to change the contribution of the features in the

[**] We use *w* and contribution of the features synonymously in this thesis.

distance calculation. Some videos may yield good results when the color feature is given more weight, whereas other videos may yield good results when the edge features are given more weight. Depending on the content or type of video, one feature may be more important than the other.

Further analysis shows that, for a given video sequence, the best *PR* results can be obtained with different *w*-values, but at different thresholds. For instance, Table 4.3 shows a more fine-grained result for one of the video sequences.

Table 4.3: Performance of "Canyon" video sequence

w	**t=0.15**	**0.18**	**0.22**	**0.26**	**0.3**	**0.33**	**0.37**	**0.41**	**0.45**
4	0.76	0.86	0.91	0.95	**1.00**	0.94	0.94	0.94	0.92
5	0.91	0.95	0.95	0.95	**1.00**	**1.00**	**1.00**	**1.00**	0.89
15	0.89	0.93	0.95	**1.00**	**1.00**	**1.00**	**1.00**	0.94	0.92
18	0.89	**0.95**	0.95	**1.00**	0.94	0.92	0.92	0.83	0.75
23	**0.93**	0.93	0.95	0.95	0.95	0.95	0.95	0.94	**0.94**
25	0.91	**0.95**	**1.00**	**1.00**	0.92	0.92	0.92	0.86	0.83
29	0.91	**0.95**	0.95	**1.00**	0.94	0.92	0.92	0.89	0.76

In the above table, the maximum performance is **1.00**. This performance is achieved at *w* = 4, 5, 15, 18, 25 and 29, but at different thresholds. Through experiments, the best threshold for each *w* can be determined. Other video sequences could require different sets of *w* in order to produce maximum performance (see Table 4.4).

Table 4.4: *w* and *t* values for the best performance of video sequences

Video	***PR***	***w***	***t***
ANTELOPE	0.96	12	0.26
CANYON	1.00	4, 5, 15, 18, 25, 29	0.26, 0.33, 0.41
DIARY	0.99	12,15, 23	0.22
JOURNAL	0.97	23, 29, 31	0.3

It is obvious from the results that the contribution of the features should change depending on the video content. The problem is that, at the video sequence level, we still

have a very coarse granularity. A more refined approach may be more appropriate, and could pave the way for complete automation.

4.3 ADAPTIVITY AT THE SCENE LEVEL

The results obtained in the previous section can be further improved by introducing adaptivity at the video scene-level. Scenes in a video vary widely in their content. Hence using the same set of features and thresholds for all the scenes may not produce the best results. Each scene should be treated differently with a different set of features and thresholds, depending upon its content and complexity. A scene with high motion produces higher values of *distance* between the successive frames and hence needs a higher scene threshold to identify the scene cut. One more measure of dissimilarity between scenes is the activity measure. Activity in a scene is related to the scene variability, and usually depends on the amount of detail or edges in the scene.

The scenes can be classified into various scene classes based on the activity and motion measures calculated from the scenes. Each scene class is treated with the best possible set of features and scene threshold. In the rest of the thesis, we will refer to the set of features and the scene threshold together as the *parameter* or the *parameter set* for processing the scene. The parameter set for each scene class is determined using a training data set containing a sufficient number of scenes from each scene class.

4.3.1 SCENE ACTIVITY

The activity in an image is a measure of the significant detail in the image. Scene activity can be determined by the activity of the first frame (or selected frames) in the scene. We consider three activity measures: A_1, A_2 and A_3.

