Environmental Audio-Visual Context Awareness

by

Brian Clarkson

S.B. in Mathematics, Massachusetts Institute of Technology (1999)
S.B. in Computer Science and Electrical Engineering, Massachusetts Institute of Technology (1999)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1999

© Massachusetts Institute of Technology, MCMXCIX. All rights reserved.

The author hereby grants to MIT permission to reproduce and distribute publicly
paper and electronic copies of this thesis document in whole or in part, and to grant
others the right to do so.

Author

Department of Electrical Engineering and Computer Science
February 1, 1999

Certified by

Alex P. Pentland
Department Head/Associate Professor, Media Arts and Sciences
Thesis Supervisor

Accepted by

Arthur C. Smith
Chairman, Departmental Committee on Graduate Theses
Environmental Audio-Visual Context Awareness

by

Brian Clarkson

Submitted to the Department of Electrical Engineering and Computer Science on February 1, 1999, in partial fulfillment of the requirements for the degree of Master of Engineering in Electrical Engineering and Computer Science

Abstract

A truly personal and reactive computer system should have access to the same information as its user, including the ambient sights and sounds. To this end, we have developed a system for extracting events and scenes from natural audio/visual input. We find our system can (without any prior labeling of data) cluster the audio/visual data into events, such as passing through doors and crossing the street. Also, we hierarchically cluster these events into scenes and get clusters that correlate with visiting the supermarket, or walking down a busy street.

Thesis Supervisor: Alex P. Pentland
Title: Department Head/Associate Professor, Media Arts and Sciences
Acknowledgements

To the best parents in the world, Thomas R. Clarkson, my hilarious, enthusiastic and tinkering Father, and Anneliese M. Clarkson, my conscientious, emotional, instinctive and beautiful Mother.

To Mr. Harris and Mrs. Barnes, the science and math teachers who gave me the confidence and drive to get to this far.

To Dina Finan, my significant other, who has (even though she is a very sophisticated and modern lady) gladly cooked me dinner on those late nights that I came home from work exhausted and delusional.

To Tony Jebara, Nitin Sawhney, and Deb Roy for all their “emotional” support.

To Sumit Basu for all the lengthy and “useless-but-maybe-not” discussions.

To Bernt Schiele who had the patience and care to teach me how to be scientific.

To Professor Pentland who has granted me so many opportunities like dancing on stage in a fashion show in Tokyo, all in the name of science!
# Table of Contents

CHAPTER 1: INTRODUCTION ............................................................................................................. 3
  GOAL ............................................................................................................................................ 3
  THE APPROACH .......................................................................................................................... 4
  OVERVIEW .................................................................................................................................. 5

CHAPTER 2: THE SUPERVISED APPROACH .................................................................................... 6
  SPEAKER IDENTIFICATION IN THE PRESENCE OF BACKGROUND NOISE ...................................... 6
    Event Detection .......................................................................................................................... 7
    Feature Extraction .................................................................................................................... 8
    Modeling .................................................................................................................................... 8
    Background Adaptation ............................................................................................................. 9
  A SIMPLE AUDITORY SCENE CLASSIFIER (SASC) ...................................................................... 11
  FEATURE EXTRACTION ................................................................................................................ 11
  SOUND OBJECT CLASSIFICATION ............................................................................................... 12
    Methods ...................................................................................................................................... 12
    Application ............................................................................................................................... 13
    Results ...................................................................................................................................... 14

CHAPTER 3: THE CLUSTERING APPROACH ................................................................................. 16
  SOUND SCENE SEGMENTATION .................................................................................................. 16
    Methods ...................................................................................................................................... 16
    Experiment ................................................................................................................................ 18

CHAPTER 4: AUDIO & VIDEO SEGMENTATION OF EVENTS & SCENES ........................................... 21
  THE PERSONAL AUDIO-VISUAL TASK ....................................................................................... 21
  THE APPROACH .......................................................................................................................... 21
  FEATURE EXTRACTION ................................................................................................................ 22
  TIME SERIES CLUSTERING .......................................................................................................... 23
  TIME HIERARCHY .......................................................................................................................... 24
  REPRESENTATION HIERARCHY ..................................................................................................... 24
  DATA COLLECTION ....................................................................................................................... 25
  RESULTS ...................................................................................................................................... 26
    Short Time Scale Object HMMs ............................................................................................. 26
    Long Time-scale Object HMMs .............................................................................................. 26
    Long Time-scale Scene HMMs ............................................................................................... 28
Chapter 1: Introduction

**Goal**

Computers have evolved into miniature and wearable systems[1]. As a result there is a desire for these computers to be tightly coupled with their user's day-to-day activities. A popular analogy for this integration equates the wearable computer to an intelligent observer and assistant for its user. To fill this role effectively, the wearable computer needs to live in the same sensory world as its human user.

Thus, a system is required that can take this natural and personal audio/video and find the coherent segments, the points of major activity, and recurring events. When this is achieved then the personal computing devices you carry can act based on events in your surroundings. While the domain is different there are efforts in the field of video surveillance and monitoring (VSAM) that have many of the same technical problems. VSAM researchers (such as the Forest of Sensors Project [2] and others listed in [3]) concentrate their sensors on a fixed location which facilitates the segmentation problem. In this work and in others [4, 5] the sensors (video cameras and microphones) are placed on the body for the purpose of observing a person's surroundings. In this case the location and sensor direction is always changing with the wearer's actions. This makes the segmentation problem very difficult and in some ways much more interesting. The field of multimedia indexing has wrestled with many of the problems that such a system creates. However, audio/video data that these researchers typically tackle are very heterogeneous and thus have little structure and what structure they do have is usually artificial, like scene cuts, and patterns in camera angle. The "eyes and ears" audio/video data that we are tackling is much more homogeneous and thus richer in structure, and filled with repeating elements and slowly varying trends.

