CONTENT-BASED TEMPORAL PROCESSING OF VIDEO

Robert A. Joyce

A DISSERTATION
PRESENTED TO THE FACULTY
OF PRINCETON UNIVERSITY
IN CANDIDACY FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE
BY THE DEPARTMENT OF
ELECTRICAL ENGINEERING

November 2002

Abstract

Multimedia information is most often stored, browsed, and transmitted as simply "raw" data, a set of opaque files. Digital video and audio in particular benefit tremendously from "content-aware" processing; as the salient content information is often temporal in nature, we study both the extraction and applications of the temporal structure of media streams.

We begin by examining some of the fundamental issues behind and goals of automated temporal processing. From there, the problem of gradual transition detection in video is explored, and we present methods to detect both dissolve and wipe-based transitions, even in the presence of special graphical effects. Combining video transition detection with neural network-based predictors, we apply the principles of content-aware processing to improve the channel multiplexing efficiency of variable bit rate video streams.

The integration of video, audio, and other data is essential to any temporal analysis of media streams. Segmentation in these modalities, as well as distance metrics between segments of the same stream, are developed. We examine issues in comparing distance metrics of different modalities, and develop a normalization scheme that takes into account both the distance metrics' statistics and prior probabilities on perceptual segment distances.

Using this distance information, we construct a matrix-based representation that allows quick identification of "idiomatic" sequences, such as dialog or character introductions, in both audio and video. This representation also has a graphical interpretation, which allows the use of shortest-path and similar algorithms, and can associate related but visually dissimilar segments by crossing the boundary between audio and video. Such a graph is itself a useful visualization tool, as it can show transitive connections between segments that would not otherwise be clear. Using detected idiomatic sequences and other criteria, we generate a hierarchy of such graphs, which allows a user to zoom in on sections of interest without being presented with hundreds of segments at once.

Acknowledgements

This thesis would not have been written were it not for the help of numerous people throughout the past half-decade. First and foremost, I am indebted to Bede Liu, whose advice, help, and enthusiasm have been unrelenting. I very much appreciate the freedom he has given to explore new ideas and avenues of research, and he has been wonderfully willing to discuss whatever off-the-wall idea, or unlikely descriptive analogy, I've come up with. Both as a teaching assistant and advisee, he as been a joy to work for.

I am similarly grateful to Wayne Wolf and Bradley Dickinson, whose advice, comments, and questions during both the thesis proposal and the writing of this document have been vital to the finished product.

Min Wu's contributions to the work on video bandwidth prediction were significant, both in terms of ideas as well as Matlab code. Discussions with Min, as well as S.-Y. Kung, Ling Guan, and Hau-San Wong, shaped the direction of our work in this area. Dr. Wong developed the neural network-based feature selection algorithm detailed in Section 3.2.3. The ideas presented here have also drawn significant inspiration from the work of Dr. Liu's previous students and others, including but certainly not limited to Minerva Young, Boon-Lock Yeo, and Heather Yu. Finally, the insights of academic siblings Peng Yin, Minghui Xia, and Scott Craver have been a near-constant source of ideas and refinements.

The State of New Jersey, the National Science Foundation, Intel Corporation, and the New Jersey Center for Multimedia Research have all generously supported our research through my years at Princeton. The U.S. Army Research Office contributed significantly to early portions of this work.

The tireless Jay Plett deserves many thanks, not only for computing support, but for friendship, sage advice, and the opportunity to learn from a guru. His insights, both technical and otherwise, will be missed. Thanks also to Stephanie Eggers and Karen Williams

for administrative support and smoothing over the bureaucratic process.

Julio Concha, Rich Radke, Brian Irvine, Norm Hess, Paul Zimmons, Matt Tangvald, Abby Bechtel, Nature, and Mom and Dad: were it not for their support, inspiration, and love, I'd have never gotten to this point.

Contents

Abstract **i**

Acknowledgements **ii**

1 Introduction **1**

2 Gradual Transition Detection **9**

2.1 Compressed-Domain Processing Preliminaries 11

2.2 Frame-Space Dissolve Detection 12

2.3 Histogram Space Wipe Detection 18

2.4 Analysis of the Correlation Statistic 25

2.5 Experimental Results 34

2.5.1 Dissolve Detection with Simple Detector 34

2.5.2 Dissolve Detection via Parametric Detector 35

2.5.3 Wipe Detection 40

3 Content Analysis for Traffic Prediction **42**

3.1 Bandwidth Renegotiation Points 45

3.2 Traffic Prediction per Interval 46

3.2.1 Media Stream Traffic Descriptors 48

3.2.2 Content Features 49

3.2.3 Feature Selection for Traffic Prediction 53

3.2.4 Consistency-Based Feature Selection 57

3.3 Experimental Results 60

3.3.1 Prediction MSE 60

3.3.2 Trace-Driven Link Utilization 63

4 Multimodal Processing **67**

4.1 Existing Audio/Video Techniques 68

4.2 Speaker Segmentation and Distance Metrics 70

4.3 Video Shot Distance Metric 74

4.4 Audio/Video Distance Normalization 75

5 Association Matrices **86**

5.1 Prior Work 86

5.2 Association Matrix Construction 88

5.3 A/V Association Matrices 89

5.4 Idiomatic Sequence Detection 99

5.4.1 Detection of Prototype Sequences 102

5.4.2 Generation of Idiomatic Sequences' Prototype Matrices 107

5.4.3 Cross-modality Idiomatic Sequences 114

5.4.4 Detector Experimental Results 116

6 Structure via Multimodal Processing **119**

6.1 Prior Temporal Structure Work 120

6.2 Association Graphs 121

6.2.1 Shortest Paths 122

6.2.2 Transitive Path Existence 123

6.3 Memory-Based Graphs and Matrices 124

6.4 Inferring Plot Threads 126

7 Visualization of Multimodal Structure **130**

7.1 Plotting a Single Graph 132

7.1.1 Horizontal Placement 132

7.1.2 Vertical Placement 134

7.1.3 Drawing the Graph 135

7.2 Node Ranking 137

7.3 Hierarchical Graphing 141

7.4 Hierarchical Graph Examples 143

8 Summary and Conclusions **154**

Bibliography **159**

Index **174**

List of Tables

3.1 Candidate content and traffic features for use in per-interval traffic prediction 52

3.2 MSE traffic prediction results, comparing features 62

3.3 MSE traffic prediction results, comparing the SFS/GRNN and consistency-based approaches 62

5.1 Detection and false alarm rates for the idiomatic sequences 117

7.1 Segment rank assignment for hierarchical graphing 140

List of Figures

2.1 Three-dimensional representation of a video sequence f in frame space during a dissolve . . . 14

2.2 The DFD-based correlation sequence $\rho_{dfd}(k, L)$ for a segment of documentary video . . . 17

2.3 Sample wipe sequences from network television, showing the wide variation possible . . . 19

2.4 The sequence $\rho_{hist}(k, L)$ for a segment of news video containing two wipes . 22

2.5 Triplet correlation values ρ_{dfd} for dissolve and non-dissolve segments . . . 26

2.6 Triplet histogram correlation values ρ_{hist} for wipe and non-wipe segments . 27

2.7 ρ_{dfd} "noise" after the dissolve indicator signal $s(k)$ is subtracted . . . 28

2.8 Distribution of dissolve transition lengths, measured from 95 dissolves . . . 29

2.9 Dissolve likelihood function $\mathcal{L}(l)$ for a sample sequence . . . 32

2.10 Distribution of $\mathcal{L}(l)$ values for dissolve and non-dissolve segments of 13 minutes of video . . . 33

2.11 ROC plot for the parametric dissolve detection method on a 13 minute television video sequence . . . 37

2.12 recall $\times$ precision for the parametric dissolve detector, over a range of $T_{\mathcal{L}}$ and T_{dist} . . . 38

2.13 ROC plot for our wipe detection algorithm over 23.5 minutes of video, including some synthetic wipes . . . 41

3.1 Traffic prediction scenarios with different delays 47
3.2 Neural network based traffic prediction, using both short-term traffic observations and content features to determine the entire shot's traffic patterns . 48
3.3 Error plot for SFS/GRNN feature selection and corresponding feature subsets 56
3.4 Four traffic classes derived by K-mean clustering on the two principal components of D-BIND, the first step in consistency-based feature selection . . 58
3.5 Sorted consistency measures for each candidate content feature, when used individually 59
3.6 Overall structure of the VBR resource predictor 60
3.7 Traffic prediction MSE using different intervals and predictor inputs 64
3.8 Network utilization for multiplexed sources, computed by trace-driven simulation 66

4.1 Empirical distributions for the cepstral mean audio shot distance metric . . 73
4.2 Empirical distributions for the two-key-frame video shot distance metric . . 76
4.3 Prior probabilities of "same," "similar," and "different" speaker segments as a function of the number of segments separating them 79
4.4 Prior probabilities of "same," "similar," and "different" video shots as a function of the number of shots separating them 80
4.5 Minimum-cost thresholds dividing measured audio distance values into three subjective regimes, "same," "similar," or "different," as a function of the number of audio segments separating the two test segments 82
4.6 Minimum-cost thresholds dividing measured video distance values into three subjective regimes, "same," "similar," or "different," as a function of the number of video shots separating the two test shots 83

5.1 Perceptually normalized distance matrices for a 7 minute Charlie Rose interview clip 93

5.2 Perceptually normalized distance matrices for the 7 minute Charlie Rose interview clip, with element widths proportional to segment lengths 94
5.3 Charlie Rose audio-audio and video-video time-normalized distance matrices superimposed over one another 95
5.4 Perceptually normalized distance matrices for a 7.5 minute CBS news clip . 96
5.5 Superimposed audio-audio and video-video time-normalized distance matrices for a 7.5 minute CBS news clip 97
5.6 Superimposed audio-audio and video-video time-normalized distance matrices for a 71 second segment from NBC's sitcom "Frasier" 98
5.7 Perceptually normalized distance matrices for the first 10 minutes of CBS's "The Late Show with David Letterman" 100
5.8 Superimposed audio-audio and video-video time-normalized distance matrices for the first 10 minutes of CBS's "The Late Show with David Letterman" 101

6.1 Schematic graph of our plot thread model 127

7.1 Plot of the memory-based graph for the first 65 seconds of the "Charlie Rose" clip 137
7.2 Plot of the memory-based graph for the entire "Late Show with David Letterman" clip 138
7.3 Hierarchical browsing interface example 142
7.4 "Charlie Rose" top level graph 144
7.5 "Charlie Rose" level 2 graph 144
7.6 "Charlie Rose" level 3 graph 145
7.7 "Charlie Rose" base level graph 145
7.8 "The Late Show with David Letterman" level 2 graph 146
7.9 "The Late Show with David Letterman" level 3 graph 147
7.10 "The Late Show with David Letterman" level 4 graph 147

7.11 “Frasier” level 2 graph 149
7.12 “Frasier” level 3 graph 149
7.13 “Frasier” base level graph 150
7.14 CBS local news level 2 graph 151
7.15 CBS local news level 3 graph 152
7.16 CBS local news level 4 graph 152
7.17 CBS local news base level graph 153
7.18 Entire CBS local news base level graph 153

Chapter 1

Introduction

Video and audio have long been a dominant means of communication, and the quantity of media in the digital domain is increasing at a rapid pace thanks to the wide availability of video digitizing hardware and editing equipment. Transmission, storage, and management of large quantities of video benefit from—and in many cases, require—knowledge of the "content" of the media at some level above that of a raw data stream.

The definition of "content" is highly application-dependent. For example, a news video library requires the annotation of important speakers and locations, as well as identification of topic changes and specialty segments, such as weather or health reports. If the library is to be used for browsing as well as keyword searches, as is the case for films and entertainment television, there must be some method for extracting a summary or preview that contains important highlights yet is easily understandable. Databases and search systems require that extracted information be stored in a compact, standardized format.

Consumer applications, while perhaps less demanding than commercial video libraries, are beset by their own set of issues: equipment must be inexpensive yet deal with possibly vague queries, video may be unedited (such as home movies) or in an otherwise unconventional organization, and summary information must be presented in an intuitive fashion, comprehensible without training. These requirements hem in the types of analysis that can be performed; for instance, it may be difficult to identify principal characters or locations,

and common editors' patterns will not be present to aid in segmentation.

Efficient transmission of multimedia streams is yet another venue for the use of content information. Knowledge of low-level content information, such as when a shot or scene change occurs, is useful in bandwidth allocation schemes. Internetwork routers can also benefit from content-aware processing, by selectively dropping less important packets or routing critical ones over more reliable paths. Depending on the memory available to the client, temporal content information can yield more efficient compression, allowing selected frames to be saved and used much later as reference images. Higher-level semantic information allows "important" spatial or temporal information to be transmitted with higher fidelity than ancillary or background content.

There are a number of commonalities in the applications of content analysis:

- **Automated extraction:** Manual annotation of large audio and video collections, in either the analog or digital domains, requires an immense investment in manpower to perform. Large news organizations can have so much incoming video that continuous manual annotation is nearly impractical. Live media sources are even more difficult to handle; automatic annotation is often the only solution. In situations where automatic annotation cannot capture the desired information, algorithms are often available that will assist in manual annotation by finding segments likely to contain items of interest.

- **Standardized annotation:** In archiving and search applications, as well as in transmission applications where there are multiple client architectures, it is vital to agree upon an application-independent representation of content information. The emerging MPEG-7 standard provides a method of codifying extracted information in an XML-like format [1, 2]. The standard will not specify how information is to be extracted, nor precisely what information must be available for a given stream.

- **Temporal information dominates:** The information to be extracted is often temporal in nature, whether it be due to segmentation, association of distant frames or

scenes, or extraction of salient fragments from the stream. Even when some of the desired information is non-temporal, it must first be determined which frames or audio intervals to use as the basis for any static detection or classification algorithm.

The dominance of temporal information forms the basis of this thesis; in the creation of indexing structures, annotations, and summaries, temporal relationships are precisely what differentiate image collections from audio and video databases.

A necessary first step in any temporal processing is to segment the media stream into more manageable chunks, typically according to "natural" temporal boundaries occurring throughout the stream. As audio and video segments are seldom precisely aligned in time, and at this stage little further content information is available, we segment audio and video independently to maintain generality.

Video streams are typically divided into semantic segments called "scenes," which are further subdivided into "shots" [3, 4, 5]. A video *shot* is defined as a single continuous camera action, from the start of recording to a subsequent stop on the same camera [6]. (The prevalence of computer-based graphic effects often blurs the definition of a shot boundary; to be precise, we consider any boundary-like action causing a dramatic transition of more than 50 percent of the image area to be a shot boundary.) A contiguous collection of related shots, usually taking place in the same location, form a *scene*.

Shot and scene segmentation of video have been long-studied, particularly when the segment transitions are abrupt (such as cuts). In Chapter 2, we develop methods of segmenting video when the transitions are gradual in nature, whether they be due to dissolves, wipes, or more complex graphical effects. When combined with a conventional cut detector, these techniques form a comprehensive video segmentor.

Similar shot and scene definitions can be constructed for audio data, complicated somewhat by the fact that audio is sometimes edited in a manner that belies its underlying structure. For example, background music can blend two "segments" together, or inserted

sound effects can falsely split segments. Nonetheless, we consider audio "shots" to be continuous spoken segments by a single speaker, or isolated musical selections. As in video, contiguous sets of related audio shots form "scenes," such as dialog. (These notions are somewhat distinct from the field of "auditory scene analysis," which more often deals with overlapping source separation in complex aural environments [7, 8].)

Due to the fact that typical audio streams contain many varying, superimposed sound sources, reliable audio segmentation has been an elusive goal. We visit some of the issues and tradeoffs involved in Chapter 4, as audio segmentation is necessary for true multi-media content extraction.

Once a reliable segmentation system is established, straightforward applications can be explored. One such application is in improving the efficiency of transmitting variable bit rate (VBR) compressed video. Situations requiring bandwidth estimation, such as statistical multiplexing and metered/scheduled channels, traditionally deal with VBR video inefficiently: bandwidth reservations for the VBR peak rate are wasteful, while those for the average rate will likely result in dropped packets during complex segments. In Chapter 3, we apply both video segmentation and mid-level content feature measurement to the problem of predicting future VBR bandwidth requirements. This prediction method outperforms those based solely on bit rate statistics, allowing greater overall channel utilization and fewer unnecessarily dropped packets.

While video data is the dominant source of information in VBR bandwidth prediction, joint video and audio content is of fundamental importance to indexing and browsing engines. Chapter 4 discusses some of the issues in merging information from multiple modalities. A central issue is how to make meaningful comparisons between measured statistics of audio and video, such as the similarity of two segments. We develop a normalization scheme that allows distance metrics within any modality (and where applicable, between sources) to be used on equal footing in later processing.

We introduce in Chapter 5 a method of compactly representing distance information

from multiple modalities in a media stream. While useful to an expert as a structural visualization method, we also focus on its applications in finding higher-level "idiomatic" events in conventionally-edited media, such as dialog and character introductions. The remaining chapters all use information directly available via this representation, which we call an "association matrix."

The primitives of shot and scene are vital in segmenting a media sequence, and are reasonable atomic units for indexing, retrieval, and transmission applications, but they do not always capture the complete temporal structure of a sequence. In addition, some media genres contain shots that cannot be sensibly collected into scenes in the traditional sense.

Obtaining intra- and cross-modality distance information allows for the exploitation of transitive distances: it is possible to associate two video segments, for instance, that are superficially dissimilar. Such methods suggest a graph-based interpretation of the combined distance information, which we use in Chapter 6 to extract relationships between distinct shots and scenes and to correlate audio and video segmentation data. Long-term transitive associations are then used in an attempt to capture semantic threads of plot. Even when such a high-level interpretation is not possible, transitive associations are still useful in generating summaries and in flagging important events, such as when two formerly distinct paths merge.

Finally, we apply the content extraction tools we have developed to the generation of hierarchical summaries of multimedia streams in Chapter 7. Hierarchical summaries are vital as stream length increases, as it is impossible to capture all the information a browsing user may wish to know in a single screen. We instead allow the user to "zoom in" on segments of interest, panning around as necessary, in a map-like paradigm. Segments are presented to the user according to a ranking determined by a number of factors, including any idiomatic sequences they belong to, where the segments occur, and their duration.

In extracting content information in its various forms from a media stream, there are a number of goals we keep in mind. Not all can be adhered to in all instances, but our

methods are developed with the following as priorities:

Generality

Automated processing systems for audio and video streams must strike a balance between generality and the degree to which real "content" information can be extracted. Algorithms that are applicable to large classes of streams, from dramatic films to sports to newscasts to home movies, can at best make heuristic guesses as to what extracted low-level information means in terms of "content," at least in any sense to which humans can relate. Likewise, it is possible to infer reliable high-level information if one is restricted to very narrow domains of video, such as recordings of specific sporting events in specific styles, or news broadcasts on particular television channels [9, 10, 11].

Our approach in this thesis is provide tools that are general, in the sense that they extract mid-level information that is useful independent of the particular domain or genre of video being analyzed. Domain-specific knowledge can be applied at higher levels of analysis to infer further information, given the mid-level information we've extracted. Such knowledge can also be used to improve our mid-level algorithms (for example, having a model of the general temporal structure of a certain news program can help in segmentation), but that is left as an implementation issue. The details of any particular domain or genre are intentionally avoided in our algorithms.

Causality

An important goal for any media processing system is causality: any information extracted on-line must depend only on audio and video data in the past. As delay is often tolerable, this requirement can be relaxed a bit in some cases; a small, fixed amount of future data may be used if the output decisions are delayed by an equal amount. Causality is particularly crucial for live sources, long-duration material, or media sources being digitized on-the-fly from analog storage. Restricting ourselves to causal processing also reins in the on-line

storage capacity needed by the algorithms while they are running. Finally, causality can prevent some possible deadlock situations when jointly processing audio and video streams.

Complexity

Naturally, computational requirements (CPU, memory) are an important consideration given the sheer size of raw video and audio data streams. Live sources require processing to be faster than real time, and even off-line but on-the-fly digitization benefits greatly from real-time processing because of the specialized analog playback hardware that would otherwise be needed.

When video and audio analysis algorithms are used on media servers, either in an on-line capacity or in building off-line databases for later use, the fact that many streams will be analyzed simultaneously must be taken into consideration. A streaming server performing on-line traffic prediction, for example, might be doing so for dozens of distinct streams and network conditions at the same time.

Finally, low complexity is advantageous because there is often much more processing left to do beyond what will be described in this thesis; even algorithms that operate in real time may not be fast enough when the whole system is taken into consideration.

Tests with Real-World Data

For any video and audio analysis algorithms to be widely useful, they must be tested "in the wild," with streams of the type likely to be encountered. While instructive, it is not enough to only test with artificially-constructed streams having specific properties to exercise certain aspects of an algorithm. In the detection of gradual transitions in video, for example, the gamut of transition types in common use eclipses any test set one could reasonably create. In addition, transitions seen in film or television video are likely not as precise or noise-free as those constructed in the lab.

To the extent possible, the tools also should be robust to common distortions that occur

in real-world video streams, such as multiple A/D and D/A conversions, transmission over the air in NTSC, and playback through consumer VHS video recorders.

With these priorities in mind, we proceed to the first step in the temporal processing of multimedia streams: the segmentation of video.

Chapter 2

Gradual Transition Detection

As we saw in the last chapter, the accurate division of a media stream into meaningful constituent segments is a vital prerequisite to temporal processing. The detection of abrupt transitions ("cuts") between video shots has been extensively studied in both the compressed and uncompressed domains. Gradual transitions, which are more likely to mark scene boundaries than are cuts, pose a much more difficult problem. Such transitions can be roughly divided into two classes: those that simultaneously but gradually affect every pixel of the image, and those that abruptly affect an evolving subset of the pixels, with the subset changing in each frame. Over a number of frames, the cumulative change—due to the summed gradual changes or to the union of pixel subsets—forms the gradual shot boundary.

The first class includes dissolve and fade-in/out effects, and one could argue that dissolves and fades are the only members of this class. Much work has been done on dissolve and fade detection, particularly with the use of reduced-resolution frames and motion vectors gathered directly from the compressed stream [12]–[21]. Comparison studies have been conducted by Boreczky, Lienhart, and Gargi, among others [22, 23, 24].

What are commonly thought of as wipe effects are members of the second class, although for brevity the term "wipe" will herein be used to describe any transition abruptly affecting an evolving subset of pixels. Wipes are often utilized in television news and sports coverage,

as well as in movies. In sports video, for example, wipes are generally used to denote the beginning and end of an instant replay; thus, detection of wipes would allow an indexing system to separate replays from live action, thereby preserving continuity in time. During newscasts, wipe transitions often signal a change in story or topic.

Qualitatively, wipe transitions are generally characterized by the slow sliding in or uncovering of an image from a new shot, while simultaneously covering up or sliding out the old shot. At any instant during the transition, the frame contains some of the old content as well as some of the new. The "edge" of the wipe—the moving spatial boundary between the old and new shots—can be a single line or a complex pattern. Multiple boundaries may also be present within the same transition. Recently there has been a trend toward using blurred wipe boundaries; attempting to detect the exact wipe edge can be difficult. Often, the transition is generated by computer, in which case three-dimensional projections or other special effects may be present. On occasion, computer-generated artwork will completely cover the image, creating an intermediate step in the wipe's progression.

One common method of wipe detection involves extraction and counting of edges in the image; this statistic will monotonically change during a transition, from the old shot's value to the new shot's value [25, 26, 27]. This generally must be performed on uncompressed video, and is computationally expensive. In the compressed domain, methods have been proposed that analyze a projection or subset of the DC DCT coefficients, looking for progressions of abrupt pixel changes [28, 29, 30]. Progressions of changes in encoder motion-prediction decisions have also been used, as have progressions in partial-frame histogram intersections [31, 32]. A method has been proposed by which the statistical characteristics of wipe sequences are detected [33]. Clustering algorithms in reduced-dimension pixel and histogram spaces have been used to detect transitions as well [34]. Kobla, *et al.*, combined pixel and histogram-based distance metrics after excluding sections of the video containing significant motion [35]. Finally, Fernando, *et al.*, used the Hough transform on spatially-reduced frames to detect and characterize the style of certain types of wipes [36]. The

majority of these algorithms depend on the video producer using only a limited amount of computer graphics or artwork, and assume little motion adjacent to and during the wipe. With the prevalence of computer-generated wipes, the sharp boundaries and simple one-directional wipe models are likely to fail on modern video; what is needed is a more general method, independent of the direction or style of wipe, and robust to any reasonable amount of producer-added effects (for instance, blurring, page-turning, shadows, and projections).

Some preliminary results of the work described here were presented in [37] and [38].

2.1 Compressed-Domain Processing Preliminaries

Ideally, the segmentation process would be done in real-time, either from a live feed or a single pass of videotape. However, the computation time required to decode compressed video and perform image-processing operations on full frames, while decreasing with progress in processor design, remains significant. This complexity constraint becomes even more troublesome when considering that shot and scene decomposition are only the first steps of the indexing process; there is much yet to do. In addition, most sizable digital video libraries are likely to be in compressed form, if only to save space. For these reasons, analyzing video streams directly in the compressed domain is advantageous.

For concreteness, we will focus on MPEG-1 video in this chapter, but the algorithms presented apply equally well to other block/transform-based video compression schemes [39].

One natural technique of compressed-domain analysis is reduced-resolution processing: using a subset of the block transform coefficients to reconstruct thumbnail-sized images. Of particular interest is the construction of so-called "DC frames," which are comprised of the lowest-order DCT coefficients of each MPEG block (and are therefore one sixty-fourth the size of the full frames). For intracoded (I) frames, construction of DC frames is trivial. Intercoded (P,B) frames require full decompression of their reference frames for exact DC reconstruction. Instead, rapid first-order estimation techniques are used to construct DC

frames for intercoded compressed frames [16]. If computation time is very critical, a slight speedup can be gained by resorting to simpler zero-order estimation techniques; the negative impact on the final results is small. Similar methods can be used to construct DC+2AC frames, which are formed from the DC and two lowest-order AC coefficients of each block. The 2×2 IDCT then required for each block is simple to compute.

Aside from their computational advantages, DC sequences are more suitable for video analysis in many respects. Primarily, the artifacts of MPEG compression and video noise are significantly reduced at the lower resolution. In addition, small amounts of camera or object motion, which dramatically affect the registration of adjacent frames' pixels at the full-frame level, are obscured at such a low resolution.

Displaced frame differences ("DFD's"), which are the pixel-by-pixel differences between frames after any motion compensation, can be computed for P frames without full decompression. DC DFD's require no computation at all, as they are just the lowest-order DCT coefficients of the residue frame, which are available directly in the coded stream. Other reduced-resolution DFD's can be computed via low-order inverse DCT's.

Unless otherwise noted, first-order estimated DC sequences are used in all calculations for the remainder of the thesis. In MPEG-1 sequences, the DC frames are typically 44×30 pixels in size.

2.2 Frame-Space Dissolve Detection

At its most basic, a dissolve or fade is a time-varying superposition of two video streams. Let $f_k(x, y)$ denote the value of pixel (x, y) in frame k of sequence f, with $g_k(x, y)$ and $h_k(x, y)$ defined similarly. A dissolve from sequence g to sequence h, lasting from frame m to frame n, can therefore be described by

$$f_k(x, y) = \alpha_k h_k(x, y) + (1 - \alpha_k)\, g_k(x, y) \tag{2.1}$$

where α_k is an increasing sequence, with $\alpha_m = 0$ at the beginning of the dissolve and $\alpha_n = 1$ at the end. It is often assumed that the sequence α_k increases linearly, but this is not necessarily the case; particularly artistic dissolves may have a pause, a long lead-in time, or some other non-linearity in α_k.

For the moment, we assume there is negligible motion in the sequences g and h. For compactness, we denote by f_k the vector formed by all the pixels of frame k (the ordering is irrelevant, as long as it is consistent). With color video, each pixel has three dimensions in color space; f_k then contains three times as many elements as there are pixels in a frame. Consider the trajectories formed by $f_b - f_a$ and $f_d - f_c$, where $m < a < b < n$ and $m < c < d < n$. Substituting the model in (2.1) yields

$$f_d - f_c = (\alpha_d - \alpha_c)(\alpha_b - \alpha_a)^{-1}[f_b - f_a] \tag{2.2}$$

during a dissolve. As α_k is an increasing sequence, $(\alpha_d - \alpha_c)(\alpha_b - \alpha_a)^{-1} > 0$. This condition is equivalent to the statement that, during a dissolve, the normalized correlation, ρ, between any two trajectory vectors is 1. If one considers each vector f_k as being in a frame space, then the video's trajectory in this space will be a straight line during a dissolve, as shown in Figure 2.1. Natural, non-dissolve motion in a stream generally does not have this characteristic; it is uncommon for all the pixels in the image to evolve in the same way, frame after frame. Note that linearity in frame space is distinct from the condition that α_k increases linearly; we make no such assumption about the time progression of the dissolve.

In order to check this condition, we are faced with four concerns: limited memory (we cannot store all the frames), limited computation time, no *a priori* knowledge of the start or end of the dissolve, and the fact that there may be some object or camera motion in the frame. Analysis of three nearby frames at a time offers a good compromise among these considerations. Using frames $k - L$, k, and $k + L$, we can compute two length-L frame differences, where

$$d_k^L(x, y) = f_k(x, y) - f_{k-L}(x, y) \quad \forall x, y \tag{2.3}$$

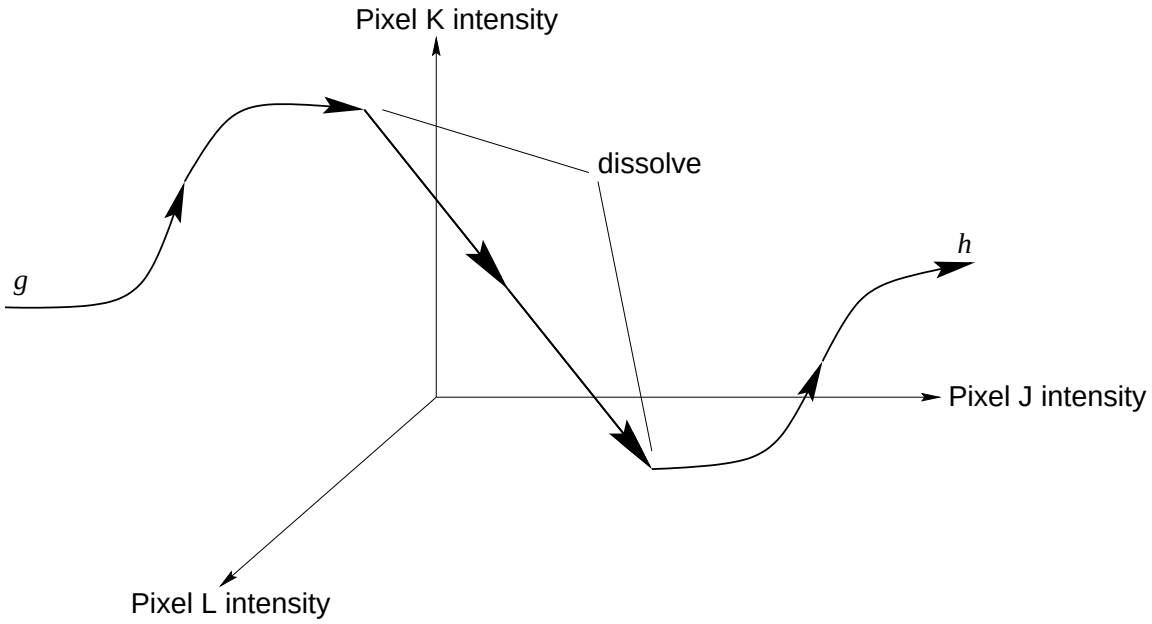


Figure 2.1: Three-dimensional representation of a video sequence f in frame space during a dissolve between sequences g and h.

is the k-th difference frame, and d_k^L is the corresponding vector in frame space. The correlation, as a function of k and L, is then

$$\rho(k,L) = \frac{\left\langle d_{k+L}^L, d_k^L \right\rangle}{\sqrt{\left\| d_{k+L}^L \right\|^2 \left\| d_k^L \right\|^2}}, \tag{2.4}$$

where $\langle \cdot, \cdot \rangle$ represents an inner product. A "straight" triplet of frames is declared if the the correlation is high enough, i.e., if

$$\rho(k,L) \geq T_{corr} \tag{2.5}$$

for an appropriate threshold T_{corr}.

In order to declare a dissolve, we require that condition (2.5) hold for every k in some sequence of frames, say from m to n. A condition on the length of this line in frame space is also needed; we require that

$$\| f_n - f_m \| \geq T_{dist}. \tag{2.6}$$

The length condition is necessary because small changes (e.g., in frame brightness) can

lead to the correlation condition being met for an isolated triplet or two. Instead of (2.6), a simpler threshold on the length in number of frames can be used, but the frame-space length condition is more robust to eliminating false detections.

The testing of (2.5) and (2.6) can be done sequentially, with no knowledge of future frames beyond $k + L$, according to the following algorithm:

```
while ( k + L ≤ total number of frames )
    if ( ρ(k, L) ≥ T_corr )
        n = k + L
        if ( m not yet set )
            m = k − L
    else if ( m is set )
        if ( ‖f_n − f_m‖ ≥ T_dist )
            declare dissolve
        unset m
    end
    k = k + L
end
```

Regarding the selection of L, we note that the effects of motion diminish as $L \to 1$, but decreasing L leads to more false alarms, as it is possible to construct a long non-straight line in frame space which has local correlations near 1. As L is increased, computational requirements are lessened, but it becomes more likely that outliers (from a straight line) will be obscured by the coarse granularity of sampled frames. Selecting $L = 3$, which means only the I and P frames in many MPEG-1 streams, provides a resonable compromise: slow motion is not destructive, and the computation time and number of false alarms are both reasonable.

While many dissolves do indeed have little motion, this is not universally true; any

rapid object or camera motion during the transition will prevent the frame-space linearity condition from holding. (Local linearity might still hold though, if L is small.) Simple object or camera motion can be compensated for by using DFD's (Section 2.1), instead of the true frame differences, in (2.4). In addition, much computation is eliminated, due to the ease of extracting (DC) DFD's. Unfortunately, this does place a dependency on how the particular MPEG encoder was designed; to maintain some consistency among computed correlation values, we restrict analysis to only P frames. (This requires us to ignore two triplets per MPEG group of pictures (GOP), namely the PPI and PIP, because one of the two required DFD's cannot be reliably computed in each case.) If we denote the DFD between frame f_k and frame f_{k-L} as $\tilde{d}_k^L$, the correlation calculation in (2.4) becomes

$$\rho_{dfd}(k,L) = \frac{\left\langle \tilde{d}_{k+L}^L, \tilde{d}_k^L \right\rangle}{\sqrt{\left\| \tilde{d}_{k+L}^L \right\|^2 \left\| \tilde{d}_k^L \right\|^2}}. \tag{2.7}$$

A plot of this ρ_{dfd} sequence for a sample documentary clip with two dissolves is shown in Figure 2.2; note the sharp increase during the dissolve frames.

Values for T_{corr} and T_{dist} should be set based on the desired false alarm rate and detection accuracy. In many cases, false alarms are not as detrimental as missed events in shot decomposition; detection accuracy can be improved if some false alarms are allowed. As the values of the frame-space correlations can depend on non-content-related factors such as frame size, video noise, and compression artifacts, the mean of the past M values of $\rho_{dfd}(k,L)$ is subtracted before the T_{corr} comparison is made. (This is equivalent to gently high-pass filtering the ρ_{dfd} sequence.) More thorough post-processing of the correlation sequence is detailed in Section 2.4, and specific experimental results are presented in Section 2.5.

The algorithm's computational requirements can be lessened, on the other hand, by sacrificing some detection probability. Roughly one third of the computation time can be saved by using only the luminance space in DFD extraction and ρ_{dfd} calculation. In addition, a significant number of multiplications can be eliminated by assuming $\left\| d_{k+L}^L \right\| \approx \left\| d_k^L \right\|$ during a dissolve; aside from the analysis to be described in Section 2.4, this is equivalent

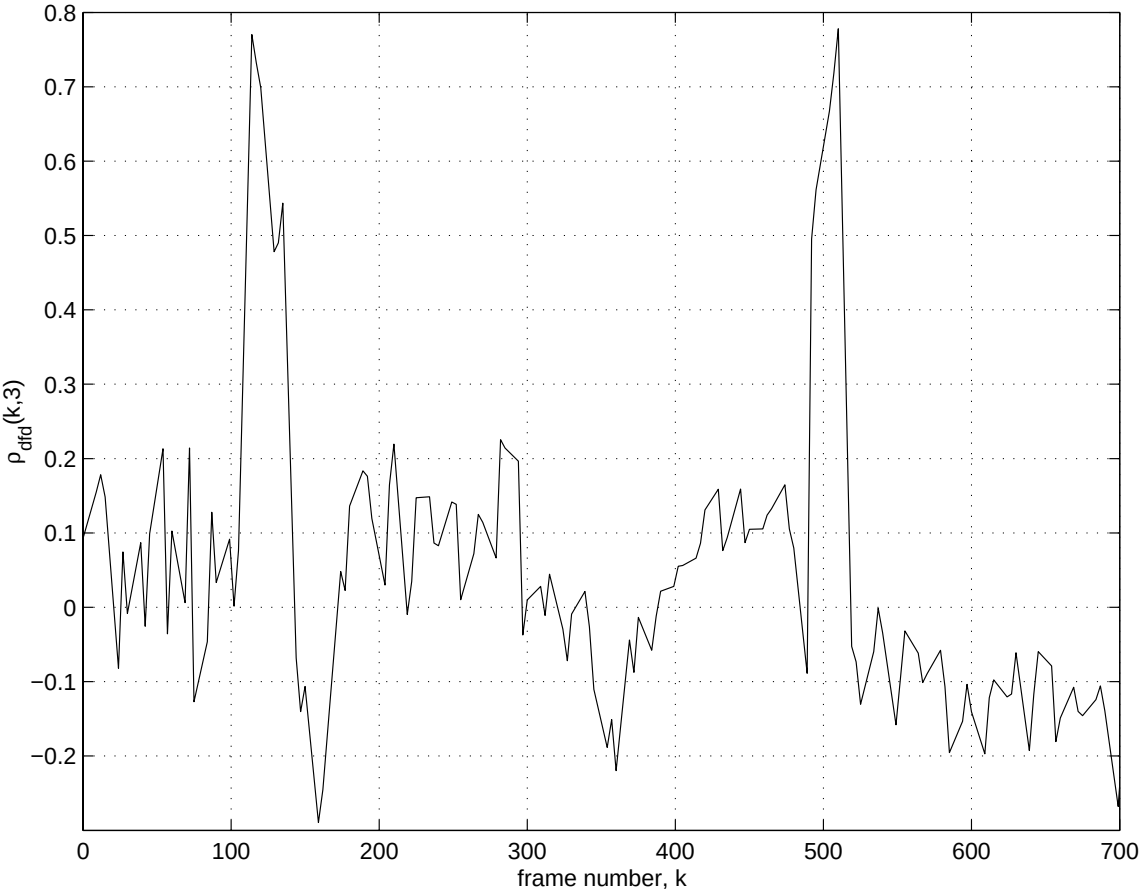


Figure 2.2: The DFD-based correlation sequence $\rho_{dfd}(k, L)$ for a segment of documentary video, with $L = 3$; dissolves occur during frames 115–140 and 492–516.

to the detection method of Yeo, *et al.*, and similar schemes [16]. This assumption limits the detector to finding dissolves with linear α_k sequences. With the same limitation, all non-trivial multiplications can be eliminated by computing, for each triplet, the raw difference between the center frame and the pixel-wise mean of the outer frames.

The dissolve detectors described here can be easily extended to the detection of partial-frame dissolves, such as caption appearances or split-screen transitions. In this application, ρ_{dfd} is calculated for the region of interest, or absent this knowledge, for a set of blocks within the image. ρ_{dfd} may be calculated for each pixel in the extreme case, yielding dissolve "frames" that show the degree to which each pixel is within a dissolve sequence.

2.3 Histogram Space Wipe Detection

One can imagine many effects in which an evolving subset of pixels changes abruptly in each frame. The simplest wipes are those in which one sequence gradually covers or replaces another, with no global movement of either sequence. More complicated wipes can involve one stream "sliding" in over another, or one "pushing" another aside. "Zoom" based wipes can also be created in this manner, with a new stream appearing from the center of the old one, expanding to fill the whole frame. Finally, complex computer-generated wipes can include page-turning effects, projections, or artistic wipe boundaries; for a few examples, see Figure 2.3. One or more frames may not even contain content from either adjacent shot; this is particularly common in sports video, where a large computer-generated object passes across the field of view to effect a transition using two back-to-back wipes. Due to the broad range of gradual transitions that fall within the wipe class, a detection method tailored to a specific wipe is likely to miss many other kinds of wipes.

As in the dissolve case, we assume a wipe transition from sequence g to sequence h, from frame m to frame n. A simple, overlap-based wipe can be described as

$$f_k(x,y) = I_k(x,y)h_k(x,y) + [1 - I_k(x,y)]\, g_k(x,y) \tag{2.8}$$

Figure 2.3: Sample wipe sequences from network television, showing the wide variation possible.

where f_k is the resulting frame k, and $I_k(x, y)$ is either 0 or 1 for each k, x, and y. $I_k(x, y) = 0$ for all x and y when $k < m$ (before the wipe), and $I_k(x, y) = 1$ when $k > n$ (after the wipe). In the case where one or both sequences slide in or out of the frame, (2.8) becomes

$$\begin{aligned} f_k(x, y) = \; & I_k(x, y) h_k \left(x + x_{h,k}, y + y_{h,k}\right) \\ & + \left[1 - I_k(x, y)\right] g_k \left(x + x_{g,k}, y + y_{g,k}\right) \end{aligned} \tag{2.9}$$

where the wipe-induced motion of the sequences is described by $x_{g,k}$, $y_{g,k}$, $x_{h,k}$, and $y_{h,k}$. Even these two models are more restrictive than one would like; they preclude the detection of many artistic wipes, for example. Natural object motion in video typically fits these models as well, yielding only limited usefulness. The important information of each model is that contained in the sequence $I_k(x, y)$; as such, we will concentrate on $\|I_k\|$. This sequence should increase from 0 to N, the number of pixels in the image, as k increases from m to n. For most wipes, $\|I_k\|$ will increase linearly or quadratically.

One representation of a video stream that allows us to examine the $\|I_k\|$ sequence, without the restrictions of specific wipe models, is the histogram. We denote the p-th bin of frame f_k's histogram as $F_k(p)$ (the number of bins, P, is a free parameter); we will use the same vector shorthand of F_k, for some arbitrary ordering of bins. Assuming for the moment that each frame's histogram is fairly uniform across different portions of the image, the histograms during a wipe can be expressed as

$$F_k(p) = \left(\frac{\|I_k\| + E_{G,k}(p)}{N}\right) G_k(p) + \left(1 - \frac{\|I_k\| + E_{H,k}(p)}{N}\right) H_k(p), \tag{2.10}$$

where $E_{G,k}(p)$ and $E_{H,k}(p)$ are error terms resulting from the spatial non-uniformity in the histograms of g and h, respectively. Note that this histogram-based wipe model has the same form as the frame-space model (2.1) for a dissolve! If the values of $E_{G,k}$ and $E_{H,k}$ are small and fairly constant in k, it also meets the conditions we imposed on the coefficients α_k from the dissolve case. Specifically, the quantity

$$\beta_k = \frac{\|I_k\| + E_{G,k}(p)}{N} \tag{2.11}$$

will be increasing in k from 0 to 1, and

$$1 - \frac{\|I_k\| + E_{H,k}(p)}{N} \approx 1 - \beta_k. \tag{2.12}$$

Such a parallel immediately suggests a wipe detection algorithm. As in the dissolve case, the correlation between any two histogram difference vectors ($F_b - F_a$ and $F_d - F_c$) will be 1 during an ideal wipe. Moreover, a wipe will appear as a straight line in a histogram space, where each dimension corresponds to one bin of the histogram. (This linearity is independent of any nonlinearity in the time progression of the wipe.) In the same manner as the dissolve case, we define the L-frame histogram difference $D_k^L(p)$ as

$$D_k^L(p) = F_k(p) - F_{k-L}(p) \quad \forall p. \tag{2.13}$$

We compute the correlation sequentially, from triplets of frames:

$$\rho_{hist}(k, L) = \frac{\left\langle D_{k+L}^L, D_k^L \right\rangle}{\sqrt{\left\| D_{k+L}^L \right\|^2 \left\| D_k^L \right\|^2}}. \tag{2.14}$$

This value is compared to a threshold, and following the algorithm presented for the dissolve detector on page 15, the value of (2.6) is computed to determine the length of the candidate wipe; a wipe transition is declared if both thresholds are met. Note that a condition similar to (2.6) could be computed in the histogram space; we have not done so, due to the unwanted constraint this imposes that the two adjacent shots must have sufficiently different histograms. The sequence ρ_{hist} for a sample stream is given in Figure 2.4.

As in the dissolve case, T_{corr} and T_{dist} should be chosen to achieve the desired balance between detection probability and false alarm rate. Once again, to counter the fact that the mean value of ρ_{hist} is somewhat dependent on the type of video and the recording/compression quality, the mean of the last M values is subtracted before thresholding against T_{corr}.

