Characterizing Audio Events for Video Soundtrack Analysis

Courtenay V. Cotton

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2013

ABSTRACT

Characterizing Audio Events for Video Soundtrack Analysis

Courtenay V. Cotton

There is an entire emerging ecosystem of amateur video recordings on the internet today, in addition to the abundance of more professionally produced content. The ability to automatically scan and evaluate the content of these recordings would be very useful for search and indexing, especially as amateur content tends to be more poorly labeled and tagged than professional content. Although the visual content is often considered to be of primary importance, the audio modality contains rich information which may be very helpful in the context of video search and understanding. Any technology that could help to interpret video soundtrack data would also be applicable in a number of other scenarios, such as mobile device audio awareness, surveillance, and robotics. In this thesis we approach the problem of extracting information from these kinds of unconstrained audio recordings. Specifically we focus on techniques for characterizing discrete audio events within the soundtrack (e.g. a dog bark or door slam), since we expect events to be particularly informative about content. Our task is made more complicated by the extremely variable recording quality and noise present in this type of audio.

Initially we explore the idea of using the matching pursuit algorithm to decompose and isolate components of audio events. Using these components we develop an approach for non-exact (approximate) fingerprinting as a way to search audio data for similar recurring events. We demonstrate a proof of concept for this idea.

Subsequently we extend the use of matching pursuit to build an actual audio fingerprinting system, with the goal of identifying simultaneously recorded amateur videos (i.e. videos taken in the same place at the same time by different people, which contain overlapping audio). Automatic discovery of these simultaneous recordings is one particularly interesting facet of general video indexing. We evaluate this fingerprinting system on a database of 733 internet videos.

Next we return to searching for features to directly characterize soundtrack events. We develop a system to detect transient sounds and represent audio clips as a histogram of the transients it contains. We use this representation for video classification over a database of 1873 internet videos. When we combine these features with a spectral feature baseline system we achieve a relative improvement of 7.5% in mean average precision over the baseline.

In another attempt to devise features to better describe and compare events, we investigate decomposing audio using a convolutional form of non-negative matrix factorization, resulting in event-like spectro-temporal patches. We use the resulting representation to build an event detection system that is more robust to additive noise than a comparative baseline system.

Lastly we investigate a promising feature representation that has been used by others previously to describe event-like sound effect clips. These features derive from an auditory model and are meant to capture fine time structure in sound events. We compare these features and a related but simpler feature set on the task of video classification over 9317 internet videos. We find that combinations of these features with baseline spectral features produce a significant improvement in mean average precision over the baseline.

Table of Contents

List of Figures

List of Tables

Acknowledgments

Most importantly I would like to sincerely thank my advisor Dan Ellis for all of his guidance and support throughout this process. He has taught me many invaluable lessons about how to approach research problems and how to ask interesting questions.

I would also like to thank Professors Shih-Fu Chang, John Kender, Xiaodong Wang and Dr. John Smith for serving on my thesis committee and for their extremely helpful feedback on this work.

I have to thank all the current and former members of LabROSA for their friendship and support, as well as their help with research ideas and of course lab beer nights.

I could never have achieved this goal without the support of my parents, Chase and Valentine, and my siblings Kirsten, Jordan, and Chase. Thank you for everything.

And to all of my friends, thank you. I couldn't have done it without you.

Chapter 1

Introduction

The human ability to easily understand and interpret the audio environment that they find themself in is a skill that has not yet been replicated by computers. There are many components to this understanding. The ability to isolate and interpret speech is one that has attracted much attention from the research community, for obvious reasons. However, there are numerous other facets to this understanding, such as identifying instances of recognizable audio events and general awareness of environment from audio characteristics. Specifically, the ability to easily identify instances of similar audio events that can be highly informative about the content and context of an audio signal is still an unsolved problem. There is no generally well-established technique or set of techniques to automate this type of audio event identification. In this domain, there is no equivalent to the off-the-shelf speech recognition systems currently commercially available. The unifying theme of this work is to explore ways in which events in ambient audio signals may be represented so as to make the process of comparing and interpreting them easier to automate.

Largely, we have been focused on the idea of discovering discrete events in unconstrained audio recordings, such as those found in the soundtracks of consumer and internet video recordings. We have been motivated by the desire to describe and automatically identify events. When we discuss audio events, we generally mean the types of sound events that humans find easy to distinguish as an isolated and recognizable occurrence, anything from a dog bark to a car alarm sound. Relatedly, we could consider the concept of audio textures, which might be made up of one or more events. But here we have largely conceived of the problem as one of identifying discrete events, which may or may not make up a larger texture. This is in contrast to a standard audio modeling approach,

which would typically treat this type of unconstrained audio signal as a bag of audio frames, rather than trying to isolate specific events in it. This work is an exploration of a number of related approaches to the problem of characterizing these type of audio events. In it we cover two related problems in dealing with audio events: the easier task of identifying exact replicas of certain events with an audio fingerprinting technique and the more difficult task of quantifying similarity between different instances of the same type of event.

The concept of an 'event' as we use it here has the potential to be somewhat ambiguous. For example, the dog's bark mentioned above could be considered an 'event'. But this bark could conceivably consist of a single syllable ('woof') or multiple syllables ('woof woof woof'), either of which might be considered a single event by a human listener. We do not go out of our way to be strict in our definition of an 'event' in this sense. In chapter 5 we describe a system for finding events based on transient detection. In this case, each distinct syllable as mentioned above would be sufficient on its own to trigger a detected 'event'. However, in other algorithms presented here, for example the non-negative matrix factorization (NMF) of chapter 6, a bark event could be learned to be the series of syllables, provided a similar sequence occurs frequently enough as a cohesive unit in the training data. We generally tend to be agnostic to the particular form that a learned 'event' may take, and instead allow data-driven methods to find a reasonable level at which to consider something a unit.

At a higher level, our concern with locating 'events' is primarily for the purpose of defining a local unit (longer than a 20-30 ms feature frame) that also has some inherent semantic meaning. The larger goal is to identify local units that correlate well with semantic concepts that can be applied to the video as a whole, for example 'animal' in the case of a dog bark. We attempt to understand the content of a video's soundtrack by identifying smaller components within it that are more indicative of the content than the surrounding audio background. In this sense, it is somewhat irrelevant how precisely an event is defined, as long as perceptually similar short units can be compared and identified as similar across multiple videos.

We have a number of motivations for wanting to describe audio content in this way. The most prominent application in our work has been the classification and tagging of video content. We believe this kind of representation is important to understanding the audio content of videos, because it mirrors how people get context clues from identifying individual audio events. Because

video content analysis is one of our main application interests, and because we have appropriate ground truth data, we have used the task of video classification to evalute our algorithms in many of the experiments described here. Regardless, the ability to characterize an audio environment by the types of audio events it contains has many potential applications. A few of these are multimedia database searching and indexing, surveillance, mobile audio awareness applications, and automated video narration. Some of these applications will be disscussed further in chapter 2. Key to note is that most of these applications require working with data that is often extremely noisy, cluttered, and processed with a wide range of different recording equipment filters.

The presence of noise inherent in amateur audio recordings is important to understanding the problems we face. For illustration, figure 1.1 displays spectrograms of 10 second segments from two different video soundtracks. The first is a clip from a professional news organization recording taken during the 2009 U.S. presidential inauguration; it was collected from YouTube as part of the dataset we discuss in chapter 4. The second is an amateur consumer video taken by a parent of a child playing at a playground; it was also collected from YouTube as part of the Columbia Consumer Video dataset we discuss in chapter 7. Just a quick look at these spectrograms demonstrates the nature of the problem we face working with amateur recordings. Both are displayed on the same intensity scale; the amateur recording clearly suffers from higher levels of background noise and has far less dynamic range. The amateur video also demonstrates a fairly common property of these recordings, a sharp cutoff filter around 6 kHz; different recording equipment has different cutoffs, which can exacerbate the problem of comparing audio features among amateur recordings. Listening to each clip reveals other differences; the amateur video contains constant high level environmental noise, probably wind noise, which is mostly responsible for the noise seen on the spectrogram. Additionally, there are intermittant noises which probably arise from the camera operator shifting the handheld equipment. There is the presence of the camera operator's voice, which is much louder than other voices in the scene. In contrast, the professionally produced video does not suffer from the camera operator's voice or other interferences. Note that both videos were recorded in an uncontrolled outdoor environment; professional videos recorded in an indoor or studio setting would contain even less ambient background noise than the professional video displayed here. All of these factors contribute to the extremely challenging problem of handling and characterizing these types of amateur video recordings.

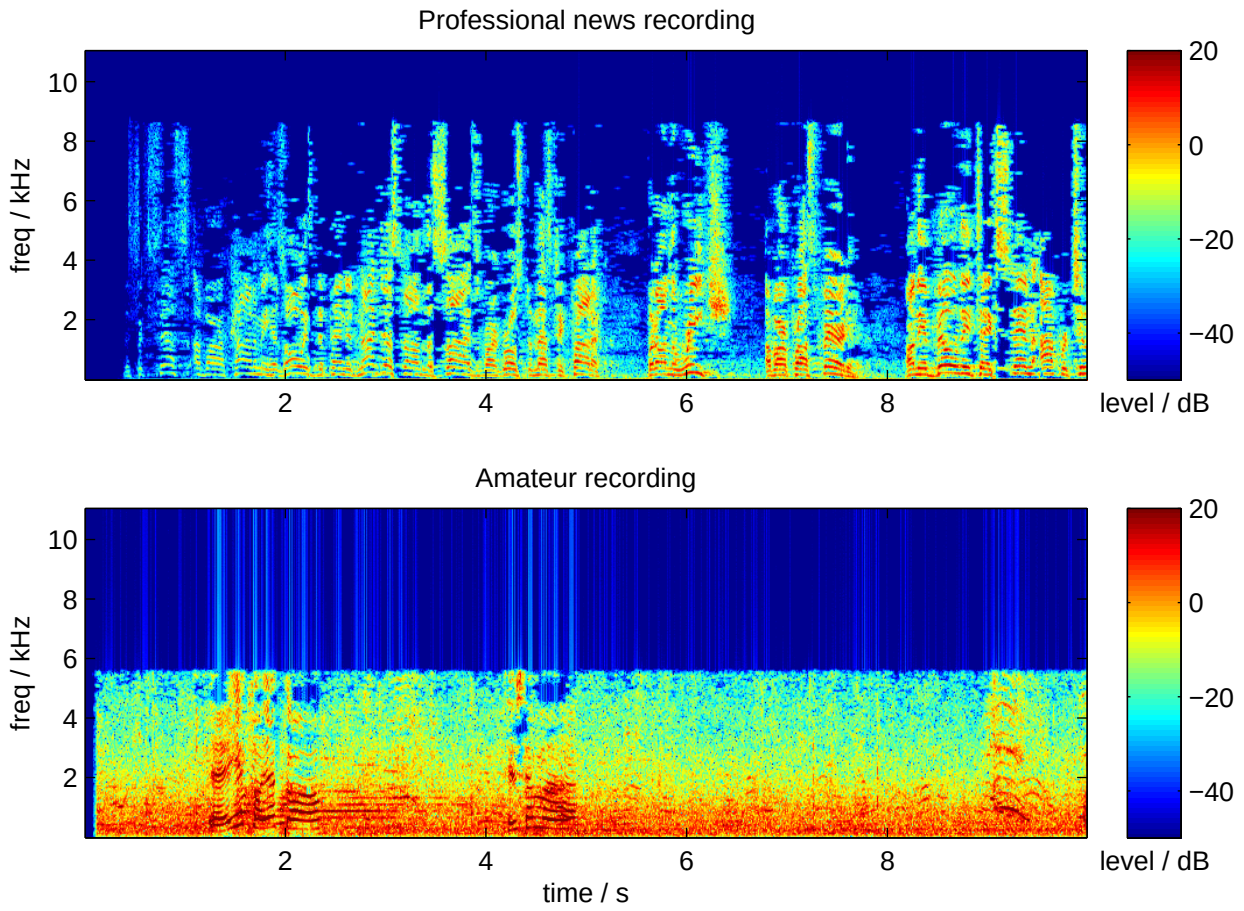

Figure 1.1: Comparison of spectrograms of professional and amateur recordings. Top: professional news organization recording; Bottom: amateur consumer recording.

1.1 Contributions

An open question is how to best represent this type of event content in ways that will be most useful for the types of detection and classification tasks we are interested in here. Generally the problem form suggests the idea of a sparse, part-based representation of some sort. To this end we have explored a number of algorithms for performing sparse signal decomposition, among them: matching pursuit, transient onset detection, and non-negative matrix factorization. Ultimately, we explore an algorithm inspired by the auditory model, which demonstrates successful performance despite not being explicitly oriented towards events. These algorithms will each be discussed in more depth in the upcoming chapters. In addition to preferring separable descriptions, the other

major requirement we have is that our feature description allow for good comparisons between similar types of events. With the exception of chapter 4, which deals with exact fingerprinting, all of the experiments described here focus on the problem of finding good representations to describe audio events in a way that will highlight the similarity between events of similar classes.

This work offers insights into several different methods for investigating the content of noisy, unconstrained audio recordings. Specifically:

- a use of the matching pursuit algorithm for identifying similar events (chapter 3)
- proof of concept for fingerprinting videos taken at the same time and place, based on audio content (chapter 4)
- an exploration of transient-based features for the purpose of comparing audio events and classifying video content (chapter 5)
- a system for using convolutional non-negative matrix factorization for detecting and classifying audio events (chapter 6)
- a system using auditory-model-inspired features for classifying video content (chapter 7)

Some of the work presented here has been previously published in [Cotton and Ellis, 2009; Cotton and Ellis, 2010; Cotton *et al.*, 2011; Cotton and Ellis, 2011].

1.2 Thesis Outline

This thesis covers a number of related approaches to describing audio signals in ways that would be helpful for video search tasks. The organization is as follows. In chapter 2 we discuss prior approaches to the three main tasks that we cover in this thesis: audio concept classification, audio event detection, and audio fingerprinting. Chapters 3 and 4 both deal with the idea of using matching pursuit as a method for characterizing audio signals. In chapter 3 we describe an initial attempt to use matching pursuit atoms along with the locality sensitive hashing technique to identify instances of similar auditory events. This exploration inspired us to investigate the further potential of the matching pursuit algorithm. In chapter 4 we investigate the use of matching pursuit with exact hashing to prototype an audio fingerprinting system that would enable the

searching of internet video for instances of videos taken by different people at the same place and time. Continuing our exploration of how to characterize similar but non-identical audio events, we investigated using time-frequency patches composed of reconstructed matching pursuit atoms as a way to build audio dictionaries in [Jiang *et al.*, 2009] (this work is not directly discussed in this thesis). This approach represented only select segments of the frequency spectrum and worked less well than we had hoped, but this led us to the idea of using full-spectrum time-frequency patches to compare audio events. In chapter 5 we describe a system composed of a transient detector and subsequent time-frequency patch feature description. This system was used to build a general purpose video soundtrack classification system. The desire to find a more reliable way of identifying meaningful time-frequency patches led us to investigate a version of non-negative matrix factorization for separating audio events into commonly occurring basis patches and activations. We used this representation to perform event detection on a set of example events in the presence of noise; these experiments are described in chapter 6. This system was promising but proved too computationally intensive to run on the type of large video databases that we were interested in experimenting on. We were still interested in finding better ways to characterize the audio of large video databases. Finally, we turned to a promising auditory-model-based classification system described in [Lyon *et al.*, 2010] and investigated ways to adapt it for more efficient audio descriptions. In chapter 7 we describe these experiments and some encouraging results on our classification system using these features.

Chapter 2

Prior Work

In this chapter, we review previous approaches to the problems we address in this thesis. We primarily address three somewhat related but distinct tasks here: general classification and tagging of audio concepts (chapters 5 and 7), acoustic event detection and classification (chapters 3 and 6), and audio fingerprinting (chapter 4).

2.1 General Audio Concept Classification and Tagging

It is somewhat difficult to cover all of the various possible work encompassed by the topic of audio classification, since similar tasks have been tackled by numerous groups in widely varying contexts and for different applications. Our primary application interest in this thesis is the indexing and search of specifically consumer-style video recordings, but audio concept classification has obvious applications in a number of other areas: professional video database search, mobile audio recording devices, surveillance, and robotics, to name a few. Each application domain has its own constraints and complications which change the nature of the systems that have been developed for each. Furthermore, it is difficult to compare among different systems due to the lack of widely used standardized datasets or even concept labels (tags). Consequently, we will do our best to provide an overview of the prior work that is most relevant to our current task.

Audio classification could deal with any of a number of possible classes of audio. A number of early works in audio classification focus mainly on the problem of discriminating between two or three very distinct classes, such as speech, music, and silence. [Saunders, 1996] builds a system

to discriminate speech from music based primarily on features concerning the zero-crossing rate. [Scheirer and Slaney, 1997] use a diverse set of 13 different audio features to distinguish between speech and music, and compare the performance of several different classification frameworks for this task. [Patel and Sethi, 1996] suggests extracting a number of features directly from MPEG encoded audio, for the purpose of classifying professionally processed video into three categories: dialog, non dialog, and silence.

[Wold *et al.*, 1996] use the statistics of five basic audio features to build a content-based retrieval system. They test this system by performing classification experiments on a database containing multiple classes of sounds, including music (different instruments), speech, and various animal, natural, and mechanical sounds. [Foote, 1997] presents improved retrieval results on this same dataset, using the standard speech recognition features mel-frequency cepstral coefficients (MFCCs) and a tree quantization scheme.

[Li *et al.*, 2001] compare low-level acoustic features with MFCC and linear predictive coefficients (LPC) features under a Bayesian classification framework. They conclude that the MFCC and LPC features were more useful for their classification task, which included speech (with different numbers of speakers), music, environmental noise, silence, and various combinations.

[Casey, 2001] describes a classification system for 19 generic audio classes, using spectral feature information and hidden markov models (HMMs).

[Slaney, 2002] details a retrieval system for generic audio categories (specifically, a collection of sound effects) based on a semantic mapping algorithm; he uses a multi-frame derivative of MFCC features as the acoustic representation.

Some of the work presented in this thesis was inspired by [Chu *et al.*, 2009]. In it Chu et al. use statistical features derived from a matching pursuit (MP) decomposition of audio, in combination with MFCC features, to obtain better classification accuracy among recordings from 14 types of audio environments (in other words, background noise that is characteristic of different locations). In chapter 3 we use an MP decomposition for identifying individual similar audio events, but not for overall background sound classification.

[Eronen *et al.*, 2006] also examine the problem of environment recognition (using recordings from 27 environment classes), comparing between MFCC, LPC, and a collection of low-level acoustic features, and using an HMM to perform classification.

[Kalinli *et al.*, 2009] develop a system for classifying sound effect clips into 21 categories, based on selecting a subset of salient points and characterizing only those points, using standard MFCC features. This approach bears some similarity to our attempt to isolate foreground transient events for the purpose of audio clip classification in chapter 5, although we try to describe these events individually using features on a longer time scale, whereas Kalinli still describes the clips with (a reduced subset of) MFCC statistics.

[Lyon *et al.*, 2010; Lyon *et al.*, 2011] present a system for doing audio retrieval of sound effects clips over a large set of possible tags; their approach is based on a feature representation called a stabilized auditory image (SAI) which is derived from an auditory model. This work serves as the starting point for our exploration in chapter 7. We re-examine the use of these features for our classification problem and then demonstrate a simlarly effective but computationally simpler set of features.

In general, our problem space resembles a few of the tasks handled in the above works. The main difference is that our data and ground truth concepts emerge from the domain of unconstrained consumer-generated video content. The data to be classified is therefore noisy and not well controlled. Our data is also labeled at a longer clip level. In comparison, many of the datasets considered above consist of cleaner, pre-segmented sound examples, with well-defined classes. Additionally, in most of the above cases, the classes to be detected are considered to be mutually exclusive, whereas most of our data instances have multiple applicable tags. The work on environment recognition [Eronen *et al.*, 2006; Chu *et al.*, 2009] is the most similar to our problem, but still uses mutually exclusive tags/classes.

2.1.1 Video Indexing and Search using Audio Information

More relevant to our work is the specific task of video content characterization using audio information. While video indexing systems are generally primarily based upon visual information, there have been some attempts to use the audio modality to perform or assist in this task.

[Brezeale and Cook, 2008] present a survey of techniques for video classification, including some audio based approaches.

[Liu *et al.*, 1997; Liu *et al.*, 1998] build a system to classify amongst five types of television programs using a collection of audio features and neural networks and hidden markov models

(HMMs), respectively.

[Zhang and Kuo, 2001] present a system for segmenting and classifying the audio in video soundtracks (television and movies) using a relatively small number of simple audio features and a rule-based classification algorithm. They consider a slightly wider variety of classes: speech, music, song, general environmental sounds, silence, and combinations of these.

[Lu *et al.*, 2003] classify audio data from a variety of sources (television, internet, music tracks) using MFCCs plus a handful of acoustic features and classify them into five classes (speech, music, silence, environmental sounds, and speech with music or noise) using support vector machines (SVMs).

Our work differs from most of the above in that we are primarily interested in analyzing the type of low and extremely variable quality consumer-generated videos that are found on YouTube and other internet video-sharing sites, rather than professionally produced content like movies and television. The presence of excessive background noise, variable recording equipment quality, and the lack of intentional audio cues found in professionally produced material make the audio analysis of internet-style video a much more difficult task.

In recent years there has been some classification work aimed specifically at internet-style video content.

[Ramachandran *et al.*, 2009] combine classifiers derived from different modalities (video features and various types of metadata) into a consensus learning scheme that outperforms any of the individual classifiers on a set of videos obtained from YouTube. Although their scheme could easily incorporate audio-based classifiers, they do not seem to actually use audio information in these experiments.

