

---

# Structure out of Sound

Michael Jerome Hawley

B.S. Computer Science, Music, Yale University, 1983

Submitted to the Program in Media Arts and Sciences,  
School of Architecture and Planning  
in partial fulfillment of the requirements for the degree of  
**Doctor of Philosophy** at the  
**Massachusetts Institute of Technology**

September 1993

© Massachusetts Institute of Technology, 1993. All rights reserved.

---

Author:

\_\_\_\_\_  
Program in Media Arts and Sciences  
August 6, 1993

Certified by:

\_\_\_\_\_  
**Marvin Minsky**  
Toshiba Professor of Media Technology, MIT  
Thesis Supervisor

Accepted by:

\_\_\_\_\_  
**Stephen A. Benton**  
Chairman, Departmental Committee on Graduate Students  
**Rotch**

MASSACHUSETTS INSTITUTE  
OF TECHNOLOGY

OCT 18 1993



## Abstract

---

# Structure out of Sound

Michael Hawley

*submitted to the Program in Media Arts and Sciences  
on August 6, 1993 in partial fulfillment of the requirements  
for the degree of Doctor of Philosophy in the field of  
Media Arts and Sciences.*

---

Sound is the predominant communication modality in the natural world, yet most current computers have no sense of sound whatsoever. They only occasionally "bleep," and are, in effect, stone-deaf. This is a serious deficiency. General audio capability must be developed if machines are to function in more natural contexts.

The thesis presented here is that machines must be able to extract meaningful structure in order to operate intelligently on sound. They must "hear" events, and recognize voices, effects, melodies, moods, room-acoustics, and many other features. This work demonstrates how such sound recognizers may be built and unified in a fruitful framework.

In particular, several implementations of sound sensors are presented, including a polyphonic pitch extractor, a music detector, and a talker recognizer for indexing conversations. These programs filter sound to produce event-list scripts that describe the contents in some sense. In this way, many sorts of sound streams may be abstracted and rendered amenable to processing by a wide variety of applications. New applications in music and speech synthesis as well as content-oriented filtering of video are shown.

---

Supervisor: **Professor Marvin Minsky**  
Title: **Toshiba Professor of Media Technology, MIT**

---


*This work was supported by sponsors of the **Movies of the Future Program**, including:  
**Apple Computer, AT&T, Bellcore, Columbia Pictures Entertainment,  
Eastman Kodak Company, Intel Corporation, Paramount Pictures Inc.,  
Warner Brothers, Inc., and Viacom.***



## Doctoral Committee

---

Advisor:




**Marvin Minsky**  
Toshiba Professor of Media Technology, MIT

Reader:



**Andrew Lippman**  
Assistant Director, MIT Media Laboratory

Reader:



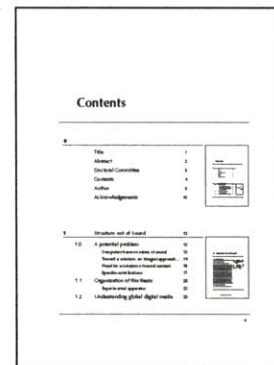
**Robert Sproull**  
Fellow, Sun Microsystems



# Contents

0

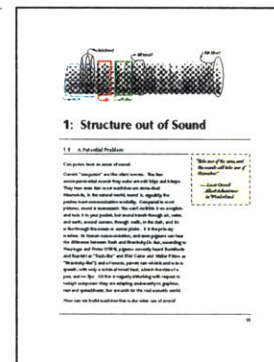
Title	1
Abstract	3
Doctoral Committee	5
Contents	7
Author	12
Acknowledgements	13



1

## Structure out of Sound 15

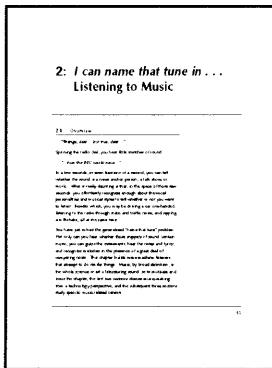
1.0	A potential problem	15
	Computers have no sense of sound	15
	Toward a solution: an integral approach...	17
	Need for a notation of sound content	19
	Specific contributions	20
1.1	Organization of this thesis	23
	Experimental apparatus	24
1.2	Understanding global digital media	25



	"How The World Was One"	25
	The information inversion	27
	<i>Parable of the ant — revisited</i>	27
	Some specifics	29
	<i>The internet</i>	30
	<i>The Library of Congress as bookstore . . .</i>	32
	Listen to the Net	34
1.3	Related work on sound understanding	35
	Visible Speech	36
	Perceptual research — auditory scene . . .	37
	Implementations of the general auditory. . .	39
	Other precedents from speech science	41
	Other developments in audio . . .	42
	<i>Low-level structural manipulations . . .</i>	42
	<i>High-level processing</i>	43
	What about the general problem?	46
	Summary	48
1.4	Remarks	49
	End of the silent computing era	49
	Broad versus deep	50

## 2 "I can name that tune in ..." 53

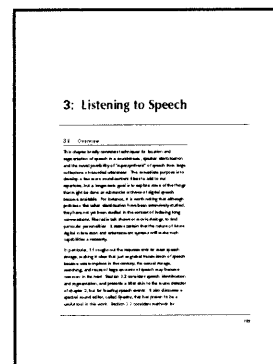
### Listening to Music



2.0	Overview	53
2.1	Revolutionary études	55
	The rise of the <i>instrument-savant</i>	55
	Historical notes	57
	<i>Cathedrals and the end of chant</i>	58
	<i>Baroque high-technology</i>	58
	<i>Pianos and the industrial onslaught</i>	60
	<i>Electronics and the wane of live music</i>	61
	<i>Musical automata</i>	62



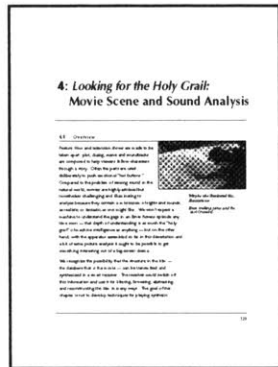
	Summary: towards the longest musical lever	64
2.2	Experiences with MIDI	65
	Musical building blocks and glue	65
	BSO in a box	65
	First studies	66
	Arranging studies	69
	A piano roll reader	75
	Summary	78
2.3	A music detector	78
	Measuring harmonic entropy to find . . .	79
	Implementing the music detector	81
	Testing the music detector	85
	Limitations and extensions	86
	<i>A music elimination filter</i>	87
2.4	A polyphonic pitch extraction filter	89
	Two clues	90
	The nature of piano sound	91
	Finding and identifying note onsets	94
	Testing the polyphonic pitch extractor	97
	Limitations and extensions	98
2.5	Name that tune	100
	Name that tune for MIDI	100
	Name that tune from polyphonic audio	102
2.6	Remarks	106
<b>3</b>	<b>Listening to Speech</b>	<b>109</b>
3.0	Overview	109
3.1	Sizing up Speech	110
3.2	Finding and Segmenting Speech	112
	A split-comb to locate voiced speech	113
	The speech event detector	115
	<i>Spectre: a spectral sound editor</i>	115



3.3	Identifying Talkers	118
	Typical approaches to speaker recognition	120
	Average baseline pitch in determining ...	121
3.4	Supersynthesis	123
3.5	Remarks	127

### Looking for the Holy Grail:

## 4 Movie Scene and Sound Analysis 129



4.0	Overview	129
4.1	A close listen to <i>Mr. Ed</i>	130
	Indexing the elements of a sitcom	130
	Footsteps and Doorslams	131
	Cows, horses, laughtracks, and other effects	137
4.2	Visual Scene Analysis	138
	Scene parsing with <i>Media Whacker</i>	137
	Limitations and extensions	140
4.3	The Holy Grail	141
	The quintessential adventure	141
	Hearing the clues	142
	Finding the grail	143
4.4	Remarks	144

## 5 Conclusions 147



5.0	Recap — What about the Pigeon?	147
5.1	Contributions	149
5.2	Lacunae and Future Work	153



a1.0 Overview



a2.0 Introduction

a2.1 What rich-text code looks like

a2.2 Quick overview

a2.3 Remarks

a2.4 Summary



## Author

---

**Michael Hawley** received degrees in music and computer science from Yale University in 1983. He has pursued research in operating systems, cognitive psychology, computer music, and digital cinema at Bell Laboratories in Murray Hill, IRCAM in Paris, and Lucasfilm in San Rafael. Recently he was a member of the team that created the NeXT computer, for which he implemented several seminal digital books, including Merriam-Webster's dictionary and the first digital edition of the works of William Shakespeare. Michael is also occasionally active as a concert pianist.

Upon completing his doctorate he will hold the J.C.R. Licklider Chair as an Assistant Professor of Media Technology in the MIT Department of Electrical Engineering and Computer Science, in conjunction with the Laboratory for Computer Science and the Media Laboratory.

## Acknowledgements

---

Deep thanks to professors Steve Benton, Tod Machover, Nicholas Negroponte, and Barry Vercoe for outstanding support and stimulation of all kinds that has improved my work and my life at MIT. My gratitude extends especially to committee members Bob Sproull and particularly Andy Lippman, for many thoughts and insights. (It was also Andy who persuaded me to take up rollerblades.) In addition, thoughtful comments from Andy Moorer were deeply appreciated, as well as Tom Stockham, who made a very special visit on my behalf. I feel privileged to have been the focus of so much goodwill.

I am also one of a few lucky people who can thank Marvin Minsky as a surrogate father. Marvin was not simply my advisor — he and the whole Minsky family (Gloria, Margaret, Julie, Henry, Milan, Adrian, Silas, Whiskers, Claudia...) essentially adopted me. I lived in the attic of the Minsky mansion in Brookline in my early time at MIT, and spent some wonderful evenings up in the garret reading, downstairs chatting or making music, or occasionally helping Marvin randomize the plumbing. That sort of nurturing relationship is more than most students dream of, and in retrospect it is hard to imagine how anyone could build a PhD without that kind of backing — or how I could possibly repay the favor. I hope my virtual family got a fraction of fun in return.

I have been the beneficiary of a good deal of equipment, from computers to grand pianos. Wayne Stahnke, Jim Turner, Hal Vincent, Tony Habig, John Amuedo, Tadao Kikumoto, Eran Egozy, Ray Kurzweil, Bill Joy, Laura Tong, Dave Cumming, Garth Zeglin and many others helped make it materialize. Greg Tucker, Bob Greene and Nicholas helped find a place to put it all.

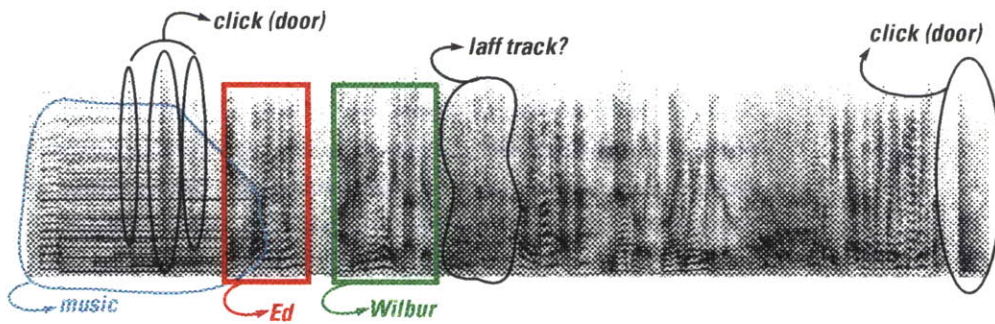
Steve Jobs has helped in ways too numerous to mention, and I am grateful and honored to have had the opportunity to contribute at NeXT. During that time, the Media Laboratory let me put my doctoral work in a somewhat ticklish position, disappearing for months at a stretch to the bit mines of California. I like to think such mingling of academic thought and worldly currents is the essence of MIT at its best, but I know this created complications, and MIT was most indulgent on my behalf, for which I am thankful.

Officemate Robert Rowe and virtual officemate Henry Massalin have shared many thoughts and ideas, leaving a deep impression on my work, which I know they will recognize. Henry in particular has furnished a number of illuminating insights about signal processing and operating systems; his fast Hartley transform code was enormously useful. Dan Ellis is one of the few others at MIT working along similar lines, and has provided terrific intellectual leads; the mere existence of a kindred soul or two is refreshing when the field seems so empty. Janet Cahn and Barry Arons provided a number of helpful hints about the nature of speech. Brian Redman revived my interest in Mr. Ed at just the right moment. Michael Schrage saw me through some lean stretches and has helped immensely all along the way. Sam Joffe has been a close friend and housemate throughout, as well as a careful and thoughtful reader. His comments greatly improved the writing. Linda Peterson miraculously held the whole academic bundle together.

Several dear friends helped hold *me* together — to risk naming far too few, Charisse Castagnoli, Dominic Frontiere, Murray Goldman, Leo Hourvitz, Gillian Lee, Mark Lucente, Mort Meyerson, Margaret Minsky, Ken Phillips, Wendy Plesniak, Olin Shivers, Kate Smith, John Underkoffler, and Gaye Williams are some of many who were there when it mattered. I hope I will be forgiven for leaving unnamed my much larger family of friends and colleagues at MIT, NeXT, Lucasfilm, and many wonderful friends who have expanded my thinking, and helped in countless ways.

I am not ashamed to admit that I've had as wonderful and fortunate a graduate life as one could wish for. This is due in no small measure to the Media Laboratory itself which has been an ideal place to live a research life; and that, in turn, is due in no small measure to the brobdingnagian vision of Nicholas Negroponte and the many sponsors that have joined in these explorations. My work was generously supported by the Movies of the Future research program.

Finally, I hope my parents, Mary Kay and George, and my brothers, Pat and Steve will enjoy finding a few familiar echoes in some of this work.



# 1: Structure out of Sound

## 1.0 A Potential Problem

**Computers have no sense of sound.**

Current “computers” are like silent movies. The few accompanimental sounds they make are odd blips and bleeps. They hear even less: most machines are stone-deaf. Meanwhile, in the natural world, sound is, arguably, the predominant communication modality. Compared to most pictures, sound is evanescent. You can’t scribble it on a napkin and tuck it in your pocket, but sound travels through air, water, and earth, around corners, through walls, in the dark, and for miles through the ocean or across plains. It is the primary medium for human communication, and even pigeons can hear the difference between Bach and Stravinsky (in fact, according to Neuringer and Porter (1984), pigeons correctly heard Buxtehude and Scarlatti as “Bach-like” and Eliot Carter and Walter Piston as “Stravinsky-like”); and of course, parrots can whistle and mimic speech, with only a minimal vocal tract, a brain the size of a pea, and no lips. All this is vaguely disturbing with respect to today’s computers: they are adapting exuberantly to graphics, text and spreadsheets, but are unfit for the real acoustic world.

How can we build machines that make wiser use of sound?

*“Take care of the sense, and  
the sounds will take care of  
themselves.”*

— Lewis Carroll  
*Alice’s Adventures  
in Wonderland*

The answer, of course, is that machines need to understand sound, and to operate all along the continuum between a sound wave and representations of its content. That is not a succinct or singular concept, however. Grand challenges, like creating conversational speech systems or interactive music systems, are among many largely unsolved problems that are subsumed therein; so are more basic functions that have recently begun to appear in workstations for recording, encoding, and rendering sounds. In the complex acoustic world, countless moving parts jiggle and collide. Their sound reflects something of the mechanics and behavior of the system, including information about features such as size, shape, location, motion, or more elusive characteristics of personality, like mood or emotion in speech or music. Understanding those sounds and all their subtleties gives rise to a perceptual mirror-world that is just as convoluted (as well as the ubiquitous tangle of patch cables).



*You can't compete with radio.*  
Rube Goldberg (1929)

In the face of this noisy complexity, human hearing shows a remarkable ability to “diagnose” sound (to use S. S. Stevens’ word). Although we have many techniques for putting structure *into* sound, taking it out is a lot harder. In most respects we have barely begun to understand the workings of the human auditory system well enough to emulate them in machines. Surely we want interfaces that can listen, understand, and converse naturally with us, and models of sound that are relevant to our sensibilities. But just as surely, we want machines to sense and synthesize sound, doing many practical



things with sound in an infinite variety of ways that humans alone cannot. As powerful and general-purpose as it is, human audio capability is nevertheless highly specialized. Consider the subtle alignment of human ears and voices: the ear canal is well-sized to work in the frequency range of speech. Elephants have much larger ears (and throats) so they can listen and “speak” in the sub-audible (to humans) 8-Hz band – a useful portion of the spectrum to pick, since low frequency sounds travel farther and are easier to orient (especially if your ears fan out like a dish antenna). Bats use high-frequency chirps for guidance, and owls have sophisticated spatial audio abilities for night hunting. Human abilities, while a relevant and lofty goal, are not the only goal, and perhaps not even the most important one. Moreover, the most interesting applications of new technologies are often the unexpected ones. Remember that when Edison and his team developed the phonograph, he thought it was going to be a voice dictating machine: he had no inkling of the fact that music would be its chief application, by far, and that making music disks would become a major entertainment industry. Certainly, future entertainment media will afford many interesting and as yet unanticipated opportunities for audio applications. In any case, the point is that while much is known about the nature of many kinds of sounds and about techniques for processing them, little of that knowledge has found its way into day-to-day systems.



*Talking to the Phonograph*

Edison phonograph  
(*Harper's Weekly*,  
March 30, 1878)

### **Toward a solution: an integral approach to sound understanding**

Would you rather be blind or deaf? Fortunately this is a choice most of us never need to make. But consider the question from a different perspective, as a designer of machines. There has already been a great deal of work on interactive graphics and visual modalities. Machine vision is a well-known field; but “machine hearing” is not. As a designer, the corresponding question might be: would you rather have a visually adept but deaf machine, or one that has limited graphics (color- or stereo-blind, say) but has a well-developed sense of sound and a voice

that is as friendly as its graphical icons? This is almost a non-question. Yet take apart today's "workstation" and you will find thousands of dollars in parts for imaging — video hardware, an expensive color monitor, custom graphics chips, rendering software, graphical input devices, and often attachments for printers. On the audio side, there may be a 3" speaker, a peculiar sound chip, and possibly an input jack for a cheap microphone. Something is wrong with this picture.

Central to this thesis is the idea that sound — *general* sound — must become an integral modality in machine-mediated communication. Information machines need to operate fluently in the acoustic world. The idea of an audio-rich computer suggests a markedly different context for "computing" than has so far been used, and that is certainly a strong departure from the "silent movie" world of window systems and spreadsheets. The future machine that understands you when you say "dim the lights," the language-translating "vidphone," tomorrow's music machines, and the entity that your home entertainment system eventually becomes, all will function as *one* machine, because the information will be shared in a single digital form. Thus, what once was a peculiar piece of audio hardware, like a "reverberator" or a musical instrument, will become weightless software in a general machine. Although there have by now been several audio workstations and a piecemeal array of audio peripherals, we need to architect the acoustic aspects of systems in a far more unified way. This trend may seem obvious to some, but it is not at all the mainstream view.

Most important, to operate intelligently on a sound ultimately depends on sensing structure in the wave that describes meaningful aspects of the source. Our inability to build machines that can do this is partly why sound has been so poorly used in interfaces. Until such senses exist, sound will continue to be too slippery: operations will grasp little of the salient content, and machines will not make good use of it.

The problem is not so much to develop machine analyzers and indexers that can identify the source attributes of particular acoustic signals — sound effects recognizers, ambience sensors, musical timbre identifiers and pitch extractors, speech identification and transcription systems, and so on — this thesis argues that the problem is more to knit them into representations that can be fruitfully processed and recomposed. In the recent scientific literature this analysis is often called *signal understanding* (a class of the general AI problem of sensor interpretation) to distinguish it from the classical signal processing that traditionally focuses on relatively low-level phenomena and linearly-filtered systems. The goal is to abstract the signal into a symbolic stream so that the most meaningful elements are exposed. In this way, other agencies can operate on “deeper” qualities of the source.

Indeed, the idea of operating on sound in qualitative ways, like altering the gender or mood of a voice, is a tantalizing prospect. In one future, not far off, it suggests telephones that let you pick the voice you wish to project (*Cary Grant... Lauren Bacall...*), shaping your sound for irresistible appeal — in fact, inexpensive DSP technology is already being built into phones to do things like this. It begins to allow machines to be sensitive to the joys and frustrations conveyed in speech. It suggests more immediate realities, too, like systems that intelligently select and synchronize sounds to fit a movie, or machines that search through thousands of hours of recorded audio to find elements of interest. One could imagine many such tools. One need only reflect on the impact of signal-oriented audio technology in the past hundred years — phones, broadcast and personal radio, recording and playback tools — to get an idea of what it will mean as content-oriented audio tools begin to take root.

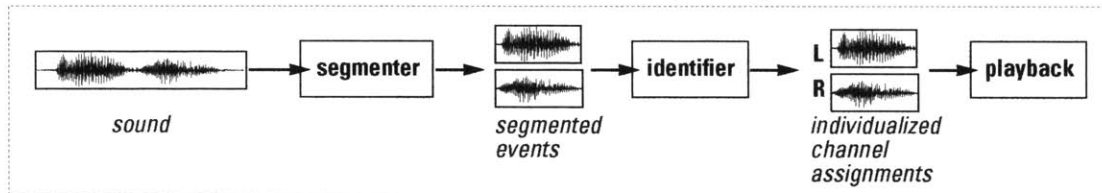


*The **Voice Changer Telephone** casts the caller's voice from male to female, child to adult, or through any of a dozen other disguises.*

### **Need for a Notation of Sound Content**

One immediate implication of this is that we need conventions for describing the features of interest in sound streams. Some of the tools in this thesis use a simple notation for indexing speech,

music, and other sound streams, extending to describe video and other sorts of data as well. It is reminiscent of data structures like “cue sheets” that support digital movie editing systems. The existence of such a notation invites a variety of tools that operate on such scripts. As an example, suppose we have a



monophonic recording of a conversation between two people and we wish to cast it to stereo, putting one talker in the left channel, another in the right (as shown in the diagram above). An *utterance segmenter* produces an initial event list, one event per utterance. This event file is then filtered through a *speaker identifier* that tags each utterance event with the name of the speaker. A speech-recognition system might do that, but simpler tools often suffice. This tagged event stream is passed through a text filter (a trivial editing script) that appends a “**pan...**” directive to each event according to the talker’s identity to indicate positioning of the sound during playback. A *playback filter* then reads this list of events and plays them back, interpreting instructions like “**pan...**” (For the moment, it is not necessary to dwell on the many subtleties of accurately isolating utterances, assigning them to talkers, or dealing with difficulties due to background noise or overlap; some of these issues are dealt with in chapter 3). This could perhaps be done in certain special purpose machines, and things like this are certainly done by hand, by Hollywood movie mixers. This little example illustrates how a few general-purpose tools, linked by the same economical event-level representation, may be composed in many ways.

*segmented utterances:*  
| 1.25-2.21 | utterance1  
| 2.37-4.52 | utterance2  
...  
*after identification:*  
| 1.25-2.21 | Harrison  
| 2.37-4.52 | Karen  
...  
*pan, according to speaker:*  
| 1.25-2.21 | Harrison | pan left  
| 2.37-4.52 | Karen | pan right  
...

### Specific contributions

This work makes some pragmatic contributions to the problem of abstracting and applying the contents of sound streams. A small one is the *event list* transcription (or *e-list*) mentioned above. An

e-list is a simple textual description of sounds that is readily amenable to processing by many other software tools. The need for representations like this is obvious; however, note that although such an encoding already exists for music (**MIDI**, the **m**usical **i**nstrument **d**igital **i**nterface), there is no comparable “MIDI for speech,” or for general sound.

The hard task is to devise ways of filtering sounds to derive such event lists. For example, a machine listener that detects “footsteps” might be built from a low-level listener that finds “clicks” and in so doing, translates a sound stream into a condensed list of “click” events; this would be followed by a higher-level analyzer that notices groups of clicks that resemble the patterns of footsteps. (This problem is discussed in chapter 4; it may seem straightforward, but in practice, a naive click-finder will trigger on many noise bursts, like consonants and glottal stops; overcoming that naiveté is not easy, and a robust footstep recognizer is not a simple thing to build). Many sound-to-event converters fit into this framework; e.g., “speech-to-text” would be a complex instance of such a filter.

Among the new filters presented here are: a polyphonic pitch finder for extracting the notes from piano recordings; a general music detector (for finding segments of music amid movie sound tracks); and a speaker identifier (for determining who-speaks-when in a conversation). All of these are difficult problems, known in the literature, and often are studied here with many limitations, as will be noted. For example, *demixing* of sound (the “cocktail party” signal separation problem, but without necessarily having the benefit of binaural information) is notoriously hard, as is the removal of ambient sounds or reverberation, or the accurate identification of effects, instrument timbres, or linguistic accents or affects. For that matter, problems like segmentation of speech utterances or music transcription have each been the topic of several dissertations. Undaunted by this, each application presented here is functioning in a useful if limited sense, and can now be studied in a productive

*“In its present form, the model requires approximately two hours of computation for every second of acoustic input.”*

— Guy Brown (1992)

programming environment. For instance, although speaker identification has been a research topic for decades, there are no systems (known to the author) that index conversations. Here, part of the solution is presented as a conventional Unix filter that reads sound and writes event lists (each event a per-speaker utterance) that can then be processed by many other programs. These analysis and resynthesis utilities often run faster than realtime.

In the course of devising these filters, some new techniques (or new twists on old techniques) were tried; these include spectral differencing (to find new notes in piano music), harmonic entropy prediction (to find music and speech), and a vowel timbral matcher (to identify speakers). Several of the filters are built atop straightforward spectral analysis (e.g., using a windowed fast Hartley transform to obtain magnitude and phase information, and tracking prominences and features in them). Some might quibble with this approach — surely a simple spectrum is not the “right” representation, and other analytic tools (wavelet-or chirp-transforms, constant-Q spectra, “correllograms,” optimal comb filters, LPC coding, etc.) are better for certain applications. Yes, that is often true, and particular signal processing details will be noted when relevant, but part of the point of this thesis is to show how many such mechanisms can and must be brought to bear in order to produce a fuller description of sound contents.

Finally, this work has produced insights into some novel ways of operating on sounds once some of the contents have been identified. In music, it appears that, at least for relatively simple recordings like a close-miked piano, one can not only find the notes, but in so doing, separate the note samples to build a model of the piano. That is to say, a recording system can be built that constructs a model of the performance by separating the information into gestural (keystroke-like) events and instrumental components. In speech processing, if an event list is sufficiently detailed so as to map the words (and their speakers)

to respective segments of sound, these can be used to fuel a powerful new class of sampled-speech synthesis systems. Whole words and phrases may be assembled into utterances, provided intonation and amplitude are contoured to fit. This is seen as a generalization of the automatic dialog replacement that is common in commercial media production. The event information gleaned from soundtracks is also used in the context of a *movie parser*, a program that filters the sound and picture parts of a film so as to help find clips and scenes of interest.

These examples illustrate ways in which a variety of high-level sound processing tasks can be combined into a more unified system.

## 1.1 Organization of this thesis

---

The remainder of this chapter provides additional background information. This includes some remarks about the emerging global context for information machines (1.2), and a discussion of relevant work that has been done to date in the area of sound understanding (1.3).

Each subsequent chapter studies a particular aspect of the problem of abstracting structure from sound and incorporating it in the larger system. These range from specific to general, as follows:

**2: *Listening to music*** — presents three sound feature detectors, one a pitch-extractor that transcribes acoustic recordings of polyphonic piano music (with certain limitations), a program that finds segments of music in general soundstreams, and a “name that tune” melody finder. Music is a particularly well-codified and highly structured form of sound, so it provides a useful point of departure. A discussion of historical trends in music and technology, as well as recent MIDI-based systems, motivates the chapter.

**3: *Listening to speech*** — the familiar problems of utterance segmentation and talker identification are extended to index

dialogs based on who-speaks-when.

The notion of smart splicing (a generalization of dialog replacement) is introduced, illustrating how spoken utterances might be constructed by future speech synthesis systems.

**4: *Movie scene and sound analysis*** — a feature film soundtrack contains a mixture of dialog, music, and effects; in fact, much of the interesting information in a film is contained in the soundtrack. It thus presents a rich and challenging domain for audio analysis. How many events in a soundtrack can be automatically detected? How can they be used in browsing? This chapter presents a movie parser that uses both sound and picture information to locate meaningful parts of a film.

**5: *Conclusions*** — a summary highlighting the interesting aspects of the work presented here, and pointers for future work.

#### **6: *References***

In addition, two appendices discuss implementation details.

This arrangement is not arbitrary. The progression develops some building blocks in the domains of music and speech (both highly structured forms of sound), applies them to full soundtracks, and integrates these components with a movie system. The fullest working taxonomy and richest synthesis of general sound is found in the movie industry: the broad categories include dialog, music, effects of several types, and ambiances. This is likely to be recapitulated in information systems, particularly as they inherit an entertainment context. The examples used here focus on dialog and music, which are most often the meaning-bearing components, with some mention of effects recognition.

#### **Experimental Apparatus**

The work done here was implemented on NeXT (release 3.0) and Sun (4.2BSD Unix) systems, using MIDI, audio, and video peripherals of several sorts. Audio input was digitized using a stereo, 16-bit, 44.1 KHz sampling microphone made by Ariel



corporation, and processed entirely in the NeXT 68040. Video was taken in through a 24-bit *NeXTDimension* board. Specific apparatus will be noted as necessary.

## 1.2 Understanding Global Digital Media

---

The science of speech processing sprouted from incentives in transmission economy: much of our audio signal processing technology derived from the acute demands of the global telephone and radio infrastructure. Making wise use of limited capacity still provides incentives, for example in managing cellular phone networks, but these seem minuscule compared to the demands of the new digital infrastructure. Sound in future interfaces will develop in that infrastructure, so it is worth reflecting briefly on it.

### “How The World Was One”

We are in the early stage of an awesome diffusion of computational power and digital information. In the global transition from analog to digital media, two properties are all-important: many diverse information formats will be unified in a single digital channel, and information of all sorts will accumulate in the collective memory of a planetful of machines.

The result will be a pervasive, information-rich climate. Future machines will be characterized not by the size of a disk or the speed of a processor, but by what they know, and by how skillfully they use that knowledge to find more. Their evolution will be spurred by the ambient information environment and implemented in software. Little more than a decade ago, the idea of a “personal computer” was a novelty: a PC the size of a suitcase could scarcely hold even a dimestore paperpack. Networks were sinewy webs tied together with telephones so that scientists could swap Fortran code by e-mail. Today, digital machines are rapidly absorbing books, music and movies, fusing with cellular phones, televisions, and pocket diaries, and working their way into the conversational mainstream. In the

*“Is it a fact . . .  
that by means of electricity,  
the world of matter has be-  
come a great nerve, vibrating  
thousands of miles in a  
breathless point of time?  
Rather, the round globe is a  
vast head, a brain instinct  
with intelligence!  
Or, shall we say, it is itself a  
thought, nothing but a  
thought, and no longer the  
substance which we deemed  
it!”*

— Nathaniel Hawthorne  
*The House of the  
Seven Gables (1851)*

space of about two decades, a machine that once was sensory deprived, large, isolated, and had little storage — essentially a *tabula rasa* — becomes one that is sensory-rich, portable, globally linked, having immense storage that is soaked with information of many kinds. Some machines will come to parse movies and television, looking for moments of interest. Some will graze global networks, ferreting for new resources. Some will be like *idiot-savants*: not generally intelligent, but able to leverage thousands of hours of music or a global electronic library in a pocket of memory. All will “compute” in the context of a vast amount of information. In sum, future “thinking machines” will rather suddenly have an awful lot to think about.

These trends are not exactly news. Recent developments are bringing cable television, phone and data networks into mass-convergence with computing. The emergence of national and global information infrastructures is a topic of daily discussion, from e-mail in the White House and federally-funded information superhighways, to a multitude of computer-mediated consumer products that are beginning to include interactive television, virtual shopping, networked education, and many other applications. Hawthorne’s vision (1851) of the electrically wired world as resembling a giant nervous system, and of the information transcending the hardware, is striking considering it dates from the dawn of telegraphy (Morse’s famous transmission of “What hath God wrought?” from Washington to Baltimore was sent in 1844, and the new transatlantic cables around 1855 would reduce the time it took news to travel between Europe and the United States from a month to a few seconds). McLuhan echoed this idea with the metaphor of a global village, as has Arthur C. Clarke in his book *How the World Was One: Towards the Tele-Family of Man* (1992).

*“In this electric age we see ourselves more and more being translated into the form of information, moving toward the technological extension of consciousness.”*

— H. Marshall McLuhan  
(1911-1980)

What *is* news to some extent is an awareness that an antidote to the numbing influence of a flood of bandwidth is the creative application of computing to get at the “meat” of those messages. Recall that at the core of McLuhan’s *Understanding Media* (1963) was the discussion of “hot” versus “cool” media.

Television, as cool, passive medium, was a sort of “unified sensorium” that “introduced a kind of *rigor mortis* into the body politic.” At that time, concepts of “interactive” computing were just being born (Ivan Sutherland’s *Sketchpad* thesis was finished in 1963), so computer applications were irrelevant to filtering sensory channels like those of television. Not surprisingly, *Understanding Media* had little to say about computers.

### The Information Inversion

Now that digital machines are beginning to acquire audio-visual modalities, and now that they clearly show the promise of helping to turn broadcast media into more engaging, imaginative and personable forms, it seems strange that so many interests are still so fixated on the dissemination of raw bandwidth instead of the interesting use of it. The nature of global tradeoffs between bandwidth and intelligence has been a frequent topic in Negroponte’s editorials (e.g., *Debunking Bandwidth*, 1993), and the argument is often a plea not for more pixels, or more channels, but for smarter machines. In any case, from the point of view of the computer scientist, there has been a marked inversion. “Computers” were once designed like cathedrals, and the problems they dealt with were bounded by whether or not they fit in through the front door in a certain sense — that is, early machines were commodious for only a few kinds of tasks. Now, computing is being built into all manner of secular appliances and even adornments, and we can see that the information that fuels these machines will, by extension through the global net, be effectively unbounded.

*“More bits per second is not an intrinsic good. In fact, more bandwidth can have the deleterious effect of swamping people and of allowing machines at the periphery to be dumb.”*

— Nicholas Negroponte  
(1993)

### *The parable of the ant — revisited*

It is helpful to recall (and reinterpret) Simon’s parable of the ant on a beach (1965/1985):

We watch an ant make his laborious way across a wind- and wave-molded beach. He moves ahead, angles to the right to ease his climb up a steep dunelet, detours around a pebble, stops for a moment to exchange information with a compatriot.

Thus he makes his weaving, halting way back to his home.  
So as not to anthropomorphize about his purposes, I sketch the path on a piece of paper. It is a sequence of irregular, angular segments – not quite a random walk, for it has an underlying sense of direction, of aiming toward a goal.

I show the unlabeled sketch to a friend. Whose path is it?  
An expert skier, perhaps, slaloming down a steep and somewhat rocky slope. Or a sloop, beating upwind in a channel dotted with islands or shoals. Perhaps it is a path in a more abstract space: the course of a search of a student seeking the proof of a theorem in geometry.

Whoever made the path. . . why is it not straight;  
why does it not aim directly from its starting point to its goal?

*An X, viewed as a behaving system, is simple. The apparent complexity of its behavior is largely a reflection of the complexity of the environment in which it finds itself.*

*X = ant,  
man,  
algorithm...*

The moral is that even a simple-seeming algorithm behaves in interesting ways once loose in an information-rich environment. Complex behavior reflects a complex world.

With this in mind, Simon and others assumed it would therefore be easy to emulate human decision-making: recall the infamous mis-forecast (Simon 1960) “*within the very near future — much less than 25 years — we shall have the technical capability of substituting machines for any and all human functions in organizations.... Duplicating the problem-solving and information-handling capabilities of the brain is not far off...[certainly] within the next decade.*” The timescale for such predictions has proven far from true. When decisions require significant general knowledge (which is to say, when the “ant” is extremely sophisticated, and when there is interesting interplay between the metaphoric ant and the world), the complexity of human responses is still only poorly emulated by machines.

Somehow, focusing on the ant seems moot. We are just starting to create a global beach where there was a void before. Understanding and interacting with the *beach* is the goal. Machine environments are growing information-rich: we need to develop ways of filtering for content. This thesis is partly an instance of that.

## Some specifics

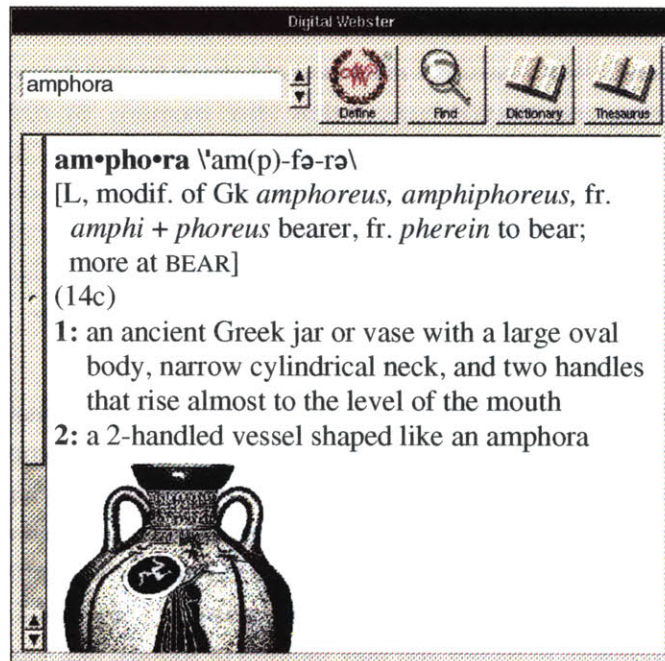
Around 1981 a personal machine held about 64,000 bytes of fast memory and 256,000 bytes of slow (disk) memory. By 2000 a.d. or shortly after, the forecast is for perhaps 1,000,000,000 bytes of fast memory and on the order of 100,000,000,000 bytes of disk, though this is hard to gauge because the virtual memory will be multiplied by millions of networked machines. Even in the time I have been engaged in doctoral research, the technology changes have been startling.

Significantly for this thesis, a few years into my work digital audio and video began to come within the grasp of common computing. In 1986, an audio-oriented computer was a high-end proposition. Workstations with microphones (and without special ancillary audio hardware) began to appear in significant numbers around 1990, and it became possible (though still uncommon) to record and play samples of sounds, like sound effects or snippets of speech, using relatively conventional computers. Now workstations commonly run in the 20-100 mips range. Thus, general-purpose machines are beginning to encroach on the territory once occupied by special-purpose audio hardware, like digital music synthesizers.

Similar progress can be seen in textual media. The limitations of paper books have been known for years, and many sermons have been given on the promise of more dynamic containers. Bush's *memex* (1945) was not the first of these, and Kahn's *knowbots* (1988) will not be the last. What is notable is that, beginning around 1985, increases in memory, software richness, and improvements in display quality and interface aesthetics, all finally combined to make interactive digital books and libraries viable *products*. Backer's *movie manual* (described in a 1988 thesis) was a research instance of that, as was Phillips' *MediaView* (1991). One of the first commercial examples was the *Digital Webster* dictionary I implemented for the NeXT computer (Hawley, 1987, 1988):

- 1945 ***memex***  
Vannevar Bush
- 1965 ***libraries of the future***  
J.C.R. Licklider
- 1969 ***intellect augmentation***  
Douglas Engelbart
- 1974 ***hypertext***  
Ted Nelson
- 1979 ***books without pages***  
Nicholas Negroponte
- 1979 ***dynabook***  
Alan Kay
- 1988 ***hypercard***  
Andy Herzfeld
- 1988 ***knowbots***  
Bob Kahn

Variations on a theme



Although forms of digital dictionaries had been in use for years (for example, in the computational linguistics community), and although the reading and writing of “e-books” was emerging as a field (e.g., Yankelovich and Meyerowitz, 1985), this was the first time it became possible to *publish* a digital “desktop” dictionary with adequate typography, embedded illustrations, and connectivity to other applications. It was, however, a bit too early to implement a talking pronunciation function. Since then there have been a flurry of digital publications, as is well known.

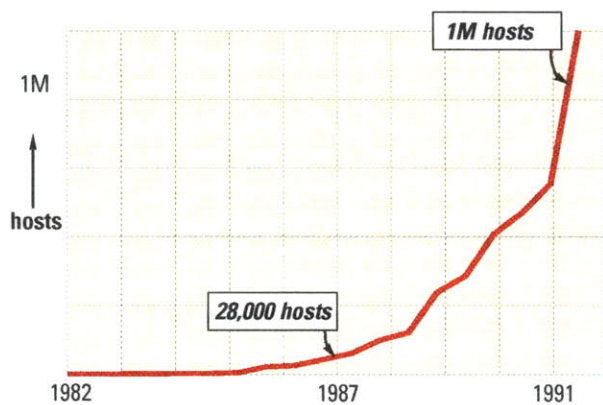


“On the Internet,  
nobody knows you’re a dog.”

— *The New Yorker*, July 5 1993.

#### *The internet*

The same trends that have brought such interfaces to fruition have also caused global networks to blossom. Again, even in the few years spanned by my doctoral research, the growth has been phenomenal. In 1987 there were about 28,000 *internet* host machines; by 1991 that number had increased exponentially to over a million (Lottor, 1992), involving somewhere between 10M and 15M users on a daily basis:



Although there are indications that growth (in North America, at least) is slowing to a linear pace (Quarterman, 1993), the factors that led to this growth recall Licklider’s argument in *Libraries of the Future* (1965) — namely, that a strong effort made to improve libraries would be repaid by ever-increasing accessibility and use of the information.