Any natural change in g or h's histograms through time (due to motion or other effects) introduces a deviation from $\rho_{hist} = 1$ in the same manner as motion in g or h did in the

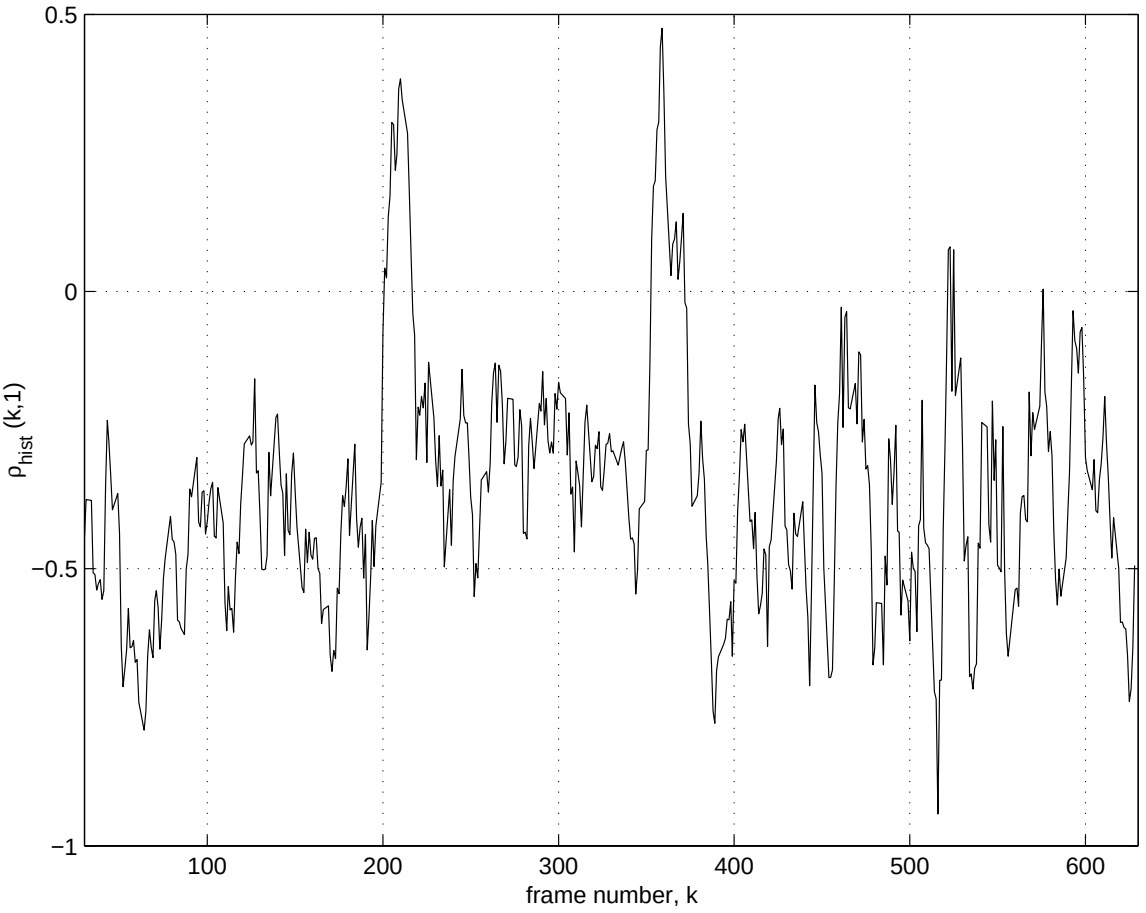


Figure 2.4: The sequence $\rho_{hist}(k, L)$ for a segment of news video containing two wipes. $L = 1$ in this case, and wipes occur in frames 197–209 and 352–364.

dissolve case. As L is decreased, such effects decrease as well, and $G_k - G_{k-L} \rightarrow 0$ and $H_k - H_{k-L} \rightarrow 0$ because the g and h histograms are very unlikely to change abruptly (except in the case of a scene cut).

In addition, as L is decreased, the effect of errors terms $E_{G,k}$ and $E_{H,k}$ in (2.10) diminishes. This is due to the use of triplets: the deviation from spatial uniformity in the histogram is only important in the region that actually changes over the frame range $k - L$ to $k + L$. In a wipe, the size of this region (a vertical slice of the image, for example) vanishes as $L \rightarrow 0$. For these reasons, we set $L = 1$ from here on; this agrees with experimental results obtained by varying L. In cases where the histogram non-uniformity is caused by an object's edge entering or exiting the region of significance, the effect on ρ_{hist} will be impulsive—the straight line in histogram space will now be piecewise linear, with some small number of vertices.

Equation (2.10) makes a computational assumption: the number of pixels in any histogram must be an integer, yet the coefficient β_k may be such that the equation requires a non-integral number of pixels in a particular bin of F_k. This quantization error, if significant, can reduce the correlation among the adjacent pair of vectors in a triplet. The error can be reduced by using fewer histogram bins, as well as by increasing the spatial resolution at which one operates (using a low resolution or a large number of bins would force very small quantities of pixels into many bins, making any quantization errors in the intermediate frame of a triplet more significant). For this reason, we perform the histograms on DC+2AC frames and use 2 to 4 histogram bins per color dimension.

Better characterizing these data-dependent quantization and histogram non-uniformity errors remains an open problem, but their effects on ρ_{hist} can be reduced by low-pass filtering or otherwise post-processing the resulting correlation sequence (under the assumption that the errors in $\rho_{hist}(k, L)$ are approximately independent in k). The generally impulsive errors due to $E_{G,k}$ and $E_{H,k}$ suggest the use of a median filter (or more generally, an n^{th}-largest filter), which while nonlinear, does have a sufficiently low-pass characteristic to help with the

quantization noise. As an added benefit, low-pass filtering helps alleviate the time-varying histogram distortions that MPEG compression and video noise can introduce.

One issue has not yet been addressed: can natural motion in video have the linear histogram-space characteristic? Pathologically structured object motion into or out of a frame can cause a straight line in the histogram-space, as can panning the camera if the image contents and histograms change radically during the pan. Experimentally, the number of false alarms attributed to object motion has been shown to be fairly small in natural video, provided the image histogram does not change radically during the pan. False detections due to panning can only be eliminated at the expense of missing "push" type wipes (which are arguably a type of panning). This can be done by computing the temporal variance of each macroblock's motion vectors—low variance corresponds to constant motion in some direction through time. Specifically, using P frame motion vectors during the candidate wipe, we calculate the temporal variance of each macroblock's X motion vector plus the temporal variance of its Y motion vector. If some fraction (T_{MF}) of the macroblocks have a variance sum greater than a threshold T_{MV}, then the candidate wipe is confirmed. If T_{MV} is not met for a sufficient number of macroblocks, a wipe is not declared. False alarms could be further reduced by adding additional constraints; one example is requiring the ratio of $\left\|D_k^L\right\|$ to $\left\|D_{k+L}^L\right\|$ to be either constant or linear.

In practice, we find that dissolves are often falsely detected as wipes by our algorithm. A dissolve does not have the linearity property in histogram space; rather, the histogram of the old shot is progressively shifted, bin by bin, toward all pixels being in the "black" bin; the new shot is correspondingly shifted bin-wise from black to its final histogram. However, if the shot histograms are fairly continuous from bin to bin (i.e., there are no spikes in particular bins, with adjacent bins nearly empty), then the dissolve can masquerade as a linear change in histogram space. This is particularly problematic when the number of bins is small; spikes are very unlikely when there are only a few bins. The simplest solution is to cascade the transition detectors: only try to detect wipes in areas previously declared

not to be dissolves. While this introduces a two-level detection dependency—dissolve false alarms will contribute to wipe misses—the results are greatly improved.

2.4 Analysis of the Correlation Statistic

Given that the detection algorithms introduced in Sections 2.2 and 2.3 are so similar once the correlation statistics are computed, it is useful to study the ρ sequences' statistics. Any information gained can be used to derive a more optimal (yet computationally expensive) detector.

Figure 2.5 shows the distributions of ρ_{dfd} values for 13 minutes of video, separated into dissolve and non-dissolve segments (after filtering out cuts); Figure 2.6 is the wipe case. The overlap in densities is not as bothersome as it might appear, because detection is done on variable-length sets of frames (using the algorithm presented in Section 2.2), not on individual triplets.

One interpretation of the distributions, particularly those of ρ_{dfd}, is that of a signal+noise detection problem where the signal of interest (denoted $s(k)$) is a binary indicator of whether there is a transition during the triplet. More precisely, during transitions we set $s(k)$ to be the mean value of the correlation statistic over all transition sequences, and outside of transitions we set $s(k)$ to be the mean ρ over all non-transition triplets. (In the wipe case, the clipping of the values to ± 1 causes the noise distribution to not be independent of $s(k)$; this could be alleviated by more sophisticated detection-theoretic techniques.) If we subtract $s(k)$ from $\rho_{dfd}(k, L)$, we find the measurement error due to motion, compression, etc., is nearly an ideal Gaussian process with sample variance 0.0387 (Figure 2.7). The process $n(k)$, where $n(k) = \rho_{dfd}(k, L) - s(k)$, is not white noise—its power spectrum is tilted toward DC—but is fairly independent of $s(k)$. (The coarseness of the Figure 2.5's distribution within transition regions is due to the relatively small number of triplets comprising it.)

The length of the transition is unknown during the detection process, yet we would

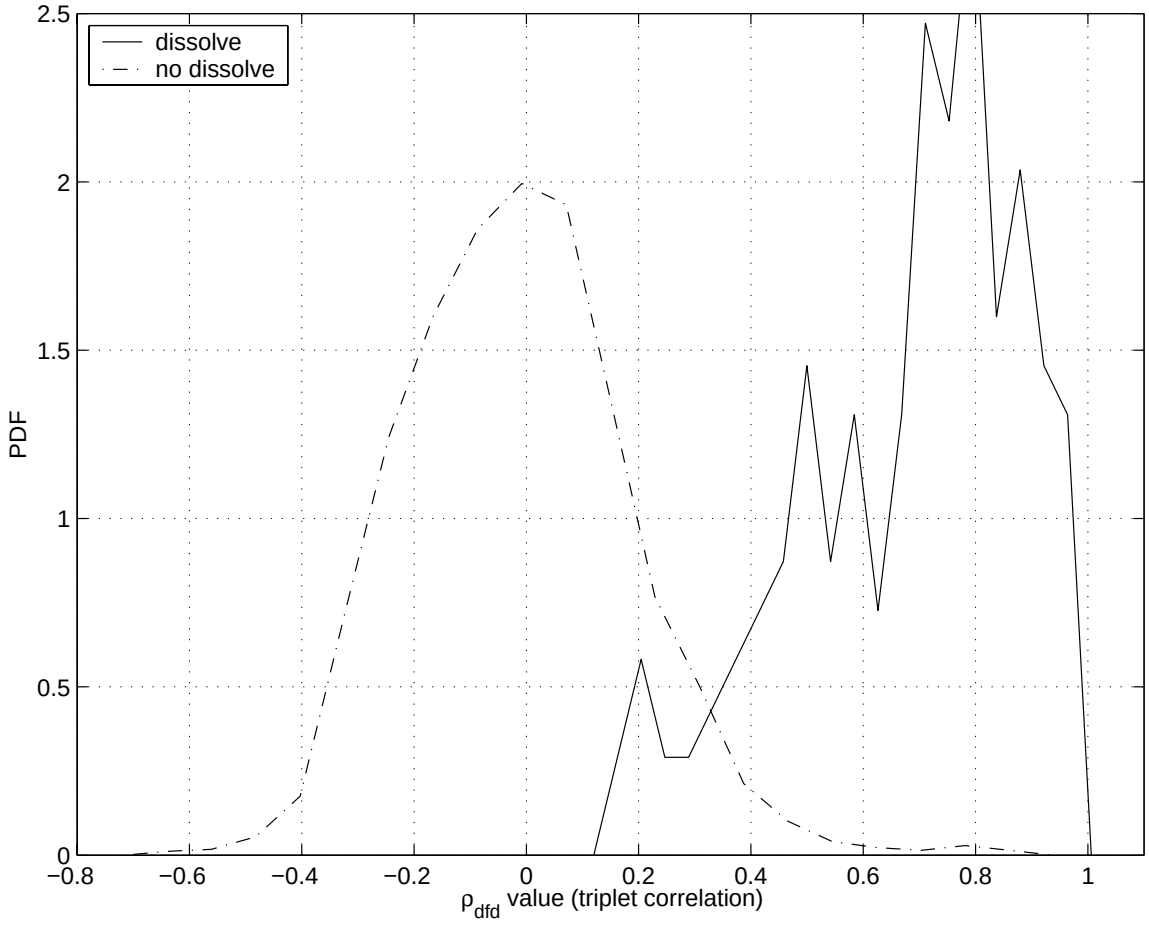


Figure 2.5: Triplet correlation values ρ_{dfd} for dissolve (solid line) and non-dissolve (dashed line) segments. The sample variance for the dissolve segments is 0.0406; for the non-dissolve segments, 0.0387.

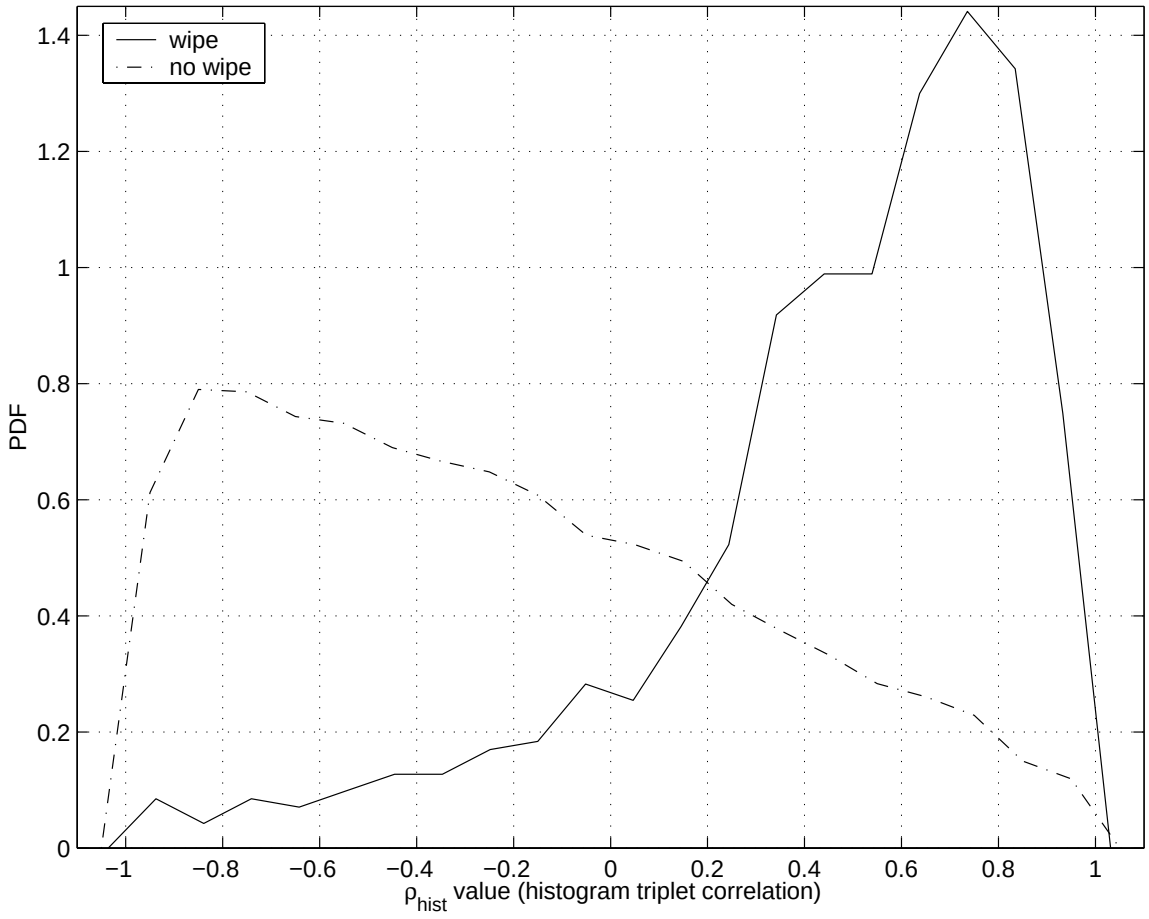


Figure 2.6: Triplet histogram correlation values ρ_{hist} for wipe (solid line) and non-wipe (dashed line) segments. The sample variance for the wipe segments is 0.1386; for the non-wipe segments, 0.2175.

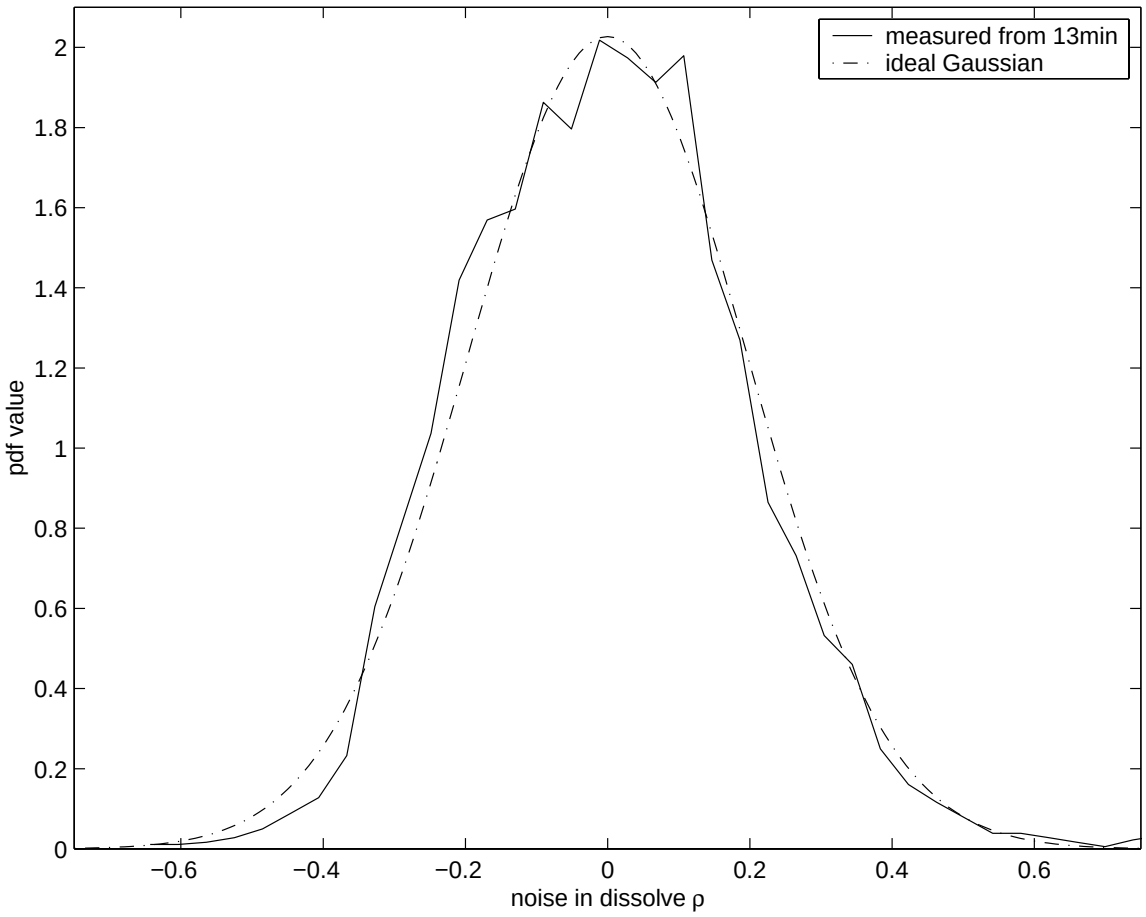


Figure 2.7: ρ_{dfd} "noise" after the dissolve indicator signal $s(k)$ is subtracted.

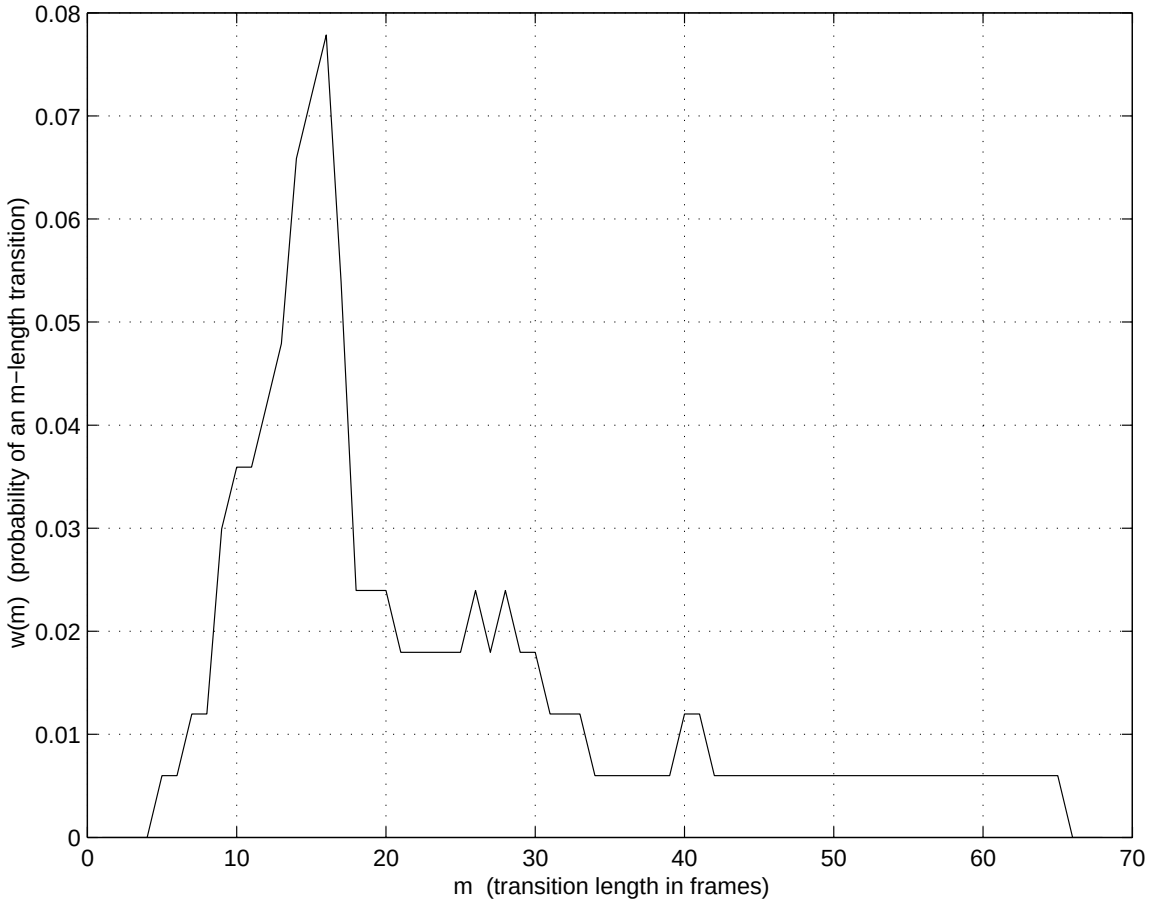


Figure 2.8: Distribution of dissolve transition lengths, measured from 95 dissolves. Some values are interpolated from neighboring samples.

like to take advantage of the interdependence of the $s(k)$ values; this can be done using a parametric detector that averages over some given distribution of test signals. In this case, the parameter is the transition length; an estimated distribution of dissolve lengths, measured from 95 transitions, is shown in Figure 2.8. We call this PMF $w(m)$. Note that, even within the framework of the simple detector presented in Section 2.2, this distribution could be used to eliminate inordinately short or long false detections.

Optimum parametric detectors in non-i.i.d. Gaussian noise are well known [40]. Such detectors by necessity work on blocks of input data, where the block in our case must be

longer than the support of the parameter's PMF. Denote this block length K; $w(m) = 0$ for $m > K$. In order to make each block's noise statistics consistent, we require that blocks begin at a compressed group-of-pictures (GOP) boundary; otherwise, we have a non-stationary noise as the block start shifts through one GOP. Block l therefore contains frames lt through $lt + K - 1$, where t is the length of a GOP.

For simplicity, all time indices that follow are given in terms of frames, whereas the detector's actual calculations need to be done in terms of triplets. Once the GOP structure of a stream is known, converting frame- to triplet-distances is straightforward. Also note that in the dissolve case, where we have $L = 3$ and use DFD's, one must account for the two skipped triplets (PPI and PIP) per GOP when calculating transition lengths; such issues are ignored in the following.

We begin by setting μ_0 to be the (estimated) mean of the ρ_{dfd} values when no transition is present; μ_1 is the mean during dissolves and fades. Denote by $R(l)$ the column vector of measurements formed by $\rho_{dfd}(k, L) - \mu_0$, where $lt < k < lt + K - 1$. We then construct a parameterized set of K-length test signal vectors, $S(p, q)$, $0 \leq p \leq t - 1$ and $1 \leq q \leq K - p$, where again t is the length of a GOP:

$$S(p, q) = [\underbrace{0 \cdots 0}_{p} \underbrace{\mu \cdots \mu}_{q} 0 \cdots 0]^T \tag{2.15}$$

where $\mu = \mu_1 - \mu_0$. Essentially, S contains every possible transition length of interest, with starting points anywhere within the first GOP (starting points within later GOP's will be detected in subsequent blocks). The stationary block-based problem can then expressed as $R(l) = N(l) + S(p, q)$ for some p and q. The density $w(m)$ must mapped into $W(p, q)$ according to the length of the transition tested in $S(p, q)$, giving[1] $W(p, q) = w(q)$.

As the noise sequence $n(k)$ is not i.i.d., neither is the noise vector $N(l)$, so both the received signal blocks $R(l)$ and the test signal vectors $S(p, q)$ must be pre-whitened; denote the whitened vectors $\bar{R}(l)$ and $\bar{S}(p, q)$, respectively. The whitening can be done by

[1] If certain triplets are skipped, such as the GOP boundary triplets mentioned in Section 2.2, this equation must be modified to account for the non-constant time increments that depend on p.

estimating the covariance matrix Σ_N and multiplying by one of its Cholesky factors [40].

Given the set of $\bar{S}$ vectors, the block PMF $W(p,q)$, and the pre-whitening matrix (C^{-1}), the transition likelihood function can be calculated for each block of correlations $R(l)$ as follows:

$$\mathcal{L}(l) = \sum_{p,q} W(p,q) \exp\left[\left(\bar{S}^T(p,q) C^{-1} R(l)\right) - \frac{1}{2}\left(\bar{S}^T(p,q)\bar{S}(p,q)\right)\right] \tag{2.16}$$

A sample plot of $\mathcal{L}(l)$ in Figure 2.9 shows that this approach extracts dissolves quite well from the ρ_{dfd} sequence. A histogram of $\mathcal{L}(l)$ for dissolve and non-dissolve segments is shown in Figure 2.10 (note the log scale); when compared to Figure 2.5, the improvement is clear. As argued in Section 2.2, it is also necessary to test the L^2 length of a candidate dissolve in frame space before declaring a transition. Specifically, a dissolve transition is declared when both the following inequalities hold:

$$log_{10}\mathcal{L}(l) > T_{\mathcal{L}} \qquad \text{and} \qquad \|f_{lt+Q-1} - f_{lt+P}\| \geq T_{dist}, \tag{2.17}$$

where P and Q are the values of p and q corresponding to the largest term in the sum (2.16). The transition in this case begins at frame $lt+P$ and ends at frame $lt+Q-1$. Transitions often span multiple GOP's, so that $\mathcal{L}(l) > T_{\mathcal{L}}$ for two or more consecutive values of l; in such cases, we declare the frames corresponding to the largest $\mathcal{L}(l)$ value as the dissolve transition. Given our "on-then-off" prototype sequences $S(p,q)$, the (locally) largest $\mathcal{L}(l)$ during a dissolve is generally the first one.

This parametric detection structure could also be useful for the more general case of gradual transition detection, where some statistic is computed per frame (or set of frames) and the probability density of transition lengths is known or estimated, provided that the "noise" in measurements is approximately Gaussian and, more crucially, independent of $s(k)$.

As can be seen in Figure 2.6, the "noise" in ρ_{hist} is highly dependent on $s(k)$ (and is not Gaussian); applying the parametric detector unaltered indeed yields poor results. In the absence of tractable independent noise models, the only way to improve detection with

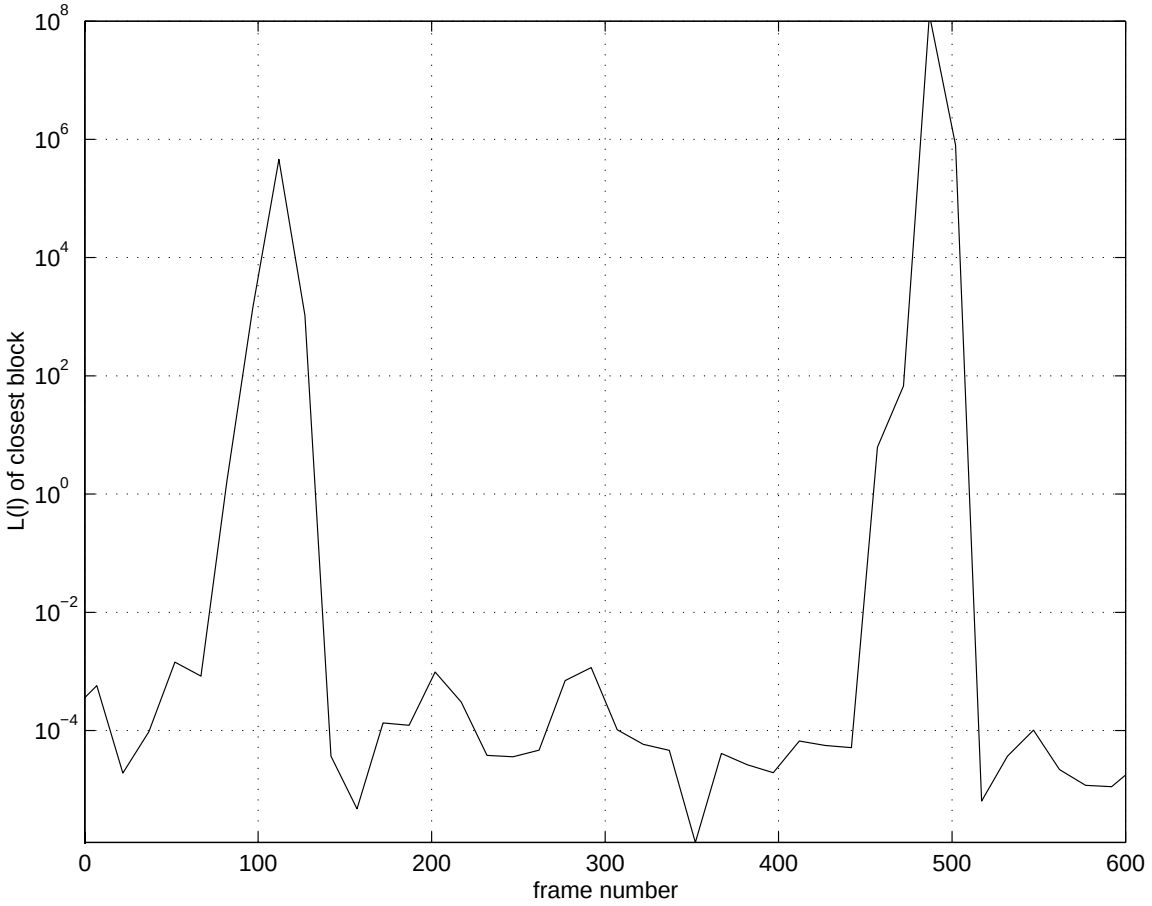


Figure 2.9: Dissolve likelihood function $\mathcal{L}(l)$, expressed in terms of the frame number beginning each block of triplets l; note the logarithmic scale. The sample MPEG stream is the same as in Figure 2.2.

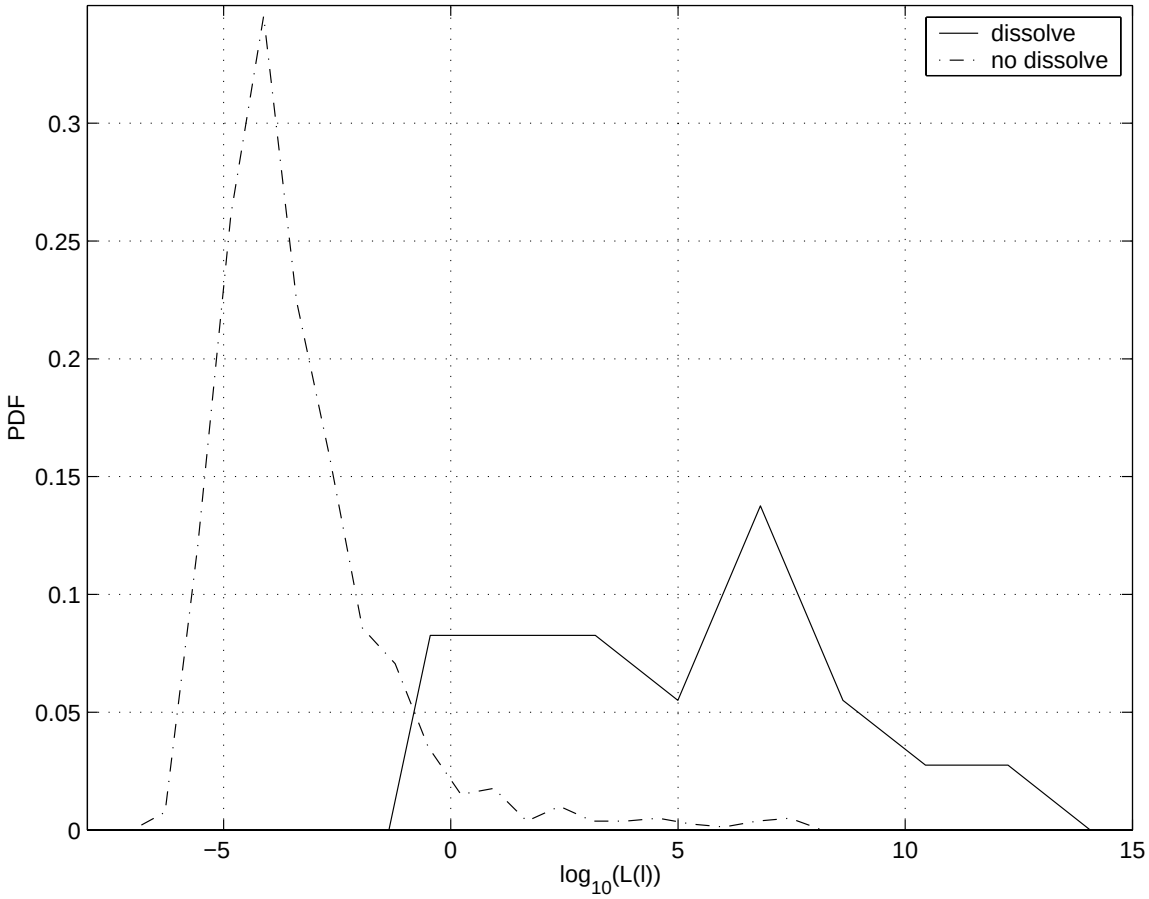


Figure 2.10: Distribution of $\mathcal{L}(l)$ values for dissolve and non-dissolve segments of 13 minutes of video; again note the log scale.

ρ_{hist} is to compare sample densities to the two in Figure 2.6. Such a comparison is likely to be problematic, as the number of samples within a given block is small (and there is no way to refine the temporal resolution beyond the block level). ρ_{hist} could certainly benefit from further analysis, however; it is possible for example that some neural network or adaptive filter structure exists which will improve the results.

2.5 Experimental Results

Each algorithm was tested with "natural" television and film footage, digitized from VHS tape sources with a hardware MPEG-1 encoder. The resulting video quality is hardly perfect, making for a good workout of each transition detector. Each test stream was digitized at a resolution of 352×240 and a frame rate of 29.97 fps. The test sets consisted of news video from different networks, documentary footage and other material. All computation was performed on a 350 MHz Sun workstation.

2.5.1 Dissolve Detection with Simple Detector

A thirteen minute collection of video was used as a "training" set, on which parameter values were selected. The training set's 23700 frames contained 59 dissolves, 115 cuts, and 6 wipes, as well as some significant object and camera motion. Most of the dissolves were clearly visible, but three were between images so similar that a casual human viewer likely would not notice the transition. The dissolves ranged in length from 12 to 65 frames, and a number of the transitions contained motion of some sort.

Testing yielded good results with $T_{corr} = 0.15$, after the mean of the past 125 values was subtracted. (Depending on the video, the effective T_{corr} was between 0.4 and 0.8.) $T_{dist} = 55000$ (normalized to the number of macroblocks per frame) and at least 3 successive above-threshold triplets were required in order to declare a dissolve. With these values, 52 out of the 59 dissolves were properly detected, with 24 false alarms (a rate of one per 2633

frames). Different thresholds can be selected to yield different detection probability versus false alarm tradeoffs. In most cases, the detector correctly identified the locations of the transition start and end to within four frames.

These results confirm that frames pace correlation is a reasonable statistic to use for dissolve detection, and that using DFD's is an effective way to combat the errors in correlation introduced by shot motion. By visual inspection of $\rho_{dfd}(k, L)$ plots, one can generally pick out all of the dissolves (even the ones that are missed); this leads us to believe that a more sophisticated detector could produce better results using the same ρ_{dfd} sequence. Results for such a detector, that described in Section 2.4, are presented in the next section.

Including the overhead due to DC frame extraction, our algorithm processed video at about 170 frames per second. In fact, about 95% of the processing time is spent parsing the MPEG stream and calculating the DC frames; once the DC frame is available, our algorithm takes only an additional 0.3 ms/frame on the test machine. Speed in parsing could likely be improved through better optimization of our partial MPEG decoder.

2.5.2 Dissolve Detection via Parametric Detector

Once the $\rho_{dfd}(k, L)$ sequence is computed, the detector outlined in Section 2.4 can be applied as an alternative to the simple detector. Using the same 23700-frame test sequence, $T_{\mathcal{L}} = 0.1$, and $T_{dist} = 55000$, the parametric detector correctly found 53 out of 56 dissolves, with 12 false alarms[2]. Of the three missed dissolves, one fell at the very end of a GOP and was short enough to not affect any DFD's; another overlapped slightly with a wipe transition. The final missed transition was likely due to unfortunate motion-compensation decisions on the part of the encoder, such that the DFD's did not suggest gradual changes in each pixel. Seven of the 12 false alarms were due to cuts, which could be eliminated if a cascaded "cut-dissolve-wipe" detector were implemented. (As a proof-of-concept, we implemented such a cascaded detector using the cut detector developed by Yeo, *et al.*; six of the seven

[2]The total number of dissolves is 56, not 59 as in the previous test, because 3 of the dissolves occur within K frames of the end of a stream, thus are never tested against the $S(p, q)$ sample vectors.

cut-induced false alarms were indeed eliminated.) Three more of the 12 were "dissolve-like" operations, such as captions or computer-graphic effects fading away. The total number of false alarms, if one discounts those due to cuts and dissolve-like effects, is 2 per 13 minutes of video.

At the expense of a higher false alarm rate, the detection probability can be pushed up to 0.982; 31 false alarms were produced in this case. Disregarding caption fades and cuts, the false alarm count falls to 15 (slightly more than one per minute). Different detection/false alarm tradeoffs are possible; Figure 2.11 shows the raw false alarm rate corresponding to a number of detection probabilities.

An alternative quality measure is to find the maximum achievable $Q = (\text{recall} \times \text{precision})$ value, where recall is the detection probability and precision is the number of correct detections divided by the total number of detections (correct or not). A perfect detector is one with $Q = 1$. (In practical applications, the quantity of false alarms is not as critical as the weighting given to them in the Q measure defined here, but this metric is a reasonable basis for comparison.) Automated detection and false alarm counting techniques were used to iteratively find the maximum Q for the parametric detector; the maximum, 0.8395, occurs when $T_{\mathcal{L}} = 1.5$ and $T_{dist} = 67000$, yielding a detection probability of 0.891 and a false alarm rate of 0.228 per minute. The Q values over a small range of $T_{\mathcal{L}}$ and T_{dist} are shown in Figure 2.12.

The parametric detector was also tested on longer streams, a total of 23 additional minutes; in addition to news and documentary footage, the longer streams contained a number of commercials. Blindly using $T_{\mathcal{L}} = 0.1$ and $T_{dist} = 55000$, 108 out of 149 dissolves were detected, with 36 false alarms (1.57 per minute). Discounting false alarms due to cuts and caption fades, the number drops to 24 (1.04 per minute). If necessary, further processing can be done to shrink the number of false alarms (for example, requiring each triplet's frame space vectors have at least a certain L^2 length during a dissolve, or utilizing other statistical properties dissolve transitions must have). Most of the missed detections occurred during

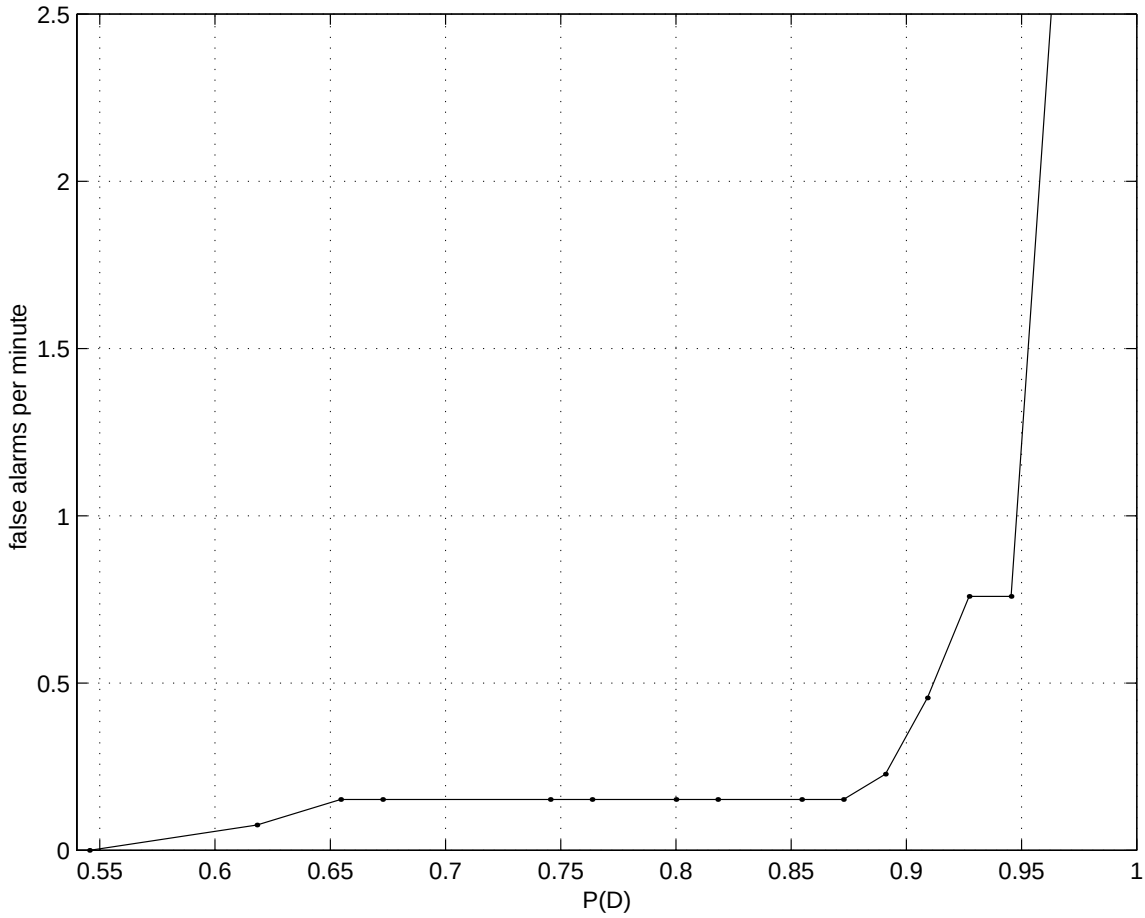


Figure 2.11: ROC plot for the parametric dissolve detection method on a 13 minute (23695 frame) television video sequence. The false alarm rate is the number of false detections divided by the total sequence length; caption fades and similar effects, when detected, were counted as false alarms.

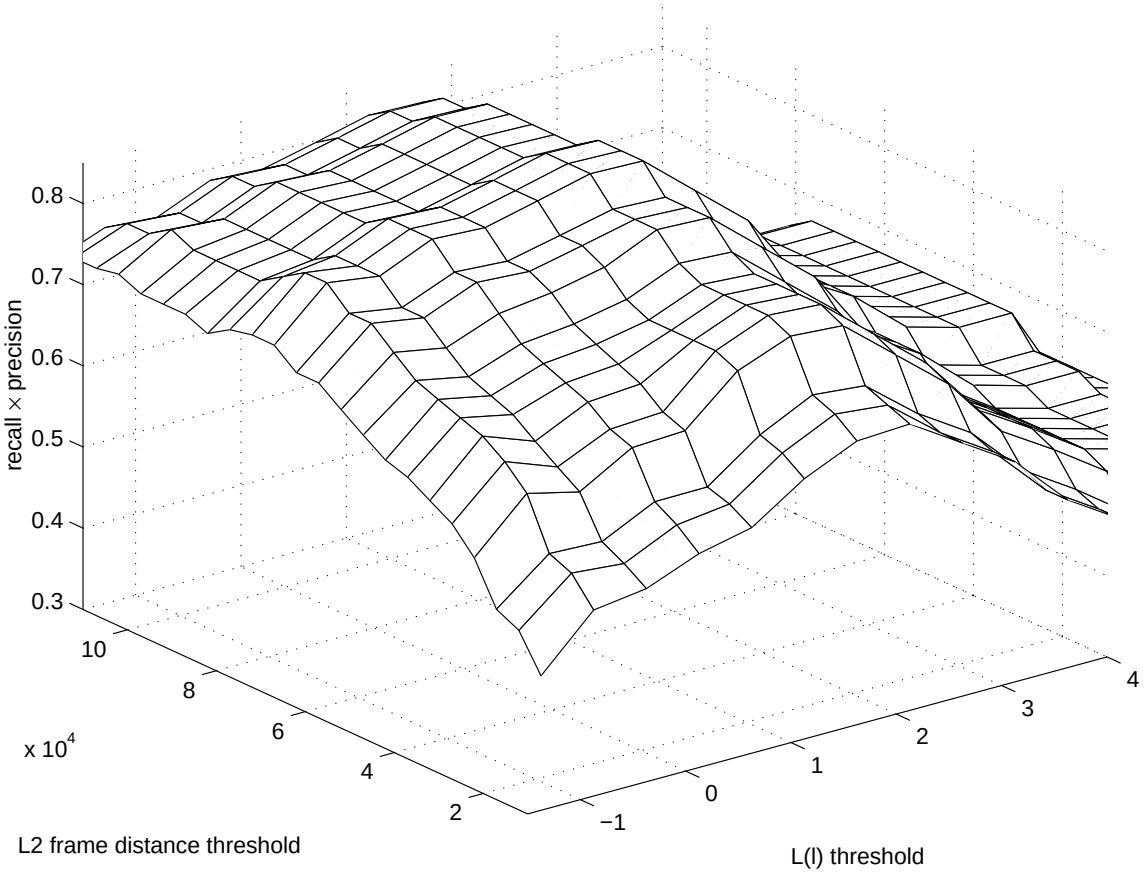


Figure 2.12: recall × precision for the parametric dissolve detector, over a range of $T_{\mathcal{L}}$ and T_{dist}; the maximum occurs at $T_{\mathcal{L}} = 1.5$ and $T_{dist} = 67000$.

the commercial segments, where large numbers of dissolves appeared right after one another (often, with few non-dissolve frames in between). As designed, our parametric detector will not detect more than one transition within K frames, which alone accounts for 10 misses; by adding more complicated 1/0 sequences to the test set $S(p, q)$, these could be detected at some computational expense. (One could make an argument that the short non-dissolve segments in cases like this aren't "shots" in the same sense as those in non-commercial segments.)

