AUTOMATIC METHODS FOR CONTENT-BASED ACCESS AND SUMMARIZATION OF VIDEO SEQUENCES

A Thesis

Submitted to the Faculty

of

Purdue University

by

Cuneyt Murat Taskiran

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2004

Anneme
sınırsız fedakarlıkları ve desteği için

Elif'ime
"*Car elle me comprend . . .*"

ACKNOWLEDGMENTS

In German one's Ph.D. advisor is known as *Doktorvater* — Ph.D. father, an apt term for the master–aprentice relationship between the doctoral candidate and the poor soul who is advising him. During the time I walked in Professor Delp's office sometime in January 1997 and the time he finally signed my dissertation, I came to appreciate just how important this relationship is. If I have finally finished my studies without being "sent home in a box," as he nicely puts it, it is because of the infinite patience and great support that Professor Delp has shown me. I want to thank Professor Edward J. Delp III for all he has taught me, academic and otherwise.

I would like to thank my committee members, Professor Jan Allebach, Professor Charles Bouman, and Professor Zygmunt Pizlo, for their generous support. If Professor Allebach had not believed in me when we first met at a Conference in Cappadocia in 1995 and had not been instrumental in finding me a teaching assistantship position, I would have never been able to come to Purdue. I thank him for giving me this wonderful opportunity. Professor Bouman inspired me many times during our long and sometimes heated discussions. I would also like to thank Professor Pizlo for his generous help in the video summarization study.

I am grateful to the C-SPAN Archives for funding me during my research at Purdue. I would like to extend my thanks to Professor Robert Browning, who was ever so patient when the project delivery dates got delayed.

Although I took only one course from him, Professor Ray DeCarlo has taught me many things about how to be a good teacher, I thank him for that. Me and Elif enjoyed the wide-ranging discussions and the hospitality of Professor and Mrs. DeCarlo.

I have stayed in EE32 for five long years. It was my fellow officemates Eugene T. Lin, Yuxin (Zoe) Liu, Hyung Cook Kim, Jinwha Yang, Jennifer Talavage, Gregory

W. Cook, and my fellow VIPER labmates Aravind K. Mikkilineni, Anthony F. Martone, Michael Igarta, Limin Liu and Zhen Li, and Yajie Sun who made this time not only bearable but very enjoyable. My special thanks go to Eugene who answered innumerable C, C++, and other programming-related questions. I have learned a lot from him. I would also like to thank Zoe for being the joy of EE32 and for tolerating me all these years. Part of the joy of being in EE32 was the no-holds-barred discussions about any subject conceivable to Man, with Eugene, Zoe, and Jinwha, ranging from the sugar consumption in Asians to religion. I would like to thank my summer officemate and roommate, Rafael Villoria, for bringing a Spanish flavor to EE32.

I want to thank all my family, my parents Ayla and Güngör Taşkıran, my sister Sibel Kandemir, my brother-in-law Levent Kandemir, my uncle and aunt Atilla and Sema İmrahor İlyas, and my cousin Mehmet Ali İmrahor İlyas for supporting me all these years. If it were not for the sacrifices made by my mother, sister, and uncle, I would not be here today. I dedicate this work to them.

Finally, I thank my wonderful wife, Elif Ekmekçi-Taşkıran, for all that she has done for me. She has been my safe haven all through these years. I would not be able to finish Ph.D. without her. Depending on what my deadline demanded, she has been my critic, secretary, experimental assistant, proofreader; but more importantly she has been a true emotional anchor to me.

TABLE OF CONTENTS

Page

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Taskiran, Cuneyt M. Ph.D., Purdue University, December, 2004. Automatic Methods for Content-Based Access and Summarization of Video Sequences . Major Professor: Edward J. Delp.

Depending on the specific information they are seeking, users desire flexible and intuitive methods to search and browse multimedia libraries. However, the cost of manually extracting the metadata to support such functionalities may be unrealistically high. Therefore, over the last decade there has been a great interest in designing and building systems that automatically analyze and index multimedia data, and retrieve its relevant parts.

In this work we describe algorithms that facilitate browsing, searching, and summarization of video sequences. We propose shot transition detection algorithms for the detection of cut and dissolve types of shot transitions based on a binary tree regression classifiers framework. Our system is able to detect these transitions with high accuracy.

We discuss stochastic models to model video program genres, such as news programs or sitcoms, and show how these can be applied to automatically detect the genre of a given program. We investigate the use of hidden Markov Models (HMMs) and stochastic context-free grammars (SCFGs) for modeling. Since the computational complexity of SCFG training is high, we develop a hybrid HMM-SCFG model that reduces the training time of the models considerably.

Deriving compact representations of video sequences that are intuitive for users and let them easily and quickly browse large collections of video data is fast becoming one of the most important topics in content-based video processing. Such representations, which we will collectively refer to as *video summaries*, rapidly pro-

vide the user with information about the content of the particular sequence being examined, while preserving the essential message. We propose an automated method to generate video skims for information-rich video programs, such as documentaries, educational videos, and presentations, using statistical analysis based on speech transcripts that are obtained by automatic speech recognition (ASR)from the audio. Ideally one would like the generated summaries to be both detailed and covering most of the important points of the full program they were derived from. Clearly, for high summarization ratios it is impossible to stiff both of these constraints. Our summarization approach quantifies these two concepts and maximizes a weighted sum of both detail and coverage functions to obtain a trade-off between the two. We also study objective evaluation methods for video summaries. We evaluate summaries produced by a number of algorithms using a question and answer evaluation scheme and discuss other methods of summary evaluation.

In the final part of the dissertation we describe a real-world video processing application that makes use of many algorithms introduced in this work. In this application we generate a list of unique people appearing in a news program using a combination of visual and audio features. This system greatly facilitates the indexing of news programs and also may be used as part of a automatic open caption insertion system.

1. INTRODUCTION

1.1 Proliferation of Video Data

Applications for digital video are undergoing an explosive growth. This increase is spurred both by the increasing availability of digital video and escalating consumer demand. The availability of digital video has rapidly increased due to developments in low-cost storage media, improved compression techniques, less costly and more sophisticated input modules, and higher transmission rates. The user demand has been affected by the appearance of diverse applications such as video conferencing, multimedia authoring systems, 3G cellular phones, surveillance systems, and movie-on-demand, among others. The Internet has also played a large role in this increase.

While these applications will generate and use vast amounts of video data, the technologies for organizing and searching through video data are still in their infancy. This presents a great obstacle in the wide-spread use of video data and the generation of huge collections of video data. According to an international survey, there are more than six million hours of feature films and video archived worldwide, with a yearly increase rate of about 10% [1]. This is an large potential waiting to be tapped.

1.2 The Content-Based Video Access Problem

1.2.1 What is "Content"?

Identifying content for a given image or video sequence is not an easy task, even for a human, since these multimedia objects may be classified using a multitude of attribute dimensions. Dyson [2] discusses some problems which exist in retrieving relatively simple symbols from a database. She has found that even when provided

with extremely detailed verbal instructions about a symbol, most of the time people were not able to draw it correctly.

For our purposes, we may define video content as "any property of a video program that the user may wish to query on." Hence, content is dependent on the particular user group querying the system.

1.2.2 Why Content-Based Retrieval?

As pointed above, recent advances has made it possible to build large image and video collections. This immediately brings about two problems: How can the data in such a collection be efficiently indexed into a database structure, and how can the users search efficiently through this database. An initial solution is to index images and video by using textual attributes. However, there are a number of drawbacks of a text-based system [3] some of which are listed below

- The search is completely dependent on the keywords derived at the time of the indexing, unless various high level language processing tools, like thesaurus look-up, are used. Hence, if the query refers to some properties that were not initially described, the search will most likely fail to retrieve the correct item.

- Some image and video properties may be impossible to describe using a linguistic representation.

- Even if all the useful properties of the multimedia content is described using text, since there is no standard image or video description language, a query may not match any of the the index keywords, e.g., an image component that is "curvy" for the indexer may be "wavy" for the user.

- Detailed indexing of data by a human operator is time-consuming. Indexing effort is directly proportional to the granularity of access desired, where a very low-granularity indexing scheme would just have the title of a video program whereas a high-granularity one will have various labels for individual shots.

The time investment for a detailed description of video material is estimated to be 10 hours of indexing effort per hour of video data [4].

Despite these disadvantages text-based systems are still used because they are the only tools available currently. In order to solve this problem, in recent years tremendous attention has been given to developing content-based methods to access video and image data and finding better ways of presenting the results to the user.

1.3 Components and Constraints for Content-Based Access Systems

Researchers working on the content-based media access topics have outlined the following requirements for content-based database management systems [5, 6]:

- *New representation models and their standardization*: There is a need of data models and knowledge structures organizing visual information at different levels of abstraction. These models have to be standardized such that common search engines and applications can gather data from a variety of databases. The MPEG-7 standard1 [7] was designed to solve this problem.

- *"Nearly-automated" indexing tools*: The indexing process for new data has to be automated as much as possible. Fully automated indexing is perhaps neither feasible nor desirable, but when one needs to index hundreds of hours of video, every operation that is automated is of importance. The indexing tools have to be able to relate some basic semantic properties of the data to low-level features fairly reliably.

- *Intuitive navigational tools*: Search interfaces and controls have to be simple and intuitive. Users use on the average 1.5–2.5 keywords per query [8] and most of them cannot master Boolean search. Browsing and navigating in a large database can be very disorienting unless the user can form a mental picture of the entire database. Only having an idea of the surroundings can offer indication of where to go next. The wider the perceptual horizon, the more

secure the navigation will be [9]. Humans are not good in digesting information from different levels of granularity simultaneously, i.e., performing depth-first search, but are much better in doing breadth-first search. Therefore, the level of the granularity of the search has to be intuitive and should be controllable by the user.

- *Different performance criteria*: It is observed on the World Wide Web that users care more for precision of the results they get from a query than knowing how many relevant items they missed. However, when the queries become ambiguous, the false alarms might correspond to data that the user didn't request but they might still find useful and a somewhat higher rate of false alarms than a typical classification application may be accepted.

- *Speed*: Users will often sacrifice accuracy for speed. The indexing and retrieval techniques has to scale up to hundreds of hours of video and tens of thousands of images. This requires efficient data storage and search strategies.

1.4 Using Non-Visual Information to Analyze Content

There exists other modes of information besides image data that the video signal provides which may give valuable information about program content. These information sources include the audio track, closed-caption text, and captions that appear in the video. Although these additional modes of gathering information about video content are valuable, they will not be discussed further in this thesis since our emphasis is on using image data.

1.4.1 Information from the Audio Track

In general shot boundaries tend to have corresponding discontinuities in the audio signal. In news programs, soap operas, talk shows, and the like, a change of the speaker is generally accompanied by a shot transition. This fact may be used to

make the shot boundary detection task much more robust. Audio information would especially be of great help in detecting shot boundaries where the visual content changes very little.

Nam, Cetin, and Tewfik [10] use speech data along with visual data to segment news video clips by detecting speaker changes that correspond to either the start of a news item or the change of a subject. After pauses are detected and voiced/unvoiced distinction is performed, voiced phonemes are extracted from voiced speech segments. Cepstral coefficients determined from the sub-band analysis of these segments are used as features. Speaker classification is performed using a hidden Markov model structure.

Patel and Sethi [11] use the sub-band information encoded in the audio MPEG stream. They derive features like average energy, band energy ratio, pause rate, and pitch from the MPEG coded audio signal to label shots as {dialog, non-dialog, silent}.

Saraceno and Leonardi [12] label audio segments using labels from the set {silence, speech, music, noise}. Silence detection is performed by processing the signal energy using a finite state machine. Speech and music detection is done by evaluating an autocorrelation measure. They have found that information on silence of audio segments can make the cut detection process much more robust, especially for news and commercial sequences.

1.4.2 Closed-Caption Information

Closed captions are captions that are normally not visible and are hidden in line 21 of the vertical blanking interval. The Television Decoder Circuitry Act mandates, since July 1993, all televisions manufactured for sale in the U.S. to contain a built-in closed-caption decoder if the picture tube is 13 inches or larger. Closed-captioning on virtually all television programming will be mandatory by January 2006 [13].

Among researchers, processing the closed-caption information has become more popular than using cues derived from the audio signal. This popularity is due to the following factors: It is very easy to extract, store, and process closed-caption data since it consists of plain text. Also, it immediately gives us a transcript of the audio and provides valuable cues about the structure through the use of special markers. Finally, efficient methods exist to process text to obtain keywords. However, one should be aware of the fact that using simple-minded closed caption analysis may lead to incorrect results. For example, during a baseball game the commentators may be talking about the exploits of Babe Ruth, who is not present in the game that is being played. Hence, a keyword derived from closed-caption containing "Babe Ruth" will incorrectly index this video sequence [14].

Mohan [15] uses the markers that exist in the closed-caption to segment news stories in news programs. These markers include ">>" for change of speaker, ">>>" for change of story, and markers like "[applause]" for various non-speech sounds. He notes a number of difficulties with this approach: First, due to the live nature of TV news captioning, there generally exists a lag between the audio and the related closed-caption. Second, a large number of mistakes are made during closed-captioning since the captioners are not given the transcript and enter story markers at their discretion. Using the heuristic that a new story almost always starts with a shot transition, shot boundary information and pause detection are used to align the story units with the video.

Nakamura and Kanade [16] spot important keywords and associate these with keyframes extracted from shots. The closed-caption text is automatically classified into the categories {speech/opinion, meeting/conference, crowd, visit/travel}. The keyframes are identified as belonging to one of the categories {face, people, outdoor scene} by an operator. They then find the correspondence between these items by matching the duration times using dynamic programming. A similar word spotting technique is used by Merlino, Morey and Maybury [17] to perform story segmentation for news programs.

1.4.3 Information from On-Screen Graphics

Extracting text from on-screen graphics provides additional information for video analysis and indexing. For example, in news programs the person or the place that is the topic of the news can be unambiguously extracted from the caption in the frame. In this particular application, extracting captions may be the easiest method to obtain such information, when compared with applying speech recognition of the audio track or performing keyword spotting on the closed-caption information.

There are two problems to solve in order to come up with a robust caption extraction scheme: the low resolution of the characters and the existence of complex backgrounds that may have hue and brightness very close to those of the caption characters.

Kannangara *et al.* [18] use nonlinear morphological filtering to extract letters from frames. They first isolate the text box using heuristics based on empirical analysis and then segment the letters using a vertical projection profile. The letters are then recognized by performing morphological opening on the thresholded text block using structuring elements for each letter. They report this method to be superior to correlation because of the small size of the letters and blurring around letter edges.

Sato *et al.* [19] extract the text regions and then enhance them by by sequentially filtering the caption during the frames that it is present. Characters are segmented using correlation with reference patterns.

1.5 Generation of the Experimental Video Dataset

We have created a large collection of video sequences to be used as a testbed for the video analysis algorithms described in this dissertation. The length of the sequences is variable, some cover 10 or 20 minutes of a program and some cover the whole program.

Storing large amounts of video data mandates the use of compression, since the amount of memory needed may be large. For example, one hour of uncompressed SIF (352×240 for NTSC) resolution video requires approximately 27 Gigabytes of storage space. In addition, retrieving uncompressed video over a network takes a long time. Mainly due to these reasons most video data is kept in compressed format. This makes processing in uncompressed domain unattractive because then the additional steps of decompressing the video to be analyzed and then compressing it back have to be performed each time the video sequence is to be accessed. Another reason which makes processing directly in the compressed domain attractive is that processing algorithms can make use of features which are computed at the time of the compression by the encoder. Because of the above advantages, all the sequences in our video data set is kept in compressed format.

The sequences were digitized using two different hardware encoders by connecting the cable feed directly to these devices. Some older sequences in the database were digitized using the CLM4500 Consumer MPEG Video Encoder from C-Cube Microsystems at a rate of 1.5 Mbits/sec in SIF format (352×240). The video and audio bitstreams created by the hardware were multiplexed using the Vitec MPEG-1 Multiplexer version 1.4 to obtain a system layer MPEG-1 file. The remaining sequences were digitized using a Optiplex VideoPlexPlus encoder card at 2.0 Mbits/sec in SIF format, which directly produces a system layer MPEG-1 file. Each of these hardware devices had an option to perform shot boundary detection while encoding. This option was turned off, since it would interfere with our experiments. Commercials in the sequences, if they exist, were edited out for all sequences in the dataset using the Mediaware M1-edit Pro software.

We have collected sequences in six genres, soap operas, news programs, movies, sitcoms, talk shows, sports programs, and programs from the C-SPAN networks. In addition to the MPEG file, each sequence in our data set has two files associated with it. An information file provides metadata about the sequences such as the content, the channel from which it was digitized, sequence length, and capture time, among

others. A ground truth file lists all the shot boundaries in the sequence. These ground truths were determined by a human operator watching the sequences.

1.6 Outline of the Dissertation

Segmenting a given video sequence into shots is generally the first step in video content analysis. In Chapter 2 we discuss how to detect the locations and types of shot transitions in compressed video sequences using features extracted directly from the compressed bit stream. We propose shot transition detection algorithms for the detection of cut and dissolve types of shot transitions based on a binary tree regression classifiers framework. Our system is able to detect these transitions with high accuracy.

There usually exists a big gap between the low-level features, like color, motion, and texture, that computer vision algorithms can reliably extract from an image and the high-level attributes, like "romance", "landscape", "action", that users want to use to query the database. In Chapter 4 we discuss how this gap may be bridged to an extent using a number of so-called pseudo-semantic shot labels. We then discuss how these labels may be used to build program genre models for content including news programs, sitcoms, and movies. Such models are very important for a number of video analysis applications. In Chapter 4 we describe in detail how these models can be applied to automatically detect the genre of a given program. However, the models can also be applied to detect important parts of video programs and to produce custom-edited video content.

We investigate the use of hidden Markov Models (HMMs) and stochastic context-free grammars (SCFGs) for modeling. Since the computational complexity of SCFG training is high, we develop a hybrid HMM-SCFG model that reduces the training time of the models considerably.

In Chapter 5 we discuss how video programs may be summarized using speech transcripts. Deriving compact representations of video sequences that are intuitive

for users and let them easily and quickly browse large collections of video data is fast becoming one of the most important topics in content-based video processing. Such representations, which we will collectively refer to as *video summaries*, rapidly provide the user with information about the content of the particular sequence being examined, while preserving the essential message. The need for automatic methods for generating video summaries is fueled both from the user and production viewpoints. With the proliferation of personal video recorder devices and hand-held cameras, many users generate many times more video footage that they can digest. On the other hand, in today's fast-paced news coverage, programs such as sports and news must be processed quickly for production or their value quickly diminishes. Such time constraints and the increasing number of services being offered, places a large burden on production companies to process, edit, and distribute video material as fast as possible.

Our goal in Chapter 5 is two-fold: First, we propose an automated method to generate video skims for information-rich video programs, such as documentaries, educational videos, and presentations, using statistical analysis based on speech transcripts that are obtained by automatic speech recognition (ASR)from the audio. We show how important phrases can be automatically extracted even from ASR text that has a relatively high word error rate. These detected phrases may be used to augment current summarization methods and visualization schemes. Ideally one would like the generated summaries to be both detailed and covering most of the important points of the full program they were derived from. Clearly, for high summarization ratios it is impossible to satisfy both of these constraints. Our summarization approach quantifies these two concepts and maximizes a weighted sum of both detail and coverage functions to obtain a trade-off between the two. This approach enables the user to change the weights and regenerate the video summary of a program with more detail or more coverage, depending on a particular application.

Our second main objective in Chapter 5 is to rigorously study the objective evaluation methods for video summaries. We evaluate summaries produced by a

number of algorithms using a question and answer evaluation scheme and discuss other methods of summary evaluation.

In the final part of the dissertation, Chapter 6, we describe a real-world video processing application that makes use of some of the algorithms introduced in this work. In this application we generate a list of unique people appearing in a news program using a combination of visual and audio features. This system greatly facilitates the indexing of news programs and also may be used as part of a automatic open caption insertion system.

2. SHOT BOUNDARY DETECTION

2.1 Introduction

A *shot*, which is the basic component of a video sequence, may be defined as a group of frames that has continuity in some general conceptual or visual sense. Often, a shot is composed of frames which depict the same physical scene, signify a single camera operation, or contain a distinct event or action. Most systems for indexing, characterization, and understanding of video rely on the identification of individual shots; hence, usually the first task in processing video data is segmenting shots by detecting shot boundaries, thereby breaking the video sequence into distinct "semantic chunks". In order to exploit the hierarchical nature of video the detected shots may then be clustered to obtain scenes, which form the next semantic level for video.

The temporal video segmentation problem can be stated as follows: Given a video sequence, segment it temporally into blocks of frames such that interblock variation of frames is much larger than the intrablock variation, using some frame similarity criterion. It should be pointed out that the term "scene change detection", frequently used to refer to the temporal segmentation problem, is a misnomer because what is being segmented out are the shots, not the scenes. We shall use the terms *shot boundary detection* or *shot transition detection* interchangeably to refer to the task of temporal segmentation of video.

For some program content it may be difficult to derive features that reflect the homogeneity of the frames in a shot. A shot may be conceptually continuous but may have large variations in all the features that were proposed in the literature to detect shot boundaries. Furthermore, shots may not represent visually and conceptually homogeneous units and may be punctuated internally, e.g. by using camera

movement. This may cause long shots which consist of sub-shot story units, a common occurrence in movies. These difficulties occur much less frequently in *structured video* where there is a strong spatial order within the individual frames of the shots, and a strong temporal order among the different shots [20]. News programs, with their predictable order of events, are a good example of structured video. Sitcoms and talk shows would be other examples to structured video.

We limit the domain of the shot boundary detection problem first by making the assumption that a shot transition has to be such that it has to contain significant contextual and visual changes. This limitation is necessary because the general detection of contextual changes in video may be an unsolvable problem. In addition, we shall only consider the detection of cuts and dissolves, as these types of shot transitions make up the overwhelming majority of all shot transitions in most video sequences.

The structure of this chapter is as follows: InSection 2.2 we introduce a mathematical model that we use to model abrupt and gradual shot transitions in video programs. A detailed survey of previous work in shot boundary detection is presented in Section 2.3. We then describe the feature extraction process we used and introduce the generalized trace feature vectors in Section2.4 and Section 2.5. In Section 2.6 basic theory of decision tree classifiers is reviewed and in Section 2.7 the application of the decision tree methodology to the shot boundary detection task is described. Results for shot boundary detection experiments are presented in Section 2.8. Finally, conclusions are provided in Section 6.7.

2.2 The Shot Transition Model

In this section we introduce the shot transition model that forms the basis of shot boundary detection approach. We assume each video sequence in our video dataset has a unique identifying number k and we use Σ^k to denote the video sequence. We will refer to the j^{th} shot in the sequence Σ^k by S_j^k. The number of shots in sequence

Σ^k is N_k. We will denote the i^{th} frame from the sequence Σ^k as $f_i^k(x, y)$, where usually the spatial indices will be suppressed and we will write f_i^k instead. When we focus on the processing of a single sequence and its identifier number is irrelevant in the discussion, we will suppress the sequence number subscripts and will write Σ, S_j, and f_i. For each shot S_j we define $b(S_j)$ and $e(S_j)$ to be the frame numbers of the first and last frames in the shot, respectively.

Shot transitions are divided into two types, *abrupt shot transitions*, where the last frame of one shot is followed immediately by the first frame of the next shot, and *gradual shot transitions*, where the transition from one shot to the other is spread out over a number of frames. An abrupt shot transition is also known as a *cut*.

There are many different forms of gradual shot transitions, the most frequent being fade ins, fade outs, and dissolves. A *fade in* is a shot transition where the frame is dark in the beginning of the transition. Gradually an image appears, brightening to full strength. A *fade out* is the opposite of a fade in. A *dissolve* is the superposition of a fade out and a fade in. In this type of shot transition the pixels of the image from one shot is gradually replaced by the image from the next shot.

We define a shot transition between shots S_j and S_{j+1} using the following video edit model

$$f_i = \begin{cases} g_i, & i < s \\ \left(1 - \frac{i-s}{T}\right) g_i + \left(\frac{i-s}{T}\right) h_i, & s \leq i \leq e \\ h_i, & i > e \end{cases} \tag{2.1}$$

where $T = e - s$ is the duration of the shot transition. For dissolves we have $g_i \in S_j$ and $h_i \in S_{j+1}$. Cuts are defined similarly with the additional constraint that $e = s + 1$. For a fade-out we have $S_{j+1} \equiv S_B$ where S_B is a shot containing the same monochrome frame; for a fade-in, the reverse is true, i.e., $S_j \equiv S_B$. Some frames from a dissolve and a fade out type of shot transition are shown in Figure 2.1.

(a) (b)

Fig. 2.1. Selected frames from (a) a dissolve and (b) a fade-out shot transition

2.3 Previous Work in Shot Boundary Detection

Since shot boundary detection is a fundamental component of video processing, considerable work has been done in the last decade in this topic. The general approach is to derive one or more low-level features from video frames, derive a dissimilarity value between frames using these features, and flag a shot transition whenever this value shows some nontypical behavior. The following is a list of some of the previous techniques that have been used in the literature. More references may be found in the survey and comparison papers [21–25].

2.3.1 Methods Based on Pixel-Wise Frame Differences

In order to detect cuts, some researchers have used the difference of pixel values averaged over the whole frame as a similarity feature between frames [26]. Shahraray [27] has proposed dividing the frame into blocks and finding the "best" matching blocks between frames for comparison, similar to the block matching technique of MPEG. An order-statistic filter is used to derive the image match value which is then processed by a finite-state machine to detect cuts. Yeo and Liu [28] use the pixel differences of the luminance component of DC frames in an MPEG sequence. Nam, Cetin, and Tewfik [10] use a method which is essentially the same as Yeo and Liu's method, but instead of the DC frames derived from MPEG video they use coarse reduced frames obtained from the 2-D wavelet transform of uncompressed frames. Ardizzone and La Cascia [29] process the pixel differences from two frames using a multi-layer perceptron to decide if they belong to the same shot.

2.3.2 Methods Based on Color Histograms

Other methods have been proposed based on obtaining the color histograms of frames, followed by the application of some form of similarity metric to the frame

histograms in order to detect shot boundaries. Histogram intersection, which is equivalent to the L_1 metric for normalized histograms, is frequently used.

Patel and Sethi [30] have experimented with various statistics and have found that the χ^2 test gives the best performance. They use the intensity histograms computed for the entire frame. The histograms are found using DC coefficients of MPEG video for only I frames assuming that every 12^{th} frame is an I frame. Ferman and Tekalp [31] use the sum of histogram differences for the Y, U, and V components. They first detect candidate cut locations by performing 2-class clustering on a coarse, temporally sampled version of the video. For each candidate location a local window is placed and 2-class clustering is performed within the window to determine the precise cut location.

2.3.3 Methods Based on Dimensionality Reduction Techniques

Idris, Kobla and Doerman [32] place DC coefficients for each frame from an MPEG coded sequence in a feature vector. They then reduce the dimensionality of these feature vectors to three by multi-dimensional scaling methods to obtain a set of points in three dimensions, which they call the *VideoTrail*. This set is then processed to obtain shot boundaries. Han and Tewfik [33] use a similar technique in uncompressed domain. They subsample each frame and form a feature vector by grouping a number of subsampled pixel values from the frames within a time window. The KL transform is then used to reduce the dimensionality to 5. These low dimensional vectors are used to locate shot boundaries.

2.3.4 Methods Based on Edge Detection

Based on the observation that during a shot transition the location of appearing edge pixels would be very different from old edge pixel locations, Zabih, Miller, and Mai [34] have proposed a cut detection scheme based on edge detection. Entering and exiting edge pixel number change fractions are found and a global threshold is

used to detect cuts. Shen, Li, and Sethi [35] have applied this technique to MPEG sequences using multi-level Hausdorff distance histograms.

2.3.5 Methods Based on Motion Vector Information

Kobla, Doerman, and Lin [36], perform cut detection for MPEG coded sequences using what they call the resemblance value between frames, which is derived from the numbers of each type of macroblocks in the compressed frames. Zhang *et al.* [37] detect cuts by thresholding the minimum of the counts of forward and backward non-zero motion vectors. Meng, Juan, and Chang [38] define various ratios of the number of macroblocks with forward, backward, and no motion compensation to perform cut detection for P and B frames of a MPEG sequence.

2.3.6 Model-Based Techniques

The techniques discussed above have posed the problem of temporal video segmentation as detecting shot transitions in arbitrary image sequences. However, defining models for video which capture its structure may provide valuable constraints for video segmentation. A number of researchers have investigated the merits of a video model in the detection of shot boundaries.

Hampapur, Jain, and Weymouth [39] have come up with a video edit model which is based on a study of the video production process. Based on this model, they introduce what they call chromatic images which are computed by dividing the change in gray level of each pixel between two images by the gray level value of that pixel in the second image. During fades, this chromatic image attains a reasonably constant value. Song, Kwon, and Kim have generalized this technique to detect dissolves [40]. They show that during dissolves, the first time partial derivative of the sequence will be "large" while the second derivative will be "small". Aigrain and Joly [4] develop a model for the probability density function of intershot pixel differences in successive frames. They then detect shot transitions by fitting

this model to pixel difference data obtained from the sequence. Vasconcelos and Lippman [41] have modeled the time duration between two shot boundaries. Using a Bayesian model and the Weibull prior distribution, they have derived a variable threshold to detect shot boundaries.

Rather than trying to model the characteristics of shots, one can try to model the application domain on which a particular video segmentation algorithm will operate. Knowledge-based approaches to structured video domains include the work of Zhang *et al.* [42] where anchorperson shots are found by examining intrashot temporal variation of frames. Swanberg, Shu, and Jain [20] used a similar approach using template matching.

2.3.7 Miscellaneous Methods

In the method used by Arman, Hsu, and Chiu [43] a fixed number of connected regions is selected and some predetermined collection of DCT AC coefficients from these regions are used to form a feature vector for the frames of motion JPEG coded video. Normalized inner products of these vectors are compared with a global threshold to detect shot transitions. Idris and Panchanathan [44] use vector quantization to compress video sequences using a codebook of size 256 and 64-dimensional vectors. The histogram of the labels computed from the codebook for each frame is used as the similarity measure and the χ^2 statistic is used to detect cuts.

2.3.8 Comparisons Between Shot Boundary Detection Techniques

Boreczky and Rowe [24] compare histogram difference methods both using a single threshold and the twin comparison technique of Zhang *et al.* [45], together with motion compensated pixel differences and the DCT vector inner product method of Arman, Hsu, and Chiu [43]. After tuning the algorithms on a small set of data, they ran them on a larger database containing 419745 frames with 2507 cuts and 506 gradual transitions. This database included television programs, news, movies,

and commercials. They found that the histogram difference method with a single threshold gave consistently good results. They also report that none of the algorithms performed well in detecting gradual scene transitions.

Dailianas, Allen, and England [25] have performed comparisons between methods based on histograms using various similarity measures, Aigrain and Joly's [4] pixel difference modeling method, the edge detection method of Mai, Miller, and Zabih [34], and a method based on moment invariants computed from pixel values. Their database consisted of news, a movie, and a training video containing 1141 cuts and 291 gradual transitions. They have found that it was difficult to distinguish wipes from large object motion in the shots, and that almost all algorithms had a high false alarm rate.

Lienhart [22] compared algorithms that use color histogram differences and the edge detection method of Mai, Miller, and Zabih [34] together with two algorithms specialized for gradual transition detection: the variance of pixel values method proposed by Meng, Juan, and Chang [38] and the edge-based contrast method. His data set had a total length of 173 minutes and consisted of a feature film, a news program, one episode of *Baywatch* and a specially generated sequence containing many dissolve transitions. The dataset had 1896 cuts and 424 gradual transitions. He has found that the edge detection method was very sensitive to motion, generated a high number of false alarms, and had great computational burden. Very high detection rates with low false alarm rates are reported for cuts and fades. The performance in detecting dissolves were less satisfactory.

Kobla, DeManthon, and Doerman [46] compare the performance of three algorithms based on their *VideoTrails* method [32] in detecting gradual transitions with the following algorithms after some modifications: The plateau detection algorithm proposed by Yeo and Liu [28], the variance of pixel values method of Meng, Juan, and Chang [38], the twin comparison histogram differencing method of Zhang *et al.* [45], and the chromatic edit model of Song, Kwon, and Kim [40]. They performed two experiments: In the first one, the algorithms were ran on a sequence consisting of

two unrelated 10 second sequences which were combined using 15 different types of special effects, most of which were nonlinear. In the second one, the algorithms were ran on approximately 50 minutes of video having diverse content and which had 441 special effect edits. The *VideoTrails*-based methods performed better in detecting highly nonlinear special effect edits in the first experiment but gave almost the same performance as the plateau, twin-comparison, and pixel variance methods in the second experiment. The performance of the plateau method depended a lot on the windowing parameter. Motion or the appearance of objects can offset the performance of the twin-comparison algorithm greatly. The chromatic edit method did not perform well using the DC images derived from MPEG video.