Neighborhood differences (A_1):

The first activity measure is calculated as the sum of the horizontal and vertical neighborhood differences. Let $I(x,y)$, be an $M \times M$ image. Let $h(x,y)$ and $v(x,y)$ be horizontal and vertical neighborhood differences respectively. Then the activity A_1 is computed as:

$$A_1 = \frac{1}{M^2}\left[\sum_{x=1}^{M-1}\sum_{y=1}^{M} h(x,y) + \sum_{x=1}^{M}\sum_{y=1}^{M-1} v(x,y)\right] \tag{4.3}$$

<u>*Sum of Squares (A_2):*</u>

The second activity measure A_2 is calculated as the sum of the square of the horizontal and vertical neighborhood differences at each pixel in the image.

$$A_2 = \frac{\sum_{x=1}^{M-1}\sum_{y=1}^{M-1}\sqrt{h(x,y)^2 + v(x,y)^2}}{M^2} \tag{4.4}$$

<u>*Image Variance (A_3):*</u>

A_3 is calculated as the standard deviation of the intensity values.

$$A_3 = \sigma_c = \frac{\alpha}{M}\left[\sum_{x=1}^{M}\sum_{y=1}^{M}\left(I(x,y) - \mu_c\right)^2\right]^{\frac{1}{2}} \tag{4.5}$$

where

$$\mu_c = \frac{1}{M}\sum_{x=1}^{M}\sum_{y=1}^{M} I(x,y) \tag{4.6}$$

It is easy to extend the above schemes under the multi-resolution framework. For instance, with A_3, the activity for r-th block at the k-th resolution is computed as follows:

$$a_r^k = \sigma_{c,r}^k = \alpha\left[\frac{1}{\left(m_r^k\right)^2 - 1}\sum_{x=1}^{m_r^k}\sum_{y=1}^{m_r^k}\left(I_r^k(x,y) - \mu_{c,r}^k\right)^2\right]^{\frac{1}{2}} \tag{4.7}$$

where

$$\mu_{c,r}^{k} = \frac{1}{\left(m_r^k\right)^2} \sum_{x=1}^{m_r^k} \sum_{y=1}^{m_r^k} I_r^k(x,y) \tag{4.8}$$

Then, activity A_3 at a given resolution k is given as the average of activities of all the blocks at the given resolution-level.

$$A_3^k = \frac{1}{2^{2k}} \sum_{r=1}^{2^{2k}} ac_r^k \tag{4.9}$$

Then, $A_3 = \sum_k w_k A_3^k$, where w_k is a weight factor.

The three activity measures were computed for a training data set and A_3 was chosen to be the best activity measure. The decision of choosing one of the three activity measures was subjective. Using the three measures, we computed quantitative estimates of the activity for some test images from different video scenes. We then chose the measure that produced results that were closest to our subjective observation on the activity in the image. The activity values obtained at different resolution levels where almost similar, and hence we chose to consider only the lowest resolution level, $k = 0$ for calculating the activity of an image (i.e., $w_2 = 0$, $w_1 = 0$, $w_0 = 1$).

Activity Classes:

Based on the activity values, a scene can be classified into one of three activity classes: Low, Medium and High activity. Table 4.5 shows the activity classes using A_3, the variance activity measure.

Table 4.5: Activity classes

Activity	Activity Class	Description
0 – 0.14	I	Low
0.15 – 0.19	II	Medium
> 0.19	III	High

The range of activity values for each class is determined using the cumulative distribution function of the activities of all the scenes in a training data set (see Fig. 4.1).