In most cases, within each test segment, the false alarms had lower likelihood function values $\mathcal{F}(l)$ than the real transitions (yet still above threshold); very few non-transition regions induced higher likelihood function values than the smallest dissolve segment's value. The cutoff region for $T_{\mathcal{L}}$ varies somewhat by stream, generally according to the content and the MPEG encoder. (This variation is significantly less than the stream-dependent variation in the simple detector's optimal T_{corr}.) One way to both address this issue and to provide the user with more control over the coarseness of the temporal segmentation is to allow presentation-time control of $T_{\mathcal{L}}$. The user would increase the $T_{\mathcal{L}}$ "knob" for coarser segmentation (avoiding caption fades, etc), and decrease it to see the less dramatic gradual transitions (for instance, those involving only a portion of the screen). This operation would require efficient storage of the $\mathcal{L}(l)$ sequence, and fairly rapid threshold testing and presentation, as the threshold would be unknown at the time of analysis.

When the largest term of the sum in (2.16) is examined as suggested at the end of Section 2.4, the temporal accuracy of the derived start and stop points for each dissolve is generally better than 4 frames. Given that the B frames are ignored, it is impossible to be more accurate than roughly 2 frames on average with our test streams.

As most of the computation time is spent extracting DC frames and DFD's, the speed decrease induced by this detector is minimal (on the order of K^2 additional multiply-adds per GOP).

2.5.3 Wipe Detection

In testing the wipe algorithm, the training video set was augmented by short clips with artificial wipes (between TV news shots) created with Adobe Premiere 4.2. Forty test clips, with varying parameters and styles of wipes, were created. The first shot of each clip contained mild object motion, and the second shot of each was a slow zoom; neither shot was motionless during the wipe. The combined length was 42100 frames, or 23.5 minutes.

Sixty out of 62 wipes were detected, with 35 false alarms, when using the parameters $T_{corr} = 0.25$ (after the running mean was subtracted), $T_{dist} = 61000$, and 2 histogram bins per color dimension (for a total of 8 bins). Motion vector temporal variance checking was used, with $T_{MF} = 0.4$ and $T_{MV} = 2.5$, to confirm potential wipes. The misses were mainly due to adjacent shots having very similar histograms: one example is a wipe between two close-up views of a basketball play, having very similar histograms; except for its white boundary, the transition was barely visible to the eye. Most of the false alarms were due to close-up panning during a tennis segment, where the histograms changed wildly.

Using the same parameter set, the algorithm was tested on a feature-length movie containing 28 wipes of varying styles, along with significant motion and special effects. 14 of these wipes were detected, with a false alarm rate of 4.0 per minute. Again, most of the false alarms were due to rapid camera motion in action sequences. The wipes that were not detected were ones with very broad borders, within which the two images were blurred together; in this region, the histograms will not be linearly combined (some of the transitions were in fact midway between a wipe and a dissolve). Naturally, the results will improve if the parameters are tailored to this particular stream, and different detection versus false alarm tradeoffs can be reached.

The wipe algorithm requires about 2.9 ms/frame of computation time; when added to the 18.1 ms/frame required to extract the DC+2AC frames, the algorithm runs at a rate of nearly 48 frames per second. If the dissolve algorithm is cascaded with the wipe algorithm, the overall processing speed is 46 frames per second.

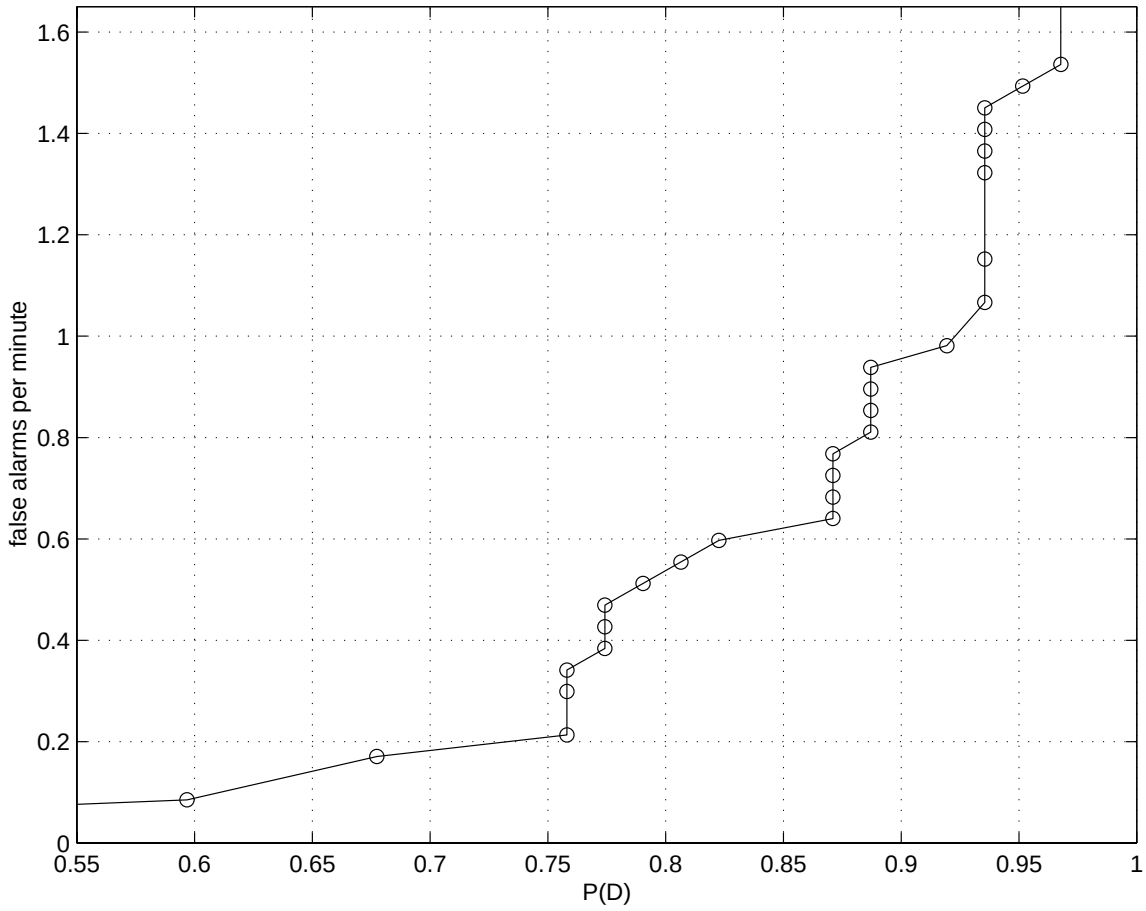


Figure 2.13: ROC plot for our wipe detection algorithm over 23.5 minutes (42147 frames) of video, including some synthetic wipes. The false alarm rate is the number of false detections divided by the sequence length.

Chapter 3

Content Analysis for Traffic Prediction

The transmission of multimedia streams over bandwidth-limited shared networks is a complex problem, as one needs to cope with the ever-changing system parameters: the number of data sources and receivers, the bandwidth required by each source stream, and the topology of the network itself. An optimal resource allocation system must dynamically consider global strategies (network-wide management) as well as local strategies (e.g., access control for individual connections); in the following sections, we focus on the local strategies only.

Bandwidth allocation and management for individual streams generally must be done at the "edges" of the network, in order to conserve computational resources on internetwork switches. Such systems will likely not have complete knowledge of the network state, and must therefore make their use of network resources as minimal as possible to maintain a given quality of service (QoS). If a source requests more bandwidth than it actually uses, the overall network utilization drops. Conversely, if the source exceeds its bandwidth request, packet loss and delay will become significant. While offline systems could compute the exact dynamic bandwidth requirements for a stream before transmitting it, on-line processing is desirable in many applications. Systems such as video conferencing and live news-on-demand absolutely require on-line processing. In addition, on-line processing is needed in any system that dynamically transcodes video, or that splices and combines segments in an

interactive manner. To keep delay and computational requirements low, any information used to make bandwidth decisions should be directly available in the compressed video stream. It is desirable to have a resource management system that can accurately estimate the required bandwidth in real-time.

We focus on the resource management of variable bit rate (VBR) video[1], which offers consistent perceptual quality. The hallmark of VBR video is that its bandwidth undergoes both short- and long-term changes, in reaction to the complexity—and therefore, compressibility—of the underlying video. No one-time bandwidth allocation will provide loss-free VBR video transmission with high utilization and low delay. For MPEG-1 and MPEG-2 streams, the bit rate variations can be up to an order of magnitude and occur on two different time scales. The shorter time scale corresponds to the duration of the GOP (group of pictures); the variation is due to the fact that intracoded (I) frames generally require more bits than forward predicted (P) frames, which in turn require more bits than bidirectionally predicted (B) frames. The brief spikes in traffic caused by I frames are generally not a problem for networks; as most MPEG compressors produce only two or three I frames per second, a small buffer can adequately smooth the traffic if some delay is tolerable. The long-term variation is brought about by changes in the semantic content of different shots and scenes. Such bandwidth changes cannot be easily absorbed by reasonable-capacity network buffers. This long-term bit rate variation is one of the biggest challenges in VBR video transmission.

Traditional IP traffic, such as that generated by file transfers and email communication, is supported by best-effort service that does not have guaranteed delay, transfer rate, or other QoS characteristics. To facilitate transmission of real-time multimedia content across the Internet, several new protocols have been proposed in recent years. Among them, the Resource Reservation Protocol (RSVP) is a network control protocol that allows Internet applications to obtain a certain QoS for their corresponding data flows [41, 42]. Within

[1]The contribution of audio to the overall bit rate is largely ignored, as video is the dominant source of data.

this protocol, a route reservation is created and periodically updated in two stages: the sender multicasts PATH messages containing traffic characteristics, then RESV messages containing resource reservation requests are forwarded from the receiver along a reverse path. In the context of QoS-guaranteed network communication, it is crucial to quantify the video traffic characteristics as precisely as possible, independent of which protocol is being used. Such quantification generally involves prediction of future and/or long-term traffic patterns because the frequency of reservation adjustment is limited in practice.

For the efficient transmission of VBR video, we study two issues: (1) at which points the bandwidth should be renegotiated, and (2) how much bandwidth to ask for at any given point.

Conventional approaches renegotiate resources according to changes in bitstream level statistics [43]. The relationship between past and future traffic was parametrically modeled in work such as that by Chong, Izquierdo, and references therein [44, 45]. Doulamis, *et al.*, derived separate models for I, P, and B frames, altering them based on a model of "activity" in a GOP [46]. More clearly content-based approaches have been introduced, motivated by the high correlation between long-term traffic characteristics and video content [47, 48]. We find that while content is a major factor in determining the bandwidth, content alone may not be sufficient for predicting future traffic and in estimating how much bandwidth to request.

A content-based prediction approach has been proposed by Bocheck, *et al.*, consisting of training and testing stages [48]. In the training stage, content features are quantized into a small number of levels (e.g., slow/medium/fast motion), and every possible combination of significant features is labeled as one content class for which the typical traffic pattern is computed. After training, the content class for each shot in the test video is identified by extracting the same features, and the typical traffic pattern of the class is used as the predicted traffic for that shot. Unfortunately, this specific prediction structure via classification can only feasibly incorporate a limited number of coarsely quantized features;

each feature is weighted equally, rather than by its relevance to traffic. In addition, the training is only applicable to a single compressor and parameter set: content alone is not useful for analyzing differently-encoded versions of the same stream. Finally, some useful and readily-available information, such as the exact bandwidth statistics of the video in the observation periods, are not incorporated.

3.1 Bandwidth Renegotiation Points

Whenever the traffic characteristics of the transmitted VBR stream change dramatically, the requested bandwidth should be renegotiated. A tradeoff in overhead must be considered, however: if the renegotiation happens too often (say, every frame), the request and negotiation packets themselves will be a significant source of traffic. In addition, the renegotiation process likely involves delay itself, and is limited by the available computational power. Renegotiating too infrequently leads to dropped packets or frames, poorer overall network utilization, and possibly wasted expense, if bandwidth is not a free commodity.

The on-line determination of bandwidth renegotiation points in VBR video generally falls into three categories: deterministic, traffic-based, and content-based. Deterministically setting the renegotiation points is the simplest method: bandwidth requests are made every n frames, where n is an empirically determined balance between request overhead and correlation of frame bit rates. Traffic-based renegotiation occurs when the stream violates a previously negotiated bandwidth request, or when utilization drops below some level. Although traffic-based renegotiation tracks the real bandwidth more closely, a single complex frame can cause the requested bandwidth to remain elevated for some time. A more "natural" set of renegotiation points is the set of shot boundaries. By studying the bits used per frame in VBR video, one sees that the most dramatic changes occur at the beginning of new camera shots [48]. Within a single shot, the traffic characteristics are relatively constant[2].

[2]If a shot has a sudden change in content features, the change can be considered a boundary as far as

There exist many approaches to finding shot boundaries in the compressed domain. For simplicity, we consider only abrupt transitions and adopt Yeo's compressed-domain cut detector [16]. This method uses a windowed relative threshold on the sum of absolute DC-frame pixel differences,

$$d_k = \sum_{i,j} |x_k(i,j) - x_{k-1}(i,j)|, \tag{3.1}$$

and allows for fast, on-line computation of renegotiation points. The gradual transition detectors described in Chapter 2 can be used to augment the cut detector.

3.2 Traffic Prediction per Interval

After selecting renegotiation points, the next step is to determine how much bandwidth to request for each interval without introducing significant delay. For natural renegotiation points such as shot boundaries, past shots' traffic generally cannot help in determining a new request, as the traffic pattern has changed. Exact traffic information for the new video shot can be obtained by measuring every frame between the current and the next renegotiation points, using this as a basis for the resource request. This is possible in an offline situation where the future video is available, but is not suitable for real-time applications because significant delay will be introduced (particularly if the shots are long). With the requirement of online processing in mind, one can predict the traffic for the entire shot based on an observation of the first few frames. This is illustrated in Figure 3.1, where the shaded areas indicate observation periods. Renegotiation is performed after the short-term observation, and if granted, the video will be transmitted using the newly reserved bandwidth. If the request is not granted, the source could attempt to transcode the video in order to fit a smaller bandwidth for the single shot; otherwise, the shot is dropped. Note that the observations will inevitably introduce some delay in renegotiation, but the video itself may be transmitted without delay, as in Figure 3.1(a). With this approach,

renegotiation is concerned. For simplicity, we will ignore such intra-shot "boundaries."

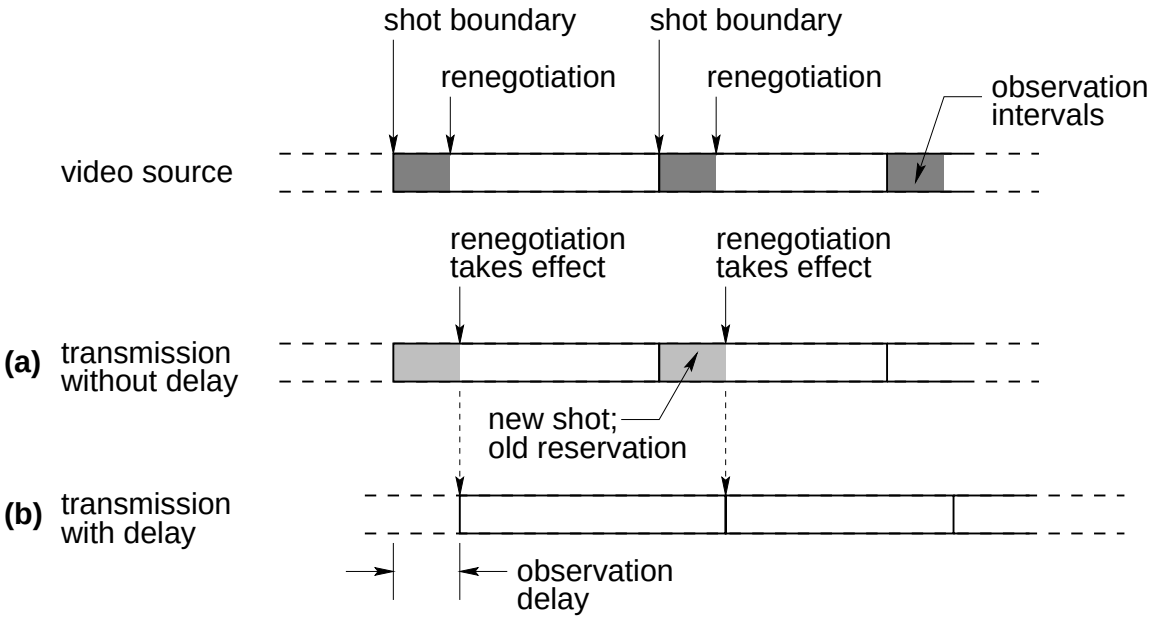


Figure 3.1: Traffic prediction scenarios with different delays.

unexpected bursty traffic during the shaded periods can only be accommodated by adding extra capacity to network buffers. For applications tolerating a short-delay, the video may be transmitted with a t-second delay as in Figure 3.1(b), so that the video traffic is always within the bounds of the negotiated agreement. While our approach can be applied to both delayed and non-delayed transmission, we shall focus on the better-performing delayed transmission case.

Although the problem of predicting future traffic based on short-term observations may be handled by parametric modeling, it is not easy to come up with a simple and effective parametric model when incorporating content features. For this reason, we use a neural network to accomplish the prediction task, as shown in Figure 3.2. The input to the network consists of selected content features and traffic descriptors from the observation period. The outputs are the principal components of the D-BIND traffic descriptor for the entire shot, as discussed in the next section. We adopt a multilayer-perceptron network with a single hidden layer and apply the back-propagation approach in supervised training [49].

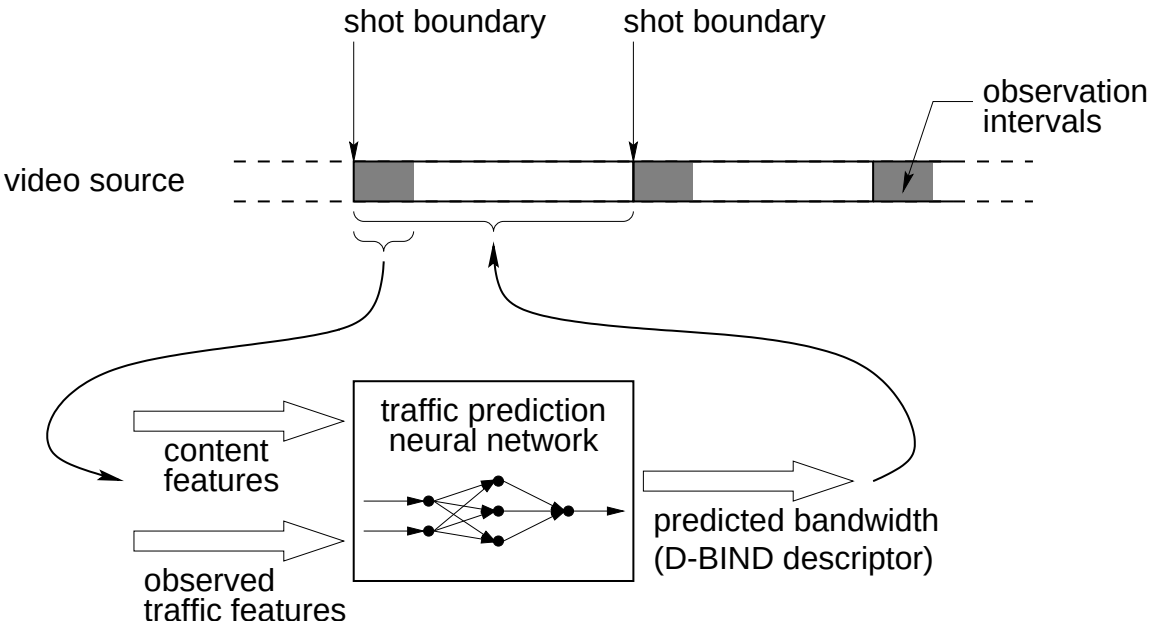


Figure 3.2: Neural network based traffic prediction, using both short-term traffic observations and content features to determine the entire shot's traffic patterns.

3.2.1 Media Stream Traffic Descriptors

In any traffic management scheme, a method is needed to quantify bandwidth requirements; many traffic descriptors have been proposed in the literature. Among them, peak rate and average rate are two very simple ones, but they do not capture traffic patterns over different time scales (a feature needed for VBR transmission). To overcome this problem, Knightly, *et al.*, proposed the D-BIND descriptor for deterministic service, which provides a worst-case performance guarantee [50]. D-BIND, or the *d*eterministic *b*ounding *in*terval *d*ependent model, is essentially a vector containing the maximum allowed arrival rate for various intervals. It is defined as follows: Let $A[\tau, \tau + t]$ be the cumulative number of bits arriving during the t-length interval beginning at time τ. The tightest bound over all time, called the "empirical envelope," is

$$B^*(t) = \sup A[\tau, \tau + t] \tag{3.2}$$

A piecewise-linear bounding function B_{W_T} is then constructed from $B^*(t)$, where $W_T = \{(q_k, t_k)|k = 1, 2, ..., p\}$ is the vector of bit arrival and interval pairs. Given a set of interval lengths t_k, the tightest such bounding function is denoted $B^*_{W_T}$. The D-BIND descriptor is usually expressed in terms of arrival rates, $R_T = \{(r_k, t_k)|k = 1, 2, \ldots, p\}$, where $r_k = q_k/t_k$. This descriptor captures both the short-term burstiness and the long-term traffic characteristics of a video segment, while being relatively simple to implement in admission control and policing.

3.2.2 Content Features

Image complexity and motion have been suggested by Bocheck, *et al.*, as significant correlates to video traffic [48]. Keeping in mind the requirement of efficient online processing, we extract fourteen features related to complexity and motion by processing the video in the compressed domain. This set of content features is likely more than necessary, but we will rely on the selection methods in the following sections to weed out redundant or irrelevant features. Other features could be incorporated as well, if they have a high relevance to traffic.

The spatial "complexity" of the intracoded (I) frames is intuitively the dominant factor determining a stream's resource requirements, because the number of bits required to encode the frame is directly dependent on the energy compaction provided by the DCT and the compaction is less dense in blocks with edges or complex textures. In order to estimate complexity, we compute the weighted sum of the magnitudes of AC coefficients in the frame (DC coefficients are differentially encoded, so high DC magnitudes do not exact much penalty in traffic). Any weighting pattern giving more weight to higher-frequency DCT coefficients could be used; we chose to weight coefficients according to the sum of their frequencies in each dimension (the L^1 "distance" from the DC coefficient).

Motion vector magnitudes can dramatically affect the resources required by predicted (P and B) frames; for simplicity we shall consider only the forward predicted frames. Higher

magnitudes mean more intense motion, and consequently more correction will likely be needed in the residue frames after motion compensation. Motion direction, for the most part, is irrelevant to traffic. We compute the mean motion vector magnitude, for the whole frame, as follows:

$$\overline{\|\text{mv}\|} = \frac{1}{M} \sum_{i,j} \|\mathbf{m}_k(i,j)\|_2 \tag{3.3}$$

where M is the number of macroblocks in the video frame and $\mathbf{m}_k(i,j)$ is frame k's forward motion vector for the macroblock (i,j). In order to identify segments with strong motion in part of the frame, but not the entire frame, we also compute the value of (3.3) for each of four spatial quadrants.

The coding efficiency of predicted frames can also be measured by counting the number of intracoded blocks in the frame; areas that could not be adequately predicted from previous frames must be encoded again, at some expense in bandwidth. The fraction of P frame macroblocks that must be intracoded, instead of intercoded, therefore is another candidate feature.

Motion compensation is less efficient if the object or frame motion is not "simple", meaning that more correction must be applied in the residue frames if different macroblocks' motion vectors point in radically different directions. We measure the motion complexity in a number of ways, and rely on the feature selection process to find the ones most important to traffic prediction. First, we form a simple directional histogram of the motion vectors, in which each intercoded macroblock's motion vector is classified into five bins: up, down, left, right, or "zero," according to the dominant axis of the vector. Complex motion corresponds to having roughly equal values in each bin, so we use the variance over these five bins as a candidate feature. An alternative way of measuring the coverage of the motion prediction over the new frame is to compute the spatial variance of the motion vector magnitudes:

$$\text{var}\left(\|\mathbf{m}_k\|\right) = \frac{1}{M} \sum_{i,j} \|\mathbf{m}_k(i,j)\|_2^2 - \left(\frac{1}{M} \sum_{i,j} \|\mathbf{m}_k(i,j)\|_2 \right)^2 \tag{3.4}$$

In addition, the spatial variances of the x and y motion vector components, as well as

their cross covariance, are calculated:

$$\mathrm{var}\left(m_{k_x}\right) = \frac{1}{M}\sum_{i,j} m_{k_x}^2(i,j) - \left(\frac{1}{M}\sum_{i,j} m_{k_x}(i,j)\right)^2 \tag{3.5}$$

$$\mathrm{var}\left(m_{k_y}\right) = \frac{1}{M}\sum_{i,j} m_{k_y}^2(i,j) - \left(\frac{1}{M}\sum_{i,j} m_{k_y}(i,j)\right)^2 \tag{3.6}$$

$$\begin{aligned}\mathrm{cov}\left(m_{k_x}, m_{k_y}\right) &= \frac{1}{M}\sum_{i,j} m_{k_x}(i,j) m_{k_y}(i,j) \\ &\quad -\frac{1}{M^2}\sum_{i,j} m_{k_x}(i,j)\sum_{i,j} m_{k_y}(i,j)\end{aligned} \tag{3.7}$$

Finally, as we are only able to observe the very beginning of each new camera shot, the ways in which motion might change throughout the shot are important to estimate. Even if the motion magnitude is small in the first few frames, it can be large later in the shot, requiring more bandwidth to represent. To make this effect more manageable, we measure the object and frame acceleration in two ways. First, motion vectors from adjacent predicted frames are subtracted to form acceleration vectors, of which we take the mean magnitude:

$$\overline{\|\mathrm{accel}\|} = \frac{1}{M}\sum_{i,j} \|\mathbf{m}_k(i,j) - \mathbf{m}_{k-1}(i,j)\|_2 \tag{3.8}$$

A high value for this mean indicates that the motion in the video is not simple, and that the residue frames will become increasingly complex (thus requiring more bits). The second candidate acceleration feature places greater emphasis on changes in speed, rather than changes in direction:

$$\overline{\Delta\,\|\mathbf{m}\|} = \frac{1}{M}\sum_{i,j} \left(\|\mathbf{m}_k(i,j)\|_2 - \|\mathbf{m}_{k-1}(i,j)\|_2\right) \tag{3.9}$$

The eighteen candidate predictor inputs (fourteen content plus four traffic) are summarized in Table 3.1. None of the fourteen content features requires full decompression of the VBR stream to compute; in MPEG-1 and 2, the amount of computation required is quite low.

Feature	Description
1	I frame complexity
2	Mean MV magnitude
3	Variance of MV directional histogram
4	Fraction of intracoded MB's
5	Mean magnitude of accel vectors
6	Mean change in MV magnitudes
7	Mean MV magnitude, upper-left
8	Mean MV magnitude, upper-right
9	Mean MV magnitude, lower-left
10	Mean MV magnitude, lower-right
11	Variance of MV x components
12	Variance of MV y components
13	Covariance of MV x and y components
14	Variance of MV magnitudes
15	Short-term D-BIND r_1 (1 frame)
16	Short-term D-BIND r_2 (2 frames)
17	Short-term D-BIND r_3 (3 frames)
18	Short-term D-BIND r_4 (4 frames)

Table 3.1: Candidate content and traffic features for use in per-interval traffic prediction.

3.2.3 Feature Selection for Traffic Prediction

There is, however, significant redundancy in the eighteen features of Table 3.1, and not all may be highly relevant to traffic prediction. The importance of selecting the relevant subset from the original feature set is closely related to the "curse of dimensionality" problem in function approximation, where sample data points become increasingly sparse when the dimensionality of the function domain increases, such that the finite set of samples may not be adequate for characterizing the original mapping [51]. In addition, the computational requirements are usually greater for implementing a high-dimensional mapping. To alleviate these problems, we reduce the dimensionality of the input domain by choosing a relevant subset of features from the original set.

Traffic statistics have a nonlinear dependence on short-term content features, leading us to utilize the sequential forward selection (SFS) procedure, combined with an easily-trained general regression neural network (GRNN)[52, 53, 54]. The SFS/GRNN feature selection method described in the following sections was developed by Dr. Hau-San Wong. Dr. Wong's results are summarized here for completeness; a more detailed development is available in [55] and [56].

Unlike typical neural network structures which require iterative training, the GRNN model parameters can be directly determined from the training data. Such one-pass training allows for rapid evaluation of individual feature subsets, but is in general sub-optimal. As the number of feature inputs increases, the approximation error becomes more and more significant, and as we shall see, an alternative approach is necessary to augment the feature set beyond a certain point.

Sequential Forward Selection (SFS)

The sequential forward selection (SFS) procedure provides for the incremental construction of salient feature subsets [52]. Given an original set F of N features, we define a sequence of subsets F'_m, $m = 0, \ldots, N$, where each set contains m features. At each iteration, F'_{m+1}

is the union of F'_m with a single (remaining) feature that is most relevant by some measure.

In our case, we are constructing feature subsets to aid in traffic prediction, so a natural relevance metric, $D_{F'_m}$, is the prediction mean square error

$$D_{F'_m} = \frac{1}{P}\sum_{p=1}^{P} \|\mathbf{y}_p - g(\mathbf{x}_{F'_m,p})\|^2 \tag{3.10}$$

where $g(\cdot)$ is the GRNN-based predictor described in the next section and P is the number of elements in the training data set. $D_{F'_1}$ is first calculated, using scalar prediction inputs, for each single feature. The feature yielding the minimum $D_{F'_1}$ is chosen to construct the final F'_1. $D_{F'_2}$ is then calculated for of the remaining $N-1$ features, combining the test feature with F'_1, and the test feature minimizing this MSE is used to form the final $F'_2 = f_{\text{minimum MSE}} \cup F'_1$. In this manner, we construct a final set of nested feature subsets, $F'_1 \subset F'_2 \subset \ldots \subset F'_N$. We can then select the smallest subset that meets a desired MSE criteria.

SFS via the General Regression Neural Network (GRNN)

In order to calculate the MSE values, $D_{F'_m}$, required by SFS, a sub-optimal—but simple to train—predictor is used. The general regression neural network (GRNN) is a two-layer special case of the radial basis function (RBF) neural network [53, 54]. The first layer applies Gaussian transfer functions to the input values, and the second is a simple linear summation layer. Unlike typical RBF networks, however, the centers and widths of the Gaussians in the GRNN are deterministic functions of the training data.

We begin with a set of P training data samples, $(\mathbf{x}_p, \mathbf{y}_p)$, where $p = 1, \ldots, P$, each $\mathbf{x}_p$ is an m-dimensional input vector, and each $\mathbf{y}_p$ is the corresponding result of the function to be approximated. Each sample point is associated with a multivariate Gaussian kernel in the first network layer, where vector $\mathbf{x}_p$ is assigned to the center of the kernel. The output of the p-th first-layer node is

$$\beta_p = \exp\left[-\frac{(\mathbf{x}-\mathbf{x}_p)^T(\mathbf{x}-\mathbf{x}_p)}{2\sigma^2}\right], \quad p = 1, \ldots, P, \tag{3.11}$$

for an arbitrary input vector x, where σ is a user-specified smoothing parameter. The network output is then the weighted sum

$$\mathbf{y} = \sum_{p=1}^{P} \alpha_p \mathbf{y}_p, \tag{3.12}$$

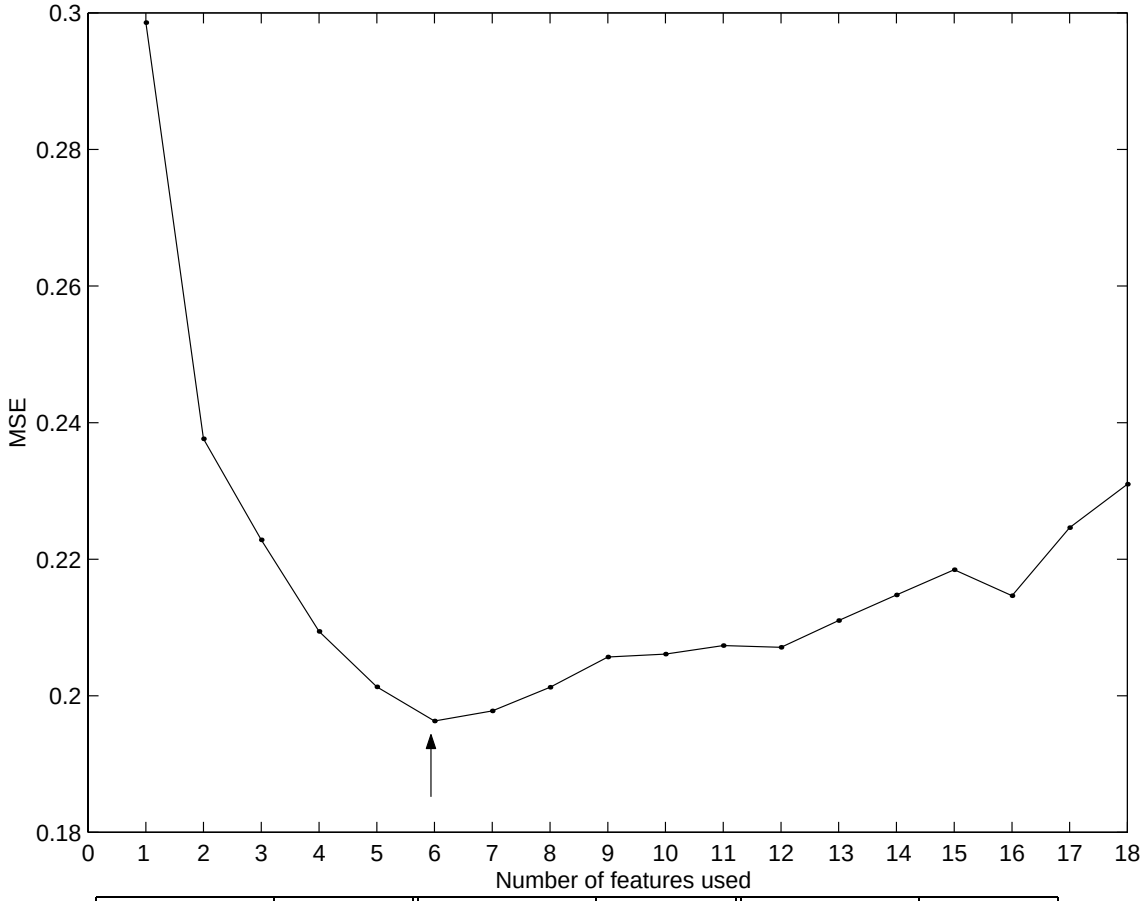
where the weights α_p are

$$\alpha_p = \frac{\beta_p}{\sum_{p=1}^{P} \beta_p}. \tag{3.13}$$

Results of SFS/GRNN Feature Selection

We now apply the SFS/GRNN feature selection technique to the eighteen candidate features in Table 3.1. As the GRNN requires separate test and data sets, we adopt the leave-one-out approach for evaluating the approximation errors. This approach, a special case of cross validation, uses the average error obtained by training with each possible set of $P-1$ samples and testing with the remaining one [51].

13175 frames of digitized cable television video, encoded in VBR MPEG-1 at an average bit rate of 2.1 Mbps, were used for the experiments. The method of Section 3.1 identified 177 intervals in the video, and short-term feature observations were computed for each interval as described in Section 3.2.2. Figure 3.3 shows the accumulated prediction error at each SFS iteration, as well as the feature subsets used. Beyond F_6', which consists of features $\{1, 6, 15, 16, 17, 18\}$, the error beings to increase. This is likely due to the suboptimal nature of the GRNN, particularly in high dimensional spaces; the number of training samples forces the GRNN kernels to only sparsely populate the increasingly large feature space. The upswing makes it clear that feature indices near and beyond the minimum do not reflect their actual order of relevance to prediction, and that an alternative approach is needed to select sets of more than 6 features.



Feature Set	Features	Feature Set	Features	Feature Set	Features
F'_1	15	F'_7	$3 \cup F'_6$	F'_{13}	$4 \cup F'_{12}$
F'_2	$18 \cup F'_1$	F'_8	$5 \cup F'_7$	F'_{14}	$2 \cup F'_{13}$
F'_3	$17 \cup F'_2$	F'_9	$7 \cup F'_8$	F'_{15}	$10 \cup F'_{14}$
F'_4	$16 \cup F'_3$	F'_{10}	$8 \cup F'_9$	F'_{16}	$9 \cup F'_{15}$
F'_5	$1 \cup F'_4$	F'_{11}	$12 \cup F'_{10}$	F'_{17}	$11 \cup F'_{16}$
F'_6	$6 \cup F'_5$	F'_{12}	$13 \cup F'_{11}$	F'_{18}	$14 \cup F'_{17}$

Figure 3.3: Cumulative error plot for SFS/GRNN feature selection; the table shows which features are included after each SFS step (cf. Table 3.1). Minimum GRNN MSE is achieved after selecting six features, as indicated by an arrow. The increasingly suboptimal nature of the GRNN in high dimensional spaces means that the feature order near and beyond the minimum may not reflect their actual relevance in prediction.

3.2.4 Consistency-Based Feature Selection

As the SFS/GRNN method cannot completely characterize the set of relevant features in the content-to-traffic mapping, we adopt a consistency-based approach as a complementary selection mechanism to evaluate the relevance of content features. Consistency measures were originally used to select features which are most effective in preserving class separability [57]. In the case of traffic prediction, this measure was used by Bocheck, *et al.*, to evaluate the relevancy of content features to video traffic [48]. Their work classifies content feature values into clusters (for example, high, medium, and low motion), implicitly imposing a clustering on traffic by the empirical content-to-traffic mapping. Instead, we propose to cluster the traffic descriptors directly, then look for the subset of feature space that most accurately preserves this clustering in the reverse mapping.

In the first step, video shots are classified into k traffic clusters based on the D-BIND traffic descriptor (Section 3.2.1). Classification can be done by K-mean, E-M, or other algorithms. A consistency measure $\mathcal{C}$ for each feature individually is then computed [48]:

$$\mathcal{C} = \frac{\text{mean inter-class distance}}{\text{mean intra-class distance}} \tag{3.14}$$

where the distances are in the space of the features under consideration. Highly relevant features will have large $\mathcal{C}$ values, as such features induce small intra-class distances and large inter-class distances among traffic clusters in feature space.

We apply K-mean clustering to classify video shots' traffic into 4 clusters. Using the first two principal components of the D-BIND descriptor of each shot, the traffic clusters are shown in Figure 3.4. Each cluster reflects a different level of complexity and action. The shots in rightmost cluster typically contain fast motion along with considerable complexity, for example. We then compute each content feature's consistency measure according to equation (3.14), with the sorted results shown in Figure 3.5. I frame complexity (feature 1) has the highest consistency measure of all content features, which is the same result as achieved by the SFS/GRNN approach in Section 3.2.3. Similarly, we find that the average

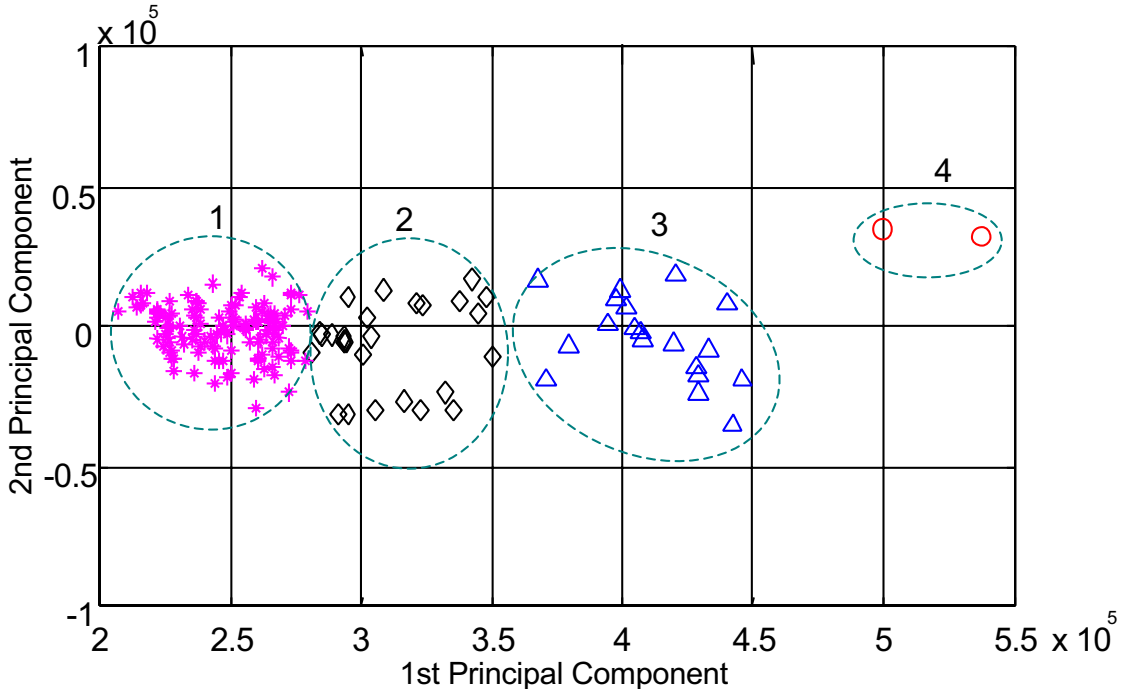


Figure 3.4: Four traffic classes derived by K-mean clustering on the two principal components of D-BIND, the first step in consistency-based feature selection.

magnitude of P frame "acceleration" (feature 5) has the second highest consistency. We also notice that features 7–10, the regional motion magnitudes, have high correlation with feature 2, the global motion magnitude, and all have similar consistency. To reduce the redundancy in the selected feature set and the prediction complexity, we exclude the regional motion features, resulting in four highly consistent content features: $\{1, 5, 2, 13\}$.

It should be pointed out that the consistency-based approach assumes features are uncorrelated and only considers features that are related with the traffic descriptor in a monotonic way as beneficial. For this class of features, a large distance in traffic space implies a large distance in feature values. Although these assumptions simplify the problem and provide a feasible way to evaluate a feature relevancy, more complicated relations between features and traffic are not captured by this approach.

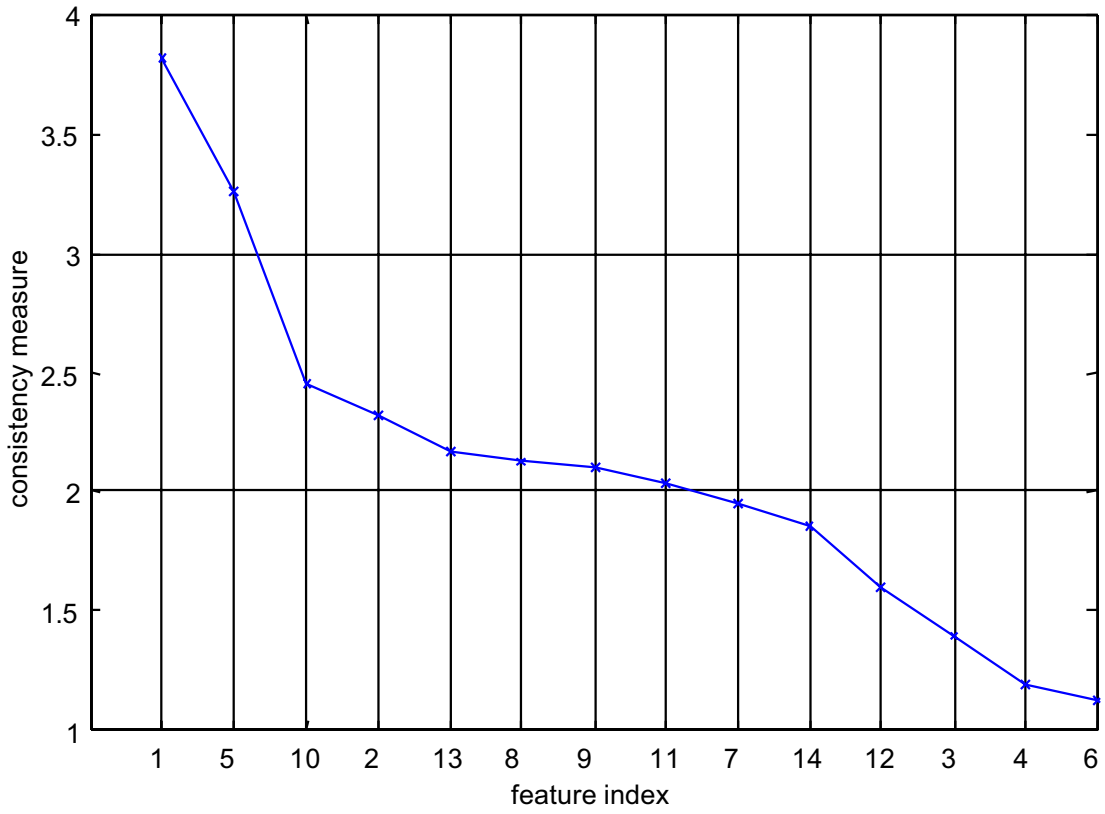


Figure 3.5: Sorted consistency measures $\mathcal{C}$ for each candidate content feature, when used individually.