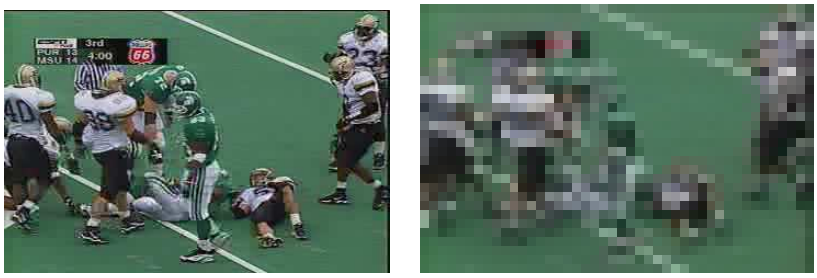
2.4 Feature Extraction for Shot Boundary Detection

In this section we describe the features that are extracted from video sequences to be used for the shot boundary detection task.

When extracting features from video frames to be used for the shot boundary detection task, one can choose to gather information from the whole frame or the so-called *DC frame* may be used. A DC frame consists of the DC coefficients of the two-dimensional discrete Fourier Transform (2D-DCT) of a video frame. For each 8×8 block used by the MPEG compression algorithm the DC coefficient is defined as

$$F_i(0,0) = \frac{1}{8} \sum_{n=0}^{7} \sum_{m=0}^{7} f_i(m,n) \tag{2.2}$$

from which we see that the DC coefficient for the 2D-DCT of each 8×8 block of the image is proportional to the mean value of pixel values of that block. Therefore, the DC frame, which consists of 2D-DCT DC coefficients of a frame, is a decimated version of that frame within a constant. A frame and its corresponding DC frame is shown in Figure 2.2.



original frame extracted DC frame

Fig. 2.2. An original decompressed frame and the corresponding DC frame.

Although using DC frames speeds up feature extraction, their lower resolution decreases the performance of shot boundary detection algorithms. Therefore, in this work we used features extracted from the whole frames.

2.5 The Generalized Trace Feature Vector

Given a sequence Σ we extract a number of feature from each frame f_i in the sequence to be used for shot boundary detection. The feature vector corresponding to f_i is denoted by $\mathbf{X}_i = [x_{i1}, \ldots, x_{in}]^T$, which we refer to as the *generalized trace* (GT). The components of the GT vector are quantities that measure the dissimilarity between consecutive frames. In the following section we describe the importance of each of these features for the shot boundary task and how the features are calculated in detail.

2.5.1 Luminance Histogram Intersection Feature

This luminance histogram reflects the lightness distribution within a frame. Frames belonging to different shots usually have different distributions of luminance, hence their luminance histograms show marked differences. Histograms are invariant to translation and rotation about the viewing axis, and change only slowly under change of angle of view, change in scale, and occlusion [47]. These properties make luminance

histograms robust indicators of frame content, which makes them good features for detecting the changes caused by shot transitions.

The luminance color histogram for frame f_i is calculated using the equation

$$H_i^Y(k) = \#\text{pixels in } f_i \text{ with } k \leq \left\lfloor \frac{L}{MaxVal} f_i^Y(x,y) \right\rfloor < k+1, \quad k = 0, \ldots, N-1 \tag{2.3}$$

where $\lfloor x \rfloor$ is the value of x rounded to the next lower integer, L is the number of bins used in the histogram, and $MaxVal$ is the maximum value a pixel can assume, which in our case equals 255. We have used 64 bins for the luminance histogram, i.e., $L = 64$.

The histogram intersection technique, first proposed by Swain and Ballard [47], is one of the many techniques to compare two given histograms. Given a template histogram, h_T, and a sample histogram, h_S, the normalized histogram intersection between them is defined as

$$\mathcal{H}(h_T, h_S) = \frac{\sum_{j=1}^{L} \min(h_T(j), h_S(j))}{\sum_{j=1}^{L} h_T(j)} \tag{2.4}$$

The numerator of the above expression represents the number of pixels from the template that have corresponding pixels of the same bin in the sample image. Hence, $0 \leq \mathcal{H}(h_T, h_S) \leq 1$ is a fractional match value between the two histograms. It is also shown in [47] that if the images represented by h_T and h_S have the same size, that is, if

$$N = \sum_{j=1}^{L} h_T(j) = \sum_{j=1}^{L} h_S(j),$$

then the histogram intersection operation defined in (2.4) reduces to the L_1 norm of the difference between the histograms

$$1 - \mathcal{H}(h_T, h_S) = \frac{1}{2N} \sum_{j=1}^{L} |h_T(j) - h_S(j)|. \tag{2.5}$$

The component of the GT feature vector based on the luminance histogram dissimilarity between frames f_{i-1} and f_i is defined as

$$x_{i1} = \frac{1}{2N} \sum_{j=1}^{64} |H_i^Y(j) - H_{i-1}^Y(j)| \tag{2.6}$$

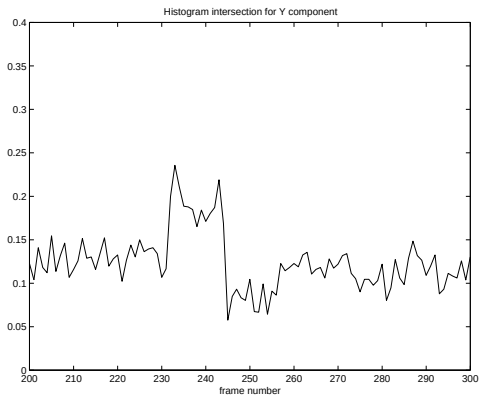


Fig. 2.3. The behavior of the luminance histogram intersection frame dissimilarity measure during a dissolve transition. The transition is between frames 233 and 244

where N is the number of pixels in the image. Since we are using NTSC SIF size images, this value is $N = 352 \times 240$.

The behavior of the luminance histogram intersection frame dissimilarity measure during a dissolve transition is shown in Figure 2.3

2.5.2 Luminance Pixel Variance and Mean Features

Frames belonging to different shots may have similar color distributions, e.g., two shots containing two different people in front of the same background. In these cases the color distribution feature is of little use and a feature that reflects the structure within a frame is required. The variance of pixel values correlates with how "edgy" a frame is, that is, if a frame has a large number of edges, the value of this feature will be high. The variance of luminance values is defined as

$$\sigma_i^2 = \frac{1}{|X||Y| - 1} \sum_x \sum_y \left(f_i^Y(x, y) - \mu_i^Y \right)^2 \tag{2.7}$$

where $|X|$ and $|Y|$ are the width and height of a frame, respectively, and the value $\mu_i^Y = \frac{1}{|X||Y|} \sum_x \sum_y f_i^Y(x, y)$ is the mean of the pixel luminance values for frame f_i.

Using the features above, we define two more components of the GT feature as

$$x_{i2} = \mu_i^Y - \mu_{i-1}^Y \tag{2.8}$$

$$x_{i3} = \sigma_i^2 - \sigma_{i-1}^2 \tag{2.9}$$

The difference of pixel luminance variance feature, x_{i3}, is a powerful tool in localizing dissolve regions. Specifically, x_{i3} varies approximately linearly from a large negative value to a large positive value during a dissolve transition. This observation may be shown mathematically using the shot transition model given in (2.1). Assuming that the pixel luminance variances, σ_i^2, within a shot have approximately the same value, we obtain the following formula for the frame variances around a dissolve

$$\sigma_i^2 = \begin{cases} \sigma_{S_j}^2, & i < s \\ \left(1 - \frac{i-s}{T}\right)^2 \sigma_{S_j}^2 + \left(\frac{i-s}{T}\right)^2 \sigma_{S_{j+1}}^2, & s \le i \le e \\ \sigma_{S_{j+1}}^2, & i > e \end{cases} \tag{2.10}$$

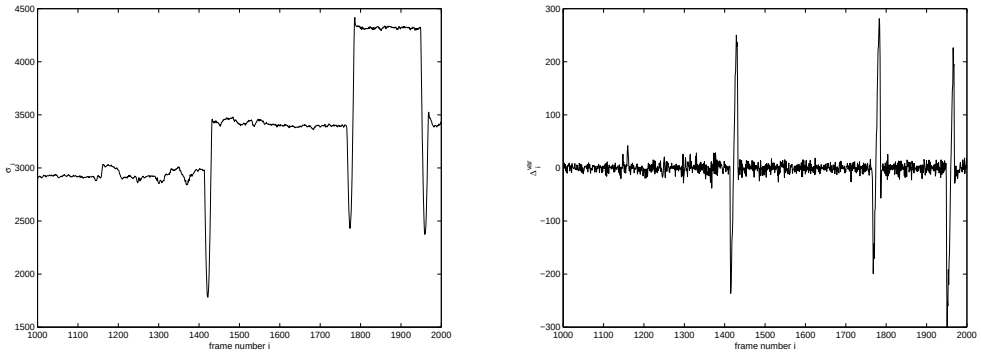
where $\sigma_{S_j}^2$ and $\sigma_{S_{j+1}}^2$ are the variances for shots S_j and S_{j+1}, respectively. The first order difference of the frame variance, x_{i3}, is then given by

$$\Delta_i^{var} = \begin{cases} 0, & i < s \\ \frac{2}{T^2}\left[-((e-i)-1)\,\sigma_{S_j}^2 + ((i-s)+1))\,\sigma_{S_{j+1}}^2\right], & s \le i \le e \\ 0, & i > e \end{cases} \tag{2.11}$$

From (2.11) we see that the value of Δ_i^{var} increases linearly from $-\frac{2T-1}{T^2}\sigma_{S_j}^2$ to $\frac{2T-1}{T^2}\sigma_{S_{j+1}}^2$ during a dissolve.

The fact that frame variances behave parabolically during a dissolve was first discussed in [48] where large negative spikes appearing near the start and near the end of a dissolve for the second difference of luminance variance was used to detect dissolves. However, since each differencing operation decreases the signal to ratio of the variance curve and hence the detectibility of the dissolve, the practical performance of this method based the second difference is not adequate.

The behavior of the luminance variance, σ_i^2, and the difference of luminance variance values, x_{i3}, during three dissolve transitions are illustrated in Figure 2.4.



(a) Luminance variances (b) Luminance variance differences

Fig. 2.4. Variance of pixel luminance values and the variance difference frame dissimilarity values for the first 1000 frames of a sequence containing three dissolve regions.

2.6 Using Decision Trees for Shot Boundary Detection

After the GT vectors are extracted from all the frames in a given sequence, we use decision trees to detect cut and dissolve locations. In this section we briefly review the decision tree approach to pattern classification.

2.6.1 Decision Tree Methodology

An intuitively appealing methodology in decision making is to proceed in multiple stages, making partial decisions along the way. The basic idea behind this methodology is to break up a complex decision boundary into a union of several simpler boundaries. Decision trees are one of the possible approaches to multistage decision making which are hierarchical in nature. The decision tree methodology is known to be very effective in a wide variety of domains. Overviews of applications of decision tree classifiers in pattern recognition can be found in [49, 50]. A general multidisciplinary survey is given in [51].

A *decision tree* consists of a tree structure with three types of nodes: the root node, internal nodes, and leaf nodes. The root node and each internal node have a rule associated with them and represent intermediary decisions in the tree. The outcome of each decision is called a *split*, because it corresponds to a splitting of the subset of the data that is at the node at which the split occurs. At the bottom of the tree are the leaf nodes that represent the final decisions of the tree, each containing a class label. The classification of a particular feature vector starts at the root node. Based on the outcome of the rule in the root node one of the splits is chosen and a descendant node is reached. This process is repeated until it terminates by arriving at a leaf node at which point the feature vector is labeled using the class label of the leaf node.

Several advantages of decision tree classifiers have been pointed out in the literature [49, 51]:

- Decision trees are ideally suited to handle nominal features, which have values that are discrete without any natural measure of similarity or ordering. The frame type flag features in the GT feature vector are examples of nominal features.

- Trees perform classification by a sequence of simple, relatively easy to understand tests whose semantics are generally clear to domain experts.

- In contrast to conventional single-stage classifiers where each data sample is tested against all classes, which reduces efficiency, in a tree classifier a sample is tested against only certain subsets of classes, thus eliminating unnecessary computations.

- In single-stage classifiers, only one set of features is used for discriminating among all classes, which is selected using a global optimality criterion. In the decision tree methodology, however, one has the flexibility of choosing different subsets of features at different internal nodes of the tree such that the feature

subset chosen is optimal for the samples in that node. This flexibility may provide performance improvement over a single-stage methodology.

- The "curse of dimensionality" for small sample size may be avoided by the decision tree structure using a small number of features at each node.

However, the decision tree approach also has some drawbacks:

- The mean subset size decreases with the depth of the decision node. Hence, there is strong tendency for inferences made near the leaves to be less reliable than those near the root. This is sometimes called the data fragmentation problem.

- Overlap between terminal nodes, especially when the number of classes is large, can cause the terminal number of nodes to be much larger than the actual number of classes and thus increase the search time and memory requirements.

2.6.2 Construction of Decision Trees

Suppose we are given a collection of pairs of jointly distributed random variables $\{(\mathbf{X}_k, Y_k)\}$, $k = 1, \ldots, N_{tr}$, where $\mathbf{X}_k \in R^n$ is a feature for sample k and $Y_k \in \{1, \ldots, L\}$ is the class label for this sample. The classification problem is to generate or "grow" a decision tree from this training set so that it can be used to estimate the class label of a previously unobserved feature vector. The regression problem, may be stated similarly; however, in this case Y_k is real-valued and is interpreted as the dependent variable which is related to the independent variables in $\mathbf{X}_k$.

An obvious strategy in growing a decision tree is to use the following recursive process: at a given node, either declare it to be a leaf and assign it a class label, or find a decision rule to split the data further into subsets. This is an example of a general tree generation methodology known as classification and regression trees (CART) [52]. In the CART approach the main design questions relating to tree construction are

1. How should the decision rule be determined at each node?

2. When should a node be declared a leaf, i.e., when should we stop splitting?

3. If a leaf node contains data from more than one class, how should a class label be assigned to it?

4. How can we avoid trees that have good performance on the training set but cannot classify new samples accurately, i.e., those that overfit the data in the training set?

The easiest problem to solve is the third one: If the data in a leaf node corresponds to more than one class than the class label for that leaf is determined by the class that has the largest number of samples in the leaf.

In order to solve the first problem a measure of "impurity", $i(\nu)$, is defined for each internal node ν in the tree. Several different impurity functions have been used [53] such as the *entropy impurity* function

$$i(\nu) = -\sum_j P(\omega_j) \log_2 P(\omega_j) \tag{2.12}$$

and the *Gini impurity* function

$$i(\nu) = \sum_i \sum_{j \neq i} P(\omega_i) P(\omega_j). \tag{2.13}$$

In these equations, $P(\omega_j)$ is the fraction of samples at node ν that belong to class j. After an impurity function is defined, at ν we choose the decision rule that maximizes the decrease in impurity as defined by

$$\Delta i(\nu) = i(\nu) - P_L i(\nu_L) - (1 - P_L) i(\nu_R) \tag{2.14}$$

where ν_L and ν_R are the left and right children of ν and P_L is the fraction of samples at node ν that will go to ν_L if this decision is used. It has been observed that the particular impurity function used does not affect the performance of the final decision tree in general [53].

The second and fourth problems are related. There are two approaches to solving these problems. In *pre-pruning algorithms* a stopping rule is used to halt tree growing. The stopping rule may be such that a node is not split if the decrease in impurity over all possible decisions is smaller than a threshold or it could be a measure depending on the global properties of the tree. This type of approach may suffer from the horizon effect, which is stopping splitting too early for overall optimal recognition accuracy. On the other hand, in *post-pruning algorithms* a full tree is first grown that minimizes impurity in all leaf nodes. Then, the resulting tree is examined in order to discard rules and conditions that only apply for the characteristics of the training set and do not reflect true regularities of all possible data. In this case tree generation requires two passes through the data and either a separate data set or cross validation methods are required. Post-pruning algorithms avoid the horizon effect but at a computational cost.

In this work, we shall only consider post-pruning methods, since these were found to be superior to pre-pruning methods in general [52]. The basic strategy is as follows

- *Growing the tree*: Starting with an empty tree, perform the following operations

    1. If all samples or a sufficient majority of them at the current node, ν, belong to the class C stop and create a terminal node with class label C.
    2. Otherwise score each possible split of the set of samples at ν using a goodness measure.
    3. Choose the best split and split the samples in ν into left and right children using this split.
    4. Recursively repeat this procedure for the subtrees rooted at the left and right children nodes.

- *Pruning the tree*: Starting with the large tree generated above, perform the following operations

1. Prune the children of node ν if the sum of the costs of its children is equal the cost of ν according to some cost criterion.

2. Continue on doing the previous step until only the root node is left, creating a sequence of pruned subtree.

3. Choose one of these subtrees in the sequence created above using some desirability criterion.

2.7 Detecting and Classifying Shot Transitions

We use the GT of a video sequence and a binary regression tree [52] to estimate the conditional probability that a given frame from the sequence belongs to a shot transition. An improvement over the CART method of building regression trees was proposed in [54] where the data set is divided into two roughly equal sets. One set is used to build a large tree that overfits the data. Then the other set is used to prune the tree back to maximize some desirability criteria. This procedure is then iterated, successively interchanging the roles of the first and second ground truth sequences until convergence.

The regression tree we used is a variation of the above technique. The difference is that the training and pruning step is used only once since we have found that the tree obtained after one iteration is adequate in classification accuracy. The training process uses two sequences of nearly equal size with known shot boundary locations, which we refer to as *ground truth sequences*. One ground truth sequence is used to build a large tree which overfits the data. The tree obtained is then pruned in a bottom-up fashion using the second ground truth sequence where we remove nodes whose deletion decreases the classification error.

The GT/regression tree method offers a number of advantages compared to other shot boundary detection techniques listed in Section 2.3. First, the GT feature vector allows a multitude of different features to be collectively used to detect shot transitions. This is important since different features may be useful in detecting

different types of shot transitions. The regression tree performs automatic feature selection and complexity reduction [54]. Second, the output of the regression tree is normalized in the range $[0, 1]$ and approximates the probability that the frame under question belongs to a shot transition, which allows consistent and meaningful thresholding. Moreover, the method is highly extensible. New features can easily be incorporated into the existing system. In conclusion, The GT/regression tree combination provides a powerful technique for detecting and classifying a wide variety of shot transitions.

We have focused on detecting two types of shot boundaries, cuts and dissolves, since these constitute the overwhelming majority of shot transitions in most programs. Cuts are point events in video sequences, i.e., they take place from one frame to another, whereas dissolves are distributed over a number of frames. Due to their dissimilar nature, we use slightly different methods to detect cuts and dissolves.

2.7.1 Detection of Cuts

A schematic diagram of the steps in cut detection is shown in Figure 2.5. To train the regression tree, we use known cut locations in the ground truth sequences and ignore gradual transitions. After the tree has been trained using two ground truth sequences, it is used to process the GT from the sequence whose cuts are to be detected. To detect if a cut has occurred between f_i and f_{i+1} we place a window of length $2W_1+1$ frames centered around f_i and the GT vectors for all the frames in this window are concatenated into one large feature vector. These agglomerate vectors are then used by the regression tree which provides a piecewise linear approximation to the conditional mean, i.e.,

$$y_i \approx E[\alpha_i \,|\, \mathbf{X}_{i-W_1} \cdots \mathbf{X}_i \cdots \mathbf{X}_{i+W_1}] \tag{2.15}$$

where α_i is the cut indicator function

$$\alpha_i = \begin{cases} 1, & \text{if a cut occurs between } f_i \text{ and } f_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{2.16}$$

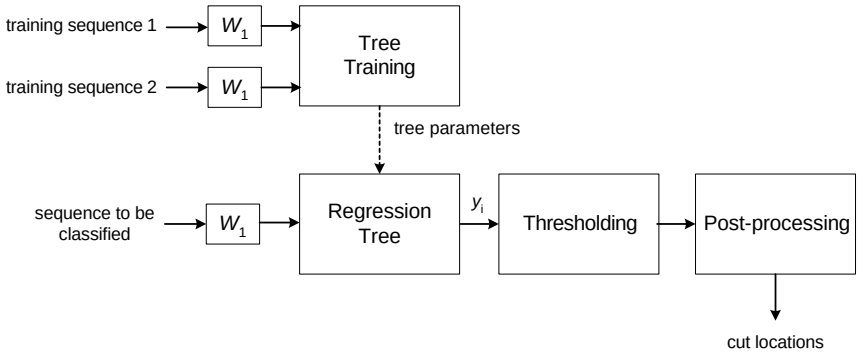


Fig. 2.5. Block diagram of cut detection system using a regression tree.

and y_i is the output of the regression tree for f_i. The output y_i can be interpreted to be the probability of a cut occurring between f_i and f_{i+1} [52].

Candidate cut locations are then determined by thresholding the output of the regression tree; if $y_i \geq \tau$, we decide that there is a cut between f_i and f_{i+1}. This approach is more robust than thresholding the value of a frame similarity feature, because the classifier chooses the best features from the GT and the threshold is not dependent on the specific video sequence.

The detected candidate cut locations are then post-processed to remove cuts that are too close together. In the experiments described in Section 2.8, if two candidate cut locations are closer than 10 frames, the candidate cut with a smaller value of y_i is deleted.

2.7.2 Detection of Dissolves

For the detection of dissolves we use a similar approach to the detection of cuts, as described in Section 2.7.1. However, since dissolves take place over an interval of frames, the size of the window used in feature windowing, W_2, is chosen to be much larger than the one used for cut detection. To train the regression tree, we use known dissolve locations in the ground truth sequences and ignore cuts.

The output, w_i, of the regression tree trained to detect dissolves then is given by the approximation

$$w_i \approx E[\beta_i \,|\, Y_{i-W_2} \cdots Y_i \cdots Y_{i+W_2}] \tag{2.17}$$

where β_i is the gradual transition indicator function

$$\beta_i = \begin{cases} 1 & \text{if } f_i \text{ belongs to a gradual transition} \\ 0 & \text{otherwise} \end{cases} \tag{2.18}$$

Candidate dissolve regions are then determined by thresholding the output of the regression tree; if $w_i \geq \tau_d$, we mark frame f_i as possibly belonging to a dissolve transition.

After the candidate dissolve locations are obtained, we perform two postprocessing operation to reduce the number of false alarms and misclassifications. First, due to classification noise, some frames within a dissolve may not be marked as a dissolve, causing the dissolve region to appear as two dissolves. We get rid of such cases by combining two dissolves that are closer than 7 frames. Second, although the regression tree used is trained just to detect dissolves, sometimes it also detects the cuts in a sequence, leading to misclassifications. In order to avoid this problem we remove dissolves that are closer than 20 frames to a detected cut. The motivation for this operation is the fact that cut detection is more reliable than dissolve detection, so if a dissolve is close to a detected cut, the chance of it being an error rather than the cut is much higher.

2.8 Shot Boundary Detection Experiments

In this section we report the results of the performance of our cut and dissolve detection algorithms on video sequences from different program genres and compare their performance with other algorithms.

Table 2.1
The number of different types of shot boundaries for the program genres used.

Genre	# frames	# cuts	# dissolves	# fades	# others
soap	67582	337	2	0	0
talk	107150	331	108	1	6
sports	78051	173	45	0	29
news	58219	297	7	0	6
movies	54160	262	15	6	1
cspan	90269	95	19	0	0
TOTAL	455431	1495	196	7	42

2.8.1 Cut Detection Experiments

For the cut detection experiments we have selected 29 sequences from our database to represent a number of different program genres. These sequences are classified into six program genres, namely soap operas, talk shows, sports, news, movies, and and C-SPAN programs. Statistical information about these sequences is summarized in Table 2.1.

To get an honest estimate of the performance of our cut detection system, we have used the following procedure which is similar to a cross-validation procedure

for each $G \in \{soap, talk, sports, news, movies, cspan\}$
  for $i = 1$ to 4
    randomly choose two sequences, S_1 and S_2, both not in G
    train a regression tree using S_1 and S_2
    classify all the sequences in G using this tree
    Average classification performance
  average the four values found above to find performance for genre G

Our experiments have shown that using a window size of $W_1 = 1$, which means that the regression tree uses 36 features, provides a reasonable balance between

complexity and performance for cut detection so we have used this value for our experiments. For all results of our method, we have used a threshold of $\tau = 0.8$ as our detection criteria.

We have compared our method with the sliding window technique proposed in [28] and often used in cut detection. In this technique, a symmetric window of size $2m+1$ is placed around f_i and a cut is declared between f_i and f_{i+1} if

1. the value of the feature value for i is the maximum within the window, and
2. it is also n times the value of the second maximum in the window.

We have used the sum of the histogram intersections of the Y, U, and V components, i.e., $g_{i,1} + g_{i,2} + g_{i,3}$ as the frame similarity feature. We chose the values $m = 7$ and $n = 2$ because these values gave the best over all performance on our data sets.

We have also compared our results with a global thresholding technique which uses the sum of the histogram intersections of the Y, U, and V components, i.e., it uses $g_{i,1} + g_{i,2} + g_{i,3}$ as the frame similarity feature. This simple method is included here as a baseline, to serve as an indication of the relative difficulty in detecting the cuts in various video genres. A global threshold value of 0.45 was found to give best results and this value was used in all our experiments. Again, we remove the cut with a lower feature value if two cuts are closer than 10 frames.

The results of these experiments are shown in Table 2.2. From this table we can make the following observations: The GT/regression tree method gives a consistent performance for video sequences with diverse content, whereas the sliding window method runs into problems with some program genres. The difference between the two algorithms is especially evident for the *sports* and *news* genres, where the the detection rate of the sliding window detector is low. There detection of shot transitions in for these genres is more challenging than the others, since there is a lot of camera and object motion in sequences belonging to these genres.

By training the regression tree used for classification using different sequences and averaging the classification result, we showed that the particular sequences used to train the regression tree have little effect on the performance of the system. The only

Table 2.2
Results for cut detection using the GT/tree classifier, the sliding window method, and simple thresholding. *D* and *M* indicate the average detection rate and missed detection, respectively, for each program genre, and *FA* is the total number of false alarms. *MC* gives the total number of misclassifies where a shot boundary other than a cut was detected at a cut location.

	tree classifier				*sliding window*				*simple thresholding*			
Genre	D	M	FA	MC	D	M	FA	MC	D	M	FA	MC
soap	0.941	0.059	13.3	0	0.916	0.084	99	0	0.852	0.145	24	0
talk	0.942	0.058	32.3	7.5	0.950	0.050	45	1	0.968	0.032	171	15
sports	0.939	0.051	82.5	34.8	0.785	0.215	59	1	0.925	0.075	251	73
news	0.958	0.042	38.0	0.75	0.886	0.114	61	0	0.926	0.074	212	1
movies	0.821	0.179	43.3	2	0.856	0.144	25	0	0.816	0.184	25	3
cspan	0.915	0.085	54.3	8.5	0.994	0.006	40	0	0.943	0.057	3	20
TOTAL	0.919	0.079	43.950	8.93	0.898	0.102	54.83	0.33	0.905	0.095	114.3	18.667

constraint to be satisfied when selecting the two sequences to train the regression tree is that they should contain a large number of cuts so that the tree can adequately be trained.

2.8.2 Dissolve Detection Experiments

For our dissolve detection experiments, we have selected 40 sequences that each contain cuts and a large number of dissolves. We have used a window size of $W_2 = 11$ frames and a threshold value of $\tau_d = 0.08$, since this value resulted in good classifier performance. In order to obtain an honest estimate of the performance of the regression tree classifier, we used the same procedure used for cut detection described in Section 2.8.1, namely we randomly selected two sequences to build a regression tree and classified the remaining sequences using this tree. This procedure was repeated four times and the performance for each tree was averaged. The results are presented in Table 2.3.

From Table 2.3 we can see that the performance of our dissolve detection is adequate for the *cspan* and *news* genres. For *sports*, although the detection rate is

Table 2.3
Results for the dissolve detection using the regression tree classifier. Detect and Miss indicate the average detection and miss probabilities, respectively. False Alarm is the average percentage of false alarms detected.

Genre	Detect	Miss	Avg. FA %
comedy	0.250	0.750	43.30
cspan	0.871	0.129	7.99
news	0.782	0.218	46.24
sports	0.757	0.243	252.73

high, there are many false alarms. Many of these are caused by other types of shot transitions and the high amount of motion in these sequences. For the *comedy* genre the classifier misses most of the dissolves. This is due to the fact the shots combined with dissolves in this genre usually have little variation in color, texture, and motion.

For comparison, we have listed three dissolve detection results reported in the literature in Table 2.4. In the literature, rather than using false alarm and detection rates, as we have done in Table 2.3, shot boundary detection results are often given in terms of *precision* and *recall*, which are defined as

$$\text{recall} = \frac{\#\,\text{true dissolves detected}}{\#\,\text{all dissolves in sequence}} = \frac{D}{D+M} \tag{2.19}$$

$$\text{precision} = \frac{\#\,\text{true dissolves detected}}{\#\,\text{all dissolves detected}} = \frac{D}{D+FA} \tag{2.20}$$

where D, M, and FA are the number of detect, miss, and false alarm locations, respectively. For the purposes of comparison we converted the results of our algorithm given in given in Table 2.3 to precision and recall values. Since the comedy genre is an outlier for the performance of our algorithm, we report the precision and recall results of our algorithm both with the results of the comedy included and with these results excluded.

From the comparison given in Table 2.4 we can make the observation that, when the comedy genre results are not included, the recall rate of our algorithm is compa-

Table 2.4
Comparison of some reported dissolve detection performances in the literature with the proposed regression tree based classifier. The precision and recall values are obtained by averaging all the dissolve detection results reported in each paper.

	# dissolves	precision	recall
Truong, Dorai, and Venkatesh [55]	297	75.1%	82.2%
Lienhart [56]	133	25.8%	55.4 %
Jun, Yoon, and Lee [57]	103	71.0%	90.3 %
ours, comedy not included	559	44.0%	80.3%
ours, comedy included	633	43.2%	66.5%

rable hiher than the result given in [56] and is comparable to the one given in [55]. However, rge precision of our dissolve detection system is lower than the results reported in the literature. This problem may be fixed by post-processing the results of our algorithm. Note that the number of dissolves tested by our system is much larger than those for the ones given by other researchers, which makes the variance of our results lower than other results.

2.9 Conclusions

In this chapter we introduced a new method to detect cut and dissolve types of shot boundaries through the use of the Generalized Trace feature vector and regression trees. In the literature shot transition detection is usually achieved using rule-based systems that have many ad-hoc parameter settings and have to be fine-tuned for each program genre. Our classifier-based approach overcomes these problems and can also be easily extended by adding extra features. Thresholding and post-processing the output of the regression tree is more robust than thresholding the value of a frame similarity feature, since the tree chooses the best features from the generalized trace feature vector and the threshold is not dependent on the specific video sequence.

We tested the performance of out cut and dissolve detection system on a large number of cuts and dissolves. We have used a methodology similar to the leave-one-out method, where we trained the regression tree on two randomly selected sequences and determined its output on the remaining sequences. This procedure was repeated four times and the classifier performance was taken to be the mean of these values. Our experiments showed that our cut detection technique has better overall performance and has a performance that is approximately constant across program genres. We also showed that the regression tree classifier is effective in detecting dissolve shot transitions.

3. DISTRIBUTION OF SHOT LENGTHS FOR VIDEO ANALYSIS

3.1 Introduction

In this chapter we investigate the problem of choosing and fitting models for the distribution of shot lengths for broadcast video programs. Modeling the distribution of shot lengths for video is important for the following applications, among others:

- *Shot boundary detection.* We discussed a number of approaches to detect shot transitions in video programs in Chapter 2. The performance of these methods can be enhanced if one uses a model for the probability distribution of shot lengths as *a priori* knowledge in a Bayesian framework to derive an adaptive threshold for shot boundary detection [41]. This method is superior to using a fixed threshold to detect shot boundaries and is more robust to changes caused by camera and object motion.

- *Video genre classification.* It is evident that, for an information source as rich as video, efficient indexing and retrieval require some form of automatic analysis of semantic content which may be achieved by assigning semantic labels to individual shots. Currently, however, automatic extraction of truly semantic features such as "young girl running," or "park scene" is not possible. One way to circumvent this problem is to define features that correlate well with high level semantic labels and hence are useful in bridging the gap between low-level image features and semantic labels [58]. We call such features *pseudo-semantic features* [59]. Shot length is one such feature. For example, action sequences contain a large number of consecutive short shots whereas for other content, like news stories and talk shows, longer shot lengths are more common. It has

been shown that shot length distribution is a simple yet effective feature which correlates well with the content and genre of a video sequence [59].

- *Video Traffic Modeling.* Accurate traffic models of variable bit rate (VBR) coded video is necessary for prediction of performance of any multimedia network. Various models have been proposed for modeling variations in the bit rate of VBR video [60,61]. In the last decade researchers have found that video conference data and packet counts per unit time in Ethernet traffic exhibit long range dependence and self-similarity, while quantities including Internet file transmission times, UNIX file sizes, and CPU time to complete a job appear to be generated by distributions with heavy tails [62,63]. This fact is disturbing since it makes invalid the classical assumptions widely used in queuing analysis such as exponential waiting times and iid-ness of events. VBR coded video bit rate also exhibits multiple-scale variations which is most probably caused by the fact that the length of shots are generated from heavy-tailed distributions. Based on these observations, models that incorporate long range dependence were recently proposed as more faithful models for VBR video traffic [64]. The accurate determination of the distribution function of shot lengths is central in these and most other traffic models.

The structure of this chapter is as follows. We discuss previous work in shot length distribution in Section 3.2. In Section 3.3 we illustrate the statistical properties of the data set that we used for our experiments. After reviewing the mathematical properties of heavy-tailed distributions in Section 3.4 we describe how the parameters for the different distributions are estimated using different methods in Section 3.5. We introduce the Kolmogorov-Smirnov test as the goodness-of-fit measure in Section 3.6 and use it to find the best distributions that fits to the given data. Finally, Section 3.7 presents our conclusions.

