Timing is of the Essence:
Perceptual and Computational Techniques for Representing, Learning, and Reproducing Expressive Timing in Percussive Rhythm

by

Jeffrey Adam Bilmes (Jeff Bilmes)

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning
on August 6, 1993, in partial fulfillment of the
requirements for the degree of
Master of Science

Abstract

This thesis has one main goal: design algorithms that computers can use to produce expressive sounding rhythmic phrases. First, I describe four elements that can characterize musical rhythm: metric structure, tempo variation, deviations, and ametric phrases. The first three elements can be used successfully to model percussive rhythm.

Second, I describe two algorithms: one, an automatic transcription algorithm, extracts stroke attack times and automatically constructs unique stroke types from a percussive performance. The other takes a percussive performance and factors out the metric structure, tempo variation, and deviations.

Third, I apply these algorithms to a performance given by the percussion group Los Muñequitos de Matanzas. Using both a synthesis of the performance and statistical analysis, I demonstrate that timing data represented in this form is not random and is in fact meaningful. In a synthesis with tempo variation removed but deviations retained, the original performance's expressive feel is preserved. Therefore, I claim that rhythmic analysis requires the study of both tempo variation and deviations.

Finally, because similar quantized rhythmic phrases have similar corresponding deviations, the smoothness assumption necessary for a function approximation approach to learning is satisfied. I describe a multi-stage clustering algorithm that locates sets of similar quantized phrases in a performance. I then describe a machine learning algorithm that can build a mapping between quantized phrases and deviations. This algorithm can be used to apply deviations to new phrases.

I claim that deviations are most important for the expressive feel of percussive music. Therefore, I have developed a new drum machine interface, a deviation experimentation program, with which deviations can be explored.

Timing is of the Essence: Perceptual and Computational Techniques for Representing, Learning, and Reproducing Expressive Timing in Percussive Rhythm

by

Jeffrey Adam Bilmes

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1993

Author .....
Program in Media Arts and Sciences
August 6, 1993

Certified by .....
Barry L. Vercoe
Professor, Media Arts and Science
Thesis Supervisor

Accepted by .....
Stephen A. Benton
Chairman, Departmental Committee on Graduate Students

Timing is of the Essence: Perceptual and Computational Techniques for Representing, Learning, and Reproducing Expressive Timing in Percussive Rhythm

by

Jeffrey Adam Bilmes

Thesis Readers

Thesis Reader .....

Michael Jordan

Associate Professor

MIT Department of Brain and Cognitive Science

Thesis Reader .....

Marvin Minsky

Toshiba Professor of Media Arts and Sciences

MIT Media Lab

Acknowledgments

Many friends and colleagues have greatly contributed to the quality of this work. I thank all of you for your help and comments.

First, this thesis and the project it describes would not have been possible without the support of my supervisor, Barry Vercoe. His enthusiasm, sincerity, perspicacity, and advice were always enormous assets.

It was an honor and privilege to use a performance given by Los Muñequitos de Matanzas for my analysis. This event would have been impossible without abundant helpful activity from the following people: Kathryn Vaughn, W. Andrew Schloss, Barry Vercoe, and Scott Wardinsky. I would also like to thank the participating members of Los Muñequitos for contributing their enormous talents to this project: Gregorio Diaz Alfonso, Jesús Alfonso Miró, Diosdado Ramos Cruz, Ricardo Cané Gómez, Israel Berriel Gonzalez, Agustín Diaz Cano, Israel Berriel Jimenez, Alberto Romero Diaz, and Rafael Navarro Pujada. The recording process would have been very difficult without help from Greg Tucker and Andy Hong. These people all gave me the opportunity to work with unsurpassable timing data. Many thanks to you all.

I have had extremely useful discussions with many people. Dan Ellis has been invaluable for his knowledge of signal processing. Few people have such intuition and mastery in their field. In addition, it was always productive to discuss an idea or problem with Chris Popat – rare is a person who is extremely talented both in science and elucidation. I must also thank my successive office mates, Beerud Sheth and Mike Casey, not only for being valuable people to discuss ideas with, but also for tolerating loud percussive music in the office. Several people reviewed copies of this thesis and provided valuable suggestions, including Tina Perry, Dan Ellis, David P. Anderson, and Rosalind Picard. Several other people have also contributed to this project in various ways including Pawan Sinha, David Wessel, and C.K. Ladzekpo. Special thanks must go out to Tina Perry for being extraordinarily warm and understanding throughout this entire process.[1]

The MIT Media Laboratory has been an ideal working environment. Not only does it have vast computer facilities, but also it maintains a stimulating atmosphere in which both rigorous research and remarkable imagination thrive. I am aware of no laboratory better for conducting this research.

I must also thank the force behind the music and cognition reading group: Judy Brown, Mike Casey, Dan Ellis, Bill Gardner, Stephen Gilbert, Eric Metois, David Rosenthal, Alan Ruttenberg, Kathryn Vaughn, Barry Vercoe, and David Yadegari. These people created a fertile environment and provided a medium in which I could test out new ideas and honestly conduct music and cognition research. May the group grow and prosper.

Other members of the Media Laboratory's music and cognition group have contributed to its unique quality and flavor including Janet Cahn, Joe Chung, Ken Haase, Mike Hawley, Andy Hong, Tod Machover, Pattie Maes, Tom Maglione, Fumi Matsumoto, Betty Lou Mcclanahan, Marvin Minsky, and Beerud Sheth. I thank them all for their contributions.

This thesis is dedicated to my brother Daniel Gregory Bilmes. Forever will he be remembered.

[1] And for never letting me put a footnote mark before a period.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> Picard*: The good doctor was kind enough to provide me with a recording of your concert. Your performance shows – – feeling.*
> Data*: As I have recently reminded others, sir, I have no feeling.*
> Picard*: It's hard to believe. Your playing is quite beautiful.*
> Data*: Strictly speaking, sir, it is not my playing. It is a precise imitation of the techniques of Jascha Heifetz and Trenka Bronkjen.*
> Picard*: Is there nothing of Data in what I'm hearing? You see, you chose the violinists. Heifetz and Bronkjen have radically different styles, different techniques, and yet, you combined them – – successfully.*
> Data*: I suppose I have learned to be – – creative, sir – – when necessary.*
>
> ST:TNG[1]
> *The Ensign of Command*

Robots, Star Ships, The Speed of Light, Creativity. In space, everything has an easy solution. Science fiction movies have long tantalized those of us trying to understand these phenomena. Although they encourage the advancement of science, they often trivialize the process, deodorizing and discounting the sweat through which solutions are found. Data merely combines the styles of Heifetz and Bronkjen. What could be easier?

What they do not tell us is *how*. How exactly is Data able to be "creative"? How is he able to combine the styles of the violinists, "successfully"? What is it in the music that he finds imitable? What features can he recognize and exploit to create his own performance? At the very least, he must have a special capability to extract a representation

[1] From Star Trek: The Next Generation. Data, an android, has completed a concert playing violin. He and Picard, the captain of the ship and an aficionado of fine music, are discussing this performance. Presumably, the birth date of *Trenka Bronkjen* is beyond the 20th century.

of the performers' style, a capability that lends itself to manipulation and combination and that perfectly describes their melody, harmony, and rhythm.

Those are the features he must understand: melody, harmony, and rhythm – they are each distinctive. They effect each other in many ways. But how do they effect each other? We need more than just a representation, we need a model. A good model is one that can accurately represent a phenomenon. A better model is additionally one the manipulation of which produces valid stylistic changes within the phenomenon. With both criteria in mind, in this thesis I address the problem of representing and reproducing *expressive* timing in musical rhythm. My primary goal is this: produce human-like expressive rhythmic phrases by computer.

Duplicating expressivity in musical performance is quite an elusive task. Duplicating human intelligence using a computer, however, is perhaps even more elusive. It is my hope that building a system that to some degree simulates musical expressivity might inform us about the way humans think. Perhaps we can learn about temporal perception, audition, or perhaps even emotion. Whatever we learn, a music system that successfully duplicates expressivity might shed some light onto nebulous concepts such as aesthetics[Min81].

There is at least one obvious similarity between the problem of understanding musical expression and that of understanding human intelligence. That is, performance evaluation. Music exists in all cultures and there are indisputably different types of music and different performances of the same style that are considered more or less expressive, a word that embraces a number of aesthetic musical qualities. Although I do not plan to delve into aesthetics, I believe (and will base my conclusions on this) that if a group of critical listeners evaluate a piece of music and reach a consensus that it contains expression, we will have satisfied the goal of simulating expression. This is a musical *Turing test*. The similarities are unequivocal.

1.1 Applications

Of what use is the successful reproduction of musical expression? First, current commercial drum machines and music sequencers represent music either by using multiples of a time quantum for inter-onset time intervals or by recording note attack times to within some time resolution. They attempt to produce a *swing* feel, but they sound artificial and cold.

The producers of these machines stand in great need of a representation that can facilitate the production of expressive rhythmic phrases. Second, in music schools, there is demand for computer programs that can teach students a rhythmic feel. A successful implementation of such a system could provide quantitative performance evaluation. Third, it would be enlightening to explore, from an evolutionary standpoint, why musical expressivity evolved. Is there some physiological reason? Did it arise out of intended or unintended fluctuations? Finally, it is interesting to note that expressive rhythm and dance are closely related. How and why does expressive rhythm stimulate our pleasure centers, and in some cases, provoke movement or dance? Understanding rhythmic expression might help decipher some of these riddles.

1.2 Scope

My investigation of rhythmic expressivity involved the utilization of techniques from many different fields including signal processing, computer science, statistics, cluster analysis, pattern recognition, and neural networks. A brief outline of the thesis follows.

Chapter 2 begins by describing my general rhythmic philosophy. It defines what I believe to be the main elements that characterize musical rhythm. It defines deviations, and asserts that they are extremely important for representing percussive rhythm. Further, it defines the concept of *tatum*, used often throughout this work. Chapter 3 consists primarily of low-level analysis. It defines a new method of automatic percussive music transcription that produces a list of attack times and pitch types from an audio signal. Transcription, however, was not my end goal, and the timing data obtained from these techniques is used in subsequent analysis. Chapter 4 defines a timing analysis algorithm that takes attack times and pitch types as defined in Chapter 3, and produces the elements described in Chapter 2. The algorithm is applied to a real performance with some very interesting results. Specifically, the elements are shown to have captured the performance's expressivity, and that the rhythmic representation contains useful data. This is demonstrated by statistical analysis and by synthesizing the performance in various ways. Chapter 5 describes a system that can be used to learn the expressive features in rhythmic phrases and can add expression to lifeless rhythmic phrases. Chapters 3, 4, and 5 are depicted in Figure 1-1. Last, Chapter 6 describes what remains to be done and suggests possible future projects.

Figure 1-1: Thesis Outline.

It is my hope that this research will open the gates for others to explore these avenues.

In summary, we first introduce a new rhythmic philosophy and representation. Next, we obtain timing data from a real percussive performance. A new rhythmic representation is extracted from the timing data. Finally, a system is defined that learns expressive timing, and can add expressivity to bland rhythmic phrases.

1.3 Symbolic versus Continuous

Historically, musical information processing has come in two forms: *symbolic* and *continuous*. Symbolic musical information processing operates on a discrete domain. Here, the main concept is quantization and there are three types. Time quantization is most common. This is when the duration between note events is a multiple of some time quantum. There is also pitch quantization, in which an instrument is represented by a small subset of the pitches it can produce. Finally, tempo quantization is when there are instantaneous jumps in tempo at various positions in a musical piece. Musical counterpoint, traditional musical harmony, grammar approaches to musical representation, and unembellished standard musical notation are all examples of representations used in symbolic musical information processing. On the other hand, continuous musical information processing operates on a continuous domain. Herein lies the expression. Here is where we find accelerando, rubato, accentuation, pause, timbre variation, crescendo, etc. Here is where we find music's life. I claim, however, that we need both. To successfully model music, we need to study both the symbolic and the numerical aspects of music. Throughout this thesis, I keep this in mind. So, let us begin.

Chapter 2

Rhythm

Rhythm is one of the principle translators between dream and reality. Rhythm might be described as, to the world of sound, what light is to the world if sight. It shapes and gives new meaning.

Dame Edith Sitwell [1887-1964]
The Canticle of the Rose [1949]

Rhythm may be defined as the movement in time of individual sounds. ...however, rhythm is not only the whole feeling of movement in music, but also the dominant feature. ...[It] provides the regular pulsation or beat which is the focal point in uniting the energies of the entire community in the pursuit of their collective destiny.

C.K. Ladzekpo
Foundation Course in African Dance Drumming[Lad89]

The perception of ordered movement in time, an apt and well-known definition. Consider the obvious analogy between tempo and velocity. It seems fine, right? Perhaps not. This definition neglects rhythm's original function. It ignores the reason we evolved to the point where we could perceive rhythm. The goal of musical rhythm is to *unify*. It is a reconciliatory force that transforms the cacophony of isolated individuals into the harmonious collectivity of an ensemble. Musical rhythm is more than just movement, it is synchronized movement. Herein, I try to remember rhythm's original purpose and strive to better understand it.

2.1 Rhythmic Elements

I claim that four elements constitute musical rhythm: *metric structure*, *tempo variation*, *deviations*, and *ametric phrases*. Each of these elements can be studied separately and will be described presently. Percussive rhythm, however, is musical rhythm in its purest form. It is rich in expression without the distraction of melody and harmony. As we will see, the first three elements are enough to represent most percussive rhythm. They will be discussed extensively.

Metric structure in musical rhythm is a symbolic representation. It can be considered both perceptually and representationally. Perceptually, musical metric structure comprises beats and hierarchies. Beats constitute the framework in which successive musical events are perceived. They make up the isochronous sequence of elements which occur in music. They are the most fundamental musical form and were perhaps originally intended to emulate the heart beat, the sound of walking, or beating a rock with a stick. Beats establish the musical tactus,[1] they give us something to synchronize our tapping foot with, and they are usually the pulse that inspires us to dance. Beats are rarely solitary, however; they are arranged in groups, and the groups are arranged in groups, and so on, forming what is called the metric hierarchy. Beat grouping, sometimes referred to as *subjective rhythmization*[Fra83], is a psychological linking of these sequential event stimuli. It is a perceptual illusion which has no apparent practical purpose. For the illusion to occur, the beats can not be too far apart (no more than two seconds), nor too close together (no less than 120 milliseconds). Indeed, these are not new concepts; previous study can be found in [Mey56, LJ83, Fra83, Han89, Lee91] but this list is not at all comprehensive.

Metric structure can also be considered representationally. In this form, it goes by such names as Western musical scripture (Figure 2-1A) or horizontal bar notation (Figure 2-1B). The latter form is often used in computer music sequencers and electronic drum machines. The common characteristic here is time quantization; the duration between all note events is a multiple of some time quantum.

Metric structure alone is not an accurate way of representing musical rhythm; it provides a fundamental framework that is necessary, but restrictive. Music represented solely in this form tends to be bland and lifeless. In fact, very little expressive timing can be de-

[1] The stroke of the hand or baton in conducting, or often the most emphasized pulse.

B: A Time Deformation

Figure 2-2: Representations for Tempo Variation.

scribed using only metric structure. We therefore leave the symbolic world, and enter the continuous one.

Tempo variation is perhaps the most studied expressive timing model. It refers to the change in speed with which the metric grid passes by as the musical piece progresses. This is similar to a real performance in which the tempo will increase (accelerando) or decrease (ritardando). Models of tempo variation often take the form of functions that map score time to performance time [Jaf85, WBS87] (Figure 2-2A), or that map time durations to deformed time durations [AK91, AB91, AB92, AK89] (Figure 2-2B). There are other models that do not appear to be tempo variation[Cli77], but in fact are. Tempo variation functions have been discovered that correspond closely to real musical performances [FERes, Rep90]. In short, although all the above models are essentially the same thing (functions that provide a duration for each beat), tempo variation is required for any valid representation of expressive timing in music. No study of rhythmic timing should go without it.

Ametric models of rhythm represent those phrases that do not have an associated beat. A musician produces these phrases without using a constant beat as a reference for note placement. These rhythmic figures are akin to those of poetry or even plain speech, and might be called rhythmic prosody. There are two musical situations in which this occurs: 1) Some music contains no perceptible beat. Rhythmic phrases contained therein

completely disregard any isochronous pulse, so no beat-based representation is suitable. 2) When there is a clearly defined beat, sometimes a performer will land on that beat only once per phrase. For example, a performer might land on a beat every eight bars, but in the interim might completely ignore all such guidelines. In this case, a model must account for the connection with high levels in the metric hierarchy and for the interim sections. I am not aware of any attempts to model these types of musical phrases. And because almost all percussive music can be represented without ametric models, I also will not consider them further. Ultimately, however, they should be investigated.

Musical expression is not always best thought of in terms of inter-onset durations. That is, we do not always conceive of a piece of music as items separated by a stretchable or shrinkable duration, like boxes and glue in a TEX document. Tempo variation is inherently a controller of inter-onset durations in which space between note events is varied; the shorter the space, the faster the tempo. Often, however, music is *event centered*: we place events at appropriate times perhaps deviating from some grid markers. In this case, the inter-onset durations play a minor role. It is the onset times themselves that make the headlines.

For example, a common difference between Western classical music and ethnic or modern music (such as African, Latin, or jazz) is that expressive timing in the former can often be represented using only tempo variation. Ethnic and modern music is event centered, however, and tempo variation is not a suitable model. Therefore, this music requires a new timing model. Enter deviations. Deviations are, like tempo variation, functions of a metric time position. Unlike tempo variation, however, a deviation function provides a duration for each metric position used to time shift any musical event falling on that position (see Figure 2-3). Such a function can therefore be used to model African music and jazz (i.e., swing) in which performers deviate in time from a relatively uniform beat.