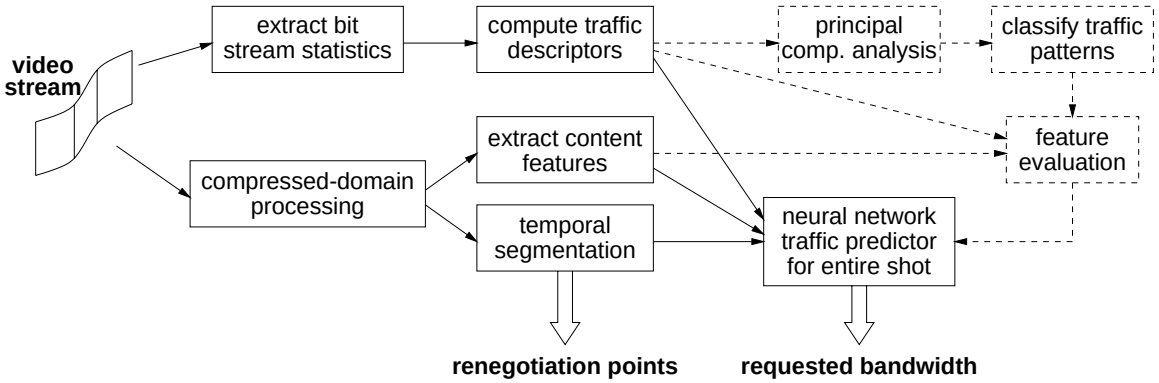


Figure 3.6: Overall structure of the VBR resource predictor; dashed lines represent connections only made during the training phase.

3.3 Experimental Results

We use D-BIND descriptors and deterministic (conservative) service in our tests, though the proposed framework is applicable to others policies. Fixing $[t_1, ..., t_p]$, D-BIND can be described by a vector $[r_1, ..., r_p]$. r_1 through r_4 of the short-term observed traffic, corresponding to intervals of 1 to 4 frames respectively, are the remaining inputs to our neural network (Figure 3.2). When describing the entire shot, the dimensionality of D-BIND is large, increasing the prediction complexity. Such an increase is rather wasteful as there is some redundancy in D-BIND (r_k approaches the mean bit rate for large k). In order to lessen redundancy and reduce prediction complexity, we apply principal component analysis (PCA) to D-BIND and use the first L principal components as traffic descriptors. The neural network will then predict these L values. The overall system structure is illustrated in Figure 3.6. Some prelminary results with this framework were presented in [58].

3.3.1 Prediction MSE

We first demonstrate the performance of our proposed framework by evaluating the prediction mean squared error (MSE), a commonly used criterion. For the traffic prediction problem, the overestimation of shot D-BIND descriptors could lower network utilization,

and underestimation could degrade QoS or even cause network buffer overflow. Our experiments were performed on the same 13175-frame MPEG-1 VBR video mentioned above; it consisted of segments from a fast-action documentary and a television drama.

To verify the selection results of the SFS/GRNN approach, the feature subset $F_6' = \{1, 6, \text{4-dim D-BIND}\} = \{1, 6, 15, 16, 17, 18\}$ (Section 3.2.3) is used for training a multilayer perceptron to predict the long-term traffic statistics. Among the 177 shots extracted from the video sequences, the first 50 shots are used as training samples for the network, and the next 127 shots are used as test data. We have listed the prediction mean square error in normalized units[3] for different numbers of hidden nodes in Table 3.2. For the purpose of comparison, we have also included the prediction results by randomly choosing 2 sets of 6 features from the original 18. We can observe that the 6 features selected by SFS and GRNN achieve the smallest error in each case. In addition, we also notice that increasing the number of hidden nodes from 10 to 20 does not significantly improve the prediction results, and for some particular feature combinations the prediction error even increases for a large hidden layer, indicating the possibility of overfitting. As all the D-BIND features rank close to the top of the feature list, it is reasonable to suggest that most of the useful information for predicting the future traffic is already embedded in these short-term statistics. To confirm this, we have also included the prediction results using the 4 short-term D-BIND features only. We observe that the resulting errors are only slightly greater than those of the original selected subset F_6', indicating that these short-term features are the most essential for predicting the long term network traffic.

From these results, we can conclude that the SFS/GRNN selection mechanism is capable of identifying the most important features—the short-term D-BIND statistics—for the current prediction problem. On the other hand, we find that the addition of content features to the D-BIND subset serves to slightly improve the prediction result. That only

[3]Note that the D-BIND principal values are on the order of 10^5 bits per frame, and the prediction MSE of these principal values is on the order of 10^{10}.

Feature Subset	number of hidden nodes	MSE (1st PCA)	MSE (1st and 2nd PCA)
F_6'	10	0.0238	0.0277
D-BIND	10	0.0247	0.0281
Random Set 1	10	0.0559	0.0695
Random Set 2	10	0.0426	0.0545
F_6'	20	0.0232	0.0268
D-BIND	20	0.0244	0.0279
Random Set 1	20	0.0579	0.0719
Random Set 2	20	0.0488	0.0617

Table 3.2: MSE traffic prediction results using content/traffic features selected by SFS/GRNN, traffic features only, and two random feature sets.

Feature Subset	MSE (1st PCA)	MSE (1st and 2nd PCA)
F_6' (SFS/GRNN)	0.0232	0.0268
F_8 (combined)	0.0215	0.0257

Table 3.3: MSE traffic prediction results, comparing features selected by SFS/GRNN and by the combined SFS/GRNN/consistency approach.

two of the 14 content features are included in the selected subset is due to our previous decision not to adopt those content features beyond the GRNN minimum-error point. Due to the GRNN dimensionality issues explained above, we have employed consistency-based selection to augment the SFS/GRNN process. To demonstrate the improvement, we list the prediction MSE's of the feature sets F_6' of the SFS/GRNN approach and $F_8 = \{1, 2, 5, 13, \text{4-dim D-BIND}\}$ of the combined approach (Section 3.2.4) in Table 3.3, where the number of hidden nodes is 20. The prediction MSE using F_8, selected by the combined approach, is smaller than that of F_6', selected by SFS/GRNN alone, especially for predicting the most significant component of the D-BIND descriptor. This confirms that incorporating the alternative selection approach can enhance prediction performance.

Finally, using feature set F_8 selected by the combined approach, we compared the prediction MSE under four different strategies. With respect to renegotiation points, we consider: (A) using equal-length request intervals (one request every 75 frames, which is the average shot length), and (B) using shot boundaries from temporal segmentation. We also consider three different neural network inputs for traffic prediction: (I) the four content features $\{1, 2, 5, 13\}$ of the observed video, (II) the four short-term D-BIND features $\{15, 16, 17, 18\}$ of the observed video, and (III) both of the above. Two sets of comparisons are shown in Figure 3.7. Comparing the two leftmost columns, (A-III) and (B-III), we observe that (B-III) gives much smaller MSE, meaning that content-based renegotiation points are by far superior to non-content-based ones. Comparing the three rightmost columns, we observe that short-term traffic (B-II) gives better prediction performance than content features (B-I) alone. In addition, we find again that using both content and short-term bandwidth of observed video (B-III) is only marginally better than using short-term bandwidth alone (B-II). This implies that most of the useful information in content features for predicting traffic is already inherent in the short-term bandwidth statistics.

3.3.2 Trace-Driven Link Utilization

We compare our proposed approach with a static peak-rate allocation and a bitstream-level dynamic scheme to demonstrate the improvement of network link utilization achievable. The R-VBR scheme, a heuristic dynamic renegotiation algorithm using D-BIND descriptors, was proposed by Zhang, *et al.*, and claims significant improvement over static peak rate allocation [43]. It raises the reserved bandwidth (described by D-BIND) by a factor α when the real bandwidth exceeds the reserved resources, and lowers it by a factor β when the real bandwidth remains below the reserved amount for K frames. The average R-VBR renegotiation frequency is determined by the triplet (α, β, K). In contrast, our proposed scheme uses the shot boundaries, obtained from content-based temporal segmentation, as

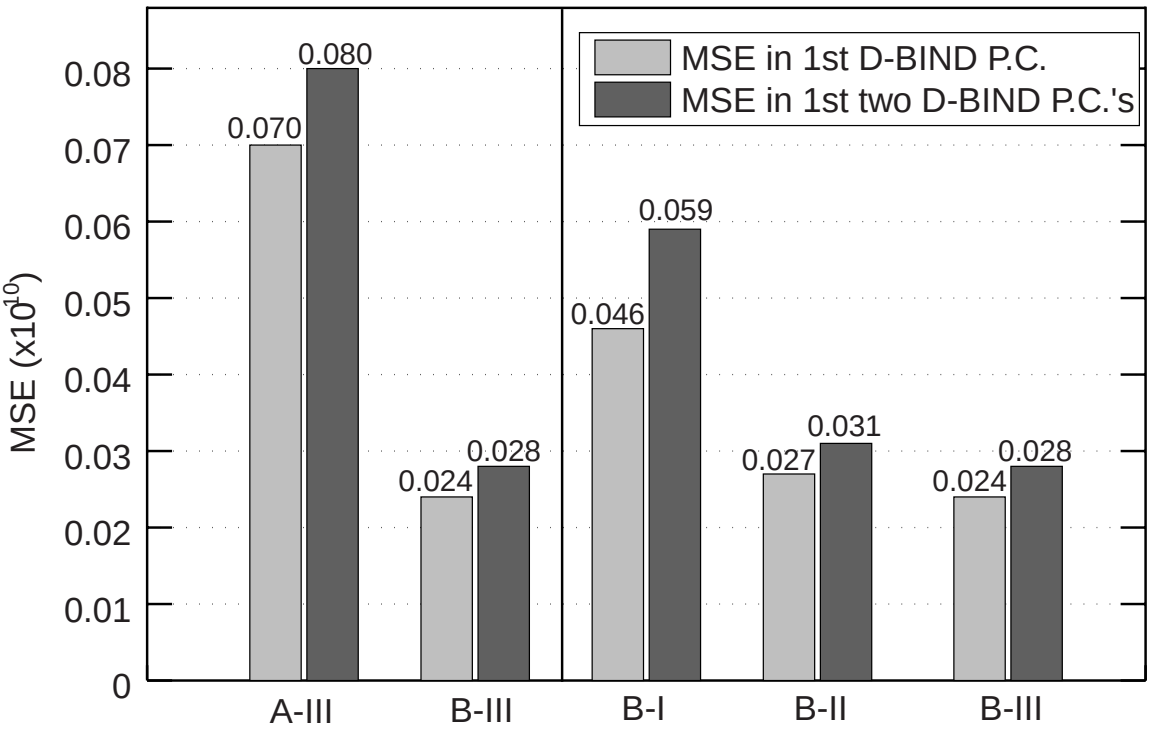


Figure 3.7: Traffic prediction MSE using (A) equal-length request intervals and (B) shot-boundary derived intervals, combined with (I) content, (II) short-term traffic, and (III) combined content/traffic-based predictors. (D-BIND values are on the order of 10^5 bits.)

renegotiation points, and a neural network traffic predictor to determine how much bandwidth to ask for at each point. For the 177-shot video used in our experiments, the full D-BIND vector of each entire shot is estimated from the two principal components which are the outputs of the neural network traffic predictor. These predicted D-BIND descriptors are the values used in bandwidth renegotiation.

Link utilization is obtained by trace-driven simulation, similar to that described in [48]. Multiple video sources, based on the above mentioned sample video but with random starting points, are multiplexed into a T3 line (link speed $c = 45$ Mbps). For simplicity, the simulation blocks a source when its resource request is rejected, and a new request is generated at the next renegotiation point. More sophisticated admission control is certainly possible, a subject for future research. A network buffer with maximum capacity Q and first-come-first-served queuing policy is used to smooth out the bursty traffic. When a renegotiation request is received from the n-th source, the worst case buffer occupancy is computed:

$$Q_W = Q_C + \max\left\{0, \max_{1\le k\le d}\left\{t_k \cdot \left[r_k(n) - c + \sum_{i\neq n} a_i \cdot r_k(i)\right]\right\}\right\}, \tag{3.15}$$

where Q_C is the current buffer occupancy, d is the dimension of D-BIND descriptor, $r_k(i)$ is the k^{th} D-BIND component of the i^{th} source, and a_i is set to 1 if the i^{th} source is admitted and 0 otherwise. The requested bandwidth is granted only if $Q_W \le Q$. Given a bound on the per-stream rejection probability (1% in our simulation), link utilization is defined as:

$$u = \frac{\text{max number of admitted sources}}{c/r_{avg}}, \tag{3.16}$$

where r_{avg} is the average rate of the entire video sequence. The simulation results of utilization versus buffer capacity are shown in Figure 3.8. With three parameter settings, $(\alpha = 1.3, \beta = 0.7, K = 30)$, $(\alpha = 1.3, \beta = 0.7, K = 60)$, and $(\alpha = 1.4, \beta = 0.7, K = 90)$, the R-VBR scheme generates requests at average rates of 0.81, 1.54, and 2.32 seconds, respectively. The corresponding utilizations are shown as dashed curves. The bottom

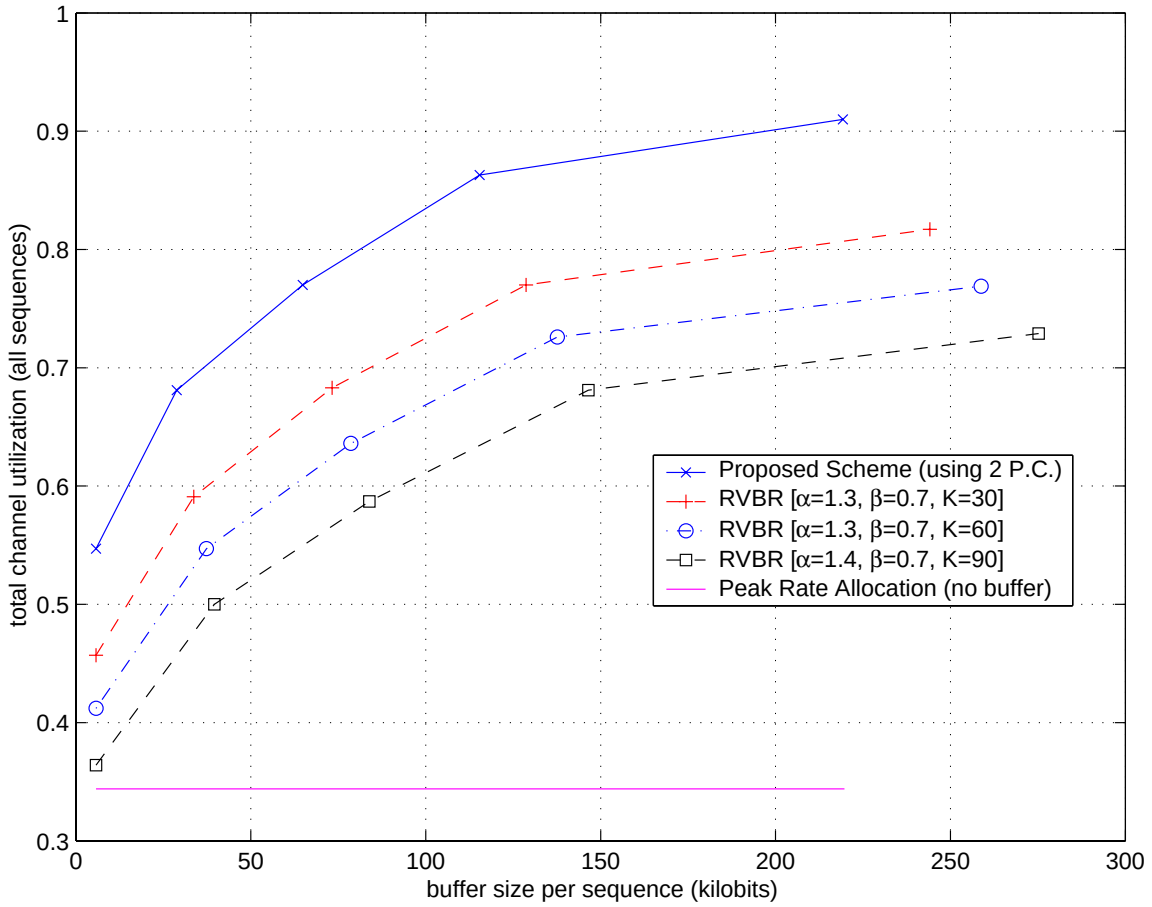


Figure 3.8: Network utilization for multiplexed sources, computed by trace-driven simulation, for our proposed resource allocation scheme, peak rate allocation, and R-VBR.

straight line shows the utilization if the peak bandwidth were allocated to each sequence, with no buffering to smooth the data. (A utilization of 1.0 would correspond to r_{avg} allocation per stream with an infinite buffer and corresponding unbounded delay.) The upper solid curve is the utilization of our proposed scheme, which renegotiates once every 2.48 seconds on average. The figure shows that our proposed scheme obtains a much higher link utilization than peak-rate allocation. Furthermore, our proposal outperforms the R-VBR scheme of similar renegotiation frequency by 18%, as well as by 9% against R-VBR with triple the renegotiation frequency.

Chapter 4

Multimodal Processing

While the segmentation and analysis of the video component of multimedia streams are important, significant information is contained in the accompanying audio tracks. Indeed, it is often possible to understand the plot and organization of a multimedia presentation by listening only to the audio track; the video track alone is generally more difficult to understand. We investigate here both the audio track of multimedia streams as well as how to meaningfully compare statistics extracted from the audio and video tracks. This information will be incorporated in the multimedia temporal analysis techniques presented in later chapters.

In terms of segmentation, we concentrate on speaker changes as the dominant transitions to detect. Most audio sequences also contain music, background sounds, and other effects; for speaker segmentation to be accurate at all, one needs to either filter out these sounds (difficult in general) or separate segments containing the sounds before considering speaker changes. In keeping with our goal of generality, we prefer audio segmentation and processing tools that do *not* require prior training to known speakers or sound sources. In speech processing parlance, we require an "open-set text-independent" speaker segmentor.

4.1 Existing Audio/Video Segmentation and Analysis Techniques

Work in the area of speaker change detection and speaker identification long predates video processing research. Given the commonly-accepted filtered excitation model of speech, the time-average mel cepstrum (spectrum of log-magnitude spectrum) is generally agreed to be the best available statistic for separating speakers [59, 60]. The argument is that, as the convolution of the excitation sequence—which distinguishes speakers—with the vocal tract filter is decomposed into a sum in the cepstral domain, the effects of the filter (i.e., what is actually being said) will be diminished by the time average. The accuracy of this method is hampered, however, by large intra-speaker variation in cepstral coefficients, as well as the fact that humans also use the amount of noise in the voice, the overall vocal-tract shape, and possibly other attributes as additional features in discriminating speakers [61]. In addition, non-voiced segments must be eliminated before computing cepstra, as fricatives and stops do not conform nicely to the filtered excitation speech model. (Ignoring speakers *per se*, more general audio classification can be done via cepstral coefficients as well [62, 63].)

Shridhar, *et al.*, published an early comparison study of distance metrics on cepstral coefficients (among other features) using two-second utterances parameterized at 50 Hz [64]. Gish, *et al.*, computed a likelihood ratio using multivariate Gaussian assumptions on cepstral coefficients, reporting a classification error rate of 10 percent on pre-segmented air-traffic controller audio [65, 66]. Their group also looked at the issue of determining segment boundaries [67]. Using HMM-based phoneme-spotting techniques, Wyse and Smoliar improved on the technique by incorporating only vowel and semi-vowel sounds in the cepstral computation [68]. Nam, *et al.*, used a similar phoneme-based method on audio, combining it with video shot detection and clustering [69]. More recent work includes that of Mori and Nakagawa, who used VQ distortion measures on averaged cepstral coefficients of broadcast news speech, where VQ models for well-known anchors and announcers were pre-trained [70].

Siegler, *et al.*, used cepstral distance metrics to segment and classify speakers, with 64% boundary detection probability, but 60% of the detected boundaries were false alarms; given correct boundaries, the clustering error rate was much lower [71].

Discriminators of speech versus music have been developed using a myriad of extracted statistics as feature vectors; Scheirer gives a good summary of the features used, which include the spectral centroid, time-based spectral difference, spectral rolloff points, zero crossing rate, and 4 Hz modulation energy [72]. Tzanetakis and Cook discuss ways to extract some of this information directly from MPEG-compressed audio streams [73]. Generic audio segmentors have also been built using these features, often combined with trained classifiers to detect certain types of audio (such as ads, "violence," etc.)[74]–[79].

In the last few years, video researchers have begun concentrating on audio as an additional source of information. Much of this work has concentrated on genre-specific heuristic combinations video and audio information. Chang, *et al.*, combined audio cheering detection and word-spotting with image edge detection and football field models to extract touchdown segments [80]. Another simple but commonly-used technique is to determine where audio and video boundaries coincide, as these locations are likely more significant transitions [81, 82, 83]. Liu and Wang combine speaker change information with face detection and clustering to form "cast lists" for media streams [84]. Foote, *et al.*, used audio and video boundary statistics for high-speed browsing of video [85]. A number of researchers have combined techniques such as face detection, shot boundaries, pre-trained speaker detection, and closed-captioning information to classify news, ads, etc., and make other inferences about "content" [86, 9, 87, 88].

From a more abstract point of view, surveys by Minami and Wactlar explored the effects of the director's "intent" on statistical features of audio and video streams [89, 90]. Finally, Pan viewed the combination of audio and video statistics and decisions as a form of information fusion [91].

4.2 Speaker Segmentation and Distance Metrics

Despite the large quantity of previous investigation into speaker segmentation issues, few authors have adequately dealt with problem of segmenting a continuous audio stream into different speakers in a text-independent manner, when no previous training is available. We have tried a number of methods, with limited success. The central problem appears to be the wide variation of cepstral coefficients within the same speaker's words; such a large variance makes it difficult to isolate changes in the underlying distribution.

Our segmentation methods all compute mel-weighted cepstral coefficients at a rate of 100 Hz on 8 kHz-sampled audio, discounting silent and unvoiced (spectrally-flat) time segments[1]. Cepstral values are averaged over two non-overlapping blocks of time, then a distance metric is computed between the cepstral means (in our case, Euclidean, based on the Gaussian model for cepstral coefficients). We tested two methods of forming the pairs of time blocks:

- Sliding pairs of fixed length blocks, with lengths between 0.5 and 2.0 seconds. As 100 cepstral vectors were calculated per second of audio, we slide the blocks in 0.01 second increments, using the above-threshold local maxima in distance metrics as indications of speaker transitions.

- Sliding pairs of blocks, where the first block contains all the cepstral vectors between the last declared transition and the current time, and the second block was of fixed length. This is beneficial, as the generally long first blocks help compensate for large intra-speaker cepstral variations.

Motivated by the phoneme-spotting technique employed by Nam and Wyse to improve their results, we used CMU's Sphinx speaker-independent speech recognition engine to extract phoneme lists from the audio stream [93]. Initially we used the phoneme lists as a

[1] Just as features such as the spectral median can be computed directly from MPEG-compressed data [73], we determined that cepstral distances could be estimated directly from the 32 subband scalefactors of MPEG-1 audio as well [92]. While these estimates are beneficial in keeping with our goal of low computational complexity, we used traditionally-computed cepstral vectors to prevent estimation errors from degrading performance.

filter to the blocks, so that we would only compute cepstra using the following vowel-based Sphinx phonemes:

/AA/, /AE/, /AH/, /AO/, /AW/, /AY/, /EH/, /EY/, /IH/,
/IY/, /OW/, /OY/, /UH/, /UW/, /Y/, /+UH+/, and /+UM+/.

As it is more sensible to compare *like* phonemes, instead of means over a large set, the final segmentor therefore performs the following[2]:

1. Construct two blocks, according to the second method above (the first block is variable-length, the second is fixed).
2. Find the vowel-like (voiced) phonemes within each block.
3. Determine which vowel-like phonemes the two blocks have in common.
4. For each common phoneme, take the (Euclidean) distance between mean cepstral vectors computed only over that phoneme.
5. Compute the mean of these distances, yielding a "phoneme-aligned" distance between the two blocks.

This algorithm was tested on a 71-second audio stream from a television sitcom, consisting entirely of a conversation between a man and a woman with little extraneous sound. 80% of the 15 segment boundaries were found to within half a second, but there were 17 false alarms. When allowing for such inaccuracies as ± 0.5 second per boundary, a false alarm rate of one per $71/17 = 4.176$ seconds becomes significant; many of the "detections" may well have been coincidental. Testing on a 5 minute stream with three participants yielded similar results: 78.6% detection rate, but with 94 false alarms. (For comparison, Siegler, whose assumptions are most similar to our own but does not use phoneme information,

[2]Although we did not implement it as its non-causal nature conflicts with our design goals, a bottom-up "agglomerative" clustering technique is another alternative here.

reported a 64% correct detection rate but 60% of the detections were false alarms [71].) While these results are respectable given only one other researcher's work with which to compare, the dominance of the false alarms makes them nearly useless for the temporal structure algorithms presented in Chapter 6.

Armed with these observations, the difficulty of the speaker segmentation problem in continuous audio streams (where there are no other cues, such as dead air in the air traffic controller case) is clear. The central problem is that the test segments are too short to adequately average out the intra-speaker variations, yet making them longer means that short segments will be completely missed (and many speech segments on commercial television are less than 2 seconds long). A number of authors use partial or complete *a priori* training of speakers in order to obtain reasonable results; such a technique could be employed here at the expense of generality and completely automated operation. Alternatively, non-audio cues, such as video shot boundaries or closed-captioning information, can be used to aid in segmentation (but neither is a very reliable indicator of speaker changes). What is needed is an audio statistic that responds more quickly and reliably to changes in speaker, thus not requiring long-term averages for discrimination.

If one is given the segments *a priori*, either via manual segmentation or from cues such as closed-captioning information (when accurate), the distance metrics discussed above are reasonably good at discriminating segments spoken by the same person from those spoken by different people. Aligning with the prior work, using simple mel-weighted cepstral averages—ignoring non-voiced segments—proved to be the best method (in fact, it even slightly beat out the "phoneme-aligned" distances described above, presumably because it uses more cepstral data in forming the means). Cepstral vectors in this case are calculated at 25 Hz, as the assumption of a single speaker makes for more slowly varying statistics. Figure 4.1 shows the measured distributions of this distance metric where the two segments in question are from the "same," "similar," or "different" speakers (more precise definitions for these terms will be given in Section 4.4).

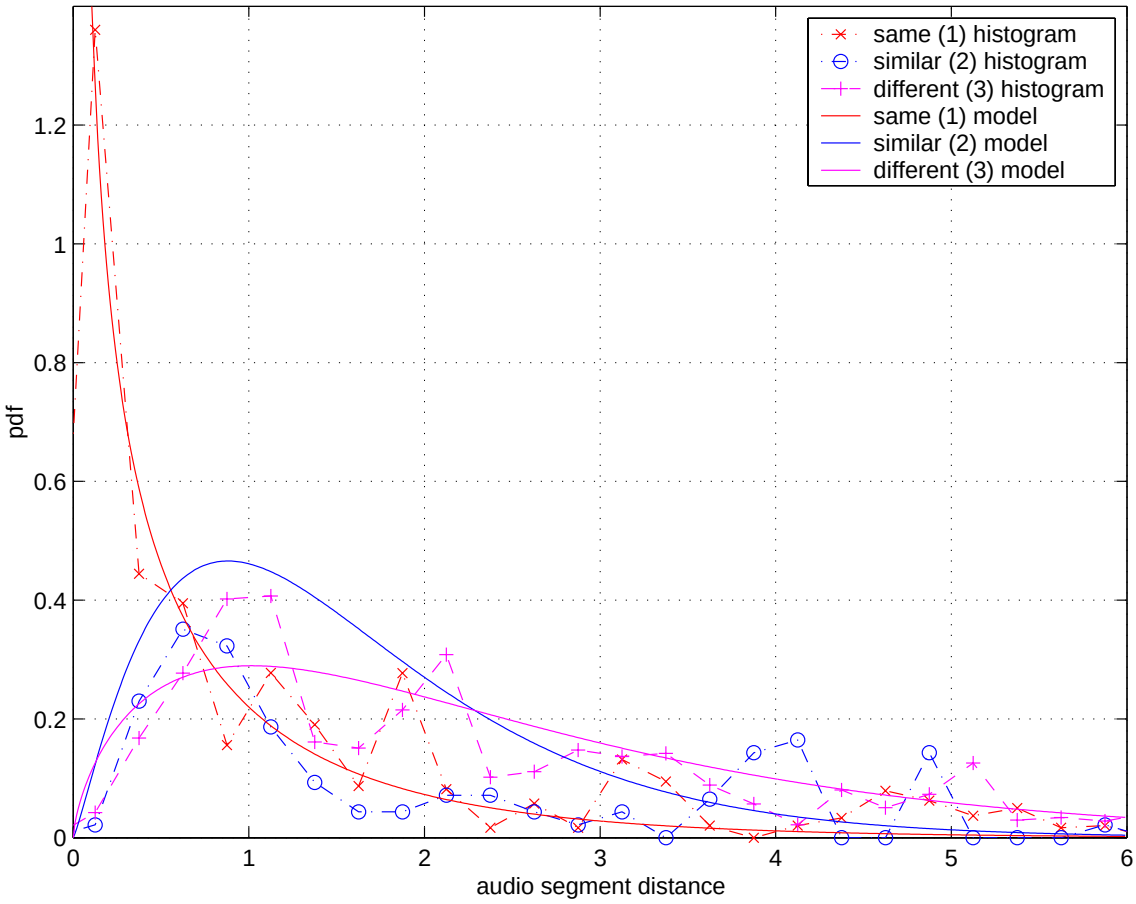


Figure 4.1: Empirical distributions for the cepstral mean audio shot distance metric, with MFCC's calculated at 25 Hz. Dashed lines are measured histograms, and solid lines represent models for each distribution, developed in Section 4.4.

4.3 Video Shot Distance Metric

Yeung, among others, developed distance metrics for measuring the similarity of two video shots [3]. These metrics generally involve choosing a "key frame" from each shot, then using an image-based distance metric between the two key frames. The key frame can simply be the first or center frame of a shot, or it can be chosen more intelligently through motion analysis and other techniques [94, 95]. A significant shortcoming, however, is that single key frames are unable to represent changes that occur in a shot; in many situations, the action at the end of a long shot has little to do with that at the beginning. Additionally, when one conceives of a distance between two shots in the *same* video stream, what is generally meant is "how similar is the *end* of the first shot to the *beginning* of the second?" (This is not, however, the prime characteristic of good distance metrics for searching through a temporally unordered shot database.) Two shots should have a low distance value if no off-camera action would be needed to transition between them in the real world; this is independent of whether the two had differing amounts of motion, for example.

We therefore develop a simple two-key-frame video shot distance metric. Using the very first and last frames of a shot as representative key frames is generally unwise, as they will be susceptible to errors in the shot segmentor (particularly for gradual transitions). Instead, we use the frames closest to 10% and 90% through the duration of the shot as the "entering" and "exiting" key frames, $K_{enter}(i)$ and $K_{exit}(i)$ for shot i, respectively[3]. In our implementation, we use easily-extracted DC-resolution key frames to avoid camera jitter and noise issues, and we restrict ourselves to I and P frames. To compute the distance between shots j and k, where $k > j$, use shot j's "exiting" key frame and shots k's "entering" key frame:

$$D_{j,k} = d\left(K_{exit}(j), K_{enter}(k)\right). \tag{4.1}$$

[3]Techniques that extract multiple key frames per shot, such as [94], may also be used; entering and exiting key frames are then selected from the set of extracted candidates.

$d(\cdot,\cdot)$ represents an image distance metric; motivated by Yeung's work and Boreczky's comparison study of image distance metrics for dissolve detection, we use a regional histogram distance [22]. This method incorporates spatial changes by splitting the image up into a number of blocks, then computing the intersection of each block's histogram across the two images:

$$d\left(f_j, f_k\right) = \frac{1}{N_{blocks}} \sum_{b=1}^{N_{blocks}} \left[\frac{1}{N_{pixels}} \sum_{h=1}^{N_{bins}} \min\left(F_{b,j}(h), F_{b,k}(h)\right) \right], \tag{4.2}$$

where $F_{b,j}(h)$ is bin h of frame j's histogram, only considering pixels from block b.

We found the greatest separation of distance values for subjectively "different" and "same" or "similar" shots when using a 3×2 array of histogram blocks per image ($N_{blocks} = 6$) and 6 histogram bins per color dimension (R, G, and B). Histograms of distance values for each of these three classes are shown in Figure 4.2. It's possible that improvement could be gained by using non-uniformly-spaced blocks or a different color space, but we have not pursued those ideas.

(Note that this distance metric might not completely characterize segment pairs in televised sports events; it is likely that most shots of a football field, for instance, would elicit a small measured distance. One possible solution in the case of sports is to use a distance metric more attuned to edge information.)

4.4 Audio/Video Distance Normalization

In order to make meaningful comparisons between audio and video distance metrics, it is necessary to normalize the distances in such a way that an audio segment distance of d is perceptually equivalent to a video segment distance of the same value. Simply dividing both distance metrics by their maximum values is not sufficient, as the perceptual distance in each case is a nonlinear function of the measured distance. For example, even if both audio and video distance metrics range from 0 to 1, a video distance of 0.5 might signify two perceptually "closer" shots than an audio distance of 0.5.

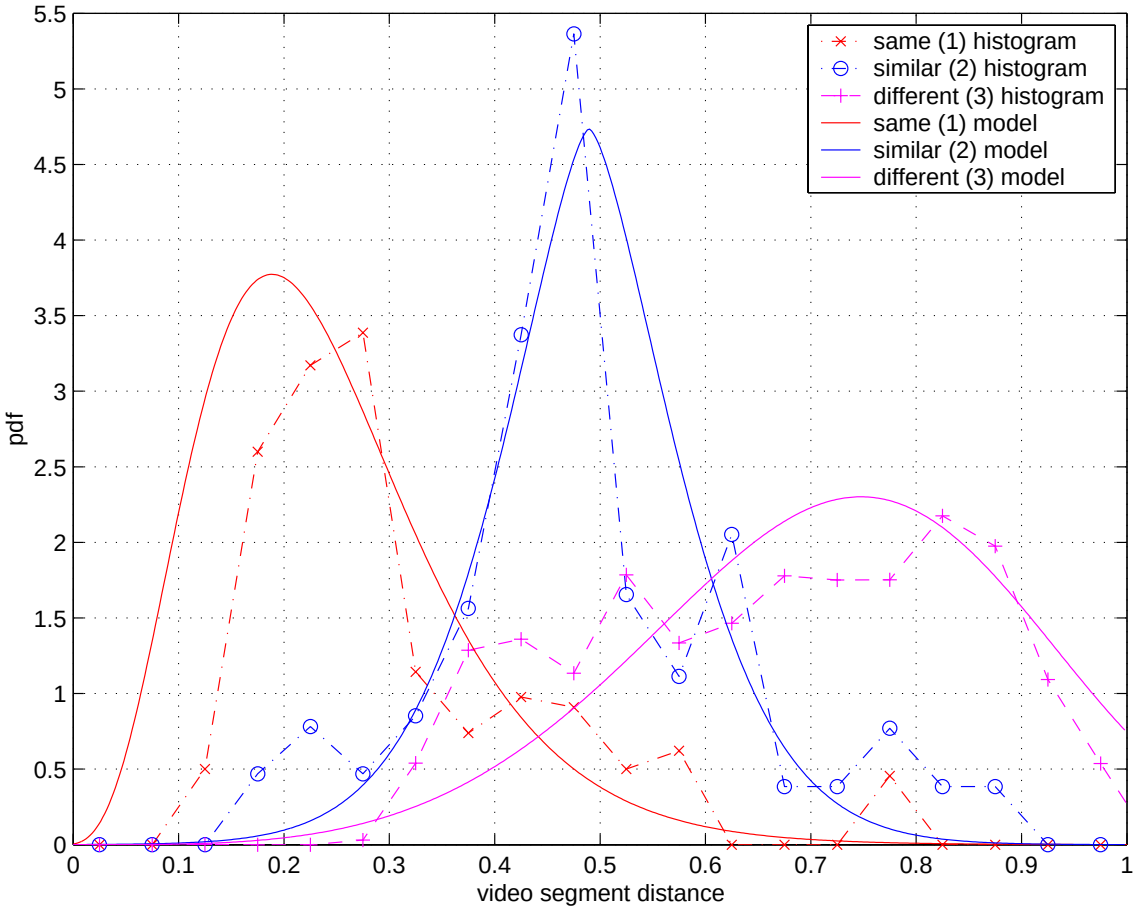


Figure 4.2: Empirical distributions for the two-key-frame video shot distance metric, with $N_{blocks} = 6$ and 6 bins per histogram dimension. Dashed lines are measured histograms, and solid lines represent models for each distribution, developed in Section 4.4.

Determining the entire nonlinear mapping between measured and perceptual distances would involve a significant psychological study with numerous subjects and sample clips, as it is difficult for most people to quantify how perceptually different two segments are (what does "twice as different" mean?). Instead, we adopt a three-step quantization approach: we declare two segments are either the "same," "similar," or "different." We identify these as classes 1, 2, and 3, respectively, and define the three classes as follows:

"same" (1): Two segments that are of the same source in the same context: two video shots that could occur without any intervening camera motion or off-camera action, or two audio segments from the same speaker with the same background sounds.

"similar" (2): Two segments of the same source material, but recorded in a different manner: two video shots of the same location but from different points of view, or two audio segments of the same speaker but under different conditions (e.g., a news correspondent on the street and the same correspondent recorded in the studio).

"different" (3): Two segments that do not fit into either class 1 or class 2; they have no clear physical relationship to each other. They may have a higher-level semantic relationship, but considering the images or sounds alone, are "different."

(Naturally, there is some overlap in these three classes, and reasonable people could argue whether a particular pair of segments fits into a given category.) This rough quantization should be sufficient, because in most cases when two shots are different, exactly how different they are is less relevant. More quantization steps would complicate the manual training and classification process, while not providing much more detailed information for our goal of distance normalization.

Given these three quantization steps, we may then view the distance normalization process as a parameterized 3-hypothesis detection problem [40]. The prior probabilities for each class depend on how many segments separate the two test segments, as segments that are temporally closer are more likely to be in classes 1 or 2. To compute these priors, as a

function of segment separation, we manually labeled distances for audio and video segments from 14.5 minutes of television data, including sitcoms, news, and documentary footage[4]. Out of 1321 pairs of audio segments, a total of 126 pairs were declared to be in the "same" class, 67 pairs were declared "similar," and 1128 were "different." 62 pairs of video shots were labeled as the "same," 61 pairs as "similar," and 2747 pairs as "different." A plot of the audio prior probabilities, as a function of segment separation, is shown in Figure 4.3; video priors are shown in Figure 4.4. The dominant feature in both prior functions is the period-2 oscillation; this is due to the predominance of conversational audio and video scenes. As segments are further separated temporally, the likelihood that they're part of the same dialog diminishes, as does the oscillation in the prior function. Naturally, the probability of shots being different also increases gradually as time separation increases.

While we could use the raw data directly, we instead model the prior functions analytically using the sum of an exponential and an exponentially-decaying sinusoid (which reduces to a $(-1)^n$ factor due to the period of 2).

$$\pi_i(s) = \alpha_i + \beta_i e^{-\gamma_i(s-1)} + \delta_i e^{-\epsilon_i(s-1)} \cos(s\pi) \tag{4.3}$$

is the prior probability of class i for two segments s apart (where $s=1$ signifies adjacent segments). Parameters α_i, β_i, γ_i, δ_i, and ϵ_i for audio and video were derived from the measured data for each class i; the resulting model prior functions for audio are also plotted in Figures 4.3 and 4.4. (Models for video priors needed a bit of tweaking at $s=1$ and $s=2$ because "similar" video shots become an important phenomenon at those separations, while audio segment similarity is not as common at $s \leq 2$.)

Next, distributions for the distance metrics to be normalized need to be considered. The commonly-used Gaussian assumption for cepstral coefficients motivates a gamma distribution model for our audio segment distance metric [65]. The two gamma distribution

[4]We neglect any errors introduced by having only one person label classes instead of a larger sampling of viewers. For a more complete characterization, both more footage and more viewers would be needed, but the process is the same.

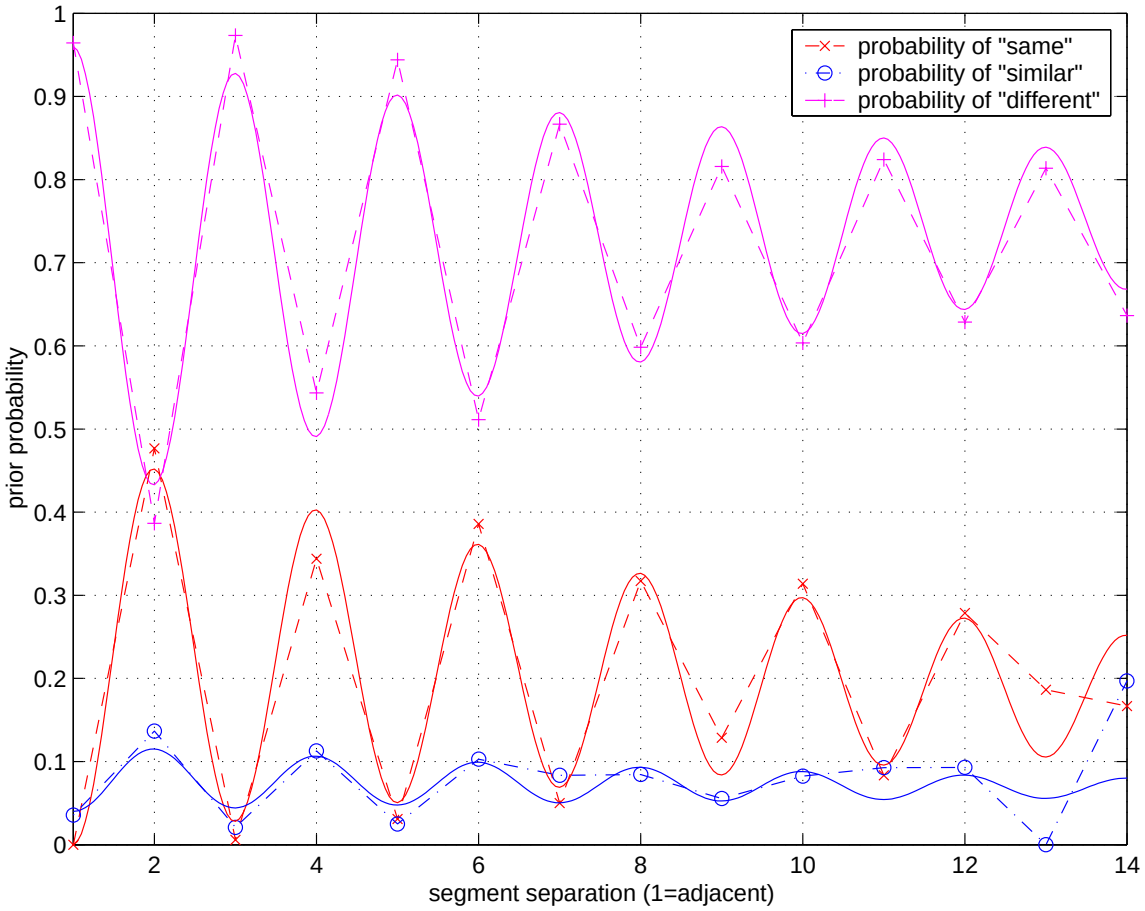


Figure 4.3: Prior probabilities of "same," "similar," and "different" speaker segments as a function of the number of segments separating them. Dashed lines represent measured values from test audio data, while solid lines are the models discussed in the text.

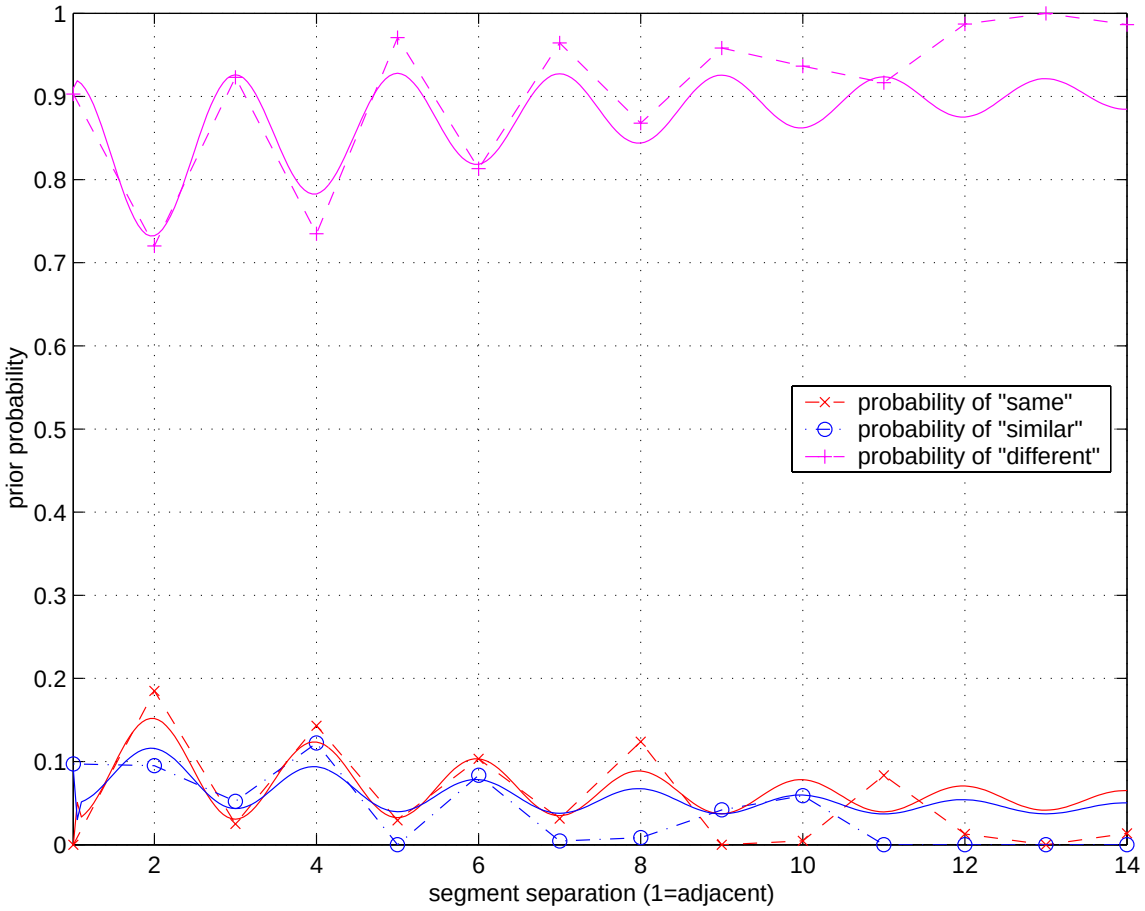


Figure 4.4: Prior probabilities of "same," "similar," and "different" video shots as a function of the number of shots separating them. Dashed lines represent measured values from test video data, while solid lines are the models discussed in the text.

parameters for each class were selected to provide the best mean-square fit to the distance histogram; the resulting models are shown in Figure 4.1. Modeling the regional histogram video shot distance metric proved more difficult, as there is no commonly agreed-upon model for image histogram intersections. Nonetheless, the data seem to reasonably fit a "skewed" Gaussian model, with the distributions then scaled to fit within the range of possible distance values. The video models are of the form:

$$p_i(x) = \kappa_i e^{\frac{-\left(x^{\lambda_i} - \mu_i\right)^{2+\eta_i}}{2\sigma_i^2}} \qquad 0 \le x \le 1. \tag{4.4}$$

Video class 3 ("different") distances fit slightly better with a Weibull distribution; Weibull parameters α and β were also selected for a best mean-square fit to the data. All three class models are shown in Figure 4.2.