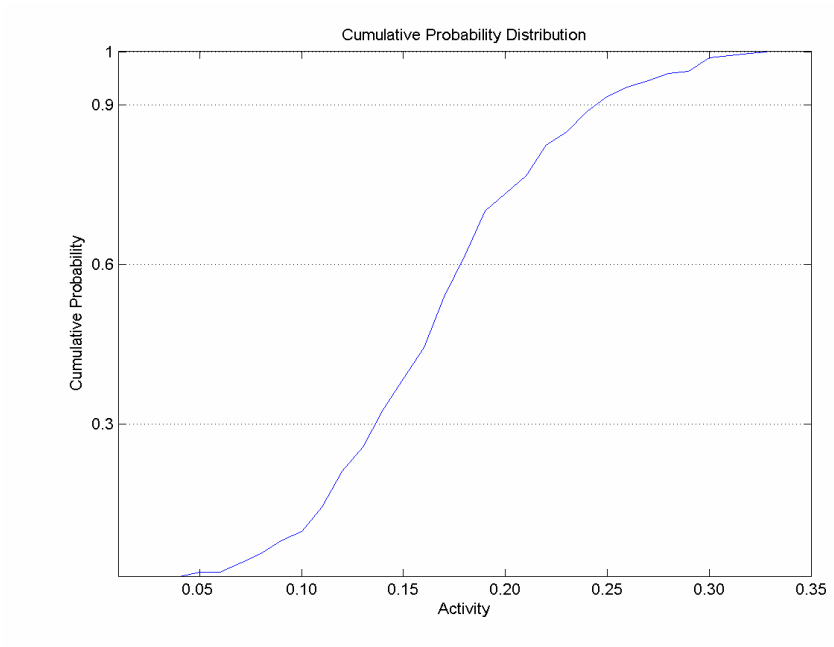

Figure 4.1: Cumulative Probability Distribution of Activity values

4.3.2 MOTION ESTIMATION

The amount of motion in a scene is also one of the factors to be considered when deciding on the parameters to be used for segmenting the video. If a scene has a lot of motion, then it may require higher thresholds for segmentation.

For motion estimation, we use the block matching technique, which is the most common and a widely used technique for motion estimation. Given a pair of successive frames, a given number of macro-blocks are considered in the first frame as shown in the Fig 4.2. Each macro-block in the current frame is matched against candidate macro-blocks within its search area in the next frame. The candidate macro-blocks are just the displaced versions of original macro-block. The best matching candidate block is found using minimum absolute difference (MAD) between the original block and the candidate

block, and its displacement (motion vector) is recorded. The total motion between the two frames is computed as the average displacement of all the macro-blocks.