The use of our system's resulting indexing differs greatly from the typical "querying for key-frames". Suppose our system has clustered its audio/video history into 100 models. Upon further use, the system notices that whenever the user requests his grocery list, model 42 is active. We would say that model 42 is the supermarket. However, the system does not need to have such a human-readable for model 42. (What
would the system do with it? The user presumably knows already that he is in the supermarket.) However, a software agent built on our system would know to automatically display the user's grocery list when model 42 activates.

**The Approach**

This work uses auditory and visual cues to classify the user's environmental context. Like "the fly on the wall" (except now the fly is on your shoulder) it does not try to understand in detail every event that happens around the user (as in [4]). Instead, we make general evaluations of the auditory and visual ambience and whether a particular environment is different or similar to an environment that the user was previously in. In other words we want to build a metric that will allow us to navigate or make queries on the user's audio-visual history much like the image retrieval task. In fact many of the techniques introduced by Photobook [6] and the QBIC project [7] are applicable here.

Of the myriad of physical sensors available to us, sight and sound can be the most compelling because the user and his computer can potentially have the same perceptions of the environment. If the computer ignores an event because it could not see or hear it, the user will naturally anticipate this as he would for his fellow man.

One way to make use of the audio-visual channel is to construct detectors for specific events. These detectors can be simple such as detecting bright light or loud sounds or very complicated such as identifying speakers, or recognizing objects. Given these detectors higher order patterns can be observed such as a correlation amongst different events. For example, a user's audio-visual environment can be broken into scenes (maybe overlapping) such as 'visiting the grocery store', 'walking down a busy street', or 'at the office' that are really collections of specific events such as 'footsteps', 'car horns', and 'crosswalks'. If we had detectors for the low-level events that make up these scenes then we can recognize the scenes. This identifies a natural hierarchy in a person's audio-visual environment. There are events that correspond to short and simple patterns and then scenes, which are collections of these events again in some distinctive pattern.

It is important to notice that recognizing these scenes is not the same as recognizing location. A scene definitely depends on the location, but there are usually other defining characteristics. For example, 'working at the grocery store' would have a
very different audio-visual pattern than 'shopping at the grocery store'. Fortunately, 'working at the grocery store on Monday' typically does not differ greatly (in the audio-visual sense) from 'working at the grocery store on Wednesday', but it might. The key point is that if two segments of a person's audio-visual history differ greatly than they are usually perceived as different scenes. This supports using the inherent variation in the audio-visual channel to find a meaningful indexing.

**Overview**

Each of the ensuing chapters describes a stage in our research towards extracting context from the audio/visual environment. Chapter 2 introduces the main methods for modeling auditory events in a traditional supervised learning framework. Assuming a constrained domain (few classes and/or stable environments), it is possible to get useful classification of auditory events (such as speaker identification, and other nonspeech sounds). Moving beyond the constrained domains requires the ability to adapt to new conditions, which are not explicitly labeled. In chapter 3 we remove the assumption that our learning algorithms are receiving labels and explore what information can be extracted from the auditory environment based on the raw data alone. Finally, Chapter 4 has the same goal but visual information is now included.
Chapter 2: The Supervised Approach

We begin with two experiments for classifying various auditory events. In our first experiment, Speaker Identification (ID), we develop a framework for building spectral models of clean auditory events and then incorporate temporal models of background noise to make these clean speech models usable in natural settings. Finally, we try these techniques in a prototype application that classifies the auditory context and utilizes it to make decisions for the user.

**Speaker Identification in the Presence of Background Noise**

In this experiment we more exactly evaluate the performance of our techniques on the speaker identification task. We concentrate on natural speech and background noise and methods for dealing with them. We use the speaker identification task as a starting point only because it is well defined. Later in this chapter we extend these speaker identification results to general sounds.

Past work has shown that text-independent speaker identification (SI) relies on the characterization of the spectral distributions of speakers. However, convolutional and additive noise in the audio signal will cause a mismatch between the model and test distributions, resulting in poor recognition performance [8, 9]. Even if the audio channel is kept consistent so as to minimize convolutional noise, there will always be the problem of additive noise in natural scenarios.

Deconvolutional techniques such as RASTA[10] have had substantial success in matching the spectral response of different auditory channels. However, severe drops in performance are still evident with even small amounts of additive noise.

Work done by [9] has suggested that the presence of noise doesn't necessarily degrade recognition performance. They compared their system's error rates on a clean database (YOHO) and a more realistic database (SESP). When training and testing were done on the same database the error rates were comparable. Building on this idea, our speaker identification system is based on a simple set of linear spectral features, which are characterized with HMMs. This simple combination is well suited for adapting the speaker models to various types of background noise.
Event Detection

The first stage in the system’s pipeline is the coarse segmentation of auditory events. The purpose of this segmentation is to identify segments of audio, which are likely to contain valuable information. We chose this route because it makes the statistical modeling much easier and faster. Instead of integrating over all possible segmentations, we have built-in the segmentation as prior knowledge.

The most desirable event detector should have negligible computational cost and low false rejection rate. The hypothesized events can then be handed to any number of analysis modules, each specialized for their classification task (e.g. speech recognizer, speaker identification, location classifier, language identification, prosody, etc.).

![Energy vs. Adaptive Energy](image)

Figure 1: The event detector uses a normalized version (bottom) of raw energy (top) to gradually ignore long-lasting sounds.