Given the densities for each class and the prior probability functions, a cost matrix $\mathbf{C}$ is needed in order to formulate the detection problem. $c_{i,j}$ is the cost of choosing class i when the perceptual distance was actually in class j. For testing purposes, we used

$$\mathbf{C} = \begin{bmatrix} 0 & 0.5 & 2 \\ 0.5 & 0 & 1 \\ 2 & 1 & 0 \end{bmatrix}, \tag{4.5}$$

as confusing classes 1 and 2 is not as serious as confusing 1 and 3. The matrix $\mathbf{C}$ need not be symmetric, if for example it is very important not to associate segments that are unrelated.

Determining the thresholds separating each class, as a function of segment separation, must then be done. Assuming the cost of correct detection, $c_{i,i}$, is zero, we choose class 1 ("same") when the following two conditions are true:

$$p_1(x)c_{3,1}\pi_1(k) \;\geq\; \pi_2(k)\left(c_{1,2} - c_{3,2}\right)p_2(x) + \pi_3(k)c_{1,3}p_3(x) \tag{4.6}$$

$$p_1(x)c_{2,1}\pi_1(k) \;\geq\; \pi_2(k)c_{1,2}p_2(x) + \pi_3(k)\left(c_{1,3} - c_{2,3}\right)p_3(x) \tag{4.7}$$

where x is the measured segment distance, k is the temporal separation between the two segments under test, $p_i(x)$ is the density function of the distance metric under class i, and

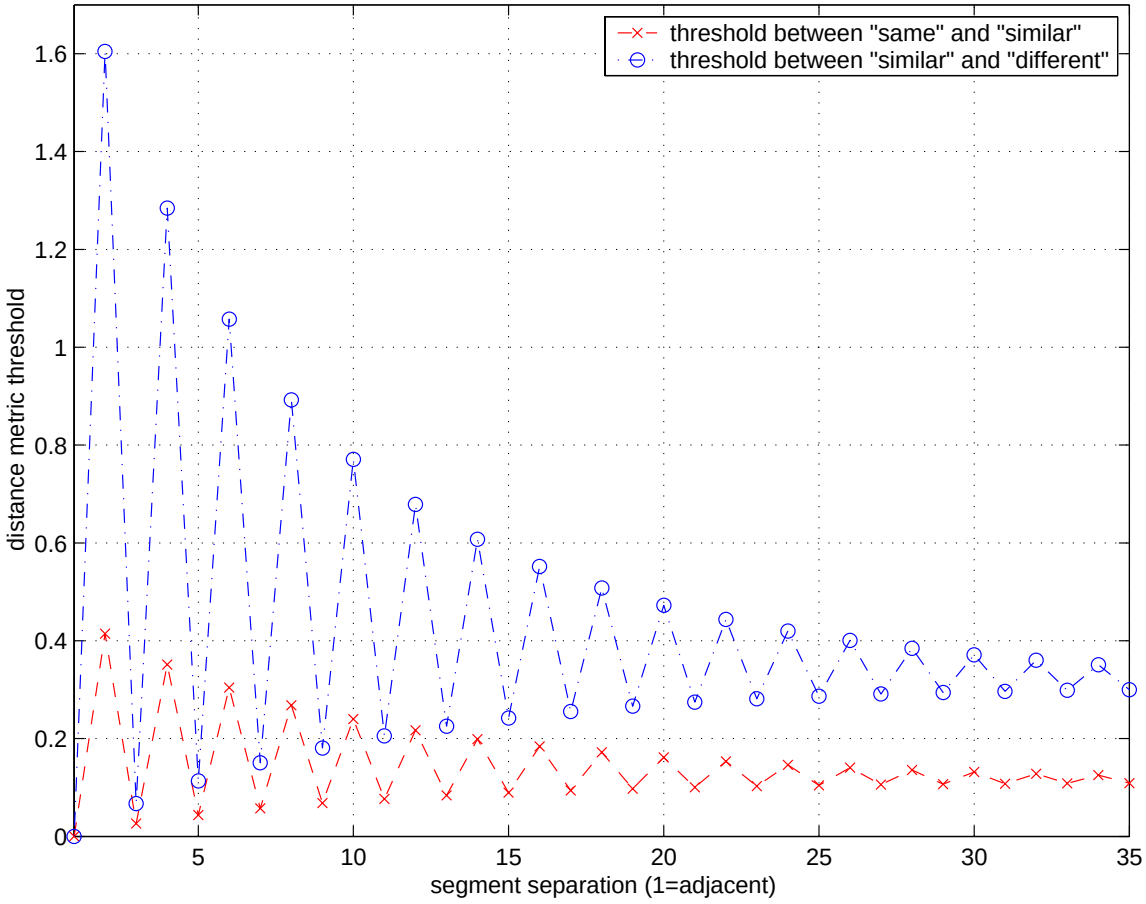


Figure 4.5: Minimum-cost thresholds dividing measured audio distance values into three subjective regimes, "same," "similar," or "different," as a function of the number of audio segments separating the two test segments.

$\pi_i(k)$ is the prior probability of class i at separation k. Similar equations can be derived for classes 2 and 3. Given our model distribution functions, the decision regions for a fixed segment separation can be represented by two thresholds: one threshold distinguishing classes 1 and 2, and another distinguishing 2 from 3. The minimum-cost threshold values computed for our cepstral audio segment distance metric, using the value for $\mathbf{C}$ given in (4.5), are shown in Figure 4.5. The thresholds for the regional histogram video shot distance metric are shown in Figure 4.6.

The errors in detection can be expressed as "confusion" matrices, where element (i, j)

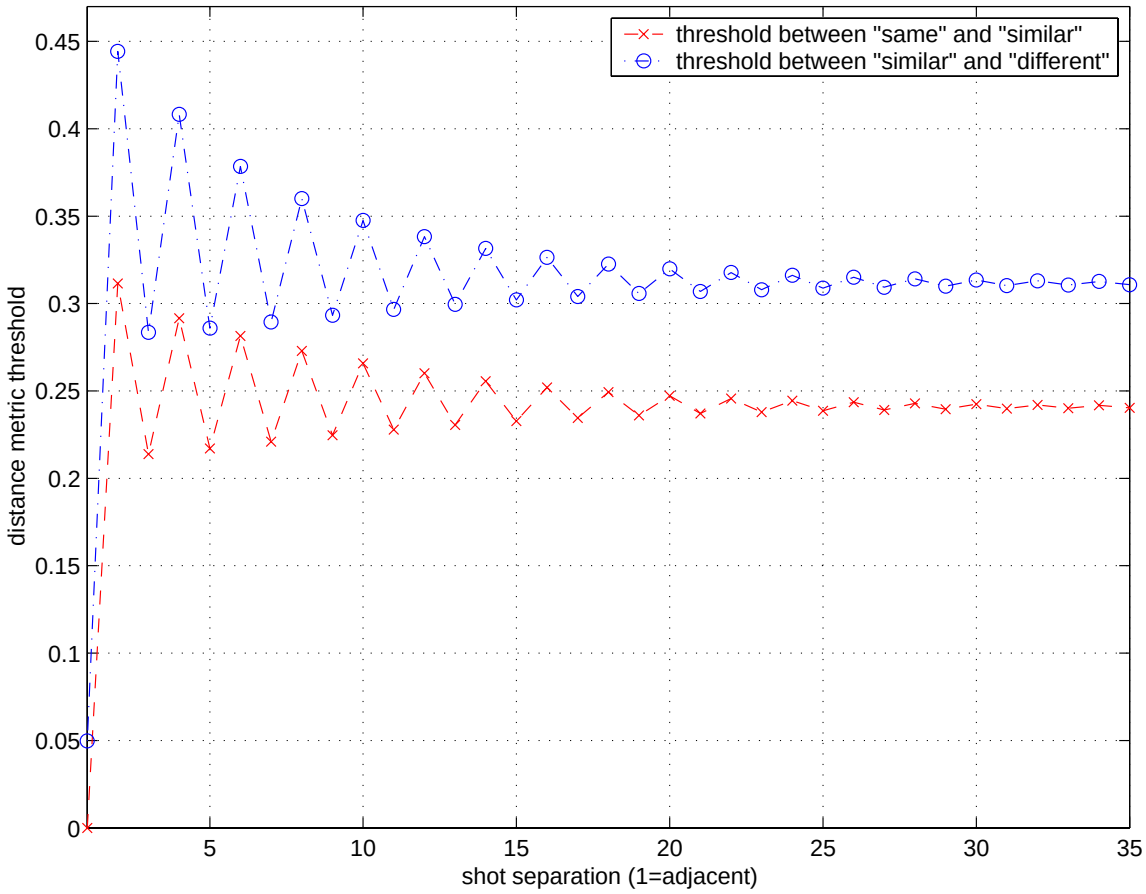


Figure 4.6: Minimum-cost thresholds dividing measured video distance values into three subjective regimes, "same," "similar," or "different," as a function of the number of video shots separating the two test shots.

is the number of distances that are detected as class i but are truly in class j. Using the same 14.5 minute set of television data, the confusion matrices for the detection process on our audio and video distance metrics are

$$\mathbf{E}_{video} = \begin{bmatrix} 33 & 9 & 2 \\ 13 & 9 & 11 \\ 16 & 42 & 2681 \end{bmatrix} \quad \text{and} \quad \mathbf{E}_{audio} = \begin{bmatrix} 17 & 1 & 9 \\ 28 & 12 & 56 \\ 81 & 54 & 1063 \end{bmatrix}. \tag{4.8}$$

These correspond to a overall error rates of 46.7%, 85%, and 0.48% for the three classes of video distances, and 86.5%, 82.1%, and 5.76% for the same classes in the audio case (even though, as the cost matrix indicates, some of the errors are less critical than others).

Looking back at this three-hypothesis detection problem, one might wonder whether the "similar" distance class (2) is really needed. In fact, the cost matrix $\mathbf{C}$ controls whether this class is detectable, and thus whether it is necessary. Cost matrices that place less relative weight on the class 1 versus 3 errors, for example

$$\mathbf{C}_2 = \begin{bmatrix} 0 & 0.5 & 1 \\ 0.5 & 0 & 0.75 \\ 1 & 0.75 & 0 \end{bmatrix}, \tag{4.9}$$

yield a number of segment separations at which the two computed thresholds are equal. This indicates a null region for class 2, meaning that only classes 1 and 3 are distinguishable. In general, the cost matrix $\mathbf{C}$ can have a dramatic effect on the errors made by the distance class detector described here; for any particular application, $\mathbf{C}$ needs to be carefully considered, particularly in light of the relatively low prior probabilities of "same" (class 1) segment pairs.

Finally, once the detector has been run over the raw distance metric output, one must then assign subjective distance values that correspond to each class. We denote the vector mapping class numbers to perceptual distances as d_{norm}. For simplicity, we assign a perceptual distance of 0 when two shots are in the "same" class and 1 when they're in the

"different" class. As a heuristic judgement, we assign a perceptual distance of 0.3 to the "similar" class, giving

$$d_{norm} = \begin{bmatrix} 0 & 0.3 & 1 \end{bmatrix}. \tag{4.10}$$

While the process of perceptual normalization is of interest in an abstract sense, we shall see in the next chapter its utility in media representations which combine audio and video distance information. The combined representations treat audio and video distance values equally and provide mechanisms for comparing segments across modalities. Without normalization, such combined representations make little sense as each distance metric's values have different real-world meaning. This process is of course not limited to the particular distance measures we use; with the development of more discriminating segment distance metrics, the normalization methods described here can be easily repeated with the new metric's set of PDF's.

In the chapters that follow, we shall assume all segment distances are normalized in this way, using the cost matrix $\mathbf{C}$ given in equation (4.5) and the d_{norm} of (4.10).

Chapter 5

Association Matrices

Individually, raw segment distance measures—even perceptually normalized ones—are only useful in specific circumstances, such as testing queries in a search engine. Instead, we would like to extract and incorporate such distances in a media representation that facilitates further processing. Such processing could include detection of "important" events (by some definition), generation of summaries, or extraction of long-term content information.

One method of visualizing and interpreting segment distance information (and thus, as we shall see in Chapter 6, temporal structure) is via sets of distance matrices. Foote generated audio self-similarity matrices by applying correlation techniques directly to audio streams [96]. Motivated by this technique, we describe a general method to combine distance matrices from multiple modalities in Section 5.2, then simplify the representation by restricting ourselves to three specific metrics in Section 5.3. A straightforward application of this representation is described in Section 5.4, where "idiomatic" a/v sequences are detected.

5.1 Prior Methods Combining Video and Audio Information

Before diving into our multimedia representation techniques, it is worthwhile to review other methods that have been used to combine video and audio content information in the

analysis of media streams.

Traditionally, multimedia content analysis work has focused on video only, so many techniques tend to use audio as a support to video analysis, rather than combining audio and video on equal ground. As noted before, Chang, *et al.*, used speech recognition to pick out key words such as "touchdown" in sporting events, then used video segmentation and edge detection to identify playing-field shots [80]. Abrupt changes in audio (but not necessarily speaker changes) were used by Huang, *et al.*, to differentiate shot boundaries from higher-level scene breaks [81]. A music versus speech discriminator formed a critical part of Minami's video browser, which used the audio information to form temporal blocks [89].

Joint audio and video information has been exploited by a number of recent authors. Saraceno combined a video cut detector and an audio classification tool (discriminating silence, speech, music, and "noise") with heuristic rules to find scenes and commercial breaks based on the coincidence of audio and video transitions. Her later work incorporated low-resolution frames and other features to recognize dialog, action, and story sequences, yielding a roughly 85% detection rate [5, 97]. In addition to his audio-based work noted in Chapter 4, Sundaram also examined the correlation between audio and video segment boundaries [79]. Carnegie Mellon's Informedia indexing system combined audio speech recognition with caption-reading and closed captioning information to generate name-to-face mappings [90].

Speaker and program recognition can only be improved by the fusion of audio and video classifiers. Liu, *et al.*, used a number of audio and video "clip"-level features as inputs to a classifier to detect news reporting; they found that audio features are far more useful in this detection problem [86]. Trained speaker identification in the audio and video (face) domains was combined by Neti, *et al.*, to improve recognition results [87]. Finally, Pan, *et al.*, looked at the joint audio/video classification problem from a more abstract point of view, dividing methods into signal-level fusion, feature-level fusion, and decision-level fusion; employing the latter form, they used a neural network to do the required joint

density function estimates [91].

5.2 Association Matrix Construction

One of the central goals of combining distance metric information from multiple modalities is to associate segments that would not otherwise be seen as similar. For example, if two video shots have visually different content, but it is determined that the same speaker dominates both, then the video shots are in some sense similar and this fact should be detectable in an automated fashion. We therefore call our stream representation, which combines the results from a number of distance metrics across multiple modalities, an "association" matrix.

The construction an association matrix starts with a given set of distance metrics, $\{m_1, \ldots, m_K\}$, taken over segments of video, audio, or other information such as closed-captioning or studio notes. m_k, for instance, may be a face recognition technique that compares two video shots to determine whether the same face appears in both. A vector of segments appropriate to the distance metrics is then composed,

$$S = [S_{m_1} \quad S_{m_2} \quad \cdots \quad S_{m_K}], \tag{5.1}$$

where each S_{m_k} is a row vector of segments appropriate to distance metric m_k. If, for example, $K = 2$, m_1 is a video shot distance metric, and m_2 is an audio speaker distance metric, then S is a row vector consisting of the video shots followed by the audio (speaker) segments. (An element in S is a "segment" in the abstract sense; the actual segment data need not be copied. One can alternatively think of S as holding segment indices.)

The association matrix $\mathbf{A}$ is a block matrix of distance matrices, where the blocks are determined by the block vector structure of S. We define the elements of $\mathbf{A}$ as follows: $a_{i,j} = \mathcal{D}_{i,j}(s_i, s_j)$, where s_i is the i-th scalar element of S and $\mathcal{D}_{i,j}$ is the distance metric appropriate to segments s_i and s_j. If s_i and s_j are both parts of the same block, k, of S, then $\mathcal{D}_{i,j} = m_k$. Otherwise, if one is defined, $\mathcal{D}_{i,j}$ is a distance metric across the respective segments (which may not be of the same media). If no distance is defined or appropriate,

$a_{i,j} = 1$ (our defined maximum distance). In order for the intra- and inter-media segment distances to be meaningfully compared, they must all be normalized to a common perceptual standard using the techniques of Section 4.4.

The block matrix structure of $\mathbf{A}$ is

$$\mathbf{A} = \begin{bmatrix} \mathbf{D}_{1,1}(S_{m_1}, S_{m_1}) & \mathbf{D}_{1,2}(S_{m_1}, S_{m_2}) & \cdots & \mathbf{D}_{1,K}(S_{m_1}, S_{m_K}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{D}_{K,1}(S_{m_K}, S_{m_1}) & \mathbf{D}_{K,2}(S_{m_K}, S_{m_2}) & \cdots & \mathbf{D}_{K,K}(S_{m_K}, S_{m_K}) \end{bmatrix}. \quad (5.2)$$

Each block, $\mathbf{D}_{k,l}$, is a distance matrix: element (i, j) is the distance from segment i to segment j. (Another possible interpretation of $\mathbf{A}$ is as a $K \times K \times 2$ array.)

The association matrix will have block-symmetry if $\mathbf{D}_{k,l} = \mathbf{D}_{l,k}^T$ for $k \neq l$, which is to say that the chosen cross-modality distance metrics are symmetric. For example, an audio-video cross-modality distance metric is symmetric if the distance from a particular video shot s_v to an audio segment s_a is equal to the distance from s_a to s_v. If in addition each distance metric m_k is self-symmetric, then each $\mathbf{D}_{k,k}$ will be symmetric, as will the entire association matrix $\mathbf{A}$.

For long streams, the association matrix can grow to unwieldy sizes at a rate of roughly N^2; one possible way to deal with this issue is to select only segments of particular interest. In the case of causal processing, one could use all the segments within some time window of the past, and beyond that point only preserve representative segments of important characters or scenes (provided that the existence of these can be detected within the time window).

5.3 A/V Association Matrices

The formulation above allows for a great deal of flexibility in choosing distance metrics and segment types; it is meant to be a general framework for combining multiple distance metrics' information in single matrix. In order to actually compute association matrices,

we need to restrict ourselves to a concrete set of distance metrics.

Given the development in Chapter 4, we will set $K = 2$ and use

m_1: the regional histogram video shot distance (Section 4.3)

m_2: the cepstral audio speaker segment distance metric (Section 4.2)

Both these distance metrics are first normalized according to the techniques of Section 4.4; their values are therefore restricted to those in equation (4.10). The segment vector $S = [S_{m_1} \quad S_{m_2}]$ is therefore the set of video segments, followed by the set of audio segments.

The accuracy of the segmentation schemes used to determine S is of paramount importance, as the distance metrics will be meaningless if they are computed over time intervals containing more than one segment! The characterization of these errors, and how they interact with the oscillating priors on the normalized distances (Figures 4.3 and 4.4) are complex issues that have not been satisfactorily studied. For the present work, *we assume that the video and audio segmentation are perfect*, which means they must be done manually. Given the video segmentation techniques described in Chapter 2, this is not an unreasonable assumption in the video domain. In the audio domain, however, perfect speaker segmentation is an elusive goal, made even more difficult by modern television's propensity for mixing music, voices, and sound effects on top of one another.

Using the two distance metrics given above, the association matrix takes on a much-simplified 2×2 block form:

$$\mathbf{A} = \begin{bmatrix} \mathbf{D}_{VV} & \mathbf{D}_{AV}^T \\ \mathbf{D}_{AV} & \mathbf{D}_{AA} \end{bmatrix}, \tag{5.3}$$

with $\mathbf{D}_{VV}$ the distance matrix among video shots and $\mathbf{D}_{AA}$ the distance matrix among audio segments. $\mathbf{D}_{AV}$, the audio-video distance matrix, is in our case simply based on the temporal overlap of the audio and video segments:

$$\mathbf{D}_{AV}(i,j) = 1 - \frac{\text{(overlap between audio segment } i \text{ and video segment } j)}{\min(\text{length of audio segment } i, \text{ length of video segment } j)} \tag{5.4}$$

(Note that video segment j is s_j, while audio segment i is s_{j+n_V}, where n_V is the number of video segments.) Under this metric, segment pairs where one is a strict temporal subset of the other have distance 0, which is appropriate when associating such audio and video segments. Non-overlapping segment pairs have distance 1, while partially-overlapping pairs have a distance roughly comparable to the perceptually normalized distance metrics among audio and video segments, so no additional normalization of $\mathbf{D}_{AV}$ is necessary.

Blocks $\mathbf{D}_{VV}$ and $\mathbf{D}_{AA}$ are symmetric[1], and $\mathbf{D}_{AV}$ is almost all ones except near the diagonal; these properties cut the computation time for $\mathbf{A}$ significantly. (As mentioned above, for very long streams or limited-memory situations, it may only be necessary to calculate $\mathbf{A}$ for the few dozen shots preceding and following one of interest, on the assumption that temporally distant events are less relevant. This abridged $\mathbf{A}$ may be augmented with audio or video segments deemed "important" by previous analysis, such as those of a news anchor.)

The association matrix for a seven minute segment of the PBS "Charlie Rose" talk show is shown in Figure 5.1; the matrix is split into blocks as in (5.3), with the upper-right matrix omitted for clarity. The first 55 seconds of this clip (video shots 1–10, audio shots 1–9) are a conversation between the host and one guest. Between the 55 and 65 second marks, the show's logo appears and music plays while a new guest is brought in. The host speaks between the 65 and 110 second marks, even though the video shots alternate (video shots 12–15). The remaining time consists almost entirely of a single guest speaking, but with a number of distinct video shots.

Although all axes in Figure 5.1 are time-based, they give little sense of the durations of different events and where events lie; many of the video shots occur in short bursts, for example. Figure 5.2 is a version of the association matrix with the columns and rows scaled in proportion to their segments' durations. Yet another visualization aid is to overlay the

[1] $\mathbf{D}_{VV}$ is, strictly speaking, only symmetric if segment pairs are temporally ordered before computing the distance in (4.1). In some limited applications, relaxing this condition may prove useful (to find stories told backward).

(time-scaled) $\mathbf{D}_{VV}$ and $\mathbf{D}_{AA}$ blocks in a translucent fashion; the result is Figure 5.3. In this plot, time segments where only video shots match are shown in magenta, time segments with the same audio (speaker) are cyan, and intervals where both audio and video match closely are shown in purple.

Another example of the association matrix and translucent A/V distance matrices is shown in Figures 5.4 and 5.5, respectively. The clip in this case is a 7.5 minute segment from the news broadcast of a local CBS affiliate. It begins with short previews of two stories, and at the 10 second mark, two anchors introduce the first story. Between 31 and 172 seconds into the clip, one reporter narrates and interviews a number of people on the street before returning to the newsroom. An unrelated but similarly-structured news story occurs between the 204 and 359 second marks. The two anchors read stories through the 424 second mark, then on-location images and a short preview appear before the clip ends.

Figure 5.6 contains the overlaid A/V matrices for a 71 second segment from the NBC sitcom "Frasier." The segment begins with 10 seconds of the character Frasier Crane walking through a radio studio, then the balance of the clip is a dialog between Frasier and another character. The audio segments form an exact dialog sequence, alternating speakers. The video, in addition to alternating between the speakers' faces, contains two additional shots with both characters in the frame (beginning at 15.2 and 62.9 seconds).

Figures 5.7 and 5.8 are generated from the first ten minutes of a broadcast of CBS's "The Late Show with David Letterman." As is clear from Figure 5.8, the first half of this clip is a monologue sequence. After that point, there is a short sequence of the show's band playing (319.8–340.3 sec), followed by a number of shots of David Letterman at his desk with only him speaking. He then interviews a staff member (403.7–547.2 sec), and finally cuts to a pre-recorded video clip of the staff member interviewing people on the street. Throughout the Letterman stream, audience applause and laughter are interspersed with the speakers; although many of the non-voice segments are properly ignored by our audio distance metric, those with mixed voices and applause only detract from the speaker

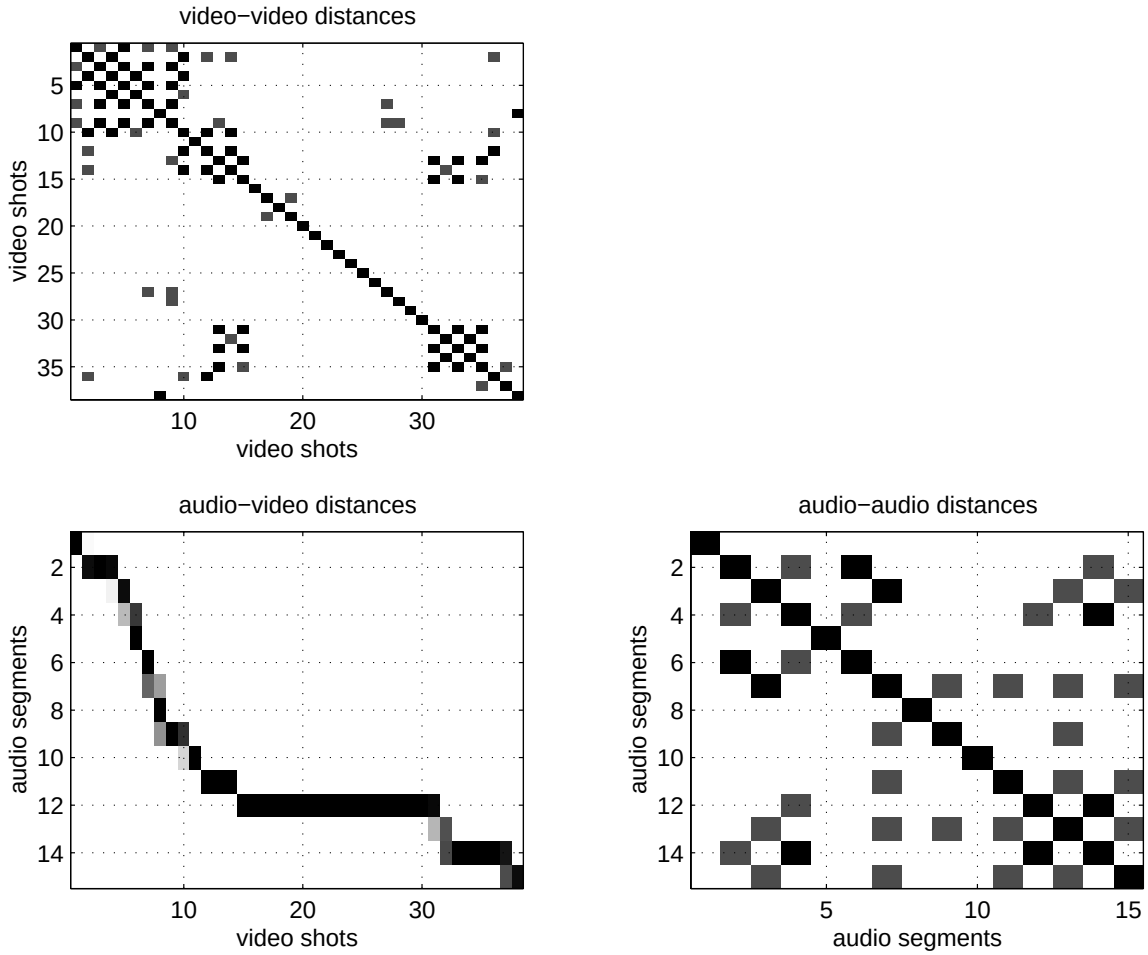


Figure 5.1: Perceptually normalized distance matrices for a 7 minute Charlie Rose interview clip. Columns of the upper- and lower-left matrices are time-aligned, as are rows for the lower-left and lower-right matrices. Darker shades represent smaller segment distances.

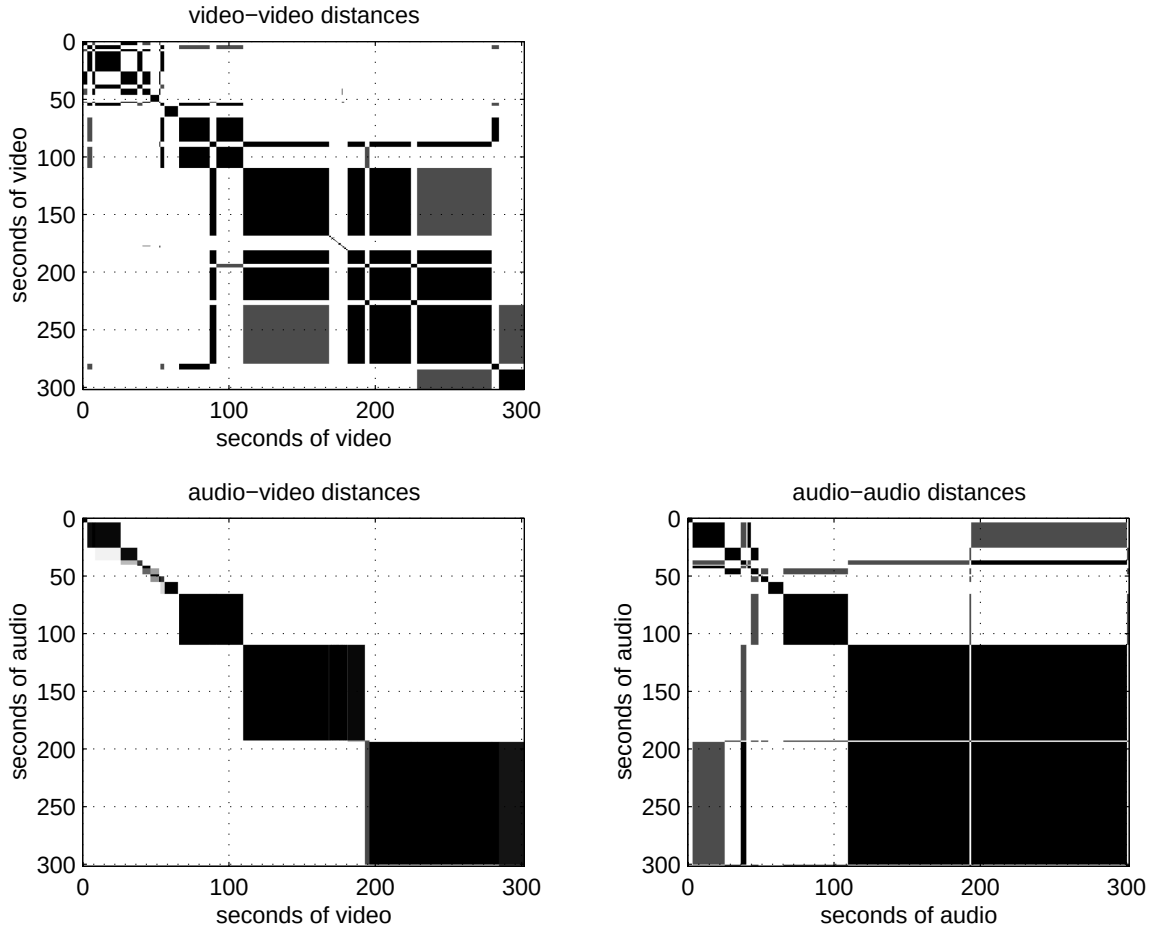


Figure 5.2: Perceptually normalized distance matrices for the 7 minute Charlie Rose interview clip, with element widths proportional to segment lengths in seconds. Columns of the upper- and lower-left matrices are aligned, as are rows for the lower-left and lower-right matrices. Darker shades represent smaller segment distances.

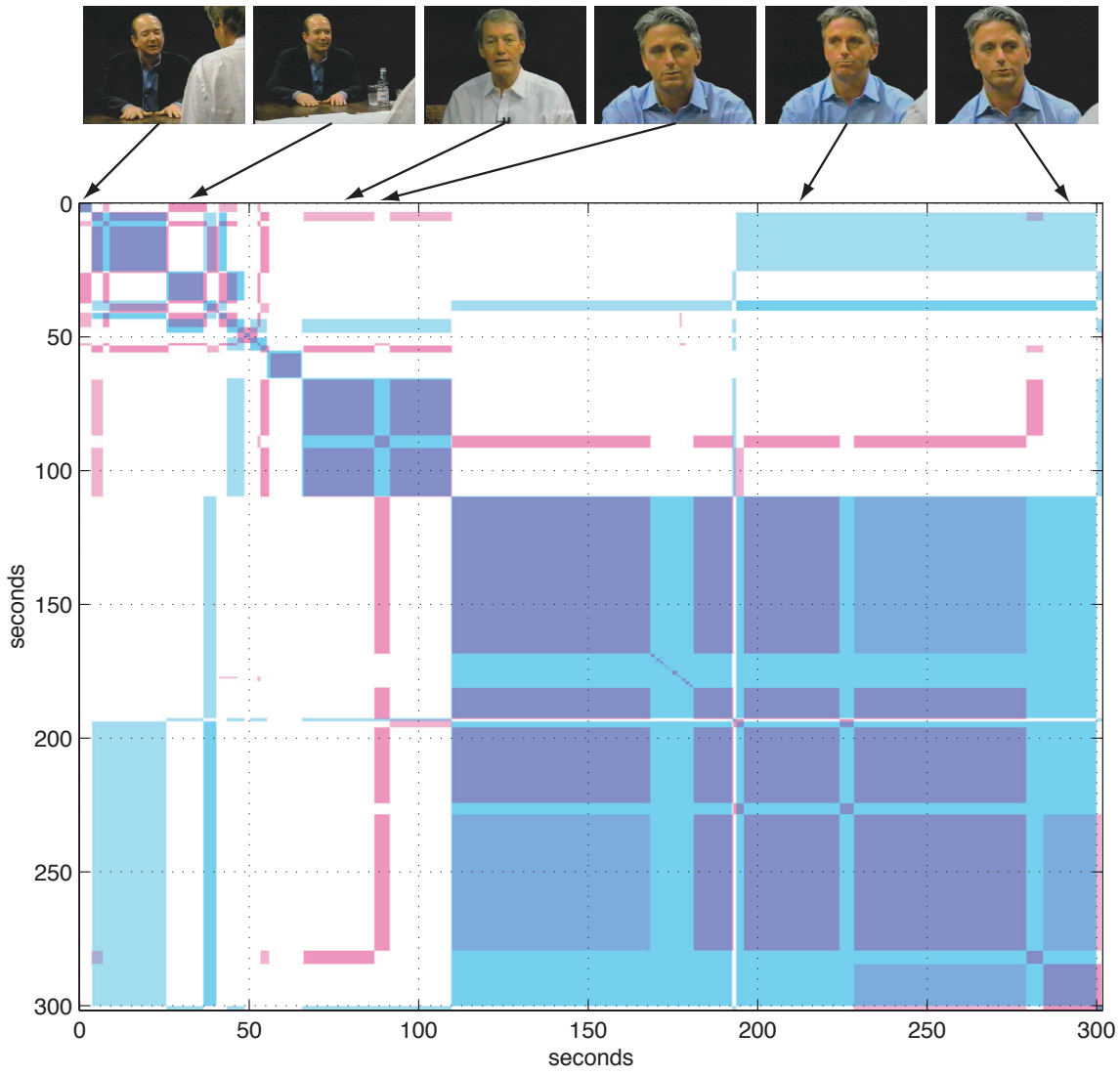


Figure 5.3: Charlie Rose audio-audio and video-video time-normalized distance matrices superimposed on one another. Magenta denotes close pairs of video segments, cyan denotes close pairs of audio segments, and purple represents intervals that are close in both the audio and video domains. Thumbnails at the top of the figure correspond to selected video shots.

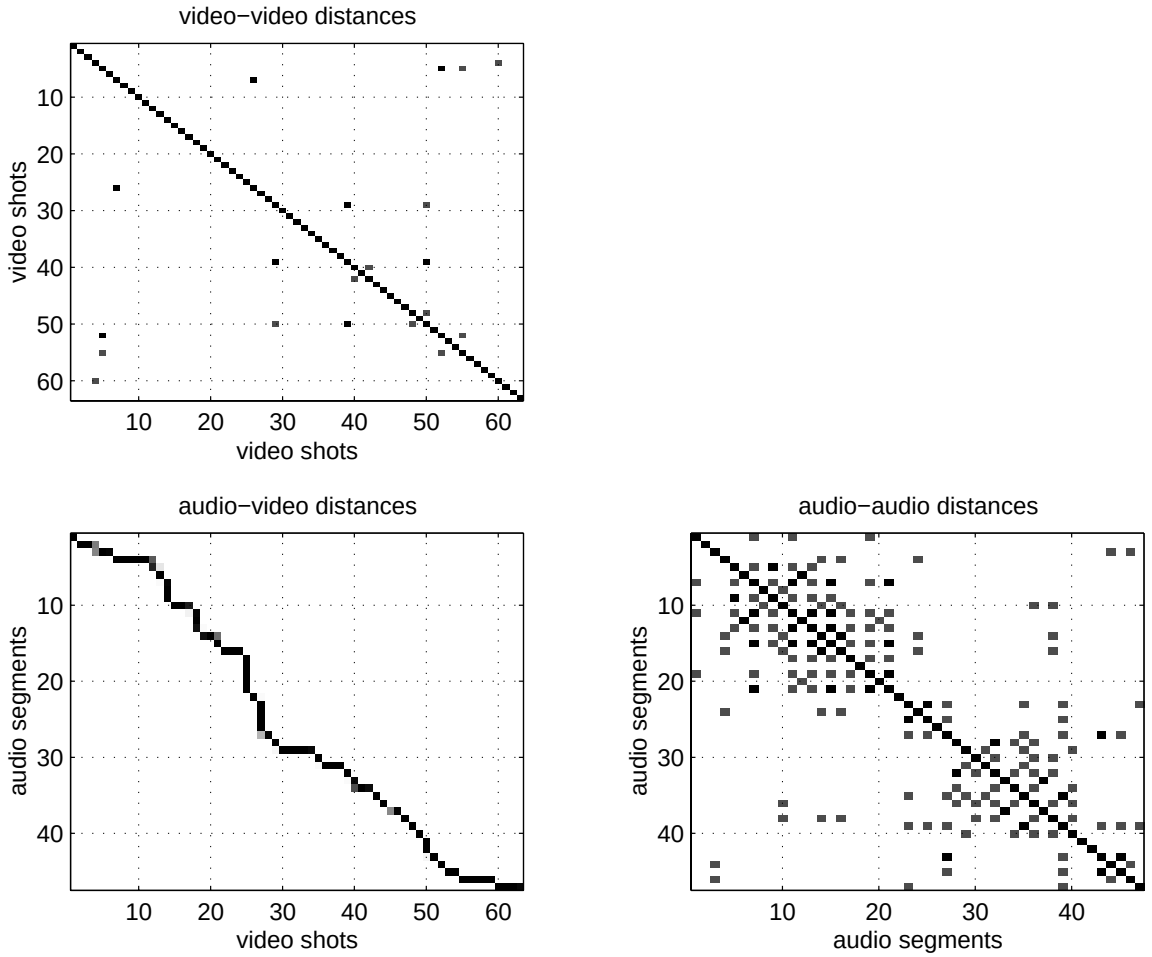


Figure 5.4: Perceptually normalized distance matrices for a 7.5 minute CBS news clip. Columns of the upper- and lower-left matrices are time-aligned, as are rows for the lower-left and lower-right matrices. Darker shades represent smaller segment distances.

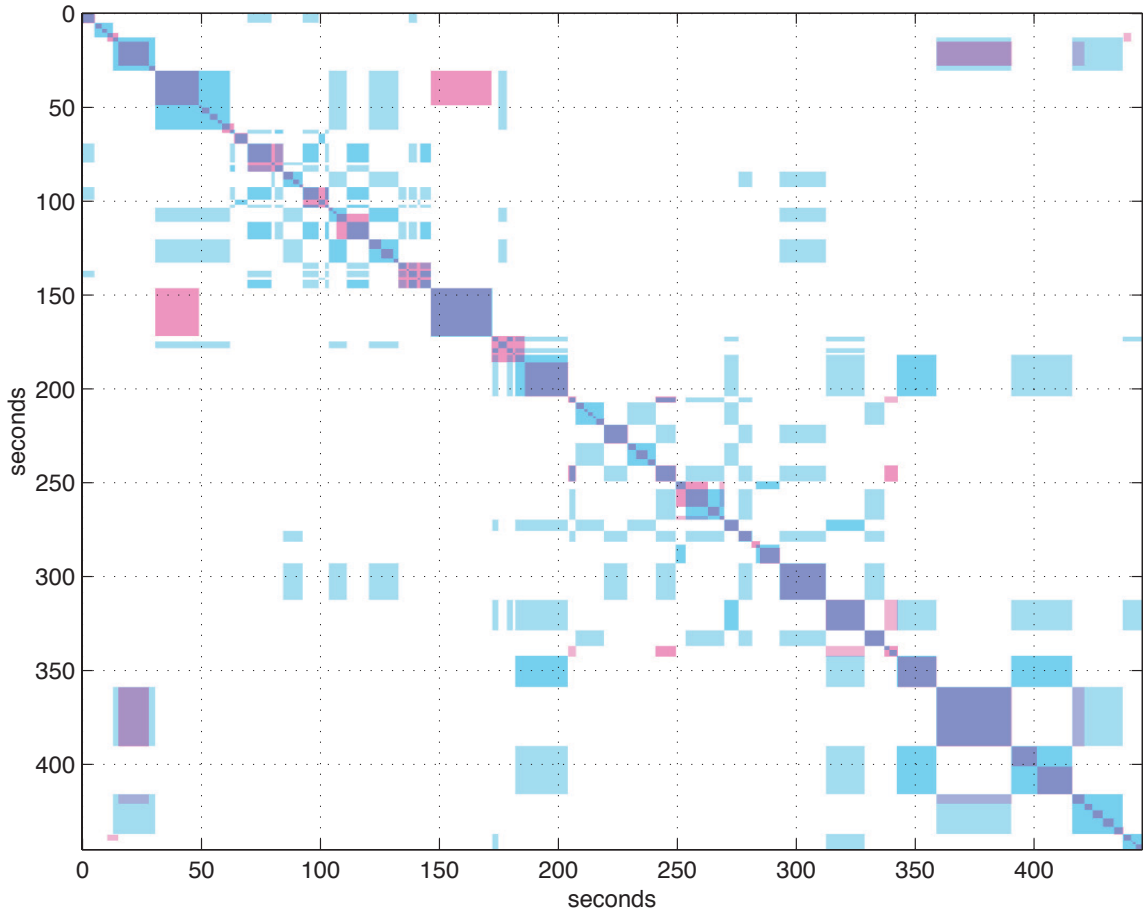


Figure 5.5: Superimposed audio-audio and video-video time-normalized distance matrices for a 7.5 minute CBS news clip. Magenta denotes close pairs of video segments, cyan denotes close pairs of audio segments, and purple represents intervals that are close in both the audio and video domains.

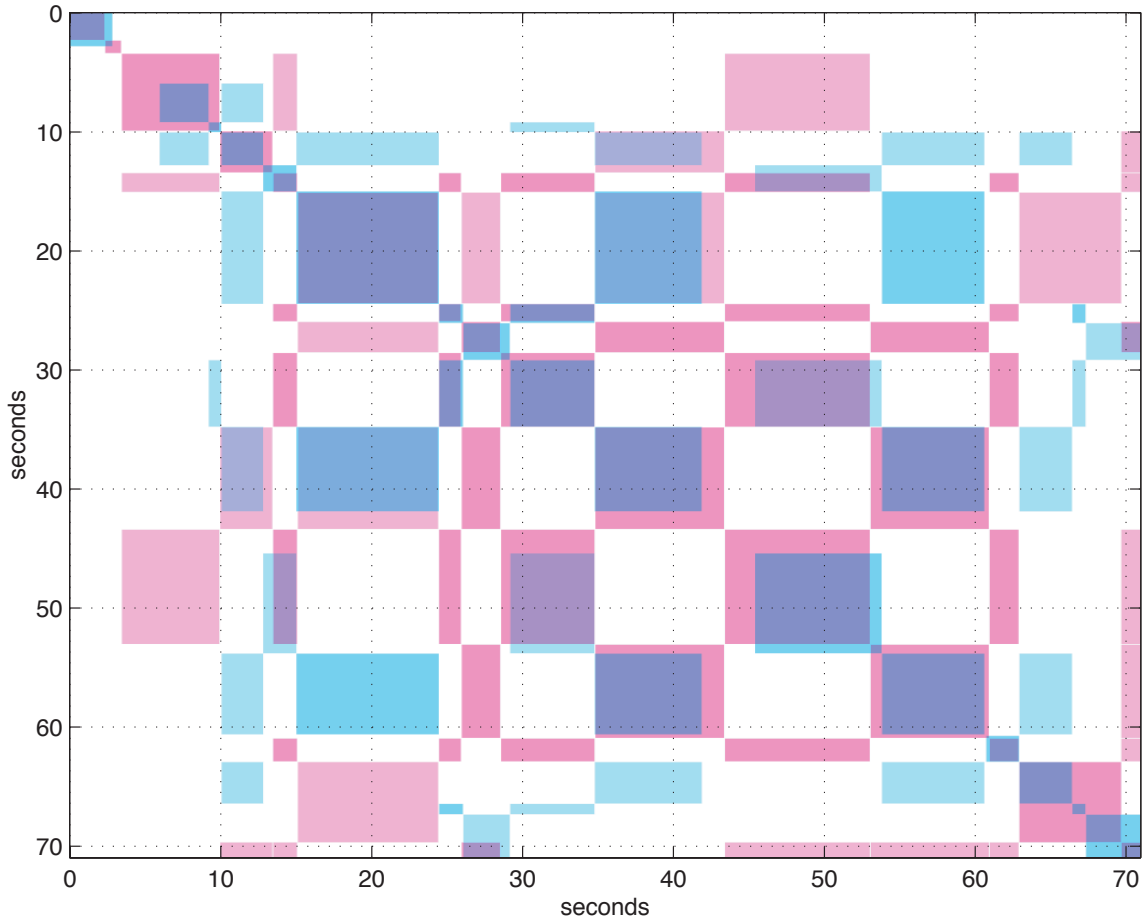


Figure 5.6: Superimposed audio-audio and video-video time-normalized distance matrices for a 71 second segment from NBC's sitcom "Frasier." Magenta denotes close pairs of video segments, cyan denotes close pairs of audio segments, and purple represents intervals that are close in both the audio and video domains.

matching accuracy. Despite this issue, Mr. Letterman's voice is properly matched, thus associating visually distinct shots such as the monologue and desk shots (as evidenced by the many cyan areas in Figure 5.8, yet very few magenta blocks).

