Intelligent Segmentation of Lecture Videos

by

Bassam H. Chaptini

Bachelor of Engineering, Civil and Environmental Engineering, American University of Beirut, 2000

Submitted to the Department of Civil and Environmental Engineering in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Civil and Environmental Engineering

at the

Massachusetts Institute of Technology

June 2002

© Massachusetts Institute of Technology 2002. All rights reserved.

Signature of Author: ____________________________

Department of Civil and Environmental Engineering

May 10, 2002

Certified by: ____________________________

Steven R. Lerman
Professor

Department of Civil and Environmental Engineering

Thesis Co-Supervisor

Certified by: ____________________________

V. Judson Harward
Principal Research Scientist

Thesis Co-Supervisor

Accepted by ____________

Oral Buyukozturk
Chairman, Departmental Committee on Graduate Studies
Intelligent Segmentation of Lecture Videos

by

Bassam H. Chaptini

Submitted to the Department of Civil and Environmental Engineering on May 10, 2002, in partial fulfillment of the requirements for the degree of Master of Science in Civil and Environmental Engineering

ABSTRACT

This thesis presents the design and implementation of a software module that improves the segmentation and automatic content extraction of recorded lecture videos. Video segmentation in general refers to the automated understanding of the semantic meanings of a multimedia document. Segmentation in the context of this research corresponds to dividing the video into semantically meaningful units corresponding to the different topics and ideas that a lecturer has introduced during a classroom lecture. Traditionally, video segmentation techniques heavily rely on visual and audio cues to determine the underlying structure of the video. Applying these traditional techniques to segment lecture videos will not be effective because lecturers do not use significant visual or audio hints to indicate a transition to a new topic. This research proposes a novel technique for segmenting lecture videos by relying on board events such as a “new line” or a “pause in writing” as strong hints that reveal the semantics of a classroom lecture. The software tool described here is the first step towards building a fully automated segmentation system for lecture videos.

Thesis Co-Supervisor: Steven R. Lerman
Title: Professor

Thesis Co-Supervisor: V. Judson Harward
Title: Principal Research Scientist
Acknowledgments

First of all, I wish to express my deepest gratitude to my advisor Prof. Steven Lerman for his advice, help and support, and to Dr. Judson Harward for the invaluable insights, guidance and assistance he offered me over the last two years. Thanks are also extended to Andy Merlino, head of the Virage Advanced Technology Lab, and to David Mycue, director of the Streaming Media Compression Services, for assisting me in developing the plug-in for the Virage VideoLogger.

I also would like to thank all the RA’s and staff at the Center for Educational Computing Initiatives for making CECI such a wonderful place to work.

In countless ways I have received support and love from my family. I would like to take this opportunity to thank them for all the love, affection, encouragement and wonderful moments they shared with me over the years.

Last but not least, I would like to thank Ralph, Christian, Jean-Pierre, Karim, Jad, Ziad, Leila, Lara, Lynne, Richard, Issam and all other friends for making my stay at MIT a memorable experience.
Content

CHAPTER 1 INTRODUCTION................................................................. 7
  1.1 SINGAPORE-MIT ALLIANCE...................................................... 9
  1.2 CURRENT PROCESS.............................................................. 9
  1.3 SEGMENTING LECTURE VIDEOS.......................................... 10
  1.4 RESEARCH OVERVIEW...................................................... 11
  1.5 ORGANIZATION OF THE THESIS........................................ 11

CHAPTER 2 VIDEO SEGMENTATION: AN OVERVIEW....................... 13
  2.1 CONTEXT............................................................................ 14
  2.2 VIDEO SEGMENTATION FRAMEWORK.................................. 14
  2.3 VIDEO SEGMENTATION TECHNIQUES.................................... 16
    2.3.1 Extracting Features..................................................... 16
      2.3.1.1 Video and Audio Features..................................... 17
      2.3.1.2 Other Features.................................................... 22
    2.3.2 Feature Interpretation................................................ 23
      2.3.2.1 Hierarchical Segmentation................................... 23
      2.3.2.2 Video Shot Detection and Classification Using HMM...... 24
    2.3.3 Scene Content Classification..................................... 25
  2.4 SEGMENTATION OF LECTURE VIDEOS................................. 25

CHAPTER 3 TECHNOLOGY OVERVIEW.............................................. 27
  3.1 CURRENT SEGMENTATION OF LECTURE VIDEOS..................... 28
  3.2 THE VIRAGE VIDEOLOGGER.............................................. 29
    3.2.1 Technical overview.................................................. 29
3.2.2 System Architecture ................................................................. 31
3.2.3 Virage Data Model ................................................................. 33
3.2.4 Integrating the VideoLogger in SMA ....................................... 34
3.3 VIDEOLOGGER SOFTWARE DEVELOPMENT KIT (SDK) ............... 35
3.4 VIRTUAL WHITEBOARD AND VIRTUAL “INK” TECHNOLOGIES .... 36
   3.4.1 Overview ............................................................................. 36
   3.4.2 Integrating Virtual Ink in SMA ........................................... 37

CHAPTER 4 THE LECTURE VIDEO PLUG-IN ........................................ 39
   4.1 OVERVIEW ............................................................................ 39
   4.2 EXTRACTED FEATURES .......................................................... 41
   4.3 CUSTOMIZATION .................................................................... 43
   4.4 IMPLEMENTATION ................................................................. 46
      4.4.1 Data File ........................................................................... 46
      4.4.2 Feature Extraction: The plug-in data model...................... 47
      4.4.3 Connecting to the VideoLogger ......................................... 49
   4.5 RESULTS AND LIMITATIONS .................................................. 50

CHAPTER 5 CONCLUSION ...................................................................... 52
   5.1 IMPROVING FEATURE DETECTION EFFECTIVENESS ................ 52
   5.2 CONSTRUCTING SEGMENTATION MODELS ............................ 53
List of Figures

Figure 2.1 General Framework................................................................. 15
Figure 3.1 Virage’s VideoLogger............................................................... 30
Figure 3.2 Virage System Architecture .................................................. 32
Figure 3.3 API Development................................................................. 36
Figure 4.1 Lecture Video Plug-in General Architecture......................... 40
Figure 4.2 Lecture Video Plug-in............................................................ 42
Figure 4.3 Configuration Panel.............................................................. 44
Figure 4.4 Class Diagram...................................................................... 47
Figure 4.5 Plug-in Processes................................................................. 48
Chapter 1

Introduction

Because of its length and unstructured format, efficient access to video is not an easy task without the help of an index that captures the semantic structure of the video content. A lack of such an index makes the tasks of browsing and retrieval inefficient; a user, searching for a particular object of interest in a given video, has to perform a linear search by browsing sequentially through the video until the segment is found. This is obviously a time consuming technique. Constructing a video index, on the other hand, is a tedious task when it is done manually. Video segmentation refers to the computerized understanding of the semantic meanings of a multimedia document and is used to construct a video index to facilitate user access.

Video segmentation is particularly important for universities interested in enriching the educational experience by publishing online multimedia content such as lectures in the form of streaming video. In this way, students who were unable to attend the event itself can benefit from an on-demand experience at their own pace and on their own schedule. They can also review parts of lectures they didn’t fully understand. To be valuable, the recorded lecture video needs to be segmented according to the topics it contains.
Segmentation in this context means dividing the video into semantically meaningful units corresponding to the different topics and ideas that a lecturer has introduced. Extracting the content of the lecture video allows students to identify the underlying structure of the lecture and easily access the parts they are interested in.