Jazz music almost always has a steady beat. But a jazz musician almost never plays on the beat. Where is the beat? In particular, where is it when everyone plays off the beat? People are said to play "behind the beat," "right on the beat," or "in front (or on top) of the beat," in which note events occur respectively later than, right on, or earlier than the time grid defined by the beat. When most of the performers play on the beat (hardly ever), the time grid is explicitly defined, but the performance sounds dead. When most of the performers play off the beat, how do we know where it lies? That is, if all performers in an ensemble play behind the beat, why do we not just perceive the time grid earlier and hear

Figure 2-3: A Deviation Curve.

the performers playing on the beat? Several reasons are possible: 1) Performers do not deviate by the same amount. In ensembles, certain instruments whose role is to define the back-beat typically play more on the beat than other instruments; for example, the bass in a jazz ensemble and a support drum in an African ensemble tend to play more on the beat. 2) The amount a performer plays behind or ahead of the beat varies. For example, during the beginning of a section, a performer might play right on the beat but deviate appreciably during the middle. Models of deviation can easily represent both of these situations.

There is no doubt that music devoid of both harmony and melody can still contain considerable expression. Percussive music is a case in point, as anyone who has truly enjoyed traditional music from Africa, India, or Central or South America knows. I claim that most percussive rhythmic figures can be represented solely using these models. Specifically, deviations are essential to any study of percussive musical phrases.

2.2 Tatums

The next concept will be used throughout this thesis. When we listen to or perform music, we often perceive a high frequency pulse, frequently a binary, trinary, or quaternary subdivision of the musical tactus. What does it mean to perceive this pulse, or as I will call it, *tatum*?[2]

The tatum is the high frequency pulse or clock that we keep in mind when perceiving or performing music. The tatum is the lowest level of the metric musical hierarchy. We

[2]When I asked Barry Vercoe if this concept had a term, he felicitously replied "Not until now. Call it *temporal atom*, or *tatom*." So, in honor of Art Tatum, whose tatum was faster than all others, I chose the word *tatum*.

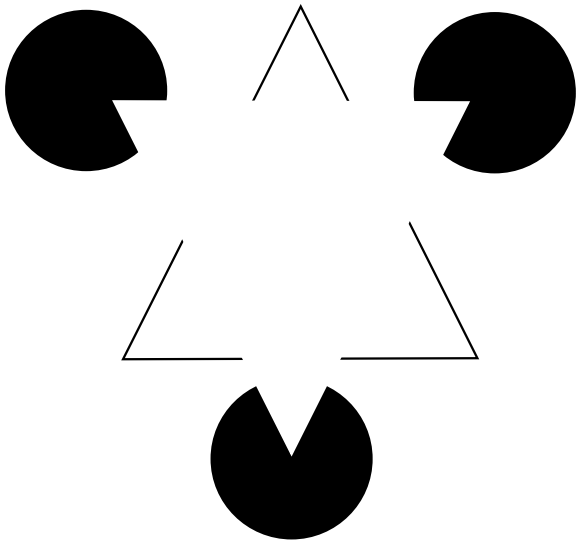


Figure 2-4: Illusory Contours.

use it to judge the placement of all musical events.

Perceiving the tatum does not necessarily imply a conscious ticking in the mind, like a clock. Often, it is an unconscious and intuitive pulse that can be brought into the foreground of one's thought when needed. Perceiving the tatum implies that the listener or performer is judging and anticipating musical events with respect to a high frequency pulse. Often, it is an illusory perception, quite similar to the illusory contours described in Marr's book[Mar82] and perceived in the upside-down triangle in Figure 2-4.

The tatum is not always explicitly stated in a piece of music. How, then, is it implied? Often, it is defined by the smallest time interval between successive notes in a rhythmic phrase. For example, two sixteenth notes followed by eighth notes would probably create a sixteenth note tatum. Other times, however, the tatum is not as apparent; then, it might best be described as that time division that most highly coincides with all note onsets.[3]

The tatum provides a useful means of defining tempo variation and deviations. Tatums pass by at a certain rate, and can be measured in tatums per minute. Therefore, tempo variation can be expressed as tatum duration (in seconds) as a function of tatum number. Similarly, deviations can be expressed as deviation (in seconds) as a function of tatum number. That is, a deviation function determines the amount of time that an event, metrically falling on a particular tatum, should be shifted when performed.

[3]Section 6.1.1 discusses situations in which the tatum rate may change in a piece.

Note that tempo variation is defined per ensemble, whereas deviations are defined per single voice of a performer. A piano, which has ten voices, would have a deviation function producing at most ten values per tatum. Therefore, deviations operating at the tatum level can encode the chord asynchronies of [Pal88] or constructs in the expressive timing calculus of [DH92]. A drum, which we study here, has one voice and would have a deviation function producing at most one value per tatum.

The tatum concept will be used in several ways throughout this thesis. Tatums are used to refer both to a perceptual concept and to a physical computer representation. Additionally, tatums are used in two ways when referring to a representation.[4] First, the n^{th} tatum in a piece of music is the n^{th} tatum that occurs in that piece. For example, if the piece is 200 measures long, and there are 16 tatums per measure, then there are 3200 tatums in the piece numbered from 1 to 3200. Second, the term *per-measure tatum* refers to a tatum in a certain position relative to every measure in a piece of music. So, the i^{th} per-measure tatum refers to the i^{th} tatum relative to the beginning of every measure.

2.3 Tempo Variation versus Deviations

One common question is if we assume tempo variation is also per person, why not just use tempo variation to represent deviations in a performance? That is, is there mathematically any difference between tempo variation and deviations? The answer is no, there is no *mathematical* difference. Either can represent performance timing. There is, however, a perceptual difference.

When listening to or playing in a drum or jazz ensemble, there are times when the tempo is considered constant, even though members are playing off the beat. Notice, the concept of being "off the beat" suggests that there is deviation from some tempo followed by the ensemble. There is no concept, however, of individual members of the ensemble slightly adjusting their own personalized tempo. Furthermore, the tempo change needed to represent deviations in a typical performance would be at an unnaturally high frequency and high amplitude. Imagine, at each successive tatum, varying the tempo between 2000 and 354 tatums per minute (Figure 2-5A). Perceptually, this seems quite impossible. However it seems quite reasonable to assume that a person could, at a constant tempo of 300 tatums

[4] Appendix E further defines representational subclasses of tatums.

Figure 2-5: Equivalent Tempo Variation and Deviation Representations.

per minute, play every other note 15 percent of a tatum early (Figure 2-5B). In other words, although they might be mathematically equivalent, tempo variation and deviations are different in more important ways – they are distinct both functionally and conceptually.

The previous paragraph suggests that there must be some (per person) upper limit on tempo oscillation frequency. That is, any timing variation in the performance not accounted for by tempo variation because of its high frequency must be caused by "deviations." This seemingly minor point, as we will see in Section 4.2, is quite important.

Chapter 3

Data Acquisition

Before we begin an analysis of timing in musical rhythm, it is necessary to obtain timing data, extracting both onset time and note category. This chapter describes a new process that I used both to extract onset times from a recorded performance of percussive music and to automatically classify musical events into different categories.

3.1 Attack Time Detection

What is an *attack time*? Most people would agree, an attack time is the point when a musical event becomes audible. But what does *audible* mean in this context? Does it refer to the time when physical energy in a signal increases infinitesimally, the time of peak firing rate of the cochlear nerve, the time when we first notice a *sound*, the time when we first perceive a *musical* event (*perceptual attack time*), or something else? Whatever it means, it is clear that perceptual attack time is not necessarily coincident with initial increase in physical energy. In his paper, [Gor84] discusses various definitions of attack time. Unfortunately he comes to no certain conclusion to the question. He notes, however, that it is easier to determine the attack time (whatever it may be) in percussive music because there is less room for error; relative to other musical instruments, the time between zero and maximum energy of a percussive musical event is almost instantaneous.

Because there is not wide agreement on the exact definition of attack time, I use a definition that, although not unreasonable, is biased towards percussive music: the attack time is a point at which the slope of the short-time energy of the high-pass filtered signal reaches a peak. The reasons for this are discussed in this chapter.

Some definitions of key terms used throughout this thesis follow. Each musical event in percussive music is called a *drum stroke* or just a *stroke*; it corresponds to a hand or stick hitting a drum, two sticks hitting together, two hands hitting together, a stick hitting a bell, etc. The attack time of a drum stroke is its onset time, therefore the terms *attack* and *onset* have synonymous meanings, and are used interchangeably. Finally, the term *drum stroke type* (or just *stroke type*) is functionally similar to pitch. It provides a means to classify the quality of percussive sounds. A drum is hit in different ways and those ways that produce sounds that are distinct from others constitute different stroke types.

3.1.1 The Recording, Environment, Instrumentation, and Personnel

They were possessed by the spirit of the drums.

Chinua Achebe
Things Fall Apart [1959]

It was an honor to have Los Muñequitos de Matanzas, the extraordinary dance and drum ensemble from Matanzas, Cuba, as participants in my collection of percussive timing data. Indeed, the members of Los Muñequitos de Matanzas are some of the best (if not the best) drummers in the world. They play with incredible feeling and technique and they could fairly be called the apotheosis of percussive musical expression. I could not have found a better musical ensemble for this study. The data obtained from this performance was used for all subsequent analysis in this thesis.

The recording took place on 6 November 1992 in the MIT Media Laboratory's Experimental Media Facility.[1] I am indebted to the people who, during the two preceding days, concentrated on the technical issues surrounding the recording.[2] They were thorough and ensured that the recording, once underway, would go smoothly and without malfunction.

Each performer was recorded onto a separate track of a multi-track tape player. My goal was to obtain a transcription of each individual drummer, i.e., a list of all pairs of attack times and stroke types for each performer. The drummers were each recorded on separate tracks to eliminate the necessity of performing source separation on the resulting signal, a procedure known to be extremely problematic [Ell92a, Mel91].

[1] i.e., the Cube.
[2] Especially Greg Tucker and Andy Hong.

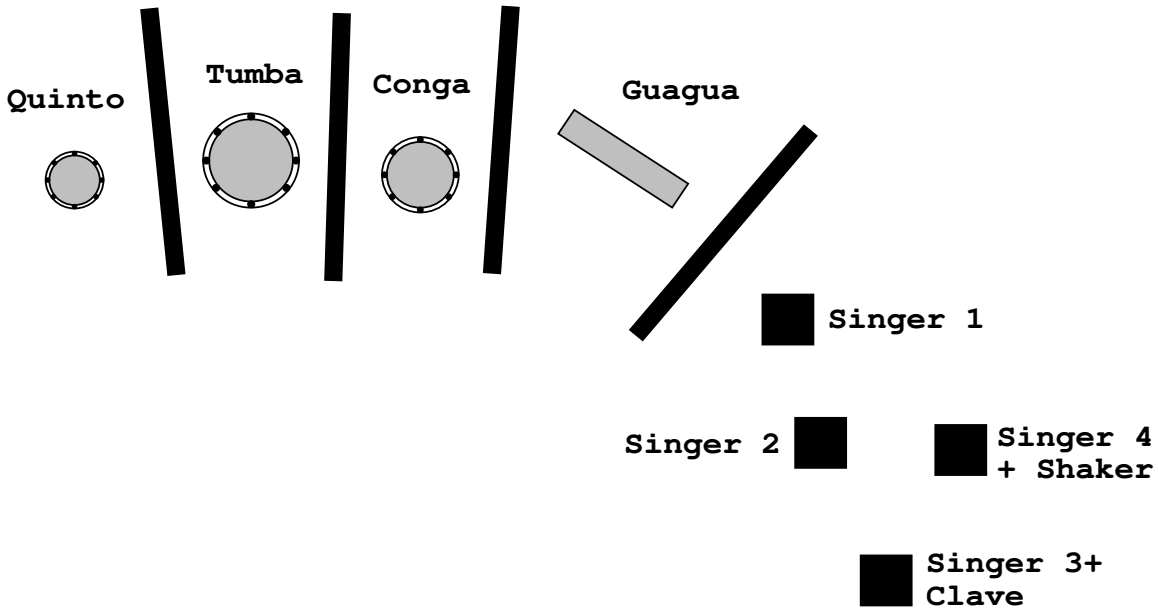


Figure 3-1: Layout of the Los Muñequitos Performance.

Eight members of Los Muñequitos de Matanzas were recorded. Three drummers each played one conga drum – a low pitched one (tumbao), a middle one (segundo) and a high one (quinto). The quinto player improvised whereas the tumbao and segundo players mainly played repetitive, predetermined parts. Another ensemble member played the guagua, a thick bamboo cylinder, about 4" diameter, hit with two sticks. In addition, there were four singers. One singer simultaneously played the clave, two cylindrical hardwood blocks that are hit together producing a high pitched sound. Another singer played the shaker, a rattle made from a dried gourd with beads webbed around the gourd's bulbous end.

The drummers were as arranged as in Figure 3-1. Sound barriers surrounded the guagua player and the three drummers (indicated as black bars in the figure), preventing as much inter-track bleed as possible (nevertheless, as we will see later, there was enough bleed to cause difficulties). The recording was made on an eight track tape machine. The recorded tracks, in order, consist of: 1) quinto, 2) tumbao, 3) segundo, 4) guagua, 5) the clave, the shaker, and voice, 6) unused, 7) voice, and 8) voice. Although the original plan assigned one performer to each recording track, technical difficulties required us to use only three tracks for the vocals, clave, and shaker. A total of six songs were recorded, five rumbas and one santeria. This amounts to a total recording time of approximately twenty three minutes.

3.1.2 Computation of Drum Attack Times

Attack time detection was not a facile task. Although inter-track bleed was minimized by the precautions taken during the recording process, some of the tracks, especially the tumbao, contained considerable bleed. The process described herein was developed specifically for this performance. Nevertheless, it would successfully produce attack times for any percussive recording without bleed.

Stroke types can generally be categorized in three ways (there are more than three stroke types, but for the purpose of this discussion, three general categories suffice):

1. High energy *open tones* with relatively long temporal extent (about 250ms). The drummer forcefully strikes the drum, immediately removing the hand.

2. High energy *slaps* with relatively small temporal extent (about 20ms). The drummer forcefully strikes the drum, without immediately removing the hand.

3. Low energy *mutes* with relatively short temporal extent. The drummer gently grounds the hand on the drum.

Unfortunately, the energy of mutes in one track was often less than the energy of bleed from the combined effects of the quinto slaps and the broad-band noise of the shaker. For example, in Figure 3-2, there are three tumbao mute attacks at 29.42, 29.6, and 29.8 seconds. The quinto attacks bleed over at 29.35 seconds and 29.525 seconds,[3] and there is broad-band, temporally-long "noise" from the shaker between 29.25 and 29.45 seconds. Clearly, this makes it difficult to extract the attack time of the tumbao mute at 29.42 seconds which is engulfed in a flood of neighboring track signal. This recurring situation caused considerable problems for the attack time detection algorithms. I tested several methods, all operating on data obtained from a 44.1kHz DAT recording of the individual tracks.

I initially implemented and tested the method described in [Sch85, Gor84]. First, the envelope of the signal is computed by sliding a min-max window over the data. The min-max window computes two values for the time point at the window's center; these values are simply the minimum and maximum valued samples in the window. The window *slides* over the signal finding minimum and maximum values for all points in the signal – except,

[3]Because the entire spectral region (from 0Hz to 20kHz) is shown, the low frequency bleed near 500Hz is at the very bottom of the spectrogram.

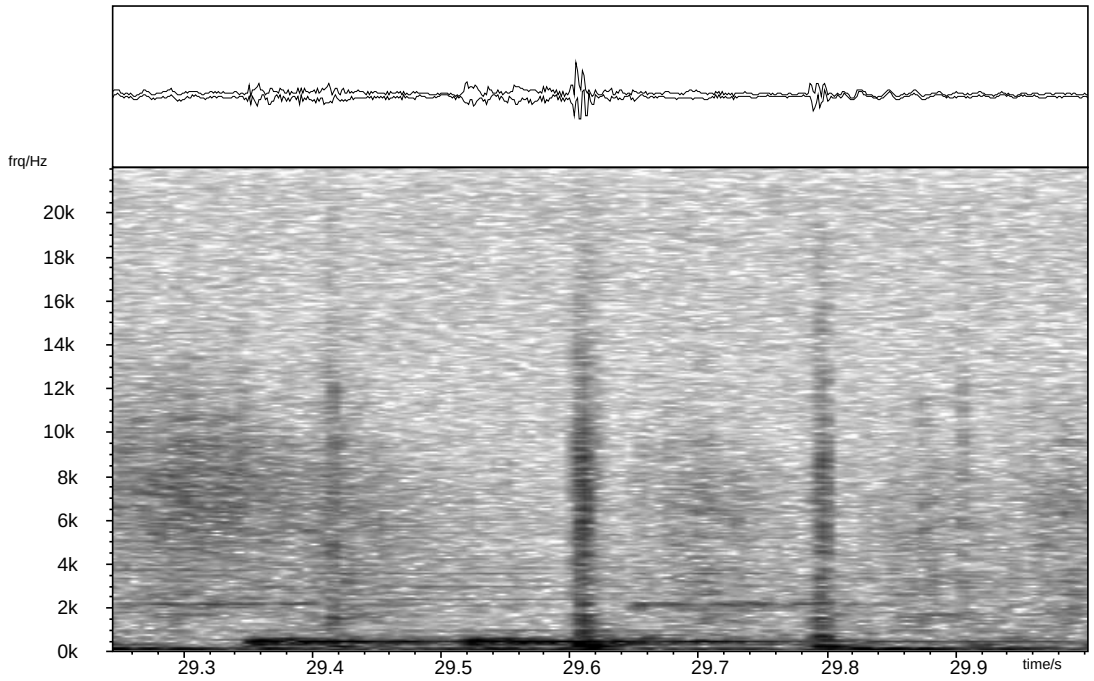


Figure 3-2: Bleed of Quinto and Shaker onto Tumbao Track.

of course, for the first and last $(\text{windowLength} - 1)/2$ points. These values for all the signal points constitute the envelope. A pseudo-derivative of the upper portion of the envelope is then computed: for each point, it and several surrounding points are used to find the slope parameter m using the method of linear least-squares fitting to a line. The slope signal is searched sequentially and, when the slope goes above some threshold, the time of the next zero-crossing (envelope maxima) is found and used as the attack time. That is, an attack time is defined as a zero-crossing in the slope (or envelope maxima) that directly follows a slope maxima above some threshold.