3.2 Previous Work

Sarkar *et al.* [61] use a geometrical distribution to model the length of shots which is expressed in number of GOPs in the shot. Frater *et al.* [60] propose the following probability mass function for the length of shots

$$f(m) = \frac{a}{m^k + b^2}$$

where $n \approx 2$ is used to model two movie sequences. Krunz and Ramasamy [64] propose a formula in which the probability mass function of shot lengths is expressed in terms of the empirical autocorrelation function. They also derive expressions which relate the performance at a video buffer to the distribution of shot lengths of video. Pareto and Weibull distributions were considered for shot lengths. Heyman and Lakshman [65] examine the gamma, Weibull, and Pareto densities for modeling shot lengths. They use a plot of $\gamma_1 = \sigma/\mu$ versus $\gamma_3 = \mu_3/\sigma^3$ where μ, σ, and μ_3 are the mean, variance, and the third central moment estimated from data, respectively, to identify the underlying distribution function for a variety of broadcast video programs. Unfortunately, such high moments may not exist if the underlying distribution has heavy tails. Their analysis revealed that different video programs followed different distributions. For some sequences no good fit was found. Dawood and Ghanbari [66] use a second order gamma distribution to model shot lengths. Jelenkovic *et al.* [67] develop a rigorous video model for queuing analysis. Based on their experiments they conclude that MPEG traffic exhibits subexponential behavior. They use a Pareto distribution to model shot lengths.

Previous studies are limited by the following facts: first, the number of video sequences for which the proposed shot length models were fit was generally too small to decide which distribution to use and to obtain accurate estimates for distribution parameters. Second, the video content used was not diverse enough, which makes it impossible to compare distributions from different program genres. Generally researchers have just used sequences containing movies or video conferences. Third, statistical analysis for the goodness of fit is lacking in some studies [41, 60].

In previous work on modeling shot lengths often one of the following distributions were used

Pareto distribution

$$\begin{aligned} f(x) &= \frac{\alpha\beta^{\alpha}}{x^{\alpha+1}} \\ F(x) &= 1-\left(\frac{\beta}{x}\right)^{\alpha}, \quad \alpha,\beta>0, x>\beta \end{aligned} \tag{3.1}$$

Gamma distribution

$$\begin{aligned} f(x) &= \frac{\beta^{\alpha}}{\Gamma(\alpha)}x^{\alpha-1}e^{-\beta x} \\ F(x) &= \Gamma(\alpha,\beta x), \quad \alpha,\beta>0, x>0 \end{aligned} \tag{3.2}$$

Weibull distribution

$$\begin{aligned} f(x) &= \frac{\alpha}{\beta}\left(\frac{x}{\beta}\right)^{\alpha-1}\exp(-(x/\beta)^{\alpha}) \\ F(x) &= 1-\exp(-(x/\beta)^{\alpha}), \quad \alpha,\beta>0, x>0 \end{aligned} \tag{3.3}$$

In this chapter we will consider only these three distributions as possible distribution models to model shot lengths for broadcast video.

3.3 Statistical Properties the Experimental Data Set

We have used sequences from two program genres for the results reported in this chapter. The locations of all the shot transitions in the sequences were recorded by a human operator. The characteristics of these two genres are summarized below.

- *cspan* (25 sequences, approximately 7.5 hours, 996 shots). C-SPAN programs generally contain minimum editing. For example, our dataset includes a shot of approximately 16 minutes which contains only a head and shoulders shot of a speaker.

- *movie* (7 sequences, approximately 1.5 hours, 917 shots). The shot lengths distribution may differ from movie to movie and from director to director. In

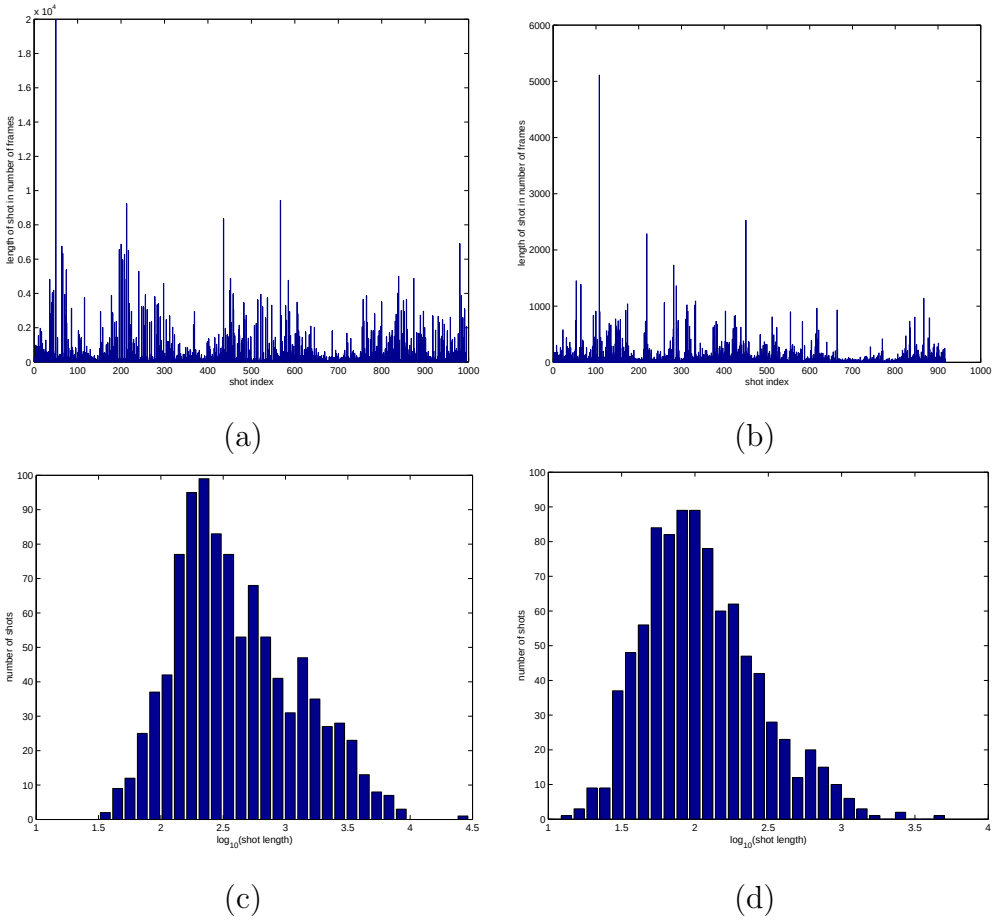


Fig. 3.1. Plots (a) and (b) show the lengths of shots, for cspan and movie program genres, respectively. Note different scales on the y-axis for these plots. Plots (c) and (d) are the log histograms of shot lengths for cspan and movie, respectively.

fact it has been shown that the shot length distribution correlates well with the director [68]. However, different editing patterns for movies generally fall within a narrow range widely used norms.

Basic statistical properties of the two data sets are given in Figure 3.1 and Table 3.1.

Table 3.1
Basic statistics for the two data sets used in our experiments.

Data	*cspan*	*movie*
n	996	917
min	32	12
median	332	98
mean	818.67	178.55
max	29703	5113
variance	2.16×10^6	7.78×10^4

3.4 Heavy-Tailed Distribution Functions

From the log histogram plots in Figure 3.1 we observe that the shot length distribution data is heavily skewed to the right. This implies that there is a non-negligible number of very long shots. This observation is verified by the data plots in Figure 3.1(a) and (b). Throughout this paper we will assume that the shot lengths are independent and identically distributed (iid).

Consider the Gaussian distribution, for which, as $x \to \infty$

$$P[X > x] \sim \frac{1}{\sqrt{2\pi}} \frac{\exp(-x^2/2)}{x} \to 0. \tag{3.4}$$

Such a distribution is said to possess light tails. In contrast, a random variable X is called *heavy-tailed* if, as $x \to \infty$, we have

$$P[X > x] \sim x^{-\alpha} L(x) \tag{3.5}$$

where $L(x)$ is a slowly varying function for which $\lim_{t\to\infty} L(tx)/L(x) = 1$. The variable α is called the *tail index* of the distribution. Heavy-tailed distributions belong to the family of *subexponential distributions*. An important property [69] of these distributions is that

$$\lim_{x\to\infty} \frac{P(X_1 + \cdots + X_n > x)}{P(\max(X_1, \ldots, X_n) > x)} = 1$$

for every $n \geq 2$. This property implies that the largest value in $\{X_n\}$ exerts a strong influence on the total distribution. If X is a heavy-tailed random variable, for $X \geq 0$, the l^{th} moment of X, $E(X^l)$, exists only if $l < \alpha$. This property implies that if $\alpha < 2$, a heavy-tailed random variable has infinite variance; if $\alpha < 1$, then it also has infinite mean.

Heavy-tailed distributions include the Pareto, log-normal, and log-gamma distributions and the Weibull distribution for $0 < \alpha < 1$. Light-tailed distributions include the normal, exponential, and gamma distributions and the Weibull distribution for $\alpha \geq 1$.

3.5 Estimating Distribution Parameters

3.5.1 The Hill Plot

One way to estimate the tail index, α, is based on the following observation: Suppose $X_1, \ldots, X_n$ are iid random variables with distribution function F. Let $X_{1,n} \geq \cdots \geq X_{n,n}$ be the order statistics for this data. If X has a Pareto distribution with $\beta = 1$

$$P(X > x) = 1 - F(x) = x^{-\alpha}, \quad x \geq 1 \tag{3.6}$$

then $Y = \log X$ will have an exponential distribution with distribution function

$$P(Y > y) = y^{-\alpha y}, \quad y \geq 0$$

Since for this distribution the mean is α^{-1}, the maximum likelihood estimate (MLE) of α is given by

$$\hat{\alpha}_n^{-1} = H_n = \frac{1}{n} \sum_{i=1}^{n} \log X_{i,n}$$

We can generalize this estimation method for the case where X has a general heavy-tailed distribution

$$1 - F(x) = x^{-\alpha} L(x)$$

by assuming that above a certain threshold $1 - F(x)$ behaves like the Pareto distribution function in (3.6). This means we should base our estimate of α on the part

of the distribution that looks most Pareto-like. Based on these arguments we can define the Hill estimator to be

$$\hat{\alpha}_{k,n}^{-1} = H_{k,n} = \frac{1}{k}\sum_{i=1}^{k} \log \frac{X_{i,n}}{X_{k+1,n}} \tag{3.7}$$

Note that only k upper statistics are used in the estimation. The Hill estimator is asymptotically normal and it can be shown that as $n \to \infty$ and $k \to \infty$ but $k/n \to 0$, we have $H_{k,n} \to \alpha^{-1}$ in probability. In practice the Hill estimator is used by plotting

$$\text{Hill Plot: } \left\{(k, H_{k,n}^{-1}),\ 1 \leq k < n\right\} \tag{3.8}$$

One should bear in mind that the Hill estimator has optimality properties only when the underlying distribution is close to a Pareto distribution. If the distribution is far from Pareto, there may be large bias even for large sample sizes. Hence it is best to use the Hill plot with other analysis techniques and compare results. The Hill plots for the datasets are shown in Figure 3.2(a) and (b). These plots suggest the tail index values $\alpha_{\text{cspan}} \approx 0.8.$ and $\alpha_{\text{movie}} \approx 1.2$

3.5.2 The Moment Estimator

Another estimate for α may be obtained using the moment estimator proposed in [70]. This method is designed to estimate the *extreme value index*, γ, where we have $\gamma = 1/\alpha$. For this method we first define the quantities

$$H_{k,n}^{(r)} = \frac{1}{k}\sum_{i=1}^{k} \left(\log \frac{X_{i,n}}{X_{k+1,n}}\right)^r$$

which for $r = 1$ is the Hill estimator described in Section 3.5.1. Then the estimate for γ is given by

$$\hat{\gamma}_{k,n} = H_{k,n}^{(1)} + 1 - \frac{1/2}{1 - (H_{k,n}^{(1)})^2/H_{k,n}^{(2)}} \tag{3.9}$$

The quantity $\hat{\gamma}_{k,n}$ is also very useful in deciding if a given data sample is drawn from a heavy-tailed distribution or not. If $\hat{\gamma}_{k,n}$ is negative or close to zero, then most probably the underlying distribution is not heavy-tailed. The use of the moment

estimator to estimate the value of γ and decide if the underlying distribution is heavy-tailed is similar to the Hill plot. We plot the graph

$$\text{Moment Plot: } \{(k, \hat{\gamma}_{k,n}),\ 1 \leq k < n\} \tag{3.10}$$

to obtain an estimate of γ. The moment estimator plots are shown in Figure 3.2(c) and (d). These plots suggest the tail index values $\gamma_{\text{cspan}} \approx 0.7$ and $\gamma_{\text{movie}} \approx 0.6$ which correspond to the values $\alpha_{\text{cspan}} \approx 1.4$ and $\alpha_{\text{movie}} \approx 1.7$. From these plots we can also conclude that heavy-tailed distribution is an appropriate model for these shot distributions.

3.5.3 The Mean Excess Function

Let X be a random variable with a right endpoint x_F. The function defined by

$$e(u) = E(X - u | X > u),\ 0 \leq u < x_F) \tag{3.11}$$

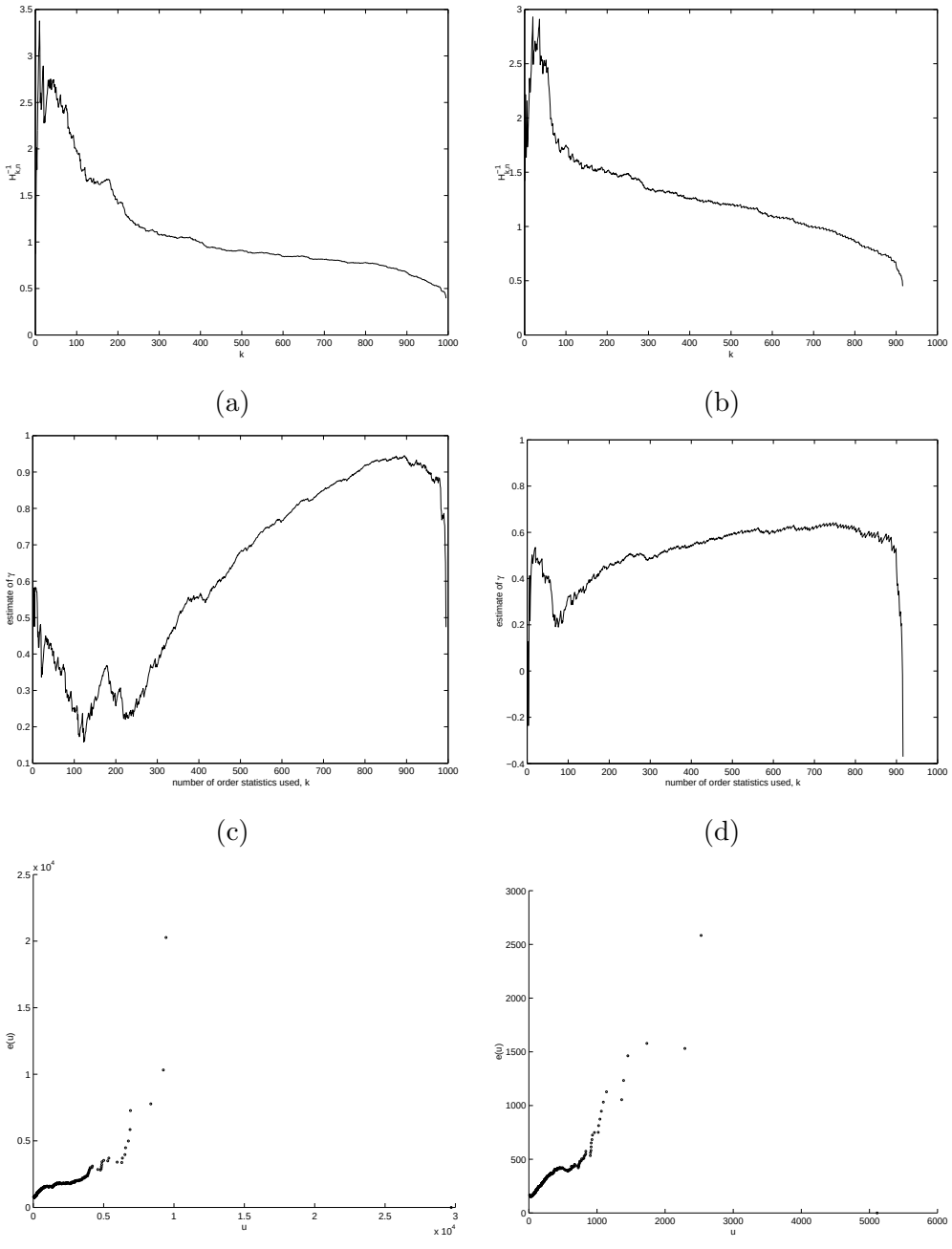
is called the *mean excess function* of X. The mean excess function provides another useful graphical tool, especially for discrimination in the distribution tails. The mean excess function of a heavy-tailed distribution function, for large values of u, typically will lie between a constant function, corresponding to the exponential distribution, and a straight line with positive slope, corresponding to the Pareto distribution. Any continuous distribution function F is uniquely determined by its mean excess function [69].

Suppose that $X_1, \ldots, X_n$ are iid and define $\Delta_n(u) = \{i : X_i > u\}$, that is, $\Delta_n(u)$ is the set of samples in $\{X_n\}$ with values larger than the threshold u. Then we define the empirical mean excess function, $e_n(u)$, to be

$$e_n(u) = \frac{1}{\text{card}\Delta_n(u)} \sum_{i \in \Delta_n(u)} (X_i - u), \quad u \geq 0 \tag{3.12}$$

where we use the convention $0/0 = 0$. The mean-excess plot is then obtained from the graph

$$\text{ME Plot: } \{(X_{k,n}, e_n(X_{k,n})),\ 1 \leq k < n\} \tag{3.13}$$



(a) (b)

(c) (d)

(e) (f)

Fig. 3.2. Plots (a), (c), and (e) are the Hill, moment estimator, and mean excess function plots for the cspan program genre. Plots (b), (d), and (f) are the same plots for the movie genre.

Table 3.2
Formulas for the mean and variance for the three distributions considered as possible models. Note that for the Pareto distribution the mean exists only if $\alpha > 1$ and the variance exists only if $\alpha > 2$.

Model	mean	variance
Pareto	$\alpha\beta/(\alpha-1)$	$\alpha\beta^2/(\alpha-1)(\alpha-2)$
Gamma	α/β	α/β^2
Weibull	$\beta\Gamma(1+1/\alpha)$	$\beta^2[\Gamma(1+2/\alpha)-\Gamma^2(1+2/\alpha)]$

where $X_{1,n} \geq \cdots \geq X_{n,n}$ are the order statistics. The mean excess plots for the two datasets are shown in Figure 3.2(e) and (f). Since the graphs are linear we have further proof that the underlying distributions are heavy-tailed. Since linear mean excess functions correspond to the Pareto distribution, we also conclude that the Pareto distribution is a good candidate to model the given data.

3.6 Selecting the distribution Using a Goodness-of-Fit Measure

In this section we examine the problem of choosing the best distribution for our shot length data. Formulas for the means and variances for the three distributions are given in Table 3.2. We shall use some of these formulas to estimate distribution parameters using the method of moments.

For the gamma distribution equating the first moments of the data, we obtain

$$\hat{\alpha}_g = \frac{\hat{\mu}^2}{\hat{\sigma}^2}, \quad \hat{\beta}_g = \frac{\hat{\alpha}_g}{\hat{\mu}}$$

where $\hat{\mu}$ and $\hat{\sigma}^2$ are the estimates for the mean and variance, respectively.

For the Pareto distribution the maximum likelihood estimate of β is $\min\{X_n\}$. We then either obtain an estimate of α from the Hill and moment estimator plots or check to see if $\alpha > 1$ from these plots and use the mean formula to obtain

$$\hat{\alpha}_p = \frac{\hat{\mu}}{\hat{\mu} - \hat{\beta}_p}, \quad \hat{\beta}_p = \min\{X_n\}, \quad \text{if } \alpha > 1$$

For the Weibull distribution there is no closed form expression for the method of moments estimators for the parameters. Therefore, for this distribution parameters have to be estimated by plotting the equation

$$\ln\left(\ln\left(\frac{1}{1-F(x)}\right)\right) = \alpha \ln x - \alpha \ln \beta$$

which is obtained by manipulating the expression given in (refeq:weibull) and fitting a line through the data points to estimate $\hat{\alpha}_w$ and $\hat{\beta}_w$.

Once the parameters of the distributions are estimated, we measure the goodness of fit of the proposed distribution to the empirical distribution function using the Kolmogorov-Smirnov (K-S) statistic. The K-S statistic is defined to be [71]

$$D = \max_{-\infty<x<\infty} |F_n(x) - F(x)| \tag{3.14}$$

where $F_n(x)$ is the empirical cumulative distribution function estimated from data using the formula

$$F_n(x) = \frac{n-k+1}{n+1}.$$

The values of the estimated parameters and the corresponding values of the K-S statistic are tabulated in Table 3.3. From this table we see that for our data set the Weibull distribution gives the best fit followed closely by the Pareto distribution. However, among the three models the Pareto distribution provides the best fit for the right tail of the shot length distribution. Therefore, it may be preferred when it is important to characterize the distribution of very long shots.

3.7 Conclusions

In this chapter we have addressed the problem of choosing a good probabilistic model for the duration of shots for broadcast video. Accurate models for shot lengths are important to model video both for content-based retrieval applications and for performing queuing analysis for the design of video buffers in multimedia networks.

Using a large dataset collected from C-SPAN programs and movies we have illustrated various exploratory data analysis techniques. We have shown that the

Table 3.3
Estimated parameter values for the three distribution models. For each model the K-S statistic, D, is also given.

	cspan		movie	
Data	α	β	α	β
Pareto	0.3	32	0.40	12
	$D = 0.2148$		$D = 0.2732$	
Gamma	0.31	3.79×10^{-4}	0.41	23.0×10^{-4}
	$D = 0.3282$		$D = 0.3319$	
Weibull	1.05	697.00	1.32	171.02
	$D = 0.1440$		$D = 0.1287$	

shot length distributions possess heavy tails which is a very important fact for queuing analysis. Using various graphical tools we have shown how the parameters of the distributions may be estimated. Finally we have compared the three possible distribution models using the Kolmogorov-Smirnov statistic.

4. AUTOMATIC DETECTION OF PROGRAM GENRE

4.1 Introduction

Video programs belonging to the same program genre have many structural similarities. This is caused by the fact that these sequences are edited by professional video editors to convey a specific message to the audience or elicit a particular viewing viewing experience. Each program genre has a number of well-established editing rules and patterns that the video editors follow. The set of rules for a particular program genre is known as the *video grammar*. As an example, consider news programs which are highly structured and follow a similar pattern, regardless of the particular television channel they were prepared for. They generally contain a sequence of one or more anchor person shots followed by a news story. Therefore a typical instantiation of a news program might have the sequence $\{AAANNANNNNAANANN\ldots\}$, where A is an anchor person shot and N is a news story shot. Of course, no two news programs, even those from the same television channel, will have the same sequence. However, given a number of news programs, it is possible to build stochastic models to capture the underlying structure of the genre. In other words, given many instantiations of news programs, we can estimate the set of rules, or the video grammar, that was used to produce these sequences. Similar observations can be made for other program genres, such as soap operas, documentary programs, talk shows, sports programs, and sitcoms.

Building a stochastic model for a program genre has many important applications. Such a model can be used to classify a given video sequence to a program genre. It can also be used to summarize programs by determining which parts of the program are more interesting to viewers according to some criteria. Finally, a program genre

model can be used in the automatic production of video, e.g. unedited video material may be edited in the style of a documentary program or a news program.

In this chapter we examine the problem of deriving stochastic program genre models discuss how these models can be applied to the task of classifying a given video sequence to a program genre. We study three different stochastic models, hidden Markov models (HMMs), stochastic context-free grammars (SCFGs), and a hybrid model that is a SCFG built on HMM processing elements.

In order to be able to build these models we need to derive a label for shots, such as the A and N labels, used for the news program example above. Ideally labels that reflect high-level semantic properties of shots would provide the most useful information in developing program genre models. Example of such true semantic labels such as "young girl running," "blue dress," and "park scene" characterize a shot based on its content. Currently, however, automatic extraction of such truly semantic features is hard and computationally expensive. One way to circumvent this problem is to define shot features that correlate well with high level semantic labels and hence are useful deriving informative shot labels. We call such features *pseudo-semantic features*. We derive shot labels by clustering pseudo-semantic features extracted from shots. These shot labels are then used to train a stochastic program genre models. A schematic diagram of our system for building the models is shown in Figure 4.1. For each video sequence Σ^k in the training set, we extract a pseudo-semantic feature vector from each shot S_j^k belonging to Σ^k. All extracted feature vectors for all the shots in the training set are combined and the number of clusters and the parameters for each cluster are estimated. We then use the estimated cluster parameters to label each shot, to obtain a symbol sequence $\mathbf{T^k}$ for the video sequence Σ^k. The symbol sequences corresponding to programs belonging to a particular program genre are then used to build a stochastic model for that genre.

The structure of this chapter is as follows: After reviewing previous work on the application of stochastic video models for various video analysis tasks in Section 4.2, we describe pseudo-semantic feature extraction and how these features are used

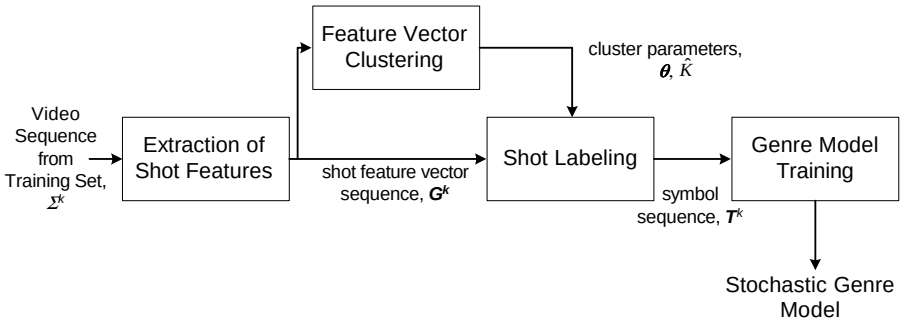


Fig. 4.1. Block diagram illustrating the training of the stochastic program genre models.

to derive shot labels in Section 4.3. The three stochastic models we use, namely HMMs, SCFGs, and the hybrid HMM-SCFG model, are introduced in Section 4.4, Section 4.5, and Section 4.6, respectively. Results for program genre classification are presented in Section 6.6. Finally, conclusions are provided in Section 6.7.

4.2 Previous Work on Stochastic Models for Video Analysis

HMMs are versatile tools to analyze time series whose statistical properties may change with time. Their applications range from spoken word recognition [72] to analysis of DNA sequences [73].

A number of researchers have applied HMMs to different video analysis tasks. Boreczky and Wilcox [74] used a HMM trained on audio features in addition to the motion features derived from adjacent frames to detect shot boundaries. Eickeler and Muller [75] use image features derived from difference images to train a HMM which in turn is used both to detect shot boundaries and classify types of shots for news sequences. Using shot labels such as *medium shot* or *close-up*, Wolf [76] has trained a HMM to detect dialogs in video sequences. Another dialog detection algorithm is described in [77] Liu et al [78] have used various features derived from audio to classify TV programs using a discrete HMM. Applications of HMMs to event detection in video programs are given in [79, 80].

SCFGs have been widely used in natural-language processing; however they have not been used as often as HMMs for video sequence analysis, except for some studies in video event recognition [81, 82].

4.3 Shot Feature Extraction and Labeling

In this section we describe how we generate discrete shot labels which are used in building stochastic models for video sequences, as will be discussed in later sections. Our goal is to derive shot labels that correlate well with the semantic content of the shots and are easily derivable from the compressed video stream with reasonable computational burden.

The first processing step in obtaining shot labels is determining the shot boundaries in the given video sequence, as described in Chapter 2. After shot boundary locations are determined, a number of features are extracted from each frame in each shot of the video sequences in the data set. These features are then aggregated to obtain a feature vector for each shot. Finally, clustering is used to derive the shot labels.

4.3.1 Shot feature extraction

We would like to extract features from shots that correlate well with semantic description of shot contents. However, it is desired that the extraction of features should not be computationally expensive To this effect, we extract a feature vector from each shot containing features that represent the editing pattern and the motion, color, and texture content of the shot. The feature vector corresponding to the shot S_j is denoted by $\mathbf{G}_j = [g_{j1}, \ldots, g_{jn}]^T$.

The first feature we use is the length of the shot defined as

$$g_{j1} = e(S_j) - b(S_j) + 1 \tag{4.1}$$

where $b(S_j)$ and $e(S_j)$ are the beginning and end frame numbers of shot S_j, respectively. The distribution of shot lengths was shown to be an important indicator of the genre and the tempo of a video program [83].

The amount of object or camera motion also provides important clues about the semantic content of the shot. Shot length and a measure of average shot activity have been shown to be useful features in classifying movie sequences to different genres [83,84]. We enhance these two basic shot features by three additional features based on the color and texture of the shot frames. The color features are obtained by averaging the pixel luminance and chrominance values within each frame and over the shot. The texture feature is calculated by averaging the variance of pixel luminance values for each macroblock within each frame and averaging these values for the shot.

In order to derive the features defined above for shot S_j, we first calculate the following features for each frame, f_k, that belong to S_j

$$m_k = \frac{1}{\#\text{MBs in } f_k \text{ with MVs}} \sum_{\text{MBs in } f_k \text{ with MVs}} (MV_x)^2 + (MV_y)^2 \tag{4.2}$$

$$T_k = \frac{1}{\#\text{MBs in } f_k} \sum_{MB_i \in f_k} \text{Var}(MB_i^Y) \tag{4.3}$$

$$Y_k = \frac{1}{XY} \sum_x \sum_y f_k^Y(x,y) \tag{4.4}$$

$$UV_k = \frac{1}{XY} \sum_x \sum_y \frac{f_k^U(x,y) + f_k^V(x,y)}{2} \tag{4.5}$$

$$\tag{4.6}$$

where the quantities that appear in these equations are defined as

MB is the 16×16 region of the frame f_k that is referred to as a macroblock in the MPEG standard. MV_x and MV_y are the horizontal and vertical components of the motion vector for each macroblock,

$\text{Var}(MB_i^Y)$ is the variance of the luminance component of the pixels in macroblock i,

$f_k^Y(x,y)$, $f_k^Y(x,y)$, and $f_k^Y(x,y)$ are the luminance and the two chrominance components of the pixel at coordinates (x,y) in f_k, and

X and Y are the width and length of f_k.

In these equations, m_k is the average value of the magnitudes of the components of the motion vectors of the blocks in the frame. This is the frame motion feature given in the MPEG-7 standard [85]. Note that this feature will be identically zero for I-frames in the MPEG stream. T_k is the average value of macroblock variances of the frame and is a measure of the texture of f_k. Finally, Y_k and UV_k are the mean luminance and chrominance values for the frame.

The shot features corresponding to these frame features are then calculated by averaging them over all the frames in the shot,

$$\{g_{j2}, g_{j3}, g_{j4}, g_{j5}\} = \frac{1}{C(S_j)} \sum_{k=1}^{C(S_j)} \{m_k, Y_k, UV_k, T_k\} \tag{4.7}$$

where $C(S_j)$ is the number of frames in S_j.

At the end of the shot feature extraction process each video sequence in the data set, Σ_k, is represented by a sequence of shot feature vectors $\{\mathbf{G}_j\}_{k=1}^{N_k}$ where N_k is the number of shots in the sequence Σ_k. Each shot feature vector $\mathbf{G}_j$ has a dimensionality of $n = 5$. We will use $\mathbf{G}_j$ to denote random feature vectors and $\mathbf{g}_j$ for their realizations.

4.3.2 Feature Clustering and Determination of Model Order

After the shot feature vectors are extracted from shots for all the video sequences in our training data set, they are modeled using a Gaussian mixture model. We use the Expectation-Maximization (EM) algorithm to estimate the parameters of the mixture model and agglomerative clustering to estimate the number of clusters from training data. In this approach the component mixtures are viewed as clusters, and starting with a large number clusters, we merge two clusters at each step until one

cluster remains. The number of clusters which maximizes a goodness-of-fit measure is chosen as the final model order.