[Wang *et al.*, 2010] devise another fusion framework for classifying YouTube videos by combining multiple data modalities. They use a number of text and video features and a couple of audio features (volume and spectrogram features). They note that their focus here is not on the particular features used but on the combination framework.

[Ekenel *et al.*, 2010] use a wide variety of features, including audio features (MFCCs, fundamental frequency, energy, and zero crossing rate) with SVMs for broadcast video genre classification. They also apply their system to a set of YouTube videos, but this set is selected to resemble the broadcast genre content, and even so the authors note that the YouTube data is significantly more

difficult to classify due to the higher variability in content.

In [Lee and Ellis, 2010] our colleague Lee presents a comparison of several approaches for audio-based consumer and internet video classification; all are fundamentally based on modeling the statistics of video clips' MFCC features.

Much of this work and audio classification work in general tends to rely on some version of MFCC features, which capture only stationary spectral information over very short time frames (around 25 ms) and which are easily corrupted by (typically irrelevant) background noise that is prevalent in consumer video. We investigate two different types of novel features as alternatives (and complements) to MFCCs. In chapter 5 we explore using spectral information on a longer timescale to describe only select portions of the audio signal which contain transient events (rather than representing all of it uniformly). In chapter 7 we extend Lyon's work on stabilized auditory images [Lyon *et al.*, 2010] for use in video classification, with novel features that are similarly based on short-term autocorrelation but are computationally less expensive.

2.2 Acoustic Event Detection and Classification

Above we discuss the characterization of pre-segmented audio clips for the classification of general, relatively high-level concepts. A related task that we address is the detection and classification of instances of specific audio events, within a longer segment of audio. As in the generic classification case above, work in this area could focus on any number of specific event types. The need for finely-labeled ground truth data in this task makes it somewhat more difficult to acquire labeled data than for generic audio classification tasks.

[Radhakrishnan *et al.*, 2005] design a two-level framework for detection of suspicious audio events in a surveillance application. They use standard MFCC features with a Gaussian mixture model (GMM) framework to adaptively model the background environment and detect frames that seem to be outliers and therefore may be indicative of suspicious behavior.

[Cai *et al.*, 2006] present a system for detecting common sound effects in movies and television, using both traditional acoustic features and some novel spectral features within an HMM framework.

[Mesaros *et al.*, 2010] explore event detection in the context of audio recorded in public environ-

ments, similar to the consumer video and environmental data discussed above. In their evaluation they use HMMs with MFCC features to try to detect 61 classes of events, but obtain relatively low detection accuracy.

An extensive investigation into acoustic event detection algorithms was performed as part of the CLEAR (CLassification of Events, Activities, and Relationships) evaluations under the European CHIL (Computer in the Human Interaction Loop) project. This project generated three databases of audio recorded in meeting room environments, which contain the sort of events that would be useful to detect for automated meeting room technology. This data is described in detail in [Mostefa *et al.*, 2007]. Each dataset contains 13 to 16 defined classes of events (such as cough, door knock, and phone ring) and ground truth labels at the event level, as shown in figure 2.1. Two of these databases contain isolated events; we use one of these for our experiments in chapter 6. The third dataset contains the same events, but interspersed with and possibly overlapping with speech and with each other.

[Temko *et al.*, 2007] describe the approaches of the three teams involved in the evaluation. In these evaluations, the teams each perform both event classification on pre-segmented instances and event detection; for brevity we will only describe the detection systems here. One of the teams' systems is based on filter-bank energies and a handful of other features. The statistics of these features are taken over a sliding window, the sliding window frames are classified by SVM (after a preliminary segmentation step), and the results are smoothed to get a final detection sequence. Another system is based on MFCC features and custom continuous density HMMs, also with a preliminary segmentation step. The last system also uses MFCCs and continuous density HMMs, but without preliminary segmentation before classification. This last system generally performed better at the detection task (had lower error rates). [Zhuang *et al.*, 2010] present improved results on this same dataset, using a feature boosting approach and two complex detection models, one HMM-based and the other using an SVM-GMM-supervector.

Unlike some of the works above, we are specifically interested in detecting events in the presence of background noise, since this is the primary problem encountered in our domain of interest (consumer video). Consequently, both of the approaches we examine are intended to have noise-robust properties, especially in comparison to standard MFCC features which are easily corrupted by additive noise. We first attempt to leverage the energy efficiency and sparsity properties of a

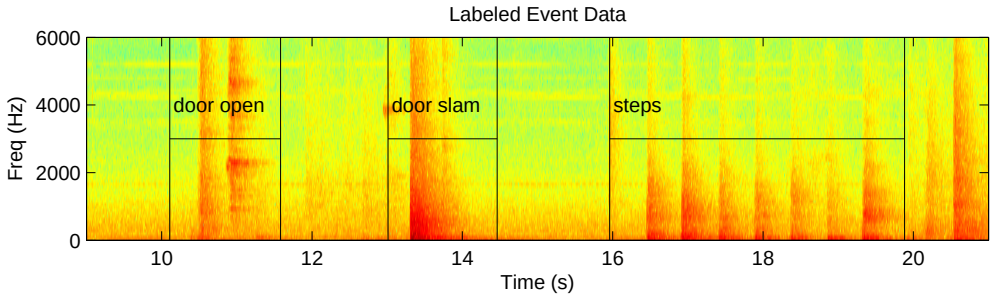

Figure 2.1: Example of labeled meeting room data collected for the CHIL project.

matching pursuit decomposition to extract components of events and then identify similar events despite the addition of a noisy background (chapter 3). Then in chapter 6 we use a formulation of non-negative matrix factorization (NMF) as another way to extract noise-robust component descriptors from individual audio events. We use features derived from these components to build an event detection system which outperforms one based on MFCCs in the presence of noise. Since NMF has an obvious application to describing parts-based components of audio, it has previously been used for similar tasks. For example [Cho and Choi, 2005] use features derived from NMF decompositions of spectrograms to classify speech, music, and environmental sounds. They do this by uniformly segmenting the spectrogram into patches and using conventional NMF on this pre-determined set of patch offsets; in comparison we take advantage of a convolutive formulation of NMF [O'Grady and Pearlmutter, 2006] to allow the algorithm to identify similar spectrogram patches without the need to pre-define the set of possible locations as segmenting them does.

2.3 Audio Fingerprinting

In chapter 4 we present an audio fingerprinting implementation and application. Audio fingerprinting is generally concerned with the problem of defining a representation of audio content which is both robust to compression, noise, and other channel variation, and is also quickly and easily searched for matching content. It has been studied in a number of different contexts. Typical applications are identifying a recording of an unknown song and monitoring audio content. Fingerprinting has typically been implemented as a concise representation of the concatenation of a sequence of audio frames using a wide variety of audio features, including MFCCs, spectral flat-

ness, and subband modulation frequencies, among others. [Cano *et al.*, 2005] present an extensive overview of various fingerprinting approaches.

Our implementation is directly based on the fingerprinting technique of Wang's Shazam system [Wang, 2006], for identifying unknown songs in potentially highly noisy environments. Distinct from a large number of prior fingerprinting approaches which aim to characterize the entire spectrum, Wang's technique is based on extracting only a subset of high-energy peak locations in the time-frequency map of the audio content. These peaks are then paired to increase the relative uniqueness of the representation and hashed for ease of search. A large set of matching peak pairs is highly indicative of the underlying audio content being a genuine match. This technique is highly robust to additive noise and channel variations, while still being very discriminative of content. In our implementation we suggest a variation on the peak-picking method, by using the matching pursuit (MP) algorithm to select peaks. We do not claim improved performance over the original technique, since we do not implement the original for comparison. We are primarily interested in exploring the relevance of this type of robust fingerprinting to a new application scenario.

In our work we investigate the use of Shazam-style time-frequency peak hashing for the purpose of identifying and exploring simultaneously-recorded video content. There is an immense amount of video being recorded by the average person and often subsequently being shared online, in a number of social media contexts. We were interested in finding a scalable way to scan this content to find videos recorded during the same time at the same location. These may be taken from different perspectives and therefore not share a lot of visual content, but should share (differently processed versions of) the same audio content. Identifying this matching but differently processed and noisy audio content bears a lot of similarity to the Shazam problem of identifying songs in noisy environments.

There have been a couple of similar attempts to use audio fingerprinting for the goal of video indexing. [Shrestha *et al.*, 2007] use a more traditional block of audio frames -based fingerprinting algorithm for the purpose of synchronizing consumer videos recorded at the same time.

The work most similar to ours is that of [Kennedy and Naaman, 2009]. Like this work, they use a version of the Shazam fingerprinting algorithm. They collect a set of videos from three concerts and use fingerprinting and other techniques to cluster, temporally align, and analyze them. They are more concerned with organizing, synchronizing and presenting videos that they already know

were recorded at the same concert then in the logistics of identifying otherwise possibly unlabeled (and therefore unidentifiable) videos taken at the same time and place. One of our contributions is a proof of concept for the viability of searching through large databases of video for simultaneously recorded content that might otherwise be undiscoverable.

Chapter 3

Identifying Similar Acoustic Events using Matching Pursuit and Locality-Sensitive Hashing

In this chapter we explore the use of matching pursuit (MP) derived features to identify repeated patterns that characterize perceptually similar acoustic events. We use locality-sensitive hashing (LSH) to efficiently search for similar events. We describe a method for detecting repetitions of events and demonstrate performance on real-life data containing similar sounding event instances.

3.1 Introduction

There are many examples of sound events which may be heard multiple times in the same recording, or across different recordings. These are easily identifiable to a listener as instances of the same sound event, although they may not be exact repetitions at the waveform level. We define an event as any short-term, perceptually distinct occurence, e.g. a door knock. The ability to identify recurrences of perceptually similar events has applications in a number of audio recognition and classification tasks. This work was motivated specifically by the desire to relate repeated audio events with the visual source of the sound, as in a video.

Our goal is to identify characteristic patterns that can be used to search for the presence of an event, i.e. identifying a kind of fingerprint for the event. There are two main challenges to

this task: First, we must find a representation that is sufficiently invariant to differences in event instances, and to context such as background sounds, to allow repeated events to be matched, yet still captures enough detail of the sound to allow perceptually distinct events to be distinguished. Second, we need a way to efficiently search for these events in very large datasets.

Our approach to the first problem is to use the matching pursuit (MP) algorithm as the basis for our audio event representation. MP [Mallat and Zhang, 1993] is an algorithm for sparse signal decomposition into an over-complete dictionary of basis functions. MP basis functions correspond to concentrated bursts of energy localized in time and frequency, but spanning a range of time-frequency tradeoffs. By allowing the analysis to choose the bandwidth/duration parameter that best fits a feature in the audio – instead of adopting a single, compromise time scale as in the conventional short-time Fourier transform, MP allows us to describe a signal with the atoms that most efficiently explain its structure. Any fixed time-frequency decomposition would be similarly limited; for example, a wavelet transform would explain elements on a variable scale, but would still only have one bandwidth at each center frequency. The sparseness of the MP representation makes this approach robust to background noise, since a particular element, representing a maximally compact local concentration of energy, will experience less proportional change than a less compact representation as the surrounding noise level increases. MP features were proposed for environmental audio classification in [Chu *et al.*, 2008].

Our work is also inspired by previous work in searching for events using a strict, exact-match fingerprint technique [Ogle and Ellis, 2007]. That algorithm efficiently identified audio excerpts that were repeated in their entirety, such as pieces of music or electronic ring tones, from environmental audio. However, it was not able to identify "organic" sounds (such as the sound of a door closing) where there was nontrivial variation between successive instances of the event. In this thesis, we use a similar representation in terms of time-frequency energy peaks taken in pairs and characterized by their time difference, but instead of an exact hash we use locality-sesitive hashing (LSH) [Andoni and Indyk, 2008], an algorithm that uses the highly efficient constant-time mechanism of hash lookups to find near neighbors in feature space instead of only exact matches. LSH has been proposed for matching similar music items e.g. remixes of particular tracks [Casey and Slaney, 2007].

Section 3.2 describes our MP representation, section 3.3 describes how we search for recurring

events, section 3.4 describes our preliminary experiments to illustrate this idea, and we conclude with a discussion of the issues raised and the prospects for unsupervised discovery of repeating acoustic events.

3.2 Matching Pursuit Representation

The basis functions we use in the MP algorithm are Gabor functions, i.e. Gaussian-windowed sinusoids. The Gabor function is evaluated at a range of frequencies covering the available spectrum, scaled in length (trading time resolution for frequency resolution), and translated in time. Each of the resulting functions is called an atom, and the set of atoms is the dictionary, which covers a range of time-frequency localization properties. The length scaling creates long atoms with narrowband frequency resolution, and short atoms (well-localized in time) with wideband frequency coverage. This amounts to a modular STFT representation with analysis windows of variable length. During MP analysis, atoms are selected in a greedy fashion to maximize the energy removed from the signal at each iteration, resulting in a sparse representation. Atoms extracted from the signal are defined by their dictionary parameters (center frequency, length scale, translation) and by parameters the algorithm estimates (amplitude, phase). Here, we use the Matching Pursuit Toolkit [Gribonval and Krstulovic, 2006], an efficient implementation of the algorithm.

The dictionary we use contains atoms at nine length scales, incremented by powers of two. For data sampled at 22.05 kHz, this corresponds to lengths ranging from 1.5 to 372 ms. These are each translated by increments of one eighth of the atom length over the duration of the signal.

3.2.1 Psychoacoustic Pruning of Atoms

Since MP is a greedy algorithm, the first (highest energy) atom extracted from a given neighborhood is the most locally informative, with subsequent lower energy atoms selected to clean up imperfections in the initial representation; these are often redundant in terms of identifying key time-frequency components.

In order to reduce the size of the representation while retaining the perceptually important elements, we prune the atoms returned by MP with post-processing based on psychoacoustic masking principles [Petitcolas, 2003; Pan, 1995]. The objective of MP is to reduce the energy of the

error residual as much as possible at each stage, but owing to the limitations of human hearing, perceptual prominence may be only weakly related to local energy. In particular, lower energy atoms close in frequency to higher-energy signal may be entirely undetectable by human hearing, and thus need not be represented. A related effect is that of temporal masking, which masks energy close in frequency and occurring shortly before (backward masking) or after (forward masking) a higher-energy signal; the forward masking effect has a longer duration, while the backward masking is typically negligible. A similar approach, which incorporates a psychoacoustic model into the matching pursuit algorithm itself, has been explored in several places, such as [Heusdens *et al.*, 2002].

Our implementation creates a masking surface in the time-frequency plane, based on the atoms in the full MP representation. Each atom generates a masking curve at its center frequency with a peak equivalent to its amplitude, which falls off with frequency difference. Additionally we consider this masking curve to persist while decaying for a brief time (around 100 ms), to emulate forward temporal masking.

Atoms with amplitudes that fall below this masking surface are therefore too weak to be perceived in the presence of their stronger neighbors; they can be removed from the representation. This has the effect of only retaining the atoms with the highest perceptual prominence relative to their local time-frequency neighborhood. This pruning emphasizes the most salient atoms, and removes less noticeable ones; it is an important step in reducing the size of the search space and improving the relevance of the atoms as features (since secondary, "cleanup" atoms are usually removed), as well as reducing the probability of false matches in the search procedure described below.

Figure 3.1 shows an example audio clip containing two distinct transient events analyzed with MP. A large set of MP atoms (171 in this case) is extracted initially and then pruned with psychoacoustic masking (leaving 76 in the example).

3.3 Pair Formation and Pattern Discovery

We want a way to define a specific type of audio event by the common relationships between its atoms, if any exist. We start with an audio sample containing many separate instances of similar

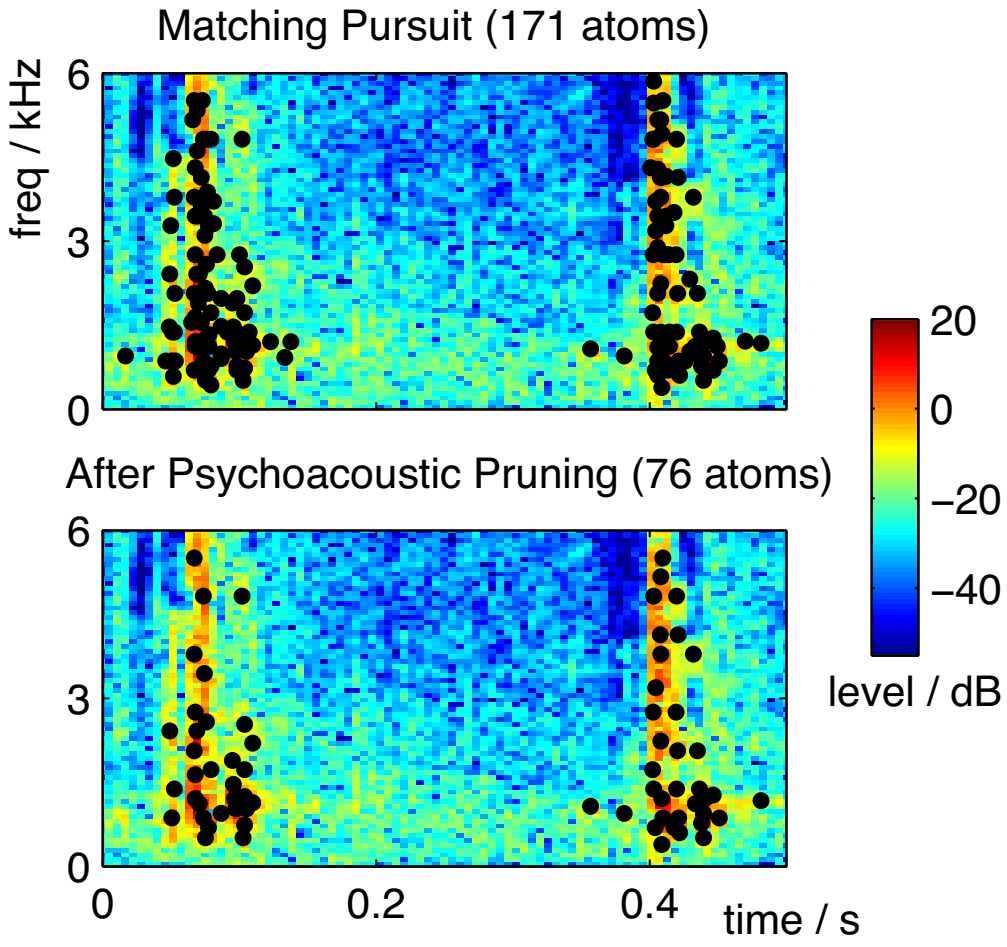

Figure 3.1: Matching pursuit atoms used to represent audio events, shown as dots at the centers in time-frequency overlaid on the spectrogram, before and after psychoacoustic pruning.

events that we would like to describe, potentially including small variations in the details of the instances. We form pairwise relationships between all pairs of atoms whose centers fall within a relatively short time window of each other. The pairs are characterized by the two center frequencies of the atoms and the time offset between them, creating a three-dimensional feature space (see an example of pair formation in figure 3.2). Bandwidth parameters could also be used to refine the space, but were not needed for the current demonstration. By excluding the energy of atoms, the representation becomes robust to variations in level and channel characteristics, provided the sound event energy remains sufficiently prominent to be included in the MP analysis.

The dimensions are normalized, then LSH is performed on the entire set of pairs. LSH makes

multiple random normalized projections of data items onto a one-dimensional axis. Items that lie within a certain radius in the original space will be sure to fall within that distance in the projection, whereas distant items have only a small chance of falling close together. The projected values are quantized, and near neighbors will tend to fall into the same quantization bin. These quantized sets are quickly recovered at query time through a conventional hash table. By consolidating the results across multiple projections, both chance co-occurrences due to unlucky projections, and the risk that nearby points will straddle a quantization boundary can be averaged out.

We start with a sound file that we know or suspect contains multiple instances of some sound event (perhaps mixed in with other, nonrepeating events). We form the MP representation, then store hashes describing all nearby atom-pairs in an LSH database. The database is then queried with every atom-pair hash in turn to identify large clusters of similar pairs i.e. atom-pairs that return large numbers of matches within some radius. Since the LSH queries are constant-time, this entire process takes a time proportional only to the number of atom-pairs, instead of the N^2 time required for exhaustive pairwise comparison. We assume clusters in atom-pair space arise from the repeating events, and we can link pairs into higher-order constellations if they share individual atoms. Thus, we arrive at a set of pairs we can use to recognize future instances of the repeating event. Each acoustic event may result in dozens of nearby atoms, leading to many local pairings. Detection does not require that *all* atoms and relations be successfully identified; it is frequently sufficient to detect only a small fraction of these "landmarks" to correctly identify a structure.

LSH requires a radius parameter be set to define how close nearest neighbors must be. This is essentially the definition of how "similar" the particular time-frequency structure of the characteristic features of two events must be for them to be considered instances of the same event class. Here the radius is tuned by hand, but this could easily be automated, for instance based on an estimate of the true number of event repetitions in this training set.

3.4 Experiments

We test our approach on a 13 second sample from a video soundtrack, containing 38 instances of the sound of horse hooves. This is a real recording, i.e. each hoof sound arises from a distinct physical event. We extract the top 3000 atoms with MP and perform psychoacoustic pruning on

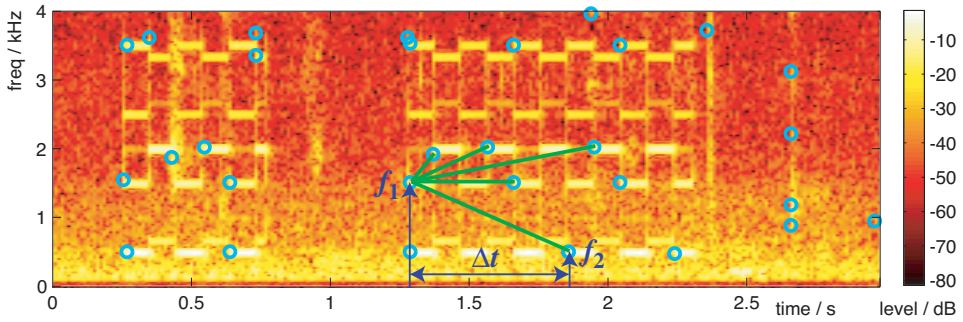

Figure 3.2: Example formation of atom pairs for hashing, defined by the triplet of (f_1,f_2,and ΔT).

the set of atoms, which reduces this number by about half. We use a ±35 ms time window for pair formation between atoms, which produces around 19,000 pairs.