Because of the large amount of inter-track bleed in the Muñequitos recordings, however, this method either produced many spurious attacks, missed many of the low energy mutes, or both. Furthermore, because the attack time is based on the slope of the envelope, and the envelope is essentially a very low-pass filter and has a very narrow pass-band, this method has very poor time resolution. Drum hits are usually characterized by rapid transients that result in a broad-band noise burst. The most accurate attack-time measurement will be aligned to this burst and is most easily computed using the upper part of the signal spectrum. Therefore, we should use only high frequency energy as an attack time determi-

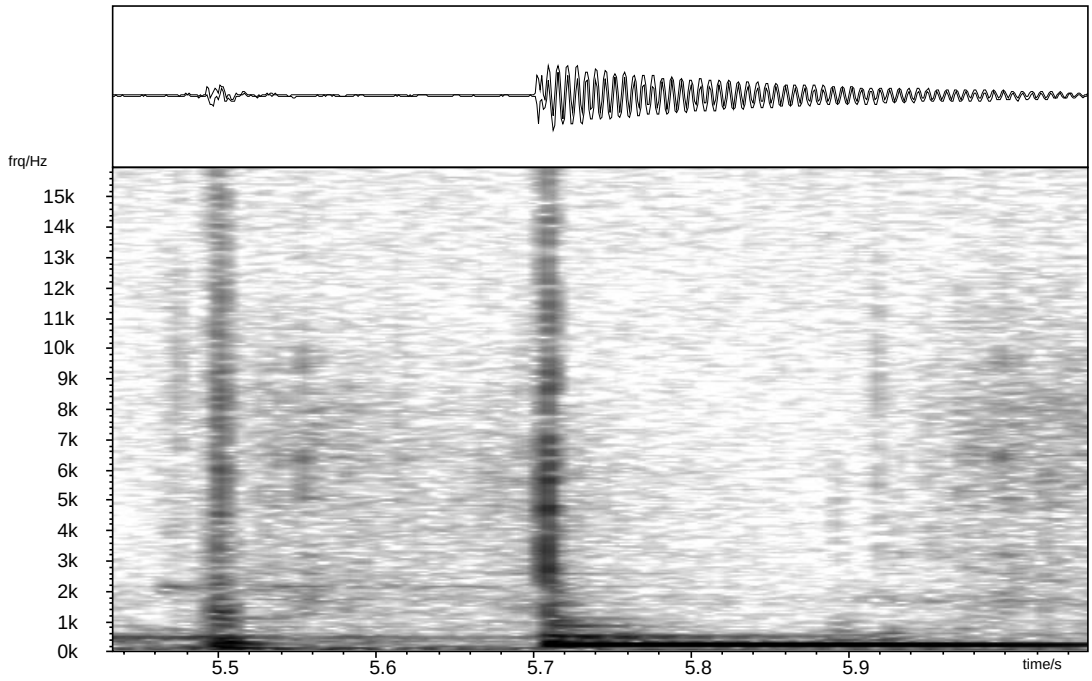


Figure 3-3: Spectrogram of a Tumbao Mute and Open Tone.

nant. With this in mind, I proceeded to develop a new attack detection method.

Observe the spectrogram of two typical attacks, the first a mute, the second an open tone (see Figure 3-3). The mute occurs at 5.5 seconds and the open tone begins at 5.7 seconds. There is essentially a broad-band short burst of energy for both attacks, but the open tone has significant temporal extension in the low frequencies (below about 500Hz). Therefore, we can filter out the low frequency energy and use the only high frequency energy as an attack time determinant. Because the high frequency energy burst is not temporally distributed and is essentially vertical and linear in the spectrogram, it makes sense to use a linear phase FIR high-pass filter. The resulting filtered signal will therefore be wide in frequency and shifted in time by the constant group-delay of the filter [OW89]. Furthermore, because the filter has a very wide pass-band (from the pass-band cutoff frequency up to half the sampling rate) we should obtain good time resolution. This suggests the following strategy which is diagrammed in Figure 3-4.

Attack Detection Strategy #1

Input: A digital signal containing drum strokes.

Figure 3-4: Attack Detection Strategy #1.

Output: The attack times for each drum stroke.

Step 1: HPF. Filter the attack signal with a linear phase FIR high-pass filter.

Step 2: ENG. Compute the short-time energy of the filtered signal using a sliding window energy process. A window slides across the signal finding the energy for the sample point at the window center. That is, we compute

$$e[n] = \sum_{i=-M_{ws}}^{i=+M_{ws}} (x[n+i])^2,$$

where $x[n]$ is the filtered signal, $2M_{ws}+1$ is the window size in samples, and $e[n]$ is the energy for position n of the filtered signal.

Step 3: SLP. Compute the slopes of the energy using a sliding window slope process. This is not just a first order difference, but is the computation of the slope m using the linear least squares method.

Step 4: THRS. Search through the slope signal. Once the slope goes above some threshold (*slopeThres*), search over a pre-defined region (*msSearchWin*) for the point at which the slope is maximum (**MAX**). Take that point as the attack time.

Step 5: Skip a bit (*skipRegion*) and go to step 4 if any signal remains to be processed.

Step 4 looks for the maximum of the slope of the short-time energy of the high-pass filtered signal – we are using, as our attack time, the point of fastest *increase* in high frequency energy. The algorithm does not look for the maximum energy of the filtered sig-

nal as in [Sch85]; it looks for a rapid increase of filtered signal energy at a time resolution dependent on the size of the slope window, the size of the energy window, and the pass-band of the high-pass filter. There are several reasons for taking the maximum of the slope rather than the maximum of the energy as the attack time:

- If we look only for absolute energy values, we might skip an attack. Consider, for example, a possible situation in which the energy increases above some threshold because of an attack, and then before decreasing, increases further because of a new higher energy attack. If we look only at absolute energy, we miss the second attack (because there is no zero crossing of the slope of the energy of the first attack, it never reaches a maximum). By looking at the maximum of the slope, however, we will detect both attacks. In each attack, the computed attack times will be the points at which the energy increases the fastest.

- Some might argue that it is important to compute the perceptual attack time (PAT), i.e., the point in time at which humans perceive the attack. However, for analysis and synthesis (which we do have planned), it might be erroneous to compute the PAT because once analyzed, synthesized, and re-listened, PAT adjustments would be counted twice. In other words, any analysis on the timing properties of percussive sounds that aims to discover what constitutes expressive timing should analyze what *will be* perceived, not actually what *is* perceived. Therefore, we are ideally looking for the intended attack time (IAT), the time the performer meant the sound to begin, not the perceptual attack time. That way, a synthesis will produce something meant to be perceived. At worst, if the PAT is identical for different percussive sounds (or at least within 1ms) and if the computed attack time is consistently a fixed distance away from the PAT, then there will be no appreciable error during synthesis.

  Furthermore, as discussed in section 3.1, we currently know little about the processing in, and the feedback between, components of the mammalian hearing mechanism, surely not enough to make definitive assertions about exactly when we perceive an attack. The cochlea, the cochlear nucleus, the medial geniculate nucleus, and the auditory cortex might all influence each other in ways still unknown to even the best researchers. And, although we have quantitative knowledge of auditory-nerve fiber responses in mammals, and are aware that such a nerve fiber follows a tuning curve

[KP88] (the threshold of responses to tones plotted as a function of frequency), we can not assume necessarily that the resulting post-threshold neural firing rate corresponds with PAT. PAT remains elusive.

It therefore may be concluded that we should choose the measure that enables us to produce the most consistent attack times without being unduly concerned with PAT.

I implemented and tested this strategy. The high-pass filter was designed using the Kaiser window method [OW89]. A full listing of all the parameters and their descriptions follow:

1. High-pass filter parameters: The stop-band cutoff frequency ω_s, pass-band cutoff frequency ω_p, and tolerance δ. The transition region ($\Delta\omega = \omega_s - \omega_p$) is defined implicitly.
2. *slopeThres*: When the slope goes above this value, we start searching for an attack.
3. *msSearchWin* (Maximum slope search window): once the slope is above *slopeThres*, this is the maximum amount of time to search for a slope peak. This reduces unnecessary searching past the real slope peak.
4. *skipRegion*: Once a slope peak is found, this is the time to skip before searching again. This is usually about 60% of the tatum rate (and is therefore tempo dependent).
5. *ews*: Energy window size: the window size for the sliding window energy process.
6. *sws*: Slope window size: the window size for the sliding window slope process.

The actual parameters used are listed in Table 3.1. I obtained many of them empirically by repeatedly running the program. Note that *slopeThres* is dependent on the recording level of the performance.

An average human can detect time deviations no smaller than about 5ms [Sch85][pg22]. It seems reasonable to assume that professional percussionists, such as Los Muñequitos de Matanzas, have more finely tuned sensitivity to time deviations. It is therefore necessary to maintain very fine-grained time resolution in our attack detection strategy. Values controlling time resolution, the filter pass-band, *ews*, and *sws*, needed to be carefully chosen.

Unfortunately, because the attacks differ significantly in overall broad-band energy, no working value for the slope threshold could be found, even after spending considerable

	Segundo	Tumbao	Quinto
ω_s	1000Hz	1000Hz	1000Hz
$\Delta\omega$	50Hz	50Hz	50Hz
δ	0.001	0.001	0.001
$slopeThres$	10^8	10^8	1.3×10^8
$msSearchWin$	2.5ms	10ms	10ms
$skipRegion$	80ms	90ms	65ms
ews	1ms	1ms	1ms
sws	0.5ms	0.5ms	0.5ms

Table 3.1: Parameters for Strategy #1.

time adjusting all the parameters. One of two things would always happen. With a low threshold, the algorithm would detect all mutes, open tones, and slaps, but would also detect many spurious attacks (i.e., bleed). With a high threshold, the algorithm would not produce any spurious attacks, but would also miss many of the mutes.

Attack Detection Strategy #2

In place of the high-pass filter, I tried many band-pass filters. I determined the pass-bands by visually inspecting the spectrogram of the attacks, finding frequency bands at which all attacks seemed to all have common energy – but to no avail. The same problems occurred.

Attack Detection Strategy #3

Next, I located frequency bands in which different attacks had comparable energy. These regions might indicate reliable characteristics in the attacks' spectra that could be used as a criterion for the indication of an genuine attack. I manually selected a set of 13 different sample attacks that contained no bleed. For each attack i, its 16k point DFT $X_i[k]$ was computed. Then, the minimum $X_{min}[k]$ and maximum $X_{max}[k]$ of the magnitude spectrum was determined:

$$X_{min}[k] = \min\left(|X_1[k]|, |X_2[k]|, \ldots, |X_n[k]|\right),$$

and

$$X_{max}[k] = \max\left(|X_1[k]|, |X_2[k]|, \ldots, |X_n[k]|\right).$$

I then found the frequency bands where the difference,

$$X_{res}[k] = X_{min}[k] - X_{max}[k],$$

was small. Those frequency bands reveal the spectral locations at which the difference between the maximum energy and minimum energy of all attacks is small. There are two useful cases when a frequency band in $X_{res}[k]$ is small:

1. $X_{max}[k]$ is large and so is $X_{min}[k]$. So, all attacks have high energy in this frequency band.

2. $X_{max}[k]$ is small and so is $X_{min}[k]$. So, all attacks have low energy in this frequency band.

Case 1 could determine pass-bands and case 2 stop-bands in a multiple band-pass filter. Therefore, the filtered signal would theoretically contain information pertinent only to the attacks. And because the pass-bands are regions at which there is little energy difference, the filtered attacks should have comparable energy. I tested this strategy (using the method of Strategy #1) with various such multiple band-pass filters, but unfortunately there were three problems which made it unusable.

First, contiguous frequency bands where $X_{res}[k]$ was very small (less than or equal to about 1% of the average value)

$$X_{res}[k]/\text{average}(X_{res}[k]) \leq 0.01$$

were extremely narrow. In fact the largest such region, centered just above 10kHz, had only a 5Hz band-width. At 10kHz, at which the energy from all drum strokes is very low, any attack detection strategy would fall prey to spurious attacks caused by the broad-band and relatively high energy shaker.

Second, a linear phase FIR filter with such a small pass-band (which implies, of course, an even smaller transition region) would be several seconds in length[OW89]. A filter of this length would be computationally unrealistic with a sampling frequency of 44.1kHz. Even worse, it would have terrible time resolution. Digital IIR filters can be designed to implement these very sharp spikes in magnitude frequency response. However, they would likewise have terrible time resolution and would also produce nonlinear phase distortions.

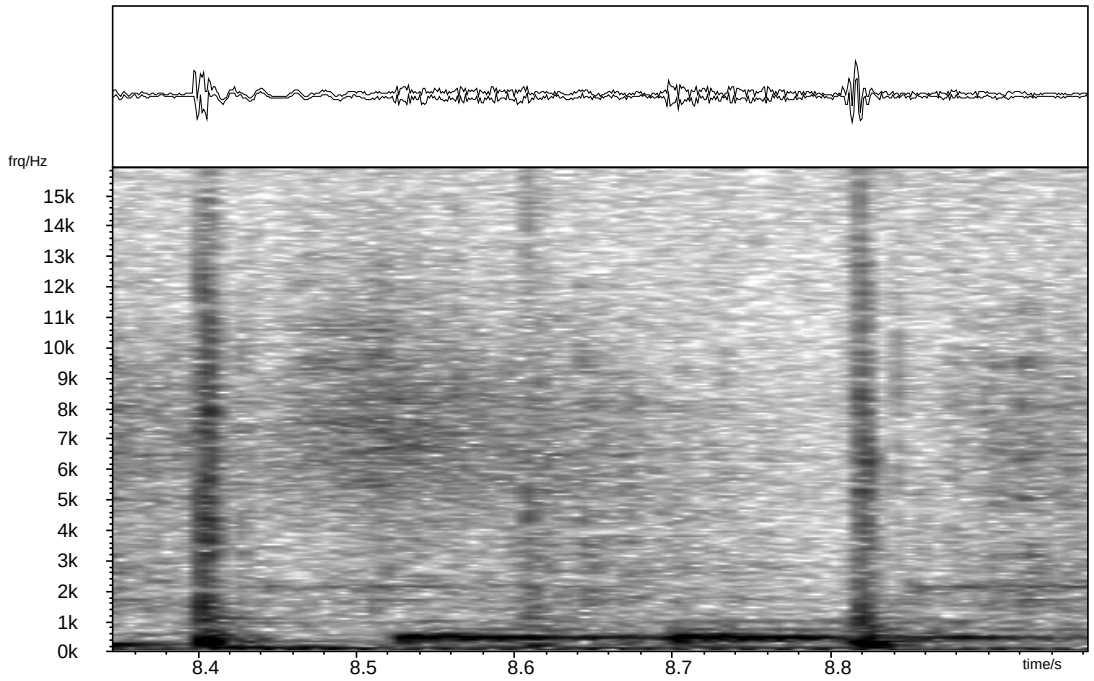


Figure 3-5: Another Example of Quinto Bleed onto Tumbao Track.

Therefore, such filters were not considered further.

Third, when looking at frequency bands wider than 5Hz and centered on frequency regions containing most of the drum energy (between 1k and 10kHz), $X_{res}[k]$ would quickly (within about 10Hz) jump to more than 50% of the average value and stay there. Therefore, using a shorter DFT with coarser frequency resolution in the computation of $X_{res}[k]$ would probably produce no frequency bands in which $X_{res}[k]$ was small. Disregarding these facts, I tried the method with a linear phase FIR filter of reasonable length. The pass-bands with band-widths ranging from 50 to 300 Hz were situated over regions at which $X_{res}[k]$ was minimum. Unfortunately, even with these narrow pass-bands, the variation in $X_{res}[k]$ was significant and the results were identical to those previous: either spurious attacks or missed mutes. Reluctantly, this strategy was abandoned.

Two main problems caused the previous three approaches to fail. The first can be seen in Figure 3-5. The genuine tumbao attacks are at 8.4, 8.61, and 8.82 seconds. At 8.52 seconds, there is moderate high-frequency and significant low-frequency bleed from the quinto open tone onto the tumbao track. There is also significant high-frequency bleed from the shaker starting near 8.46 seconds. There is a high slope peak near 8.5 seconds because of

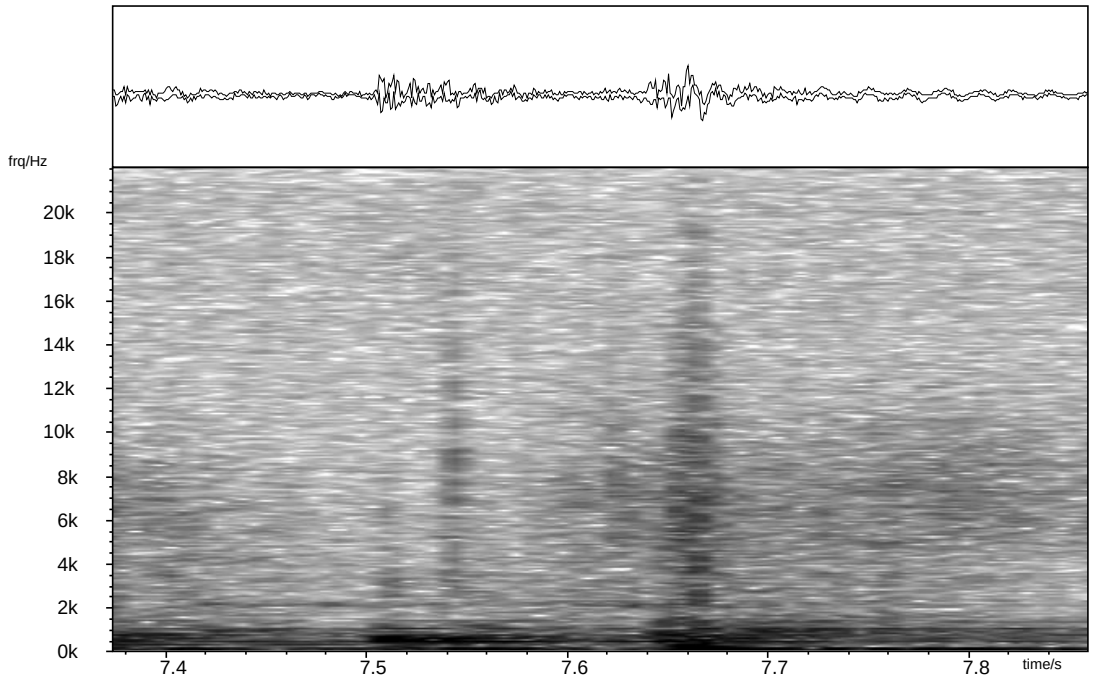


Figure 3-6: High Frequency Quinto Bleed onto Tumbao Track.

the high-frequency bleed. This is higher in fact than the slope peak near the real tumbao attack at 8.61 seconds. The above strategies produced either a spurious attack near 8.5 seconds (with a high *slopeThres*) or a missed attack at 8.61 seconds (with a low *slopeThres*).