The work presented in this thesis is part of the Singapore-MIT Alliance (SMA) development program. SMA is an innovative engineering education and research collaboration among the National University of Singapore (NUS), Nanyang Technological University (NTU), and the Massachusetts Institute of Technology (MIT).

The focus of this research is on automatic content extraction of recorded lecture videos. The work presented here is a first step in addressing SMA’s desired objective for fully automating the indexing of lecture video streams. In particular, this thesis will study:

- Recent advances in video segmentation techniques. The objective of this review is to set a framework for developing a specialized system to segment lecture videos.
- The integration of an “off the shelf” video segmentation engine into the current lecture video segmentation process. The limitations of this commercial system will be further examined.
- Our approach of improving the video segmentation by designing and implementing a software module that looks at the whiteboard activity as a source of information for segmenting lecture videos.
- Some ideas on productive directions for future research.
1.1 Singapore-MIT Alliance

SMA classes are held in specially equipped classrooms at the Singaporean institutions and at MIT using live video transmission over Internet. The synchronous transmission allows participants at both locations to see each other and speak normally. However, because of the 12 time zone difference, SMA has made great efforts to find and develop tools to enhance asynchronous learning. Converting a live classroom experience into randomly accessible asynchronous content allows students to review lecture materials and professors to reuse discrete portions of the recorded class sessions to answer questions or supplement lectures in subsequent semesters. This asynchronous access to lecture material can be achieved by having the classroom activity video recorded and then posted on the Web for asynchronous access.

SMA lectures are given daily, and it is expensive to process, segment and label them through manual methods. Immediate availability of the lectures is also important because the courses are fast-paced and build sequentially on earlier lectures. Techniques for efficient segmentation and retrieval of lecture videos have to be developed in order to cope with the volume of lecture video data produced by the SMA program.

1.2 Current Process

The current process of converting the live classroom experience into randomly accessible, asynchronous resources is labor intensive. A skilled individual has to watch the master recording and manually insert the cuts corresponding to topic changes in the lecture. This is not efficient for two reasons: First, the individual is not generally familiar with the subject matter, nor can he be expected to switch the program perfectly. Second,
transforming the lecture into a series of static images, transferring it to a website and constructing the links is a labor-intensive process [31].

While some of the preparation, systems issues, and quality control are unavoidable, a considerable part of the labor could be automated yielding to two major benefits:

- Lower costs per lecture on converting the video to useful, asynchronous learning resources.
- Faster turnaround time

We need to investigate and design tools to automate most parts of this segmentation. More specifically, we need a tool to automate the web publishing part, and a tool that will facilitate topic change detection and segment the lecture video accordingly.

### 1.3 Segmenting Lecture Videos

Most automated analysis systems rely heavily on visual and audio cues to segment videos. Using these systems to extract the semantics of a classroom lecture will not be effective because lecturers aren’t dramatic in giving their presentation. They can move from topic to topic seamlessly without significant visual or audio cues. In order to intelligently and effectively segment lecture videos, a data source more relevant to the classroom activity needs to be analyzed. There are a number of data sources, such as observing presentation slides or detecting the whiteboard activity, which can reveal pertinent information about the lecture structure. This study is interested in analyzing one such source of information: whiteboard activities.
1.4 Research Overview

Using Virage's VideoLogger™, an "off the shelf" automated segmentation system, would be the first step in fulfilling SMA's requirements of transforming the lecture presentation into a series of static images, transporting them to a web site and performing the appropriate linking to the streaming video. The segmentation engine of the VideoLogger, however, is not effective on lecture videos.

Using the open architecture of the VideoLogger and Virtual Ink's Mimio™, a product that electronically captures handwriting and drawing on a standard whiteboard, a software module that extracts whiteboard activity was developed. This software was successful in systematically detecting all the cues and all board events such as an "Erasure" or a "Pause in writing". However, not all board events necessarily correspond to a topic change in the lecture. Consequently, using the plug-in directly as a content extraction system will not yield good results. Future work needs to investigate and design a separate content extraction system that will be based on board events and other cues to decide when a topic transition occurs.

1.5 Organization of the Thesis

This thesis is organized as follows. Chapter 2 presents an overview of classical and recently proposed techniques for the segmentation of video sequences. As an introduction to the problem, basic concepts and a framework about the segmentation process are outlined. In the following sections, the three steps of the proposed segmentation framework are discussed with an emphasis on the features extraction step. Feature extraction is the first and most basic phase of the process. Finding and implementing a
feature extraction model appropriate for the segmentation of lecture videos is the focus of
the current research.

Chapter 3 is a detailed description of the technologies used in this research. It first
describes the current techniques used to segment lecture videos. It then presents the
Virage VideoLogger, a third party application that performs sophisticated, automated
logging of video assets. This section will also include a description of the VideoLogger’s
impact on the current segmentation process, and its limitation in accomplishing an
effective lecture video segmentation. The remainder of the chapter introduces the two key
technologies that were used to develop a plug-in for the Virage VideoLogger that helps
convert a live synchronous lecture into multiple asynchronous segments each focused on
a single meaningful topic. The Virage VideoLogger Software Developer's Kit provides
developers and systems integrators access to the full range of VideoLogger functions,
which enables the VideoLogger to be tailored to suit our particular needs of segmenting
video lectures. The second main technology used in this research is Virtual Ink’s Mimio
device, which electronically captures handwriting and drawing on a standard whiteboard.
Chapter 4 addresses the design and development of this plug-in. First, the general
architecture of the system is discussed. A presentation of the plug-in’s functionalities and
implementation follow. A discussion of the result of the plug-in and its limitations
conclude this chapter.

Chapter 5 is dedicated to the presentation of possible developments of the work presented
here.
Chapter 2

Video Segmentation: an overview

In this chapter, we will propose an overview of the main developments in the field of video segmentation. The objective is to review recent advances in using multimedia information to accomplish automatic content-based retrieval, and to use these advances as a framework for developing specialized systems for segmenting lecture videos.

This chapter is organized as follows. Sec. 2.2 defines and sets the context and motivation of video segmentation. Section 2.3 presents the framework that is the basis of most segmentation systems, and introduces and defines terminologies that are commonly used in the video segmentation literature. Section 2.4 introduces segmentation techniques that are based on the previous framework. More specifically, Section 2.4.1 reviews feature extraction techniques that are used by most segmentation systems. Section 2.4.2 introduces “Feature Interpretation” which is the next step in the video segmentation framework. Feature interpretation is the first step in delimiting topic segments. Section 2.4.3 is dedicated to the last step, which is the actual content determination of the segments. Finally Section 2.5 introduces the direction of this research based on the review presented in this chapter.
2.1 Context

Video segmentation consists of extracting the semantics of a multimedia document such as a video sequence with an accompanying audio track. Those semantics are expressed in multiple forms that are often complementary to each other. Therefore it is necessary to analyze all types of data: image frames, sound tracks, texts that can be extracted from image frames, and spoken words that can be deciphered from the audio track. This usually involves segmenting the video into meaningful units, classifying each unit into a scene type, and indexing and summarizing the document for efficient retrieval and browsing [34].

First, we need to define terms that are commonly used in the video segmentation literature. A video sequence usually consists of separate scenes, and each scene includes many shots. The ultimate goal of video segmentation is to automatically group shots into what human beings perceive as a scene. Using movie production terminology, a scene is a sequence of semantically coherent shots, a shot is a continuous recording of audiovisual material, a video shot is a successive collection of frames, and a frame is a 2-dimensional image sample [35].