The internet constitutes a global associative memory, and has become the *de facto* library for some fields like software engineering. It is a somewhat anarchic network of networks connecting a universe of universities, companies, and private users. As the net has filled, it has given rise to a number of schemes for *finding* things. The concept of *resource discovery* (a phrase coined by Michael Schwartz around 1991) refers to processes for finding relevant information in that space. In the past three years over a dozen systems for indexing and browsing the contents of the net have been developed. Danzig (1992) has analyzed several of these; some that may be familiar to the reader include *Gopher* (Alberti *et al* 1992), *Archie* (Emtage *et al* 1992), *Prospero* (Neuman, 1991), the *World Wide Web* (Berners-Lee *et al* 1992), and *WAIS* (Kahle, 1991). Archie, for example, indexes about 3M public files (about 200 gigabytes of material) on 1500 machines. Its searching service, distributed on machines around the world, essentially provides filename lookup. All of these systems can be viewed as part of a perennial process: burgeoning information, whether in clay tablets, urns,

“The **system** of man’s development and use of knowledge is regenerative. If a strong effort is made to improve that system, then the early effort’s results will facilitate subsequent phases of the effort, and so on, progressively, in an exponential crescendo.”

— J.C.R. Licklider  
*Libraries of the Future*  
(1965)



Digitized manuscript from the Library of Congress’ online exhibit of Treasures from the Vatican; found on the net using Archie.



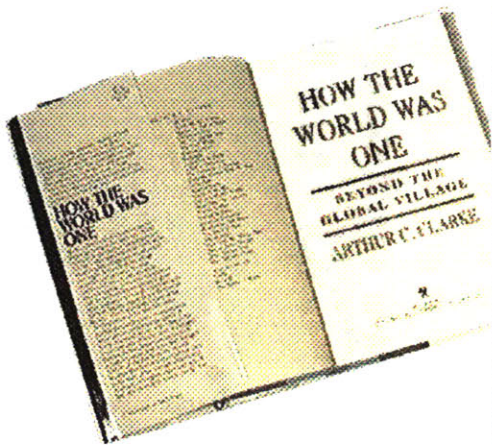
Scrolls from the Dead Sea  
(on-line Library of Congress exhibit)

*"[with] computer consoles installed in every home . . . everybody will have access to the Library of Congress"*  
— John McCarthy (1966)

bookshelves, file cabinets or personal computers, has spurred the development of filing and indexing systems, from Callimachus' first catalog of the scrolls in the Alexandrian library, to Kahle's Connection Machine data-parallel retrieval system. More and more, we will seek ways of structuring and exploring these globally networked resources, and coupling them to intuitively and aesthetically appealing personal interfaces.

### *The Library of Congress as bookstore front-end*

Consider the following Library of Congress interface (Hawley, 1992). The *query* window (shown below) communicates with the Library of Congress card catalog (a resource available on the internet, as are hundreds of other library catalogs). A user can fetch book records by subject, title, author, and so on. Catalog entries are retrieved and formatted. The following illustration shows a book record as the user sees it.:



Library of Congress Complete Service

**how the world was one**

title

Title search: "Rowing" finds all items beginning with "Rowing."  
"Rowing for the hell of it" finds that book. Punctuation, upper/lower case, and "the", "an", "a" at the start of a title, will be ignored.

---

Clarke, Arthur Charles — **How the world was one: beyond the global village** [1992]

Clarke, Arthur Charles — **How the world was one: towards the tele-family of man** [1992]

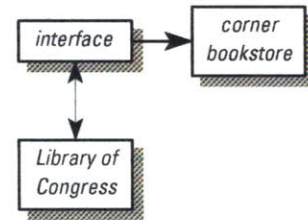
---

<b>Type of Material:</b>	Book
<b>LC Call Number:</b>	TK5102.2 .C53 1992
<b>Author:</b>	<b>Clarke, Arthur Charles, 1917-</b>
<b>Title:</b>	<b>How the world was one : towards the tele-family of man / Arthur C. Clarke.</b>
<b>Publication Info:</b>	New York, N.Y. : Bantam Books, c1992.
<b>Phys. Description:</b>	<b>xvi, 296 p. 24 cm.</b>
<b>Notes:</b>	Includes bibliographical references
<b>Subjects:</b>	Telecommunication systems--History.
<b>LC Card Number:</b>	92003558
<b>ISBN:</b>	0-553-07440-7

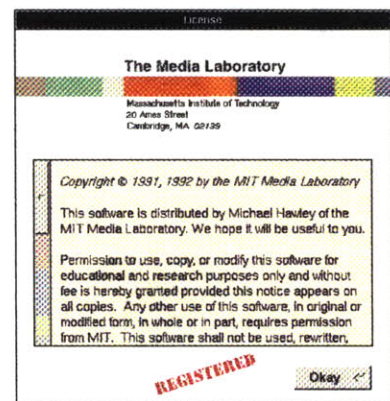
ready!
[Order the book...](#)



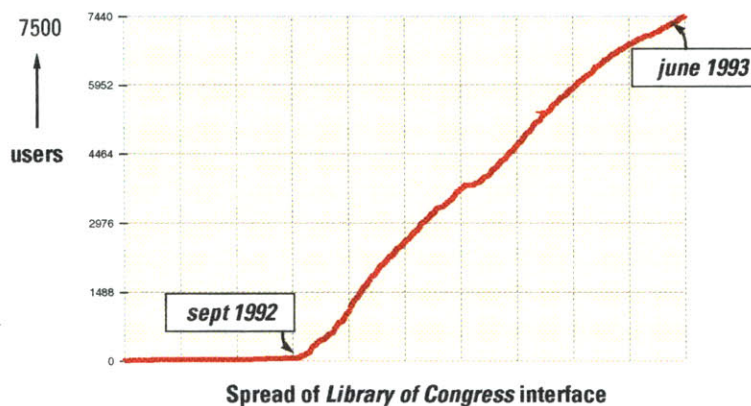
What is going on here is straightforward: the interface essentially contains the same components as *Webster*, minus the dictionary service, plus a connection to the networked catalog (which in this case, resides at Data Research Associates in St. Louis, MO, although the Library of Congress recently placed its own catalog on line). Recent books are tagged with an international standard book number (*ISBN* code). The “Order...” button sends this catalog information along with billing information by e-mail to a purchasing service — the book, if it can be found, may be shipped by overnight mail.



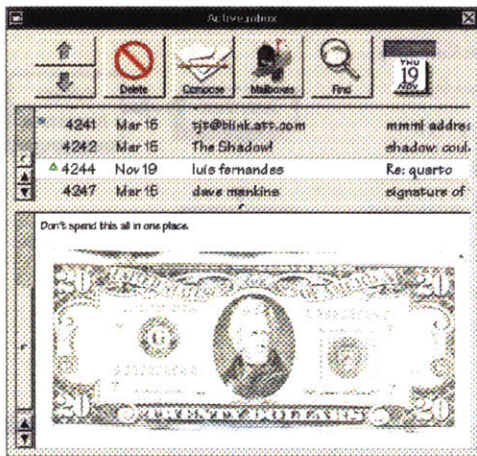
In addition to a design that encompasses the browsing of digital books as well as the purchasing of physical ones, the software itself was distributed to users through the network using a similar mechanism: the author dragged the icon representing the *Library of Congress* application into an archival program, called *Opener*, that shipped the software automatically to several hub archive sites around the world. The *copyright* notice in the *Library of Congress* program is self-registering — that is, receipts are collected from users when they acknowledge the copyright notice. In this way, we can observe how far and how fast the software spreads around the world. In this case, about 7500



Copyright notice/registration panel.



users registered over a 9-month period; the growth rate is roughly linear with slight inflections at the beginning and ends of academic semesters (there seems to be a slight



A "shareware" profit?.

acceleration in the early fall and spring, and a deceleration around Christmas and the beginning of summer, reflecting the preponderance of academic users; however, there are users in hundreds of private companies and institutions in over 40 countries around the world). This degree of connectivity between an author and users was unprecedented before the advent of global networks. It prompts many users to send in comments and corrections to the code, which have greatly improved the software. One grateful user mailed in a digitized \$20 bill!

### Listen to the Net

At this point, the reader might well ask, *what has this got to do with sound?* A discussion of the global information organism and the numinous medium beyond is all well and good, but how does it advance this thesis?

The phenomenon, seen in the rapid expansion of computer networks, is that systems are evolved and adapted in response to the demands of ambient information. But current networks, built as they are from machines of the sensory-deprived sort, are largely based on rudimentary forms of text, a highly abstract form of information, and one can regard well-known text indexing and file-browsing techniques as having been extended into the net in natural ways. Nevertheless, with a bit of cleverness, some surprisingly imaginative products and services can be built. A similar pattern should follow as audio and video become common datatypes in the net. In fact, "talk radio" and forms of broadcast television have already begun to appear.

Unfortunately, machines are currently unable to do many structurally useful things with audio and video streams. We compute from models to audio-visual renderings with difficulty, and the reverse process — abstracting useful models

Station: *Internet Multicasting Service*  
 Channel: **Internet Town Hall**  
 Program: **National Public Radio**  
*meets the Internet*  
 Release: *May 21, 1993, 2-3PM EDT*  
 Content: **Talk of the Nation/  
 Science Friday**

On May 21, we will be joining the Internet to National Public Radio for a special edition of Talk of the Nation/Science Friday. Host Ira Flatow will field questions from

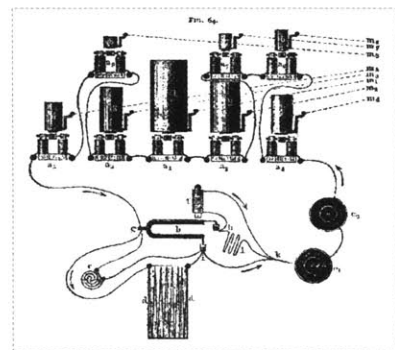
users sitting in front of computers as well as users sitting next to telephones. Questions from the Internet will come from video-conferencing tools on the Multicast Backbone (**MBONE**) using a gateway provided by Ron Fredrick and Steve Deering of Xerox PARC. (To learn more about the multicast backbone, **ftp** to **isi.edu** and get the file **/mbone/faq.txt**. If you do have MBONE connectivity, check SD for a listing for Internet Town Hall.)

*Notice regarding new internet media services*

from sounds or images — is much more poorly understood. The era of global digital systems is rapidly approaching, but without meaningful, integrated approaches to structuring sound and video, the promise of it will be largely unfulfilled.

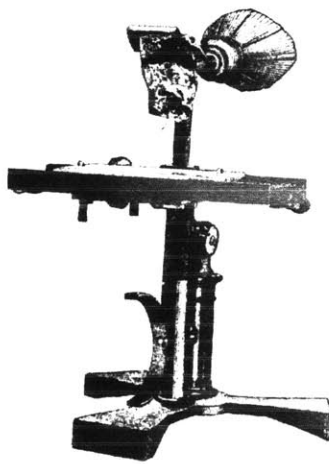
### 1.3 Related work on sound understanding

One of the great scientific texts of all time is Helmholtz's *On the Sensations of Tone* (1863) which ties together physical and physiological acoustics with musical science and aesthetics. That work is significant to this thesis for many reasons, of course, but especially because of Helmholtz's explicit goal: to unify "the horizons of physics, philosophy, and art" which had grown too widely separate. That goal is consonant with the view of the general digital channel as a "unimedium," and with the specific approach of this thesis through sound. Helmholtz was interested in the structure of sounds, how human hearing was able to sense this, and the consequent structure of music. His analysis of the perception of difference tones was done in 1856, and two years later he published his theory of the cause of musical harmony as rooted in the mixing of partials, as well as a study of the timbre of vowels and music instruments. One of the chief difficulties Helmholtz faced was simply that sound is hard to see. The acoustic analogue of prisms was made with great difficulty



*Helmholtz's electromechanical vowel synthesizer. (cf. Helmholtz, p.399)*

from tuning forks and resonators, or sand on vibrating metal plates, but there was no real way to view a sound spectrum; the *phonautograph*, for transcribing sound pressure waves onto smoked glass, was a new development, but offered little insight into the strengths of the “partials” in a given sound. Often it was easier to do analysis by synthesis: his vowel synthesizer is a good example of a tool for that purpose. By adjusting the resonant tubes, Helmholtz could approximate vowel sounds and thus make a rough map of formant frequency.



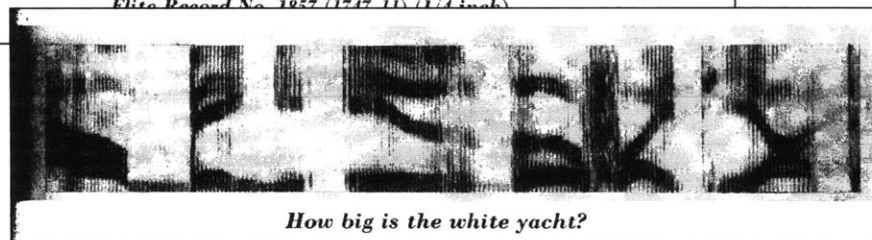
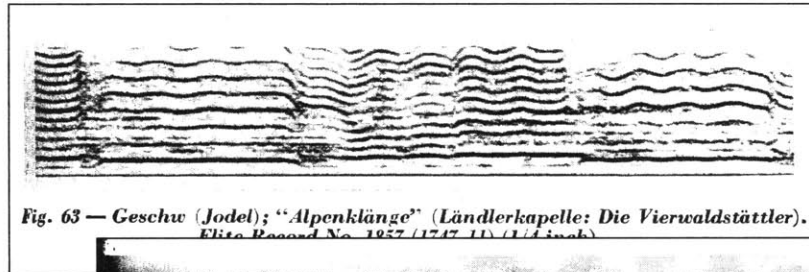
Bell's "ear phonautograph"

Confronted with a similar problem, Alexander Graham Bell built an “ear phonautograph” in 1874, a somewhat grisly device made from a human ear (taken from a cadaver at Harvard) and a piece of straw attached to the tympanum to scribe the vibration on smoked glass. Speech waveforms could then be observed. Bell was looking specifically to find ways of improving on other mechanical phonautographs by using the couplings in a real ear. It was the new understanding he gained from studying the structure of speech sound at the wave level — seeing *pictures* of vowels and the way they were transduced — that directly sparked his development of the telephone (Millman, p.5). For decades after that, as phonographic tools became developed, many similar studies of waveform traces were done; numerous examples can be found in Scripture (*Elements of Experimental Phonetics*, 1904).

### Visible Speech

In the 1930's, Homer Dudley and others developed filterbank approaches to the analysis of sound, leading to vocoder analyzers and the “voder” speech synthesizer. These tools provided a much better analytic view of the frequency content of sound. By 1944, these techniques finally culminated in the real-time *sound spectrograph*, which Ralph Potter and his group at Bell Labs used in an interesting early study looking (literally) at structure in the sound (1947). Their book, *Visible Speech*, was an atlas of sound spectrograms, containing hundreds of pages depicting spectra, ranging from a phonetic pronunciation guide,

to pictures of the sounds of Caruso and Bing Crosby, bird songs, speech in unusual languages (like Icelandic and Ojibway) or speech of the speech- or hearing-impaired. Even the ululations of a tobacco auctioneer were included.



Two of Potter’s spectrograms: yodelling (above), and speech (below).

Potter picked up a thread from Melville Bell (Alexander Graham Bell’s father) whose own book, *Visible Speech: The Science of Universal Alphabets*, was published in 1867. Bell had devised a symbolic phonemic system to aid the deaf, a mission that was shared somewhat by Potter, who pointed out that seeing the spectra, and observing the patterns in them, was deeply informative and educational: the spectrograph, he said, was like “the pattern of a rug, spread out so that it is clearly visible,” unlike the wave which resembled the unravelled thread. In tests with experimental classes, Potter *et al* found that “reading” the speech spectra in realtime was difficult, but that humans could acquire vocabularies with about the same facility as in learning a foreign language (Millman, p.106). The science of speech recognition has essentially taken the same tack ever since.

### Perceptual research — auditory scene analysis

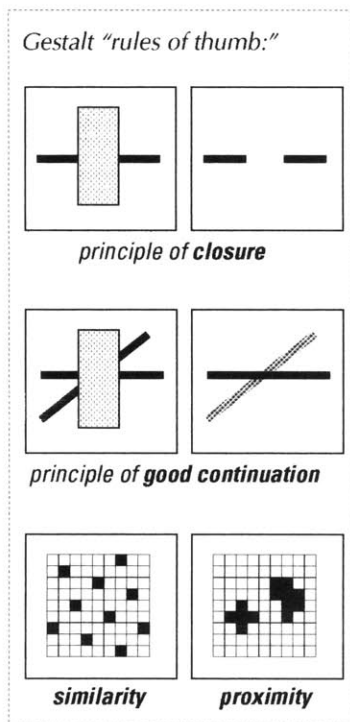
Approaches to machine transmission of sound have long been informed by some knowledge of human audio perception, although this was not codified until recently. Beginning his

lengthy book *Auditory Scene Analysis* (1990), Albert Bregman wrote (page xi):

In the late 1960's, when I began to do research on the perceptual organization of sound, I naively thought that surely the questions that I was puzzling about had been studied to death.

Indeed, Bregman was one of the first to embrace the problem of audio perception and to attempt to establish a comprehensive framework for study. The phrase "auditory scene analysis" and much of the flavor of his approach were borrowed from models of machine vision, particularly the exemplary studies of David Marr (1982). By attempting to understand how sound may be "parsed" in a computable way, this represents something of a departure from the work of more physical or physiological predecessors like Helmholtz or von Békésy.

Bregman's discussion cannot be summed up in a paragraph or two, but the idea is to work toward understanding how humans hear acoustic events — performing primitive auditory scene analysis, segregating and integrating those cues, sensing "timbre," recognizing "schemas of 'known' sounds," and so on.



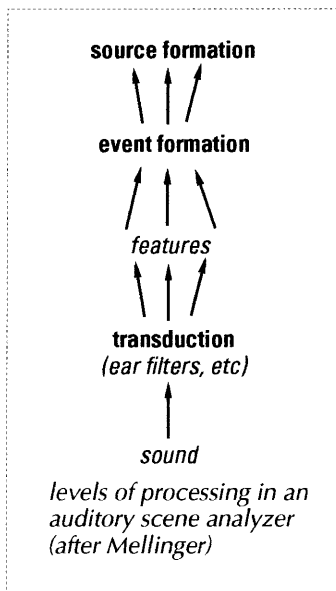
As an experimental psychologist, much of Bregman's work has involved generating listening tests and aural puzzles (akin to optical illusions), then observing how human listeners resolve them so as to postulate the underlying auditory perceptual mechanisms. (Helmholtz approached the study of difference tones in much the same way). He discusses these mechanisms mainly by analogy to the visual features that are observed in images of magnitude spectra. Certain principles of perceptual organization well-known from vision also pertain to audition: for example, *closure* refers to the tendency to complete missing or obscured parts of a figure; *good continuation* refers to the way that ambiguous crossings are resolved by following the most predictable paths; *similarity* refers to the way that elements with the same attributes (e.g., color, size, shape, amplitude, etc) are grouped together, and *proximity* refers to the way nearby elements (in time, space, etc) are chunked into groups. A great deal of Bregman's discussion is devoted to applying and

extending these principles toward the organization of the auditory scene. Bregman notes how computer scientists have also applied insights gleaned from human perception when building systems to untangle complicated soundstreams. For example, although musical deconvolution processes need not model hearing directly, they nevertheless benefit from using similar principles to collect harmonics into notes. In this thesis, although these rules are not explicitly developed, many of the ways in which spectra are taken apart depend on finding lines to locate clicks or harmonics, grouping proximal elements together, and so on.

### **Implementations of the general auditory scene analysis model**

Overall, work on computer implementations of auditory scene analyzers so far has focused on particular problems in music or speech processing. Moorer's thesis, *On the Segmentation and Analysis of Continuous Musical Sound by Computer* (1975) was a pioneering effort to perform automatic polyphonic music transcription (from sound to symbolic score). Moorer's system, though not "practical" at the time, did accomplish the task of music transcription on a limited basis (only periodic tones, no vibrato or trills, no instances of a fundamental overlapped by other harmonics, no more than two voices, etc.). Much of the work was classically rooted, but penetrated into the murkier realm of perceptual processing, and of heuristics needed for high-level work. The thesis ended optimistically, suggesting that the process could be advanced to a relatively high level by straightforward extensions. It is notable that, nearly 20 years later, there are still no systems for automatic music transcription in practical use known to the author. This is not for lack of research attention or want of industrial interest.

Mellinger's thesis, *Event Formation and Separation in Musical Sound* (1992), is an example of a low-level perceptually-modelled musical event parser. In particular, he builds on Lyons' cochlear models (1986) as the first-stage processor, and follows a



tack similar to that of Marr (1982) in constructing a multi-level auditory model. In the model, sound is first transduced by an “ear” filter to yield a *correlogram* (similar to a “constant-Q” spectrum, but incorporating mechanisms for gain control, dynamic encoding, and a process representing basilar membrane action). This is followed by a *feature extractor* that finds cues such as harmonic onsets and offsets, or variations in frequency and amplitude (like vibrato). The next stage collects features into *events* in a more or less gestalt sense. Finally, events are assigned to specific *sources*, using a fairly complex process that has knowledge of heuristics, patterns, grammars, or possibly even culture (Mellinger p.12). It is a bottom-to-top approach that maps a stream of sound to a collection of logical sources.

A similar method was used by Guy Brown (*Computational Auditory Scene Analysis: a Representational Approach*, 1992). Like Mellinger, Brown also concentrates on a source separation problem (the “cocktail party” problem of separating a voice from surrounding noise). Brown makes the compelling argument that most work in audio content analysis, like speech recognition, has attempted to leap from low-level spectral details to high level symbolic structures with little intervening representation — and without much success. For this reason, he attempts to build a fuller representational layer, essentially paralleling Marr (and Mellinger; the differences in their approaches, as well as critical reviews of the literature on neurophysiological and perceptual modelling of hearing can be found in both theses). Brown also notes that his implementation (and other current implementations) is based on unlearned, “primitive” processing. That is, they make no use whatsoever of learned patterns that might aid the processing, or help to complete higher level gaps. This role is filled somewhat by Markov models in speech processing, but in general, auditory models use little pattern memory. Thus, existing implementations concentrate on low-level segmentation of sources, not recognition of events.

There are other shortcomings besides this in applying formal perceptual models to practical problems in auditory scene



analysis. One is overkill. It is often neither necessary nor desirable to force a particular sound recognition task through a human-like processing model. It may be simpler and more effective from an engineering standpoint to model the source. For instance, one does not need a human to identify DTMF telephone “touch tones” — in fact, it takes an unusually skilled human to do it, whereas a simple filter works perfectly well. But, bear in mind that even implementing a touch-tone recognizer in software is awkward at best in current systems! Another problem is extensibility. Perceptual models are not generally coupled to synthetic processes. By its nature, a perceptual filter seeks to throw away irrelevant or redundant information so as to isolate the important parts. The process is not always reversible. Although a perceptual system might inform a synthetic process in a uniquely meaningful way, and although approaches to synthesis are being explored (Ellis, 1993), the model is essentially analytic. It also tends to be monolithic, with customized connections between transduction filters, feature extractors, and so on, making it difficult to apply to other problems.

### **Other precedents from speech science**

The speech recognition field, also a highly structured domain like music, has a pressing need to create complex, high-level sound analyzers that are integrated into general systems. The literature on speech processing is full of approaches to extracting content from spoken utterances — talker identification, separation and segmentation of mixed conversations, speech identification (speech to text), studies of intonation (affect), and so on. There is no need to review the speech literature here (the reader might refer to the recent perspectives offered by Flanagan and Del Riesgo, 1990, or Sondhi and Furui, 1992, or any of a number of texts, like Parsons, 1986).

One of the more demonstrably useful recurring techniques in recent research speech processing systems is the use of

“blackboard” architectures for coordinating the interpretation of multiple sensors. The term originated with *Hearsay* (Lesser, *Organization of the Hearsay-II Speech-Understanding System*, 1975) and has become a central part of many approaches to sensor interpretation. The model is that of a group of experts watching solutions being developed to a problem, as if on a blackboard: whenever an expert sees room for a contribution or a correction, it goes to the blackboard and makes the changes. In this way, many interpreters can inform one another. One of the notable things about this approach is that it is highly unstructured: there is no predetermined order in which expertise is applied, and solutions tend to adapt well to a variety of situations. Thus it provides some of the representational richness that Brown, Mellinger, and others were after, but without the “hardwiring.”

### **Other developments in audio programming**

#### *Low-level structural manipulations of sound*

In his 1988 doctoral thesis, Xavier Serra described an analysis/synthesis system that worked by separating a sound into predictable (pitched) parts, and residual (stochastic) noise. The gist of his idea was to model the pitched component sinusoidally (much like the method used by Macaulay and Quatieri in 1986), then subtract the modelled sound from the original to derive a residual noise signal. A piano note could thus be separated into the resonant string sound, and the “crack” of the hammer. In this way, the two components could be separately operated upon and then recombined to reconstitute the original exactly, or a modified version of it. This has to be done with care — for instance, phase must be properly dealt with — but it is an intuitive and effective decomposition.

Serra’s thesis came at a time when thousands of timbres and sampled sounds were beginning to crop up in synthesizers, but there was no graceful way to mix or interpolate between them. Such mixing is known as “cross-synthesis,” and forms of it

have been used for many years. Moorer discussed the application of linear prediction for this purpose in 1978, e.g., to make a “talking trumpet.” Lansky and Steiglitz (1981) used a similar “warped” form of linear prediction to change the apparent size of the source resonator (turning a violin into a ‘cello, or a man into a boy, for example). But linear prediction can be difficult to do well, particularly when the sounds contain noise, and it has been too costly to compute for common use in synthesizers. For relatively simple musical sounds, Serra’s method provides a natural approach, and at the time, it was just becoming feasible to consider incorporating that degree of computation in commercial synthetic instruments. A similar method was reported by Dannenberg, Rubine, and M. H. Serra (1990), which was based on the interpolation of frequency spectra to reproduce short-time variations in a signal. In that case, time-varying magnitude spectra were used to modulate the output of a bank of waveform oscillators. Like Serra’s technique, this made it possible to mix sound in a more structurally-oriented way, using a few intuitive parameters. In any case, all of these methods are approaches to the problem of operating upon low-level structural aspects of sound.

#### *High-level processing*

The advent of MIDI (Loy 1985) spurred a great deal of work on high-level music processing. Prior to that, though, Buxton et al. (*The Use of Hierarchy and Instance in a Data Structure for Computer Music; Design Issues in the Foundation of a Computer-Based Tool for Music Composition*, 1978) showed how a fertile set of musical tools could be built given a well-conceived underlying data structure. Later, MIDI-based tools provided a wealth of music making possibilities built on a standard for encoding the contents of musical data streams. Using MIDI, Langston provided a more modern rendition of Buxton’s artful arrangement of software tools by grafting the general “little language” methodology well-known from Unix system design

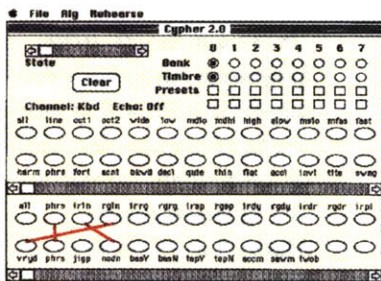
*“In the last decade, **little languages** have emerged to support a multitude of tasks ranging from complex statistical calculations to construction of lexical parsers. Meanwhile, in the last half-decade, a multitude of computer-controlled sound synthesis devices have become available. Unfortunately there has been little overlap of these new development areas and the software to support these new devices has been rudimentary at best.”*

— Peter Langston  
(1990)

into musical tasks (*Little Languages for Music*, 1990). The synergy of such an approach is nicely seen in Langston's work.

A more recent example is the NeXT *Music Kit* (Jaffe *et al*, 1992) which extends musical data structures familiar from the earliest "Music-N" languages of Mathews (1969) into a fully object-oriented system. The Music Kit has a full-featured score language intended to make programmatic use of all the musical objects. NeXT was also the first company to provide some general sound processing software with its *Sound Kit* (closely coupled to the Music Kit), though most of the work is simply in managing recording and playback of multiformat audio data. It was, however, an important first step towards making sound a first-class data type in general computing (around 1988). Of all these examples, and many others besides, Langston's best illustrates the expressive power to be had by crafting well-designed tools and languages. For example, he shows how entertaining incidental music (pseudo-Mozart, bluegrass, etc) may be generated on demand to suit given constraints (e.g., time, key, tempo, etc).

In the analytic area, Rowe's recent thesis (*Machine Listening and Composing*, 1991) showed how many high-level musical feature detectors could be combined to do quasi-intelligent sensing of musical input. These cooperating agents were able to inform a synthetic performer, shaping its behavior to suit the musical content. Rowe's program, called *Cypher*, was a Macintosh application that provided a construction kit for assembling and experimenting with high-level musical sensors. Because the tasks were musical ones, a great deal of processing involved listening for and operating on patterns of *notes*, like motives, harmonies, and rhythms. In this way, the system was able to act on musically salient features in the live performance, and modify its improvisations accordingly. This tied together high-level musical content analysis and synthesis in an interactive system, marvelously demonstrating that one does not need to imbue a machine with the clairvoyant ability to follow a human's musical intentions: rather, because music is so highly patterned, useful

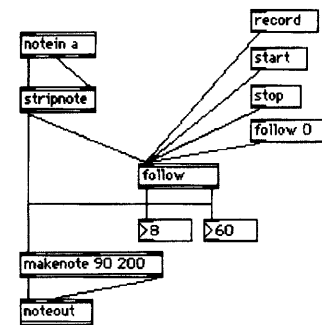


Rowe's interface to *Cypher*, for interconnecting "listener" and "player" objects in an interactive MIDI music system.

and intriguing synthetic performers can be built to serve as gestural amplifiers by sensing and acting on those patterns.

One would also like to build applications in this vein that are connected not just to MIDI sensors, but to sound sensors, and Miller Puckette has taken a step in that direction by extending his MAX program (a graphical MIDI dataflow-style application, 1991) and *Animal* (Lindemann and Puckette, 1991) to include signal objects. In particular, MAX was adapted to run on the IRCAM Signal Processing Workstation (a NeXT computer equipped with an add-on i860 accelerator). This was one of the first attempts at using general-purpose processors for sophisticated realtime audio applications. The machine was fast enough to compute and invert FFTs in real-time along with a network of other signal and control filtering algorithms, all of it embedded in the framework of an engineering workstation — an important step towards achieving computational continuity between signal and content. Rowe surveys these issues more fully in his book, *Interactive Music Systems* (1993).

A large-scale attempt to integrate sound and other multimodal input was made by Flanagan's group at AT&T Bell Laboratories, with its *HuMaNet* experiment (Berkley and Flanagan, 1990). This was a group teleconferencing experiment that used talker and speech recognition, speech synthesis, spatial positioning and spatial sensing of audio sources, along with computer-mediated videoconferencing. It demonstrated how these technologies could combine harmoniously to provide an information and communication interface that was natural and easy to use. It also demonstrated how several sorts of structural information in the sound were needed to achieve that. The "guts" of the control system was a finite-state grammar that allowed rapid definition of application interface commands and easy changes to existing systems: the sensory inputs (chiefly speech commands, after the correct talker had been physically located and identified) were parsed through the grammar to trigger actions.



A MAX patch for a recording and score-following function. (cf. Rowe, 1993)

*“System architectures are desirable that permit specific techniques and data representations from signal processing and artificial intelligence to be integrated in a more sophisticated manner than allowed by classical signal understanding systems”*

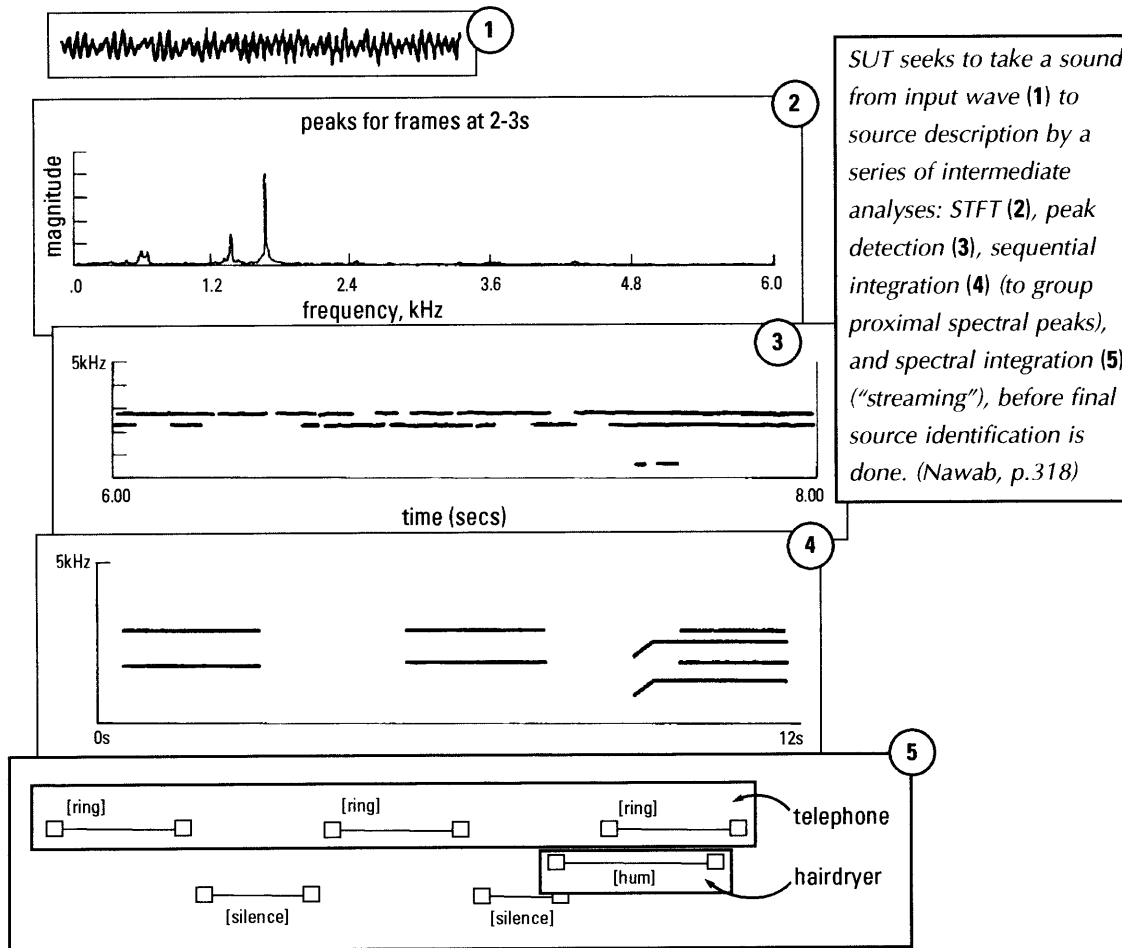
— H. Nawab  
(1992)

### What about the general problem?

Little of this recent systems work has much to say about the problem of general sound understanding. The research of Nawab’s group is the sole exception that proves this rule: he describes a general “sound understanding testbed” (in *Symbolic and Knowledge-Based Signal Processing*, 1992, ch. 9; Nawab & Lesser, *High-Level Adaptive Signal Processing*, 1989). Their system lets one experiment with techniques for understanding sound in a household setting (recognizing telephone rings, electrical appliance sounds, smoke alarms, and the like). Robot hearing ability was the goal, although more specific systems, like a helicopter signal tracking system, and a pitch analyzer for a speech system, were also discussed.

Nawab showed how their general sound understanding system attempts to detect the rings of a telephone and the whir of a hair drier turning on: roughly speaking, a spectrum is obtained by short-time Fourier analysis of the input, the spectrum is reduced by a peak-finder, and peaks that are close in a harmonic and temporal sense are grouped into *contours* (Bregman calls this process sequential integration), then contours are collected into *streams* by a process of spectral integration. Streams are then grouped to form *source* hypotheses. All of these activities were steered by a sophisticated blackboard system. This is the first attempt known to the author to build exploratory tools for a general-purpose high-level sound recognition system that can function in a realistic setting.

One of Nawab’s chief observations was that, because of the enormous variety of input signal types and system goals, classical signal understanding methods were inappropriate. The “classical approach” is one that seeks to formulate the problem mathematically, and then designs a particular system to achieve those objectives if possible. It was Nawab’s contention that the wealth of inputs demanded a similar wealth of processing algorithms, and that the focus is on “providing the capability of



utilizing the most appropriate algorithm on a when-needed basis as well as providing mechanisms for deciding which algorithm is needed in particular situations." Whether or not much of this has been put into practice, as by integration in current operating systems frameworks, is unclear.

Recently, Nawab, Beyerbach, and Dorken (1993) have begun to look more deeply into the problem of tracking arbitrary sounds using a principal component analysis of time-frequency distributions. The idea is to reduce time-frequency data, like a spectrum, to three one-dimensional "principal decomposition functions" — a spectral function, a frequency-shift function, and an amplitude function. This is reminiscent of linear predictive

analysis, which similarly yields a representation of pitch, amplitude, and a time-varying filter function, though Nawab's PDF's do not seek to provide an invertible analysis. Because pitch corresponds to the driving mechanism, and filtering relates to the resonating components, this sort of analysis corresponds fairly well with many physical sound sources.

### Summary

*"The nature of the computations that underlie perception depends more upon the computational problems that have to be solved than upon the particular hardware in which their solutions are implemented...an algorithm is likely to be understood more readily by understanding the nature of the problem being solved than by examining the mechanism (and the hardware) in which it is embodied."*

— David Marr  
(1982)

Taken as a whole, this work is clearly heading in a common direction, albeit under many different rubrics. Awareness to the topic of auditory scene analysis is increasing, and there is a growing opportunity to embed knowledge of this in machines. In particular, computers are acquiring enough computational power to make audio processing capability common. Massalin gave a convincing demonstration of that with his *Synthesis* operating system (1992). It essentially proved that previous attempts to build audio systems will be combinable in a unified software environment.

Abstracting structure from sound is an ancient problem, and this brief survey shows three styles of approach. In contrast to the formal perceptual modelling approach, and the classical signal processing approach, and the neo-classic approach used by Nawab, this thesis seeks representational richness by concentrating on high-level *events*, and on ways to develop a flexible repertoire of programming tools for working with them. That does not mean that traditional research themes are not present: the use of event lists in this thesis serves a similar purpose as blackboard architectures (except that it serves to make the content of sound available to the whole system); gestalt-like grouping mechanisms, edge enhancement and attack-detection filters, and other analytic primitives familiar from the traditional perceptual and analytic literature are used frequently. Lessons learned from studies of integration (like Langston's music languages, Rowe's improviser, or the *HuMaNet* conferencing system) have been pleasantly synergistic, which is encouraging.



## 1.4 Remarks

---

### End of the silent computing era

The topic of general sound understanding was presented as a fertile area for future work in rich-media information systems. Other sensory modalities are important, too — a machine with taste buds, or some haptic sensitivity would have many wonderful applications — but the lack of audio capability is a grave deficiency in machines, much as it is in humans.

Outside of speech processing and computer music, work on sound understanding systems to date has been sparse and disjoint. Areas like “machine hearing” and “audio interfaces” remain virtually unknown compared with their graphical analogues. This is not simply because of the more tangible nature of images compared to sounds, or the prevalence of applications that require pictures, or even because simple still pictures and sketches can serve as a natural and useful interface on practical machines, whereas “still” sounds *per se* don’t exist and “sketchy” sounds are not tolerable for long. Rather, it is because operating effectively on sound requires a grasp of the underlying structure. Cave paintings evolved before written languages for essentially this reason — much the same reason that recent graphical interfaces necessarily preceded audio interfaces.

The drastically changing climate of general digital information systems was also discussed. By its nature, this change is global and highly synthetic, reflecting a mass-merger of computing and all manner of digital information. Simon’s parable of the ant on a beach reminds us that, although organism and environment mix to shape intelligent behavior, the rapid buildup of digital information infrastructures relatively suddenly creates a vast “beach” where there was none before. That imposes itself on the organism. Faced with information overload, we evolve discerning sensory abilities. The stimulating case of the free-

*“Whether deafness is ‘preferable’ to blindness, if acquired in later life, is arguable; but to be born deaf is infinitely more serious than to be born blind. For the prelingually deaf, unable to hear their parents, risk being severely retarded, if not permanently defective, in their grasp of language....and to be defective in language, for a human being, is one of the most serious calamities, for it is only through language that we enter fully into our human estate and culture, communicate freely with our fellows, acquire and share information. If we cannot do this, we will be bizarrely cut off — whatever our desires, or endeavors, or native capacities.”*

— Oliver Sacks  
**Seeing Voices** (1989)

flowing wealth of services already populating the internet versus the torpid world of broadcast television, makes a good argument for why we must develop ways of abstracting meaningful structure from sound and video.

### **Broad versus deep**

Unlike most doctoral studies, which contain a lot of information about particular problems, this one attempts to cut some practical inroads in a large general problem. A difficulty with this sort of work is that, although certain low-level signal analysis techniques are often of interest in and of themselves (in fact, to the author, they are frequently the most interesting parts), the higher-level mixtures that tie recognition systems together seem to be described by a hodgepodge of heuristics, or a babel of little languages. As with other “AI” problems in sensor interpretation, problems in understanding sound are often qualitative. They can be quirky and subjective in nature, revealing the complexities of the source as well as the context in which the sound occurs. Consequently they tend not to be defensible in a crisp, mathematical sense, as has been noted by Moorer, Rowe, Nawab, and a great many others.

This is not a failure of science, simply a reflection of the world. The job of perception is to build a useful model of reality from sensory input — that is the beginning of deeper understanding. To cope with this, as Miller said, it is generally better to know a little about a lot of things. Roger Shepard (1981) used the phrase “psychophysical complementarity” to describe the tangled way that mental mechanisms evolve to map the tangled structure of their surroundings. Marvin Minsky’s *Society of Mind* (1986) told a similar story on a much larger scale. So did Georg von Békésy (1960) when he described his approach to the convoluted problem of hearing as a “mosaic” that studied many small pieces of the problem. And in a way, so did Albert Bregman (1991) when, after nearly 700 pages of discourse on the subject of auditory scene analysis, he concluded:

*“In order to survive in a constantly fluctuating world, it is better to have a little information about a lot of things than to have a lot of information about a small segment of the environment.”*  
— G. A. Miller  
(1956)

There is no chance that the reader will have arrived at this point in the discussion with the sense that anything is settled. Even if one accepts the general framework of auditory scene analysis there still remains the problem of filling it out . . .

Like any other sense, the sense of sound is plural, and the implementation of it is an endless process of understanding and integration. This thesis is a step in that direction.



