Creating Music by Listening

by

Tristan Jehan

Diplôme d'Ingénieur en Informatique et Télécommunications
IFSIC, Université de Rennes 1, France, 1997

M.S. Media Arts and Sciences
Massachusetts Institute of Technology, 2000

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, 2005

Author ....
Program in Media Arts and Sciences
June 17, 2005

Certified by ....
Tod Machover
Professor of Music and Media
Thesis Supervisor

Accepted by ....
Andrew B. Lippman
Chairman, Departmental Committee on Graduate Students

Creating Music by Listening

by Tristan Jehan

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning, on June 17, 2005,
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Abstract

Machines have the power and potential to make expressive music on their own. This thesis aims to computationally model the process of creating music using experience from listening to examples. Our unbiased signal-based solution models the life cycle of listening, composing, and performing, turning the machine into an active musician, instead of simply an instrument. We accomplish this through an analysis-synthesis technique by combined perceptual and structural modeling of the musical surface, which leads to a minimal data representation.

We introduce a music cognition framework that results from the interaction of psychoacoustically grounded causal listening, a time-lag embedded feature representation, and perceptual similarity clustering. Our bottom-up analysis intends to be generic and uniform by recursively revealing metrical hierarchies and structures of pitch, rhythm, and timbre. Training is suggested for top-down unbiased supervision, and is demonstrated with the prediction of downbeat. This musical intelligence enables a range of original manipulations including song alignment, music restoration, cross-synthesis or song morphing, and ultimately the synthesis of original pieces.

Thesis supervisor: Tod Machover, D.M.A.
Title: Professor of Music and Media

Thesis Committee

Thesis supervisor ....

Tod Machover
Professor of Music and Media
MIT Program in Media Arts and Sciences

Thesis reader ....

Peter Cariani
Research Assistant Professor of Physiology
Tufts Medical School

Thesis reader ....

François Pachet
Researcher
Sony Computer Science Laboratory

Thesis reader ....

Julius O. Smith III
Associate Professor of Music and (by courtesy) Electrical Engineering
CCRMA, Stanford University

Thesis reader ....

Barry Vercoe
Professor of Media Arts and Sciences
MIT Program in Media Arts and Sciences

Thesis Committee

Thesis supervisor

Tod Machover
Professor of Music and Media
MIT Program in Media Arts and Sciences

Thesis reader

Peter Cariani
Research Assistant Professor of Physiology
Tufts Medical School

Thesis reader

François Pachet
Senior Researcher
Sony Computer Science Laboratory

Thesis reader

Julius O. Smith III
Associate Professor of Music and (by courtesy) Electrical Engineering
CCRMA, Stanford University

Thesis reader

Barry Vercoe
Professor of Media Arts and Sciences
MIT Program in Media Arts and Sciences

Acknowledgments

It goes without saying that this thesis is a collaborative piece of work. Much like the system presented here draws musical ideas and sounds from multiple song examples, I personally drew ideas, influences, and inspirations from many people to whom I am *very* thankful for:

My committee: Tod Machover, Peter Cariani, François Pachet, Julius O. Smith III, Barry Vercoe.

My collaborators and friends: Brian, Mary, Hugo, Carla, Cati, Ben, Ali, Anthony, Jean-Julien, Hedlena, Giordano, Stacie, Shelly, Victor, Bernd, Frédo, Joe, Peter, Marc, Sergio, Joe Paradiso, Glorianna Davenport, Sile O'Modhrain, Deb Roy, Alan Oppenheim.

My Media Lab group and friends: Adam, David, Rob, Gili, Mike, Jacqueline, Ariane, Laird.

My friends outside of the Media Lab: Jad, Vincent, Gaby, Erin, Brazilnut, the Wine and Cheese club, 24 Magazine St., 45 Banks St., Rustica, 1369, Anna's Taqueria.

My family: Micheline, René, Cécile, François, and Co.

"*A good composer does not imitate; he steals.*"

– Igor Stravinsky

Table of Contents

List of Figures

CHAPTER ONE
Introduction

"*The secret to creativity is knowing how to hide your sources.*"

– Albert Einstein

Can computers be creative? The question drives an old philosophical debate that goes back to Alan Turing's claim that "a computational system can possess all important elements of human thinking or understanding" (1950). Creativity is one of those things that makes humans special, and is a key issue for artificial intelligence (AI) and cognitive sciences: if computers cannot be creative, then 1) they cannot be intelligent, and 2) people are not machines [35].

The standard argument against computers' ability to create is that they merely follow instructions. Lady Lovelace states that "they have no pretensions to *originate* anything." A distinction, is proposed by Boden between psychological creativity (P-creativity) and historical creativity (H-creativity) [14]. Something is P-creative if it is fundamentally novel for the individual, whereas it is H-creative if it is fundamentally novel with respect to the whole of human history. A work is therefore granted H-creative only in respect to its *context*. There seems to be no evidence whether there is a continuum between P-creativity and H-creativity.

Despite the lack of conceptual and theoretical consensus, there have been several attempts at building creative machines. Harold Cohen's AARON [29] is a painting program that produces both abstract and lifelike works (Figure 1-1). The program is built upon a knowledge base full of information about the morphology of people, and painting techniques. It plays randomly with thousands of interrelated variables to create works of art. It is arguable that the creator in this case is Cohen himself, since he provided the rules to the program, but more so because AARON is not able to analyze its own work.

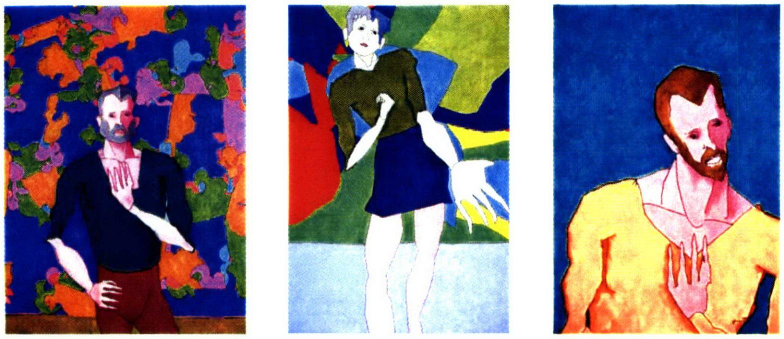


Figure 1-1: Example of paintings by Cohen's computer program AARON. In Cohen's own words: *"I do not believe that AARON constitutes an existence proof of the power of machines to think, or to be creative, or to be self-aware, to display any of those attributes coined specifically to explain something about ourselves. It constitutes an existence proof of the power of machines to do some of the things we had assumed required thought, and which we still suppose would require thought, and creativity, and self-awareness, of a human being. If what AARON is making is not art, what is it exactly, and in what ways, other than its origin, does it differ from the* real thing*?"*

Composing music is *creating* by putting sounds together. Although it is known that humans compose, it turns out that only few of them actually do it. Composition is still regarded as an elitist, almost mysterious ability that requires years of training. And of those people who compose, one might wonder how many of them really innovate. Not so many, if we believe Lester Young, who is considered one of the most important tenor saxophonists of all time:

> *"The trouble with most musicians today is that they are copycats. Of course you have to start out playing like someone else. You have a model, or a teacher, and you learn all that he can show you. But then you start playing for yourself. Show them that you're an individual. And I can count those who are doing that today on the fingers of one hand."*

If truly creative music is rare, then what can be said about the rest? Perhaps, it is not fair to expect from a computer program to either become the next Arnold Schönberg, or not to be creative at all. In fact, if the machine brings into existence a piece of music by assembling sounds together, doesn't it compose music? We may argue that the programmer who dictates the rules and the constraint space *is* the composer, like in the case of AARON. The computer remains an instrument, yet a sophisticated one.

The last century has been particularly rich in movements that consisted of breaking the rules of previous music, from "serialists" like Schönberg and Stockhausen, to "aleatorists" like Cage. The realm of composition principles today is so disputed and complex, that it would not be practical to try and define a set of rules that fits them all. Perhaps a better strategy is a generic modeling tool that can accommodate specific rules from a corpus of examples. This is the approach that, as modelers of musical intelligence, we wish to take. Our goal is more specifically to build a machine that defines its own creative rules by listening to and learning from musical examples.

Humans naturally acquire knowledge, and comprehend music from listening. They automatically hear collections of auditory objects and recognize patterns. With experience they can predict, classify, and make immediate judgments about genre, style, beat, composer, performer, etc. In fact, every composer was once ignorant, musically inept, and learned certain skills essentially from listening and training. The act of composing music is an act of bringing personal experiences together, or "influences." In the case of a computer program, that personal experience is obviously quite non-existent. Though, it is reasonable to believe that the musical experience is the most essential, and it is already accessible to machines in a digital form.

There is a fairly high degree of abstraction between the digital representation of an audio file (WAV, AIFF, MP3, AAC, etc.) in the computer, and its mental representation in the human's brain. Our task is to make that connection by modeling the way humans perceive, learn, and finally represent music. The latter, a form of memory, is assumed to be the most critical ingredient in their ability to compose new music. Now, if the machine is able to *perceive* music much like humans, *learn* from the experience, and *combine* the knowledge into creating new compositions, is the composer: 1) the programmer who conceives the machine; 2) the user who provides the machine with examples; or 3) the machine that makes music, influenced by these examples?

Such ambiguity is also found on the synthesis front. While composition (the creative act) and performance (the executive act) are traditionally distinguishable notions—except with improvised music where both occur simultaneously—with new technologies the distinction can disappear, and the two notions merge. With machines generating sounds, the composition, which is typically represented in a symbolic form (a score), can be executed instantly to become a performance. It is common in electronic music that a computer program synthesizes music *live*, while the musician interacts with the parameters of the synthesizer, by turning knobs, selecting rhythmic patterns, note sequences, sounds, filters, etc. When the sounds are "stolen" (sampled) from already existing music, the authorship question is also supplemented with an ownership issue. Undoubtedly, the more technical sophistication is brought to computer music tools, the more the musical artifact gets disconnected from its creative source.

The work presented in this document is merely focused on composing new music automatically by recycling a preexisting one. We are not concerned with the question of transcription, or separating sources, and we prefer to work directly with rich and complex, polyphonic sounds. This sound collage procedure has recently gotten popular, defining the term "mash-up" music: the practice of making new music out of previously existing recordings. One of the most popular composers of this genre is probably John Oswald, best known for his project "Plunderphonics" [121].

The number of digital music titles available is currently estimated at about 10 million in the western world. This is a large quantity of material to recycle and potentially to add back to the space in a new form. Nonetheless, the space of *all* possible music is finite in its digital form. There are 12,039,300 16-bit audio samples at CD quality in a 4-minute and 33-second song[1], which account for $65{,}536^{12{,}039{,}300}$ options. This is a large amount of music! However, due to limitations of our perception, only a tiny fraction of that space makes any sense to us. The large majority of it sounds essentially like random noise[2]. From the space that makes any musical sense, an even smaller fraction of it is perceived as unique (just-noticeably different from others).

In a sense, by recycling both musical experience and sound material, we can more intelligently search through this large musical space and find more efficiently some of what is left to be discovered. This thesis aims to computationally model the process of creating music using experience from listening to examples. By recycling a database of existing songs, our model aims to compose and perform *new* songs with "similar" characteristics, potentially expanding the space to yet unexplored frontiers. Because it is purely based on the signal content, the system is not able to make qualitative judgments of its own work, but can listen to the results and analyze them in relation to others, as well as recycle that new music again. This unbiased solution models the life cycle of listening, composing, and performing, turning the machine into an active musician, instead of simply an instrument (Figure 1-2).

In this work, we claim the following hypothesis:

> *Analysis and synthesis of musical audio can share a minimal data representation of the signal, acquired through a uniform approach based on perceptual listening and learning.*

In other words, the analysis task, which consists of describing a music signal, is equivalent to a *structured compression* task. Because we are dealing with a perceived signal, the compression is perceptually grounded in order to give the most compact and most meaningful description. Such specific representation is analogous to a music cognition modeling task. The same description is suitable

[1] In reference to Cage's silent piece: 4'33".

[2] We are not referring to heavy metal!

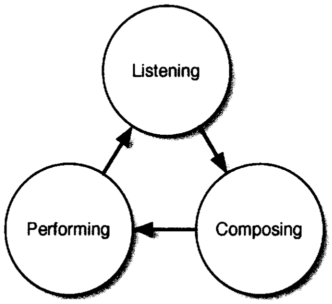


Figure 1-2: Life cycle of the music making paradigm.

for synthesis as well: by reciprocity of the process, and redeployment of the data into the signal domain, we can resynthesize the music in the form of a waveform. We say that the model is *lossy* as it removes information that is not perceptually relevant, but "optimal" in terms of data reduction. Creating new music is a matter of combining multiple representations before resynthesis.

The motivation behind this work is to *personalize* the music experience by seamlessly merging together listening, composing, and performing. Recorded music is a relatively recent technology, which already has found a successor: synthesized music, in a sense, will enable a more intimate listening experience by potentially providing the listeners with precisely the music they want, whenever they want it. Through this process, it is potentially possible for our "metacomposer" to turn listeners—who induce the music—into composers themselves. Music will flow and be live again. The machine will have the capability of monitoring and improving its prediction continually, and of working in communion with millions of other connected music fans.

The next chapter reviews some related works and introduces our framework. Chapters 3 and 4 deal with the machine listening and structure description aspects of the framework. Chapter 5 is concerned with machine learning, generalization, and clustering techniques. Finally, music synthesis is presented through a series of applications including song alignment, music restoration, cross-synthesis, song morphing, and the synthesis of new pieces. This research was implemented within a stand-alone environment called "Skeleton" developed by the author, as described in appendix A. The interested readers may refer to the supporting website of this thesis, and listen to the audio examples that are analyzed or synthesized throughout the document:

`http://www.media.mit.edu/~tristan/phd/`

CHAPTER TWO
Background

> *"When there's something we think could be better, we must make an effort to try and make it better."*
>
> – John Coltrane

Ever since Max Mathews made his first sound on a computer in 1957 at Bell Telephone Laboratories, there has been an increasing appeal and effort for using machines in the disciplines that involve music, whether composing, performing, or listening. This thesis is an attempt at bringing together all three facets by closing the loop that can make a musical system entirely autonomous (Figure 1-2). A complete survey of precedents in each field goes well beyond the scope of this dissertation. This chapter only reviews some of the most related and inspirational works for the goals of this thesis, and finally presents the *framework* that ties the rest of the document together.

2.1 Symbolic Algorithmic Composition

Musical composition has historically been considered at a symbolic, or *conventional* level, where score information (i.e., pitch, duration, dynamic material, and instrument, as defined in the MIDI specifications) is the output of the compositional act. The formalism of music, including the system of musical sounds, intervals, and rhythms, goes as far back as ancient Greeks, to Pythagoras, Ptolemy, and Plato, who thought of music as inseparable from numbers. The automation of composing through formal instructions comes later with the *canonic* composition of the 15th century, and leads to what is now referred to as *algorithmic composition.* Although it is not restricted to computers[1], using algorithmic programming methods as pioneered by Hiller and Isaacson in "Illiac Suite" (1957), or Xenakis in "Atrées" (1962), has "opened the door to new vistas in the expansion of the computer's development as a unique instrument with significant potential" [31].

Computer-generated, automated composition can be organized into three main categories: *stochastic* methods, which use sequences of jointly distributed random variables to control specific decisions (Aleatoric movement); *rule-based* systems, which use a strict grammar and set of rules (Serialism movement); and *artificial intelligence* approaches, which differ from rule-based approaches mostly by their capacity to define their own rules: in essence, to "learn." The latter is the approach that is most significant to our work, as it aims at creating music through *unbiased* techniques, though with intermediary MIDI representation.

Probably the most popular example is David Cope's system called "Experiments in Musical Intelligence" (EMI). EMI analyzes the score structure of a MIDI sequence in terms of recurring patterns (a signature), creates a database of the meaningful segments, and "learns the style" of a composer, given a certain number of pieces [32]. His system can generate compositions with surprising stylistic similarities to the originals. It is, however, unclear how automated the whole process really is, and if the system is able to extrapolate from what it learns.

A more recent system by Francois Pachet, named "Continuator" [122], is capable of learning *live* the improvisation style of a musician who plays on a polyphonic MIDI instrument. The machine can "continue" the improvisation on the fly, and performs autonomously, or under user guidance, yet in the style of its teacher. A particular parameter controls the "closeness" of the generated music, and allows for challenging interactions with the human performer.

2.2 Hybrid MIDI-Audio Instruments

George Lewis, trombone improviser and composer, is a pioneer in building computer programs that create music by interacting with a live performer through acoustics. The so-called "Voyager" software listens via a microphone to his trombone improvisation, and comes to quick conclusions about what was played. It generates a complex response that attempts to make appropriate decisions about melody, harmony, orchestration, ornamentation, rhythm, and silence [103]. In Lewis' own words, "the idea is to get the machine to pay attention to the performer as it composes." As the performer engages in a dialogue,

[1] Automated techniques (e.g., through randomness) have been used for example by Mozart in "Dice Music," by Cage in "Reunion," or by Messiaen in "Mode de valeurs et d'intensités."

the machine may also demonstrate an independent behavior that arises from its own internal processes.

The so-called "Hyperviolin" developed at MIT [108] uses multichannel audio input and perceptually-driven processes (i.e., pitch, loudness, brightness, noisiness, timbre), as well as gestural data input (bow position, speed, acceleration, angle, pressure, height). The relevant but high dimensional data stream unfortunately comes together with the complex issue of *mapping* that data to meaningful synthesis parameters. Its latest iteration, however, features an unbiased and unsupervised learning strategy for mapping timbre to intuitive and perceptual control input (section 2.3).

The piece "Sparkler" (composed by Tod Machover) exploits similar techniques, but for a symphony orchestra [82]. Unlike many previous works where only solo instruments are considered, in this piece a few microphones capture the entire orchestral sound, which is analyzed into perceptual data streams expressing variations in dynamics, spatialization, and timbre. These instrumental sound masses, performed with a certain freedom by players and conductor, drive a MIDI-based *generative algorithm* developed by the author. It interprets and synthesizes complex electronic textures, sometimes blending, and sometimes contrasting with the acoustic input, turning the ensemble into a kind of "Hyperorchestra."

These *audio-driven* systems employ rule-based generative principles for synthesizing music [173][139]. Yet, they differ greatly from *score-following* strategies in their creative approach, as they do not rely on aligning pre-composed material to an input. Instead, the computer program *is* the score, since it describes everything about the musical output, including notes and sounds to play. In such a case, the created music is the result of a compositional act by the programmer. Pushing even further, Lewis contends that:

> ***"[...] notions about the nature and function of music are embedded in the structure of software-based music systems, and interactions with these systems tend to reveal characteristics of the community of thought and culture that produced them. Thus, Voyager is considered as a kind of computer music-making embodying African-American aesthetics and musical practices."*** [103]

2.3 Audio Models

Analyzing the musical content in *audio* signals rather than *symbolic* signals is an attractive idea that requires some sort of perceptual models of listening. Perhaps an even more difficult problem is being able to synthesize meaningful audio signals without intermediary MIDI notation. Most works—often driven by a particular synthesis technique—do not really make a distinction between *sound* and *music*.

CNMAT's Additive Synthesis Tools (CAST) are flexible and generic real-time analysis/resynthesis routines based on sinusoidal decomposition, "Sound Description Interchange Format" (SDIF) content description format [174], and "Open Sound Control" (OSC) communication protocol [175]. The system can analyze, modify, and resynthesize a live acoustic instrument or voice, encouraging a dialogue with the "acoustic" performer. Nonetheless, the synthesized music is controlled by the "electronic" performer who manipulates the interface. As a result, performers remain in charge of the music, while the software generates the sound.

The Spectral Modeling Synthesis (SMS) technique initiated in 1989 by Xavier Serra is a powerful platform for the analysis and resynthesis of monophonic and polyphonic audio [149][150]. Through decomposition into its *deterministic* and *stochastic* components, the software enables several applications, including time scaling, pitch shifting, compression, content analysis, sound source separation, instrument modeling, and timbre morphing.

The Perceptual Synthesis Engine (PSE), developed by the author, is an extension of SMS for monophonic sounds [79][83]. It first decomposes the audio recording into a set of streaming signal coefficients (frequencies and amplitudes of sinusoidal functions) and their corresponding perceptual correlates (instantaneous pitch, loudness, and brightness). It then *learns* the relationship between the two data sets: the high-dimensional signal description, and the low-dimensional perceptual equivalent. The resulting *timbre* model allows for greater control over the sound than previous methods by removing the *time dependency* from the original file[2]. The learning is based on a mixture of Gaussians with local linear models and converges to a unique solution through the Expectation-Maximization (EM) algorithm. The outcome is a highly compact and unbiased synthesizer that enables the same applications as SMS, with intuitive control and no time-structure limitation. The system runs in real time and is driven by audio, such as the acoustic or electric signal of traditional instruments. The work presented in this dissertation is, in a sense, a step towards extending this monophonic timbre model to polyphonic structured music.

Methods based on data-driven *concatenative* synthesis typically discard the notion of analytical transcription, but instead, they aim at generating a *musical surface* (i.e., what is perceived) through a set of compact audio descriptors, and the concatenation of sound samples. The task consists of searching through a sound database for the most relevant segments, and of sequencing the small units granularly, so as to best match the overall *target* data stream. The method was first developed as part of a text-to-speech (TTS) system, which exploits large databases of speech phonemes in order to reconstruct entire sentences [73].

[2] The "re" prefix in *re*synthesis.

Schwarz's "Caterpillar" system [147] aims at synthesizing monophonic musical phrases via the concatenation of instrumental sounds characterized through a bank of descriptors, including: unit descriptors (location, duration, type); source descriptors (sound source, style); score descriptors (MIDI note, polyphony, lyrics); signal descriptors (energy, fundamental frequency, zero crossing rate, cutoff frequency); perceptual descriptors (loudness, sharpness, timbral width); spectral descriptors (centroid, tilt, spread, dissymmetry); and harmonic descriptors (energy ratio, parity, tristimulus, deviation). The appropriate segments are selected through *constraint-solving* techniques, and aligned into a continuous audio stream.

Zils and Pachet's "Musical Mosaicing" [181] aims at generating *music* from arbitrary audio segments. A first application uses a probabilistic generative algorithm to compose the music, and an overlap-add technique for synthesizing the sound. An overall measure of concatenation quality and a constraint-solving strategy for sample selection insures a certain continuity in the stream of audio descriptors. A second application uses a target song as the overall set of constraints instead. In this case, the goal is to replicate an existing audio surface through granular concatenation, hopefully preserving the underlying musical structures (section 6.5).

Lazier and Cook's "MoSievius" system [98] takes up the same idea, and allows for real-time interactive control over the mosaicing technique by fast *sound sieving*: a process of isolating sub-spaces as inspired by [161]. The user can choose input and output signal specifications in real time in order to generate an interactive audio mosaic. Fast time-stretching, pitch shifting, and k-nearest neighbor search is provided. An (optionally pitch-synchronous) overlap-add technique is used for synthesis. Only few or no audio examples of Schwarz's, Zils's, and Lazier's systems are available.

2.4 Music information retrieval

The current proliferation of compressed digital formats, peer-2-peer networks, and online music services is transforming the way we handle music, and increases the need for automatic management technologies. *Music Information Retrieval* (MIR) is looking into describing the *bits* of the digital music in ways that facilitate searching through this abundant world without structure. The signal is typically tagged with additional information called *metadata* (data about the data). This is the endeavor of the MPEG-7 file format, of which the goal is to enable content-based search and novel applications. Still, no commercial use of the format has yet been proposed. In the following paragraphs, we briefly describe some of the most popular MIR topics.

Fingerprinting aims at describing the *audio surface* of a song with a compact representation metrically distant from other songs, i.e., a musical signature. The technology enables, for example, cell-phone carriers or copyright management services to automatically identify audio by comparing unique "fingerprints" extracted live from the audio with fingerprints in a specially compiled music database running on a central server [23].

Query by description consists of querying a large MIDI or audio database by providing qualitative text descriptors of the music, or by "humming" the tune of a song into a microphone (query by humming). The system typically compares the entry with a pre-analyzed database metric, and usually ranks the results by similarity [171][54][26].

Music similarity is an attempt at estimating the closeness of music signals. There are many criteria with which we may estimate similarities, including editorial (title, artist, country), cultural (genre, subjective qualifiers), symbolic (melody, harmony, structure), perceptual (energy, texture, beat), and even cognitive (experience, reference) [167][6][69][9].

Classification tasks integrate similarity technologies as a way to cluster music into a finite set of classes, including genre, artist, rhythm, instrument, etc. [105][163][47]. Similarity and classification applications often face the primary question of defining a *ground truth* to be taken as actual facts for evaluating the results without error.

Thumbnailing consists of building the most "representative" audio summary of a piece of music, for instance by removing the most redundant and least salient sections from it. The task is to detect the boundaries and similarities of large musical structures, such as verses and choruses, and finally assemble them together [132][59][27].

The "Music Browser," developed by Sony CSL, IRCAM, UPF, Fraunhofer, and others, as part of a European effort (Cuidado, Semantic Hi-Fi) is the "first entirely automatic chain for extracting and exploiting musical metadata for browsing music" [124]. It incorporates several techniques for music description and data mining, and allows for a variety of queries based on *editorial* (i.e., entered manually by an editor) or *acoustic metadata* (i.e., the sound of the sound), as well as providing browsing tools and sharing capabilities among users.

Although this thesis deals exclusively with the extraction and use of acoustic metadata, music as a whole cannot be solely characterized by its "objective" content. Music, as experienced by listeners, carries much "subjective" value that evolves in time through communities. *Cultural* metadata attached to music can be extracted online in a textual form through web crawling and natural-language processing [125][170]. Only a combination of these different types of metadata (i.e., acoustic, cultural, editorial) can lead to viable music management and retrieval systems [11][123][169].

2.5 Framework

Much work has already been done under the general paradigm of *analysis/resynthesis*. As depicted in Figure 2-1, the idea is first to break down a sound into some essential, quantifiable components, e.g., amplitude and phase partial coefficients. These are usually altered in some way for applications including time stretching, pitch shifting, timbre morphing, or compression. Finally, the transformed parameters are reassembled into a new sound through a synthesis procedure, e.g., additive synthesis. The *phase vocoder* [40] is an obvious example of this procedure where the new sound is directly an artifact of the old sound via some describable transformation.

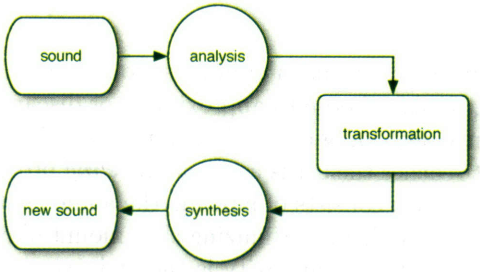


Figure 2-1: Sound analysis/resynthesis paradigm.

2.5.1 Music analysis/resynthesis

The mechanism applies well to the modification of audio signals in general, but is generally blind[3] regarding the embedded musical content. We introduce an extension of the sound analysis/resynthesis principle for *music* (Figure 2-2). Readily, our music-aware analysis/resynthesis approach enables *higher-level* transformations independently of the sound content, including beat matching, music morphing, music cross-synthesis, music similarities.

The *analysis* framework characterizing this thesis work is the driving force of the *synthesis* focus of section 6, and it can be summarized concisely by the following quote:

> *"Everything should be made as simple as possible but not simpler."*
>
> – Albert Einstein

[3] Perhaps we should say deaf...

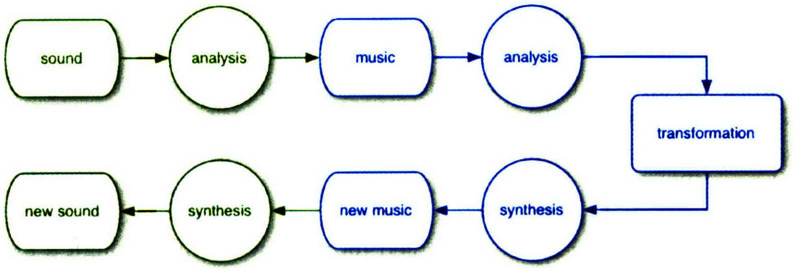


Figure 2-2: Music analysis/resynthesis procedure, including sound analysis into music features, music analysis, transformation, music synthesis, finally back into sound through sound synthesis.

We seek to simplify the information of interest to its minimal form. Depending on the application, we can choose to approximate or discard some of that information, consequently degrading our ability to resynthesize. Reconstructing the original signal as it reaches the ear is a *signal modeling* problem. If the source is available though, the task consists of labeling the audio as it is being perceived: a *perception modeling* problem. Optimizing the amount of information required to describe a signal of given complexity is the endeavor of information theory [34]: here suitably *perceptual information theory.*

Our current implementation uses *concatenative synthesis* for resynthesizing rich sounds without having to deal with signal modeling (Figure 2-3). Given the inherent granularity of concatenative synthesis, we safely reduce the description further, resulting into our final *acoustic metadata*, or music characterization.

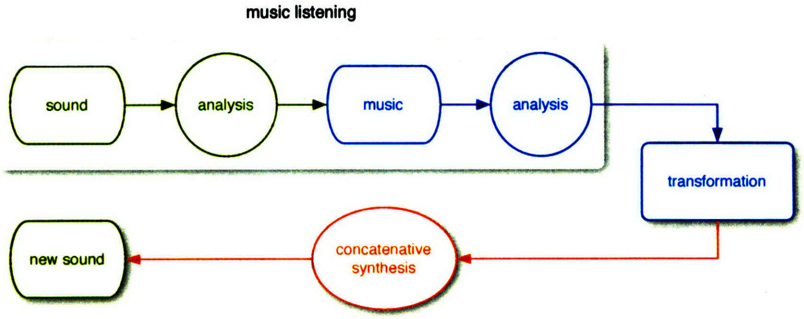


Figure 2-3: In our music analysis/resynthesis implementation, the whole synthesis stage is a simple concatenative module. The analysis module is referred to as *music listening* (section 3).

We extend the traditional *music listening* scheme as described in [142] with a *learning* extension to it. Indeed, listening cannot be disassociated from learning. Certain problems such as, for example, downbeat prediction, cannot be fully solved without this part of the framework (section 5.3).

Understanding the mechanisms of the brain, in particular the auditory path, is the ideal basis for building perceptual models of music cognition. Although particular models have great promises [24][25], it is still a great challenge to make these models work in real-world applications today. However, we can attempt to mimic some of the most basic functionalities of music perception, and build a *virtual listener* that will process, interpret, and describe music signals much as humans do; that is, primarily, from the *ground-up*.

The model depicted below, inspired by some empirical research on human listening and learning, may be considered the first *practical* attempt at implementing a "music cognition machine." Although we implement most of the music listening through deterministic signal processing algorithms, we believe that the whole process may eventually be solved via statistical learning approaches [151]. But, since our goal is to make music, we favor practicality over a truly uniform approach.

2.5.2 Description

We propose a four-building-block diagram, where each block represents a signal *reduction* stage of another. Information flows from left to right between each stage and always corresponds to a simpler, more abstract, and slower-rate signal (Figure 2-4). Each of these four successive stages—hearing, feature extraction, short-term memory, and long-term memory—embodies a different concept, respectively: filtering, symbolic representation, time dependency, and storage. The knowledge is re-injected to some degree through all stages via a top-down feedback loop.

music cognition

The first three blocks roughly represent what is often referred to as *listening*, whereas the last three blocks represent what is often referred to as *learning*. The interaction between music listening and music learning (the overlapping area of our framework schematic) is what we call *music cognition*, where most of the "interesting" musical phenomena occur. Obviously, the boundaries of music cognition are not very well defined and the term should be used with great caution. Note that there is more to music cognition than the signal path itself. Additional external influences may act upon the music cognition experience, including vision, culture, emotions, etc., but these are not represented here.