A number of simple temporal properties are directly evident from the association matrices. Dialogs, for example, appear as strict checkerboard patterns in $\mathbf{D}_{AA}$ and near-checkerboards in $\mathbf{D}_{VV}$. A single audio segment covering a number of video shots signifies a narrated scene or set of scenes (for instance, nearly everything after the 110 second mark in Figure 5.3 can be considered a narrated segment). Isolated far off-diagonal "same" (low distance) matrix elements indicate two temporally-distant shots have something in common: either similar video if in $\mathbf{D}_{VV}$, or the same speaker if in $\mathbf{D}_{AA}$. "Action" sequences manifest themselves as areas with only the main diagonal having non-1 elements, most likely in $\mathbf{D}_{VV}$ and to a lesser degree in $\mathbf{D}_{AA}$.

5.4 Idiomatic Sequence Detection

Indeed, most local temporal properties of multimedia streams can be interpreted in the framework of association matrices by expressing them as local matrix properties. With such descriptions, the properties can then be detected using correlation-type techniques on regions of one or more of the blocks of $\mathbf{A}$. Using the association matrix to detect local temporal properties of media streams can be seen as a generalization of Yeung's work in identifying dialog and action events in video-only streams [3]. As we shall see, it also allows for easy implementation of cast-detection techniques, such as that of Liu and Wang [84].

While the notion of "important" temporal properties can be very context- and producer-dependent, we aim for generality by detecting sequences that are so common that they can be called "idiomatic" within the set of possible editing sequences. These idiomatic sequences are by nature multimodal, as the media stream is edited as a whole, not video then audio independently. As the alignment of audio and video transitions may not be perfect, however,

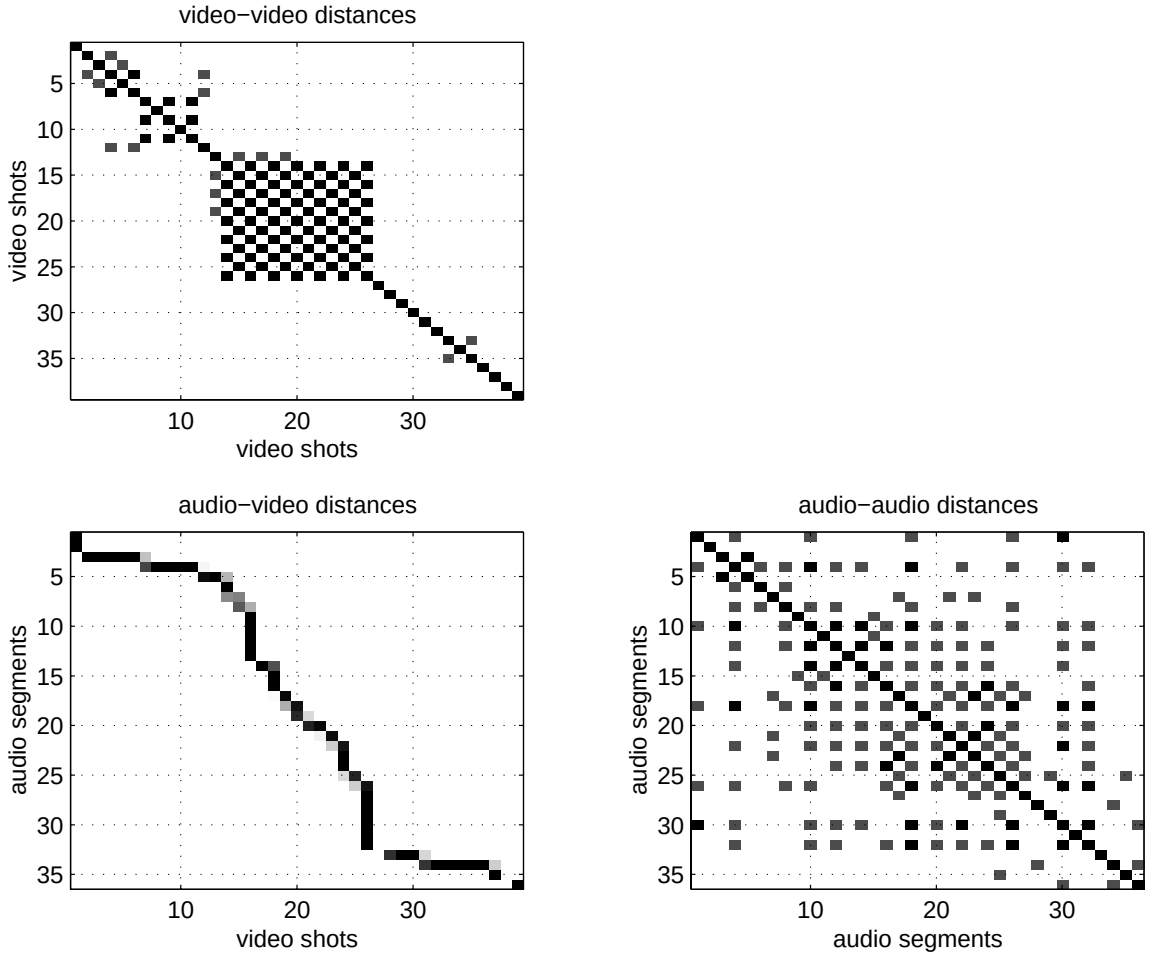


Figure 5.7: Perceptually normalized distance matrices for the first 10 minutes of CBS's "The Late Show with David Letterman." Columns of the upper- and lower-left matrices are time-aligned, as are rows for the lower-left and lower-right matrices. Darker shades represent smaller segment distances.

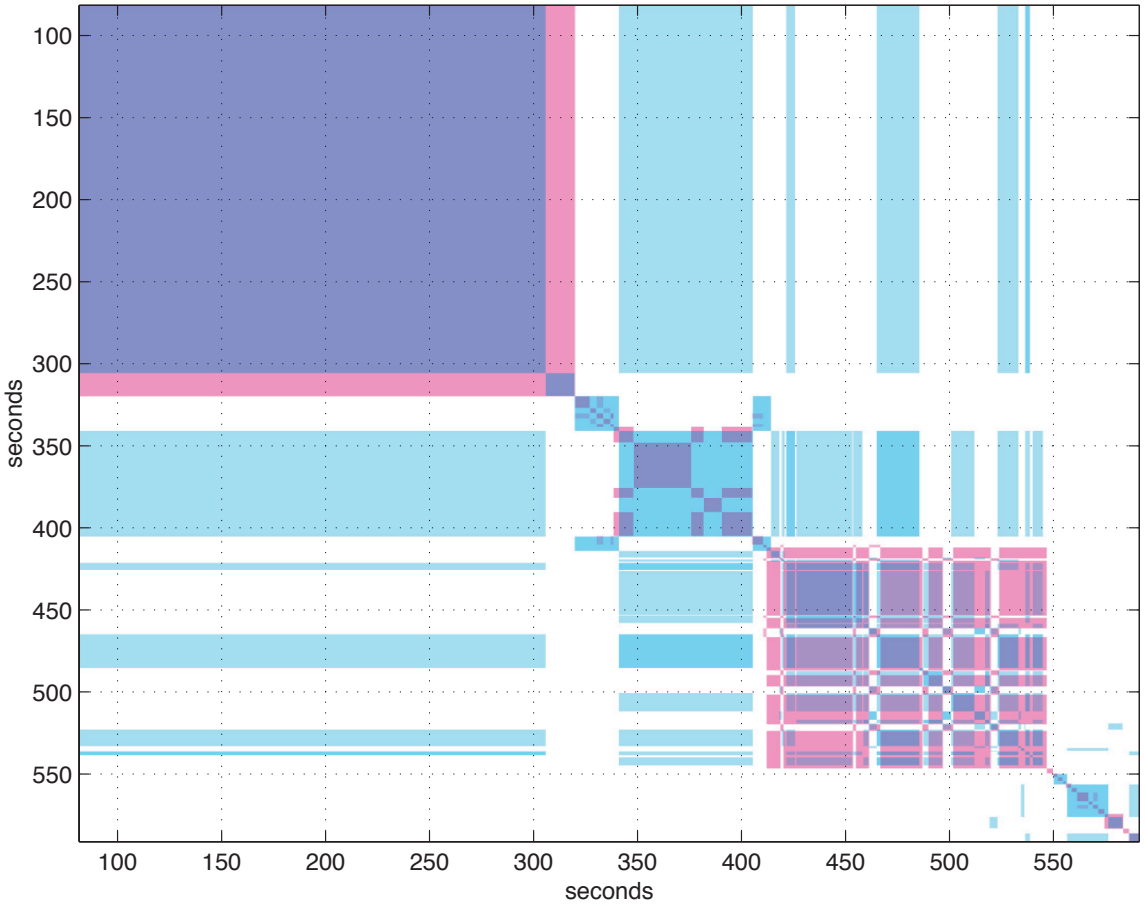


Figure 5.8: Superimposed audio-audio and video-video time-normalized distance matrices for the first 10 minutes of CBS's "The Late Show with David Letterman." Magenta denotes close pairs of video segments, cyan denotes close pairs of audio segments, and purple represents intervals that are close in both the audio and video domains.

our approach is to detect such sequences in each modality independently, then find time intervals where the detections overlap. (Exceptions to this approach will be dealt with in Section 5.4.3.)

In keeping with the simplified A/V association matrix approach of Section 5.3, we use the upper-left (video-video) and lower-right (audio-audio) blocks of a stream's association matrix. The detection algorithm is then applied to each of these two distance matrices in the same manner; from this point on, the single-modality distance matrix under consideration will simply be referred to as $\mathbf{D}$, with $\mathbf{D} = \mathbf{D}_{VV}$ or $\mathbf{D} = \mathbf{D}_{AA}$ as appropriate.

We will first develop the algorithm in general, then describe how to apply it to a number of specific idiomatic sequences.

5.4.1 Detection of Prototype Sequences

Given a single-modality $M_D \times M_D$ distance matrix $\mathbf{D}$, we wish to find instances of certain properties within $\mathbf{D}$ that indicate temporal sequences of interest (dialogs, character introductions, etc.). A block correlation-based approach is adopted, where the correlation between a small prototype matrix and a subset of $\mathbf{D}$ is computed and thresholded[2]. An $M_P \times N_P$ prototype matrix $\mathbf{P}$ contains elements with one of three values:

-0.5: corresponding segment pairs should have low distance ("same," class 1)

$+0.5$: corresponding segment pairs should have high distance ("different," class 3)

0: don't care about the corresponding segment pairs; their distance is irrelevant to the decision of whether the prototype fits the measured distance matrix

The construction of prototype matrices will be discussed in Section 5.4.2.

[2]Although we haven't pursued it here, it would be interesting to study this as a more formal detection problem. A difficult issue, however, is that the "noise" isn't additive, but instead typically involves insertions and deletions of time segments. Another source of noise is the fallibility of the segment distance metrics and resulting incorrect perceptual distance classes.

We define the block correlation function, starting at element (k, l), as follows:

$$\rho_{blk}(\mathbf{D}, \mathbf{P}, k, l) = \frac{\sum_{i=1}^{M_P} \sum_{j=1}^{N_P} \left(d_{i+k-1, j+l-1} - \frac{1}{2} \right) p_{i,j}}{\sum_{i=1}^{M_P} \sum_{j=1}^{N_P} p_{i,j}^2}. \tag{5.5}$$

Recall that $\mathbf{D}$ will have elements ranging from 0 to 1, so 1/2 is subtracted to remove the bias in correlation (yielding, in the ideal case, values equal to those in $\mathbf{P}$). Also note that we don't normalize by the L^2 norm of $\mathbf{D} - \frac{1}{2}$; this is for two reasons. First, the value is nearly constant: only "similar" segments have squared values that are not $\frac{1}{4}$. The second reason is that it allows us to have "don't care" regions that don't detract from the overall correlation (as they implicitly would if we divided by the size-dependent norm of $\mathbf{D}$). We are essentially calculating the fraction of "do care" segments that match.

There are a small number of cases, as we shall see in Section 5.4.2, where a raw count of the non-matching segment pairs is more appropriate than the fractional correlation calculated in equation (5.5). To differentiate it from the correlation calculation's threshold, will denote this absolute threshold as "X segments" when it is necessary.

In order to handle a wider variety of temporal prototypes, we divide the prototype sequences to detect into two classes:

Self-similar prototypes: if temporal interval T has the desired property, then *all* sub-intervals of T also have the desired property (e.g., subsets of dialog sequences are themselves dialogs).

Non-self-similar prototypes: if temporal interval T has the desired property, there exists some sub-interval of T that does *not* have the property (e.g., returning to a news anchor at the beginning and end of a story).

Regardless of their self-similarity, prototypes may have a global minimum length; a 2-segment dialog, for instance, is meaningless. We also characterize the prototypes as either:

Local: whether an interval T fits the prototype depends only on segments within T.

Global: testing whether an interval T fits the prototype requires knowledge of some segments and distances outside T.

While the general techniques are similar and relatively straightforward, the detection of each class of prototype sequences differs in the details.

Local, Self-similar Prototypes

The simplest case, local and self-similar prototype sequences are characterized by a single square prototype matrix for a given sequence length; we denote the $l \times l$ prototype matrix by $\mathbf{P}_l$. Self-similarity allows us to detect the sequences of interest by starting with a short detected sequence and growing it until it reaches segments that do not fit the prototype. The following algorithm essentially makes its way through the segments, attempting to find the longest sequence whose correlation with the prototype is above a threshold T_P

(discarding sequences shorter than a minimum length, T_l):

```
i = 1
while( i ≤ M_D )
    l = T_l − 1
    while ( ρ_blk(D, P_{l+1}, i, i) ≥ T_P )          (*)
        l = l + 1
        if ( l + i − 1 > M_D )
            break
    end
    if ( l ≥ T_l )
        declare idiomatic sequence starting at segment i, length l
        if ( events cannot overlap )
            i = i + l − 1
    end
    i = i + 1
end
```

The test in line 10 asks whether it is reasonable for detected events to overlap in time; for most types of idiomatic sequences, it is not, so the next l segments are skipped once a detection is made. Dialog is an example: were it not for this test, a 10 segment dialog would be detected $11 - T_l$ times (once for each segment, all with the same ending segment). As we will see in Section 5.4.2 though, there are cases where two separate instances of a particular idiomatic sequence can reasonably overlap in time.

Global, Self-similar Prototypes

Global prototype sequences can be seen as a special case of local sequences, where the prototype matrix $\mathbf{P}$ is of a fixed size, $M_D \times M_D$, the same as $\mathbf{D}$. The prototype matrix is

now dependent on the segment offset within $\mathbf{D}$ currently being tested, as well as the length of the candidate sequence, so we denote it by $\mathbf{P}_{l,i}$, where i is the segment number beginning the candidate sequence. Once again, self-similarity allows us to grow candidate sequences by increasing l until one is found that does not sufficiently match. The only change to the algorithm presented on page 105 is in the line labelled (*):

`while (` $\rho_{blk}(\mathbf{D}, \mathbf{P}_{l+1,i}, 1, 1) \geq T_P$ `)`

(note the fixed offset, $(1, 1)$, in the correlation calculation).

Local, Non-self-similar Prototypes

A lack of self-similarity prevents prototype matrices in this class from being "grown" until the largest match is found, because some of the smaller prototype sequences will fail the correlation test even if a larger one would pass. We can handle a large number of such cases by imposing a two-step test: one prototype matrix is used to grow the potential sequence as large as possible, then another prototype matrix is used as a confirming test on the final size. Specifically, we replace the "declare idiomatic sequence..." line of the algorithm on page 105 with the following:

`if (` $\rho_{blk}(\mathbf{D}, \mathbf{P}_l^C, i, i) \geq T_C$ `)`

declare idiomatic sequence starting at segment i, length l

`end`

where $\mathbf{P}_l^C$ is the confirming prototype matrix and T_C is the correlation threshold on this matrix. Again, in some cases, it is more appropriate for T_C to be a raw number of non-matching segments.

While this two-step method greatly expands the number of idiomatic sequences we can detect, it cannot handle every non-self-similar possibility. An example is the "independent event" sequence of Section 5.4.2; it is non-self-similar, but has other properties which make

it easy to detect with only a slight modification to the framework for global, self-similar prototypes. In addition, for idiomatic sequences that can only be one segment in length (such as "character introduction"), the method of this section can be used to effectively provide two independent correlation tests, with two thresholds, on the measured distance matrix.

Global, Non-self-similar Prototypes

To handle idiomatic sequences having global, non-self-similar prototypes, one only need combine the modifications described in the two previous sections. The resulting algorithm uses two prototype matrices, each $M_D \times M_D$ in size: $\mathbf{P}_{l,i}$ and $\mathbf{P}^C_{l,i}$.

5.4.2 Generation of Idiomatic Sequences' Prototype Matrices

The format of the prototype matrix entirely determines what types of idiomatic sequences it will detect and how it will perform. As the prototype matrices are not constant—to allow for varying-length sequences, and for global prototypes, different starting point offsets—we will describe their general form. In most cases, the prototype form for a particular idiomatic sequence can be easily derived by thinking about which segments must be the same speaker/image, and which must not, in order to construct an archetypical sequence of the desired form.

The most straightforward idiomatic sequence to detect is dialog. Dialog sequences consist of two unrelated speakers/images, with the audio or camerawork alternating between them. This results in every other segment pair being "same," creating a checkerboard pattern in the distance matrix. (Real dialogs may have intermediate segments with both actors in the frame, something we currently account for only in the selection of the threshold T_P. More general time alignment techniques may help.)

The idiomatic sequences we are currently able to detect, within both the audio and video domains, are summarized below. This list is not an exhaustive set of possible detectable

sequences; it only represents common idioms whose forms are simple to determine. Below each entry are the details of its detection, including empirically-determined correlation thresholds. The prototype matrices are examples to show the general form, as the actual matrices will depend on the parameters l and i; in each case, ■ represents -0.5 ("same"), □ denotes $+0.5$ ("different"), and × indicates 0 ("don't care"). Elements along the main diagonal are normally "don't care" because every segment is identical to itself, giving a normalized distance of 0. For global prototype matrices, the (i, i)-th element is denoted ⊗ to show the offset of the center portion within the ellipses.

- **Dialog:** Segments alternate between two primary characters (speakers in audio, images in video). In detecting conversations, audio dialogs are more meaningful than video dialog sequences, although the latter can offer important information. In the "Charlie Rose" clip (Figure 5.1), for example, one guest speaks for a long time (a single audio segment), while the camera switches back and forth between the two participants. Video shots during an audio dialog occasionally do not follow the typical back-and-forth format (being interspersed with wide-angle shots of both participants, for example), making detection of the audio event all the more important.

Global/Local:	local	**Min Length (T_l):**	4
Self-Similar:	yes	**Testing T_P:**	0.75
Can Overlap:	no		

$$\mathbf{P}_l : (l = 6) \quad \begin{bmatrix} \times & \square & \blacksquare & \square & \blacksquare & \square \\ \square & \times & \square & \blacksquare & \square & \blacksquare \\ \blacksquare & \square & \times & \square & \blacksquare & \square \\ \square & \blacksquare & \square & \times & \square & \blacksquare \\ \blacksquare & \square & \blacksquare & \square & \times & \square \\ \square & \blacksquare & \square & \blacksquare & \square & \times \end{bmatrix}$$

- **Action:** A sequence of segments that are wholly independent of one another, such as a sequence of video shots following a moving character. Such sequences generally indicate a progression of events, rather than an interactive or situational scene.

Global/Local: local **Min Length (T_l):** 5
Self-Similar: yes **Testing T_P:** 0.999
Can Overlap: no

$$\mathbf{P}_l : (l = 6) \quad \begin{bmatrix} \times & \square & \square & \square & \square & \square \\ \square & \times & \square & \square & \square & \square \\ \square & \square & \times & \square & \square & \square \\ \square & \square & \square & \times & \square & \square \\ \square & \square & \square & \square & \times & \square \\ \square & \square & \square & \square & \square & \times \end{bmatrix}$$

- **Return to Anchor:** The first and last segments of the sequence are the same speaker/image, but the intermediate segments are dissimilar from both the first and last. This sequence is typical of television news shows, where an anchor desk shot appears between successive stories.

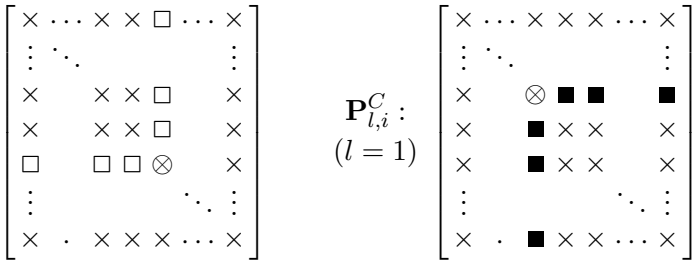
Global/Local: local **Min Length (T_l):** 6
Self-Similar: no **Testing T_P:** 0.99
Can Overlap: yes **Testing T_C:** 0.995

$$\mathbf{P}_l : (l = 6) \quad \begin{bmatrix} \times & \square & \square & \square & \square & \square \\ \square & \times & \times & \times & \times & \times \\ \square & \times & \times & \times & \times & \times \\ \square & \times & \times & \times & \times & \times \\ \square & \times & \times & \times & \times & \times \\ \square & \times & \times & \times & \times & \times \end{bmatrix} \qquad \mathbf{P}_l^C : (l = 6) \quad \begin{bmatrix} \times & \square & \square & \square & \square & \blacksquare \\ \square & \times & \times & \times & \times & \square \\ \square & \times & \times & \times & \times & \square \\ \square & \times & \times & \times & \times & \square \\ \square & \times & \times & \times & \times & \square \\ \blacksquare & \square & \square & \square & \square & \times \end{bmatrix}$$

- **Character Introduction:** Although not strictly limited to human characters, this idiomatic "sequence" of one segment occurs the first time a new video shot or speaker is introduced. To distinguish introductions other events, we require that we see the same speaker/image at least one more time during the media stream. (As such, detection of character introductions is not a causal operation unless an *a priori*-unknown delay is allowed in processing.) Detection of character introductions benefits from using an absolute threshold (T_C) during the confirmation step, as we don't care so much that a particular fraction of future segments contain the given speaker/image, only that a raw absolute number of segments do.

Global/Local: global **Min Length (T_l):** 1 (only possible size)
Self-Similar: no **Testing T_P:** 0.99
Can Overlap: no **Testing T_C:** 1 segment

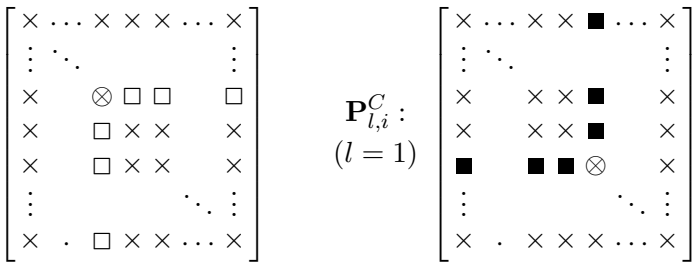
$\mathbf{P}_{l,i}$: $(l=1)$ $\mathbf{P}^C_{l,i}$: $(l=1)$



- **Character Departure:** The opposite of a character introduction, a departure occurs when the current segment has been seen at least once before, but is never seen again. Detection of character departures can not be done causally, as the entire remaining portion of the stream must first be known. As in character introductions, an absolute threshold is used during the confirmation step because it is only the raw number of matching segments that matters.

Global/Local: global **Min Length (T_l):** 1 (only possible size)
Self-Similar: no **Testing T_P:** 0.99
Can Overlap: no **Testing T_C:** 1 segment

$\mathbf{P}_{l,i}$: $(l=1)$ $\mathbf{P}^C_{l,i}$: $(l=1)$

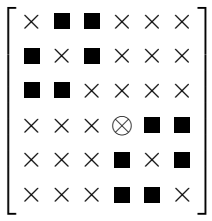


- **Topic Change:** As the name implies, these are sequences containing exactly two unrelated subsequences (no "same" segment pairs between them), where each subsequence does contain some intra-subsequence associations. Topic change sequences are characterized by a 2×2 block-identity form for their distance matrices, and are thus not strictly self-similar. Subsets of topic change sequences that center about the same

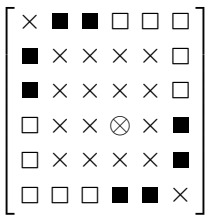
point, however, are themselves topic changes, so we can detect this idiomatic sequence by modifying the local self-similar sequence detector slightly: instead of growing test sequences from the upper-left corner, we instead grow them from the center point. P_l is thus $2l \times 2l$ in size; $\otimes$ in the following three matrices represents the "center" element from which the prototype matrices are grown. (The equal lengths of the two subsequences in the prototype matrix does not impose a similar requirement on the two topics, as matching candidate sequences are grown from the change point, thus some subset will have equal length subsequences.) In order to prevent the $\mathbf{P}_l$ prototype sequence from growing too far, thus unnecessarily failing the $\mathbf{P}_l^C$ test, we use only the outer elements of the prototype when growing (except for the initialization step, where $l = T_l$, which must include all interior elements).

Global/Local:	local	**Min Length (T_l):**	5
Self-Similar:	no	**Testing T_P:**	-0.2
Can Overlap:	yes	**Testing T_C:**	0.9

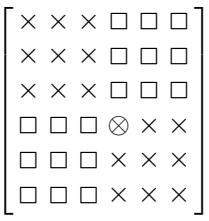
$\mathbf{P}_l$: $(l = T_l)$



$\mathbf{P}_l$: $(l > T_l)$



$\mathbf{P}_l^C$: $(l = 3)$



- **Independent Event:** An independent event sequence is one that is completely unrelated to any past or future segments. Examples include non-repeated commercials, short segments dividing larger portions of a show, and flashbacks and similar interludes[3]. Segments within an independent event may, however, be associated with one another. Despite having a global, non-self-similar prototype, independent event sequences cannot be detected using the two-step method presented above. Instead, they can be found by starting, at each segment offset i, with the largest l that will fit

[3]For another approach to commercial break detection, see the works of Lienhart and McGee, which use statistics such as cut rate, mean volume, and number of black frames [98, 99].

($l = M_D - i + 1$), then shrinking the prototype matrices until an above-threshold correlation with the distance matrix is found. While independent events can be nested in complex ways, we choose to prevent overlapping detections in order to avoid a number of degenerate cases. (This is an issue that deserves further study, as the non-overlapping assumption prevents detection of a number of interesting events.)

Global/Local: global **Min Length (T_l):** 1
Self-Similar: no **Testing T_P:** 0.99
Can Overlap: no

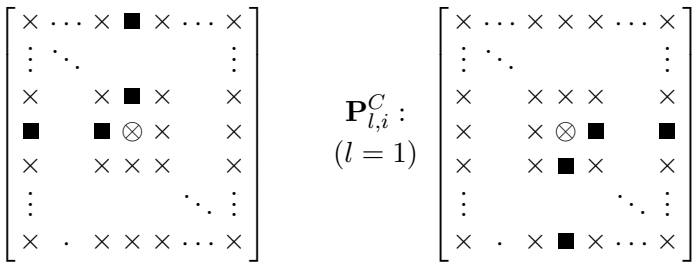
$\mathbf{P}_{l,i}$: ($l = 2$)

$$\begin{bmatrix} \times & \cdots & \times & \square & \square & \cdots & \times \\ \vdots & \ddots & & & & & \vdots \\ \times & & \times & \square & \square & & \times \\ \square & & \square & \otimes & \times & & \square \\ \square & & \square & \times & \times & & \square \\ \vdots & & & & & \ddots & \vdots \\ \times & \cdot & \times & \square & \square & \cdots & \times \end{bmatrix}$$

- **Path Split:** The precise meaning of "path" will be explored in Section 6.4, but a split essentially means that two or more (distinct) future segments are directly or transitively related to the current one, and one or more past segments are similarly associated. Ideally, at a high level, this corresponds to a segment where a single (transitive) plot path splits into two parallel story lines. Detection of path split segments requires the use of memory-based distance matrices, described in Section 6.3, instead of the standard distance matrices given above. (The memory-based distance matrices make use of transitive links, but include only the temporally-closest links from each particular segment.)

Global/Local: global **Min Length (T_l):** 1 (only possible size)
Self-Similar: no **Testing T_P:** 1 segment
Can Overlap: no **Testing T_C:** 2 segments

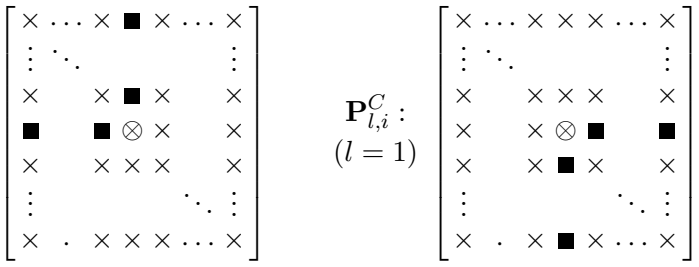
$\mathbf{P}_{l,i}$: $(l = 1)$ $\mathbf{P}^C_{l,i}$: $(l = 1)$



- **Path Merge:** Merges are the temporally-reversed analog to splits, where multiple distinct segments from the past are transitively associated with the current one, which is in turn associated with at least one future segment. Path merges correspond to the semantic ideal of two parallel story lines (linked by transitive close associations) merging into a single path. It is possible for a single segment to be both a merge and a split if it ties together multiple parallel transitive links that continue both into the past and the future. Like path splits, detection requires the memory-based distance matrices of Section 6.3.

Global/Local: global **Min Length (T_l):** 1 (only possible size)
Self-Similar: no **Testing T_P:** 2 segments
Can Overlap: no **Testing T_C:** 1 segment

$\mathbf{P}_{l,i}$: $(l = 1)$ $\mathbf{P}^C_{l,i}$: $(l = 1)$



In light of this listing, one can visually pick out certain idiomatic sequences within the sample clips of Figures 5.1–5.8; low-distance segment pairs are shown in black in both cases. The video distance matrix for the CBS news clip (Figure 5.4), for instance, contains

a number of "return to anchor" segments. Video segment 11 and audio segment 5 (which coincide in time) of the Charlie Rose clip fit the prototype of "independent event" sequences; that interval contains a short musical interlude between two interviews. Finally, in the David Letterman clip (Figure 5.8), the three dominant scenes, respectively, are a monologue, a narration (discussed in the next section), and a dialog where one participant has much less on-air time than the other. Note that particular segments may belong to multiple idiomatic sequences.

Depending on the locality and self-similarity of the prototype under consideration, one of the four algorithms presented in Section 5.4.1 is used to locate instances of the idiomatic sequence. Even for very long streams, local prototypes are simple to handle as they require only knowledge of a small block matrix around the current segment. Global prototypes require at least some matrix elements to be preserved, and can result in large matrix tests if the source video is more than a few hundred shots long (roughly one half-hour of television with commercials, or a feature-length film). It is worth recalling that the distance matrices are quantized into 3 values, thus requiring only 2 bits per element in the ideal case, and the correlation calculation (5.5) can be easily vectorized. As a (non-vectorized) timing example, with a video of 552 shots and 290 audio segments, finding global-prototype introductory segments takes 94 seconds (16 times faster than real time), while finding local-prototype dialog sequences takes 15 seconds, both on a 350 MHz Sun workstation. Specific experimental results are given in Section 5.4.4.

5.4.3 Cross-modality Idiomatic Sequences

Each of the idiomatic sequences described above can be detected in only one modality, audio or video, at a time. In order to form a truly multimodal description of the stream, one should correlate the detections based on time intervals (not shot numbers) where they overlap: intervals with a particular idiomatic sequence in both audio and video are flagged.

Another class of idiomatic sequences involves the interaction of audio and video. The

$\mathbf{D}_{AV}$ (lower-left) block of the association matrix is the only useful statistic in this case. Among the most straightforward examples of its use are:

- **Narration:** Narrated intervals consist of a single audio segment overlapping with a number of video segments. At present, we ignore intra-modality associations among the video segments, concentrating only on $\mathbf{D}_{AV}$; future work should certainly take them into account. Narrated intervals can be detected by simply counting the number of 0 (or nearly 0) distance elements in each row of $\mathbf{D}_{AV}$. Rows with five or more 0 elements are audio segments of narration.

- **Group Audio:** The transpose of narration, this sequence consists of a number of speaker segments during the same video shot. Generally such a sequence occurs during a wide-angle video shot of a number of participants in a discussion. Detection is done in the same manner as for narrated intervals, except that $\mathbf{D}_{AV}^T$ is used as the base distance matrix (and the results are video shots with group audio).

- **A/V Alignment:** If a sub-matrix of $\mathbf{D}_{AV}$ is the identity (or more precisely, $1 - I$), the corresponding audio and video segments align precisely in time. Straightforward heuristic algorithms can be used to test for this condition. The semantic/idiomatic meaning of such an alignment is somewhat hazy, but its presence is occasionally noteworthy as typical television and film media are not aligned in this way.

More complex $\mathbf{D}_{AV}$ sequences could also be be detected using correlation-type algorithms similar to those in Section 5.4.1. Although it has not been investigated, it would certainly possible to construct idiomatic sequences that require certain forms within audio and video distance matrices, combined with a certain form of the $\mathbf{D}_{AV}$ cross-distance. Such idioms would likely be rather complex and application-specific.

5.4.4 Detector Experimental Results

In addition to the four television clips named above, we also used short clips from ABC and NBC news in testing the idiomatic sequence detectors. The total test set consisted of 46060 frames, or 25.6 minutes, of digitized video and audio. Detected sequences were studied in light of the "ground truth," meaning what an actual viewer would perceive (given the prescribed minimum sequence lengths and overlap prohibitions), as well as in terms of how well the detectors could possibly do given a likely imperfect association matrix. Any discrepancy in the results is due to inadequacies in the video and (especially) audio segment distance metrics discussed in Chapter 4. Table 5.1 summarizes the results.

The ground truth sequences listed in the table were extracted by manually gauging similarity between the same shots and audio segments used by the automated techniques. As described in Section 5.4.2, detected idiomatic sequences are restricted by minimum length and overlap constraints; we obeyed these same constraints when generating the ground truth sequence lists. Ground truth sequences are more ambiguously-defined than those in terms of association matrices, as the continuum of real-world distances between shots makes it difficult to pinpoint some events. Sequences like topic-change can sometimes be ill-defined for real video, even though it is unambiguous in quantized association matrices. Occasionally, the semantic meaning of the idioms gets distorted as well: a small number of dialog and return-to-anchor sequences had no human characters for example, and the non-overlap restriction prevented a many independent-event sequences from being properly detectable.

The discrepancy between the two sets of results in the table makes clear the fact that the distance metrics—particularly audio—are the limiting factor. A single incorrect distance can cause a number of misses and false alarms, as the events to be detected are not temporally localized. Remember that, because the segmentation is done manually in these tests, the $\mathbf{D}_{AV}$ distance matrix is by definition correct; narration, group audio, and alignment detection results are therefore the same with respect to the association matrix as with the

Idiomatic Sequence		against ground truth		against assoc. matrix	
		$P(D)$	#FA	$P(D)$	#FA
Dialog	(video)	6/6 (100%)	0	6/6 (100%)	1
	(audio)	4/7 (57%)	4	9/9 (100%)	0
Action	(video)	6/6 (100%)	0	7/7 (100%)	0
	(audio)	3/4 (75%)	3	6/7 (86%)	0
Return to Anchor	(video)	5/11 (45%)	0	5/6 (83%)	0
	(audio)	0/2 (0%)	1	1/1 (100%)	0
Character Introduction	(video)	15/23 (65%)	4	19/19 (100%)	0
	(audio)	11/19 (58%)	14	26/26 (100%)	0
Character Departure	(video)	14/23 (61%)	6	20/20 (100%)	0
	(audio)	11/19 (58%)	11	25/25 (100%)	0
Topic Change	(video)	0/1 (0%)	0	0/1 (0%)	0
	(audio)	0/2 (0%)	0	0/0 (-%)	0
Independent Event	(video)	2/3 (67%)	0	5/5 (100%)	0
	(audio)	8/13 (62%)	10	18/18 (100%)	0
Path Split	(video)	1/1 (100%)	1	2/2 (100%)	0
	(audio)	0/2 (0%)	2	2/2 (100%)	0
Path Merge	(video)	0/1 (0%)	1	1/1 (100%)	0
	(audio)	0/2 (0%)	1	1/1 (100%)	0
Narration	(A/V)	6/6 (100%)	0	6/6 (100%)	0
Group Audio	(A/V)	3/3 (100%)	0	3/3 (100%)	0
A/V Alignment	(A/V)	2/2 (100%)	0	2/2 (100%)	0
Totals		97/157 (62.2%)	58	164/167 (98.2%)	1

Table 5.1: Detection and false alarm rates for the idiomatic sequence detector over 25.6 minutes of digitized television. As the distance metrics in the association matrix are the dominant source of error, results are given with respect to the real (human-perceived) events as well as against the information offered by the distance matrices.

ground truth events.

It would be worthwhile to expand the scope of statistics available to the idiomatic sequence detection algorithms. Likely useful information includes segment durations (in real time), the interaction of other detected sequences in dependency chains (so, e.g., the first two segments of a dialog aren't the last two of an action sequence), and intra-shot information such as face detection.

Chapter 6

Structure via Multimodal Processing

A key concept in the analysis of multimedia streams is the determination of how the plot manifests itself in connections between audio and video shots and scenes. Deriving the long-term temporal structure, beyond the notion of simple scene clustering, therefore yields significant information about the stream in question. As we saw in the last chapter, fairly general heuristics can be used to pinpoint short sequences of interest, such as segments that are narrated by a third party, or instances where programs return to a common shot such as that of a news anchor. Once again, domain or genre-specific processing can yield further information about what the temporal structure means in terms of "content." Even without domain-specific intelligence, the temporal structure is useful in composing quick summaries of the stream's content; this idea will be further developed in Chapter 7.

One method of visualizing and interpreting segment association information (and thus, temporal structure) is via the matrices described in Section 5.3. The association matrix summarizes the distance information, but only shows direct, not transitive, links between segments. Temporal transitivity allows us to associate distant segments as well as construct directed graphs and trees of the content. In particular, transitivity across modalities is the key to associating video shots that are not visually similar and audio segments that are not of the same speaker. Shortest-path algorithms and related ideas can be used on

these graphs to answer specific queries, attempt to reconstruct plot lines, and provide an automated graphical summary of the media stream.

The use of transitive links also compensates for some imperfections in distance metrics, as missed direct associations may be "routed around" by multiple links. For example, if segments s_1 and s_2 should have a low perceptual distance, but instead are detected as "different," it is possible that some segment s_k exists such that s_1 is close to s_k and s_k is close to s_2. The converse, unfortunately, is also possible: if s_1 and s_2 should have a high perceptual distance, the presence of s_k will falsely lower it. And not all types of issues can be corrected by transitivity; a major impediment to reliably determining lines of plot progression is the presence of incorrectly characterized segment pairs (close associations that are missed, or truly different segments that are falsely detected as "same" or "similar").

6.1 Prior Temporal Structure Work

In the mid 1990's, researchers began looking for ways to summarize video-only streams using temporal sequences. Yeung developed a "scene transition graph" to aid in visualization and analysis; the graph allowed for easy identification of temporal events, clusters, and progression from one scene to another [3, 100]. Time-adaptive clustering of video shots, based not only on key frames but statistics such as motion and histogram changes as well, was further explored by Rui [101]. Lienhart, *et al.*, used audio features, face detection, and temporal rules to cluster shots into dialog scenes, scenes with similar settings, and audio-based scene-like intervals [102].

Domain-specific knowledge was exploited by Furht, *et al.*, to help in temporally parsing news programs (classifying intervals into, for example, anchor-people, stories, weather reports, and commercials)[103, 104]. Story segmentation and classification of news streams, using video, audio, and transcript information, has also been studied [105, 106].

Wolf used a hidden Markov model (HMM) to classify sequences based on the size of

the dominant face in each shot's key frame [107]. This HMM design was later applied to the structural analysis of documentaries and home video [108, 109]. The authors found that using more than a handful of sequence types (HMM states) tends to result in over-segmentation into nearly-equivalent states.

6.2 Association Graphs

While the association matrices are useful in showing immediate connections between segments, they do not directly show segments that are close due to transitivity. Cross-modality transitivity can be illustrated by selecting a single element from $\mathbf{D}_{VV}$, using the local minimum in the same column of $\mathbf{D}_{AV}$ to find an associated audio segment, finding a small-distance element in that row of $\mathbf{D}_{AA}$ to use as an association in audio, using $\mathbf{D}_{AV}$ to "translate" back to a video shot, and so on.

An interpretation in terms of weighted directed graphs shows the transitivity more clearly. The vertices in these graphs are the set of audio and video segments, S in equation (5.1), while the edge weights are the corresponding perceptually normalized distances. Given a judicious choice of d_{norm} in Section 4.4, the "cost" of a path through the graph can be computed simply by summing the edge weights. Using the sum is intuitively justified, as if segment s_1 is "similar" to s_2, and s_2 is "similar" to s_3, s_1 and s_3 are at least remotely similar. Extending the chain by one more link can yield segments that are essentially "different" (and given d_{norm} in (4.10), would have a total distance of 0.9). Note that this summed cost is only an upper bound on the true perceptual distance between the end segments along the path, and it may well be greater than 1 (our maximum distance). An alternative cost formulation, which we will use in Sections 6.2.2 and 6.3, is to select the maximum edge weight as the total cost of a transitive path.

6.2.1 Shortest Paths

The formulation given above lends itself perfectly to the use of shortest-path algorithms to find semantic "paths" through a media stream. While one cannot claim such paths always follow the real plot evolution of a stream, they are nonetheless useful in attempting to answer the question, "What sequence of events led from A to B?" Without multiple modalities and a cross-modality distance metric, it would be impossible to answer that question if A and B were not intrinsically similar to begin with. Given the association matrix as a starting point, methods such as Dijkstra's algorithm allow us to answer such chain-of-event queries relatively quickly [110]. Commercials and other unrelated interrupting sequences will automatically be ignored, as they will not normally have any close associations with story segments.

In determining transitive associations of segments, allowable paths through the directed graph can be restricted in a number of ways. Restrictions are implemented by setting the weight of disallowed edges to ∞ (equivalently, replacing certain elements of $\mathbf{A}$ with ∞). Possible restrictions include the following, as well as combinations thereof:

- A **forward-time only** requirement prohibits travel via any edge leading from a segment that occurs (temporally) after the one it leads to. "After" must be defined carefully, as audio and video segments can overlap; for our purposes, the segments' start times are used.

- **Reverse-time only** restrictions only allow travel to segments that have already concluded; this allows for strictly causal processing. (If some delay is tolerable, this restriction can be relaxed slightly to allow concurrent segments by waiting until all have finished before deciding. Deadlock issues may arise, though, if this variable-delay scheme is not implemented carefully.)

- A **contiguous segment** condition only allows travel via edges corresponding to overlapping segments. This condition forces the creation of many disconnected subgraphs,

as any complete break in the audio and video action will not be traversable (and it will be difficult to determine from the graph if an associated segment is returned to at a later time in the stream).

- **Threshold** restrictions can be set, above which any edge weight is considered to be ∞. Employing a threshold of 0.99 prohibits "different" segments from ever being associated, a generally good idea.

After selecting two segments of interest, possibly from different modalities, it is straightforward to run Dijkstra's algorithm on the (possibly restricted) association graph. If run repeatedly, independent sequences (such as commercials) will be split out into separate connected graphs, while closely-associated segments will cluster in well-connected subgraphs.

6.2.2 Transitive Path Existence

While in theory one could run Dijkstra's algorithm over every possible segment pair to determine all transitive connections, such a computationally-intensive method is overkill. In addition, the number of resulting (transitive) associations would be prohibitive for any further processing, such as visualization or detection of idiomatic sequences. (Adjacent segments are commonly related through transitive associations, for instance; this would foil a dialog detector.) More often what is desired is to know whether any path exists connecting two apparently unrelated segments. The raw distance in this case is not a concern, provided each segment pair in the path is associated to some desired degree. The maximum edge weight along a path, therefore, is what we will use as the "cost" of the path.

This formulation bears a number of similarities to the flow graph problem, but our situation is even simpler due to fact that we ignore any edges above a certain cost. We can simply threshold the raw association matrix $\mathbf{A}$ as above, then utilize the resulting graph without regard to edge weights. The coarse 3-step quantization of the perceptually-normalized distance matrices $\mathbf{D}_{VV}$ and $\mathbf{D}_{AA}$ makes the selection of a proper threshold

straightforward; the only real variable is how much cross-modality overlap to require (as $\mathbf{D}_{AV}$ is not quantized). 0.8 is a reasonable distance threshold to use, as it allows 20% or more time overlap between video and audio segments to count as associating them.

In order to answer the question of whether a suitable path exists between segments s_k and s_l, one need only use the thresholded (thereby unweighted) graph in a breadth-first search algorithm, starting at s_k, and terminating when s_l is included in the tree or when all nodes have been tested. The time required to answer the query is proportional to the number of segments plus number of association matrix elements that meet the threshold. Due to the nature of the breadth-first search algorithm, it is advantageous to compute the set of all nodes transitively connected to a given node at the same time.