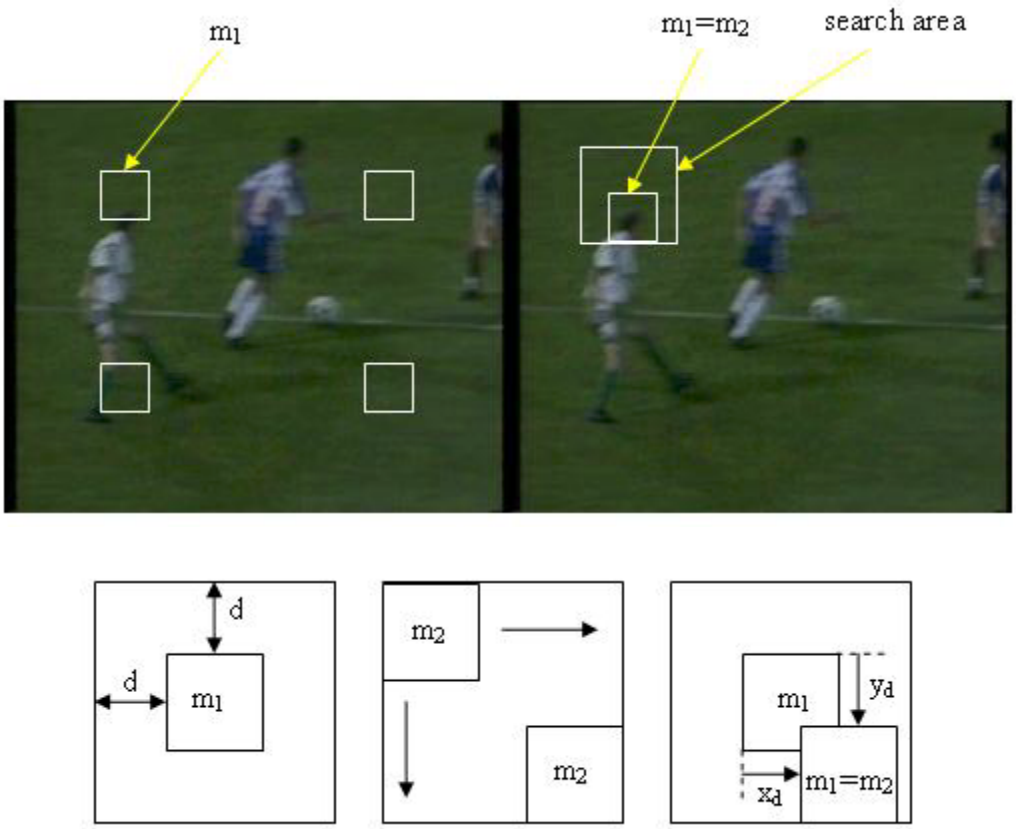

Figure 4.2: Motion Estimation

Let m_i be a macro-block of size $m \times n$ in frame i, and S be the corresponding search area in frame $i+1$ of size $(m+2d)\times(n+2d)$ as shown in figure. The minimum absolute difference (MAD) between the original macro-block m_i and the candidate macro-blocks is computed using Eqn. 4.10.

$$MAD(m_i, m_j) = \sum_x \sum_y \left| m_i(x,y) - m_j(x,y) \right| \tag{4.10}$$

The best matching block, m_2 is determined as:

$$m_2 = \arg\min_{j \in S} \{ MAD(m_i, m_j) \} \tag{4.11}$$

Let (x_1, y_1) be the coordinates of the centroid of macro-block m_1, and (x_2, y_2) be the centroid of the best matching macro-block m_2, then the motion vector of m_1 is determined in terms of displacement from position m1 to m2.

$$d_1 = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4.12}$$

The total motion between the two frames is computed using Eqn. 4.13:

$$\mathrm{M} = \frac{1}{n_b} \sum_{i=1}^{n_b} d_i \tag{4.13}$$

where n_b is the number of macro-blocks. In our experiments, the value of n_b was taken as 4, and d=10.

Motion Classes:

Based on the motion values, a scene can be classified into one of three motion classes: Low, Medium and High motion. Table 4.6 shows the motion classes.

Table 4.6: Motion classes

Motion	Motion Class	Description
0 – 4.71	I	Low motion
4.72 – 9.42	II	Medium motion
9.43 – 14.4	III	High motion

The range of motion values for each class is determined by uniformly dividing the range 0 – 14.4 into 3 partss.

4.3.3 SCENE CLASSIFICATION

Using the activity and motion class, we define a simple mapping function $f(.)$ to define the overall scene class.

$$S_c = f\big(A_3(I_i), \mathrm{M}(I_i, I_{i+1}),\big) = \begin{cases} I, & (A_3, \mathrm{M}) = (I, I) \\ II, & (A_3, \mathrm{M}) = (II, I) \\ III, & (A_3, \mathrm{M}) = (I, II) \\ IV, & (A_3, \mathrm{M}) = (III, I) \\ V, & (A_3, \mathrm{M}) = (I, III) \\ VI, & (A_3, \mathrm{M}) = (II, II) \\ VII, & (A_3, \mathrm{M}) = (III, II) \\ VIII, & (A_3, \mathrm{M}) = (II, III) \\ IX, & (A_3, \mathrm{M}) = (III, III) \end{cases}$$

Hence, the scenes in a video sequence are classified into 9 scene classes. Fig. 4.3 gives the pictorial representation of the above mapping function.

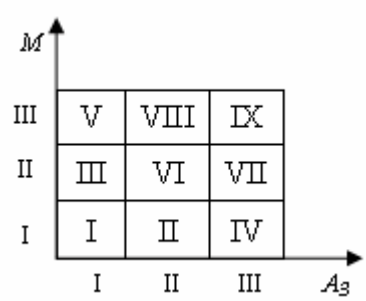

Fig 4.3: Scene Classes

4.3.4 ADAPTIVE FEATURES AND ADAPTIVE SCENE THRESHOLDS

Having characterized and classified the scene based on the motion and activity measures, the next question is to determine the parameters for video scene partitioning for a given scene. Formally, given a video scene s_j, we classify it into a certain scene class , c_i $c_i \in \{I, II, III, ..., IX\}$. The problem of scene-level adaptation then is to determine the parameter set (i.e. the (w, τ) pair) that will produce the best results for all scenes, $s_j \in c_i$, $\forall i, j$. Here, best results are defined in terms of information retrieval measures of precision and recall.