The second problem occurred when high frequencies bled from another drum track (Figure 3-6). This example shows the quinto at 7.5 seconds bleeding over, not only in the low frequencies but also in the high frequencies above 1kHz. The tumbao mute at 7.54 is almost smothered. Furthermore, the shaker bleed once again is significant in high frequency between 7.61 seconds therefore drowning the tumbao at 7.66 seconds.

From these two situations, we may deduce the following criteria about a valid current-track attack: it contains a coincident increase in high and low frequency energy; the low frequency energy is much larger than the high frequency energy; and the high frequency energy is a broad-band temporally-narrow vertical burst. Therefore, it seems logical to retain Strategy #1's attack criteria on the high frequency energy, but accept the attack only if it coincides with a sharp increase in low frequency energy. This leads to the following algorithm also depicted in Figure 3-7:

Filter Process A

Figure 3-7: Attack Detection Strategy #4.

Input: A digital signal containing drum strokes.

Output: *slopesA*, a signal containing the slope of the energy of the high-pass filtered input.

A1: HPF. Filter the attack signal with a linear phase FIR high-pass filter.

A2: ENG. Compute the short-time energy of the signal using a sliding window energy process.

A3: SLP. Compute the slopes of the energy using a sliding window slope process. Call this *slopesA*.

Filter Process B

Input: A digital signal containing drum strokes.

Output: *slopesB*, a signal containing the slope of the energy of the low-pass filtered input.

B1: LPF. Filter the attack signal with a linear phase FIR low-pass filter. designed with a pass-band cutoff frequency equal to the frequency below which the majority of an average attack's energy is found.

B2: ENG. Compute the short-time energy of the signal using a sliding window energy process.

B3: SLP. Compute the slopes of the energy using a sliding window slope process. Call this *slopesB*.

Attack Detection Strategy #4

Input: A digital signal containing drum strokes.

Output: The attack times for each drum stroke.

step 1: Run Filter processes A and B.

step 2: Once *slopesA* goes above *slopeAthreshold*, if *slopesB* is also above *slopeBthreshold1* (**THRS**), search (**MAX**) over some pre-defined region (*msSearchWin*) for the point at which *slopesA* is maximum. Accept the attack time only if *slopesB* is above *slopeBthreshold2*.

step 3: Skip a bit more. Go to step 2 if any signal remains to be processed.

The attack time is therefore the point at which the high frequency energy is increasing the fastest, contingent on the low frequency energy increasing at a certain rate. Of course, this process adds considerably to the number of parameters. In fact, the parameters now consist of:

1. High-pass filter parameters: The stop-band cutoff frequency ω_{A_s}, pass-band cutoff frequency ω_{A_p}, and tolerance δ_A. Again, the transition region ($\Delta\omega_A = \omega_{A_s} - \omega_{A_p}$) is implicitly defined.

2. *slopeAThres*: When the slope goes above this value, we begin searching for an attack.

3. *ewsA*: Energy window size, the window size for the sliding window energy process.

4. *swsA*: Slope window size, the window size for the sliding window slope process.

5. Low-pass filter parameters: The pass-band cutoff frequency ω_{B_p}, stop-band cutoff frequency ω_{B_s}, and tolerance δ_B.

6. *slopeBThres*: Same, but for process B.

7. *ewsB*: Same, but for process B.

8. *swsB*: Same, but for process B.

9. *slopeBThres2*: The value that *slopesB* must be above for a candidate attack to be accepted. This reduces the likelihood that a small bump in low frequency would cause the process to accept a spurious attack.

	Segundo	Tumbao	Quinto
ω_{B_s}	1000Hz	1000Hz	1000Hz
$\Delta\omega_B$	50Hz	50Hz	50Hz
δ_B	0.001	0.001	0.001
slopeBThres	10^7	0.2×10^7	10^6
slopeBThres2	0	0	0
ewsB	1ms	1ms	1ms
swsB	15ms	20ms	15ms

Table 3.2: Strategy #2 Parameters.

10. *skipRegion*: Same as in Strategy #1.

11. *msSearchWin*: Same as in Strategy #1.

I implemented and tested this strategy. The FIR filters were again designed using the Kaiser window method. Many parameters here are identical to Strategy #1 and have the same final values, so Table 3.2 contains only the additional ones. Notice that the low-pass filters pass only the frequencies that the high-pass filter removes.

The parameter *slopeBThres2*, always set to zero, ensured that *slopesB* was not negative at the attack time. Therefore, we never accepted an attack time when the low frequency energy was decreasing. This reduced the chance of spurious attacks. In addition, this parameter, along with *msSearchWin*, limited the search extent for high frequency slope peaks.

This algorithm worked well. Figure 3-8 gives an example in which dotted lines indicate computed attacks. Nevertheless, I spent considerable time adjusting parameters. There was method to this madness however; deactivating the attack criteria on *slopesB*, I adjusted the attack criteria on *slopesA* until all real attacks and some spurious attacks were detected. Then, looking at the values of *slopesB* at the spurious attack times, I set the *slopesB* attack criteria parameters to reject only the spurious attacks. Then, reactivating the *slopesB* attack criteria, I ran the program a final time.

This algorithm, along with its preventatively set parameters, eliminated all of the spurious attacks while allowing the genuine ones through. Still, there were some problems in the adjustment of *swsB*. If it was too small, a spurious attack occurring just before a genuine attack might cause a large *slopesB* value and would be accepted. If *swsB* was too large, mutes were rejected because values of *slopesB* were essentially zero at mute attacks even though values of *slopesA* were above threshold.

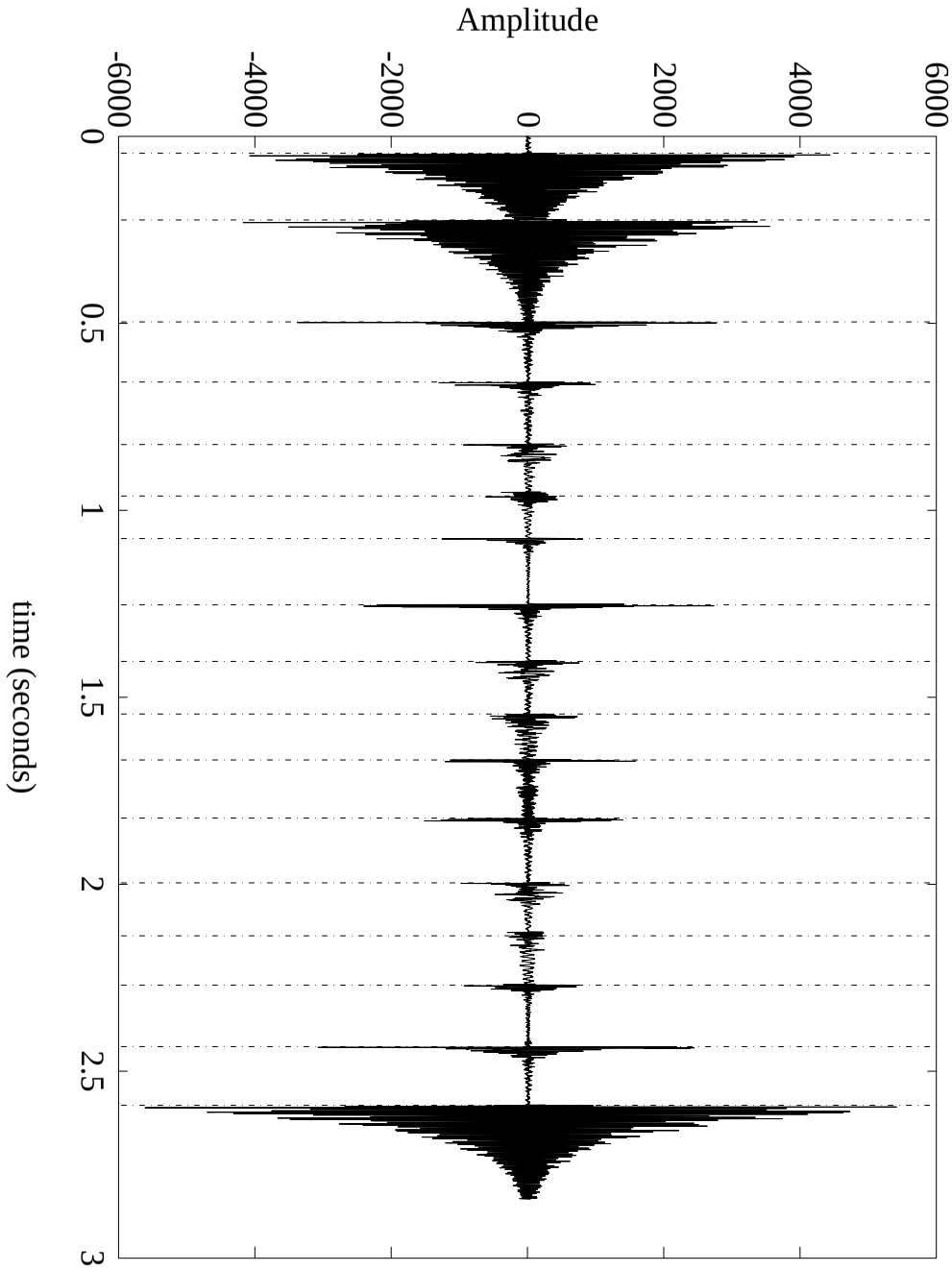


Figure 3-8: Segundo Attack Detection Example.

The tumbao track was most intransigent. This drum was placed next to the quinto (the loudest drum) during the performance, so there was quite a bit of bleed. Moreover, tumbao open tones were occasionally followed immediately by a soft mute, and *slopesB* was negative even though *slopesA* was large and positive. In addition, fast runs of open quinto tones caused problems detecting quinto attacks; the previous tone had not completely decayed when the next tone started, so *slopesB* was not very large.

The parameters listed in Table 3.2 were tuned to avoid spurious attacks and miss as few mutes as possible. Presumably, it is better to get valid data and less than 100% of what exists (missed attacks), than invalid data and more than 100% of what exists (spurious attacks).

After all attacks were detected, I assiduously examined and verified each one, correcting any mistakes. This step was necessary because subsequent portions of this project rely on perfectly accurate data. I also used the corrected attack time lists to judge the performance of the attack detector. On all tracks, the attack detector found 100% of the open tones and slaps, about 95% of the mutes, and produced less than 3% spurious attacks.

Bleed caused most of the problems. If there was no inter-track bleed, I believe this approach would be 100% successful. In an ideal recording session with drummers playing in isolation booths, this attack detection strategy should work perfectly.

3.1.3 Computation of the Guagua Attack Times

Computation of the guagua attacks *was* a facile task, indeed. Although there was track bleed, the guagua attacks were significantly higher in energy. More importantly, all the guagua attacks had about equal energy and the impulse responses of the guagua attacks were all almost identical. Therefore, I used Attack Detection Strategy #1, find the fastest increase in energy of the filtered signal. There was one major difference, however. Rather than using a linear phase filter, I used a matched filter. The matched filter's impulse response was obtained by simply taking a time reversed sample of a clean (no bleed) guagua stroke. Convolving the guagua track with this filter was therefore equivalent to cross-correlating it with the clean guagua impulse response. That is, normal discrete time con-

volution of a signal and a filter is defined as

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k],$$

where $x[k]$ is the signal and $h[n]$ is the filter impulse response. If $h[n]$ is the impulse response of the guagua, we compute

$$y[n] = x[n] * h[-n] = \sum_{k=-\infty}^{\infty} x[k]h[k-n],$$

which is the definition of cross-correlation between $x[n]$ and $h[n]$. The peaks of the slope of the energy were found just as in Strategy #1. This worked flawlessly.

3.1.4 Implementation Note

I implemented this whole process in C++. Entire signals, 5 minutes of 44.1kHz stereo 16bit sampled sounds,[4] were much too large to process whole. Therefore, processing was done in segments. The filtering method consisted of linear convolution using FFTs in which adjacent segments were combined using the overlap add method[OW89]. For the same reason, the program computed alternating segments of *slopesA* and *slopesB* rather than one at a time. Because ewsA $\neq$ ewsB and swsA $\neq$ swsB, it was possible to encounter a value in *slobesB* whose corresponding *slopesA* value was not in the current segment (and vice versa). Therefore, the program contained some interesting bookkeeping.

I implemented the sliding windows attempting to take advantage of previously computed information whenever possible. The sliding energy-window was implemented as a C++ class. Through obvious means, each single sample slide-right required two floating-point multiplies, two floating-point adds, and one pointer increment. The sliding slope-window was also implemented as a C++ class. Each single sample slide-right required four floating point adds, and five floating point multiplies, regardless of the window size. Assuming the definitions of S_x, S_y, S_{xx}, and S_{xy} given in [PTVF92], let $S_x[i]$, $S_y[i]$, $S_{xx}[i]$, and $S_{xy}[i]$ be the corresponding values for position $i + (M-1)/2$ in the sequence $Y[i]$. Because

$$S_{xy}[i+1] - S_{xy}[i] = -T \times (Y_{i+1} + Y_{i+2} + \ldots + Y_{i+M-1} - (M-1)Y_{i+M}),$$

[4] I look forward to the day when I can look back and smile at this paltry value.

where T is the sampling period, each slide right may be computed as

```
Sy -= (double)Y[0];
// Y is actually a float* pointing to the first data location.
Y++;
Sxy -= period*(Sy - (double)(windowLength-1)*
                                    (double)Y[windowLength-1]);
Sy += (double)Y[windowLength-1];
```

Roundoff could be a problem after many slides. Therefore, computation is done in double precision and, for each new segment, S_y and S_{xy} are calculated anew.

With these optimizations, the FFT code consumed 60% of the user time in each run. Additional optimizations were not possible.

3.1.5 Conclusion

The wasting of time is an abomination to the spirit.[5] Yet many hours of time spent massaging parameters for a given test case motivated one major realization: This is not the way humans detect attack times in percussive music. When I, a human, listen to the performance, I have no trouble attending to only the current drum, ignoring the bleed (similar to the well-known cocktail party effect [Bre90]). In fact, I have no trouble attending to an instrument only via its bleed (I can easily listen to the quinto on the tumbao track, for example). The energy process approach would certainly kick up its heels with that one.

In effect, humans have the capability of listening to only what they wish. We can isolate events from an amalgamation of sounds; we can zoom in on what interests us. All of these things, I believe, require a high degree of knowledge about the sound source. Humans must have a representation of what they are attending to (e.g., a particular percussive instrument), they must have unconscious knowledge of what to listen for, and they must have expectations about when to listen. This representation perhaps significantly influences low-level auditory processing, something [KP88] affirms. Indeed, I expect any low level processing like the method I describe above to be completely fruitless in an endeavor to obtain any perceptually genuine measure of auditory event onset times. Nevertheless, as stated above, for our purposes we require not the perceptual attack time, but rather an attack time mea-

[5] The Maxims of Ptahhotpe [c. 2350 B.C.]

Figure 3-9: Automatic Stroke Classification Strategy.

sure based on the physical energy in the data signal itself and that can be used for analysis and synthesis. So, although it is a fascinating problem to determine how humans perceive percussive attacks (or for that matter, any sound at all), we have restricted ourselves here to the task at hand, studying what humans produce.

3.2 Automatic Stroke Classification

I initially classified drum strokes by hand – another extremely tedious process. While sufficient for obtaining stroke types, this manual process became quite time consuming. I therefore decided to build an automatic stroke classification system.

The general approach, depicted Figure 3-9, is described as follows. For each drum stroke, extract numerical features from the input signal. Use these features in an unsupervised learning strategy that discovers *natural* clusters in the feature space. Finally, assume each cluster corresponds to a separate stroke type. Three processes constitute the system: 1) the segmentation process, 2) the feature extraction process, and 3) the clustering algorithm.

3.2.1 Drum Stroke Segmentation

When extracting features, it is necessary to know the portion of the signal constituting each drum stroke, i.e., the delimiting time points. This is done by finding the point after each attack time at which the energy has significantly decayed.

First, compute the short-time energy signal using the sliding window energy process.

Use a relatively large window size (*sEws*) that guarantees high-frequency (i.e., short time) and large-magnitude energy variation does not mistakenly signal the end of the stroke.

Second, find the point of maximum energy between the attack time and *breakOff* seconds before the next stroke. Therefore, both *breakOff* and the next attack time determine the maximum distance to search.

Finally, starting at the attack time, search for the point when the energy signal becomes smaller than a fraction, *energyFraction*, of the previously found maximum energy. The end of the stroke is determined by the point of submergence. If the energy signal never gets there, however, the stroke end is *breakOff* seconds time before the next attack time.

When I tested this algorithm, I adjusted the parameters to provide the best segmentation. I judged this by visually inspecting the stroke delimiters. The results were then given to the next stage of processing.

3.2.2 Feature Extraction

Obtaining good features for each drum stroke was quite straightforward: run each stroke segment through a four-stage processing step consisting of: 1) segmentation, 2) feature extraction, 3) normalization, and 4) feature dimensionality reduction.

Feature Definition

The segmented signal is subjected to spectral analysis resembling the transfer function between auditory sound events and action potentials of cochlear nerve fibers in the mammalian ear. The spectral components provide features. Similar to the low-level auditory processing of humans who do so magnificently in this task, such biologically plausible features should make it easier for a pattern classifier to discriminate between different strokes. The analysis technique I used is an approximation to the constant Q transform[Bro91, Ell92b].

A constant Q transform has components that have constant Q, or constant center frequency ω to band-width $\Delta\omega$ ratio (i.e., $\omega/\Delta\omega = Q$). Such a transform may be calculated by evaluating[Bro91]

$$X^{cq}[k_{cq}] = \sum_{n=1}^{N[k_{cq}]-1} \omega[n, k_{cq}]x[n]e^{-jn\omega_{k_{cq}}},$$

where $X^{cq}[k_{cq}]$ is the k_{cq}'th component of the transform, $x[n]$ is the sampled time function, and $\omega[n, k_{cq}]$ are window functions (kernels) of length $N[k_{cq}]$ that determine the frequency

characteristics (center frequencies and band-widths) of each transform component.