We collect the shot feature vectors from all video sequences in the training set and number them consecutively, obtaining the collection $\mathbf{G}_{tr} = \cup_{k=1}^{N_{tr}} \{\mathbf{G}_1^k, \ldots, \mathbf{G}_{N_k}^k\}$, where N_{tr} is the total number of training sequences from all program genres in the data set. We model this collection using K clusters where the probability density function (pdf), p_k, for each cluster k is multivariate Gaussian. The parameters for p_k are $\boldsymbol{\theta}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, which represent the mean vector and the covariance matrix of the cluster, respectively. In this case the pdf for a shot feature vector $\mathbf{G}_j$, given that it belongs to the k^{th} cluster, is

$$p_k(\mathbf{g}_j; \boldsymbol{\theta}_k) = \frac{1}{(2\pi)^{M/2}} |\boldsymbol{\Sigma}_k|^{-1/2} \exp\left\{ -\frac{1}{2} (\mathbf{g}_j - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\mathbf{g}_j - \boldsymbol{\mu}_k) \right\} \tag{4.8}$$

Then, assuming that the shot feature vectors are independent and identically distributed (i.i.d.), we can write the log-likelihood for $\mathbf{G}_{tr}$ as

$$L(\boldsymbol{\Psi}) = \sum_{i=1}^{N_{tr}} \log \left(\sum_{k=1}^{K} \pi_k p_k \left(\mathbf{g}_i; \boldsymbol{\theta}_k \right) \right) \tag{4.9}$$

where $\boldsymbol{\Psi} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K; \pi_1, \ldots, \pi_{K-1})$ is the complete set of parameters specifying the model and π_k are the mixture probabilities, subject to the constraint $\sum_{k=1}^{K} \pi_k = 1$. For a fixed value of K, we use a EM-based approach to find a local maximum of the likelihood function to obtain the maximum likelihood estimate (MLE) of the parameter vector, $\hat{\boldsymbol{\Psi}}_{ML}$.

In the above analysis we assumed that the number of clusters, K, were known *a priori*. In practice this is not the case and K must also be estimated from data. It is not possible to estimate K using an approach similar to above, since the MLE for the number of clusters, $\hat{K}_{ML}$, is not well-defined. This is due to the fact that $L(\boldsymbol{\Psi})$ can always be made larger by increasing the number of clusters for $\hat{K} \leq N$. Methods for estimating model order generally require the addition of an extra term

to the log-likelihood of (4.9) that penalizes higher order models. We have used the minimum description length (MDL) criterion [86], which is defined as

$$MDL(K, \mathbf{\Psi}) = -L(\mathbf{\Psi}) + \frac{1}{2} R \log(Nn) \tag{4.10}$$

where R is the number of real-valued numbers required to specify the parameters of the model and n is the dimensionality of the feature vectors. In our case we have

$$R = K\left(1 + n + \frac{n(n+1)}{2}\right) - 1 \tag{4.11}$$

and $n = 5$. The minimization of the above criterion is performed iteratively using the EM algorithm. We start with a high number of initial clusters, usually 2–3 times the anticipated number of clusters, and at each step merge the two clusters which cause the maximum decrease in the MDL criterion. This process is continued until only one cluster is left. Then, the number of clusters for which the minimum value of MDL was achieved is chosen as the estimate of the number of clusters for the model, $\hat{K}$.

4.3.3 Shot Labeling

The cluster mixture model estimated using the procedure described in Section 4.3.2 is used to obtain a discrete label for each shot feature vector. The label for each shot is determined by the cluster number that the shot feature vector is most likely to belong to, that is, given the shot feature vector, $\mathbf{G}_j$, we determine the corresponding shot label symbol, t_j, using

$$t_j = \arg \max_{k \in \{1,\ldots,\hat{K}\}} p_k(\mathbf{g}_j;\, \boldsymbol{\theta}_k) \tag{4.12}$$

where the shot label t_j is an integer in the range $\{1, \ldots, \hat{K}\}$.

4.4 Hidden Markov Models for Program Genre

In this section we briefly review the theory for HMMs; for an in-depth analysis of HMMs refer to [72]. In a HMM an underlying and unobserved sequence of states

follows a Markov chain with finite state space, and the probability distribution of the observation at any time is determined only by the current state of that Markov chain [87]. A discrete-time, discrete-symbol HMM, λ, with N states and M output symbols is a 5-element structure $\langle \mathcal{S}, \mathcal{W}, \mathbf{A}, \mathbf{B}, \boldsymbol{\pi} \rangle$ where $\mathcal{S} = \{s_1, \ldots, s_N\}$ is the set of states, $\mathcal{W} = \{w_1, \ldots, w_M\}$ is the set of output symbols, $\mathbf{A}$ is the $N \times N$ state transition probability matrix, $\mathbf{B}$ is the $N \times M$ observation symbol probability distribution matrix, and $\boldsymbol{\pi}$ is the $N \times 1$ initial state distribution vector. Once the initial state of the HMM is chosen using the $\boldsymbol{\pi}$ vector, at each value of the discrete time t, the HMM emits a symbol according to the symbol probability distribution in current state, chooses another state according to the state transition probability distribution for the current state, and moves onto that state.

4.4.1 Hidden Markov Model Topology

An important consideration in building HMMs is the particular model topology selected. There are no general rules for selecting the topology, it has to be determined based on the properties of the system to be modeled. One possibility is to use an *ergodic* or fully connected HMM, for which all the elements of the transition matrix are nonzero, which implies that every state can be reached from any other state. Another model, which is widely used in speech recognition, is the *left-right* model which has the properties that $a_{ij} = 0$ for $j < i$, i.e., no transitions are allowed to states whose indices are lower than the current state, and that the sequence always starts in state 1 and ends in state N. In [75] a left-right HMM was used to segment and classify semantic components of news programs. However, we believe that an ergodic model is more suitable for modeling the structure of video programs because they generally exhibit recurring characteristics. Hence, we have used an ergodic HMM for genre modeling.

4.4.2 Distance Measure on Hidden Markov Models

For some applications we might want to compare two HMMs to determine their similarity. For example, one might to know if a HMM trained on soap operas is in some sense similar to one trained on sitcoms. The similarity between HMMs cannot be easily determined by inspection of the model parameters. It can be shown that the parameters for two HMMs, λ_1 and λ_2, may look different, yet the HMMs may be equivalent in the sense that the statistical properties for the observation symbols are the same, i.e., $E\{O_t = v_k|\lambda_1\} = E\{O_t = v_k|\lambda_2\}$ [88].

A directed distance measure between two HMMs, λ_i and λ_j, may be defined as

$$D(\lambda_i, \lambda_j) = \lim_{T\to\infty} \frac{1}{T} \left(\log P(O^{(i)}|\lambda_i) - \log P(O^{(i)}|\lambda_j)\right) \tag{4.13}$$

where $O^{(i)} = O_1 O_2 \cdots O_T$ is a sequence of observations generated by model λ_i. For ergodic HMMs it can be shown that $D(\lambda_i, \lambda_j)$ is the Kullback-Leibler divergence between the two probability distribution functions $p(\cdot|\lambda_0)$ and $p(\cdot|\lambda_0)$ [88]. The distance measure in defined by (4.13) is non-symmetric. One can obtain a symmetric distance between models as

$$d(\lambda_i, \lambda_j) = \frac{D(\lambda_i, \lambda_j) + D(\lambda_j, \lambda_i)}{2}. \tag{4.14}$$

The distance $d(\lambda_i, \lambda_j)$ represents a measure of the difficulty of discriminating the model λ_i from the model λ_j.

4.5 Stochastic Context-Free Grammars

Most video programs have a hierarchical structure where shots are grouped into scenes and scenes are grouped into larger segments. Such a hierarchical model for video suggests that shots that are far apart in the program may actually be semantically related. Linear models, such as HMMs, fail to model such long-range dependencies within sequences. In order for a model not only the linear order of shots but also the hierarchical nature of a sequence it has to be able to allow recursive embedding. The simplest probabilistic model with such property is a stochastic

context-free grammar (SCFG). SCFGs offer the simplest and most natural probabilistic model for tree structures and the algorithms for them follow from a natural extension from ones developed for HMMs. The predictive power of a SCFG, as measured by entropy, is greater than that for HMMs with the same number of parameters [89]. In this section we briefly review the theory for SCFGs; for an in-depth analysis of SCFGs refer to [89].

Let $\mathcal{W} = \{w_1, \ldots, w_M\}$ be a set of symbols that we refer to as *terminals*. These symbols are equivalent to the set of output symbols that we defined for the HMMs. We define another set of symbols, $\mathcal{I} = \{I_1, \ldots, I_N\}$, which are called *nonterminals*. Symbols of the nonterminal type are capable of generating other symbols whereas symbols of the terminal type are not. We define a *production rule* to be either a unary or binary mapping [1] of the form

$$\begin{aligned} I_i &\rightarrow w_l, \\ I_i &\rightarrow I_j I_k, \quad I_i, I_j, I_k \in \mathcal{I},\ w_l \in \mathcal{W}. \end{aligned} \tag{4.15}$$

We assign a probability to each production rule, satisfying the set of constraints

$$\sum_j \sum_k P(I_i \rightarrow I_j I_k) + \sum_j P(I_i \rightarrow w_j) = 1, \quad i = 1, \ldots, N, \tag{4.16}$$

which implies that any symbol of nonterminal type gets transformed into another symbol with unit probability.

A SCFG, γ, is then specified as a 5-element construct $\langle \mathcal{I}, \mathcal{W}, \mathcal{R}, \mathcal{P}, \boldsymbol{\pi}_{root} \rangle$ where $\mathcal{R}$ is the set of all unary and binary production rules of the form given in (4.15), $\mathcal{P}$ is a set of probabilities associated with each rule in $\mathcal{R}$, and $\boldsymbol{\pi}_{root}$ is the initial probability distribution that determines which nonterminal is chosen as the first symbol, which is called the root symbol. After the root nonterminal is chosen using $\boldsymbol{\pi}_{root}$, at each value of the discrete time t, the SCFG chooses one of the rules originating from the

[1] The type of SCFG defined here is actually based on a special case of context-free grammars called the Chomsky normal form. However, there is no loss of generality since it can be shown that any SCFG can be transformed into an identical grammar in the Chomsky normal form in the sense that the languages produced by the two grammars will be identical.

current nonterminal and replaces the current node with the symbols on the right side of the rule. This process is continued in a recursive fashion until there are no more nonterminal symbols to be expanded, producing a tree structure which is called a *parse tree*. The probability for a given parse tree is found by multiplying the probabilities of all the rules that are used in the derivation of the tree. Given a sequence of symbols, the probability assigned to it by the SCFG is the sum of the probabilities for all the parse trees that could have produced the given sequence.

In order to train a SCFG, the grammar to be used is fixed and all production rules are assigned random a probability values uniformly distributed in the interval $[0, 1]$. Then, rule probabilities are estimated using the training sequences by maximizing the cumulative probability for the sequences in the training set. Similar to the Baum-Welch procedure for HMM training, an algorithm exists, known as the *inside-outside algorithm*, for the training of a SCFG [89–91].

Let a given video sequence Σ^k containing N_k shots be represented as a sequence of shot labels $\mathbf{T} = t_1 \cdots t_{N_k}$. We will use the notation t_{pq} to refer to the subsequence of symbols $t_p \cdots t_q$ in $\mathbf{T}$. Starting from the nonterminal I_j, if, after applying a series of production rules we can rewrite I_j as the sequence of symbols t_{pq}, then we say that the nonterminal I_j dominates the symbols t_{pq} and write I_j^{pq}. We define the *outside probability*, $\alpha_j(p, q)$, and the *inside probability*, $\beta_j(p, q)$, for the sequence $\mathbf{T}$ as

$$\begin{aligned} \alpha_j(p,q) &= P(t_{1(p-1)}, I_j^{pq}, t_{(q+1)L}) \\ \beta_j(p,q) &= P(t_{pq} \mid I_j^{pq}) \end{aligned} \tag{4.17}$$

The inside probability, $\beta_j(p, q)$, is the total probability of generating the subsequence of symbols t_{pq}, given that we start at the nonterminal I_j. The outside probability, $\alpha_j(p, q)$, is the probability of starting with the root symbol and generating the non-

terminal I_j and all the symbols of the sequence outside t_{pq}. The inside probabilities may be calculated recursively using the formula

$$\beta_j(p,q) = \begin{cases} P(I_j \to t_p), & p = q \\ \sum_{r,s} \sum_{d=p}^{q-1} P(I_j \to I_r I_s)\, \beta_r(p,d)\, \beta_s(d+1,q), & p < q \\ 0, & p > q \end{cases} \tag{4.18}$$

where $r, s \in \{1, \ldots, N\}$. Similarly, using the boundary conditions

$$\alpha_j(1, N_k) = \begin{cases} 1, & j = 1 \\ 0, & j \neq 1 \end{cases} \tag{4.19}$$

the outside probabilities are recursively calculated as

$$\alpha_j(p,q) = \begin{cases} \sum_{f,g \neq j} \sum_{e=q+1}^{N_k} \alpha_f(p,e)\, P(I_f \to I_j I_g)\, \beta_g(q+1,e) & \\ \quad + \sum_{f,g} \sum_{e=1}^{p-1} \alpha_f(e,q)\, P(I_f \to I_g I_j)\, \beta_g(e,p-1), & p < q \\ 0, & p > q \end{cases} \tag{4.20}$$

where $f, g \in \{1, \ldots, N\}$. Using the inside and outside probabilities, the inside-outside SCFG training algorithm reestimates the production rule probabilities at each iteration. Under the assumption that the training sequences are independent, the the likelihood of the training set is the product of the probabilities of the sequences it contains. Therefore, in the inside-outside algorithm we calculate the contribution of each sequence to the rule probabilities and sum these over the whole training set. The update formula for unary production rule probabilities at iteration k is given by

$$\tilde{P}^k(I_j \to t_k) = \frac{\sum_{k=1}^{N_{tr}} \sum_{h=1}^{N_k} \alpha_j(h,h)\, \beta_j(h,h)\, P(t_h = t_k)}{\sum_{k=1}^{N_{tr}} \sum_{p=1}^{N_k} \sum_{q=p}^{N_k} \alpha_j(p,q)\, \beta_j(p,q)} \tag{4.21}$$

where N_{tr} is the number of shot sequences in the training set and $P(t_h = w_k)$ indicates if the h^{th} shot label in $\mathbf{T}$ is equal to the k^{th} terminal symbol, i.e., we have

$$P(t_h = w_k) = \begin{cases} 1, & h^{th} \text{ shot label in } \mathbf{T} \text{ is equal to } w_k \\ 0, & \text{otherwise} \end{cases}$$

The update formulas for binary production rule probabilities at iteration k is given by

$$\tilde{P}^k(I_j \rightarrow I_r I_s) = \frac{\sum_{k=1}^{N_{tr}} \sum_{p=1}^{L-1} \sum_{q=p+1}^{L} f_i(p,q,j,r,s)}{\sum_{k=1}^{N_{tr}} \sum_{p=1}^{N_k} \sum_{q=p}^{N_k} \alpha_j(p,q)\, \beta_j(p,q)} \tag{4.22}$$

where

$$f_i(p,q,j,r,s) = \sum_{d=p}^{q-1} \alpha_j(p,q)\, \beta_r(p,d)\, \beta_s(d+1,q)\, \tilde{P}^{k-1}(I_j \rightarrow I_r I_s)$$

A drawback of SCFGs is that they contain many more parameters to be trained. An M terminal, N nonterminal SCFG has $N^3M + MN$ parameters to be compared to $N^2M + MN$ parameters that a HMM with the same output symbols and N states contains. Therefore, compared with linear models like HMMs, the computational complexity of the inside-outside algorithm is high. For each training shot label sequence Σ_k, each iteration of the inside-outside algorithm takes $O(N^3 N_k^3)$ computations [91]. This fact poses more problems for the application of SCFGs to video program genre modeling than natural language modeling, since the number of symbols in a shot label sequences, i.e., the number of shots in a video program, is usually much larger compared to the number of words in a sentence.

4.6 The Hybrid HMM-SCFG Model

In Section 4.5 we pointed out the advantages and drawbacks of HMMs and SCFG. Training of HMMs have low computational complexity while they fail to capture the hierarchical structure that is inherent in many video programs. On the other hand, SCFGs are more suited to model video grammar but their training has high computational complexity. In this section we describe a hybrid HMM-SCFG model that incorporates the advantages of both models. The model consists of a HMM bank that processes segments of a given symbol sequence. The output of these HMMs are then used to train a SCFG. This way, the SCFG can correct for any classification errors made by the HMMs. Also, since the SCFG is trained on the

output of segments and not the symbols themselves computational complexity of training is greatly reduced, compared to the standard SCFGs.

In our hybrid HMM-SCFG model, we introduce a new set of nonterminal symbols, $\tilde{\mathcal{I}} = \{1, \ldots, L\}$, that we call *preterminals*, which can only appear on the right side of a production rule. Using these preterminal symbols, we modify the form of the unary SCFG rules from the form given in (4.15) to the form

$$I_j \rightarrow \tilde{I}_l, \quad I_j, \in \mathcal{I}, \tilde{I}_l \in \tilde{\mathcal{I}}. \tag{4.23}$$

This implies that the output of the HMM-SCFG model is not shot labels, as was in the SCFG model, but the values of the preterminals.

The interpretation of the values assumed by the preterminals is as follows: Let the number of program genres that a given sequence can be classified into be L. First, we train a HMM for each genre as explained in Section 4.4, using full-length shot label sequences, thereby obtaining L HMMs, $\lambda_1, \ldots, \lambda_L$. Then, we break up a given training shot label sequence $\mathbf{T}$ into P consecutive segments, $\mathbf{t}_j, j = 1, \ldots, P$. After this step the segment $\mathbf{t}_1$ will contain the shot labels $t_1 \cdots t_{\lfloor L/P \rfloor}$, and similarly for the other segments. For each segment $\mathbf{t}_j$, we obtain L likelihood values, $P(\mathbf{t}_j|\lambda_l)$, using the L HMMs. In this case the preterminal $\tilde{I}_l$ takes on integer values in the interval $[1, L]$ and specifies the particular HMM that was used to obtain the likelihood value for segment $\mathbf{t}_j$. This implies that in order to calculate the unary rule probability $P(I_j \rightarrow t_l)$ for the SCFG, we need to sum the contributions from all L HMMs, that is, we have

$$P(I_j \rightarrow \mathbf{t}_k) = \sum_{l=1}^{L} P(I_j \rightarrow l) P(l \rightarrow \mathbf{t}_k). \tag{4.24}$$

The probabilities $P(l \rightarrow \mathbf{t}_k) = P(\mathbf{t}_k|\lambda_l)$ are obtained using the HMM models, whereas the probabilities $P(l \rightarrow \tilde{I}_l)$ have to be estimated, similar to the estimation of the unary rule probabilities for the inside-outside algorithm for standard SCFGs. The hybrid HMM-SCFG model is illustrated in Figure 4.2.

Dividing the input shot label sequences into segments, as was described above, has two advantages. First, it reduces he length of the input sequence from N_k to

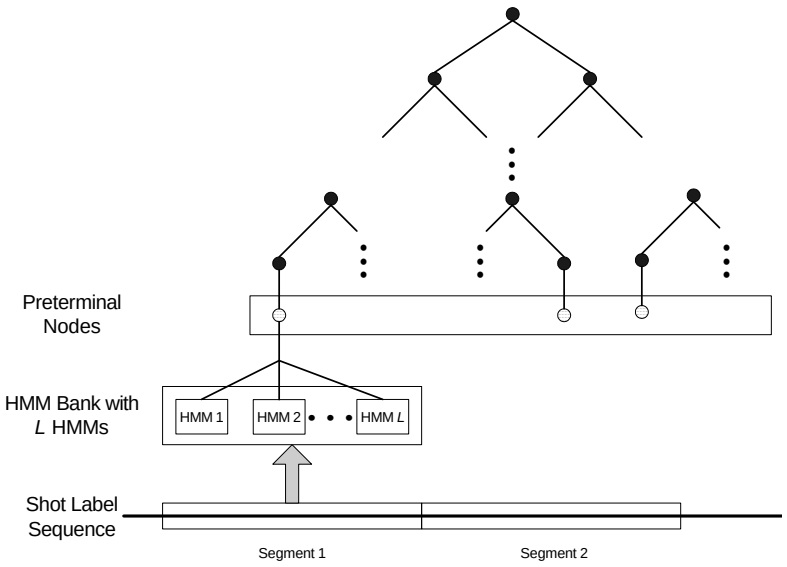


Fig. 4.2. The Hybrid HMM-SCFG model diagram illustrating the use of preterminal nodes.

P, where $P \ll N_k$. Second, it alleviates the problem of shot label sequences being possibly nonstationary over long intervals.

Based on the changes above for the HMM-SCFG model, The calculation of inside probabilities $\beta_j(p,q)$, defined in (4.18), is modified for the case of $p = q$ to

$$\beta_j(p,p) = \sum_l P(I_j \to \tilde{I}_l)\, P(\tilde{I}_l \to \mathbf{t}_p). \tag{4.25}$$

The formula for the reestimation of the unary rule probabilities given in (4.21) also needs to be modified to

$$\tilde{P}^k(I_j \to t_k) = \frac{\sum_{k=1}^{N_{tr}} \sum_{h=1}^{N_k} \alpha_j(h,h)\, P(I_j \to \tilde{I}_l) P(\tilde{I}_l \to \mathbf{t}_h)}{\sum_{k=1}^{N_{tr}} \sum_{p=1}^{N_k} \sum_{q=p}^{N_k} \alpha_j(p,q)\, \beta_j(p,q)} \tag{4.26}$$

The reestimation of the binary rule probabilities given in (4.22) is unchanged.

As pointed above in Section 4.5 the complexity of the unmodified training algorithm for a SCFG with N nonterminals on sequence Σ_k with N_k shots is $O(N^3 N_k^3)$ The training complexity of our approach is $O(N^3 P^3)$. The difference is significant, since in general we have $N_k \gg P$. For example, a typical sitcom program has

approximately 300 shots per episode, therefore $N_k = 100$ for such a program. If we use a value of $P = 10$, then the time complexity of training a program model for an episode using our HMM-SCFG model will be $\frac{300^3}{10^3} = 27000$ smaller than the unmodified inside-outside algorithm.

4.7 Maximum a posteriori Classification of Sequences using Program Genre Models

Our main goal in this chapter was to classify a given video sequence into a predetermined number of program genres. We use the model probabilities for the classification of sequences. For each type of model, HMM, SCFG, or hybrid HMM-SCFG, we first train program genre models using a number of training sequences with known genre labels. We extract features and perform feature vector clustering for the sequences in the training set, as described in Section 4.3. The number of output symbols was taken to be equal to the number of shot labels, i.e., we take $M = \hat{K}$. For HMMs, the order of the model was also taken to equal to $\hat{K}$. For SCFGs, different values for the number of internal terminals, N, were tries and the value that provides the highest cumulative model probability for the sequences in the training set was selected.

Let there be L program genres that we want to use for classification. In order to perform genre classification, we train L program genre models $\{\omega_1, \ldots, \omega_L\}$. For a given video sequence to be classified, we first calculate the shot label sequence, $\mathbf{T}$, corresponding to it. Then, the a posteriori probabilities that each of the L models could have produced $\mathbf{T}$ are calculated. We use the maximum a posteriori rule and classify the given sequence the genre with the highest a posteriori probability. Assuming all genres are equally likely, the classification of a given video sequence Σ with shot label sequence $\mathbf{T}$ is performed using the equation

$$\text{genre of } \Sigma = \max_{k \in \{1,\ldots,L\}} P(\mathbf{T}|\omega_k). \tag{4.27}$$

Table 4.1
Statistical information about the sequences used in the genre recognition experiments.

program genre	# sequences	# shots	mean sequence length (min)	mean number of shots per sequence
soap	11	1541	14.35	140.1
comedy	11	2908	20.16	264.4
cspan	29	1279	21.96	44.1
sports	14	1185	12.30	84.6
news	13	1483	12.02	114.1
movie	12	2310	15.95	192.5

4.8 Experimental Results

In this section we present the results of genre classification experiment we performed using HMMs.

4.8.1 The Experimental Data Set

We have selected 90 sequences from our database belonging to 6 different program genres with a total duration of more than 25 hours and containing a total of 10706 shots. The details of how these sequences were digitized were described in Section 1.5. Statistical information about the sequences used in the genre detection experiments is given in Table 4.1.

We have randomly divided the data set into two subsets, a training set and a test set, each containing approximately the same number of sequences from each genre. The sequences in the training set are used to build the stochastic program genre models while the sequences in the test set are used to measure the classification performance of the models built.

4.8.2 HMM Genre Recognition Experiments

We followed the following procedure in determining the classification performance of HMMs for genre detection

for $i = 1$ to 100

build program genre HMMs, $\lambda_1, \ldots, \lambda_6$, using training set

classify all sequences in a given data set using $\lambda_1, \ldots, \lambda_6$

obtain classification performance

average classification performance over all iterations

The experiment was repeated 100 times in order to average the performance of HMMs which were trained using a different set of initial values for parameters. At each iteration we compute the misclassification rate for each genre and average these values to obtain an average misclassification rate for that particular set of HMMs.

The average confusion matrices for the training and test sets for 100 iterations of the HMM genre detection experiment are given in Table 4.2 and Table 4.3. From these results we can make the following observations: Soap operas, sitcoms and C-SPAN programs are classified with high accuracy. There is some confusion between sports programs and news programs. This is to be expected since the fast-paced syntax of some news stories resemble sports programs in some respects. The worst classification performance is for the movie sequences, which get confused with soap operas. This is due to the fact that most of the movie sequences we had consisted mostly of dialogs that are similar to soap operas.

Note that the classification performance results tabulated in Table 4.2 and Table 4.3 are not symmetric with respect to genres. For example, from Table 4.3 we see that soap opera programs are confused with sitcoms approximately 25% of the time; however, sitcom sequences are not confused with soap opera programs at all. We can explain this asymmetry in classifier performance by calculating the model distances between HMMs using the directed model distance defined in (4.13). These distance

Table 4.2
Genre classification results using HMMs on the training set.

	soap	comedy	cspan	sports	news	movie
soap	6.00	0.00	0.00	0.00	0.00	0.00
comedy	0.00	6.00	0.00	0.00	0.00	0.00
cspan	0.00	0.00	7.00	0.00	0.00	0.00
sports	0.00	0.00	0.11	7.50	0.38	0.01
news	0.00	0.00	0.00	0.00	8.00	0.00
movie	1.02	0.75	0.00	0.00	0.00	4.23

Table 4.3
Genre classification results using HMMs on the test set.

	soap	comedy	cspan	sports	news	movie
soap	4.00	1.00	0.00	0.00	0.00	0.00
comedy	0.00	4.01	0.00	0.00	0.99	0.00
cspan	0.00	0.04	5.81	0.06	1.09	0.00
sports	0.00	0.00	0.04	4.92	1.04	0.00
news	0.00	0.00	0.00	1.67	3.33	0.00
movie	2.52	0.95	0.00	0.00	0.25	2.28

values are listed in Table 4.4. Continuing the soap–comedy genre example above, from Table 4.4 we get the value for the distance of the soap opera genre model to the sitcom genre model, $d(\lambda_{soap}, \lambda_{comedy}) = 2.4113$, and the distance of the sitcom genre model to the soap opera genre model as $d(\lambda_{comedy}, \lambda_{soap}) = 0.2867$. These distance values suggest that the similarity of the comedy model to the soap opera model is much smaller than the similarity for the soap opera model to the sitcom model. Since models that are similar will tend to be confused when classifying sequences, the HMM distances explain why soap opera sequences are sometimes labeled as sit-

Table 4.4
Directed distance values between program genre hidden Markov models calculated using (4.13).

	soap	comedy	cspan	sports	news	movie
soap	0.0000	2.4113	3.2045	3.4945	2.4346	2.3435
comedy	0.2867	0.0000	3.6094	3.8543	2.4113	2.2085
cspan	1.4372	1.6027	0.0000	2.9248	2.7695	3.4184
sports	1.2848	1.9622	1.0860	0.0000	2.4437	2.9806
news	0.4425	0.7548	1.0704	0.4737	0.0000	2.9293
movie	0.7965	0.8395	2.1587	1.5237	0.7525	0.0000

coms while the reverse error is much less likely. The same argument also explains the other classifier performance asymmetries found in Table 4.2 and Table 4.3.

4.9 Experimental Results for comparison of HMM and HMM-SCFG

In a second experiment we compared the performance of the HMM with the proposed hybrid HMM-SCFG model on a different data set. This second data set consisted of a total of 23 sequences selected from four program genres, namely soap operas, sitcoms, C-SPAN programs, and sports programs.

The sequences in each genre were divided into sets containing roughly the same number of sequences. One of these sets were used as the training set, the other as the test set for the algorithms. We clustered the shot feature vectors obtained from the sequences in the training set, using the method described in Section 4.3.2. The cluster parameters so obtained were then used to label the shots of the sequences in both the training and test sets. We used six clusters, so the number of terminal symbols, $M = 6$.

We performed two genre classification experiments. In Experiment I a HMM for each genre was trained using the the training set and then used these HMMs to classify the sequences in the test set where the genre of each sequence was determined

Table 4.5
HMM genre classification confusion matrix. HMMs of order 6 were used.

	Classifier Output			
True Label	soap	comedy	cspan	sports
soap	4	1	0	0
comedy	0	5	0	0
cspan	0	1	6	0
sports	0	0	0	6

Table 4.6
SCFG-HMM genre classification confusion matrix. The same HMMs as the ones provided the results in Table 2 were used with a SCFG of 4 nonterminal nodes.

	Classifier Output			
True Label	soap	comedy	cspan	sports
soap	5	0	0	0
comedy	0	5	0	0
cspan	1	0	6	0
sports	0	0	0	6

using Equation 4.27. The number of states of each HMM was set to four. All the sequences in the training set were correctly classified. The results for the test set are given in Table 2.

In Experiment II we used the same HMMs that were used for Experiment I but we now used the hybrid SCFG-HMM model that was described in Section 4.6. The number of terminal nodes of the SCFG was set to four. Again, all the sequences in the training set were correctly classified. The results for the test set are shown in Table 3.

4.10 Conclusions

In this chapter we have described a method to analyze video sequences from different types of TV genres using stochastic models such as hidden Markov models and stochastic context-free grammars. We have proposed a hybrid HMM-SCFG scheme that greatly reduces the computational complexity of SCFG training. The stochastic models are built using shot labels derived from pseudo-semantic shot feature vectors. We have shown that agglomerative clustering of feature vectors using a Gaussian mixture model is an efficient way to determine the order of stochastic models if an a priori model for video structure does not exist.

Our genre classification experiments using the HMMs have shown that the classification performance is excellent for soap operas, sitcoms, and C-SPAN programs. The performance for sports programs and news programs is also good. Classification performance for movies was not good; we believe this was due to the fact that most movie sequences in our data set contained dialog scenes.

5. SUMMARIZATION OF VIDEO PROGRAMS

Deriving compact representations of video sequences that are intuitive for users and let them easily and quickly browse large collections of video data is fast becoming one of the most important topics in content-based video processing. Such representations, which we will collectively refer to as *video summaries*, rapidly provide the user with information about the content of the particular sequence being examined, while preserving the essential message. The need for automatic methods for generating video summaries is fueled both from the user and production viewpoints. With the proliferation of personal video recorder devices and hand-held cameras, many users generate many times more video footage that they can digest. On the other hand, in today's fast-paced news coverage, programs such as sports and news must be processed quickly for production or their value quickly diminishes. Such time constraints and the increasing number of services being offered, places a large burden on production companies to process, edit, and distribute video material as fast as possible.

Summarization, for a video, audio, or text document, is a challenging and ill-defined task since it requires the processing system to make decisions based on high-level notions such as the semantic content and relative importance of the parts of the documents with respect to each other. Objective evaluation of resulting media summaries is also a problem, since it is hard to derive quantitative measures of summary quality.

Our goal in this paper is two-fold: First, we propose an automated method to generate video skims for information-rich video programs, such as documentaries, educational videos, and presentations, using statistical analysis based on speech transcripts that are obtained by automatic speech recognition (ASR)from the audio. We

show how important phrases can be automatically extracted even from ASR text that has a relatively high word error rate. These detected phrases may be used to augment current summarization methods and visualization schemes. Ideally one would like the generated summaries to be both detailed and covering most of the important points of the full program they were derived from. Clearly, for high summarization ratios it is impossible to satisfy both of these constraints. Our summarization approach quantifies these two concepts and maximizes a weighted sum of both detail and coverage functions to obtain a trade-off between the two. This approach enables the user to change the weights and regenerate the video summary of a program with more detail or more coverage, depending on a particular application.

Our second main objective in this paper is to rigorously study the objective evaluation methods for video summaries. We evaluate summaries produced by a number of algorithms using a question and answer evaluation scheme and discuss other methods of summary evaluation.

The outline of this chapter is as follows: We first characterize various aspects of video summarization methodology, such as different application domains and summary visualization schemes in Section 5.1 and Section 5.2. The current state of the art in automatic video summarization is reviewed in Section 5.4. Our proposed summarization algorithm is described in detail in Section 5.5. In Section 5.10 we review some of the video summary evaluation schemes that were used in the literature. We describe the design of our user study,present experimental results, and provide a detailed analysis in Section 5.11. Finally, conclusions are given in Section 5.12.

5.1 Summary Application Domains

The goal of video summarization is to process video programs, which usually contain semantic redundancy, and make them more interesting or useful for users. The properties of a video summary depends on the application domain, the charac-

teristics of the sequences to be summarized, and the purpose of the summary. Some of the purposes that a video summary might serve are listed below.

- Intrigue the viewer to watch the whole video. Movie trailers, prepared by highly skilled editors and with high budgets, are the best examples of summaries of this type.

- Let the viewer decide if the complete program is worth watching. Summaries of video programs that may be used in personal video recorders are examples of this category of summaries. The user may have watched the episode that was recorded or may have already watched similar content so might not want to watch the program after seeing the summary.

- Help the viewer locate specific segments of interest. For example, in distance learning, students can skip parts of a lecture that they are familiar with and, instead, concentrate on new material.