We take a pragmatic approach to the problem of determining the parameters. Using a training set of video scenes, we use a simple clustering technique to determine

the (w,τ) pairs that produce the best results for each scene class in the training set. We then use these pairs for analysis of the test video sequences.

Let $P=(w,\tau)$ be the weight-threshold pair that defines the parameter set for video segmentation. Let N be the number of video sequences used for the training set. We use the edge-response vectors to analyze the video scenes in the training set, using all the available weights and thresholds (i.e. 32 weights, and 9 thresholds in all). Let P_j^c denote the set of (w,τ) pairs that produced correct partitioning results for the class *c* scenes in video sequence *j*. To select the best (w,τ) pair for a given scene class, *c*, all we need is the intersection of P_j^c, for all the *N* sequences:

$$P^c = P_1^c \cap P_2^c \cap P_3^c \cdots \cap P_N^c \tag{4.14}$$

When we have $\left|P^c\right| > 1$, then any member of P^c can be used as the best parameter set. The major problem is when $P^c = \varnothing$, that is, the intersection is empty, implying that no single parameter set always produced correct results for all the class *c* scenes in the test sequences. Two approaches can be used to address this problem.

For each scene in a given video sequence, we define an array $a_{i,j}$, $i = 1,2,...,w_{\max}, j = 1,2,...,\iota_{\max}$, such that $a_{i,j} = 1$ if the scene is correctly partitioned with the parameter set (i,j) pair, and $a_{i,j} = 0$ otherwise. We use $w_{\max} = 32, \iota_{\max} = 9$ in our implementation. Let $a_{i,j}^c(k)$ denote the $a_{i,j}$ arrays for all the class *c* scenes in video sequence *k*. Then, the best parameter set for the class *c* scenes is determined as :

$$P^c = \arg\max_{i,j}\left\{a_{i,j}^c\right\}, \text{ where, } a_{i,j}^c = \sum_{k=1}^{N} a_{i,j}^c(k) \tag{4.15}$$

The above selects the parameter set that produced the best overall result, over all the scenes of a given class in the training set. This could be dominated by one video sequence that has many scenes of the given class. A variation could be to use the parameter set that produced the best result over the scenes of a given class from most of

the video sequences, although it may not necessarily produce the best results over all the scenes. That is,

$$P^c = \arg\max_k \{P^c(k)\}, \text{ where, } P^c(k) = \arg\max_{i,j} \{a^c_{i,j}(k)\} \tag{4.16}$$

A similar problem can arise in determining the thresholds. Here, one can choose the average threshold when two or more thresholds are producing the best result, or to choose the minimum threshold to avoid potential false misses in the segmentation. Table 4.8 shows the adaptive parameters determined for each of the 9 scene classes using a training data set shown in Table 4.7.

Table 4.7: Distribution of scene classes in the training data set

Scene Class	ANTELOPE	JOURNAL	LAS	Total
I	4	19	8	**31**
II	7	11	17	**35**
III	10	14	8	**32**
IV	2	10	5	**17**
V	3	4	2	**9**
IV	3	5	2	**10**
VII	0	2	1	**3**
VIII	1	2	1	**4**
IX	2	0	0	**2**
Total	**32**	**67**	**44**	**143**

Table 4.8: Weight-Threshold pairs for the scene classes

Scene Class	Weights (*w*)	Threshold (*t*)
I	12	0.22
II	9	0.18
III	13, 24	0.15, 0.22
IV	12	0.22, 0.33
V	1	0.18, 0.22
VI	15	0.33
VII	28, 29	0.37
VIII	6	0.45
IX	2	0.41, 0.45