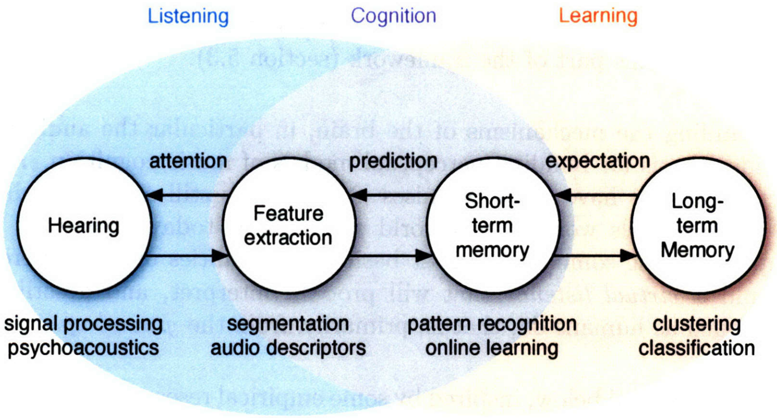


Figure 2-4: Our music signal analysis framework. The data flows from left to right and is reduced in each stage. The first stage is essentially an auditory filter where the output data describes the audio surface. The second stage, analyzes that audio surface in terms of perceptual features, which are represented in the form of a symbolic "musical-DNA" stream. The third stage analyzes the time component of the streaming data, extracting redundancies, and patterns, and enabling prediction-informed decision making. Finally, the last stage stores and compares macrostructures. The first three stages represent *listening*. The last three stages represent *learning*. The overlapping area may represent *musical cognition*. All stages feedback to each other, allowing for example "memories" to alter our listening perception.

hearing

This is a filtering stage, where the output signal only carries what we hear. The ear being physiologically limited, only a portion of the original signal is actually heard (in terms of coding, this represents less than 10% of the incoming signal). The resulting signal is presented in the form of an *auditory spectrogram*, where what appears in the time-frequency display corresponds strictly to what is being heard in the audio. This is where we implement *psychoacoustics* as in [183][116][17]. The analysis period here is on the order of 10 ms.

feature extraction

This second stage converts the auditory signal into a symbolic representation. The output is a stream of symbols describing the music (a sort of "musical-DNA" sequence). This is where we could implement *sound source separation.* Here we may extract perceptual features (more generally audio descriptors) or we describe the signal in the form of a *musical surface.* In all cases, the output of this stage is a much more compact characterization of the musical content. The analysis period is on the order of 100 ms.

short-term memory

The streaming music DNA is analyzed in the time-domain during this third stage. The goal here is to detect patterns and redundant information that may lead to certain expectations, and to enable prediction. Algorithms with a built-in temporal component, such as symbolic learning, pattern matching, dynamic programming or hidden Markov models are especially applicable here [137][89][48]. The analysis period is on the order of 1 sec.

long-term memory

Finally, this last stage clusters the macro information, and classifies the analysis results for long-term learning, i.e., storage memory. All clustering techniques may apply here, as well as regression and classification algorithms, including mixture of Gaussians, artificial neural networks, or support vector machines [42][22][76]. The analysis period is on the order of several seconds or more.

feedback

For completeness, all stages must feedback to each other. Indeed, our prior knowledge of the world (memories and previous experiences) alters our listening experience and general musical perception. Similarly, our short-term memory (pattern recognition, beat) drives our future prediction, and finally these may direct our attention (section 5.2).

2.5.3 Hierarchical description

Interestingly, this framework applies nicely to the metrical analysis of a piece of music. By analogy, we can describe the music locally by its instantaneous sound, and go up in the hierarchy through metrical grouping of sound segments, beats, patterns, and finally larger sections. Note that the length of these audio fragments coincides roughly with the analysis period of our framework (Figure 2-5).

Structural hierarchies [112], which have been studied in the frequency domain (relationship between notes, chords, or keys) and the time domain (beat, rhythmic grouping, patterns, macrostructures), reveal the intricate complexity and interrelationship between the various components that make up music. Deliège [36] showed that listeners tend to prefer grouping rules based on attack and timbre over other rules (i.e., melodic and temporal). Lerdahl [101] stated that music structures could not only be derived from pitch and rhythm hierarchies, but also from timbre hierarchies. In *auditory scene analysis*, by which humans build mental descriptions of complex auditory environments, abrupt events represent important sound source-separation cues [19][20]. We choose to first detect sound events and *segment* the audio in order to facilitate its analysis, and refine

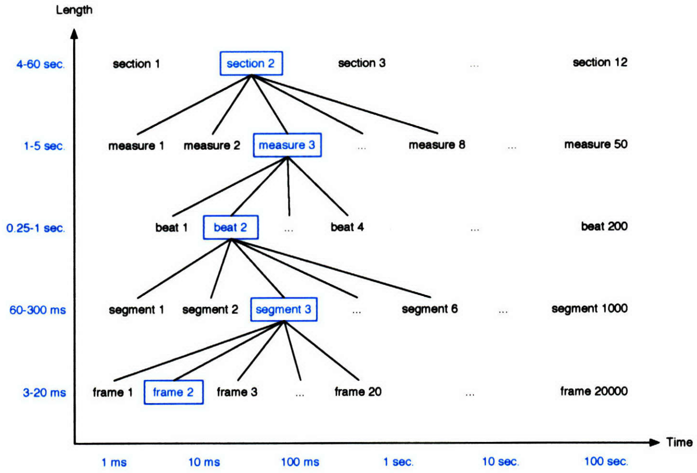


Figure 2-5: Example of a song decomposition in a tree structure.

the description of music. This is going to be the recurrent theme throughout this document.

2.5.4 Meaningful sound space

Multidimensional scaling (MDS) is a set of data analysis techniques that display the structure of distance-like data as a geometrical picture, typically into a low dimensional space. For example, it was shown that instruments of the orchestra could be organized in a *timbral space* of three main dimensions [66][168] loosely correlated to brightness, the "bite" of the attack, and the spectral energy distribution. Our goal is to extend this principle to all possible sounds, including polyphonic and multitimbral.

Sound segments extracted from a piece of music may be represented by data *points* scattered around a multidimensional space. The music structure appears as a *path* in the space (Figure 2-6). Consequently, musical patterns materialize literally into geometrical loops. The concept is simple, but the outcome may turn out to be powerful if one describes a complete music catalogue within that common space. Indeed, the boundaries of the space and the dynamics within it determine the extent of knowledge the computer may acquire: in a sense, its *influences*. Our goal is to *learn* what that space looks like, and find meaningful ways to navigate through it.

Obviously, the main difficulty is to define the similarity of sounds in the first place. This is developed in section 4.4. We also extend the MDS idea to other scales of analysis, i.e., beat, pattern, section, and song. We propose a three-way hierarchical description, in terms of pitch, timbre, and rhythm. This is the main

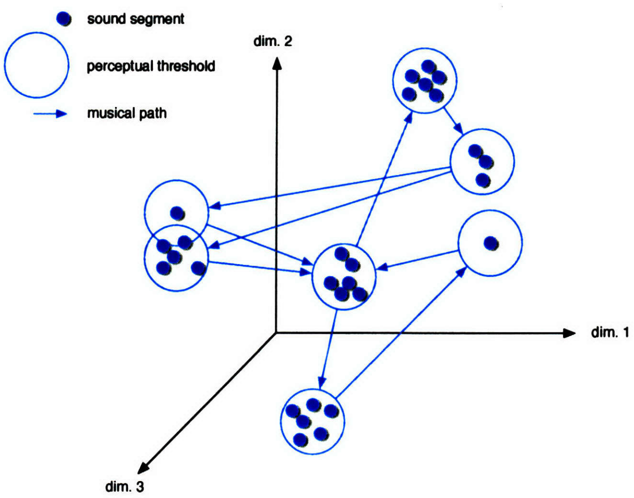


Figure 2-6: Geometrical representation of a song in a perceptual space (only 3 dimensions are represented). The sound segments are described as data points in the space, while the music structure is a path throughout the space. Patterns naturally correspond to geometrical loops. The perceptual threshold here is an indication of the ear resolution. Within its limits, sound segments sound similar.

topic of chapter 4. Depending on the application, one may project the data onto one of these musical dimensions, or combine them by their significance.

2.5.5 Personalized music synthesis

The ultimate goal of this thesis is to characterize a *musical space* given a large corpus of music titles by only considering the acoustic signal, and to propose novel "active" listening strategies through the automatic generation of new music, i.e., a way to navigate freely through the space. For the listener, the system is the "creator" of personalized music that is derived from his/her own song library. For the machine, the system is a synthesis algorithm that manipulates and combines similarity metrics of a highly constrained sound space.

By only providing *machine listening* (chapter 3) and *machine learning* (chapter 5) primitive technologies, it is our goal to build a *bias-free* system that learns the structure of particular music by only listening to song examples. By considering the structural content of music, our framework enables novel transformations, or *music processing*, which goes beyond traditional audio processing.

So far, music-listening systems have been implemented essentially to query music information [123]. Much can be done on the generative side of music management through acoustic analysis. In a way, our framework elevates "audio" signals to the rank of “music” signals, leveraging music cognition, and enabling various applications as described in section 6.

CHAPTER THREE
Music Listening

"My first relationship to any kind of musical situation is as a listener."

– Pat Metheny

Music listening [68] is concerned with the understanding of how humans perceive music. As modelers, our goal is to implement algorithms known as *machine listening*, capable of mimicking this process. There are three major machine-listening approaches: the *physiological* approach, which attempts to model the neurophysical mechanisms of the hearing system; the *psychoacoustic* approach, rather interested in modeling the effect of the physiology on perception; and the *statistical* approach, which models mathematically the reaction of a sound input to specific outputs. For practical reasons, this chapter presents a psychoacoustic approach to music listening.

3.0.6 Anatomy

The hearing system is physiologically limited. The torso, head, and outer ear filter the sound field (mostly below 1500 Hz) through shadowing and reflection. The *outer ear* canal is about 2-cm long, which corresponds to a quarter of the wavelength of frequencies near 4000 Hz, and emphasizes the ear sensitivity to those frequencies. The *middle ear* is a transducer that converts oscillations in the air into oscillations in the *inner ear*, which contains fluids. To avoid large losses of energy through reflection, impedance matching is achieved by a mechanical lever system—eardrum, malleus, incus, stapes, and oval window, as in Figure 3-1—that reaches an almost perfect match around 1000 Hz.

Along the basilar membrane, there are roughly 3000 inner *hair cells* arranged in a regular geometric pattern. Their vibration causes ionic flows that lead to the "firing" of short-duration electrical pulses (the language of the brain) in the nerve fibers connected to them. The entire flow of information runs from the inner ear through approximately 30,000 afferent nerve fibers to reach the midbrain, thalamus, and finally the temporal lobe of the cerebral cortex where is is finally perceived as *sound*. The nature of the central auditory processing is, however, still very much unclear, which mainly motivates the following psychophysical approach [183].

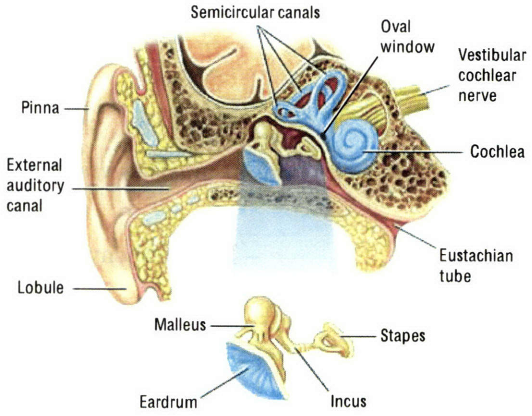


Figure 3-1: Anatomy of the ear. The middle ear is essentially a transducer that converts air oscillations in the outer ear (on the left) into fluid oscillations in the inner ear (on the right). It is depicted with greater details in the bottom drawing. The vestibular cochlear nerves connect the cochlea with the auditory processing system of the brain. Image from [44].

3.0.7 Psychoacoustics

Psychoacoustics is the study of the subjective human perception of sounds. It connects the physical world of sound vibrations in the air to the perceptual world of things we actually hear when we listen to sounds. It is not directly concerned with the physiology of the hearing system as discussed earlier, but rather with its effect on listening perception. This is found to be the most practical and robust approach to an application-driven work. This chapter is about modeling our perception of music through psychoacoustics. Our model is *causal*, meaning that it does not require knowledge about the future, and can be implemented both in real time, and faster than real time. A good review of reasons that motivate and inspire this approach can also be found in [142].

Let us begin with a monophonic audio signal of arbitrary length and sound quality. Since we are only concerned with the human appreciation of music, the signal may have been formerly compressed, filtered, or resampled. The music can be of any kind: we have tested our system with excerpts taken from jazz, classical, funk, electronic, rock, pop, folk and traditional music, as well as speech, environmental sounds, and drum loops.

3.1 Auditory Spectrogram

The goal of our auditory spectrogram is to convert the time-domain waveform into a reduced, yet perceptually meaningful, time-frequency representation. We seek to remove the information that is the least critical to our hearing sensation while retaining the most important parts, therefore reducing signal complexity without perceptual loss. The MPEG1 audio layer 3 (MP3) codec [18] is a good example of an application that exploits this principle for compression purposes. Our primary interest here is understanding our perception of the signal rather than resynthesizing it, therefore the reduction process is sometimes simplified, but also extended and fully parametric in comparison with usual perceptual audio coders.

3.1.1 Spectral representation

First, we apply a standard Short-Time Fourier Transform (STFT) to obtain a standard spectrogram. We experimented with many window types and sizes, which did not have a significant impact on the final results. However, since we are mostly concerned with timing accuracy, we favor short windows (e.g., 12-ms Hanning), which we compute every 3–6 ms (i.e., every 128–256 samples at 44.1 KHz). The Fast Fourier Transform (FFT) is zero-padded up to 46 ms to gain additional interpolated frequency bins. We calculate the power spectrum and scale its amplitude axis to decibels (dB SPL, a measure of sound pressure level) as in the following equation:

$$I_i(dB) = 20\ \log_{10}\left(\frac{I_i}{I_0}\right) \tag{3.1}$$

where $i > 0$ is an index of the power-spectrum bin of intensity I, and I_0 is an arbitrary threshold of hearing intensity. For a reasonable tradeoff between dynamic range and resolution, we choose $I_0 = 60$, and we clip sound pressure levels below -60 dB. The threshold of hearing is in fact frequency-dependent and is a consequence of the outer and middle ear response.

3.1.2 Outer and middle ear

As described earlier, physiologically the outer and middle ear have a great implication on the overall frequency response of the ear. A transfer function was proposed by Terhardt in [160], and is defined in decibels as follows:

$$A_{dB}(f_{KHz}) = -3.64\ f^{-0.8} + 6.5\ \exp\big(-0.6\ (f-3.3)^2\big) - 10^{-3}\ f^4 \qquad (3.2)$$

As depicted in Figure 3-2, the function is mainly characterized by an attenuation in the lower and higher registers of the spectrum, and an emphasis around 2–5 KHz, interestingly where much of the speech information is carried [136].

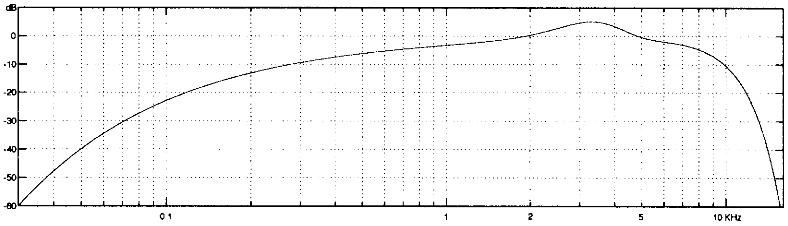


Figure 3-2: Transfer function of the outer and middle ear in decibels, as a function of logarithmic frequency. Note the ear sensitivity between 2 and 5 KHz.

3.1.3 Frequency warping

The inner ear (cochlea) is shaped like a 32 mm long snail and is filled with two different fluids separated by the *basilar membrane*. The oscillation of the oval window takes the form of a traveling wave which moves along the basilar membrane. The mechanical properties of the cochlea (wide and stiff at the base, narrower and much less stiff at the tip) act as a *cochlear filterbank*: a roughly logarithmic decrease in bandwidth (i.e., constant-Q on a logarithmic scale) as we move linearly away from the cochlear opening (the oval window).

The difference in frequency between two pure tones by which the sensation of "roughness" disappears and the tones sound smooth is known as the *critical band*. It was found that at low frequencies, critical bands show an almost constant width of about 100 Hz, while at frequencies above 500 Hz, their bandwidth is about 20% of the center frequency. A *Bark* unit was defined and led to the so-called *critical-band rate scale*. The spectrum frequency f is warped to the Bark scale $z(f)$ as in equation (3.3) [183]. An Equivalent Rectangular Band-

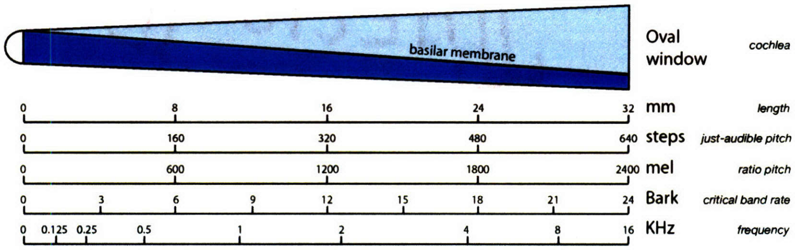


Figure 3-3: Different scales shown in relation to the unwound cochlea. *Mel* in particular is a logarithmic scale of frequency based on human pitch perception. Note that all of them are on a linear scale except for frequency. Tip is shown on the left and base on the right.

width (ERB) scale was later introduced by Moore and is shown in comparison with the Bark scale in figure 3-4 [116].

$$z(f) = 13\ \arctan(0.00076f) + 3.5\ \arctan\left((f/7500)^2\right) \qquad (3.3)$$

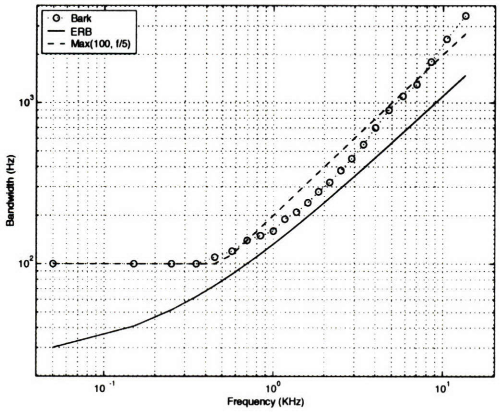


Figure 3-4: Bark critical bandwidth and ERB as a function of frequency. The rule-of-thumb Bark-scale approximation is also plotted (Figure adapted from [153]).

The effect of warping the power spectrum to the Bark scale is shown in Figure 3-5 for white noise, and for a pure tone sweeping linearly from 20 to 20K Hz. Note the non-linear auditory distortion of the frequency (vertical) axis.

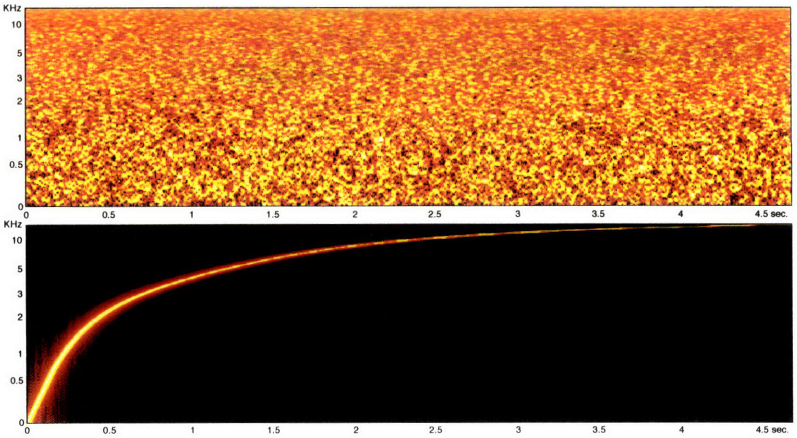


Figure 3-5: Frequency warping onto a Bark scale for [top] white noise; [bottom] a pure tone sweeping linearly from 20 to 20K Hz.

3.1.4 Frequency masking

Simultaneous masking is a property of the human auditory system where certain *maskee* sounds disappear in the presence of so-called *masker* sounds. Masking in the frequency domain not only occurs within critical bands, but also spreads to neighboring bands. Its simplest approximation is a triangular function with slopes +25 dB/Bark and -10 dB/Bark (Figure 3-6), where the lower frequencies have a stronger masking influence on higher frequencies than vice versa [146]. A more refined model is highly non-linear and depends on both frequency and amplitude. Masking is the most powerful characteristic of modern *lossy* coders: more details can be found in [17]. A non-linear *spreading function* as found in [127] and modified by Lincoln in [104] is:

$$SF(z) = (15.81 - i) + 7.5(z + 0.474) - \ (17.5 - i)\sqrt{1 + (z + 0.474)^2} \qquad (3.4)$$

$$\begin{array}{rcl} \text{where } i & = & \min\big(5 \cdot PS(f) \cdot BW(f), 2.0\big) \\ BW(f) & = & \begin{cases} 100 & \text{for } f < 500 \\ 0.2f & \text{for } f \geq 500 \end{cases} \\ PS & \text{is} & \text{the power spectrum,} \\ \text{and } z & \text{is} & \text{defined in equation 3.3.} \end{array} \qquad (3.5)$$

Instantaneous masking was essentially defined through experimentation with pure tones and narrow-band noises [50][49]. Integrating spreading functions in the case of complex tones is not very well understood. To simplify, we compute the full spectral mask through series of individual partials.

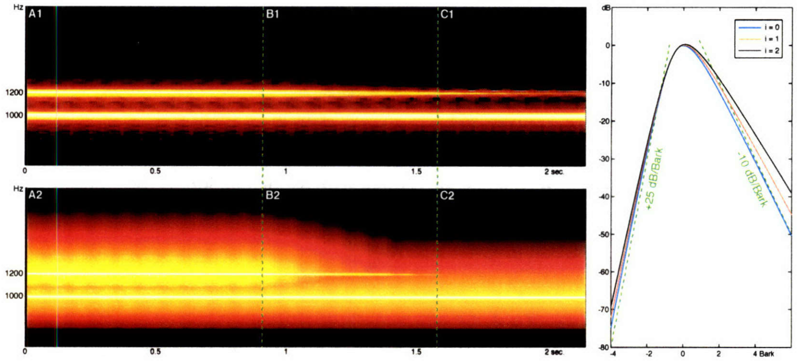


Figure 3-6: [right] Spectral masking curves in the Bark scale as in reference [104], and its approximation (dashed-green). [left] The effect of frequency masking is demonstrated with two pure tones at 1000 and 1200 Hz. The two Bark spectrograms are zoomed around the frequency range of interest. The top one is raw. The bottom one includes frequency masking curves. In zone A, the two sinusoids are equally loud. In zone B and C, the amplitude of the tone at 1200 Hz is decreased exponentially. Note that in zone C1 the tone at 1200 Hz is clearly visible, while in zone C2, it entirely disappears under the masker, which makes it inaudible.

3.1.5 Temporal masking

Another perceptual phenomenon that we consider as well is *temporal masking*. As illustrated in Figure 3-7, there are two types of temporal masking besides simultaneous masking: pre-masking and post-masking. Pre-masking is quite unexpected and not yet conclusively researched, but studies with noise bursts revealed that it lasts for about 20 ms [183]. Within that period, sounds softer than the masker are typically not audible. We do not implement it since signal-windowing artifacts have a similar smoothing effect. However, post-masking is a kind of "ringing" phenomenon which lasts for almost 200 ms. We convolve the envelope of each frequency band with a 200-ms half-Hanning (raised cosine) window. This stage induces smoothing of the spectrogram, while preserving attacks. The effect of temporal masking is depicted in Figure 3-8 for various sounds, together with their *loudness* curve (more on loudness in section 3.2).

The temporal masking effects have important implications on the perception of rhythm. Figure 3-9 depicts the relationship between *subjective* and *physical* duration of sound events. The physical duration of the notes gives an incorrect estimation of the rhythm (in green), while if processed through a psychoacoustic

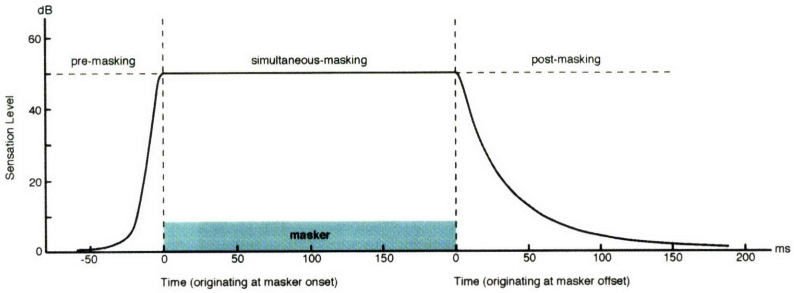


Figure 3-7: Schematic drawing of temporal masking, including pre-masking, simultaneous masking, and post-masking. Note that post-masking uses a different time origin.

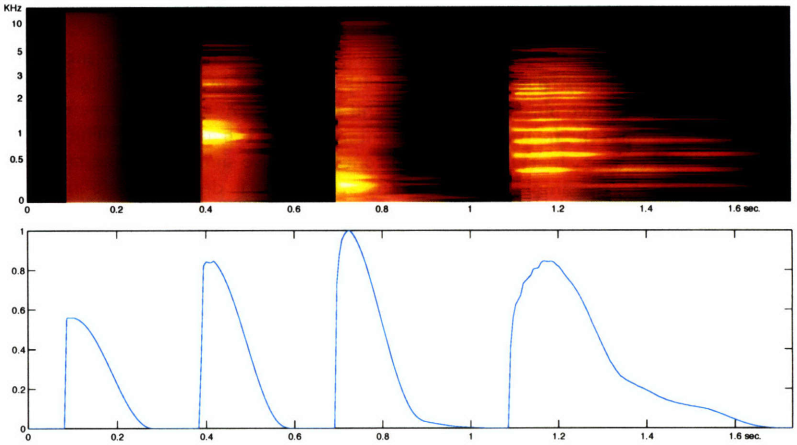


Figure 3-8: Bark spectrogram of four sounds with temporal masking: a digital click, a clave, a snare drum, and a staccato violin sound. Note the 200-ms smoothing effect in the loudness curve.

model, the rhythm estimation is correct (in blue), and corresponds to what the performer and audience actually hear.

3.1.6 Putting it all together

Finally, we combine all the preceding pieces together, following that order, and build our *hearing* model. Its outcome is what we call the *audio surface*. Its graphical representation, the *auditory spectrogram*, merely approximates a "what-you-see-is-what-you-hear" type of spectrogram, meaning that the "just visible" in the time-frequency display corresponds to the "just audible" in the underlying sound. Note that we do not understand *music* yet, but only *sound*.

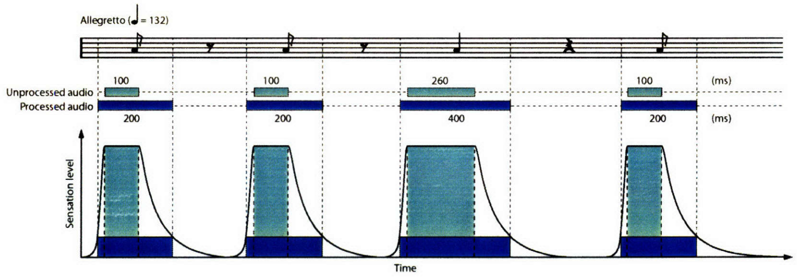


Figure 3-9: Importance of *subjective duration* for the estimation of rhythm. A rhythmic pattern performed by a musician (see staff) results in a subjective sensation (blue) much different from the physical reality (green)—the physical duration of the audio signal. A temporal model is implemented for accurate duration analysis and correct estimation of rhythm.

Figure 3-10 displays the audio surface of white noise, a sweeping pure tone, four distinct sounds, and a real-world musical excerpt.

3.2 Loudness

The area below the audio surface (the zone covered by the mask) is called the *excitation* level, and minus the area covered by the threshold in quiet, leads to the sensation of *loudness*: the subjective judgment of the intensity of a sound. It is derived easily from our auditory spectrogram by adding the amplitudes across all frequency bands:

$$\mathrm{L_{dB}(t)} = \frac{\sum_{k=1}^{N} E_k(t)}{N} \tag{3.6}$$

where E_k is the amplitude of frequency band k of total N in the auditory spectrogram. Advanced models of loudness by Moore and Glasberg can be found in [117][57]. An example is depicted in Figure 3-11.

3.3 Timbre

Timbre, or "tone color," is a relatively poorly defined perceptual quality of sound. The American Standards Association (ASA) defines timbre as "that attribute of sensation in terms of which a listener can judge that two sounds having the same loudness and pitch are dissimilar" [5]. In music, timbre is the

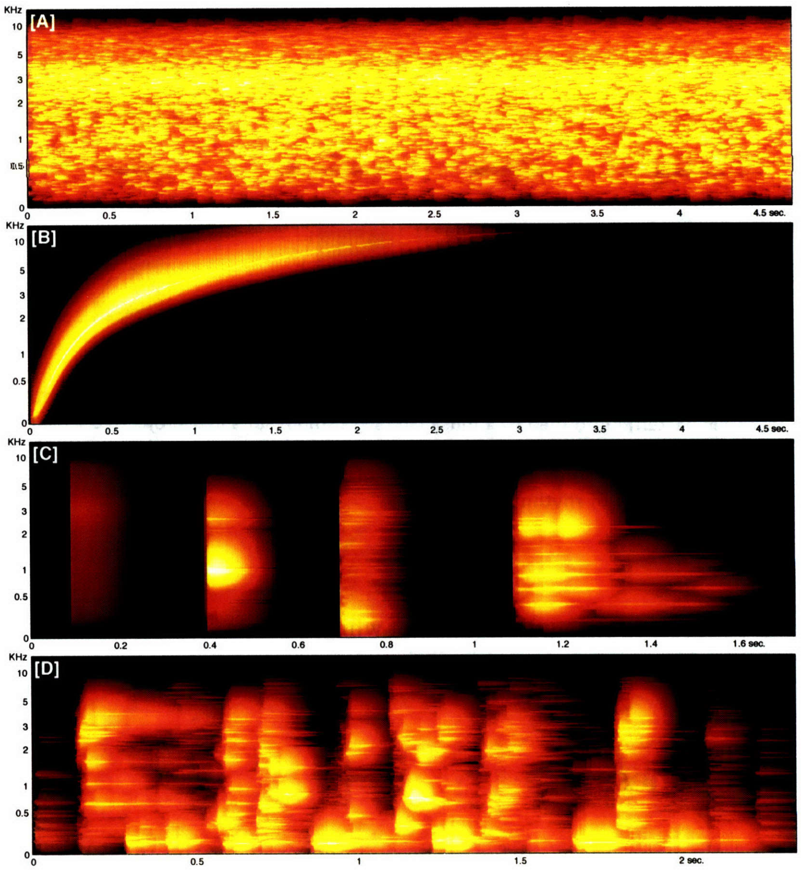


Figure 3-10: Auditory spectrogram of [A] white noise; [B] a pure tone sweeping linearly from 20 to 20K Hz; [C] four short sounds, including a digital click, a clave, a snare drum, and a staccato violin sound; [D] a short excerpt of James Brown's "Sex machine."

quality of a musical note that distinguishes musical instruments. It was shown by Grey [66] and Wessel [168] that important timbre characteristics of the orchestral sounds are attack quality (temporal envelope), spectral flux (evolution of the spectral distribution over time), and brightness (spectral centroid).