We used a simple and efficient event detector, constructed by thresholding total energy and incorporating constraints on event length and surrounding pauses. These constraints were encoded with a finite-state machine. This method’s flaw is the possibility of arbitrarily long events. An example is walking into a noisy subway where the level of sound always exceeds the threshold. A simple solution is to adapt the threshold or equivalently scale the energy. The system keeps a running estimate of the energy statistics and continually normalizes the energy to zero mean and unit variance (similar to Brown’s onset detector [11]). The effect is that after a period of silence the system is hypersensitive and after a period of loud sound the system grows desensitized. Figure 1 shows the effects of adaptation for a simple tone (actual energy is on top and adapted energy is on the bottom). Notice that after 500 frames (8 secs), the system is ready to detect other sounds despite the continuing tone.
Feature Extraction

After segmentation the (16 kHz sampled) audio is filtered with a weak high-pass filter (preemphasis) in order to remove the DC offset and boost the higher frequencies. We calculate Mel-scaled frequency coefficients (MFCs) for frames of audio that are spaced 16 ms apart and are 32 ms long. This frame size sets the lower limit on the frequency measurement to approximately 30 Hz. Mel-scaling increases the resolution for lower frequencies, where speech typically occurs.

MFC is a linear operation on the audio signal, so additive noise does not cause a nonlinear distortion in our features. This useful because it allows us to detect additive noise given a model of the noise in isolation.

Modeling

Our system uses continuous HMMs to capture the spectral signature of each speaker. An HMM for each person is estimated from examples of their speech. The estimation was achieved by first using segmental k-means to initialize HMM parameters and then Expectation-Maximization (EM) to maximize (locally) the model likelihood. Since the examples of speech are text-independent there is no common temporal structure amongst the training examples. This situation requires the use of fully-connected (ergodic) HMMs.

In order to find the optimal model complexity for our task, we varied the number of states and number of Gaussians per state until the recognition rate was optimized. We tested HMMs with 1 to 10 states and 1 to 100 Gaussians. The best performance was achieved with a 1 state HMM with 30 Gaussians per state or, equivalently, a mixture of 30 Gaussians. This is not surprising given the lack of temporal structure in our text-independent training and testing examples. Arguably this makes the use of HMMs unnecessary. However, the use of HMMs is justified for our background noise adaptation.
Background Adaptation

Statistical models trained on clean speech (or speech in any specific environment) will perform badly on speech in a different environment. The changing environment causes distortions in the speech features, which create a mismatch between the test speech and model distribution. Primarily, differing microphone and sound card types, and microphone and sound source location cause convolutional noise. Additive noise is caused by the presence of other sound sources. We will assume that the microphone type and location is constant and concentrate on additive noise only.

The goal is to be able to adapt models of clean speech for use in noisy environments. However, the adaptation cannot require samples of the speech in the noisy environment because usually they are not available. So given only the clean speech models and recordings of the background noise, our adaptation technique can estimate the appropriate noisy speech models.

The model adaptation procedure (which is related to the parallel model combination algorithm of [12]) is based on estimating HMMs for noisy speech from HMMs separately trained on speech and noise. Since the background noise might have temporal structure, such as repetitive noises like motor noise, or randomly occurring changes like thunder in a rain storm, it is appropriate to use an HMM to model it. The feature extraction and HMM training was the same as above.

If the background noise and speech are assumed independent and the features are extracted using only linear operators then the distributions can be easily estimated. Let $B$ be the background noise HMM with $M$ states, $S$ the clean speech HMM with $N$ states, and $S'$ the noisy speech HMM. The combination of the two HMMs, $S$ and $B$, is the HMM $S'$ with $MN$ states in the state space constructed from the outer product of the $S$ and $B$ state spaces. The probability distributions for each state in $S'$ are the convolution of the distributions in $S$ with the distributions in $B$.

This adaptation was evaluated using the speech of 26 people (the data collection is described in the Appendix) and an auditory background scene of a street in a thunder storm. The noise scene contains frequent thunder and occasional passing cars against a constant background of rain. We created two sets of audio data: a Speech Only set with uncorrupted speech, and a Speech + Noise set which was constructed by adding the
background recordings to the audio clips in the *Speech Only* set. They were mixed at a Signal-to-Noise Ratio (SNR) of 7dB. Each of these sets was further divided into training and test sets.

A single state HMM, $S_i$, was trained on the speech of each individual from the *Speech Only* set. A 3-state HMM, $B$, was trained on the background sounds. This HMM was used to adapt the $S_i$ HMMs thereby creating a new set of HMMs, $S_i'$, which should match the speech in the *Speech + Noise* set. Although it is not an option for real-time adaptation, we also trained HMMs, call them $C_i$, on the *Speech + Noise* training set to evaluate the effectiveness of the adaptation. The performance of $C_i$ is theoretically the upper limit for the performance of $S_i'$ and the performance of $S_i$ is the upper limit of performance for both.

Finally we test all HMMs on both the *Speech Only* and *Speech + Noise* test sets. Table 1 contains the recognition rates for two sets of 130 audio clips. As shown by the extremely poor performance of the $S$ HMMs on the *Speech + Noise* test set, the background scene has clearly caused a mismatch between the speech models and the audio. However, the 69.2% performance of the $C$ HMMs (very near the performance of the clean speech models, 71.5%) shows that the additive noise is not hindering discrimination very much. The adaptation is able to regain 95% of the performance if we assume the $C$ HMMs are exactly matched to the *Speech + Noise* set.

<table>
<thead>
<tr>
<th>HMM Models</th>
<th>Test Sets</th>
<th>Speech Only</th>
<th>Speech + Noise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speech Only (S)</td>
<td></td>
<td>71.5%</td>
<td>23.1%</td>
</tr>
<tr>
<td>Corrupted (S')</td>
<td></td>
<td>N/A</td>
<td>69.2%</td>
</tr>
<tr>
<td>Adapted (C)</td>
<td></td>
<td>N/A</td>
<td>65.4%</td>
</tr>
</tbody>
</table>

Table 1: Recognition rates for the clean speech, corrupted speech and adapted speech models.
**A Simple Auditory Scene Classifier (SASC)**

In this chapter, we explore the classification of some simple auditory scenes (user is alone, user is in a conversation, user is near a conversation but not directly involved) and its use in a wearable audio-only messaging system. The SASC system is designed for real-time real world I/O. No special considerations were made in the selection of the microphone except that it should be stable, small and unobtrusive. Currently we use a wireless lavalier microphone (Lectrosonics M-185) because it can be easily integrated into personal mobile computing platforms. We believe that wearable systems will have the most to gain from the SASC system because the auditory environment of a mobile user is dynamic and structured. The user’s day-to-day routine contains recurring auditory events that can be correlated amongst each other and the user’s tasks.