# 2: *I can name that tune in . . .* Listening to Music

---

## 2.0 Overview

---

*"Strange, dear ... but true, dear ..."*

Spinning the radio dial, you hear little snatches of sound:

*"...from the BBC world news..."*

In a few seconds, or even fractions of a second, you can tell whether the sound is a news anchor person, a talk show, or music. What is really daunting is that, in the space of those few seconds, you effortlessly recognize enough about the vocal personalities and musical styles to tell whether or not you want to listen! Besides which, you may be driving a car one-handed, listening to the radio through static and traffic noise, and sipping a milkshake, all at the same time.

You have just solved the generalized "name that tune" problem. Not only can you hear whether those snippets of sound contain music, you can guess the instruments, hear the notes and lyrics, and recognize melodies in the presence of a great deal of competing noise. This chapter builds some machine listeners that attempt to do similar things. Music, by broad definition, is the whole science or art of structuring sound, so to motivate and focus the chapter, the first two sections discuss music-making from a technology perspective, and the subsequent three sections study specific music-related sensors.

---

In particular, section **2.1** notes trends in the interplay of tools and musical forms. Its purpose is to sketch some of the ways they have mingled, and offer insights to those who might not be well-acquainted with music. The material is nontechnical. Section **2.2** describes experiments and experiences with a MIDI music system I built, including such problems as arranging a full romantic piano concerto for synthetic orchestra and computerized Bösendorfer concert grand piano “soloist,” as well as the quaint but interesting reciprocal problem of optically reading paper piano rolls to derive musical performance data.

To begin the discussion of music signal analysis, section **2.3** presents a *music detector*, a filter that attempts to find the musical segments in a sound stream, converting a sound stream to a list of “music” events. At first this seems an impossibly broad notion, but it is shown how quite a good music finder can be written in about a dozen lines of code. Its performance is examined. Section **2.4** investigates an approach to the knotty problem of *polyphonic pitch extraction*, specifically for the case of filtering acoustic recordings of piano music to derive the performance gesture. This is another example of a sound-to-eventlist conversion process, where the input sound is piano music, and the output event list is the gist of the performance. Can we really build a filter to convert an audio piano recording to a performance list? What are its limitations and implications? These questions are considered. Then, having found some music, and having extracted a few notes, section **2.5** discusses the “*name that tune*” problem, in which we want to recognize an input melody. That problem hinges not so much on the lexical aspect of matching a melody-string in a dictionary, as it does on finding ways to locate snatches of music in (often noisy) sound streams. Section **2.6** summarizes the results.

## 2.1 Revolutionary études: music and technology

---

### The rise of the *instrument-savant*

Have you heard the organ at *Notre Dame* recently?

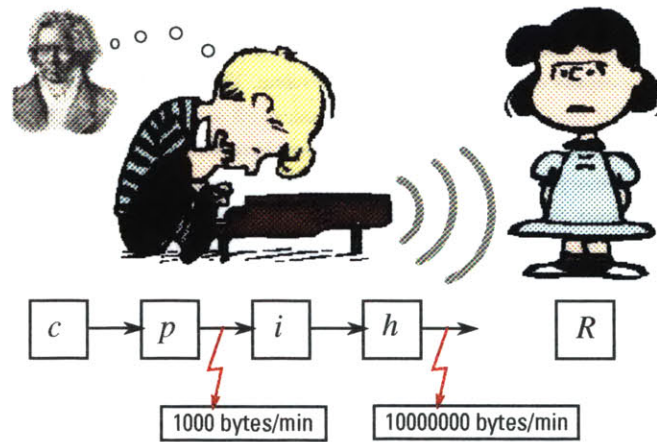
*PARIS (December 7, 1992)* – The organ of Notre Dame cathedral can now boast digitized sound transmission, video monitors and three IBM PS/2 PCs on a 16 megabit/sec Token Ring local-area network. Following its three-year, 11 million franc (US\$2m) investment from the French state, French software firm *Synaptel SA* unveiled the restored instrument last week. A PC supervises the transmission of data between the organ's five keyboards and pedals and the 7,800-pipe organ case as well as a MIDI (Musical Interface Digital Instrument) gateway. The system allows the organist to program the sensitivity of the keys and the digitization of the musical signal avoids problems of dust in the organ's old pneumatic tubes and valves. The sequencer of the network's MIDI interface lets the organist record, recall and edit anything he plays. Instead of a key to modulate the tension parameters of the keyboard console, a memory card memorizes and establishes the preferred configuration of each organist. Synaptel also installed synthetic voice capability on headphones to warn blind organists of important mixing information as they play. The organ of Notre Dame cathedral was first reconstructed in 1730-1735 when a mechanical traction system was put in place. Electricity was installed in the keyboard/pedal console in 1959. (© IDG News Service)

The inevitable infiltration of digital technology into the organ console at Notre Dame represents a far more drastic change than simply eliminating the troubles caused by dusty pneumatics. The change, of course, is that instruments are no longer relatively simple mechanical systems. Instruments now have memory and the ability to receive digital information. They may *render* music in a deeply informed way, reflecting the stored impressions of many instruments, halls, performers, compositions, and a number of environmental or contrived variables. At first, digital technology let us clumsily analyze and synthesize sound; later, it became possible to build digital control into instrument interfaces, so that musician's gestures were captured and could be operated upon; and now, we can see that it will eventually be possible to compute seamlessly between the wave and the underlying musical content. That is a distinct departure from the traditional idea of an instrument.

*“As you know, my own life has been chiefly devoted to the development of instruments of communication. But however important these may be, they are at best only instrumentalities. Their function is only to transmit. In the final analysis, they will be judged by what they transmit.”*

— David Sarnoff  
in a letter (1954) to  
Arturo Toscanini

What is interesting about the trend is that it injects intelligence and autonomy into instruments. Their function is no longer “only to transmit” or transduce: they become sources and interpreters of deeper musical information in their own right. Mathews articulated this early on (1969); Vercoe’s work with “synthetic accompanists” (1989) and Machover’s work with “hyperinstruments” (1990) are two of many recent examples of the ways that artistic and research opportunities flourish when instruments become active partners. The consequences of this are both immediate and far-reaching. To understand how, it helps to reconsider the canonical flow of musical information:



Information from a *composer* is transmitted to a *performer*, who adds interpretation and inputs it to an *instrument*. Typically, that is a transducer that converts gesture to sound, so that the wave imparts compositional and performance information as well as information about the instrument. As it radiates through a *hall* or through a recording system, the sound acquires features of the acoustics or the recording setup. The listener catches this signal and takes away some sense of the room, the instrument, and possibly the recording methods, but unless the listener is a piano tuner, an acoustician or a recording engineer, most of the attention is concentrated on the music and how it is played.

There are many variations on this theme, and there are vital feedback paths, too (e.g., a performer will play differently depending on the instrument, the room, or on interaction with



other performers) but essentially, the source network is a process of information expansion, and a reciprocal process of reduction occurs in the listener. Measurements indicate that the data flow between a performer and an instrument is on the order of 1,000 bytes per minute (e.g., piano-playing is typically about 3Kbytes per minute, depending on the music). By contrast, a digital recording of the sound at acceptable commercial rates is about 10,000,000 bytes per minute, much of it redundant over time.

Many people (even information scientists, who should know better) are startled by the fact that musical gesture, like the keystrokes in piano-playing, flows at such a paltry rate. There are about four orders of magnitude in compression to be had by distilling the audio signal to its gestural content. At gestural rates, all of Joplin's piano rags occupy about a megabyte, and one present-day compact disc would hold years of music(!). Consider how this has changed the music production infrastructure has: today, a composer can earn \$25,000 for a day's work by synthesizing 30 seconds of music for a soft drink commercial — the music is composed, arranged, performed, synthesized, edited, mixed, synchronized with video, and printed to a master digital tape, all by *one* musician-engineer working in a digital studio. Most of the tools that make this possible are less than ten years old — many, less than five.

This is a short-term symptom of a long-term trend. The new concept of an "instrument savant" piano that recalls a Chopin étude, or that can render any piece of music in its repertoire, and can shape the timing and expression to suit some other purpose, improvising as need be, is indeed revolutionary. As more information accumulates in instruments, it is fair to say that instruments themselves must begin to understand something more musical about the music that flows through them.

### **Historical notes**

The prospects of increasingly intelligent instruments are actually part of a long history of using technology to amplify music-making ability. A digression helps put this in perspective.

Consider the marriage of music and musical tools through the early mechanical, industrial, electrical, and now digital eras. Until the advent of the phonograph and telephone around 1875, there was no medium for recording and transmitting sound *per se*. Early musical instruments were purely mechanical. Music was always “live.” The other musical communication medium was printed scores, which were scarce. Although it is often forgotten, the interplay between prevailing technology and the changing shapes of music can readily be seen in each era — the development of musical form is largely written around its tools. Here are several examples.

#### *Cathedrals and the end of chant*

For a monk sitting in a small chapel, it was easy to adhere to Gregory I’s divinely-inspired reforms to the *schola cantorum* (c.590) that there be no instruments, and that all singing be done in monophonic, rhythm-free chant (so-called gregorian chant). That broke down with the rise of great cathedrals (c.1000 AD). In huge rooms, monophony gave way to polyphony, instruments that were loud enough to be heard had to be used to extend the voices, and a strong sense of rhythm was needed to keep many participants together. The fact that Andrea and Giovanni Gabrieli in Venice wrote so much polyphonic brass and multi-choir music is unsurprising given Saint Mark’s cathedral with its many vaults and domes, and remarkable acoustics. These are clear cases of building architecture influencing musical style.



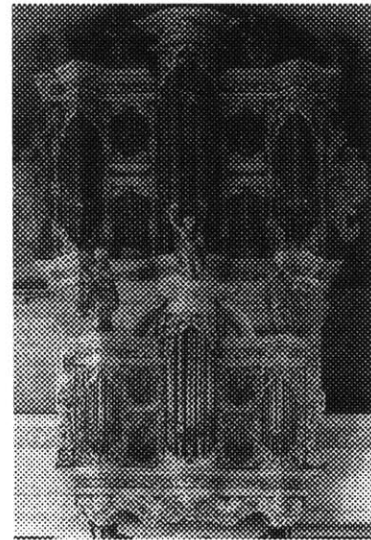
*Air pump for the Halberstadt organ in the 16th century (from Praetorius).*

#### *Baroque high-technology*

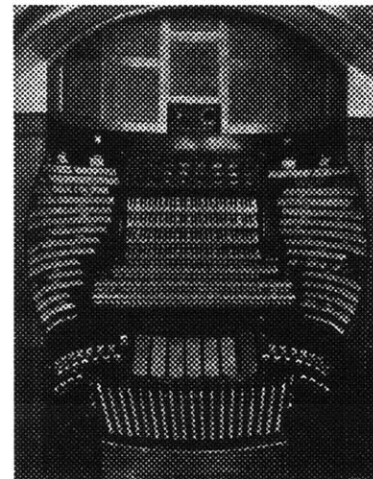
Pipe organs furnish another example. Even though the organ is one of the most ancient instruments (it was described in detail by Hero of Alexandria in the first century), organs remained rather simple until the 1500’s, with a few exceptions in the larger cathedrals (Note Dame in Paris probably had around 2,500 pipes in the 1400’s). Then, in 1517, with the first printed books on organ building and the wider availability of clockwork for making control mechanisms, designs and inventions flourished (Sumner, *The Organ*, 1973): the richness and quality of organ

construction, and consequently, organ playing, improved dramatically. The art of organ building reached a zenith in the work of Arp Schnitger, who built some of the finest organs that J.S. Bach played on, and held a virtual monopoly on the organ-building business in 17th century Germany. Schnitger was able to muster a great deal of technology for glittering metalwork and cabinetry to build perhaps the grandest, most complex and inspiring machines of the time. Nearly all the pipework was metal: wood pipes did not fare well in the dank northern European climate. The corresponding clarity made it favorable to play many-voiced fugues at brisk speeds. The pipes were mostly lead-tin alloys (the show pipes were almost pure tin, imported from England), and the keys were ivory and ebony, so a considerable trade and shipping base was required to supply the materials. Schnitger and his sons built more than 150 organs in northern Europe, many of them magnificent four-manual instruments with upwards of 60 ranks of pipes, requiring substantial air power. It is no wonder that the organ became the hub of so much musical activity. Bach was actively interested in organ mechanics, constantly testing them and working with builders to improve them. The phrase “pulling out all the stops” may have originated with Bach and his habit of opening all the ranks at once when trying out an organ for the first time (cf. David and Mendel, *The Bach Reader*, 1966).

Note that organs have always presented a challenge to interface designers. The word *organ* derives from the Greek *ergon* (tool, instrument) and organ consoles have long been an ergonomic adventure. The problem of linking fingers and feet to tens of thousands of pipes and switches commends itself to contemporary interface designers. One extreme is found in Atlantic City, where the world’s largest organ not surprisingly has the world’s most complex console (seven keyboards and hundreds of stops, switches, and indicators, all within an arm’s or leg’s reach). Computer mediation finally has made it possible shield the user from this overt complexity, and provide many enhanced functions besides, by adding software control as in the new console at Notre Dame.



*The 1695 Schnitger organ at Johanniskirche, Hamburg.*



*The world's largest organ console in Atlantic City, NJ.*

### *Pianos and the industrial onslaught*

Industrialization in the 19th century made it possible to mass-produce and mass-distribute instruments. The “modern” grand piano was thus partly a byproduct of the railroad industry, not only for shipping, but for factory tools to make iron frames and wire strings. The piano was *the* home entertainment medium for a century: manufactured in tremendous quantities, it naturally became a nexus of inventive musical, social and industrial activity of all kinds. If the 17th century organ was a time-shared mainframe machine, the piano in the 19th century can be seen as a much more personal one. The romantic concept of the solo “recital” — a virtuoso with mane of hair and concert dress, playing in profile to an audience of thousands in a large concert hall — was pioneered with overwhelming success by Franz Liszt, and made possible by dint of the overwhelming success of the piano. The majority of musical forms for a hundred years revolved around the piano; for example, all important chamber and symphonic music circulated as piano transcriptions. It became the broadcast channel for music.

Arthur Loesser (*Men, Woman, and Pianos*, 1954) mentions that in 1891, for instance, a real estate developer built a row of apartments on the west side of Harlem in New York — and built an upright piano into every one of the four dozen parlors! From 1890 to 1910 the number of pianos in the U.S. multiplied about 6 times as fast as the population; on the frontier it was not uncommon to find a piano in a log cabin, even if there was no bathtub present. There were many other musical channels opening at the time, to be sure — valved brass instruments, woodwinds with keys, and the flute as we now know it, all matured, and the symphony orchestra grew to its modern size (*cf.*, Sachs 1940). Nevertheless, the dominance of the piano as a unified musical channel is interesting. A full-blooded social and musical history can be written around a single instrument.

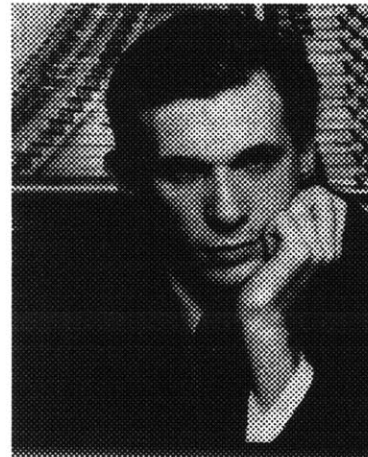
### *Electronics and the wane of live music*

For about the past hundred years, the momentum of electronic communication and the newfound ability to manipulate and transmit sound signals have reshaped music and the way we hear it, in ways both obvious and subtle. The science and craft of analog electronics entered into the design of instruments, from the curious *telharmonium* (1908), a sort of combination organ and telephone network that was a precursor to *muzak*; and the *theremin* (among the first electronic instruments, adapted from radio parts in the 1920's) to the Gibson guitar, the Hammond organ and Moog's synthesizers. The centralized concert experience that led to the large symphony orchestra and mass gatherings is now fragmenting, due to the influence of home and personal audio machines, as well as the ability to amplify and broadcast. The "unified" channel once occupied by the highly interactive piano has been replaced by highly passive receivers, like radios and televisions.

With tape recording, splicing and mixing became the building blocks of a non-live musical art form. Pierre Schaeffer's *musique concrète* in the 1940's was essentially a montage form of composition using taped elements. The Canadian pianist Glenn Gould was also a fiendish splicer of tape, removing clinkers or overdubbing extra lines in pursuit of an ideal contrapuntal extreme (cf. Kazdin, *Glenn Gould at Work: Creative Lying*, 1989; Gould/Page, *The Glenn Gould Reader*, 1984). Gould shunned live concerts, preferring to make music exclusively in the studio, freezing the impression on tape a bit like a sculptor, and thus he became a prime example of an artistic extreme fostered by electronic and recorded media. Much as Bach involved himself in the design of organs and early pianos to improve his musical craft, Gould embraced analog studio technology (e.g. Gould, *Strauss and the Electronic Future*, 1964): "it will not, it seems to me, be very much longer before a more self-assertive streak is detected in the listener's participation, before, to give but one example, 'do it yourself' tape-editing is the prerogative of every

*"I predicted that the public concert as we know it today would no longer exist a century hence — that its function would have been entirely taken over by electronic media. It had not occurred to me that this statement represented a particularly radical pronouncement."*

— Glenn Gould  
(1966)



Glenn Gould: "Let's ban applause."

*“Technology... is not primarily a conveyor belt for the dissemination of information; not primarily an instantaneous relay system; not primarily a memory bank to hold the achievements of man... It should not be treated as a noncommittal voyeur... it must be exploited.”*

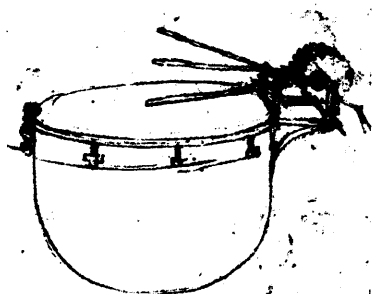
— Glenn Gould (1975)

conscientious consumer of recorded music (the *Hausmusik* of the future, perhaps). And I would be most surprised if the consumer involvement were to terminate at that level. In fact, implicit in electronic culture is an acceptance of the idea of multilevel participation in the creative process.” Gould’s somewhat introspective and paradoxical focus on recordings, which goes against the grain of self-assertiveness in the listener, was, it seems to me, more of an effort to engage the only available electronically-mediated outlet.

The interaction between recorded and live forms of music also makes itself felt in subtler ways: the advent of the “long playing” record caused many musicians to become “completists,” recording lists of pieces that were never really intended to be played as a set (all Chopin’s ballades, or Bach’s partitas, say). This has flavored live concert programming, as musicians now will play “boxed sets” in a single sitting, instead of more balanced menus. And of course new technology this century has brought a merger with movies and television, creating new forms. Chances are, when a composer today writes an orchestral score, it is destined for a film soundtrack: Hollywood studios and routes to the personal stereo have replaced royal courts, cathedrals, *salons*, and pianos in family rooms — another example of large-scale architectural influences on musical forms.

#### *Musical automata*

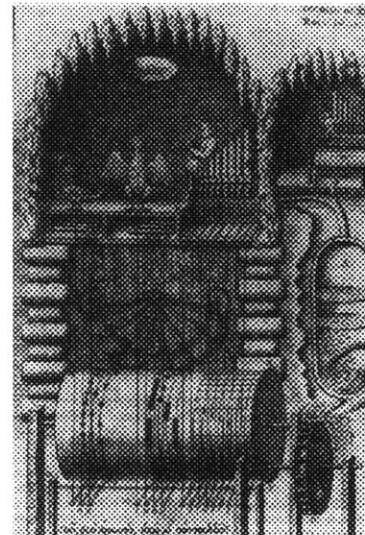
Instruments have been built to play themselves for centuries. Leonardo da Vinci (1452-1519) designed and probably built a number of self-playing music machines; the drum machine shown here was one of many, and along with player mechanisms for bells, organs, and trumpets, was probably conceived for providing automatic fanfares for plays and parties (Winternitz, *Leonardo da Vinci as a Musician*, 1982). Leonardo was certainly not the sole inventor of such things. We know that automatic water-organs were built by Arab and Byzantine engineers in the 9th century (the palace of the Khalif al-Muqtadir



*Leonardo's drum machine.*

had a famous musical tree; automatic musical instruments of many kinds are described by al-Jazari, 1974). These technologies migrated to Italy by about 1400, and during the Renaissance, became quite common. Self-playing organs and musical clocks were well known in the 1500's and 1600's, using pinned barrels to store the notes. The organ shown here was one of the most famous, built for the Tivoli gardens at the Villa d'Este around 1550. It stood about 20 feet high, was powered by a fabulous waterfall, and was described as playing "madrigals and many other things" (the engraving shows it also flapping the wings on puppet birds; see Williams and Owen, *The Organ*, 1988). Contraptions like these were built for popes and for royalty all over Europe, reaching the heights of extravagance at Versailles. By the early 1700's mechanical organs were sophisticated enough to be used to accompany hymns in church. Composers were called upon to write novelty pieces for these automata; Mozart and Beethoven were two of many who composed musical clock and organ music. By the early 1800's barrel-programmed organs were considered to be note-perfect.

These fascinating ancient automatic instruments are sometimes forgotten because there has been a deluge of self-playing instruments in the 19th and early 20th centuries. The decades around the turn of the century are littered with thousands of varieties of player pianos, organs, music boxes, orchestrions, player violins, player banjos, roll-driven movie sound systems, organs, harps, and percussion among other things; they were built into furniture and into handheld instruments, merged with record players, radios, and telephones, powered by steam, electric, pneumatic, or hand crank systems, programmed with rolls, barrels, discs or sheets of punched cardboard, paper, or brass. The player pianos achieved a degree of perfection: instruments built by Ampico, Duo-Art, and Welte-Mignon were capable of reproducing the finest nuances in piano playing. Nearly every prominent pianist until about 1930 cut piano rolls, and regarded it as the preferred medium. Mussolini and Babe Ruth owned Duo-Arts. The current industry built around MIDI



Barrel organ (c. 1550).



Roll-Monica is very popular, being constantly available by playing and old. Simply insert Music Roll a musical number and turn playing handle. Right-hand-glass with hand. Illustration, see Music Rolls, in every music store each full size.



Some handheld player instruments. Precursors to pocket computers?

has recaptured some of the sales volume, but not the charm of these mechanical wonders.

### **Summary — towards the longest musical lever**

To sum up, the couplings between musical forms and musical tools are as intimate and intricate as one's *embouchure* is to a flute. When architecture provided big rooms, polyphony and masses of instruments were developed to fill them, and rhythm was needed to coordinate them. When industrialization provided avenues for massive production and shipping, general-purpose instruments like pianos became a kind of broadcast medium and piano scores became the *lingua franca* for musical information exchange. When electric technology took root, recording and broadcast spawned a new art of studio music-making which transcends space and time at the expense of traditional live concert experiences. All of these tools have had a drastic impact on musical forms, the way music is made, and the way it is heard.

Unlike previous mechano-electric instruments, computer-mediated techniques impinge on the whole gamut of musical sources — composers, performers, instruments, rooms, and so on. In principle, the entire musical process can be controlled and composed. Now that gestural information has come within the realm of cheap synthesis, computer instruments are developing from music boxes into “instrument savants,” with a lot of memory, not much general intelligence, but the beginnings of an ability to listen and shape a response. The next section, on MIDI-related software, illustrates how some of the high-level structure in music can be arranged and sensed.



## 2.2 Experiences with MIDI

---

### Musical building blocks and glue

The present music synthesis industry began to bloom in the early 1980's with two developments: digital music synthesis with interesting sounds and keyboard control became technically feasible (the Yamaha DX7 began that beguine, using FM synthesis which was both computationally inexpensive and acoustically rich); and the MIDI protocol took root, connecting synthesizers to general computers for high-level control. This was welcome news, for it brought computer music as a field out of the mire of low-level synthesis, and into the realm of high-level gestural control. Now more than 40% of all cinema and television music is produced synthetically. Curiously, however, MIDI has not spawned much of a recording industry: you cannot really buy a "record" or "piano roll" of MIDI data as rendered by some famous artist (or algorithm) and play it on your system. This is primarily because MIDI encodes instrument information in a device-specific way (and the devices are quite peculiar). Despite that drawback, many imaginative possibilities have emerged.

*"Extraordinary how potent cheap music is."*

— Noel Coward (1932)

### BSO in a box

Around 1986 I began to build a MIDI system (Hawley, *The Personal Orchestra*, 1990). The idea was to construct a fairly aggressive music system (at the time), and explore.

The system consisted of a Sun-3 workstation (25MHz, 8Mb, 1600x1280 bitmapped display) with a custom VME-bus interface that controlled 4 real-time MIDI processors (Roland MPU-401), for a total of 64 channels of MIDI output, and 4 channels of MIDI input. The Sun ran a local window system (Hawley 1985) and some MIDI system software (Hawley 1986). An IBM PC-AT with a Z80-based controller handled realtime piano control, and connected to the Sun through a serial port. Other musical apparatus included a Yamaha KX-88 keyboard controller (a silent keyboard with weighted keys that generates MIDI data), an MJC8 MIDI junction box, a number of synthesizers (Kurzweil, E-mu,

*"Suppose we had a digital Boston Symphony Orchestra in a box as a personal stereo add-on, and the apparatus to control it. What would people do with such stuff?"*

Roland, and Yamaha), a mixer, amplifier, and some speakers. This kind of configuration was novel at the time, but since then has become conventional.



*Bösendorfer concert grand piano;  
Solenoid actuators below keyboard.*

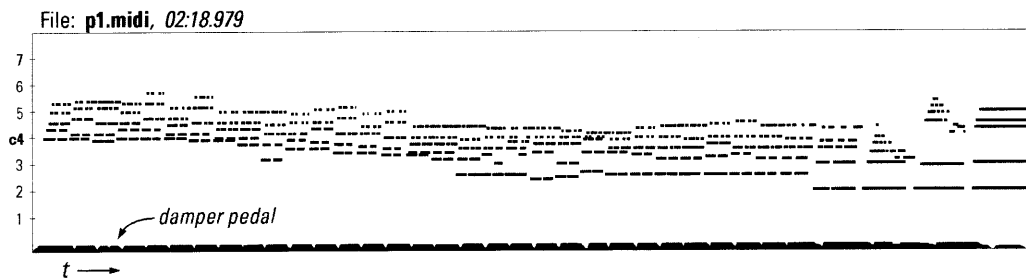
There was also a Bösendorfer grand piano, which warrants a special word. The piano was an unusual addition to the band: it was a 9'6" "imperial" concert grand, the largest piano in commercial production, weighing 1400 pounds, with 97 keys (nine extra bass notes), and three pedals. Designed by Wayne Stahnke for the Kimball corporation, it records hammer velocities optically (at a sample rate of about 800 Hz, to 10 bits of precision), and reproduces the playing with a solenoid stack. It is the ultimate player piano (as well as the most expensive one: at about \$120,000 it cost twice as much as the other equipment combined). Note that by adequately capturing pedal and keystroke data, the piano completely characterizes its musical input. The meaning of the Nyquist limit for sampling of gestural input is not so straightforward to interpret as in the linear signals case, but temporal resolution and dynamic range are such that reproduction is perceptually flawless. Unless the piano is poorly calibrated, one cannot tell by listening whether a performance is live or reproduced.

The idea was that such apparatus would be common in about 10 years — a control processor sufficient to drive 64 synthetic "performers," a synthesis system capable of rendering sound of near-orchestral richness, and content processing as well as musical archives that make interesting interaction possible. While one would not expect to find a Bösendorfer in every living room, a sampled piano and the rest of the apparatus would certainly be condensed into a small box in the foreseeable future.

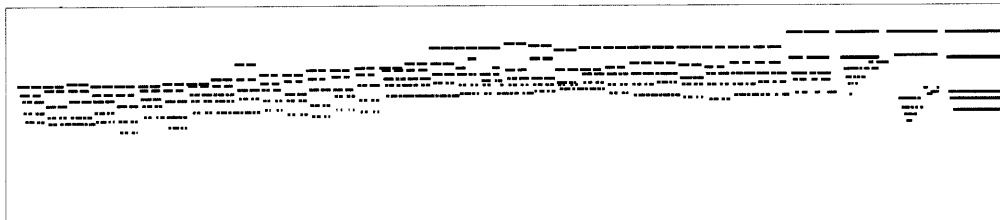
### **First studies**

When musical information begins to flow through a computer, one of the first interesting things to do is count the bits, studying statistics of timing, pitch content, and so forth. Even the simplest-seeming measurements are often illuminating because they

capture meaningful gestures. Simon and Sumner (1968) considered aspects of this; for example, that patterns of note occurrence in music, like letters in text, accumulate in characteristic distributions. Rearrangements of pitches and timings can therefore have an interesting impact on the music. Shown here is the first prelude in C major from the *Well Tempered Clavier*. The data was recorded on the Bösendorfer and filtered through a “piano roll” plotting program:



By inverting the pitches (that is, by re-mapping pitches as if the piano keyboard were flipped, which is done with a software filter) the image of the music is flipped as well, as expected,

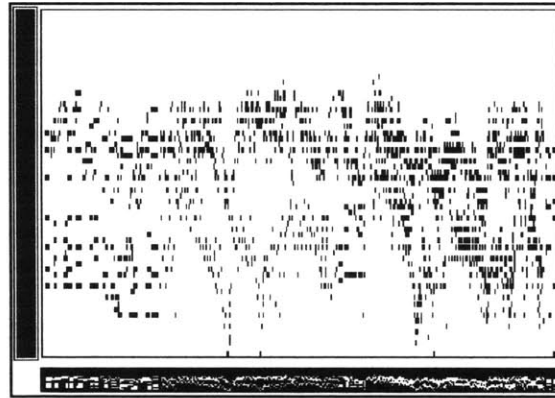


but what is a little more interesting is how the tonal content is affected. “Major” becomes “minor” and vice versa (so a C major harmony becomes an f minor harmony), and traditional tonal chord progressions often become “modal” ones; e.g, a **I-V-I** harmonic progression becomes a **i-iv-i**. When this happens, the music no longer sounds like Bach, although it still has some strange but beautiful progressions. The weird nature of this effect is belied by the picture of a simple visual flip.

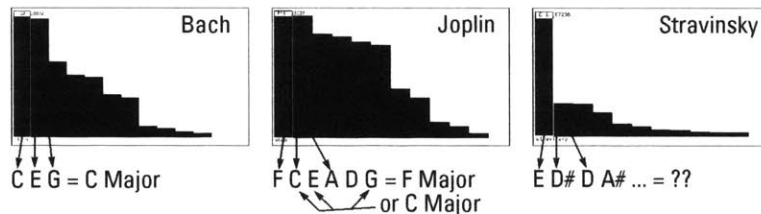


Obviously, the distribution of pitch content is related to the sense of musical key. Tonality *per se* is defined more by the grammar of harmonic progression, but raw pitch distribution is informative

in other ways. The following is a time-compressed picture of the first movement of Bach's c minor partita, and the dark horizontal striations indicate frequent notes:



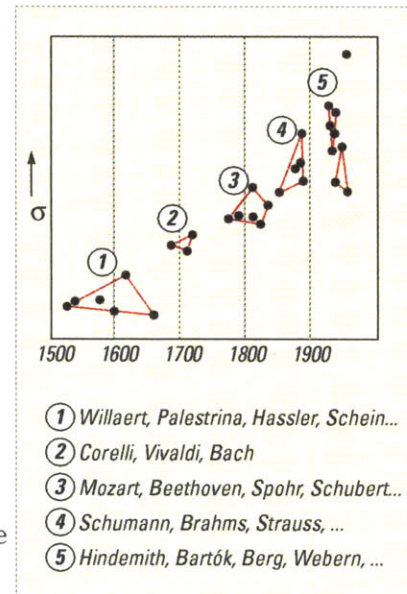
This suggests that a histogram of pitch content (folded into one octave) correlates with the overall key most of the time — provided there *is* one:



In most cases, when the music is of the straightforward tonal kind, the three or four commonest notes indicate the triad of the principal key, as in the Bach prelude. In the case of the Joplin piece (*Solace*, a rag) the bulge in the distribution reflects what turns out to be the bi-modal tonality of the music (it begins in **C** and ends in **F**). The Stravinsky example (from *Les Noces*) is a peculiar distribution, with a big spike on **E**, then a succession of pitches in a non-tonal order. It is an atonal piece.

This measure — one of the simplest measures of musical content imaginable — is interesting not because it measures the musical key. In fact, it doesn't do that in a musical sense, and to do so is not simple (Rowe devoted a considerable part of his thesis work towards getting his *Cypher* program to do just that, with

somewhat mixed success). Nevertheless, the nature of the pitch distribution is frequently all that is needed to distinguish between broad categories of music (for example, tonal vs. atonal, or “is Eliot Carter more like Stravinsky than Bach?”). In fact, in 1962 Wilhelm Fucks studied the frequency distribution of pitch and showed that kurtosis and standard deviation both increased significantly from about 1500 to the present. Furthermore, the choice of key in western music is a significant one — the key of c# minor, for instance, is generally associated with somber or bittersweet music; G major is for “happier” music, by and large. There are undoubtedly many reasons why these associations have developed over the centuries; early tuning systems made some keys more dissonant than others, for instance, but even after equal tempering the tendency for composers to copy successful pieces helped to propagate these conventions. As a result, the statistics of pitch distribution reveal aspects of the music that are deeply ingrained in the culture. So, it appears that pitch probability can be used to roughly pinpoint the century in which the music was written, the approximate key, the degree to which the music is tonal, and it can certainly be used as a kind of “hash” function for finding similar music. These methods will be interesting to develop more fully as libraries of digital music data become available.



*Kurtosis in pitch distribution over four centuries (after Fucks, 1962)*

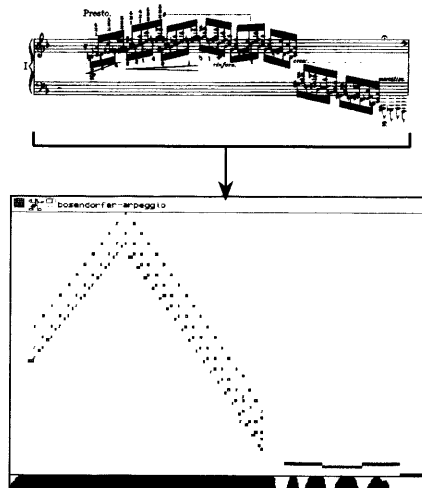
### Arranging studies

MIDI technology is mainly intended for *making* music, and one project was concerned with building arrangements. In particular, I wanted to investigate the feasibility of assembling scores of orchestral complexity and wondered what soft tools would be required. This problem is discussed at length in Hawley (*Totentanz: an Experiment in Symphony Emulation*, 1989); the purpose of what follows is to provide a feeling for the overall approach.

Briefly, the problems encountered in arranging involve assembling the quiltwork of many musical elements, fitting and adjusting the parts, and mapping the parts to instruments in the

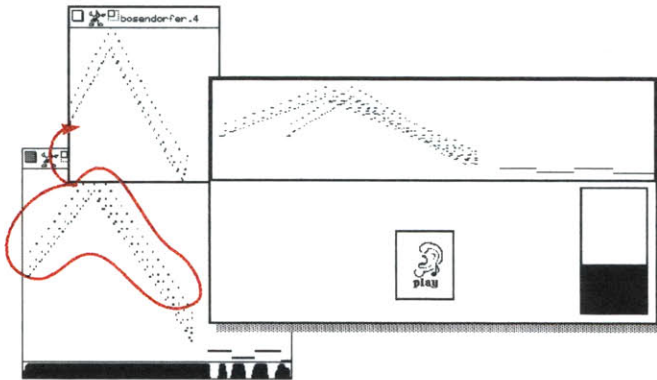
synthetic orchestra. There are a number of subtleties involved; for instance, special editing capability is required to deal with details in piano data like pedalling, and in general, it is not at all straightforward to cast parts between different synthetic instruments because the nuances of the different instruments are highly variable. As an example, MIDI amplitude values are interpreted differently by different synthesizers, and often by different voices within the same synthesizer; there is seldom a helpful relationship between these values and the output sound amplitude, in dB, say. Another challenge is maintaining good data structures to couple the underlying music to a manipulable graphical interface; this topic was discussed at length by Buxton (1978), and by many others since.

The approach taken here was to treat notes, phrases, instrumental parts, and whole score assemblies as lists of “Events,” and build a graphical editing system to arrange these elements in a visual score. This is easily understood with an example. Music is entered, as by playing a part on the piano keyboard:

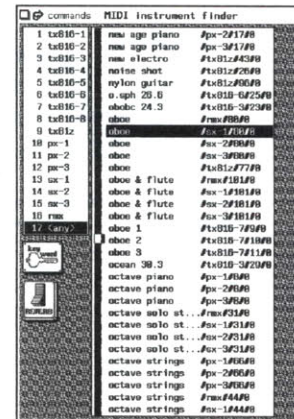


This fragment includes pedal data as well as key velocity information (drawn as grey scale in the note events). This part then becomes an object that can be edited at several levels — one may alter the parameters of individual notes, arbitrary

collections of notes, or whole parts, using a variety of filters (for example, to transpose, stretch, “make staccato,” “make legato,” apply dynamics or rhythmic rubato). To add emphasis to this particular virtuoso riff, the notes in the arpeggio were copied out into a second part that was composited with the first, but with a slight delay to make a double flourish:



Many such parts are then assembled into a visual score, reminiscent of a traditional score but using this graphical performance list notation. A large symphonic piece will have several hundred parts. At a higher level, these parts need to be positioned in time, and the editor supports automatic alignment and stretching to facilitate this (one can graphically “stretch” a part in time to fit it into a particular niche); there were, however, no facilities for aligning notes within parts or for specifying temporal dependencies (for example, it would be useful to somehow “link” the beats so that stretching the solo piano part would cause the rest of the associated orchestral parts to stretch as well). Each part is routed to some number of instruments. A central palette provides patching such that any of several thousand synthetic instruments may be attached to a part — the complexities of MIDI channel numbers, program maps, and other device specifics are subsumed in that abstraction. Because of that, a score needs to be *compiled* to generate performance files (like MIDI files, special piano control files, and perhaps audio event files) so that the instrumental patches can be



centralized patch menu

resolved. Implicit in this is the need to deal gracefully with conflicts if instrumental channels are exhausted, or, for example, to find “best fit” matches if a particular voice requested by a part can only be approximated by the local setup. The compiler dealt with some of these issues, however the problem of timbral mapping remains a research question. It should be noted that we currently lack automated mechanisms for organizing and relating musical timbres, and that means there is no good way to take a request for a particular violin and approximate it with the local palette. The compiler was designed with the idea that at least the instrument names could be used, so a request for a generic “violin” would find something, but this capability was not developed much further.

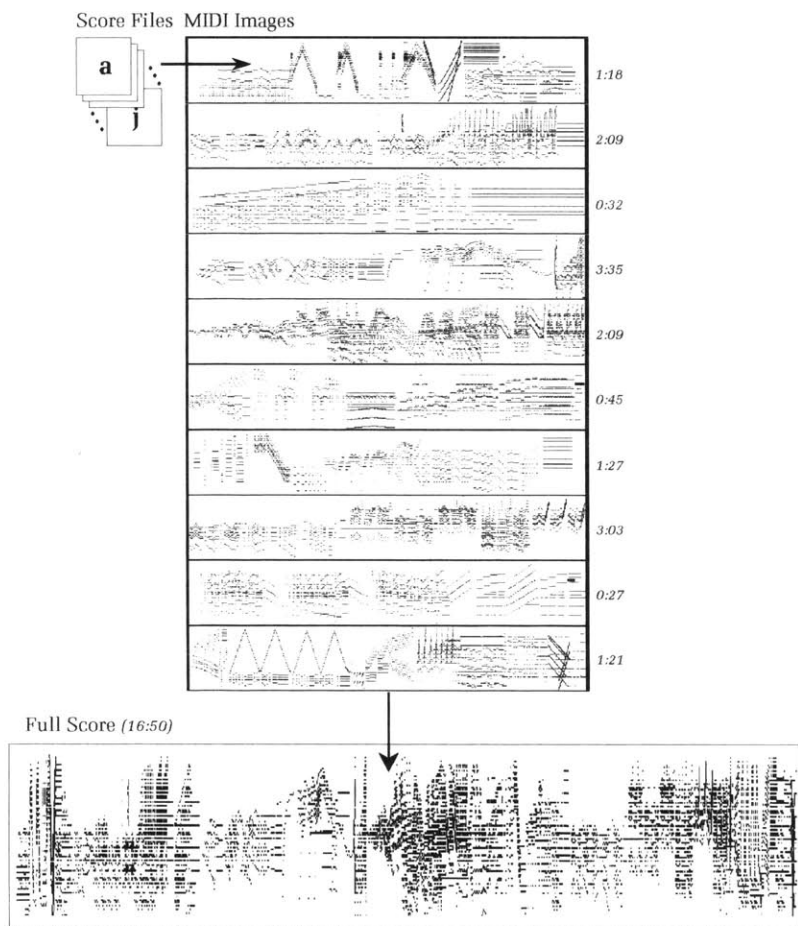
Upon selecting a number of graphical parts, a “play” command runs the compiler to generate a performance data list:

The image displays a musical score for the piece "Totentanz" by Franz Liszt. The score is divided into two main parts: "Piano Solo" and "Piano II (Orchestra)". The tempo is marked "Andante". The score includes various musical notations such as notes, rests, and dynamic markings like "piano" and "crescendo". A diagram on the left side of the score illustrates the structure of the music, with labels for "orchestra (theme)", "piano & timpani", "arpeggio", "composite arpeggio", and "super-glissando". A "cymbal crash" is also indicated. The score is presented in a clear, professional layout, with the title "Totentanz." and the composer's name "Franz Liszt." prominently displayed at the top.