In fact, this psychoacoustician's waste basket includes so many factors that the latest trend for characterizing timbre, sounds, and other high-level musical attributes consists of using a battery of so-called low-level *audio descriptors* (LLD), as specified for instance in the MPEG7 standardization format [118]. Those can be organized in various categories including *temporal descriptors* computed from the waveform and its envelope, *energy descriptors* referring to various energy measurements of the signal, *spectral descriptors* computed from the STFT, *harmonic descriptors* computed from the sinusoidal harmonic modeling of the signal, and *perceptual descriptors* computed using a model of the human hearing process [111][133][147].

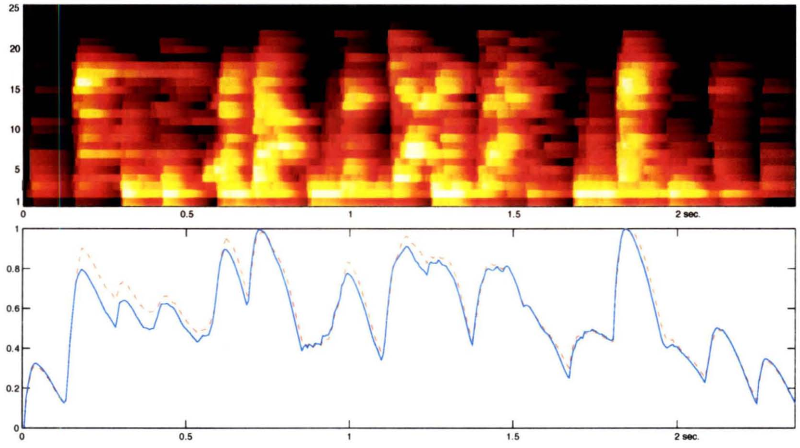


Figure 3-11: 25 critical Bark bands for the short excerpt of James Brown's "Sex machine" as in Figure 3-10, and its corresponding loudness curve with 256 frequency bands (dashed-red), or only 25 critical bands (blue). The measurement of loudness through critical band reduction is fairly reasonable, and computationally much more efficient.

The next step typically consists of finding the combination of those LLDs, which hopefully best matches the perceptive target [132]. An original approach by Pachet and Zils substitutes the basic LLDs by primitive operators. Through genetic programming, the Extraction Discovery System (EDS) aims at composing these operators automatically, and discovering signal-processing functions that are "locally optimal" for a given descriptor extraction task [126][182].

Rather than extracting specific high-level musical descriptors, or classifying sounds given a specific "taxonomy" and arbitrary set of LLDs, we aim at representing the timbral space of complex polyphonic signals with a meaningful, yet generic description. Psychoacousticians tell us that the critical band can be thought of as a frequency-selective *channel* of psychoacoustic processing. For humans, only 25 critical bands cover the full spectrum (via the Bark scale). These can be regarded as a reasonable and perceptually grounded description of the instantaneous timbral envelope. An example of that spectral reduction is given in Figure 3-11 for a rich polyphonic musical excerpt.

3.4 Onset Detection

Onset detection (or segmentation) is the means by which we can divide the musical signal into smaller units of sound. This section only refers to the most atomic level of segmentation, that is the smallest rhythmic events possibly found in music: individual notes, chords, drum sounds, etc. Organized in time, a sequence of sound segments infers our perception of *music*. Since we are not concerned with sound source separation, a segment may represent a rich and complex polyphonic sound, usually short. Other kinds of segmentations (e.g., voice, chorus) are specific aggregations of our minimal segments which require source recognition, similarity, or continuity procedures.

3.4.1 Prior approaches

Many applications, including the holy-grail *transcription* task, are primarily concerned with detecting onsets in a musical audio stream. There has been a variety of approaches including finding abrupt changes in the energy envelope [38], in the phase content [10], in pitch trajectories [138], in audio similarities [51], in autoregressive models [78], in spectral frames [62], through a multifeature scheme [162], through ICA and hidden Markov modeling [1], and through neural networks [110]. Klapuri [90] stands out for using psychoacoustic knowledge; this is the solution proposed here as well.

3.4.2 Perceptually grounded approach

We define a sound segment by its onset and offset boundaries. It is assumed perceptually "meaningful" if its timbre is consistent, i.e., it does not contain any noticeable abrupt changes. Typical segment onsets include abrupt loudness, pitch or timbre variations. All of these events translate naturally into an abrupt spectral variation in the auditory spectrogram.

We convert the auditory spectrogram into an *event detection function* by calculating the first-order difference function of each spectral band, and by summing across channels. The resulting signal reveals *peaks* that correspond to onset transients (Figure 3-12, pane 4). Transients within a 50-ms window typically *fuse* perceptually into a single event [155]. We model fusion by convolving the raw event detection signal with a Hanning window. Best results (i.e., with segments greater than 50 ms) are obtained with a 150-ms window. The filtering generates a smooth function now appropriate for the *peak-picking* stage. Unlike traditional methods that usually rely heavily on designing an adaptive threshold mechanism, we can simply select the local maxima (Figure 3-12, pane 5). We may reject the flattest peaks through threshold as well, but this stage and settings are not critical.

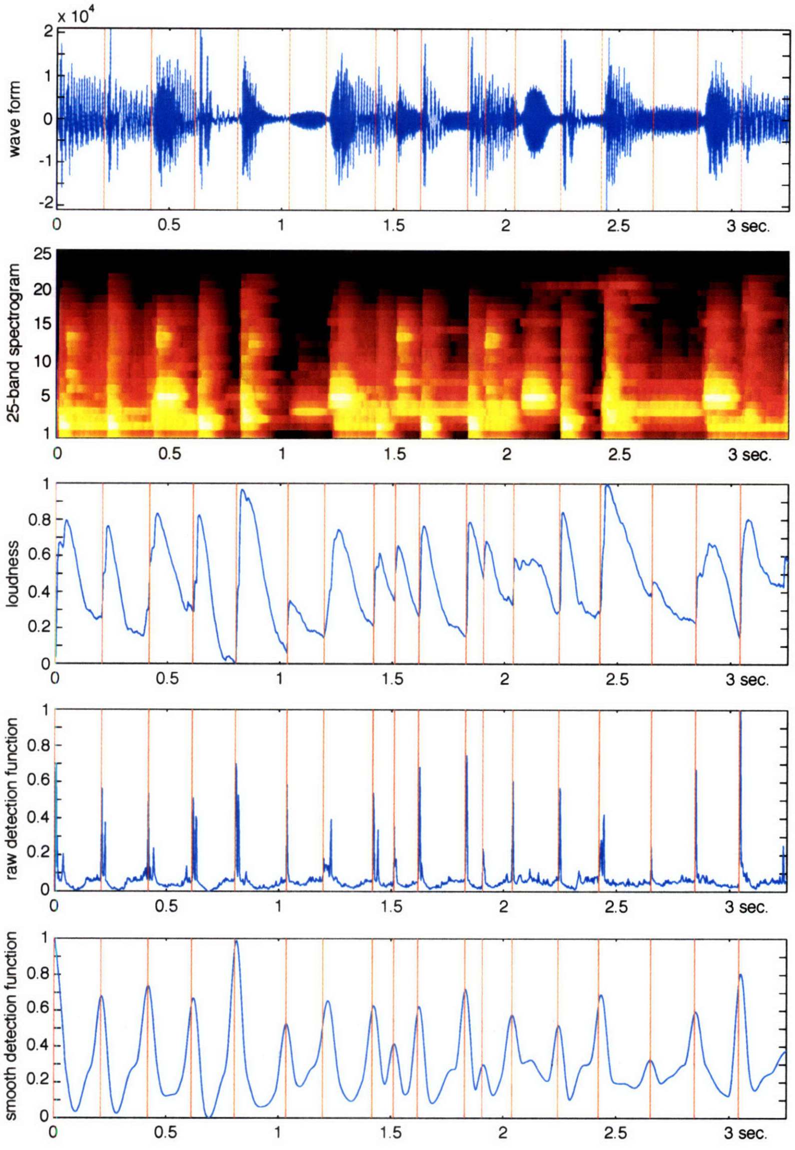


Figure 3-12: A short 3.25 sec. excerpt of "Watermelon man" by Herbie Hancock. [1] waveform (blue) and segment onsets (red); [2] auditory spectrogram; [3] loudness function; [4] raw event detection function; [5] smoothed detection function.

Since we are concerned with reusing the audio segments for synthesis, we refine the onset location by analyzing it in relation with its corresponding *loudness function*. An onset generally occurs with an increase variation in loudness. To retain the entire attack, we seek the previous local minimum in the loudness signal (in general a small time shift of at most 20 ms), which corresponds to the softest pre-onset moment, that is the best time to cut. Finally, we look within the corresponding waveform, and search for the closest zero-crossing, with an arbitrary but consistent choice of direction (e.g., from negative to positive). This stage is important to ensure signal continuity at synthesis.

3.4.3 Tatum grid

Segment sequencing is the reason for musical perception, and the inter-onset interval (IOI) is at the origin of the metrical-structure perception [74]. The *tatum*, named after jazz pianist "Art Tatum" in [12] can be defined as the lowest regular pulse train that a listener intuitively infers from the timing of perceived musical events: a time quantum. It is roughly equivalent to the time division that most highly coincides with note onsets: an equilibrium between 1) how well a regular grid explains the onsets, and 2) how well the onsets explain the grid.

The tatum is typically computed via a time-varying IOI histogram [64], with an exponentially decaying window for past data, enabling the tracking of accelerandos and ritardandos [148]. The *period* is found by calculating the greatest common divisor (GCD) integer that best estimates the histogram harmonic structure, or by means of a two-way mismatch error procedure as originally proposed for the estimation of the fundamental frequency in [109], and applied to tatum analysis in [65][67]. Two error functions are computed: one that illustrates how well the grid elements of period candidates explain the peaks of the measured histogram; another one illustrates how well the peaks explain the grid elements. The TWM error function is a linear combination of these two functions. *Phase* is found in a second stage, for example through circular mean in a grid-to-onset alignment procedure as in [148].

Instead of a discrete IOI histogram, our method is based on a moving autocorrelation computed on the smooth event-detection function as found in section 3.4. The window length is chosen adaptively from the duration of x past segments to ensure rough salience stability in the first-peak estimation of the autocorrelation (e.g., $x \approx 15$). The autocorrelation is only partially calculated since we are guaranteed to find a peak in the $\pm(100/x)\%$ range around its center. The first peak gives the approximate tatum period. To refine that estimation, and detect the phase, we run a search through a set of *templates*.

Templates are patterns or filters that we aim to align against the signal. We pre-compute dozens of regular pulse trains in the range 1.5–15 Hz through a series of click trains convolved with a Hanning window: the same used to smooth the detection function in section 3.4.2. To account for memory fading, we shape the templates with a half-raised cosine of several seconds, e.g., 3–6 sec. The templates are finally normalized by their total energy (Figure 3-13, left). At a given estimation time, the optimal template is the one with highest energy when cross-correlated with the current smoothed detection function. For maximum efficiency, we only estimate templates within the range ±10% of our rough period estimation. We limit the cross-correlation lag search for the optimal template, to only the tatum period length $\Delta\tau$, since it contains the peak that will account for phase offset ϕ and allows us to predict the next tatum location: $\tau[i+1] = \tau[i] + \Delta\tau[i] - c \cdot \phi[i]$ where c is a smoothing coefficient and $\phi[i] \in [-\Delta\tau[i]/2, +\Delta\tau[i]/2[$. The system quickly phase locks and is efficiently updated at tatum-period rate.

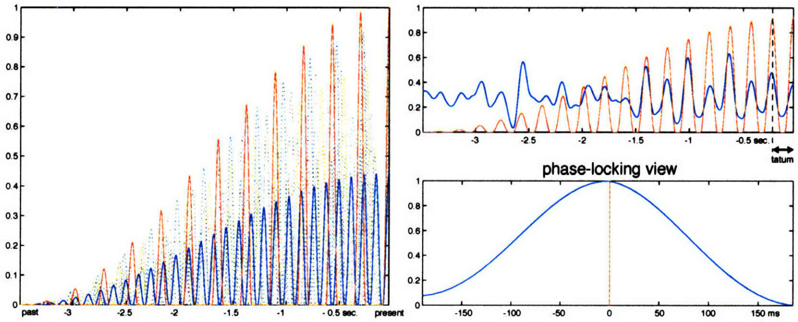


Figure 3-13: Tatum tracking. [left] A bank of dozens of templates like the ones displayed here are pre-computed: eight are shown, with a memory decay of about 3.5 seconds: *present* is on the right; *past* is on the left. [right] Example of tracking "Misery" by The Beatles. The top pane shows the smooth detection function (blue) and the current best template match (red). The bottom pane displays the cross-correlation response around the predicted phase for the optimal template. Here the template is in perfect phase with the signal.

3.5 Beat and Tempo

The beat (or *tactus*) is a perceptually induced periodic pulse that is best described by the action of "foot-tapping" to the music, and is probably the most studied metrical unit. It defines *tempo*: a pace reference that typically ranges from 40 to 260 beats per minute (BPM) with a mode roughly around 120 BPM. Tempo is shown to be a useful time-normalization metric of music (section 4.5). The beat is a down-sampled, aligned version of the tatum, although there is no clear and right answer on how many tatum periods make up a beat period: unlike tatum, which is derived directly from the segmentation signal, the beat sensation is cognitively more complex and requires information both from the temporal and the frequency domains.

3.5.1 Comparative models

Beat induction models can be categorized by their general approach: top-down (rule- or knowledge-based), or bottom-up (signal processing). Early techniques usually operate on quantized and symbolic representations of the signal, for instance after an onset detection stage. A set of heuristic and gestalt rules (based on accent, proximity, and grouping) is applied to infer the underlying metrical structure [99][37][159][45]. More recently, the trend has been on signal-processing approaches. The scheme typically starts with a front-end subband analysis of the signal, traditionally using a filter bank [165][141][4] or a discrete Fourier Transform [59][96][91]. Then, a periodicity estimation algorithm—including oscillators [141], histograms [39], autocorrelations [63], or probabilistic methods [95]—finds the rate at which signal events occur in concurrent channels. Finally, an integration procedure combines all channels into the final beat estimation. Goto's multiple-agent strategy [61] (also used by Dixon [38][39]) combines heuristics and correlation techniques together, including a chord change detector and a drum pattern detector. Klapuri's Bayesian probabilistic method applied on top of Scheirer's bank of resonators determines the best metrical hypothesis with constraints on continuity over time [92]. Both approaches stand out for their concern with explaining a hierarchical organization of the meter (section 4.6).

3.5.2 Our approach

A causal and bottom-up beat tracker based on our front-end auditory spectrogram (25 bands) and Scheirer's bank of resonators [141] is developed. It assumes no prior knowledge, and includes a *confidence* value, which accounts for the presence of a beat in the music. The range 60–240 BPM is logarithmically distributed to a large bank of *comb filters*, whose properties are to resonate at a given tempo. The filters are tested on multiple frequency channels of the auditory spectrogram simultaneously, and are tuned to fade out within seconds, as a way to model short-term memory. At any given time, their internal energy can be summed across channels by tempo class, which results in a *tempo spectrum* as depicted in Figure 3-14 (bottom). Yet, one of the main drawbacks of the model is its unreliable tempo-peak selection mechanism. A few peaks of the spectrum may give a plausible answer, and choosing the highest is not necessarily the best, or most stable strategy. A template mechanism is used to favor the extraction of the fastest tempo in case of ambiguity[1]. Section 5.3, however, introduces a *bias-free* method that can overcome this stability issue through top-down feedback control.

Figure 3-14 shows an example of beat tracking a polyphonic jazz-fusion piece at supposedly 143 BPM. A tempogram (middle pane) displays the tempo knowledge gained over the course of the analysis. It starts with no knowledge, but slowly the tempo space emerges. Note in the top pane that beat tracking was

[1] It is always possible to down-sample by a tempo octave if necessary.

stable after merely 1 second. The bottom pane displays the current output of each resonator. The highest peak is our extracted tempo. A peak at the sub octave (72 BPM) is visible, as well as some other harmonics of the beat. A real-time implementation of our beat tracker is available for the Max/MSP environment [180].

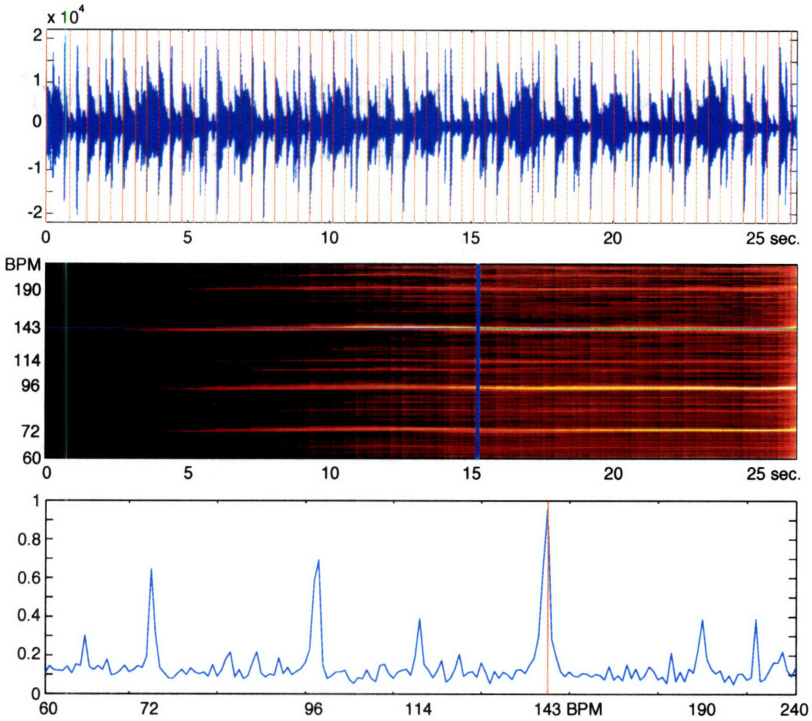


Figure 3-14: Beat tracking of a 27 sec. excerpt of "Watermelon man" by Herbie Hancock. [top] waveform (blue) and beat markers (red); [middle] tempogram: the system starts with no knowledge (black area) and gets gradually more confident; [bottom] tempo spectrum after 15 sec. of tracking.

3.6 Pitch and Harmony

The atomic audio fragments found through sound segmentation in section 3.4 represent individual notes, chords, drum sounds, or anything timbrally and harmonically stable. If segmented properly, there should not be any abrupt variations of pitch within a segment. Therefore it makes sense to analyze its pitch content, regardless of its complexity, i.e., monophonic, polyphonic, noisy. Since polyphonic pitch-tracking is yet to be solved, especially in a mixture of sounds that includes drums, we opt for a simpler, yet quite relevant 12-

dimensional *chroma* (a pitch class regardless of its register) description as in [8]. A *chromagram* is a representation of chromas against time. It was previously used for chord recognition [178], key analysis [131], chorus detection [60], and thumbnailing [27].

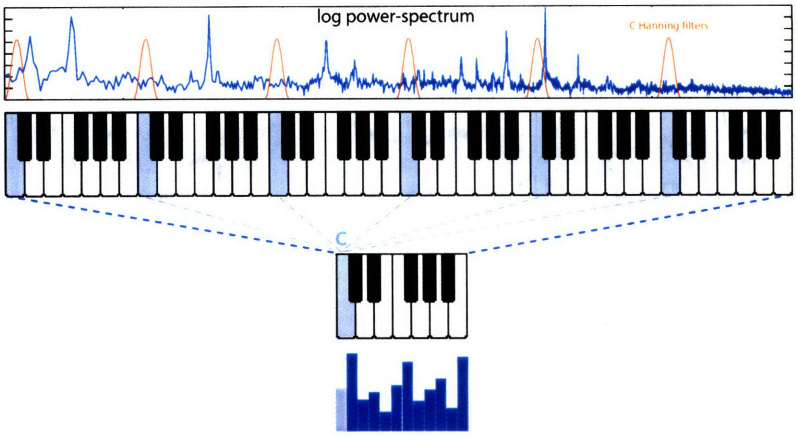


Figure 3-15: Computing schematic for building a chromagram. The power spectrum energy is accumulated into 12 pitch classes through a bank of filters tuned to the equal temperament chromatic scale.

We compute the FFT of the whole segment (generally between 80 to 300 ms long), which gives us sufficient frequency resolution. A standard Hanning window is applied first, which slightly attenuates the effect of noisy transients while emphasizing the sustained part of the segment. A chroma vector is the result of folding the energy distribution of much of the entire power spectrum (6 octaves ranging from C1 = 65 Hz to B7 = 7902 Hz) into 12 discrete pitch classes. This is a fair approximation given that both fundamental and first harmonic correspond to the same pitch class and are often the strongest partials of the sound. The output of the 72 logarithmically spaced Hanning filters of a whole-step bandwidth—accordingly tuned to the equal temperament chromatic scale—is accumulated into their corresponding pitch class (Figure 3-15). The scale is best suited to western music, but applies to other tunings (Indian, Chinese, Arabic, etc.), although it is not as easily interpretable or ideally represented. The final 12-element *chroma vector* is normalized by dividing each of its elements by the maximum element value. We aim at canceling the effect of loudness across vectors (in time) while preserving the ratio between pitch classes within a vector (in frequency). An example of a segment-synchronized chromagram for four distinct sounds is displayed in Figure 3-16.

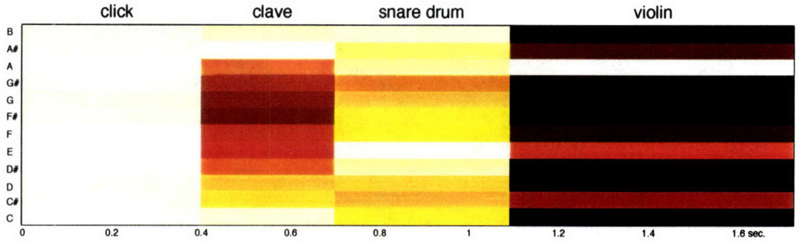


Figure 3-16: Four segment-synchronous chroma stpectra for the sound example in Figure 3-8: a digital click, a clave, a snare drum, and a staccato violin sound. The broadband click sound is the noisiest (flat). The violin sound is the most harmonic: the pitch played is an A. Even the visible A# emphasis is recognizable by listening to the sound of the noisy clave sound.

Our implementation differs significantly from others as we compute chromas on a segment basis. The great benefits of doing a *segment-synchronous*[2] computation are:

1. accurate *time* resolution by short-window analysis of onsets;
2. accurate *frequency* resolution by adaptive window analysis;
3. computation *speed* since there is no need to overlap FFTs; and
4. efficient and meaningful *representation* for future use.

The usual time-frequency paradox and our "simple" solution are shown in Figure 3-17 in the case of a chromatic scale. Our method optimizes both time and frequency analysis while describing the signal in terms of its *musical* features (onsets and inter-onset harmonic content). Figure 3-18 demonstrates the benefit of our pitch representation in the case of monotimbral music. Indeed, the critical-band representation, suitable for timbre recognition (section 3.3), is not suitable for pitch-content analysis. However, the segment-synchronous chromagram, which discards the timbre effect through its normalization process, appears suitable for describing pitch and harmonic content.

3.7 Perceptual feature space

In this chapter, we have modeled human perception through psychoacoustics. We have constructed a low-level representation of music signals in the form of a *sequence* of short audio segments, and their associated perceptual content:

[2] The term is borrowed from its equivalent in pitch analysis, as in "Pitch-Synchronous Overlap Add" (PSOLA).

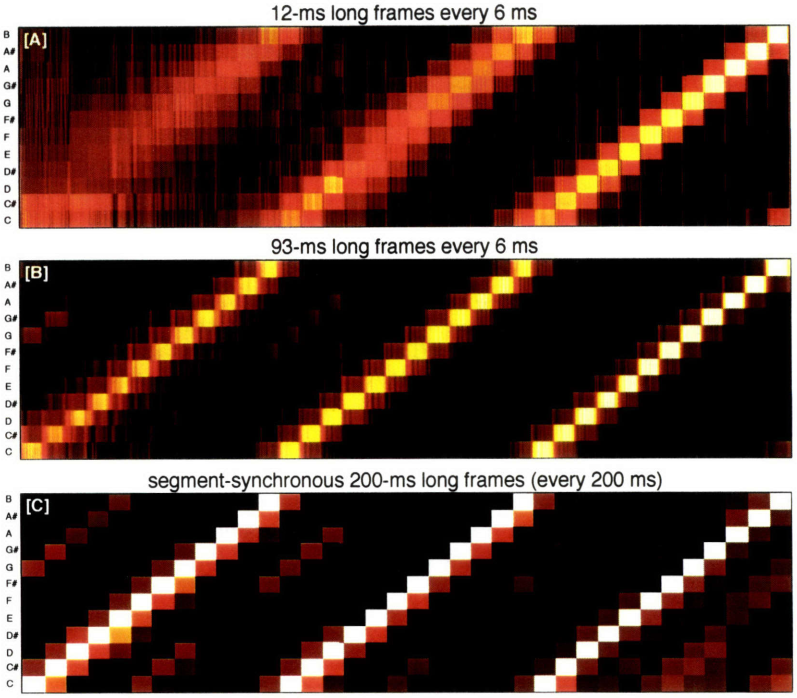


Figure 3-17: Time-frequency resolution paradox. A three-octave chromatic scale (from C2 to B4) is played on a piano at a rate of 5 notes per second. [A] Example of the distortion effect in the low range of the spectrum when computing the chromagram, due to the use of short windows (512 samples at 44.1 KHz); [B] long windows (4096 samples at 44.1 KHz) mostly fix the frequency resolution issue, but 1) compromises on temporal accuracy, and 2) is computationally more costly; [C] our solution employs short windows for the segmentation, and one adapted window for computing the frequency content of a segment. The result is both accurate and fast to compute.

1. **rhythm** represented by a loudness curve[3];
2. **timbre** represented by 25 channels of an auditory spectrogram;
3. **pitch** represented by a 12-class chroma vector.

Although the pitch content of a given segment is now represented by a compact 12-feature vector, loudness, and more so timbre, are still large—by an order of magnitude, since we still describe them on a frame basis. Empirically, it was found—via resynthesis experiments constrained only by loudness—that for a given segment, the *maximum* value in the loudness curve is a better approximation of the overall perceived loudness, than the *average* of the curve. For a more accurate representation, we describe the loudness curve with 5 dynamic features: loudness at onset (dB), maximum loudness (dB), loudness at offset (dB), length of the segment (ms), and time location of the maximum loudness

[3] We use the terms rhythm, loudness function, and dynamic features interchangeably.

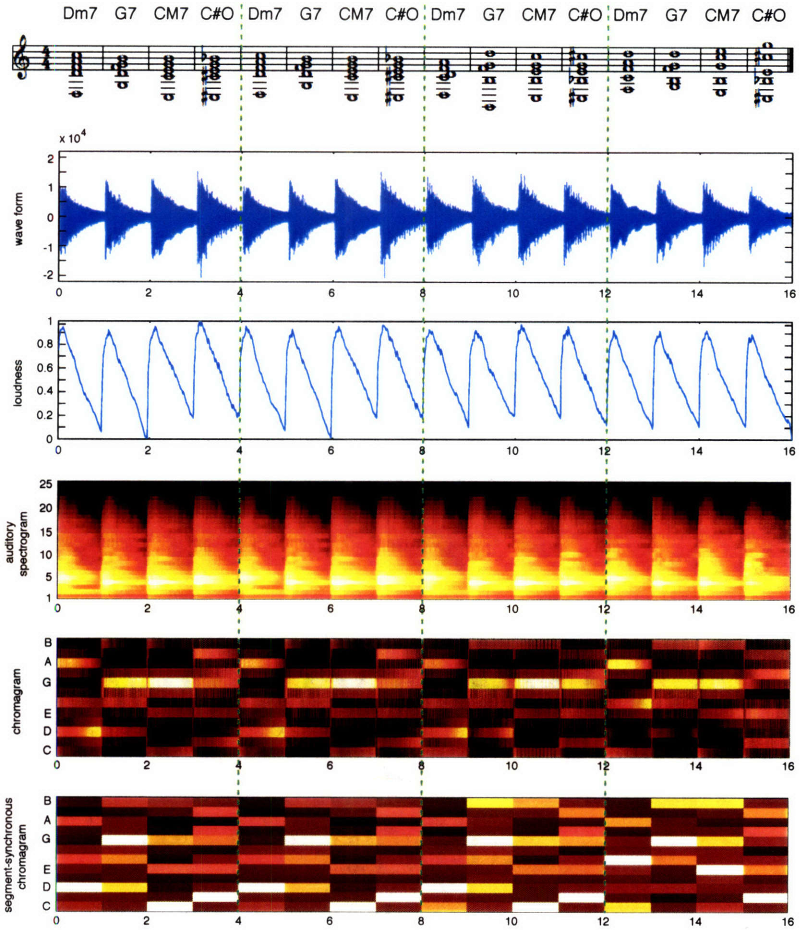


Figure 3-18: Typical II-V-I progressions played on a piano. No distinction can be made by looking at the waveform, the loudness curve, or the 25-critical band auditory spectrogram since this is only "one" timbre. However the chromagram allows us to differentiate chords and recognize similarities between chords of the same class even when they are inverted. The last pane shows the chromagram computed on a per-segment basis, while the previous one is computed every 6 ms with a 93-ms long window. Our representation is much simpler (only one chroma per segment), and preserves the most important information.

relative to the onset (ms). A currently rough approximation of timbre consists of averaging the Bark features over time, which reduces the timbre space to only 25 Bark features per segment. We finally label our segments with a total of 5 rhythm features + 25 timbral features + 12 pitch features = 42 features[4].

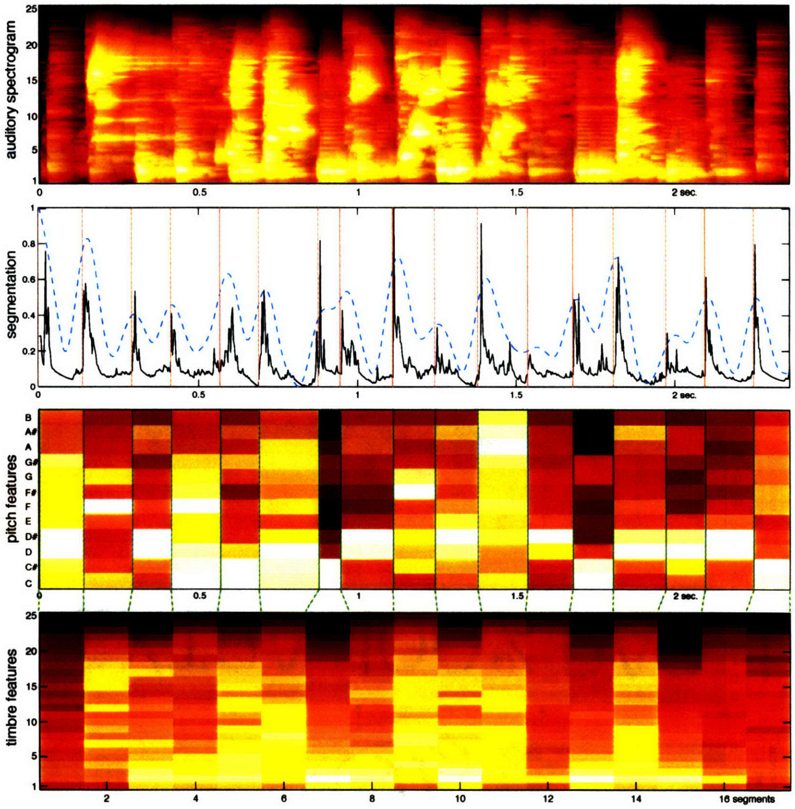


Figure 3-19: A short excerpt of "Sex Machine" by James Brown illustrating the 12 pitch features in synchrony with segments, and corresponding timbral features, equally distributed in time.