We use a small LSH radius of 0.085 (in the normalized feature space) to cluster this set of pairs; this radius was tuned to give a relatively small number of hits, such that only very similar, hopefully characteristic pairs will be matched with each other. For this radius, the most common pair pattern had 35 nearest neighbors. We then select patterns with at least 20 nearest neighbors, which yields 25 pair patterns, each of which we infer occurs in at least half of the hoof sound events. Because the events we are describing here are very short in time, most of these pairs appear to be nearly simultaneous (i.e. very little time offset between the two atom locations) although some show a small but consistent time skew between center frequencies.

3.4.1 Event Detection

To test our algorithm, we mix the signal with a second soundtrack containing speech and general background noise. Listening to the mixtures, as a second signal is added many of the horse hoof events become less audible, especially those that overlap with speech, but those that are audible remain distinctly identifiable; it is this effect that we hope to reproduce. Even as the confusing speech and noise is increased, the atoms and pairs representing audible events should hopefully remain reasonably similar.

Figure 3.3 shows the percentage of nearly identical atoms retained from the original (pruned) atom represenation of the clean signal, as the mixing proportion changes. Lower SNR corresponds to more of the second, masking signal being added.

We perform our pair extraction process (MP, psychoacoustic pruning, and pair formation) on the mixture at varying SNRs, form atom-pairs and store them in an LSH database, then query this database using our previously saved 25 queries generated from the original clean signal. As an experimental variation, we try several different LSH radii including the "true" radius used to define the characteristic query pairs. At each SNR level, the matches found for each of the 25 queries are scored as true or false positives; each query is scored separately based on the locations of its matches in the original signal. True positives are matches found within a small window (5 ms) near the time of a similar query match in the original signal. False positives are any other matches found in the mixture. Although each of the hoof events usually contains multiple pairs, one pair found within an event is considered sufficient to detect that event.

Figure 3.4 shows the precision and recall of the system, at three different LSH radii (0.025, 0.085, 0.15). Recall is defined as the number of events (out of 38 total) with at least one match detected. Precision is defined as the number of correct matches found over all queries divided by the sum of all query matches.

Figure 3.5 shows a portion of the original signal, with the locations of query pairs marked; below it is the signal in a mixture, with true matches (black) and false positives (magenta) marked. To the right of the figure is a magnified view of a single event.

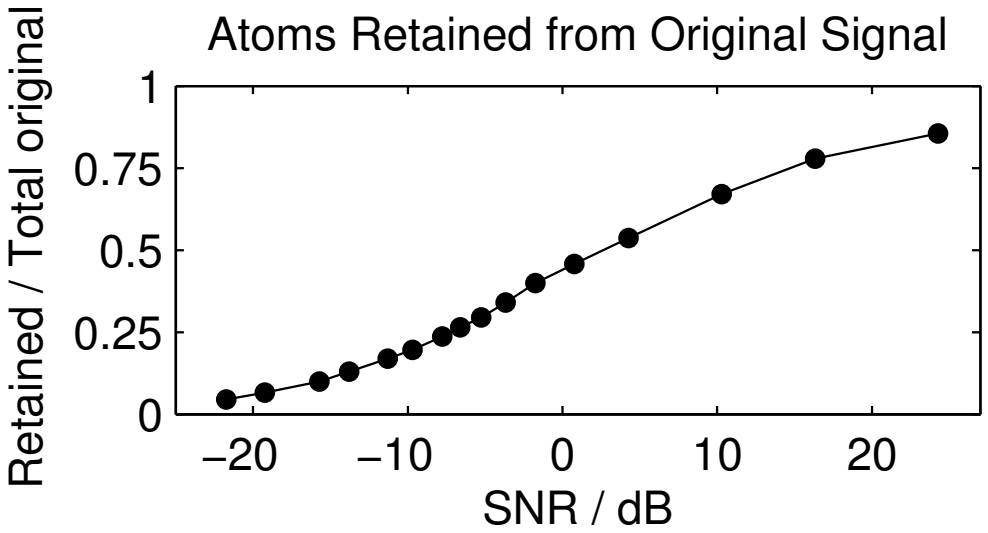

Figure 3.3: Percent of MP atoms retained from the original signal as the noise increases.

3.5 Discussion

Figure 3.3 shows that the pruned atom representation of the original events degrades reasonably gracefully as a second signal is added, with almost 25% of the atoms remaining essentially unchanged even at -10 dB SNR. These many MP atoms that stay nearly identical, even in noise, indicate that the atom pair patterns have the potential to form a noise-robust basis for identifying the sound events.

Figure 3.4 demonstrates that recall is reasonably stable over a range of SNR. This follows from our simple approach of OR-ing together all of the queries to detect the 38 events. It is promising, however, that nearly all of the events maintain a similar enough representation so that at least one nearly identical pair persists in each, even at fairly low SNR. This indicates that the atomic representation of the original events is staying relatively constant as the second signal is added.

The tradeoff between precision and recall is seen in the variation with radius. Results at a radius of 0.085 correspond to the radius for which the queries were originally chosen. Choosing a larger radius improves recall, if only slightly, at the cost of precision. A smaller radius will improve precision (slightly), but with a low recall rate.

Figure 3.5 shows examples of successful and unsuccessful event detection in noise. The top shows all instances of the 25 originally selected patterns. Below, the atom pairs in black are those that have been found nearly identically at low SNR as they were in the original signal. The magenta pairs are false positives. There are also a few events which have been lost entirely due to the presence of noise.

3.6 Conclusions

We demonstrate a promising approach for the detection of repeated events in large amounts of audio data. The patterns identified here are robust enough to be useful for event detection, even in the presence of noise. Practically, the main shortcoming is probably low precision, indicating that the patterns are not entirely unique to the event under consideration. However, this can probably be improved upon by tuning parameters such as the atom pruning threshold and LSH radius in both the discovery and detection stages of the algorithm. When selecting queries, we could also consider not just its commonality in that event, but the frequency with which it is found in other

generic audio; this would allow us to select patterns more unique to a specific event type and therefore improve precision.

There are several obvious enhancements which could make the algorithm more robust. Pair representations could be made more specific by incorporating other atom parameters into the feature space, such as atom length or amplitude difference between the pair. Pairs often found together could be joined into constellations, producing something closer to a fingerprint (or realistically, a set of potential fingerprints) for an event. A set of common constellations could be stored as potential queries for the detection task. Alternately, the pairs do not necessarily need to be explicitly linked together; they could each form an individual event detector, the results of which could be combined probabilistically for greater stability.

A set of atom pairs or constellations could be identified to define sets of events of different types. As demonstrated here, LSH would allow for efficient searching of any generic audio recording for the presence of patterns matching any of these events, which could be very promising for audio-visual analysis, among other applications.

In the next chapter, we extend our use of matching pursuit to the domain of actual audio fingerprinting, since this is another analysis tool that would be useful in video indexing.

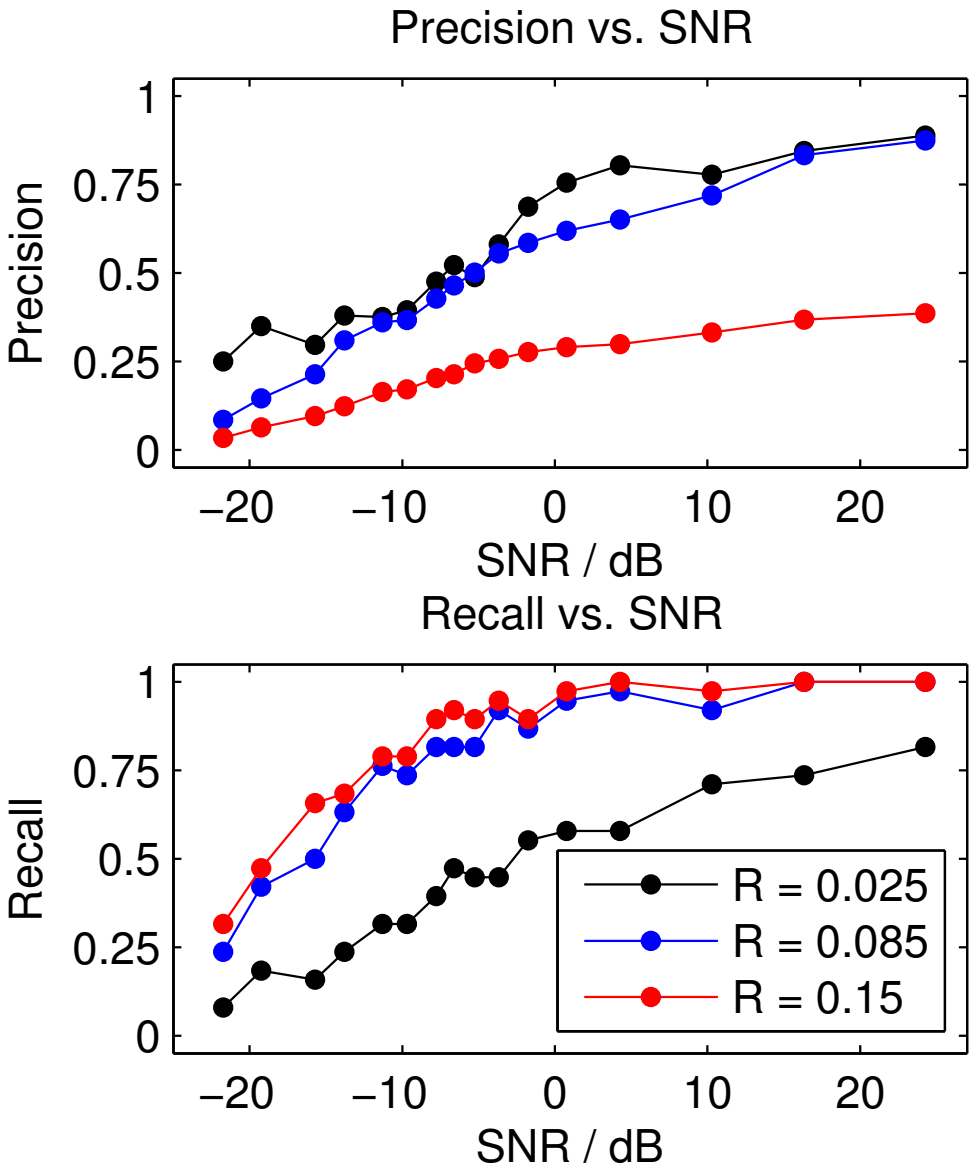

Figure 3.4: Detection, precision, and recall vs. SNR at three LSH radii.

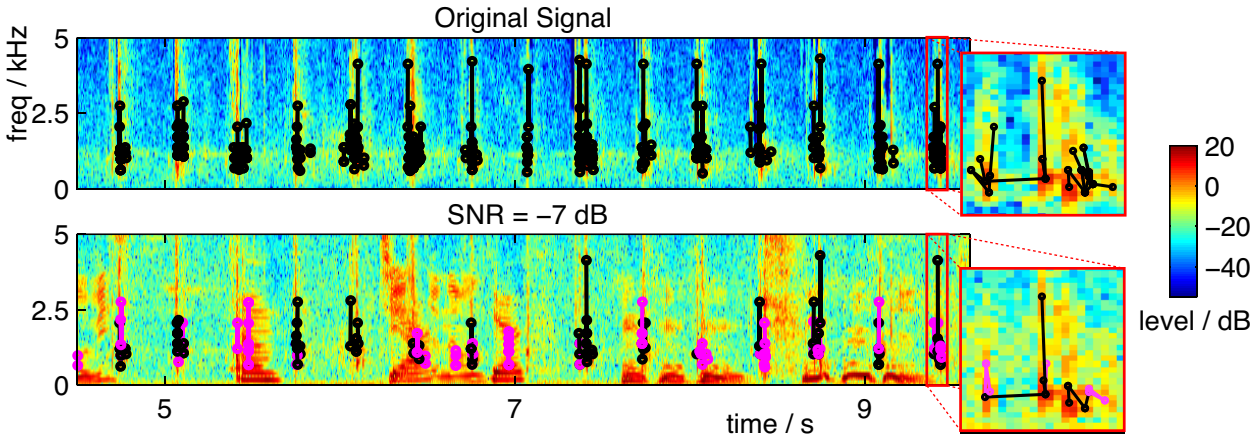

Figure 3.5: Original query patterns (top), and mixture detection results (bottom). Black pairs are correct detects; magenta are false positives.

Chapter 4

Audio Fingerprinting to Identify Simultaneously Recorded Videos

The proliferation of consumer recording devices and video sharing websites makes the possibility of having access to multiple recordings of the same occurrence increasingly likely. It should be possible to identify these co-synchronous recordings by their audio tracks, despite local noise and channel variations. In this chapter we explore a robust fingerprinting strategy to do this. Matching pursuit is used to obtain a sparse set of the most prominent elements in a video soundtrack. Pairs of these elements are hashed and stored, to be efficiently compared with one another. This fingerprinting is tested on a corpus of over 700 YouTube videos related to the 2009 U.S. presidential inauguration. Reliable matching of identical audio sequences in different recordings is demonstrated, even under difficult conditions.

4.1 Introduction

With the proliferation of recording devices and public sharing of video footage, there is an increasing likelihood of having access to multiple recordings taken during the same time at the same location. Due to the nature of amateur and online video it is likely that these recordings would not be labeled or tagged similarly enough to identify them with text information. We are interested in automatically discovering these alternate recordings, with the goal of improving online video search capabilities. It would be extremely difficult to identify these videos conclusively using visual

information, since different recordings are likely to be taken from entirely different viewpoints. It is, however, possible to consider doing this with the audio, since the same sequence of audio should be captured well by any recording made in the same vicinity.

The problem under consideration has similarities with that of identifying identical musical recordings in the presence of noise and channel variations. In both cases, we expect to see a lot of invariant underlying structure (e.g. spectral peaks) in the same relative time locations, but possibly corrupted with different channel effects and mixed with varying levels and types of noise. This problem is addressed in [Wang, 2006] by using the locations of pairs of spectrogram peaks as robust features for matching. A similar approach was applied for the identification of repeated events in environmental audio in [Ogle and Ellis, 2007]. A variation of this technique, using the matching pursuit (MP) algorithm to select peaks rather than a standard spectrogram representation, was presented in chapter 3 to group similar but nonidentical audio events.

In section 4.2 we present a strategy for using MP to select salient elements of a signal, pairing these elements to create distinguishing landmarks, and efficiently searching for matching landmarks. In section 4.3 we describe the video data used to test this strategy, and in section 4.4 we examine the precision of our search results.

4.2 Algorithm

4.2.1 Matching Pursuit

MP [Mallat and Zhang, 1993] is an algorithm for sparse signal decomposition into an over-complete dictionary of basis functions. MP basis functions, called atoms, correspond to concentrated bursts of energy localized in time and frequency, but spanning a range of time-frequency tradeoffs. The MP algorithm iteratively selects atoms corresponding to the most energetic points in the signal, as long as they can be approximated by a basis function in the dictionary. In contrast to selecting peaks with a fixed-window spectrogram representation, MP can capture salient features in the signal at varying time-frequency scales.

In our fingerprinting, each video soundtrack is decomposed into an MP representation in order to identify a sparse set of the most salient elements it contains. It is most straightforward to decompose the entire length of a video at once, in order to avoid issues with windowing and

boundary overlaps. A variable number of atoms are extracted from each video, roughly a few hundred atoms per second, although these are not uniformly distributed throughout the video. Selecting a larger number of elements than will actually be used from the signal as a whole will tend to sufficiently cover both louder and quieter portions of the signal; then a smaller number of atoms in each local area can be selected from these.

We use the efficient implementation of MP from [Gribonval and Krstulovic, 2006]. Our dictionary contains Gabor atoms (Gaussian-windowed sinusoids) at nine length scales, incremented by powers of two. For data sampled at 22.05 kHz, this corresponds to window lengths ranging from 1.5 to 372 ms. These are each translated in time by increments of one eighth of the atom length over the duration of the signal.

4.2.2 Landmark Formation and Hashing

As in chapter 3 a landmark consists of a pair of atoms, and is defined only by their two center frequencies and the time difference between their temporal centers (see figure 3.2). These values are quantized to allow effecient matching between landmarks. The time resolution is 32 ms. The frequency resolution is 21.5 Hz, with only frequencies up to 5.5 kHz considered; this results in 256 discrete frequencies.

Note that we do not perform psychoacoustic pruning in this experiment as we did in chapter 3; the idea of optimizing for perceptual similarity is less relevant for matching truly identical audio than it is for identifying different but similar-sounding events.

For every block of 32 time steps (around 1 second), the 15 highest energy atoms are selected. Each of these is paired with other atoms only in a local target area of the frequency-time plane. Here, each atom is paired with up to 3 others; if there are more than 3 atoms in the target area, the closest 3 in time are selected. This leads to approximately 45 landmarks per second. The target area is defined as the frequency of the initial atom, plus or minus 667 Hz, and up to 64 time steps after the initial atom.

The landmark values as quantized above can be described as a unique hash of 20 bits: 8 bits for the frequency of the first atom, 6 bits for the frequency difference between them, and 6 bits for the time difference. A hash table is constructed to store all the locations of each landmark hash value. Landmark locations are stored in the table with an identification number from the

originating video and a time offset value, which is the time location of the earlier atom relative to the start of the video.

4.2.3 Query Matching

To find instances of the same audio sequence as found in a query video, the query is decomposed with MP as described above. The video is then divided into five-second (non-overlapping) clips, and landmarks are formed from the atoms in each clip and hashed, as described in section 4.2.2. Each clip will contain an average of 225 landmarks. We break the query into these shorter pieces to improve the opportunity for matching subportions of videos, as well as to provide independent tests of matches between longer videos, as described below. The hash table is queried for each of the landmarks found in the five second clip. The start time of each query landmark is treated as a reference time; this is subtracted from the offset times for landmarks returned from the table. A likely match will therefore return multiple landmarks from the same video, all reporting the same relative offset time from their corresponding query landmarks.

4.3 Video Database

We wanted to test this algorithm on a set of videos likely to contain multiple versions of the same sequence of audio. We chose to consider videos taken during the 2009 American presidential inauguration. We assumed there were likely to be many different professional and personal recordings of the ceremony available, given the massive public attendence and news coverage. We obtained a set of videos from YouTube using the query "inauguration obama". YouTube query results are limited to 1000 items; this and other complications (e.g. videos with no soundtrack) limited our actual database set to 733 videos. Other than this, no hand selection or filtering was done on the video set. The set comprises 56.2 hours of video. All audio is sampled at 22.05 kHz. The list of YouTube video IDs for this dataset can be found at `http://www.ee.columbia.edu/~cvcotton/InaugurationDataset.txt`.

4.4 Results

4.4.1 Match Evaluation

Each video was processed as above and stored in the hash table. Then each video was divided into five second (non-overlapping) segments, and used as a query to the database. Matches to the query video itself were discounted. Matches were returned based on the proportion of identical landmarks matched to the total number of landmarks in the query segment. A lower threshold proportion will result in more matches returned. In this experiment, all matches containing at least 5% of the query landmarks and at least 10 actual landmarks were considered.

Two videos with a long stretch of matching audio will result in a number of sequential query segments matching the same video, with the same time offset. For the purpose of simplifying evaluation, all matches occurring between the same two videos at the same offset are collapsed into a single match, spanning the time from the start of the first matching segment to the end of the last matching segment. This is a reasonable assumption, since it is unlikely for two videos to match at multiple points with the same offset unless they are truly part of the same long matching segment.

4.4.2 Estimating Precision

The procedure described above produced 34,247 individual matches, with matches at the level of 5% (and containing at least 10 landmarks) being the minimum considered. Of these, 91 matches occurred above the level of 40%; manual examination of these results revealed them to be largely matches between videos in several different 'series', each with a signature introduction sound or music at the beginning of the video. There were also six pairs of identical or nearly identical videos in this set. All these matches are accurate, but not particularly interesting. The 'series' videos in general did not contain footage of the inauguration itself. For simplicity, they and (one copy of) the six exact duplicate videos were all removed from the database. This removal left a set of 31,756 matches. Of these matches, 8681 (27%) involved matches between multiple five-second segments in a row (that is, the match was detected over a longer time period than a single five-second clip). Given the small probability of two videos matching with the same time offset in multiple places by chance, it is reasonable to assume that most of these long matches are accurate. This was confirmed by random checking of these long matches. Similarly, even short matches with a relatively high

proportion (over 15%) of matching landmarks seemed generally accurate on the basis of casual spot-checks.

We therefore wanted to examine and quantify the precision of short matches (a single five-second clip) with low proportions of matching landmarks. In order to estimate the precision of these, we randomly sampled 1.5% of them (or at least 20 samples, whichever was higher) to hand check at each match percentage level. A large number of the incorrect matches were between clips which either both contained music or crowd noise. Further examination revealed that a large number of these spurious matches contained a long chain of landmarks in a single frequency bin. Since this mistake seemed easy to avoid, we subsequently repeated the fingerprinting experiment with a modified landmark formation rule.