The editing process proceeded quickly — parts were entered, assembled, tested, and refined into a performable score. The



final composition was built by compiling and concatenating all the principal sections into a single score:



It is illuminating to study the visual patterns and “chicken scratchings” in this musical data, since this level of coding is, after all, much of what we would like to derive from acoustic information. At a glance, the eye easily picks out interesting shapes or phrases anywhere in the 17 minute work. It is not hard to imagine many applications in audio browsing that render the structure of a lot of sound in some sort of “atlas”-like form to make it rapidly accessible. Commercial music arranging tools are beginning to develop some of this graphical richness, but that has been sorely constrained given the tiny screens and

limited capacities of current machines. Nevertheless, there is, after all, a reason why composers write symphonic scores on large pieces of paper, and for the same reason, I chose to design with a large graphical layout in mind.

What of the music? Was it good? And what of the process of making it? Was it effective? Much of the editing system was written in a six-month coding “blitz,” and test pieces were orchestrated along the way. The *Totentanz* was assembled in about a month, although that was not a full-time effort, and it was compounded by making a number of changes to the editing system along the way. A score of that complexity could certainly be built in about a week. As for the musical quality, a stated goal was to see if one could approach commercial acceptability. Cheap synthesizers are of course a very rough approximation to a symphony orchestra; in particular, most commercial synthesizers at the time operated at around 10KHz sample rates, and because of the lack of high frequency information and overall richness, the net effect was somewhat “muddy.” The piano part was fine, of course, and surpasses what any human could do; it can be criticized on musical grounds. Overall, the performance was good enough to be considered “real” music (and a CD was issued in conjunction with the *USENIX Computing Systems Journal*, 3(2), 1990).

Two interesting anecdotal observations came from this. First, using an editing system of this sort, it is easy to take a good performance and destroy it, but it is very hard to take a bad performance and make it good. At the moment, it is emphatically the case that time invested in generating good musical input to begin with — that is, “practicing” — saves a great deal of time in the editing stage. I view this mainly as a limitation of the editing tools: there is a lack of control over high-level phrasing, but one could imagine some solutions to that. For instance, it should be possible to take a note-perfect version of a score and impose it on a lively but sloppy performance to remove the wrong notes completely yet preserve the character of the “live” performance. One could also imagine controls for

editing rubato and dynamics; currently these are implemented simplistically (e.g., a choice of scaling methods to use for ritardando) but they could be done in much more interesting ways. For example, after assembling a score, one would like to “conduct it” into shape, by supplying a rhythm (tapping it would do) and thereby synchronizing the score to fit. Methods like this are used in interactive systems like synthetic accompanists, and they have also been used to synchronize digital sound playback with SMPTE or MIDI timing data (Poor, 1992), but they have not fully worked their way into editing systems. The second anecdotal observation is that, although the synthetic orchestra is of middling quality by itself, when the counterpoint contains two or three convincing-sounding instruments (the piano, and a good synthetic flute or woodwind) the overall effect is remarkably good — one doesn’t really notice the weaknesses in the background instruments. This is a perceptual phenomenon that is not clearly understood. It is reminiscent of a similar observation made by Tom Holman (1991) regarding film sound: when mixing a movie soundtrack a black-and-white workprint is used, but when the color reel is finally run, the dialog seems clearer, the localization of panning is more sharply defined, and overall, the sound seems quieter, by perhaps 2-3dB. It is as if the new or interesting information occupies more of the viewer’s perceptual capacity, leaving somewhat less attention for the other information, but generally sharpening the overall sense of “awareness.”

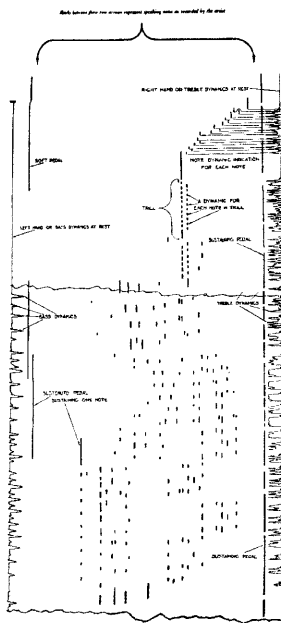
### **A piano roll reader**

Working with Eran Egozy, a paper player piano roll reader was constructed. The idea was to convert the roll data to MIDI as a source of additional musical information. This is a project that is interesting on simple musical grounds: there are about 100,000 piano rolls in existence from the 1910-1930 era, and a number of these are “reproducing” rolls containing dynamic information. The fact that the gestural data rate is so low is what made it possible to record and reproduce it accurately using punched paper and an analog, air-powered “computer” in the first place.

Music rolls are the “fossils” that document the transitional era from little or no recording technology, to plenty of it. Like daguerrotypes they provide a snapshot of a different world.

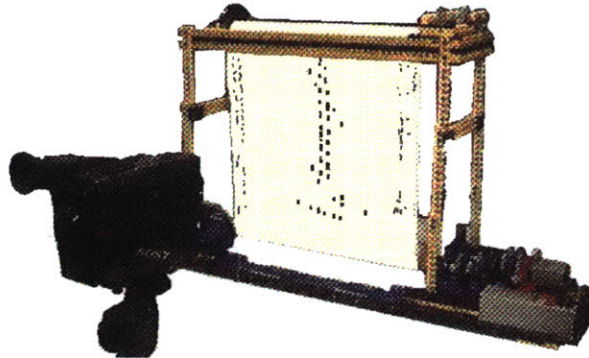
Paper rolls were made to exacting mechanical standards. For instance, Clarence Hickman designed a bellows-controlled servomechanism for Ampico to keep the paper roll accurately aligned with the tracker bar: its mechanical tolerance was .005” (cf. Bowers, 1972). Temporal accuracy is such that one must scan about 100 horizontal lines per second to accurately read a Welte-Mignon roll in realtime, and for that reason, the governor that controls the air motor in a player piano is also precisely engineered. Thus, though we know the data rate will compress to a few thousand bytes per minute, the raw data rate from image is about 100 scan lines per second, where each scanline requires about 500 samples (for about 100 columns of holes and the gaps between them) or about 50k samples per second.

Originally the idea was to “fax” the roll as it rolled by, using a homebrew processor with a 1x1024 pixel CCD array. That approach had a few drawbacks: it required a special board, a dedicated processor and the software for it, custom optics, and because the particular processor being considered was a bit slow, an optical shaft encoding system for a servocontrol loop would be needed to keep the paper running smoothly. In time, however, a cleaner solution presented itself. Instead of building custom apparatus for scanning, or a specially-wired tracker bar, it became possible (using the movie parsing system described in chapter 4) to read and analyze a live stream of video. So, using a Sony “Handycam” for input, we were able to filter the image of the roll and compress it to MIDI in realtime, and all of this was (barely) within the tolerances required. For instance, the camera provides a bit more than 500 pixels of horizontal resolution and was just sharp enough to read the small “fang mark” holes that encode Duo-Art dynamics; and the NeXT Dimension video system provided just enough speed to let us sample the video image in realtime (the roll-filtering process eventually consumes about 64k pixels per second). Fortuitously,

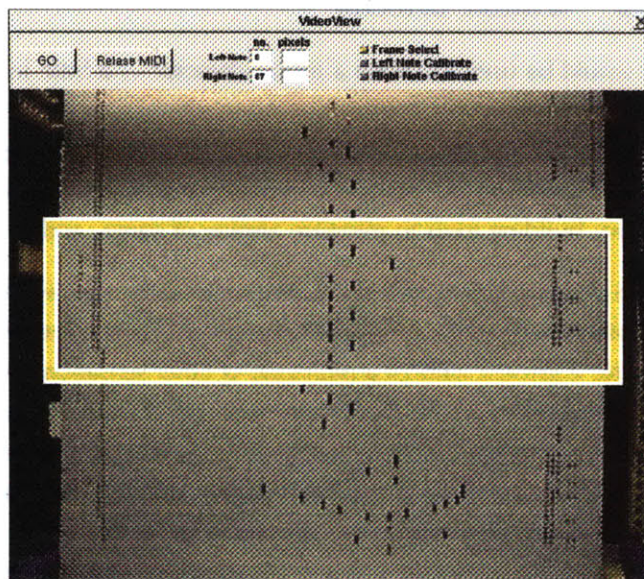


*Welte master roll with dynamic indications. (after Bowers, 1972)*

when reading the image this way, it is not necessary to build a negative-feedback loop to servocontrol the paper speed: the system simply aligns the incoming frames by matching up the holes in the images. Because of this, it was convenient to snap together a “spoolbox” for the rolls out of LEGO toys:



The scanning application, on the NeXT computer, displays the video input, and filters the roll data to produce MIDI in realtime:



It does not yet do a thorough job of interpreting Ampico, Welte, and Duo-Art dynamic codings, although that would not be hard to develop. (One would scan the calibration rolls built for these respective pianos, since they provide exhaustive dynamic tests). The LEGO apparatus is a bit jiggly, although it has worked surprisingly well.

## Summary

The purpose of this discussion has been to demonstrate with a variety of examples some of the expressive power that derives from achieving a dense coding of musical content. It only scratches the surface of what has been done recently, of course; for instance, I scarcely mentioned the work that has been done on interactive music systems (e.g., Rowe, 1993) or the great deal of other recent music systems work. There has been a curious tendency on the part of computer scientists to trivialize MIDI, but for all its faults, it delivers much of the content that is, after all, the goal of a great deal of musical signal analysis. The following sections deal with abstracting some of that structure directly from sound, but the musical examples just mentioned help to remind us of some of what to expect when we succeed.

## 2.3 A music detector

---

It would be useful to have a *music detector* — a little listener a bit like Neuringer's Stravinsky-sensing pigeons, to comb sound waves and find music when it occurs. This would be a good step towards finding the interludes between segments in a television show, building tools to measure airplay of songs, and so on.

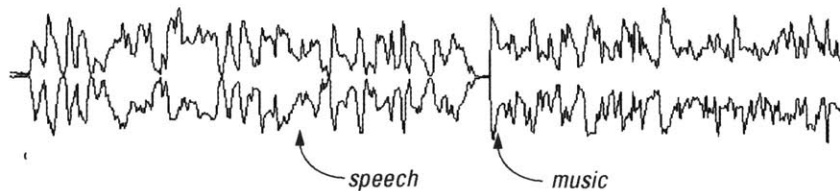
The problem is that, in a sense, one man's data may well be another man's music. How can one reliably determine whether music is present without a multitude of other sensations that more or less describe all the things that music is? It would certainly seem to help if one could first recognize the constitutive components — the timbre of an instrument, the rhythmic tapping of a drummer, popular melodies, and so on, to know whether or not the sounds make music. One could reasonably conclude that a "proper" music detector ought to weigh a collection of sensibilities to come up with a believable response. Therein lie many theses.

Instead of that, we'll try the opposite approach. What are the acoustic features that characterize music? Can we build a workable detector based on that model? As it turns out, what we will examine is an exceedingly simple little listener that very

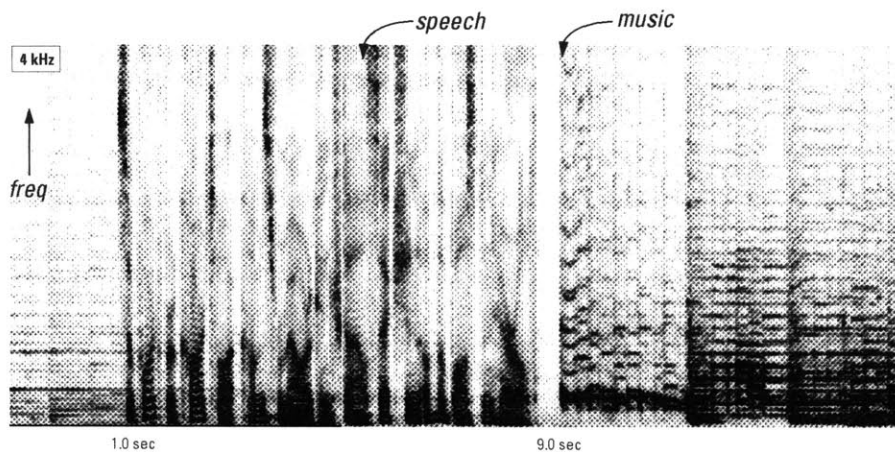
much “combs” signals looking for music, and it works surprisingly well for many common cases.

### Measuring harmonic entropy to find pitched notes

Let us first simplify the problem somewhat by considering only music that has “pitches” — notes with frequencies that remain relatively fixed for a finite period of time. That immediately rules out most drumming, some non-Western musics, and forms like “rap,” but nevertheless, even drums are often pitched, and a great deal of what we commonly call music has pitched notes and chords that make melodies and harmonies. An amplitude envelope, of course, does not reveal that information:



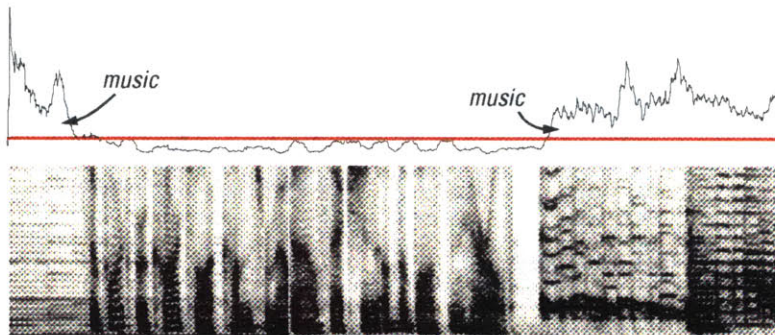
but a spectrum does:



Each vertical slice of the spectrum represents one frame of log-magnitude frequency samples; the greater the magnitude, the darker the sample, so concentrations of energy at peaks appear as black points. This spectrum shows about 13s of sound: about 1s of lead-in music, followed by 8s of speech, then more music.

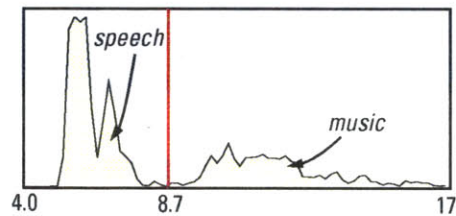
When music is present, combs of relatively fixed-frequency harmonics are seen, giving the spectrum a striated appearance: the music at either end contrasts sharply with the speech in the middle, which also contains combs of harmonics, but unlike most music, the harmonics of speech vary rather rapidly in pitch.

One way to measure this is with a filter that senses “striation” by sliding an analysis window over the spectrum, collecting the peaks in frequency magnitude, and keeping a running tally of the average duration of a stationary frequency peak. This reflects the predictability of a peak (high for music, lower for speech), or inversely, its entropy (higher for speech, lower for music). When the predictability of a peak exceeds a threshold, there is likely to be music present:



Output of a harmonic prediction filter (a music detector)

In much of the sound I have studied, and especially in entertainment media, the distinction between music and speech is so sharp, and there is sufficient music in the sound, that a histogram of the prediction filter output reveals the two modes quite clearly:



(The horizontal axis is the average length of a frequency peak in frames; the threshold here of 8.7 frames is about .05s at 8KHz)



with 48-sample frame increments). Now what we wish to do is, arrange things so that a sound filtering process will output a list of *events* or segments that contain music. It is useful at this point to consider the implementation in a bit of detail because it yields components that will be handy in other situations. Then we will test it on some long samples of sound and consider its limitations.

### Implementing the music detector

The form of many filters like this is that of a function applied to sequential frames of input data, as is familiar from short-time Fourier analysis. This is implemented in a straightforward way:

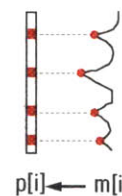
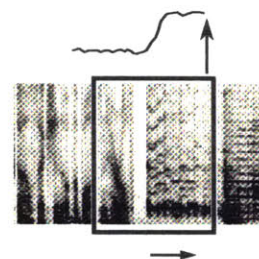
```
filter( func, offset, num_frames)
```

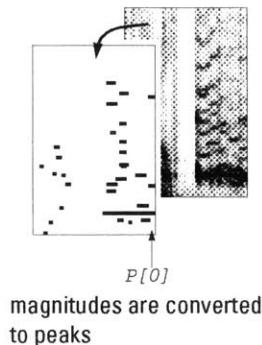
slides a filtering function, *func*, over the input sound (optionally beginning at some offset in the stream, and running for some number of frames; external variables control the frame size and increment). As the analysis progresses, *func*(*s*, *n*) is called with successive frames of *n* samples, *s*[]; in most cases it is convenient to use one-channel 32-bit samples.

The *music* filter maintains a sliding *window* of frames (think of it as a 2-d rectangle sliding over the image of the spectrum), and within the window, keeps a moving average of peak duration. When a new sample frame arrives, the oldest frame is removed, the new frame is inserted (effectively stepping the window ahead one frame in time), the peak tallies are updated, and the new estimate of peak predictability is output. To “insert” the new frame, we compute its magnitude spectrum

$$m[i] = \log|F[i]|, \text{ where } F[i] \text{ is the discrete Fourier term at frequency } = i * \text{SampleRate} / \text{FrameSize}$$

then locate the peaks in the spectrum *m*[] to obtain a *peak frame*

$$p[i] = 1 \text{ if } m[i] \text{ is a local maximum (in } i), \text{ else } 0$$




It is these peak frames that are kept in the analysis window. This is implemented as a modulo-indexed delay buffer,  $P$ , where  $P(0)$  gives the current peak frame,  $P(t)$  gives the  $t$ -th previous peak frame, and  $P(t)[i]$  is true if the  $i$ -th bin of the  $t$ -th previous frame contained a peak. In this way, the original window of magnitudes is efficiently reduced to a binary window of peaks. We then compute the average length of a run of peaks in that window, but since we wish to emphasize runs if they are present, we first drop the singletons (the isolated peaks) and we “fill in the dotted lines” with a rule that plugs small gaps in runs of peaks (the rule is simply: **1011**  $\rightarrow$  **1111**). This process of peak detection in frequency followed by peak tracking in time can be thought of as analogous to convolution of the 2d spectral “image” with a horizontal line detection filter. In this way we obtain an average that adequately reflects the “longish” runs that characterize sustained musical tones.

The **C** code for the music filter is this:

```
music(int *s, int n){
    int m[N], h[N], *p3, *p2, *p1, *p0, *pl, n2=n/2;

    // decrement peak accumulators:
    for (p0=P(0)+MinBin, pl=P(0)+MaxBin; p0 < pl; p0++)
        if (*p0 > minRun) peakN--, peakT -= *p0;

    // find a new frame of peaks
    Spectrum(m,h,s,n); // m=magnitude data; h=hartley data
    findPeaks(P(0),m,n2); // P(0)[i] = 1 if m[i] is a peak

    // 4-pt smoother; emphasize sustained peaks: 1101=>1111
    smoothPeaks();

    // merge incoming column of peaks
    // and increment peak accumulator
    for (p2=P(4)+MinBin, p3=P(3)+MinBin, pl=P(3)+MaxBin;
        p3<pl; p2++, p3++)
        if (*p2 && *p3 && *p2 < maxRun)
            *p3 += *p2, *p2 = 0;
        else if (*p2 > minRun)
            peakN++, peakT += *p2;

    output((float)peakT/(float)(peakN? peakN:1));
    CurFrame++;
}
```

This relatively tidy function follows the method outlined above (drop the oldest peak frame, decrement counters, find a new peak frame and insert it, smoothing the peak runs), but a few other details should be noted. The spectrum is computed using a discrete fast Hartley transform (DHT) with a raised cosine window. This is analogous to a discrete Fourier transform (DFT) except that the *complex* “cis” kernel in the Fourier transform

$$F(f) = \int x(t)e^{-i2\pi ft} dt = \int x(t)\text{cis}(2\pi ft) dt$$

becomes a *real* “cas” kernel in the Hartley transform:

$$H(f) = \int x(t)(\cos(2\pi ft) + \sin(2\pi ft)) dt = \int x(t)\text{cas}(2\pi ft) dt$$

Thus, Hartley is related to Fourier as:

$$H(f) = F_{\text{real}}(f) - F_{\text{imag}}(f)$$

The Hartley transform was due to R.V.L. Hartley (*A More Symmetric Transform*, 1942) and has been extensively described by R. Bracewell (1984, 1986). One of its chief virtues is that the same real kernel is used in both analysis and synthesis equations. That is, Hartley is its own inverse (and “more symmetric”), whereas in Fourier, the complex exponential kernel changes sign between analysis and synthesis equations, which is what gives rise to its conjugate pairings and Hermitian matrices. That represents a twofold redundancy in terms that Hartley avoids. In computer implementations of Hartley, a single function suffices for both analysis and synthesis, and the multiplications are all-real, not complex. There are of course well-known methods for eliminating redundant calculation in implementations of Fourier, but use of the Hartley transform avoids these nits from the start.

As an additional expedient, the choice of the cosine window means that instead of multiplying the input function by a

*“Since, as has been known for a century and a half, Fourier’s coefficients are complex, it is perhaps hard to accept that the complex numbers are a construct of the human mind rather than of nature. Of course we all accept that a power spectrum, such as an optical spectrum, is described by a real function of a real frequency variable,  $f$ , but is not the strength of  $a_f^2 + b_f^2$  of that power spectrum to be computed as the squared modulus of the complex coefficient  $a_f + ib_f$ ? Well, yes, that is one way. The other way is to compute  $H(f)^2 + H(-f)^2$ , where  $H(f)$  is the (real) Hartley transform.”*

— Ronald Bracewell  
(1986)

window function and then taking the transform,  $H(x \cdot w)$ , one may convolve the transform of the input with the transform of the window,  $H(x) * H(w)$ , which in the case of a cosine (*hanning*) window is simply convolution with  $[\cdot 5, 1, \cdot 5]$ , and that can be done with adds and shifts. Synthesis can then be performed by overlap/add. When combined with a particularly fast implementation of the discrete fast Hartley transform (by Massalin, 1992) the result is a neat and efficient function for spectral analysis. The music filter with conservative settings shown here runs about 8% faster than real-time on a Motorola 68040 computer; with settings appropriate for skimming hours of sound, like movie soundtracks, it runs about 5x faster than realtime.

There are just a few other subtleties to mention regarding the *music* function. One is that, in this implementation, the tracking of peaks is restricted to a band of about 150–1000Hz. This decreases the computation by a considerable factor but still captures most of the musical information. Computation may also be reduced by skipping the filtering function more quickly along in the input stream (the default increment is quite conservative). Another small point is that peak-runs are truncated if they exceed a certain duration (about 1/3 the duration of the peak-sampling window). This has the effect of reducing the impact of long and often non-musical drones, like harmonics from a buzzing power supply (of course, it may also reduce the overall responsiveness to very slow-moving music). Finally, the frame size is chosen so that the discrete frequency samples in the spectrum are spaced about 16Hz apart, which is wide enough to track small frequency variations like light vibrato, yet narrow enough so that most of the musical harmonics are distinct while most of the pitch variation in speech crosses the bins.

In order to segment the output of this function as musical *events* it suffices to threshold the peaks and then collect these segments (dropping segments that are too short, and collecting segments

that are close together into a single segment). Because this last clustering operation is done without regard to spectral content it may truncate or attach extra “leader” to segments.

The final result of all this is a Unix-style filter that outputs a list of “music” events, tagged with a weight indicating the likelihood that they actually do contain music:

```
music file.snd
file.snd| 0.00.00| 0.09.55|music p=7.94
file.snd| 1.25.25| 0.31.63|music p=7.43
file.snd| 1.59.95| 0.27.26|music p=8.31
. . .
```

There are certainly many other ways in which a similar overall measurement might be made, but it is likely that they will seek essentially the same features. This implementation has the advantage of being intuitively clear and straightforward to implement. Now let us see how well it works.

### Testing the music detector – All Things Considered

In his master’s thesis (*A Graphical Interface to Audio News*, 1993) Chris Horner used this music filter to parse broadcast radio programs. Music is used as a kind of “signpost” to delimit the segments of many radio shows, so it is a valuable cue to find. After a little bit of trial-and-error work, Horner set thresholds so that the filter ran about 2x realtime (that is, a 30-minute radio show was parsed in 15 minutes) and it found most of the musical segments with relatively good success — perhaps 2 errors per hour. What is somewhat more interesting is the degree to which music figures, structurally, in radio shows.

Logs of the National Public Radio program *All Things Considered* showed an average of seven musical segues per show. Frequently the music is used to tag unannounced stories – not so much “hard news” pieces as interviews, editorials or color stories. Some radio programs are sufficiently well-structured that

<i>min</i>	<i>dur</i>	<i>desc</i>
0:00	65	summary/top story
7:00	38	still to come...
12:55	10	feature story
15:39	30	corporate news
17:27	46	stock report
18:38	10	daily business press
19:36	10	feature story 2
23:36	10	feature story 3
26:37	91	final notes

The **Marketplace** radio format  
(after Horner, 1993)

the segments within a show occur at approximately the same interval, and are tagged with music in consistent ways. In particular, the half-hour NPR show *Marketplace* consists of about 9 segments punctuated with musical segues ranging between 10s and 91s in duration. These segments are rather reliably positioned (for instance, segment 3 is generally a feature story, and segment 5 the stock report). With the music detector, Horner’s software could use the overall pattern of musical interludes to identify the radio program, as well as to locate individual segments within the program like the feature stories. A browsing interface could then fetch particular segments as needed. Horner noted that sometimes the musical segues (the long ones) overlapped the next introduction (that is, the announcer starts talking while the “band plays on”). By design, the music filter still signals high while the music is fading, even though other sounds (like an announcer) may be mixed in, so other methods were needed to find accurate starting locations for some segments.

The music detector makes two sorts of errors — errors of omission (failing to find a segment of music), and false hits. Very roughly, these rates were about 2 errors per hour, and generally, they were false hits. In the *Marketplace* study, Horner found that during a week of broadcasts, only two of the music tags were not detected at all by the *music* filter, and there were an additional eight false hits (1.6 per day) one of which actually was music playing in the background during an interview (an appropriate find, but it might confuse a higher-level pattern matcher).

**Limitations and extensions**

The music detector as described works surprisingly well considering what should be obvious shortcomings. For instance, touch tones and car horns would be marked as probable musical segments. Perhaps they are, in a certain definition, but many such mislabellings can be correctly dropped by higher-level

agencies. It is often reasonable, for example, to ignore short segments (we expect music to last for more than 5 or 10 seconds) and treat the quick bursts as strongly-pitched effects to be handled by some other agency. In addition, when sustained stretches of music occur, there are sometimes lulls or pauses that break up what should be a contiguous segment, and the short dropouts can be ignored. This amounts to gestalt-like clustering of events according to temporal proximity.

The music detector also misses certain things. As already mentioned, it won't catch most rhythm-only tracks. However, these might be found by autocorrelating the power spectrum to obtain the regular beats, and if not that, by using a click finder to send beats to a rhythm extraction process. Rosenthal (1993) showed some ways in which a stream of individual events can be parsed to find their rhythmic underpinnings, if any. Mont-Reynaud and Goldstein (1985) have also studied this problem, although their interest was more in analyzing pitched music. The trouble is, in the absence of other intelligence, *any* periodic tapping (like footsteps, or a clanking piece of machinery) would pass by that measure. This suggests that some sense of timbre be used — e.g., recognize the specific percussion instruments. Although Schloss (1985) had some luck at transcribing drum sounds, percussion instrument sounds can not yet be identified well. Likewise, the effect of "Hitchcock strings" (the shrieking violin glissandi famous from *Psycho*) would never be construed as music unless one could infer that the sound came from an orchestra. Probably most people can't (and certainly few do when they are watching a horror movie climax). There will always be a gray area between special effects that function musically, or quasi-musical elements used for effect.

#### *Music removal*

In the classic Dr. Seuss movie *The 5000 Fingers of Dr. T*, the hero invents an atomic "music fix" to suck music out of the air, the key to stopping the evil Dr. Terwilliger. It is often useful to separate the music from other sounds for quite pragmatic

reasons. This is not easy to do well, in general, but the music detection filter can be adapted into a music separation filter (and in fact, this is a useful way to verify that the music detector is actually latching correctly on musical information).

The idea is to select only the music-bearing components for resynthesis (or the non-musical ones as the case may be). The problem, of course, is to accurately identify those components, and as suggested, one way to do this is by applying an edge enhancement filter to the spectral image so that primarily the sustained musical harmonics are marked for resynthesis (or removal). The program *musyn* (appendix 1) does exactly this; in particular, the command

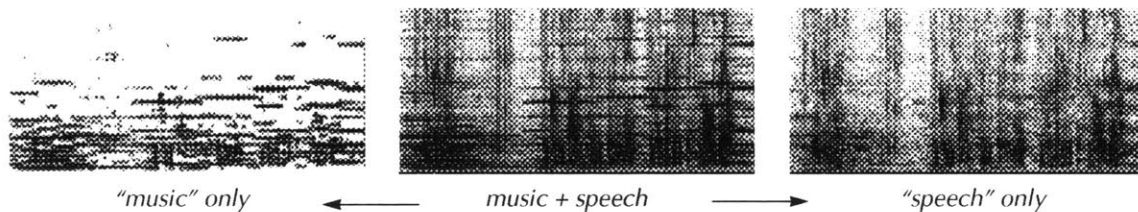
```
musyn [-m] soundfile | pplay
```

extracts and plays the “musical” part of a soundfile (or if the *-m* option is given, plays the nonmusical portion). This is a useful technique for roughly separating speech from background music, and in this way the music can be operated upon independently; at least, enough of the primary pitch content is present to be useful, as will be seen in chapter 4.

There are, of course, several shortcomings to this method. First, the musical portion of the signal, in practice, is not completely characterized by line spectra: vibrato, particularly in high harmonics, will be missed, as will noise or the blurring of peaks near attacks, and harmonics from monotonous speech will be collected by mistake with the music. Second, the filter function of the collective musical source is not easily ascertained, which means that simply combing out peaks will not cleanly remove the musical parts. Third, when other sources are mixed, unless one takes care to estimate the amplitude and phase of sustained musical harmonics so that the right amount of energy can be removed, unwanted bursts of sound will be included in those cases.



There have been other approaches to this sort of separation task. The homomorphic vocoder (Oppenheim, 1969) uses cepstral analysis (the Fourier transform of the log-magnitude spectrum) to determine pitch and impulse response of speech signals, and Miller (1973) used this technique to separate singing voices from orchestral backgrounds, but because the cepstrum tracks the most prominent pitch (which is not always the singer) numerous errors had to be corrected by hand. That is, in the absence of other disambiguating information, it is difficult to precisely sift out the components of the signal that belong to a particular source, especially one so variable as “music.” However, voiced speech is so distinctive that it often can be separated with some degree of accuracy. Parsons (1976) showed a method of detecting and selecting the harmonics in speech when another unwanted voice overlapped; the method worked well but had difficulty in dealing with consonants and non-vocalic speech. Ellis (1993) has been developing perceptually-oriented approaches. One can imagine several ways of dealing with this problem, but the simple approach used here works well enough for many applications. This illustration shows a sample of sound



separated into music and speech by this method. The “music only” separation contains some speech harmonics, and the “speech only” separation includes some residual music, but the separation is sufficiently clear to be useful.

## 2.4 A polyphonic pitch extractor: the inverse piano

The “inverse piano” is reminiscent of an ordinary piano, except instead of putting gesture in and getting piano sound out, you put piano *sound* in to get *gestural* data out. Thus it is a special instance of an audio event filter, but nonetheless an interesting

*“Several years ago, I recorded four piano notes — the lowest four D’s, **mf**, no pedal. They were digitized and then analyzed (a week’s work).”*  
— Richard Cann (1980)

one, for several reasons. First, polyphonic pitch transcription is an extremely difficult problem, but the piano is a relatively simple instrument: notes are struck strings with no vibrato or amplitude modulation, and notes are largely superposed to form chords. But the apparent simplicity of the instrument contrasts with what happens when it is vigorously played: it is one thing to identify an isolated note, but quite another thing entirely to transcribe Art Tatum playing at full speed. Second, it is a useful problem in and of itself, for reasons not unlike those that apply to reading piano rolls, except that in the case of recorded piano music, there is nearly a century of interesting musical information that would potentially be available for analysis if this technique were perfected. And third, as we will see, it leads to other ideas for future recording systems.

### Two Clues

Here are two clues before we begin, one dealing with the distribution of harmonic energy in conventional piano playing, the other dealing with the nature of piano chords.

Consider an analysis/synthesis filter that, for each input frame of sound samples, obtains a magnitude spectrum, finds and sorts the peaks in order of magnitude, and then resynthesizes only the  $N$  most prominent peaks in each frame. The following Unix filter

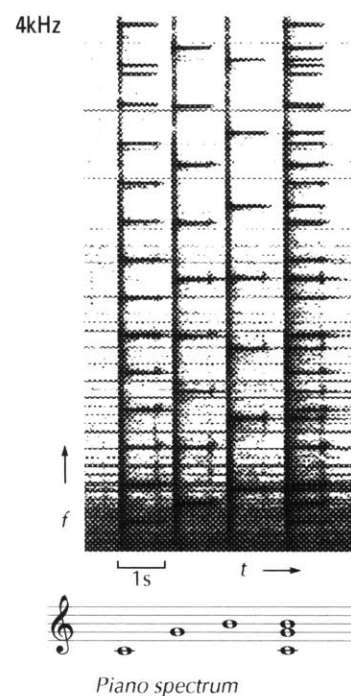
```
anasyn -p 4 soundfile | pplay
```

does exactly that. We observe that by resynthesizing 1 peak in speech (in a 512-sample frame) the result is unintelligible — a skittering sine wave that roughly follows the inflectional contour. But with 2 peaks, the speech can almost be understood. With 4 peaks, the speech can be understood and the speaker can be pretty reliably recognized. With 8 peaks it sounds like a slightly strange but basically acceptable analog telephone connection. These represent compression rates that are already in the 50:1 to 100:1 range (about 80 bytes/second) and we have done little to intelligently track and code the formants and many redundant harmonics.

In music, the same thing is observed: a piano recording of Glenn Gould playing Bach is resynthesized using just 4 peaks, and we perceive most all the polyphony. Of course, with just 4 peaks, the noisy “crack” of the attack is not resynthesized, so the effect is rather like a glass harmonica. Nevertheless, most all the pitch content comes shining through. This immediately tells us that, as a “first pass” in a pitch extraction process, we can narrow the search from several hundred frequency magnitude samples to a small handful (a dozen, say).

The second clue involves what happens with piano notes when chords are played. A physicist excited about the prospect of finding chaotic, nonlinear couplings came to my lab eager to take a close look at vibrating piano strings: surely there would be some interesting interactions between the notes. I was skeptical, since simple sampling synthesizers sounded pretty good, but he insisted that with a little nonlinear coupling between the string synthesis elements they would sound even better. We tried the following little experiment: the Bösendorfer was programmed to play three notes, **C4**, **G4**, **B4**, first in sequence, then as a chord. Each note was played for 1s at the same amplitude, as was the chord (the spectrum and the notes are shown). The piano’s performance was recorded (in stereo, at 44.1KHz, with 16-bit samples). Using this soundfile, the same 3-note chord was then synthesized by simply taking the three individual note samples and adding them together (this was done by simply eyeballing the attack points). We then compared the two chords, listening and looking at spectra; could we perceive a difference?

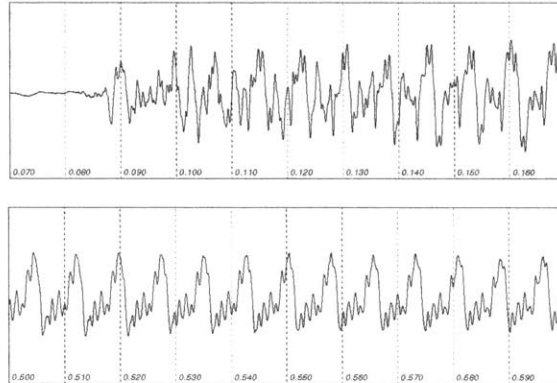
The answer was: no. The chords were utterly indistinguishable. This doesn’t mean that there aren’t interesting cross-couplings to be found, only that the piano sound is additive to a very high degree. These two clues give us some hope that we can indeed write a program to perform this deconvolution.



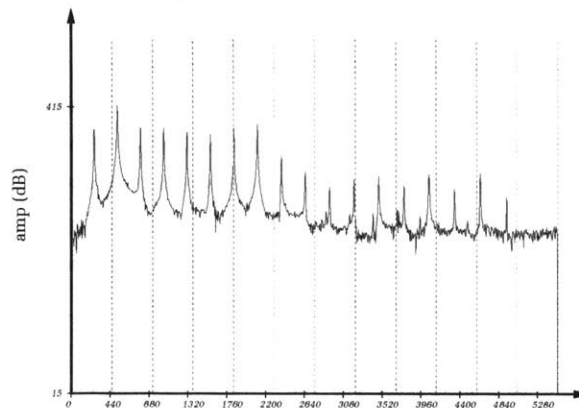
### The nature of piano sound

Before plunging into an implementation, it is useful to examine the sound of piano notes more closely. A typical note begins

with a noisy “crack” and then quickly settles into periodic oscillation as the resonant strings decay:



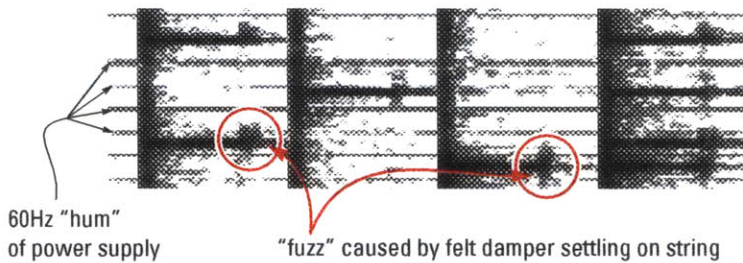
The attack portion of this note (a **C3** around 140 Hz) begins at .088s and by .110s (.02s into the sound) a just-discernible period begins to appear. In fact, if just a few cycles of the note at the attack are played, say from .088-.115s, the pitch can often be discerned! This is somewhat puzzling until one realizes that the sensation of pitch at that frequency may be conveyed by a few high frequency harmonics, before the fundamental settles in — if indeed it ever does: many low bass notes do not have a discernible fundamental (which makes sense since the hammer is striking a long string very near one end, so it is not inclined to vibrate in one piece). The case of the missing fundamental is well-known, and easily understood when one looks at the



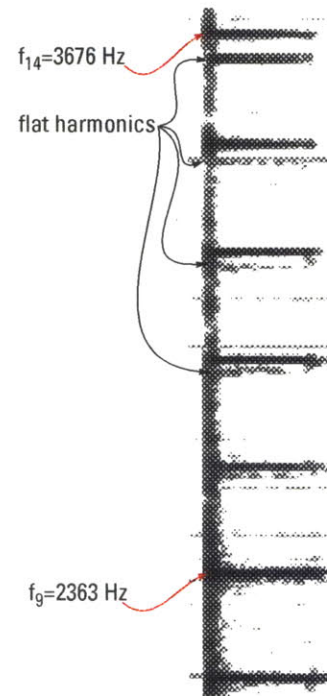
frequency spectrum of a note. Shown above are the first 18

harmonics of **C4**,  $f_0=262\text{Hz}$ , and the upper partials occur in a comb at roughly integer multiples of the fundamental,  $f_n = nf_0$  (in the case of pianos, string stiffness causes the harmonics to stretch sharp slightly). We remember from Helmholtz' studies of combination tones that when two tones  $f_1$  and  $f_2$  are played, we may also perceive a "beat" frequencies or difference tones at  $|f_1 - f_2|$  and  $f_1 + f_2$ . A similar effect applies here, and the gaps between harmonics in the comb all indicate  $f_0$ .

When the note is struck, a noisy "crack" appears as a broadband click (a vertical line) in the spectrum, although if it is played very softly, that click may not be present. After the note is struck, the strings decay exponentially (which is to say, linearly in decibels) until the note is released, at which point the felt damper settles upon the string. That contact often makes a very faint "fuzz" as can be seen in this close-up of the spectrum:



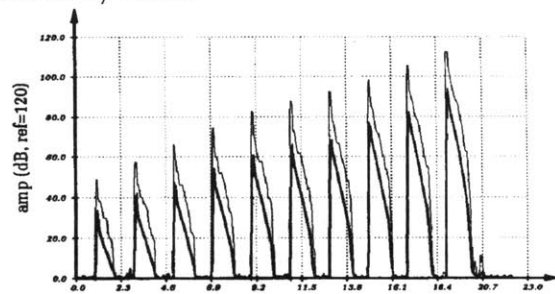
In addition, many of the notes on the piano are composed of multiple strings (on the Bösendorfer, notes from **D2** on up have three strings), and if one of these slips slightly flat, the tone sounds "shimmery" because of the summation tones caused by the not-quite-aligned combs. The diagram at the right shows this. We can see that the first **C** is pulling slightly flat: starting at about the 9th harmonic (3676 Hz), the "ghost" of a flatter harmonic begins to appear just underneath the "true" one, and it becomes quite prominent around the 13th harmonic (3676 Hz). This indicates some of the precision with which we can filter. A great deal more information about piano acoustics can be found in Hall's series of six articles in the *Journal of the Acoustical Society of America* (1992).



One of the three strings has pulled slightly flat.