Resembling natural cochlear filtering, the center frequencies are geometrically spaced and the band-widths of the individual elements increase with frequency. The center frequencies are often based on a musical scale and may be calculated using

$$\omega_{k_{cq}} = (2^{c/12})^{k_{cq}} \omega_{min},$$

where ω_{min} is the lowest center frequency, and c is the number of semitones between successive center frequencies. This leads to a Q of

$$Q = \frac{\omega_{k_{cq}}}{\omega_{k_{cq}+1} - \omega_{k_{cq}}} = \frac{1}{2^{c/12} - 1}.$$

In [BP91], a constant Q transform is calculated efficiently using the output of a discrete Fourier transform (DFT). However, I chose to approximate the constant Q transform using a DFT by simply averaging the spectral values at different band-widths for different center frequencies. The implementation is quite easy and, as we will see, seems to produce distinguishable features. Specifically, I define the approximated constant Q transform as

$$X^{acq}[k_{cq}] = \frac{1}{M_{k_{cq}} - N_{k_{cq}} + 1} \sum_{k=N_{k_{cq}}}^{M_{k_{cq}}} |X[k]|,$$

where

$$N_{k_{cq}} = (\omega_{k_{cq}} - \frac{\omega_{k_{cq}}}{2Q})\frac{N}{2\pi},$$

$$M_{k_{cq}} = (\omega_{k_{cq}} + \frac{\omega_{k_{cq}}}{2Q})\frac{N}{2\pi},$$

and where $X^{acq}[k_{cq}]$ is the k_{cq}'th component of the transform, $\omega_{k_{cq}}$ are the desired center frequencies, $X[k]$ is the DFT of drum stroke time signal $x[n]$, and N is the DFT length. I implemented and tested this strategy with various values for c and ω_{min}.

Two additional features were computed for each stroke. The first one is the length-normalized energy of the drum stroke sample; that is,

$$x_{avg} = \frac{1}{L+1} \sum_{n=0}^{L} x_e[n]^2,$$

where L is the length of the stroke segment.

The second one, used also in [Sch85], encodes temporal extent by exploiting a drum stroke's near exponential energy decay. The goal is to fit an exponential decay function to the energy of the drum stroke sample. The energy is once again computed using the sliding window energy process, with window size *fEws*. Specifically, we find τ, the decay constant for which the energy of $x[n]$ most closely matches the function $y[n] = e^{-nT/\tau}$, where T is the sampling period. I first considered using nonlinear least squares methods [PTVF92] to fit the decaying exponential to the energy of the signal, but decided against it because the following approach was much simpler and probably just as effective.

I use autoregressive (AR) or all-pole signal modeling[LO88]. Specifically, I use the autocorrelation method of linear prediction and obtain a first-order model of the drum stroke energy signal $x_e[n]$; the parameters of the first-order model determine the time constant τ. The derivation of τ which follows is quite different than that found in [Sch85].[6]

We wish to model the signal $x_e[n]$, the drum stroke energy sample of length $L+1$ and with z-transform $X_e(z)$. By finding a signal $a[n] = a_0 - a_1\delta[n-1]$ with z-transform $A(z) = a_0 - a_1 z^{-1}$ such that

$$X_e(z)A(z) = \mathbf{1} + E(z),$$

or

$$x_e(n) * a(n) = \delta(n) + e(n), \tag{3.1}$$

where the error $e[n]$ is minimized, $A(z)$ becomes an approximation to the inverse of $X_e(z)$. We may therefore approximate $X_e(z)$ using the inverse of $A(z)$; that is,

$$\hat{X_e}(z) = \frac{1}{A(z)} = \frac{1}{a_0 - a_1 z^{-1}} \approx X_e(z).$$

Because we are interested only in the time decay and not the amplitude of the approximation, we may multiply by a_0 and obtain

$$\hat{X_e}'(z) = a_0 \frac{1}{A(z)} = \frac{1}{1 - a_1' z^{-1}},$$

where $a_1' = a_1/a_0$ giving

$$\hat{x_e}[n] = {a_1'}^n u[n] \approx x_e[n]/a_0.$$

[6] A more general derivation of the autocorrelation method can be found in [Jac89, LO88, Mak75].

Equation 3.1 can be represented in matrix form

$$\mathbf{X_e}\vec{a} = \vec{\delta} + \vec{e},$$

where

$$\mathbf{X_e} = \begin{pmatrix} x_e[0] & 0 \\ x_e[1] & x_e[0] \\ x_e[2] & x_e[1] \\ \vdots & \vdots \\ x_e[L] & x_e[L-1] \\ 0 & x_e[L] \end{pmatrix},$$

$$\vec{a} = [a_0 \quad -a_1]^T,$$

$$\vec{\delta} = [1\ 0 \ldots 0]^T,$$

and

$$\vec{e} = [e_0\ e_1 \ldots e_{L+1}]^T.$$

Applying the least-squares error criterion, we seek to minimize

$$E = \vec{e}^T\vec{e} = (\mathbf{X_e}\vec{a} - \vec{\delta})^T(\mathbf{X_e}\vec{a} - \vec{\delta}).$$

Setting the vector derivative to zero, we get:

$$\frac{\partial E}{\partial \vec{x_e}} = 2\mathbf{X_e}^T\mathbf{X_e}\vec{a} - 2\mathbf{X_e}^T\vec{\delta} = \vec{0},$$

or

$$\mathbf{X_e}^T\mathbf{X_e}\vec{a} = \mathbf{X_e}^T\vec{\delta} = x_e[0]\vec{\delta},$$

which is called the normal equation [Str88] and has solution

$$\vec{a} = (\mathbf{X_e}^T\mathbf{X_e})^{-1}x_e[0]\vec{\delta}.$$

In the first order case, we have

$$\mathbf{X_e}^T\mathbf{X_e} = \begin{pmatrix} \sum_{n=0}^{L}(x_e[n])^2 & \sum_{n=0}^{L-1} x_e[n]x_e[n+1] \\ \sum_{n=0}^{L-1} x_e[n]x_e[n+1] & \sum_{n=0}^{L}(x_e[n])^2 \end{pmatrix},$$

and because we are not interested in the amplitude of the exponential approximation, we can write the normal equation as

$$\mathbf{X_e}^T\mathbf{X_e}\vec{a}' = (x_e[0]/a_0)\vec{\delta},$$

where $\vec{a}' = \vec{a}/a_0$. The second equation in this system of two equations implies

$$\sum_{n=0}^{L-1} x_e[n]x_e[n+1] - a_1' \sum_{n=0}^{L}(x_e[n])^2 = 0,$$

or

$$a_1' = \frac{\sum_{n=0}^{L-1} x_e[n]x_e[n+1]}{\sum_{n=0}^{L}(x_e[n])^2}.$$

Assuming that $a_1' = e^{-T/\tau}$ with sampling period T, we get

$$\tau = \frac{-T}{\ln(a_1')}.$$

This provides the time constant value.

The previous process provides us with, for each of the P strokes, length N vectors,

$$\vec{F'}_i = [cq_{i0}\, cq_{i1}\ \ldots\ cq_{iN-3}\ \ \tau_i\ \ x_{avg_i}]^T,$$

where for $i = 1 \ldots P$, cq_{ij} is the j^{th} approximated constant Q transform component, τ_i is the time decay constant, and x_{avg_i} is the average energy for the i^{th} drum stroke.

Normalization

Because each vector component should carry equal importance in the classifier, the features were normalized to possess zero mean and unit variance. This was done by subtracting the means and dividing by the standard deviations:

$$\vec{F}_{i\,j} = (\vec{F'}_{i\,j} - m_j)/\sigma_j,$$

$$\vec{m} = \frac{1}{P}\sum_{i=1}^{P} \vec{F'}_i,$$

$$\vec{\sigma}_j = \left(\frac{1}{P}\sum_{i=1} P(\vec{F'}_{i\,j} - m_j)^2\right)^{1/2}.$$

Dimensionality Reduction

If c, the number of semitones between successive center frequencies of the transform, is too large, we loose frequency resolution and therefore loose ability to discriminate between different stroke types. Therefore, I determined the best values of c and ω_{min} experimentally. A value of 4 semitones (third octaves) resulted in 26 transform components and feature vector lengths of 28 – quite large for the number of drum strokes per data set (approximately 1000).

To reduce the feature space to a more manageable size, I used principle component analysis (also known as the Karhunen-Loève transform). This method enables us to represent a vector $\vec{F}_i$ of dimension N with a linear combination of $M < N$ vectors from an appropriate orthonormal basis $\vec{u}_j, j = 1 \ldots N$. This approximation is

$$\hat{\vec{F}} = \alpha_1 \vec{u}_1 + \alpha_2 \vec{u}_2 + \ldots + \alpha_M \vec{u}_M,$$

where

$$\alpha_i = \vec{F}_i \vec{u}_i.$$

For a given value of M, we desire the difference between the original and the approximation,

$$\vec{\epsilon} = \vec{F}_i - \hat{\vec{F}} = \sum_{j=M+1}^{N} \alpha_j \vec{u}_j,$$

to be minimal.

It can be shown [The89] that the basis vectors that minimize $\vec{\epsilon}^T \vec{\epsilon}$ for a given M are the eigenvectors corresponding to the M largest eigenvalues of the correlation matrix of vectors $\vec{F}_i$,

$$E[\vec{F}\vec{F}^T].$$

The maximum likelihood estimation of the correlation matrix, called the sample correla-

tion matrix[DH73], is defined as

$$\mathbf{R} = \frac{1}{P}\sum_{i=1}^{P} \vec{F}_i \vec{F}_i^T,$$

and is what I used. Further, it can be shown that this set of eigenvectors accounts for as much of the data's variances as possible for a given M [HKP91, The89] (the eigenvector corresponding to the largest eigenvalue points in the direction of the greatest variance of the data, the eigenvector corresponding to the next largest eigenvalue points in the direction of the next greatest variance, and so on). So, by choosing these eigenvectors as a basis, we are representing the features using the most *important* directions.

We therefore compute the N eigenvectors and eigenvalues of the sample correlation matrix, sort them, and then choose M eigenvectors where the sum of the largest M corresponding eigenvalues is some percentage p of the total sum of the eigenvalues. The parameter p is determined empirically. It is chosen with the goal of minimizing the number of dimensions while retaining the ability for the classifier to discriminate between feature vectors.

3.2.3 Classification

I considered several unsupervised learning strategies, including Hebbian self-organizing feature extractors [HKP91] and a competitive learning strategy based on adaptive resonance theory [HKP91, CG91]. However, I decided on a more traditional clustering approach because it was much easier to implement. I used the K-means clustering procedure [The89, DH73] (also called the Basic ISODATA procedure) which I define here. Note that in this definition, the words *cluster*, *class*, and *stroke type* are used interchangeably.

K-Means Algorithm

Input: A list of feature vectors of dimension M and the upper limit K of the number of clusters to assign to the feature vectors.

Output: For each feature vector, a number between 1 and K indicating the cluster to which it belongs

step 1: Create K clusters. Arbitrarily assign each feature vector $\vec{\hat{F}}_i$ to one of the K clusters.

step 2: Compute the means of each cluster.

step 3: Reassign each feature vector to the cluster to which it is nearest.

step 4: If any reassignments occurred in step 3, go to step 2, otherwise stop.

It can be shown [DH73] that the simple K-means procedure optimizes the sum-of-squared-error criterion function, defined by

$$\mathbf{m}_i = \frac{1}{n_i} \sum_{\vec{F} \in \mathcal{F}_i} \vec{F}$$

and

$$J_e = \sum_{i=1}^{K} \sum_{\vec{F} \in \mathcal{F}_i} \|\vec{F} - \mathbf{m}_i\|^2,$$

where $\mathcal{F}_i$ is the set of vectors currently contained in cluster i, and n_i is the size of $\mathcal{F}_i$. That is, for a given K, the K-means procedure attempts to find a clustering of the samples for which J_e is minimum. From the equations, it can be seen that this is essentially a minimum variance criterion; K-means tries to find a clustering where the intra-cluster variance is minimum.

A major problem with the simple K-means algorithm is determining the number of clusters K. This is known as the problem of validity. Normally, if K is known, K-means works well for dense, well-separated clusters (i.e., small intra-cluster variance and large inter-cluster variance). But because our data set is essentially unknown, we have no *a priori* means of determining K. Much literature exists on testing the validity[7] of a given K. The approaches are usually either based on statistical assumptions about the process generating the data or assume an optimizeable criterion function [DH73, And73, JD88, Bij73, DHJO87, AB84, ACD87].

Rather than formulating the problem in such a way, and because I did have rough bounds on K (i.e., $K \geq 3$ and $K \leq 15$, a rather small range), I determined it empirically: I ran K-means several times and found the K which worked best according to my ear.

3.2.4 Results of Automatic Classification

I applied the classification procedure to the tumbao, quinto, and segundo data. I corrected any errors by hand because they were unacceptable for later analysis. The results which follow are a comparison between the direct output of the algorithm and the hand-corrected

[7] We also encounter this problem once again in Chapter 5.

	Tumbao	Quinto	Segundo
Segmentation			
sEws	5ms	5ms	5ms
engergyFraction	0.5%	0.5%	0.5%
breakOff	10ms	10ms	10ms
Features			
c	4	4	4
ω_{min}	50	30	30
fEws	5ms	5ms	5ms
Resulting Num Features	31	31	28
Karhunen-Loève			
Percent p	90%	90%	90%
Resulting Num Features	4	5	9
K-Means Procedure			
K	5	4	5
Correct Num Classes	6	8	10

Table 3.3: Classification Parameters and Results.

versions. The programs were run with the parameters listed in Table 3.3. I will describe the results for the Tumbao and Quinto only.

I ran K-means several times trying different values for K. In addition, select samples were manually extracted from the signal. I used the output of each K-means run to synthesize the performance by matching cluster numbers to hand-extracted samples. A synthesis program[8] retriggered the hand extracted samples at the appropriate time using the stroke types determined by K-means. The classification that produced the most correct sounding performance was accepted.

I accepted a value of $K = 5$ for the K-means algorithm on the tumbao data. There are 997 tumbao samples. Once manual correction had been completed, there turned out to be 6 distinct stroke types. K-means, with $K = 5$, correctly classified 80% of the drum strokes, as can be seen in Table 3.4. Notice that we have done quite well (above 85%) with those drum strokes that have high relative frequency (above 20%); and we have done poorly on those strokes that have low relative frequency. This confirms our expectation about the K-means algorithm: if there are not enough samples in a class to create a relatively dense

[8]Csound[Ver].

Class	Total Occurrences	Relative Freq	Num Correct	% Correct
1	42	0.0421264	0	0%
2	384	0.385155	339	88.2812%
3	212	0.212638	211	99.5283%
4	292	0.292879	252	86.3014%
5	64	0.0641926	0	0%
6	3	0.00300903	0	0%
Totals	997	1	802	80.4413%

Table 3.4: Classification Results for Tumbao, $K = 5$.

Class	Total Occurrences	Relative Freq	Num Correct	% Correct
1	42	0.0421264	0	0%
2	384	0.385155	338	88.0208%
3	212	0.212638	134	63.2075%
4	292	0.292879	250	85.6164%
5	64	0.0641926	0	0%
6	3	0.00300903	3	100%
Totals	997	1	725	72.7171%

Table 3.5: Classification Results for Tumbao, $K = 6$.

cloud, samples from that class will tend to be adopted by another.

Once I knew the correct number of classes, out of curiosity I re-ran the K-means algorithm with $K = 6$; the results are shown in Table 3.5. The most striking feature of these results is that the total percent correct decreased from 80% to 73%, especially because class 6, with three samples, is classified 100% correctly.

Table 3.6 shows the K-means results with $K = 4$, the accepted value on the quinto data. I again corrected any errors manually – there were eight distinct stroke types – and, once again ran K-means on the quinto data, this time with $K = 8$. The results are shown in Table 3.7. Once again, we see that the total percent correct went from 80%, when using an incorrect number of classes, down to 52%, when using the correct number.

This phenomenon can be explained by considering the minimum variance criterion function J_e which K-means attempts to optimize[DH73]. When the K-means algorithm is allowed to create more classes, rather than create a new cluster surrounding a small number of points that constitute their own class, K-means will try to split a heavily populated class into smaller ones. This is because creating a cluster surrounding a small group will

Class	Total Occurrences	Relative Freq	Num Correct	% Correct
1	299	0.318424	299	100%
2	339	0.361022	336	99.115%
3	209	0.222577	135	64.5933%
4	12	0.0127796	8	66.6667%
5	41	0.0436635	0	0%
6	4	0.00425985	0	0%
7	1	0.00106496	0	0%
8	34	0.0362087	0	0%
Totals	939	1	778	82.8541%

Table 3.6: Classification Results for Quinto, $K = 4$.

Class	Total Occurrences	Relative Freq	Num Correct	% Correct
1	299	0.318424	295	98.6622%
2	339	0.361022	86	25.3687%
3	209	0.222577	83	39.7129%
4	12	0.0127796	5	41.6667%
5	41	0.0436635	5	12.1951%
6	4	0.00425985	0	0%
7	1	0.00106496	0	0%
8	34	0.0362087	17	50%
Totals	939	1	491	52.2897%

Table 3.7: Classification Results for Quinto, $K = 8$.

have only a small effect on J_e, whereas splitting a dense and large cluster will create two smaller clusters, each with much less variance, having a large overall decrease in J_e. For perfect clustering, therefore, it is important to have well balanced classes when using the K-means procedure. Unfortunately, my data was only moderately balanced.

Is 80% correct the best we can do? When I listen to some of the incorrectly classified drum strokes in isolation, I also have difficulty determining which stroke was played. However, once the stroke is in context – within a musical phrase performed relative to the metric form – I have no trouble identifying the stroke. The basic problem is this: too many drum strokes sound alike when out of context. I would not be surprised to find an illusory effect of some kind, an *illusory difference*, in which drum strokes which sound identical out of context sound completely different in a musical context (more on this in Section 6.1.5). Therefore, without fundamental information such as the drum stroke's tatum number, its position in a musical phrase, and perhaps even a deeper understanding of the performance itself, it seems unlikely to produce a classification scheme that rivals human performance.