4.4.3 Removing Single-Frequency Chains

We repeated the experiment, removing long strings of matches in the same frequency bin by not allowing atoms at the same frequency to form landmark pairs with each other in the initial hashing. This change reduced the total number of matches discovered by almost 40%, from 34,247 to 20,805, and likely reduced the number of spurious matches significantly. We followed the same methodology described above in removing the exact duplicates and other uninteresting pairs. Again, we divided the final set of matches into those lasting longer than five seconds (which we assume are highly likely to be correct), and those that are only five seconds long (of which we would like to evaluate the precision). We summarize the total numbers of matches discovered in this experiment and in the original experiment in table 4.1. Note that the number of matches longer than five seconds (presumed correct matches) has remained relatively constant, while the number of short matches has been reduced dramatically.

Again, we sampled the new set of five-second matches to evaluate the precision, using the same strategy as described above. Table 4.2 shows a breakdown of the number of matches discovered at each percent matching level, between 5% and 17%, and the number that we hand checked in each case to estimate precision at that level. Once again, we assumed based on our preliminary assessment that matches over the 17% level were highly likely to be accurate, and did not check above this level. Figure 4.1 displays the precision found by hand checking the new set, compared with the precision found in the original experiment above.

As expected, removing the single-frequency landmarks increased the precision at all proportion levels. This second experiment seems to be a more fair appraisal of the algorithm's actual potential for large-scale discovery. Although precision still falls at very low percentages (5-6%), the precision remains surprisingly high above that level. In this experiment, it is nearly perfect down to 15%. When we do start observing failures, around the 14% level, they are mostly accounted for by 'quiet' audio segments, where there are presumably fewer atoms and fewer landmarks, and therefore 14% of them may be quite a low absolute number. Another common failure case is two clips with different music but similar beats.

We would like to make it clear that the point of these evaluations was to quantify the exact level at which the precision of this fingerprinting algorithm breaks down. From our manual checking, we can easily say this algorithm is robust enough that matches longer than five seconds and/or with more than 17% of atoms matching are almost invariably correct. Therefore our goal was to characterize the precision of the shortest (five-second) and lower percent matches. We feel this evaluation fairly characterizes the algorithm's precision down to low levels, although we cannot quantify the recall since we do not know the ground truth for the full set of matches that may occur in this database.

Number of Matches	Original	Removed Single-Freq
Total	34,247	20,805
Minus exact dups	31,756	19,025
Longer than 5 sec	8681 = 27%	8645 = 45%
5 sec only	23,075 = 73%	10,380 = 55%

Table 4.1: Number of video matches discovered.

4.4.4 Identifying Unique Recordings

In the process of examining matches above, a number of different types of accurate matches were observed. The most common at high landmark proportion levels were between videos taken by different news organizations at the same time during the inauguration. Another set were between videos which were obviously derived from the same original news recording, but with various levels of additional processing. Some of these were re-broadcasts by a news organization in another country,

Percent Match	Original		Removed Single-Freq	
	Num Found	Num Checked	Num Found	Num Checked
5%	8398	126	3182	48
6%	7143	107	2671	40
7%	2850	43	1236	20
8%	1494	22	861	20
9%	894	20	550	20
10%	620	20	429	20
11%	439	20	304	20
12%	302	20	292	20
13%	225	20	197	20
14%	169	20	137	20
15%	124	20	129	20
16%	89	20	64	20
17%	79	20	86	20
$> 17\%$	249	0	242	0
Total Num	23,075	478	10,380	308

Table 4.2: Number of five-second matches, by percent of atoms that match.

with additional narration or translation over the original footage. Others had been remixed with music. A surprisingly large number seemed to be videos taken of television screens. A very small number were discovered which had been taken by amateurs in attendance at the inauguration.

An interesting question is how many of these independent recordings exist in the database. We observed that each of the various professional news recordings represented in the database tend to match each other well, since they are all very clean long-duration recordings of exactly the same sequence of audio. We attempted to estimate this subset of professional recordings by selecting any videos that match each other in at least 15% of the total landmarks, contain at least 25 actual matching landmarks, and are at least 20 seconds long. This described 691 matches, between 118 separate videos.

We expect amateur recordings to also match one or more of these professional videos, but likely

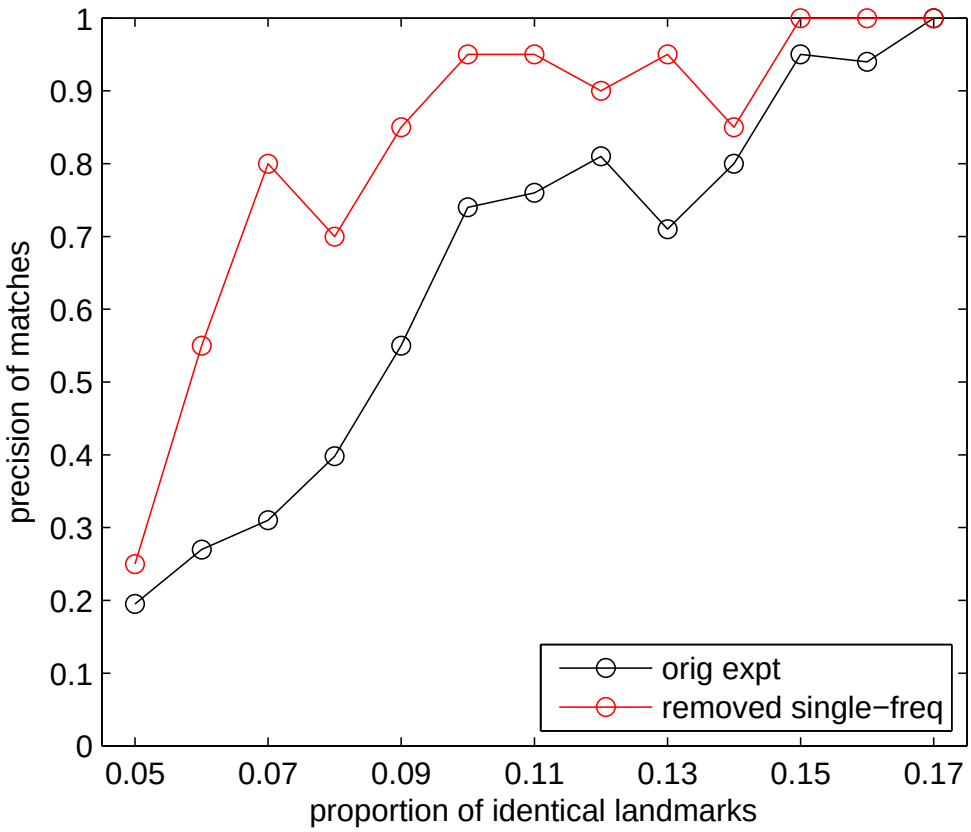

Figure 4.1: Precision of five-second fingerprint matches, with and without single-frequency chains removed.

for a shorter duration and/or at a smaller landmark percentage level. We therefore looked at the set of videos which match any of the presumed professional set described above, in at least 10% of the landmarks, with at least 20 actual landmarks, and with no minimum duration. This yielded a set of 2130 matches, between 189 videos (in additional to the 118 above). For each of these videos, the top (highest proportion of landmarks) match was returned for examination. Many of these videos turned out to be heavily processed or remixed versions of a professional recording. A few were actually incorrect, but commonly mistaken, videos containing either music or crowd noise. A number of them (14) were actually discovered to be independently recorded videos of the inauguration ceremony or related festivities, that were reliably matched with professional footage of the same ceremony. Fig. 4.2 demonstrates the matching landmarks in one of these amateur video results; figure 4.3 shows frames from each video. These and other examples of the matches

described here can be viewed at `http://www.ee.columbia.edu/~cvcotton/vidMatch.htm`.

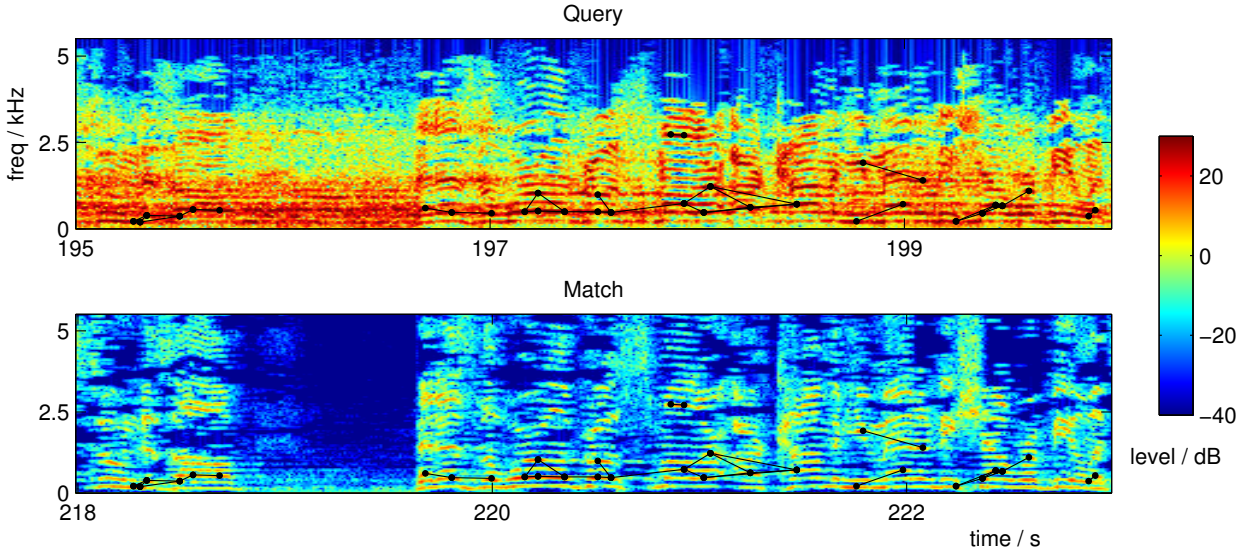

Figure 4.2: Top: a clip of an amateur video of the inauguration speech; Bottom: a CNN broadcast. The two share 59 common landmarks over a 10 second period (only five seconds shown). The matching landmarks are drawn in black.

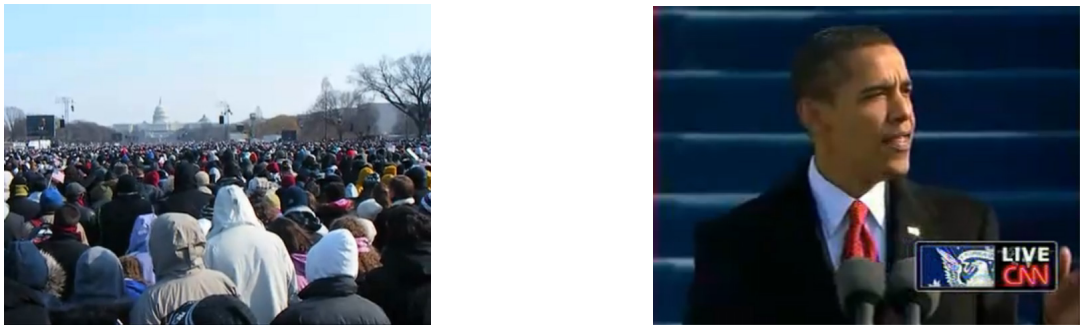

Figure 4.3: Frames from two matching videos, the left amateur and the right professional.

4.5 Conclusions

The fingerprinting procedure outlined here was demonstrated to be robust to high levels of noise and channel differences. The system as demonstrated reliably returns accurate matches with very few false positives at a match threshold of around 15% of landmarks.

We believe that this type of robust audio fingerprinting would be a valuable tool for online video search, since the landmark hashing method makes the algorithm theoretically efficient enough to perform over large video databases. The goal of the search might be strictly finding co-synchronous video recordings, as described here, or might include other related applications. For example, this technique has the potential to be very useful in the task of content-based video copy detection, as described in [Anguera and Adamek, 2011].

Chapter 5

Audio Transient Event Features for Video Soundtrack Classification

In this chapter we present our initial results on video classification using event-based features. Unlike many approaches in the literature which describe the audio via statistics of mel-frequency cepstral coefficient (MFCC) features calculated on uniformly-spaced frames, we investigate an approach to focusing our representation on audio transients corresponding to soundtrack events. These event-related features can reflect the "foreground" of the soundtrack and capture its short-term temporal structure better than conventional frame-based statistics. We evaluate our method on a test set of 1873 YouTube videos labeled with 25 semantic concepts. Retrieval results based on transient features alone are comparable to an MFCC-based system, and fusing the two representations achieves a relative improvement of 7.5% in mean average precision (MAP).

5.1 Introduction

The enormous volumes of video being captured by consumers, stored on computers, and uploaded to the Internet, presents an urgent need for automatic tools for video classification and retrieval – since they are often insufficiently labeled by their creators. While visual content is the most obvious basis for automatic analysis, the soundtrack of a video also contains important information about a clip's content, information that may be complementary to the video stream, and that may also be easier to process or recognize. Our colleagues have been investigating the use of soundtracks in

video classification for several years [Chang *et al.*, 2007; Lee and Ellis, 2010].

A common approach to modeling audio is to extract features from uniformly-spaced short-time frames (e.g. 25 ms) extracted from the entire length of the soundtrack. A video's soundtrack, however, may have information that is very unevenly and sparsely distributed – such as an occasional dog barking, or other foreground sound event. Short, sparse events of this kind may have relatively little statistical impact when mixed in with all the frames in the clip, and their information may be lost.

To address this risk, we have developed a system for representing the soundtrack based on identifying and modeling the individual audio transients it contains. By analyzing only a subset of points in the soundtrack that are likely to contain distinct event onsets, our goal is to develop an approach that is complementary to the typical global background model. At each transient event time, we also model the local temporal structure over a relatively long window (e.g., 250 ms instead of 25 ms), which we hope will be able to further capture the temporal characteristics of these events.

Some related work oriented towards extracting a sparse subset of relevant points in an audio track for the purpose of classification can be found in [Kalinli *et al.*, 2009; Chu *et al.*, 2008]. Our work differs in a number of ways, including our application which is based around a set of 25 labels derived from a study with actual users [Loui *et al.*, 2007].

5.2 Proposed Algorithm

The following section details the processing stages of our algorithm. Figure 5.1 shows a block diagram of the system and example data.

5.2.1 Automatic Gain Control

A major problem in dealing with YouTube-style amateur video is the wide variation in background noise characteristics, recording equipment, and quality. Since our goal is to characterize individual transients according to their underlying cause, we would like to minimize the extent to which differences in recording conditions will result in irrelevant variability in the extracted features. We attempt to address this problem by applying automatic gain control (AGC) as a pre-processing

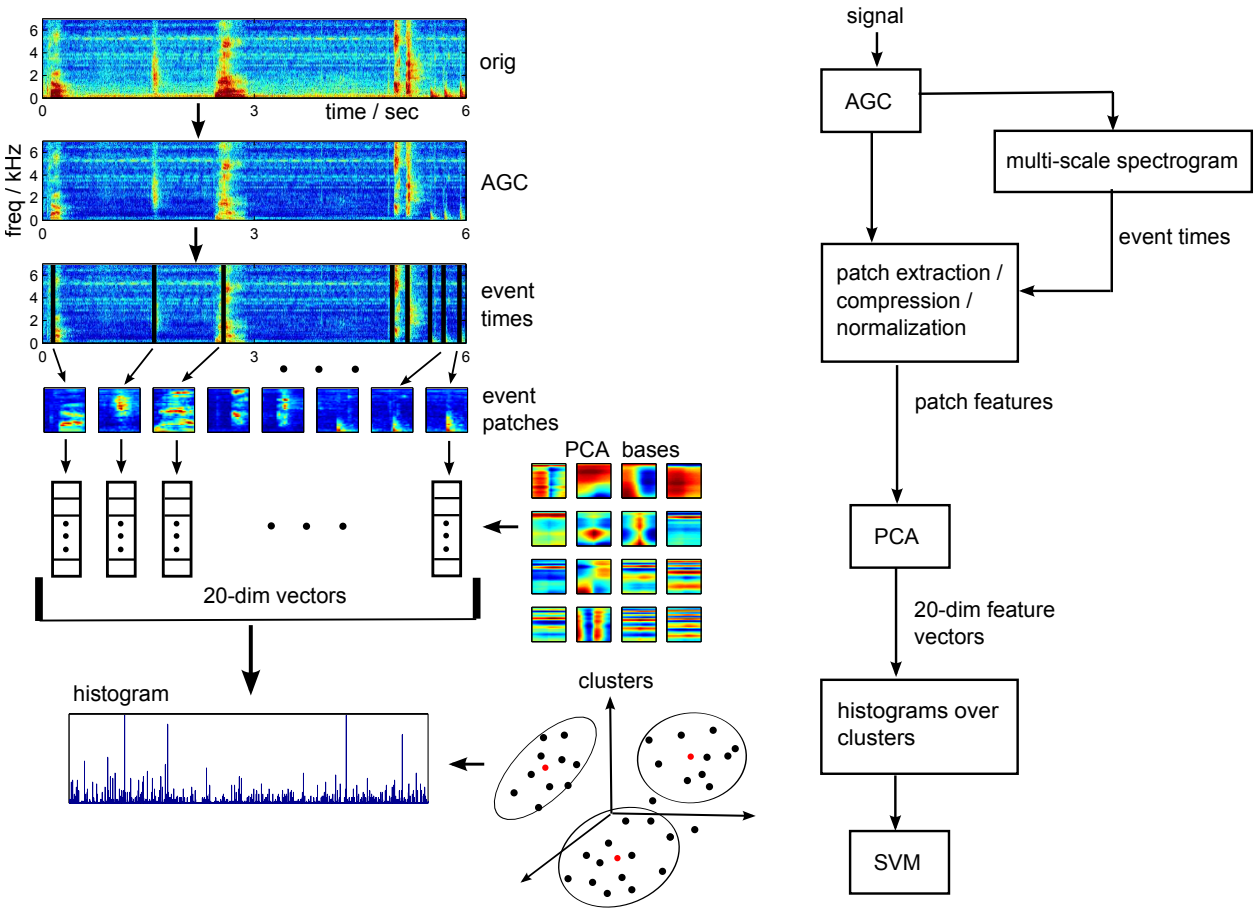

Figure 5.1: Block diagram of transient feature system, with examples of data.

step to our feature extraction process. In addition to reducing irrelevant variation, this stage can also make the subsequent transient detection more accurate.

Although many cameras and other recording devices have built-in audio AGC functionality, the AGC we discuss here is an additional process we apply to the recorded data. In recording devices the native AGC usually adapts fairly slowly (slow-release) because it is meant to be transparent and unnoticeable to a listener. In contrast, our AGC processing stage can adapt faster for better normalization since it does not need to be imperceptible to a listener. Our AGC also performs gain control separately in multiple audio bands, whereas a native AGC in a recording device would not.

Our automatic gain control equalizes the energy in both time and frequency by first converting the signal into an invertible short-time Fourier transform (STFT) using 32 ms windows. The magnitude of this representation is then smoothed across time and frequency, using a fixed time

window, and a frequency window defined in terms of an auditory frequency axis, leading to wider integration windows (in Hertz) for higher center frequencies. We use a mel frequency mapping. The local average energy obtained by this smoothing of the energy surface is then divided out of the STFT magnitude prior to inverting back to an audio waveform using overlap-add synthesis. The code for this AGC is available[1].

The AGC parameters were tuned for our task. We used symmetric non-causal smoothing with frequency integration on the order of 1 mel and temporal smoothing on the order of 4 seconds.

5.2.2 Transient Detection and Feature Extraction

After applying the AGC, the STFT (or spectrogram) of the signal is taken for a number of different time-frequency tradeoffs, corresponding to window lengths between 2 and 80 ms. We use multiple scales to be able to locate events of different durations. High-magnitude bins in any spectrogram result in a candidate transient event at the corresponding time. A threshold is set to some amount above the local (temporal) mean in each frequency band, and bins with values higher than this threshold are recorded. Additionally, a limit is set on the minimum distance between successive events. In this case, the overall system was tuned to produce an average of around 4 events per second.

For each event time, a short window of the signal is extracted centered on the event time. This window is 250 ms long in order to capture the temporal structure of the transient. For this short snippet, we again take the STFT, this time at a single scale of 25 ms with 10 ms hops, and integrate the frequency dimension into 40 mel-frequency bands. The result is an event patch consisting of 23 successive time frames, each consisting of 40 frequency bins. We restrict the spectrum to 7 kHz to compensate for differences in the high-frequency cutoff characteristics of different recording equipment, which would otherwise affect the comparisons between event patches.

Rather than take the log of a patch's magnitude values (as we would do if we were producing MFCCs), we raise the magnitude to a fractional exponent to compress larger values. This was determined empirically to perform better than taking the log. The specific exponent (0.2) was arrived at through tuning.

We finally normalize the patches by scaling the maximum value to be 1.

[1] `http://labrosa.ee.columbia.edu/matlab/tf_agc/`

5.2.3 PCA and Clustering

The resulting feature patches have 920 dimensions, which is too large to efficiently compare. We perform principal component analysis (PCA) on the training data, and use the top 20 bases to reduce the dimensionality down to 20 elements. Looking at these top basis elements in figure 5.1 we observe that some appear to describe sharp temporal onsets, as would be expected for transient-like sounds, while others seem to represent the contribution of stationary spectra to the event patches.

After representing the points with PCA, we then perform k-means clustering on the resulting 20-dimensional training data to arrive at a set of K clusters. Here, K is selected to be 1000. We store the means and covariances of these clusters.

5.2.4 Soundtrack Feature Representation

For each test video, we again extract patches around all event times detected by our algorithm. We then characterize the video as a histogram of its events as they are distributed over the K learned clusters. Initially, we took histograms using hard assignment of each event descriptor to a single cluster. However, we improved performance by distributing an event's weight proportionally amongst all clusters. Specifically, we assign weight to the histogram bins according to the posterior probability that each patch comes from each cluster according to a Gaussian distribution given the cluster's mean and covariance. Each video's histogram is normalized by the total number of events extracted from that clip.