Such compact low-rate segment-synchronous feature description is regarded as a musical signature, or *signal metadata*, and has proven to be efficient and meaningful in a series of applications as demonstrated in chapter 6. In the following chapter, we interpret this musical metadata through a set of *hierarchical structures* of pitch, rhythm, and timbre.

[4] As already predicted by supercomputer *Deep Thought* in [2].

CHAPTER FOUR
Musical Structures

"One has no right, under pretext of freeing yourself, to be illogical and incoherent by getting rid of structure."

– Thelonious Monk

This chapter is a natural extension of *music listening* as described in chapter 3. The goal is to extract higher-level musical structures from the low-level musical *metadata* already derived from the audio. Therefore, the work here implies using more *pattern-recognition* techniques [42] than signal processing. With perception in particular, pattern recognition has much to do with *similarity*, as we are dealing with "interpreted" signals—there is no absolute truth or consensus on how to organize audio signals. Musical similarity though, remains to be defined properly [81].

4.1 Multiple Similarities

Measuring similarities in music audio is a particularly *ambiguous* task that has not been adequately addressed. The first reason why this problem is difficult is simply because there is a great variety of criteria (both objective and subjective) that listeners must consider and evaluate *simultaneously*, e.g., genre, artist, instrument, tempo, melody, rhythm, timbre. Figure 4-1 depicts this problem in the visual domain, with at least two obvious classes of similarities that can be considered: outline and texture.

Another major difficulty comes from the *scale* at which the similarity is estimated. Indeed, measuring the similarity of sound segments or entire records raises different questions. Previous works have typically focused on *one* single

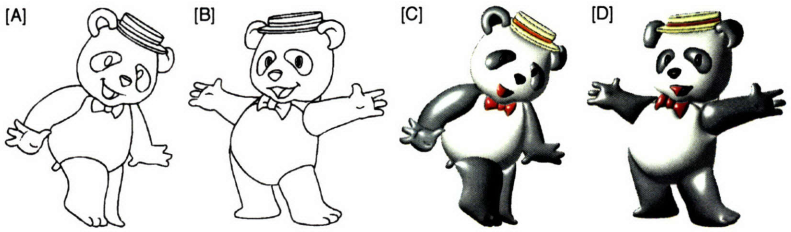


Figure 4-1: Simple example of the similarity problem in the visual domain. When asked "How would you group these four images?" about a 1/3 of Media Lab students would group images as A–B and C–D, 1/3 would group them as A–C and B–D, and the rest would not answer. The panda outlines are from César Coelho. The computer-generated 3D renderings are from Hedlena Bezerra.

task, for instance: instruments [111], rhythmic patterns [46], or entire songs [6]. Our approach, on the other hand, is an attempt at analyzing multiple levels of musically meaningful similarities, in a *uniform* fashion.

4.2 Related Work

Recent original methods of mapping music into a subjective *semantic* space for measuring the similarity of songs and artists have been proposed [11]. These methods rely on statistical models of labeled signals—for example, acquired through web-crawling and natural language processing (NLP)—which do not easily scale down in measuring similarities within songs. Here, we are only concerned with "objective" acoustic criteria (or musical dimensions) measurable directly from the signal with no prior knowledge, such as the ones extracted through music listening: pitch, rhythm, and timbre. The analysis task is challenging for the following reasons:

1. non-orthogonality of the musical dimensions; and
2. hierarchical dependency of the various scales of analysis.

"Non-orthogonality" because the pitch, loudness and timbre percepts depend on a shared underlying signal structure. "Hierarchical" because the sensation of tatum, tempo, or pattern all depend on the lowest-level segment structure. We believe that a *multi-scale* and *multi-class* approach to acoustic similarities is a step forward towards a more generalized model.

4.2.1 Hierarchical representations

Hierarchical representations of pitch and rhythm have already been proposed by Lerdahl and Jackendoff in the form of a musical grammar based on Chomskian linguistics [100]. Although a questionable oversimplification of music, among other rules their theory includes the *metrical structure*, as in our representation. However, in addition to pitch and rhythm, we introduce the notion of hierarchical *timbre* structure, a perceptually grounded description of music audio based on the metrical organization of its *timbral surface*: the perceived spectral shape in time, as described in section 3.3.

4.2.2 Global timbre methods

Global timbre analysis has received much attention in recent years as a means to measure the similarity of songs [11][6][69]. Typically, the estimation is built upon a pattern-recognition architecture. The algorithm aims at capturing the overall timbre distance between songs, given a large set of short audio frames, a small set of acoustic features, a statistical model of their distribution, and a distance function. It was shown, however, by Aucouturier and Pachet, that these generic approaches quickly lead to a "glass ceiling," at about 65% R-precision [7]. They conclude that substantial improvements would most likely rely on a "deeper understanding of the cognitive processes underlying the perception of complex polyphonic timbres, and the assessment of their similarity."

It is indeed unclear how humans perceive the superposition of sounds, or what "global" means, and how much it is actually more significant than "local" similarities. Comparing most *salient* segments, or patterns in songs, may perhaps lead to more meaningful strategies.

4.2.3 Rhythmic similarities

A similarity analysis of rhythmic patterns is proposed by Paulus and Klapuri [130]. Their method, only tested with drum sounds, consists of aligning tempo-variant patterns via dynamic time warping (DTW), and comparing their normalized spectral centroid, weighted with the log-energy of the signal. It is pointed out that aligning patterns turns out to be the most difficult task.

Ellis and Arroyo [46] present an approach to rhythm similarity called "eigen-rhythm" using Principle Component Analysis (PCA) of MIDI drum patterns from 100 popular songs. First, the length and downbeat of input patterns are estimated via autocorrelation and by alignment to an average "reference" pattern. Finally, through PCA analysis, they reduce the high-dimensionality data to a small space of combined "basis" patterns that can be used for classification and visualization.

In [129], Parry and Essa extend the notion of rhythmic *elaboration* to audio, first proposed by Tanguiane [158] for symbolic music. They divide the amplitude envelope via beat tracking, and measure the pattern length as in [130]. This method stands out for its ability to estimate the *complexity* of transitioning between intra- or inter-song patterns.

4.2.4 Self-similarities

In 2001, Foote and Cooper [52] were first to introduce *self-similarities* as a way to visualize musical structures. The method consists of building a square matrix where time runs from left to right, as well as from top to bottom. Cells $\{t_i, t_j\}$ of the matrix represent the audio similarity at time t_i and t_j in the waveform. Typically the *distance measure* is symmetric (i.e., $dist(t_i, t_j) = dist(t_j, t_i)$), thus the matrix is symmetric, and the diagonal is null as shown for the example in Figure 4-7.

From matrix to segmentation

Foote and Cooper propose the cosine distance between Mel-Frequency Cepstral-Coefficient (MFCC[1]) feature vectors. They suggest that "psychoacoustically motivated parameterizations [...] may be especially appropriate if they match the similarity judgments of human listeners." From their matrix can be derived a representation of the rhythmic structure that they call *beat spectrum*. In [30], they propose a statistically based framework for segmenting and clustering larger audio segments via *Singular Value Decomposition* (SVD). The analysis can, for instance, return the structural summarization of a piece by recognizing its "most representative" sections, such as chorus and verse.

Peeters et al. also propose to summarize songs by detecting occurrences of large sections in self-similarity matrices [132]. Their approach stands out for introducing *dynamic* features, which model the temporal evolution of the spectral shape over a fixed duration of time. Interestingly, they consider *various scales* of analysis (up to 1Hz), and employ an unsupervised grouping strategy of successive states, through K-means and a hidden Markov model (HMM). In a first pass, states are defined as *templates*, by segmenting the matrix.

From segmentation to matrix

Our approach starts with a series of significant segmentations, naturally derived from perceptual models of listening, and hierarchically organized. Our similarity matrices differ from others in the nature of the audio representation, and in how we derive larger segment similarities (i.e., sound, beat, pattern). The standard approach usually consists of 1) computing a fine-grain matrix of "generic" similarities, through short windows of fixed duration, and 2) segmenting the matrix at a given scale by detecting sequence repetitions (upper/lower diagonals). Instead, we *first* compute meaningful segmentations of "specific"

[1] MFCC is a popular choice front end for state-of-the-art speech recognition systems.

perceptual characteristics (i.e., pitch, rhythm, timbre), and only *then* compute their similarity matrices. Our benefits are:

1. multiple representations for more specific applications;
2. segment boundaries accurately preserved; and
3. independence to time (tempo) variations.

In a sense, our segmented and synchronous matrices (i.e., segment, beat, pattern), which are orders of magnitude smaller than usual frame-based matrices (Figure 4-2), are graphical artifacts of intermediate structure-analysis results, rather than an algorithmic starting point, as is usually the case. We are *not* particularly concerned with extracting very large structures from the lowest-level matrices directly. Nonetheless, these algorithms still apply at higher levels, where the typical $O(N^2)$ computation cost might originally be a limitation with standard frame-based representations.

Our hierarchical representation is *recursively* computed using *dynamic programming* (section 4.3), which itself is a recursive algorithm. We use larger matrices (lower-level) to infer the smaller ones (higher-level), such that the pattern-synchronous self-similarity matrices (our smallest) are computed via dynamic programming from the beat-synchronous self similarity matrices, which themselves are computed from the segment-synchronous self-similarity matrices, which are found via dynamic programming at the frame scale.

4.3 Dynamic Programming

> *"If you don't do the best with what you have happened to have got, you will never do the best with what you should have had."*
>
> – Rutherford Aris

Dynamic programming[2] (DP) is a method for solving dynamic (i.e., with time structure) optimization problems using recursion. It is typically used for text processing and *alignment* and similarity of biological sequences, such as protein or DNA sequences. First, a *distance* function determines the similarity of items between sequences (e.g., 0 or 1 if items match or not). Then, an *edit* cost penal-

[2] The word *programming* in the name has nothing to do with writing computer programs. Mathematicians use the word to describe a set of rules, which anyone can follow to solve a problem.

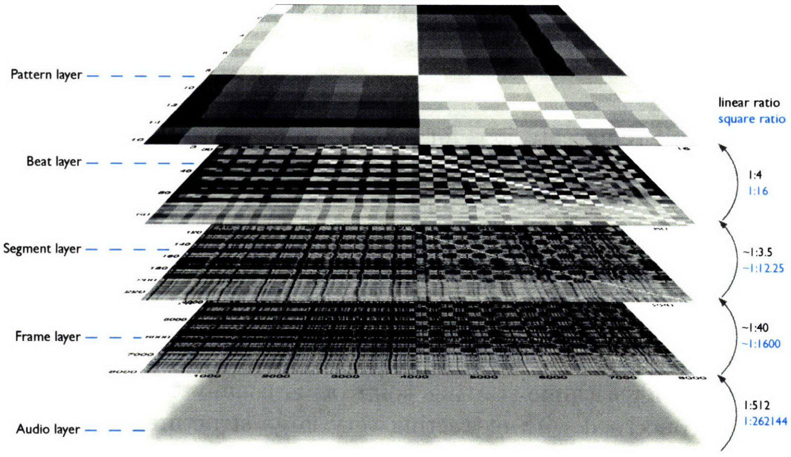


Figure 4-2: 3D representation of our hierarchical similarity representation in the *timbre* case for a 47-second excerpt of the song "Feel no pain" by Sade. [right] The first number (black) represents the *linear ratio* (i.e., per dimension) going from one scale to another. The second number (blue) is the *square ratio* (i.e., number of cells in the matrix).

izes alignment changes in the sequence (e.g., deletion, insertion, substitution). Finally, the total accumulated cost is a measure of dissimilarity.

A particular application of DP is known as *dynamic time warping* (DTW) [33]. The concept of dynamic time alignment has been applied to solve the problems associated with spectral sequence comparison of speech in automated speech recognition (ASR) systems [72].

The principle illustrated in Figure 4-3 is the following: given a test pattern $\mathbf{T} = \{\mathbf{t}_1, \cdots, \mathbf{t}_N\}$ and a reference pattern $\mathbf{R} = \{\mathbf{r}_1, \cdots, \mathbf{r}_M\}$, a particular point-to-point mapping $\phi = (\phi_t, \phi_r)$ of length K_ϕ time aligns $\mathbf{T}$ and $\mathbf{R}$.

$$\mathbf{t}_{\phi_t(k)} \Leftrightarrow \mathbf{r}_{\phi_r(k)} \quad \text{with} \quad 1 \leq k \leq K_\phi \tag{4.1}$$

The optimal alignment minimizes the overall distortion $D(\mathbf{T}, \mathbf{R})$, where $D(\mathbf{T}, \mathbf{R})$ is based on a sum of local distances between elements, i.e., $d(\mathbf{t}_i, \mathbf{r}_j)$, where d is a distance defined as Euclidean, Cosine, Mahalanobis, Manhattan, Minkowski, etc., and $\mathbf{t}_i$, $\mathbf{r}_j$ are specific feature vectors. The recursive algorithm incrementally evaluates all possible paths to the desired endpoint $\{\mathbf{t}_N, \mathbf{r}_M\}$. Once the endpoint is reached, the optimal path can be determined by backtracking through the minimum cost function:

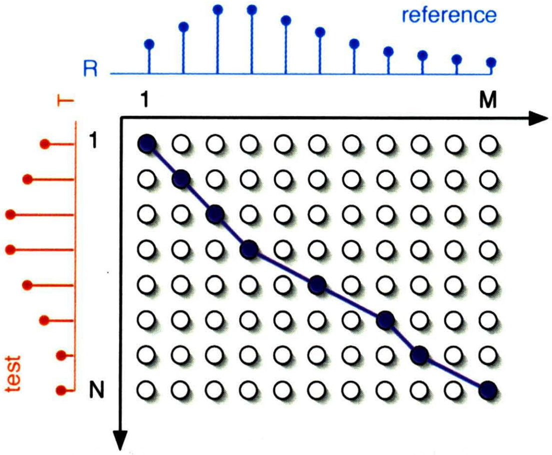


Figure 4-3: Example of dynamic time warping (DTW) in the case of one-dimensional signals. In our case, we deal with two-dimensional signals: the distance between each point $\mathbf{t}_i$ and $\mathbf{r}_j$ is measured as a distance between two feature vectors (e.g., two 25-dimensional Bark frames).

$$\begin{aligned} D(\mathbf{T},\mathbf{R}) &= \min_{\phi} D_\phi(\mathbf{T},\mathbf{R}) && (4.2) \\ D_\phi(\mathbf{T},\mathbf{R}) &= \frac{1}{M_\phi}\sum_{k=1}^{K_\phi} d(\mathbf{t}_{\phi_t(k)},\mathbf{r}_{\phi_r(k)})\cdot m_k && (4.3) \end{aligned}$$

with the following endpoint, and monotonicity constraints,

$$\begin{aligned} \phi_t(1) &= 1 \\ \phi_r(1) &= 1 \\ \phi_t(K) &= N \\ \phi_r(K) &= M \\ \phi_t(k+1) &\geq \phi_t(k) \\ \phi_r(k+1) &\geq \phi_r(k) \end{aligned}$$

M_ϕ is a path normalization factor that allows to compare between different warps, e.g., with different lengths:

$$M_\phi = \sum_{k=1}^{K_\phi} m_k \qquad (4.4)$$

and m_k is a path weight tuned according to the problem being solved as in Figure 4-4, or can be discarded: $m_k = 1$ for $k \in (1, K_\phi)$.

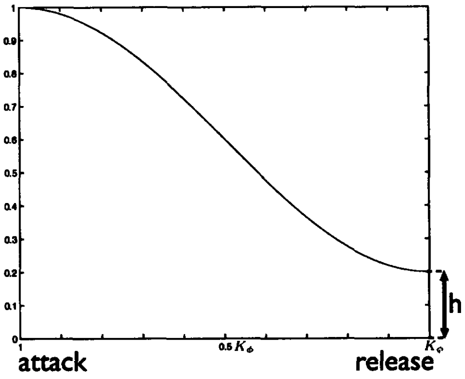


Figure 4-4: Weight function used for the timbre similarity of sound segments. The parameter "h" is chosen empirically.

4.4 Sound Segment Similarity

In a first task, we seek to measure the *timbral similarity* of the segments found through section 3.4. A first approximation of $D(\mathbf{T}, \mathbf{R})$ consists of measuring the Euclidean distance (equation 4.5) between the 25-dimensional feature vectors proposed in section 3.7. However, since timbre depends highly on temporal resolution, if we have access to the auditory spectrogram, a much more advanced solution is to time align the segments using dynamic time warping. In this case, $d(\mathbf{t}_i, \mathbf{r}_j)$ is the distance between the 25-dimensional feature vectors of the *auditory spectrogram*, and $D(\mathbf{T}, \mathbf{R})$ is defined by the optimal path in the DTW algorithm (section 4.3). The Euclidean distance (equation 4.5), defined as the straight line distance between two points, should be appropriate since the auditory spectrogram is perceptually normalized, i.e., the geometric distance between segments is proportional to the perceptual distance, as defined psychoacoustically. It is computed for two points $x = (x_1, \cdots, x_N)$ and $y = (y_1, \cdots, y_N)$ in Euclidean N-space as:

$$D_{Euclidean} = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2} \qquad (4.5)$$

It was shown in [66] that attacks are more important than decays for timbre recognition. We dynamically weigh the path (parameter m_k in equation 4.3) with a half-raised cosine function, as displayed in Figure 4-4, therefore increasing the alignment cost at the attack more than at the decay. Two parameters are chosen empirically (edit cost, and weight function value h), which could be optimized in future implementations. We compute the ***segment-synchronous self-similarity matrix of timbre*** as displayed in Figure 4-7 (top-right). Note that the structural information—visible in the frame-based self-similarity matrix—remains, although we are now looking at a much lower, yet musically informed analysis rate (a matrix size ratio of almost 1600 to 1).

The pitch similarity of segments is computed directly by measuring the distance between chroma vectors, since it was shown that time resolution is not really an issue (section 3.6). We choose the Euclidean distance. However, here is a place to insert specific heuristics on the perceptual distance between chords: for example, CM7 may sound closer to Em7 than to C7 [93]. A simple example of decorrelating timbre from pitch content in segments is shown in Figure 4-6. Note how timbre boundaries are easily detected regardless of their pitch content, and how chord patterns are clearly identified, regardless of their timbre. Finally, the dynamic-loudness similarity of segments[3] can be computed by DTW of the one-dimensional loudness curve.

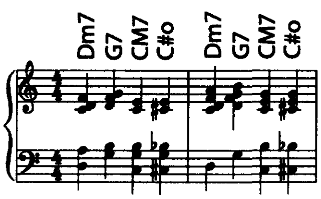


Figure 4-5: Chord progression played successively with various timbres, as in the example of Figure 4-6.

4.5 Beat Analysis

The beat analysis (section 3.5) reveals the underlying musical metric on which sounds arrange. It is generally found between 2 to 5 segments per beat. Using the segment-synchronous self-similarity matrix of timbre as a *new* distance function $d(\mathbf{t}_i, \mathbf{r}_j)$, we can repeat the DP procedure again, and *infer* a beat-synchronous self-similarity matrix of timbre. Although we do not consider it, here is a place to insert more heuristics, e.g., by weighting *on*-beat segments more than *off*-beat segments. Another option consists of computing the simi-

[3] We cannot really talk about rhythm at this level.

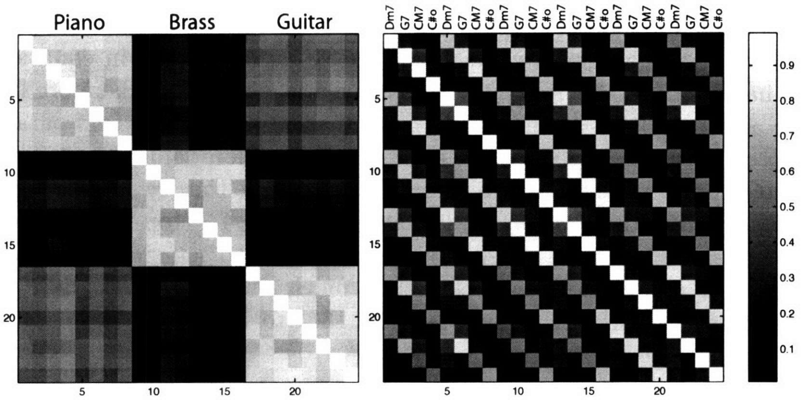


Figure 4-6: Test example of decorrelating timbre from pitch content. A typical II-V-I chord progression (as in Figure 4-5) is played successively with piano, brass, and guitar sounds on a MIDI instrument. [left] segment-synchronous self-similarity matrix of timbre. [right] segment-synchronous self-similarity matrix of pitch.

larity of beats directly from the auditory spectrogram as in section 4.4, which is more costly.

It musically makes sense to consider pitch similarities at the beat level as well. We compute the beat-synchronous self-similarity matrix of pitch much like we do for timbre. Unlike sound-synchronous self-similarity matrices, beat-synchronous self-similarity matrices are perfectly normalized in time, regardless of their local tempo. This is very important in comparing music especially where tempos are not perfectly steady. Note in Figure 4-7 (bottom-left) that the much smaller matrix displays a series of upper/lower diagonals, which reveal the presence of musical patterns.

4.6 Pattern Recognition

Beats can often be grouped into patterns, also referred to as *meter* and indicated by a symbol called a *time signature* in western notation (e.g., 3/4, 4/4, 12/8). This section, however, deals with patterns as perceived by humans, rather than their original score notation, as organized by *measures*.

4.6.1 Pattern length

A typical method for finding the length of a pattern consists of applying the autocorrelation function of the signal energy (here the loudness curve). This is a good approximation based on dynamic variations of amplitude (i.e., the rhythmic content), but it does not consider pitch or timbre variations. Our

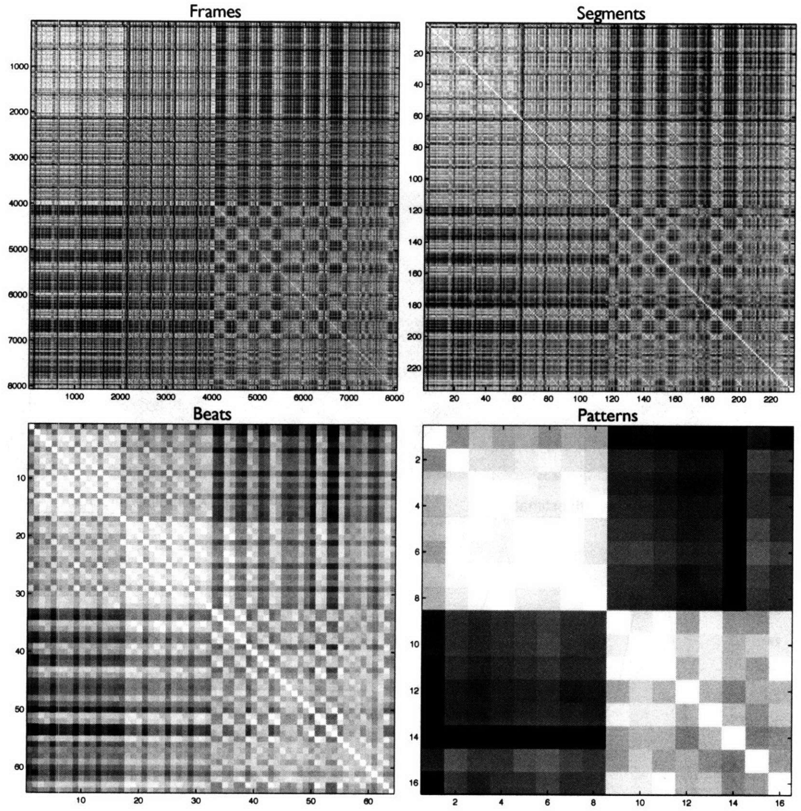


Figure 4-7: First 47 seconds of Sade's "Feel no pain" represented hierarchically in the timbre space as self-similarity matrices: [top-left] frames; [top-right] segments; [bottom-left] beats; [bottom-right] patterns. Note that each representation is a scaled transformation of the other, yet synchronized to a meaningful musical metric. Only beat and pattern representations are tempo invariant. This excerpt includes 8 bars of instrumental introduction, followed by 8 bars of instrumental plus singing. The two sections appear clearly in the pattern representation.

system computes pattern similarities from a short-term version of the beat-synchronous self-similarity matrices (only considering a limited section around the main diagonal), therefore it synchronizes the analysis to the beat.

We run parallel tests on a beat basis, measuring the similarity between successive patterns of 1- to 11-beat long—typical patterns are 3, 4, 5, 6, or 8 beats long—much like our bank of oscillators with the beat tracker (section 3.5). We pick the first peak, which corresponds to a particular number of beats (Figure 4-8). Note that patterns in the pitch dimension, if they exist, could be of different lengths than those found in the timbre dimension or rhythm dimension. An example where only analyzing the pitch content can characterize the length of a pattern is shown in Figure 4-9. A complete model should include all representations, such that the length L of an ideal pattern is found as the first peak in a combination of all similarities. However, this is not currently implemented in our system: we choose timbre similarities for finding the pattern length.

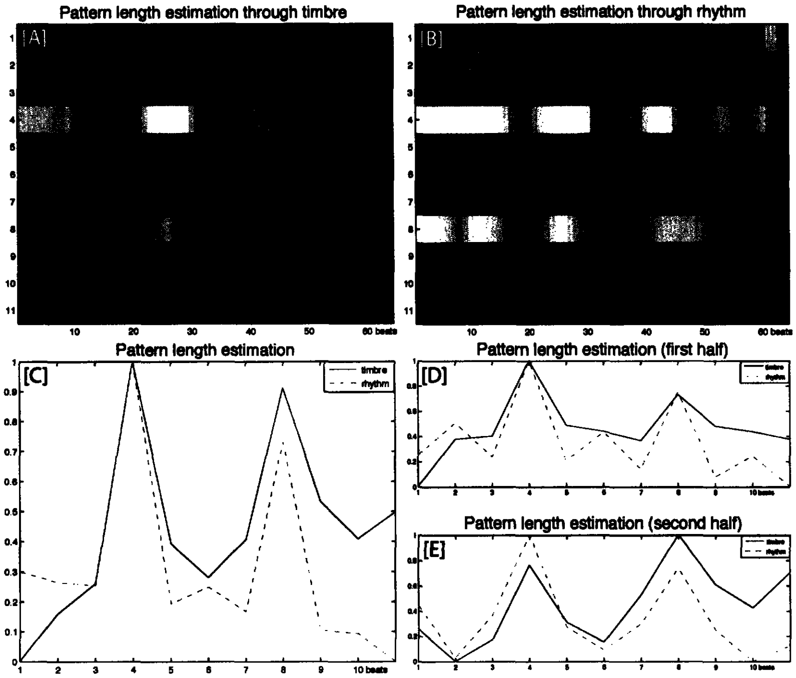


Figure 4-8: Causal analysis of pattern length through timbre [A] or rhythm (via the autocorrelation of our loudness function) [B] for the Sade excerpt of Figure 4-7. Overall, both timbre and rhythm agree on a first peak at 4 beats [C]. In the first half of the excerpt however, the rhythm's first peak is at 2 beats while at 4 beats in the timbre space [D]. In the second half of the excerpt, both show a pattern length of 4 beats again [E].

A drawback of autocorrelation methods is that they do not return the phase information, i.e., where patterns begin. This problem of *downbeat* prediction is addressed in section 5.3 by taking advantage of our current multi-class representation, together with a training scheme. However, we now present a simpler heuristically based approach, generally quite reasonable with popular music.

4.6.2 Heuristic approach to downbeat detection

We assume that chord changes are most likely to occur on the downbeat as opposed to other beats: a fair assumption with most western music, typically annotated on a measure basis. We suppose that we already know the length L (in number of beats) of a pattern, as found in section 4.6.1. We define the *dissimilarity* D between two successive patterns by the Euclidean distance (equation 4.5) between their averaged chromas over L. For a given pattern i, the downbeat estimation consists of finding the maximum likelihood $\max_{\phi} D_{\phi_j}[i]$ in a set of L dissimilarity evaluations, i.e., for all beat phase ϕ_j, where $0 \leq j \leq L-1$.

If L can be divided by two, then it is likely that the minimum likelihood $\min_{\phi} D_{\phi_j}[i]$ occurs at opposite phase $((\phi_j + L/2) \mod L)$ compared to the maximum likelihood. Indeed, averaging chromas over two chords is more likely to minimize dissimilarities. Therefore, a more robust strategy first computes the absolute difference between pairs of dissimilarities in phase opposition, and chooses the best candidate (maximum likelihood) from the pair with highest absolute difference.

The process is causal, although it has a lag of $2L$ beats as demonstrated in Figure 4-9 for a simple synthesized example, and in Figure 4-10 for a real-world example. The lag can be completely removed through a general predictive model of downbeat prediction, as proposed in section 5.3. However, if real-time analysis is not a concern, then overall the present approach is statistically reasonable.

4.6.3 Pattern-synchronous similarities

Finally, we derive the pattern-synchronous self-similarity matrix, again via dynamic programming. Here is a good place to insert more heuristics, such as the weight of strong beats versus weak beats. However, our current model does not assume any weighting. We implement pitch similarities from beat-synchronous chroma vectors, and rhythm similarities using the *elaboration* distance function proposed in [129], together with our loudness function. Results for an entire song can be found in Figure 4-11. Note that the elaboration distance function is not symmetric: a simple rhythmic pattern is considered more similar to a complex pattern than vice versa.

4.7 Larger Sections

Several recent works have dealt with the question of thumbnailing [28], music summarization [132][30], or chorus detection [60]. These related topics all deal with the problem of extracting large non-periodic sections in music. As can be seen in Figure 4-7 (bottom-right), larger musical structures appear in the matrix of pattern self-similarities. As mentioned in section 4.2.4, advantages of our system in extracting large structures are 1) its invariance to tempo, and 2) segmentation is inherent to its representation: finding section boundaries is less of a concern as we do not rely on such resolution.

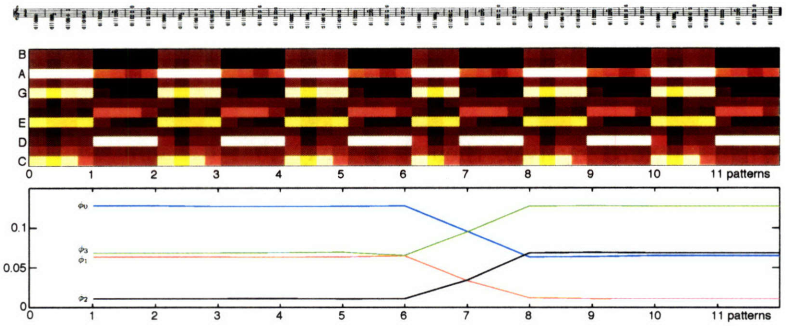


Figure 4-9: [top] A series of eight *different* chords (four Am7 and four Dm7) is looped 6 times on a piano at 120 BPM, minus one chord discarded in the middle, i.e., 47 chords total. [middle] The chromagram depicts beat-synchronous chromas. [bottom] Four evaluations that measure the *dissimilarity* of two consecutive patterns of four beats, are run in parallel every four beats, for ϕ_0 (blue), ϕ_1 (red), ϕ_2 (black), and ϕ_3 (green). While D_{ϕ_0} is clearly the highest during the first half (i.e., the first beat is the downbeat), the highest beat phase then shifts to ϕ_3 (i.e., the downbeat has shifted back by one beat).