2.2 Video Segmentation Framework

Woudstra [35] presents a general framework for modeling multimedia information. The proposed framework consists of different properties modeled in a three-layer architecture (figure 2.1):
The lowest layer is the data layer. It contains the raw data, format, and attribute properties. *Raw data* is a sequence of elementary data units, *format* is a representation of the raw data (e.g. MPEG-1), and *attribute* is a characterization of the raw data that cannot be extracted from the raw data and its format (e.g. creation date).

The next layer is the feature layer. The feature property is a domain independent representation of the raw data and format (e.g. color histogram).

The last layer is the concept layer. The concept property is the semantic interpretation of the other property types (e.g. commercial, dialog between two persons, high action scene, etc.).

The feature and concept layers contain information that can be determined (respectively via feature extraction and concept extraction) using the properties of the lower layers and
extraction algorithms for features and knowledge rules for concepts. As opposed to features, concepts are context dependent. Different concepts can be inferred from the same lower layers. For example, a ‘thumb pointing upwards’ has a different interpretation for scuba-divers than for stock brokers. The context only influences the usefulness of particular features.

A widely accepted first step towards video segmentation is feature extraction. Features are interpretation independent characteristics. Examples are pitch and noise for audio, and color histogram and shape for images. Quite a number of features have been identified and many feature detection algorithms already exist [1, 12, 24]. The next step is adding semantics to collections of features. For example, from a collection of features (e.g. silence in audio, black frame…) one can infer that a scene change has occurred.

2.3 Video Segmentation Techniques

Studying the algorithms behind the segmentation engines or studying the mechanisms behind the feature extraction techniques is beyond the scope of this study. I will rather present in this section a list of the most useful features (audio, visual and other) that are used in the video segmentation arena. I then discuss some of the segmentation techniques that are based on these features.

2.3.1 Extracting Features

The key to the success of any video segmentation algorithm is the type of features employed for the analysis. Many features have been proposed for this purpose. Some of them are designed for specific tasks, while others are more general and can be useful for a variety of applications. This section will review some of these features.
2.3.1.1 Video and Audio Features

Wang, Liu and Huang [34] studied recent advances in using audio and visual information jointly for segmenting a video. As we have previously mentioned, this usually means dividing the entire video into scenes so that each segment corresponds to a story unit. Sometimes it is necessary to divide each scene into shots, so that the audio and/or visual characteristics of each shot are coherent. An important step that follows the segmentation is the classification of a scene or shot to some predefined category, which can be very specific (an Opera performance), less precise (a music performance), or even less explicit (a scene dominated by music). A key to the success of all these tasks is the extraction of appropriate audio and visual features which will serve as a basis for scenes and shot determination.

Once again, [34] provides a good reference for the most used audio/video features that different video segmentation systems use.

2.3.1.1.1 Audio Features

There are many features that can be used to characterize audio signals. Audio features usually occur on two levels:

- Long term clip level: for a feature to reveal the semantic meaning of an audio signal, the analysis has to be made on a period of time that spans usually from one to several tens of seconds.

- Short term frame level: which can be considered as a group of neighboring samples that last usually from 10 to 40ms, roughly the scale of audio corresponding to a single frame. Frames can be used to detect short-term features such as gunshots.
Volume is the most widely used and most easy to compute feature. Volume is a reliable indicator for silence detection, which may help to segment an audio sequence and to determine clip boundaries. Moreover, the temporal variation of the volume in a clip can reflect the scene content.

Zero Crossing Rate (ZCR) is a count of number of times that the audio waveform crosses the zero axis. Speech can include voiced regions as well as unvoiced regions. Voiced speech includes hard consonants such as "g" and "d", while unvoiced speech includes fricatives such as "f" and "s". Typically, unvoiced speech has low volume and low energy levels and thus can mistakenly be classified as silent. ZCR is one of the most indicative and robust measures to discern unvoiced speech frames from being classified as silence. ZCR has a normal number of cross-overs during silence while it significantly increases during unvoiced speech. By using the combination of ZCR and volume, one can prevent low energy unvoiced speech frames from being classified as silent. For the ZCR to reveal the semantics of an audio signal, the analysis has to be made on a period of time usually from one to several tens of seconds. With a speech signal, low and high ZCR are interlaced. This is because voice and unvoiced audio often occur alternatively. A commercial clip on the other hand usually has a relatively smooth contour because it has a strong background music.

Pitch is the fundamental frequency of an audio waveform. Normally only voiced speech and harmonic music have a well-defined pitch. But the pitch can still be used as a low level feature to characterize the fundamental frequency of any audio waveforms. The typical pitch frequency for a human being for example is between 50-450Hz, whereas the
pitch range for music is much wider. It is difficult to derive the scene content directly from the pitch level of isolated frames, but the dynamics of the pitch contour over successive frames appear to reveal more about the scene content. For instance, in a commercial clip with a music background, there are usually many discontinuous pitch segments; within each segment the pitch value is almost constant. This pattern is due to the music background. The pitch frequency in a speech signal is primarily influenced by the speaker (male or female), whereas the pitch of a music signal is dominated by the strongest note that is being played.

*Spectral Features* refers to the Fourier Transform of the samples of the audio signal. Analyzing the audio signal from a frequency point of view allows the use of powerful signal analysis methods. For instance, one can filter and divide an audio sequence into different segments, so that each segment contains music of the same style or speech from one person. Speech recognition and speaker recognition are other areas where spectral representation of an audio signal is extremely useful.

### 2.3.1.1.2 Video Features

Several papers have been written summarizing and reviewing various visual features useful for video segmentation [19, 29, 34]. The following is a brief review of some of the visual features.

*Color:* The color histogram, which represents the color distribution in an image, is one of the most widely used color features. It is invariant with respect to image rotation, translation, and viewing axis. Statistically, it denotes the joint probability of the
intensities of the three-color channels. Histograms are the most common method used to
detect shot boundaries. The simplest histogram method computes gray level or color
histograms of two frames. If the difference between the two histograms is above the
threshold, a shot boundary is assumed [4].

Texture refers to the visual patterns that have properties of homogeneity that do not result
from the presence of only a single color or intensity [30]. It is an innate property of
virtually all surfaces, including clouds, trees, bricks, hair, and fabric. It contains
important information about the structural arrangement of surfaces and their relationship
to the surrounding environment [14]. Texture is an important feature of a visible surface
where repetition or quasi-repetition of a fundamental pattern occurs.

Shape features can be represented using traditional shape analysis such as moment
invariants, Fourier descriptors, autoregressive models, and geometry attributes [19]. They
can be classified into two categories:

- Global features are properties derived from the entire shape. Examples of global
  features are roundness or circularity, central moments, eccentricity, and major
  axis orientation.
- Local features are those derived by partial processing of a shape and do not
depend on the entire shape. Examples of local features are size and orientation of
  consecutive boundary segments, points of curvature, corners, and turning angles.

Compression: Little, et al [20] used differences in the size of JPEG compressed frames to
detect shot boundaries as a supplement to a manual indexing system.
Motion is an important attribute of video. Motion information can be generated by block-matching or optical flow techniques. Motion features such as moments of the motion field, motion histogram, or global motion parameters can be extracted from motion vectors. High-level features that reflect camera motions such as panning, zooming, and tilting can also be extracted [29]. Because shots with camera motion can be incorrectly classified as gradual transitions, detecting zooms and pans increases the accuracy of shot boundary detection algorithms [4].