5.2.5 Concept Detection

We use support vector machines (SVMs) to compare videos using their histogram features. We train one SVM per concept. The SVM's gram matrix is computed as the Mahalanobis distance between histogram vectors (with covariance estimated from the entire training set), and SVM parameters are tuned on a validation data set. To produce retrieval results for a given concept, test videos are ranked according to their decision value (margin) under that concept's SVM.

5.3 Experimental Results

The evaluation task is to retrieve videos in the order most relevant to each of the 25 concepts. We report average precision (AP) as the performance metric. Average precision is defined as the average over precision values evaluated at the depth of each true result in the ranked list.

The data used is a set of 1873 consumer videos downloaded from YouTube, and labeled with one or more of 25 semantic concepts, as described in [Lee and Ellis, 2010]. We use five-fold cross-validation in our experiments, where each fold of the data is divided randomly into 40% training, 20% validation, and 40% testing data.

We compare our results with a baseline approach in which MFCC features are extracted from every frame of the audio clip. The parameters used to extract MFCC's mirror those of our patch extraction stage: 25 ms frames with 10 ms hops, and 40 mel-frequency bands covering up to 7 kHz. Twenty-five coefficients are retained. Each clip's frames are modeled as a single Gaussian, where the clip's feature representation is the 25 means and 325 (unique) covariance values of that Gaussian. A set of SVM's are then trained on this feature set, again using the Mahalanobis distance between these statistical parameters to characterize the distance between clips. This follows the single Gaussian modeling procedure of [Lee and Ellis, 2010].

Lastly, we fuse the results from the two approaches. We do late fusion, wherein we add the normalized decision values from each SVM, using a weighting factor to trade off between the two decision values. The weight factor is optimized for each class, based on our expectation that the event-based system will be more effective at detecting some concepts and the global system will be more effective on others. The fusion weight used for each concept is tuned over the validation data.

Figure 5.2 shows average precision results over the 25 concepts and mean AP for all concepts for our algorithm, the MFCC model, and the fusion of the two. The AP that would result from guessing randomly is included for reference.

Table 5.1 shows mean AP values for the system as described above (original), and for some alternate parameter settings: without the AGC, with the event threshold adjusted to give an average of 2 events per second rather than 4, for larger and smaller values of K, and for 2 different settings of the patch compression exponent.

Figure 5.3 shows spectrograms of example event patches from three different clusters that are well-correlated with the labels 'baby', 'birthday', and 'playground', respectively. These clusters

were identified by computing the mutual information between all labels and clusters and examining the clusters with the highest mutual information for a given label.

Our original goal was to discover semantically meaningful types of transient sounds which would cluster well with similar sounds and be meaningfully related to the label categories. To some degree we were able to do this, identifying several clusters that are clearly indicative of a particular concept. For example, listening to examples from the 'baby' cluster in figure 5.3 reveals that they generally correspond to similar-sounding instances of people laughing, while the 'birthday' cluster contains mostly singing voices. Additional investigation of the clusters revealed some other obvious patterns, such as a cluster containing many fireworks explosion sounds, which was well-correlated with concepts like 'show' and 'parade'. Overall, however, we discovered somewhat less consistency in the clustered data than we had hoped for originally. Despite this, the clustering still seems to capture enough relevant information about the event patches to give statistical leverage that is useful in classifying the videos they come from.

5.4 Discussion and Conclusions

We demonstrate that focusing a soundtrack representation specifically on the subset of the signal indicated by transient events can improve concept retrieval performance over simply modeling MFCC frames globally for an entire audio clip. Our fusion of these two models achieves a 7.5% relative improvement in the mean average precision, from 0.38 to 0.41, over the global model alone. Although this improvement is not huge, the fused system achieves better average precision than either of the individual systems on 20 out of the 25 concepts. This result would be unlikely if the fusion was not a statistically significant improvement over the baseline.

The event model is especially helpful for predicting some semantic concepts (such as "playground" and "animal") and less useful for others (such as "cheer" and "ski"). This is reasonable since some of the concepts in this test set would be expected to have more distinct types of events associated with them than others.

In addition to improving overall retrieval performance with our fusion method, we achieve comparable performance to a global model with our method alone. This is promising because it allows the possibility of building a classification framework around these type of sound events. The

concept labels used in this task are a proxy for attempting to determine what is happening in a video. By building a concept detection system around events that also have some semantic meaning themselves, we can learn more about what is happening at an event level in the video. This has the potential to enhance search and retrieval capabilities for video based on the occurrence of specific audio events.

In the next chapter we attempt to improve upon the event detector described here by investigating a slightly different method of identifying spectrogram patches to describe events.

Parameter settings	MAP
original (AGC, 4 events/sec, K = 1000, comp. exp. = 0.2)	**0.385**
without AGC	0.332
2 events/sec	0.284
K = 500	0.378
K = 2000	0.375
patch compression exponent 0.125	0.375
patch compression exponent 0.4	0.335

Table 5.1: Mean AP results of transient system, for some alternate parameter settings

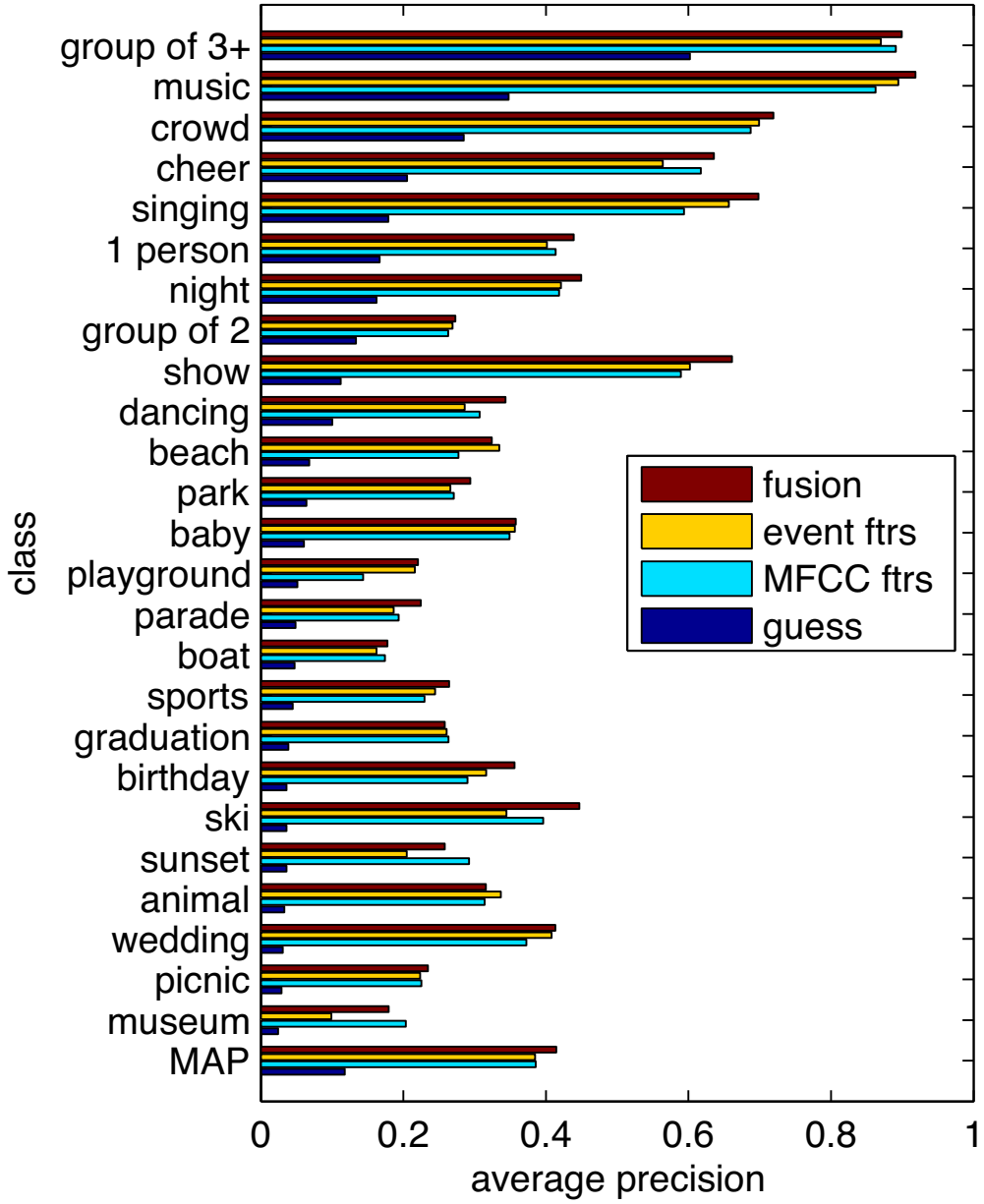

Figure 5.2: Average precision results for each class, and mean AP.

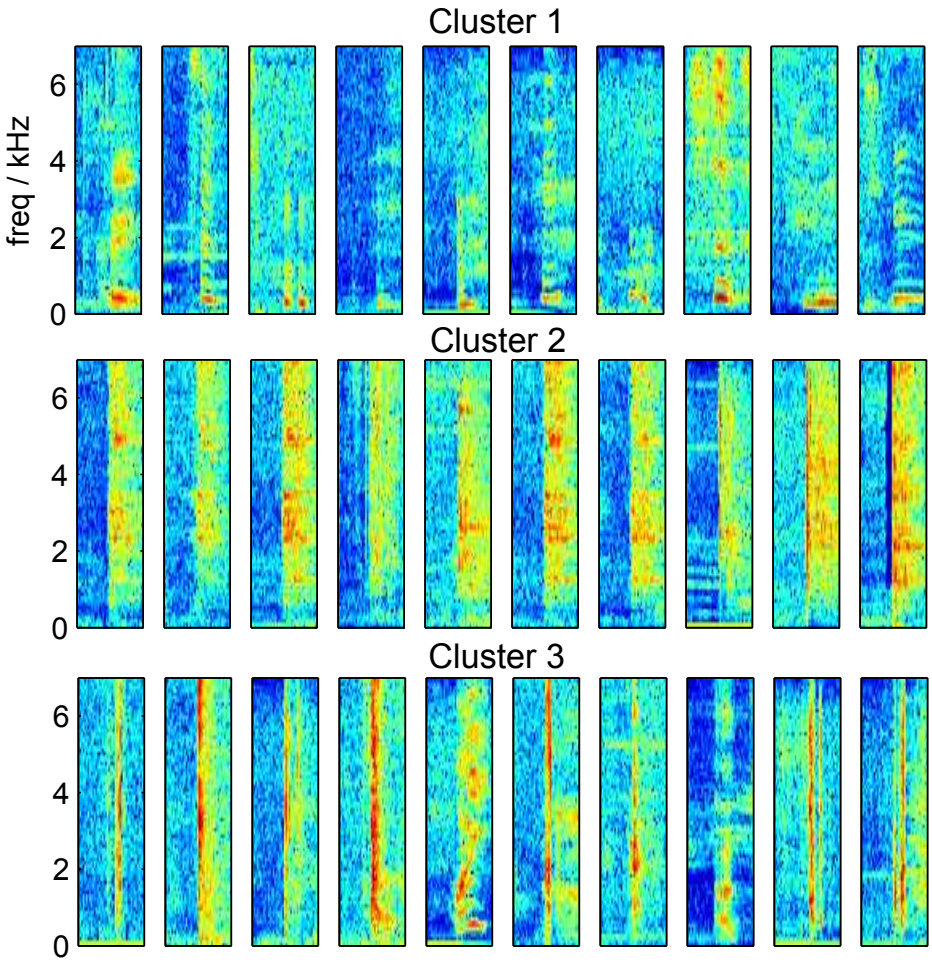

Figure 5.3: Examples of event patches from three clusters that are well-correlated with the 'baby', 'birthday', and 'playground' concepts, respectively.

Chapter 6

Spectral vs. Spectro-Temporal Features for Acoustic Event Detection

Automatic detection of acoustic events is an interesting problem on its own, with obvious relevance to our goal of interpreting soundtrack data. Typical approaches to the problem use short-term spectral features to describe the audio signal, with additional modeling on top to take temporal context into account. In this chapter we propose an approach to detecting and modeling acoustic events that directly describes temporal context, using convolutive non-negative matrix factorization (NMF). NMF is useful for finding parts-based decompositions of data; here it is used to discover a set of spectro-temporal patch bases that best describe the data, with the patches corresponding to event-like structures. We derive features from the activations of these patch bases, and perform event detection on a database consisting of 16 classes of meeting-room acoustic events. We compare our approach with a baseline using standard short-term mel frequency cepstral coefficient (MFCC) features. We demonstrate that the event-based system is more robust in the presence of added noise than the MFCC-based system, and that a combination of the two systems performs even better than either individually.

6.1 Introduction

Detection and classification of acoustic events is important in a number of applications. In particular, we are motivated to examine this problem in the context of automatically finding and tagging

events that occur in an unconstrained environmental audio stream, such as the soundtrack of a YouTube video.

Standard approaches to acoustic event detection utilize sequences of feature vectors that each capture spectral information over a very short time window, e.g., 30 ms. Additional modeling of temporal structure, for example a hidden Markov model (HMM), may then be used to reincorporate the temporal context that has been lost. In contrast, we are interested in taking advantage of temporal context directly when detecting and comparing acoustic events, rather than trying to describe events only by the statistics of their component frames. Intuitively, we would think of an acoustic event as something that is defined by both its spectral energy and its characteristic temporal shape.

In chapter 5 we explored an approach to discovering and comparing acoustic events with their temporal context. We did this by extracting spectro-temporal patches around "transient" points that exhibit a large increase in spectral energy. However, this method can be very sensitive to the parameters of the transient detector, and therefore have poor robustness to environmental factors. The extracted features also suffer because there is no separation between an event's energy and the background noise floor, which will also be captured in the surrounding spectro-temporal patch.

To address these problems, we present an algorithm based on non-negative matrix factorization (NMF). NMF is an algorithm for describing data as the product of a set of bases and a set of activations, both of which are non-negative; it was first described in its current form in [Lee and Seung, 1999]. It is useful for finding parts-based decompositions of data; since all components are non-negative, each basis contributes only additively to the whole. This promotes a solution in which high-energy foreground events and constant low-level background energy may be described by different bases, and therefore separated in the feature representation.

Most applications of NMF to audio processing decompose spectral magnitude frames (e.g., columns of a spectrogram), and have each NMF basis consist of a single short time frame [Smaragdis and Brown, 2003]. Since we are interested in learning bases that correspond to entire events, we use the convolutive formulation of NMF [O'Grady and Pearlmutter, 2006; Smaragdis, 2007]. In this version, bases consist of spectro-temporal patches of a number of spectral frames stacked together. The pattern described by each frame is then activated as a whole to contribute to the reconstruction of the data. Additionally we would like to ensure some level of sparsity in the activations of these

bases. This is in order to encourage the bases to learn more foreground event patterns and fewer patterns that mimic the background, which would be activated non-sparsely over large segments of the data.

This NMF algorithm allows us both to locate transients in time, and to learn a dictionary of event-patch codewords, within a single optimization framework, avoiding the separate transient detection and patch clustering of our previous approach.

Figure 6.1 demonstrates a convolutive NMF decomposition using three bases on a piece of toy data containing two different audio events repeated three times each, with background noise throughout. (In this case, the sparsity constraint on basis 1 has been relaxed, which is why it has been used to represent some of the background noise, while bases 2 and 3 have learned fairly good approximations of the two foreground events.)

6.2 Event Detection with Convolutive NMF

Our algorithm for the detection of acoustic events is as follows. We downsample all data to 12 kHz, and take a standard STFT or spectrogram of the entire signal, using 32 ms frames and 16 ms hops. We warp the frequency axis to mel-frequency as though we were taking MFCC features, using 30 mel-frequency bands from 0 to 6 kHz. We then concatenate all training data spectrograms and perform convolutive NMF across the entire set of training data, using 20 basis patches which are each 32 frames wide (approximately 500 ms). This produces a set of basis patches W and a set of basis activations H. The number of bases was tuned for performance; this also seems like a reasonable number to use since we are modeling data containing 16 classes of acoustic events, as described in section 6.5. In our experiments we use the implementation of sparse convolutive NMF described in [O'Grady and Pearlmutter, 2006], with KL divergence as the objective and a sparsity parameter $\lambda = 1$. For testing, we then use the fixed set of learned patch bases and perform NMF to find the corresponding activation values for the test files. An example set of 20 patch bases learned from training data is shown in figure 6.3. Figure 6.4 shows an example event ('door knock') and the activation patterns of the three bases that contributed the most energy in representing it. It also shows the event as reconstructed by NMF using the full set of bases. Note that the activations appear to occur somewhat before the onset of energy; this is because the patch activation is placed

at the left-hand edge of the 500 ms patch.

We believe the NMF algorithm captures a relevant set of event-like patches, but we still need a reasonable way to represent the (continuous) activation patterns of these bases as discrete event-like features. In order to do this, we use a sliding window of 1 s, with hops of 250 ms. Within this window, we summarize the local basis activation pattern by taking the log of the maximum of each activation dimension, producing a set of 20 features per window. These activation values are pre-normalized such that each basis has a maximum activation of 1 over the entire dataset.

In order to perform event detection with these features, we use a simple HMM. The dataset we use for evaluation (described in section 6.5) labels specific time intervals as containing acoustic events of a given class. Our HMM consists of a single state for each of the 16 classes and a 17th state for the background. The observation matrix is trained using the interval labels of the training data, with the simple assumption that the observations for each event class can be modeled as a single Gaussian. The transition matrix is trained on the stream of labels, which in practice prohibits direct transitions between two classes other than background (since the training data has no overlap or adjacency between different classes). The topology of the HMM used can be seen in figure 6.2. A stream of predicted labels is produced for each test file and scored as explained in section 6.6. Finally, in order to produce a reliable event stream, events shorter than 6 frames are removed.

6.3 Baseline MFCC Event Detector

In order to evaluate our algorithm, we compare it with a baseline which has a similar HMM structure, but that employs a standard set of short-term MFCC features. We extract MFCC features from all data (25 ms frames with 10 ms hops, 40 mel-frequency bands) and retain 25 coefficients as features. We feed this into an HMM, trained in the same way described above. Again, we use a single state for each class, plus a background state, and observations are modeled as a single Gaussian for each class. We generate a series of predicted labels at the frame level, as above.

Because the MFCC frame spacing is much shorter than the NMF system's frame spacing (10 ms vs. 250 ms), we need to post-process the predicted label stream to evaluate it fairly. We first

use a median filter across 250 frames (2.5 sec), and then we remove any remaining events that are less than 100 frames long. Both these parameters were selected to optimize performance on clean test data.

6.4 Combined System

Since we have built two event detectors based on different sets of features, we were interested to see if they could be combined in a complementary way to produce better performance. We did this by taking the two predicted event streams and requiring that they both agree on a predicted event label for some overlapping period of time. If this is true, then we consider that predicted event to extend to the entire period of time that either system has predicted the event. This yields a combined prediction event stream that can be evaluated alongside the two individual systems. Requiring both component systems to agree tends to reduce insertions while increasing deletions; however, it turns out that insertions were the dominant problem in noise, so this approach can be beneficial.

6.5 Database

In order to focus on the task of detecting specific acoustic events other than speech, we tested our approaches on the FBK-Irst database of isolated meeting-room acoustic events [CHIL, 2008]. This is a dataset which was originally collected under the CHIL (Computer in the Human Interaction Loop) project. Event detection and classification using data of this type has been extensively examined by Temko and others [Temko, 2007].

The data we used consisted of 9 sessions, each around 7 minutes long. Each session was recorded by multiple microphones, although we only use one channel in our experiments. This is because we would like to develop algorithms that will also work on less controlled data, such as video soundtracks which would only have one or two channels available.

The database contains 16 semantic classes of acoustic events: door knock; door open; door slam; steps; chair moving; cough; paper wrapping; falling object; laugh; keyboard clicking; key jingle; spoon, cup jingle; phone ring; phone vibration; MIMIO pen buzz; and applause. Each session contains around 4 repetitions of each of the 16 classes of events, so there are around 36 examples

of each event in the database. Approximately 50 repetitions per event class were recorded.

The data labels consist of short intervals that contain instances of the labeled sound (see figure 2.1). Consecutive events of different classes do not overlap with each other in this dataset.

In our experiments, we use two folds of the data. In each fold, the data split is 6 training files and 3 test files. For efficiency reasons, in the NMF algorithm only 3 of the training files are used to actually learn the patch bases; the remaining 3 are added back in and used to train the HMM and learn the observation distributions.

6.6 Experiment and Metric

We are interested in examining the ability of our NMF algorithm to discover acoustic events in the quiet meeting room environment in which this data was recorded, but also in the midst of noisy environments. Our hope is that the additive nature of the NMF algorithm will allow it to represent acoustic events that occur in noise more consistently than standard MFCC features, which will be corrupted by added noise.

In order to test this idea, we performed all experiments with varying levels of additive noise. The noise added was a short clip of background chatter and activity recorded in a cafeteria. For each noise level, the test data was left clean while this noise clip was added to the training data at the specified SNR.

For evaluation we use the acoustic event error rate (AEER) that was used in CHIL evaluations for the event detection task, as described in [Temko, 2007]. This is defined as: $AEER = 100(D + I + S)/N$, where D is the number of deletions, I the number of insertions, S the number of substitutions, and N the total number of events that occur in the ground truth labels.