6.3 Memory-Based Graphs and Matrices

Given that we can now determine the existence of any transitive path as needed, we must devise a way to find all "interesting" paths without finding so many that the resulting graph is too cluttered for visualization or other uses. The selection method we will employ is based on a memory-driven interpretation of temporal associations. Basically, given a particular audio or video segment, we claim that the most useful associations to know of are the temporally closest ones. If a low-distance path can be found, that is used; if not, we search for the temporally-closest node with a path of a slightly higher total distance. This idea is rooted in the way viewers are likely to perceive a segment, via a mental search for something they've already seen or heard that is the same as (or similar to) the current image or sound. Most likely, the mental search will settle on the most recent instance within the viewer's memory. It is also not unlikely that a segment introducing the type similar to the current one, or some other key, similar segment in the past, will be the first one remembered; the former is useful for clustering but not transitive linking, and the latter is rather difficult to define in an algorithmic sense. We will therefore consider only the temporally-closest links

from this point on.

Let $n_1, n_2, \ldots, n_N$ be the set of nodes (segments) for one modality, either audio or video in our case. The causal memory-based graph for this modality is constructed as follows:

```
for ( i = 1 to N )
    foreach d in [0 0.3]
        // find the set of transitively-connected nodes via edges of distance ≤ d
        tNodes = BreadthFirstSearch( n_i, d )
        tNodes = subset of tNodes with indices < i
        if ( NotEmpty( tNodes ) )
            j = max( tNodes )
            add edge from node j to node i
            break // out of foreach() loop
        end
    end
end
```

Where non-causal processing is possible, this algorithm is repeated on a time-reversed version of the stream[1]. This is done by reversing the direction of the outer loop and the inequality in line 5, substituting `min()` for `max()` in line 7, and by reversing the orientation of the edge addition in line 8 (so they still point forward in time, even though the search is backward). If an *a priori*-unknown delay is permissible in computing some nodes' forward associations, the anti-causal "memory" based graph can be computed by keeping a running list of nodes that have not yet been forward-associated. The two-pass method allows multiple (forward-time) links to be found leading to the same node, whereas the strictly causal algorithm is restricted to generating sets of temporally-expanding trees. Stated differently, the anti-causal pass allows paths of associations to merge in time, as well as split.

[1] In some cases, a search vector of [0 0.25] in the second line generates better results due to the large number of false "similar" audio segment pairs with distance 0.3.

By treating the "**add edge from node** j **to node** i" line above as simply $\mathbf{D}'(j,i) = 1$, predecessor matrices can be constructed that play the same role as the distance matrices used to derive them. Such two-pass augmented distance matrices are used in the path-split and path-merge idiomatic sequence detectors described in Section 5.4.2.

6.4 Inferring Plot Threads

Ideally, the diverging and converging paths of the memory-based graph will follow the semantic chains of plot through the media stream. While higher-level human intelligence can make further connections among segments by inferring off-camera actions, television and film directors often use common visual and aural cues in segments to reinforce these connections (which with luck would cause measured distances to be lower as well). We model the semantic progression through a stream as shown in Figure 6.1, and the automatically-generated memory-based graph follows this pattern well. The figure shows a single modality from a ficticious stream, where each node is a segment and each edge is found by the two-pass algorithm above (thus likely using cross-modality information, even though the other modality is not shown). Parallel horizontal paths in the model are independent yet simultaneous chains of association through the stream; we will call such parallel paths "threads," with the hope that they truly capture concurrent threads of plot through the stream.

Certain events in such a threaded graph are of particular importance. Nodes with no incoming edges signal the beginning of a new thread, unassociated with any previously-seen segments either directly or transitively. The transitivity check is what distinguishes this "introductory" segment from the "character introduction" segments of Section 5.4.2. The complementary check, for path-ending nodes with no exiting edges, can also be easily performed.

Vertices where the paths converge in time are also important in the evolution of the plot,

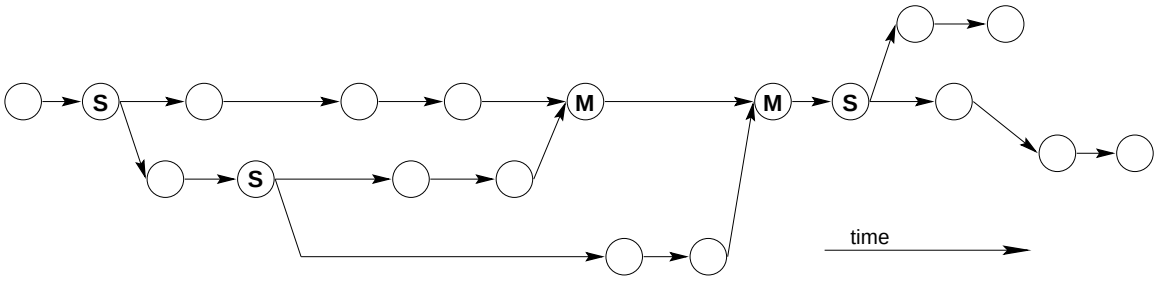


Figure 6.1: Schematic graph of our plot thread model. Merge nodes are indicated by an M, split nodes by an S. Concurrent chains of association are shown as parallel horizontal sets of nodes.

and even short summaries would likely benefit from their inclusion. If low-level similarity accurately reflects high-level semantics, such merge nodes would occur when two different story lines finally meet, joining into a single plot thread. Similarly, path split nodes have multiple exiting edges and a single entering edge; these correspond to a semantic split into two "sibling" threads.

Path merge and split nodes can be detected by the methods of Section 5.4.2, using the memory-based distance matrix described in the previous section. Path beginning and ending nodes can also be detected, using the same prototypes as the character introduction and departure idioms, by substituting the memory-based distance matrix $\mathbf{D}'$ for $\mathbf{D}$. Due to their complex interaction with any errors in both modalities' measured distance matrices, detection of path-merge/split/start/end segments is not as accurate as that of basic idiomatic sequences, such as dialog.

To facilitate the generation of plots like Figure 6.1, we assign thread numbers to nodes

according to the following heuristic:

```
maxThr = 0
for ( i = 1 to N )
    if ( NumEnteringEdges( n_i ) ≠ 1 )
        // either root or merge node; allocate new thread
        maxThr = maxThr + 1
        nodeThr[n_i] = maxThr
    else
        n_e = the single node with an edge to n_i
        if ( NumExitingEdges( n_e ) > 1 )
            // we have new siblings; form new thread
            maxThr = maxThr + 1
            nodeThr[n_i] = maxThr
        else
            // same thread as parent
            nodeThr[n_i] = nodeThr[n_e]
        end
    end
end
```

On termination, the vector `nodeThr[]` is the list of thread numbers, one per node. This method will tend to over-allocate threads, as it creates a new thread for every merge and split segment. In the absence of further information, this is unavoidable due to the fact that we cannot always judge which of the two merging/splitting threads is "closer" to the segment in question, and therefore which of the multiple threads should be preserved after (before) the merge (split).

In the next chapter, we will examine an application of the transitive linking principles

outlined here. By displaying memory-based (Section 6.3) transitive links in a visual summary, in addition to relevant direct links, viewers can more easily see distant connections. In addition, the threading algorithm presented on page 128 will be vital to the placement of nodes in the graphing methods of Section 7.1.

Chapter 7

Visualization of Multimodal Structure

The graphical interpretations of association matrices, presented in the last chapter, have a natural extension in the realm of visualization. In graphical summaries of media streams, two conflicting goals are both priorities: the summary must be compact enough that the user can easily decide whether the stream is one that merits further study, preferably without any scrolling or mouse movement, yet it must contain enough information to answer rather detailed queries without forcing the user to watch the whole stream. Hierarchical visualization systems are a perfect fit to these goals.

Ideally the visual summary would be completely intuitive: a user with no prior experience should be able understand the summary and, with some degree of confidence, know what to expect when they "zoom in" on a section. In the case of multimodal summaries, it should be visually clear which segments are simultaneous or overlapping and which are not. As people often think of time in a linear fashion, we construct visual summaries that use time as the horizontal dimension. This restriction can force the graphs to be unreasonably large, but a hierarchical representation should mitigate this problem. In contrast, Yeung, *et al.*, avoided the issue in their scene transition graphs by requiring human placement of thumbnails and clusters. A downside to their approach is that the sense of time is often lost in large or complex scene transition graphs; this is particularly true when the same

clusters are visited multiple times during a program, making it difficult to determine which transitions happened when.

There have been a number of attempts to visualize multimedia streams in a compact fashion, with applications from video editing to indexing and search systems. Toklu developed a linear video browser, aimed at editing applications, using video shots as atomic playback units and image cross-sections for a time-based representation alongside colored tracks indicating different speakers (found from closed-caption text)[111]. Minami, *et al.*, developed a similar browser, adding the ability to deal with musical segments [89]. Christel, *et al.*, concatenated carefully selected short clips to construct "skims" roughly one-tenth the length of the original stream; their studies found that using fewer, longer clips helped viewers understand the summary better, even if it meant that the summary was incomplete [112]. Peaks in motion intensity were used by Nam and Tewfik to produce similar skim sequences, duplicating frames as necessary to maintain the rhythm of the original presentation [113]. In addition to the scene transition graph summary representation, Yeo and Yeung also developed a pictorial summary scheme for news stories and short clips [100, 114]. Davis discussed some of the semantic and representational issues in constructing video summaries while developing his complementary iconic and episodic pair of video interfaces [115]. In a more general setting, Tufte examined ways to visualize multiple streams of cause and effect simultaneously yet in an intuitive fashion [116].

Existing hierarchical methods tend to focus on a (possibly implicit) user-selectable threshold to determine the amount of detail to present. DeMenthon, Kobla, and Doermann, for example, constructed a video player that let the user select the coarseness of key frame generation via a threshold control [117]. The threshold determined how many vertices were computed in their high-dimensional curve simplification, where feature vectors consisted of the sizes, positions, and movement of the four largest image blobs within a short time frame. Foote, *et al.*, employed a similar user-controllable threshold on audio and video-based confidence scores to determine the granularity of displayed index points [85].

Minami's "Video in Time" system generated condensed videos at 6 levels of detail, presenting the user with a time-based graph indicating the density of frames used at each level [118]. A two-dimensional representation of feature presence versus time was used by Ponceleon and Dieberger in a hierarchical fashion to summarize sets of video segments [119].

7.1 Plotting a Single Graph

Before considering a hierarchical presentation, we will first construct single layouts based on the association graphs of Chapter 6. The dominant consideration here is node placement, which should be automated yet intuitive. As discussed above, we will let time dictate the horizontal dimension to demonstrate the temporal order of the summary.

Given that we are using segments of video and audio (and possibly other modalities), node layout must consider whether to group modalities together or allow segments of different types to mix. For clarity, we decided on the former approach; placing nodes of different types in close proximity is potentially confusing to the viewer. Separating audio and video in the graph also facilitates the placement of partially-overlapping segments. Our designs will therefore lay out the vertical dimension of audio and video segments independently, placing the time-aligned video and audio layouts one above another in the final plot. Time alignment allows us to maintain the intuitive relationship between coincident audio and video segments. (While the audio and video graph layouts are independent, it should be remembered that the graph's edges are based on transitive associations that *do* use cross-modality information.)

7.1.1 Horizontal Placement

Properly time-aligning the two modalities is slightly tricky, as we cannot simply use a linear time scale on the horizontal axis. The shortest segment in a stream—particularly one with commercial spots—can have a duration well under a second. Making node widths

proportional to segment lengths means that the shortest segment must still be wide enough to be visible, and ideally large enough to contain information such as the start time and a thumbnail image. Calibrating the horizontal scale to this shortest node will result in an unusably wide graph layout. Keeping node widths constant introduces a complementary problem: the interval between node start times in the same thread must be wide enough to see the adjacent edge, yet the absolute duration of the shortest such interval may be very small. Overlapping nodes would result, an even more egregious problem than a very wide graph.

As a compromise, we use a non-linear time scale on the horizontal axis. To assist the viewer's sense of duration, faint vertical lines are drawn at fixed intervals in real time. To keep the graph uncluttered, node widths are constant and independent of their segments' respective durations. The following horizontal layout approach is adopted:

1. Place all video nodes sequentially, independent of their start times, packed so just enough space appears between them to see any adjacent edge.

2. Form a list of the start times of each placed video node.

3. Interpolating between time-stamps in this list, set each audio node's horizontal position from its start time, without regard to its potentially overlapping another node.

4. Starting with the leftmost audio node, see if the audio node immediately to its right is too close (the width of the node plus a small inter-node spacing). If it is too close, shift all audio *and* video nodes to the right of this node by the amount necessary to introduce sufficient space.

5. Repeat step 4 for each audio node, working from left to right.

The iteration in steps 4 and 5 effectively spreads out the graph in a nonlinear fashion. Node start times are used in each case to avoid the temporal anomalies induced by having

fixed-width nodes of different durations. Once the node spreading is done, the locations of the vertical cue lines are computed by interpolating among all known time/location pairs.

7.1.2 Vertical Placement

Once the horizontal position of each node is fixed, we must then determine the set of vertical positions; as noted above, this is done independently for audio and video. Even in this constrained one-dimensional problem, node placement algorithms are nearly always heuristic in nature and depend very much on the intent of the presentation [120]. A reasonable placement scheme can be devised from the "plot thread" information already extracted in Section 6.4. Although the schematic layout given in Figure 6.1 works for carefully constructed artificial examples, real media streams tend to have a large number of plot threads throughout their duration, and the connections between these threads can be complex. Showing each independent thread on its own parallel line would lead to dozens of such lines and a huge resulting layout. As node placement is mostly an aesthetic issue, we make the following compromise between showing all plot lines separately and having a compact graph: parallel lines can be re-used for unrelated plot threads, provided the threads do not overlap in time.

The following method is used for vertical placement within each modality, where `nodeThr[]` (computed by the algorithm on page 128) is the vector of thread numbers for each node:

1. Using `nodeThr[]`, determine the (temporally) first and last node occupying each thread.

2. Starting with thread 2, see if any other thread t, where $t < 2$, has a last-occupancy time smaller than thread 2's first node. (If multiple other threads meet this criteria, pick the smallest among them to use as t.) If the condition is true, reassign thread 2 to thread t: update thread t's last-occupancy node number, set thread 2's last occupancy time to 0, and change elements of `nodeThr[]` as appropriate.

3. Repeat step 2 for each remaining thread.

4. For each thread still in use, set the vertical position of all its nodes to be just below all existing nodes.

The thread re-use procedure outlined here is greedy. We make no claim that it will produce the smallest possible layout, but it runs quickly and guarantees that once a thread starts, it stays on the same parallel line and is never interrupted. As discussed in Section 6.4, it is often difficult to know to which of the multiple parents (children) a merge (split) node is "closest;" after thread re-use, it is possible for the merge (split) node to end up on an unintuitively-placed parallel line with respect to the parents (children). Note that threading and relative node placement are done based on the whole set of segments, even though at coarse summaries in Section 7.3's hierarchies, many of the influential segments are hidden.

Vertical placement is first done for all video nodes, then for all audio nodes, forcing audio thread lines to appear below all video threads.

7.1.3 Drawing the Graph

Now that all node placement decisions have been made, we can construct the graphical representation. W3C's recently-standardized scalable vector graphic format, SVG, is used here to allow a familiar web-browser interface that includes zooming and panning facilities as well as hyperlinks [121]. We have implemented an SVG graph generator based on the specifications above and in this section.

Video nodes are represented red and audio nodes in blue. All nodes contain an ordinal index as well as the segment's start and times. Thumbnails are also included for video nodes, generated from the DC+2AC (88×60) versions of frames 10% through their respective shots. When the user points their mouse at one of these thumbnail images, it enlarges to be more visible. (Although the current implementation doesn't support it, using the full uncompressed frame would be even better.) It would be possible to include thumbnails for

audio segments, likely based on the corresponding "character introduction" segments, but any incorrectly detected introduction has the potential to be very misleading if used in this way.

Intra-modality edges, derived from the memory-based graphs, are drawn with arrows indicating forward time. In a simplistic attempt to prevent edges from crossing unrelated nodes, some are drawn as arcs instead of straight lines. Links between audio and video segments, below some distance threshold, are shown without directional arrows. The vertical alignment of associated audio and video segments reinforces the visual aspect of these links.

As mentioned above, faint lines are drawn at fixed intervals in stream time so that the user can sense the non-linear temporal dimension. We chose to draw such lines at every 5% of the stream's duration.

Figure 7.1 shows an example of such a plot, taken from the beginning of the Charlie Rose interview clip. Among its successes are the splitting of the two characters of the initial conversation, joining them at the end, and in finding the isolated logo segment near the end that is unrelated to any previous audio or video. Also apparent from the figure is the fact that the editor chose not to precisely align the audio and video segment boundaries in producing the show; audio segment 2, for instance, contains three video shots. The paths' semantic interpretations, however, are interrupted a bit by video segment 8's association with a similar-looking shot of different characters. Another issue is the lack of audio associations between a number of the initial segments; this is a failure of the audio distance metric described in Chapter 4.

The entire graph for the "Late Show with David Letterman" clip is shown in Figure 7.2. While virtually indecipherable on paper, this large graph is usable via the zoom and pan controls of SVG viewers such as Adobe Systems' web browser plug-in. Like a topographic map, the density of time-stamp lines early in the graph indicates that the first segments have a very long duration. The top thread of video segments, which continues through shot 26, is the main story-line of the show, namely David Letterman speaking either in

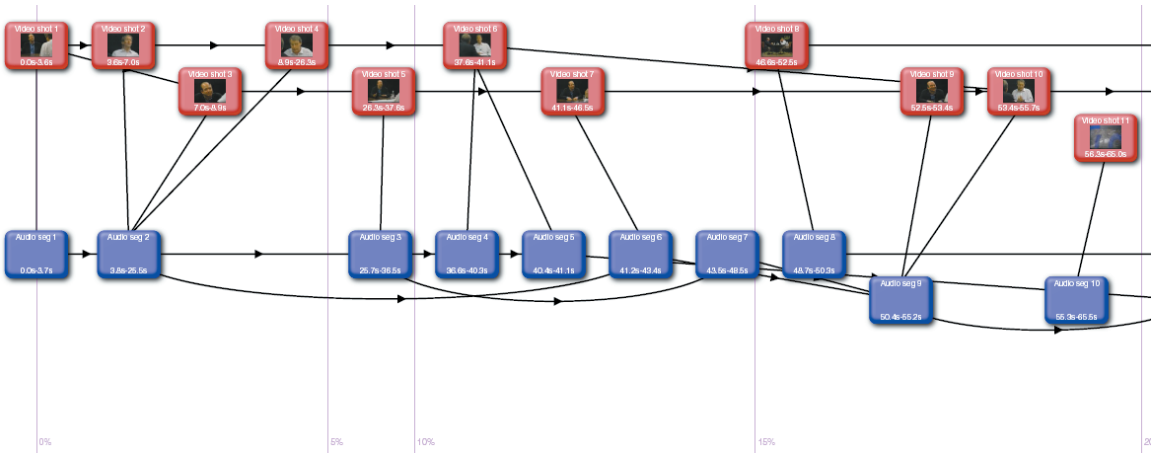


Figure 7.1: Plot of the memory-based graph for the first 65 seconds of the "Charlie Rose" clip. Red (upper) nodes are video shots, blue (lower) are audio segments, and the vertical lines signify constant time intervals.

monologue or at his desk. Toward the end of this thread is an interview, where the guest spends little time talking. Unlike in the Charlie Rose clip, the two participants in the interview are combined in one thread of transitive associations, due both to the minimal guest audio and the numerous wide-angle shots showing both participants speaking at the same time. The second video thread, positioned just below the first from shots two through thirteen, corresponds to times when the show's band is playing. Subsequent video threads, which are all re-assigned to the top row of segments, are portions of a pre-recorded sequence with music and interview shots of a number of people around Los Angeles.

7.2 Node Ranking

Given the size of the graph for even the 10-minute Letterman clip, it is apparent that hierarchical graphing methods will be beneficial. There are a number of methods for hierarchically collapsing the graphs, some domain-specific and some more general. Our approach is to collapse graphs by selecting certain segments to be displayed at each hierarchical level. To

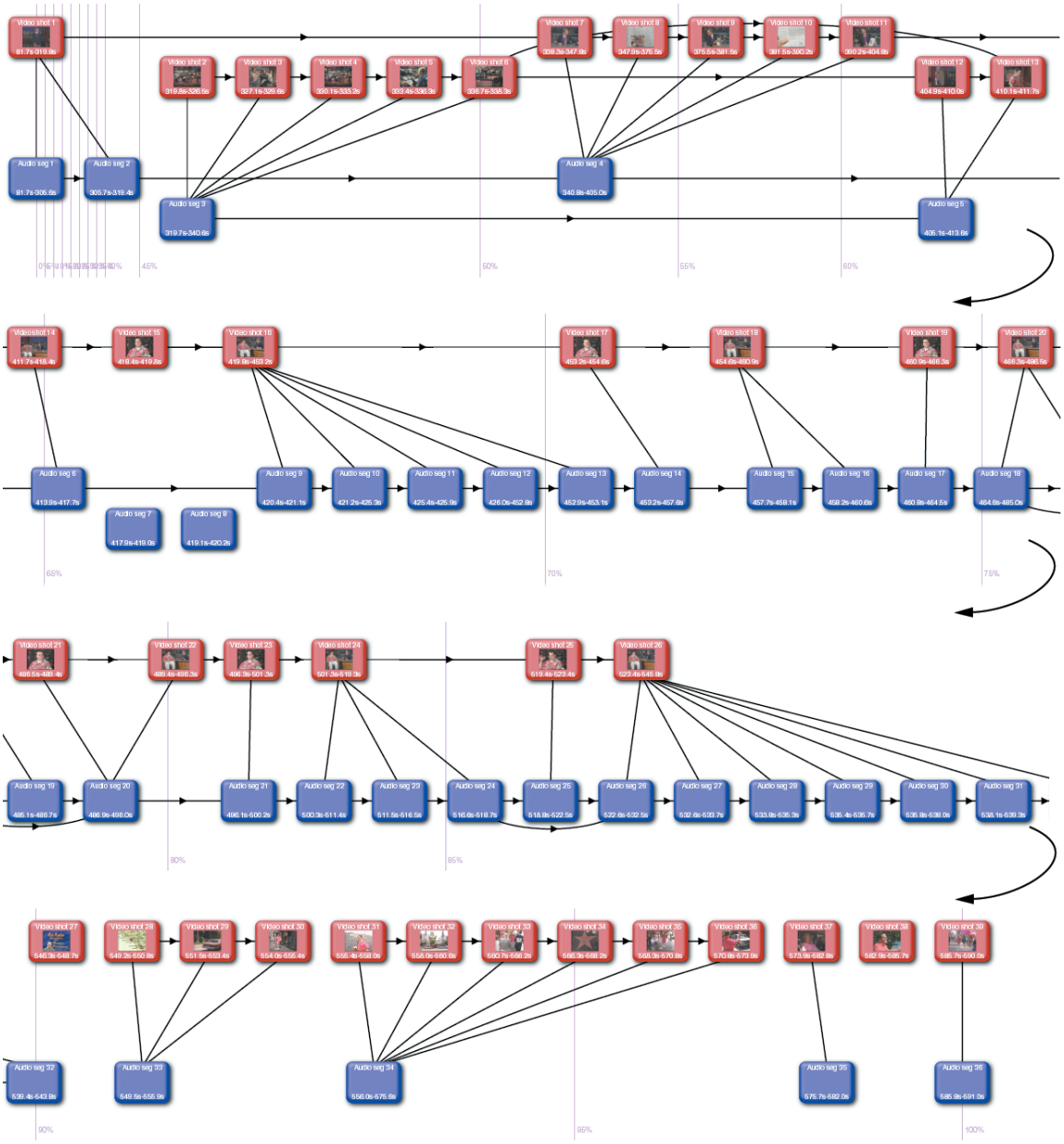


Figure 7.2: Plot of the memory-based graph for the entire "Late Show with David Letterman" clip. Red (upper) nodes are video shots, blue (lower) are audio segments, and the vertical lines signify constant time intervals. Due to width constraints, the plot is wrapped into four rows.

do this, we need to rank segments according to their priority; rankings are based on inclusion in various idiomatic sequences. Although we have not explored it, there would likely be some benefit to incorporating non-temporal segment statistics, such as mean volume of audio or motion vector magnitudes in video.

We rank nodes by their inclusion in idiomatic sequences as shown in Table 7.1; this is by no means the only possible ordering. Rank 1 signifies the most "important" nodes, thus the top level coarse summary would include only nodes from that rank. The next coarsest summary, level 2, would consist of nodes from ranks 1 and 2; the level k summary would include nodes of all ranks $\leq k$. Nodes are members of multiple idiomatic sequences will be assigned the highest appropriate rank (but no additional bonus, another option). Any empty ranks are eliminated by shifting all lower-ranked nodes up by one in rank. The exact manner of linking and drawing edges will be discussed in the next section.

The large number of possible ranks may cause the hierarchical structure to be too deep, consisting of many nearly-identical levels. As such, we elect to equalize rank sizes by enforcing a growth factor γ (in our case, $\gamma = 2.0$). Once the initial ranks are computed according to Table 7.1, nodes are ordered by rank in a single list, with nodes of equal rank in a shuffled order. The first four nodes in the list—typically the first and last audio and video segments—are then chosen as the new rank 1. The next 4γ nodes are given rank 2 and so on, with rank k having $4\gamma^{k-1}$ segments. There may not be enough nodes to fill the highest rank, but the corresponding hierarchy level will contain all segments. (The intra-rank shuffling is to prevent early segments from being biased toward higher ranks by being added first when old ranks are split.) Note that the number of nodes *added* in each new level is what increases the factor γ, so the total number of nodes at level l is $\sum_{k=1}^{l} 4\gamma^{k-1}$. The logarithmic nature of this rank reassignment should flatten even the longest stream's hierarchy to an acceptable number of levels.

Rank	Description
1	First and last audio and video segments
2	video ∩ audio
3	video only — Character introduction segments
4	audio only
5	video ∩ audio
6	video only — Path merge segments
7	audio only
8	video ∩ audio
9	video only — Path split segments
10	audio only
11	video ∩ audio
12	video only — Topic change sequences' first segments (i.e., where the topic change occurs)
13	audio only
14	video ∩ audio
15	video only — Return-to-anchor sequences' first segments (i.e., the "anchor")
16	audio only
17	video ∩ audio
18	video only — Interlude sequences' first and last segments
19	audio only
20	video ∩ audio
21	video only — Character departure segments
22	audio only
23	video ∩ audio
24	video only — Action sequences' first and last segments
25	audio only
26	video ∩ audio
27	video only — Dialog sequences' first two segments (i.e., both participants)
28	audio only
29	Segments aligned in audio and video
30	Video shots greater than 7 seconds and audio segments greater than 10 seconds long

Table 7.1: Segment rank assignment for hierarchical graphing, where rank 1 signifies the most important nodes. Inclusion in idiomatic sequences is determined by the methods of Section 5.4. video ∩ audio indicates intervals that belong to the corresponding idiomatic sequence in both the video *and* audio domains.

7.3 Hierarchical Graphing

Given a set of node rankings over the entire video stream, it is straightforward to construct a hierarchical set of graphs. At each level, the selected nodes (of ranks less than or equal to the level number) are drawn, and any memory-graph style transitive paths between those nodes are displayed. (The techniques of Section 6.3 are used to find such transitive paths among the subset of nodes.) Appropriate below-threshold elements of $\mathbf{D}_{AV}$ are drawn as edges between the corresponding audio and video nodes.

Moving from one level of the hierarchy to the next level is done by clicking on a node. The next level is then displayed, centered on the node of interest (which is guaranteed by rank to be included in the new graph). Beyond the third or fourth level, it is likely that the whole graph will no longer fit within the viewport; the user will have to use the panning tools to scroll if they would like to see beyond what is shown.

At each level, the entire graph is drawn and, as noted above, the initial view is centered on the node of interest. This is done to keep with a map-like approach. An alternative would be to draw only those nodes surrounding the one of interest, dropping the rest of the graph. We consider this to be an unintuitive loss of information, particularly if the user wants to find the context of a particular scene. As the initial view at each level is zoomed-in on the segments of interest, and much of the graph is off-screen, the resulting quantity of information in our approach is not excessive.

An example of the hierarchical browsing interface is shown in Figure 7.3; for compactness, all subsequent figures will show only the graphs themselves. For large graphs, links are provided to quickly pan to the beginning or end of the graph, as well as to zoom out such that the whole graph fits in the window, for a bird's-eye view.

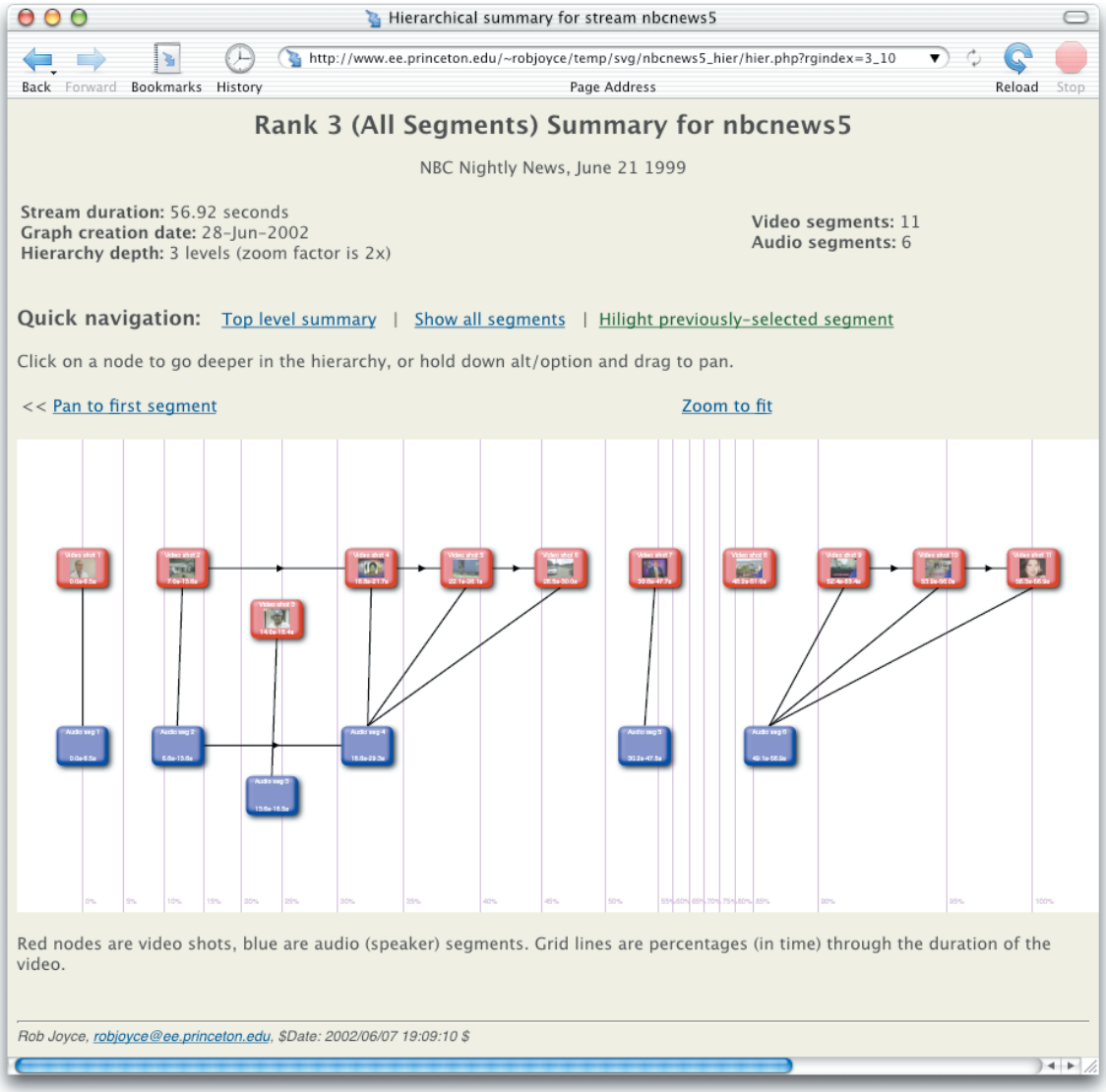


Figure 7.3: The complete interface to our hierarchical multimedia browser.

7.4 Hierarchical Graph Examples

We conclude this chapter with some detailed examples of hierarchical graphs. This is done both to get a sense of how the graphs work and are interrelated, as well as to determine what types of semantic events are commonly captured and which are missed.

"Charlie Rose" Clip

Figures 7.4–7.7 show a typical series of hierarchical views for the five minute PBS "Charlie Rose" video clip. The top level summary, Figure 7.4, shows just the first and last segments of each modality. Clicking to zoom in on video shot 1 yields the display shown in Figure 7.5, consisting of roughly the first sixty percent of the show's duration. In the audio domain, the graph correctly represents the end of the first interview thread, in that video shot 17 and beyond are not part of it. The unfortunate video association between shot 8 and a subsequent similar shot of different people appears in this view as a separate thread, whereas semantically shot 8 is better associated with earlier shots.

Zooming in on video shot 17 (the far right-hand segment) results in Figure 7.6, a level 3 summary. Most of the nodes in this view are of various video games, narrated by a single speaker. While the level 3 summary does contain audio segments, neither of the two occurring during the seven displayed video shots was deemed important enough to plot; one must pan to the left or right to see audio. It would be reasonable in this case to augment the graph with well-aligned audio segments where they give some context to the video. A more general solution would be to design a joint video-audio segment ranking that ensures some measure of equality among nodes added from each available modality.

Clicking on video shot 15 yields Figure 7.7, the base level summary, which introduces the audio segment we want. It is here we see the narration clearly, and can infer that the blue-shirted individual in shot 15's thumbnail is the one describing the subsequent images.

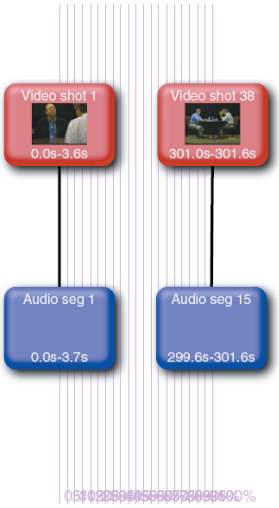


Figure 7.4: "Charlie Rose" top level graph, showing just the first and last segments of audio and video.

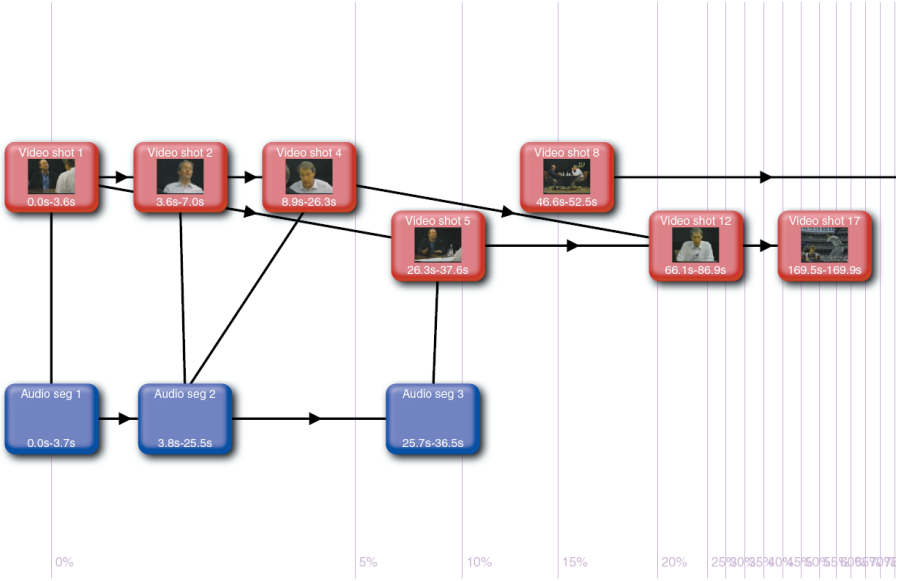


Figure 7.5: "Charlie Rose" level 2 graph, resulting from clicking on video shot 1 of Figure 7.4.

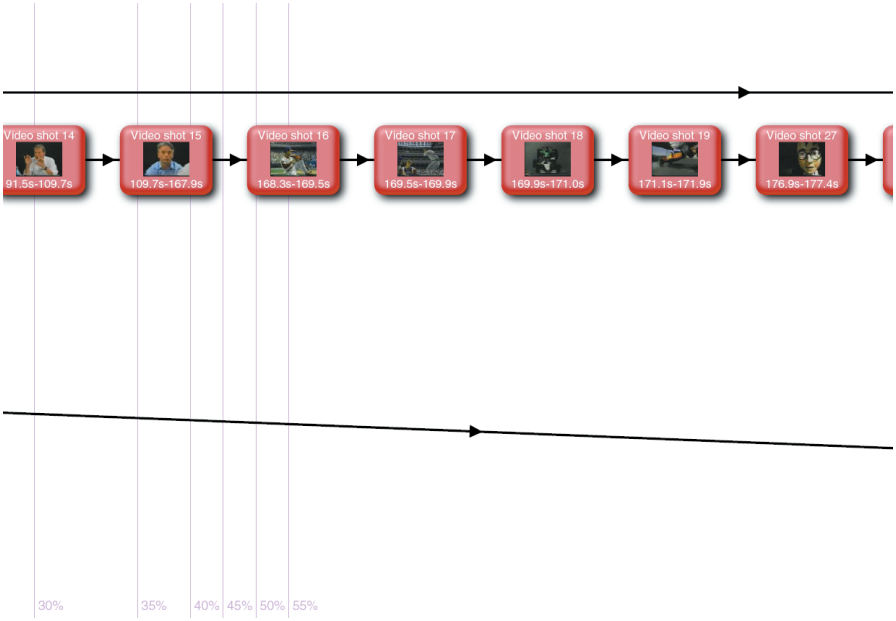


Figure 7.6: "Charlie Rose" level 3 graph, resulting from clicking on video shot 17 of Figure 7.5.

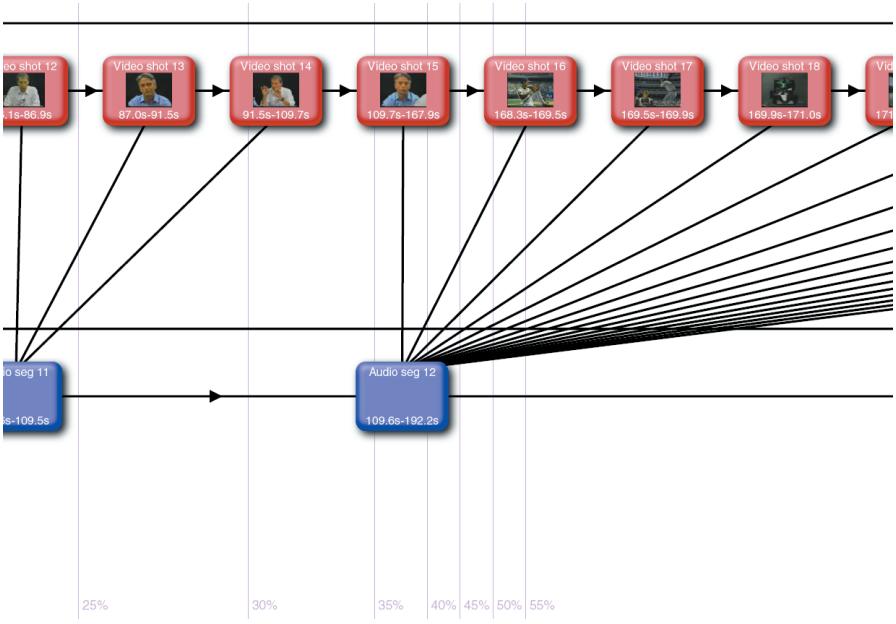


Figure 7.7: "Charlie Rose" level 4 graph, showing all video and audio segments within the displayed interval.

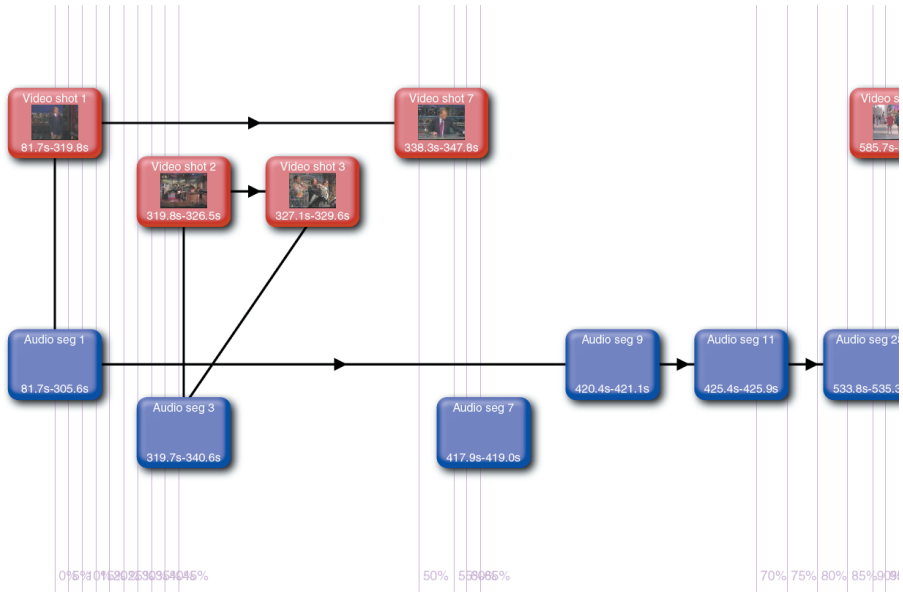


Figure 7.8: "The Late Show with David Letterman" level 2 graph, produced by clicking on the first video segment in the top level graph.

"Late Show with David Letterman" Clip

The first nine minutes of an episode of CBS's "The Late Show with David Letterman" are examined in Figures 7.8–7.10. The top-level summary is not shown, as its 4-segment structure is the same as the Charlie Rose summary of Figure 7.4. Clicking on the earliest video shot at the top level results in the view of Figure 7.8. The triangle of associations among video shots 2 and 3 and audio segment 3 is the show's band playing, an event roughly independent of David Letterman's dialog and subsequent desk shots; the graph shows this independence from the continuing story-line quite well.

From this point on, the user continually decides to zoom in on video shot 7, yielding Figure 7.9 followed by Figure 7.10. Clicking on shot 7 there results in the base level graph view, which in this particular case is identical to Figure 7.10; it already contains all the nodes within the visible interval.

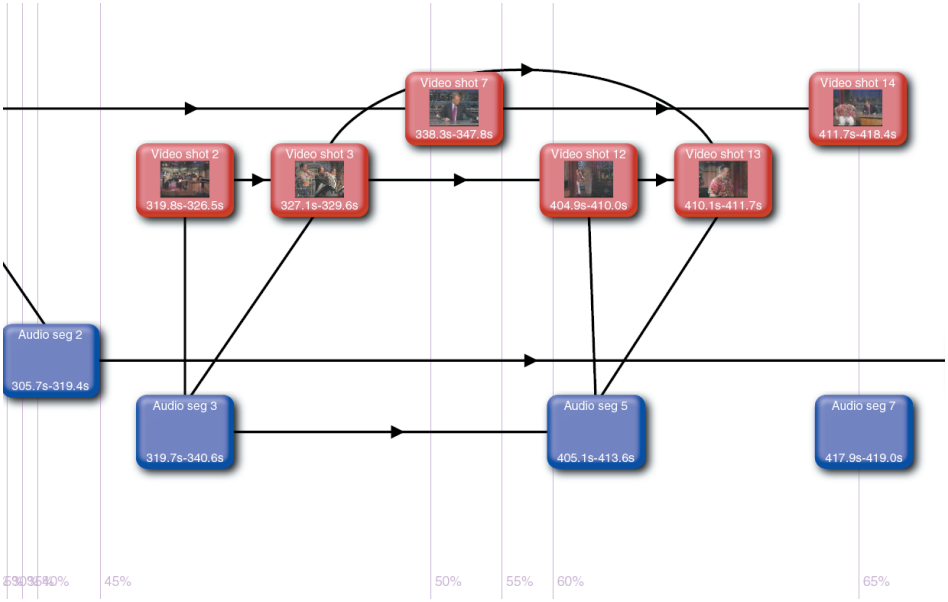


Figure 7.9: "The Late Show with David Letterman" level 3 graph, produced by clicking on video shot 7 in Figure 7.8.

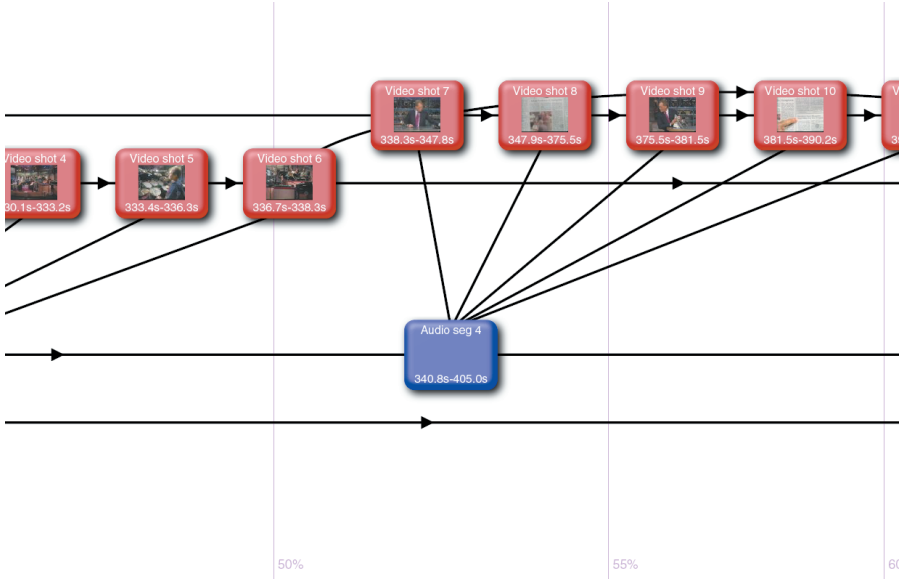


Figure 7.10: "The Late Show with David Letterman" level 4 graph, produced by clicking on video shot 7 in Figure 7.9. Although not the base level graph, this view shows all the segments in the given interval, so further zooming yields no more information.

"Frasier" Clip

A one-minute clip from NBC's "Frasier" is examined in Figures 7.11–7.13; as in the Letterman example, the top level summary is omitted. Clicking on the first audio segment results in Figure 7.11, showing the first portion of the clip. The first three audio and video segments are well-represented as three separate semantic episodes, but beyond that point the graph is less instructive. A dialog begins in audio segment 5, but that is not apparent due to the lack of segment 6 in this level's summary. A similar problem occurs in the video domain; the structure is not apparent from visible nodes.