The visual features we have just presented can be used in different image retrieval systems to effectively perform a video segmentation. Here, we will select a few representative systems and highlight their distinct characteristics.

Query By Image Content (QBIC) [9, 23, 8] was the first commercial content-based image retrieval system. Its system framework and techniques have had profound effects on later image retrieval systems. QBIC supports queries based on example images, user-constructed sketches and drawings, and selected color and texture patterns. This is achieved by using color feature, color histograms, texture feature, and shape feature [29].

Virage is a content-based image search engine developed at Virage Inc. Similar to QBIC, Virage [2, 13] supports visual queries based on color, composition (color layout), texture, and structure (object boundary information). But Virage goes one step further than QBIC. It also supports arbitrary combinations of the above four atomic queries.
Photobook [26] is a set of interactive tools for browsing and searching images developed at the MIT Media Lab. Photobook consists of three subbooks from which shape, texture, and face features are extracted, respectively. Users can then query, based on the corresponding features in each of the three subbooks.

### 2.3.1.2 Other Features

Several other features have been explored to perform specific tasks.

*Close Captioning:* The closed caption transcript that is broadcast together with the video can be used to detect specific keywords, which might signal a scene cut. In addition, useful information can be derived from both the presence and absence of closed captioning texts at certain times during the video [17].

*Speech recognition* can be used in two different ways [16, 36, 37]:

- **Alignment:** One needs to know exactly when each word in the transcript is spoken. This information is not available from the closed captioning and must be derived by other means. A speech recognition system can provide this information.

- **Keyword detection:** Closed Captioning might not be available. A large vocabulary speech recognition system can be used to detect specific keywords.

*Speaker Identification:* Speaker recognition is the process of determining the identity of a person based on her speech utterance [25], or based on face recognition techniques [28].
Using speaker recognition, a system can generate a descriptive index that indicates the presence of a person in a particular video segment.

### 2.3.2 Feature Interpretation

Referring back to figure 2.1, we have discussed different techniques in feature extraction. In this part, we will briefly review two approaches to video segmentation and feature interpretation.

In “2.3.1.1.2 Video Features”, we have presented techniques such as QBIC, which can be considered as a low level interpretation. These are early models of video segmentation, which are primarily focused on visual information. The resulting segments usually correspond to individual camera shots.

The next two models that we will present are more advanced and are concerned with clustering “shots” into “scenes”. Strictly speaking, such clustering depends on the understanding of the semantic content of the video. By joint analysis of certain features, however, it is sometimes possible to recognize shots that are related by location or events, without actually invoking high-level analysis of semantic meanings [34].

#### 2.3.2.1 Hierarchical Segmentation

In [18], a hierarchical segmentation approach was proposed that detects scene breaks and shot breaks at different hierarchies. At the lowest level, audio, color and motion breaks are detected. The next level determines shot segments by combining the results from detecting both color and motion breaks. At the highest level, scene breaks are detected by looking for frames for which both visual and audio breaks are identified. The algorithm is based on the observation that a scene change is usually associated with simultaneous
changes of color, motion, and audio characteristics, whereas a shot break is only accompanied with visual changes. For example, a TV commercial often consists of many shots, but the audio in the same commercial follows the same rhythm and tone. The algorithm proceeds in two steps. First, significant changes in audio, color, and motion characteristics are detected separately. Then shot and scene breaks are identified depending on the coincidence of changes in audio, color, and motion. Audio continuity is used to recombine certain shots into “scenes”.

2.3.2.2 Video Shot Detection and Classification Using HMM

A common approach to detect shot boundaries is to compute the difference between the luminance or color histograms of two adjacent frames and compare it to a preset threshold. A problem with this approach is that it is hard to select a threshold that will work with different shot transitions. To circumvent this problem, an alternative approach has been proposed. It uses “Hidden Markov Models” (HMM) to model a video sequence and accomplish shot segmentation and classification simultaneously [3]. Selecting the thresholds is usually done manually. However, with HMMs these parameters can be learned automatically. The use of HMM for modeling a video sequence is motivated by the fact that a video consists of different shots connected by different transition types and camera motions. The states used by these HMMs are the shots themselves, the transitions between them (cuts, fades, and dissolves), and camera motions (pans and zooms). An introduction to the key concepts of HMM is included in chapter 5.
2.3.3 Scene Content Classification

The last step in the video segmentation process is to label a segment or a scene as belonging to one of several predefined semantic classes. Referring back to figure 2.1, this corresponds to choosing one of the many possible contexts. At the lowest level, a segment can be categorized into some elementary classes such as speech vs. music, indoor vs. outdoor. At the next level, some basic scene types can be distinguished such as dialog between two people, an indoor concert, etc. The highest level involves determining the semantic meaning of the scene.

2.4 Segmentation of Lecture Videos

In this part of the thesis, we have introduced a general framework employed for video segmentation systems. The first step in this framework is to extract features and information from the raw data, which usually is a multimedia stream. The more general features are usually audio (e.g. volume, pitch, etc.) and video (e.g. color, texture, etc), but other features such as speaker recognition and close captioning have proven, in some contexts, to be effective. Video segmentation systems use a joint analysis of the different features to cluster shots into scenes and meaningful stories. As features are extracted, a video segmentation model deduces the semantics of the video.

In the last chapter, we will look at MITRE’s Broadcast News Navigator (BNN) as an example of a segmentation system. This system has concentrated on automatic segmentation of stories from news broadcasts. This system is comparable to the segmentation of lecture videos because the BNN system is heavily tailored towards a specific news format, namely CNN prime time [17]. This is in the same spirit as the objective of this research, and should serve as a basis for future work.
As previously mentioned, the most important and basic step in a video segmentation engine is to extract the right features. As we have mentioned in chapter 1, professors aren’t usually dramatic in giving their lectures. They can move from topic to topic without dramatic visual or audio cues. Visual cues such as a camera zoom or switch aren’t correlated with topic changes in a lecture setting. The audio stream often provides more hints. Detecting silences and background sound helps determine the beginning and end of a lecture. Speaker recognition helps identify a “question and answer” session, where the audio switches back and forth from teacher to students. But these different features are not sufficient to extract the different topics that the professor has discussed in a lecture. One potential source of information more relevant to the classroom dynamics is the whiteboard activity. There are many other features, such as observing presentation slides, which can be used to identify the lecture structure. Extracting a comprehensive set of features related to the classroom activity and developing a fully automated segmentation system that includes the three steps of a segmentation process (feature extraction, feature interpretation and content classification) is beyond the scope of this initial study in automating the segmentation of lecture videos. This research is rather interested in investigating the first step of the segmentation process: identifying and extracting one set of features, the whiteboard activity, and exploring its relevance to the classroom and teacher’s activity. Ultimately, the whiteboard activity should be combined with other classroom specific features, namely presentation slides, and other features such as audio and video to construct a model that intelligently segments lecture videos. The semantic interpretation of the features will also be the subject of future work.
Chapter 3

Technology Overview

An introduction to the various technologies used in developing the software module that extracts whiteboard events is presented here. Section 3.1 describes the current techniques used to segment lecture videos. The Virage VideoLogger is then introduced in section 3.2. An overview of the algorithms, architecture and data model will be presented first. This will be followed by a description of the VideoLogger’s impact on the current segmentation process, and its limitation in accomplishing an effective lecture video segmentation. The next two sections introduce two key technologies that were used to develop the plug-in. Section 3.3 presents the Virage VideoLogger Software Developer's Kit which provides developers access to the full range of VideoLogger functions and which enables the VideoLogger to be tailored to suit our particular needs of segmenting video lectures. Section 3.4 is dedicated to an overview of the Virtual Whiteboard and Virtual “Ink” Technologies, and their impact on the SMA program. Discussion of the plug-in itself is deferred to chapter 4.
3.1 Current Segmentation of Lecture Videos

Chapter 1 has established the motivation and need to segment lecture videos: students need to locate a particular topic discussed by the lecturer without having to watch the entire lecture again. They should be presented with links corresponding to the underlying structure of the lecture so that they have immediate random access to the portions of the lecture corresponding to the topic they’re looking for.