4.4 ADAPTIVE VIDEO SEGMENTATION

From the foregoing, we can describe the steps involved in adaptive video segmentation as follows:

1. Calculate the activity and motion measures for the first frame in the video.
2. Determine the scene class for the first scene based on the activity and motion measures.
3. Calculate the *distance* between the successive frames using the feature set pre-determined for the given scene class. Compare the *distance* using the pre-determined scene threshold for the given scene class.
4. If the *distance* between the two frames exceeds the scene threshold, record a scene cut.
5. Repeat the above procedure on the first frame in the next scene, until the entire video sequence is analyzed. Hence all the scene cuts are detected by tuning the parameters according to the scene classes.

CHAPTER 5: EXPERIMENTAL RESULTS

5.1 EXPERIMENTAL DATA AND EXPERIMENTAL ENVIRONMENT

5.1.1 EXPERIMENTAL DATA

Apart from Canyon and Crops, the other video sequences used in the experiments were taken from MPEG-7 videos. Table 5.1 shows the list of the videos used in the experiments:

Table 5.1: Summary of the experimental data

Video	Duration min:sec	Total # of Frames	Total # of Scene Cuts	Frame Size	Description
ANTELOPE	4:10	6271	32	288x352x3	Wild Life
CANYON	1:46	3183	18	240x352	Documentary
CROPS	1:31	2198	13	60x160x3	Farming
CULTURE	6:00	9000	33	288x352x3	Documentary
DIARY	4:13	6337	29	288x352x3	News
JOURNAL	6:04	9100	67	288x352x3	News, Sports
LAS	4:05	6142	44	288x352x3	Documentary
Total	**27:49**	**42231**	**236**		

5.1.2 EXPERIMENTAL ENVIRONMENT

The experiments were carried out in a MATLAB 6.5 environment, using Zenith personal computers with AMD Athlon[TM] MP 1800+, AT/AT Compatible, dual processor, running at 1.5 GHz with 1GB RAM.

5.2 PERFORMANCE MEASURES

The performance of our method is evaluated using Precision and Recall measures. Precision is defined as the proportion of the correct scene detections out of the total scene detections. Let C be the set of all the correctly detected scenes, and S be the set of all the scenes detected by the system, then precision Pr is given as:

$$\Pr = \frac{|C|}{|S|}$$

Recall is defined as the proportion of the correct scene detections out of the total true scene cuts. Let R be the set of true scene cuts, then recall Rc is given as:

$$Rc = \frac{|C|}{|R|}$$

We measure the overall system performance in terms of the average precision and recall, $PR = \frac{(\Pr + Rc)}{2}$. An ideal system will perform at $\Pr = 1$ and $Rc = 1$. The performance can also be measured in terms of the time taken for the segmentation process.

5.3 RESULTS FOR NON-ADAPTIVE VIDEO PARTITIONING

Table 5.2 shows the results obtained using non-adaptive approach for partitioning the video. The results show average precision to be 92% and average recall to be 88%, which constitute on an average of 90% system performance. This shows that the edge-based features provide a significant measure of the video content, and hence are the important features for video analysis.

Table 5.2: Results for Non-Adaptive Video Partitioning

Video	Scenes	Retrieved	Correct	False	Miss	Pr	Rc	PR
ANTELOPE	32	37	30	7	2	0.81	0.94	0.88
CANYON	18	18	18	0	0	1.00	1.00	1.00
CROPS	13	16	13	3	0	0.81	1.00	0.90
CULTURE	33	21	20	1	13	0.95	0.61	0.78
DIARY	29	25	24	1	5	0.96	0.83	0.90
JOURNAL	67	70	65	5	2	0.93	0.97	0.95
LAS	44	35	35	0	7	1.00	0.83	0.92
Average						**0.92**	**0.88**	**0.90**

5.4 RESULTS FOR ADAPTIVE VIDEO PARTITIONING

5.4.1 ADAPTATION AT VIDEO SEQUENCE LEVEL

Table 5.3 shows the results obtained by the video partitioning algorithm using adaptation at the sequence-level. The last two columns show the weights and the corresponding thresholds used for each video sequence.