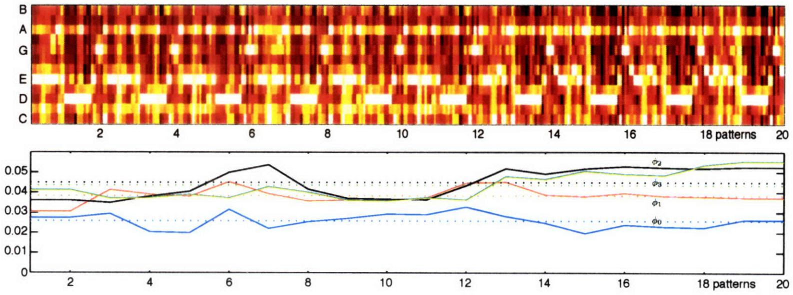


Figure 4-10: Real-world example using a 1-minute excerpt of "Lebanese blonde" by Thievery Corporation, a pop tune that includes drums, percussion, electric piano, voice, sitar, flute, and bass. The song alternates between underlying chords Am7 and Dm7, with a fair amount of syncopation. The beat tracker takes about a measure to lock on. The ground truth is then ϕ_2 as depicted by the black line. The most unlikely beat phase is ϕ_0, as shown in blue. We find through pairs in opposition phase that $|D_{\phi_2}[i] - D_{\phi_0}[i]| > |D_{\phi_3}[i] - D_{\phi_1}[i]|$, for $1 < i < 20$, which allows us to choose $\phi_2[i]$ with $\max_\phi(D_{\phi_0}[i], D_{\phi_2}[i])$. Dotted lines show the average phase estimation.

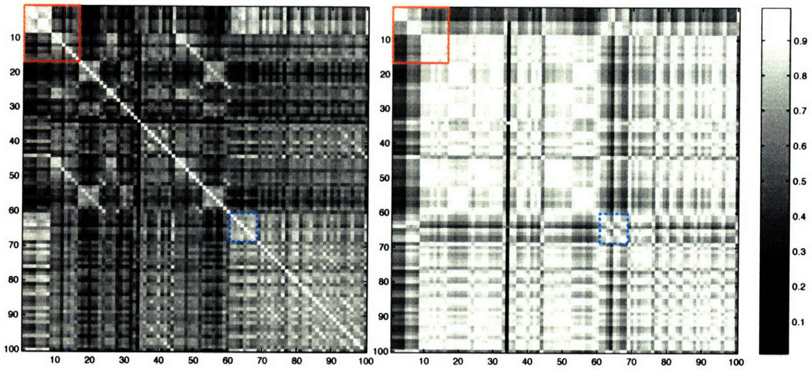


Figure 4-11: Pitch [left] and rhythm [right] self-similarity matrices of patterns for the entire Sade song "Feel no pain." The red squares outline the first 16 measures studied in Figure 4-7 with timbre similarities. Note that a break section (blue dashed squares) stands out in the rhythm representation, where it is only the beginning of the last section in the pitch representation.

Although this is not currently implemented, previous techniques used for extracting large structures in music, such as the Gaussian-tapered "checkerboard" kernel in [30] or the pattern matching technique in [28], apply here, but at a much lower resolution, increasing greatly the computation speed, while preserving the temporal accuracy. Note that the pattern level is a fair assumption of section boundary in most popular music. In future work, we may consider combining the different class representations (i.e., pitch, rhythm, timbre) into a single "tunable" system for extracting large sections.

4.8 Chapter Conclusion

In this chapter, we propose a *recursive* multi-class (pitch, rhythm, timbre) approach to the structure analysis of acoustic similarities in popular music. Our representation is hierarchically organized, where each level is musically meaningful. Although fairly intensive computationally, our dynamic programming method is time-aware, causal, and flexible. Future work may include inserting and optimizing heuristics at various stages of the algorithm. Our representation may be useful for *fast* content retrieval (e.g., through vertical, rather than horizontal search—going down the tree structure towards the best similarity rather than testing all the leaves); improved song similarity architectures that include specific content considerations; and music synthesis, as described in chapter 6.

CHAPTER FIVE
Learning Music Signals

> "*The beautiful thing about learning is that no one can take it away from you.*"
>
> – B.B. King

Learning is acquiring knowledge or skill through study, experience, or teaching. Whether a computer system "learns" or merely "induces generalizations" is often a subject of debate. Indeed, learning from data or examples (similarity-based learning) is another way of speaking about generalization procedures and concept representations that are typically "simplistic and brittle" [119]. However, we argue that a music-listening system is not *complete* until it is able to improve or adapt its performance over time on tasks similar to those done previously. Therefore, this chapter introduces learning strategies that apply to the music analysis context. We believe that state-of-the-art "learning" algorithms are able to produce robust models of relatively complex systems, as long as the data (i.e., musical features) is consistent and the learning problem well posed.

5.1 Machine Learning

Machine learning [115], a subfield of artificial intelligence [140], is concerned with the question of how to construct computer programs (i.e., agents) that automatically *improve* their performance at some task with *experience*. Learning takes place as a result of the interaction between the agent and the world, and from *observation* by the agent of its own decision-making processes. When learning from measured or observed data (e.g., music signals), the machine learning algorithm is also concerned with how to *generalize* the representation of that data, i.e., to find a regression or discrimination function that best describes the data or category. There is a wide variety of algorithms and techniques, and their description would easily fill up several volumes. There is no ideal one: results usually depend on the problem that is given, the complexity of implementation, and time of execution. Here, we recall some of the key notions and concepts of machine learning.

5.1.1 Supervised, unsupervised, and reinforcement learning

When dealing with music signals and extracting perceptual information, there is necessarily a fair amount of ambiguity and imprecision (noise) in the estimated data, not only due to the analysis technique, but also to the inherent fuzziness of the perceptual information. Therefore, statistics are widely used and will often play an important role in *machine perception*—a machine that can recognize patterns grounded on our senses. If an external *teacher* provides a *category* label or *cost* for each pattern (i.e., when there is specific feedback available), the learning is said to be *supervised*: the learning element is given the true output for particular inputs. It adapts its internal representation of a correlation function to best match the information provided by the feedback. More formally, we say that an example (or sample) is a pair $(x, f(x))$, where x is the input and $f(x)$ is the output of the function applied to x. *Induction* is the task that, given a collection of examples of f, returns a function h (the hypothesis) that approximates f. Supervised learning can be *incremental* (i.e., update its old hypothesis whenever a new example arrives) or based on a representative *training* set of examples. One must use a large enough amount of training samples, but one must keep some for validation of the hypothesis function (typically around 30%).

In *unsupervised learning* or *clustering*, there is no explicit teacher, and the system forms "natural" clusters (groupings) of the input patterns. Different clustering algorithms may lead to different clusters, and the number of clusters can be specified ahead of time if there is some *prior knowledge* of the classification task. Finally, a third form of learning, *reinforcement learning*, specifies only if the tentative classification or decision is right or wrong, which improves (reinforces) the classifier.

For example, if our task were to classify musical instruments from listening to their sound, in a supervised context we would first train a classifier by using a large database of sound recordings for which we know the origin. In an unsupervised learning context, several clusters would be formed, hopefully representing different instruments. With reinforcement learning, a new example with a known target label is computed, and the result is used to improve the classifier.

5.1.2 Generative vs. discriminative learning

One current way of categorizing a learning approach is differentiating whether it is *generative* or *discriminative* [77]. In generative learning, one provides domain-specific knowledge in terms of structure and parameter *priors* over the joint space of variables. Bayesian (belief) networks [84][87], hidden Markov models (HMM) [137], Markov random fields [177], Kalman filters [21], mixture of experts [88], mixture of multinomials, mixture of Gaussians [145], and so forth, provide a rich and flexible language for specifying this knowledge and subsequently refining it with data and observations. The final result is a probabilistic distribution that is a good generator of novel examples.

Conversely, discriminative algorithms adjust a possibly non-distributional model to data, optimizing a specific task, such as classification or prediction. Popular and successful examples include logistic regression [71], Gaussian processes [55], regularization networks [56], support vector machines (SVM) [22], and traditional artificial neural networks (ANN) [13]. This often leads to superior performance, yet compromises the flexibility of generative modeling. Jaakkola et al. recently proposed Maximum Entropy Discrimination (MED) as a framework to combine both discriminative estimation and generative probability density [75]. Finally, in [77], *imitative* learning is presented as "another variation on generative modeling, which also learns from examples from an observed data source. However, the distinction is that the generative model is an agent that is interacting in a much more complex surrounding external world." The method was demonstrated to outperform (under appropriate conditions) the usual generative approach in a classification task.

5.2 Prediction

Linear systems have two particularly desirable features: they are well characterized and are straightforward to model. One of the most standard ways of fitting a linear model to a given time series consists of minimizing squared errors between the data and the output of an ARMA (autoregressive moving average) model. It is quite common to approximate a nonlinear time series by *locally* linear models with slowly varying coefficients, as through Linear Predictive Coding (LPC) for speech transmission [135]. But the method quickly breaks down when the goal is to generalize nonlinear systems.

Music is considered a "data-rich" and "theory-poor" type of signal: unlike strong and simple models (i.e., theory-rich and data-poor), which can be expressed in a few equations and few parameters, music holds very few assumptions, and modeling anything about it must require a fair amount of *generalization* ability—as opposed to memorization. We are indeed interested in extracting regularities from *training* examples, which transfer to new examples. This is what we call *prediction.*

5.2.1 Regression and classification

We use our predictive models for *classification*, where the task is to categorize the data into a finite number of predefined classes, and for nonlinear *regression*, where the task is to find a smooth interpolation between points, while avoiding *overfitting* (the problem of fitting the noise in addition to the signal). The regression corresponds to mapping a high-dimensional input data stream into a (usually) one-dimensional nonlinear output function. This one-dimensional signal may be the input data itself in the case of a signal forecasting task (section 5.2.5).

5.2.2 State-space forecasting

The idea of forecasting future values by using immediately preceding ones (called a time-lag vector, or tapped delay line) was first proposed by Yule [179]. It turns out that the underlying dynamics of nonlinear systems can also be *understood*, and their geometrical behavior inferred from observing delay vectors in a time series, where no or *a priori* information is available about its origin [157]. This general principle is known as *state-space reconstruction* [53], and inspires our predictive approach. The method has previously been applied to musical applications, e.g., for determining the predictability of driven nonlinear acoustic systems [144], for musical gesture analysis and embedding synthesis [114], and more recently for modeling musical compositions [3].

5.2.3 Principal component analysis

Principal component analysis (PCA) involves a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called *principal components*. The first principal component accounts for the greatest possible statistical variability (or entropy) in the data, and each succeeding component accounts for as much of the remaining variability as possible. Usually, PCA is used to discover or reduce the dimensionality of the data set, or to identify new meaningful underlying variables, i.e., patterns in the data. Principal components are found by extracting the *eigenvectors* and *eigenvalues* of the covariance matrix of the data, and are calculated efficiently via singular value decomposition (SVD). These eigenvectors describe an *orthonormal basis* that is effectively a rotation of the original cartesian basis (Figure 5-1).

5.2.4 Understanding musical structures

When a time-lag vector space is analyzed through PCA, the resulting first few dimensions characterize the statistically most significant underlying degrees of freedom of the dynamical system. A data set, which appears complicated when

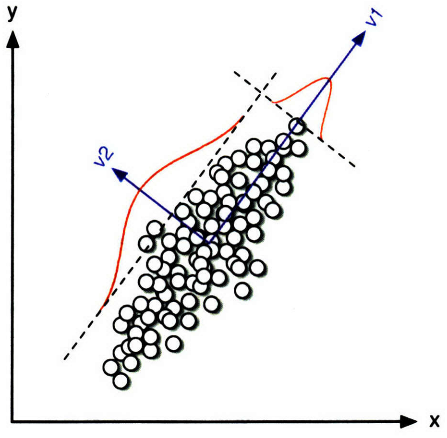


Figure 5-1: Graphical representation of a PCA transformation in only two dimensions. The variance of the data in the original cartesian space (x, y) is best captured by the basis vectors v1 and v2 in a rotated space.

plotted as a time series, might reveal a simpler structure, represented by a *manifold* when plotted as a stereo plot (Figure 5-2). This apparent simpler structure provides the evidence of some embedded redundant patterns, which we seek to model through *learning*. Our primarily goal is to build a generative (cognitive) representation of music, rather than "resynthesize" it from a rigid and finite description. Nevertheless, a different application consists either of transforming the manifold (e.g., through rotation, scaling, distortion, etc.), and unfolding it back into a new time series; or using it as an attractor for generating new signals, governed by similar dynamics [3].

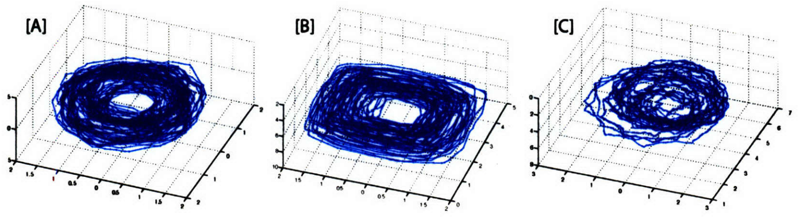


Figure 5-2: Manifold of the first three dimensions of the time-lag vector in the eigenspace. 4000 loudness points for [A] electronic piece "Lebanese blonde" by Thievery Corporation; [B] funk piece "It's all right now" by Eddie Harris; [C] jazz piece "One last pitch" by Harry Connick Jr. The manifold structure shows the existence of rhythmic patterns.

5.2.5 Learning and forecasting musical structures

We project a high-dimensional space onto a single dimension, i.e., a correlation between the time-lag vector $[x_{t-(d-1)\tau}, \cdots, x_{x_{t-\tau}}]$ and the current value x_t [48]. We use machine-learning algorithms to infer the mapping, which consequently provides an insight into the underlying behavior of the system dynamics. Given an initial set of d data points, the previously taught model can exploit the embedded geometry (long-term memory) to predict a new forthcoming data point x_{t+1}. By repeating the procedure even further through data points $x_{t+2}, \cdots, x_{t+(\delta-1)\tau}$, the model forecasts the time series even farther into the *future*.

We model rhythm in this way, through both the iterative mixture of Gaussian framework (CWM) provided by Schoner [145], and a support vector machine package (SVM^{light}) provided by Joachims [85] (Figure 5-3). The outcome is an extremely compact "rhythm" synthesizer that learned from example, and can generate a loudness function given an initialization data set. We measure the robustness of the model by its ability to predict 1) previously trained data points; 2) new test data points; and 3) the future of the time series (both short-term and long-term accuracy). With given numbers of delays and other related parameters, better overall performances are typically found using SVM (Figure 5-3). However, a quantitative comparison between learning strategies goes beyond the scope of this thesis.

5.2.6 Support Vector Machine

Support vector machines (SVM) [164] rely on preprocessing the data as a way to represent patterns in a *high* dimension—typically much higher than the original feature space. With appropriate nonlinear mapping into the new space, and through basis function, or *kernel*—such as Gaussian, polynomial, sigmoid function—data can always be regressed (and classified) with a *hyperplane* (Figure 5-4). Support vector machines differ radically from comparable approaches as SVM training always converges to a *global* minimum, i.e., the search corresponds to a *convex* quadratic programming problem, typically solved by matrix inversion. While obviously not the only machine learning solution, this is the one we choose for the following experiments.

5.3 Downbeat prediction

Other than modeling and forecasting one-dimensional signals, such as the loudness (rhythm) signal, we can also predict new musical information given a multidimensional input. A particularly interesting example is downbeat prediction based on surface listening, and time-lag embedded learning. The model is causal, and tempo independent: it does not require beat tracking. In fact, it

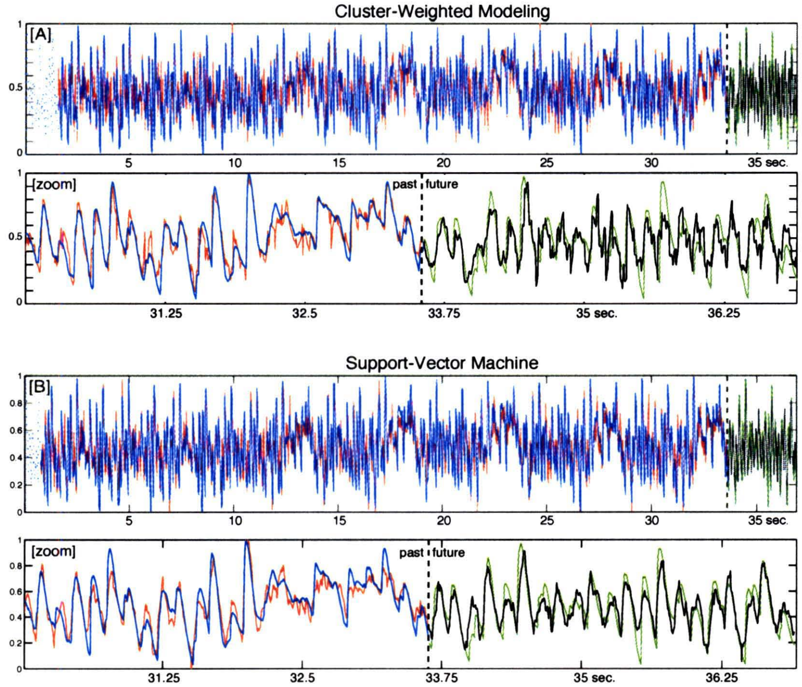


Figure 5-3: Rhythm prediction of funk song "It's all right now" by Eddie Harris using: [A] CWM with 12 Gaussians, 20-point tap delay, 20 past data points; [B] SVM with Gaussian kernel, 15-point tap delay, 15 past data points; [blue] Training loudness function; [dotted blue] initial buffer; [red] predicted loudness function; [green] unknown future ground truth; [black] forecasted future. The dotted black marker indicates the present.

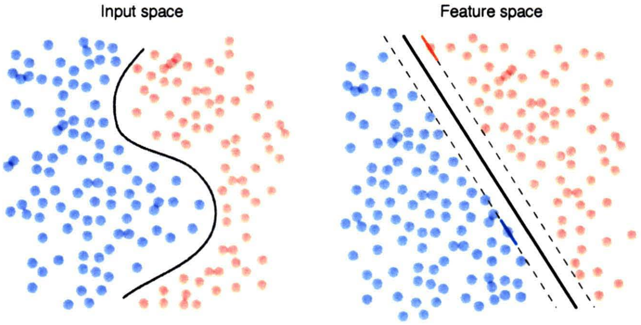


Figure 5-4: Typical classification task using SVM. The classifier must discriminate between two types of labeled data with a nonlinear discriminative function [input space]. The data is elevated into a higher dimensional space where the data can be discriminated with a hyperplane [feature space]. Dotted line represent *support vectors*. The SVM algorithm tries to maximize the distance between support vectors of opposite classes.

could appropriately be used as a phase-locking system for the beat tracker of section 3.5, which currently runs in an open loop (i.e., without feedback control mechanism). This is left for future work.

5.3.1 Downbeat training

The downbeat prediction is supervised. The *training* is a semi-automatic task that requires little human control. If our beat tracker is accurate throughout the whole training song, and the measure length is constant, we label only one beat with an integer value $p_b \in [0, M-1]$, where M is the number of beats in the measure, and where 0 is the downbeat. The system extrapolates the beat-phase labeling to the rest of the song. In general, we can label the data by tapping the downbeats along with the music in real time, and by recording their location in a text file. The system finally labels segments with a phase location: a float value $p_s \in [0, M[$. The resulting segment phase signal looks like a sawtooth ranging from 0 to M. Taking the absolute value of its derivative returns our *ground-truth* downbeat prediction signal, as displayed in the top pane of Figure 5-5. Another good option consists of labeling tatums (section 3.4.3) rather than segments.

The listening stage, including auditory spectrogram, segmentation, and music feature labeling, is entirely unsupervised (Figure 3-19). So is the construction of the time-lag feature vector, which is built by appending an arbitrary number of preceding multidimensional feature vectors. Best results were obtained using 6 to 12 past segments, corresponding to nearly the length of a measure. We model short-term memory *fading* by linearly scaling down older segments, therefore increasing the weight of most recent segments (Figure 5-5).

Training a support vector machine to predict the downbeat corresponds to a regression task of several dozens of feature dimensions (e.g., 9 past segments × 42 features per segments = 378 features) into one single dimension (the corresponding downbeat phase of the next segment). Several variations of this principle are also possible. For instance, an additional PCA step (section 5.2.3) allows us to reduce the space considerably while preserving most of its entropy. We arbitrarily select the first 20 eigen-dimensions (Figure 5-6), which generally accounts for about 60–80% of the total entropy while reducing the size of the feature space by an order of magnitude. It was found that results are almost equivalent, while the learning process gains in computation speed. Another approach that we have tested consists of selecting the relative features of a running self-similarity triangular matrix rather than the original absolute features, e.g., $((9 \text{ past segments})^2 - 9)/2 = 36$ features. Results were found to be roughly equivalent, and also faster to compute.

We expect that the resulting model is not only able to predict the downbeat of our training data set, but to generalize well enough to predict the downbeat

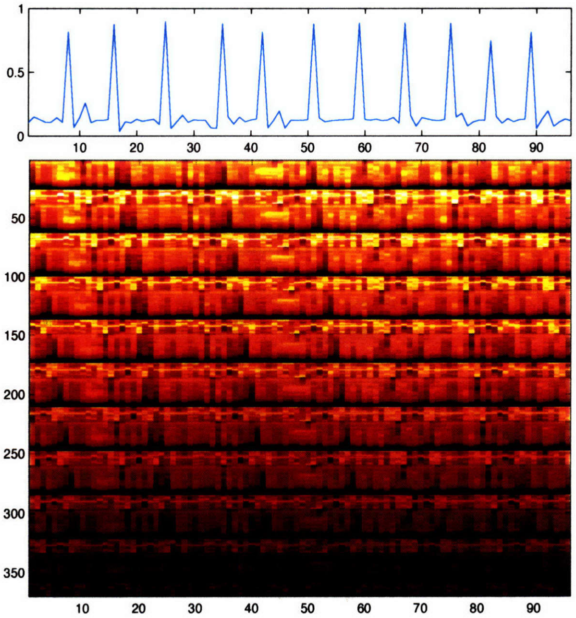


Figure 5-5: [bottom] Time-lag embedded feature vectors (9 segments of 37 features in this example) for a dozen measures of Beatles's "Love me do" song. Note that past features (greater index number) are faded out. [top] Corresponding target downbeat prediction signal.

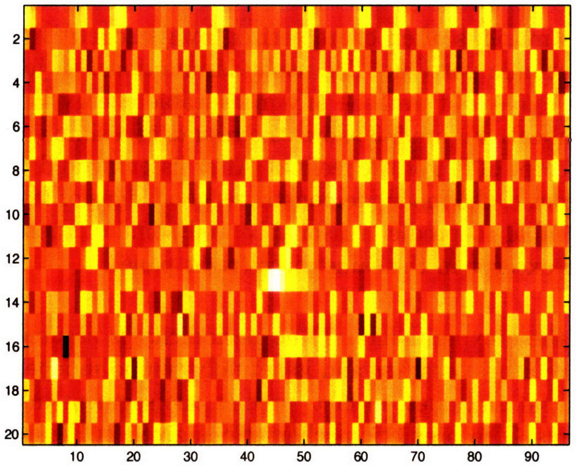


Figure 5-6: PCA reduction of time-lag feature vectors of Figure 5-5. The space is properly normalized by mean and variance across dimensions. Because each dimension represents successively axes of remaining maximum entropy, there is no more possible interpretation of the features.

of new input data—denominated *test* data in the following experiments. An overall schematic of the training method is depicted in Figure 5-7.

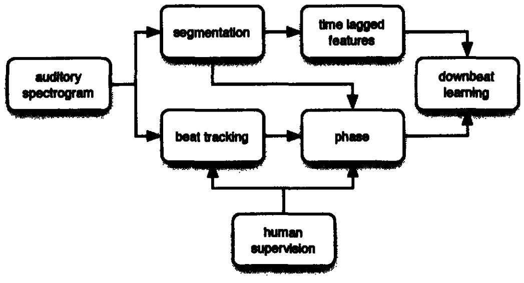


Figure 5-7: Supervised learning schematic.

Although downbeat may often be interpreted through harmonic shift [61] or a generic "template" pattern [92], sometimes neither of these assumptions apply. This is the case of the following example.

5.3.2 The James Brown case

James Brown's music is often characterized by its repetitive single chord and syncopated rhythmic pattern. The usual assumptions, as just mentioned, do not hold. There may not even be any measurable energy in the signal at the perceived downbeat. Figure 5-8 shows the results of a simple learning test with a 30-second excerpt of "I got the feelin'." After listening and training with only 30 seconds of music, the model demonstrates good signs of learning (left pane), and is already able to predict reasonably well some of the downbeats in the next 30-second excerpt in the same piece (right pane).

Note that for these experiments: 1) no periodicity is assumed; 2) the system does not require a beat tracker and is actually tempo independent; and 3) the predictor is causal and does, in fact, predict one segment into the future, i.e., about 60–300 ms. The prediction schematic is given in Figure 5-9.

5.3.3 Inter-song generalization

Our second experiment deals with a complex rhythm from the northeast of Brazil called "Maracatu." One of its most basic patterns is shown in standard notation in Figure 5-10. The bass-drum sounds are circled by dotted lines. Note that two out of three are syncopated. A listening test was given to several

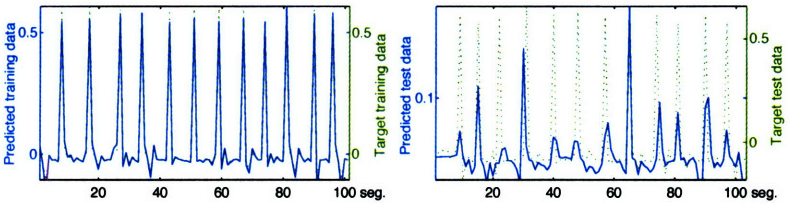


Figure 5-8: Downbeat prediction for James Brown's "I got the feelin'" song. [left] Training data set, including a dozen measures. [right] Test data set: the dotted green line represents the ground truth, the blue line is our prediction. Note that there can be a variable number of segments per measure, therefore the distance between downbeats may vary in the plot.

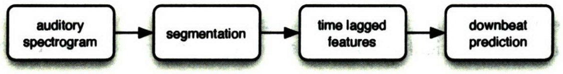


Figure 5-9: Causal downbeat prediction schematic.

musically trained western subjects, none of whom could find the downbeat. Our beat tracker also performed very badly, and tended to lock onto syncopated accents.

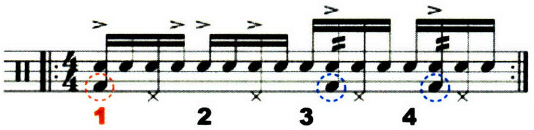


Figure 5-10: Typical Maracatu rhythm score notation.

We train our model with six Maracatu pieces from the band Maracatu Estrela Brilhante. Those pieces have different tempi, and include a large number of drum sounds, singing voices, choirs, lyrics, and a variety of complex rhythmic patterns. Best results were found using an embedded 9-segment feature vector, and a Gaussian kernel for the SVM. We verify our Maracatu "expert" model both on the training data set (8100 data points), and on a new piece from the same album (1200 data points). The model performs outstandingly on the

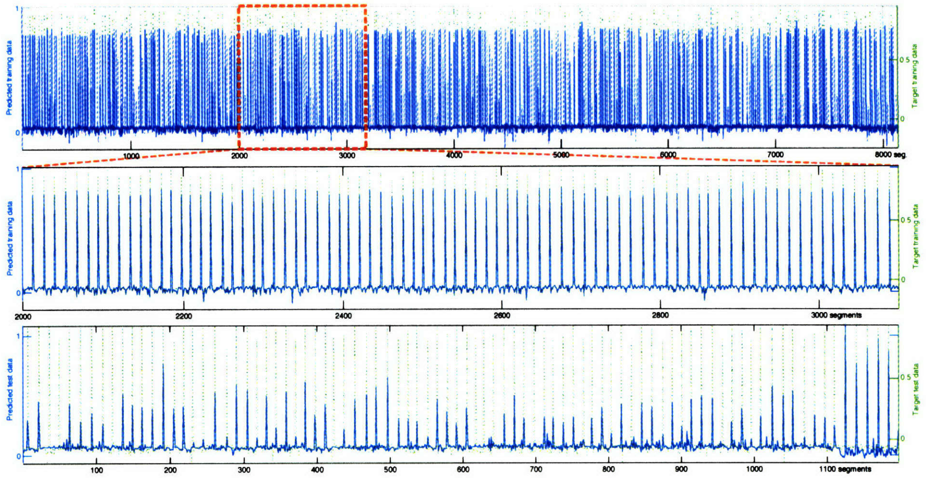


Figure 5-11: Downbeat prediction results for the Maracatu expert model. [top] Full training data set (6 songs) including training downbeat (dotted green) and predicted downbeat (blue). Note that learning is consistent across all data. [middle] Zoom (red section) in the training data to show phase prediction accuracy. [bottom] Results for the new test data (1 song). Note that the last few measures are particularly accurate: most songs in fact tend to end in the same way.

training data, and does well on the new untrained data (Figure 5-11). Total computation cost (including listening and modeling) was found to be somewhat significant at training stage (about 15 minutes on a dual-2.5 GHz Mac G5 for the equivalent of 20 minutes of music), but minimal at prediction stage (about 15 seconds for a 4-minute song).

This experiment demonstrates the workability of extracting the downbeat in arbitrarily complex musical structures, through supervised learning, and without needing the beat information. Although we applied it to extract the downbeat location, such framework should allow to learn and predict other music information, such as, beat location, time signature, key, genre, artist, etc. But this is left for future work.

5.4 Time-Axis Redundancy Cancellation

Repeating sounds and patterns are widely exploited throughout music. However, although analysis and music information retrieval applications are often concerned with processing speed and music description, they typically discard the benefits of sound redundancy cancellation. Our method uses unsupervised clustering, allows for reduction of the data complexity, and enables applications such as *compression* [80].

5.4.1 Introduction

Typical music retrieval applications deal with large databases of audio data. One of the major concerns of these programs is the meaningfulness of the *music description*, given solely an audio signal. Another concern is the efficiency of *searching* through a large space of information. With these considerations, some recent techniques for annotating audio include psychoacoustic preprocessing models [128], and/or a collection of frame-based (i.e., 10–20 ms) perceptual audio descriptors [70] [113]. The data is highly reduced, and the description hopefully relevant. However, although the annotation is appropriate for sound and timbre, it remains complex and inadequate for describing *music*.

In this section, two types of clustering algorithms are proposed: nonhierarchical and hierarchical. In nonhierarchical clustering, such as the k-means algorithm, the relationship between clusters is undetermined. Hierarchical clustering, on the other hand, repeatedly links pairs of clusters until every data object is included in the hierarchy. The goal is to group similar segments together to form clusters whose centroid or representative characterizes the group, revealing musical patterns and a certain organization of sounds in time.