Converting the lectures to useful asynchronous resources would therefore involve two steps:

- Segmenting the lecture according to the topics it contains and performing the appropriate linking to the streaming video.
- Processing and transferring the lecture video to a website.

The current process of converting the live classroom experience into a randomly accessible asynchronous resource is labor intensive. This is because:

- Segmenting the lecture requires a skilled individual to watch the master recording and manually insert the cuts corresponding to a topic change in the lecture.
- Transforming the lecture into a series of static images, transferring it to a website and constructing the links is a tedious and time-consuming job.

Because of the quantity of lectures produced by the SMA program and the need for immediate availability of the lecture information, we need to automate a considerable part of the labor to yield a lower cost per lecture and a faster turnaround. The next section introduces the Virage’s VideoLogger, an “off the shelf” automated segmentation system that we evaluated as a first step in automating the segmentation of lecture videos.
3.2 The Virage VideoLogger

Virage is a provider of video and rich media communication software. The VideoLogger is Virage’s cataloging product that enables extraction of video content. By utilizing media-analysis algorithms that watch, listen to, and read a video stream in real time, the Virage VideoLogger automatically creates a frame-accurate index that can be accessed randomly.

3.2.1 Technical overview

This technical overview is based on the information presented in [10]. As the VideoLogger software ingests a video, one set of algorithms slices the video into logical segments based on changes in the visual content. The VideoLogger extracts distinctive thumbnail images, known as keyframes, that are time coded to represent each segment, resulting in a frame-accurate storyboard.

Separate sets of algorithms profile the audio track, perform speech-to-text transcription, and handle speaker identification. When embedded textual information, such as close captioning or teletext, is available, the VideoLogger also extracts and time-codes the text to form a searchable index. Optionally, one can supplement this information by marking and annotating clips using the keyframes in the storyboard as in and out points in the media stream.

At the same time that it indexes the video, the VideoLogger can control the digitization and encoding of one or more streamable versions of the original content. Synchronized encoding and indexing allows one to navigate through the video using the index, then to “click” on the extracted keyframes and play the digitized version from that point. Some of the key design points of the VideoLogger system are:
Separation of content and metadata: Metadata is the pertinent data about the video data (content), and serves to index and provide access to the video content. Any feature or information that is extracted from the raw video data, such as speaker identification or keyframes, is stored as metadata. Using this metadata layer, most of the extracted features have an identical interface, namely an “in-time” and an “out-time” which enables search and browse activities to infer temporal relationships between extracted features. Figure 3.1 shows Virage’s VideoLogger. A keyword spotting algorithm, part of the Virage’s AudioLogger, is employed to provide keyword spotting when the audio signal contains speech.

Figure 3.1 Virage’s VideoLogger (http://www.virage.com)
Every entry in the lower right window (Title: Speech) is a metadata having an “in” and “out” time, and a description of the corresponding data (in this case the detected keywords). Metadata form the basis of managing the extracted media indices. The separation between metadata and video data offers modularity with respect to digital content and allows the VideoLogger to integrate with a wide variety of formats and technologies.

*Open Architecture:* Virage provides a set of developer toolkits to allow easy customization and integration. These toolkits provide a number of API’s and plug-in interfaces for use by software developers to integrate the metadata capture process with other media management activities. These toolkits are grouped under the Virage VideoLogger SDK, which will be discussed later in this chapter.

### 3.2.2 System Architecture

Figure 3.2 depicts a high level schematic overview of the VideoLogger cataloging environment. The diagram is mainly conceptual, and the boxes do not necessarily map onto hardware or software processes.
Video comes into the system from either analog or digital sources, and is ingested by the Virage VideoLogger, and optionally by the Virage AudioLogger. Simultaneously, it can be encoded to some form of digital content. Metadata can emerge from the VideoLogger in a variety of forms. Presentation can be as simple as HTML files browsable through a search engine mechanism, or as complex as a DBMS-based media asset management system. The open architecture allows integrators to configure various combinations of hardware, databases, search engines, and digital video encoding to meet specific needs.
3.2.3 Virage Data Model

The VideoLogger software watches, listens, reads and allows annotation of video. The most important feature extraction engines are the following:

**VideoLogger:** The VideoLogger analysis engine creates a storyboard that visually depicts the content of the video. The Virage keyframing engine was introduced in chapter 2. As mentioned previously, the keyframing engine analysis operates on color space, on shape and edges, and on the motion of visual elements.

**AudioLogger:** The Virage media cataloging system includes an optional AudioLogger, which can be used in combination with the VideoLogger. The AudioLogger provides an audio engine that performs signal analysis and processing of the audio signal associated with the video. The AudioLogger extracts three distinct types of audio metadata:

- Audio Classification: The audio engine classifies the audio signal into several predefined categories such as Speech, Silence, Music, etc.
- Speech to Text: Real time, speaker-independent continuous speech identification converts audio signal into an index of the spoken words.
- Speaker ID: The AudioLogger uses voice characteristics to identify individual speakers.

**Video Reader:** When the VideoLogger application is combined with decoding equipment capable of extracting encoded text such as close captioning or teletext, the VideoLogger places the text in a metadata track and time-stamps it.
3.2.4 Integrating the VideoLogger in SMA

The deployment of the VideoLogger into the current process would be beneficial on several levels:

- It would help automate the synchronization of the video lecture with the PowerPoint presentation by capturing the lecturer’s timing of the slideshow exactly as presented in class. This can be achieved by using a plug-in that Virage developed; it helps integrating streaming video with slides. Once a set of timings is determined, the process of integrating the timings into the streaming presentation, once a tedious task and a source of human error, is now an automated procedure.

- The VideoLogger would greatly simplify the process of transforming the lecture presentation into a series of static images, transporting them to a web site, and performing the appropriate linking to the streaming video.

The internal segmentation engine of the VideoLogger, however, does not address the issue of segmenting lecture videos. This internal segmentation engine works best when a topic or scene change comes with distinctive video and audio cues as occur in TV newscasts. Usually, in TV news, when a topic or story change occurs, there is a dramatic change in the setting, where the camera might move from the studio to a reporter on the field, or from one person to another. Moreover, reporters tend to introduce themselves and end their stories by using specific keywords. This is not the case for classroom lectures. Lecturers are much less dramatic and predictable when giving their presentations. Although the VideoLogger would solve a good deal in automating the synchronization and web-publishing of video lectures, there is still a need to develop an analysis engine that detects topic changes in a video lecture. As mentioned previously,
this analysis engine should extract features that are more relevant to classroom dynamics than visual and audio cues. The VideoLogger’s ability to be adapted to different media streams allowed us to develop a plug-in that looks at the teacher’s activity on the whiteboard. The plug-in is based on two key technologies:

- The Virage VideoLogger SDK, which exposes the VideoLogger APIs and plug-in interfaces.
- Virtual “Ink” Technologies, which are used to detect and monitor the writing on the whiteboard.