- Let users judge if a video clip returned by a video database system in response to their query is relevant. In content-based image database applications the results of a user query are shown as thumbnail images, which can be judged at a glance by the user for relevance to the query. Judging the relevance of query results is time-consuming for video sequences, since search results may contain long sequences containing hundreds of shots. Presenting the summaries of the results would be much more helpful.

- Enable users of pervasive devices, such as personal digital assistants, palm computers, or cellular phones, to view video sequences, which these devices otherwise would not be able to handle due to their low processing power. Using summaries also result in significant downloading cost savings for such devices. An example of such an application is described in [92], It makes use of annotations based on the MPEG-7 standard. With the increased consumption

of video using cellular phones, some form of summarization will become very important.

- Give the viewer all the important information contained in the video. These summaries are intended to replace watching the whole video. Executive summaries of long presentations or videoconferences, would be an example of this type of summary.

The above list, while not exhaustive, illustrates the wide range of different types of video summaries one would like to generate. For most applications video summaries mainly serve two functions: the *indicative function*, where the summary is used to indicate what topics of information is contained in the original program; and the *informative function*, where the summaries are used to cover the information in the source program as much as possible, subject to the summary length. Clearly these two summary functionalities are not independent. Video summarization applications often will be designed to achieve a mixture of the two functionalities. the viewer's time and the environment where the summary will typically be consumed by the user, as well as the display characteristics of the device used are also important factors in what type of summaries to generate. For example, a summary of a lecture must contain the main results and conclusions, while a movie trailer must not reveal the punch line. Naturally, there is no single approach, neither manual nor automatic, that will apply to the generation of all types of video summaries.

5.2 Types of Program Content

An important distinction we make when considering video summarization algorithms is the type of the program content that will be summarized. For the purposes of video summarization we categorize video program content into two broad classes:

- *Event-based content.* Video programs of this type contain easily identifiable story units that form either a sequence of different events, or a sequence of

events and non-events. Examples of the first kind of programs are talk shows and news programs where one event follows another and their boundaries are well-defined. For talk shows each event contains an interview with a different guest while for news programs each event is a different news story. The best example of programs where sequence of events and non-events occur are sports programs. Here, the events may correspond to important events in games, such as touchdowns, home runs, or goals.

- *Uniformly informative content.* These are programs which cannot easily be broken down to a series of events as event-based content. For this type of content, most parts of the program may be equally important for the user. Examples of this type of content are sitcoms, presentation videos, documentaries, soap operas, and home movies.

Note that the distinction introduced above is not clear cut. For example, for sitcoms one can define events according to the appearance of audience laughter. Movies form another example: most action movies have a clear sequence of action and non-action segments.

For event-based content, since the types of events of interest are well-defined, one can use knowledge-based event detection techniques. In this case, the processing is generally domain-specific and a new set of events and event detection rules must be derived for each application domain, which is a disadvantage. However, the summaries produced will be more reliable than those generated using general-purpose summarization algorithms. This class of algorithms is examined in Section 5.4.4.

If domain-based knowledge of events is not available or if one is dealing with uniformly informative content, one has to resort to more general summarization techniques. This is generally done by first dividing the program to be summarized into a number of segments. Then, segments are selected for the summary either by deriving an importance score for each program segment and selecting segments with high scores, or by clustering similar segments together. the scores with high

segments are selected for the summary. Another approach is to cluster frames from the video directly. Once the segments to be included in the summary are identified, each segment is represented using either keyframes extracted from the segment or portions of video within the segment.

5.3 Summary Visualization Methods

After the parts of the video to be included in the summary are determined they have to be displayed to the user in an intuitive and compact manner. Depending on the desired summary type and length, information from all detected video segments in the full program may be used. Alternatively, importance scores may be assigned to each segment using various combinations of visual, audio, textual, and other features, and only portions or keyframes extracted from the segments with the highest scores may be included in the summary. We will use the term *video summarization* to denote any general method that can be used to derive a compact representation of a video program. Once a summary is obtained after processing the source video, it has to be presented to the user. We will use the term *summary visualization*, to refer to the method that is used to present the summary to the user. In this section we review some of the summarization display approaches that have been proposed. These methods mainly fall into two categories: Video abstracts [93–97] based on keyframes extracted from video and video skims [98–103] where portions of the source video are concatenated to form a much shorter video clip. For video skims the duration of the summary is generally specified by the user in terms of the *summarization ratio* (SR), which is defined as the ratio of the duration of the video skim to the duration of the source video. For video abstracts the user may specify the number of keyframes to be displayed.

5.3.1 Static Visualizations or Video Abstracts

The simplest static visualization method is to present one frame from each video segment, which may or may not correspond to an actual shot, in a storyboard fashion, sometimes accompanied by additional information such as timestamps and closed caption text. The problems with this method are that all shots appear equally important to the user and the representation becomes unpractically large for long videos.

One way to alleviate this problem is to rank the video segments and to display only the representative key-frames belonging to the segments with highest scores. In order to further reflect the relative scores of the segments the keyframes may be sized according to the score of the segment. This approach have been used in [93] and [96] where keyframes from segments are arranged in a "video poster" using a frame packing algorithm In their PanoramaExcerpts system Taniguchi, Akutsu, and Tonomura [94] use panoramic icons, which are obtained by merging consecutive frames in a shot, in addition to keyframes. In the Informedia project time ordered keyframes, known as filmstrips, were as video abstracts [104].

Although video abstracts are compact, since they do not preserve the time-evolving nature of video programs, they present fundamental drawbacks. They are somewhat unnatural and hard to grasp for non-experts, especially if the video is complex. Most techniques just present keyframes to the user without any additional metadata, like keywords, which can make the meaning of keyframes ambiguous. Finally, static summaries are not suitable for instructional, and presentation videos, as well as teleconferences, where most shots contain a talking head, and most of the relevant information is found in the audio stream. These deficiencies are addressed by dynamic visualization methods.

5.3.2 Dynamic Visualizations or Video Skims

In these methods the segments with the highest scores are selected from the source video and concatenated to generate a video skim. While selecting portions of the source video to be included in the video skim, care must be exercised to edit the video on long audio silences, which generally correspond to spoken sentence boundaries. This is due to the experimentally verified fact that users find it very annoying when audio segments in the a video skim begin in mid-sentence [98, 101].

5.3.3 Other Types of Visualizations

There are also some video browsing approaches which may be used to visualize video content compactly and hence may be considered a form of video summarization.

As part of the Informedia Project Wactlar [105] proposes video collages, which are rich representations that display video data along with related keyframes, maps, and chronological information in response to a user query. In their BMOVIES system Vasconcelos and Lippman [106] use a Bayesian network to classify shots as action, close-up, crowd, or setting based on motion, skin tone, and texture features. The system generates a timeline that displays the evolution of the state of the semantic attributes throughout the sequence. Taskiran *et al.* [107] cluster keyframes extracted from shots using color, edge, and texture features and present them in a hierarchical fashion using a similarity pyramid. In the CueVideo system, Amir *et al.* [108] provide a video browser with multiple synchronized views. It allows switching between different views, such as storyboards, salient animations, slide shows with fast or slow audio, content-based accelerating fast playback, and full video while preserving the corresponding point within the video between all different views. Ponceleon and Dieberger [109] propose a grid, which they call the movieDNA, whose cells indicate the presence or absence of a feature of interest in a particular video segment. When the user moves the mouse over a cell, a window shows a representative frame and other metadata about that particular cluster. A system to build a hierarchical rep-

resentation of video content is discussed in Huang *et al.* [110] where audio, video, and text content are fused to obtain an index table for broadcast news. Aner et al. [111] compose mosaics of scene backgrounds in sitcom programs. The mosaic images provide a compact static visual summary of the physical settings of scenes. By matching keyframes with mosaics, shots are clustered into scenes based on their physical locations, resulting in a compact scene-based representation of the program and an automatic reference across multiple episodes of the same sitcom.

5.4 Previous Approaches to Video Summary Generation

In this section we investigate various algorithms that have been proposed to generate video summaries.

5.4.1 Speedup of playback

A simple way to compactly display a video sequence is to present the complete sequence to the user but increase the playback speed. A technique known as time scale modification can be used to process the audio signal so that the speedup can be made with little distortion [112]. The compression allowed by this approach, however, is limited to a summarization ratio (SR) of 0.4–0.7, depending on the particular program genre [113]. Based on a comprehensive user study, Amir *et al.* report that for most program genres and for novice users, a SR of 0.59 can be achieved without significant loss in comprehension. However, this SR value is not adequate for most summarization applications, which often require a SR between 0.1 and 0.2.

5.4.2 Techniques Based on Frame Clustering

Dividing a video sequence into segments and extracting one or more keyframes from each segment was long recognized as one of the simplest and most compact way of representing video sequences. For a survey of keyframe extraction techniques

the reader is referred to [114]. These techniques generally focus on the image data stream only. Color histograms, because of their robustness, have generally been used as the features for clustering.

One of the earliest work in this area is by Yeung and Yeo [93], which uses time-constrained clustering of shots. Each shot is labeled according to the cluster it belongs to and three types events, *dialogue*, *action*, and *other*, are detected based on these labels. Representative frames from each event are then selected for the summary. The Video Manga system by Uchihashi *et al.* [96] clusters individual video frames using YUV color histograms. Iacob, Lagendijk, and Iacob [115] propose a similar technique. However, in their approach video frames are first divided into rectangles, whose sizes depend on the local structure, and YUV histograms are extracted from these rectangles. Ratakonda, Sezan, and Crinon [116] extract keyframes for summaries based on the area under the cumulative action curve within a shot, where the action between two frames is defined to be the absolute histogram difference between color histograms of the frames. They then cluster these keyframes into a hierarchical structure to generate a summary of the program.

Ferman and Tekalp [117] select keyframes from each shot using the fuzzy c-clustering algorithm, which is a variation of the k-means clustering method, based on alpha-trimmed average histograms extracted from frames. Cluster validity analysis is performed to automatically determine the optimal number of keyframes from each shot to be included in the summary. This summary may then be processed based on user preferences, such as the maximum number of keyframes to view, and cluster merging may be performed if there are too many keyframes in the original summary.

The approaches proposed in [114] and [118] both contain a two-stage clustering structure, which is very similar to the method used in [117] but instead of performing shot detection segments are identified by clustering of video frames. Hanjalic and Zhang use features and cluster validity analysis techniques that are similar to those in [117]. Farin, Effelsberg, and de With [118] propose a two-stage clustering technique based on luminance histograms extracted from each frame in the sequence.

First, in an approach similar to time-constrained clustering, segments in the video sequence are located by minimizing segment inhomogeneity, which is defined as the sum of the distances of all frames within a segment to the mean feature vector of the segment. Then, the segments obtained are clustered using the Earth-Mover's distance [9]. Yahiaoui, Merialdo, and Huet [119] first cluster frames based on the L_1 distance between their color histograms using a procedure similar to k-means clustering. Then, a set of clusters is chosen as to maximize the coverage over the source video sequence.

An application domain which poses unique challenges for summarization is home videos. Since home videos generally have no plot or summary and contain very little editing, if any, the editing patterns and high level video structure, which offer strong cues in the summarization of broadcast programs, are absent. However, home videos are inherently time-stamped during recording. Lienhart [100] proposes a summarization algorithm where shots are clustered at four time resolutions using different thresholds for each level of resolution using frame time stamps. Very long shots, which are common in home videos, are shortened by using the heuristic that during important events audio is clearly audible over a longer period of time than less important content.

5.4.3 Techniques Based on Frame Clustering by Dimensionality Reduction

These techniques perform a bottom-up clustering of the video frames selected at fixed intervals. A high dimensional feature vector is extracted from each frame and this dimensionality is then reduced either by projecting the vectors to a much lower dimensional space [120, 121] or by using local approximations to high dimensional trajectories [95,97] Finally, clustering of frames is performed in this lower dimensional space.

DeMenthon, Kobla, and Doerman [95] extract a 37-dimensional feature vector from each frame by considering a time coordinate together with the three coordinates of the largest blobs in four intervals for each luminance and chrominance channel. They then apply a curve splitting algorithm to the trajectory of these feature vectors to segment the video sequence. A keyframe is extracted from each segment. Stefanidis *et al.* [97] propose a similar system; however, they split the 3-dimensional trajectories of video objects instead of feature trajectories.

Gong and Liu [120] use singular value decomposition (SVD) to cluster frames evenly spaced in the video sequence. Each frame is initially represented using three-dimensional RGB histograms, which results in 1125-dimensional frame feature vectors. Then, SVD is performed on these vectors to reduce the dimensionality to 150 and clustering is performed in this space. Portions of shots from each cluster are selected for the summary. Cooper and Foote [121] sample the given video sequence at a rate of one frame per second and extract a color feature vector from each extracted frame. The cosine of the angle between feature vectors is taken to be the similarity measure between them and a non-negative similarity matrix is formed between all pairs of frames. Non-negative matrix factorization (NMF) [122], is used to reduce the dimensionality of the similarity matrix. NMF is a linear approximation similar to SVD, the difference being the fact that the basis vectors are non-negative.

5.4.4 Techniques Using Domain Knowledge

As discussed in Section 5.2, if the application domain of the summarization algorithm is restricted to event-based content, it becomes possible to enhance summarization algorithms by exploiting domain-specific knowledge about important events. Summarization of sports video has been the main application for such approaches. Sports programs lend themselves well for automatic summarization for a number of reasons. First, the interesting segments of a program occupy a small portion of the whole content; second, the broadcast value of a program falls off rapidly after the

event so the processing must be performed in near real-time; third, compact representations of sports programs have a large potential audience; finally, often there are clear markers, such as cheering crowds, stopped games, and replays, that signify important events.

Summarization of soccer has received a large amount of attention recently (see [123] for a survey of work in soccer program summarization). Li, Pan, and Sezan [124] develop a general model for sports programs where events are defined to be the actions in a program that are replayed by the broadcaster. The replay is often preceded by a close-up shot of the key players or the audience. They apply their approach to soccer videos where they detect close-up shots by determining if the dominant color of the shot is close to that of the soccer field. Ekin and Tekalp [123] divide each keyframe of a soccer program into 9 parts and use features based the color content to classify shots into long, medium, and close-up shots. They also detect shots containing the referee and the penalty box. Goal detection is performed similar to [124] by detecting close-up shots followed by a replay. Cabasson and Divakaran [125] detect audio peaks and a motion activity measure to detect exciting events in soccer programs. Based on the heuristic that the game generally stops after an exciting event, they search the program for sequences of high motion followed by very little motion. If an audio peak is detected near such a sequence it is marked as an event and included in the summary.

Domain knowledge can be very helpful even for uniformly informative content. For example, He *et al.* [101] have proposed algorithms based on heuristics about slide transitions and speaker pitch information to summarize presentation videos.

5.4.5 Techniques Using Closed-Captions or Speech Transcripts

For some types of programs a large portion of the informational content is carried in the audio. News programs, presentation videos, documentaries, teleconferences, and instructional videos are some examples of such content. Using the spoken text to

generate video summaries becomes a powerful approach for these types of sequences. Content text is readily available for most broadcast programs in the form of closed captions. For sequences, like presentations and instructional programs, where this information is not available, speech recognition may be performed to obtain the speech transcript. Once the text corresponding to a video sequence is available, one can use methods of text summarization to obtain a text summary. The portions of the video corresponding to the selected text may then be concatenated to generate the video skim. Processing text also provides a high level of access to the semantic content of a program that is hard to achieve using image content only.

Agnihotri *et al.* [126] search the closed-caption text for cue words to generate summaries for talk shows. Cues such as "please welcome" and "when we come back" in addition to domain knowledge about program structure are used to segment the programs into parts containing individual guests and commercial breaks. Keywords are then used to categorize the conversation with each guest into a number of predetermined classes such as *movie* or *music*. In their ANSES system Pickering, Wong, and Rueger [127] use key entity detection to identify important keywords in closed-caption text. Working under the assumption that story boundaries always fall on shot boundaries, they perform shot detection followed by the merging of similar shots based on the similarity of words they contain. They then detect the nouns in text using a part of speech tagger and use lexical chains [128] to rank the sentences in each story. The highest scoring sentences are then used to summarize each news story.

An example of a technique that uses automatic speech recognition (ASR) is the one proposed by Taskiran *et al.* [98]. The usage of ASR makes their method applicable to cases where the closed-caption text is not available, such as presentations or instructional videos. In their approach the video is first divided into segments at the pause boundaries. Then, each segment is assigned a score using term frequencies within segments. Using statistical text analysis, dominant word pairs are identified

in the program and the scores of segments containing these pairs are increased. The segments with highest scores are selected for the summary.

5.4.6 Approaches Using Multiple Information Streams

Most current summarization techniques focus on processing one data stream, which is generally image data. However, multi-modal data fusion approaches, where data from images, audio, and closed-caption text are combined, offer the possibility to greatly increase the quality of the video summaries produced. In this section we look at a few systems that incorporate features derived from multiple data streams.

The MoCA project [103], one of the earliest systems for video summarization, uses color and action content of shots, among other heuristics, to obtain trailers for feature films. The Informedia project constitutes a pioneer and one of the largest efforts in creating a large video database with search and browse capabilities. It uses integrated speech recognition, image processing, and natural language processing techniques for the analysis of video data [102, 105]. Video segments with significant camera motion, and those showing people or a text caption are given a higher score. Audio analysis includes detection of names in the speech transcript. Audio and video segments selected for summary are then merged while trying to maintain audio/video synchronicity.

Ma *et al.* [129] propose a generic user attention model by integrating a set of low-level features extracted from video. This model incorporates features based on camera and object motion, face detection, and audio. An attention value curve is obtained for a given video sequence using the model and portions near the crests of this curve are deemed to be interesting events. Then, heuristic rules are employed, based on pause and shot boundaries, and the SR, to select portions of the video for the summary. Another model-based approach is the computable scene model proposed by Chang and Sundaram [130], which uses the rules of film-making and

experimental observations in the psychology of audition. Scenes are classified into four categories using audio and video features.

5.5 Summary Generation Using Speech Transcripts

As can be seen from the survey of video summarization approaches given in Section 5.4, most methods for video summarization do not make use of one of the most important sources of information in a video sequence, the spoken text or the natural-language content, Informedia, CueVideo and the system proposed in [110] being some exceptions. Speech transcripts are readily available for TV programs in the form of closed captions. For video programs like seminars and instructional programs, where this information is not available, speech recognition may be performed on audio to obtain the transcript. Once the text corresponding to a video sequence is available, one can use methods of text summarization to obtain a text summary. The portions of the video corresponding to the selected text are then concatenated to generate the video skim.

The techniques used in text summarization may be roughly divided into two groups:

- *Statistical analysis based on information-retrieval techniques.* In this approach, the problem of summarization is reduced to the problem of ranking sentences or paragraphs in the given text according to their likelihood of being included in the final summary. This likelihood is a high level semantic quality of a sentence, compromising notions of meaning, relative importance of sentences, style, and the like. In these techniques, instead of employing natural language understanding methods, various features are extracted from the text which were shown to be correlated with the "abstract-worthiness" of a sentence, and the ranking is done using a combination of these features.

- *Natural Language Processing (NLP) analysis based on information-extraction techniques.* This paradigm, making use of techniques from artificial intelli-

gence, entails performing a detailed semantic analysis of the source text to build a source representation designed for a particular application. Then a summary representation is formed using this source representation and the output summary text is synthesized [131].

Methods using statistical processing to extract sentences for the summary often generate summaries that lack coherence. These methods also suffer from the *dangling anaphor problem*. Anaphors are pronouns, demonstratives, and comparatives like "he," "this," or "more," which can only be understood by referring to an antecedent clause appearing before the sentence in which the these words occur. If the antecedent clause has not been selected for the summary, anaphors may be confusing for the user. Although techniques based on NLP generate better summaries, the knowledge base required for such systems is generally large and complex. Furthermore such systems are specific to a narrow domain of application and are hard to generalize to other domains. Moreover, attempts to directly apply NLP methods on text decoded by speech recognition usually fail because of the poor quality of the decoded text, which is reflected in a high word error rate and lack of punctuation. We will use the statistical analysis approach in processing the speech transcripts. A block diagram of the proposed system is shown in Figure 5.1.

5.6 Program Segmentation Using Audio and Speech

The first step in our video summarization system is the segmentation of the given program into a number of segments. One approach to segmentation is to use shots as segments, thereby dividing the video into visually coherent groups. However, shot boundaries are not aligned with audio boundaries in most types of video programs, that is, a shot boundary may occur in the middle of a sentence in audio. In user studies of summarization algorithms it has been found that users find it annoying when audio segments in the summary begin in the middle of a sentence [101]. Therefore, instead of using shot boundaries to segment video into segments, we use long

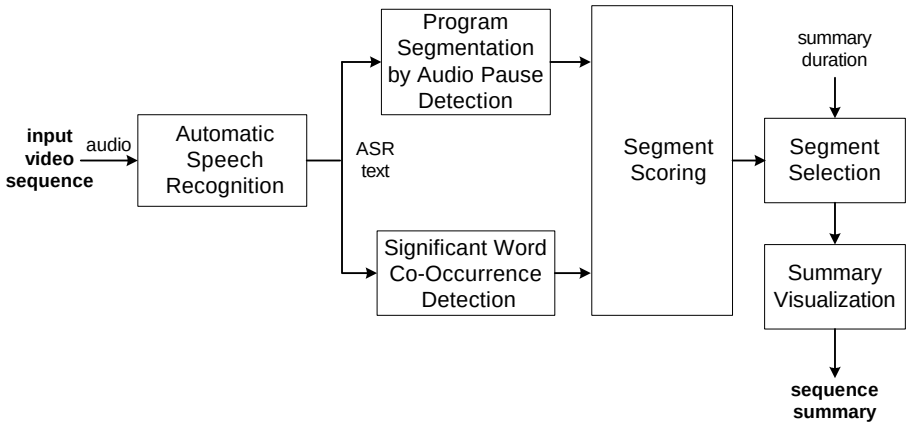
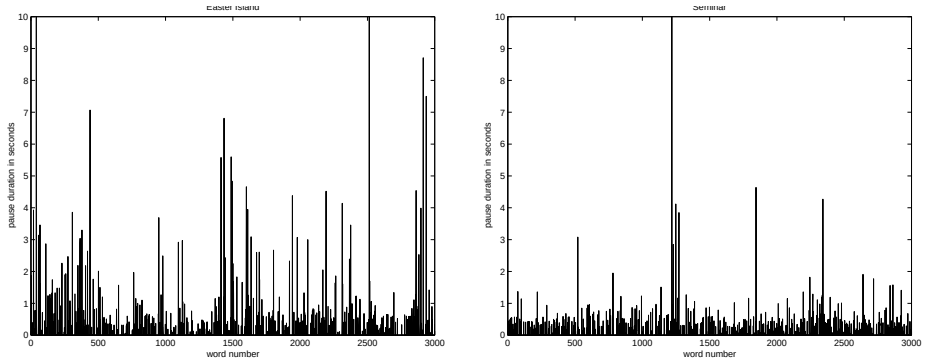


Fig. 5.1. Block digram of the video summarization system described in this chapter.

inter-word pauses in audio, which generally coincide with sentence boundaries. We define the duration of video between two consecutive long pauses to be a *segment*, which form the smallest processing units for our summarization system. An advantage of this approach to segmentation is that it avoids having very long segments which could occur if each shot is used as a segment, due to long shots that commonly occur in videos of presentations and meetings.

There are many techniques available to detect prolonged silence in speech. However, we use a simple yet accurate heuristic using the speech recognition text generated by the large vocabulary IBM ViaVoice speech recognition system [132]. This text contains the time stamp for each word detected by the speech recognition system together with its estimated duration. Based on these timestamps we calculate the duration between words w_i and w_{i+1} as $p_i = t_{i+1} - (t_i + l(w_i))$, where t_i and t_{i+1} are the start timestamps of words w_i and w_{i+1} in the video, and $l(w_i)$ is the estimated duration of word w_i. The distribution of interword durations of two video sequences, a wildlife documentary and a seminar, are shown in Figure 5.2. The durations between words in the documentary are generally larger since this is an educational program with a well-defined script and the speaker speaks at a relaxed



(a) documentary (b) seminar

Fig. 5.2. The distribution of pauses between consecutive words detected by automatic speech recognitions in two programs with different types of content for the first 3000 words.

pace for the audience to easily follow. On the other hand, the pauses in the seminar video are generally quite short, corresponding to free-flowing natural speech, with vocalized pauses, such as "ah" or "um," when the speaker is thinking about how to phrase a thought. The variability between these plots imply that a global threshold cannot be used to detect segment boundaries.

We use a sliding window scheme to robustly detect long pauses in audio that is similar to the cut detection method given in [55]. Let $\mathbf{W} = \{p_{i-m}, \ldots, p_i, \ldots, p_{i+m}\}$ be a sliding window of size $2m+1$ centered at the inter-word duration that is being currently considered for marking as a long pause. We declare the pause p_i to be a segment boundary is the following conditions are met

1. The value p_i is the maximum within $\mathbf{W}$.
2. The value p_i is also n times larger than the mean value of $\mathbf{W}$, not including the value p_i, that is

$$p_i \geq n \left(\frac{\sum_{j=i-m, j\neq i}^{i+m} p_j + c}{2m} - c \right).$$

A small constant, C, is added to pause duration values to guard against cases where all pause durations in the window are small and the mean of $\mathbf{W}$ is close to zero. We have used the value $C = 0.2$.

The parameter n controls detection sensitivity, increasing it will decrease the detection rate but also decrease the false alarm rate. The value of m controls the granularity of the resulting segments by setting a lower limit on the number of words a segment can have. A small value of m causes the video to be divided into a large number of small segments which may make the resulting video skim seem "choppy" and hard to follow. On the other hand, using too large a value for m will cause segments to be too long. We have used the values $n = 2$ and $m = 21$ in our experiments.

After the long pauses in the program are detected using the above procedure the video sequence is divided into segments using these pauses as boundaries. Each word detected by the ASR system in the program is then assigned to a segment based on the time of occurrence. In our system we use what is referred to as the "bag of words" approach where all sentence structure and word ordering is ignored.

5.7 Calculation of Segment Scores

5.7.1 Segment Scoring Using Speech Recognition Text

After the video is segmented using pause boundaries, as described in Section 5.6, we compute a score for each segment based on the words they contain. Not all the words belonging to a segment are used in deriving the segment score; some words, called *stopwords*, which have grammatical function but contribute little to the information content of the segment are ignored. Generally, stopwords include articles, pronouns, adjectives, adverbs, and prepositions. We have used a list of 371 stopwords in our system.

A score is calculated for each of the words that remain in a segment after the stopwords are eliminated. We use a scoring method that is related to a family of word

scoring schemes commonly used information retrieval, referred to as term frequency – inverse document frequency (tf.idf) methods. In these scoring schemes, the score of a word is calculated as a function of *term frequency*, $n_{i,w}$, the number of times a word has appeared in a video program, reflecting how salient the word is within the program and *document frequency*, the number of segments that a word appears in, indicating how semantically focused the word is. Based on these ideas, for each word w in segment Σ_i we compute the score, $s_{i,w}$, which measures the statistical importance of w using the formula [133]

$$s_{i,w} = \frac{(k+1)\, n_{i,w}}{k[(1-b) + b\frac{L_i}{AL}] + n_{i,w}} \log \frac{N}{n_w}, \tag{5.1}$$

where

$n_{i,w}$ - number of occurrences of word w in segment i

n_w - total number of occurrences of word w in the video sequence

L_i - number of words in segment i

AL - average number of words per segment in the video sequence

N - total number of segments in the video sequence

k, b - tuning parameters

The constant k determines the sensitivity of the first term to changes in the value of the term frequency, $n_{i,w}$. If $k = 0$ this term reduces to a function which is 1 if word w occurs in segment i and 0 otherwise; if k is large, the term becomes nearly linear in $n_{i,w}$. Since not all segments have the same number of words, the segment score has to be normalized by the number of words in the segment. A simple normalization would be to divide $n_{i,w}$ by L_i, which corresponds to the case of $b = 1$ in the above formula. This normalization is based on the assumption that if two segments are about the same topic but of different lengths, this is just because the longer is more wordy. Extensive experiments reported in [133] suggest the values $k = 2$ and $b = 0.75$. We have used these values in our experiments.

After the scores for individual words are computed using (5.1), the score for segment i is computed as the sum of the scores for the words it contains as

$$S_i = \sum_{w \in \Sigma_i} s_{i,w} \tag{5.2}$$

5.7.2 Detection of Dominant Word Co-Occurrences

One problem with using the words detected by a speech recognition system is that, the detection accuracy is low, especially for video programs containing rare words and phrases. In this study we used the IBM ViaVoice, large vocabulary continuous speech recognition (LVCSR) real-time system with the Broadcast News language model of 60,000 words [134]. The word error rate of LVCSR systems may vary substantially depending on the content, from 19% on prepared speech read by an anchor in a studio as reported by [134], to up to 35-65% for a wide variety of real-world, spontaneous speech data that may include combinations of speech with background music, noise, degraded acoustics, and non-native speakers. The difficulty is that speech recognition programs are generally trained using a large, well-balanced collection of content in which words and phrases like "animal kingdom," "breeding season," "compressed air," or "Moore's Law" are rare. However, such terms and proper names may play an important role in understanding a program and should be taken into account by the video summarization algorithm when selecting segments for the summary. Such groups of words are referred to as *collocations*, which are defined as sequences of two or more consecutive words that have characteristics of a syntactic or semantic unit [89]. Since we ignore the ordering of words in segments, we generalize the definition of a collocation and include pairs of words that are strongly associated together but do not necessarily occur in consecutive order. We refer to such pairs of words as *co-occurrences*. We measure the association of words by deriving a likelihood measure that measures how dependent the use of one word in a segment is on the presence on the other word.

Detecting co-occurrences in the speech transcript of a video program may be viewed as a hypothesis test: given the occurrence distribution of the words w_1 and w_2 in the segments detected in the program, the null hypothesis is that they occur independently and the alternate hypothesis is that the occurrences are dependent. The likelihood ratio for this hypothesis test may be written as

$$\lambda = \frac{\max_p P(w_1, w_2; p \mid H_0)}{\max_{p_1,p_2} P(w_1, w_2; p_1, p_2 \mid H_1)}, \tag{5.3}$$

where H_0 and H_1 are the null and alternate hypotheses, respectively, and p_1 and p_2 are the estimates of the occurrences of w_1 and w_2. The word counts that summarize the occurrence distribution of w_1 and w_2 are given by

$a = n_{w_1,w_2}$ is the number of segments where w_1 and w_2 both appear,

$b = n_{w_1,\neg w_2}$ is the number of segments where w_1 appears but not w_2,

$c = n_{\neg w_1,w_2}$ is the number of segments where w_2 appears but not w_1, and

$d = n_{\neg w_1,\neg w_2}$ is the number of segments where neither w_1 nor w_2 appear.

The maximum likelihood estimates for the probabilities are then given by $p = \frac{a+b}{N}$, $p_1 = \frac{a}{a+c}$, and $p_2 = \frac{b}{b+d}$. Substituting these values in (5.3) and assuming that the probability that a word will appear in a segment is modeled using a binomial distribution, the log-likelihood ratio becomes [135]

$$\begin{aligned} \log\lambda \;=\; & (a+b)\log(a+b) + (a+c)\log(a+c) + (b+d)\log(b+d) \\ & + (c+d)\log(c+d) - a\log a - b\log b - c\log c - d\log d \\ & - N\log N. \end{aligned} \tag{5.4}$$

Using the likelihood ratio in (5.4) to find significant word co-occurrences in speech transcripts has many advantages over other methods [89]. It is more accurate for sparse data than other methods and, since the quantity $-2\log\lambda$ is asymptotically χ^2 distributed, the critical value to reject the null hypothesis of independency for a given confidence level can easily be calculated.

In Table 5.1 we list the 20 co-occurring word pairs that have the highest $-2\log\lambda$ values for three documentary programs we have used in our experiments. *MarkTwain*

Table 5.1
Twenty word co-occurrences with the highest log-likelihood values for three documentary programs.

MarkTwain	$-2\log\lambda$	*SeaHorse*	$-2\log\lambda$	*BrooklynBridge*	$-2\log\lambda$
mark–twain	26.64	gets–pregnant	17.52	air–compressed	28.64
history–shakespeare	15.78	setting–term	15.72	new–york	18.86
forced–times	15.78	lives–tiny	15.72	favorite–institute	16.06
ball–put	15.78	change–color	15.72	cure–decided	16.06
got–talked	15.78	shallows–shore	15.72	favorite–polytechnic	16.06
twelve–miles	15.78	female–takes	13.14	polytechnic–institute	16.06
boat–brothers	15.78	animal–kingdom	13.14	greatest–existence	16.06
ambition–steamboat	15.78	unfold–wild	12.40	brooklyn–york	13.88
aside–spring	15.78	care–pregnancy	12.40	top–chamber	12.74
american–bowl	15.66	half–human	12.40	size–sons	12.74
people–understood	14.28	breeding–season	12.40	compressed–water	12.44
sam–clemens	13.60	watch–wild	12.40	civil–washington	12.00
father–financial	12.46	carefully–know	12.40	people–views	11.26
dollars–jones	12.46	carries–eggs	12.40	inside–required	11.26
long–want	12.46	cash–takes	12.40	city–half	11.26
go–literature	12.46	arrested–need	12.40	back–feet	10.22
look–night	12.46	arrested–think	12.40	feet–level	10.22
race–understood	12.46	females–males	11.74	existence–structure	10.22
nearby–summer	12.46	sea–horses	11.68	greatest–structure	10.22
friends–nearby	12.46	consider–long	10.90	structure–plans	10.22

is documentary detailing Mark Twain's biography, *SeaHorse* includes information about the life and mating habits of sea horses, and *BrooklynBridge* is about the building of the Brooklyn Bridge in New York City. From this table we observe that the word pairs give valuable information about the contents of the programs. For a confidence level of $\alpha = 0.05$, the critical value of the χ^2 distribution with one degree of freedom is 7.88. Therefore for all the word pairs listed in Table 5.1 the null hypothesis can be rejected and there is strong statistical association between the words in the pair.