To evaluate a stream of predicted labels, it is broken into predicted events. A predicted event is any string of consecutive frames with the same label. If the center of a predicted event (of the correct class) falls anywhere within the true event's label interval, then it is considered a correctly predicted event. Any predicted event that does not fall within a true event of the same class is considered an insertion or substitution; we count these errors together. Any true event that does not have a (correctly) predicted event fall within it is considered a deletion.

6.7 Results

Figure 6.5 shows the performance of the three algorithms under the AEER metric. Table 6.1 breaks these results down into deletions, insertions, and again the overall AEER for each algorithm. Each system has been tuned (by balancing insertions and deletions) to optimize its performance at 30 dB SNR (nearly clean noise conditions). In clean conditions, the MFCC-based system performs much better than the event-based one, and about the same as the combination of the two.

We then examine how each system breaks down in the presence of added noise. All three systems produce deletions at a roughly comparable rate as noise increases. The MFCC-based system produces a large number of insertions in high noise conditions, while the NMF-based system largely does not (insertion and deletion rates for NMF stay roughly balanced). The combination system achieves even better performance as the noise level increases. This is mostly true because it is limiting the number of insertions by requiring that the two systems agree on events.

		SNR (dB)				
		10	15	20	25	30
Deletions	MFCC	41	28	17	13	13
	NMF	55	40	32	26	22
	Combined	54	41	27	21	18
Insertions	MFCC	133	105	91	29	6
	NMF	43	50	29	22	18
	Combined	33	32	15	9	3
AEER	MFCC	282	216	175	68	30
	NMF	158	147	100	77	64
	Combined	140	118	69	48	34

Table 6.1: Average number of deletions and insertions contributing to AEER.

6.8 Discussion and Conclusions

Despite the relative crudeness of our NMF-based features, we demonstrate that these type of large-scale event features can be usable in the detection and classification of acoustic events. Although

our system is not competitive with a conventional short-frame-based system in clean conditions, it proves useful when the test data is even slightly more noisy than the training data. Features based on NMF basis activations seem to be fairly robust under moderate noise conditions (i.e. both systems using NMF features do not degrade much between 30 and 20 dB SNR). The MFCC-based system, on the other hand, performs much more poorly under moderate noise conditions. Presumably this is because the MFCC features are being corrupted by background noise while the NMF-based system is allowing prominent events to be represented by the same bases as they would have in the clean test data. This would therefore yield feature descriptions that are theoretically more constant as the background noise increases.

Since our interest in event detection extends to varied types of data and recording conditions, it is important for an algorithm to be able to detect similar events that occur in the midst of different types of noise. Event modeling based on convolutive NMF bases seems promising for developing noise-robust systems of the type necessary to detect acoustic events in all types of unconstrained videos and other audio data.

Ideally we would like to use the convolutional NMF algorithm explored here on the large video corpuses described in chapters 5 and 7. The only limitation to doing so is that convolutional NMF is rather computationally expensive. For example, in these experiments the training procedure (learning both bases W and activations H) was performed on only 3 of the 9 files (20 minutes of data) and took around 5 hours to learn 20 bases (on the order of 15x real time). Also, this computational expense does not scale linearly with more data, due to memory constraints. Additionally in a larger scale experiment more than 20 bases would likely be used, adding to the training time as well. All of this presents a challenge to using convolutional NMF -based algorithms on large datasets, although it is probable that improvements in computer processing power will make this work feasible in the future.

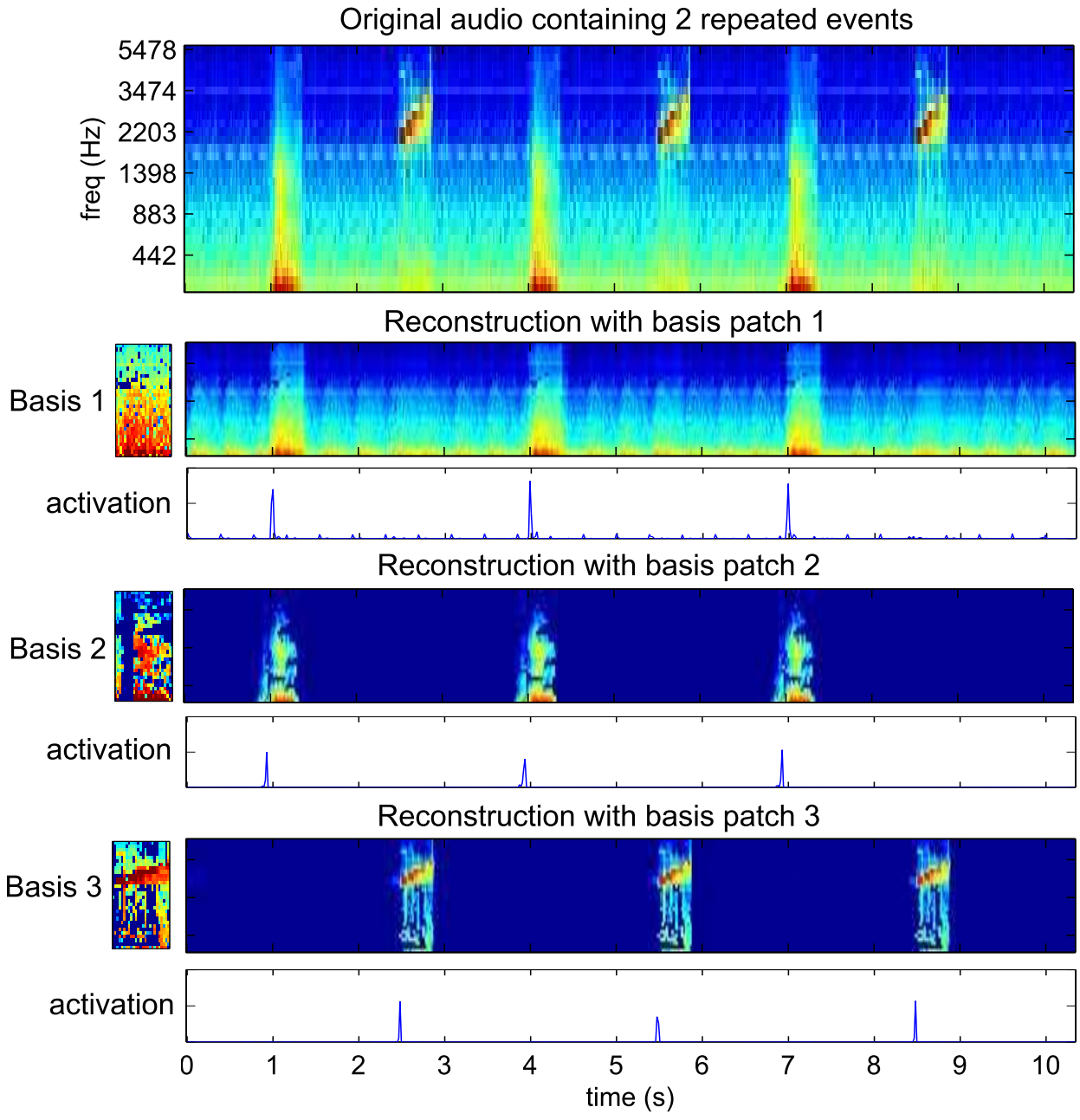

Figure 6.1: Convolutive NMF decomposition example showing three basis patches and their corresponding activations, as learned from a piece of toy data.

Figure 6.2: Topology of the learned HMM.

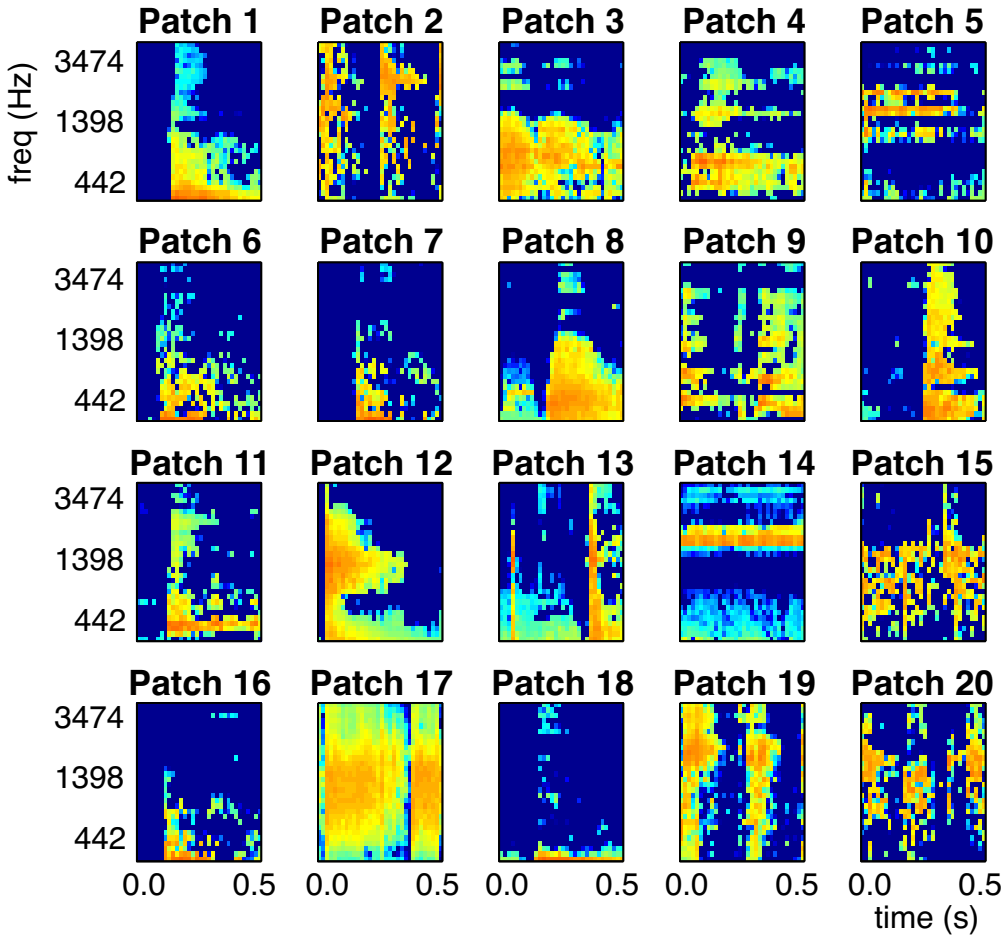

Figure 6.3: 20 NMF patch bases learned on training data (20 minutes of meeting room sound event recordings).

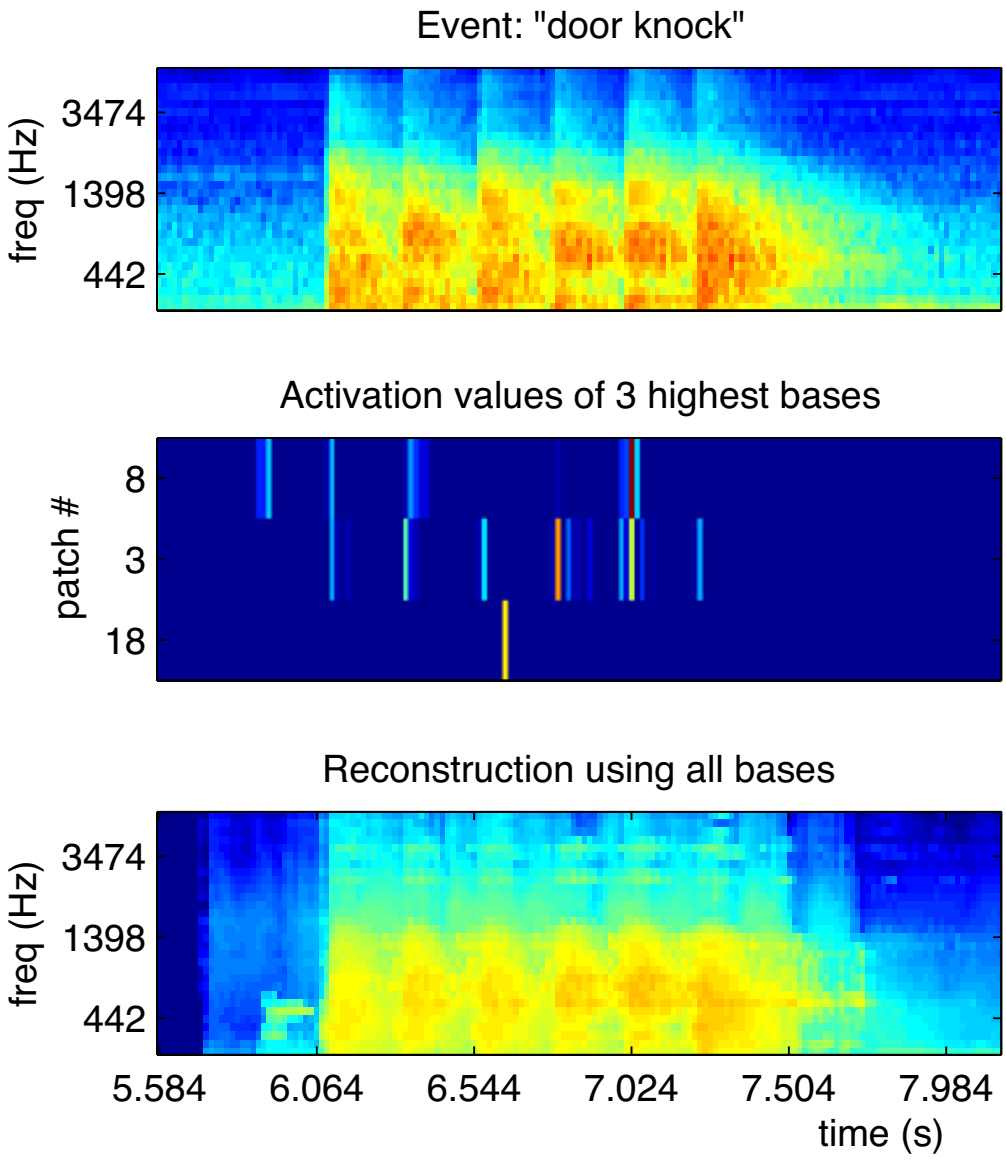

Figure 6.4: Example of a 'door knock', the top 3 bases used to represent it, and a reconstruction of the event.

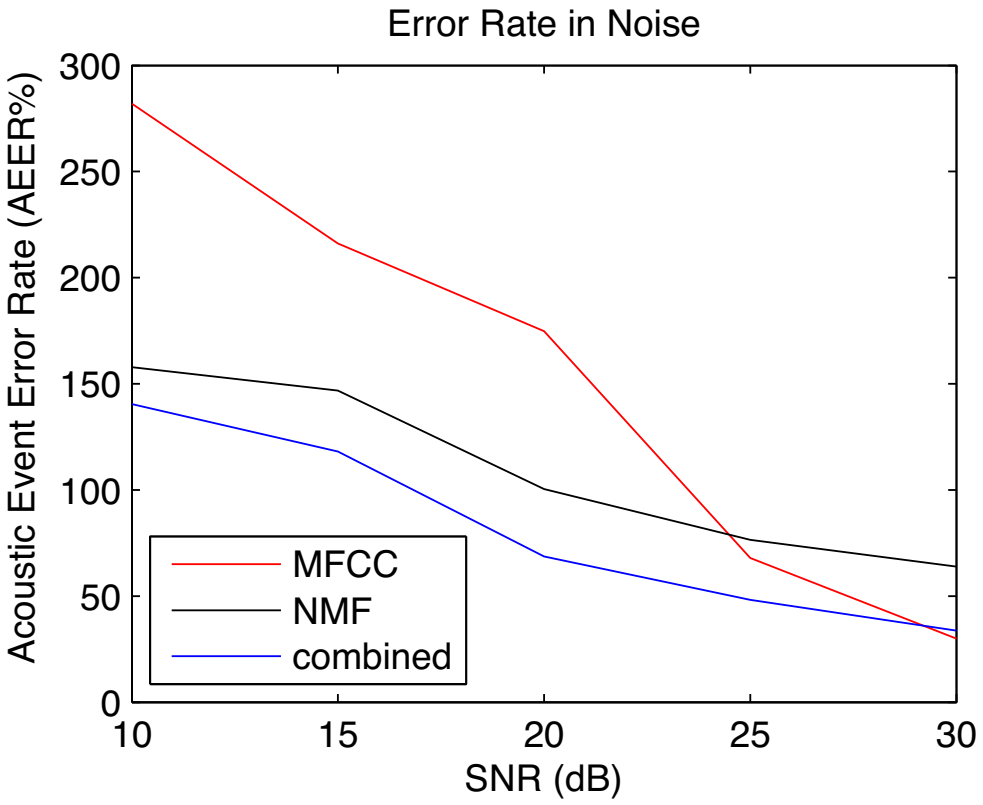

Figure 6.5: Acoustic event error rate results in noise.

Chapter 7

Subband Autocorrelation Features for Video Soundtrack Classification

In this chapter we investigate another set of features for use in video classification. Inspired by the system presented in [Lyon *et al.*, 2010], we develop novel auditory-model-based features that preserve the fine time structure lost in conventional frame-based features. While the original auditory model is computationally intense, we present a simpler system that runs about ten times faster but achieves equivalent performance. We use these features for video soundtrack classification with the Columbia Consumer Video dataset, showing that the new features alone are roughly comparable to traditional MFCCs, but combining classifiers based on both features achieves a substantial mean Average Precision improvement of 18% over the MFCC baseline.

7.1 Introduction

As the means to collect and share video and audio become increasingly ubiquitous and cheap, tagging and retrieval of multimedia content become increasingly important. Although much of this work focuses on the visual content of a video, modeling the audio content can also prove helpful for the purpose of search and indexing. A standard approach to characterizing audio content uses mel-frequency cepstral coefficients (MFCCs), which are short-time spectral features. There are on-going efforts to identify other useful features in this domain and novel methods for employing them in retrieval tasks.

In [Lyon *et al.*, 2010; Lyon *et al.*, 2011], features based on an auditory model were presented for use in audio recognition (specifically of soundtrack clips). In contrast to traditional features which average the signal spectrum over 20-30 ms windows, the auditory model features attempt to preserve the fine temporal structure of the sound via a "stabilized image" of the waveform. These features were used in conjunction with the "passive-aggressive" model for image retrieval (PAMIR) as the learning mechanism. The authors showed that these features performed as well as or better than traditional MFCC features for retrieval tasks, and that they are particularly useful for the identification of sounds in mixtures. Since we are working with broadly similar problems of classifying unconstrained environmental audio, we investigated this system. We began by attempting to replicate their system as closely as possible and test it on a retrieval task on a corpus of tagged consumer video soundtracks.

The next sections introduce our data/domain, and then describe our results using an available implementation of the auditory model front end, both with the original PAMIR retrieval model, and with more conventional Support Vector Machine (SVM) classifications. Sections 7.5 and 7.6 describe our modifications to the original system to reduce the dimensionality of the representation, and to simplify the overall calculation to reduce its computational burden. Section 7.7 describes the further improvements we obtained by fusing these novel features with the existing baseline MFCCs.

7.2 Dataset and Task

We performed all evaluations on the Columbia Consumer Video (CCV) dataset described in [Jiang *et al.*, 2011]. This is a set of 9,317 video clips from YouTube, comprising 210 hours of video. The clips are tagged with 20 semantic categories. For all our experiments, the metric used was average precision of retrieval results for each category, with the mean average precision (mAP) over all categories serving as the main objective index of performance.

7.3 Stabilized Auditory Image Features

The system of [Lyon *et al.*, 2010; Lyon *et al.*, 2011] has a multi-step feature generation process. First the signal is passed through a time-varying filterbank intended to model the cochlea, including its

local loudness adaptation (through changes in individual filter resonance). The filterbank outputs are then integrated using what the authors call strobed temporal integration. Strobe (peak) points are identified, and the signal is cross-correlated with a sparse function that is zero except at these strobe points. This is done separately in each filter channel, resulting in a two-dimensional (number of channels $\times$ time lag) image, termed the stabilized auditory image (SAI). (In lieu of a more detailed description, please see the presentation of our simplified auditory model features in section 7.6). In their experiments an SAI is generated every 20 ms to characterize the audio signal at that point. Then a sequence of SAIs is converted into features using a sparse code representation as follows: Each SAI is overlaid with a set of rectangular patches of different sizes. Each of these rectangles defines a local region of interest on the SAI. The set of rectangle features is collected over all data, and each rectangle region is vector quantized (VQ) separately. A single SAI is then represented by a sparse code whose dimensionality is the number of rectangles times the size of each VQ codebook. An audio clip is represented as the sum of its SAI codes (essentially, a set of histograms).

To approximate this system, we used a publicly-available C++ codebase, AIM-C [Walters, 2010], that computes stabilized auditory images that are similar though not completely identical to those described in [Lyon *et al.*, 2010]. The audio data is first downsampled to 16 kHz and processed with AIM-C to produce a series of SAIs. The SAIs were then cut into 24 rectangles, using the box-cutting method described in [Lyon *et al.*, 2010], where the smallest boxes were 32 frequency channels by 16 time steps. Each dimension was then doubled systematically until the edges of the SAI were reached. Figure 7.1 shows an example SAI with the 24 rectangles overlaid in red. Specifically, the rectangles emcompass four frequency ranges (channels 1 to 32, 17 to 48, 33 to 64, and 1 to 64), and six timescales at each frequency range. Figure 7.2 shows the contents of the resulting 24 rectangles. We then downsampled and quantized the margin features (sums of values in each of the rectangle's two dimensions) for each of the 24 rectangles with a 1000-codeword dictionary learned by k-means on the training set. This leads to a representation of each video clip as a sparse 24,000-element vector which is essentially the concatenation of the histograms over each of the 24 rectangles.