### 3.3 VideoLogger Software Development Kit (SDK)

The VideoLogger application contains a group of APIs and plug-in interfaces for extensibility and customization by developers. The VideoLogger SDK exposes these interfaces (figure 3.3). The SDK is packaged as a dynamically linkable library of “C” entry points that provide access to the metadata and extensibility of the VideoLogger [11, 32, 33].

The Remote API is especially useful for our purposes. The VideoLogger Remote API serves two main functions:

- Remote Control: Provides a remote control mechanism for unattended logging activities, such as driving the VideoLogger from a centralized scheduling application.
- Media Analysis: Provides a programmatic metadata input mechanism to allow external processes to perform independent analysis on visual, audio and other sources of information, and to feed that metadata into the VideoLogger. The Virage AudioLogger cooperates with the VideoLogger in this fashion.
3.4 Virtual Whiteboard and Virtual “Ink” Technologies

3.4.1 Overview

Virtual Ink has created a product called Mimio; it electronically captures handwriting and drawing on a standard whiteboard using a combination of ultrasound/infrared capture bars attached to the side of the board. Special pressure sensitive pen cases hold standard whiteboard pens. The software can distinguish the holders, and therefore can match the pen colors to the software display. A special tool permits erasure.

The Mimio application captures the pen-motions and displays them on the PC connected to the capture bar. This display can then be forwarded across the Internet using the standard application sharing mechanism of NetMeeting [15].

Virtual Ink has created a “virtual ink” file format that can be synchronized with audio using the BoardCast software. In effect, BoardCast captures a time-coded stream of
vector-based pen motion in an “ink” file and audio is recorded in a standard Real Audio file. The two are synchronized using a short SMIL\textsuperscript{1} script. BoardCast turns the pen motions into a streaming media type compatible with all the other Real streaming media types. The Real Player G2 will automatically download the Virtual Ink plug-in from the Real.com site the first time it encounters an ink file [15].

3.4.2 Integrating Virtual Ink in SMA

Integrating Virtual Ink’s Mimio and BoardCast technologies in SMA lectures had the following advantages:

- With a virtual whiteboard, the lecturer writes and draws on a full size whiteboard for the local class. The BoardCast is synchronously captured and transmitted to the distant classroom using NetMeeting. The streaming ink can later be synchronized with the video of the class and presented as adjacent screens on a PC.

- Synchronizing handwriting with audio is an expressive way to deliver technical presentations.

- BoardCast sessions are small enough to be efficiently stored or transferred over a network.

- There is an experimental plug-in that enables recording BoardCast files from mouse devices, such as WACOM tablets (http://www.wacom.com). Using this plug-in, BoardCast files can be created from any PC equipped with a WACOM tablet.

Moreover, having the whiteboard handwriting captured electronically allowed us to develop our plug-in that takes a BoardCast stream and extracts time stamped pen

\textsuperscript{1} Synchronized Multimedia Integration Language is a markup language that enables Web developers to divide multimedia content into separate files and streams (audio, video, text, and images), send them to a user's computer individually, and then have them displayed together as if they were a single multimedia stream. It defines the commands that specify whether the various multimedia components should be played together or in sequence.
motions. The plug-in extracts board events, and using the Virage Media Analysis Plug-in Framework, feeds this new type of metadata into the VideoLogger. Because metadata is distinct and separate from content, and because the analysis plug-in within the VideoLogger is running in its own computational thread, the VideoLogger can still perform video and audio analysis on the lecture video. The plug-in is effectively performing a feature extraction on a new stream of data to improve the interpretation of the video/audio stream. What features are being extracted and how are they interpreted? The next chapter will address these two questions.
Chapter 4

The Lecture Video Plug-in

This chapter is devoted to the design and development of the plug-in. Section 4.1 presents an overview of the plug-in and discusses the general architecture of the system. Section 4.2 and 4.3 list the different features that the plug-in can extract and customize. Section 4.4 goes through the actual implementation of the plug-in by first presenting the BoardCast format, then by explaining the feature extraction mechanism, before showing how the Media Analysis Plug-in interface was used to establish a connection with the VideoLogger. Section 4.5 concludes the chapter by discussing the results and limitations of the plug-in.

4.1 Overview

The video lecture plug-in was developed using C++ and Virage’s Media Analysis Plug-in Framework (VPF). It is a separate analysis engine that runs in its own thread within the VideoLogger. It takes a BoardCast file (which is synchronized with the video lecture), identifies specific board events, and commands the VideoLogger to keyframe the video stream at the marked time stamps. Using the video segmentation framework we have
described in chapter 2, the BoardCast file acts as the raw data while the plug-in operates as a feature extraction system.

Figure 4.1 shows a general overview of the plug-in architecture. The diagram is mainly conceptual, and the boxes do not necessarily map to software modules and processes.

The figure shows two inputs to the system: a BoardCast file, which is analyzed by the plug-in, and a recorded lecture video, which is streamed by the VideoLogger. Note that the analysis of the BoardCast file occurs before the lecture video is ingested in the VideoLogger.

The first step that the plug-in takes is to translate the BoardCast file, which is in binary format, back into a time-coded stream of vector-based pen motion, or more concretely, to
extract the coordinates of the pen motions vs. time. The board event detector matches observed pen motion to patterns defined by a set of cues (e.g. sudden drop in the y coordinate), and saves the time code whenever the requirements of one of the predefined cues are satisfied.

Afterwards, the VideoLogger streams the lecture video that is synchronized with the BoardCast file. The VPF gives the plug-in access to the VideoLogger timecode track. Whenever the VideoLogger’s timecode corresponds to the timestamp of an already detected board event, the plug-in sends a command to the VideoLogger to mark the time and create a keyframe in the corresponding video lecture.

Because the plug-in detects all the board events and all the cues contained in the BoardCast before the lecture video starts playing, the plug-in can look ahead in the pen-motion stream, and in certain cases, command the VideoLogger to keyframe the lecture video before the board event occurs.

4.2 Extracted Features

The plug-in detects several types of board events. It inserts a keyframe whenever one or a combination of the following board activities is detected:

- Clear Board: corresponds to a new screen where the lecturer resets the virtual board.
- Erasure: corresponds to the lecturer using the eraser on the board.
- Writing Stops: occurs when the lecturer stops writing on the board for a given time delay.
- Writing Resumes: occurs when the lecturer starts writing after a given time delay.
- Dividing line: occurs when the lecturer draws a long horizontal or vertical line on the board.
- New Line: occurs when a vertical drop greater than a given value separates two distinctive pen strokes.

There are many other cues that can potentially mark a change in the topic. One example is a change in the letter size to indicate a title. Because of the prototype nature of this plug-in, cues that were easy to implement with a simple algorithm were selected.

Figure 4.2 shows a view of the VideoLogger with the plug-in running in the background.
The figure shows a presentation that demonstrates the plug-in capabilities. The video stream that has been synchronized with the BoardCast recording is shown in the upper left window (title: Monitor). The tabs in lower right window (title: Topic Change) show the different plug-ins that are available (e.g. Closed Caption, Audio, Speech, etc.). All of these media analysis modules come integrated with the VideoLogger application, except for the selected one, Topic Change, which is the plug-in that I have developed. The window shows the different board events that have been detected so far. Each entry in the topic change list gives the time and type of the detected board activity. Every entry has a corresponding keyframe (the lower left window shows the inserted keyframes; title: Keyframes). For instance, the keyframe with a time stamp of “1:36:15” corresponds to the “New line” event highlighted in the list.