It is clearly possible to identify a single pitch reliably — it is safe to assume that the strongest harmonic is either  $f_0$  or  $f_1$  and proceed to use an interpolating comb to single out the pitch. Similarly, the amplitude of the note may be taken to be the sum of the energy in the separate harmonics: Bruno Repp (1993) has shown that overall amplitude may be adequately determined from the power of just the first two harmonics. Amplitude may be subjectively altered, but in any case it is easy enough to derive a table to map dB in amplitude to amplitude (hammer velocity) on the reproducing instrument; for the Bösendorfer, the scaling is approximately linear:

midi-amp	dB@120
60	34.00
70	40.12
80	46.17
90	53.28
100	60.79
110	68.01

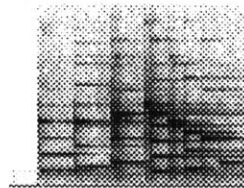


### Finding and identifying note onsets

To detect note onsets, at first it seems sufficient to look for that “crack” of the hammer (which appears as a vertical line in the spectrum:



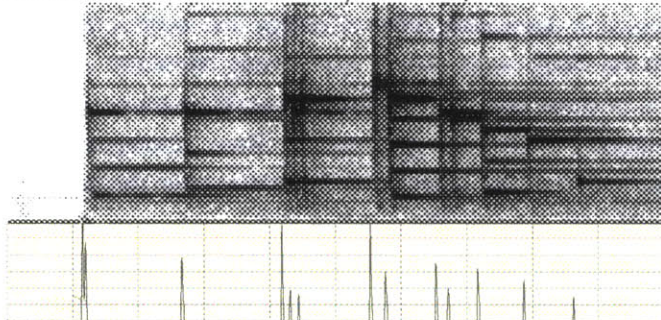
*J. S. Bach, Goldberg Variations  
Bach-Gesellschaft edition (1863)  
(Dover, 1970)*



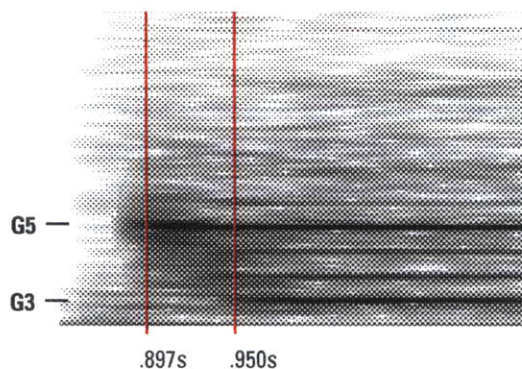
*Glenn Gould  
Salzburg Music Festival (1959)  
(Price-Less CD D15119)*

but on closer inspection we see that very soft note attacks don't have much noise component; the last **D4** in the example above

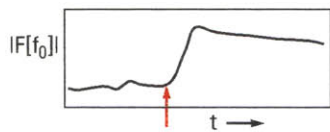
is quiet enough so that the new harmonic is the primary (and almost the only) cue. New notes are better characterized by the advent of new harmonics. Filtering for this (essentially by convolution with a bipolar operator, which smooths and differentiates) we can accurately find the possible onsets:



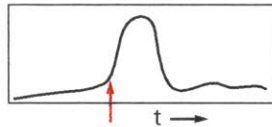
In this case, the differentiation is being done on log-magnitude values obtained by incrementing the analysis frame in 32-sample (.003s) "hops." Notice that the first attack is slightly "fat," *i.e.*, spread in time: in fact, two attacks are barely discernible. That is because the first two notes of the aria, **G3** and **G5**, perceived as an octave chord, are actually skewed by about .05s (other studies of piano playing, *e.g.*, Palmer 1989, have shown that melody/accompaniment attacks are often slightly skewed in time). This can be easily seen in a closer look at the spectrum:



(This image was made by sliding the analysis window in .001s increments). This is not quite a just-noticeable difference, although it can be measured.



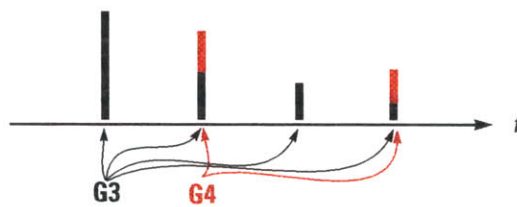
Harmonic of a probable attack



... a nearby harmonic shows the noise of the attack, but not the sustained resonant energy.

Having located probable attacks, we now identify the pitches by analyzing the new harmonics. In particular, around a given attack time,  $t$ , we analyze the magnitude spectra to find those harmonics that have increased significantly in magnitude, taking care to look for “steps,” not impulses. We then sort the harmonics in decreasing order of energy and build a table to weight the likely notes: that is, each strong harmonic could be  $f_0$ ,  $f_1$ , or  $f_2$  of some note, and the table  $n[pitch]$  contains the number of harmonics tallied for a given *pitch*. Then, starting with the loudest low note, we subtract its harmonics from the spectrum at that point and consider the residual in the same way until all the new note onsets have been identified. Having found the note onsets and identities, the harmonics are summed to obtain amplitudes, and tracked to locate offsets.

Implicit in the subtraction is some sense of the shape of the harmonic envelope: upper partials decay in energy. A common, troublesome idiom is shared harmonics due to fifths, and especially octave-playing, since all the harmonics overlap:



If the upper note is so soft that it appears to be the *second* harmonic of the lower one, the problem is more difficult, and can be compounded by phase cancellation. This case is quite common with right-hand (treble) octaves, and often to an astute human listener, the “octaveness” of the sound is barely discernible as a slightly brighter timbre caused by the “tilt” due to the energy added in the upper partials by the higher note. To the extent that the spectrum of the lower note is accurately known, it can be accurately subtracted, leaving the residue of the upper note which will be collected by a subsequent iteration. That will work.



## Testing the polyphonic pitch extractor

The result of all this work is another event-producing filter to add to our repertoire:

```
polyp aria.snd
| 0.886| 1.045| G5 798.15|25.23
| 0.956| 1.652| G#3 203.45|20.98
# Hmm; seems a little sharp! A440 => 446.713074
# start| dur|note| db(abs rel)|freq
/tmp/t1.snd
| 0.851| 1.069| G5|25.23 62.12| 798.152
| 0.891| 0.652| G#3|20.98 49.21| 203.451
| 1.929| 1.142| G5|25.97 45.67| 798.152
| 3.011| 0.092| A5|28.15 54.33| 892.052
| 3.107| 0.148| G5|27.96 54.28| 798.152
| 3.195| 0.471| A5|27.76 41.84| 892.052
| 3.974| 0.256| C6|28.41 52.29|1017.603
| 4.137| 0.871| A5|29.06 50.34| 892.052
| 4.684| 0.683| G5|27.85 53.35| 798.152
| 4.816| 0.423| F#5|26.75 45.48| 751.202
| 5.136| 1.422| E5|27.25 54.87| 672.952
| 5.635| 1.302| D5|26.54 52.04| 594.702
| 6.178| 0.962| D4|23.45 52.67| 297.351
. . .
```

Two small mistakes were made: the second **G3** (203.45Hz) was marked as **G#3**, and the high **B5** (1017.25 Hz) was marked as **C6**. These errors are because the piano on this recording is slightly sharp, by about 1% (for G5,  $798.15/783.991 = 1.02\%$ ), so **A4** should be set at about  $440 \times 1.02 = 448\text{Hz}$ . The addition of an auto-tuner fixes these note-spelling problems:

```
# Hmm; seems a bit sharp! A440 => 446.713074
```

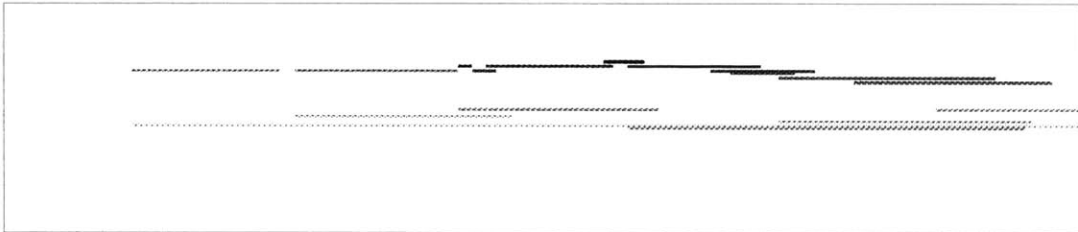
This is straightforward: relative to **A4=440**, some of the pitches will be either sharp or flat; if the majority of notes are sharp (say), a few sharp notes frequencies are compared with their correct frequencies at 440Hz to obtain the percentage by which to retune the filterbank.

Now we can run the test sample through:

```
polyp aria.snd | e2midi | midi2ps
```

The two filters convert to MIDI and then PostScript, respectively (gray scale indicating relative amplitude):

*Aria -- 00:07.147*



The tracking of sustained notes is somewhat delicate; essentially, the process follows resonant harmonics until they decay rapidly or fall below a threshold. These constants are currently set by measuring the recording, which is not yet done automatically. Setting the decay threshold too low produces a staccato effect which is useful for rapid auditioning:

*Aria -- 00:06.305*



### Limitations and Extensions

Built into this extraction process are basic properties of piano acoustics — for instance, that chords are superposed; that note onsets are relatively abrupt and can thus be detected by a simple differentiator; that note decays are exponential, so offsets can be found similarly; that the relative strength of the harmonics is such that the strongest harmonic is usually  $f_0$  or  $f_1$ ; and that there is no vibrato or amplitude modulation to speak of, which means the filter is well-suited to the task of deconvolving piano or piano-like sound as opposed to saxophone playing. But there is no knowledge of higher-level idioms that could be used to “fill in

the blanks” for instance, to find notes that may be so poorly articulated or buried in the texture of a thick chord that other expectations are required in order to sense them. Clean, two-voice Bach playing poses few problems; fast left-hand boogie-woogie octave tremolos are nasty.

Pedalling is essential in beautiful piano playing, but it is currently not dealt with at all in this filtering process. It is expected that damper-pedalling will change the decay rate overall and perhaps boost the higher harmonics. It changes the sound quality in ways that can be appreciated but are still difficult to precisely intuit even for an expert human transcriber. In any case, the problem has not been well-studied. The “soft” pedal (or *una corda* pedal, so-called because it shifts the hammers so that they strike fewer than the three strings, or in the case of the Bösendorfer, simply causes the hammers to strike the strings with less-worn parts of the felt) also changes the sound quality, “softening” the tone. Again, this should be measurable in the spectrum, but is even subtler than the use of the damper pedal, and likewise has not been studied.

This study reveals a few important things. First, as hoped, piano sound is straightforward enough so that automatic transcription can be done. However, filtering must be done extremely precisely, and the detection of faint or obscured notes needs to be improved either by knowledge of the musical idioms (octave playing is a common example) or by precise knowledge of the instrument itself. The use of idiomatic knowledge depends on the source or style of playing; for example, Bach playing is often simple enough that all the notes can be clearly heard by a human as well as a machine, but blues and jazz require more information to resolve. On the other hand, the small study of chord superposition indicates that, because notes are primarily added, they can also be subtracted cleanly. Thus, given a complete “sampled” model of the piano we should be able to remove notes from the texture to accurately “unravel” complex chords. Further research is required to perfect the low-level

filtering and to explore the use of higher-level knowledge to improve the process.

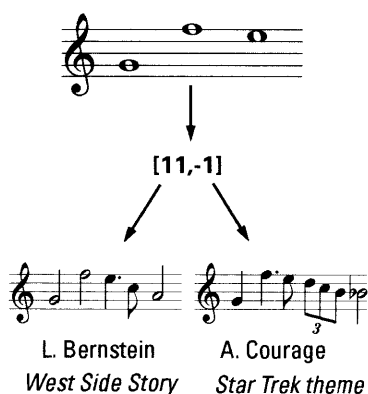
## 2.5 Name that tune

*“Permeating the work of many of the great and prolific composers, we find a certain combination of notes, a certain ‘melos.’ This ‘melos’ or melodic line seems to be a strong ingredient of their style.”*  
— Sam Morgenstern (1948)

“The essential basis of music is *Melody*,” wrote Helmholtz, noting that harmony was a relative latecomer to western music, and that many non-western musics have no harmony, but nearly all have melody. Like the words in speech, melodies carry much of the critical meaning in music. The ability to appreciate a melody clearly ought to be in the skillset of a listening machine; it is not just a game of “name that tune”!

In 1948 Clarence Barlow and Sam Morgenstern assembled a dictionary of musical themes — a collection of about 10,000 famous melodies, indexed by title and by tune. As Morgenstern was acutely aware, one can tell a lot from a melody: obviously, the tune bears a certain “fingerprint” of the composer, but it is also true that melodies from the same era sound similar, as do melodies from the same country (Hungarian tunes, Russian tunes, French tunes, and so on). The indexing method used by Barlow and Morgenstern consisted of transposing the melody to the key of C; thus, only the relative pitch sequence was used.

### Name that tune for MIDI



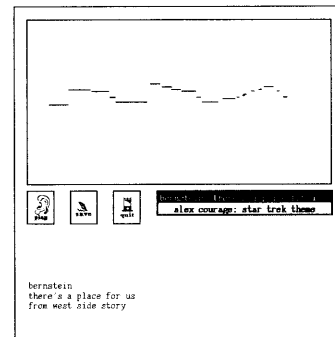
In 1988 I wrote a “name that tune” interface for the MIDI system. The `ntt` program took a melody played at the keyboard (*i.e.*, input from MIDI) and found possible matches in a database of musical themes. Melody lookup was straightforward: the input MIDI events were filtered so as to remove very short notes, like ornaments or glitches; the relative pitch interval sequence was taken as the sequence of pitch changes in half steps (which, for a melody like *There’s a Place for Us*, **G4 F5 E5**, is [11, -1], that is, a rising major 7th followed by a descending half-step); and this was then mapped around ‘O’ in the middle of the ASCII

table to obtain a printable string (in this case, 'ZN'). A system search command, like Unix *grep*, could then be invoked to find the melody string in the known list. This was evidently the first interactive melodic dictionary, although recently commercial products have begun to appear in this vein.

One idea implicit here, and in Barlow and Morgenstern, is that relative pitch is generally a more salient feature than rhythm. This relates to the combinatorial fact that a melody of  $N$  notes can contain  $12^N$  pitch strings if leaps are restricted to an octave. Rhythm provides less variation (it is often roughly binary, so the exponential base is small), hence less information. Psychological studies of human memory support this. For example, Dewitt (1986) studied factors of rhythm and pitch, and concluded that pitch contour is the more important feature. Sloboda (1985) studied the memory of a musical *idiot-savant*, an individual who could play back a piano piece after only 2 or 3 hearings. He observed that gross and highly structural substitution errors occurred later: for example, the *savant* subject mistakenly folded the rhythm from one phrase onto the pitches of another. All of this also tends to favor pitch as the more persistent feature, and thus, although there are other ways to do melodic string matching, there is some basis for this method.

The program worked fine but progressed no further, for two main reasons. First, recognizing melodies is productive when there is a large dictionary on hand but annoying when there isn't (we didn't want to keystroke all of Barlow and Morgenstern or the ASCAP collection), and second, the need to use it at the moment is fairly infrequent. Of course, it is not hard to imagine a music system that lets the user hum a tune to retrieve songs; in fact, if the song library were large, a "hum a few bars" interface might often be preferred.

It would be interesting to make a more thorough study of melodies along these lines — for example, attempting to create a similarity metric for melodies so that labels like "French" or



a name-that-tune interface

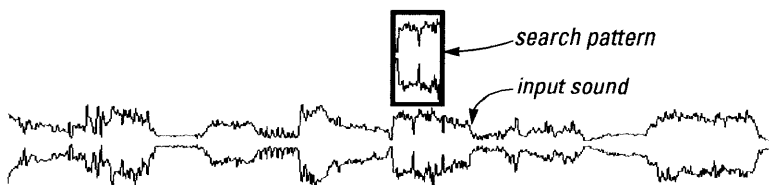
“early 19th century” could be applied, or attempting to find melodies in polyphonic MIDI data. To my knowledge, that has not yet been done.

### **Name That Tune from polyphonic audio**

Now, suppose we want to extend the smart radio mentioned earlier so that it not only recognizes music, but recognizes particular musical tags, like theme songs. Is there some way to do melodic recognition from a complex audio signal?

The natural approach seems clear: if we can find the music, and extract the pitches, and then if we can locate the melody-bearing line, we can pass the result into a melodic dictionary as just described. There are two main difficulties with this. First, for the case of complex musical textures, a robust polyphonic pitch extractor does not yet exist, and second, determining which line is the melody may not be easy, particularly in an orchestral situation. As it turns out, though, there are many useful applications for which neither of these restrictions is severe.

One way to find specific pieces of music in an input signal is by brute force — sampling the melody to make a search pattern, then correlating it against the input music stream:



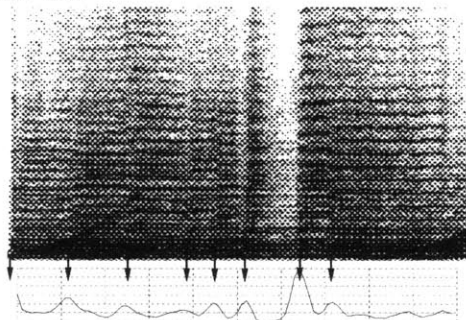
This will work, and simple, literal string searching (or matched filtering) is not necessarily a bad approach. As with other forms of string searching, there are many ways to improve performance. One would of course perform the correlation in magnitude to avoid distortions due to phase changes; and, the signals can be considerably compressed to reduce computation,

and it is sufficient to probe for the search pattern at significant attack-points in the signal. So, for example, a system could operate on greatly subsampled audio streams, and a note-detector (the one-note case of the music filter already mentioned) could skip along in the input stream to invoke further testing as necessary. Because much of the musical content is conveyed in a few peaks, the frequency data can be compressed by a factor of 100 or so before matching. One could imagine using this method to find specific sound segments, or equivalently, in a video stream to spot particular key frames. It is a practical search technique and doubtless will find many applications. There are obvious drawbacks, however. This method is sensitive to changes in pitch, tempo, instrumentation, and so forth; it may have difficulty if there is a voice-over or other sound in the mix; and it requires a search sample that is already known.

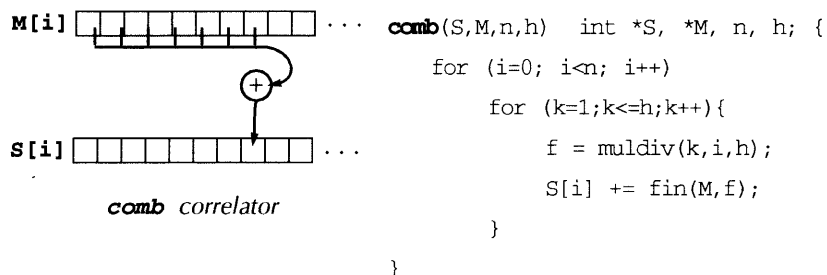
What we would like to do, of course, is perform the search in a transform domain that escapes these problems. Although we cannot quite extract an arbitrary melody, we can get close enough to perform an effective match, and fortunately, in a great number of musical situations, the composer makes the melody easy to spot: it is usually the loudest, highest line in the music. Consider the following tune:



This is the theme from *I Love Lucy*; the spectrum of that melody sample looks like this:



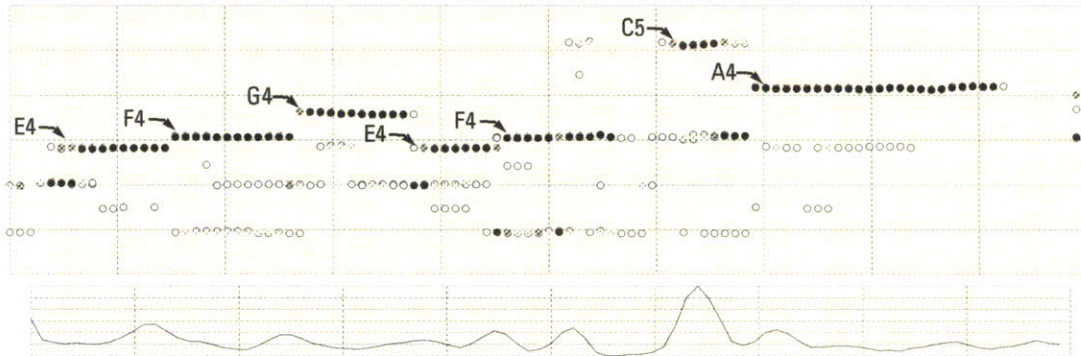
The syncopated “rest” is easy to see (it is the vertical white band in the spectrum). The upward-sweeping harp glissando and note changes are faint but can be made out; bongos and the bass line are very difficult to see, but can be heard. Drawn below the spectrum is the output of an attack detector (the positive-going difference in the magnitude spectrum,  $d[i] = M[i]_{new} - M[i]_{old}$  or zero when  $M[i]_{new} < M[i]_{old}$ ) is shown below. At each attack point we wish to locate the most prominent note. The method used is a correlation made by summing a combed sum of  $h$  harmonics stretching over the samples of the magnitude spectrum,  $M$ , namely:



where  $muldiv(k, i, h) = k*i/h$  in double-precision, and  $fin(array, i)$  is the linearly interpolated value of  $array[i]$  where  $i$  is a floating-point index. This was found to be preferable to a cepstrum because, unlike speech which generally exhibits a dozen or more well-defined harmonics and consequently shows a sharp peak in the cepstral “quefreny” response, musical harmonics, particularly in old recordings, are often fuzzy and fewer in number, and the cepstrum is noisy. For the same reason, the piano pitch separation method no longer works because one cannot assume that the first or second harmonic of a note is the strongest one, and because the “peaks” are often ill-formed. Additionally, this comb is convenient to apply within a narrow band of frequencies, and the resulting frequencies are interpolated typically to within about 2Hz (that is, if samples in  $M$  are spaced by  $f$  Hz, then samples in  $S$  are spaced by  $f/h$  Hz, where  $h$  is the number of harmonics in the comb). Initially, the result of combing in this way is a set of



possible pitches; the following plot shows the four best choices for each input magnitude frame, with the likeliest pitch in black:



Collecting the horizontal runs into notes and favoring the strongest note at each attack produces this:



Once all the “votes” are tallied, the pitch extraction does indeed tend to collect the melody-bearing notes. Notice that one note was missed: the pitches as found are: **E4 F4 G4 E4 F4 (C5) C5 A4**, which corresponds to an intervallic string of **[1, 2, -3, 1, 7, (0), -3]**:



The missing note won't matter when doing literal searching provided the same pitch extraction filter is used on the input, and that the matching software is tolerant of some variance.

At this point, having obtained some representation of the primary pitch content, the matching may proceed in a number of ways. For very fast matching, a string search on the relative pitch string will do just what we expect: it will find the prominent pitch sequences and is fairly resilient to transpositions and tempo changes. In this way, the transformed input stream can be efficiently compared with a set of melodic templates. This method is explored further in chapter 4. A more thorough approach that attempts to avoid the problem of ambiguous melody lines might check for a melodic pattern beginning at each possible new note in the input; this might have to be used in listening to “fugues,” for example, in order to find lines that are buried in the counterpoint. That has not been tried, although it seems straightforward.

## 2.6 Remarks

---

The buildup of musical tools leads to a certain synthetic extreme. One would say “leading steadily” were it not for the boom of information technology which, practically and potentially, puts so much musical information in such a malleable form. This makes a remarkable contrast with the historical marriage of music and technology to date. That is easy to see in the “thought experiment” that shows that years of music could fit on today’s 600-megabyte compact disk, but concrete examples of savant-like instruments and experiments in synthesizing both musical content and signal were cited.

*“In music, the sensations of tone are the material of the art.”*

— Hermann Helmholtz

To make intelligent use of that musical information, machines need to listen in many ways. If the information is concentrated, like MIDI data is, that can involve statistical sensors to guess the key, find the rhythm, recognize the melody, and so on. That is an abstract and powerful form of listening, since it is attached to such a dense form of content, and quickly provided the basis for Rowe’s approach to interactive music systems (1993). But some basic questions were also asked. Forgetting about guessing

composers, idioms, and affects, one of the most basic questions of all is, how can a machine recognize that a sound might be music? How can this be implemented today? How can that ability find a useful place in the larger repertoire of listening processes? The simple solution (measuring the stability of harmonics) not only worked well, but in the context of listening to radio shows, demonstrates how useful high-level structure can be gleaned. Without recognizing theme songs or lyrics, but simply knowing that music is playing, one can often do a surprisingly good job of recognizing which radio show is on, and what segment (headlines, features, etc) is playing, provided the format of the show is distinctive and well-known.

Beyond that, of course, we naturally do want to know what the musical content is — what notes are being played, what instruments are being used, what composer wrote them, and so on. Study of the piano transcription problem showed that that problem appears to be largely solvable, if not quite solved. Some improvements are needed to make it work — namely, much as speech understanding improves when one anticipates the syntax and semantics of the language, music deconvolution needs to be improved with more knowledge of musical idioms (a sense of tonality, melody, harmony, and special-case figures like octaves and rolled chords). In any case, further research should consider not only making the piano gestural deconvolver work, but ought to seek to derive a fuller *model*: the analytic process should yield not just the gesture, but a sampled piano and perhaps some sense of the room. That would be an exciting and practical direction for future music recording systems to pursue.

Having found the notes, one is in a position to “name that tune.” Examples from MIDI processing showed how melody recognition is analogous to dictionary word lookup. Indeed, the **ntt** program showed how a relative interval sequence (like [0,0,-4] for the opening theme of Beethoven’s Fifth) makes a simple and effective search key. But what about the problem of searching from the musical audio signal? That is quite a bit

trickier since so much interesting music is polyphonic, polyinstrumental, and may contain other noise besides. Thus what constitutes the melody line in a complex musical texture is not always clear. But some assumptions often fit with reality. A pitch extraction process that attempts to trace that loudest upper line will often find the useful melody string. This yields a search string that has the happy property of being independent of key and tempo, so that other transpositions of the melody can be found later. In fact, this method is not so kludgy as it seems, and is intuitively reasonable.

Overall the desire has been to show how we might couple low-level audio sensors to high-level processes that leverage the dense musical content. Although we are still not adept at teasing that information out of musical sound a number of interesting operations can be implemented and applied. All of this can be readily seen in the domain of music, for which so many sorts of sound-structuring rules, and content-oriented manipulative technologies already exists. Surprisingly, that is much less true of speech, which is the topic of the next chapter.

# 3: Listening to Speech

---

## 3.0 Overview

---

This chapter briefly considers techniques for location and segmentation of speech in a soundstream, speaker identification, and the novel possibility of “supersynthesis” of speech from large collections of recorded utterances. The immediate purpose is to develop a few more sound-content filters to add to our repertoire; a longer-term goal is to explore some of the things that might be done as substantial archives of digital speech become available. For instance, it is worth noting that although problems like talker identification have been extensively studied, they have not yet been studied in the context of indexing long conversations, like radio talk shows or movie dialogs, to find particular personalities. It seems certain that the nature of future digital information and entertainment systems will make such capabilities a necessity.

In particular, **3.1** roughs out the requirements for mass speech storage, making it clear that just as global transmission of speech became commonplace in this century, the casual storage, searching, and reuse of large amounts of speech may become common in the next. Section **3.2** considers speech identification and segmentation, and presents a filter akin to the music detector of chapter 2, but for locating speech events. It also discusses a spectral sound editor, called *Spectre*, that has proven to be a useful tool in this work. Section **3.3** considers methods for

identifying the talker, or at least approximately segregating utterances based on talker, and section 3.4 looks at the intriguing opportunities for synthesis of speech by assembling sampled words and phrases into whole utterances.

### 3.1 Sizing up Speech

---

There are many things we listen to and listen for in speech — not simply the textual content of the message, of course, but also the ways in which it was said: for example, who spoke (age, gender, identity), or what expression, intonation, or emotional affect was used. We can tell in an instant whether the voice is happy, angry, or sad, whether it belongs to a little girl or a big man, to someone who is sick, tired, drunk, or, by hearing the trace of an accent, whether the speaker grew up in Yonkers or Houston. Qualities like these are all natural indices to envision using to browse or manipulate large amounts of speech.

A somewhat startling fact about spoken information is how little of it there actually is. As with music, we expect that speech accumulates at a finite rate, but the coding of large amounts of speech, and consequently how much of it might be retained in the memory of machines or appliances, is a little more difficult to estimate than the keystroking of a piano because the “instrument” is not so easily separated from the “gesture.”

Supposing that an individual speaks for about 4 full hours in every 24 hour day (after compressing away the silences), we might assume roughly 1500 hours of speech per person per year. At a coding rate of 8,000 bytes per second, that is about 40 gigabytes per person per year — still a lot by today’s standards although it certainly won’t seem so forever. Fortunately the redundancies in speech make it compressible. Honda and Itakura (1992) evaluated a “split-band” adaptive-predictive speech coder at low to medium bit rates (from 4K to 16K bits per second); in general, for “toll-quality” telephony, coding rates down to about 1000 bits per second are cited as a

tenable lower bound (Rabiner, 1993; *cf.* fig 2.2), which puts the size at about 670 megabytes per person-year of speech. Of course, such coders use relatively small codebooks and are designed for purposes of voice transmission, like telephony, and that is rather different than coding for content, or coding to facilitate the manipulation of expressive or deep structural qualities. The textual content of a year of speech would be something closer to 50 megabytes per person (it takes about a minute to read two hundred words out loud) but as is well-known, people's speaking vocabularies are relatively fixed and quite small compared to writing or reading vocabularies — though just *how* small is not easy to estimate and likely to be highly dependent on the individual. One could imagine that a codebook of several thousand entries for a talker's commonly-used words and phrases could be employed in conjunction with information to abstract inflection. That would achieve a dense compression.

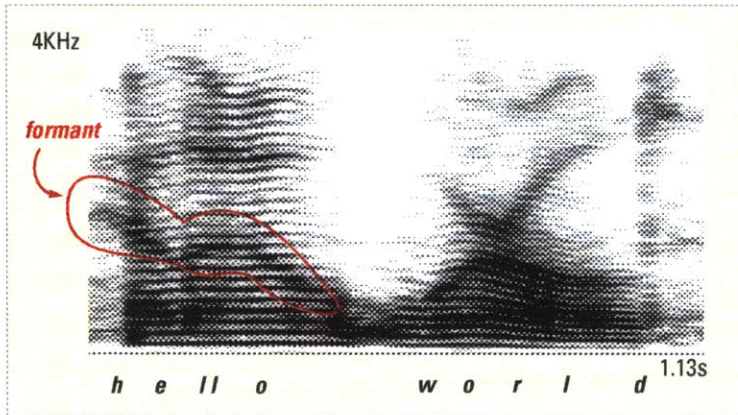
In any case, even ignoring what are likely to be considerable savings to be had due to long-term redundancies in individual vocabularies and articulatory habits, it seems safe to assume that a person-year of speech at modest code rates is about a gigabyte, which could be reduced another ten-fold by Lempel-Ziv-like methods. Given the rate at which memory capacity of personal information appliances is increasing, the idea of a medium that records a lifetime of spoken output may not be so far-fetched. Of course, all this is speculative since nobody has ever attempted to code substantial amounts of speech in this way, and no application has really demanded it yet. Nevertheless, considering the effect that massive, pervasive transmission of speech has had in the past hundred years, as well as trends in digital media technology, it would be wise to contemplate ways of working with large amounts of recorded speech. If nothing else, speech will accompany televised information, which we will certainly wish to index by spoken content.

**speech:**

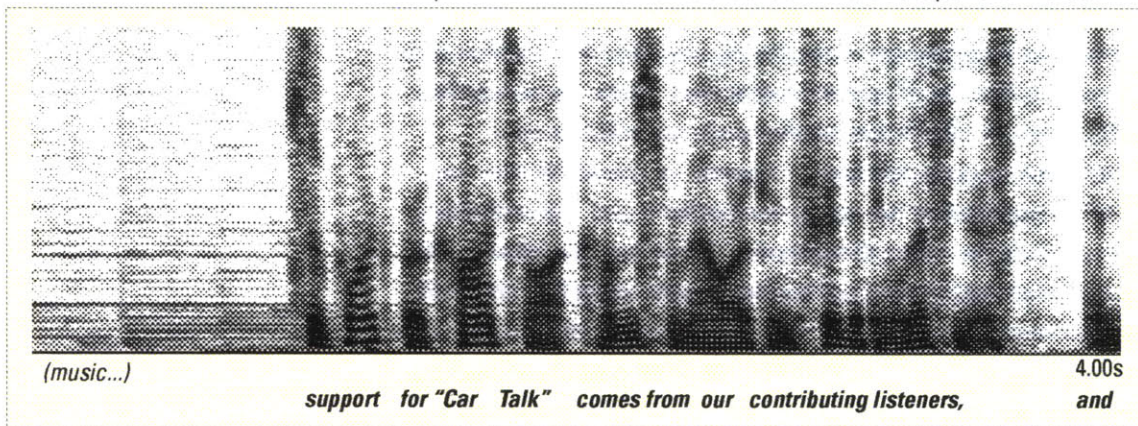
*1 gigabyte per person-year*  
*70 gigabytes per lifetime*

### 3.2 Finding and Segmenting Speech

As with music, before understanding the content, like words or notes, we would like to simply locate segments of speech. Just as a musical spectrogram reveals certain distinctive characteristics, so too does a speech spectrogram. The most apparent features are the harmonics of the vocal cords (the thin horizontally-oriented stripes); the darker, thicker bands imposed on them are the *formants* that shape the vowels:



Of course if the speech is unvoiced (like whispering) one will see formants without vocal cord harmonics. In longer phrases, there are pauses and consonants that look like short puffs of noise:



There are several ways one might devise a filter to detect speech. If the problem is just to distinguish speech from background noise, a thresholded measure of average magnitude,



possibly with some adaptive ability, will often suffice (Lamel 1981). The presence of speech can also often be found by considering zero-crossing rates (ZCR), linear predictive coding (LPC) coefficients, and autocorrelation measures to detect periodic energy. Usually these are combined with subsequent heuristics to fill in short gaps or remove isolated “islands” so that a sensible grouping of utterances is found. Techniques like these have been reviewed and applied most recently by Hindus (1992) in her work on capture and display of telephone conversations, and by Arons (1993) in his work on speech browsing interfaces.

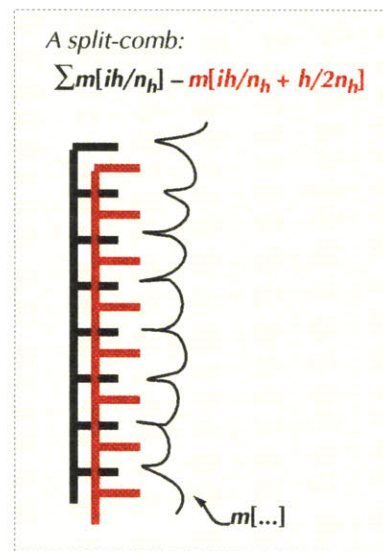
### A split-comb to locate voiced speech

A slightly different segmentation method is used here. Because we expect background music and sounds of many sorts, because we wish to measure pitch accurately, and because we may already have done the work to obtain short-time spectra, a comb filter is used to sense harmonic energy. In particular, when speech is present we expect to see several harmonics pitched within a speech-like band, so the detector stretches a comb of  $n_h$  harmonics over a range of frequency samples. When each harmonic settles on or very near a magnitude peak the comb output will be maximized and the pitch of the speech will be the frequency of the topmost harmonic in the comb divided by  $n_h$ . However, a simple comb will also respond to broadband noise and may mistakenly collect energy pitched at multiples of the correct frequency. To prevent this a “split comb” is used. This simply involves subtracting the same comb shifted up in frequency by half the pitch:

$$c[i] = \sum m[ih/n_h] - m[ih/n_h + h/2n_h]$$

where

- $m$  = log-magnitude frequency spectrum
- $h$  = harmonic index in comb (from 1 to  $n_h$ )
- $n_h$  = number of harmonics in comb
- $i$  = magnitude index of topmost harmonic  
(at frequency  $i \cdot \text{SampleRate}/\text{FrameSize}$ )



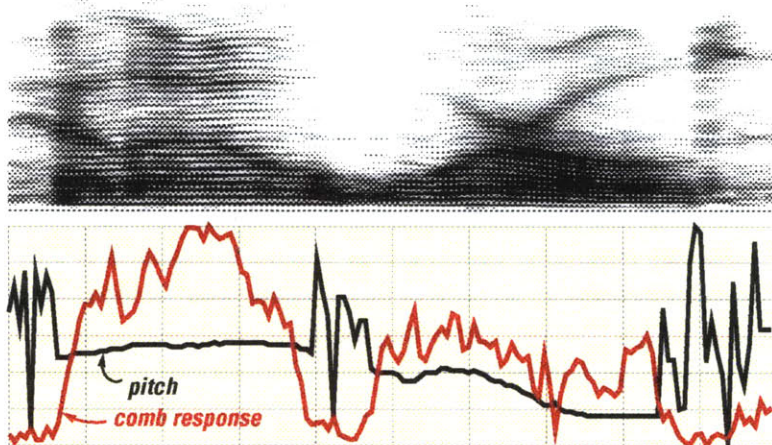
The index  $i$  is run over the range of frequencies of interest (so that the comb can be implemented to efficiently follow in the vicinity of a known frequency). In this way, when the positive comb settles on the peaks and the negative comb settles in the valleys between them, an emphatic maximum is found and the pitch of the fundamental is simply  $i \cdot \text{SampleRate}/\text{FrameSize}$ . Since the teeth of the comb generally do not all fall at integral multiples in the frequency sample array it is often desirable to interpolate between samples, replacing references to

`m[i]`

by an interpolating index such as

```
#define fin(d, f) (((1.0-((f)-(int)(f)))*(d)[(int)(f) ]) +\
                 ((f)-(int)(f)) * (d)[(int)(f+1)])
fin(m, i)
```

This works quite nicely for detecting presence and pitch of voiced speech. For example, running the comb over the “hello, world” spectrum above produces this output:



The red trace indicates the response of the comb; it rises sharply when new voiced speech and stressed syllables occur. During these time segments, the pitch output (in black) is valid. For this utterance, the pitch ranges from 86 to 135 Hz.

### The speech event detector

Speech can be readily distinguished from stationary-pitch information like music by filtering for the frequently-interrupted and frequency-varying pitches that are characteristic of speech. This then gives us the information we need to locate voiced segments of speech, including syllables and stressed utterances if necessary, and additionally, to accurately determine pitch contours. Thus, we have another filter to add to our kit:

```
speech file.snd
file.snd| 10.82| 2.64| speech
. . .
```

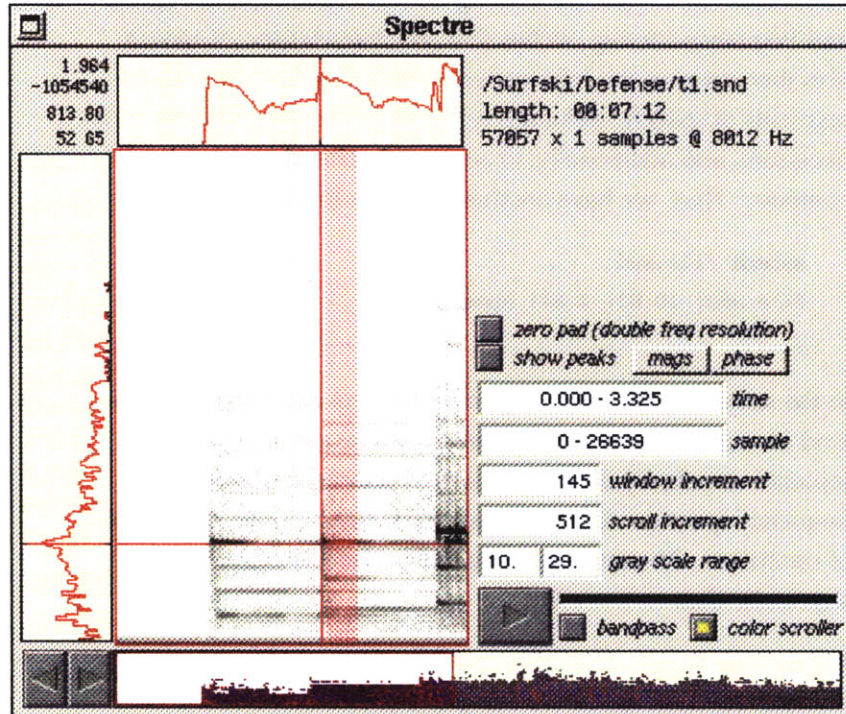
In the same way that the music detector was adapted to make a kind of music separator, the speech detector can be used to separate vowels, by using the comb to select samples in the frequency spectrum for resynthesis, or simply to modulate a bank of oscillators. The method of Macaulay and Quatieri (1986) is essentially that.

There are certainly many other ways to derive similar measurements for speech. For instance, the Rabiner-Gold algorithm (described in Rabiner (1975), p. 681) uses a bank of pitch period estimators to detect coincident harmonics, and this can also easily be adapted to do voiced-unvoiced detection. Hess (1983) presents a 700-page monograph on the subject of pitch alone. The problem here is somewhat different in that we need to locate speech in what may be a rather noisy soundtrack containing interesting background information. In addition, we will want to perform other sorts of tasks, like speaker identification, which will be derived from measurements of the speech spectrum. For these reasons, combing for vowels has proven useful.

### ***Spectre*: a spectral sound editor**

Techniques like these were developed much more rapidly using a spectrally-oriented sound editor, called *Spectre*, written for this

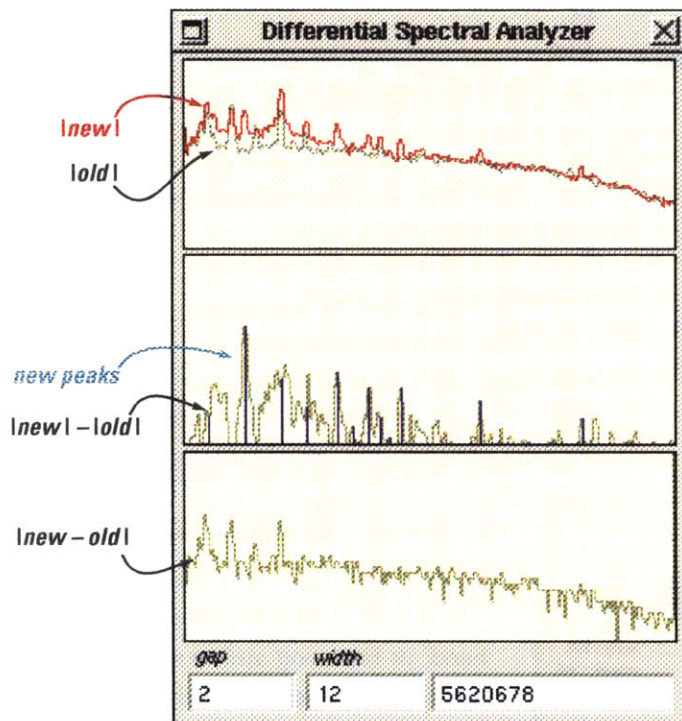
purpose. This displays sound spectra with options for zooming, scaling, peak-finding, and other such functions. A typical display looks like this:



The crosshairs over the main spectral view select time and frequency slices of magnitude data to be displayed in the adjoining red line graphs; the lower horizontal view (in red and blue), essentially a scroller, shows the full spectrum of the file (with red indicating increasing energy, blue indicating decreasing energy, and the clear portion of the scroller visible in the main spectral view). In this way it is easy to inspect values, adjust frame “hop” sizes, and play back portions of the data. For instance, one can play bands of frequencies by setting the “bandpass” switch and selecting a block of data: the result is synthesized by the *anasyn* program, mentioned in section 2.4.