Table 5.3: Results for Sequence-level Adaptive Video Partitioning

Video	Scenes	Retrieved	Correct	False	Miss	Pr	Rc	PR	Weight w	Threshold t
ANTELOPE	32	31	30	1	2	0.97	0.94	0.96	12	0.26
CANYON	18	18	18	0	0	1.00	1.00	1.00	15,18, 25,29	0.26,0.30, 0.33,0.37
CROPS	13	13	13	0	0	1.00	1.00	1.00	15	0.37
CULTURE	33	33	33	0	0	1.00	1.00	1.00	9,23	0.18
DIARY	29	30	29	1	0	0.97	1.00	0.99	12,15, 23	0.22
JOURNAL	67	71	67	4	0	0.94	1.00	0.97	23	0.3
LAS	44	45	44	1	0	0.98	1.00	0.99	23,27, 28	0.18
Average						**0.98**	**0.99**	**0.99**		

We can clearly see that different video sequences use different weights to give the best performance in terms of scene cuts detection. Also the weight-threshold pair is not fixed. It changes from one video to the other depending on the content of the video. For instance, $w = 12$ uses 0.26 as the threshold for ANTELOPE video sequence, whereas the same w uses 0.22 as the threshold for DIARY video sequence.

The results shown in the table were obtained by running the 32 weights with each of the 9 thresholds, and the best results were recorded in the table. The multiple entries in the weight column in the table indicate that all the listed weights gave the same results. Same is the case with the thresholds. All the listed thresholds gave the same results. Defining the correct weight-threshold pair for a given video sequence is a difficult task because characterizing an entire video sequence into just one type is very difficult.

Hence we go for scene-level adaptivity, where scenes can be easily characterized and classified into scene classes as described in section 4.3.3.

5.4.2 ADAPTATION AT THE SCENE-LEVEL

Table 5.4 shows video partitioning results obtained using adaptation at the scene-level. The scene-level adaptation approach gives 91% performance in detecting scene cuts. This is slightly better than the non-adaptive approach, but worse than the sequence-level adaptation scheme. A better scene classification method could further improve the results.

Table 5.4: Results for Scene-level Adaptive Video Partitioning

Video	Scenes	Retrieved	Correct	False	Miss	Pr	Rc	PR
ANTELOPE	32	35	30	5	2	0.86	0.94	0.90
CANYON	18	19	18	1	0	0.95	1.00	0.98
CROPS	13	14	13	1	0	0.93	1.00	0.97
CULTURE	33	18	18	0	15	1.00	0.55	0.78
DIARY	29	26	23	3	6	0.88	0.79	0.84
JOURNAL	67	70	64	6	3	0.91	0.96	0.91
LAS	44	44	43	1	1	0.98	0.98	0.98
Average						**0.93**	**0.89**	**0.91**

CHAPTER 6: SUMMARY AND FUTURE WORK

6.1 SUMMARY

A scene-adaptive content-based video segmentation technique has been proposed which uses the color feature along with the edge-based features. The significance of edge-based features has been studied, and also experiments have been conducted with different combinations of these features. An adaptive approach has been used both at the video sequence level and at the scene level. From the experimental results, it is clear that the video-level adaptivity gives better results than a non-adaptive approach. The scene-level adaptivity provides a more automated method for choosing the analysis parameters, and also performs better than the non-adaptive scheme. The multi-resolution approach helps us compare the video frames in a better way taking into account both local and global changes in the video. With the proposed approach, the system could successfully detect 98% of all the scene cuts in a video using sequence-level adaptation, and 91% using scene-level adaptation.

6.2 FUTURE WORK

The efficiency of the system can be improved by introducing a temporal skip factor. This will result in a smaller number of frames that need to be processed. Adaptivity can also be introduced at resolution-level in order to get more accurate results. Further research is needed in order to determine scene thresholds dynamically from the scene content rather than by pre-determining them using a training data set. This can lead to more accurate results. Also the effect of combining edge-features with texture and shape features can be studied. The proposed approach for the detection of gradual transitions needs further study.