4.3 Customization

The fact that different lecturers have different styles of writing on the board complicates the task of the plug-in. For instance, some lecturers leave a lot of space between lines; others write in a more compact way; this makes it difficult for the plug-in to determine when it should mark a new line. To mitigate this problem, a configuration panel enables the user to fine-tune the detection cues.

Figure 4.3 shows the configuration panel. The top line indicates the BoardCast file the plug-in is set to analyze. The “Options“ frame, allows the user to customize the detection of the different board activities. The user can deactivate the detection of any of the cues by unchecking the corresponding option.
Some of the board events can be fine-tuned:

- **Dividing Line**: is detected as a user defined percentage of the total board width or height (e.g. 70%).

- **Writing Stops**: is detected as a time delay when the lecturer actually stops writing. If no writing has been detected for a period of time greater than the time delay specified on the configuration panel, a board event is detected. As previously mentioned, the plug-in analyzes the entire BoardCast file and detects all the board events before the VideoLogger starts ingesting the corresponding video. If the plug-in didn’t have the ability to look ahead in the BoardCast stream, the only way to determine a time delay during which there was no writing would be to wait until the lecturer started writing.
again, and to mark the video at that point, i.e. to mark the resumption of writing rather than the pause in writing. By preprocessing the BoardCast stream, the plug-in can determine the time delays in advance, giving it the ability to mark a “Writing Stops” the moment the lecturer stops writing. It is useful to note that, in case a topic transition was associated with a pause in writing, the lecturer usually continues talking about the topic for a while after the writing has stopped. It is, therefore, more interesting to mark and keyframe the video a certain amount of time after the “Writing Stops” is detected. The configuration panel allows the user to specify this time delay.

- **Writing Resumes:** is detected when a lecturer resumes writing after having stopped for a given time delay. This board activity works in a similar fashion as the “Writing Stops” where the ability of the plug-in to look ahead in the BoardCast file allows it to mark the video a given time delay before the writing actually begins. The rationale behind this is similar to the “Writing Stops”; the lecturer is expected to start talking about a new topic a certain amount of time before she starts writing on the board.

- **New Line:** The plug-in analyzes the BoardCast file for vertical drops in the coordinates. If the vertical drop exceeds a certain limit (user specified as a percentage of the whiteboard height), a new line is detected. The difference with a vertical “Dividing Line” is that a “New Line” is associated with two distinctive pen strokes separated by a vertical drop in the coordinates. A vertical “Dividing Line” on the other hand is a single pen stroke. Sometimes, the plug-in issues multiple “New Line” events in a small time frame. This usually happens when the lecturer draws a figure on the white board, or writes an itemized list such as an outline. Although we would like to detect the start of the figure drawing and the beginning of the itemized list
(these are strong cues for a new topic), we don’t need to detect the rapid “New Lines” events. Using the configuration panel, the user can specify the minimum delay required for “New Line” events to be detected. For instance, if this delay is set to five seconds, the plug-in will not mark a “New Line” unless it’s five seconds away from the previous one.

Being able to customize the board event detection allows users to:

- Find the right settings for the plug-in that will give a good feature extraction performance regardless of the lecturer giving the presentation. This should be done after we have accumulated a large virtual whiteboard archive so that we can select a sample of lectures and perform extensive testing to determine the ideal settings for the plug-in.
- Customize the plug-in for the style of a lecturer. This is a good solution in case there is no training data, or in case some teachers have a particular way of lecturing which will always require customization.

### 4.4 Implementation

#### 4.4.1 Data File

Given the segmentation framework we have defined in chapter 2, the data layer in the case of this research work consists of the BoardCast file, which is a proprietary binary message format. Referring back to figure 1.1, the data layer holds three properties: attributes, format and raw data. The mapping of these properties to the BoardCast data file is as follows:

- **Format** is the format or extension of the file itself i.e. “ink” (.mbc) file.
- **Attributes** are the date, message size, version, and board dimensions stored in the file.
- **Raw data** is the pen motions and other activities stored in binary form.

Virtual Ink provided us with enough documentation to reverse the binary code into pen motions. Pen motions are divided into pen strokes. The key attributes of pen strokes are:
- **Time stamp**: which gives the time when the pen stroke took place.
- **Pen Id**: which defines the type of pen (different colors) or eraser is being used.
- **Data points and number of data points**: which translates, after some post-processing, into coordinates.

Other types of information, such as board clears, can be located by detecting special messages contained within the BoardCast file.

### 4.4.2 Feature Extraction: The plug-in data model

A simplified class diagram for the plug-in is presented in figure 4.4.
All the board events inherit from the parent abstract class `BoardEvent`. All the board events have a common property, the time stamp, and a static method `detect()`. Because it is defined as an abstract method in the parent class, `detect()` is implemented in each of the `BoardEvent` subclasses. Its function is to locate the corresponding board event based on the BoardCast file.

![Diagram](image)

**Figure 4.5 Plug-in Processes**

Figure 4.5 shows the different processes that the plug-in performs before connecting to the VideoLogger. The first step is to parse the BoardCast file into units such as pen...
strokes and clear boards. The `detect()` method of every `BoardEvent` class is sequentially applied to the parsed BoardCast. If the `detect()` method of a particular class detects an event, a corresponding new `BoardEvent` object is added to a linked list. The objects in the list are sorted according to their time stamp.

### 4.4.3 Connecting to the VideoLogger

The Virage Media Analysis Plug-in Framework is structured as a set of C++ files defining entry points to the plug-in. Those entry points expose the Virage plug-in APIs as dynamically linkable entry points to the plug-in DLL. The Developer’s plug-in should be derived from the Virage Media Analysis Plug-in class and should implement all the entry points. A detailed description on the behavior of the different entry points is included in [33].

In addition to the classes shown in figure 4.4, the analysis plug-in developed in this research has three C++ classes:

- A class that functions as a wrapper to the C++ interface and takes care of basic functionality like returning the name of the plug-in.
- A class that takes care of the analysis itself (section 4.4.2) and gets passed around as the plug-in instance.
- A class for the configuration dialog (section in 4.3).

The plug-in starts by analyzing the BoardCast stream. The analysis is performed when the user chooses the BoardCast file in the configuration panel and clicks on the “OK” button (Figure 4.3).
The plug-in then launches the thread that is needed during the analysis session. The VideoLogger uses the VPF interface to pass the current time stamp of the lecture video. The plug-in compares this time stamp with the first element in the BoardEvent linked list. When the VideoLogger's time stamp is equal to the time stamp of the first element (with a specified tolerance), the plug-in issues a command to the VideoLogger to keyframe the video at the current time and generates a new metadata entry. The plug-in also specifies the caption to be included in the metadata (see Figure 4.2). The caption depends on the type of the BoardEvent element being examined. Finally, this element is removed from the list.

4.5 Results and Limitations

The plug-in was successful in combining two very promising technologies: Virage's VideoLogger and Virtual Ink's Mimio and BoardCast. By combining these technologies, the plug-in was able to achieve its main objective of extracting features relevant to the classroom activities. The results were predictable: the plug-in systematically marks the video lecture every time it detects a board event.