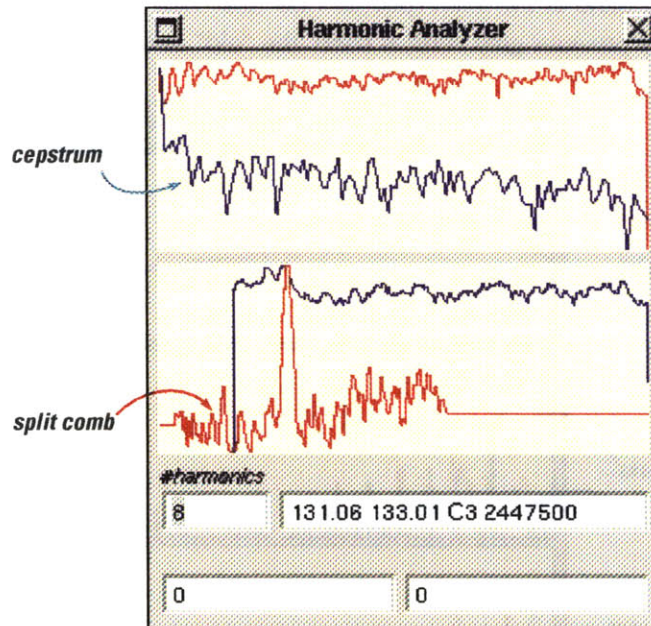
In addition to these basic functions, other inspectors have been added as needed to provide other analytic views. This *Differential* view shows the magnitude change between the red

vertical band of the main spectral view (the “new” information) and the previous grey vertical band (the “old” information):



The upper view shows new and old frequency samples (in red and grey, respectively); the middle view shows their difference (with blue segments to indicate the novel peaks); and the lower view shows the phase-correct difference. This is especially convenient for inspecting things like subtle note onsets in music. One maneuvers the crosshairs with the mouse while watching the differential harmonic changes at the same time.

In the following figure, the *Harmonic Analyzer* shows cepstral, split-comb, peak-comb, and possibly other data views for a selected timeslice. This also illustrates why the split-comb is often preferable to a cepstrum for tracking pitch: as noted in 2.5, even when the cepstrum exhibits no distinct peak, the split-comb response shows a strong one, and the pitch has already been interpolated typically to within one or two hertz.



start	end	source	length
487.33	35.00	connery + ford []	
668.85	20.19	doody + ford	
1541.68	20.00	connery + ford	
0.00	0.00		
26.70	20.00		
950.73	17.80	doody + donovan	
1239.80	17.50	(donovan)+ford	
1273.78	16.00	ford	
1370.73	35.00	connery + ford	
1462.27	15.00	ford	
1480.50	45.00	[fugal]	
874.833333	7.05		

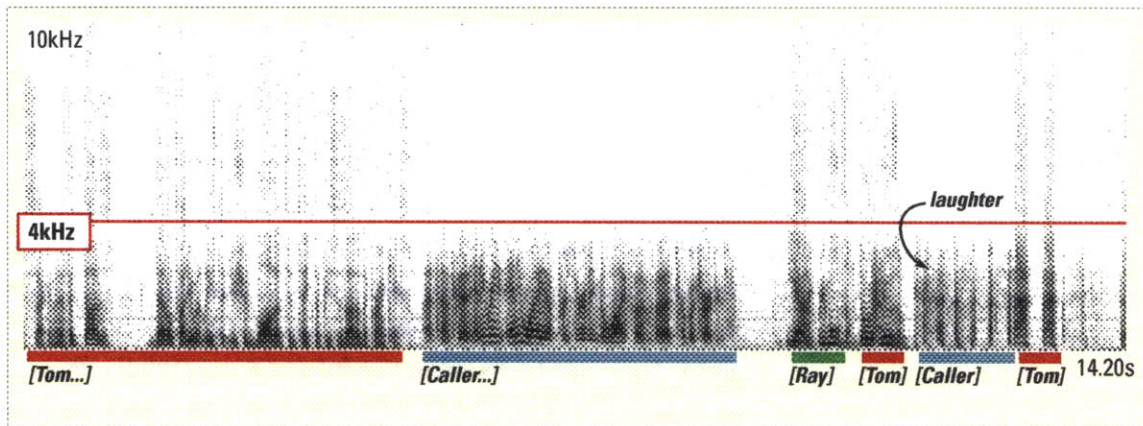
event list editor

Finally, *Spectre* incorporates an *event list editor* to facilitate logging soundfiles by hand, or browsing and touching up event lists that were generated elsewhere.

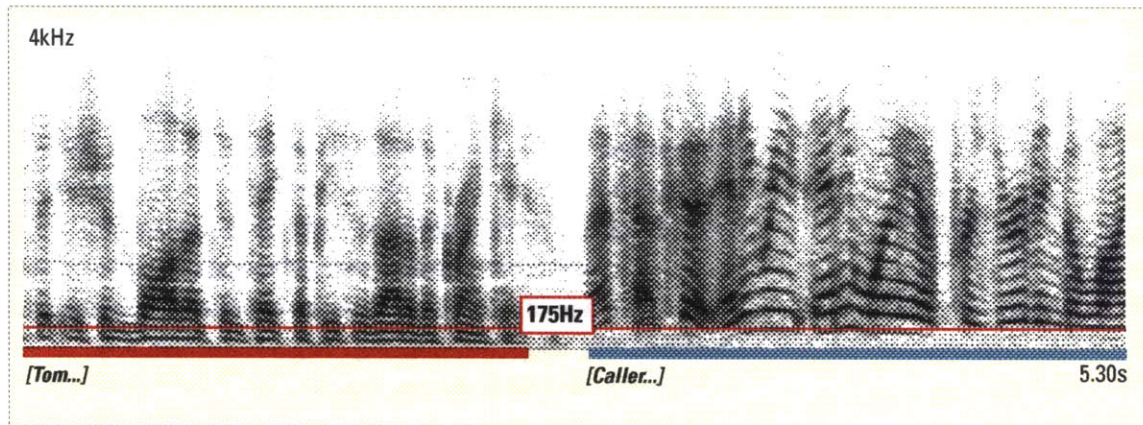
For example, speech-to-text systems can currently produce only rough event lists to annotate a sound file; using *Spectre*, it is easy to precisely spot the locations of consonants and vowels so that endpoints of words can be accurately adjusted.

### 3.3 Identifying Talkers

Having located utterances we would like to segregate them according to who is speaking. This may occasionally be done using gross spectral differences; for example, to automatically find the callers in a phone-in radio talk show like *Car Talk*, it is easy enough to find speech that is band-limited to about 4KHz, as this spectrum shows:



It is also the case that the “telephone” spectrum on radio is generally weak in low frequencies (it tends to have very little energy below about 175Hz) which gives it a characteristically muffled sound:



Indeed, the gap in the lower frequency band has proven the more reliable discriminator, and let us just note the utility of this by mentioning that the filter *find-caller* writes out an event list of speech segments from phone-in voices:

```
find-caller cartalk.snd
cartalk.snd| 4.17| 5.23| phone-speech
. . .
```

The command **eplay** plays back event lists (with a number of options to control concatenation, mixing, etc), so one can skim the radio show to play only the phone-in comments in this way:

```
find-caller cartalk.snd | eplay
```

In most cases, of course, the problem of determining speaker identity is considerably more complex than looking for a band-limited voice. Speaker identification has been studied for several decades. Comprehensive summary articles by A. E. Rosenberg and S. Furui can be found in Sondhi and Furui (1992); others can be found in Parsons (1986), Saito (1992), or most recent texts on speech science.

### **Typical approaches to speaker recognition**

Briefly, the problem of speaker recognition is usually divided into *verification* (accepting or rejecting the claimed identity of a speaker typically by comparing pronunciation of a password with a known key; cf. Jayant, 1990); and *identification* (determining which speaker among a known set most closely matches an input voice). In many cases, the procedures for doing this are so well-established and robust that systems are in some ways superior to humans; the technology is expected to be applied in place of other signatures in numerous secured-access situations, like telephone banking. Algorithms naturally concentrate on vowels, for the steady-state resonance of the vocal tract and formation of vowels captures much of the speaker's vocal physiology and performance (Johnson 1990).

A typical algorithm, like the one described by Furui (1992), extracts a number of statistical features (mean, standard deviation, cross correlation, and Fourier coefficients derived from linear predictive filter parameters and fundamental frequency) and performs a correlation to determine the speaker. Furui showed error rates (false acceptance and false rejection) of about 2% for a task involving identification of an unknown speaker from a known population of 34 talkers, both over the phone and

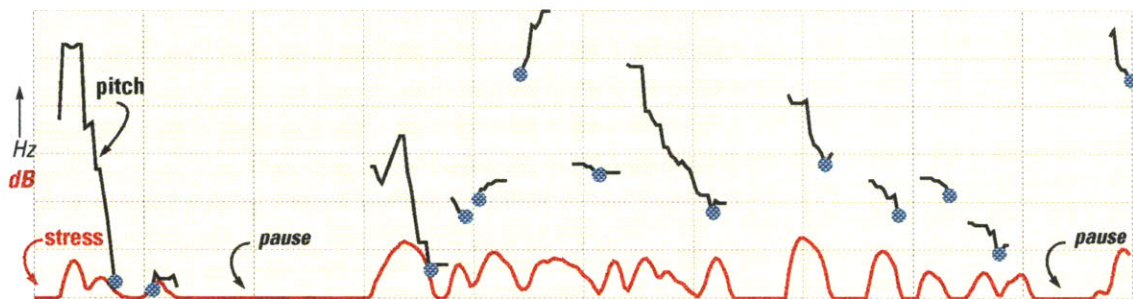


in the laboratory. Using “high-quality” audio and a fixed password, the reliability rate was 100% for a pool of nine speakers. An algorithm can also look for specific vowels, and Fakotakis *et al.* (1993) used a “vowel spotting” technique for identifying speakers independent of spoken text.

In any case, there are many robust techniques to pick from. For the purposes of this work, we need only segregate utterances by deciding from among a small number of speakers (perhaps half a dozen in a given movie), which is a simple form of the identification problem. As it turns out, many of the alternations in speaker can be detected by considering the change in the average baseline pitch of  $f_0$ , which can be conveniently implemented as follows.

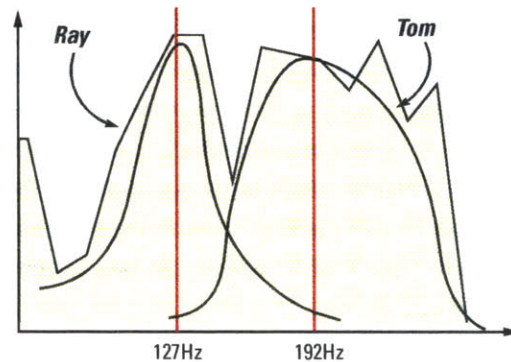
### Average baseline pitch in determining speaker identity

The speech segmenter, *speech*, is modified slightly to append to each utterance the statistics of fundamental pitch. In particular, each utterance consists of a number of pitched (voiced) segments, and the “baseline” pitch is taken to be the low point of each segment. For example, if an utterance has the following pitch contour:



the average baseline pitch is the average of the minima of the frequency of the voiced segments (the points shown in blue; a similar statistic, instead of collecting baseline pitch, is to simply sample the pitch whenever the stress contour is a local maximum). This approximately reflects the “natural” frequency

of the voice. Such a measure is adequate for most male/female adult/child distinctions, but it has proven to work well for segregating among a few talkers in general, like actors in a film. For example, in *Car Talk*, the two hosts, Tom and Ray Magliozzi, speak at slightly different average baseline pitch:



Tom, who has the higher-pitched voice, also tends to yell a lot more, so the upper mode of the histogram is wider.

The output of the `speech` filter now looks like this:

```
speech cartalk.snd
cartalk.snd| 4.17| 5.23| phone-speech| f0:158 175.8 140.6
. . .
```

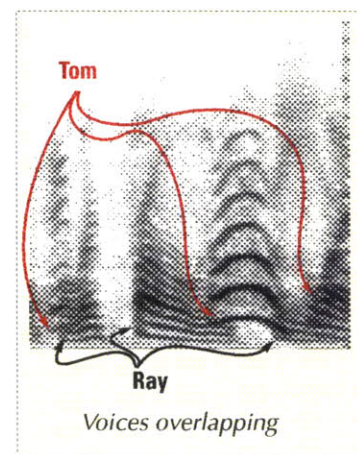
indicating that in the given utterance,  $f_0$  consisted of two segments with minimum pitch 175.8 Hz and 140.6 Hz, the average of which was 158 Hz. It is straightforward to organize the utterances by considering the distribution of baseline pitch. This then lets us build a postfilter for the speech segmenter to assign voices to utterances:

```
speech cartalk.snd | speaker -f 127:"Ray" -f 192:"Tom"
cartalk.snd| .45| 3.32| speech| Ray
cartalk.snd|4.25| 2.57| speech| Tom
. . .
```

We know this is not an especially accurate method for speaker identification, and there are certainly many other techniques in

the literature that could be used. Nevertheless it works well enough for many applications, and in particular, for distinguishing a small number of actors in a movie soundtrack. What is somewhat more important is that the speech and its pitch are fairly accurately found in the presence of other noise; after that, the work on speaker identification begins.

Another problem that should be mentioned is overlap of voices. When two voices overlap in what appears to be a single utterance, a more thorough pitch analysis usually reveals two distinct pitch paths. This is the basis of typical speaker separation techniques, like that of Parsons (1978). Here, a second stage in speech segmentation could consider each candidate utterance more closely to determine whether or not it contains overlapping voices. In natural conversation, voices overlap all the time; for instance, the two hosts of *Car Talk* constantly interrupt one another, and squelch their callers. A more realistic segmenter would take this into consideration. In slightly less natural situations, like movies, the dialog rarely overlaps, besides which there is frequently other pitched noise in the background that might be confusing to a naive algorithm. For the purpose of the work done here, voice overlap detection has not been implemented.



### 3.4 Supersynthesis

If recorded speech is of consistent quality and contains a sufficient vocabulary, it can be used in high-level synthesis by reassembling whole words and phrases. This can be illustrated with a small example.

The printer in the NeXT computer uses a number of spoken error messages:

*Your printer is out of paper.*

*Your printer is waiting for paper.*

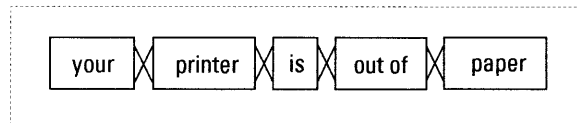
*Paper is jammed in your printer.*

*Your printer cover is open.*

In the NeXT system these utterances are stored as soundfiles to be played as needed. Words may be tagged as events as follows:

```
nopaper.snd | 0.006 | 0.152 | printer lady | your  
| 0.157 | 0.511 | printer lady | printer  
| 0.644 | 0.225 | printer lady | is  
| 0.863 | 0.365 | printer lady | out of  
| 1.216 | 0.483 | printer lady | paper
```

To play back a message, these events can be piped through **eplay** as mentioned, which assembles a sound by joining the segments with micro-fades (of about .015s):



The result is indistinguishable from the original (each word may be positioned in the mix at the same position at which it occurred in the original, or the words may simply be concatenated, as suggested in the example). **eplay** also interprets directives that may be appended as extra fields to the events. For instance,

```
| 0.157 | 0.511 | printer lady | printer | @gain 1.5
```

multiplies the amplitudes by the given gain envelope (in this case, 1.5x). The following directives are recognized:

```
@dur n      truncate segment to n seconds  
@stretch n   stretch segment to n seconds  
@gain n1 n2... multiply the amplitudes by the given envelope  
@rev n      reverse the segment  
@t n        position the segment at time t in the output  
@pan n      pan the segment left or right
```

These are meant to be illustrative, not comprehensive; many more processing functions could be added.

Now, new sentences may be generated by assembling appropriate words:

*"Your printer cover is jammed."*

*printeropen.snd | 0.004 | 0.161 | printer lady | **your**  
| 0.148 | 0.428 | printer lady | **printer**  
| 0.570 | 0.449 | printer lady | **cover**  
| 1.005 | 0.218 | printer lady | **is**  
paperjam.snd | 0.60 | 0.54 | printer lady | **jammed** | @stretch 1.02*

*"Jam is in your paper."*

*paperjam.snd | 0.60 | 0.45 | printer lady | **jam** | @gain 1.5  
printeropen.snd | 1.005 | 0.218 | printer lady | **is**  
paperjam.snd | 1.14 | 0.3 | printer lady | **in your**  
nopaper.snd | 1.216 | 0.483 | printer lady | **paper***

These utterances sound perfectly natural provided they are grammatically correct, and the words join together smoothly.

There are a number of ways in which a good fit between words can be found. By taking care to preserve as much global context as possible, and by beginning with a reasonably homogeneous speech corpus, the overall sentence contour can be preserved; e.g., one can replace objects and verbs in a template sentence:

*"Your **x** is out of **y**."*

When picking a word to fill a slot, in general there may be several performances of it to choose from in the database. There are various criteria one might use in selecting the best fit, including picking the replacement word that occurs in a similar position in the sentence as the slot to be filled, and picking a replacement that occurred in a surrounding context that best matches the phonemes in the destination context. This will help to ensure that the transitions at the head and tail of the replacement are phonetically smooth, and that it fits the sentence intonation. I have not yet automated these functions, but anecdotal experience in assembling sentences by hand is promising.

In addition, once the replacement word or phrase is put in place, just as we cross-fade in amplitude to remove clicks, it would be useful to smooth the fit by taking care to keep the pitch locally smooth, and perhaps also altering the emphasis by changing pitch and amplitude. Intonational structure is intricate but the work of Silverman (1987) on fundamental frequency contours and Pierrehumbert (1981) on intonation, as well as Cahn's work on synthetic affect (1990) all indicate that intonation can be analyzed and synthesized. In the examples above, for instance, certain words were emphasized by stretching or boosting amplitude slightly when putting them into sentence-initial and sentence-final positions. At the moment, **eplay** lacks the analysis and resynthesis functions required to facilitate proper pitch and time scaling (e.g., pitch shifting without affecting formant positions or consonants); nor can it make more meaningful intonational adjustments.

By contrast, previous techniques for synthesizing speech have generally focused at lower levels. Formant synthesis from text has been popular for years (Rabiner 1967), and the work of Klatt and others is well-known through DECtalk and MITalk (cf. Allen, 1987). The general difficulty with these systems is that overall sentence intonation and rhythm is awkward, and vocal timbre is unnatural. The timbral problem can be overcome by using sampled data (for instance, Hauptman (1993) at CMU has been experimenting with the application of the Sphinx speech recognition system for the automatic assembly of triphone databases specifically for use in rule-based synthesis). However, triphone samples can be difficult to assemble cleanly, and the overall contour of the speech produced is still not convincing. By using large speech corpora as suggested here, one can rearrange whole sentences so that the overall affect remains natural. This then leaves the problem of what to do about missing words; probably they would be cobbled together by a phonemic synthesizer.

### 3.5 Remarks

---

The problem of finding speech events has been considered very briefly, mainly because it is a large topic about which a great deal is already known. The goal here was to implement a few techniques relevant to our event filtering framework so that at least some useful information can be gotten out of the speech parts of soundtracks. To that end, a number of event filters were introduced, including **speech** (a speech segmenter), **speaker** (to segregate utterances by speaker), and **eplay** (an event list playback utility), as well as **Spectre** (a general application for investigating audio spectra).

One could easily imagine that a future speech transcription system would deal with all the issues discussed here, and others besides, but even in that scenario the value of the simple event list approach and composable parts used here should be easy to see. For instance, in movie-like applications, it is common to want to rearrange dialog, for purposes of synchronization, replacement, enhancement or muting of the emotional content, and so on. This will probably be a common operation in future entertainment systems as well. When the structure in the soundtrack is well-annotated the parts can be easily rearranged. Although the conventional wisdom is that the “Frankenstein” approach to synthesis from a large speech database won’t work, this discussion argued that not only is storage of large amounts of speech quite feasible, and even imminent, but with a few operations sensitive to grammatical, phonetic, and articulatory continuity, the resulting speech sounds natural. In fact, even without second-order operations for smart splicing of speech, it is often possible to put together phrases quite convincingly. This is therefore a promising area for future research.





# 4: *Looking for the Holy Grail:* Movie Scene and Sound Analysis

---

## 4.0 Overview

Feature films and television shows are made to be taken apart: plot, dialog, scene and soundtracks are composed to help viewers follow characters through a story. Often the parts are used deliberately to push emotional “hot buttons.” Compared to the problem of sensing sound in the natural world, movies are highly artificial but nonetheless challenging and illuminating to analyze, because they contain a microcosm of sights and sounds, as realistic or fantastic as one might like. We won’t expect a machine to understand the gags in an Ernie Kovacs episode any time soon — such depth of understanding is as much the “holy grail” of machine intelligence as anything — but on the other hand, with the apparatus assembled so far in this dissertation and a bit of extra picture analysis it ought to be possible to get something interesting out of a big-screen drama.

We recognize the opportunity that structure in the film — the database that *is* the movie — can be transmitted and synthesized in a smart receiver. The receiver could switch off this information and use it for filtering, browsing, abstracting, and reconstructing the film in many ways. The goal of this chapter is not to develop techniques for playing synthetic movies; it is to explore the lower level picture and sound



*“May he who illuminated this...  
illuminate me.”*

*(from **Indiana Jones and the  
Last Crusade**)*

contents to find ways of locating important parts. Of course, the accessibility of those parts invites new ways of watching them.

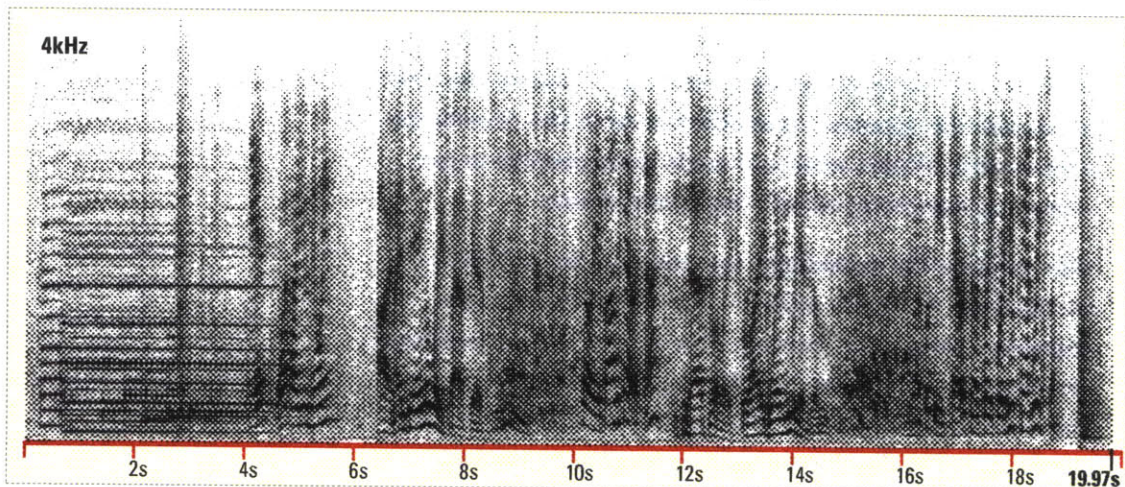
Section 4.1 takes the audio content filters already developed and applies them to a mini-example — taking apart some of the soundtrack to *Mr. Ed*, the 1950's TV sitcom. Section 4.2 presents a movie analysis application that performs real-time image analysis (detecting clips, fades, wipes, pans, etc); this tool makes it possible to index the gross picture elements in a feature film, in effect parsing out the “storyboard.” This is used in section 4.3, which studies an entire movie (the third “Indiana Jones” film: *Indiana Jones and the Last Crusade*, in which Indy (Harrison Ford) and his father (Sean Connery) race against the Nazis to uncover the Holy Grail). This is a classic adventure film, and telltale features can be found in both picture and soundtrack. Of particular interest is John Williams’ musical score which contains a multitude of leitmotivic musical clues that point the way the way to the Grail. Section 4.4 discusses this work overall.

## 4.1 A Close Listen to *Mr. Ed*

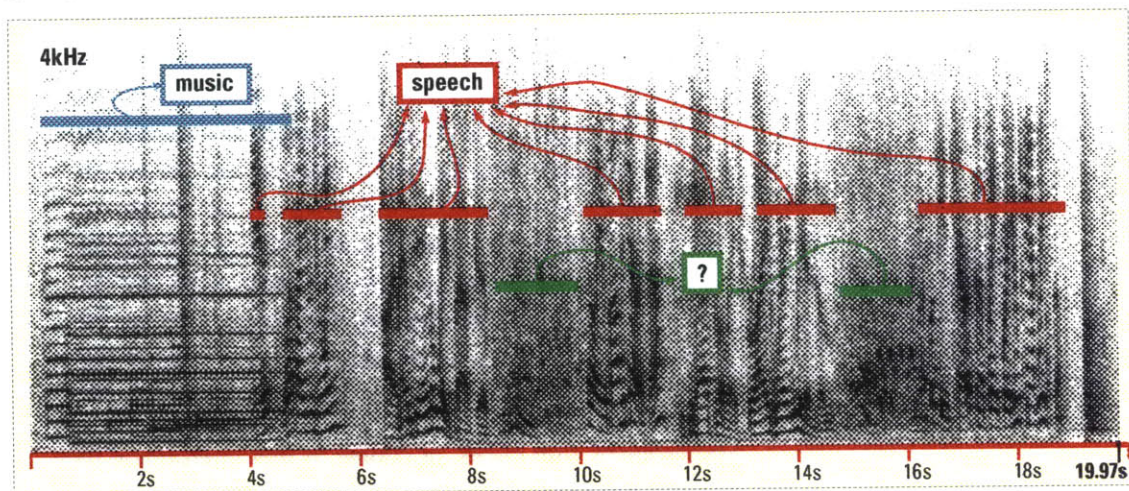
---

### Indexing the elements of a sitcom

Here is 20s of sound from the beginning of a scene in *Mr. Ed*:



By now, we are adept at recognizing various features in the audio spectrum: the lead-in music is easy to spot (through about 4.75s; the vibrato that becomes noticeable around the 15th harmonic may indicate string or orchestral music of some kind); speech occurs intermittently (starting at about 4s); there are some broadband vertical clicks (between 2s and 4s, and at about 19s); and then there is some fuzzy-looking stuff (from 8s – 10s, and 15s – 17s, respectively). The *music* and *speech* filters account for much of this sound:



This leaves gaps at 6s, 8.5s, 11.5s, 13s, 15s, and 19s, of which all but two contain silence (background hiss); the two gaps that are still unaccounted for (shown in green) contain the “fuzzy-looking stuff” that turns out to be laugh tracks. In addition, the broadband clicks have not yet been labelled; we will consider these next.

### Footsteps and doorslams

It is useful to add a click detector to the set of event filters as a building block for finding footsteps and other noises. The program *click* does just that (by looking for abrupt broadband amplitude increases), however when we run it over this segment of sound, we get more than we bargained for:

**click** ed.snd

ed.snd

```
| 00.04|0.20|click|11.88|tilt - 30 26 27
| 02.10|0.20|click|15.40|tilt v 25 22 38
| 02.78|0.20|click|16.67|tilt - 15 15 18
| 03.17|0.20|click|14.15|tilt v 36 30 39
| 03.37|0.20|click|14.27|tilt v 39 34 41
| 05.68|0.20|click|13.14|tilt - 30 25 27
| 05.78|0.20|click|12.34|tilt - 25 25 28
| 06.15|0.20|click|12.10|tilt - 29 31 22
| 06.71|0.20|click|15.45|tilt - 23 33 34
| 12.38|0.20|click|14.97|tilt / 4 9 21
| 12.71|0.20|click|14.24|tilt - 22 24 26
| 14.12|0.20|click|15.01|tilt v 17 13 23
| 16.56|0.20|click|14.20|tilt v 30 16 31
| 16.58|0.20|click|14.86|tilt / 17 24 30
| 16.83|0.20|click|14.49|tilt / 13 17 31
| 17.26|0.20|click|14.90|tilt ^ 29 33 24
| 18.11|0.20|click|14.83|tilt v 15 10 28
| 19.30|0.63|click|16.45|tilt / 20 25 34
```

The problem with clicks, of course, is that interpreting their origin is not easy. The clicks in bold are the ones we will be interested in — the ones that stand out on visual inspection. The remainder are “false” hits, although they are not so easy to dismiss. They might be removed by appropriate thresholding (ignoring clicks fainter than 16dB except when they occur in temporal proximity), but what is an “appropriate” or proper threshold is not easy to reliably ascertain. The first click (at 00.04s) is the onset of sound: this particular file begins with some blank leader, complete silence after which the sound, starting with background noise just before the music, is spliced directly in. That click causes only a faint amplitude change (11.88 dB) and can be dropped for that reason alone.

The remaining clicks are mostly artifacts of speech — hard consonants like **/k,g,t,d,ch/** and a few sharp vowel onsets (so sharp that they are almost glottal stops; remember that actors in old television and radio programs often spoke in pointed,

nasal tones so their speech would carry over the airwaves). Without understanding the speech well enough to label these clicks as part of an utterance (as opposed to part of a footstep that might be mixed in with the speech), they are tricky to reliably separate. One heuristic that works fairly well is to simply ignore all the clicks that co-occur with speech. That is essentially the “eyeball” heuristic of ignoring all but the ones that “stand out,” visually — which is to say, the Gestalt principle of similarity. In this case, that actually does remove all but the boldface clicks we are interested in, but in general it will throw away “good” clicks that are mixed with speech. To avoid throwing out the “good” clicks, apart from understanding the speech, it might be helpful to look for clicks that cut across a vowel (as opposed to clicks that occur at the onset of a vowel, or in the transition between two vowels). It is hypothesized that this could be done by looking at the pitch contour of the vowels (for instance, as obtained by the combing method described earlier), and keeping only the clicks that occur in the midst of a smoothly-varying pitched vowel. That of course is the Gestalt principle of “good continuation” mentioned in 1.3. Brown’s thesis (1992) considered problems like this in more detail; these ideas are not implemented here.

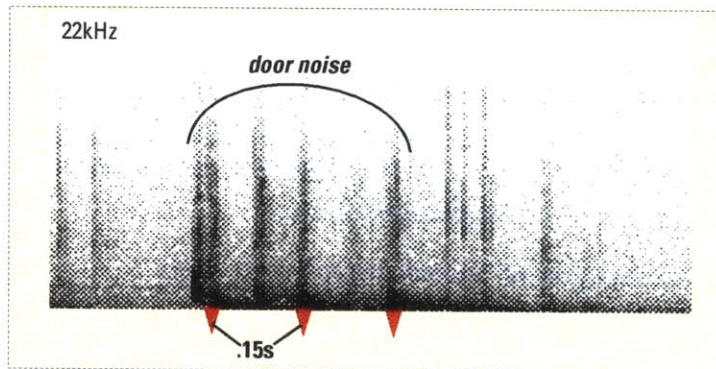
In any case, we are left with the following clicks:

```
ed.snd
| 02.10|0.20|click|15.40|tilt v 25 22 38
| 02.78|0.20|click|16.67|tilt - 15 15 18
| 03.17|0.20|click|14.15|tilt v 36 30 39
| 03.37|0.20|click|14.27|tilt v 39 34 41
| 19.30|0.63|click|16.45|tilt / 20 25 34
```

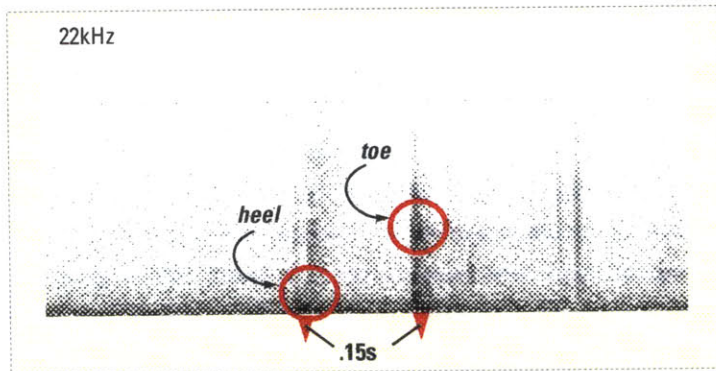
In general a click or scrape could originate from a wide variety of sources — not just doors or shoes but chewing noises, coughs, or mechanical collisions of all sorts — and there is no easy way, *a priori*, to resolve them to a particular source without understanding the surrounding pattern or context. In this case, both clicks are door noises. That would be a likely supposition anyway, since at the opening of a sitcom or some other dramatic

scene, our expectation is that the entrance of a character will be accompanied by walking-in noises.

In many cases, footsteps are distinctive enough that they can be identified as such somewhat reliably: a single footstep is usually performed after the fact by a “Foley” artist, and although they come in many varieties (for at least as many kinds of shoes and Foley-pit walkways as are on-hand during the mix), they are generally exaggerated for effect. A footstep is most always a “heel-TOE” pair of clicks; it is occasionally a singleton and rarely a triplet. In addition, footsteps often are vaguely pitched — they have a somewhat clunky, “hollow” sound, usually higher for women’s shoes than for men’s but in either case, with energy skewed toward the bottom of the frequency spectrum. Door noises may be pitched, too, but typically have a louder, more even, broadband “slap” before the pitched nature becomes apparent. For instance, here is a doorclick:



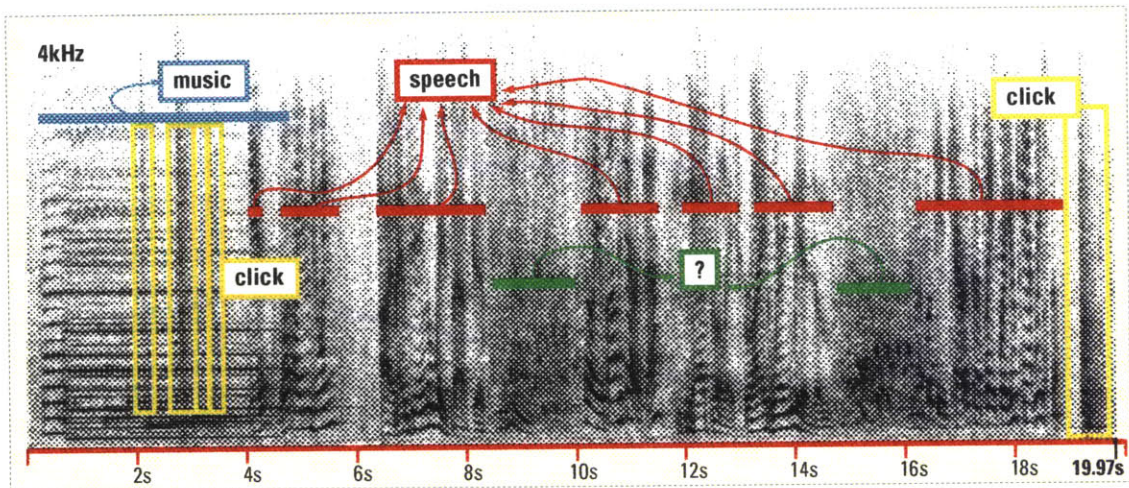
In this case, the picture of the broadband clicks is a little misleading. Of the main group of five dark, vertical lines, what is actually heard is a triplet, “clunk-clunk-clunk,” spaced at .15s intervals denoted by the red wedges; notice that most of the energy, in black, is concentrated in the “blobs” at the bottom of the spectrum. The subsequent (and preceding) clicks are almost too faint to be heard, and are echoes, or part of the latch noise. By contrast, a typical footstep looks rather different: there are usually two noises, the heel (which is a lower-pitched “clunk”) and the toe (a louder and higher-pitched “chink”):



Certainly not all footsteps look like this, but the pattern is quite common. The “**tilt**” field in the events output by the **click** program roughly indicates the skew of the spectrum so that these kinds of characterizations can be made: e.g, “/” indicates that the energy is tilted toward the high end; “-” indicates fairly flat, broadband noise; and “^” indicates a bulge in the middle. Most door noises are of these sorts, whereas most footsteps consist of a heel-noise tilted toward the low end (“\”) followed by a toe-noise tilted to the middle or high end (“^” or “/”). If a pattern such as this can be reliably matched, that of course provides a third way to detect footsteps that might be mixed in with speech.

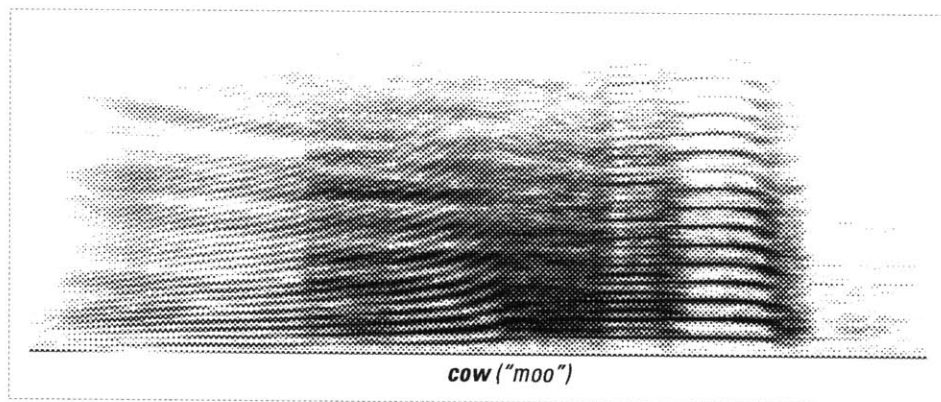
```
| 02.10|0.20|click|tilt v 25 22 38
| 02.78|0.20|click|tilt - 15 15 18
| 03.17|0.20|click|tilt v 36 30 39
| 03.37|0.20|click|tilt v 39 34 41
| 19.30|0.63|click|tilt / 20 25 34
```

In any case, for the *Mr. Ed* example, the upshot of this analysis is that the output of the click detector is filtered by ignoring clicks that co-occur with speech, leaving us with the door noises at the start and end of the segment:

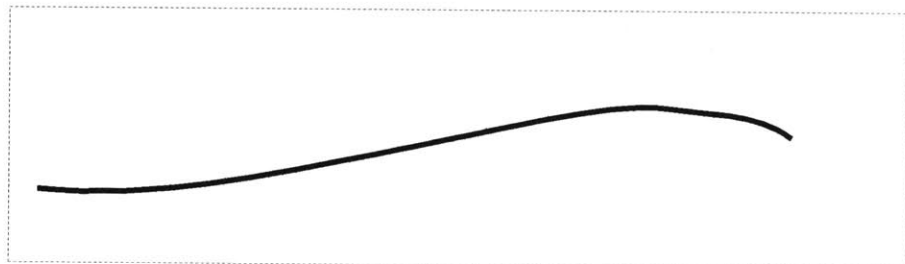


### Cows, horses, laughtracks and other effects

There is no easy way to recognize that the unaccounted sounds (in green) happen to be laughter, and this is a good opportunity to introduce the general topic of effects recognition. As Serra's thesis implicitly showed, and our intuitions based on visual inspection of audio spectra corroborate, we observe *pitched* sounds of many sorts (vowels, pitched music, etc; in general, these are sounds that arise from a vibrating, "driving" source in a system of resonators), and *noisy* sounds (clicks, scrapes, and longer-lasting noise of a highly stochastic nature, like whirring fans, ripping cloth or bubbling water). It appears that a well-refined analysis of pitch and the resonating filter system around it, perhaps combined with a little knowledge of noisebursts (as we have begun to indicate through the discussion of clicks) will go a long way towards helping to formalize, at least statistically, many sorts of sounds. For instance, a "cow-like" pattern:

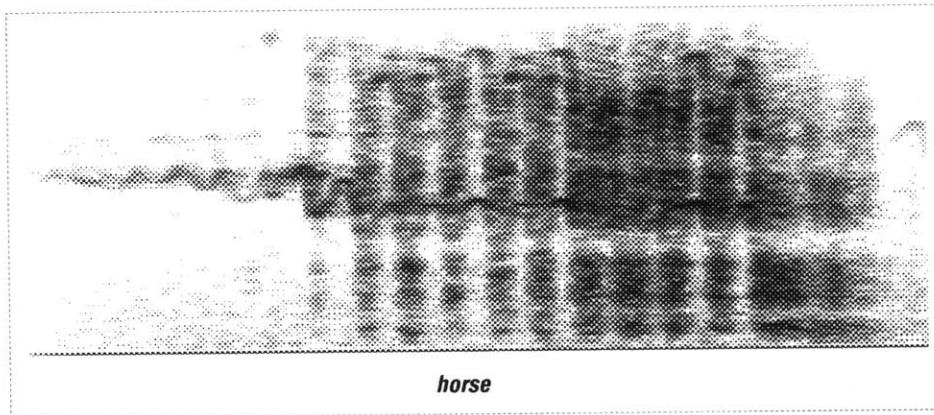


can often be easily recognized by the characteristic "hook" shape of its pitch contour:

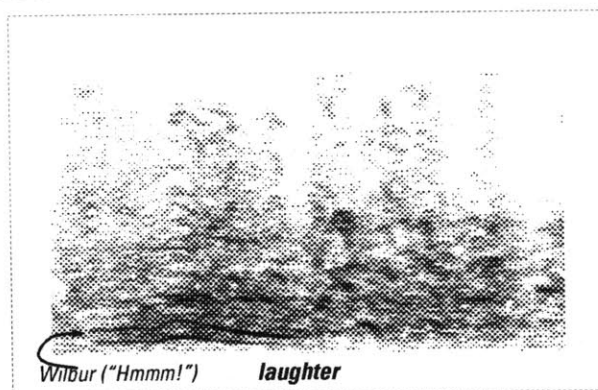




A “whinnying” horse also has quite distinctive patterns of pitch and noise:



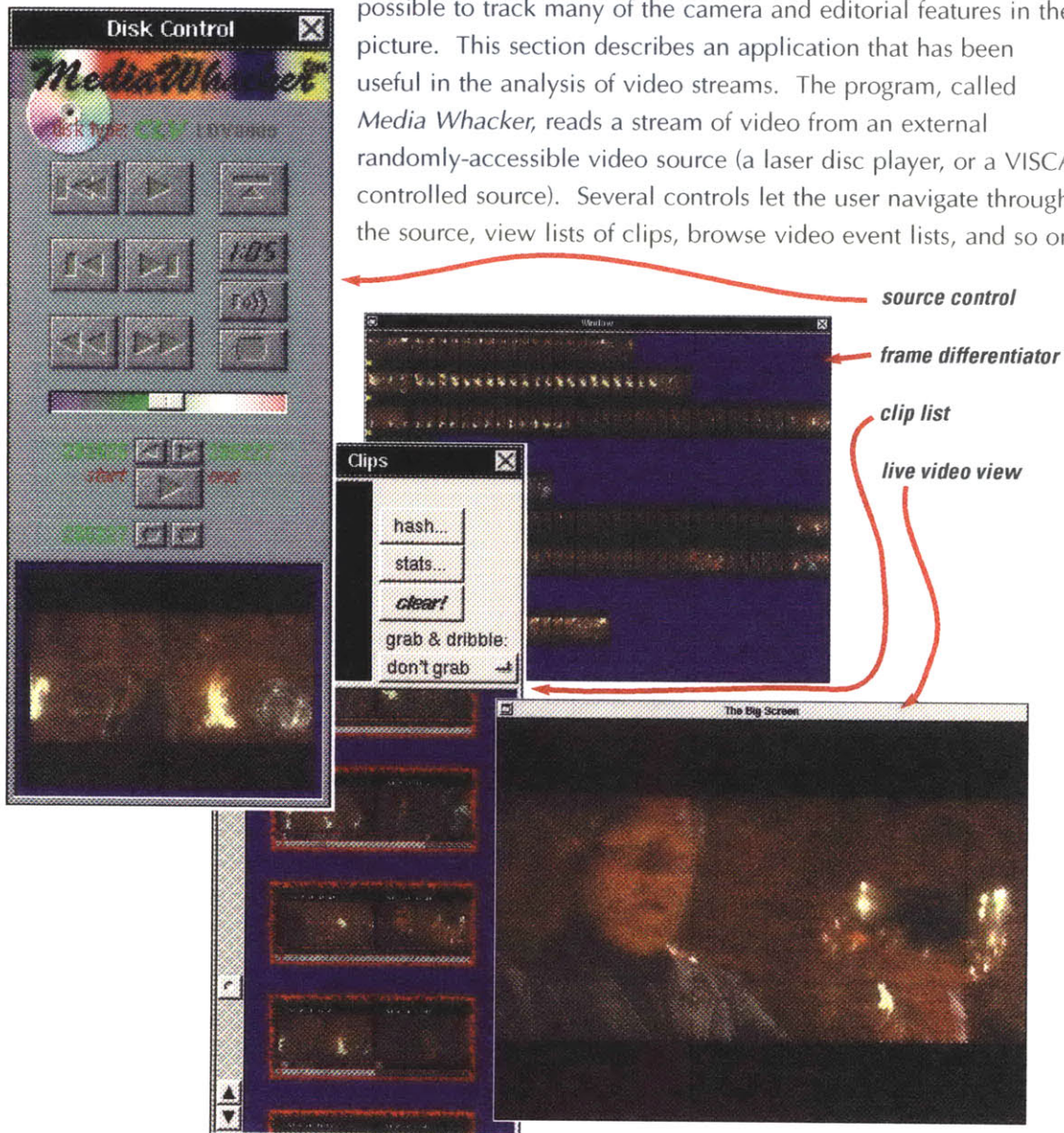
On the other hand, the recognition of highly stochastic sounds, including complex mixtures like shattering glass or applause, which can be thought of as “aural textures,” has not yet been studied. The noisy, canned audience laughter in *Mr. Ed* is an example of that:



This is a case where straightforward spectral analysis does not reveal much obvious structure. A comb-filtered analysis of the coherent pitched components does show some hoots, but not enough to form discernible patterns. More research will have to be done in this area. The best a naive event filter can do for now is label such sounds as “unidentified” in the hope that the source can be determined by some subsequent analyzer.