In our system the log-likelihood ration in (5.4) is calculated for all possible word pairs in the speech transcript. Then 30 pairs with the highest $-2\log\lambda$ values are

chosen to be the significant co-occurrences for the program, which are used to update segment scores according to the equation

$$S'_i = S_i \, C^{L_i^{\text{sig}}} \tag{5.5}$$

where L_i^{sig} is the number of significant co-occurrences that segment i contains and C is a constant. We have used the value $C = 1.2$ in our system. This score increase is based on the assumption that the segments including more significant co-occurrences are more important.

5.8 Segment Selection for the Skim Using a Greedy Algorithm

After the segment scores are calculated using constituent non-stopwords and updated using the significant word co-occurrences detected in the transcript, they are used to select the segments that will form the video skim. The duration of the skim is specified by the user in terms of the *abstraction ratio*, f, which is defined as the ratio of the duration of the video skim to the duration of the source video. We then have

$$T_{\text{summary}} = f \, T_{\text{full}} \tag{5.6}$$

Since the score of a segment is proportional to its "semantic importance," our goal in generating the skim is to select those segments that maximize the cumulative score for the resulting summary while not exceeding T_{summary}. This may be viewed as an instance of the 0–1 knapsack problem (KP) which is defined mathematically as follows [136]

$$\begin{aligned} &\text{maximize} && \textstyle\sum_{i=1}^{N} S'_i \, x_i \\ &\text{subject to} && \textstyle\sum_{i=1}^{N} T_i \, x_i \leq T_{\text{summary}} \end{aligned} \tag{5.7}$$

where T_i is the duration of the segment Σ_i, N is the number of segments in the program, and the binary variable x_i is defined as

$$x_i = \begin{cases} 1 & \text{if segment } \Sigma_i \text{ is selected for the skim} \\ 0 & \text{otherwise} \end{cases}$$

Although efficient branch-and-bound algorithms exist that can solve this problem [136], in our system we have chosen to implement a greedy algorithm to solve the 0–1 KP due to its simplicity and fast execution speed. Although the solution found by the greedy algorithm will be suboptimal, the difference between the greedy solution and the optimal solution has been shown to be small in many cases [137].

We define the *efficiency* of segment Σ_i to be the ratio of its score to its duration, i.e., $e_i = S_i'/T_i$. Clearly, we would like to include segments with high efficiency in the summary. Therefore, the first step in solving the above constrained maximization problem is to sort the segments in a program according to their efficiencies to have a list with the property

$$e_i \geq e_j \quad \text{when} \quad i < j.$$

Assuming that the segments $\{\Sigma_1, \ldots, \Sigma_j\}$ are selected for the summary, the duration of the summary will be

$$\bar{T}_j = \sum_{i=0}^{j} T_i, \qquad j = 1, \ldots, N$$

The greedy selection process proceeds as follows: Starting from $j = 1$, segments are added to the summary as long as $T_j \leq T_{\text{summary}} - \bar{T}_{j-1}$, that is, as long as there is still time left in the summary to accommodate the next segment. The first item that cannot be included in the summary is denoted the *break item*, b. Thus the break item satisfies

$$\bar{T}_{b-1} \leq T_{\text{summary}} < \bar{T}_b.$$

Then, the greedy algorithm selects the segments $\{\Sigma_1, \ldots, \Sigma_{b-1}\}$ for the summary. Note that this solution leaves an unused capacity $T_{\text{summary}} - \bar{T}_{b-1}$. In order to include as many segments in the summary as possible, we have modified the basic greedy algorithm so that once it encounters the break item, it continues to search the list of segments to fins a segment that will fit in the unused time. The selected segments are then concatenated to generate the video skim.

5.9 Increasing Summary Coverage Using a Dispersion Measure

The summary segment selection algorithm described in Section 5.8 generates summaries that tend to be focused on a few important points in the segments. Such summaries are detailed but have low overall program coverage. When the goal of video summarization is to preserve as much information about the original program as possible in the video summary, the advantage of this algorithm of over mechanical summarization procedures, such as randomly selecting the segments to be included in the summary, may be limited. This is the coverage–detail dilemma of summarization discussed the beginning of this chapter, i.e., it is impossible to maximize both the coverage and detail of the summary of a program.

One way to alleviate this problem is to add another term to the maximization problem in (5.7) that is a function of the coverage of a summary and maximize both the cumulative score and coverage for the generated summary. In order to formalize the notion of coverage, we introduce a measure of *dispersion*, $d(\mathcal{S})$, for a summary, $\mathcal{S}$. We want the quantity $d(\mathcal{S})$ to be small when $\mathcal{S}$ contains segments that are close together in the full program, and to be large when they are distributed uniformly across the full program.

Let the original video be divided into $P = \lfloor 1/f \rfloor$ temporal intervals. Given a set of segments, $\mathcal{S} = \{\Sigma_1, \ldots, \Sigma_N\}$, we estimate their temporal distribution over the original program by counting the number of segments contained in each temporal piece,

$$h(p) = \sum_{\Sigma_i \in \mathcal{S}} 1_{\frac{(p-1)}{P} T_f \leq b(\Sigma_i) < \frac{p}{P} T_f}, \quad p = 1, \ldots, P$$

where $b(\Sigma_i)$ is the start time of segment Σ_i. Then, we define the dispersion of $\mathcal{S}$ to be equal to its entropy, that is,

$$d(\mathcal{S}) = -\sum_{p=1}^{P} h(p) \log h(p). \tag{5.8}$$

Several other measures of dispersion of elements of a set can be defined, such as the Gini or the misclassification measures, which are commonly used to measure the

impurity of a node in decision tree growing [53]. The particular dispersion function selected is not vital to the algorithm and any of the other choices can also be used.

At each iteration k of the selection algorithm, we want to add the segment that leads to the greatest increase in the dispersion value of the current summary, $\mathcal{S}^k$. To this end, we define the increase in dispersion caused by adding the segment Σ_i to $\mathcal{S}^k$ as

$$\Delta d_i^k = d\left(\mathcal{S}^k \cup \{\Sigma_i\}\right) - d(\mathcal{S}^k). \tag{5.9}$$

Then, the dispersion score of Σ_i at iteration k is defined as

$$d_i^k = e^{C\Delta d_i^k}, \tag{5.10}$$

where C is a constant used to accentuate the difference in dispersion between segments. The algorithm is not sensitive to the value of C, we have used the value $C = 20$ in our experiments.

In order to be able to compare the word-based score for a segment and its dispersion score, at each iteration we normalize each of these scores by the score of the segment that has the highest value for that score. We then employ a greedy algorithm, selecting the segment Σ^* that has the highest combined score using

$$\Sigma^* = \arg\max_i \left(w_{\text{word}}\, \tilde{S}_i' + w_{\text{disp}}\, \tilde{d}_i^k \right), \tag{5.11}$$

where $\tilde{S}'$ and $\tilde{d}_i^k$ are the normalized word-based and dispersion scores, respectively, and w_{word} and w_{disp}, both in the interval $[0, 1]$ and satisfying $w_{\text{word}} + w_{\text{disp}} = 1$, are the weights for the two scores. By changing the values of these weights one can change the relative importance of detail versus coverage for the generated summaries. Our experiments suggest that the values $w_{\text{word}} = 0.7$ and $w_{\text{disp}} = 0.3$ constitutes a good trade-off between detail and coverage.

5.10 Evaluation of Video Summaries

Since automatic video summarization is still an emerging field, serious questions remain concerning the appropriate methodology in evaluating the quality of the

generated summaries. Most video summarization studies do not include any form of quantitative summary evaluations. Evaluation of the quality of automatically generated video summaries is a complicated task because it is difficult to derive objective quantitative measures for summary quality. In order to be able to measure the effectiveness of a video summarization algorithm one first needs to define features that characterize a good summary, given the specific application domain being studied. As discussed in Section 5.1, summaries for different applications will have different sets of desirable attributes. Hence, the criteria to judge summary quality will be different for different application domains.

Automated text summarization dates back at least to Luhn's work at IBM in the 1950s [138], which makes it the most mature area of media summarization. We will apply the terminology developed for text summary evaluation to evaluation of video summaries. Methods for the evaluation of summaries can be broadly classified into two categories: *intrinsic* and *extrinsic* evaluation methods [139, 140]. In intrinsic evaluation methods, the quality of the generated summaries is judged directly based on the analysis of summary. The criteria used may be user judgment of fluency of the summary, coverage of key ideas of the source material, or similarity to an "ideal" summary prepared by humans. On the other hand, in extrinsic methods the summary is evaluated with respect to its impact on the performance for a specific information retrieval task.

For event-based content, e.g., a sports program, where interesting events are unambiguous, summaries might be judged on their coverage of these events in the source video. Two such evaluations are given in [123] and [117]. Ekin and Tekalp [123] give precision and recall values for goal, referee, and penalty box detection, which are important events in soccer games. In Ferman and Tekalp's study [117], the video summary was examined to determine, for each shot, the number of redundant or missing keyframes in the summary. For example, if the observer thought that an important object in a shot was important but no keyframe contained that object, this resulted in a missing keyframe. Although they serve as a form of quantitative

summary quality measure, the event detection precision and recall values given in these studies do not reflect the quality of the summaries from the user's point of view.

For uniformly informative content, where events may be harder to identify, different evaluation techniques have been proposed. He *et al.* [101] determines the coverage of summaries of key ideas from presentation videos by giving users a multiple choice quiz derived from the full program video before and after watching a video skim extracted from it. The quizzes consist of questions prepared by the presentation speakers which are assumed to reflect the key ideas of the presentation. The quality of the video skims were judged by the increase in quiz scores. A similar technique was used by Taskiran *et al.* [98] in evaluating video skims extracted from documentaries. The quiz method has some drawbacks: First, it was found that it may have difficulty differentiating between different summarization algorithms depending on program content [98,101]. Second, it is not clear how quiz questions can be prepared in an objective manner, except, perhaps, by authors of presentations who are not usually available. Finally, the concept of a "key idea" in a video program is ambiguous and may depend on the particular viewer watching the skim. An interesting extrinsic evaluation method was used by Christel *et al.* [102]. In this study video skims extracted from documentaries were judged based on the performance of users on two tasks: fact-finding, where users used the video skims to locate video segments that answered specific questions; and gisting, where users matched video skims with representative text phrases and frames extracted from source video.

In intrinsic evaluation of text summaries generally summaries created by experts are used [139]. Using a similar approach would be much more costly and time consuming for video sequences. Another scheme for evaluation may be to present the segments from the original program to a large number of viewers and let them select the segments they think should be included in the summary, thereby generating a ground truth. This seems to be a promising approach although agreement among human subjects becomes an issue for this scheme. A commonly used intrinsic eval-

uation method is to have users rate the skims based on subjective questions, e.g. "Was the summary useful?" and "Was the summary coherent?" Such surveys were used in [100–102].

5.11 Experiments and Results

The experiments we have performed to detect the differences in the quality of summaries generated by different algorithms are described in detail in the following sections. We conducted experiments based on both intrinsic and extrinsic evaluation methods, which were described in Section 5.10. The summary quality measure that we used was the ratio of information contained about the full program in the video summary, i.e., the more information a summary contains about the full program, the higher its quality was deemed to be. Given this summary quality criterion, a natural method to test the effectiveness of video summarization algorithms is the *question and answer method*, where viewers answer questions derived from the full program just by watching the summary and the number of correctly answered questions is accepted as a quantitative measure of summary quality. We used this task as the extrinsic summary evaluation method.

As the intrinsic evaluation scheme, we tested how many of the key points of the full program were covered by the summaries by determining how many of the questions extracted from the full program is in a summary. We also asked the subjects who participated in our experiment their assessment of the summaries they watched.

5.11.1 Sequences and Algorithms Used in the Experiments

We chose three documentary programs to use in our study. Since the full documentaries were too long to be used in our experiment, we used approximately the first 20 minutes of each program as the full-length programs from which summaries were extracted. We selected program content to minimize the chance that the subjects

Table 5.2
Information about the full-length documentary programs used in the experiment.

Sequence name	**length (min.)**	**Content description**
Seahorse	22.10	Life and mating habits of seahorses [141]
MarkTwain	21.19	Detailed biography of Mark Twain [142]
WhyDogsSmile	22.53	Mother-child bond and other emotions in animals [143]

participating in the study have prior knowledge about the programs. Information about the three documentary programs used in our experiments is given in Table 5.2. All sequences were processed using the CueVideo system to obtain shot boundaries and speech transcripts.

Three algorithms, abbreviated as FREQ, RAND, and DEFT, were used on the full programs to generate three different skims for each full program. We denote the summary of the Seahorse program using the RAND algorithm by Seahorse+RAND, and similarly for the other summaries. The algorithm FREQ is the summary generation algorithm based on word-frequency and dispersion scores derived from program segments, as described in Section 5.5. The RAND algorithm detects the segments in the program, using the same pause detection algorithm as was used for FREQ, but does not calculate any segment scores and randomly selects the segments to be included in the summary. The DEFT, or default, algorithm is the simplest video summarization algorithm possible, consisting of subsampling the video program temporally at fixed intervals. The DEFT algorithm divides the full program into $\lfloor T_{\text{summary}}/T_s \rfloor$ temporal intervals, and selects the first T_s seconds of each interval for the summary. Our initial experiments suggest a value of $T_s = 5$ seconds to be the minimum value that viewers feel comfortable with; values smaller than this lead to "choppy" summaries that are hard to understand. It is clear that the DEFT algorithm will not generate high quality summaries. It is included in the study to act a baseline against which the performances of the other two algorithms will be compared.

Table 5.3
The ordering of the three video skims watched by subjects in each group in the experiment.

Group	**first summary**	**second summary**	**third summary**
Group 1	MarkTwain+FREQ	Seahorse+RAND	WhyDogsSmile+DEFT
Group 2	WhyDogsSmile+RAND	MarkTwain+DEFT	Seahorse+FREQ
Group 3	Seahorse+DEFT	WhyDogsSmile+FREQ	MarkTwain+RAND

A factor that can influence evaluation results is the value of the summarization ratio used to obtain the video skims. The evaluation results of the same summarization system can be significantly different when it is used to generate summaries of different lengths [139]. In our experiments we have kept the summarization ratio constant at $f = 0.1$, that is, the skims generated are approximately one-tenth the duration of the original programs. This value was also used in other video summarization studies and represents a reasonable amount of compaction for practical applications.

5.11.2 Experimental Design

The 48 subjects participating in the experiment were randomly divided into three groups of 16. Each subject watched three video skims. In order to minimize learning and other unforeseen cross-over effects, both the order of the content and the algorithms were different for the three groups, as shown in Table 5.3. Therefore, each subject watched skims generated from all programs by all three algorithms. Subjects were Purdue University students and employees. Each subject received $10 for participation in the experiment. Subjects participated in the experiment individually. Each subject used a personal computer on which the graphical user interface for the experiment was run. After watching each video skim, the subjects first answered three multiple questions based on their assessment of the quality of the summary they just watched. Then, they answered ten multiple choice questions derived from

the full program and containing important facts about the content presented in the skim. Each question offered four answer choices. Only one choice was correct for each question. The questions used in the experiment are listed in the Appendix. While watching the summaries the users were not able to pause the video, jump to a specific point in the video, or take notes. No time limit was imposed for completing the experiment.

The questions that were presented to the subjects were determined by two judges. One judge was the first author and the other judge was naive about the algorithms used. Each judge independently marked the parts of the closed-caption transcripts of the programs that they deemed were the important points of the programs without watching any of the summaries [1]. The intersection of these two lists of marked locations were used to generate the questions with the constraint that the questions be as uniformly distributed over the full program as possible. Ten questions were generated for each program, together with one correct and three incorrect multiple choice answers. The questions were kept simple to ensure that if the answer to a question appears in a summary than a viewer who watches that summary can easily answer the question correctly.

5.11.3 Results for the Questions and Answer Task

Statistics for the performance of the subjects in the question and answer task are shown in Table 5.4 and represented graphically in Figure 5.3. The number of correct answers out of 10 questions that the subjects scored after watching the summaries generated by the same algorithm were summed for the three experimental groups to obtain these total results. In Figure 5.3 the error bars represent an interval of $\hat{\mu} \pm \hat{\sigma}_M$, where $\hat{\mu}$ is the estimated mean number of correct answers for the subjects, the quantity $\hat{\sigma}_M = \hat{\sigma}/\sqrt{n}$ is the standard error for the mean, which gives a measure

[1] Source transcripts are available from the author.

Table 5.4
Statistics for the number of correct answers for 10 questions and $n = 48$ subjects for the question and answer task. Scores were aggregated for the three experimental groups.

	FREQ	RAND	DEFT
estimated mean, $\hat{\mu}$	6.542	5.313	4.604
estimated standard deviation, $\hat{\sigma}$	1.650	1.652	1.865
estimated standard error, $\hat{\sigma}_M = \hat{\sigma}/\sqrt{n}$	0.238	0.238	0.269

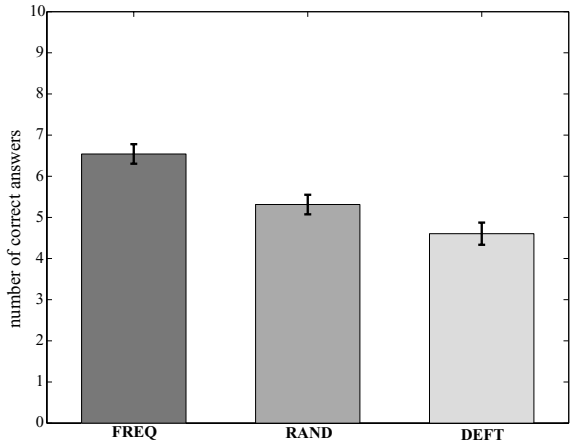


Fig. 5.3. Graphical representation of the data in Table 5.4.

how much sampling fluctuation the mean will show, $\hat{\sigma}$ is the estimated standard deviation of subject scores, and $n = 48$ is the number of subjects.

From these results we can see that the FREQ algorithm performed better than RAND and DEFT. The performance of RAND and DEFT were comparable, with RAND outperforming DEFT slightly. This result was expected since neither of these algorithms took program content into account while generating the summary. In order to assess the statistical significance of the results in Table 5.4, we list pairwise comparison p-values in Table 5.5 obtained using a two-sided t-test. The t-values were calculated with the assumption of equal variances for the all populations. Table 5.4 lists p-values for each of the three pairwise comparisons. The p-value refers to the probability of incorrectly rejecting the hypothesis that two populations

Table 5.5
Pairwise p-values using the two-sided t-test to test the statistical significance of the difference in scores between algorithms.

	FREQ	RAND
RAND	4.348×10^{-4}	
DEFT	5.189×10^{-7}	5.183×10^{-2}

have identical mean values. From Table 5.5 we see that one can safely conclude that FREQ and DEFT represent different populations, i.e., these two algorithms produce summaries of different quality, since the probability that this conclusion is wrong is less than 10^{-6}. The true p-value is larger, however. The reason is that Table 5.4 shows the results of testing three hypotheses, not one, and the probability of incorrectly rejecting at least one hypothesis out of three is higher than the probability of rejecting one out of one. In particular, if one takes $p = 0.05$ as the cut-off value below which the difference between two populations is considered significant, then each pair of hypotheses in Table 5.4 should be evaluated by using a cut-off value equal to $p = 0.05/3 = 0.0167$. We see from the Table 5.5 that using this criterion the differences in scores between FREQ and RAND, and FREQ and DEFT are statistically significant, while the difference between RAND and DEFT is not statistically significant.

In order to check the experimental assumption that subjects had little prior knowledge about the topics in the programs, we asked them to rate their prior knowledge of the program content after watching each summary. The statistics for the responses are listed in Table 5.6. These figures agree well with the experimental assumption that the subjects had little prior knowledge about the program content that they were tested on.

In our previous study with a smaller group of subjects using the question and answer method [98], we found that the results for this evaluation method were correlated with the particular documentary program used to extract the summaries. In

Table 5.6
Results for self-assessment of subjects' prior knowledge on program content. Subjects rated on a scale of 1 (not familiar at all) to 5 (very familiar).

	Seahorse	MarkTwain	DogSmile
estimated mean response, $\hat{\mu}$	1.813	1.583	1.792
estimated standard error, $\hat{\sigma}_M = \hat{\sigma}/\sqrt{n}$	0.151	0.145	0.130

order to investigate the effect of program content on the performance of the algorithms, the mean correct scores that the subjects obtained from the multiple choice questions for the three documentary programs are compared in Table 5.7. The graphical representation of the means and standard errors for the data in Table 5.7 is given in Figure 5.4. From this figure we see that the performance of the subjects who watched the summaries produced by the FREQ algorithm are consistently larger than the scores for subjects who watch the summaries from the other two algorithms, implying that our algorithm produced more informative summaries. Specifically, assuming the value $p = 0.05$ as the cut-off value for deciding statistical significance, and correcting for the fact that we are performing nine hypotheses tests, statistically significant differences were found between FREQ–RAND and FREQ–DEFT for MarkTwain, and between all three algorithms for DogSmile.

The theoretical maximum scores, c_{the}, that the subjects can obtain for the three programs are shown as horizontal lines on the bars in Figure 5.4. These values were calculated using the relation

$$c_{\text{the}} = c_{\text{inc}} + 0.25(10 - c_{\text{inc}}),$$

where c_{inc} is the number of answers included in the summary of each program, which are listed in Table 5.9, and 0.25 is the probability of getting an answer correct by guessing. This simple model assumes that the subjects have perfect memory and full attention, therefore always correctly answering the questions that were covered by the summary. As seen in Figure 5.4 there are systematic differences between the

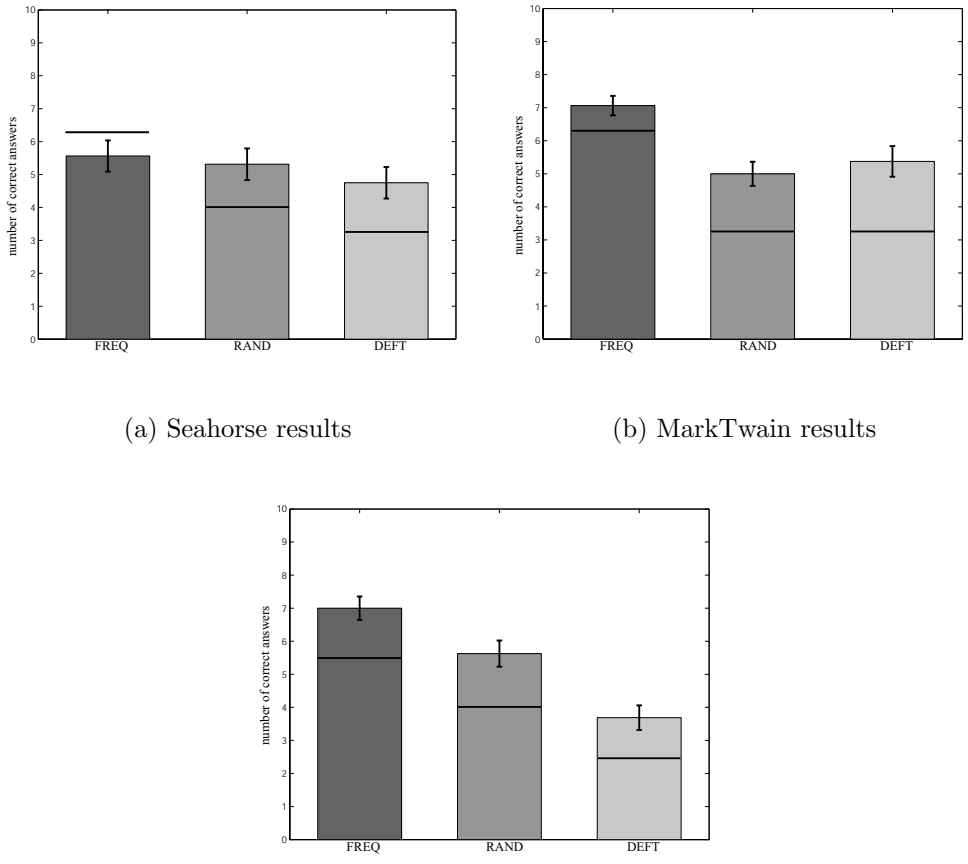
Table 5.7
Statistics for the number of correct answers for 10 questions and $n = 16$ subjects in each group for each program.

	Seahorse			**MarkTwain**			**WhyDogsSmile**		
	FREQ	RAND	DEFT	FREQ	RAND	DEFT	FREQ	RAND	DEFT
$\hat{\mu}$	5.563	5.313	4.750	7.063	5.000	5.375	7.000	5.625	3.688
$\hat{\sigma}$	1.896	1.922	1.915	1.181	1.461	1.857	1.414	1.586	1.493
$\hat{\sigma}_M$	0.474	0.481	0.479	0.295	0.365	0.464	0.354	0.397	0.373

maximum score predicted and the mean score of the subjects, which are caused by the simplicity of the above model. These differences are mainly due to two factors: First, while we tried to minimize previous knowledge about program content in the experiment, it is impossible to totally eliminate it. For some questions, subjects have used their previous knowledge to determine the correct answer although it was not in the summary. Second, although the answer to a particular question may not be included in a summary, it is still possible for subjects to use visual cues and inference to "guesstimate" the correct answer.

We should note that our question preparation strategy of distributing the questions as uniformly as possible over the total duration of the original program introduced a bias towards algorithms that select segment randomly over the full program, i.e., towards RAND and DEFT. This means that the performance of the subjects actually overestimates the quality of these algorithms.

Finally, we noticed some systematic differences in scores between subjects who were native speakers of English and those who were not. Statistics for the scores, separated according to native speaker status, are listed in Table 5.8 Although not found to be statistically significant, these results nevertheless suggest that there may be differences in performance for native and non-native speakers for the questions and answer task. This factor should be taken into account when designing video summarization evaluations using this task. From the table we can see that the



(a) Seahorse results

(b) MarkTwain results

(c) DogSmile results

Fig. 5.4. Graphical representation of the subject performance data in Table 5.7. The vertical lines represent the theoretical maximum scores.

largest difference between the scores occurs for the DEFT algorithm. Due to the poor quality of the video summaries produced by DEFT, for the viewers, information gathering from the summaries is the most challenging for this algorithm. The language understanding capabilities of viewers need to be used to the maximum for DEFT summaries, which accounts for the relatively large difference between native and non-native speaker performances for this algorithm.

Table 5.8
Statistics for the number of correct answers grouped according to whether a subject was native speaker of English (NS) or not (NNS).

	Seahorse		**MarkTwain**		**WhyDogsSmile**	
	NS	NNS	NS	NNS	NS	NNS
estimated mean, $\hat{\mu}$	6.778	6.400	5.611	5.133	5.056	4.333
estimated standard deviation, $\hat{\sigma}$	2.074	1.354	1.685	1.634	1.893	1.826
estimated standard error, $\hat{\sigma}_M = \hat{\sigma}/\sqrt{n}$	0.489	0.247	0.397	0.298	0.446	0.333

Table 5.9
The number of important items covered by each summary, out of 10 items extracted from the full programs.

	FREQ	RAND	DEFT
Seahorse	5	2	1
MarkTwain	5	1	1
DogSmile	4	2	0

5.11.4 Results for the Intrinsic Evaluation

We examined each of the nine summaries for each documentary–algorithm pair and determined how many answers each summary contains. These numbers are tabulated in Table 5.9. As can be seen from this table, the information covered by summaries generated using FREQ contain significantly more information about the full programs compared to the other two algorithms. Since the summarization ratio was $f = 0.1$ we would expect the random summaries to contain one answer on the average. The results for RAND and FREQ in Table 5.9 agree well with this prediction.

As the second intrinsic evaluation scheme, after watching each summary we asked the subjects to answer two assessment questions. The questions were "I found the summary to be clear and easy to understand" and "I feel that I can skip watching the whole program because I watched this summary." Subjects rated both of these

Table 5.10
Results for summary assessment questions aggregated over three sequences for $n = 48$ subjects. The questions were (I) "I found the summary to be clear and easy to understand" and (II) "I feel that I can skip watching the whole program because I watched this summary." Subjects rated on a scale of 1 (strongly disagree) to 5 (strongly agree).

		FREQ	RAND	DEFT
Question I	estimated mean response, $\hat{\mu}$	3.146	2.979	1.896
	estimated standard deviation, $\hat{\sigma}$	1.052	1.329	0.928
	estimated standard error, $\hat{\sigma}_M = \hat{\sigma}/\sqrt{n}$	0.152	0.192	0.134
Question II	estimated mean response, $\hat{\mu}$	2.292	2.313	1.500
	estimated standard deviation, $\hat{\sigma}$	1.071	1.151	0.825
	estimated standard error, $\hat{\sigma}_M = \hat{\sigma}/\sqrt{n}$	0.155	0.166	0.119

statements on a scale of 1 to 5, 1 being strongly disagree and 5 being strongly agree. The statistics of the subjects responses for the three algorithms, aggregated over the three programs, are shown in Table 5.10. For both of these questions the differences between FREQ and DEFT, and RAND and DEFT are statistically significant while the differences between FREQ and RAND are not statistically significant, using a two-tailed t-test at the level $p = 0.05/3 = 0.0167$, taking into account the correction described in Section 5.11.3. Although the informativeness of the summaries generated by DEFT was comparable to those generated by RAND, as evidenced by the results in Table 5.9, there is a large difference between these algorithms in terms of viewer summary assessment. This result implies that for the viewers one of the most important criteria in judging video summaries is that the segments selected for summaries not start in mid-sentence.

5.12 Conclusions

In this paper we proposed an algorithm to automatically summarize video programs. We use concepts from text summarization, applied to transcripts obtained using automatic speech recognition. The log-likelihood ratio criterion was used for detecting significant co-occurring words as a powerful tool in identifying important topics in video programs. A measure of summary dispersion over the full program was derived and was shown to be effective in managing the trade-off between detail and coverage of the generated video summaries.

In the second part of this work we developed an experimental design and a user study to judge the quality of the generated video summaries. We used both extrinsic and intrinsic schemes to evaluate the summaries generated using a summarization factor of 0.1 by three algorithms, our algorithm and two random algorithms. For extrinsic evaluation, we used a question answering strategy, where the questions are derived from the full programs while the subjects who participated in our experiment watched only the summaries generated from those programs before answering the questions. A quantitative measure of the informativeness of the generated summaries, and therefore the effectiveness of the summarization algorithms, was taken to be the number of correct answers given by the subjects. For the intrinsic evaluation of the generated summaries we counted the number of important points about the programs each summary contains. We also asked the subjects to assess the quality of the summaries.

For the extrinsic evaluation user experiment, the mean number of correct answers out of 10 questions for 48 subjects were 6.542, 5.313, and 4.604 for our algorithm and the two random algorithms, respectively. The differences in these mean scores were found to be statistically significant, implying that our algorithm produced more informative summaries.

For the intrinsic evaluation scheme, the number of important items that were contained in the summaries produced by our algorithm were significantly higher than

those produced by the random algorithms, which had performance at chance level, as expected. Although slightly better user assessment scores were obtained for our algorithm, the difference was not statistically significant between our algorithm and the random algorithm that takes pauses in audio into consideration while segmenting the full program. The assessment results imply that viewers have a strong preference for summaries where summary segments start and end on sentence boundaries. They find it annoying and confusing when segments in the summary start in mid-sentence.

6. DETECTION OF UNIQUE PEOPLE IN NEWS PROGRAMS

6.1 Introduction

In this chapter we address the problem of detecting unique people appearing in television news programs. For each person in the program, our goal is to determine the temporal regions where that person appears in the program. We will be interested specifically in news programs from the C-SPAN networks, although the techniques developed are applicable to any other new programs with similar structure. C-SPAN is a private, non-profit company in the United States that broadcasts meetings, sessions for the U.S. Senate and House sessions, interviews, and other news programming, without commercials as a public service. C-SPAN programs usually use 1-2 cameras and view-points and there is not much camera or object motion. For C-SPAN programs a change of speaker is generally marked with a shot boundary so each shot usually contains a single person of interest. In this case the detection of unique people may be achieved by grouping all shots containing the same speaker. The C-SPAN Archives was established to record, index, and archive all C-SPAN programming. As of January 2002, the Archives contained 167,267 hours of C-SPAN programs. Although all programs in the Archives are hand-indexed using a large number of descriptors, the indexing system currently used cannot generate shot-based person labels, which would be too time consuming. However, such labels are highly desirable since they would enable searches with higher granularity based on video shots rather than complete programs. These labels can also be used to automatically generate on-screen speaker identification graphics and chapters for the DVD versions of C-SPAN programs [144].