On the other hand, using the plug-in directly as a content extraction system does not yield good results. Although it clearly detects all the cues, a board event doesn’t necessarily correspond to a topic transition. Not every pause in a lecturer’s writing or every jump to a new line implies that she’s moving to a new topic. The plug-in in this respect oversegments the video lecture. It might be easier now to go over all the keyframes that the plug-in has detected and decide which ones to keep, as opposed of reviewing the entire lecture, but it would be more effective to have the plug-in accomplish its dedicated task of feature extraction and then construct a separate content extraction system based on the
combination of different features, namely BoardCast, video and audio, to decide when a
topic transition occurs. As future work we need to develop and evaluate artificial
intelligent based segmentation techniques and customize them to lecture video
segmentation.

Finally the current implementation of the plug-in is heavily dependent on Virtual Ink’s
technologies. It only works for video lectures recorded with a corresponding
synchronized BoardCast file. However, the layered architecture of the plug-in allows it to
be easily adapted to other virtual whiteboard products. A standard internal time stamped
vector format was established. The detect() methods rely on this intermediate layer to
extract board events. Adapting any whiteboard product will therefore be reduced to
mapping its time stamped pen motions to the plug-in’s internal time stamped vector
format. The mapping can be achieved given that the proper documentation for parsing the
virtual whiteboard format is provided.
Chapter 5

Conclusion

The plug-in was successful in achieving its goal of a feature extraction system. However, this is only the first step towards the larger goal of intelligently segmenting video lectures. Future work should focus on improving the effectiveness of board event features and on developing and evaluating artificial intelligence based segmentation techniques and customizing them to segment lecture video.

5.1 Improving Feature Detection Effectiveness

Now that a board event detector has been implemented, we need to study thoroughly which board events are the most correlated with topic transitions. With the existing features that the plug-in can detect we need to:

- Evaluate the correlation of each (or the combination) of these features to a topic change.

- Improve the effectiveness of certain features. Taking the example of erasures, the investigation might show that detecting partial erasures on the white board has a lower correlation with topic changes than erasing the entire board. Partially erasing
the board usually means that the lecturer simply needs more space to write on, while
erasing the entire board might strongly suggest that the lecturer is passing to a new
topic. Taking this a step further, differentiating whether the erasure occurred while
the white board was completely or partially full might be even more useful.

New features that might play an important role in segmenting a lecture video need to be
investigated (e.g. observing presentation slides). A further study needs to be conducted
on the usefulness of audio and video cues when combined with board event features.

5.2 Constructing Segmentation Models

MITRE’s Broadcast News Navigator (BNN) is perhaps the most similar system to the
lecture video segmentation system this research is interested in. The BNN system is
heavily tailored towards a specific news format, namely CNN prime time [5].

The BNN system exploits the temporal structure of the show’s format, using knowledge
about time, such as the fact that a particular CNN logo is displayed at the start of the
broadcast. The system also makes extensive use of phrase templates to detect segments.
For example, using the knowledge that, in this show, news anchors introduce themselves
at the beginning and the end of the broadcast, the system tries to detect phrase templates
such as “Hello and welcome”, “I’m <person-name>”, which signal the introduction to the
show. The objective of BNN is to use multi-stream analysis of such features as speaker
change detection, scene changes, appearance of music and so forth to achieve reliable and
robust story segmentation. The BNN system also aims to provide a deeper level of
understanding of story content than is provided through simple full text search, by
extracting and identifying, for example, all the named entities (persons, places and locations) in a story [6].

The MITRE Corporation has developed two models for their system: the “Finite State Machine”, and the “Learned Hidden Markov Models” [5], [6], [22]. Those same two models might be good candidates to build the lecture segmentation model.

**Finite State Machine (FSM):**

This technique is a multi-source technique that can be used to correlate the video, audio and BoardCast features to detect when a topic segment occurs. The entire lecture needs to be divided into a series of “states” such as “start of lecture”, “student interaction”, “new topic”, “questions”, “topic introduction”, “topic conclusion”, and “end of lecture”. The multi-source cues, including time, are then used to detect when a state transition occurred. The state boundaries are detected through the correlation of the features. The transitions of the FSM are usually represented in a relational database where each record stores the start state, the end state and the feature that induced this transition. A topic segment is detected each time there is a transition to a “new topic” state [6].

**Learned Hidden Markov Models (HMM)**

HMM are used to model systems where what we wish to predict is not what we observe, i.e. the underlying system is hidden. In our example, the observed sequence would be the extracted features, and the hidden system would be the lecture states (“Start of lecture”, “new topic”, etc.). A HMM structure is presented in three pieces [22]:
Visible “observations” or features such as a combination of audio, video or BoardCast cues;

Hidden states that are abstract and not directly observable such as “start of lecture”, “new topic”, etc;

A set of three probability vectors that determine how the model is supposed to behave:

- A = transition probability matrix = \( a_{ij} = P(q_j \text{ at } t \mid q_i \text{ at } t-1) \): The probability the model is in state \( q_j \) in an interval of time, given that it was in \( q_i \) in the previous interval.

- B = observation probability matrix = \( b_j(k) = P(O_k \mid q_j) \): The probability that observation \( O_k \) was witnessed in an interval of time, given that the model is in state \( q_j \).

- \( \Pi \) = initial probability matrix = \( \pi_i = P(q_i \text{ at } t=0) \): The probability the model is in state \( q_i \) at the first time interval [7].

There are three basic problems of interest that must be solved for the model to be useful [38]:

The evaluation problem: given a model and a sequence of observations, how do we compute the probability that the model produced the observed sequence. In other words, how well does a given model match a given observation sequence. This is particularly useful in case we are trying to choose among several competing models. Consider the situation in which we have a number of HMMs describing different systems, and a sequence of observations. We may want to know which HMM most probably generated the given sequence.
- **The learning problem:** which consists of taking a sequence of observations (from a known set), known to represent a set of hidden states, and fitting it to the most probable HMM; that is, determine the \((\Pi, A, B)\) triple that most probably describes what is observed. The probability vectors have to be calculated from the analysis of training data. A set of algorithms should be developed to keep count of all state transitions that occurred, and to keep track of observations and the states in which they were witnessed. Using this machine learning technique, the state transition probabilities and the observation probabilities in a given state can be estimated. Note that we might need to estimate several models and probability vectors to cope with the differences in lecturing styles between teachers, subjects (e.g. Calculus vs. American History), or even schools (e.g. Engineering vs. Humanities). The probability vectors need to be estimated based on an appropriate set of training data collected by teacher, subject or school.

- **The decoding problem:** another related problem, and the one usually of most interest, is to find the hidden states that generated the observed output. In many cases we are interested in the hidden states of the model since they represent something of value that is not directly observable. In practical situations, there is no “correct” state sequence but an optimal one. Once the HMM has been trained, segmenting the video into its states (i.e. start of the lecture, topic transition, etc.) is performed using the Viterbi algorithm, a standard technique for segmentation and recognition with HMMs [22]. Given a sequence of features, the Viterbi algorithm produces the sequence of states most likely to have generated these features. The state sequence is time-aligned with the feature sequence, so that the video is segmented according to the times corresponding to a “new topic” state.
Similarly to the FSM model, the HMM uses a set of states to describe the different sections of a lecture video. Conversely, the detection of topic segments in the FSM model is based on non-adaptable, manually created transitions between these states, while HMM uses machine learning to optimize the estimates of its transition vectors. A study in [6] compares the performance of FSM and HMM when used to segment the “CNN Prime News” program. The HMM and FSM gave comparable accuracy in segmenting broadcast videos, with the FSM performing slightly better. On the other hand, the average time to create the FSMs is much longer than the average time to create training data and learn probabilities for HMMs.
Bibliography