**Feature Extraction**

The SASC system uses Mel-scaled filterbank coefficients (MFCs) and pitch estimates to discriminate a variety of speech and non-speech sounds. Although for the purposes of this work we restrict ourselves to a single set of features, we strongly believe that our system should include mechanisms for generating new features candidates as needed, and automatically selecting the appropriate features for the task.

To get a sense of what information this particular feature set extracts, we can compare the voices of different speakers. Figure 2 below shows the MFC features for 8 speech events (extracted with the event detection algorithm). There are 2 examples for each speaker to show some

![Figure 2: Comparison of speakers using 15 mel-scaled filter-banks. Notice that the gross spectral content is distinctive for each speaker. (Frequency is vertical and time is horizontal.)](image-url)
possible variations. These diagrams use 15 mel-scaled filter-banks (ranging from 0 to 2000Hz log-scale, each about 0.5 secs long) to show the rough spectral peaks for these speakers. Discrimination of speaker identification (ID) for 4 speakers is quite simple as indicated by our 100% recognition accuracy (on a test set) using HMMs on these features. As more speakers are registered with the system (using only 15 mel-scaled filter-banks), the accuracy drops drastically. Adding pitch as an additional feature increases accuracy. An actual system for auditory scene analysis would need to be able to add features like this automatically. More complex methods would allow the discrimination of more speakers, but usually physical context (such as being in the office vs. at home) can restrict the number and identity of expected speakers.

**Sound Object Classification**

Methods

The features extracted from an event form a time series in a high dimensional space. Many examples of the same type of sound form a distribution of time series which our system models with a HMM. Hidden Markov Models capture the temporal characteristics as well as the spectral content of an event. Systems like Schreirer’s [5], Saint-Arnaud [4], and Foote [12] ignore this temporal knowledge.
Since our SASC system is event-driven, the process of compiling these training examples is made easier. The event detection produces a sequence of events such as in Figure 3. Only those events that contain the sound object to be recognized need to be labeled. Also, it is not necessary to specify the extent of the sound object (in time) because the event detector provides a rough segmentation. Since the event might span more time than its sound object (as it usually does when many sound objects overlap), it implicitly identifies context for each sound object.

Once HMMs have been estimated for the needed sound objects, new sounds can be compared against each HMM. Similar sound objects will give high likelihoods and dissimilar objects low likelihoods. At this point the system can either classify the sound as the nearest sound object (highest HMM likelihood) or describe the sound in terms of the nearest N sound objects. The last option is necessary for detecting the case where more than one sound object may be present in the event.

Application

Nomadic Radio is a wearable computing platform that provides a unified audio interface to a number of remote information services [8]. Messages such as email, voice mail, hourly news, and calendar events are automatically downloaded to the device throughout the day, and the user must be notified at an appropriate time. A key issue is that of handling interruptions to the listener in a manner that reduces disruption, while providing timely notifications for relevant messages. This approach is similar to prior work by [9] on using perceptual costs and focus of attention for a probabilistic model of scaleable graphics rendering.

In Nomadic Radio the primary contextual cues used in the notification model include: message priority level from email filtering, usage level based on time since last user action, and the conversation level estimated from real-time analysis of sound events in the mobile environment. If the system detects the occurrence of more than several speakers over a period of time (10-30 seconds), that is a clear indication of a conversational situation, then an interruption may be less desirable. The likelihood of speech detected in the environment is computed for each event within (10-30 second) window of time. In addition, the probabilities are weighted, such that most recent time
periods in the window are considered more relevant in computing the overall speech level. A weighted average for all three contextual cues provides an overall notification level. The speech level has an inverse proportional relationship with notification i.e. a lower notification must be provided during high conversation.

The notification level is translated into discrete notification states within which to present the message (i.e. as an ambient or auditory cue, spoken summary or preview and spatial background or foreground modes). In addition, a latency interval is computed to wait before playing the message to the user. Actions of the user, i.e. playing, ignoring or deactivating the message adjust the notification model to reinforce or degrade the notification weights for any future messages during that time period.

Results

We trained our HMM classifier on 3 classes: Background, User Speech, and Friend Speech in used these class likelihoods to detect the occurrence of 3 classes: User is Alone, User is Busy (i.e. in conversation with his friend), User is Semi-busy (i.e. his friend is nearby, but the User is not directly in conversation with him). We used the occurrence of these 3 classes to set Nomadic Radio to one of 3 notification levels: Full Message, Just Beep, and Message Summary, corresponding to the 3 scenes, respectively. Table 2 gives frames from a video we presented to the Perceptual User Interface Workshop in 1998 that shows the device being used.
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The User is alone:</strong></td>
<td>Here the user is not involved in a conversation with anyone so the Nomadic Radio reads the entire message to the User.</td>
</tr>
<tr>
<td><strong>The User is busy:</strong></td>
<td>The User is engaged in conversation with another person, hence Nomadic Radio mildly signals (a low beep) an incoming message.</td>
</tr>
<tr>
<td><strong>The User is semi-busy:</strong></td>
<td>The User is near a conversation, but is not actively engaged in it. In this case the Nomadic Radio reads only a message summary.</td>
</tr>
</tbody>
</table>

Table 2: Our simple auditory scene classifier is able to distinguish between these 3 scenarios. The SASC classifier is shown here combined with Nitin Sawhney's Nomadic Radio[13].
Chapter 3: The Clustering Approach

As we noted in the previous chapter, it is quite possible once we have trained models for the desired auditory events, that we can detect and extract auditory context. However, this assumes that we are able to describe, beforehand, the necessary auditory events to the extent where we can segment and labels them for training. In this chapter, we explore the possibility of automatically finding the appropriate auditory events to achieve some level of auditory scene segmentation.