5.4.2 Nonhierarchical k-means clustering

K-means clustering is an algorithm used for partitioning (clustering) N data points into K disjoint subsets so as to minimize the sum-of-squares criterion:

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} |x_n - \mu_j|^2 \tag{5.1}$$

where x_n is a vector representing the n^{th} data point and μ_j is the geometric centroid of the data points in S_j. The number of clusters K must be selected at onset. The data points are assigned at random to initial clusters, and a re-estimation procedure finally leads to non-optimized minima. Despite these limitations, and because of its simplicity, k-means clustering is the most popular clustering strategy. An improvement over k-means, called "Spectral Clustering," consists roughly of a k-means method in the eigenspace [120], but it is not yet implemented.

We start with the segment metadata as described in section 3.7. That MDS space being theoretically normalized and Euclidean (the geometrical distance between two points is "equivalent" to their perceptual distance), it is acceptable to use k-means for a first prototype. Perceptually similar segments fall in the same region of the space. An arbitrary small number of clusters is chosen depending on the targeted accuracy and compactness. The process is comparable to *vector quantization*: the smaller the number of clusters, the smaller the lexicon and the stronger the quantization. Figure 5-12 depicts the segment distribution for a short audio excerpt at various *segment ratios* (defined as the number of retained segments, divided by the number of original segments). Redundant segments get naturally clustered, and can be *coded* only once. The resynthesis for that excerpt, with 30% of the original segments, is shown in Figure 5-14.

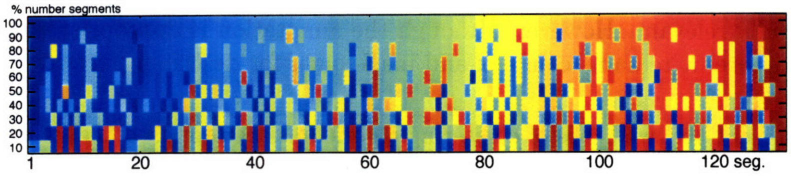


Figure 5-12: Color-coded segment distribution for the 129 segments of the "Watermelon Man" piece by Herbie Hancock, at various segment ratios. 100% means that all segments are represented, while 10% means that only 13 different segments are retained. Note the time-independence of the segment distribution, e.g., here is an example of the distribution for the 13 calculated most perceptually relevant segments out of 129:

33 33 66 66 23 122 23 15 8 112 42 8 23 42 23 15 112 33 33 66 66 66 108 23 8 42 15 8 128 122 23 15 112 33 66 115 66 122 23 15 8 128 42 66 128 42 23 15 112 33 66 115 8 108 23 15 8 42 15 8 128 122 23 115 112 33 66 115 86 128 23 33 115 112 42 8 128 42 23 115 112 8 66 8 66 108 86 15 23 42 15 8 128 122 23 115 112 8 66 115 86 128 23 122 8 112 42 8 108 42 23 115 112 8 66 115 66 108 86 122 23 42 122 23 128 122 23 128 128

One of the main drawbacks of using k-means clustering is that we may not know ahead of time how many clusters we want, or how many of them would ideally describe the perceptual music redundancy. The algorithm does not adapt to the type of data. It makes sense to consider a hierarchical description of segments organized in clusters that have subclusters that have subsubclusters, and so on.

5.4.3 Agglomerative Hierarchical Clustering

Agglomerative hierarchical clustering is a bottom-up procedure that begins with each object as a separate group. These groups are successively combined based on *similarity* until there is only one group remaining, or a specified termination condition is satisfied. For n objects, $n-1$ mergings are done. Agglomerative hierarchical methods produce *dendrograms* (Figure 5-13). These show hierarchical relations between objects in form of a tree.

We can start from a similarity matrix as described in section 4.2.4. We order segment pairs by forming clusters hierarchically, starting from the most similar pairs. At each particular stage the method joins together the two clusters that are the closest from each other (most similar). Differences between methods arise because of the different ways of defining distance (or similarity) between

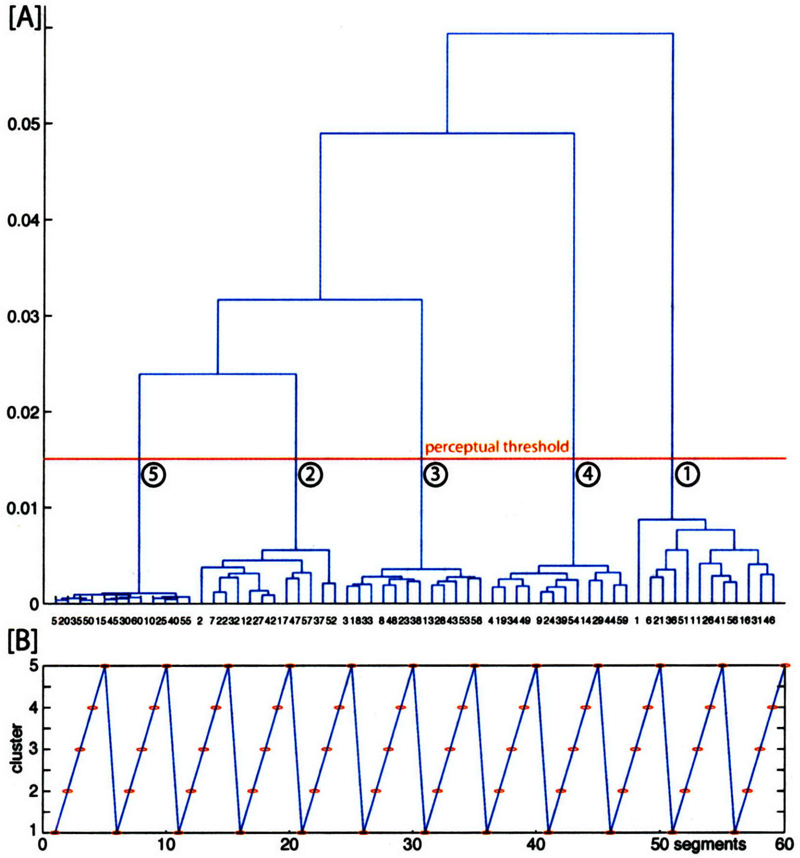


Figure 5-13: [A] Dendrogram of a drum loop made of 5 distinct sounds repeating 12 times. The *perceptual threshold*, selected manually at 0.015, groups the sounds into 5 clusters. Each cluster is composed of 12 sounds. [B] Synthesized time-series of the musical path through the 5 clusters. The 12 loops are recovered.

clusters. The most basic agglomerative model is *single linkage*, also called *nearest neighbor*. In single linkage, an object is linked to a cluster if at least *one* object in the cluster is the closest. One defect of this distance measure is the creation of unexpected elongated clusters, called the "chaining effect." On the other hand, in *complete linkage*, two clusters fuse depending on the most distant pair of objects among them. In other words, an object joins a cluster when its similarity to *all* the elements in that cluster is equal or higher to the considered level. Other methods include average linkage clustering, average group, and Ward's linkage [42].

The main advantages of hierarchical clustering are 1) we can take advantage of our already computed perceptual similarity matrices; 2) the method adapts its number of clusters automatically to the redundancy of the music; and 3) we can choose the level of resolution by defining a similarity threshold. When that threshold is high (fewer clusters), the method leads to rough quantizations of the musical description (Figure 5-13). When it is low enough (more clusters) so that it barely represents the *just-noticeable* difference between segments (a perceptual threshold), the method allows for reduction of the complexity of the description without altering its perception: redundant segments get clustered, and can be *coded* only once. This particular evidence leads to a *compression* application.

5.4.4 Compression

Compression is the process by which data is reduced into a form that minimizes the space required to store or transmit it. While modern lossy audio coders efficiently exploit the limited perception capacities of human hearing in the frequency domain [17], they do not take into account the perceptual redundancy of sounds in the time domain. We believe that by canceling such redundancy, we can reach further compression rates. In our demonstration, the segment ratio indeed highly correlates with the compression rate that is gained over traditional audio coders.

Perceptual clustering allows us to reduce the audio material to the most perceptually relevant segments, by retaining only one representative (near centroid) segment per cluster. These segments can be stored along with a list of indexes and locations. Resynthesis of the audio consists of juxtaposing the audio segments from the list at their corresponding locations (Figure 5-14). Note that no cross-fading between segments or interpolations are used at resynthesis.

If the threshold is chosen too high, too few clusters may result in *musical distortions* at resynthesis, i.e., the sound quality is fully maintained, but the musical "syntax" may audibly shift from its original form. The ideal threshold is theoretically a *constant* value across songs, which could be defined through empirical listening test with human subjects and is currently set by hand. The clustering algorithm relies on our matrix of segment similarities as introduced in 4.4. Using the agglomerative clustering strategy with additional supervised feedback, we can optimize the distance-measure parameters of the dynamic programming algorithm (i.e., parameter h in Figure 4-4, and edit cost as in section 4.3) to minimize the just-noticeable threshold, and equalize the effect of the algorithm across large varieties of sounds.

5.4.5 Discussion

Reducing audio information beyond current state-of-the-art perceptual codecs by structure analysis of its *musical* content is arguably a bad idea. Purists would certainly disagree with the benefit of cutting some of the original material altogether, especially if the music is entirely performed. There are obviously great risks for music distortion, and the method applies naturally better to

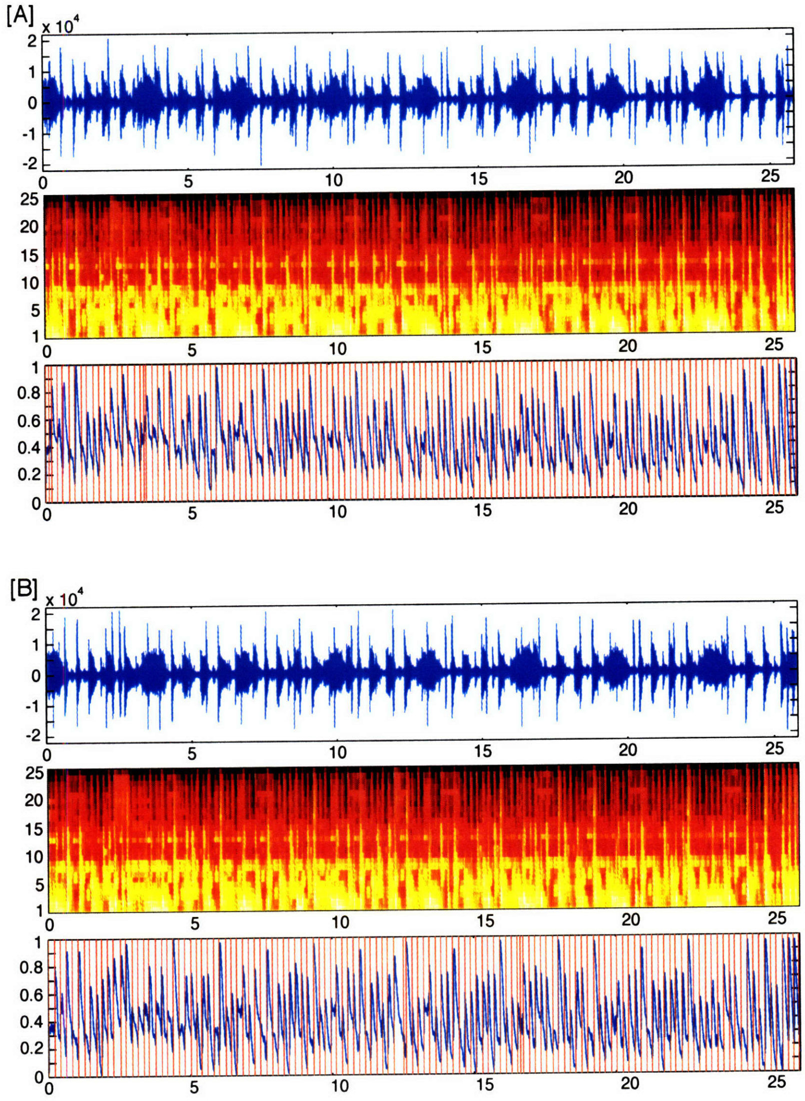


Figure 5-14: [A] 26-second audio excerpt of "Watermelon Man" by Herbie Hancock. From top to bottom: waveform, auditory spectrogram, and loudness curve with segmentation markers (129 segments of around 200 ms). [B] Resynthesis of the piece with about 30% of the segments (less than 8 seconds of audio) determined automatically through agglomerative clustering. From top to bottom: new waveform, new auditory spectrogram, new loudness curve with segmentation markers. Note that there are few noticeable differences, both in the time and frequency domains.

certain genres, including electronic music, pop, or rock, where repetition is an inherent part of its qualities. Formal experiments could certainly be done for measuring the *entropy* of a given piece and *compressibility* across sub-categories.

We believe that, with a real adaptive strategy and an appropriate perceptually grounded error estimation, the principle has great potential, primarily for devices such as cell phones, and PDAs, where bit rate and memory space matter more than sound quality. At the moment, segments are compared and concatenated as raw material. There is no attempt to transform the audio itself. However, a much more refined system would estimate similarities independently of certain perceptual criteria, such as loudness, duration, aspects of equalization or filtering, and possibly pitch. Resynthesis would consist of transforming *parametrically* the retained segment (e.g., amplifying, equalizing, time-stretching, pitch-shifting, etc.) in order to match its target more closely. This could greatly improve the musical quality, increase the compression rate, while refining the description.

Perceptual coders have already provided us with a valuable strategy for estimating the perceptually relevant audio surface (by discarding what we cannot hear). Describing musical structures at the core of the codec is an attractive concept that may have great significance for many higher-level information retrieval applications, including song similarity, genre classification, rhythm analysis, transcription tasks, etc.

CHAPTER SIX
Composing with sounds

"If it sounds good and feels good, then it IS good!"

– Duke Ellington

The motivation behind this analysis work is primarily *music synthesis.* We are interested in composing with a database of sound segments—of variable sizes, usually ranging from 60 to 300 ms—which we can extract from a catalog of musical samples and songs (e.g., an MP3 collection). These segments can be rearranged into *sequences* themselves derived from timbral, harmonic, and rhythmic structures extracted/learned/transformed from existing songs[1]. Our first application, however, does not rearrange short segments, but rather manipulates entire sections of songs by smoothly transitioning between them.

6.1 Automated DJ

A DJ (Disc Jockey) is an individual who selects, mixes and plays pre-recorded music for the enjoyment of others. Often, DJs are expected to demonstrate greater technical aspects of their performance by manipulating the songs they play in order to maintain a given tempo and energy level. "Even simply playing records allows for the DJ to bring his own creative ideas to bear upon the pre-recorded music. Playing songs in sequence offers the opportunity to observe relationships forming between different songs. Given careful attention and control, the DJ can create these relations and encourage them to become more expressive, beautiful and telling" [172]. This is called the art of "programming," or track selection.

[1] Another approach for combining segments consists of using *generative* algorithms.

6.1.1 Beat-matching

Certainly, there is more to the art of DJ-ing than technical abilities. In this first application, however, we are essentially interested in the problem of *beat-matching* and *cross-fading* songs as "smoothly" as possible. This is one of DJs most common practices, and it is relatively simple to explain but harder to master. The goal is to select songs with similar tempos, and align their beat over the course of a transition while cross-fading their volumes. The beat markers, as found in section 3.5, are obviously particularly relevant features.

The length of a transition is chosen arbitrarily by the user (or the computer), from no transition to the length of an entire song; or it could be chosen through the detection of salient changes of structural attributes; however, this is not currently implemented. We extend the beat-matching principle to downbeat matching by making sure that downbeats align as well. In our application, the location of a transition is chosen by selecting the most similar rhythmic pattern between the two songs as in section 4.6.3. The analysis may be restricted to finding the best match between specific sections of the songs (e.g., the last 30 seconds of song 1 and the first 30 seconds of song 2).

To ensure perfect match over the course of long transitions, DJs typically adjust the playback speed of the music through specialized mechanisms, such as a "relative-speed" controller on a turntable (specified as a relative positive/negative percentage of the original speed). Digitally, a similar effect is implemented by "sampling-rate conversion" of the audio signal [154]. The procedure, however, distorts the perceptual quality of the music by *detuning* the whole sound. For correcting this artifact, we implement a *time-scaling* algorithm that is capable of speeding up or slowing down the music without affecting the pitch.

6.1.2 Time-scaling

There are three main classes of audio time-scaling (or time-stretching): 1) the *time-domain* approach, which involves overlapping and adding small windowed fragments of the waveform; 2) the *frequency-domain* approach, which is typically accomplished through phase-vocoding [40]; and 3) the *signal-modeling* approach, which consists of changing the rate of a parametric signal description, including deterministic and stochastic parameters. A review of these methods can be found in [16], and implementations for polyphonic music include [15][94][102][43].

We have experimented with both the time-domain and frequency-domain methods, with certain original properties to them. For instance, it is suggested in [16] to preserve *transients* unprocessed in order to reduce artifacts, due to the granularity effect of windowing. While the technique has previously been used with the phase vocoder, we apply it to our time-domain algorithm as well. The task is relatively simple since the signal is already segmented, as in section 3.4. For each sound segment, we pre-calculate the amount of required *stretch*, and we apply a fixed-size *overlap-add* algorithm onto the decaying part of the sound, as in Figure 6-1. In our frequency implementation, we compute a segment-long FFT to gain maximum frequency resolution (we assume stationary frequency content throughout the segment), and time-scale the strongest partials of the spectrum during decay, using an improved phase-vocoding technique, i.e., by preserving the correlation between phases of adjacent bins for a given partial [97]. Our frequency implementation performs well with harmonic sounds, but does not do well with noisier sounds: we adopt the simple and more consistent time-domain approach.

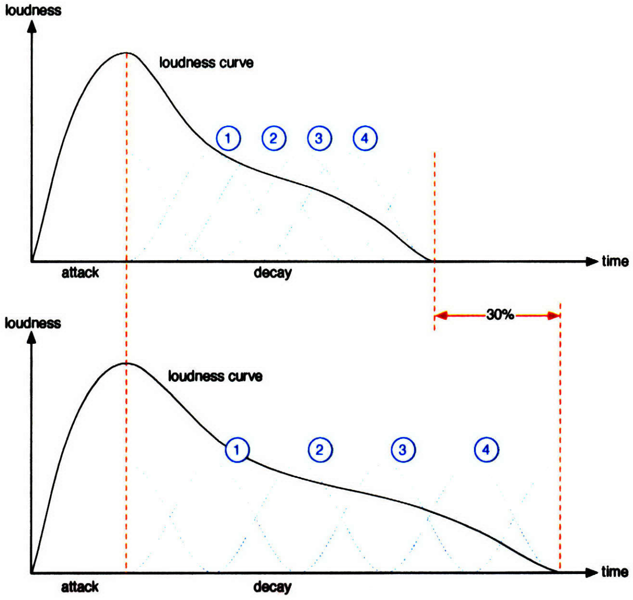


Figure 6-1: Time-scaling example of a typical sound segment. Note that we only process the decay part of the sound. The energy is preserved by overlapping and adding Hanning windows by 50%. In this example we stretch the whole segment [top] by an additional 30% [bottom].

There are several options of cross-fading *shape*, the most common being linear, exponential, S-type, and constant-power. We choose to implement a constant-power cross-fader in order to preserve the perceptual energy "constant" through the transition. This is implemented with a logarithmic fade curve that starts quickly and then slowly tapers off towards the end (Figure 6-2). In order to avoid clipping when two songs overlap, we implement a simple compression algorithm. Finally, our system adapts continuously to tempo variations by interpolating gradually one tempo into another over the course of a transition, and by time stretching every audio segment appropriately to preserve perfect beat alignment (Figure 6-3). Beat-matching and automatic transitions are successfully accomplished for arbitrary songs where tempos differ by up to 40%. Another interesting variant of the application transitions a song with *itself*, allowing for extensions of that song to infinity[2].

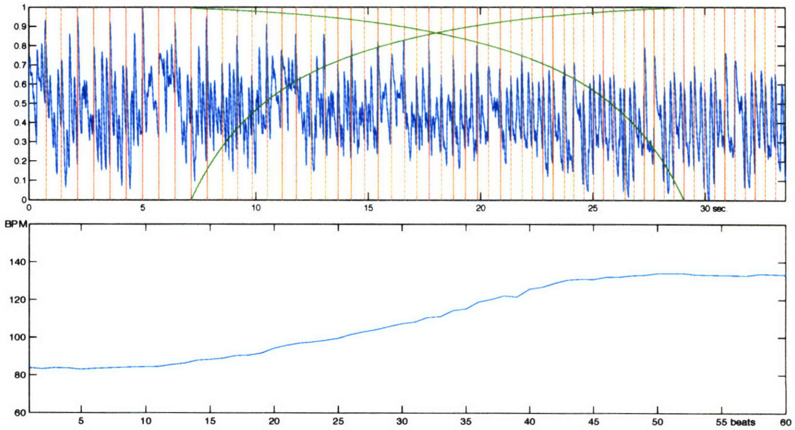


Figure 6-2: Example of beat-matched transition between a funk piece at about 83 BPM and a salsa piece at about 134 BPM. Note in the loudness curve [top] that the overall loudness is transfered smoothly from one song to another. The cross-fading zone and shape are shown in green. The beat markers (red) get closer from each other as the music speeds-up gradually. As a result, the tempo curve increases [bottom].

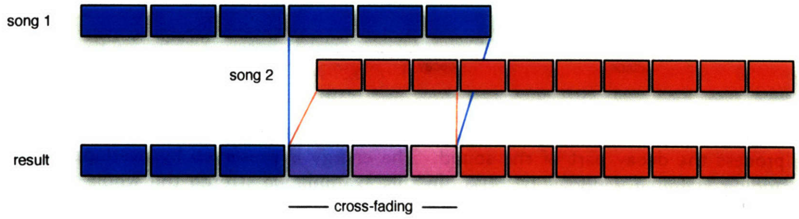


Figure 6-3: Schematic of the beat-matching procedure. Each rectangle represents a musical pattern as found in section 4.6. Rather than adjusting the speed (i.e., time-scaling) of the second song to match the first one, we may choose to continuously adjust both songs during the transition period.

A sound segment represents the largest unit of continuous timbre, and the shortest fragment of audio in our representation. We believe that every segment could

[2] And beyond...

eventually be described and synthesized by analytic methods, such as additive synthesis, but this thesis work is only concerned with the issue of synthesizing *music*, i.e., the structured juxtaposition of sounds over time. Therefore, we use the sound segments found in existing pieces as primitive blocks for creating new pieces. Several experiments have been implemented to demonstrate the advantages of our segment-based synthesis approach over an indeed more generic, but still ill-defined frame-based approach.

6.2 Early Synthesis Experiments

Our synthesis principle is extremely simple: we *concatenate* (or string together) audio segments as a way to "create" new musical sequences that never existed before. The method is commonly used in speech synthesis where a large database of *phones* is first tagged with appropriate descriptors (pitch, duration, position in the syllable, etc.). At runtime, the desired target utterance is created by determining the best chain of candidate units from the database. This method, also known as *unit selection* synthesis, gives the best results in speech synthesis without requiring much signal processing when the database is large enough.

Our concatenation module does not process the audio: there is no segment overlap, windowing, or cross-fading involved, which is typically the case with granular synthesis, in order to avoid discontinuities. Since segmentation is performed psychoacoustically at the most strategic location (i.e., just before an onset, at the locally quietest moment, and at zero-crossing), the transitions are generally artifact-free and seamless.

6.2.1 Scrambled Music

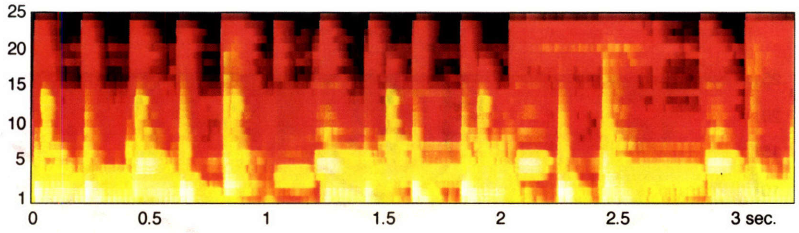


Figure 6-4: A short 3.25 sec. excerpt of "Watermelon man" by Herbie Hancock.

This first of our series of experiments assumes no structure or constraint whatsoever. Our goal is to synthesize an audio stream by randomly juxtaposing short sound segments previously extracted from a given piece of music—about two to eight segments per second with the music tested. During segmentation, a list of pointers to audio samples is created. Scrambling the music consists simply of rearranging the sequence of pointers randomly, and of reconstructing the corresponding waveform. While the new sequencing generates the most unstructured music, and is arguably regarded as the "worst" possible case of music resynthesis, the *event-synchronous* synthesis sounds robust against audio clicks and glitches (Figure 6-5).

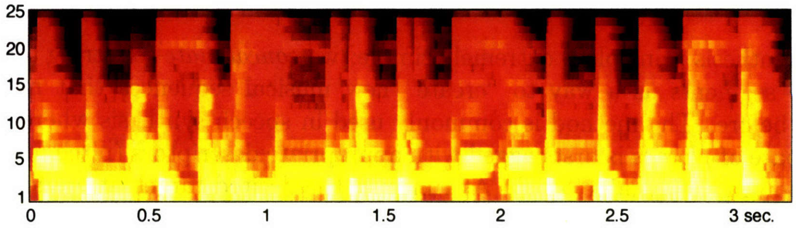


Figure 6-5: Scrambled version of the musical excerpt of Figure 6-4.

The underlying beat of the music, if it exists, represents a perceptual metric on which segments lie. While beat tracking was found independently of the segment organization, the two representations are intricately interrelated with each other. The same scrambling procedure is applied onto the *beat segments* (i.e., audio segments separated by two beat markers). As expected, the generated music is metrically structured, i.e., the beat can be extracted again, but the underlying harmonic, and melodic structure are now scrambled.

6.2.2 Reversed Music

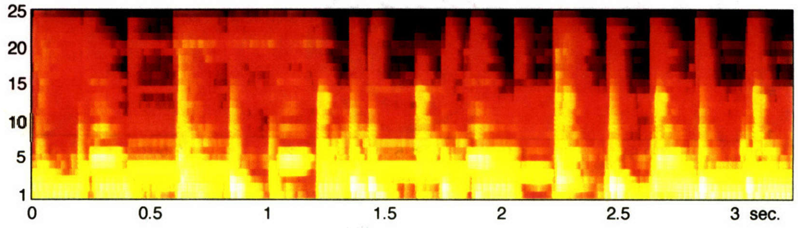


Figure 6-6: Reversed version of the musical excerpt of Figure 6-4.

The next experiment consists of adding a simple structure to the previous method. This time, rather than scrambling the music, the order of segments is reversed, i.e., the last segment comes first, and the first segment comes last. This is much like what could be expected when playing a score backwards, starting with the last note first, and ending with the first one. This is however very different from reversing the audio signal, which distorts "sounds," where reversed decays come first and attacks come last (Figure 6-6). Tested on many kinds of music, it was found that perceptual issues with unprocessed concatenative synthesis may occur with overlapped sustained sounds, and long reverb—certain musical discontinuities cannot be avoided without additional processing, but this is left for future work.

6.3 Music Restoration

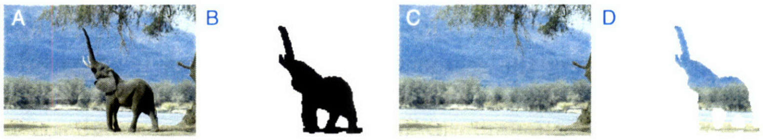


Figure 6-7: Example of "Fragment-Based Image Completion" as found in [41]. [A] original image; [B] region deleted manually; [C] image completed automatically; [D] region that was "made-up" using fragments of the rest of the image.

A well-defined synthesis application that derives from our segment concatenation consists of restoring corrupted audio files, and streaming music on the Internet or cellular phones. Some audio frames may be corrupted, due to a defective hard-drive, or missing, due to lost packets. Our goal, inspired by [41] in the graphical domain (Figure 6-7), is to replace the corrupt region with original new material taken from the rest of the file. The problem differs greatly from traditional restoration of degraded audio material such as old tapes or vinyl recordings, where the objective is to remove clicks, pops, and background noise [58]. These are typically fixed through autoregressive signal models and interpolation techniques. Instead, we deal with localized *digital* corruption of arbitrary length, where standard signal filtering methods do not easily apply. Error concealment methods have addressed this problem for short durations (i.e., around 20 ms, or several packets) [166][156][176]. Our technique can deal with much larger corrupt fragments, e.g., of several seconds. We present multiple solutions, depending on the conditions: 1) file with known metadata; 2) streaming music with known metadata; 3) file with unknown metadata; and 4) streaming music with unknown metadata.

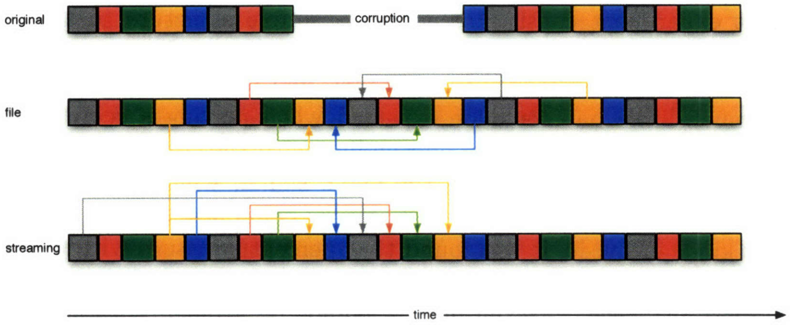


Figure 6-8: Schematic of a restoration application, in the case of a file, or with streaming music (causal). Blocks with same colors indicate audio segments that sound *similar*, although perfect match and metrical organization are not required.

6.3.1 With previously known structure

The metadata describing the segments and their location (section 3.7) is extremely small compared to the audio itself (i.e., a fraction of a percent of the original file). Even the self-similarity matrices are compact enough so that they can easily be embedded in the header of a digital file (e.g., MP3), or sent ahead of time, securely, in the streaming case. Through similarity analysis, we find segments that are most similar to the ones missing (section 4.4), and we concatenate a new audio stream in place of the corruption (Figure 6-9). Knowing the music structure in advance allows us to recover the corrupt region with decent quality, sometimes hardly distinguishable from the original (section 5.4.4). Harder cases naturally include music with lyrics, where the "new" lyrics make no sense. We consider the real-time case: the application is causal, and can synthesize the new music using past segments only. This application applies to streaming music. The quality generally improves as a function of the number of segments available (Figure 6-8).