## 4.2 Visual Scene Analysis

We now turn our attention toward the more integrated analysis of picture as well as sound in a film. Recognizing objects in the picture, like faces of actors or settings, is still a daunting job and not likely to be practical anytime soon. On the other hand, it is possible to track many of the camera and editorial features in the picture. This section describes an application that has been useful in the analysis of video streams. The program, called *Media Whacker*, reads a stream of video from an external randomly-accessible video source (a laser disc player, or a VISCA-controlled source). Several controls let the user navigate through the source, view lists of clips, browse video event lists, and so on:

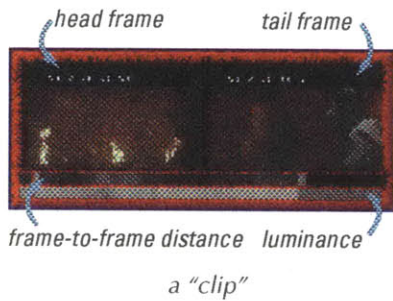


### Scene parsing with *Media Whacker*

Several research projects have considered or implemented the parsing of a movie into its constituent “clips.” As early as 1986 Sasnett described a scene boundary detector based on the use of a sampling grid (Sasnett, 1986). David Small (1992) used a differentiator trained on the red-green-blue color distance between successive frames in a decimated stream. Elliott (1993) used a collection of heuristics (above all, the continuity of image along the edges of the frame) to locate shot boundaries. Ueda *et al* (1992) used optical flow estimates on small samples of a frame (like Sasnett) to obtain information about cut boundaries, panning, and zooming. Tennenhouse and his group (1993) have also built a video clip detector and experimented with a number of programmatic applications of video event filtering (for example, matching of key frames — like title frames — to index television broadcasts).

Much of this work seems to have occurred around 1992 with the general advent of some amount of free-flowing video input and output in common computers. *Media Whacker* is a similar case in point. The basic program, including video analysis and automatic clip detection, was written in a day and worked the first time. It was needed mainly to provide primitive parsing and inspection of video streams, much like *Spectre* for sound. The program is chiefly used to read video and write out *clip lists*, which are event lists of the clips that make up a film. Like other similar analyzers, it does frame-to-frame differentiation on a decimated stream to locate clips in running video streams. However, this alone misses a number of cuts, particularly fades, wipes, and cuts in low-light scenes. The low-light problem can be avoided by using an automatic gain control that is keyed to the overall luminance in the frame (that is, sensitivity to changes is increased in low-light conditions, so that dark cuts will not be missed). Fades and wipes are detected by special processes that oversee a longer buffer (a few seconds). This seems to find nearly all the principal shot boundaries. In any case, the errors that remain are difficult to correct. They involve false hits (mis-

classifying explosions and other visually disruptive effects), as well as missing other visual effects (like very long visual crossfades, mixtures, or extremely fast, choppy cutting). In practice, those are small problems. A comparison with by-hand analysis shows that the automatic cut detection for a feature film is about 95% accurate; typical two-hour movies have about 1200 cuts (that number is sometimes higher in the case of "action" films).



The clip list output by *Media Whacker* includes not only the basic shot boundary information, but for each clip, its overall luminance and "pixel activity" profiles. These represent measures of luminance and interframe distance over the frames in the clip. For example, a slow-moving clip (like a conversation over a dinner table) has rather low pixel activity. The image analysis software for a time also tracked camera motion, but since this has not been found to be a particularly meaningful index into the content (except for anomalous cases, or perhaps for film editing applications), it was dropped.

### Limitations and extensions

Because of the environment in which it runs (a NeXTDimension 24-bit video system), incoming frames are analyzed only with difficulty. In particular, to do the scene parsing on a live video stream as described here the system is able to perform about 6 frame comparisons per second on frames that have been decimated to 64x48 pixels. It was found that frames as small as 16x12 could be used, although accuracy began to degrade noticeably. Some video sources, like disks, can be stepped a frame at a time, but although it would be possible to compute in much more depth, this has not been necessary. Much of the same video analysis software was used in the piano-roll reader described in chapter 2. While this does demonstrate the utility of having a somewhat flexible video input software library, it does not extend as far as Tennenhouse and others have gone,

primarily due to the awkward nature of managing video frame analysis in the NeXTDimension i860 architecture.

For similar reasons, it is not really feasible yet to integrate live audio and video analysis from signal into a compelling system. Although this can be done with MIDI, VISCA, and other control-protocol systems, general systems are not quite advanced enough yet to afford the signal processing capacity required. This is a natural avenue to work along, however.

### 4.3 The Holy Grail

#### The quintessential adventure

We are now in a position to begin to consider some of the high-level features observable in the picture and sound of a full-length feature film. *Indiana Jones and the Last Crusade* (©1989 Lucasfilm, LTD; 1990 Paramount Pictures) is the third film in the Indiana Jones trilogy that began with *Raiders of the Lost Ark* (1981). It is an archetypal adventure movie: Indy and his father, Henry Jones, are hot on the trail toward recovering the Holy Grail. The path to the Grail is revealed through a series of clues — a fragment of a stone tablet, a knight's tomb, a sphynx-like series of riddles to gain access to the grail — and the chase scenes involve cars, trains, horses, boats, motorcycles, airships, and armored vehicles, among other things.<sup>†</sup>

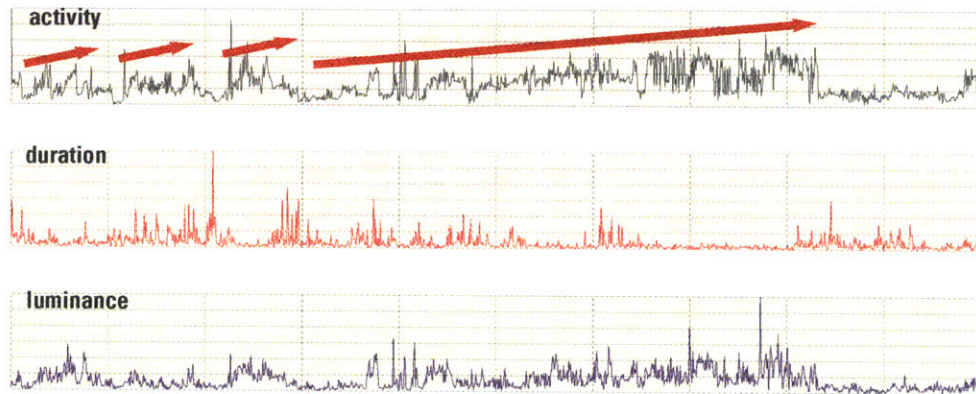
As a first pass, we filter the video through the clip-finder in *Media Whacker*. This yields approximately 1200 clips, the statistics of which are interesting. The following plot shows the “activity” in a clip (that is, the average amount of net pixel change during the clip), the duration of the clip, and the brightness of the clip, all shown as a function of time during the film:



*“The Holy Grail, Doctor Jones...  
The chalice used by Christ during the last supper.”*

*(from Indiana Jones and the  
Last Crusade)*

<sup>†</sup>There is a compelling theory (c.f. Hancock, 1992) that the grail that was the object of Parsifal's quest and the crusades of the Knights Templar was not the chalice used at the Last Supper, but rather, stones bearing a message from God — i.e., the stones carrying the ten commandments, housed in the Ark of the Covenant. The “one, true” Ark is thought to reside in the church of Mary in the holy city of Axum, Ethiopia.



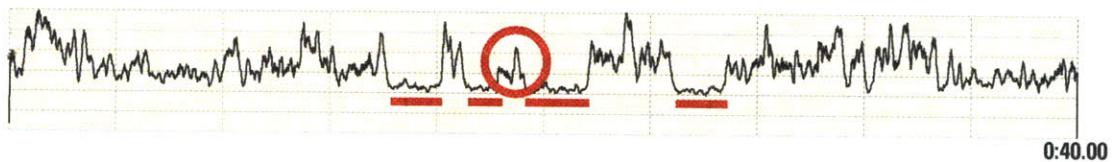
*Indiana Jones and the Last Crusade*

2:06.00

It is easy to see from the general activity level that there are four main action sequences in the film — three introductory chases followed by a fourth long, slow-building sequence that gives way to the final denouement. This is mirrored somewhat in the other statistics: the average duration of a clip tends to get shorter as the action picks up — that is, when the “activity” graph begins to run high, the “duration” graph runs low. The same is somewhat true even of luminance: action-packed scenes tend to be brighter. All of these measures reflect the long-term pace of the film. It would be interesting to study other less-adventure-prone movies along these lines.

### Hearing the clues

At first, the soundtrack yields up its clues less willingly. For example, a cursory run of the music detection filter shows us that there is music throughout:



0:40.00

In fact, in the first 40 minutes, there are only about 7 minutes *without* music. As it happens, it is these *non-musical* moments

that are most interesting, for they contain set-up scenes and dialog that reveal essential parts of the plot — dialog that is too important to be obscured by music. In fact, in the second gap Indy learns that the Grail is about to become the object of his search; when the music comes in (highlighted with a gong in the score, circled in red on the graph), a very special tune is played: it is the stately “Grail” melody that is used throughout the film to underscore the clues to the Grail as they unfold.

Using our other content filters proves somewhat less interesting; while it is possible to find dialog uttered by Indiana Jones, or by other characters, this is somewhat difficult to associate with features in the plot particularly since most of the principal characters speak throughout.

### Finding the *Grail*

The important elements in this movie are all given special melodies in the soundtrack: the familiar “Raiders March” is played whenever Jones makes a safe getaway; the Nazis always are accompanied by an evil tune; and the objects of the search, like the Ark of the Covenant, the Holy Grail, and the knights who guard it, all have special melodies accompanying their appearance, or often even a hint of their presence. For instance, whenever Indy leafs through his father’s “Grail diary,” we hear the Grail theme. In all, the Grail theme appears 19 times in the movie, of which the melody-finding program discussed in chapter 2 currently finds about half. It misses the others due to noise or subtlety in the soundtrack; for example, in one instance, the melody is played quietly, by slightly detuned instruments for a spooky effect while there is rustling paper and speech in the acoustic foreground; this fools the pitch extractor. In another case, at a particularly confusing moment (the “Leap of Faith” scene), the Grail theme is turned into a sort of fugue played quickly on high strings, probably unnoticeable to all but the most musically astute listeners anyway. But the recognizer does find the longest, loudest rendition of the theme, which occurs at the climax, the moment when Indy finally finds the Grail.

**The “Raiders” march:**



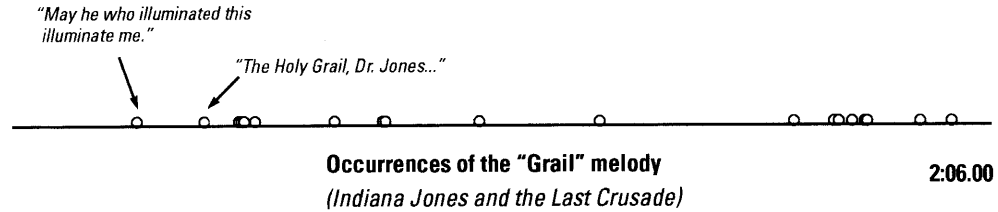
[1, 2, 5, -10, 2, 1]

**The “Grail” theme:**



[5, -2, 2, 2, 1, -1, -2, -1]

The image shows two musical staves. The first staff is titled "The 'Raiders' march:" and contains a short melodic phrase in treble clef with a key signature of one flat. Below the staff is the pitch contour [1, 2, 5, -10, 2, 1]. The second staff is titled "The 'Grail' theme:" and contains a longer melodic phrase in treble clef with a key signature of one flat. Below the staff is the pitch contour [5, -2, 2, 2, 1, -1, -2, -1].



Because the shot boundaries are known, the clips that bracket each appearance of the Grail theme can be selected to abstract a certain thread of the story. It takes about 8 minutes to watch this condensed version of the film.

#### 4.4 Remarks

This chapter brought together most of the audio event filtering techniques developed in earlier chapters, and worked towards a more unified tool for analysis of cinematic material. It pointed out how easy it is to locate the gross editorial features in a picture stream, like shot boundaries. Picture *content* analysis on live video is much harder. Nevertheless, the rudimentary information available from analysis of clips correlates quite well with the general content of the film — for example, one can easily find the important chase scenes, and the “crescendo” of action sequences in a movie. To find equivalent sorts of information in the soundtrack seems harder, at first, for the sound needs to be transformed, filtered, possibly separated to a degree, and analyzed for content. It is thus quite a bit more difficult to determine “aural continuity” than “pictorial,” for instance, because there is no straightforward way to measure such things as the coherence of instruments or sound sources, compared to simply tracking overall color coherence in the picture.

On the other hand, we now do know how to locate useful events in the soundtrack: speech, speakers to a degree, music, melodies, and a few basic sound effects. Even these primitive techniques can afford a powerful index into the content of a movie. For instance, by doing melodic searching throughout the



soundtrack we showed how a movie can be abstracted into its key thematic components. Because melodic extraction from orchestral sound is still a noisy process, it is not quite possible to perform a “common melodic substring” search on the full musical score in order to find all the important themes in advance. Clearly, though, that is something that needs to be done, because it is only by recognizing the repetition of elements like melodies, that we ourselves learn to associate them with essential aspects of the plot.

The choice of a leitmotivic film like the epic Indiana Jones adventure used here can be called contrived. Very few movie scores (or “real life” experiences) make such powerful use of melodies. Of course, it is precisely because the associations are so powerful that they beg to be noticed: after two hours of action, when the “Holy Grail” theme appears for the nineteenth time and is finally played *fortissimo* by the brass section, that is not a subtle event. It is essential to first be able to understand the obvious, recurring symbols if one hopes to eventually sense the subtler things in life.



# 5: Conclusions

---

## 5.0 Recap — What about the Pigeon?

This thesis began with an extremely broad premise — that lack of general audio capability is a serious detriment; that machines ought to function fluently in the acoustic world; and that the ability to do this hinges on understanding the meaningful structures conveyed in sound. Therefore, to enable functional as well as imaginative applications, we need to build a deeper understanding of general sound into the architecture. This may seem so blatantly obvious as to be self-evident, but it was not until 1990 that the problem of auditory scene analysis in and of itself was even well-articulated, and until quite recently, audio applications were at best peripheral and piecemeal in the world of general computing. For example, digital television will drive sound through computers perforce, and increased computational power will leave plenty of room for audio processing, but even that does not mean that sound will be more than an adjunct — unless integral schemes for understanding the sound more deeply are developed.

This thesis has taken a fundamentally different tack from all previous approaches. Classical methods seek to model systems mathematically and then derive a system design that meets those constraints; this is often difficult to relate to high-level tasks. Perceptual and other "neo-classic" methods (including implementations of auditory scene analyzers, speech systems that

*“There are two things which I am confident I can do very well. One is an introduction to any literary work, stating what it is to contain, and how it should be executed in the most perfect manner; the other is a conclusion, shewing from various causes why the execution has not been equal to what the author promised to himself and to the public.”*

— Samuel Johnson  
(1709-1784)

use blackboard architectures and other AI techniques) have met with better success, generally (it is argued) because of a richer and more robust representational core. But these systems are seldom enjoyed for their pragmatic utility.

This thesis argued for rich representations and the need to apply classical and other techniques, but above all, it emphasized abstracting the contents of sound in a way that fuels a fertile and flexible general information environment. I showed that event filters can be built to perform such diverse tasks as speaker identification, music transcription, and indexing of sounds in movie soundtracks. These are all viewed as instances of audio recognizers or indexers, and they can all write into the same representation, and hence interoperate to a large degree.

At first, making all those special purpose parts might seem sloppy or *ad hoc*. So, why not model human perception, which is, after all, our touchstone for this domain?

The pigeon reminded us that even without much general intelligence a machine can glean enough from an audio signal to draw important conclusions about its content. As I argued in chapter 1, humans and other animals themselves accrete a lifetime memory of special-purpose mental mechanisms that, to some extent, mirror the structures in the outside world. We gain deeper understanding by learning how to fit those mental pieces together. In much the same way, it is natural to build a repertoire of sensors and learn to connect them in the larger system. The thesis demonstrated that with several examples.

Implicit in the argument was the idea that there is much to learn by studying not only speech, but a wide variety of sounds. A bit like cinema in the last 70 years, information systems are undergoing a transition into an epoch of fuller and richer audio and video. Movie fans may recall that, by the 1970's, sound in feature films became sufficiently complex that a new specialist was needed — the *sound designer* (first played by Ben Burtt in his 1977 Oscar-winning role for *Star Wars*). As heirs to broadcast and entertainment technology, computers will

recapitulate some of this. Film craft also offers a simple *de facto* taxonomy for organizing sounds (dialog, music, effects, and ambiences). It was useful to operate in that scheme not only because of the variety of sounds, but because the important sounds are generally played up in the mix to ensure that they are heard. Dialog does not often overlap, for instance. This gives modest software a chance in some applications.

## 5.1 Contributions

---

Besides articulating the problem of abstracting structure from sound as one that is of compelling and increasing interest in general computing, a number of specific questions were addressed in the course of this work. To briefly summarize them:

- How can many high-level audio sensors be integrated?

Normalize audio sensors so that they produce uniform descriptions of acoustic events. The event lists used here are one of the simplest and most useful ways of doing that; like MIDI data formats, or ASCII codes, they permit an adequate notation of audio contents around which many tools can be built. This serves as a convenient link to the rest of the system.

- How is a music event finder made?

If “music” is defined to be a “sound containing waves created by instruments that play periodic notes” (*i.e.*, it has to have “melodies” or “chords”), then a music event finder can be built by measuring the average longevity of harmonics (in other words, how stationary the frequency content is, or how striated it appears in a spectral image over time), and then collecting the segments that exceed an acceptable threshold.

- Yes, but does it work?

It works well for detecting what commonly passes for music; this was demonstrated on film soundtracks and radio

programs. Because the spectrum is computed at modest resolution, vibrato is not generally a difficulty. Occasionally, false hits occur (like a telephone ringing, or low, monotone speech), though it is not unreasonable to consider these as minimalist music; anyway, a higher-level filter often disposes of spurious “blips” of music, or collects the dropouts. The filter can be extended or adjusted in straightforward ways for other models of music (like rap, or purely rhythmic forms), or to ignore too-short segments (like a doorbell) or sounds that contain no pitch variability (like a beeper). Obviously there are many other cases that are ambiguous — the “Hitchcock” string effect in horror movies, for example, consists of screeching high glissandi that might be called music only if one can discern that it is being played on a violin. Certainly many people can’t.

- How is a polyphonic pitch extractor for piano music made?

The process first finds note onsets (places where new harmonic information abruptly enters the signal). At each such onset, the previous (resonant and decaying) wave is compared to the new information by considering the differential change in the spectrum (accounting for phase). This contains the harmonics that identify the new note(s). Chords are resolved subtractively.

- Does it work?

With some acute shortcomings. Partly because of the simple method used here (a short-time magnitude spectral estimate), low bass note chords, rolled chords, octave playing, and very fast moving bass notes are fuzzy or otherwise ambiguous in the spectrum. The amount of ambiguity depends on the kind of music — two-voice Bach playing is fairly straightforward, but two-fisted boogie-woogie is not. The current chord-resolver is buggy, and only about 75-80% accurate (and is particularly erroneous when low bass notes are played). It is expected that high-level knowledge (of melodies and harmonies, for instance) could resolve many of these

problems. In addition, the current process does not attempt to sense pedalling; except for certain obvious cases (like long sustained notes) this may prove difficult to estimate.

- What about indexing of conversations?

As with music detection, the process seeks a comb of harmonics, but a comb that varies rapidly in pitch (unlike music). This is characteristic of vowels. The window around each vowel is extended to include surrounding consonants. The measurement of average baseline pitch was, somewhat surprisingly, found to be adequate for distinguishing between many speakers.

- Does that really work?

The algorithm has been found to perform well except when voices cross, when there is much background noise, and when multiple speakers really do have the same intonational frequency. It would be interesting to know, by studying many conversations, how often those problems actually arise. Crossings and mixtures are difficult to resolve (that is the source separation problem, for which methods of attack are known, but no general working solution exists). Guessing the speaker identity based on pitch has shortcomings, but tends to work well in small conversations. One extension would be to filter vowels per speaker for average timbre. Because voice quality can change dramatically in an entertainment piece, one will inevitably need other information to make robust talker assignments; for instance, it may be necessary to understand what is being said.

- How does the soundtrack analyzer work?

It works with several predictable difficulties. Gross features (music, speech, occasionally footsteps and door slams) can be found providing the texture is not a “swampy” mix. Even when the mix is complicated, in movie or radio soundtracks, the principal elements are usually so prominently mixed that

they are easily found. Lacking good models for many sorts of sound effects, though, the only recourse for the many unknown sounds is to try and lump them together and label them as such. When sounds contain a distinctively pitched component, they can likely be matched from a collection of known sounds. With only a few exceptions (some footsteps and doorslams) we have no apparatus for matching the large class of sounds of a stochastic nature (such as bubbling water, sniffing and sneezing, motors running, or other complex sounds).

- How does the movie parser work?

The picture stream is segmented into clips: a frame-to-frame comparator finds hard “butt” cuts, most fades and fast wipes. The clips are grouped according to events in the soundtrack. For example, a long run of dialog between two characters, or a long segment of music indicates that several clips may be grouped into a meaningful unit. It is then possible to index the film based on the logical content of the sound; retrieving the clips containing a particular actor’s dialog, or the clips containing a particular musical theme (a “leitmotif”) were demonstrated.

- What are its limitations?

The picture parser has some shortcomings (it does not catch long dissolves, occasionally misses cuts in low-light conditions and sometimes falsely detects special effects, like explosions); most of these problems can be fixed in a straightforward manner, particularly as video processing systems become more powerful.

The limitations previously mentioned in the case of the soundtrack analyzer naturally carry over. Operations like indexing the film on some aspect of acoustic content are, of course, successful to the extent that the sound is correctly identified, and has meaning in the context of the film.



## 5.2 Lacunae and Future Work

---

This thesis presented a framework in which many problems in content-oriented sensing and manipulation of sound can now be explored. Each of the examples invites further study, but a few prospects seemed to be of particular interest.

In the area of music processing, the piano deconvolution problem can probably be brought to a level acceptable for commercial application. A more interesting implication, though, is that it appears an analyzer could be built that not only finds the pitches, but goes further so as to winnow out the samples for every note. Because experiments as well as experience with sampling music synthesizers show that coupled interactions between vibrating strings are often so small as to be barely noticeable, it seems likely that research in this area would be a useful step towards building a music recording system that produces both gestural and instrumental data in separate forms. The same question applies to concert hall effects: we know that any large and acoustically interesting room has a response that is not information-preserving (that is, the impulse response contains zeros), and further, is time-varying. But surely some component of the hall — like its principal echoes — can be detected and cancelled. This would allow some dereverberation of the signal, and may provide enough information to intuit some possible hall sizes. The question is, how well can the room and the instruments be separated, and is this enough to be interesting? Questions like these are important because the ability to abstract a full model of the components in a musical performance will have a deep impact on future audio systems.

In the area of speech processing, the small experiments in working with sampled speech databases for purposes of adjusting rearranging dialog were very promising. In the past, speech synthesis from large samples of continuous speech was thought to be untenable. Not only is this not true, but as significant databases of conversation begin to accumulate and are indexed

by speech recognition systems, they will furnish an engine for many interesting speech applications. This thesis sketched the beginnings of an approach. When a sentence is well-tagged, so that a system can operate on each word or phrase, it is possible to alter the inflection or timing of words, or to substitute wholly new words or phrases. These can be selected from other utterances in the database and spliced into place, taking care to warp the replacement word to fit smoothly. To make an analogy, when computer memory became plentiful and processing grew fast enough, sampled music synthesizers and "layered" bitmapped displays became feasible. A similar process seems likely to occur when large sampled speech databases and some speech manipulation technology become common. Linear predictive analysis and synthesis should probably be brought to a refined state for this; that was not explored in this thesis.

Many other topics were unexplored here, of course, but in the words of Helmholtz, "a natural philosopher is never bound to construct systems about everything he knows and does not."

## 6: References

---

*article, journal, book, network link*

---

- |   |        |   |
|---|--------|---|
| Robert Alberti<br>Farhad Anklesaria<br>Paul Lindner<br>Mark McCahill<br>Daniel Torrey | 1992   | <i>The internet Gopher protocol</i><br>University of Minnesota Microcomputing<br>and Workstation Networks Center<br><a href="ftp://boombox.micro.umn.edu:/pub/gopher">ftp boombox.micro.umn.edu:/pub/gopher</a> |
| Jonathan Allen<br>M. Sharon Hunnicutt<br>Dennis H. Klatt                              | 1987   | <b>From Text to Speech: the MITalk System</b><br>Cambridge University Press, Cambridge; New York.   |
| Barry Arons   | (1993) | <i>Speech Skimmer: Interactively Skimming Recorded Speech</i><br><b>Proceedings, ACM User Interface Software and Technology</b><br><b>Conference</b> , Atlanta.   |
| J. S. Bach  | (1970) | <b>Aria mit 30 Veränderungen</b><br>in <b>Bach Keyboard Music</b> , Dover, New York.  |
| David Backer  | 1988   | <b>Structures and Interactivity of Media:<br/>A Prototype for the Electronic Book</b><br>MIT Media Laboratory PhD dissertation.   |
| Harold Barlow<br>Sam Morgenstern  | 1948   | <b>A Dictionary of Musical Themes</b><br>Crown Publishers, New York.  |
-

- Georg von Békésy                      1960      **Experiments in Hearing**  
McGraw-Hill, New York.
- David Berkley                              1990      *HuMaNet: an experimental human-machine  
communications network based on ISDN wideband audio*  
**AT&T Technical Journal**, September 1990, p87.
- Jon Bentley  
Donald Knuth  
M. D. McIlroy                              1986      *Programming Pearls: a Literate Program*  
**Communications of the ACM**, 29(6):364, May 1986.
- Tim Berners-Lee                            1992      *World-Wide Web: the information universe*  
Robert Cailliau  
Jean-Francois Groff  
Bernd Pollerman                            **Electronic Networking: Research, Applications,  
and Policy**, 1(2), Spring 1992.
- Q. David Bowers                            1972      **Encyclopedia of Automatic Musical Instruments**  
Vestal Press, Vestal NY.
- Albert S. Bregman                            1990      **Auditory Scene Analysis**  
MIT Press, Cambridge.
- Ronald Bracewell                            1984      *The Fast Hartley Transform*  
**Proceedings of the IEEE**, 72(8), August 1984.
- Ronald Bracewell                            1986      **The Hartley Transform**  
Oxford University Press, New York.
- Guy Brown                                    1992      **Computational Auditory Scene Analysis:  
a Representational Approach**  
University of Sheffield PhD Thesis.
- C. S. Burrus  
M. T. Heidemann  
D. H. Johnson                              **1988      *Archive for History of the Exact Sciences***
- Vannevar Bush                              1945      *As We May Think*  
**Atlantic Monthly** 176:101–108.

- Vannevar Bush 1967 **Science is Not Enough**  
William Morrow, New York.
- William Buxton 1978 *Design Issues in the Foundation of a Computer-Based Tool for Music Composition*  
University of Toronto CSRG-97, October 1978.
- William Buxton 1978 *The Use of Hierarchy and Instance in a Data Structure for Computer Music*  
William Reeves  
Ronald Baecker  
Leslie Miezzi  
**Computer Music Journal**, 5(3):50-56.
- Janet Cahn 1990 *Generating Expression in Synthetic Speech*  
MIT Media Laboratory Master's thesis.
- John Chowning 1982 **Intelligent Systems for the Analysis of Digitized Acoustic Signals**  
Loren Rush  
Bernard Mont-Reynaud  
Chris Chafe  
W. Andrew Schloss  
Julius Smith  
Stanford Department of Music Report STAN-M-15.
- Arthur C. Clarke 1992 **How the World Was One: toward the tele-family of man**  
Bantam, New York.
- Peter Danzig 1992 *Internet Resource Discovery Services*  
Katia Obrazcka  
Shih-Hao Li  
USC-LA Computer Science Department  
*e-mail* [danzig@usc.edu](mailto:danzig@usc.edu)
- Hans T. David 1966 **The Bach Reader: a life of Johann Sebastian Bach in letters and documents.**  
Arthur Mendel  
W. W. Norton, New York.
- Lucinda Dewitt 1986 *Recognition of Novel Melodies after Brief Delays*  
Robert Crowder  
**Journal of Music Perception**, 3(3):259-274, spring 1986.
- Edward Lee Elliott 1993 *Watch – Grab – Arrange – See*  
MIT Media Laboratory Master's thesis.

- Daniel P. W. Ellis                      1993      *A Computer Implementation of Psychoacoustic Grouping Rules*  
MIT Media Laboratory, Perceptual Computing TR#224.
- Daniel P. W. Ellis                      1993      *Hierarchic Models of Hearing for Sound Separation  
and Reconstruction*  
MIT Media Laboratory, Perceptual Computing TR#219.
- Alan Emtage                              1992      *archie: An electronic directory of services for the internet*  
Peter Deutsch                              **Usenix Conference Proceedings**, January 1992.  
[ftp sifon.cc.mcgill.ca/pub/archie](ftp:sifon.cc.mcgill.ca/pub/archie)
- D. C Engelbart                            1969      *Human intellect augmentation techniques*  
NASA; for sale by the Clearinghouse for Federal Scientific  
and Technical Information.
- N. Fakotakis                              1993      *A text-independent speaker recognition system based on  
A. Tsopanoglou                              vowel spotting*  
G. Kokkinakis                              **Speech Communication**, 12(1):57-68.
- James Flanagan                          1990      *Speech Processing: a perspective on the science  
Charles Del Riesgo                          and its applications*  
**AT&T Technical Journal**, September 1990, p2.
- Scott Foster                              1982      *Toward an Intelligent Editor of Digital Audio:  
W. Andrew Schloss                          Signal Processing Methods*  
A. Joseph Rockmore                          **Computer Music Journal**, 6(1):42-51.
- Otto Friedrich                            1989      **Glenn Gould: a life and variations**  
Random House, New York.
- Wilhelm Fucks                            1962      *Mathematical Analysis of Formal Structure of Music*  
**IRE Transactions on Information Theory**, 18(5):224-229,  
September 1962.
- Sadaoki Furui                            1992      *A Speaker Recognition System for Telephone Speech*  
reprinted in Saito (1992):300. cf. Sondhi & Furui (1992).
- Glenn Gould                              1964      *Strauss and the Electronic Future*  
**Saturday Review**, May 30 (cf. Gould and Page, 1984).

- |                                  |      |  |
|----------------------------------|------|--|
| Glenn Gould                      | 1983 | <b>Glenn Gould, by himself and his friends</b><br>Doubleday, Toronto.  |
| Glenn Gould<br>Jonathan Cott     | 1984 | <b>Conversations with Glenn Gould</b><br>Little, Brown, Boston.  |
| Glenn Gould<br>Tim Page          | 1984 | <b>The Glenn Gould Reader</b><br>Knopf, New York.  |
| Donald E. Hall                   | 1992 | <i>Piano string excitation: VI: Nonlinear Modelling</i><br><b><i>Journal of the Acoustical Society of America</i>, 92(1):95.</b> |
| Graham Hancock                   | 1992 | <b>The Sign and the Seal:<br/>The quest for the lost Ark of the Covenant</b><br>Simon & Schuster, New York.                      |
| Stephen Handel                   | 1989 | <b>Listening: an introduction to the perception of<br/>auditory events</b><br>MIT Press, Cambridge.                              |
| R. V. L. Hartley                 | 1928 | <i>The Transmission of Information</i><br><b><i>Bell System Technical Journal</i></b>  |
| R. V. L. Hartley                 | 1942 | <i>A More Symmetric Transform</i><br><b><i>Proceedings of the IRE</i></b>  |
| Michael Hawley<br>Samuel Leffler | 1985 | <i>Windows for Unix at Lucasfilm</i><br>Usenix Proceedings, summer 1985.   |
| Alex Hauptmann                   | 1993 | <i>Speak EZ: High-Quality Speech Synthesis at CMU</i><br><b><i>Proceedings, Eurospeech</i>, 1993.</b>                            |
| Michael Hawley                   | 1986 | <i>MIDI Music Software for Unix</i><br>Usenix Proceedings, summer 1986.  |
| Michael Hawley                   | 1987 | <i>The Digital Librarian</i> (NeXT software)<br>NeXT Computer, Inc, Redwood City, CA.  |
| Michael Hawley                   | 1988 | <i>Webster (first digital edition)</i> (NeXT software)<br>NeXT Computer, Inc, Redwood City, CA.                                  |

- |  |      |  |
|--|------|--|
| Michael Hawley                                       | 1989 | <i>Totentanz: an experiment in symphony emulation</i><br>MIT Media Laboratory memorandum.  |
| Michael Hawley                                       | 1990 | <i>The Personal Orchestra, or,<br/>Audio Data Compression by 10000:1</i><br><b><i>Usenix Computing Systems Journal</i></b> , 3(2)<br>University of California Press, Berkeley. |
| Michael Hawley                                       | 1992 | <i>Library of Congress interface</i> (NeXT software)<br><a href="ftp://sonata.cc.purdue.edu/pub/next">ftp://sonata.cc.purdue.edu/pub/next</a>                                  |
| Hermann Helmholtz                                    | 1863 | <b>On the Sensations of Tone</b><br>Dover, New York (1954).  |
| Wolfgang Hess  | 1983 | <b>Pitch Determination of Speech Signals</b><br>Springer-Verlag, New York.   |
| Debbie Hindus  | 1992 | <i>Semi-Structured Capture and Display of<br/>Telephone Conversations</i><br>MIT Media Laboratory Master's thesis.   |
| Tomlinson Holman                                     | 1991 | <i>New Factors in Sound for Cinema and Television</i><br><b><i>Journal of the Audio Engineering Society</i></b> , 39(7):529-539.   |
| M. Honda<br>F. Itakura                               | 1992 | <i>Low Bit-Rate Speech Waveform Coding</i><br>reprinted in Saito, p. 189.  |
| Chris Horner   | 1993 | <i>NewsTime: a Graphical Interface to Audio News</i><br>MIT Media Laboratory Master's thesis.  |
| Jean-Marie Hullot                                    | 1989 | <i>Interface Builder</i> (NeXT software)<br>NeXT Computer, Inc: Redwood City, CA.  |
| David Jaffe<br>Julius Smith<br>Douglas Fulton et al. | 1992 | <i>NeXT Music Kit Documentation</i><br><a href="ftp://ccrma-ftp.stanford.edu/pub/MusicKit.README">ftp://ccrma-ftp.stanford.edu/pub/MusicKit.README</a>                         |



- N. Jayant 1990 *Speaker Verification: a Tutorial*  
**IEEE Communications Magazine**, 28:42-48 (January).
- Ibn al-Razzaz al-Jazari (Donald Hill, trans.) 1974 **The Book of Knowledge of Ingenious Mechanical Devices**  
Reidel Dordrecht, Boston.
- Keith Johnson 1990 *The Role of Perceived Speaker Identity in F0 Normalization of Vowels*  
**Journal of the Acoustical Society of America**, 88:642-654.
- Brewster Kahle 1991 *Wide-area information server concepts, alpha release documentation*  
[ftp think.com:/wais](ftp://think.com:/wais)
- Brewster Kahle 1991 *An information system for corporate users: Wide-area Information Servers*  
**ConneXions – the Interoperability Report**, 5(11) November 1991.
- Robert Kahn 1988 *The digital library project, volume 1: The world of knowbots*  
Corporation for National Research Initiatives
- Vinton Cerf
- James F. Kaiser 1983 *Some observations on vocal tract operation from a fluid flow point of view*  
**Proceedings of the conference on physiology and biophysics of voice**, Iowa City, Iowa, May 4-7 1983. (ISBN 0-936947-53-5)
- Alan Kay 1979 *Personal Dynamic Media*  
**Computer**, 10(3), March 1977.
- Adele Goldberg
- Alan Kay 1984 *Computer Software*  
**Scientific American**, 251(3):52-59.
- Andrew Kazdin 1989 **Glenn Gould at work: creative lying**  
Dutton, New York.

- Donald Knuth 1992 **Literate Programming**  
CSLI (Stanford University), Palo Alto.
- L. F. Lamel  
L. R. Rabiner  
A. E. Rosenberg  
J. G. Wilpon 1981 *An Improved Endpoint Detector  
for Isolated Word Recognition*  
**IEEE Trans. Acoustics, Speech, and Signal Processing**  
ASSP-29:777-785.
- Peter Langston 1990 *Little Languages for Music*  
**Usenix Computing Systems**, 3(2).
- Paul Lansky  
Kenneth Steiglitz 1978 *Synthesis of Timbral Families by Warped Linear Prediction*  
**Computer Music Journal**, 5(3):45-49.
- V. Lesser  
R. D. Fennell  
L. Erman  
D. R. Reddy 1975 *Organization of the Hearsay-II Speech Understanding  
System*  
**IEEE Trans. on Acoustics, Speech, and Signal Processing**  
ASSP-23, 11-24.
- J. C. R. Licklider 1965 **Libraries of the Future**  
MIT Press, Cambridge.
- Eric Lindemann  
Miller Puckette 1991 *Animal — a rapid prototyping environment for  
computer music systems*  
**Computer Music Journal**, 15(3):78-100.
- Arthur Loesser 1954 **Men, Women & Pianos**  
Simon and Schuster, New York.
- M. Lottor 1992 *Internet Growth (1981-1991)*  
<ftp://nic.ddn.mil/rfc/rfc1296.Z>
- Gareth Loy 1985 *Musicians make a standard: the MIDI phenomenon*  
**Computer Music Journal**, 13(1):36-46.
- Richard Lyon 1986 *Experiments with a computational model of the cochlea*  
**Proc. IEEE International Conference on Acoustics,  
Speech, and Signal Processing**, 1986.