The unique person detection (UPD) system was developed for the C-SPAN Archives to be part of a larger Automatic Video Graphic Insertion (AVGI) system. The AVGI system will be used by the C-SPAN Archives to automate the insertion of on-screen graphics in programs to a large extent by processing unkeyed video programs. A simplified block diagram of the AVGI system is shown in Figure 6.1. The AVGI system consists of three subsystems, two of them are completely automated while the other subsystem needs a human operator, which are briefly described below

- The **Unique Person Detection subsystem** detects the shot boundaries in the programs and generates a list of images of all unique people appearing in the program, together with the time codes for the shots they appear in.

- The **Speaker Labeling subsystem** presents the list generated above to a human operator using a graphical user interface. The operator then associates the image of each person in the program with the corresponding C-SPAN database entry for that person.

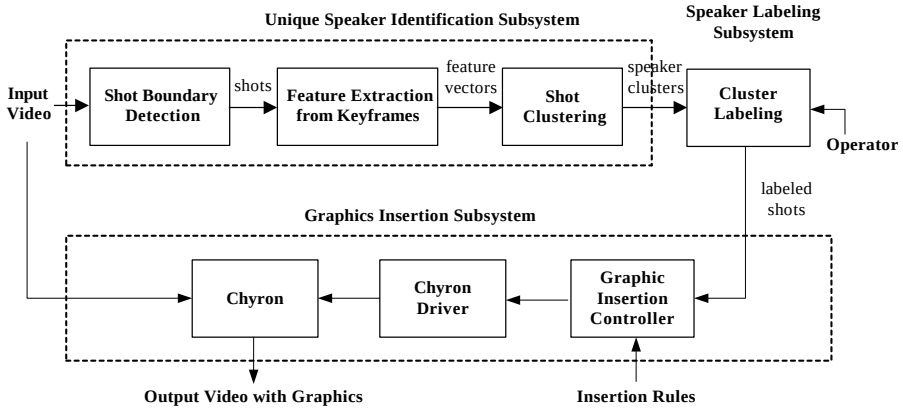
- The **Graphics Insertion subsystem** processes the list of images for unique people and their identifying information and inserts on-screen graphics into the program using the Chyron graphics generator. This subsystem uses heuristic rules to determine when to put up graphics, how long to keep them up, and what types of graphics to insert.

The AVGI system is currently operational and is being adopted by the C-SPAN Archives to be used in video indexing.

Besides the indexing application mentioned above, there are many other applications of in content-based multimedia analysis where the UPD task is important. For example, the accuracy of speech recognition may be improved by performing speaker adaption over each group of shots [145].

In this chapter we describe our approach to the UPD problem that uses a combination of visual and audio features to cluster shots belonging to same person. The part of the ACGI system shown in Figure 6.1 that will be discussed in this chapter is

Fig. 6.1. Block diagram of the proposed automatic graphics insertion system.



marked as the "Unique Speaker Identification Subsystem" in that figure. The output of our UPD system will be a list of clusters of shots, one for person appearing in the program. We assume no prior knowledge about the number of persons appearing in sequences. We illustrate our technique on a collection of video content obtained from the C-SPAN Video Archives.

The chapter is organized as follows: After discussing some previous approaches to the problem in Section 6.2, we describe the features extracted from each shot and how pairwise shot distances are calculated based on these distances in Section 6.3. Section 6.4 explains the distance normalization procedure we use to convert distance values to confidence values. Our hierarchical clustering procedure is explained in Section 6.5. The detection results are presented in Section 6.6 and Section 6.7 presents conclusions.

6.2 Previous Approaches to Unique Person Detection

Many systems were proposed to perform the person detection problem using only the audio information [146, 147]. An example of video indexing using multimodal features is found in [148] where audio transcripts, video captions, and face detection was used to associate faces and names in videos.

6.3 Shot Feature Extraction and Distance Calculation

The first step in our unique person detection system is shot boundary detection, as shown in Figure 6.1. However, for the results reported in this chapter we have used shot boundary ground truths marked by a human operator, rather than locations automatically determined by our algorithm described in Chapter 2. When shot transition locations determined by an algorithm is used there will be some errors in these locations, which will lead to errors in the shot clustering system. Since we wanted to focus on the performance of shot clustering and examine the errors due to clustering only, we have used shot transition location ground truths.

After the video sequence is divided into shots we extract a number of features from each shot for the person detection task. We use color histograms of face regions, color histograms of shot keyframes, audio, and shot adjacency as our features. After these features are extracted from each shot, we calculate pairwise shot distances as described below.

6.3.1 Shot Distance Based on Face Region Histograms

Faces provide powerful cues in comparing different speakers. We detect faces in shots and use the histogram of the face regions as one of our features. The face detection was performed by Professor Alberto Albiol Colomer from the Communications Department of the Technical University of Valencia. For face detection we examine every 10^{th} frame, which reduces the computational load without decreasing detection accuracy for most video programs. In this work we have used the face detector described in [149], which is based on a boosted cascade of simple classifiers and yields a highs detection rate. The result of face detection is a list of face regions that are represented by their bounding boxes. The pixels within each face box are converted from the original RGB space to normalized RGB values using the formulas

$$R_n = \frac{R}{\Sigma}, G_n = \frac{G}{\Sigma}, B_n = \frac{B}{\Sigma}, \tag{6.1}$$

where $\Sigma = R+G+B$. Since one of the normalized components is redundant, we use only the R_n and G_n color components. This color normalization step is important to reduce changes in pixel values caused by the automatic gain control available in many video cameras.

After the bounding boxes for faces are detected, we derive a normalized histogram for the R_n and G_n color components of the pixels within the face region. We then obtain average face histograms, $HF^i_{R_n}$ and $HF^i_{G_n}$, for shot S_i by averaging the face histograms for the frames in S_i. The face histogram distance between two shots, S_i and S_j, both containing faces, is calculated using

$$d_f(i,j) = \frac{1}{2}\sum_{k=1}^{BF} |HF^i_{R_n}(k) - HF^j_{R_n}(k)| + \frac{1}{2}\sum_{k=1}^{BF} |HF^i_{G_n}(k) - HF^j_{G_n}(k)|, \qquad (6.2)$$

where BF is the number of bins for face histograms. Currently we use the value $BF = 256$, which produced good results for our data set.

6.3.2 Shot Distance Based on Keyframe Histograms

There are cases where face detection fails to detect a person appearing in a shot or the detected face region is too small to provide reliable comparison between different persons. In order to handle these cases we use an additional color histogram feature, which is derived from the keyframe of each shot. For simplicity, we have selected the middle frame of each shot as the keyframe. We calculate the normalized histograms for the R_n and G_n color components for each keyframe. When calculating the color histograms, an area that is approximately the bottom one-third of the keyframe is ignored. In C-SPAN programs, as well as most other news programs, this area contains on-screen graphics, with similar color content for different people, illustrated in Figure 6.2. In order to make the keyframe histogram feature more sensitive to differences between shots containing different people, we do not include this region in the histogram. For each shot S_i we obtain the keyframe histograms $HK^i_{R_n}$ and

Fig. 6.2. A typical frame extracted from a C-SPAN program. The lower part or the frame containing on-screen graphics is ignored when calculating the keyframe histogram feature for the frame.

$HK^i_{G_n}$. The keyframe histogram distance between shots is computed similar to the face histogram distance using

$$d_k(i,j) = \frac{1}{2}\sum_{k=1}^{BK} |HK^i_{R_n}(k) - HK^j_{R_n}(k)| + \frac{1}{2}\sum_{k=1}^{BK} |HK^i_{G_n}(k) - HK^j_{G_n}(k)|, \qquad (6.3)$$

where BK is the number of bins for keyframe histograms. Currently we use the value $BK = 128$ that provided a good compromise in our tests. However, the performance of the system is not very sensitive to the particular choice of BK and a smaller number of bins can be used.

6.3.3 Shot Distance Based on Audio Features

For each shot we extract acoustic feature vectors consisting of the 20 mel-frequency cepstral coefficients obtained every 10 ms using a 20 ms Hamming window. The extraction of these acoustic feature vectors was done by Professor Alberto Albiol Colomer from the Communications Department of the Technical University of Valencia.

Let $\mathbf{x}^i = \{x^i(k) \in \Re^n, k = 1, \ldots, N_i\}$ and $\mathbf{x}^j = \{x^j(k) \in \Re^n, k = 1, \ldots, N_j\}$ be acoustic vectors extracted from shots S_i and S_j, respectively, where N_i denotes the number of 10ms segments for shot S_i and n is the dimensionality of the acoustic

vectors, in our case $n = 20$. We define $\mathbf{x}^{i,j} = \{\mathbf{x}^i \cup \mathbf{x}^j\}$ to be the combined set of acoustic vectors for shots S_i and S_j.

The extracted acoustic vectors are modeled using multivariate Gaussian distributions. Three different models $\mathcal{M}_i = \{\mu_i, \Sigma_i\}$, $\mathcal{M}_j = \{\mu_j, \Sigma_j\}$, $\mathcal{M}_{i,j} = \{\mu_{i,j}, \Sigma_{i,j}\}$ for $\mathbf{x}^i$, $\mathbf{x}^j$ and $\mathbf{x}^{i,j}$ are built, where μ_i and Σ_i are the mean vector and covariance matrix for the vectors in shot S_i, respectively. The dissimilarity between two audio segments corresponding to two shots is based on the comparison of their statistical models using the Bayesian Information Criterion (BIC) [150]. The BIC, which has been introduced in the statistics literature for model selection, is a likelihood criterion penalized by model complexity, which is represented by the number of model parameters. The key idea behind the BIC is that, although a higher likelihood can be obtained using separate models for audio segments i, j, this is done at the expense of a greater complexity, i.e., more parameters, that reduces the BIC value.

We use a similar approach to [150] and define the audio distance between two shots using the BIC as follows

$$\begin{aligned} d_a(i,j) &= 0.5\,(n + 0.5n(n+1))\,\log(N_i + N_j) \\ &\quad + N_i|\Sigma_i| + N_j|\Sigma_j| - (N_i + N_j)|\Sigma_{i,j}|. \end{aligned} \tag{6.4}$$

One important restriction of this distance is that enough data must be available to model the acoustic vectors. Based on the results given in [150], we calculate the audio distance, $d_a(i,j)$, only when both the audio segments corresponding to shots S_i and S_j are longer than 2 seconds.

6.4 Feature Normalization

For each pair of shots in a given sequence, we use the distance values described in Section 6.3 to derive a single confidence value for each pair of shots, which is a continuous value in $[0, 1]$, where 0 indicates that the two shots contain the same speaker with high confidence, and 1 indicates that the shots contain different speakers. This

distance normalization step is necessary in comparing shot distance values that have different ranges.

Let $p(d\,|S)$ and $p(d\,|D)$ be the probability density functions (pdfs) of the distance between two shots, $d(S_i, S_j)$, when they contain the same person and different persons, respectively. Empirical pdfs obtained from our training data set for the three shot distances introduced in Section 6.3 are shown in Figure 6.3. We then can obtain the cumulative distribution functions for shot distances for the case of same and different speakers as

$$\begin{aligned} P(d\,|S) &= \textstyle\int_{-\infty}^{d} p(y\,|S)\,dy \\ P(d\,|D) &= \textstyle\int_{d}^{\infty} p(y\,|D)\,dy. \end{aligned} \tag{6.5}$$

This normalization is performed on all three of the shot distances. For each value of the shot distance, d, these cumulative distribution functions give the conditional probabilities that the shots contain same or different speakers, respectively. Using these conditional probabilities we define the normalized distance between shots as

$$d_n = \frac{P(d|S)}{P(d|S) + P(d|D)}. \tag{6.6}$$

Once shots distances corresponding to different features are calculated and normalized using (6.6), we derive a single distance value that is used to cluster similar shots together. Note that, as described in Section 6.3, not all features are available to calculate distance values for all pairs of shots S_i and S_j. Therefore, for each pair we derive the single shot distance values using the available shot feature distances, as illustrated in Figure 6.4. In our analysis of news programs we have found that consecutive shots very rarely contain the same person. For this reason, we set the confidence value between consecutive shots to be equal to 1.

6.5 Shot Clustering

After pairwise shot distances are calculated for a sequence, we use agglomerative or bottom-up clustering to group shots containing the same person together. Let N

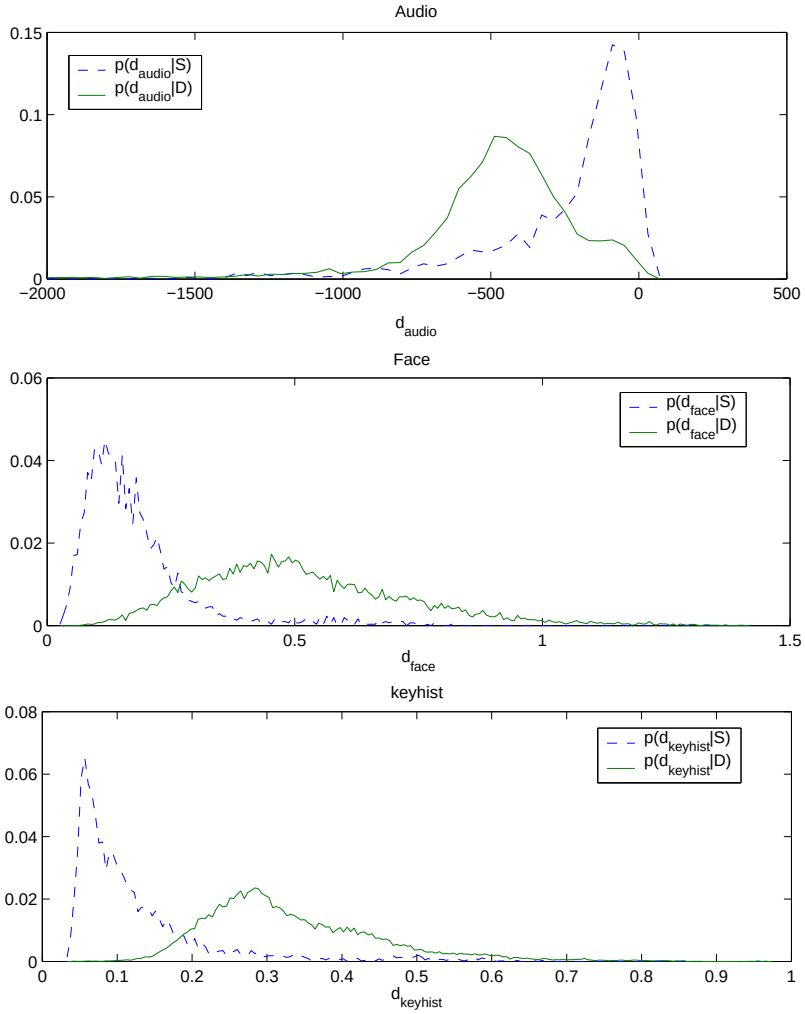


Fig. 6.3. Conditional probability density functions for the audio, face histogram, and keyframe histogram distances.

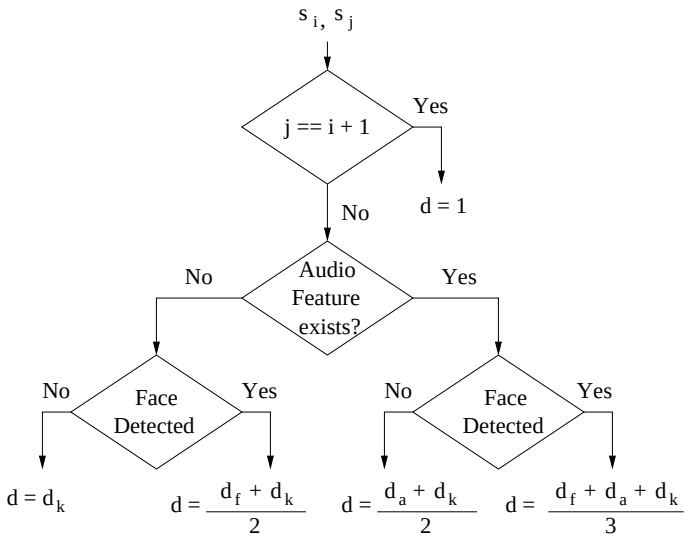


Fig. 6.4. Scheme used to derive a single shot distance value from multimodal feature distance values.

be the number of shots in the given video sequence to be processed. At the start of the algorithm all shots are placed in separate clusters so we start with N clusters. Then, at each iteration of the agglomerative clustering algorithm we search through the distance matrix, $\{d_{ij} = d(i,j)\}, 1 \leq i < j \leq N$, and find the two clusters, r and s, having the minimum distance. These two clusters are merged into a new cluster k and the distances from the newly formed cluster k to all remaining clusters are updated. We have used the complete-link update rule [151] for updating distances, which is given as

$$d_{hk} = \max(d_{hr}, d_{hs}),\ 1 \leq h \leq N,\ h \neq k. \tag{6.7}$$

The algorithm stops if either the number of clusters is equal to N_{min} or the smallest distance found is less than a threshold τ. We assume that there will be at least two people in all programs, so we used the value $N_{min} = 2$. For the threshold on cluster distances we have used the value $\tau = 0.5$. Note that this value of the threshold is not an arbitrary choice, but reflects the upper bound on the normalized confidence value for shots containing the same person. This is due to our normalization process, described in Section 6.4.

Table 6.1
Information about the C-SPAN sequences used in the unique person detection experiments.

sequence name	program type	number of shots	duration (min.)
cspan3	house committee	14	20
cspan4	senate committee	23	20
cspan7	forum	26	20
cspan11	senate proceeding	9	40
cspan12	senate proceeding	22	40
cspan15	house proceeding	21	40
cspan17	house proceeding	42	40
cspan18	house committee	36	40
cspan20	forum	23	40
cspan21	house committee	41	40
cspan22	house proceeding	85	80
cspan25	forum	61	60
cspan27	panel	59	55
TOTAL		462	535

6.6 Unique Speaker Detection Results

For our speaker detection experiments we have used 13 C-SPAN sequences that contain more than two persons. Information about these sequences is listed in Table 6.1. For each sequence, we cluster the keyframes extracted from each shot from the sequence to obtain the unique persons appearing in the program.

In evaluating the accuracy of our person detection system we have used the following rules

1. Only keyframes containing people of interest were considered in our error analysis, that is, key frames containing the audience, wide shots, etc. were ignored.

2. A *false split* was declared if the keyframes belonging to the same speaker are split into two different clusters. For example, if the same person is split into

Table 6.2
Unique person detection results.

sequence name	number of speakers	false split	wrong grouping
cspan3	3	0	0
cspan4	4	0	0
cspan7	3	2	1
cspan11	4	0	0
cspan12	4	0	0
cspan15	9	0	1
cspan17	12	0	1
cspan18	6	4	2
cspan20	6	0	0
cspan21	5	1	0
cspan22	11	4	2
cspan25	8	2	3
cspan25	8	0	0
TOTAL	**82**	**13**	**10**

three clusters, we count this as two false splits. However, shots of the same person obtained using different camera angles were not considered as false splits if they are grouped in different clusters.

3. A *wrong grouping* was declared if a keyframe for a person is grouped with that of another one in the same cluster. For example, if a cluster contains keyframes from three different people, we count this as two wrong groupings.

The unique person detection results for our data set are tabulated in Table 6.2. For each sequence we list the number of unique speakers who appear in that sequence and the number of false split and wrong grouping errors. Example of false split and a wrong grouping errors are shown in Figure 6.5.

From the results given in Table 6.2 we observe that our system is accurately able to detect unique people appearing news programs having in a wide variety of

Fig. 6.5. Keyframes from clusters 12, 13, and 14 arranges in rows from the sequence *cspan17*. First and third clusters is an example of a false split while the third cluster has a wrong grouping error.

content. The errors are mostly localized to three sequences, cspan18, cspan22, and cspan25. The main source of errors for these sequences is what we call "voice overs," which occur when a shot of a person contains the audio for another person. These errors may be handled by matching the audio to the video to determine if the audio belongs to the person person appearing in the video, such as the system described in [152].

6.7 Conclusions

In this chapter we have described a system that uses multimodal features to detect unique people appearing in news programs and illustrated our results using data obtained from C-SPAN. Our system uses features based on images, face regions, and audio to cluster shots containing the same person together. The experimental results show that our system is able to accurately detect the unique people in most news programs. The false split percentage 15.9% was and the wrong grouping percentage

was 12.2% on a collection of programs containing a total number of 82 speakers. The errors were mainly concentrated on a few programs where we had cases for which the audio for a shot does not belong to the person appearing in the shot. This problem may be solved by implementing measures based on audio continuity where video and audio segments are matched for the sequence.

We have also described how shot distance measures based on different modalities and having different ranges can be combined into a single shot distance measure using normalization. Given the three shot distance values, we perform this normalization using empirical distributions of these distances obtained using a training set.

7. CONCLUSIONS

In this dissertation we have addressed basic problems in video program indexing and analysis, namely detection of shot boundaries and building grammar models for video program structure. In addition, we have developed systems for various higher level applications, such as summarization of video programs and the detection of unique people in news programs.

7.1 Contributions of this Work

The contributions of this work may be listed as follows.

- We have developed shot transition detection algorithms based on regression tree classifiers. We have shown how abrupt transitions, such as cuts, and gradual transitions, such as dissolves, may be detected using the same framework. In the literature shot transition detection is usually achieved using rule-based systems that have many ad-hoc parameter settings and have to be fine-tuned for each program genre. Our classifier-based approach overcomes these problems and can also be easily extended by adding extra features. Thresholding and post-processing the output of the regression tree is more robust than thresholding the value of a frame similarity feature, since the tree chooses the best features from the generalized trace feature vector and the threshold is not dependent on the specific video sequence. We have shown that our cut detection technique has better overall performance and has a performance that is approximately constant across program genres. The regression tree methodology is also effective in detecting dissolve shot transitions.

- We examined the use of stochastic models to capture the structure of program genres and shown how such models may be used to automatically classify a given video sequence to one of a predetermined number of program genres. We developed the novel concept of representing video sequences as sequences of discrete-valued shot labels. We also showed how these shot labels may be obtained by clustering feature vectors extracted from video sequences. For hidden Markov models we have used the distance between models as a predictor for genre classification performance.

- One of the important contributions of this work was the development of the hybrid model that incorporates both the advantages of hidden Markov models and stochastic context-free grammars in modeling video sequences. Although stochastic context-free grammars (SCFGs) are well-suited to model the hierarchical structure of video sequences, the high computational time required for their training limits their applicability for video programs. Using our proposed hybrid model, we showed how the training time for SCFGs may be significantly reduced, which makes the hybrid model an practical tool in video sequence analysis.

- Compact representation of video sequences is fast becoming one of the most important topics in video analysis. Most of the previous work in the literature has focused on image-based methods for video summarization. In this work we have introduced a new method that uses speech transcripts, obtained using automatic speech recognition, that can be used to summarize a wide variety of video content, including documentary programs, presentations, and video conferences, that would be hard to address using only image-based approaches. An important advantage of the proposed summarization algorithm is that it quantifies the information contained in the summary and the summary coverage and maximizes a weighted sum of both detail and coverage functions to obtain a trade-off between the two. Therefore, our algorithm enables a user

to change relative weights for detail and coverage and regenerate the video summary of a program with more detail or more coverage, depending on a particular application.

- In most studies on video summarization, an evaluation of the proposed algorithm is omitted. However, we believe that rigorous testing of proposed algorithms are a crucial part of the video summarization research. To this effect, we conducted a large-scale user study to examine various summary evaluation schemes. We examined various factors that may affect the results of such evaluation experiments, such as program content and whether the experimental subjects were native speakers or not. Our user study and analysis is the most comprehensive summary evaluation study so far in the literature, to the best of our knowledge. The results of our study suggests that our algorithm generates better summaries than two other algorithms, in the sense that the summaries are more informative and their usage ratings are higher.

- We also proposed a system to detect unique people appearing in news programs. This system forms the automatic video graphics insertion system that will be deployed by C-SPAN to streamline the on-screen graphics insertion process for their programs. We showed how a mixture of image, audio, and face features can be used to cluster the segments of a program containing the same person. An important contribution of this system is the feature normalization procedure that is required to combine different shot distances.

- The three types of shot distances used in the unique speaker detection system affect the final shot distance measure equally. A better way to calculate the shot distance measure is to use a weighted sum of the three types of shot distances. These weights may be automatically determined from ground truth data.

7.2 Directions for Future Research

The work presented in this dissertation can be extended in many directions, we discuss some of these in this section.

- We will compare the shot transition detection performance of our regression tree based classifier with other patter classification methods, such as support vector machines.

- One can use the a priori distribution of shot lengths in a Bayesian hypothesis testing system to post-process the results of the shot transition classifier and reject candidate transition locations that have low likelihood according to the length distributions. We have done some preliminary studies on the distribution of shot lengths for different program genres and have found that these distributions can be modeled by so-called long-tailed probability distributions. We will extend this analysis and incorporate the shot length likelihood function in our shot transition detector.

- Our summarization algorithm currently uses only the information extracted from speech transcripts of programs. We will investigate the use of multimodal data, such as those derived from images and on-screen graphics, in conjunction with the speech transcripts, to increase the quality of the summaries produced by our algorithm.

- We will also examine the ways in which the stochastic program genre models can be used to identify important parts of video programs, which improve the quality of the summaries generated by video summarization algorithms.

- Most of the errors in the current version of the unique person detection system is caused by "voice overs," that is, shots where the audio does not match the speaker appearing in the shot. This problem may be solved by segmenting the audio track according to the speakers. Then, a dynamic programming

algorithm can be used to the align the audio segments with shots containing unique people.

APPENDIX

APPENDIX A
QUESTIONS USED IN THE VIDEO SUMMARY EVALUATION USER STUDY

Quiz questions for the SeaHorse sequence

Question 1

Seahorses are

(a) a type of aquatic mammal

(b) neither fish nor mammal

(c) a kind of fish

(d) related to reptiles

Question 2

In seahorses

(a) the female gives birth

(b) the male gives birth

(c) either the male or the female can give birth

(d) neither the male or the female gives birth but sperm and eggs mix in water

Question 3

Seahorses protect themselves from predators by

(a) camouflage

(b) their poisonous tail

(c) their speed

(d) their pointed teeth

Question 4

The bodies of seahorses are designed for

(a) speed rather than stability

(b) flexibility rather than stability

(c) flexibility rather than speed

(d) stability rather than speed

Question 5

Seahorses eat

(a) tiny shrimp

(b) small pieces of grass and weed

(c) small fish

(d) shellfish and oysters

Question 6

The breeding season for seahorses is

(a) one month

(b) seven months

(c) the whole year

(d) only during spring

Question 7

During the breeding season female seahorses mate males

(a) daily, loyal to their male partner

(b) occasionally, with any male they find

(c) daily, with any male they find

(d) occasionally, loyal to their male partner

Question 8

Seahorses grow

(a) until they are one year old

(b) until they are 18 months old

(c) until they are 2 years old

(d) for their entire life period

Question 9

The male seahorse is distinguished from the female by its

(a) teeth

(b) tail

(c) pouch

(d) dorsal fin

Question 10

The males carry fertilized eggs until they develop

(a) only in seahorses among animals

(b) in seahorses and some small mammals

(c) in seahorses and some other fish, like the pipefish

(d) in all species of fish

Quiz questions for the Mark Twain sequence

Question 1

Mark Twain's childhood home was in

(a) Missouri

(b) New York

(c) Georgia

(d) Louisiana

Question 2

Which of the following best describes Mark Twain's genius?

(a) Humor

(b) Speaking the voice of America

(c) Use of spirituality in his work

(d) Journalistic skills

Question 3

Mark Twain's position with respect to racial problems in America can be best described as

(a) He supported slavery because it made the South rich

(b) He was ignorant of the problem

(c) He thought that slavery was supported by Christianity

(d) He understood that the problem of race needed to be solved if America was going to be a nation

Question 4

Mark Twain's real last name was

(a) Clemens

(b) Jones

(c) Culpepper

(d) Hannibal

Question 5

Which of the following is true of Mark Twain's father?

(a) He was a successful businessman

(b) His jokes made a strong impression on Mark Twain

(c) Mark Twain never saw him laugh

(d) He was a farmer in Connecticut

Question 6

The most powerful and vivid storytelling voices of Mark Twain's early childhood belonged to

(a) his father

(b) hid grandparents

(c) black people

(d) local indians

Question 7

When his father died, Mark Twain had to leave school and work as a

(a) barber's apprentice

(b) printer's apprentice

(c) writer's secretary

(d) telegraph operator

Question 8

On the Mississipi River steamboats Mark Twain worked as a

(a) clerk

(b) cub pilot

(c) janitor

(d) telegraph operator

Question 9

What event caused great sorrow for Mark Twain and a great sense of remorse that he never lost?

(a) The death of his father, John

(b) The death of his mother, Jane

(c) The death of his older brother, Orion

(d) The death of his younger brother, Henry

Question 10

Mark Twain's way to repair his sorrow caused by his younger brother's death was through

(a) humor

(b) spirituality

(c) womanizing

(d) drinking

Quiz questions for the Why Dogs Smile sequence

Question 1

Why is it impossible to teach an animal to communicate by speaking?

(a) They do not have similar emotions to us

(b) They lack vocal chords

(c) Their brain capacity is insufficient to learn how to speak

(d) It is unnatural for them

Question 2

In 1966 psychologists Alan and Beatrice Gardner began to teach a young chimpanzee called Washoe to communicate using

(a) English

(b) Universal Sign Language

(c) American Sign Language

(d) A simplified language of sounds and gestures

Question 3

Why do humans have trouble attributing emotions to animals?

(a) Because we are more focused on understanding our own kind

(b) Because animals show no detectable signs of emotion

(c) Because their social structure is different than ours

(d) Because we feel superior to animals

Question 4

Which one is true about the mother-child bond?

(a) It has a biochemical foundation

(b) It is purely psychological in nature

(c) It is not observed in some mammals

(d) It is also observed in reptiles

Question 5

From the program we learn that elephants

(a) do not have as strong a mother - child bond as dolphins

(b) have a sense of humor

(c) have a very strong maternal devotion

(d) can communicate with signs

Question 6

Scientists think that the skin color of an octopus reflects its emotions. Which colors are connected with anger and fear?

(a) anger - blue, fear - pale

(b) anger - red, fear - pale

(c) anger - pale, fear - blue

(d) anger - red, fear - blue

Question 7

Which of the following is true of the biochemical glue that bonds mother and infant together in mammals?

(a) It is called beta-endorphin

(b) It is related to insulin

(c) It is called seratonin

(d) Its effect on the brain is similar to chocolate

Question 8

Mammals were able to compete with the reptiles because

(a) they are born pre-programmed

(b) they lived in social groups and learned from each other

(c) they had better hunting skills due to their better eyesight

(d) being warm-blooded they could move faster

Question 9

In nature mothers provide guidance to their children about the dangers of the world

(a) in a positive way, by rewarding them

(b) in an indirect way by setting an example

(c) by ignoring them after a dangerous act

(d) in a negative way, by punishing them after a dangerous act

Question 10

The studies done in the University of Wisconsin in the 1960s demonstrated that the most important factor for the physical growth and brain development of an infant is

(a) nutrition provided by the mother

(b) physical contact with the mother

(c) interaction with siblings

(d) maintaining the necessary sleep cycle

LIST OF REFERENCES

LIST OF REFERENCES

[1] Haitao Jiang and Ahmed K. Elmagarmid. Extracting visual content representation in video databases. In *Proceedings of CISST'97, International Conference on Imaging Science, Systems, and Technology*, pages 189–196, Las Vegas, NE, July 1997.

[2] Mary C. Dyson. How do you describe a symbol? the problems involved in retrieving symbols from a database. *Information Services and Use*, 12:65–76, 1992.

[3] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The qbic project: Querying for image content using color, texture and shape. In *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 1908, pages 173–181, San Jose, CA, February 1993.

[4] Philippe Aigrain and Philippe Joly. The automatic real-time analysis of film editing and transition effects and its applications. *Computation and Graphics*, 18(1):93–103, 1994.

[5] A. Del Bimbo. A perspective view on visual information retrieval systems. In *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Databases*, pages 108–109, Santa Barbara, CA, June 1998.

[6] Dragutin Petkovic. Challenges and opportunities in search and retrieval for medial databases. In *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Databases*, pages 110–111, Santa Barbara, CA, June 1998.

[7] ISO/IEC JTC1/SC29/WG11. MPEG-7: Requirements document (v12) w3548. In *MPEG Meeting*, Beijing, China, July 2000.

[8] Tessa Lau and Eric Horvitz. Patterns of search: Analyzing and modeling web query refinement. In *Proceedings of the Seventh International Conference on User Modeling*, Banff, Canada, June 20 – 24 1999.

[9] Yossi Rubner, Leonidas Guibas, and Carlo Tomasi. The earth mover's distance, multi-dimenasional scaling, and color-based image retrieval. In *Proceedings of the ARPA Image Understanding Workshop*, May 1997.

[10] Jeho Nam, Enis Cetin, and Ahmed Tewfik. Speaker identification and video analysis for hierarchical video shot classification. In *Proceedings of IEEE International Conference on Image Processing*, pages 550–553, Santa Barbara, CA, October 1997.

[11] Nilesh V. Patel and Ishwar K. Sethi. Audio characterization for video indexing. In *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases IV*, volume 2670, pages 373–384, San Jose, CA, February 1996.

[12] Caterina Saraceno and Riccardo Leonardi. Audio as a support to scene change detection and characterization of video sequences. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, pages 2597–2600, Munich, Germany, April 1997.

[13] http://www.fcc.gov/cgb/consumerfacts/closedcaption.html.