Nevertheless, even at 80% correct, the tedious process of hand classification was reduced to a much easier and faster process of checking for errors, a time saver indeed.

Chapter 4

Timing Analysis

4.1 Introduction

To err is human. Yet most users of drum machines and music sequencers strive to eliminate errors in musical performance. In fact, some computer musicians[1] never turn off the quantize option, destroying forever "flaws that make the performance sound sloppy." On the other hand, some computer musicians complain about the mechanical quality of computer music. They call for the development of techniques that would enable computers to sound better, i.e., more "human."

There are two orthogonal criteria of performance. The first is sheer technical proficiency. Clearly, computers have long surpassed humans on this axis – no one can perform faster than a computer. The other is expressivity, something more elusive, something that gives music its emotion, its feeling, its joy and sorrow, and its humanity. Music exudes humanity; computer music exudes uniformity. This, I strive to eliminate.

4.2 Timing Extraction Algorithm

The algorithm presented herein extracts the quantized score, the tempo variation, and the deviations from a musical performance. Combined with the algorithms given in Chapter 3, it can be considered an automatic transcriber, with the added feature of providing expressive timing information. The input to the algorithm is a list of attack times. Section 2.3

[1] I use "computer musician" to refer to anyone who uses a computer to create music and "computer music" to refer to music created thereby.

suggests that there must be some (per person) upper limit on tempo oscillation frequency. That is, any timing variation in the performance not accounted for by tempo variation because of its high frequency must be owing to deviations. This is a crucial point and the timing extraction algorithm was developed with this assumption in mind.

The main goal of the algorithm is to obtain information that can transform quantized musical phrases into expressive ones. This task becomes easier if we use the knowledge of a performer. Therefore, the algorithm is given complete metric knowledge. That is, it knows the time signature, the number of tatums per beat, the number of beats per measure, and where the beginning of the measure is (the answer to "where is one?"). The algorithm is not a music quantizer. Music quantizers are often used by MIDI sequencers, score editing programs, and electronic drum machines; they produce a performance in which the duration between all note onsets is a multiple of some time quantum. This algorithm not only extracts the metric score and tempo track (similar in function to the tempo tracking in [Ver84, DMR87]) but also extracts the deviations.

There are two versions of the algorithm: the first is primarily for percussive music; the second is slightly more general. The algorithms both require a *reference instrument* and a *performance instrument*. The reference instrument is used to extract tempo. The algorithms produce the expressive timing of a performance instrument relative to the tempo defined by the reference instrument. In an ensemble, any instrument (including the reference instrument) may be considered a performance instrument.

In version I of the algorithm, the reference instrument must repeatedly play a known predetermined pattern. The period of the pattern must be an integer multiple of the measure duration. Percussive music normally contains such an instrument (e.g., a bell, a clave, etc.) so this is not an unreasonable requirement.

The algorithm first computes a tempo function using the reference instrument. The tempo function is then transformed into a tatum duration function – tatum duration as a function of tatum number. The tatum duration function determines a tempo-normalized metric grid, i.e., a time grid spaced so that grid markers determine the time points of each tatum. The metric grid is then used as a reference to judge the performance instrument. For each performance instrument attack, the deviation is its distance to the nearest grid marker.

Let L be the number of tatums per measure, R be the number of reference instrument attacks per measure, $x[n]$ be the n^{th} reference instrument attack time, $y[n]$ be the n^{th}

performance instrument attack time, and let $z[n] = x[n \times R]$ be our estimate of the starting time of the n^{th} measure (if reference instrument attacks do not fall on the measure starting points, we interpolate, add entries to $x[n]$, and pretend that it does). $y[0]$ must lie past the first measure's starting point.

For $n = 0 \ldots R-1$, we compute $P[n]$, the average fraction of a measure of the time between reference instrument attacks n and $n+1$:

$$P[n] = \frac{1}{M} \sum_{m=0}^{M-2} \frac{x[mR+n+1] - x[mR+n]}{z[m+1] - z[m]},$$

where M is the number of measures in the performance. If the performer is playing very uniformly (i.e., nearly quantized), $P[n]$ may be obtained directly from the pattern (or score) rather than from the attack times.

Next, we compute the *rough* tempo function

$$T'[n] = \frac{x[n+1] - x[n]}{P[n \bmod R]}.$$

$T'[n]$ provides an estimate, at time $x[n]$, of the measure duration. The reference instrument informs the ensemble what the tempo is at any one point in time. The performance instrument, if it is dominant in the ensemble (e.g., a lead drum), controls when the tempo speeds up and slows down. In short, the reference instrument *defines* the tempo while the performance instrument *controls* the tempo. Because the actual tempo might take a while to be reflected in the reference instrument, when obtaining the timing of a dominant performance instrument we look slightly ahead, and compute

$$T[n] = \frac{1}{C+1} \sum_{k=n}^{n+C} T'[k],$$

where C is a parameter determining how far into the future we should look. C depends on the performance instrument, and could equal zero. Accordingly, $T[i]$ might or might not be an anticipatory measure duration estimate.

Creating a continuous time function, we next linearly interpolate[2]

$$D(t) = T[n] + (T[n+1] - T[n]) \times \frac{t - x[n]}{x[n+1] - x[n]},$$

where

$$n = \{n : x[n] \leq t < x[n+1]\}.$$

It follows that $D(t)$ is an estimate at time t of the measure duration.

Clearly, as the tempo decreases, $D(t)$ increases and $1/D(t)$ decreases. The goal is to find the tempo-normalized time points that define the tempo. These time points are used to compute the deviations. Therefore, for each measure, we find the time points that divide the area under $1/D(t)$ into L equal area regions. The time points provide a tatum time function: a function that gives the time point of each tatum.

So, for each measure m, $0 \leq m < M - 1$, and each tatum i in measure m, $1 \leq i < L$, we find $b_L[mL + i]$ where

$$z[m] \leq b_L[mL + i] < z[m+1],$$

and

$$\frac{\int_{z[m]}^{b_L[mL+i]} 1/D(t)dt}{\int_{z[m]}^{z[m+1]} 1/D(t)dt} = i/L.$$

The array $b_L[n]$ is the L-tatum per measure time location of tatum n.

Next, we compute the first order forward difference by linear convolution,

$$d'[n] = b_L[n] * (\delta[n+1] - \delta[n]) = b_L[n+1] - b_L[n],$$

where $\delta[n]$ is the unit sample sequence.[3] Note that this is a discrete first-order differentiation process.

To filter out high frequency variation, we again use linear convolution and compute

$$d[n] = d'[n] * h[n],$$

where $h[n]$ is either an FIR low-pass filter with a desired stop-band, or a Savitzky-Golay

[2]Higher order interpolation methods could be used here, but they probably would not significantly alter the final outcome.

[3]$\delta[0] = 1, \delta[n \neq 0] = 0$.

smoothing filter[PTVF92, LO88]. Because we are going to re-integrate, the filters must have a unity DC component. That is, each filter must have $H(0) = 1$, where $H(\omega)$ is the filter's frequency response. This step removes high frequency timing variation from $d'[n]$. Therefore, the array $d[n]$ is our estimate of the duration of the n^{th} tatum. Next, we recover the tatum positions from $d[n]$ by convolving with the unit step sequence $u[n]$,[4]

$$b[n] = (d[n-1] + b_L[0]\delta[n]) * u[n] = b_L[0] + \sum_{i=0}^{n-1} d[n].$$

This is a discrete integration process. The array $b[n]$ then provides the time position of tatum n.

For each performance instrument attack $y[n]$, we find the closest tatum. The distance from $y[n]$ to the closest tatum is the stroke's deviation. That is,

$$\text{devs}[n] = y[n] - b[j]$$

and

$$\text{quants}[n] = j,$$

where

$$j = \underset{j}{\text{argmin}} \mid y[n] - b[j] \mid \ .$$

The array devs$[n]$ is the deviation function and quants$[n]$ is an array of tatum numbers. Therefore, the quantized score is given (in tatums) by quants$[n]$, the tempo variation by $b[n]$, and the deviations by devs$[n]$. A positive deviation means the attack occurred after the tatum and a negative one means it occurred before.

Version II of this algorithm does not require the reference instrument to play repetitively. It does require the complete score or quantized representation. Some methods for automatic score transcription are provided in [Ros92, CRMR+84, Moo77]. The score may be used directly if it is already known and the goal is only to obtain the tempo variation and deviations.

The main distinction between versions I and II of this algorithm is that in version II,

[4] $u[n < 0] = 0, u[n \geq 0] = 1.$

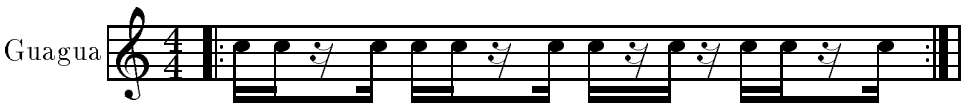


Figure 4-1: The Guagua Pattern.

$P[n]$ is not computed from the performance. Instead, it is obtained directly from the score. That is, $P[n]$ becomes the measure fraction of the time duration between reference instrument attacks n and $n+1$ according to the score. For example, an eight note in $\frac{4}{4}$ time would produce a value of $1/8$ and a quarter note in $\frac{7}{8}$ time a value of $2/7$. The starting time of each measure $z[n]$ is computed from $x[n]$ also according to the score. If no $x[n]$ falls at the beginning of a particular measure, we interpolate and create an estimation of the measure starting time. The only other difference is the following:

$$T'[n] = \frac{x[n+1] - x[n]}{P[n]}.$$

The tempo variation $b[n]$ and the deviations devs$[n]$ are computed as in version I. Version II, however, is intended for the case of identical reference and performance instruments (e.g., solo piano) and for obtaining timing data from a performance of written music. It computes the tempo variation as the low-pass filtered performance timing variation. High frequency performance variation remains as the deviations. The trick is to find the desired stop-band cutoff frequency, something that largely depends on the musical style.

4.3 Timing Analysis Results

I applied Version I of the timing extraction algorithm to the data obtained from Los Muñequitos de Matanzas described in Chapter 3. I obtained timing data from the following performance instruments: the quinto (high), the segundo (middle), and the tumbao (low) drum. The reference instrument was the guagua. An approximation of the reference instrument pattern can be seen in Figure 4-1. The computed values of $P[n]$ are listed in Table 4.1, along with the values that would be obtained from a perfectly quantized performance. Clearly, this reference instrument is far from quantized. I ran the algorithm with $C = 3$ and $h[n] = (u[n] - u[n-5])/5$. Therefore, $h[n]$ is a rectangular moving window average. What follows are the results of the segundo only.

n	0	1	2	3	4	5	6	7	8	9	10
$P[n]$	0.057	0.122	0.069	0.059	0.120	0.065	0.114	0.144	0.053	0.122	0.074
Fig. 4-1	0.063	0.125	0.063	0.063	0.125	0.063	0.125	0.125	0.063	0.125	0.063

Table 4.1: Computed Values of $P[n]$.

4.3.1 Tempo Variation and Deviations

The tempo variation $d[n]$ is plotted in Figure 4-2. Although it appears to contain considerable high frequency variation, the abscissa scale informs us otherwise. Figure 4-3 shows the DFT magnitude of $d[n]$. The abscissa is measured in normalized frequency units where 0.5 corresponds to the Nyquist frequency. The frequency units, however, are in cycles per tatum, not in Hz. The DFT magnitude is not plotted for $f > 0.1$ (variation of less than 10 tatums per cycle) since the energy quickly drops off to insignificance (below 0.01). Not shown in the plot is the large DC value caused by the always positive tempo. Also, the peaks at 0.0620 and 0.0630 correspond to 16.13 and 15.9 tatums per cycle respectively. It is probably more than coincidental that 16 is both the number of tatums per measure and a large component in tempo variation's frequency response.

Figure 4-4 shows a plot of the deviations for the segundo performance. In this format it is difficult to discern any structure in the data. Furthermore, the deviations for the quinto and tumbao look roughly the same when plotted in this way.

Figure 4-5, shows a 30 bin histogram of the deviations for the quinto, segundo, and tumbao, respectively. Notice that they are all slightly centered to the left of zero, implying that, in general, they are all playing slightly on top of the beat. Also, the histograms of the quinto and segundo look approximately Gaussian distributed. Could the deviations simply be the result of an independent and identically distributed random Gaussian process? We shall see (also see Section 5.1.1).

The deviation plots displayed above provide a deviation value whenever there is a stroke on a tatum. In between strokes, there are essentially *missing samples*. Fortunately, a form of spectral analysis is possible. The Lomb normalized periodogram [PTVF92] is a method for magnitude spectrum analysis of either unevenly sampled signals or signals with missing samples. It is commonly applied to astrophysical data where regular sampling is not possible.

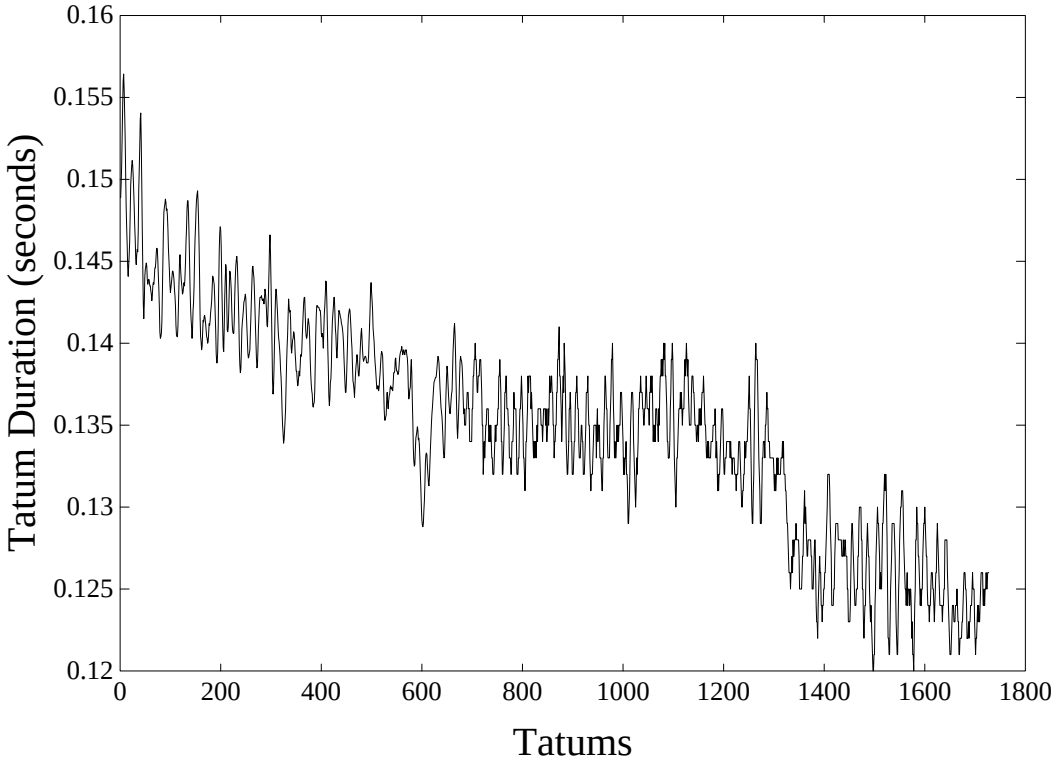


Figure 4-2: Muñequitos Tempo Track $d[n]$.

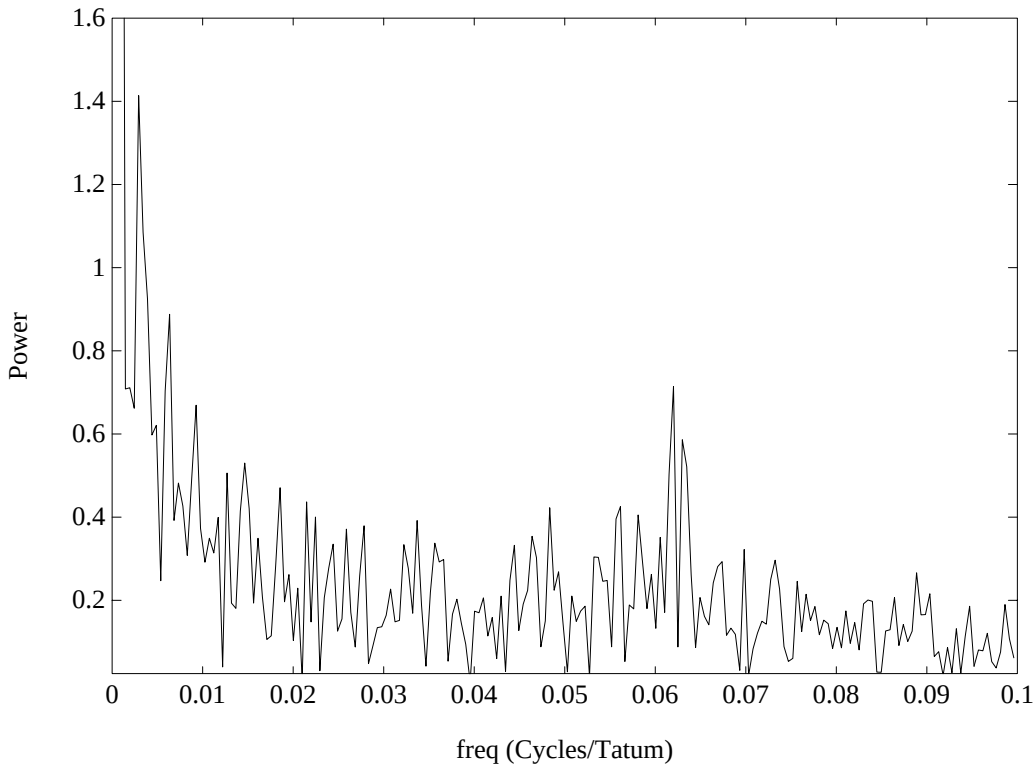


Figure 4-3: DFT Magnitude of the Tempo.