6.3.2 With no prior knowledge

Two more solutions include not knowing anything about the music beforehand; in such cases, we cannot rely on the metadata file. In a non-real-time process, we can run the full analysis on the corrupt file and try to "infer" the missing structure: the previous procedure applies again. Since detecting regions of corruption is a different problem in and of itself, we are not considering it here, and we delete the noise by hand, replacing it by silence. We then run the segmentation analysis, the beat tracker, and the downbeat detector. We assume that the tempo remains mostly steady during the silent regions and let the beat tracker run through them. The problem becomes a constraint-solving problem that consists of finding the smoothest musical transition between the two boundaries. This could be achieved efficiently through dynamic program-

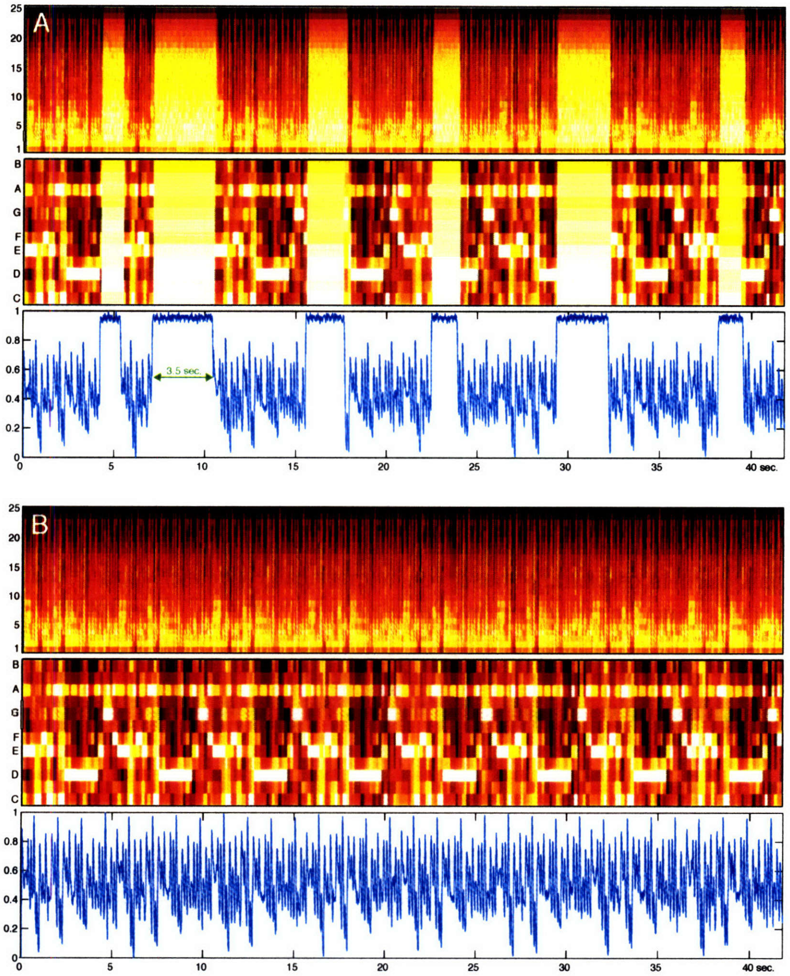


Figure 6-9: Example of "Segment-Based Music Completion" for a 42-second excerpt of "Lebanese Blonde" by Thievery Corporation. [A] is the corrupted music, including timbral, harmonic metadata, and loudness function. We simulate the corruption with very loud grey noise. [B] is the restored music.

ming, by searching for the closest match between: 1) a sequence of segments surrounding the region of interest (reference pattern), and 2) another sequence of the same duration to test against, throughout the rest of the song (test pattern). However, this is not fully implemented. Such a procedure is proposed in [106] at a "frame" level, for the synthesis of background sounds, textural sounds, and simple music. Instead, we choose to fully unconstrain the procedure, which leads to the next application.

6.4 Music Textures

A true autonomous music synthesizer should not only restore old music but should "invent" new music. This is a more complex problem that requires the ability to *learn* from the time and hierarchical dependencies of dozens of parameters (section 5.2.5). The system that probably is the closest to achieving this task is Francois Pachet's "Continuator" [122], based on a structural organization of Markov chains of MIDI parameters: a kind of prefix tree, where each node contains the result of a *reduction function* and a list of continuation indexes.

Our problem differs greatly from the Continuator's in the nature of the material that we compose from, and its inherent high dimensionality (i.e., arbitrary polyphonic audio segments). It also differs in its underlying grouping mechanism. Our approach is essentially based on a metrical representation (tatum, beat, meter, etc.), and on grouping by similarity: a "vertical" description. Pachet's grouping strategy is based on temporal proximity and continuity: a "horizontal" description. As a result, the Continuator is good at *creating* robust stylistically-coherent musical phrases, but lacks the notion of beat, which is essential in the making of popular music.

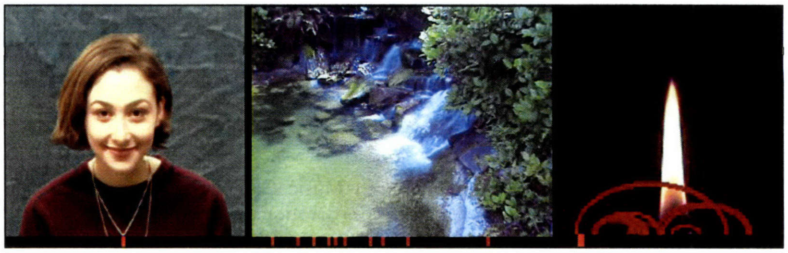


Figure 6-10: Screen shots of three different *video textures* as in [143]: a woman, a waterfall, and a candle. Each movie is made infinite by jumping seamlessly between similar frames at playback (as shown for instance through arcs in the candle image), creating smooth transitions unnoticeable for the viewer.

Our method is inspired by the "video textures" of Schödl et al. in [143], a new type of visual medium that consists of extending short video clips into smoothly infinite playing videos, by changing the order in which the recorded frames are played (Figure 6-10). Given a short musical excerpt, we generate an *infinite* version of that music with identical tempo, that sounds similar, but that never seems to repeat. We call this new medium: "music texture." A variant of this called "audio texture," also inspired by [143], is proposed at the frame level in [107] for textural sound effects (e.g., rain, water stream, horse neighing), i.e., where no particular temporal structure is found.

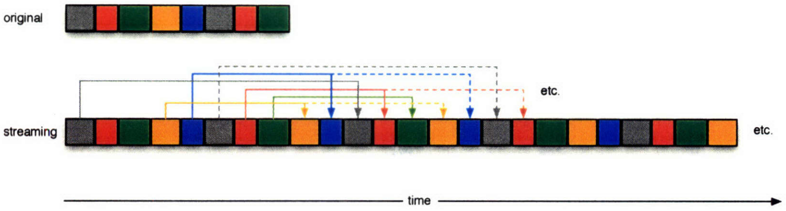


Figure 6-11: Schematic of the music texture procedure. Colors indicate relative metrical-location similarities rather than segment similarities.

Our implementation is very simple, computationally very light (assuming we have already analyzed the music), and gives convincing results. The downbeat analysis allowed us to label every segment with its relative location in the measure (i.e., a float value $t \in [0, L[$, where L is the length of the pattern). We create a music texture by *relative metrical location* similarity. That is, given a relative metrical location $t[i] \in [0, L[$, we select the segment whose relative metrical location is the closest to $t[i]$. We paste that segment and add its duration d_s, such that $t[i+1] = (t[i] + \delta_s) \mod L$, where mod is the modulo. We reiterate the procedure indefinitely (Figure 6-11). It was found that the method may quickly fall into a repetitive loop. To cope with this limitation, and allow for variety, we introduce a tiny bit of jitter, i.e., a few percent of Gaussian noise ε to the system, which is counteracted by an appropriate time stretching ratio c:

$$\begin{aligned} t[i+1] &= (t[i] + c \cdot \delta_s + \varepsilon[i]) \mod L && (6.1) \\ &= (t[i] + \delta_s) \mod L && (6.2) \end{aligned}$$

While preserving its perceive rhythm and metrical structure, the new music never seems to repeat (Figure 6-12). The system is tempo independent: we can synthesize the music at an arbitrary tempo using time-scaling on every segment, as in section 6.1.2. If the source includes multiple harmonies, the system creates patterns that combine them all. It would be useful to impose additional constraints based on continuity, but this is not yet implemented.

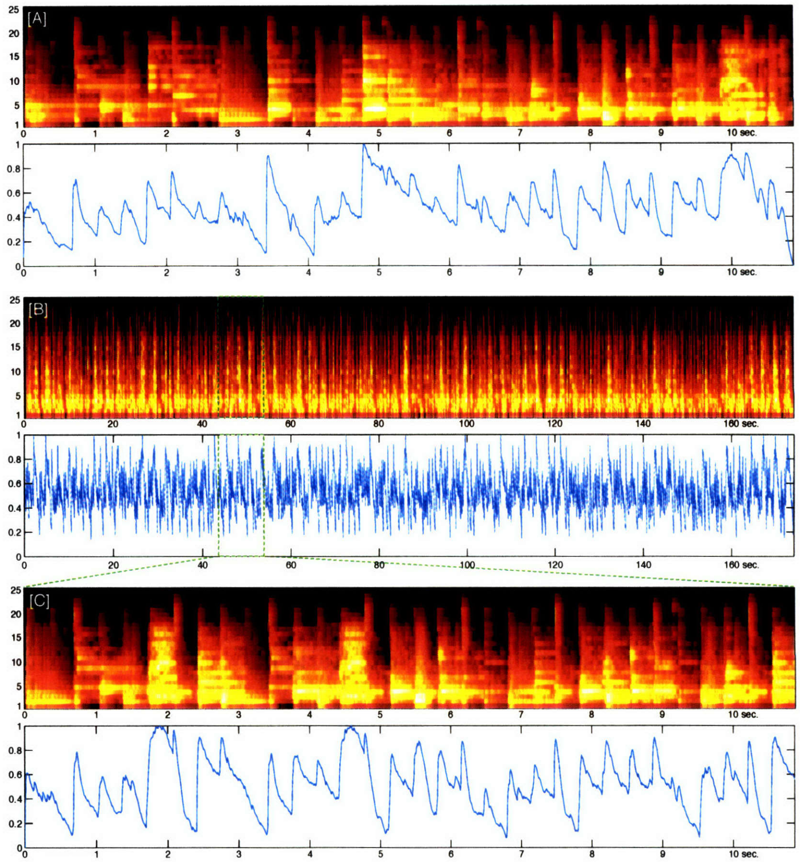


Figure 6-12: [A] First 11 seconds of Norah Jones' "Don't know why" song. [B] Music texture, extending the length of excerpt [A] by 1600%. [C] 11-second zoom in the music texture of [B]. Note the overall "structural" similarity of [C] and [A] (beat, and pattern length), although there is no similar patterns.

Instead, a variant application called "intelligent hold button" only requires *one* pattern location parameter p_{hold} from the *entire* song. The system first *preselects* (by clustering, as in 5.4.3) a number of patterns harmonically similar to the one representing p_{hold}, and then applies the described method to these patterns (equation 6.1). The result is an infinite loop with constant harmony, which sounds similar to pattern p_{hold} but which does not repeat, as if the music was "on hold."

6.5 Music Cross-Synthesis

Cross-synthesis is a technique used for sound production, whereby one parameter of a synthesis model is applied in conjunction with a different parameter of another synthesis model. Physical modeling [152], linear predictive coding (LPC), or the vocoder, for instance, enable *sound* cross-synthesis. We extend that principle to *music* by synthesizing a new piece out of parameters taken from other pieces. An example application takes the music structure description of a *target* piece (i.e., the metadata sequence, or musical-DNA), and the actual sound content from a *source* piece (i.e., a database of unstructured labeled audio segments), and creates a completely new *cross-synthesized* piece that accommodates both characteristics (Figure 6-13). This idea was first proposed by Zils and Pachet in [181] under the name "musaicing," in reference to the corresponding "photomosaicing" process of the visual domain (Figure 6-14).

Our implementation, however, differs from this one in the type of metadata considered, and, more importantly, the event-alignment synthesis method introduced in 6.2. Indeed, our implementation strictly preserves musical "edges," and thus the rhythmic components of the target piece. The search is based on segment similarities—most convincing results were found using timbral and dynamic similarities. Given the inconsistent variability of pitches between two distinct pieces (often not in the same key), it was found that it is usually more meaningful to let that space of parameters be constraint-free.

Obviously, we can extend this method to larger collections of songs, increasing the chances of finding more similar segments, and therefore improving the *closeness* between the synthesized piece and the target piece. When the source database is small, it is usually found useful to primarily "align" source and target spaces in order to maximize the variety of segments used in the synthesized piece. This is done by normalizing both means and variances of MDS spaces before searching for the closest segments. The *search* procedure can be greatly accelerated after a *clustering* step (section 5.4.3), which dichotomizes the space in regions of interest. The hierarchical tree organization of a dendrogram is an efficient way of quickly accessing the most similar segments without searching through the whole collection. Improvements in the synthesis might include processing the "selected" segments through pitch-shifting, time-scaling, amplitude-scaling, etc., but none of these are implemented: we are more interested in the novelty of the *musical artifacts* generated through this process than in the closeness of the resynthesis.

Figure 6-15 shows an example of cross-synthesizing "Kickin' Back" by Patrice Rushen with "Watermelon Man" by Herbie Hancock. The sound segments of the former are rearranged using the musical structure of the latter. The resulting new piece is "musically meaningful" in the sense that its rhythmic structure is preserved, and its timbral structure is made as close as possible to the target piece given the inherent constraints of the problem.

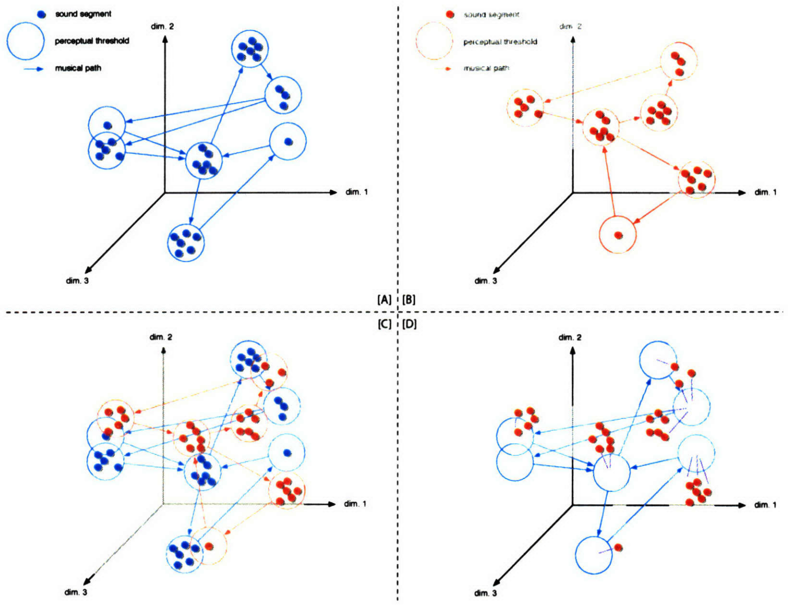


Figure 6-13: Our cross-synthesis application takes two independent songs, as shown in MDS spaces [A] and [B] (section 2.5.4), and as represented together in a common space [C]. A third song is created by merging the musical path of *target* [A] with the sound space of *source* [B], using segment similarity, and concatenative synthesis, as shown in [D].

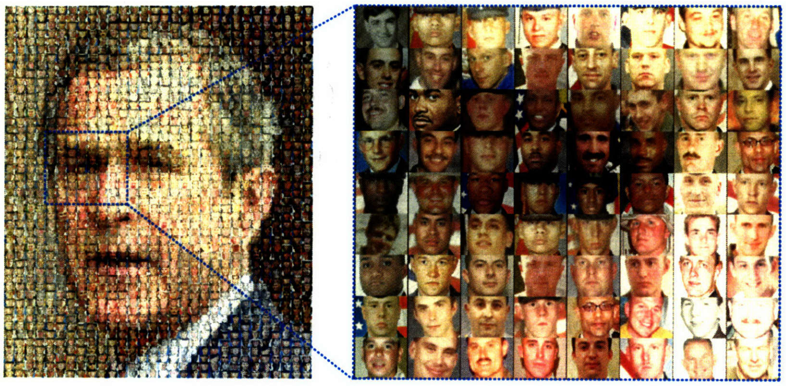


Figure 6-14: Example of *photomosaic* by [86] made out of hundreds of portraits of Americans who have died at war in Iraq during the last few years.

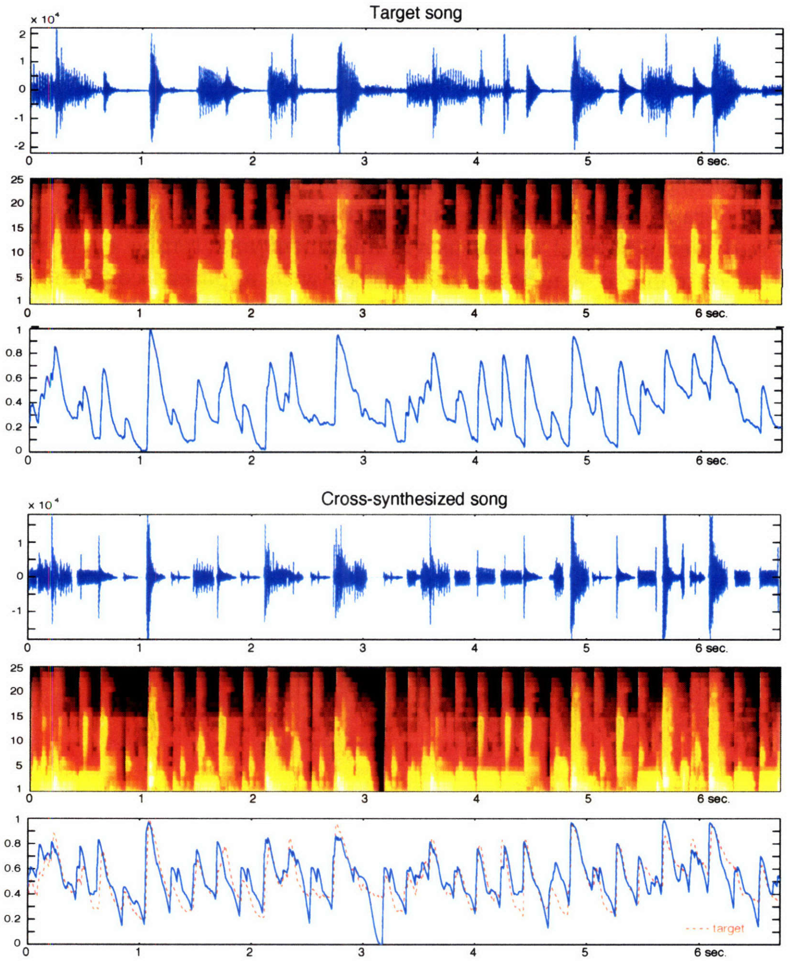


Figure 6-15: Cross-Synthesis between an excerpt of "Kickin' Back" by Patrice Rushen (source) and an excerpt of "Watermelon Man" by Herbie Hancock (target). [top] Target waveform, its auditory spectrogram, and its loudness function. [bottom] Cross-synthesized waveform, its auditory spectrogram, and its loudness function. Note the close timing and spectral relationship between both pieces although they are composed of different sounds.

6.6 Putting it all together

The final goal is to automate the creation of entirely new compositions. Our current solution is to arbitrarily combine the previous applications. Evolutionary methods could help at orienting the system; at selecting a group of sound sources, rhythmic patterns, and song structures; and at transitioning or interpolating between the parameters in such a way that the musical output reflects some of the various elements that constitute it. This part remains to be implemented. However, the system already allows us to build entire pieces with only little guidance from the user. For instance, he or she may select a set of songs that represent the source database, and a set of songs that represent the target musical structures. The system can build arbitrarily long structures, and seamlessly transition between them. It can map groups of sounds to the structures, and merge several creations via alignment and tempo curve-matching. The final outcome is a new sounding piece with apparent coherence in both the use of the sound palette, and the underlying musical structure. Yet, due to inherent constraints of our analysis-resynthesis procedure, the created music is new and different from other existing music.

CHAPTER SEVEN
Conclusion

"I'll play it first and tell you what it is later."

– Miles Davis

This chapter draws conclusions, presents our contributions to the fields of music cognition and automatic music analysis-synthesis, and finally discusses future directions.

7.1 Summary

The goal of this thesis was to computationally model the process of creating music by listening to audio examples. We aimed to close and automate the life cycle of listening, composing, and performing music, only feeding the system with a song database. Our bias-free system has been given generic listening and learning primitives, and was programmed to analyze the sounds and structures of music uniformly from the ground up. It was designed to arbitrarily combine the extracted musical parameters as a way to synthesize new and musically meaningful structures. These structures could be used to drive a concatenative synthesis module that could recycle audio material from the sound database itself, finally creating an original song with convincing quality in sound and form.

7.2 Discussion

Because of its highly subjective nature, it is difficult, if not impossible to evaluate the *quality* of a musical production. Quality is one of those things that cannot be quantified, and it is extremely contextual [134]. The interested readers may judge for themselves by listening to some of the preliminary and intermediary examples produced in the context of this work at: `http://www.media.mit.edu/~tristan/phd/`. However, the work can be considered successful in regards with synthesizing music that expands the database. The new music can be analyzed in its turn, combined with more songs, and recycled again.

Although the goal of the thesis was primarily to create music, most of the work emphasis has been on analyzing and representing music through our music cognition framework (chapters 3, 4, and 5). It was indeed hypothesized that synthesis could share the same knowledge as acquired from a uniform analysis procedure based on perceptual listening and learning. This has been demonstrated in section 5.4.4 with a novel musically-intelligent compression algorithm, and in chapter 6 through a collection of synthesis applications, which rely exclusively on this common metadata representation.

7.3 Contributions

This thesis is a practical implementation of machine intelligence for music analysis and synthesis. However, it is our desire to build a generic perceptual model of music cognition, rather than scattered and self-contained algorithms and techniques. The system was carefully designed to meet theoretical requirements (e.g., causality, psychoacaoustics), and we genuinely avoided using possibly more reliable, although less justifiable, signal processing methods.

7.3.1 Scientific contributions

- The thesis is a contribution to machine intelligence, and more precisely music intelligence. It is shown that a computer program can close the cycle of listening, composing, and performing music through audio signals.
- The data representation resulting from the analysis can be minimal and common to the synthesis data.
- Our analysis is based on a generic four-stage music cognition framework that combines both machine listening and machine learning, augmenting the standard pure signal processing approach of music analysis.
- We present the first implementation of a general and unbiased downbeat predictor. We show for instance that downbeat cannot be computed only from signal processing, and requires training.

- Superior compression can be achieved through the analysis and clustering of redundant audio segments in time.

- Musical signals such as rhythm can be "forecasted" by learning a time-lag embedded feature space.

- This work demonstrates the advantage of a segment-based approach over a still ill-defined frame-based approach in a music analysis task. We highlight the importance of making a distinction between sound and music.

- It is demonstrated that the chroma content of a piece of music can be more accurately estimated, and preserves temporal resolution if the onset and length of the Fourier analysis are synchronized to sound events. It also gains in computation speed.

- Our synthesis is based on a concatenative module that arranges arbitrarily complex sounds in time without overlapping or processing the audio. Its simplicity reveals the benefits of our psychoacoustically grounded segmentation over previous methods.

- We show that music similarities are more adequately represented as a combination of three classes: pitch, rhythm, and timbre, rather than through the generic timbral class.

- The music similarities are computed recursively *after* a perceptual segmentation stage, and are represented hierarchically for each class. As a result, our self-similarity matrices are orders of magnitude smaller, yet musically meaningful.

7.3.2 Engineering contributions

The work and examples presented in this thesis is the result of a stand-alone application named "Skeleton," as described in appendix A. The environment combines a collection of algorithms, player, GUI, database management, and visualizations, which put together allow to test the applications presented in chapter 6. In itself, Skeleton is an engineering contribution that highly facilitates the development of audio applications dealing with machine listening and machine learning technologies.

- Our DJ application leverages the difficulty of mixing music by autonomous tempo and beat alignment, which surpasses semi-automatic current commercial DJ programs.

- The restoration application goes beyond traditional approaches by fixing corrupted sections of up to several seconds. This is made possible on the account of our musical description rather than on the audio signal directly.

- We introduce a new medium called "music texture" that stretches a piece of music to an arbitrary length without affecting its tempo. The system must "invent" new similar music.
- We improve the rendering of a "musical mosaicing" application by selecting segments with perceptual criteria, and by synchronizing these to onsets of the target piece.

7.3.3 Artistic contributions

Throughout the development of this thesis work, we have generated a few hundreds of more or less elaborated musical examples that testify to the artistic potential of our system. Several acclaimed musicians have shown interest in using Skeleton as part of their creative process. The musical artifact can become the source material for a larger production. Others encouraged its immediate application as an interactive art piece: the outcome is the product of an artistic act, which must be attributed either to the user who biases the machine through music examples, the programmer who built the software, the machine itself that synthesizes unpredicted new music, or perhaps a collaboration between these. Such appreciations let us believe that this thesis contributes on the aesthetic and artistic fronts as well.

7.4 Future directions

It goes without saying that this work represents only a few steps in the vast field of machine intelligence for music analysis and synthesis. There is much to be done, if not only by improving the accuracy and robustness of our current algorithms. Here is a short list of the most immediate work that could be done upon our current architecture.

- Although we have experimented with a variety of applications using a few hundreds of musical sources from many genres, when scaling-up such system, several issues of reliability and computation typically come up.
- This thesis only deals with acoustic metadata. Many more valuable contextual results could be achieved by considering other types of metadata.
- We could explore ways of interacting with the system for freely navigating the musical space currently delimited by a database.
- We have not considered sound source separation, transcription, or multichannel music. There are many opportunities for other research ventures based on our representations.
- Although our listening and synthesis methods are fully causal, we have not explored any real-time options on the synthesis front.

- Some of this work has potential venues in more specific areas of the field, including information retrieval, where structure understanding is sometimes key.
- The ultimate synthesizer will not only synthesize music, but will also synthesize the source material. This would lead to much more compact ways of storing and playing music.

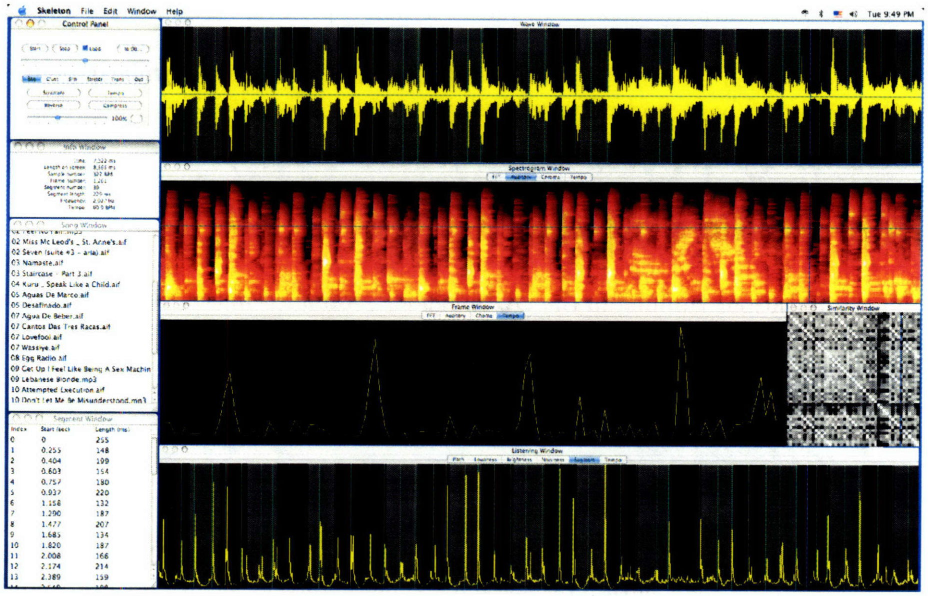
7.5 Final Remarks

Whether machines can be creative or not is certainly not answered in this work. The machine here creates with a small "c," but has no *intention* to do so. And it is not able to evaluate the quality of its own work although it can analytically compare it with others. It is unaware of any social or cultural context, which makes the music somewhat "meaningless." However, the machine is faster than humans at listening to a song database, and at generating original music. They can produce more with less, and are not biased like humans are. Those differences account for the usefulness and potential of computers in creating *new* music.

APPENDIX A
"Skeleton"

"A score is like a skeleton."

– John Zorn



Time: 7,322 ms
Length on screen: 8,365 ms
Sample number: 322,888
Frame number: 1,261
Segment number: 30
Segment length: 220 ms
Frequency: 2,027 Hz
Tempo: 90.0 BPM

02 Miss Mc Leod's _ St. Anne's.aif
02 Seven (suite #3 - aria).aif
03 Namaste.aif
03 Staircase - Part 3.aif
04 Kuru _ Speak Like a Child.aif
05 Aguas De Marco.aif
05 Desafinado.aif
07 Agua De Beber.aif
07 Cantos Das Tres Racas.aif
07 Lovefool.aif
07 Wassiye.aif
08 Egg Radio.aif
09 Get Up I Feel Like Being A Sex Machin
09 Lebanese Blonde.mp3
10 Attempted Execution.aif
10 Don't Let Me Be Misunderstood.mp3

Index	Start (sec)	Length (ms)
0	0	255
1	0.255	148
2	0.404	199
3	0.603	154
4	0.757	180
5	0.937	220
6	1.158	132
7	1.290	187
8	1.477	207
9	1.685	134
10	1.820	187
11	2.008	166
12	2.174	214
13	2.389	159

Figure A-1: Skeleton software screenshot.

The author has developed this work within his own environment named "Skeleton." Skeleton is both a stand-alone Mac OS X application with a simple GUI (Figure A-1), and an API primarily designed to speed up, standardize, and simplify the development of new applications dealing with the analysis of musical signals. Grounded upon fundamentals of perception and learning, the environment consist of machine listening, and machine learning tools, supported by flexible data structures and fast visualizations. It is being developed as an alternative to more generic and slower tools such as Matlab. It is composed of a set of original Objective-C frameworks, and open-source C libraries encapsulated by Objective-C frameworks. The software architecture of Skeleton is depicted in Figure A-2, and is described below:

A.1 Machine Listening

The machine listening software includes: pitch, loudness, brightness, noisiness, Bark, frequency masking, time-domain masking, outer ear, auditory spectrogram, segmentation, tatum, beat, pattern analysis, chromagram. Most of the listening software is also implemented for real-time use in the Max/MSP environment, and is available at: `http://www.media.mit.edu/~tristan/`.

A.2 Machine Learning

The machine learning software includes: dynamic programming, matrix manipulations, distance measures, support vector machines, artificial neural networks, cluster-weighted modeling (mixture of Gaussians), k-means, downbeat prediction, segment, beat, and pattern similarities.

A.3 Music Synthesis

The applications running in the GUI include: scrambled music, reversed music, compression, cross-synthesis, music texture, music restoration, beat-matching and cross-fading.

A.4 Software

The software is written for Macintosh in the native Objective-C language. It includes a GUI built with interface builder, an audio player using Core Audio, fast and flexible displays using Open-GL, fast linear algebra with BLAST, FFT and convolutions with Altivec, read and write audio files using sndLib, MP3 decoding with LibMAD, database management with mySQL, machine learning with packages SVM^{light}, CWM, nodeLib. The hierarchical clustering, as well as certain graphical representations (dendrogram, downbeat, state-space reconstruction) are currently implemented in MATLAB.

A.5 Database

The software creates and connects automatically to a mySQL server, which can store and efficiently retrieve the analyzed data. Songs can be pre-analyzed and the result of their analysis is stored in the database. The various applications typically retrieve the metadata directly from the database. The database initially contains four tables, namely: **SONGS, SEGMENTS, BEATS, PATTERNS** with the following fields:

SONGS: fileName, ID, path, sampleRate, numSamples, numFrames, hopSize, numSegments, numBeats, meanTempo, signature, pitchSignal, loudnessSignal, brightnessSignal, noisinessSignal, segmentSignal, barkSpectrogram

SEGMENTS: songID, ID, startInSec, lengthInMs, sampleIndex, numSamples, loudness, phase, c0 ... c11

BEATS: songID, ID, startInSec, tempo, sampleIndex, numSamples, segmentIndex, numSegments, phase, c0 ... c11

PATTERNS: songID, ID, sampleIndex, numSamples, frameIndex, numFrames, segmentIndex, numSegments, beatIndex, numBeats, c0 ... c11

Each time a new song is being analyzed, it adds one new row to the **SONGS** table, and appends multiple rows to the other tables. It can also create a series of self-similarity matrix tables. Each of these tables contains many columns (as many as there are segments, beats, or patterns in the song) and as many rows.