REFERENCES

1. D. A. Adjeroh and M. C. Lee, "Robust and efficient transform domain video sequence analysis: An approach from the generalized color ratio model", *Journal of Visual Communication and Image Representation*, 8, 2, 182-207, 1997.
2. D. A. Adjeroh and M. C. Lee, "Scene-adaptive transform domain video partitioning", *IEEE Transactions on Multimedia,* 6, 1, 58-69, 2004.
3. G. Ahanger and T. D. C. Little, "A survey of technologies for parsing and indexing digital video", *Journal of Visual Communication and Image Representation*, 7, 1, 28-43, 1996.
4. F. Arman, A. Hsu and M-Y Chiu, "Image processing on encoded video sequences", *Multimedia Systems*, 1: 211-219, 1994.
5. P. Bouthemy, M. Geglon and F. Ganansia, "A unified approach to shot change detection and camera motion characterization", *IEEE Transactions Circuits & System for Video Technology*, 9, 7, 1030-1044, 1999.
6. J. Canny, "A computational approach to edge detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8, 6, 679-698, 1986.
7. Z. Cernekova, C. Nikou, I. Pitas, "Shot detection in video sequences using entropy-based metrics", *International Conference on Image Processing 2002 (ICIP2002)*, Vol. 3, pp. 421-424, 2002.
8. J. D. Courtney, "Automatic video indexing via object motion analysis", *Pattern Recognition*, 30, 4, 607-627, 1997.
9. F. Dufaux and F. Moscheni, "Motion estimation techniques for digital TV: a review and a new contribution", *IEEE Proceedings*, 83, 6, 858-876, 1995.

10. A. Hampapur, R. C. Jain, and T. Weymouth, "Production model based digital video segmentation", *Multimedia Tools and Applications*, Vol.1, No. 1, pp. 9-46, Mar.1995.

11. M. K. Mandal, F. Idris, and S. Panchanathan, "A critical evaluation of image and video indexing techniques in the compressed domain", *Image and Vision Computing*, 17, 513-529, 1999.

12. A. Nagasaka and Y. Tanaka, "Automatic video indexing and full-motion search for object appearances", *In Proc. Second Working Conference on Visual Databases Systems*, pp. 113-127, Sept. 1991.

13. J-H Park, S-Y Park, S-J Kang and W-H Cho, "Content-based scene change detection of video sequence using hierarchical hidden markov model", *The 6th International Conference on Discovery Science*, Sapporo, Japan, October 17-19, 2003.

14. S. Porter, M. Mirmehdi and B. Thomas, "Detection and Classification of Shot Transitions", *Proceedings, British Machine Vision Conference*, Manchester, UK, 2001.

15. S. W. Smoliar and H. J. Zhang, "Content-based video indexing and retrieval", *IEEE Multimedia*, 1, 2, 62-72, 1994.

16. B-L. Yeo and B. Liu, "Rapid scene analysis on compressed video", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 5, No. 6, pp. 533-544, December 1995.

17. Y. Yusoff, W. Christmas and J. Kittler, "Video shot cut detection using adaptive thresholding", *Proceedings, British Machine Vision Conference*, Bristol, UK, September 11-14, 2000.

18. R. Zabih, J. Miller, and K. Mai, "A feature-based algorithm for detecting and classifying production effects", *ACM Journal of Multimedia Systems*, Vol. 7, pp. 119–128, 1999.

19. H. J. Zhang, A. Kankanhalli, and S. Smoliar, "Automatic partitioning of full-motion video", *Multimedia Systems*, Vol. 1, No. 1, pp.10-28, 1993.

20. H-J Zhang, L. C. Yong, S. W. Smoliar, "Video partitioning and browsing using compressed data", *Multimedia Tools and Applications*, 1, 1, 91-111, 1995.