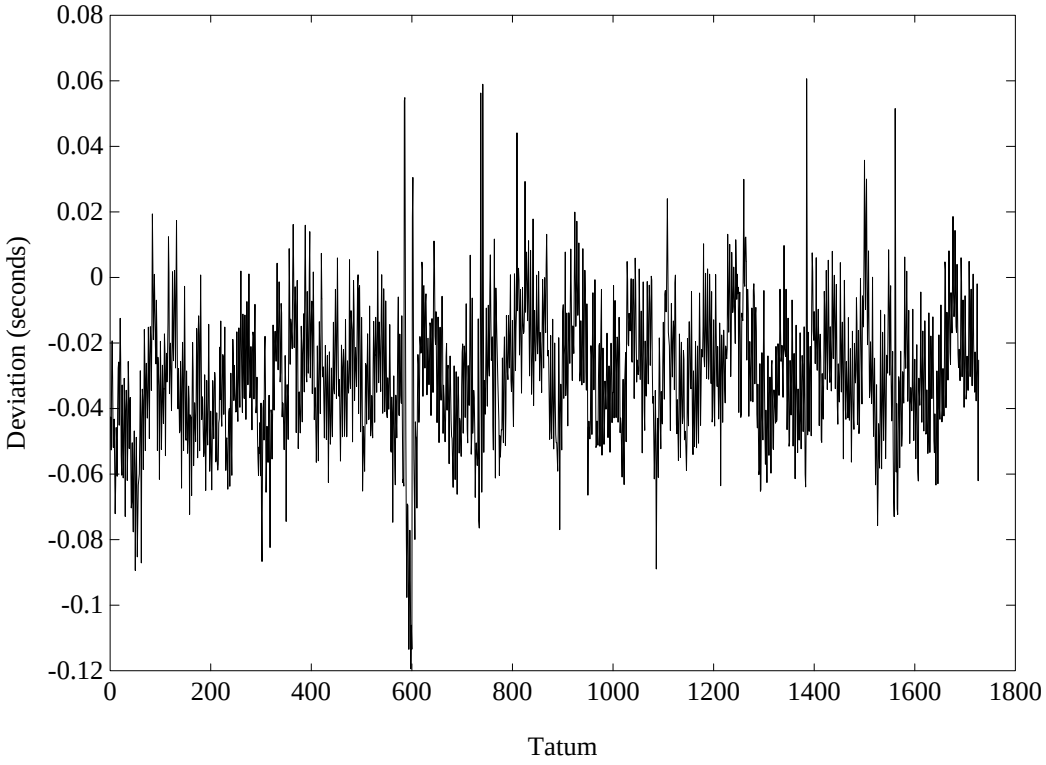


Figure 4-4: Segundo Deviations.

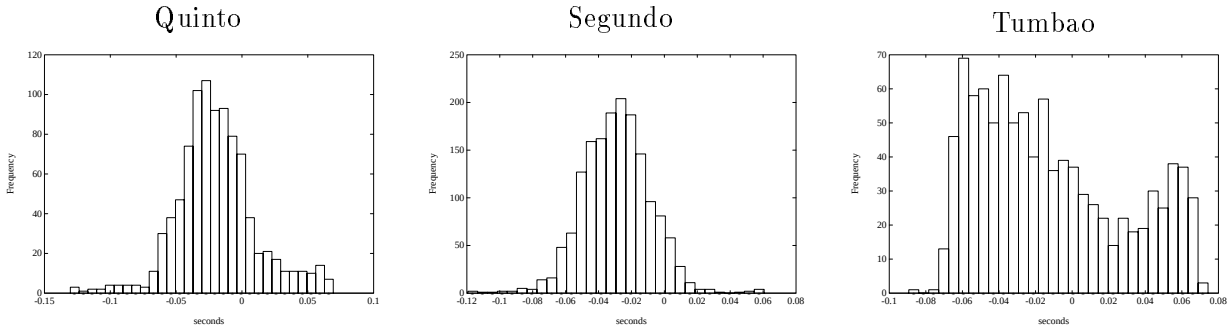


Figure 4-5: 30 Bin Deviation Histograms.

For each desired frequency point ω, the Lomb periodogram is equivalent to running linear least squares fitting to the model,

$$h(t) = A\cos(\omega t) + B\sin(\omega t).$$

The power of a spectral component at frequency ω is proportional to the degree to which the signal fits this model (the best possible values of A and B). The power also corresponds to statistical significance levels against the null hypothesis of random Gaussian noise. The significance level is the probability that a random Gaussian process would generate this power level.

Spectral analysis of unevenly sampled signals can be performed by interpolating, obtaining samples at evenly spaced time points, and then running a DFT. This method, however, can lead to large errors, particularly when large gaps in the original signal lead to excessive low frequency in the interpolated signal. Because the Lomb periodogram is essentially doing least squares analysis, only the signal's original sample points are used to determine the spectral content. It is therefore insensitive to this sort of error.

When dealing with evenly sampled data, the highest representable frequency is half the sampling rate, or the Nyquist frequency. Therefore, a DFT of a real signal provides useful spectral information only up to this rate (assuming there is no aliasing). In unevenly sampled data, however, the highest frequency represented in the signal is not so easily determined. We therefore do not know the highest spectral component the Lomb periodogram should compute. The approach used in [PTVF92] bases things on the "average" Nyquist frequency, defined as

$$f_c = N/(2D),$$

where N is the number of samples in the signal and D is the total duration. This would be the real Nyquist frequency if the signal was evenly sampled and we were using a normal DFT method. We therefore ask for spectral components up to some multiple, *hifac*, of the average Nyquist frequency. An additional parameter, the over-sampling factor *ofac*, controls the frequency spacing of the spectral plot. Specifically, the frequency spacing in the spectral plot is $\Delta f = 1/(D \times ofac)$. If we sample up to the frequency $f_{hi} = f_c \times hifac$, then

this implies that the number of different frequencies provided as output is

$$N_f = f_c/\Delta f = \frac{ofac \times hifac}{2} N.$$

For most applications, $hifac = 2$ and $ofac = 4$. In our case, however, we know that because the deviations are a function of tatum number, we can never have a frequency faster than half the tatum rate (the Nyquist sampling theorem gives us that result). Therefore, the upper limit f_{hi} should always be 0.5 and *hifac* should always be 1.

As with any spectral method, a short-time [LO88] version can be developed that finds the spectral components over segments of the input signal, rather than over the entire signal at once. We can thus see the spectral content of the signal evolve over time, rather than viewing it as a static one-shot image. In our case, the segments of the input are fixed in size and are specified in units of tatums. Most of the time, however, a particular segment will contain a different number of drum strokes than other segments. Then, the high frequency computation described above will produce different values of N_f for different segments. To avoid this problem, we compute the per-segment *hifac* value as

$$hifac_i = \frac{\text{segBounds}_i}{\text{segStrokes}_i},$$

where segBounds_i is the bounds, in tatums, of the i^{th} segment (the tatum distance between the first and last stroke within the segment), and segStrokes_i is the number of strokes in the i^{th} segment. Therefore, f_{hi} will be 0.5 and N_f will be the same for all segments.

Hoping to uncover some structure in the data, I applied this *short-time Lomb periodogram* to the quinto, segundo, and tumbao deviations. Because the deviation functions are plotted as a function of tatum number, frequencies are neither in Hz, nor cycles per sample, but rather in units of cycles per tatum. A frequency of f cycles per tatum means that a single sinusoidal period will complete in $1/f$ tatums.

Like the short-time DFT, the choice of the window size (or D, in our case) is crucial for creating these plots. If D is too small, we will get good temporal resolution (any quick change in deviation patterns will appear in the plot) but poor frequency resolution (discriminating between adjacent frequencies will be difficult). The opposite is true if D is too large. Choosing the window size, however, is not obvious. Therefore, when displaying the periodogram, I show two spectral deviation plots per drum, one with a window size (ws) of

32 tatums (two measures) and overlap (ov) of 24 and the other with a window size of 100 (an arbitrary number) and overlap of 80 tatums.

Figure 4-6 shows the short-time Lomb normalized periodogram for the segundo deviations listed in Figure 4-4. The front axis is, again, measured in normalized frequency units (cycles per tatum) where 0.5 is the Nyquist frequency. Notice the strong peak at 0.25 cycles per tatum, implying considerable deviation periodicity near 4 tatums per cycle. The segundo performance, in fact, largely consists of a repeating 4 tatum phrase. Figure 4-7 shows the short-time periodogram of the segundo with a larger window size. The better frequency resolution is apparent: the peak significantly narrows centered right on 0.25, and other small peaks appear at 0.125, 0.166, and 0.333. Clearly, this confirms that structure does exist in the deviations.

Figures 4-8 and 4-9 show the short and long window size versions of the periodogram for the quinto and tumbao tracks. These plots distinctly show that there is structure in the signals, but how significant are the results? Table 4.2 shows the distribution of the significance values. Each row gives a significance value (left column). The right columns show the number of drum segments whose peak value is less significant than the value given in the left most column. The last row shows the minimum significance value for each drum. The *Random* column shows the results for deviations generated from an independent and identically distributed Gaussian random process with the same mean and variance as the segundo deviations. The quinto deviations are not as significant as the others, mainly because the quinto is primarily an improvisational instrument and does not often play periodic phrases. Overall, though, the results for the real deviations are much more significant than the random process. In fact, the segundo and tumbao results are striking.

The spectral analysis of the tempo variation and deviation signals confirms that they contain valid information and that they are not merely random noise. In other words, *there's gold in them thar signals.*

4.3.2 Performance Synthesis

The performance was synthesized[5] by triggering select samples of the original performance. The automatic note classification algorithm developed in Chapter 3 completed the score.

[5]Using Csound[Ver].

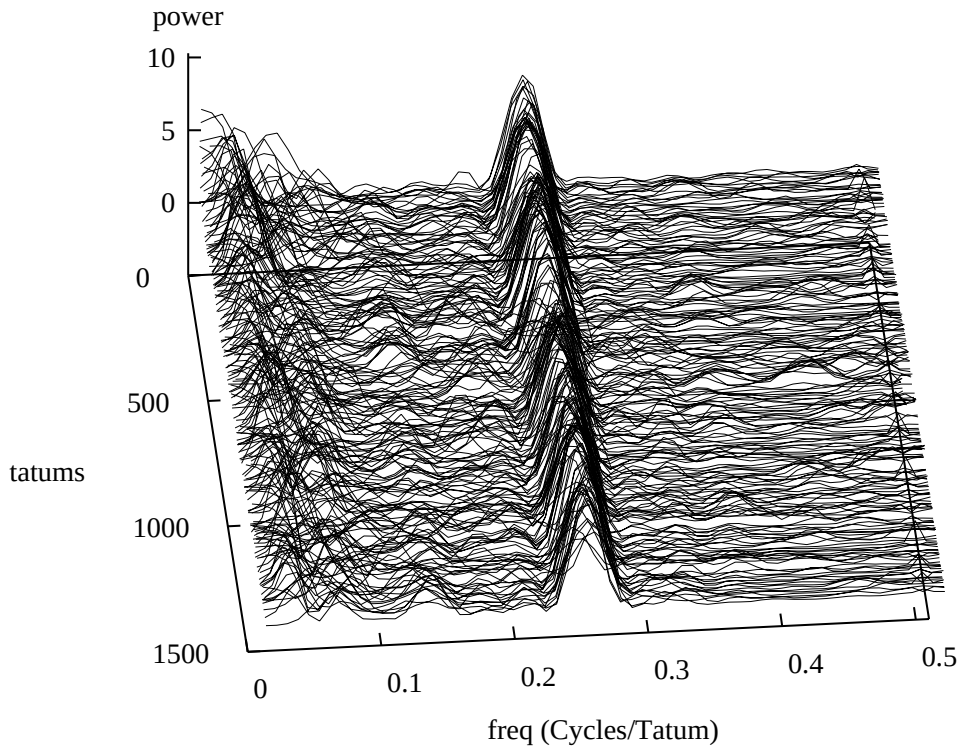


Figure 4-6: Lomb Normalized Periodogram: Segundo Deviations, $ws = 32$,$ov = 23$.

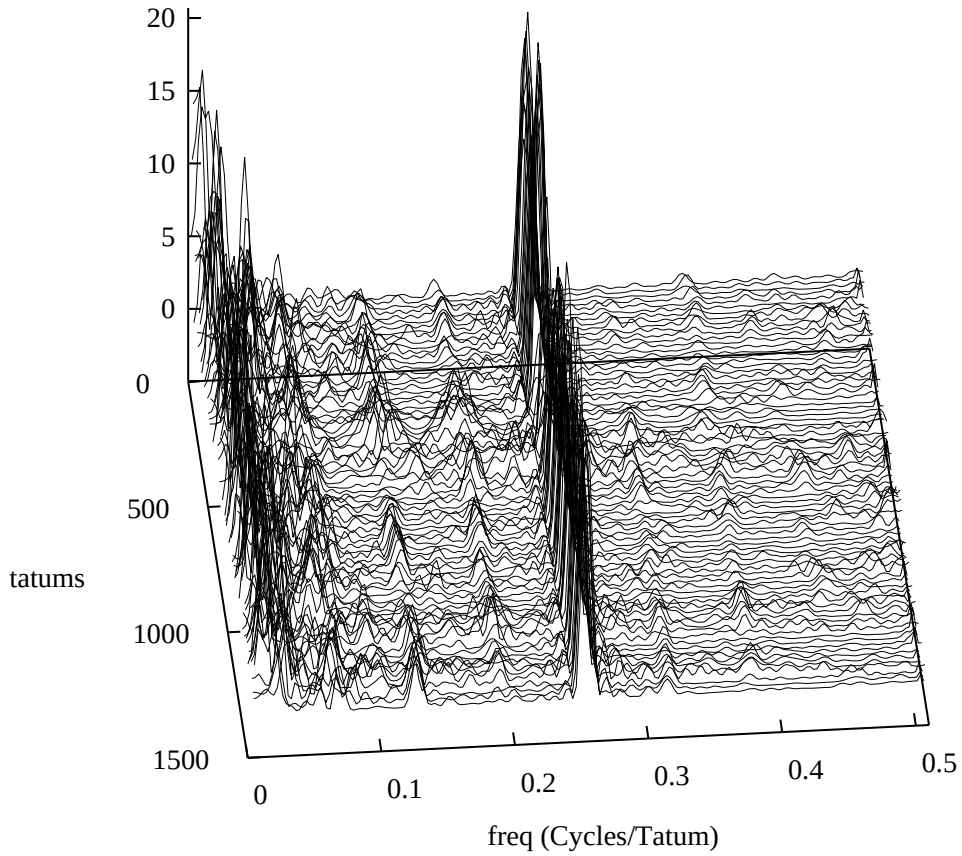
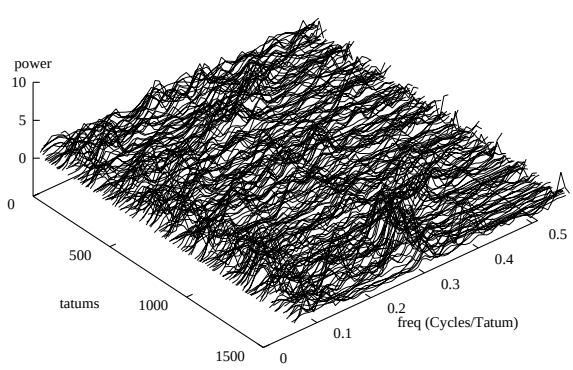
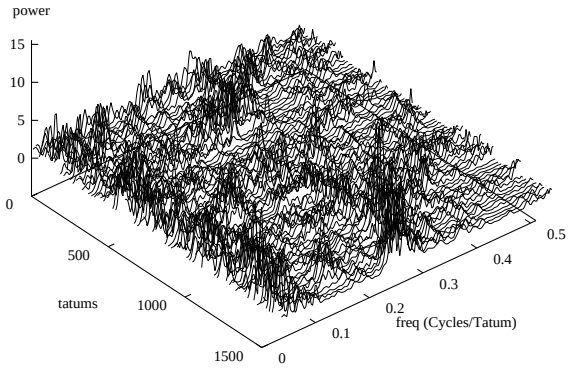


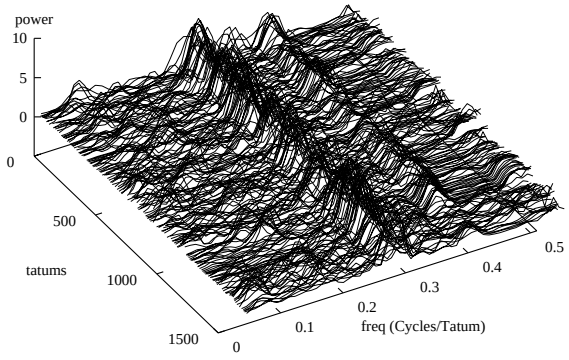
Figure 4-7: Lomb Normalized Periodogram: Segundo Deviations, $ws = 100$,$ov = 80$.

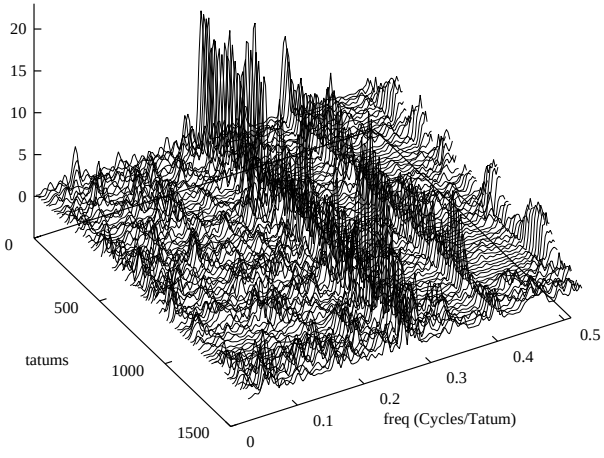




$ws = 32, ov = 24$ $ws = 100, ov = 80$

Figure 4-8: Lomb Normalized Periodogram: Quinto Deviations.





$ws = 32, ov = 24$ $ws = 100, ov = 80$

Figure 4-9: Lomb Normalized Periodogram: Tumbao Deviations.