The user then clicks on video shot 6 in order to get a better view, producing Figure 7.12. The dialog is clearly apparent here; in particular, notice how video shots 6 and 8 are properly associated in the dialog, even though their visual contents are very different! Aside from the two diagonal edges leading from audio segments 6 and 8, this part of the graph is a perfect example of a dialog. A more intuitive representation of such dialogs might benefit a casual user.

The base level view, obtained by zooming in on video shot 4, is shown in Figure 7.13. At this point, the user can see what happens just before the dialog begins: the side character in video shot 2 never speaks, the main character walks into the hall in video shot 3, and he encounters the woman in video shot 4. The only shortcoming is that audio segment 4 should be associated with segment 6, as segment 4 is the true beginning of the dialog; this is the fault of the audio segment distance metric.

CBS Local News Clip

Summary level 2 of a 7.5 minute CBS local news clip appears in Figure 7.14. The segments presented roughly correspond to the major elements of this section of the newscast: video shots 1 and 63 are the beginning of previews of news to come, video shot 4 introduces the newscast, and video shot 5 begins the first story. Video shots 7 and 29 are the first non-anchor shots of two respective stories. Shot 40 corresponds to the same story as shot

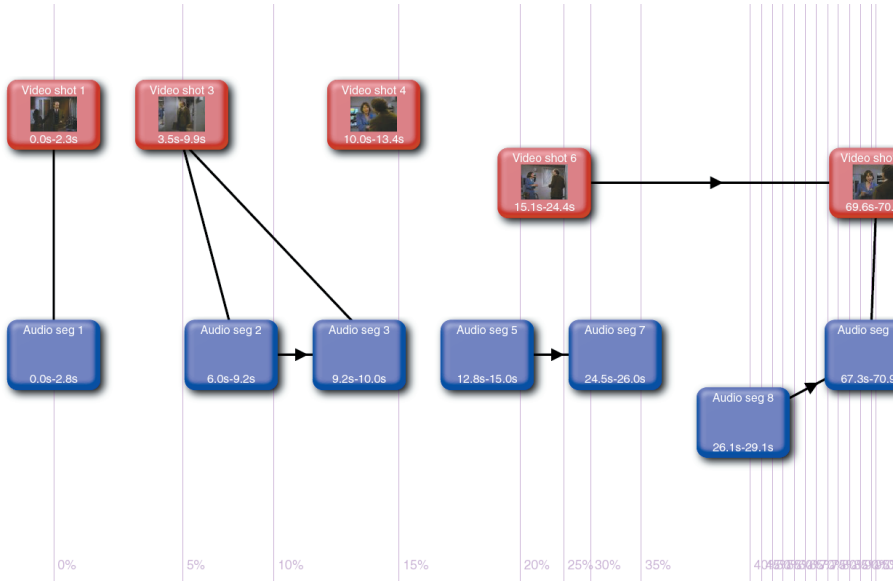


Figure 7.11: "Frasier" level 2 graph, generated by zooming in on video shot 1 at the top level.

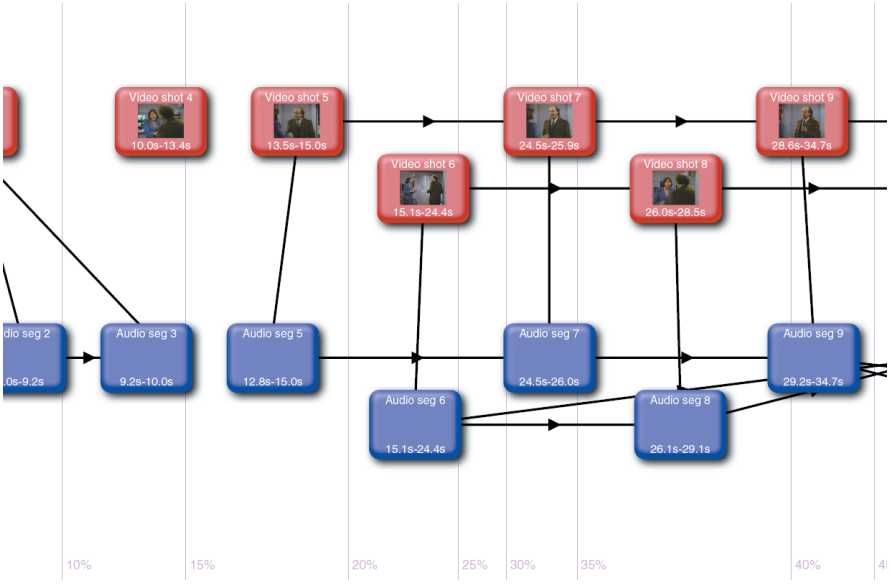


Figure 7.12: "Frasier" level 3 graph, generated by zooming in on video shot 6 in Figure 7.11.

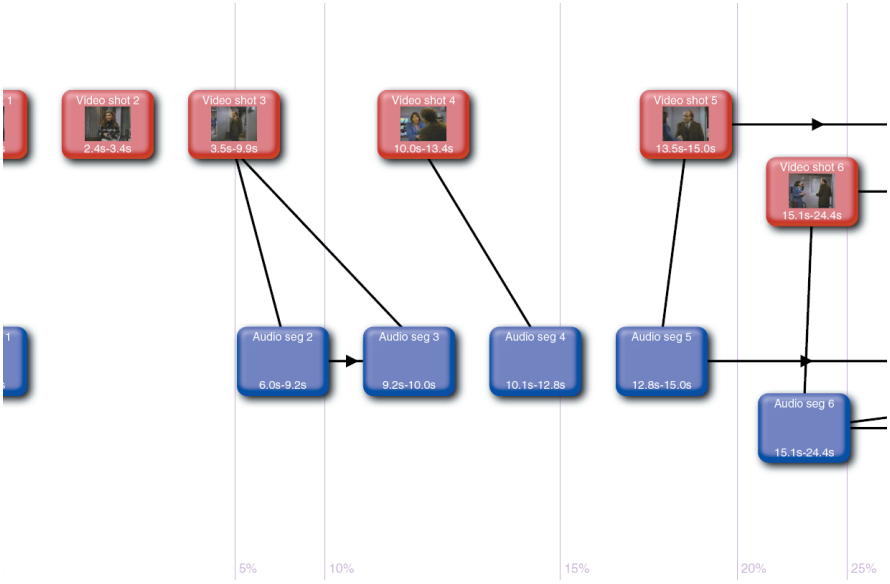


Figure 7.13: "Frasier" base level graph, generated by zooming in on video shot 4 in Figure 7.12.

29, although it takes place in a different location. Missing, though, is a shot representing the final full story of the newscast, consisting of shots 55–60.

Clicking on video shot 7 gives the layer 3 summary in Figure 7.15. Unfortunately, this summary does not show more information about the corresponding news story, ostensibly what the user had intended. Looking at the plot, it is clear that audio information without corresponding video is not too useful—particularly in the absence of audio "thumbnails." As we mentioned in the "Charlie Rose" example, it might be beneficial to balance the number of audio and video segments presented in any particular time interval.

Not finding what they had intended, the user pans around the graph to see what has been excluded from the viewport and settles on video shot 55. Zooming in on that shot yields the view in Figure 7.16. This plot shows the independence of the preview section at the end of the clip, and indicates that a longer-term story ends just before that point. Clicking on video shot 54, at the end of that story, gives the base level graph view shown in Figure 7.17. It is here we finally see the short, independent story from shots 55 through 59;

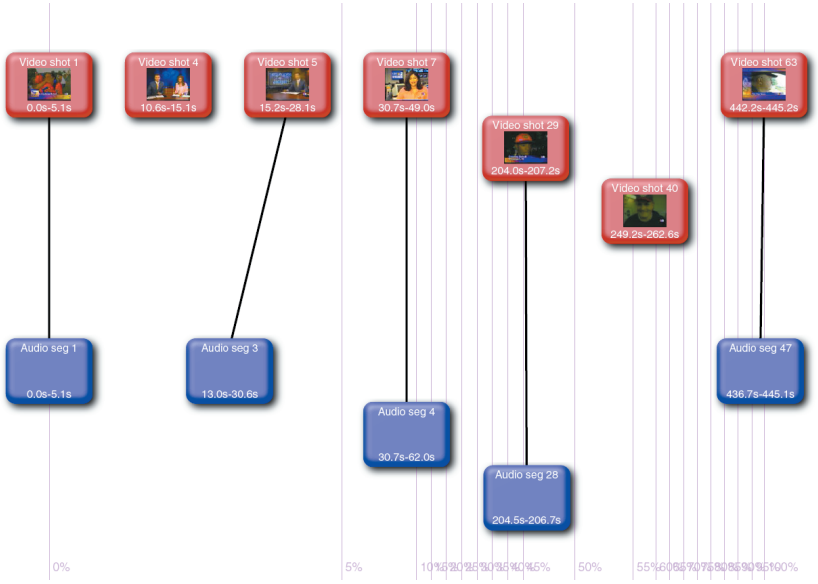


Figure 7.14: CBS local news level 2 graph, produced by clicking on video shot 1 at the top level.

perhaps domain-specific information would have helped in selecting a representative shot from this story to be given a high rank.

Finally, Figure 7.18 gives a bird's-eye view of the same base level graph; this is obtained by clicking on the "zoom to fit" link of the interface in Figure 7.3. It is impossible to see the thumbnails and shot times at this scale, but the rough structure is visible. Narrated sections are clear, as are independent scenes. The upper parallel lines among audio segments show how different portions of the clip are connected by common anchor-people. Segments that deviate from these two lines involve other speakers, likely correspondents or interviewees in a particular story. While not quite intuitive to an unfamiliar user, this broad overview is very useful in gaining some perspective on the clip as a whole.

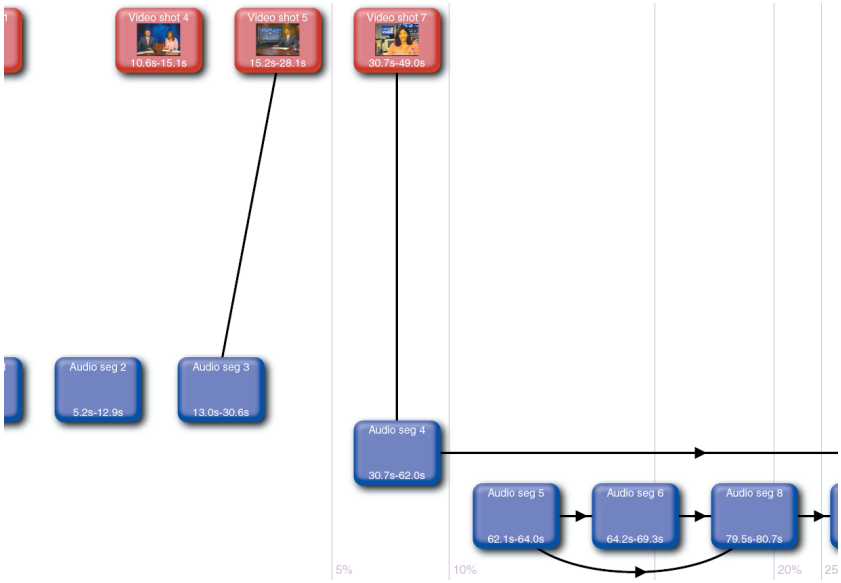


Figure 7.15: CBS local news level 3 graph, produced by clicking on video shot 7 in Figure 7.14.

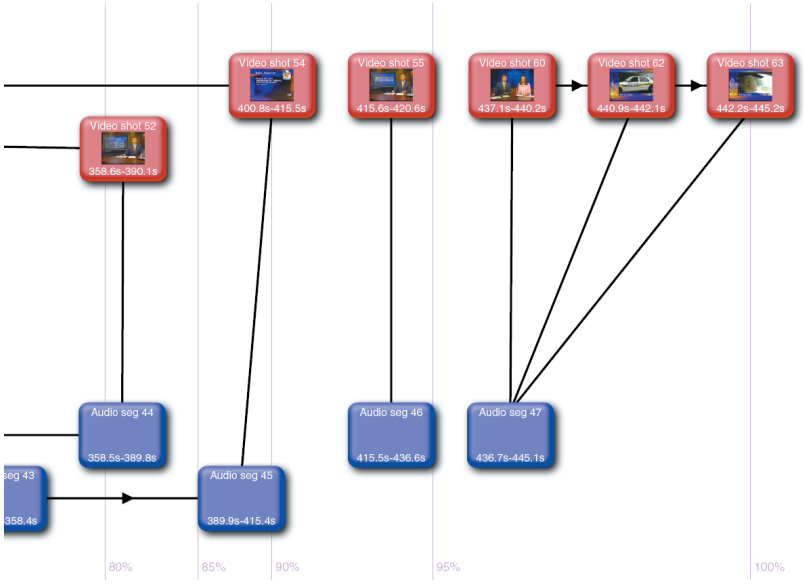


Figure 7.16: CBS local news level 4 graph, produced by clicking on video shot 55 in the level 3 graph.

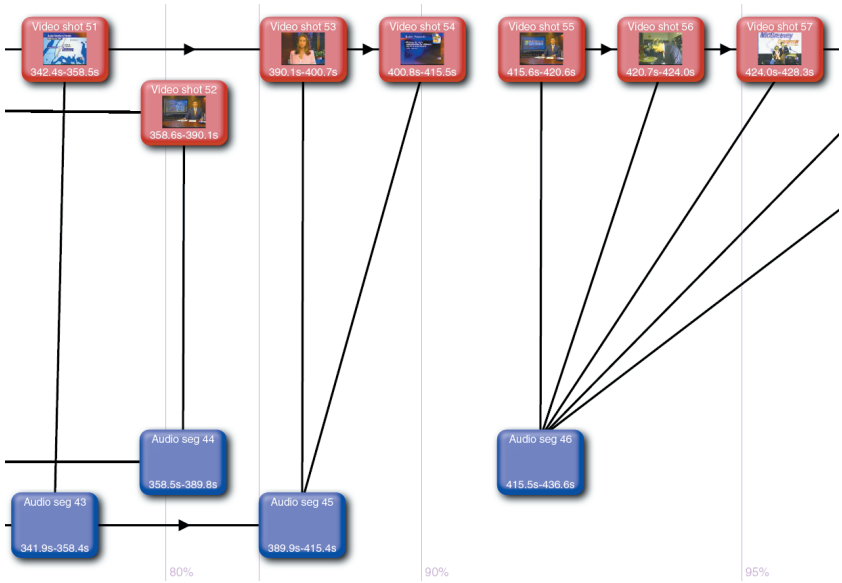


Figure 7.17: CBS local news base level graph, produced by clicking on video shot 54 in Figure 7.16.

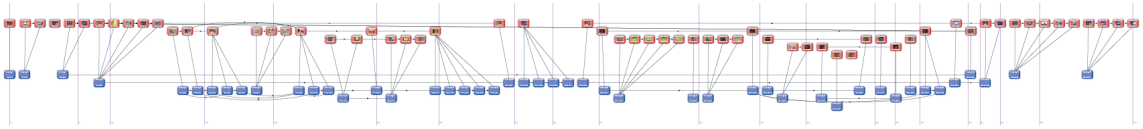


Figure 7.18: Entire CBS local news base level graph, showing all audio and video segments.

Chapter 8

Summary and Conclusions

We have proposed a framework for the temporal analysis of multimedia streams together with some applications of this framework. Temporal analysis involves three general steps: segmentation of the media stream, determination of which segments are associated with one another, and using the association information to determine higher-level events.

The thesis begins with a further development in Chapter 2 of existing video segmentation techniques, allowing for the detection of gradual transitions of a number of types. Dissolve and fade transitions are studied and properties of frame-triplet correlations are used to develop a parametric detector. Transitions affecting evolving subsets of pixels, often incorporating computer effects, are then examined. These transitions, commonly known as wipes, have a triplet correlation structure in histogram-space that is easy to detect. The dissolve detector is able to detect 53 of 56 dissolves in 13 minutes of video, with 6 false alarms, when cascaded with an existing cut detection algorithm. Incorporating a wide variety of wipes within the test set, 60 of 62 are detected, with 35 false alarms per 23.5 minutes; the false detections are largely due to significant histogram changes during motion.

Chapter 3 explores the application of temporal segmentation and video processing techniques to the prediction of variable bit rate video bandwidth requirements. Using such "content-based" boundaries as bandwidth renegotiation points dramatically improves overall prediction performance. Within each interval, short-term traffic observations are found

to be better predictors of traffic than content features, but the latter are beneficial as well. Combining content-based renegotiation with short-term observations and a neural network predictor, an improvement of 18% in link utilization is observed over techniques using only traffic information for prediction. Our work utilizes only the content boundary and intra-shot information; there are likely significant benefits to incorporating the temporal structure information of later chapters in bandwidth prediction. In particular, knowledge that a new shot is similar to one we've already seen offers a solid starting point for prediction, as we know the exact bandwidth characteristics of the old shot.

With the video segmentation problem relatively well-solved, we look in Chapter 4 at issues in combining audio information with video for true multimodal processing. Segmentation of audio according to speaker is found to be unsatisfactory for the task of determining long-term temporal structure, but given the segment boundaries, distance metrics can be developed which distinguish speakers relatively well. In a similar manner, we define a video segment distance metric that takes into account the temporal progression of actions by comparing the end of one shot to the beginning of another. As it is difficult to meaningfully compare these distance metrics, we develop a perceptually-based normalization technique based on 3-state hypothesis testing. In the video domain, detection probabilities of 53% and 99% are found for identical and different shot pairs, respectively. The audio distance metric fares worse, with a "same" segment-pair detection rate of 14% and different segment-pair detection rate of 94%. As we see in later chapters, the contributions of these metrics' errors are significant; the audio metric in particular deserves further study.

Given the problems with audio segmentation, and the complex interaction of video segmentation issues with the segment pair normalization, we choose to hand-segment the media streams used in the remainder of the thesis. Characterizing the effects of segmentation errors, not only on segment-pair distances but on temporal structure inferences as well, remains an open issue that merits exploration.

In Chapter 5 we introduce a representation for audio, video, and cross-modality distance

information that will be used throughout the remainder of the work. The general form of our "association matrix" allows for any number of modalities and distance metrics within and between them; we use a restricted form consisting of one video and one audio distance metric, plus a temporal-overlap based cross-modality metric. With streams roughly one half-hour or less in length, the entire association matrix can reasonably be used at once; a method for dealing with longer streams is needed that can discard some earlier segments (those deemed unimportant) in order to save memory by focusing on the most recent associations. This becomes even more critical if more than two modalities or three distance metrics are used.

Normalizing the size of association matrix columns to the length of the respective segments allows for an at-a-glance overview of the structure of the stream under study, as we see through a number of examples. Perhaps more importantly, the association matrix allows for straightforward algorithms to detect "idiomatic" sequences such as dialogs, character introductions, and returning shots to anchor people. We present a few examples, but an enormous variety is possible. Further, exploring sequences that have both audio and video requirements seems very worthwhile, as does incorporating non-temporal shot information. Domain-specific knowledge could also be used to construct event detectors using the association matrix.

A graphical interpretation of the association matrices is used in Chapter 6 to facilitate the study of long-term transitive connections among segments. Shortest path algorithms are applied to the resulting graphs—possibly with restricted edge sets—to determine the smallest set of connections between two given endpoints. If all the distance metrics are perfect, this should correspond to the semantic question, "What sequence of events leads from A to B?" Transitive links' existence is used to form memory-based graphs, where only the temporally-closest (possibly transitive) connections are shown. Heuristics are then applied to infer parallel semantic "threads," as well as places where such threads merge and split.

Chapter 7 combines the thread information from the memory-based graphs with the idiomatic sequence detectors of Chapter 5 to form hierarchical representations of media streams. Segments are first ranked according to their inclusion in various idiomatic sequences, and subsets are chosen to display at each level of the hierarchy. At a given level, the threaded memory-based graphs are plotted for video and audio side-by-side, with temporal alignments also indicated. Although this is one way of simultaneously representing audio and video information, future work should consider alternatives that do not require parallel graphs, yet are intuitive enough to allow inexperienced users to determine plot connections and events.

Several examples are presented at the end of the chapter. In most cases, important events are well-represented by the plots and zoomed-in views. The choice of nodes to include can seem a bit arbitrary at times; a better method for balancing audio and video segment additions, as well as spreading them out over the duration of the stream, would be beneficial. Also needed is a way to provide thumbnails for the audio segments; short sound clips are one possibility, but ideally an image-based thumbnail, based on the speaker, should be used if the speaker's face can be *reliably* determined. An incorrect thumbnail is worse than none at all.

Another avenue for exploration is the incorporation of features other than temporal connections in the graphs. For example, the mean volume of an audio segment, or the amount of motion in a video shot, could be used to influence node rankings. Such statistics could also be graphically represented in plots as a visual aid. Even more sophisticated would be to do database or web searches based on particular segments, such as character introductions, to gauge their level of importance to a typical user (for instance, an interview of a well-known personality or news anchor is likely more pertinent than one of a man on the street).

In this thesis, we propose algorithms for extraction of low-level audio and video temporal information (segment boundaries) as well as higher-level relationships, and examine

a number of ways to exploit knowledge of such boundaries and relationships. It cannot be said that we have exhaustively derived the temporal structure of a media stream at the highest level. We have, however, provided a flexible method of representing available data and a number of tools to interpret it.

Bibliography

[1] M. Abdel-Mottaleb, N. Dimitrova, L. Agnihotri, S. Dagtas, S. Jeannin, S. Krishnamachari, T. McGee, and G. Vaithilingam, "MPEG 7: A content description standard beyond compression," in *Proc. IEEE 42nd Midwest Symposium on Circuits and Systems*, August 1999.

[2] MPEG Requirements Group, "Overview of the MPEG-7 standard," MPEG Singapore Meeting, edited by J. M. Martinez, `http://www.cselt.it/mpeg/standards/mpeg-7/mpeg-7.htm`, March 2001.

[3] Minerva Ming-Yee Yeung, *Analysis, Modeling and Representation of Digital Video*, Ph.D. thesis, Princeton University, Department of Electrical Engineering, November 1996.

[4] Ruud M. Bolle, Boon-Lock Yeo, and Minerva M. Yeung, "Video query: Beyond the keywords," Research Report RC 20586, IBM, October 1996.

[5] C. Saraceno and R. Leonardi, "Identification of story units in audio-visual sequences by joint audio and video processing," in *Proceedings of the IEEE International Conference on Image Processing*, October 1998, vol. 1, pp. 363–367.

[6] David Bordwell and Kristin Thompson, *Film Art: An Introduction*, McGraw-Hill, 3rd edition, 1990.

[7] Albert S. Bregman, *Auditory Scene Analysis: The Perceptual Organization of Sound*, MIT Press, 1990.

[8] Eric D. Scheirer, *Music-listening Systems*, Ph.D. thesis, MIT, June 2000.

[9] 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*, 1999, vol. 6, pp. 3025–3028.

[10] Yap-Peng Tan, Drew D. Saur, Sanjeev R. Kulkarni, and Peter J. Ramadge, "Rapid estimation of camera motion from compressed video with application to video annotation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 1, pp. 133–146, February 2000.

[11] Stanley Boykin and Andrew Merlino, "Machine learning of event segmentation for news on demand," *Communications of the ACM*, vol. 43, no. 2, pp. 35–41, February 2000.

[12] HongJiang Zhang, Atreyi Kankanhalli, and Stephen W. Smoliar, "Automatic partitioning of full-motion video," *Multimedia Systems*, vol. 1, pp. 10–28, 1993.

[13] Jianhao Meng, Yujen Juan, and Shih-Fu Chang, "Scene change detection in a MPEG compressed video sequence," in *Digital Video Compression: Algorithms and Technologies*. Proceedings of the SPIE, February 1995, vol. 2419, pp. 14–25.

[14] Hain-Ching H. Liu and Gregory L. Zick, "Scene decomposition of MPEG compressed video," in *Digital Video Compression: Algorithms and Technologies*. Proceedings of the SPIE, February 1995, vol. 2419, pp. 26–37.

[15] Ke Shen and Edward J. Delp, "A fast algorithm for video parsing using MPEG compressed sequences," in *Proceedings of the IEEE International Conference on Image Processing*, October 1995, vol. 2, pp. 252–255.

[16] Boon-Lock Yeo and Bede Liu, "Rapid scene analysis on compressed video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, no. 6, pp. 533–544, December 1995.

[17] Yasuyuki Nakajima, Kiyono Ujihara, and Akio Yoneyama, "Universal scene change detection on MPEG-coded data domain," in *Visual Communications and Image Processing*. Proceedings of the SPIE, February 1997, vol. 3024, pp. 992–1003.

[18] Masaru Sugano, Yasuyuki Nakajima, Hiromasa Yanagihara, and Akia Yoneyama, "A fast scene change detection on MPEG coding parameter domain," in *Proceedings of the IEEE International Conference on Image Processing*, October 1998, vol. 1, pp. 888–892.

[19] H. B. Lu, Y. J. Zhang, and Y. R. Yao, "Robust gradual scene change detection," in *Proceedings of the IEEE International Conference on Image Processing*, October 1999, vol. 3, pp. 304–308.

[20] W. A. C. Fernando, C. N. Canagarajah, and D. R. Bull, "Fade and dissolve detection in uncompressed and compressed video sequences," in *Proceedings of the IEEE International Conference on Image Processing*, October 1999, vol. 3, pp. 299–303.

[21] Aya Aner and John R. Kender, "A unified memory-based approach to cut, dissolve, key frame and scene analysis," in *Proceedings of the IEEE International Conference on Image Processing*, October 2001, vol. 3, pp. 370–373.

[22] John S. Boreczky and Lawrence A. Rowe, "Comparison of video shot boundary detection techniques," in *Storage and Retrieval for Still Image and Video Databases IV*. Proceedings of the SPIE, February 1996, vol. 2670, pp. 170–179.

[23] Rainer Lienhart, "Comparison of automatic shot boundary detection algorithms," in *Storage and Retrieval for Image and Video Databases VII*. Proceedings of the SPIE, January 1999, vol. 3656, pp. 290–301.

[24] Ullas Gargi, Rangachar Kasturi, and Susan H. Strayer, "Performance characterization of video-shot-change detection methods," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 1, pp. 1–13, February 2000.

[25] Hong-Heather Yu and Wayne Wolf, "A multi-resolution video segmentation scheme for wipe transition identification," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1998, vol. 5, pp. 2965–2968.

[26] Hong-Heather Yu, *Digital Multimedia Library Indexing and Retrieval*, Ph.D. thesis, Princeton University, Department of Electrical Engineering, July 1998.

[27] Ramin Zabih, Justin Miller, and Kevin Mai, "A feature-based algorithm for detecting and classifying production effects," *Multimedia Systems*, vol. 7, no. 2, pp. 119–128, 1999.

[28] Min Wu, Wayne Wolf, and Bede Liu, "An algorithm for wipe detection," in *Proceedings of the IEEE International Conference on Image Processing*, October 1998, vol. 1, pp. 893–897.

[29] Hyeokman Kim, Sung-Jun Park, Jinho Lee, Woonkyung M. Kim, and S. Moon-Ho Song, "Processing of partial video data for detection of wipes," in *Storage and Retrieval for Image and Video Databases VII*. Proceedings of the SPIE, January 1999, vol. 3656, pp. 280–289.

[30] C. W. Ngo, T. C. Pong, and R. T. Chin, "Camera break detection by partitioning of 2D spatio-temporal images in MPEG domain," in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, June 1999, vol. 1, pp. 750–755.

[31] Soo-Chang Pei and Yu-Zuong Chou, "Efficient and effective wipe detection in MPEG compressed video based on the macroblock information," in *Proceedings of the IEEE International Conference on Image Processing*, September 2000, vol. 3, pp. 953–956.

[32] Mark S. Drew, Ze-Nian Li, and Xiang Zhong, "Video dissolve and wipe detection via spatio-temporal images of chromatic histogram differences," in *Proceedings of the IEEE International Conference on Image Processing*, September 2000, vol. 3, pp. 929–932.

[33] Adnan M. Alattar, "Wipe scene change detection for use with video compression algorithms and MPEG-7," *IEEE Transactions on Consumer Electronics*, vol. 44, no. 1, pp. 43–51, 1998.

[34] Vikrant Kobla, David Doermann, and Christos Faloutsos, "VideoTrails: Representing and visualizing structure in video sequences," in *Proceedings of the ACM Conference on Multimedia*, November 1997, pp. 335–346.

[35] Vikrant Kobla, Daniel DeMenthon, and David Doermann, "Special effect edit detection using VideoTrails: A comparison with existing techniques," in *Storage and Retrieval for Image and Video Databases VII*. Proceedings of the SPIE, January 1999, vol. 3656, pp. 302–313.

[36] W. A. C. Fernando, C. N. Canagarajah, and D. R. Bull, "Wipe scene change detection in video sequences," in *Proceedings of the IEEE International Conference on Image Processing*, October 1999, vol. 3, pp. 294–298.

[37] Robert A. Joyce and Bede Liu, "Temporal segmentation of video using frame and histogram space," submitted to *IEEE Transactions on Multimedia*, July 2001.

[38] Robert A. Joyce and Bede Liu, "Temporal segmentation of video using frame and histogram space," in *Proceedings of the IEEE International Conference on Image Processing*, September 2000, vol. 3, pp. 941–944.

[39] Didier Le Gall, "MPEG: A video compression standard for multimedia applications," *Communications of the ACM*, vol. 34, no. 4, pp. 46–58, April 1991.

[40] H. Vincent Poor, *An Introduction to Signal Detection and Estimation*, Springer-Verlag, 2nd edition, 1994.

[41] "Resource Reservation Protocol (RSVP)," white paper, Cisco Systems Inc., `http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/rsvp.htm`, June 1999.

[42] Raj Jain, "Recent advances in networking," Course web page, Ohio State University, `http://www.cis.ohio-state.edu/~jain/cis788-99/`, 1999.

[43] H. Zhang and E. W. Knightly, "RED-VBR: A new approach to support delay-sensitive VBR video in packet-switched networks," in *Proceedings of NOSSDAV*, April 1995, pp. 258–272.

[44] S. Chong, S. Li, and J. Ghosh, "Predictive dynamic bandwidth allocation for efficient transport of real-time VBR video over ATM," *IEEE Journal on Selected Areas of Communication*, vol. 13, no. 1, pp. 12–23, January 1995.

[45] M. R. Izquierdo and D. S. Reeves, "A survey of statistical source models for variable bit-rate compressed video," *Multimedia Systems*, vol. 7, no. 3, pp. 199–213, 1999.

[46] Nikolaos D. Doulamis, Anastasios D. Doulamis, George E. Konstantoulakis, and George I. Stassinopoulos, "Efficient modeling of VBR MPEG-1 coded video sources," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 1, pp. 93–112, February 2000.

[47] A. M. Dawood and M. Ghanbari, "MPEG video modelling based on scene description," in *Proceedings of the IEEE International Conference on Image Processing*, October 1998, vol. 2, pp. 351–355.

[48] Paul Bocheck and Shih-Fu Chang, "Content-based VBR traffic modelling and its application to dynamic network resource allocation," Research Report 48c-98-20, Columbia University, January 1998.

[49] S.-Y. Kung, *Digital Neural Networks*, Prentice Hall, 1993.

[50] E. W. Knightly and H. Zhang, "D-BIND: An accurate traffic model for providing QoS guarantees to VBR traffic," *IEEE Transactions on Networking*, vol. 5, no. 2, pp. 219–231, April 1997.

[51] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 1999.

[52] J. Kittler, "Feature set search algorithms," in *Pattern Recognition and Signal Processing*, C. H. Chen, Ed. Sijthoff & Noordhoff, 1978.

[53] D. F. Specht, "A general regression neural network," *IEEE Trans. Neural Networks*, vol. 2, no. 6, pp. 568–576, 1991.

[54] H. Schioler and U. Hartmann, "Mapping neural network derived from the Parzen window estimator," *Neural Networks*, vol. 5, no. 6, pp. 903–909, 1992.

[55] Min Wu, Robert A. Joyce, Hau-San Wong, Ling Guan, and Sun-Yuan Kung, "Dynamic resource allocation via video content and short-term traffic statistics," *IEEE Transactions on Multimedia*, vol. 3, no. 2, pp. 186–199, June 2001.

[56] Hau-San Wong, Min Wu, Robert A. Joyce, Ling Guan, and Sun-Yuan Kung, "A neural network approach for predicting network resource requirements in video transmission systems," in *Proceedings of the IEEE Pacific Rim Conference on Multimedia*, December 2000.

[57] Keinosuke Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, 2nd edition, 1990.

[58] Min Wu, Robert A. Joyce, Sun-Yuan Kung, and Ling Guan, "Dynamic resource allocation via video content and short-term traffic statistics," in *Proceedings of the IEEE International Conference on Image Processing*, September 2000, vol. 3, pp. 58–61.

[59] Lawrence Rabiner and Biing-Hwang Juang, *Fundamentals of Speech Recognition*, Prentice-Hall, 1993.

[60] Bruce P. Bogert, M. J. R. Healy, and J. W. Tukey, "The quefrency alanysis of time series for echoes: Cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking," in *Proceedings of the Symposium on Time Series Analysis*. Brown University, June 1962, pp. 209–243.

[61] Perry R. Cook, Ed., *Music, Cognition, and Computerized Sound: An Introduction to Psychoacoustics*, MIT Press, 1999.

[62] Don Kimber and Lynn Wilcox, "Acoustic segmentation for audio browsers," in *Proc. 28th Symposium on the Interface of Computing Science and Statistics*, July 1996, pp. 295–304.

[63] Jonathan Foote, "A similarity measure for automatic audio classification," in *Proc. AAAI Spring Symposium on Intelligent Integration and Use of Text, Image, Video, and Audio Corpora*, March 1997.

[64] M. Shridhar, N. Mohankrishnan, and M. A. Sid-Ahmed, "A comparison of distance measures for text-independent speaker identification," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 1983, vol. 2, pp. 559–562.

[65] Herbert Gish, Man-Hung Siu, and Robin Rohlicek, "Segregation of speakers for speech recognition and speaker identification," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 1991, vol. 2, pp. 873–876.

[66] Herbert Gish and Michael Schmidt, "Text-independent speaker recognition," *IEEE Signal Processing Magazine*, vol. 11, no. 4, pp. 18–32, October 1994.

[67] Man-Hung Siu, George Yu, and Herbert Gish, "An unsupervised, sequential learning algorithm for the segmentation of speech waveforms with multiple speakers," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 1992, vol. 2, pp. 189–192.

[68] Lonce Wyse and Stephen W. Smoliar, "Toward content-based audio indexing and retrieval and a new speaker discrimination technique," in *Proceedings of the International Joint Conference on Artificial Intelligence*, August 1995.

[69] Jeho Nam, A. Enis Çetin, and Ahmed H. Tewfik, "Speaker identification and video analysis for hierarchical video shot classification," in *Proceedings of the IEEE International Conference on Image Processing*, October 1997, vol. 2, pp. 550–553.

[70] Kazumasa Mori and Seiichi Nakagawa, "Speaker change detection and speaker clustering using VQ distortion for broadcast news speech recognition," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2001, vol. 1, pp. 413–416.

[71] Matthew A. Siegler, Uday Jain, Bhiksha Raj, and Richard M. Stern, "Automatic segmentation, classification and clustering of broadcast news audio," in *Proceedings of the DARPA Speech Recognition Workshop*, February 1997, pp. 97–99.

[72] Eric D. Scheirer and Malcolm Slaney, "Construction and evaluation of a robust multifeature speech / music discriminator," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 1997, vol. 2, pp. 1331–1334.

[73] George Tzanetakis and Perry Cook, "Sound analysis using MPEG compressed audio," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, June 2000, vol. 2, pp. 761–764.

[74] Zhu Liu, Jincheng Huang, Yao Wang, and Tsuhan Chen, "Audio feature extraction and analysis for scene classification," in *Proceedings of the IEEE Workshop on Multimedia Signal Processing*, June 1997, pp. 343–348.

[75] Tong Zhang and C.-C. Jay Kuo, "Heuristic approach for generic audio data segmentation and annotation," in *Proceedings of the ACM Conference on Multimedia*, October 1999, pp. 67–76.

[76] Thomas Kemp, Michael Schmidt, Martin Westphal, and Alex Waibel, "Strategies for automatic segmentation of audio data," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, June 2000, vol. 3, pp. 1423–1426.

[77] Silvia Pfeiffer, Stephan Fischer, and Wolfgang Effelsberg, "Automatic audio content analysis," in *Proceedings of the ACM Conference on Multimedia*, 1996, pp. 21–30.

[78] George Tzanetakis and Perry Cook, "Multifeature audio segmentation for browsing and annotation," in *Proc. 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, October 1999, pp. 103–106.

[79] Hari Sundaram and Shih-Fu Chang, "Audio scene segmentation using multiple features, models, and time scales," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, June 2000, vol. 4, pp. 2441–2444.

[80] Yuh-Lin Chang, Wenjun Zeng, Ibrahim Kamel, and Rafael Alonso, "Integrated image and speech analysis for content-based video indexing," in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, June 1996, pp. 306–313.

[81] Jincheng Huang, Zhu Liu, and Yao Wang, "Integration of audio and visual information for content-based video segmentation," in *Proceedings of the IEEE International Conference on Image Processing*, October 1998, vol. 3, pp. 526–530.

[82] Jeho Nam and Amhed H. Tewfik, "Combined audio and visual streams analysis for video sequence segmentation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1997, vol. 4, pp. 2665–2668.

[83] Hari Sundaram and Shih-Fu Chang, "Video scene segmentation using video and audio features," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, August 2000, pp. 1145–1148.

[84] Zhu Liu and Yao Wang, "Major cast detection in video using both audio and visual information," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2001, vol. 3, pp. 1413–1416.

[85] Jonathan Foote, John Boreczky, Andreas Girgensohn, and Lynn Wilcox, "An intelligent media browser using automatic multimodal analysis," in *Proceedings of the ACM Conference on Multimedia*, September 1998, pp. 375–380.

[86] Zhu Liu and Qian Huang, "Detecting news reporting using audio/visual information," in *Proceedings of the IEEE International Conference on Image Processing*, October 1999, vol. 3, pp. 324–328.

[87] Chalapathy V. Neti and Andrew Senior, "Audio-visual speaker recognition for video broadcast news," in *Proceedings of the DARPA Broadcast News Workshop*, March 1999.

[88] Sofia Tsekeridou and Ioannis Pitas, "Content-based video parsing and indexing based on audio-visual interaction," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 4, pp. 522–535, April 2001.

[89] Kenichi Minami, Akihito Akutsu, Hiroshi Hamada, and Yoshinobu Tonomura, "Video handling with music and speech detection," *IEEE Multimedia*, vol. 5, no. 3, pp. 17–25, September 1998.

[90] Howard D. Wactlar, Alexander G. Hauptmann, Michael G. Christel, Ricky A. Houghton, and Andreas M. Olligschlaeger, "Complementary video and audio analysis for broadcast news analysis," *Communications of the ACM*, vol. 43, no. 2, pp. 42–47, February 2000.

[91] H. Pan, Z.-P. Liang, T. J. Anastasio, and T. S. Huang, "A hybrid NN-Bayesian architecture for information fusion," in *Proceedings of the IEEE International Conference on Image Processing*, October 1998, vol. 1, pp. 368–371.

[92] ISO-IEC/JTC1 SC29/WG11, "ISO 11172, MPEG-1: Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbits/s," 1993.

[93] School of Computer Science Carnegie Mellon University, "Sphinx-II speech recognition engine," `http://www.speech.cs.cmu.edu/speech/sphinx/`, 2000.

[94] Wayne Wolf, "Key frame selection by motion analysis," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 1996, vol. 2, pp. 1228–1231.

[95] Yueting Zhuang, Yong Rui, Thomas S. Huang, and Sharad Mehrotra, "Adaptive key frame extraction using unsupervised clustering," in *Proceedings of the IEEE International Conference on Image Processing*, October 1998, vol. 1, pp. 866–870.

[96] Jonathan Foote, "Visualizing music and audio using self-similarity," in *Proceedings of the ACM Conference on Multimedia*, October 1999, pp. 77–80.

[97] Caterina Saraceno, "Video content extraction and representation using a joint audio and video processing," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, March 1999, vol. 6, pp. 3033–3036.

[98] Rainer Lienhart, Christoph Kuhmünch, and Wolfgang Effelsberg, "On the detection and recognition of television commercials," in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, 1997, pp. 509–516.

[99] T. McGee and N. Dimitrova, "Parsing TV programs for identification and removal of non-story segments," in *Storage and Retrieval for Image and Video Databases VII*. Proceedings of the SPIE, January 1999, vol. 3656, pp. 243–251.

[100] Minerva Yeung, Boon-Lock Yeo, and Bede Liu, "Extracting story units from long programs for video browsing and navigation," in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, 1996, pp. 296–305.

[101] Yong Rui, Thomas S. Huang, and Sharad Mehrotra, "Exploring video structure beyond the shots," in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, July 1998, pp. 237–240.

[102] Rainer Lienhart, Silvia Pfeiffer, and Wolfgang Effelsberg, "Scene determination based on video and audio features," in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, June 1999, vol. 1, pp. 685–690.

[103] HongJiang Zhang, Shuang Yeo Tan, Stephen W. Smoliar, and Gong Yihong, "Automatic parsing and indexing of news video," *Multimedia Systems*, vol. 2, no. 6, pp. 256–266, 1995.

[104] Borko Furht, Stephen W. Smoliar, and HongJiang Zhang, *Video and Image Processing in Multimedia Systems*, chapter 24, Kluwer Academic Publishers, 1995.

[105] Andrew Merlino, Daryl Morey, and Mark Maybury, "Broadcast news navigation using story segmentation," in *Proceedings of the ACM Conference on Multimedia*, 1997, pp. 381–391.

[106] Frederick Walls, Hubert Jin, Sreenivasa Sista, and Richard Schwartz, "Topic detection in broadcast news," in *Proceedings of the DARPA Broadcast News Workshop*, March 1999.

[107] Wayne Wolf, "Hidden markov model parsing of video programs," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, April 1997, vol. 4, pp. 2609–2611.

[108] Tiecheng Liu and John R. Kender, "A hidden markov model approach to the structure of documentaries," in *Proceedings of the IEEE Workshop on Content-based Access of Image and Video Libraries*, June 2000, pp. 111–115.

[109] John R. Kender and Boon-Lock Yeo, "On the structure and analysis of home videos," in *Proceedings of the Asian Conference on Computer Vision*, January 2000.

[110] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, *Introduction to Algorithms*, MIT Press, 1990.

[111] Candemir Toklu and Shih-Ping Liou, "Image and audio sequence visualization and interaction mechanisms for structured video browsing and editing," in *Proceedings of the IEEE International Conference on Image Processing*, September 2000, vol. 2, pp. 263–266.

[112] Michael G. Christel, Michael A. Smith, C. Roy Taylor, and David B. Winkler, "Evolving video skims into useful multimedia abstractions," in *Proceedings of the ACM Conference on Human Factors in Computing Systems*, April 1998, pp. 171–179.

[113] Jeho Nam and Ahmed H. Tewfik, "Video abstract of video," in *Proceedings of the IEEE Third Workshop on Multimedia Signal Processing*, 1999, pp. 117–122.

[114] Boon-Lock Yeo and Minerva M. Yeung, "Retrieving and visualizing video," *Communications of the ACM*, vol. 40, no. 12, pp. 43–52, December 1997.

[115] Marc Davis, "Knowledge representation of video," in *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI)*, 1994, vol. 1, pp. 120–127.

[116] Edward R. Tufte, *Visual Explanations: Images and Quantities, Evidence and Narrative*, Graphics Press, 1997.

[117] Daniel DeMenthon, Vikrant Kobla, and David Doermann, "Video summarization by curve simplification," in *Proceedings of the ACM Conference on Multimedia*, 1998, pp. 211–218.

[118] Kenichi Minami, Akihito Akutsu, Hiroshi Hamada, and Yoshinobu Tonomura, "Enhanced video handling based on audio analysis," in *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, 1997, pp. 219–226.

[119] Dulce Ponceleon and Andreas Dieberger, "Hierarchical brushing in a collection of video data," in *Proceedings of the 34th Hawaii International Conference on System Sciences*, January 2001, pp. 116–123.

[120] Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, and Kiem-Phong Vo, "A technique for drawing directed graphs," *Proceedings of the IEEE Transactions on Software Engineering*, vol. 19, no. 3, pp. 214–230, 1993.

[121] World Wide Web Consortium, "Scalable vector graphics (SVG) 1.0 specification," W3C recommendation, `http://www.w3.org/TR/SVG/`, September 2001.

Index

action sequences, **108**

aligned a/v sequences, **115**

association matrix, 5, **86**

breadth-first search, 124

causality, 6

cepstrum, 68–73, 78, 82, 90

character departure segments, **110**

character introduction segments, 5, **109**

compressed-domain processing, 9, 10, **11**, 43, 46, 49, 69, 70

consistency measure, **57**, 62

correlation

- analysis of, 25
- frame space, 14, 35
- histogram space, 21

cut, 3, 9, 23, 46, 87, 111

D-BIND traffic descriptor, 47, **48**, 49, 52, 57, 60–65

DC frame, 10, 11, 35, 39, 74

DC+2AC frame, 12, 23, 40, 135

dialog sequences, 5, 107, **108**

displaced frame difference (DFD), **12**, 16, 30, 35, 39

dissolve, 3, 9, **12**, 34, 35

fade, *see* dissolve

general regression neural network (GRNN), 53, **54**, 55–57, 61, 62

generality, as design goal, 6

group audio sequences, **115**

hierarchical summaries, 5

histogram space, 18, 21

idiomatic sequences, 5

- cross-modality, 114
- detection, 102
- prototype structure, 104
- prototypes, 107

independent event sequences, **111**

low complexity, 7

MPEG-7, 2

narration sequences, **115**

path merge segments, 5, **113**, 127

path split segments, **112**, 127

quality of service (QoS), 42–44, 61

resource reservation protocol (RSVP), 43
return-to-anchor sequences, **109**

scalable vector graphics (SVG), 135, 136
scene, 3, 5
sequential forward selection (SFS), **53**, 55–57, 61, 62
shot, 3, 5

testing methods, 7
threads, semantic, 5
topic change sequences, **110**

variable bit rate (VBR), 4, 43–45, 48, 51, 55, 60, 61

wipe, 3, 9–11, **18**, 40

XML, 2