Sound Scene Segmentation

Methods

A sound scene is composed of sound objects. The sound objects within a scene can be randomly dispersed (e.g. cars and horns on the street) or have a strict time-ordered relation (e.g. the process of entering your office building). Thus, to recognize a sound scene it is necessary to recognize both the existence of constituent sound objects and their relationships in time.

We want to avoid manually labeling sound scenes in order to build their models. Thus, the approach we take is to build a scene segmentor using only unsupervised training. Such a segmentor does not need to perform with high accuracy. However, a low miss rate is required because the boundaries produced will be used to hypothesize contiguous sound scenes. Actual models of sound scenes can then be built with standard MMI (maximum mutual information) techniques.
Figure 4: The basis of scene change detection is the shift in sound object composition.

Since most sound scenes are identifiable by the presence of a particular group of sound objects it may be possible to segment sound scenes before knowing their exact ingredients. An unsupervised training scheme, such as clustering, can segment data according to a given distance metric. The appropriate distance metric for segmenting sound scenes (i.e. detecting scene changes) would measure the difference in sound object composition (as in Figure 4).

The first step is to cluster audio segments (obtained with event detector of Chapter 2) by their average spectral content. We use the K-Means clustering algorithm to cluster the segments into \( N \) different clusters. The events in each of these clusters are used to train \( N \) HMMs, \( \{ c_1, ..., c_N \} \). These HMMs are our temporal models for the \( N \) classes of events. They are the canonical sound objects. Next we represent the “distribution” of sound objects by the vector of likelihoods of the sound object HMMs just extracted. The likelihood for each HMM is calculated with the Forward algorithm using a fixed sliding window (see Appendix for details). The result of is a likelihood function \( L_i[n] \) in time, \( n \), for each HMM, \( i \). So at time, \( n \), our representation for the distribution of sound objects is the likelihood vector, \( D[n] \):

\[
D[n] = \begin{bmatrix}
L_1[n] \\
\vdots \\
\vdots \\
L_N[n]
\end{bmatrix}
\]

Now, during a particular sound scene the trajectory of \( D[n] \) clusters in a particular section of the space. During a sound scene *change* the trajectory should shift to a new
region. Therefore, it is the rate of change of the vector, \( D[n] \), that we are using to measure the change in scene object composition. The rate is calculated as:

\[
    r[n] = \| D[n] - D[n-1] \|
\]

with a simple low-pass filter to smooth out the spiking that occurs on the order of 1-10 seconds*. Finally, \( r[n] \) was normalized to be between 0 and 1, using the observed maximum over the data. The hypothesis that we want to test is whether we can use this score, \( r[n] \), to detect scene changes. We conducted an experiment to test this hypothesis as follows.

**Experiment**

We recorded audio of a subject making a few trips to the supermarket. These data sets were collected with a lavalier microphone mounted on the shoulder and pointed forwards (see Figure 5).

Supermarket Data Set: (approx. 1 hour)

1. Start at the lab. (late night)
2. Bike to the nearest supermarket.
3. Shop for dinner.
4. Bring groceries home.
5. (turn off for the night)
6. Start at home. (morning)
7. Walk to the nearest supermarket.
8. Shop for breakfast.
9. Ride to work

![Figure 5](attachment://image.png)

**Figure 5:** A rough transcript of the data set used to test the scene segmentor. (left) The microphone setup used for the data collection. (right)

The Supermarket data was then processed for events. For the purpose of model initialization, these events were clustered into 20 different sound objects using K-Means, which uses only spectral content and ignores temporal relationships. 20 HMMs, \( \{ c_1, ..., c_{20} \} \) were then trained for each of the 20 sound objects. These HMMs

* The assumption here is that a scene must last longer than 10 seconds.
represent the canonical sounds appearing throughout the data and use the temporal and spectral information in each event for discrimination. Likelihoods in time were calculated for each of these HMMs and a time-trajectory in a 20-dimensional space (of canonical sounds) is the result (the $D[n]$ vector). After calculating the rate of change of this trajectory and filtering we obtain the plot in the top panel of Figure 6. In order to evaluate these results we also hand-labeled some scene transitions (middle and last panel).

It turns out that we have with typical unsupervised clustering techniques been able to detect 9 of the 10 scene transitions (to within a minute) using a threshold set at 0.2. The graph also shows a few transitions (2 of 11) detected where the hand-labels have none. The long-term goal is to use these hypothesized scene boundaries to build scene classifiers. Hence, the cost of these false positives is low since breaking a scene into pieces still allows us to recognize the whole. The cost of missing a possible scene boundary is high because combining two scenes will prevent the detection of either in isolation.

Interestingly, the most prominent false detection (see Figure 7) was caused by some construction occurring in the supermarket that day. In terms of the scenes we were interested in (biking, home, supermarket) there was no scene change (the construction was a part of the supermarket scene). However, this raises the question of how pertinent the labeling is. If anything it should indicate how difficult it is to manually label auditory (or visual) scenes.
Scene Segmentation Results

Figure 6: Detection score, \( r[n] \) plotted against the Ground Truth labeling. The threshold of 0.2 is used to calculate the accuracies.

90% Detection  18% False Alarm

Is this really an error?

Figure 7: The circled error was triggered because of the loud sounds of construction going on in the supermarket.
Chapter 4: Audio & Video Segmentation of Events & Scenes

In this chapter we introduce a new modality, vision, that will allow us to distinguish more types of environmental context. We develop clustering algorithms for dealing with the visual channel in addition to the auditory one. The major difference being in the definition of 'events' for these two channels. We also redefine the task slightly to show its similarities and dissimilarities to the multimedia indexing task.