[14] Shih-Fu Chang. Linking features to semantics - accessing visual content at multiple levels. In *Proceedings of IMAGINA'98, The Monte-Carlo Television Festival*, Monte-Carlo, Monaco, March 1998.

[15] Rakesh Mohan. Text-based search of tv news stories. In *Multimedia Storage and Archiving Systems*, volume 2916, pages 2–13, Boston, MA, November 1996.

[16] Yuichi Nakamura and Takeo Kanade. Semantic analysis for video contents extraction - spotting by association in news video. In *Proceedings of ACM Multimedia'97: The Fifth ACM International Multimedia Conference*, pages 393–401, Seattle, WA, November 9-13 1997.

[17] Andrew Merlino, Daryl Morey, and Mark Maybury. Broadcast nesw navigation using story segmentation. In *Proceedings of ACM Multimedia'97: The Fifth ACM International Multimedia Conference*, pages 381–391, Seattle, WA, November 1997.

[18] Sharmila Kannangara, Eduardo Asbun, Robert X. Browning, and Edward J. Delp. The use of nonlinear filtering in automatic video title capture. In *Proceedings of the 1997 IEEE/EUROSIP Workshop on Nonlinear Signal and Image Processing*, Mackinac Island, MI, September 1997.

[19] Toshio Sato, Takeo Kanade, Ellen K. Hughes, and Michael A. Smith. Video ocr for digital news archive. In *IEEE International Workshop on Content-Based Access of Video and Image Databases*, pages 52–60, Los Alamitas, CA, January 1998.

[20] Deborah Swanberg, Chiao-Fe Shu, and Ramesh Jain. Knowledge guided parsing in video databases. In *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 1908, pages 13–24, San Jose, CA, February 1993.

[21] Ullas Gargi, Rangachar Kasturi, and Susan H. Strayer. Fast scene change detection using direct feature extraction from MPEG compressed videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(1):1–13, February 2000.

[22] Rainer Lienhart. Comparison of automatic shot boundary detection algorithms. In *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 290–301, San Jose, CA, January 1999.

[23] C. O'Toole, A. Smeaton, N. Murphy, and S. Marlow. Evaluation of automatic shot boundary detection on a large video test suite. In *Proceedings of the 2nd UK Conference on Image Retrieval*, Newcastle, UK, February 25-26 1999.

[24] John Boreczky and Lawrence Rowe. Comparison of video shot boundary detection techniques. In *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases IV*, volume 2670, pages 170–179, San Jose, CA, February 1996.

[25] Apostolos Dailianas, Robert B. Allen, and Paul England. Comparison of automatic video segmentation algorithms. In *Proceedings of SPIE Photonics East'95: Integration Issues in Large Commercial Media Delivery Systems*, pages 1–16, Philadelphia, PA, October 1995.

[26] Arun Hampapur, Ramesh Jain, and Terry Weymouth. Digital video segmentation. In *Proceedings of the 2nd ACM International Conference on Multimedia*, pages 357–364, San Francisco, CA, October 1994.

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

[28] Boon-Lock Yeo and Bede Liu. Rapid scene analysis on compressed video. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(6):533–544, December 1995.

[29] Edoardo Ardizzone and Marco La Cascia. Multifeature image and video content-based storage and retrieval. In *Proceedings of SPIE Conference on Multimedia Storage and Archiving Systems*, volume 2916, pages 265–276, Boston, MA, November 18-19 1996.

[30] Nilesh V. Patel and Ishwar K. Sethi. Video shot detection and characterization for video databases. *Pattern Recognition*, 30(4):583–592, April 1997.

[31] A. Mufit Ferman and A. Murat Tekalp. Multiscale content extraction and representation for video indexing. In *Proceedings of SPIE Conference on Multimedia Storage and Archiving Systems II*, pages 23–31, Dallas, TX, November, 3-4 1997.

[32] Vikrant Kobla and David Doermann. Videotrails: Representing and visualizing structure in video sequences. In *Proceedings of the Fifth ACM International Multimedia Conference*, pages 335–346, Seattle, WA, November 9-13 1997.

[33] Keesok J. Han and Ahmed Tewfik. Eigen-image based video segmentation and indexing. In *Proceedings of IEEE International Conference on Image Processing*, pages 538–541, Santa Barbara, CA, October 1997.

[34] Ramin Zabih, Justin Miller, and Kevin Mai. A feature-based algorithm for detecting and classifying scene breaks. In *Proceedings of the 3rd ACM International Conference on Multimedia*, San Francisco, CA, November 5-9 1995.

[35] Bo Shen, Donnge Li, and Ishwar K. Sethi. Cut detection via compressed domain edge extraction. In *IEEE Workshop on Nonlinear Signal and Image Processing*, Mackinac Island, MI, September 1997.

[36] Vikrant Kobla, David Doerman, and King-Ip Lin. Archiving, indexing, and retrieval of video in the compressed domain. In *Proceedings of SPIE Conference on Multimedia Storage and Archiving Systems*, volume 2916, pages 78–89, Boston, MA, November 1996.

[37] HongJiang Zhang, Chien Yong Low, Yihong Gong, and Stephen W. Smoliar. Video parsing using compressed data. In *Proceedings of SPIE Conference on Image and Video Processing II*, volume 2182, pages 142–149, San Jose, CA, February 1994.

[38] Jianhao Meng, Yujen Juan, and Shih-Fu Chang. Scene change detection in a MPEG compressed video sequence. In *Proceedings of SPIE Conference on Multimedia Computing and Networking*, volume 2417, pages 180–191, San Jose, CA, February 1995.

[39] Arun Hampapur, Ramesh Jain, and Terry Weymouth. Production model based digital video segmentation. *Multimedia Tools and Applications*, 1(1):1–38, March 1995.

[40] S. Moon-Ho Song, Tae-Hoon Kwon, and Woonkyung M. Kim. On detection of gradual scene changes for parsing of video data. In *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases VI*, volume 3312, pages 404–413, San Jose, CA, January 1998.

[41] Nuno Vasconcelos and Andrew Lippman. A Bayesian video modeling framework for shot segmentation and content characterization. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, San Juan, Purto Rico, October 4-7 1997.

[42] Hong Jiang Zhang, Shuang Yeo Tan, Stephen Smoliar, and Gong Yihong. Automatic parsing and indexing of news video. *Multimedia Systems*, 2(6):256–266, 1995.

[43] Farshid Arman, Arding Hsu, and Ming-Yee Chiu. Feature management for large video databases. In *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 1908, pages 2–12, San Jose, CA, February 1993.

[44] F. Idris and S. Panchanathan. Indexing of compressed video sequences. In *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases IV*, volume 2670, pages 247–253, San Jose, CA, February 1996.

[45] Hong Jiang Zhang, Jianhua Wu, Di Zhong, and Stephen Smoliar. An integrated system for content-based video retrieval and browsing. *Pattern Recognition*, 30(4):643–658, April 1997.

[46] Vikrant Kobla, Daniel DeManthon, and David Doermann. Special effect edit detection using *videotrails*: A comparison with existing techniques. In *Proceedings of SPIE Conference on Storage and Retrieval for Image and Video Databases VII*, volume 3656, pages 302–313, San Jose, CA, January 1999.

[47] Michael Swain and Dana H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.

[48] Adnan Alattar. Detecting and compressing dissolve regions in video sequences with a dvi multimedia image compression algorithm. In *Proceedings of IEEE Symposium on Circuits and Systems*, pages 13–16, Chicago, IL, May 1993.

[49] S. Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE Transactions on Systems Man and Cybernetics*, 21(3):660–674, May 1991.

[50] G. R. Dattatreya and L. N. Kanal. Decision trees in pattern recognition. In L.N. Kanal and A. Rosenfeld, editors, *Progress in Pattern Recognition*, volume 2, pages 189–239. Elsevier Science Publishers B.V. (north-Holland), 1985.

[51] Sreerama K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery Journal*, 2(4), 1998.

[52] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA, 1984.

[53] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley & Sons Inc., New York, NY, 2001.

[54] Saul Gelfand, C. Ravishankar, and Edward Delp. An iterative growing and pruning algorithm for classification tree design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(2):163–174, February 1991.

[55] Ba Tu Truong, Chitra Dorai, and Svetha Venkatesh. New enhancements to cut, fade, and dissolve detection processes in video segmentation. In *Proceedings of the 8th ACM Multimedia Conference*, pages 290–301, Los Angeles, CA, October 30 - November 4 2000.

[56] Rainer W. Lienhart. Reliable dissolve detection. In *Proceedings of SPIE Conference on Storage and Retrieval for Media Databases 2001*, volume 4315, pages 219–230, San Jose, CA, January 2001.

[57] Sung-Bae Jun, Kyoungro Yoon, and Hee-Youn Lee. Dissolve transition detection algorithm using spatio-temporal distribution of MPEG macro-block types. In *Proceedings of the 8th ACM Multimedia Conference*, Los Angeles, CA, October 30 - November 4 2000.

[58] D. Stan and I. K. Sethi. Mapping low-level image features to semantic concepts. In *Proceedings of SPIE Conference on Storage and Retrieval for Media Databases 2001*, volume 4315, pages 172–179, San Jose, CA, January 2001.

[59] Cuneyt Taskiran, Charles Bouman, and Edward J. Delp. Discovering video structure using the pseudo-semantic trace. In *Proceedings of the SPIE Conference on Storage and Retrieval for Media Databases 2001*, pages 571–578, San Jose, CA, January 24-26 2001.

[60] M. R. Frater, J. F. Arnold, and P. Tan. A new statistical model for traffic generated by VBR coders for television on the broadband ISDN. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(6):521–526, December 1994.

[61] U. K. Sarkar, S. Ramakrishnan, and D. Sarkar. Segmenting full-length VBR video into shots for modeling with Markov-modulated gamma-based framework. In *SPIE Symposium on the Convergence of Information Technologies and Communications (ITCom 2001)*, Denver, CO, 19–24 August 2001.

[62] S. I. Resnick. Heavy tail modeling and teletraffic data. *The Annals of Statistics*, 25(5):1805–1869, 1997.

[63] M.E. Crovella, M.S. Taqqu, and A. Bestavros. Heavy-tailed probability distributions in the world wide web. In A.R.E. Feldman and M.S. Taqqu, editors, *A Practical Guide to Heavy Tails*, pages 3–25. Birkhauser, 1998.

[64] M. Krunz and A. M. Ramasamy. The correlation structure for a class of scene-based video models and its impact on the dimensioning of video buffers. *IEEE Transactions on Multimedia*, 2(1):27–36, March 2000.

[65] D. P. Heyman and T. V. Lakshman. Long-range dependence and quueing effects for VBR video. In K. Park and W. Willinger, editors, *Self-similar Network Traffic and Performance Evaluation*, pages 285–318. John Wiley and Sons, 2000.

[66] A. M. Dawood and M. Ghanbari. Content-based MPEG video traffic modeling. *IEEE Transactions on Multimedia*, 1(1):77–87, March 1999.

[67] The effect of Multiple Time Scales and Subexponentiality in MPEG video streams on queuing behavior. P. r. jelenkovic and a. a. lazar and n. semret. *IEEE Journal on Selected Areas in Communications*, 15(6):1052–1071, August 1997.

[68] Shih-Fu Chang, John R. Smith, Mandis Beigi, and Ana Benitez. Visual information retrieval from large distributed online repositories. *Communications of the ACM*, 40(12):63–71, 1997.

[69] P. Embrechts, C. Kluppelberg, and T. Mikosch. *Modeling Extremal Events for Insurance and Finance*. Springer, New York, NY, 1997.

[70] A. Dekkers and L. D. Haan. On the estimation of the extreme value index and large quantile estimation. *The Annals of Statistics*, 17:1883–1855, 1989.

[71] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in C:The Art of Scientific Computing*. Cambridge University Press, Cambridge, MA, 1988.

[72] Lawrence Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[73] J. Henderson S.Salzberg and K. Fasman. Finding genes in human dna with a hidden markov model. *Journal of Computational Biology*, 4(2):127–141, 1997.

[74] John Boreczky and Lynn D. Wilcox. A hidden Markov model framework for video segmentation. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, pages 3741–3744, Seattle, WA, May 1998.

[75] Stefan Eickeler and Stefan Muller. Content-based video indexing of tv broadcast news using hidden markov models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 2997–3000, Phoeniz, AZ, March 1999.

[76] Wayne Wolf. Hidden Markov model parsing of video programs. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, pages 2609–2611, Munich, Germany, April 1997.

[77] A. Aydin Alatan, Ali N. Akansu, and Wayne Wolf. Multi-modal dialog scene detection using hidden Markov models for content-based multimedia indexing. *Multimedia Tools and Applications*, 14(2):137–151, 2001.

[78] Zhu Liu, Jincheng Huang, and Yao Wang. Classification of TV programs based on audio information using hidden Markov model. In *IEEE Second Workshop on Multimedia Signal Processing*, pages 27–32, Redondo Beach, CA, December 1998.

[79] Matthew Brand and Vera Kettnaker. Discovery and segmentation of activities in video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):844–851, 2000.

[80] Lexing Xie, Shih-Fu Chang, Ajay Divakaran, and Huifang Sun. Structure analysis of soccer video with hidden Markov models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Orlando, Fl, May 13–17 2002.

[81] Yuri A. Ivanov and Aaron Bobick. Recogition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000.

[82] Darnell Moore and Irfan Essa. Recognizing multitasked activities from video using stochastic context-free grammar. In *Workshop on Models versus Exemplars in Computer Vision in IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Kauai, Hawaii, December 9–14 2001.

[83] Brett Adams, Chitra Dorai, and Svetha Venkatesh. Study of shot length and motion as contributing factors to movie tempo. In *Proceedings of the ACM International Conference on Multimedia*, pages 353–355, Los Angeles, CA, October 30 - November 4 2000.

[84] Nuno Vasconcelos and Andrew Lippman. Statistical models of video structure for content analysis and characterization. *IEEE Transactions on Image Processing*, 9(1):3–19, 2000.

[85] Sylvie Jeannin and Ajay Divakaran. MPEG-7 visual motion descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(6):720 – 724, 2001.

[86] J. Rissanen. A universal prior for integers and estimation by minimum description length. *Annals of Statistics*, 11(2):417–431, 1983.

[87] Iain L. MacDonald and Walter Zucchini. *Hidden Markov and Other Models for Discrete-valued Time Series*. Chapman and Hall, London, UK, 1997.

[88] B. H. Juang and L. R. Rabiner. A probabilistic distance measure for hidden markov models. *AT & T Technical Journal*, 64(2):391 – 408, February 1985.

[89] Christopher D. Manning and Hinrich Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, 1999.

[90] J. Baker. Trainable grammars for speech recognition. In D. Klatt and J. Wolf, editors, *Speech Communication papers for the 97th meeting of the Acouistical Society of America*, pages 557–550. MIT, Cambridge, MA, 1979.

[91] K. Lari and S. J. Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56, 1990.

[92] Belle L. Tseng, Ching-Yung Lin, and John R. Smith. Video summarization and personalization for pervasive mobile devices. In *Proceedings of SPIE Conference on Storage and Retrieval for Media Databases 2002*, volume 4676, pages 359–370, San Jose, CA, 23-25 January 2002.

[93] Minerva M. Yeung and Boon-Lock Yeo. Video visualization for compact presentation and fast browsing of pictorial content. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(5):771–785, October 1997.

[94] Y. Taniguchi, A. Akutsu, and Y. Tonomura. Panorama excerpts: Extracting and packing panoramas for video browsing. In *Proceedings of the ACM Multimedia*, pages 427–436, Seattle, WA, November 9-13 1997.

[95] Daniel DeMenthon, Vikrant Kobla, and David Doerman. Video summarization by curve simplification. In *Proceedings of the ACM Multimedia Conference*, pages 211–218, Bristol, England, September 12-16 1998.

[96] Shingo Uchihashi, Jonathan Foote, Andreas Girgenson, and John Boreczky. Video Manga: Generating semantically meaningful video summaries. In *Proceedings of ACM Multimedia'99*, pages 383–392, Orlando, FL, October 30 - November 5 1999.

[97] A. Stefanidis, P. Partsinevelos, P. Agouris, and P. Doucette. Summarizing video datasets in the spatiotemporal domain. In *Proceedings of the International Workshop on Advanced Spatial Data Management (ASDM'2000)*, Greenwich, England, September 6-7 2000.

[98] Cuneyt M. Taskiran, Arnon Amir, Dulce Ponceleon, and Edward J. Delp. Automated video summarization using speech transcripts. In *Proceedings of SPIE Conference on Storage and Retrieval for Media Databases 2002*, volume 4676, pages 371–382, San Jose, CA, 20-25 January 2002.

[99] JungHwan Oh and Kien A. Hua. An efficient technique for summarizing videos using visual contents. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'2000)*, New York, NY, July 30-August 2 2000.

[100] Rainer Lienhart. Dynamic video summarization of home video. In *Proceedings of SPIE Conference on Storage and Retrieval for Media Databases 2000*, volume 3972, pages 378–389, San Jose, CA, January 2000.

[101] Liwei He, Elizabeth Sanocki, Anoop Gupta, and Jonathan Grudin. Auto-summarization of audio-video presentations. In *Proceedings of the 7th ACM International Multimedia Conference*, pages 489–498, Orlando, FL, 30 October - 5 November 1999.

[102] Michael G. Christel, Micheal A. Smith, Roy Taylor, and David B. Winker. Evolving video skims into useful multimedia abstractions. In *Proceedings of the ACM Computer-Human Interface Conference (CHI'98)*, pages 171–178, Los Angeles, CA, April 18-23 1998.

[103] Silvia Pfeiffer, Rainer Lienhart, Stephan Fischer, and Wolfgang Effelsberg. Abstracting digital movies automatically. *Journal of Visual Communication and Image Representation*, 7(4):345–353, December 1996.

[104] Michael Christel, Alexander G. Hauptman, Adrienne S. Warmack, and Scott A. Crosby. Adjustable filmstrips and skims as abstractions for a digital video library. In *Proceedings of the IEEE Conference on Advances in Digital Libraries*, Baltimore, MD, May 19-21 1999.

[105] Howard D. Wactlar. Informedia - search and summarization in the video medium. In *Proceedings of the IMAGINA 2000 Conference*, Monaco, January 31 - February 2 2000.

[106] Nuno Vasconcelos and Andrew Lippman. Bayesian modeling of video editing and structure: Semantic features for video summarization and browsing. In *Proceedings of IEEE International Conference on Image Processing*, Chicago, IL, October 4-7 1998.

[107] Cuneyt Taskiran, Jau-Yuen Chen, Alberto Albiol, Luis Torres, Charles A. Bouman, and Edward J. Delp. Vibe: A compressed video database structured for active browsing and search. *IEEE Transactions on Multimedia*, 6(1):103–118, February 2004.

[108] A. Amir, S. Srinivasan, and D. Ponceleon. Efficient video browsing using multiple synchronized views. In A. Rosenfeld, D. Doermann, and D. DeMenthon, editors, *Video Mining*. Kluwer Academic Publishers, Boston, MA, August 2003.

[109] Dulce Ponceleon and Andreas Dieberger. Hierachical brushing in a collection of video data. In *Proceedings of the 34th Hawaii International Conference on System Sciences (HICSS'34)*, Maui, Hawaii, January 3-6 2001.

[110] Qian Huang, Zhu Liu, Aaron Rosenberg, David Gibbon, and Behzad Shahraray. Automated generation of news content hierarchy by integrating audio, video, and text information. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3025–3028, Phoenix, AR, March 15-19 1999.

[111] Aya Aner, Lijun Tang, and John R. Kender. A method and browser for cross-referenced video summaries. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'2002)*, Lausanne, Switzerland, August 2002.

[112] Arnon Amir, Dulce B. Ponceleon, Brian Blanchard, Dragutin Petkovic, Savitha Srinivasan, and G. Cohen. Using audio time scale modification for video browsing. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences (HICSS-33)*, Maui, Hawaii, 4-7 January 2000.

[113] Barry Arons. Speechskimmer: A system for interactively skimming recorded speech. *ACM Transactions on Computer Human Interaction*, 4(1):3–38, 1997.

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

[115] S. M. Iacob, R. L. Lagendijk, and M. E. Iacob. Video abstraction based on asymmetric similarity values. In *Proceedings of SPIE Conference on Multimedia Storage and Archiving Systems IV*, volume 3846, pages 181–191, Boston, MA, September 1999.

[116] Krishna Ratakonda, Ibrahim M. Sezan, and Regis J. Crinon. Hierarchical video summarization. In *Proceedings of SPIE Conference Visual Communications and Image Processing*, volume 3653, pages 1531–1541, San Jose, CA, January 1999.

[117] A. M. Ferman and A. M. Tekalp. Two-stage hierarchical video summary extraction to match low-level user browsing preferences. *IEEE Transactions on Multimedia*, 5(2):244–256, 2003.

[118] Dirk Farin, Wolfgang Effelsberg, and Peter H. N. de With. Robust clustering-based video-summarization with integration of domain-knowledge. In *Proceedings of the IEEE International Conference on Multimedia and Expo 2002 (ICME'2002)*, pages 89–92, Lausanne, Switzerland, 26-29 August 2002.

[119] I. Yahiaoui, B. Merialdo, and B. Huet. Comparison of multi-episode video summarization algorithms. *EURASIP Journal on Applied Signal Processing*, 3(1):48–55, 2003.

[120] Yihong Gong and Xin Liu. Video summarization using singular value decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 174–180, 13-15 June 2000.

[121] Matthew Cooper and Jonathan Foote. Summarizing video using non-negative similarity matrix factorization. In *International Workshop on Multimedia Signal Processing*, St. Thomas, US Virgin Islands, December 9-11 2002.

[122] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. *Advanced Neural Information Processing Systems*, 13:556–562, 2001.

[123] Ahmet Ekin and A. Murat Tekalp. Automatic soccer video analysis and summarization. In *Proceedings of SPIE Conference on Storage and Retrieval for Media Databases 2003*, volume 5021, pages 339–350, Santa Clara, CA, 20-24 January 2003 2003.

[124] Baoxin Li, Hao Pan, and Ibrahim Sezan. A general framework for sports video summarization with its application to soccer. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, pages 169–172, Hong Kong, April 6-10 2003.

[125] Romain Cabasson and Ajay Divakaran. Automatic extraction of soccer video highlights using a combination of motion and audio features. In *Proceedings of SPIE Conference on Storage and Retrieval for Media Databases 2003*, volume 5021, pages 272–276, Santa Clara, CA, 20-24 January 2003.

[126] Lalitha Agnihotri, Kavitha V. Devera, Thomas McGee, and Nevenka Dimitrova. Summarization of video programs based on closed captions. In *Proceedings of SPIE Conference on Storage and Retrieval for Media Databases 2001*, volume 4315, pages 599–607, San Jose, CA, January 2001.

[127] Marcus J. Pickering, Lawrence Wong, and Stefan M. Rueger. ANSES: Summarization of news video. In *Proceedings of the International Conference on Image and Video Retrieval*, volume LNCS 2728, pages 425–434, Urbana, IL, July 24-25 2003.

[128] Regina Barzilay and Michael Elhadad. Using lexical chains for text summarization. In *Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS'97)*, Madrid, Spain, July 1997.

[129] Yu-Fei Ma, Lie Lu, Hong-Jiang Zhang, and Mingjing Li. A user attention model for video summarization. In *Proceedings of ACM Multimedia'02*, pages 533–542, Juans Les Pins, France, December 1-6 2002.

[130] Shih-Fu Chang and Hari Sundaram. Structural and semantic analysis of video. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME'2000)*, New York, NY, July 30-August 2 2000.

[131] Karen Sparck Jones. What might be in a summary? In Knorz, Krause, and Womser-Hacker, editors, *Information Retrieval 93: Von der Modellierung zur Anwendung*, pages 9–26, Konstanz, DE, September 1993. Universitätsverlag Konstanz.

[132] L. R. Bahl, S. Balakrishnan-Aiyer, J. R. Bellegarda, M. Franz, P. S. Gopalakrisnan, D. Nahamoo, M. Novak, M. Padmanabhan, M. A. Picheny, and S. Roukos. Performance of the IBM large vocabulary continuous speech recognition system on the ARPA Wall Street Journal task. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, pages 41–44, Detroit, MI, May 8-12 1995. no pdf.

[133] Karen Sparck Jones, Steve Walker, and Stephen E. Robertson. A probabilistic model of information retrieval: Development and status. *Information Processing and Management*, 36(6):779–840, 2000.

[134] S. S. Chen, E. M. Eide, M. J. F. Gales, R. A. Gopinath, D. Kanevsky, and P. Olsen. Automatic transcription of broadcast news. *Speech Communication*, 37(1–2):69–87, 2002.

[135] Ted E. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

[136] Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations*. J. Wiley and Sons, Chichester, NY, 1990.

[137] D. Pisinger. An expanding-core algorithm for the exact 0–1 knapsack problem. *European Journal of Operational Research*, 87:175–187, 1995.

[138] P. H. Luhn. Automatic creation of literature abstracts. *IBM Journal*, 2(2):159–165, 1958.

[139] Hongyan Jing, Regina Barzilay, Kathleen McKeown, and Michael Elhadad. Summarization evaluation methods: Experiments and analysis. In *Proceedings of the AAAI Symposium on Intelligent Summarization*, Palo Alto, CA, March 23 - 25 1998.

[140] I. Mani, D. House, G. KLein, L. Hirschman, L. Obrst, T. Firmin, M. Chrzanowski, and B. Sundheim. The TIPSTER SUMMAC text summarization evaluation. Technical report, National Institute of Standards and Technology, October 1998.

[141] Kingdom of the Seahorse, PBS NOVA, 1997.

[142] Mark Twain, PBS Home Video, 2002.

[143] Why Dogs Smile and Chimpanzees Cry, Discovery Home Video, 1999.

[144] C. Taskiran, A. Martone, R. Browning, , and E. Delp. A toolset for broadcast automation for the c-span networks. In *5th International Workshop on Image Analysis for Multimedia Interactive Services*, Lisboa, Portugal, April 21-23 2004.

[145] Zhi-Peng Zhan, Sadaoki Furui, and Katsutoshi Ohtsuki. On-line incremental speaker adaptation with automatic speaker change detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Istanbul, Turkey, 5 – 9 June 2000.

[146] H.J. Nock, G. Iyengar, and C. Neti. Speaker localisation using audio-visual synchrony: An empirical study. In *Proceedings of the International Conference on Image and Video Retrieval (CIVR 2003)*, pages 488 – 499, Urbana-Champaign, IL, 2003.

[147] Lie Lu and Hong-Jiang Zhang. Speaker change detection and tracking in real-time news broadcasting analysis. In *Proceedings of the 10th ACM International Conference on Multimedia*, pages 602 – 610, Juan-les-Pins, France, December 1 – 6 2002.

[148] S. Satoh, Y. Nakamura, and T. Kanade. Name-it: Naming and detecting faces in news videos. *IEEE Multimedia Magazine*, 6(1):22 – 35, January 1999.

[149] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Conference on computer vision and pattern recognition*, Kauai, HI, December 8-14 2001.

[150] S. S. Chen and P. S. Gopalakrishnan. Speaker, environment, and channel change detection and clustering via the Bayesian Information Criterion. In *DARPA Speech Recognition Workshop*, pages 127–132, Landsdowne, VA, February 1998.

[151] Anil K. Jain and Richard C. Dubes, editors. *Algorithms for Clustering Data*. Prentice Hall, New Jersey, 1988.

[152] Alberto Albiol, L. Torres, and E. J. Delp. The indexing of persons in news sequences using audiovisual data. In *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing*, Hong Kong, China, April 6–10 2003.

VITA

VITA

Cuneyt Taskiran was born in Istanbul, Turkey. He received the B.S. and M.S. degrees, both in Electrical Engineering from Bogazici University, Istanbul, Turkey, in 1990 and 1993, respectively. During his M.S. studies he concentrated on image classification for automatic target detection application

Currently, he is pursuing the Ph.D. degree in the School of Electrical and Computer Engineering and the M.A. degree in the Interdepartmental Program in Linguistics of Purdue University, West Lafayette, Indiana. He has worked as a teaching assistant for a number of undergraduate courses. During Fall 2002 and Summer 2004 semester he taught the undergraduate EE301 Signals and Systems and EE302P probabilistic Methods in Electrical and Computer Engineering courses.

During his Ph.D. studies he has worked at Sharp Labs of America, Camas, Washington in the summer of 1999 and IBM Almaden Research Center, Almaden, California in the summer of 2000. At Sharp Labs, he worked with Dr. Richard Qian and Dr. Ibrahim Sezan on text analysis algorithms to automatically fetch and filter business news from the Internet and investigated the use of such a system for a prospective home audio-visual system. This work resulted in a U.S. patent. At IBM Almaden, he worked with Dr. Arnon Amir and other members of the IBM CueVideo team to develop video summarization techniques for rapid navigation in large video databases and integrated this system into the CueVideo system for video management.

Since Fall 2000, Cuneyt Taskiran has been funded by the C-SPAN Archives, located in West Lafayette, Indiana. He has developed video analysis and indexing tools for the Archives during this time. These tools are currently being used by the Archives.

His research interests include context-based multimedia retrieval, video understanding and analysis using stochastic models, pattren recognition, and statistical text analysis for natural language processing applications.

Cuneyt Taskiran is a member of Eta Kappa Nu (HKN), the National Electrical Engineering Honor Society, and a student member of the IEEE.

Cuneyt Taskiran's publications for his research at Purdue include

Journal Articles

1. C. Taskiran, J. Y. Chen, A. Albiol, L. Torres, C. A. Bouman, and E. J. Delp, "ViBE: A Compressed Video Database Structured for Active Browsing and Search," *IEEE Transactions on Multimedia*, vol. 6, no. 1, pp. 103-118, February 2004.
2. C. Taskiran, Z. Pizlo, A. Amir, D. Ponceleon, and E. Delp, "Video Summarization Using Speech Transcripts," to be submitted to *IEEE Transactions on Multimedia*.
3. C. Taskiran, I. Pollak, C. Bouman, and E. Delp, "Stochastic Models of Video Structure for Program Genre Detection," to be submitted to *IEEE Transactions on Circuits and Systems for Video Technology*.

Book Chapter

C. Taskiran, and E. Delp, "Summarization of Video Sequences," *Digital Image Sequence Processing*, T. Reed, ed., CRC Press, June 2004

Conference Papers

1. M. Karahan, U. Topkara, C. Taskiran, E. Lin, M. Atallah, and E. Delp, "A Hierarchical Protocol for Increasing the Stealthiness of Steganographic Methods," *Proceedings of the ACM Multimedia and Security Workshop*, Magdeburg, Germany, September 20-21, 2004.
2. C. Taskiran, A. Albiol, L. Torres, and E. Delp, "Detection of Unique People in News Programs Using Multimodal Shot Clustering," *Proceedings of the IEEE*

International Conference on Image Processing (ICIP), October 24-27, 2004, Singapore.

3. C. Taskiran, A. Martone, R. Browning, and E. Delp, "A Toolset for Broadcast Automation for the C-SPAN Networks," *Proceedings of the 5th International Workshop on Image Analysis for Multimedia Interactive Services*, April 21-23, 2004, Lisboa, Portugal.

4. A. Martone, C. Taskiran, and E. Delp, "Automated Closed-Captioning Using Text Alignment," *Proceedings of the SPIE Conference on Storage and Retrieval for Multimedia Databases 2004*, January 18-22, 2004, San Jose, CA.

5. C. Taskiran, I. Pollak, C. Bouman and E. Delp, "Stochastic Models of Video Structure for Program Genre Detection," *Proceedings of the International Conference on Visual Content Processing and Representation (VLBV'03)*, September 18-19, 2003, Madrid, Spain.

6. C. Taskiran, A. Amir, D. Ponceleon, and E. J. Delp, "Automated Video Summarization Using Speech Transcripts," *Proceedings of the SPIE Conference on Storage and Retrieval for Media Databases 2002*, January 20-25, 2002, San Jose, CA.

7. C. Taskiran and E. J. Delp, "A Study on the Distribution of Shot Lengths for Video Analysis," *Proceedings of the SPIE Conference on Storage and Retrieval for Media Databases 2002*, January 20-25, 2002, San Jose, CA.

8. C. Taskiran, C. A. Bouman, and E. J. Delp, "Discovering Video Structure Using the Pseudo-Semantic Trace," *Proceedings of the SPIE Conference on Storage and Retrieval for Media Databases 2001*, vol. 4315, pp. 571-578, January 24-26, 2001, San Jose, CA.

9. C. Taskiran, C. A. Bouman, and E. J. Delp, "The ViBE Video Database System: An Update and Further Studies," *Proceedings of the SPIE Conference on Storage and Retrieval for Media Databases 2000*, vol. 3972, pp. 199-207, January 26-28, 2000, San Jose, CA.

10. C. Taskiran, J. Y. Chen, C. A. Bouman and E. J. Delp, "A Compressed Video Database Structured for Active Browsing and Search," *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, October 4-7, 1998, Chicago, IL.

11. C. Taskiran and E. J. Delp, "Video Scene Change Detection Using the Generalized Sequence Trace," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 12-15, 1998, Seattle, WA.

12. J.-Y. Chen, C. Taskiran, E. J. Delp, and C.A. Bouman, "ViBE: A New Paradigm for Video Database Browsing and Search," *Proceedings of the 1998 IEEE Workshop on Content-Based Access of Image and Video Databases*, June 21, 1998, Santa Barbara, CA.