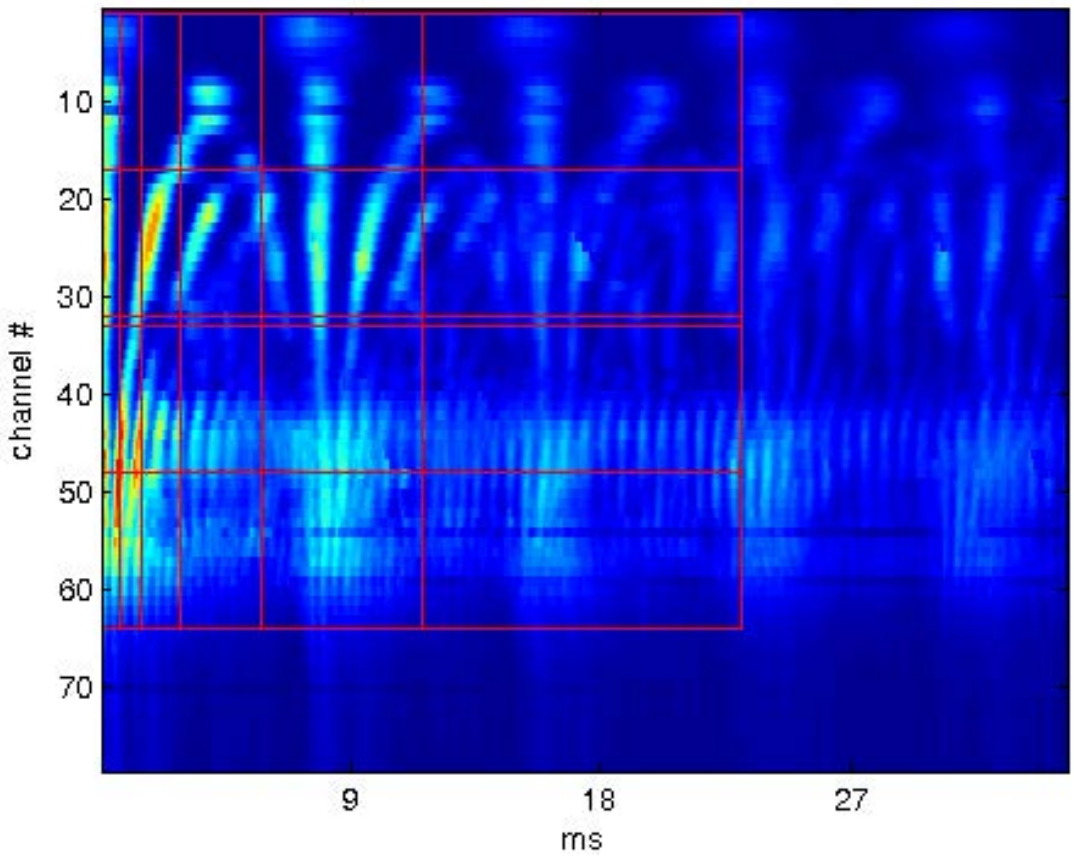

Figure 7.1: Example SAI with 24 rectangles overlaid in red.

7.4 PAMIR versus SVM Learning

As in [Lyon *et al.*, 2010], we initially used PAMIR as the learning method in our system. PAMIR is an algorithm for learning a linear mapping between input feature vectors and output classes or tags. PAMIR is especially efficient to use on sparse feature vectors (such as the extremely high dimensional histograms described above), which is one reason the authors chose it.

We were unable to get particularly good performance from PAMIR. PAMIR is theoretically useful for learning associations reasonably quickly when the scale of the data is very large. However, in reality our experiments consisting of thousands, but not millions, of data items, were not large scale enough to necessitate the use of PAMIR. We realized that we could obtain better results by combining SAI features with more standard learning techniques such as support vector machines

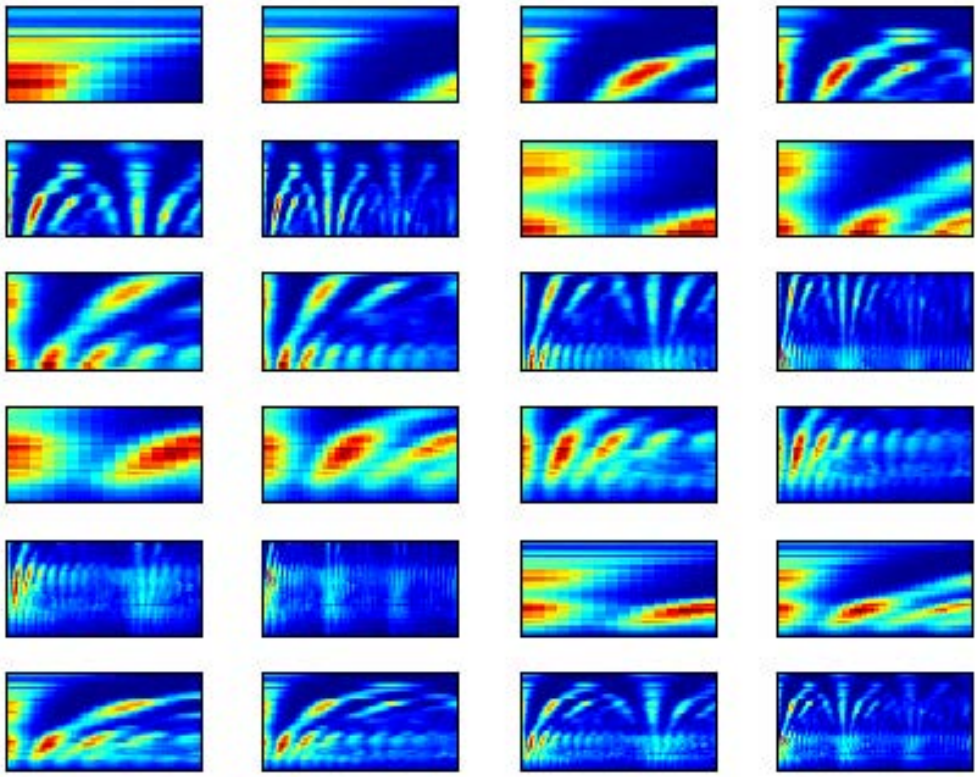

Figure 7.2: 24 rectangles extracted from an example SAI.

(SVMs).

Figure 7.3 compares the performance of SAI features using PAMIR and SVM learning techniques. As in [Lyon *et al.*, 2010], we compare the novel SAI features with a baseline system using standard MFCC features. Here, we used 20 MFCC coefficients and also added deltas and double deltas, for 60 dimensional feature vectors. For consistency with the SAI features, MFCC frames were vector quantized and collected into a single 3000-codeword histogram representation for each video clip. Figure 7.3 also shows results using these MFCC features with both learning methods. In our experiments, SVM learning significantly outperforms PAMIR on both feature sets. SAI and MFCC features perform roughly comparably to each other although MFCCs perform slightly better under both learning methods.

7.5 Reduction of Feature Set Size

We were interested in investigating how the set of rectangle features selected influenced the final results. Specifically we wondered to what extent we could minimize the number of rectangles (in order to reduce feature vector size) while retaining a similar level of performance. The authors of [Lyon *et al.*, 2010] experimented with numerous rectangle cutting strategies but did not offer strong conclusions about the extent to which larger numbers of rectangles can lead to improved performance. Since their cutting method results in rectangles that overlap, there is presumably some duplicate information. Our goal was to see if we could minimize the number of rectangles while maintaining high performance.

We experimented with reducing the set of rectangles in various ways. The original set of 24 rectangles consists of rectangles covering four different frequency ranges (low frequency, high frequency, mid frequency overlapping both low and high, and all frequency bands together), at each of six timescales (where each timescale is twice as long as the previous one). We were able to achieve performance very close to the full set using only eight rectangles. Specifically, we removed all rectangles from the mid frequency (channels 17 to 48) and full frequency (channels 1 to 64) ranges, keeping only high and low frequency rectangles (channels 1 to 32, and 33 to 64). We also removed the largest two timescales, keeping only the shortest four. Figure 7.4 compares the SAI and SVM system using all 24 rectangle features (SAI) with only these eight rectangle features (SAI reduced), demonstrating that performance remains very similar between the two. The figure also includes our reformulated auditory model features, described below.

7.6 Subband PCA Features

Even using a reduced number of rectangles in the final quantization, the calculation of SAI features is a relatively slow process. Since our target application is for very large multimedia archive (up to thousands of hours), we wanted to see if we could retain the performance of this type of feature but with simpler processing. The goal was to identify a simpler feature set that could capture information similar to the cross-correlations of the strobed temporal integration process used to produce SAIs. We decided to try a set of features based on subband autocorrelations. These features were based on earlier work in pitch tracking by our colleague Lee [Lee and Ellis, 2012]

and consist of the first ten coefficients obtained from principal component analysis (PCA) on the normalized autocorrelations in each of 24 frequency subbands spanning center frequencies from 100 Hz to 1600 Hz with six bands per octave, and a quality factor ($Q = \frac{centerfrequency}{bandwidth}$) of 8. Like the SAI features, we hoped these would capture some of the fine temporal structure not typically captured in traditional MFCC features. Unlike SAIs, the filterbank is time-invariant, and the correlation does not depend on any strobe instant selection. Analogously to the SAI rectangle features, we divided the 24 subbands into 4 (non-overlapping) frequency ranges, and vector quantized each of these 10 $\times$ 6 subband coefficient feature sets into 1000 codewords, for a total of 4000 dimensions. Figure 7.5 illustrates the entire calculation process for these features.

Figure 7.4 also includes the performance of this subband PCA (SBPCA) feature set compared to the full- and reduced-dimensionality SAI features. Although the SBPCA features show a slight drop in performance, they perform nearly as well as the SAI features.

Significantly, calculating and training the system with SBPCA features is much faster than using the SAI features. Both feature sets are computed using reasonably optimized compiled C++ code, but SAI features can take on the order of 5$\times$ longer than real time to calculate. In contrast, SBPCA features can be calculated in less than 1.5$\times$ real time. Especially when working with large amounts of data, this difference is enormous.

To give a clearer picture, table 7.1 summarizes the factors that contribute to computational time for each of the three systems considered here (MFCC, SAI, and SBPCA). The 'feature extraction' time gives an estimate of the total length of time it would take to process the 9317 videos (210 hours of data) in the CCV corpus, using a single processor core on one of our machines. The time it takes to learn k-means codebooks and compute histograms over them is a function of the number and size of the codebooks and (to a lesser extent) the dimensionality of the data points; these factors are listed for each feature set. Finally, the SVM training time is primarily a function of the dimensionality of the final feature vector used to characterize each video, since a distance matrix must be computed to create the kernel for the SVM; this dimensionality is listed in the table. The distance matrix calculation takes a non-trivial smount of time, especially when using the chi-square distance as we do here (chi-square typically works well for computing distances between histograms but is quite slow to compute). Cumulatively, considering the raw feature extraction time as well as the larger set of codebooks involved, the SAI system can end up being around an order of magnitude

slower than the SBPCA system.

Processing Step	MFCC	SAI (reduced SAI)	SBPCA
feature extraction	5.6 hrs for 210 hrs data	1087 hrs for 210 hrs data	310 hrs for 210 hrs data
k-means codebooks and histogram calc.	1 3000-word book ftr dim = 60	24 (or 8) 1000-word books ftr dim = 48	4 1000-word books ftr dim = 60
svm kernel matrix calculation	3000 dim	24000 dim (or 8000)	4000 dim

Table 7.1: Table of compution times and factors for each type of feature: MFCC, SAI, SBPCA.

7.7 Improvement with Classifier Fusion

At this point we have developed two sets of features that perform relatively comparably with traditional MFCC features, but are based on very different processing chains. In the past we have observed that feature sets capturing diverse information about the data will combine in a complementary way to produce a noticeable performance improvement. We therefore tried the same approach here, and used margin fusion (adding together the output decision value of each SVM classifier) to create classifiers based on different feature sets. We combined each of the three single feature classifiers (MFCC, SAI, SBPCA) with each other and also tried the combination of all three classifiers. Figure 7.6 shows the performance of the three individual systems and the four combinations. Adding either SBPCA or SAI features to MFCCs gives a very substantial increase in mAP, with SBPCA features slightly better than SAIs. The baseline mAP performance of 0.34 for MFCCs alone improves to 0.40 in combination with SBPCAs, a relative improvement of around 18%. Adding SAI and SBPCA features performs better than either individually, but not as well as the combinations with MFCCs. Combining the margins of all three classifiers performs the best, with an mAP of 0.42, a 24% relative improvement over the MFCC baseline.

7.8 Discussion and Conclusions

In the course of these experiments, we investigated a number of different approaches to video soundtrack classification. We draw several conclusions. Initially, we verified that SAI features perform well for audio classification, although they did not actually outperform traditional MFCC features in our scenario (which is significantly different from the isolated sounds used by [Lyon *et al.*, 2010; Lyon *et al.*, 2011]). We observed that a standard machine learning technique (SVMs) significantly outperformed the PAMIR approach (although PAMIR may prove more useful on very large amounts of data where SVMs are infeasible). We demonstrated that the SAI feature dimensionality can be reduced significantly without significantly lowering performance. We discovered that a novel feature set, SBPCA, compares favorably with SAI features but with significantly less processing overhead. Finally, we demonstrated that both SAI and SBPCA features can be combined with MFCC features for an overall performance improvement that is considerably better than our previous MFCC baseline. The combination of all three features performs better still. Since SBPCA features are reasonably fast to calculate (at least relative to SAIs), we believe that they are a promising direction to investigate for capturing information from fine temporal structure that is excluded from traditional feature. We believe this can significantly improve the performance of future audio classifier systems, especially when used in conjunction with more traditional features.

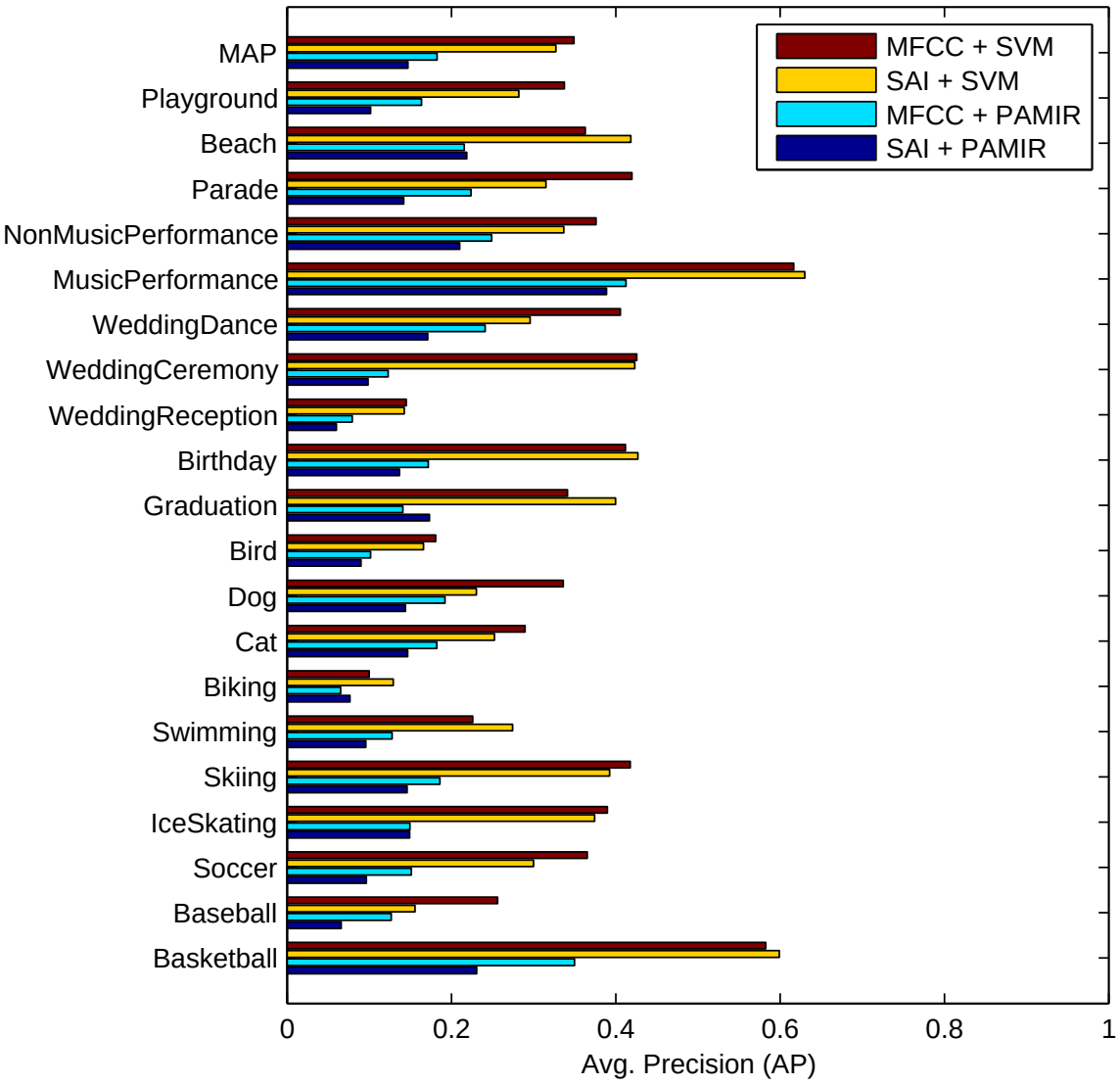

Figure 7.3: Baseline system comparisons: MFCC and SAI features, in conjunction with both PAMIR and SVM learning methods.

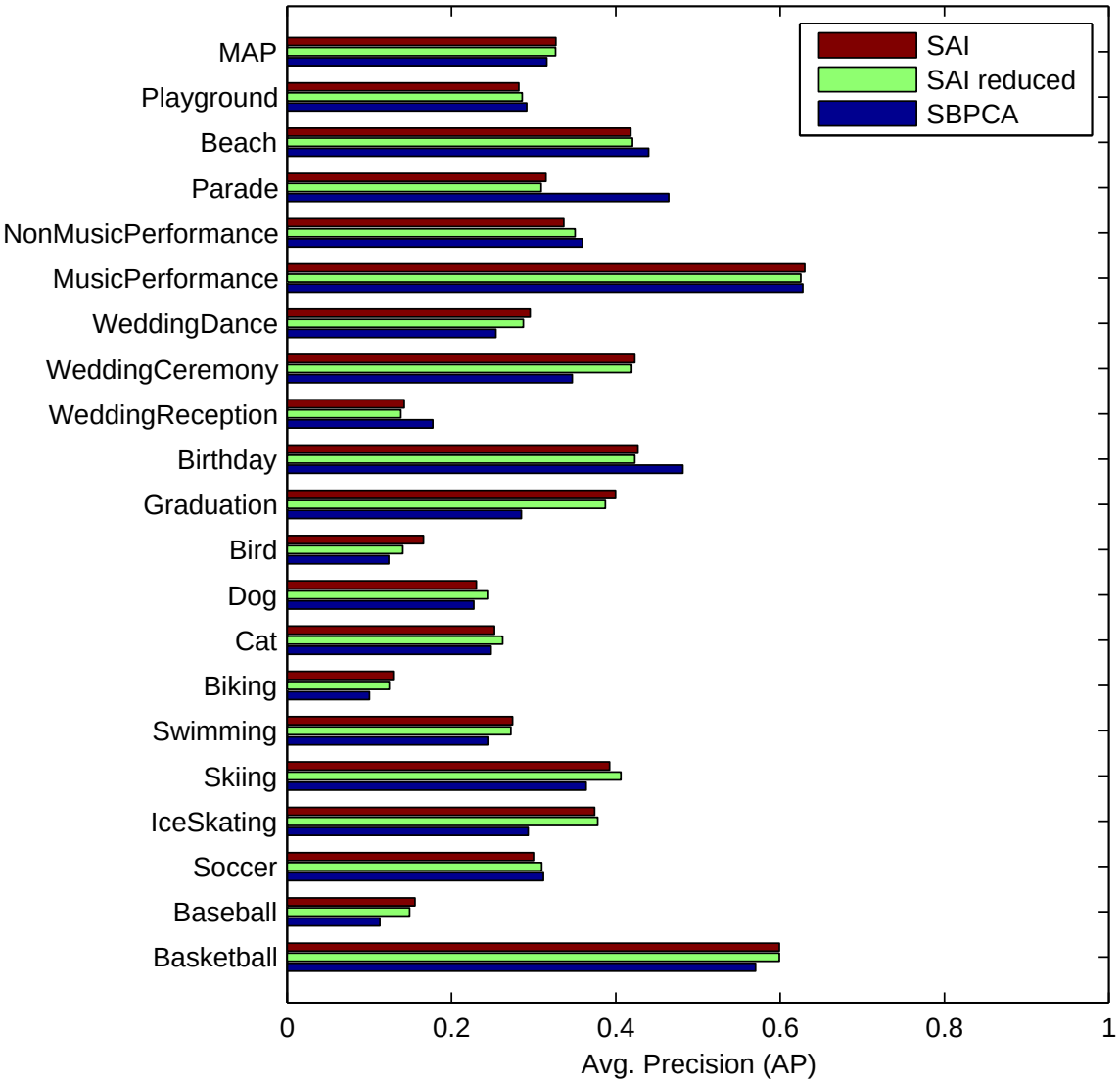

Figure 7.4: Comparison of SVM systems using: full set of 24 SAI rectangles (24,000 dimensions), a reduced set of 8 SAI rectangles (8,000 dimensions), and the simpler SBPCA features (4,000 dimensions).