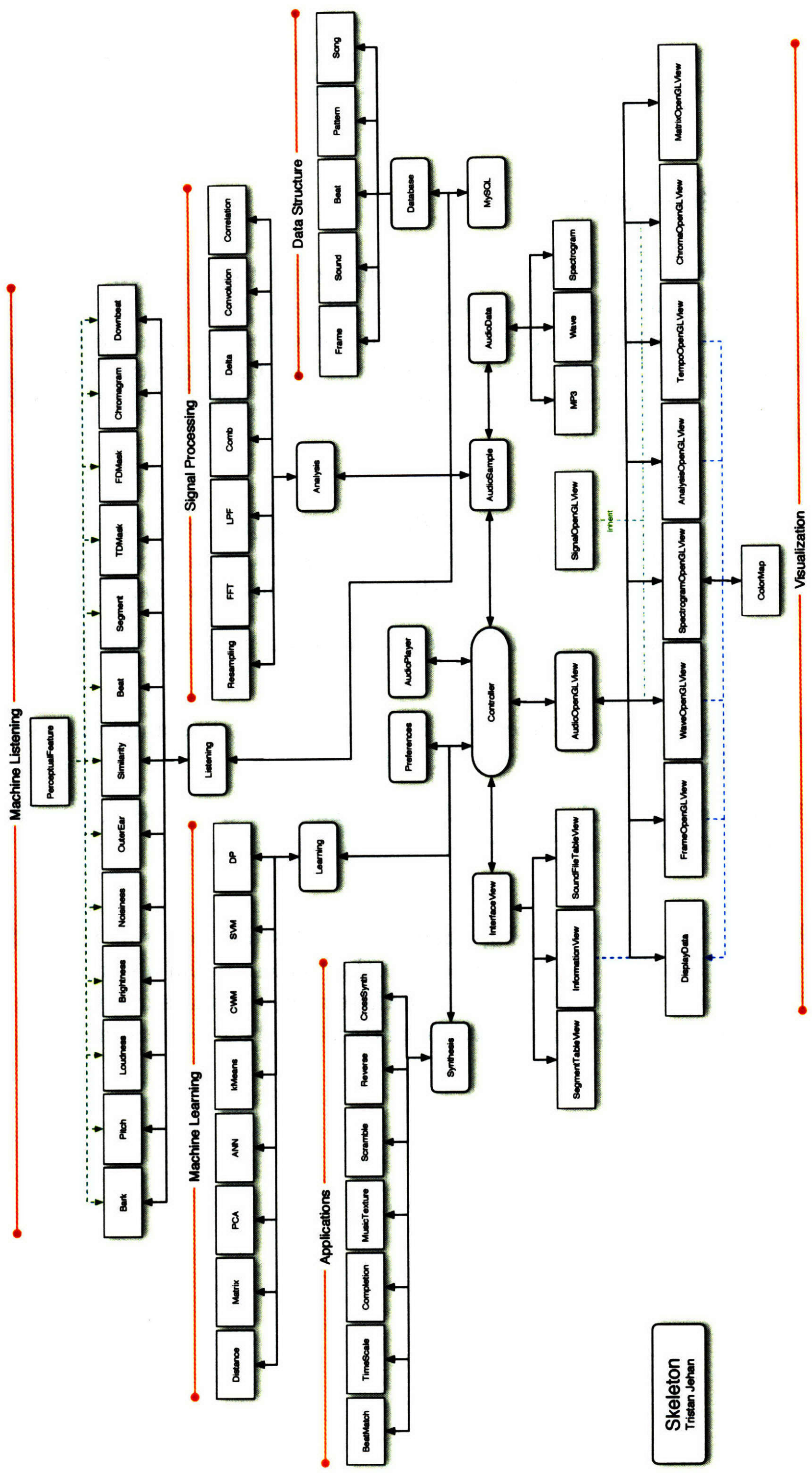


Figure A-2: Skeleton software architecture.

"*I'm saying: to be continued, until we meet again. Meanwhile, keep on listening and tapping your feet.*"

– Count Basie

Bibliography

[1] S. Abdallah and M. Plumbley. Unsupervised onset detection: a probabilistic approach using ICA and a hidden markov classifier. In *Proceedings of Cambridge Music Processing Colloquium*, Cambridge, UK, 2003.

[2] D. Adams. *The Hitchhiker's Guide to the Galaxy.* Pocket Books, New York, NY, 1979.

[3] V. Adan. Hierarchical music structure analysis, modeling and resynthesis: A dynamical systems and signal processing approach. Master's thesis, MIT Media Laboratory, 2005.

[4] M. Alonso, B. David, and G. Richard. Tempo and beat estimation of musical signals. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Barcelona, Spain, October 2004. Universitat Pompeu Fabra.

[5] A. S. Association. American standard acoustical terminology. definition 12.9. timbre, 1960.

[6] J.-J. Autouturier and F. Pachet. Finding songs that sound the same. In *Proceedings of IEEE Workshop on Model based Processing and Coding of Audio.* University of Leuven, Belgium, November 2002. Invited Talk.

[7] J.-J. Autouturier and F. Pachet. Improving timbre similarity: How high is the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1(1), 2004.

[8] M. A. Bartsch and G. H. Wakefield. To catch a chorus: Using chroma-based representations for audio thumbnailing. In *Proceedings of IEEE Wokshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 15–18, Mohonk, NY, October 2001.

[9] S. Baumann and T. Pohle. A comparison of music similarity measures for a p2p application. *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx-03)*, Septembre 2003.

[10] J. P. Bello and M. Sandler. Phase-based note onset detection for music signals. In *proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2003.

[11] A. Berenzweig, D. Ellis, B. Logan, and B. Whitman. A large scale evaluation of acoustic and subjective music similarity measures. In *Proceedings of the 2003 International Symposium on Music Information Retrieval*, Baltimore, MD, October 2003.

[12] J. A. Bilmes. Timing is of essence. Master's thesis, Massachusetts Institute of Technology, 1993.

[13] C. M. Bishop, editor. *Neural Networks for Pattern Recognition.* Clarendon Press, Oxford, 1995.

[14] M. Boden. The creative mind: Myths and mechanisms. *Behavioural and Brain Sciences*, 17(3), 1994.

[15] J. Bonada. Automatic technique in frequency domain for near-lossless time-scale modification of audio. In *Proceedings of International Computer Music Conference 2000*, Berlin, Germany, 2000.

[16] J. Bonada. *Audio Time-Scale Modification in the Context of Professional Post-Production.* PhD thesis, Universitat Pompeu-Fabra, Barcelona, 2002.

[17] M. Bosi and R. E. Goldberg. *Introduction to Digital Audio Coding and Standards.* Kluwer Academic Publishers, Boston, December 2002.

[18] K. Brandenburg and G. Stoll. ISO MPEG-1 audio: A generic standard for coding of high quality digital audio. *Journal of the Audio Engineering Society*, 10:780–792, October 1994.

[19] A. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound.* MIT Press, 1990.

[20] J. Brown and M. Cooke. Computational auditory scene analysis. *Computer Speech and Language*, 8(2):297–336, 1994.

[21] R. G. Brown and P. Y. Hwang. *Introduction to Random Signals and Applied Kalman Filtering.* John Wiley & Sons, New York, 2nd edition, 1991.

[22] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, June 1998.

[23] P. Cano, E. Batlle, T. Kalker, and J. Haitsma. A review of algorithms for audio fingerprinting. In *In International Workshop on Multimedia Signal Processing*, US Virgin Islands, December 2002.

[24] P. Cariani. Neural timing nets for auditory computation. In S. Greenberg and M. S. (Eds.), editors, *Computational Models of Auditory Function*, pages 235–249, Amsterdam, 1999. IOS press.

[25] P. Cariani. Temporal codes, timing nets, and music perception. *Journal of New Music Perception*, 30(2), 2001.

[26] W. Chai. Melody retrieval on the web. Master's thesis, MIT Media Laboratory, 2001.

[27] W. Chai. *Automated Analysis of Musical Structure.* PhD thesis, MIT Media Laboratory, 2005.

[28] W. Chai and B. Vercoe. Music thumbnailing via structural analysis. In *Proceedings of ACM Multimedia Conference*, November 2003.

[29] H. Cohen. The further exploits of AARON, painter. *Stanford Humanities Review, Constructions of the Mind: Artificial Intelligence and the Humanities*, 4(2), 1997.

[30] M. Cooper and J. Foote. Summarizing popular music via structural similarity analysis. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Mohonk, NY, October 2003.

[31] D. Cope. *New Directions in Music.* William C. Brown, Dubuque, Iowa, 1984. 4th edition.

[32] D. Cope. *Experiments in Music Intelligence.* A-R Editions, Madison, WI, 1996.

[33] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (Second Edition).* MIT Press and McGraw-Hill, Cambridge, MA, 2001.

[34] T. Cover and J. Thomas. *Elements of Information Theory.* John Wiley & Sons, Inc., New York, 1991.

[35] T. Dartnall. *Artificial Intelligence and Creativity: an Interdisciplinary Approach.* Kluwer, Dordrecht, 1994.

[36] I. Deliège. Grouping Conditions in Listening to Music: An Approach to Lerdhal and Jackendoff's grouping preferences rules. *Music Perception*, 4:325–360, 1987.

[37] P. Desain and H. Honing. Computational models of beat induction: the rule-based approach. *Journal of New Music Research*, 28(1):29–42, 1999.

[38] S. Dixon. Automatic extraction of tempo and beat from expressive perfomances. *Journal of New Music Research*, 30(1), March 2001.

[39] S. E. Dixon. An empirical comparison of tempo trackers. In *Proceedings of 8th Brazilian Symposium on Computer Music*, Fortaleza, Brazil, July 2001.

[40] M. Dolson. The phase vocoder: a tutorial. *Computer Music Journal*, 10(4):14–27, 1986.

[41] I. Drori, D. Cohen-Or, and H. Yeshurun. Fragment-based image completion. In *Siggraph 2003, Computer Graphics Proceedings*, New York, NY, USA, 2003. ACM Press / ACM SIGGRAPH / Addison Wesley Longman.

[42] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Interscience, New York, 2nd edition, 2000.

[43] C. Duxbury, M. Davies, and M. Sandler. Improved time-scaling of musical audio using phase locking at transients. In *Proceedings of the 112th AES Convention*, Munich, Germany, May 2002.

[44] Ear anatomy, 2004. Medical Encyclopedia website ADAM, accredited by the American Accreditation HealthCare Commission. `http://www.nlm.nih.gov/medlineplus/ency/imagepages/1092.htm`.

[45] D. Eck. A positive-evidence model for rhythmical beat induction. *Journal of New Music Research*, 30(2):187–200, 2001.

[46] D. P. Ellis and J. Arroyo. Eigenrhythms: Drum pattern basis sets for classification and generation. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Barcelona, Spain, October 2004.

[47] D. P. Ellis, B. Whitman, A. Berenzweig, and S. Lawrence. The quest for ground truth in musical artist similarity. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Paris, France, October 2002.

[48] J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[49] H. Fletcher. Auditory patterns. *Rev. Mod. Phys.*, 12:47–55, January 1940.

[50] H. Fletcher and W. Munson. Relation between loudness and masking. *Journal of Acoustic Society of America*, 9(1):1–10, July 1937.

[51] J. Foote and M. Cooper. Automatic audio segmentation using a measure of audio novelty. In *Proceedings of IEEE International Conference on Multimedia and Expo*, pages 452–455, New York, USA, 2000.

[52] J. Foote and M. Cooper. Visualizing musical structure and rhythm via self-similarity. In *Proceedings International Computer Music Conference*, La Habana, Cuba, 2001.

[53] N. A. Gershenfeld and A. S. Weigend. The future of time series: Learning and understanding. In A. S. Weigend and N. A. Gershenfeld, editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pages 1–63, Reading, MA, 1993. Addison–Wesley.

[54] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith. Query by humming: Musical information retrieval in an audio database. In *ACM Multimedia*, pages 231–236, 1995.

[55] M. Gibbs. *Bayesian Gaussian Processes for Regression and Classification*. PhD thesis, University of Cambridge, 1997.

[56] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–269, 1995.

[57] B. R. Glasberg and B. C. J. Moore. A model of loudness applicable to time-varying sounds. *J. Audio Eng. Soc.*, 50:331–342, 2002.

[58] S. Godsill, P. Rayner, and O. Capp'e. Digital audio restoration. In *Applications of Digital Signal Processing to Audio and Acoustics*, Norwell, MA, 1996.

[59] M. Goto. An audio-based real-time beat tracking system for music with or without drum sounds. *Journal of New Music Research*, 30:159–171, 2001.

[60] M. Goto. Smartmusickiosk: music listening station with chorus-search function. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 31–40, November 2003.

[61] M. Goto and Y. Muraoka. Real-time beat tracking for drumless audio signals: Chord change detection for musical decisions. *Journal of Speech Communication*, 27:311–335, 1999.

[62] F. Gouyon, L. Fabig, and J. Bonada. Rhythmic expressiveness transformations of audio recordings: Swing modifications. In *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx-03)*, London, UK, September 2003.

[63] F. Gouyon and P. Herrera. A beat induction method for musical audio signals. In *In Proceedings 4th WIAMIS Special Session on Audio Segmentation and Digital Music*, Queen Mary University, London, 2003.

[64] F. Gouyon, P. Herrera, and P. Cano. Pulse-dependent analysis of percussive music. In *Workshop on Digital Audio Effects, DAFx-98*, pages 188–191, Barcelona, Spain, 1998.

[65] F. Gouyon, P. Herrera, and P. Cano. Pulse-dependent analysis of percussive music. In *Proceedings of the AES 22nd International Conference on Virtual, Synthetic and Entertainment Audio*, 2002.

[66] J. Grey. Timbre discrimination in musical patterns. *Journal of the Acoustical Society of America*, 64:467–472, 1978.

[67] M. Gruhne, C. Uhle, C. Dittmar, and M. Cremer. Extraction of drum patterns and their description within the MPEG-7 high-level-framework. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Barcelona, Spain, October 2004. Universitat Pompeu Fabra.

[68] S. Handel. *LISTENING: An Introduction to the Perception of Auditory Events.* MIT Press, Cambridge, Massachusetts, 1989.

[69] J. Herre, E. Allamanche, and C. Ertel. How similar do songs sound? towards modeling human perception of musical similarities. In *Proceedings of IEEE Wokshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Mohonk, NY, October 2003.

[70] P. Herrera, X. Serra, and G. Peeters. Audio descriptors and descriptor schemes in the context of MPEG-7. In *Proceedings of International Computer Music Conference*, Beijing, China, 1999.

[71] D. Hosmer and S. Lemeshow. *Applied Logistic Regression.* Wiley and Sons, New York, 1989.

[72] X. Huang, A. Acero, and H.-W. Hon. *Spoken Language Processing: A Guide to Theory, Algorithm and System Development.* Prentice-Hall, Englewood Cliffs, N.J., 2001.

[73] A. Hunt and A. Black. Unit selection in a concatenative sound synthesis. In *Proceedings ICASSP*, Atlanta, GA, 1996.

[74] V. S. Iyer. *Microstructures of Feel, Macrostructures of Sound: Embodied Cognition in West African and African-American Musics.* PhD thesis, University of California, Berkeley, 1998.

[75] T. Jaakkola, M. Meila, and T. Jebara. Maximum entropy discrimination. *Advances in Neural Information Processing Systems 12*, 1999.

[76] A. K. Jain, J. C. Mao, and K. M. Moniuddin. Artificial neural networks: A review. *IEEE Computer Special Issue on Neural Computing*, March 1996.

[77] T. Jebara. *Discriminative, Generative and Imitative Learning.* PhD thesis, Massachusetts Institute of Technology, February 2002.

[78] T. Jehan. Music signal parameter estimation. Master's thesis, IFSIC, Rennes, and CNMAT, Berkeley, September 1997.

[79] T. Jehan. Perceptual synthesis engine: an audio-driven timbre generator. Master's thesis, MIT Media Laboratory, September 2001.

[80] T. Jehan. Perceptual segment clustering for music description and time-axis redundancy cancellation. In *Proceedings of the 5th International Conference on Music Information Retrieval*, Barcelona, Spain, October 2004.

[81] T. Jehan. Hierarchical multi-class self similarities. In *Proceedings of IEEE Wokshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Mohonk, NY, October 2005. (submitted).

[82] T. Jehan, T. Machover, and M. Fabio. Sparkler: An audio-driven interactive live computer performance for symphony orchestra. In *Proceedings International Computer Music Conference*, Göteborg, Sweden, 2002.

[83] T. Jehan and B. Schoner. An audio-driven, spectral analysis-based, perceptual synthesis engine. *Journal of the Audio Engineering Society*, 2001.

[84] F. V. Jensen. *An Introduction to Bayesian Networks.* Springer-Verlag, New York, 1996.

[85] T. Joachims. *Learning to Classify Text Using Support Vector Machines.* Kluwer, Boston, MA, 2002.

[86] Joe. War president, 2004. Posted at `http://amleft.blogspot.com` on April 1st.

[87] M. Jordan, editor. *Learning in Graphical Models.* MIT Press, Cambridge, Massachusetts, 1998.

[88] M. Jordan and R. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214, 1994.

[89] H. Kantz. Noise reduction by local reconstruction of the dynamics. In A. Weigend and N. Gershenfeld, editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, pages 475–490, Reading, MA, 1993. Addison–Wesley.

[90] A. Klapuri. Sound onset detection by applying psychoacoustic knowledge. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3089–3092, 1999.

[91] A. P. Klapuri. Musical meter estimation and music transcription. In *In Proceedings Cambridge Music Processing Colloquium*, pages 40–45, Cambridge University, UK, 2003.

[92] A. P. Klapuri, A. J. Eronen, and J. T. Astola. Analysis of the meter of acoustic musical signals. *IEEE Transaction on Speech and Audio Processing (in Press)*, 2005.

[93] T. Kuusi. *Set-Class and Chord: Examining Connection between Theoretical Resemblance and Perceived Closeness.* PhD thesis, Sibelius Academy, 2001.

[94] J. Laroche. Time and pitch scale modification of audio signals. In M. Kahrs and K. Brandenburg, editors, *Applications of Digital Signal Processing to Audio and Acoustics*, pages 279–310. Kluwer Academic Publishers, Boston, 1998.

[95] J. Laroche. Estimating, tempo, swing, and beat locations in audio recordings. In *Proceedings of IEEE Wokshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pages 135–138, Mohonk, NY, October 2001.

[96] J. Laroche. Efficient tempo and beat tracking in audio recordings. *Journal of the Audio Engineering Society*, 51(4):226–233, April 2003.

[97] J. Laroche and M. Dolson. Improved phase vocoder time-scale modification of audio. *IEEE Transactions on Speech and Audio Processing*, 7(3):323–332, May 1999.

[98] A. Lazier and P. Cook. Mosievius: Feature driven interactive audio mosaicing. *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx-03)*, Septembre 2003.

[99] C. S. Lee. *The Perception of Metrical Structure: Experimental Evidence and a Model*, pages 59–127. Academic Press, London, 1991.

[100] F. Lerdahl and R. Jackendoff. *A Generative Theory of Tonal Music.* MIT Press, Cambridge, Mass., 1983.

[101] F. Lerdhal. Timbral hierarchies. *Contemporary Music Review*, 2:135–160, 1987.

[102] S. Levine. *Audio Representations for Data Compression and Compressed Domain Processing.* PhD thesis, CCRMA, Stanford University, 1998.

[103] G. E. Lewis. Too many notes: Computers, complexity and culture in *Voyager*. *Leonardo Music Journal*, 10:33–39, 2000.

[104] B. Lincoln. An experimental high-fidelity perceptual audio coder. Technical report, CCRMA, Stanford University, 1998.

[105] L. Lu, H. Jiang, and H.-J. Zhang. Robust audio classification and segmentation method. In *Proceedings of the 9th ACM International Multimedia Conference and Exhibition*, pages 103–211, 2001.

[106] L. Lu, L. Wenyin, and H.-J. Zhang. Audio textures: Theory and applications. *IEEE Transactions on Speech and Audio Processing*, 12(2), march 2004.

[107] L. Lu, L. Wenyin, H.-J. Zhang, and Y. Mao. Audio textures. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 1761–1764, 2002.

[108] T. Machover. Hyperinstruments: A progress report 1987-1991. Technical report, MIT Media Laboratory, 1992.

[109] M. Maher and J. Beauchamp. Fundamental frequency estimation of musical signals using a two-way mismatch procedure. *Journal of the Acoustical Society of America*, 95(4):2254–2263, 1994.

[110] M. Marolt, A. Kavcic, and M. Privosnik. Neural networks for note onset detection in piano music. In *Proceedings International Computer Music Conference*, Göteborg, Greece, September 2002.

[111] K. D. Martin. *Sound-Source Recognition: A Theory and Computational Model.* PhD thesis, MIT Media Lab, 1999.

[112] S. McAdams. Contributions of music to research on human auditory cognition. In *Thinking in Sound: the Cognitive Psychology of Human Audition*, pages 146–198. Oxford University Press, 1993.

[113] M. McKinney and J. Breebaart. Features for audio and music classification. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Baltimore, MD, October 2003.

[114] E. Métois. *Musical Sound Information: Musical Gestures and Embedding Synthesis*. PhD thesis, MIT Media Lab, 1996.

[115] T. M. Mitchell. *Machine Learning*. The McGraw-Hill Companies, Inc., Singapore, 1997.

[116] B. C. J. Moore. *An Introduction to the Psychology of Hearing*. Academic Press, New York, 1997.

[117] B. C. J. Moore and B. R. Glasberg. A revision of zwicker's loudness model. *Acta Acustica*, 82:335–345, 1995.

[118] Moving Picture Experts Group. The official MPEG website. `http://www.chiariglione.org/mpeg/`.

[119] B. A. Myers. *Peridot: Creating user interfaces by demonstration*, pages 125–153. MIT Press, Cambridge, MA, 1993.

[120] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, Vancouver, Canada, December 2001. MIT Press.

[121] J. Oswald. Plunderphonics' web site: audio piracy as a compositional prerogative, 1999. `http://www.plunderphonics.com`.

[122] F. Pachet. The continuator: Musical interaction with style. In *Proceedings International Computer Music Conference*, Göteborg, Sweden, 2002.

[123] F. Pachet. *Knowledge Management and Musical Metadata*. Idea Group, 2005. In Encyclopedia of Knowledge Management, edited by Diemo Schwarz.

[124] F. Pachet, J.-J. Aucouturier, A. L. Burthe, A. Zils, and A. Beurive. The cuidado music browser: an end-to-end electronic music distribution system. *Multimedia Tools and Applications*, 2004. Special Issue on the CBMI03 Conference.

[125] F. Pachet, G. Westerman, and D. Laigre. Musical data mining for electronic music distribution. In *Proceedings of the 1st WedelMusic Conference*, 2001.

[126] F. Pachet and A. Zils. Evolving automatically high-level music descriptors from acoustic signals. *Springer Verlag LNCS*, 2771, 2003.

[127] T. Painter and A. Spanias. A review of algorithms for perceptual coding of digital audio signals. In *Proceedings of the International Conference of Digital Signal Processing*, pages 179–205, July 1997.

[128] E. Pampalk, S. Dixon, and G. Widmer. Exploring music collections by browsing different views. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Baltimore, MD, October 2003.

[129] M. Parry and I. Essa. Rhythmic similarity through elaboration. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Baltimore, MD, October 2003.

[130] J. Paulus and A. Klapuri. Measuring the similarity of rythmic patterns. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, pages 175–176, Paris, 2002. IRCAM.

[131] S. Pauws. Musical key extraction from audio. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Barcelona, Spain, October 2004. Universitat Pompeu Fabra.

[132] G. Peeters, A. L. Burthe, and X. Rodet. Toward automatic music audio summary generation from signal analysis. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Paris, 2002. IRCAM.

[133] G. Peeters, S. McAdams, and P. Herrera. Instrument description in the context of mpeg-7. In *Proceedings of International Computer Music Conference*, Berlin, Germany, 2000.

[134] R. M. Pirsig. *Zen and the Art of Motorcycle Maintenance.* Morrow, 10th edition, May 1974.

[135] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, New York, 2nd edition, 1992.

[136] T. F. Quatieri. *Discrete-Time Speech Signal Processing, Principles and Practice.* Prentice Hall, Upper Saddle River, NJ, 2002.

[137] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[138] S. Rossignol, X. Rodet, J. Soumagne, J.-L. Collette, and P. Depalle. Automatic characterization of musical signals: Feature extraction and temporal segmentation. *Journal of New Music Research*, 28(4):281–295, 1999.

[139] R. Rowe. *Interactive Music Systems.* MIT Press, 1992.

[140] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, Inc., Upper Saddle River, New Jersey, 1995.

[141] E. Scheirer. Tempo and beat analysis of acoustic musical signals. *Journal of the Acoustic Society of America*, 103(1), January 1998.

[142] E. Scheirer. *Music Listening Systems.* PhD thesis, MIT Media Laboratory, 2000.

[143] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa. Video textures. In K. Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 33–42. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[144] B. Schoner. State reconstructiion for determining predictability in driven nonlinear acoustical systems. Master's thesis, MIT/RWTH Aachen, 1996.

[145] B. Schoner. *Probabilistic Characterization and Synthesis of Complex Driven Systems.* PhD thesis, MIT Media Laboratory, 2000.

[146] M. Schroeder, B. Atal, and J. Hall. Optimizing digital speech coders by exploiting masking properties of the human ear. *Journal of the Acoustical Society of America*, 66:1647–1652, 1979.

[147] D. Schwarz. The caterpillar system for data-driven concatenative sound synthesis. In *Proceedings of the 6th International Conference on Digital Audio Effects (DAFx-03)*, Septembre 2003.

[148] J. Seppänen. Tatum grid analysis of musical signals. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, Mohonk, NY, October 2001.

[149] X. Serra. *A system for Sound Analysis/Transformation/Synthesis Based on a Deterministic Plus Stochastic Decomposition.* PhD thesis, CCRMA, Department of Music, Stanford University, 1989.

[150] X. Serra and J. O. Smith. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):12–24, 1990.

[151] P. Smaragdis. *Redundancy Reduction for Computational Audition, a Unifying Approach.* PhD thesis, MIT Media Lab, 2001.

[152] J. O. Smith. Physical modeling using digital waveguides. *Computer Music Journal*, 6(4), 1992.

[153] J. O. Smith and J. S. Abel. Bark and ERB bilinear transforms. *IEEE Transactions on Speech and Audio Processing*, 7(6):697–708, November 1999.

[154] J. O. Smith and P. Gosset. A flexible sampling-rate conversion method. *International Conference on Acoustics, Speech, and Signal Processing*, 2:19.4.1–19.4.2, 1984.

[155] B. Snyder. *Music and Memory: an Introduction.* MIT Press, Cambridge, MA, 2000.

[156] A. Stenger, K. Younes, R. Reng, and B. Girod. A new error concealment technique for audio transmission with packet loss. In *Proceedings European Signal Processing Conference (EUSIPCO 96)*, pages 1965–1968, Trieste, Italy, September 1998.

[157] F. Takens. Detecting strange attractors in turbulence. In D. Rand and L. Young, editors, *Dynamical Systems and Turbulence*, volume 898 of *Lecture Notes in Mathematics*, pages 366–381, New York, 1981. Springer-Verlag.

[158] A. S. Tanguiane. *Artificial Perception and Music Recognition.* Springer-Verlag, New York, 1993.

[159] D. Temperley. *The Cognition of Basic Musical Structures.* MIT Press, Cambridge, 2001.

[160] E. Terhardt. Calculating virtual pitch. *Hearing Research*, 1:155–182, 1979.

[161] G. Tzanetakis. *Manipulation, Analysis, and Retrieval Systems for Audio Signals.* PhD thesis, Princeton University, June 2002.

[162] G. Tzanetakis and P. Cook. Multifeature audio segmentation for browsing and annotation. In *Proceedings IEEE Workshop on applications of Signal Processing to Audio and Acoustics*, October 1999.

[163] G. Tzanetakis and P. Cook. Automatic musical genre classification of audio signals. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Bloomington, USA, October 2001.

[164] V. N. Vapnik. *Estimation of Dependences Based on Empirical Data [in Russian].* Nauka, Moscow, 1979. (English translation: Springer-Verlag, New York, 1982).

[165] B. Vercoe. Computational auditory pathways to music understanding. In I. Deliège and J. Sloboda, editors, *Perception and Cognition of Music*, pages 307–326. Psychology Press, Hove, UK, 1997.

[166] Y. Wang. A beat-pattern based error concealment scheme for music delivery with burst packet loss. In *Proceedings International Conference Multimedia and Expo (ICME'01)*, pages 73–76, 2001.

[167] M. Welsh, N. Borisov, J. Hill, R. von Behren, and A. Woo. Querying large collections of music for similarity. Technical report, Computer Science Division, University of California, Berkeley, 1999.

[168] D. L. Wessel. Timbre space as a musical control structure. *Computer Music Journal*, 3(2):45–52, 1979. Republished in *Foundations of Computer Music*, Curtis Roads (Ed., MIT Press).

[169] B. Whitman. *Learning the Meaning of Music.* PhD thesis, MIT Media Laboratory, 2005.

[170] B. Whitman and S. Lawrence. Inferring descriptions and similarity for music from community metadata. In *Proceedings International Computer Music Conference*, pages 591–598, Göteborg, Greece, September 2002.

[171] B. Whitman and R. Rifkin. Musical query-by-description as a multiclass learning problem. In *Proceedings of the IEEE Multimedia Signal Processing Conference*, St. Thomas, USA, December 2002.

[172] Wikipedia website. Definition of: Disc Jockey, May 2005. `http://www.wikipedia.org/wiki/DJ`.

[173] T. Winkler. *Composing Interactive Music: Techniques and Ideas Using Max*. MIT press, 1998.

[174] M. Wright, A. Chaudhary, A. Freed, S. Khoury, and D. Wessel. Audio applications of the sound description interchange format standard. In *Proceedings of the 107th AES Convention*, New York, New York, September 1999.

[175] M. Wright and A. Freed. OpenSound control: A new protocol for communicating with sound synthesizers. In *Proceedings International Computer Music Conference*, pages 101–104, Thessaloniki, Greece, 1997.

[176] L. Wyse, Y. Wang, and X. Zhu. Application of a content-based percussive sound synthesizer to packet loss recovery in music streaming. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 335–338, Berkeley, CA, USA, 2003. ACM Press.

[177] J. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. *Advances in Neural Information Processing Systems*, 2000.

[178] T. Yoshioka, T. Kitahara, K. Komatani, T. Ogata, and H. G. Okuno. Tempo and beat estimation of musical signals. In *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Barcelona, Spain, October 2004. Universitat Pompeu Fabra.

[179] G. U. Yule. On a method of investigating periodicities in disturbed series with special reference to Wolfer's sunspot numbers. *Philosophical Transactions Royal Society London Ser. A*, 226:267–298, 1927.

[180] D. Zicarelli. An extensible real-time signal processing environment for Max. In *Proceedings International Computer Music Conference*, pages 463–466, Ann Arbor, Michigan, 1998.

[181] A. Zils and F. Pachet. Musical mosaicing. In *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFx-01)*, Limerick, Ireland, December 2001.

[182] A. Zils and F. Pachet. Automatic extraction of music descriptors from acoustic signals using eds. In *Proceedings of the 116th AES Convention*, May 2004.

[183] E. Zwicker and H. Fastl. *Psychoacoustics: Facts and Models*. Springer Verlag, Berlin, 2nd edition, 1999.