The Personal Audio-Visual Task

We now move from thinking about the problem of extracting environmental context as a classification task to an indexing task. The goal is to learn similarity measures that will allow applications to make predictive and inductive decisions about the current situation by observing past situations. In contrast to subject-oriented video or audio [14], such as TV [15, 16], movies, and video recordings of meetings [17], our goal is to use audio & video to monitor an individual’s environment and extract a useful description of his history. Literally, the camera and microphone become an extra set of senses for the user. [11, 18]

The Approach

Since we are still trying to extract a scene-level description of the environment, it is necessary to describe first the events that make up the scenes. However, the energy-based event detector we have been using so far is not applicable to the visual channel. There are now three types of events: 1. The usual audio-only event such as speech, 2. The vision-only event such as someone turning on the lights, and 3. Audio-visual events where the auditory and visual events are tightly coupled such as the arrival of a subway train. A method for extracting these events should be able to deal with all of these situations.

We now describe a method for doing this by first combining the audio and visual data into one simple representation and then finding events by clustering in both time and
model spaces. We will find that this event clustering procedure is also well-suited for finding scenes (i.e. the relationships amongst events).

**Feature Extraction**

Unlike the typical features used for face and speech recognition, we require features that are much less sensitive. We want our features to respond only to the most blindingly obvious events—walking into a building, crossing the street, riding an elevator. Since, our system is restricted to unsupervised learning, it is necessary to use robust features that do not behave wildly or respond to every change in the environment—only enough to convey the ambiance.

**Video:** First the $(r,g,b)$ pixel values were separated into (pseudo) luminance and chrominance channels:

\[
I = r + g + b \quad I_r = r/I \quad I_g = g/I
\]

The visual field of the camera was divided into 9 regions that correspond strongly to direction. The following features were extracted from the $(I,I_r,I_g)$ values of each region:

For each region the following were calculated:

Mean: $[\bar{I} \quad \bar{I}_r \quad \bar{I}_g]$

Covariance:

\[
\begin{bmatrix}
I^2 & I_r I & I_g I \\
I_r I & I_r^2 & I_r I_g \\
I_g I & I_r I_g & I_g^2
\end{bmatrix}
\]

Table 3: The 9 features on the right were extracted from each of the 9 regions depicted in the center.

Hence, we are collapsing each region to a Gaussian in color space. This rough approximation lends robustness to small changes in the visual field, such as distant
moving objects and small amplitude camera movement (the human body is not a stable camera platform).

*Audio:* Auditory features were extracted with 25 Mel-scaled filter banks. The triangle filters give the same robustness to small variations in frequency (especially high frequencies), not to mention warping frequencies to a more perceptually meaningful scale. Both the video and the audio features were calculated at a rate of 10Hz.

**Time Series Clustering**

The algorithm we used to cluster time series data is a variation on the Segmental K-Means algorithm [19]. The procedure is as follows:

1. *Given:* $N$, the number of models, $T$ the number of samples allocated to a state, $S$, the number of states per model, $f$ the expected rate of class changes.

2. *Initialization:* Select $N$ segments of the time series each of length $T\delta$, spaced approximately $1/f$ apart. Initialize each of the $N$ models with a segment, using linear state segmentation.

3. *Segmentation:* Compile the $N$ current models into a fully-connected grammar. A nonzero transition connects the final state of every model to the initial state of every model. Using this network, resegment the cluster membership for each model.

4. *Training:* Estimate the new model parameters using the Forward-Backward algorithm [19] on the segments from step 3. Iterate on the current segmentation until the models converge and then go back to step 3 to resegment. Repeat steps 3 and 4 until the segmentation converges.

We constrained ourselves to left-right HMMs with no jumps and single Gaussian states.
**Time Hierarchy**

Varying the frame-state allocation number directs the clustering algorithm to model the time-series at varying time scales. In the *Initialization* step, this time scale is made explicit by $T$, the frame-state allocation number, so that each model begins by literally modeling $TS$ samples. Of course, the reestimation steps adaptively change the window size of samples modeled by each HMM. However, since EM is a local optimization the time scale will typically not change drastically from the initialization. Hence, by increasing the frame-state allocation we can build a hierarchy of HMMs where each level of the hierarchy has a coarser time scale than the one below it.

**Representation Hierarchy**

There are still important structures that just clustering at different time scales will not capture. For example, suppose we wanted a model for a supermarket visit, or a walk down a busy street. As it stands, clustering will only separate specific events like supermarket music, cash register beeps, walking through aisles, for the supermarket, and cars passing, crosswalks, and sidewalks for the busy street. It will not capture the fact that these events occur together to create scenes, such as the supermarket scene, or busy street scene. (Notice that simply increasing the time scale and model complexity to cover the

![Figure 8: The representation hierarchy of HMMs that allows us to model and cluster scenes HMMs as collections of events HMMs.](image-url)
A typical supermarket visit is not feasible for the same reasons that speech is recognized at the phoneme and word level instead of at the sentence and paragraph level. We address this shortcoming by adapting a hierarchy of HMMs much like a grammar (see Figure 8). So beginning with a set of low-level HMMs, which we will call event HMMs (like phonemes), we can encode their relationships into scene HMMs (like words). The process is as follows:

1. **Detection**: By using the Sliding Forward algorithm (see Appendix) with a sliding window of length \( \Delta t \), obtain the likelihood, \( L_\lambda(t) = P(O_1, \ldots, O_{t+\Delta t} | \lambda) \), for each object HMM, \( \lambda \), at time, \( t \).

2. **Abstract**: Construct a new feature space from these likelihoods,

\[
F(t) = \begin{bmatrix}
L_1(t) \\
. \\
. \\
L_N(t)
\end{bmatrix}
\]

3. **Cluster**: Now cluster the new feature space, \( F(t) \), into scene HMMs using the algorithm from the Time Series Clustering section.