- R. Macaulay  
T. Quatieri 1986 **Speech analysis/synthesis based on a sinusoidal representation**, IEEE Trans ASSP-34:744-754.
- Tod Machover  
Joe Chung 1989 *Hyperinstruments: Musically Intelligent and Interactive Performance and Creativity Systems*  
**Proceedings International Computer Music Conference**  
Computer Music Association, San Francisco.
- Tod Machover  
Joe Chung  
Andy Hong  
Neil Gershenfeld 1991 *Hyperinstruments: Musically Intelligent/Interactive Performance and Creativity Systems*  
Yearly report to the Yamaha Corporation,  
MIT Media Laboratory.
- Pattie Maes  
Beerud Sheth 1993 *Evolving Agents for Personalized Information Filtering*  
**IEEE Conference on Applications of Artificial Intelligence**  
Orlando, Florida.
- Robert Maher 1989 *An Approach for the Separation of Voices in Composite Musical Signals*  
University of Illinois (Urbana) PhD dissertation.
- David Marr 1982 **Vision: a computational investigation into the human representation and processing of visual information**  
W. H. Freeman, San Francisco.
- Henry Massalin 1987 *SuperOptimizer: A Look at the Smallest Program*  
in **Proceedings of the Second Conference on Architectural Support for Programming Languages and Operating Systems**, Palo Alto, California.
- Henry Massalin 1990 Fast Discrete Hartley Transform implementation  
(personal communication).
- Henry Massalin 1992 **Efficient Implementation of Fundamental Operating System Services**  
Columbia University PhD dissertation.
- Max V. Mathews  
Joan E Miller 1969 **The Technology of Computer Music**  
MIT Press, Cambridge.

- Max V. Mathews  
John R. Pierce                      1989      **Current Directions in Computer Music Research**  
MIT Press, Cambridge.
- John McCarthy                      1966      *Information*  
**Scientific American**, 215(3):65-95, September 1966.
- H. Marshall McLuhan              1964      **Understanding Media: The Extensions of Man**  
McGraw-Hill, New York.
- David Mellinger                    1992      **Event Formation and Separation in Musical Sound**  
Stanford University PhD dissertation.  
<ftp://ccrma-ftp.stanford.edu/pub/Theses/DAVEMThesis.ps.Z>
- Jonathan Miller                    1971      **Marshall McLuhan**  
Harcourt, Brace & World, New York.
- G. A. Miller                         1956      *The Magic Number Seven, Plus or Minus Two:  
some limits on our capacity for processing information*  
**Psych. Review**, 63(2):81.
- Neil Miller                         1975      **Filtering of singing voice signal from noise by synthesis**  
University of Utah PhD Thesis.
- S. Millman (ed)                    1975      **A History of Engineering and Science in the Bell System:  
the Early Years**  
AT&T Bell Laboratories, Indiana.
- S. Millman (ed)                    1984      **A History of Engineering and Science in the Bell System:  
Communication Sciences**  
AT&T Bell Laboratories, Indiana.
- Marvin Minsky                    1986      **The Society of Mind**  
Simon and Schuster, New York.
- Bernard Mont-Reynaud  
Mark Goldstein                    1985      *On finding rhythmic patterns in musical lines*  
**Proceedings, International Computer Music Conference**  
p391-397.

- Bernard Mont-Reynaud      1989      *Source Separation by Frequency Co-Modulation*  
David K. Mellinger      **Proceedings, First International Conference on Music Perception and Cognition**, Kyoto, Japan.
- James A. Moorer      1975      **On The Segmentation and Analysis of Continuous Musical Sound by Digital Computer**  
Stanford University PhD Dissertation.
- James A. Moorer      1979      *The Use of Linear Prediction of Speech in Computer Music Applications*  
**Journal of the Audio Engineering Society**, 27(3):134-140.
- James A. Moorer      1984      *The Lucasfilm ASP*  
**Computer Music Journal**.
- S. Hamid Nawab      1989      *High-Level Adaptive Signal Processing*  
V. Lesser      NAIC Final Report, 17.
- S. Hamid Nawab      1992      **Symbolic and Knowledge-Based Signal Processing**  
Alan Oppenheim (eds)      Prentice-Hall, Englewood Cliffs, NJ.
- S. Hamid Nawab      1993      *Principal Decomposition of Time-Frequency Distributions*  
Daniel Beyerbach      Unpublished manuscript (Boston University, Department of Electrical, Computer, and Systems Engineering).  
Erkan Dorken
- Nicholas Negroponte      1970      **The Architecture Machine**  
MIT Press, Cambridge.
- Nicholas Negroponte      1979      *Books Without Pages*  
**IEEE Proceedings**.
- Nicholas Negroponte      1993      *Debunking Bandwidth: from Shop Talk to Small Talk*  
**WIRED 1.3**, July/August.
- Theodor H. Nelson      1974      **Computer Lib / Dream Machines**  
Reissued (1987) by Microsoft Press, Redmond, WA.

- B. Clifford Neuman 1991 *The Prospero File System*  
University of Washington (Seattle)  
Department of Computer Science  
<ftp://cs.washington.edu/pub/pfs>
- Allen Neuringer 1984 *Music Discrimination by Pigeons*  
Debra Porter ***Journal of Experimental Psychology: Animal Behavioral Processes*, 10:138-148**
- Harry Nyquist 1924 *Certain Factors Affecting Telegraph Speed*  
***Bell System Technical Journal*, 3:324-326.**
- Harry Nyquist 1928 *Certain Topics in Telegraph Transmission Theory*  
***Transactions of the American Institute for Electrical Engineering*, 47:617-644.**
- Harry Ferdinand Olson 1967 **Musical Engineering**  
Dover Publications, New York.
- Alan Oppenheim 1969 *Analysis-synthesis system based on homomorphic analysis of speech*  
***Journal of the Acoustical Society of America*, 45(2):458-465.**
- Arthur W. J. G Ord-Hume 1984 **Pianola: the History of the Self-Playing Piano**  
George Allen & Unwin, London and Boston.
- Caroline Palmer 1989 **Timing in Skilled Music Performance**  
Cornell University PhD dissertation.
- Thomas W. Parsons 1976 *Separation of speech from interfering speech by means of harmonic selection*  
***Journal of the Acoustical Society of America*, 60(4):911-918.**
- Thomas W. Parsons 1986 **Voice and Speech Processing**  
McGraw-Hill, New York.
- Richard Phillips 1991 *MediaView: a General MultiMedia Digital Publishing System*  
***Communications of the ACM*, 34(7):75-83.**

- Janet Pierrehumbert      1981      *Synthesizing Intonation*  
*Journal of the Acoustical Society of America*, 70(1):985-995.
- Rob Poor      1992      *TimeWarp* (NeXT software)  
*personal communication.*
- Ralph K. Potter      1947      **Visible Speech**  
George Kopp  
Harriet Green  
Van Nostrand: New York.
- Miller Puckette      1991      *Combining Event and Signal Processing in the MAX*  
*Graphical Programming Environment*  
*Computer Music Journal*, 15(3):68-77.
- John S. Quarterman      1993      *personal communication.*
- L. R. Rabiner      1967      **Speech Synthesis by Rule: an Acoustic Domain Approach**  
MIT PhD dissertation.
- L. R. Rabiner      1975      **Theory and Application of Digital Signal Processing**  
Bernard Gold  
Prentice Hall: New York.
- L. R. Rabiner      1993      **Fundamentals of Speech Recognition**  
Prentice Hall: New York.
- Bruno Repp      1993      *Some empirical observations on sound level properties of*  
*recorded piano tones*  
*Journal of the Acoustical Society of America*, 93(1):1136-44.
- Curtis Roads      1985      **Foundations of Computer Music**  
John Strawn  
MIT Press, Cambridge.
- David Rosenthal      1993      **Machine Rhythm: Computer Emulation of Human**  
**Rhythm Perception**  
MIT Media Laboratory PhD dissertation.
- Robert Rowe      1991      **Machine Listening and Composing: Making Sense of**  
**Music with Cooperating Real-Time Agents**  
MIT Media Laboratory PhD dissertation.

- |                         |      |  |
|-------------------------|------|--|
| Robert Rowe             | 1993 | <b>Interactive Music Systems</b><br>MIT Press, Cambridge.  |
| Curt Sachs              | 1940 | <b>The History of Musical Instruments</b><br>W. W. Norton & Company, New York.   |
| Oliver Sacks            | 1989 | <b>Seeing Voices: A Journey into the World of the Deaf</b><br>University of California Press, Berkeley.  |
| Shuzo Saito, ed.        | 1992 | <b>Speech Science and Technology</b><br>IOS Press, Washington.   |
| David Sarnoff           | 1968 | <b>Looking Ahead</b> ; the papers of David Sarnoff<br>McGraw-Hill, New York.   |
| Russell Mayo Sasnett    | 1986 | <i>Reconfigurable Video</i><br>MIT Master's thesis.  |
| W. Andrew Schloss       | 1985 | <b>On the Automatic Transcription of Percussive Music:<br/>From Acoustic Signals to High-Level Analysis</b><br>Stanford University PhD dissertation.   |
| Michael Schwartz        | 1991 | <i>Resource discovery in the global internet</i><br>TR CU-CS-555-91, University of Colorado, Boulder.<br><a href="ftp://cs.colorado.edu/pub/cs/techreports/schwartz">ftp://cs.colorado.edu/pub/cs/techreports/schwartz</a> |
| Edmund Willer Scripture | 1904 | <b>Elements of Experimental Phonetics</b><br>Charles Scribner, New York.   |
| Xavier Serra            | 1988 | <b>An Environment for the Analysis, Transformation,<br/>and Resynthesis of Musical Sounds</b><br>Stanford University PhD dissertation.   |
| Claude Shannon          | 1948 | <i>A Mathematical Theory of Communication</i><br><b>Bell System Technical Journal</b> , 27:379-423.  |



- Mark Sheldon  
David Gifford  
Pierre Jovelot  
James O'Toole, jr. 1991 *Semantic File Systems*  
**Proc IEEE InfoCom**, June 1990.
- Roger Shepard 1981 *Psychophysical complementarity*  
in **Perceptual Organization**, Kubovny & Pomerantz (eds)  
Erlbaum Press, Hillsdale, New Jersey.
- Kim E. A. Silverman 1987 **The Structure and Processing of Fundamental Frequency  
Contours**  
Cambridge University PhD dissertation.
- Herbert A. Simon 1960 *The Shape of Automation*  
in **Perspectives on the Computer Revolution**, Z. W. Pylyshyn (ed.), Prentice-Hall, Englewood Cliffs, NJ (1970).
- Herbert A. Simon 1968 *Pattern in Music*  
Richard K. Sumner in **Formal Representations of Human Judgement**,  
B. Kleinmuntz (ed.), John Wiley and sons, New York.
- Herbert A. Simon 1985 **The Sciences of the Artificial** (second edition)  
(1969) MIT Press, Cambridge
- John Sloboda 1985 *An Exceptional Musical Memory*  
B. Hermelin **Journal of Music Perception**, 3(2):155-170, winter 1985.  
N. O'Connor
- David Small 1992 *"Wizard of Oz" scene detector*  
*personal communication.*
- Mohan Sondhi (ed.) 1992 **Advances in Speech Signal Processing**  
Sadaoki Furui M. Dekker, New York.
- William Leslie Sumner 1981 **The Organ: its Evolution, Principles of Construction and Use**  
St. Martin's Press, New York, N.Y.

- |  |      |   |
|--|------|---|
| Ivan Edward Sutherland                                   | 1963 | <b>Sketchpad: a man-machine graphical communication system</b><br>MIT PhD dissertation.   |
| David Tennenhouse  | 1993 | <i>Research group annual report</i><br>MIT Lab for Computer Science.  |
| Hirotsada Ueda<br>Takafumi Miyatake<br>Satoshi Yoshizawa | 1992 | <i>Impact: an Interactive Natural-Motion-Picture<br/>Dedicated Multimedia Authoring System</i><br>Hitachi Central Research Lab memorandum<br><b>email: <a href="mailto:ueda@crl.hitachi.co.jp">ueda@crl.hitachi.co.jp</a></b> |
| Barry Vercoe   | 1984 | <i>The Synthetic Performer in the Context of Live Performance</i><br><b>Proceedings of the International Computer Music Conference</b><br>San Francisco, CA.  |
| Pierre Wellner   | 1991 | <i>The Digital Desk Calculator: Tactile Manipulation on a<br/>Desktop Display</i><br><b>Proc. ACM Symposium on User Interface Software and<br/>Technology</b> , November 1991, p27-33.  |
| Peter F. Williams<br>Barbara Owen                        | 1988 | <b>The Organ</b><br>W.W. Norton, New York.  |
| Emanuel Winternitz                                       | 1982 | <b>Leonardo da Vinci as a Musician</b><br>Yale University Press, New Haven.   |
| Nicole Yankelovich<br>Norman Meyerowitz                  | 1985 | <i>Reading and Writing and the Electronic Book</i><br><b>IEEE Computer Magazine</b> , 18(10), October.  |

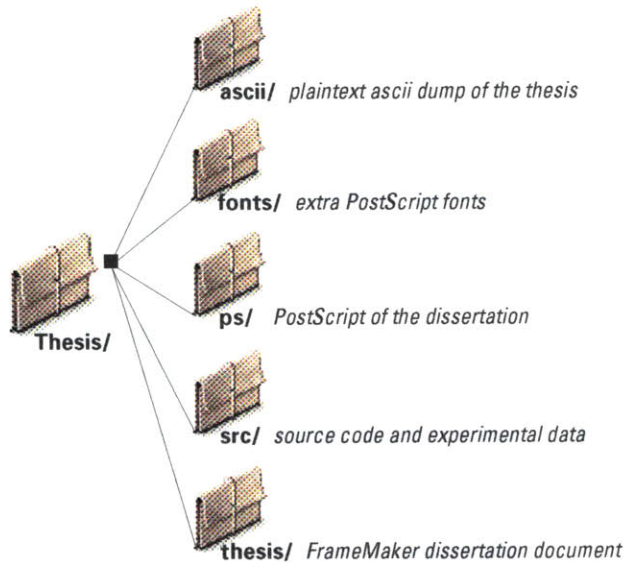
## appendix 1:

# Software in this thesis

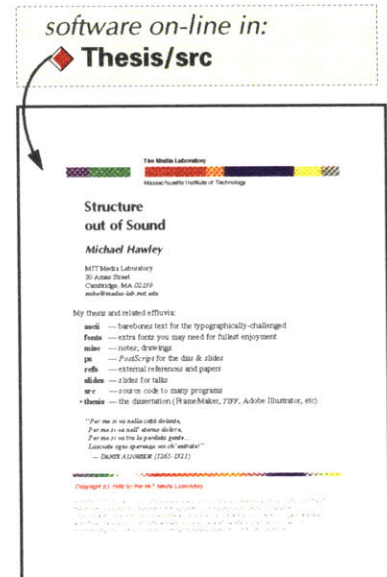
## a1.1 Overview

This appendix provides a summary of the software discussed in this dissertation. The document (a *Frame Maker*™ document) and most of the software described in it were written on a NeXT computer. The material may be available on-line and if you have a NeXT system, it is worthwhile to read it that way.

On-line organization is as follows:



The source code to most of the programs mentioned here (like the *music* and *speech* filters, the *Spectre* sound editor, etc)



are in *Thesis/src*. So are a number of general utilities (including the Library of Congress program). Most of the code is in C or Objective-C written for *NeXTSTEP* 3.0, and some of it was written in an experimental “rich text” format (see the next appendix for details on that). All of it is evolving: much of this is impromptu-style research software, not a consumer product, and what is frozen in this thesis is just a snapshot. Following are descriptions of the main parts.

### Software in *src/...*

Contains four categories of software:

- sound** a library of sound analysis utilities (*libana1.a*), shell-level commands (like *music* and *speech*), and other audio applications (like *Spectre*). Discussed in more detail below.
- midi** a small library of midi-related utilities and applications, including:
  - libmidi.a* — C utilities for midifile processing
  - da* — a midifile dis-assembler
  - ra* — a midifile re-assembler
  - midi2ps* — render a midifile as PostScript
  - mpu* — convert MPU-401-format to midifile format
  - play* — play a midifileThis code follows the rough paradigm established by Hawley (*Personal Orchestra*, 1992) and Langston (1990).
- event** a library of utilities for manipulating fielded event lists. Discussed in more detail below.
- misc** miscellaneous software and applications, including:
  - Opener* — software archiving utility
  - Library of Congress* — book-buying interface
  - Weather* — network weather interface
  - PianoRoll* — experimental piano roll reader
  - MediaWhacker* — laser disc movie indexer

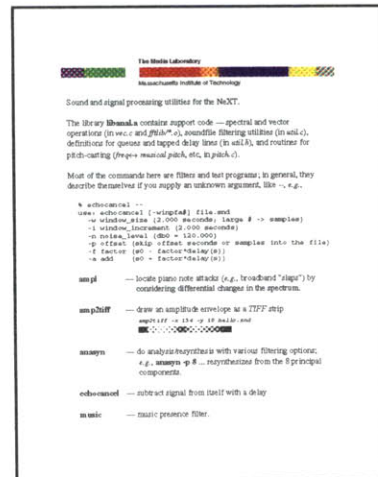
- quarto** — PostScript pagination utility
- ref-o-matic** — reference formatter
- rtf-utilities** — see appendix 2
- graph** — simple graph-plotting software

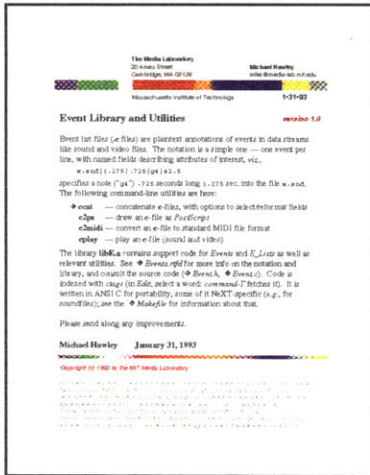
### Audio Software in *src/sound/...*

The **libana1.a** library contains utilities for reading and writing sound streams. For writing event filters it is usually sufficient to work with 8KHz monophonic audio data, but in any case, the library provides a simple, uniform interface wherein input sound data is converted to 1-channel 32-bit samples. The Hartley transform functions (for obtaining log-magnitude spectra), vector arithmetic functions, and other filtering elements (tapped delay lines, data queues, etc) are all had by linking with this library.

Shell-level utilities include:

- amp** soundfile ⇒ amplitude envelope graph
- amp2tiff** soundfile ⇒ amplitude intensity graph
- anasyn** analysis/resynthesis/filtering of a soundfile
- avgspec** compute average spectrum for a soundfile
- bias** calculate background noise bias
- click** locate “click” events in a soundfile
- echocancel** comb filtering utility
- find-caller** locate “telephone”-like speech in talk shows
- music** music event detection filter
- musyn** music removal filter
- pcorr** pitch correlation filter
- pitch** pitch calculation utility
- polyp** polyphonic pitch extractor
- pplay** play a soundfile from piped input
- sndps** soundfile ⇒ postscript waveform
- splay** play cross-faded soundfiles; cf. **Event/eplay**
- spec2tiff** render a spectrum (all the spectral illustrations in this document were generated with this)
- speaker** attempt to recognize speakers
- speech** locate “speech” events in a soundfile





## Event Lists in *src/event/...*

The **libE.a** library contains utilities for reading and writing *Event* lists. These are textual annotations of events in data streams like sound and video files. The idea is to have a simple way to describe the meaningful contents of such data, at most any level of detail. By convention, the notation is one line per event, with named fields describing attributes of interest:

**x.snd | 1.275 | .725 | g4 | 42.5**

describes a sound that in the file **x.snd** starting at **1.275** seconds, is **.725** seconds long, and is a note (“**g4**”) of amplitude **42.5** db. Event lists have provisions for collecting events into groups (that is, they form a general graph, not a strict hierarchy). The location information is flexible and may be in seconds, video timecode, samples, or some other forms.

Two simple shell-level commands are included (**e2midi**, to convert an event list to midifile format; and **eplay**, to play an audio event file, with various options to control the manner of mixing and splicing). Several **sound/...** utilities read and write event lists, as do the **Spectre** and **MediaWhacker** applications.

## appendix 2:

# Rich Text Programming

*“What is the use of a book,” thought Alice,  
“without pictures or conversations?”*

— Lewis Carroll  
*Alice’s Adventures in Wonderland*

### a2.0 Introduction

... Or, what is the use of source code written in monofont ASCII? We increasingly demand multimedia systems with high aesthetic quality, yet most coding is still done using the crudest sort of plaintext, trimmed to fit the 24x80-character view of an antique glass terminal. This is worth reconsidering.

Some of the programs discussed in this dissertation were written in a format that is replete in its use of typography, color, illustrations, and other attachments or links. In this case, “rich text format” (RTF) was used, and parsers (as for **C**, **awk**, **make**, and other development tools) were modified slightly to cast RTF input to plaintext ASCII as needed.

This appendix describes the way I did that, and a few other tools germane to programming with rich text. These include RTF-tolerant system tools and other filters for RTF documents (for example, a template filter, a rubricator, etc).

Brief mention is made of similar work along these lines.

software on-line in:  
♦ **Thesis/src/rtf-utilities**

The Make Laboratory  
20 Ames Drive  
Cambridge, MA 02139  
Microsoft Member  
Microsoft Office 4.0/4.0a  
Microsoft's Institute of Technology  
3-31-93

## RTF Utilities

version 1.0

RTF (Microsoft® Rich Text Format) is a document markup language commonly used as an interchange format. RTF is fairly well-supported by NEXTSTEP™, there will be further enhancements in the future.

The following utilities are here:

- ♦ **rtf-ascii** — convert RTF to plaintext ASCII
- ♦ **app, ccpp.c** — RTF-tolerant C preprocessor
- ♦ **rtfcat** — concatenate RTF files to the standard output, optionally converting to Postscript, etc.
- ♦ **rtfdiff** — extra de-plaintext files through RTF templates
- ♦ **rtfmail** — export to NEXTSTEP RTF files
- ♦ **rubric** — highlight words or regular expressions in an RTF file

`rtfcat -i -o out.rtf in.rtf -s howley -f size.rtf`  
converts occurrences of "Howley" (-i) (ignore case) to "Howley".

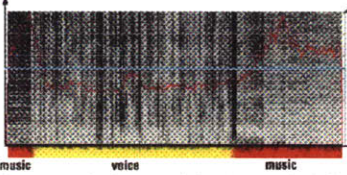
- ♦ **samples/** — sample RTF files
- ♦ **load/** — other RTF-tolerant tools (grep, make, ctags, etc)
- ♦ **RTF-specrft** — specification of the RTF format

The Makefile contains general instructions for installation. To make the C compiler RTF-tolerant, make sure you put the C preprocessor in `/usr/bin/Makefile`, you then compile using the `-B` option, viz.,  
`cc -B/usr/bin/Makefile *.c`  
where `.c` is an RTF file (or possibly a link to a .rtf directory).  
(If you are doing, you can put it in `lib` and dispense with the `-B` business).  
In that way you can put attachments or links directly in the code, an example may be found in ♦ `Ar2.c` and ♦ `Makefile`.

## a2.1 What rich-text code looks like

**Music Detection Filter**

One indication of the presence of music in a sound stream is frequencies that are sustained over several frames of a short-time magnitude spectrum. The horizontal striations are probably pitched musical notes:



This spectrum shows a segment of music, a segment of voice, and a segment of music. Overlaid on the spectrum in red is the output of the music function, below, that can be thresholded to find music-like segments. The function essentially measures how stationary the peaks in the signal are, as follows:

Slide the analysis frame in short hops over spectrum, creating a vector  $p[i] = 1$  if  $magn_{sumFrames}(128)$ , with some filtering to if a particular peak  $p[i]$  goes 1101... it's if a peak run is longer than  $minRun$  frames. The average length of the peak runs over "longish" peak passes a threshold (above  $1101$ ).

We only consider runs that are long enough to be set for analyzing 8kHz input, although input. Perhaps hissy components cloud note.

Note that we only consider peaks in the spectrum. Most of the information we need lies in the spectrum.

M. J. Hawley  
MIT Media Laboratory  
mjh@media-lab.mit.edu  
October, 1992

```
#include "util.h"
#define MaxPeaks 128
#define WinSize 512
```

The compiler and development tools are enhanced to accept code containing typographic detail (for clarity) and other attachments. This program contains a spectrum illustrating the output of the function, and the sound that was input. It also contains a link to an external document (a header file in this case). Programs should be dynamic, first-class documents, not low-grade plaintext.

```
#define HalfWin (WinSize/2)
#define N (WinSize*2)

int minRun = 3; // shortest run counted as a held peak
int maxRun = MaxPeaks/3; // count really long runs early
int peakT = 0; // # of held runs in frame delay buffer
int Peak[MaxPeaks][HalfWin]; // total length of the run
int numPeaks = MaxPeaks; // the delay buffer (see P(...) macro)
int MinBin = 10, MaxBin = 70; // length of the delay buffer
int CurFrame = 0; // consider frequencies in this range
float Offset = 0.0; // the current frame number // starting position in file

#define P(n) Peak[(CurFrame-n)%numPeaks] // nth-delayed frame of peaks
#define Spectrum(m,h,s,n) WindowedHartley(h,n,s,n).sp_LogMag16(m,h,n)

findPeaks(int *p, int *m, int n)
// m is a magnitude spectrum; p[i] ← isPeak(m[i])? 1 : 0
int i;
bzero(p,n*sizeof(int));
for (i=MinBin,m=MinBin,p+=MinBin;i<MaxBin;
    if (m[i]>=m[i+1] && m[i]>=m[i-1] && m[i]>=m[i+2] && m[i]>=m[i-2])
        *p = 1, i += 3, m += 3, p += 3;
    else
        i++, m++, p++;
}

music(int *s, int n)
int m[N], h[N], *p3, *p2, *p1, *p0, *pl, n2=n/2;
float f;

// decrement peak accumulators:
for (p0=P(0)+MinBin, p1=P(0)+MaxBin; p0 < p1; p0++)
    if (*p0 > minRun) peakN--, peakT -= *p0;

// find a new frame of peaks
Spectrum(m,h,s,n);
findPeaks(P(0),m,n2);

// 4-point smoother - emphasize sustained peaks: ...1101 => 1111
p3=P(3)+MinBin, p2=P(2)+MinBin, p1=P(1)+MinBin, p0=P(0)+MinBin;
for (p1<p0+MaxBin; p0<p1; p3++, p2++, p1++, p0++)
    if (*p3 && *p2 && !*p1) *p1 = *p0;

// merge incoming column of peaks and increment peak accumulator
for (p2=P(4)+MinBin, p3=P(3)+MinBin, p1=P(3)+MaxBin; p3<p1; p2++, p3++)
    if (*p2 && *p3 && *p2 < maxRun)
        *p3 += *p2, *p2 = 0;
    else
        if (*p2 > minRun)
            peakN++, peakT += *p2;
f = (float)peakT/(float)(peakN? peakN:1);

main(ac,av) char *av[]; int ac; {
    int i;

    winSize = WinSize; winIncr = 48;
    for_each_argument {
        case 'i': winIncr = atoi(argument);
        case 'p': Offset = atoi(argument);
        case 't': minRun = atoi(argument);
        case 'B': MaxBin = atoi(argument);
        case 'b': MinBin = atoi(argument);
        case 'n': numPeaks = atoi(argument);
        Default : use();
    }

    for (;i<ac;i++)
        dofile(av[i]);
    exit(0);
}
```



## a2.2 Quick overview

---

### 1. If you have a NeXT system, read this on-line:

start with  **Thesis/src/rtf-utilities/Readme.rtf**

### 2. What is RTF source code?

Code written in RTF (“rich text format”) may contain fonts and other formatting information, and, per NeXT’s extensions, embedded graphics, attached files, and active “links.”

RTF source files come in two varieties: simple files (with no graphics or attachments), or directories (named with a **.rtfd** extension) that contain the RTF file and any included pictures.

If the file begins

```
{\rtf0\ansi . . .
```

it’s an RTF file.

*RTF specification on-line in:*  
 **rtf-utilities/RTF-spec.rtf**


### 3. Why would a programmer want to compose in RTF?

A program is a communication vehicle for people as well as machines. Most programmers who trouble to write clear code appreciate being able to insert a bit of mathematics, or a picture or a sound when it helps describe what’s going on, or being able to change the typeface or the color when code needs to be highlighted. Much more code is read than written, so a clear presentation is valuable. It is reasonable, in this day and age, to expect to be able to write or read a program using the same fully-featured word processor† you use to write a business letter.

### 4. My compiler won’t appreciate that.

Probably not; your compiler will need to be fixed.

The instructions in **rtf-utilities** describe a way to do that.

For the case of **C**, it suffices to change the preprocessor (**cpp**)  so that on opening a file, if it turns out to be RTF it is first run through a filter to convert it to palatable ASCII. A modified version of the *GNU* preprocessor is included.

## 5. What about my other development tools?

### What about the rest of Unix, and NT, and . . . ? !

You may want to fix those to be RTF-literate as well.

The utilities here include enhanced versions of: **ctags**, **lex**, **gawk**, **grep**, and **make**. ♦ In each case the source code is given along with a description of the modification, which usually involves simply inserting an **rtf-ascii** conversion when appropriate, and that has been packaged in the form of routines that can be used in place of **open()/fopen()**.

## 6. Ugh. You really must be joking.

### What if I want to use this code but I can't edit in RTF?

You probably can, using *WriteNow*<sup>™</sup>, *Word*<sup>™</sup> or some other contemporary word processor. Otherwise, convert the RTF material to ASCII. (The **rtf-utilities/...** themselves are in plain ASCII for that reason.) Use the command:

```
rtf-ascii [files...]
```

It is a conventional Unix filter, so you will need to cache the output in a temporary file:

```
rtf-ascii x.c > /tmp/x; cp /tmp/x x.c
```

## 7. What other tools are present?

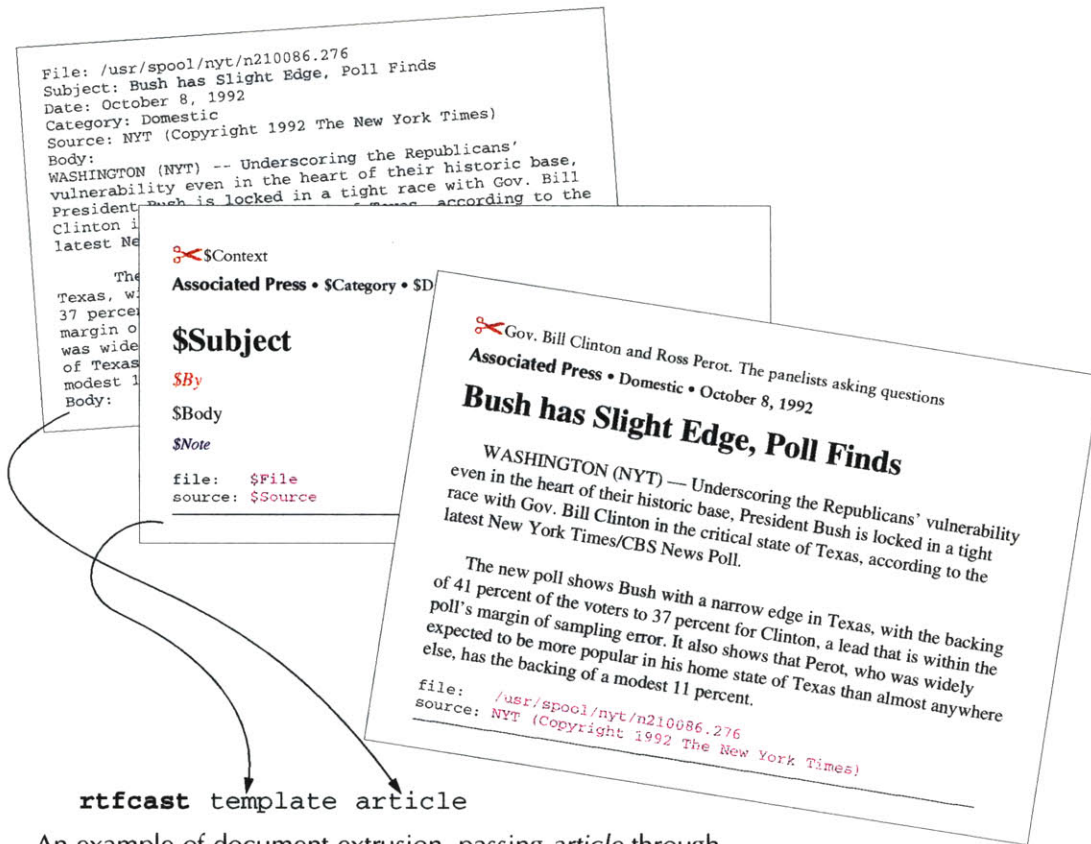
The subdirectory **samples** ♦ contains a number of examples, and the **Makefile** (*n.b.*, an RTF Makefile) runs through them. A highlighter, called **rubric**, can be used to filter RTF documents and highlight words (*rubric* comes from the Latin *ruber* meaning red, and referring to the red words in old illuminated bibles). This provides one example of a way to edit the typography of a document; it has been used, for example, in a news filtering and delivery system.

Another example is **rtfcast** which extrudes plaintext files through rich-text templates. This is useful when one wants to shape text according to a particular form, as shown:

The new poll shows Bush with a narrow edge in Texas, with the backing of 41 percent of the voters to 37 percent for Clinton, a lead that is within the poll's margin of sampling error

```
rubric -c red,bold,italic  
-w clinton
```

The new poll shows Bush with a narrow edge in Texas, with the backing of 41 percent of the voters to 37 percent for **Clinton**, a lead that is within the poll's margin of sampling error



An example of document extrusion, passing *article* through an RTF *template*.

**8. How is this relevant to the thesis *on sound*? It seems prissy.**

In this field, software is the medium for communicating research, and software ought to be dynamic, not frozen in a book. Persons interested in the audio work here may also be interested in the code, so it was important to take time to try to write it clearly, and since they will inevitably stumble on the RTF parts, an explanation was warranted. When programs operate in obscure ways on *sound*, it can be very instructive to add an illustration and a sound to the source code; at least, it was helpful for me.

**9. It seems obsolescent. There are better text representations than RTF, better languages than C, and much of your code is twangingly tuned to the NeXT environment anyway.**

Systems will continue to change. C and RTF are both somewhat baroque languages, but they are sufficiently widespread that they are likely to continue being readable through software for some time. There are choices other than RTF for the purpose of enhancing the format of programs; almost any reasonable choice will serve the purpose of helping to evolve systems that are more fluent in a variety of document types. While I would not leap to the defense of RTF (or NeXT's extensions) on intrinsic technical grounds, at the present time, RTF is the most prevalent markup language for document exchange, and it is integrated fairly well in the NeXT system, which has become popular in the audio research community. More important than the mechanics, though, is the idea of advancing systems by improving the quality of interactive documents for programmers as well as end users. (In fact, it could be argued that compound document architectures won't improve much until programmers use them for their daily coding.)

**10. Hasn't this been done already? by Knuth, for instance?**

Read the next section.

## **a2.3 Remarks**

---

### **Better things for better programming . . .**

Information science is metalingual. It has learned that particular problems are often best described in private notations, or "little languages," to use Kernighan's phrase. Above all it has learned that it is useful to have languages for writing languages.

Languages are most often textual, but they certainly don't have to be. People communicate with schematic pictures and through

sign languages, and there are a number of important “visual” languages in use today. Music scores are an obvious example of a quasi-textual graphical language; so are computer “window” systems and nearly all graphical applications, as well as some of the tools for creating them. Hullot’s *Interface Builder* (1988) is an example of a graphical meta-tool that allow one to sketch the layout and interconnection of algorithmic building blocks. Many such tools sprouted from *Sketchpad* (1963), but have flourished noticeably in recent years, perhaps simply because the supply of graphical workstations and the demand for well-designed graphical interfaces in turn favors design tools that are suited to that domain. The tasks managed by a visualization system are seldom well-described in Fortran.

There are many other not-overtly-textual programming systems that permit the user to communicate through gestures, spoken utterances, or other sorts of data samples so as to induce agencies to perform the desired task. Often, though, when a description is boiled down to its most abstract and powerful form, it is expressed as text, and most any tool that amplifies the ability to work with forms of text also improves one’s ability to manipulate the problem described in it. (What we have come to think of as “text” is not a purely alphabetic concept — like “technology” it derives from a craftly weaving together of ideas and symbols, as in *texere*, τεκτον, and τεχνη).

Computer representations of languages have improved in fits and starts, from the early days of physical wiring, punched cards, and op-codes, through to “softer” and richer representations, like EBCDIC and ASCII. The sheer volume of code being produced has required legible programs, and humane coding practice — hence the trend towards “structured” languages, and a heightened sensitivity to good programming style. To that end, Kernighan and Plauger’s *Elements of Programming Style* (1974) attempted to do for programs what Strunk and White did for prose.

### **From 5-bit Baudot to 8-bit ASCII to . . . ?**

The time is ripe for another transition. 16-bit character coding standards like Unicode are being developed as the successor to ASCII to accommodate worldwide languages. At the same time, digital document structures are improving to bring computer media up to par in design quality with traditional print media. Document casting and translation mechanisms will be needed. One can imagine that, someday, it will be common for users to request that whatever material being presented be cast into their preferred language and style. Clearly, the future holds not just a bigger “byte” but a variety of compound document architectures, and it is time that they became a part of day-to-day programming.

### **WEB considered harmful.**

Recognizing these problems and trends, Knuth (1984) began to develop a system for what he called *literate programming* — he viewed programs as artistic works of literature: “Instead of imagining that our main task is to instruct a *computer* what to do, let us concentrate rather on explaining to *human beings* what we want a computer to do.” For this he devised a language called *WEB* that fused documentation and code into a single package: essentially, it mixed the typesetting facilities of T<sub>E</sub>X with the programming style of Pascal, along with a bit of “glue” for indexing and cross-referencing of variables and modules. The *WEB* program could be filtered through T<sub>E</sub>X to generate a printed book, or through a PASCAL interpreter to generate a working program. (Refer to Knuth’s *Literate Programming* for a lengthy review of the subject, as well as numerous examples of *WEB* programs).

There are some disturbing problems with this. One can hardly disagree with the desire to write human-readable code — but let’s disagree anyway. Programs are not illuminated manuscripts. Fundamentally, a program is *not* a work of literature, and a programmer’s job is not, first and foremost, to

explain his or her intentions to other people. It is important that code be human- as well as machine-readable, but a program is a tool for cutting through problems, and its form follows that function: it needs to fit the problem, and also needs to afford some economy of expression and usability for the programmer. Programs do not necessarily benefit from fancy presentation. To dress a program up as literature can be like taking a chainsaw and making it into a comfortable piece of sculpture.

Another difficulty with Knuth's approach is that it tends to violate a number of engineering sensibilities. WEB extends only with difficulty to other languages (a new dialect of WEB must be wrapped onto any new language, and although there are tools to automate this, the result is — another new dialect). Worse, it is not conducive to debugging, since a special debugger must be constructed to interpret WEB code in the dialect to which it has been adjusted (alternatively, one can debug in the intermediate language, thus working with two versions of the source). Overall, by overemphasizing the literary bent and by attempting to mingle the previously separate structures of programming languages with document structures, WEB code tends to yield what McIlroy once called "rococo, Fabergé eggs" — ornate museum pieces, instead of snappy, constructive components (c.f. Bentley 1986 for McIlroy's discussion of why his one-line shell script was better than Knuth's 17-page WEB solution to a word counting problem). That is a serious liability, which is unfortunate because the original motivation, to improve the quality and communicability of software, can exist in harmony with the utilitarian aspects.

### Is a rich-text approach any better?

Yes. It does not suffer from the fatal flaws of Knuth's approach. One can still write an unadorned program, but the option of using fonts or illustrations to highlight a point is available.

*right:*

```
/* A working program. */
#include <stdio.h>

main(){
    printf("hello world.\n");
}
```

*wrong:*

```
/* De olde chestnutte. */
#include <stdio.h>
#define ye
#define olde
/* here beginneth ye main() */
ye olde main(){
    ye printf("hello world.\n");
}
/* It endeth. S.D.G */
```

*programs do not necessarily benefit from fancy presentations*

It encourages writing nice-looking code but not at the expense of writing workable code. Because it does not tangle with the private syntax of programming languages, any program including compilers and debuggers can transparently process rich forms of text given a bit of system support for converting documents. In fact, this suggests an important missing piece in current system architectures; namely, that a process that is reading or writing data be able to request that a high-level format converter be interposed (e.g., “open as English text”). Ritchie’s streams for 8th edition Unix (1989) did this to a degree; missing is the support for writing converters, and for describing formats so that these casts can be automatically managed.

A drawback of sorts is that rich-text documents can be complicated structures, and that may require considerable supporting code. For example, the NeXT “text object” that provides enough of an interface for multifont editing, editing of links and attachments, etc, is extremely complex.

Another drawback is that it can be difficult to edit rich-text documents on antique, non-graphical terminals, of which there are still many in the world, but that is a little like pointing out that it can be difficult to pedal a bicycle after a 50-horsepower engine has been bolted onto the frame. This restriction will not apply in a few years.

## **a2.4 Summary**

---

There are virtues in applying richer document forms to programming. The approach with RTF taken here shows a path towards a more humane and dynamic form of code. In that sense it achieves most of the goals of Knuth’s approach without the insidious problems. It is also immediately useful because once good rich-text support exists in the system, with negligible overhead, other components like compilers can function transparently with the richer forms of text. More important, this is a path that clears the way toward making the overall system



fluent in compound document architectures. That is useful because it could make the richest forms of documents as useable as the simplest ones.

In addition, in keeping with the well-known property of good abstraction, when the design of an integral document representation is extended, the whole system benefits. For example, extensions to RTF already include attachments and links that are useful in programs as well. It is not hard to imagine including “active” objects like meters or video windows. This may seem unnecessary in the minimalist view of a “hello, world” program, but can be extremely useful when writing programs to manipulate other media, like some of the audio code in this thesis. Related to structurally-oriented sound processing, a promising extension not yet tried is the idea of attaching soundfile pointers to text in the document. For example, a speech-to-text converter might produce as output a rich-text document in which each textual word is mapped to the underlying sound samples. In this way one could build a dialog editor that works by cutting and pasting text to effect analogous operations on the underlying sounds, perhaps with some intelligence added to improve the splicing and editing of the sounds (e.g., selecting a word and italicizing it could add emphasis to the underlying utterance).