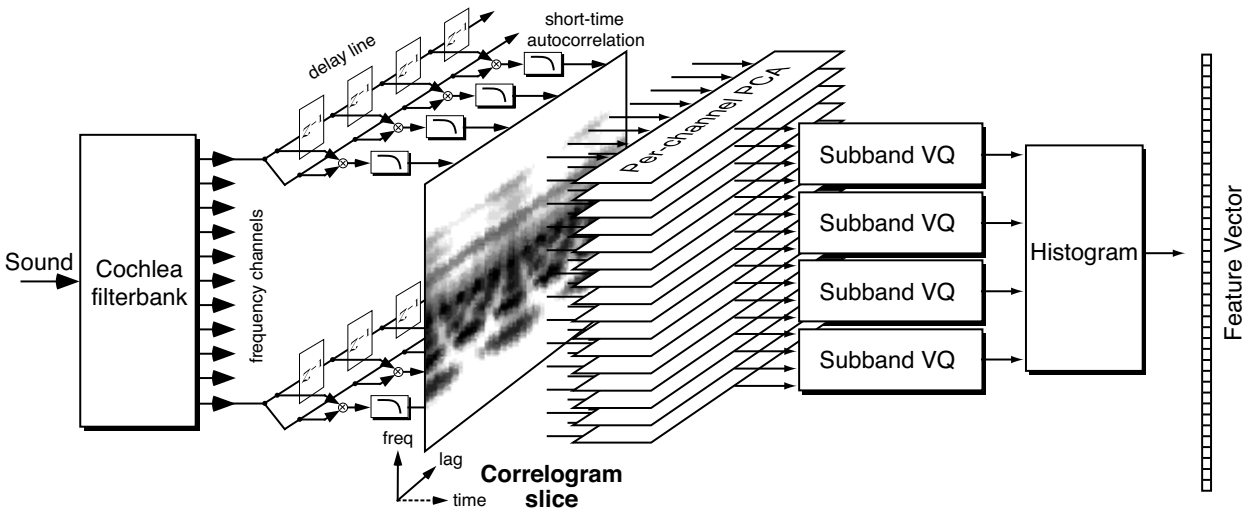

Figure 7.5: Block diagram of the calculation of the subband autocorrelation PCA feature vectors.

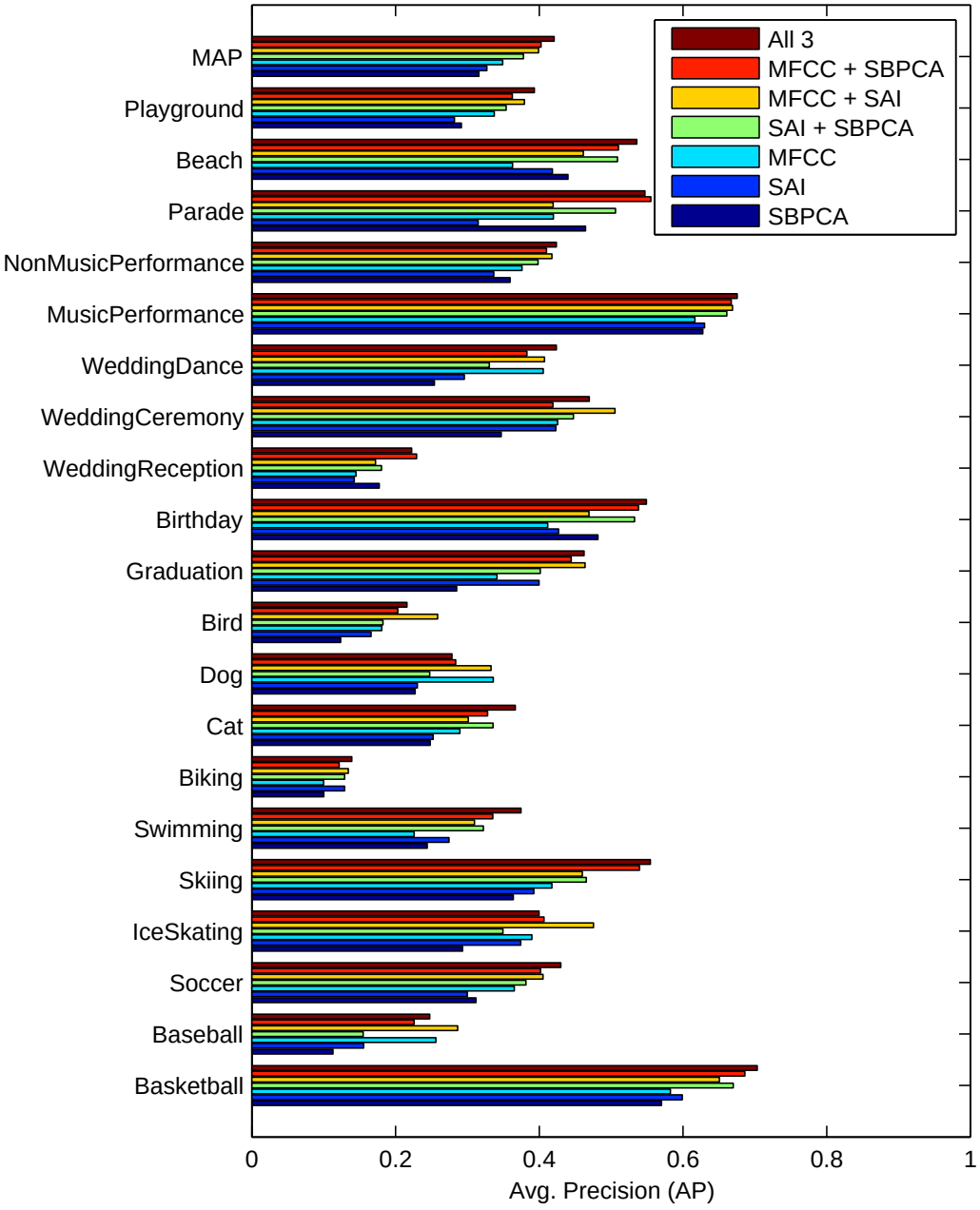

Figure 7.6: SVM results with each individual feature set (MFCC, SAI, SBPCA), fusion of each pair, and fusion of all 3.

Chapter 8

Conclusions

8.1 Thesis Summary

In this thesis, we have explored a variety of novel representations for unconstrained audio content. We started with an exploration of how matching pursuit decompostions could be used to generate features to match similar sound events to each other. Here, we demonstrated the feasibility of this idea, by showing that similar patterns of MP atoms do recur in the decomposition of similar sound events. Following this exploration, we further pursued the use of MP as a tool for matching identical audio sequences in a large database of real professional and consumer recordings. We presented a fingerprinting system that successfully identifies segments of audio recorded at the same time and place, despite widely varying recording conditions and background noise.

Our interest in the matching pursuit representation was inspired by a desire to locate and characterize transients. However, we found the resulting MP representations too sparse and brittle to develop good features from. Building upon this idea, we next investigated features based on identifying transients and describing the full time-frequency patch of the spectrum around them. We demonstrated some promising results using these features on another large video database. Searching for a better way to determine transient event locations, we next turned to a version of non-negative matrix factorization. Using features based on time-frequency patches discovered using convolutive NMF, we built an event detection system for meeting room recordings. We demonstrated that our NMF-based features were more robust to noise than a comparable MFCC-based detection system.

Looking for other features to use in classification that might be complementary to MFCC features, we finally investigated a re-implementation of an auditory model-based feature set described in [Lyon *et al.*, 2010] and ran our own experiments on it. We compared these features with similar but easier to compute features, and used both in combination with MFCCs to achieve greatly improved concept retrieval performance over a much larger video database.

Broadly, we can think about three basic approaches to audio understanding that we addressed here: matching pursuit-based (chapters 3 and 4), time-frequency patch-based (chapters 5 and 6), and auditory model-based (chapter 7). Each of these approaches was useful for audio characterization in the kinds of noisy scenarios found in consumer and internet videos. We found that an especially robust approach to many of our problems was to combine one of these novel representations with more conventional short-time spectral (MFCC) features for improved performance.

Our original goal was to learn something about the types of individual events that indicate semantic concepts in videos. Ultimately, we did not find as many clearly identifiable 'event' units as we had hoped in the large video corpuses we examined. Despite this, our systems still identified statistical correlations that proved useful in classifying these videos. Given the things we have learned over the course of this work, it still seems like a feasible idea to identify events that have semantic meaning in a video. It is not clear what all the factors might be, but it seems that the main obstacles may be only a lack of well-labeled examples from which to learn or a lack of understanding of the actual relevant contents of these large video corpuses. Our final conclusion is that an event detection system from which more obvious semantic meaning may be derived is probably a feasible goal. It may only take more investigation into such systems using a dataset with more diverse and more closely-labeled examples.

8.2 Future Work

There are several potential directions to focus on going forward. One area would be the continued integration of the approaches described here with visual features to create general-purpose video search systems. We have done some of this type of work already (for example in [Jiang *et al.*, 2009]), and efforts in this direction continue (although they are somewhat outside the scope of this thesis). There is also the potential to integrate the type of specialized event detectors explored here

with explicit speech and music detectors to build video search systems with more powerful search capabilities.

We would also like to continue to explore and improve those features that seem most promising for video classification. Ideally we would like to be able to perform the NMF decomposition of chapter 6 on the large video datasets and compare its classification ability with the transient and auditory-model systems. Unfortunately this is currently too computationally expensive with the implementation we have; it is possible we could develop a more efficient implementation or some approximation of the algorithm. Alternatively, we could try to modify our transient detector (chapter 5) to select events more sparsely and reliably or prune them somehow to obtain transient patches more similar to those selected by NMF (although the patches would still suffer from additive background noise). In general, we are interested in performing more tuning of the transient detection system parameters. We believe that the system could be refined to reduce the number of transients selected and allow us to better isolate which transients are discriminative for classification.

Furthermore, we are interested in combining the ideas introduced by the auditory model features of chapter 7 with our transient system. Both the SAI and SBPCA features seem useful for classification, but they still describe an entire noisy soundtrack clip with standard uniform frames. We still believe that the informative information in a soundtrack is likely to be unevenly distributed around transient events. One interesting idea would be to take SBPCA (or SAI) features only around times selected by a (possibly better tuned) transient detector, such that the system is again only modeling a subset of each soundtrack.

Bibliography

[Andoni and Indyk, 2008] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[Anguera and Adamek, 2011] X. Anguera and T. Adamek. Multimodal video copy detection using local features. *IEEE Communications Society Multimedia Communications Technical Committee E-Letter*, 6(1), Jan 2011.

[Brezeale and Cook, 2008] D. Brezeale and D.J. Cook. Automatic video classification: A survey of the literature. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(3):416–430, 2008.

[Cai *et al.*, 2006] R. Cai, L. Lu, A. Hanjalic, H.J. Zhang, and L.H. Cai. A flexible framework for key audio effects detection and auditory context inference. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(3):1026–1039, 2006.

[Cano *et al.*, 2005] P. Cano, E. Batlle, T. Kalker, and J. Haitsma. A review of audio fingerprinting. *The Journal of VLSI Signal Processing*, 41(3):271–284, 2005.

[Casey and Slaney, 2007] M. Casey and M. Slaney. Fast recognition of remixed music audio. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages IV–1425–1428, Hawai'i, 2007.

[Casey, 2001] M. Casey. General sound classification and similarity in mpeg-7. *Organised Sound*, 6(02):153–164, 2001.

[Chang *et al.*, 2007] S.-F. Chang, D. Ellis, W. Jiang, K. Lee, A. Yanagawa, A.C. Loui, and J. Luo. Large-scale multimodal semantic concept detection for consumer video. In *Proc. ACM Multimedia, Information Retrieval Workshop*, Sept 2007.

[CHIL, 2008] CHIL. FBK-Irst database of isolated meeting-room acoustic events. `http://catalog.elra.info/product_info.php?products_id=1093`, 2008.

[Cho and Choi, 2005] Y.C. Cho and S. Choi. Nonnegative features of spectro-temporal sounds for classification. *Pattern Recognition Letters*, 26(9):1327–1336, 2005.

[Chu *et al.*, 2008] S. Chu, S. Narayanan, and C.C.J. Kuo. Environmental sound recognition using MP-based features. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2008.

[Chu *et al.*, 2009] S. Chu, S. Narayanan, and C.C.J. Kuo. Environmental sound recognition with time–frequency audio features. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(6):1142–1158, 2009.

[Cotton and Ellis, 2009] C. Cotton and D.P.W. Ellis. Finding similar acoustic events using matching pursuit and locality-sensitive hashing. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Mohonk, NY, October 2009.

[Cotton and Ellis, 2010] C. Cotton and D. Ellis. Audio fingerprinting to identify multiple videos of an event. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Dallas, TX, March 2010.

[Cotton and Ellis, 2011] C. Cotton and D. Ellis. Spectral vs. spectro-temporal features for acoustic event detection. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Mohonk, NY, October 2011.

[Cotton *et al.*, 2011] C. Cotton, D. Ellis, and A. Loui. Soundtrack classification by transient events. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 473–476, Prague, May 2011.

[Ekenel *et al.*, 2010] H.K. Ekenel, T. Semela, and R. Stiefelhagen. Content-based video genre classification using multiple cues. In *Proceedings of the 3rd international workshop on Automated information extraction in media production*, pages 21–26. ACM, 2010.

[Eronen *et al.*, 2006] A.J. Eronen, V.T. Peltonen, J.T. Tuomi, A.P. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi. Audio-based context recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(1):321–329, 2006.

[Foote, 1997] J. Foote. Content-based retrieval of music and audio. In *Proc. SPIE*, volume 3229, pages 138–147, 1997.

[Gribonval and Krstulovic, 2006] R. Gribonval and S. Krstulovic. MPTK, The Matching Pursuit Toolkit, `http://mptk.irisa.fr/`, 2006.

[Heusdens *et al.*, 2002] R. Heusdens, R. Vafin, and W.B. Kleijn. Sinusoidal modeling using psychoacoustic-adaptive matching pursuits. *IEEE Signal Processing Letters*, 9(8):262–265, Aug. 2002.

[Jiang *et al.*, 2009] W. Jiang, C. Cotton, S.-F. Chang, D. Ellis, and A. Loui. Short-term audio-visual atoms for generic video concept classification. In *Proc. ACM International Conference on MultiMedia*, Beijing, October 2009.

[Jiang *et al.*, 2011] Y.-G. Jiang, G. Ye, S.-F. Chang, D. Ellis, and A.C. Loui. Consumer video understanding: A benchmark database and an evaluation of human and machine performance. In *Proc. ACM International Conference on Multimedia Retrieval (ICMR)*, Apr. 2011.

[Kalinli *et al.*, 2009] O. Kalinli, S. Sundaram, and S. Narayanan. Saliency-driven unstructured acoustic scene classification using latent perceptual indexing. In *Proc. IEEE International Workshop on Multimedia Signal Processing (MMSP)*, Oct 2009.

[Kennedy and Naaman, 2009] L. Kennedy and M. Naaman. Less talk, more rock: automated organization of community-contributed collections of concert videos. In *Proc. 18th International Conconference on World Wide Web, ACM*, page 311320, October 2009.

[Lee and Ellis, 2010] K. Lee and D.P.W. Ellis. Audio-based semantic concept classification for consumer video. *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, 18(6):1406–1416, Aug 2010.

[Lee and Ellis, 2012] B.-S. Lee and D. Ellis. Noise robust pitch tracking by subband autocorrelation classification. In *Proc. Interspeech-12*, Sept. 2012.

[Lee and Seung, 1999] D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 204:788–791, 1999.

[Li *et al.*, 2001] D. Li, I.K. Sethi, N. Dimitrova, and T. McGee. Classification of general audio data for content-based retrieval. *Pattern recognition letters*, 22(5):533–544, 2001.

[Liu *et al.*, 1997] Z. Liu, J. Huang, Y. Wang, and T. Chen. Audio feature extraction and analysis for scene classification. In *Multimedia Signal Processing, 1997., IEEE First Workshop on*, pages 343–348. IEEE, 1997.

[Liu *et al.*, 1998] Z. Liu, J. Huang, and Y. Wang. Classification tv programs based on audio information using hidden markov model. In *Multimedia Signal Processing, 1998 IEEE Second Workshop on*, pages 27 –32, dec 1998.

[Loui *et al.*, 2007] A.C. Loui, J. Luo, S.-F. Chang, D. Ellis, W. Jiang, K. Lee, L. Kennedy, and A. Yanagawa. Kodak's consumer video benchmark data set: Concept definition and annotation. In *Proc. ACM Multimedia, Information Retrieval Workshop*, Sept 2007.

[Lu *et al.*, 2003] L. Lu, H.J. Zhang, and S.Z. Li. Content-based audio classification and segmentation by using support vector machines. *Multimedia systems*, 8(6):482–492, 2003.

[Lyon *et al.*, 2010] R.F. Lyon, M. Rehn, S. Bengio, T.C. Walters, and G. Chechik. Sound retrieval and ranking using sparse auditory representations. *Neural Computation*, 22(9), Sept. 2010.

[Lyon *et al.*, 2011] R.F. Lyon, J. Ponte, and G. Chechik. Sparse coding of auditory features for machine hearing in interference. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, May 2011.

[Mallat and Zhang, 1993] S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12), Dec. 1993.

[Mesaros *et al.*, 2010] A. Mesaros, T. Heittola, A. Eronen, and T. Virtanen. Acoustic event detection in real life recordings. In *18th European Signal Processing Conference*, 2010.

[Mostefa *et al.*, 2007] D. Mostefa, N. Moreau, K. Choukri, G. Potamianos, S.M. Chu, A. Tyagi, J.R. Casas, J. Turmo, L. Cristoforetti, F. Tobia, et al. The chil audiovisual corpus for lecture and meeting analysis inside smart rooms. *Language Resources and Evaluation*, 41(3):389–407, 2007.

[Ogle and Ellis, 2007] J. Ogle and D. Ellis. Fingerprinting to identify repeated sound events in long-duration personal audio recordings. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume I, pages 233–236, 2007.

[O'Grady and Pearlmutter, 2006] Paul D. O'Grady and Barak A. Pearlmutter. Convolutive non-negative matrix factorisation with a sparseness constraint. In *Proc. IEEE Machine Learning for Signal Processing (MLSP)*, pages 427–432, Maynooth, September 2006.

[Pan, 1995] Davis Pan. A tutorial on MPEG audio compression. *IEEE Multimedia Magazine*, 2(2):60–74, 1995.

[Patel and Sethi, 1996] N.V. Patel and I.K. Sethi. Audio characterization for video indexing. In *Proceedings SPIE on Storage and Retrieval for Still Image and Video Databases*, volume 2670, pages 373–384, 1996.

[Petitcolas, 2003] F. Petitcolas. MPEG for MATLAB, `http://www.petitcolas.net/fabien/software/mpeg`, 2003.

[Radhakrishnan *et al.*, 2005] R. Radhakrishnan, A. Divakaran, and A. Smaragdis. Audio analysis for surveillance applications. In *Applications of Signal Processing to Audio and Acoustics, 2005. IEEE Workshop on*, pages 158–161. IEEE, 2005.

[Ramachandran *et al.*, 2009] C. Ramachandran, R. Malik, X. Jin, J. Gao, K. Nahrstedt, and J. Han. Videomule: a consensus learning approach to multi-label classification from noisy user-generated videos. In *Proceedings of the 17th ACM international conference on Multimedia*, pages 721–724. ACM, 2009.

[Saunders, 1996] J. Saunders. Real-time discrimination of broadcast speech/music. In *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, volume 2, pages 993 –996 vol. 2, may 1996.

[Scheirer and Slaney, 1997] E. Scheirer and M. Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 2, pages 1331 –1334 vol.2, apr 1997.

[Shrestha *et al.*, 2007] P. Shrestha, M. Barbieri, and H. Weda. Synchronization of multi-camera video recordings based on audio. In *Proc. of the 15th International Conference on Multimedia, ACM*, volume 25, page 545548, 2007.

[Slaney, 2002] M. Slaney. Mixtures of probability experts for audio retrieval and indexing. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, volume 1, pages 345–348. IEEE, 2002.

[Smaragdis and Brown, 2003] P. Smaragdis and J. Brown. Non-negative matrix factorization for polyphonic music transcription. In *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 177–180, Mohonk, 2003.

[Smaragdis, 2007] P. Smaragdis. Convolutive speech bases and their application to supervised speech separation. *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, 15(1):1–12, 2007.

[Temko *et al.*, 2007] A. Temko, R. Malkin, C. Zieger, D. Macho, C. Nadeu, and M. Omologo. Clear evaluation of acoustic event detection and classification systems. *Multimodal Technologies for Perception of Humans*, pages 311–322, 2007.

[Temko, 2007] A. Temko. *Acoustic Event Detection and Classification (Ph.D. Thesis)*. Department of Signal Theory and Communications, Universitat Politecnica de Catalunya, Barcelona, Spain, 2007.

[Walters, 2010] Tom Walters. Aim-c, a c++ implementation of the auditory image model. http://code.google.com/p/aimc/, 2010.

[Wang *et al.*, 2010] Z. Wang, M. Zhao, Y. Song, S. Kumar, and B. Li. Youtubecat: Learning to categorize wild web videos. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 879–886. IEEE, 2010.

[Wang, 2006] A. Wang. The Shazam music recognition service. *Communications of the ACM*, 49(8):44–48, Aug. 2006.

[Wold *et al.*, 1996] E. Wold, T. Blum, D. Keislar, and J. Wheaten. Content-based classification, search, and retrieval of audio. *MultiMedia, IEEE*, 3(3):27–36, 1996.

[Zhang and Kuo, 2001] T. Zhang and C.-C.Jay Kuo. Audio content analysis for online audiovisual data segmentation and classification. *Speech and Audio Processing, IEEE Transactions on*, 9(4):441 –457, may 2001.

[Zhuang *et al.*, 2010] X. Zhuang, X. Zhou, M.A. Hasegawa-Johnson, and T.S. Huang. Real-world acoustic event detection. *Pattern Recognition Letters*, 31(12):1543–1551, 2010.