**Data Collection**

In order to sample the visual and aural environment of a mobile person adequately, the sensors should be small and have a wide field of reception. The environmental audio was collected with a lavalier microphone (the size of a pencil eraser) mounted on the shoulder and directed away from the user. The environmental video was collected with a miniature CCD camera (1/4” diameter, 2” long) attached to a backpack (pointing backwards). The camera was fitted with a 180° wide-angle lens giving an excellent view of the sky, ground, and horizon at all times. The system was worn around the city for a few hours, while the wearer performed typical actions, such as shopping for groceries, renting a video, going home, and meeting and talking with acquaintances. The resulting recording
covered early to late afternoon. The camera's automatic gain control was used to prevent saturation in daylight.

**Results**

We evaluated our performance by noting the correlation between our emergent models and a human-generated transcription. Each cluster plays the role of a hypothesis. A hypothesis is verified when its indexing correlates highly with a ground truth labeling. Hypotheses that fail to correlate are ignored, but kept as "garbage classes". (Hence, it is necessary to have more clusters than "classes" in order to prevent the useful models from having to model everything.) In the following experiments we restricted the system to two levels of representation (i.e. a single object HMM layer and a single scene HMM layer). The time scales were varied from 3 secs to 100 secs for the object HMMs, but kept at 100 secs for the scene layer.

**Short Time Scale Object HMMs**

In this case, we used a 3 sec time-scale for each object HMM and set the expected rate of class changes, \( f \), to 30 secs. As a result, the HMMs modeled events such as doors, stairs, crosswalks, and so on. To show exactly how this worked, we give the specific example of the user arriving at his apartment building. This example is representative of the performance during other sequences of events. Figure 9 shows the features, segmentation, and key frames for the sequence of events in question. The image in the middle represents the raw feature vectors (top 81 are video, bottom 25 are audio). Notice that you can even see the user's steps in the audio spectrogram.

**Long Time-scale Object HMMs**

Here we increase the time-scale of the object HMMs to 100 secs. The results are that HMMs model larger scale changes such as long walks down hallways and streets. We give some preliminary results for the performance of classification as compared to some hand-labeled ground truth. Since we did no training with labeled data, our models did not
get the benefit of embedded training or garbage-modeling. Hence frequently the models are overpowered by a few that are not modeling anything useful. Typically this is where the system would make an application-driven decision to eliminate these models. As an alternative we present the correlation coefficients between the independently hand-

![Figure 9: This is a 2 minute sequence of the subject entering his apartment. Key frames are shown above and below, the audio/visual feature vector is shown in the middle. The vertical lines are the segmentation provided by the clustering and the thin horizontal lines are mapped to model ID.](image)

...labeled ground truth and the output likelihood of the highest correlating model. The table below shows the classes that the system was ably to reliably model from only 2hrs. of data:

<table>
<thead>
<tr>
<th>Label</th>
<th>Correlation Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Office</td>
<td>0.91</td>
</tr>
<tr>
<td>Lobby</td>
<td>0.79</td>
</tr>
<tr>
<td>Bedroom</td>
<td>0.86</td>
</tr>
<tr>
<td>Cashier</td>
<td>0.83</td>
</tr>
</tbody>
</table>

Table 4: Correlation coefficients between the clustered Event models and human-labeled ground truth.
Long Time-scale Scene HMMs

We also constructed a layer of scene HMMs that are based on the outputs of the Short Time-scale Object HMMs from above. Where before we were unable to clean classes for more complex events, like the supermarket visit and walk down a busy street, now this level HMMs is able to capture them. The following table gives the correlations for the best models:

<table>
<thead>
<tr>
<th>Label</th>
<th>Correlation Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dorms</td>
<td>0.80</td>
</tr>
<tr>
<td>Charles River</td>
<td>0.70</td>
</tr>
<tr>
<td>Necco Area</td>
<td>0.75</td>
</tr>
<tr>
<td>Sidewalk</td>
<td>0.78</td>
</tr>
<tr>
<td>Video Store</td>
<td>0.98</td>
</tr>
</tbody>
</table>

Table 5: Correlation Coefficient between clustered Scene clusters and human-labeled ground truth.

Figures 10 and 11 show the model likelihoods for the models that correlated with “walking down a sidewalk” and “at the video store”. While the video store scene has elements that overlap with other scenes, the video store model is able to cleanly select only the visit to the video store. See Appendix C for a diagram depicting all the scenes extracted from the data.
Figure 10: The Sidewalk Scene: above is the independently hand-labeled ground truth, below is the likelihood of the most correlated model.

Figure 11: The Video Store Scene: above is the independently hand-labeled ground truth, below is the likelihood of the most correlated model.
Appendix A

Data Collection for Speaker Identification in a Different Environments

We collected our data for training and testing using an Automated Teller Machine (ATM) scenario. The data collection was part of a larger project to recognize people using both audio (speaker identification) and video (face recognition). For more details on the project and how the classifiers were fused please refer to [20] The setup included a single camera and microphone placed at average head height. A speech synthesis system was used to communicate with the subjects rather than displaying text on a screen. The reasons for this are two-fold. First, the subjects won't be constrained to face the screen at all times. Second, it is more natural to answer with speech when the question is spoken as well. The subjects were instructed to behave as if they were at an actual ATM. No constraints were placed on their movement and speech.

The session begins when the subject enters the camera's field of view and the system detects their face. The system then greets the person and begins the banking transaction. A series of questions were asked and after each question the system waited for a speech event before proceeding to the next question. A typical session was as follows:

Wait for a face to enter the scene
System: "Welcome to Vizbank. Please state your name"
User: "Joe Schmoe."
System: "Would you like to make a deposit or a withdrawal?"
User: "Ummm, withdrawal."
System: "And the amount please?"
User: "Fifty dollars."
System: "The transaction is complete. Thank you for banking with us"
Wait for the face to leave the scene
Go back to step 1 and repeat.
During the transaction process the system saves audio at 16 KHz. Data was collected from 26 people. Since the replies were so short we also had each person recite Lewis Carroll’s *Through the Looking-Glass* poem:

And, as in uffish thought he stood,
The Jabberwock, with eyes of flame,
Came whiffling through the tulgey wood,
And burbled as it came!

One, two! One, two! And through and through
The vorpal blade went snicker-snack!
He left it dead, and with its head
He went galumphing back.

“And hast thou slain the Jabberwock?
Come to my arms, my beamish boy!
O frabjous day! Callooh! Callay!”
He chortled in his joy.

‘Twas brillig, and the slithy toves
Did gyre and gimble in the wabe:
All mimsy were the borogoves,
And the mome raths outgrabe.
Appendix B

The Sliding Forward Algorithm

This is a modification of the well-known Forward Algorithm [19] for computing a running likelihood of an observation sequence given the HMM. The traditional Forward Algorithm operates on a fixed-length window of observations, \( O(t) = O_{t+1}O_{t+2}...O_{t+T} \), to calculate the probability of observation, \( P(O(t) \mid \lambda) \), given HMM \( \lambda \). It does this efficiently in \( O(N^2T) \) time, where \( N \) is the number of HMM states. The length of the window, \( T \), depends on the details of model, \( \lambda \), and how far back the user is willing to let observations affect the answer. However if we want to compute the function, \( P(O(t) \mid \lambda) \), for \( t = 0,1,2,\ldots \) using the traditional Forward Algorithm, we would need \( O(N^2T) \) calculations for each time step. This is wasteful if \( T > 1 \) because then the windows overlap. The Sliding Forward Algorithm uses the results of calculating \( P(O(t) \mid \lambda) \) to speed up the calculation of \( P(O(t+1) \mid \lambda) \) in a manner similar to the moving average. Before going further you should be familiar with the scaling procedure in Rabiner’s HMM Tutorial [19].

When using the scaled Forward Algorithm, the model likelihood is calculated as:

\[
\log[P(O \mid \lambda)] = -\sum_{t=1}^{T} \log c_t
\]

where \( c_t \) is the scale factor for time \( t \). The crucial step is to realize that we can calculate \( c_t \) from \( c_{t-1} \). Since, the scale factors are defined as,

\[
c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)}
\]

where \( \alpha_t(i) \) is the forward-variable of state \( i \) at time \( t \) (i.e. \( P(O_1...O_t,q_t = i \mid \lambda) \)). These are found recursively as:

\[
\alpha_{t+1}(j) = \left[ \sum_{i=1}^{N} \alpha_t(i)a_{ij} \right] b_j(O_{t+1})
\]
where $a_{ij}$ is the transition probability from state $i$ to state $j$, and $b_j(O)$ is the emission probability of observation $O$ by state $j$. Therefore we can obtain $c_i$ from only the $\alpha_{t-1}(i)$'s. Which finally let’s us define the running likelihood as (also refer to ?):\[
L_\lambda(t) = P(O(t) | \lambda) = \sum_{\tau=-T}^{t} \log c_\tau
= \sum_{\tau=-T}^{t-1} \log c_\tau + \log c_t - \log c_{t-T}
= L_\lambda(t-1) + \log c_t - \log c_{t-T}
\]
where $T$ is the window length. So in conclusion, this algorithm only requires that we keep the $\alpha_{t-1}(i)$’s of the last time step, and a circular array of $c_i$’s for the last T time steps. This is $O(N^2)$ calculations per time step.

**Figure 12:** A visual description of the Sliding Forward algorithm as a moving window (length $T$) that accumulates the log-scale factors ($c_\tau$) which are successively calculated from the forward variables. The middle panel shows the observations ($O$) which are by the way a male saying “one, two, three” and then a female saying “one, two, three”. The bottom panel shows the running likelihood ($L_\lambda$) of an HMM trained on the male’s speech. Notice how the likelihood is higher when the male speaks and lower elsewhere.
Appendix C

Scene Segmentation

<table>
<thead>
<tr>
<th>Campus</th>
<th>Charles River</th>
<th>Necco Area</th>
<th>Home</th>
<th>Super-Market</th>
<th>Home</th>
<th>Video Store</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Campus.jpg" alt="Image" /></td>
<td><img src="Charles_River.jpg" alt="Image" /></td>
<td><img src="Necco_Area.jpg" alt="Image" /></td>
<td><img src="Home.jpg" alt="Image" /></td>
<td><img src="Super-Market.jpg" alt="Image" /></td>
<td><img src="Home.jpg" alt="Image" /></td>
<td><img src="Video_Store.jpg" alt="Image" /></td>
</tr>
</tbody>
</table>

2 hrs.

<table>
<thead>
<tr>
<th>Media Lab</th>
<th>Campus</th>
<th>Small Grocer</th>
<th>Home Entrance</th>
<th>Green Street</th>
<th>Super-Market</th>
<th>Central Square</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Media_Lab.jpg" alt="Image" /></td>
<td><img src="Campus.jpg" alt="Image" /></td>
<td><img src="Small_Grocer.jpg" alt="Image" /></td>
<td><img src="Home_Engress.jpg" alt="Image" /></td>
<td><img src="Green_Street.jpg" alt="Image" /></td>
<td><img src="Super-Market.jpg" alt="Image" /></td>
<td><img src="Central_Square.jpg" alt="Image" /></td>
</tr>
</tbody>
</table>
Appendix D

Software and Interface for HMM training and classification:

This picture shows the labeling and training interface for creating the HMMs to be used in the classifier.
This picture shows the classifier in action, with the top 6 panels displaying HMM normalized likelihoods for the real-time data streaming by on the bottom panel (as a scrolling spectrogram).
Bibliography