Table 4.2: Number of Spectral Segments with Significance $< p$, $ws = 100$,$ov = 80$.

p	Random	Segundo	tumbao	quinto
1	82	82	82	82
0.5	40	82	82	62
0.2	16	82	75	41
0.1	6	82	67	29
0.05	3	82	62	17
0.02	2	79	55	11
0.01	1	79	52	8
0.005	1	77	49	7
0.002	0	74	47	5
0.001	0	70	45	4
0.0005	0	68	42	3
0.0002	0	64	38	3
0.0001	0	57	32	2
Min. Sig.	0.002	3.338×10^{-9}	9.966×10^{-9}	1.760×10^{-5}

The synthesized examples consist of the following:[6]

1. Direct – by triggering the samples at the appropriate time.
2. Quantized – using a constant tempo equal to the overall average.
3. Quantized – using $b[n]$ as the tempo.
4. Quantized – with devs$[n]$ added to the n^{th} attack.
5. Quantized – with random Gaussian deviations added to each attack time. The Gaussian process had the same mean and variance as the devs$[n]$ array.
6. Quantized – with per-tatum random Gaussian deviations added to each attack time. In this case, there were 16 independent Gaussian processes, each with a different mean and variance. The mean and variance for the i^{th} process was the same as the mean and variance of devs$[n \bmod i]$.[7]

Most people who listen to these examples say that number 4 sounds most like the original, observing that only 4 contains the "feel" of the original performance. In addition,

[6] Also see Appendix C.
[7] Also see Section 5.1.1.

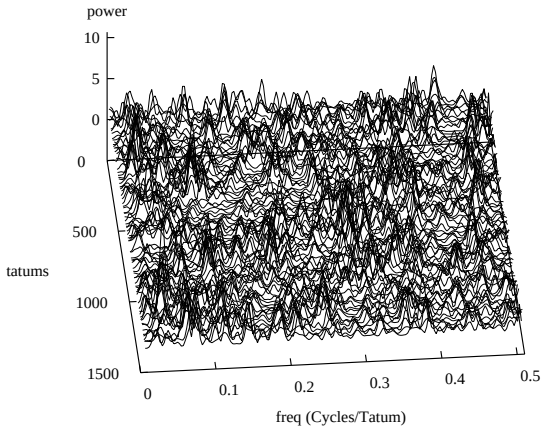


Figure 4-10: Lomb Normalized Periodogram: Random Gaussian Deviations.

numbers 5 and 6 are considered, in general, to sound "sloppy" and "random" (also see Section 5.1.1). Accordingly, Figure 4-10, showing a periodogram for the deviations in synthesis 5, confirms that there is a lack of structure. As expected, synthesis 2 sounds mechanical. Unexpectedly, even synthesis 3 sounds mechanical; tempo variation apparently does not matter. In general, without the correct deviations, the performance sounds colorless and cold – with them, it sounds rich and alive.

Consequently, I propose that, in addition to the ongoing studies of tempo variation, we begin a concentrated study on performance deviations.[8] Combining both tempo variation and deviations could eventually produce the full effect of rhythmic expressivity.

4.4 Conclusion

In this chapter, I have used the separate rhythmic elements defined in Chapter 2 for rhythmic analysis and have demonstrated the importance of deviations for representing and reproducing expressive timing in percussive musical phrases. Furthermore, I have demonstrated that the deviations extracted from a performance are indeed meaningful. The Lomb normalized periodogram provides quantitative evidence and the synthesis provides empiri-

[8] Appendix E describes a new electronic drum machine interface for the exploration of deviations.

cal evidence that the timing data is not random noise – that, in fact, deviations play a vital role in expressive timing and, therefore, should be analyzed, comprehended, and utilized in the pursuit of electronic means to reproducing human musicality.

Chapter 5

Deviation Learning System

The least initial deviation from the truth is multiplied later a thousandfold.

Aristotle
On the Heavens

In this chapter we bridge the gap between symbolic and numerical approaches to music information processing. Specifically, we have the following goal: given a quantized musical phrase, produce the deviations that make it sound natural – plainly stated, take a divested construct, and impart both life and beauty.

5.1 First Attempts

In Figure 5-1 the first 170 tatums of the quinto score can be seen. The top plot shows deviation as a function of tatum number and the bottom one shows the stroke type (the displayed section uses only three different stroke types). The vertical dashed lines denote the measure boundaries (16 tatums per measure). As can be seen, there is no obvious structure in the plots and from this standpoint the data even appears random. Similarly, Figure 5-2 displays the first 120 tatums of the segundo score. There seems to be slightly more structure; the segundo is playing a repetitive phrase. Still, no obvious plan seems to be generating these deviations.

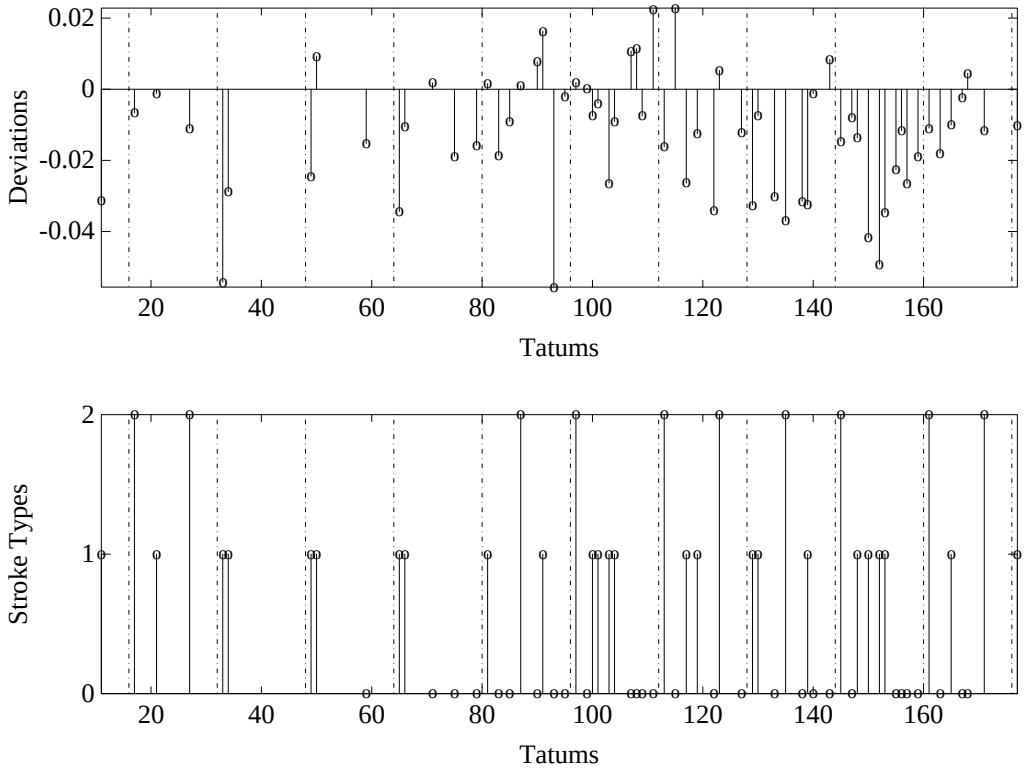


Figure 5-1: Quinto Score. First 170 Tatums.

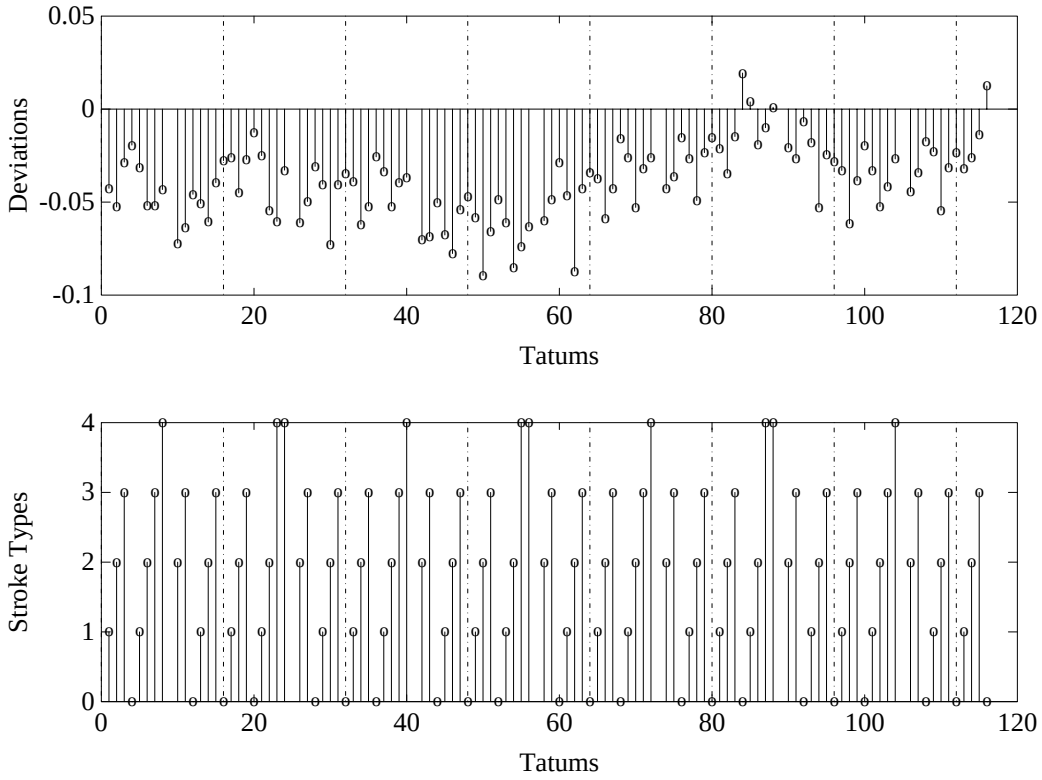


Figure 5-2: Segundo Score. First 120 Tatums.

5.1.1 Simple Random Processes

The deviation distributions might provide some insight into the underlying structure of the data. In Figure 5-3, we see a 30 bin histogram of the quinto deviations. The plot looks approximately Gaussian, centered slightly to the left of zero. Promisingly, this agrees with our intuitive feeling that the quinto player is, on average, playing slightly in front of the beat.

Could modeling the deviations as a simple independent and identically distributed (i.i.d.) random Gaussian process provide the mechanism we are looking for to model deviations? Probably not. Nevertheless, as discussed in Section 4.3.2, taped example number 5 is a synthesis of such a performance. In that case, the real deviations are substituted with ones generated from a simple Gaussian process with the same mean and variance as the original deviations. When listening to the tape, we perceive it as being obviously wrong. In fact, it sounds terrible. The deviations are no longer related to the current musical event and the performance seems filled with random mistakes. As expected, a simple i.i.d. random process alone can not accurately represent the deviations in a musical data stream.

A possible problem with the above approach is that it ignores per-measure tatum[1] deviations. It assumes the same deviation distribution for all tatums in the performance. Is there a special importance about the i^{th} tatum of a measure? Perhaps. Figures 5-4 and 5-5 show the per-measure tatum deviation histograms.[2] For each value i ranging from $0 \ldots 16$, I have computed a histogram using deviations from the i'th tatum of all measures. The plot labeled "tatum i" shows that histogram. As can be seen, the histograms look somewhat akin to each other; they all look Gaussian with slightly different means and variances.

Could we model the deviations as 16 i.i.d. random processes, the i^{th} one providing a deviation value for a drum stroke landing on the i^{th} per-measure tatum? In Section 4.3.2, taped example number 6 is a synthesis of such a performance. Unfortunately, it sounds just as bad as if not worse than example number 5. Therefore, there must be some contextual dependence: the deviations must somehow depend on the phrase in which they exist. The deviations are not just randomly generated, as some commercial drum machine manufacturers think.[3] Although a sophisticated random process model might have more success

[1] Per-measure tatums were defined in Section 2.2.

[2] These plots also show the tendency for the quinto to play *off beat.* In fact, 37% of the strokes were on beat (tatums 0,2,4,...) and 63% were off beat (tatums 1,3,5,...). Therefore, algorithms based on the minimum syncopation principle defined in [LHL84, Lee85] would incorrectly parse these phrases.

[3] Some drum machines have a *human feel button* which, when activated, slightly perturbs each percussive

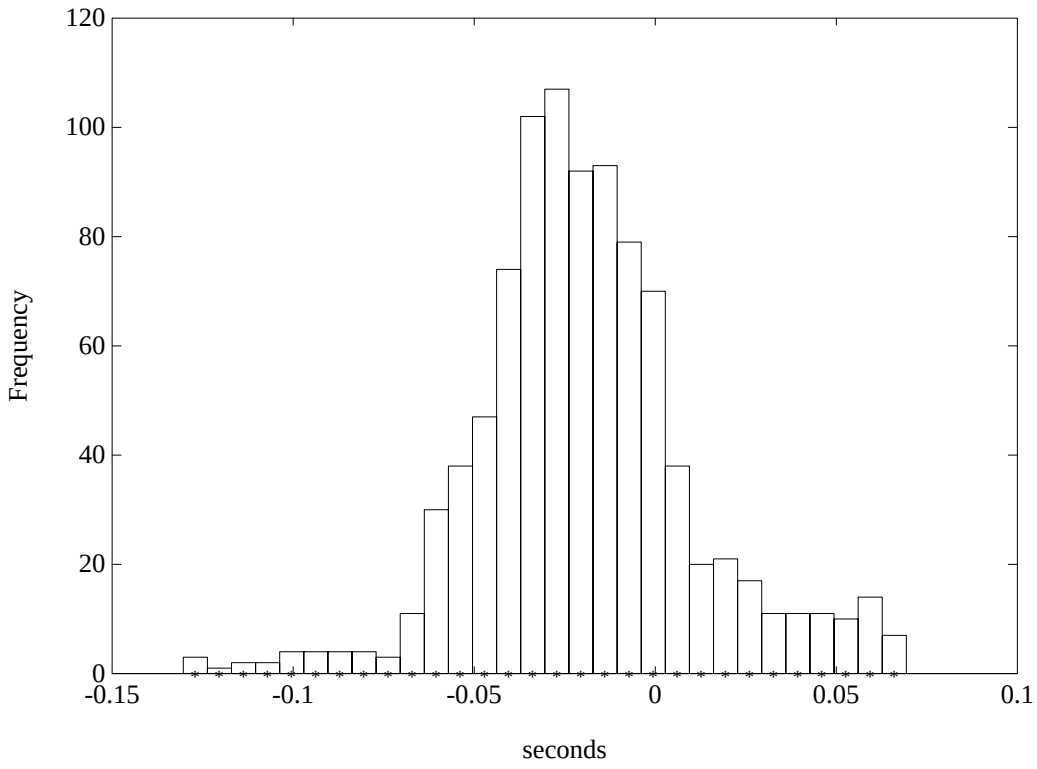


Figure 5-3: General Histogram of Quinto Deviations.

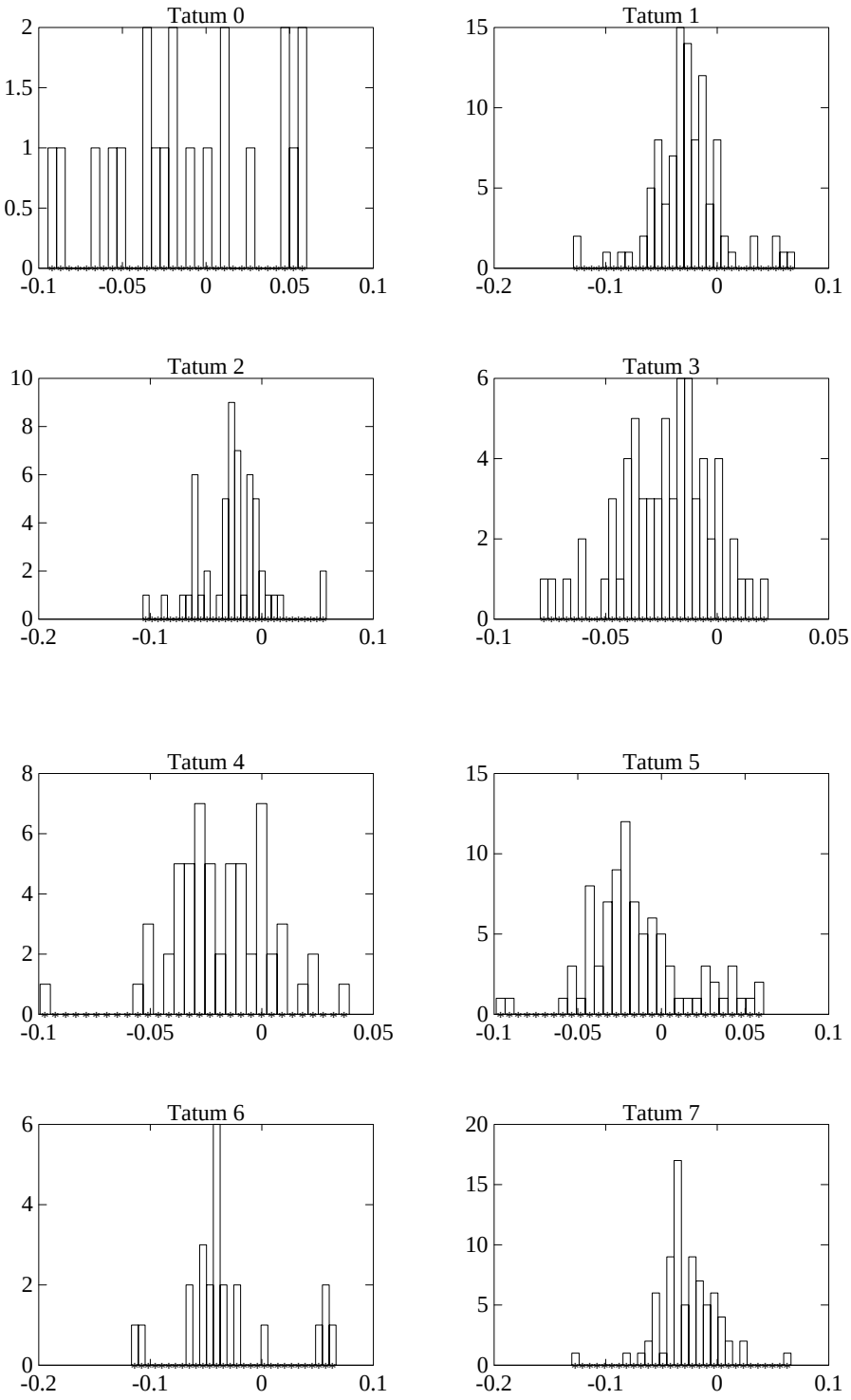


Figure 5-4: Per-Tatum Histogram of Quinto Deviations, tatums 0-7.

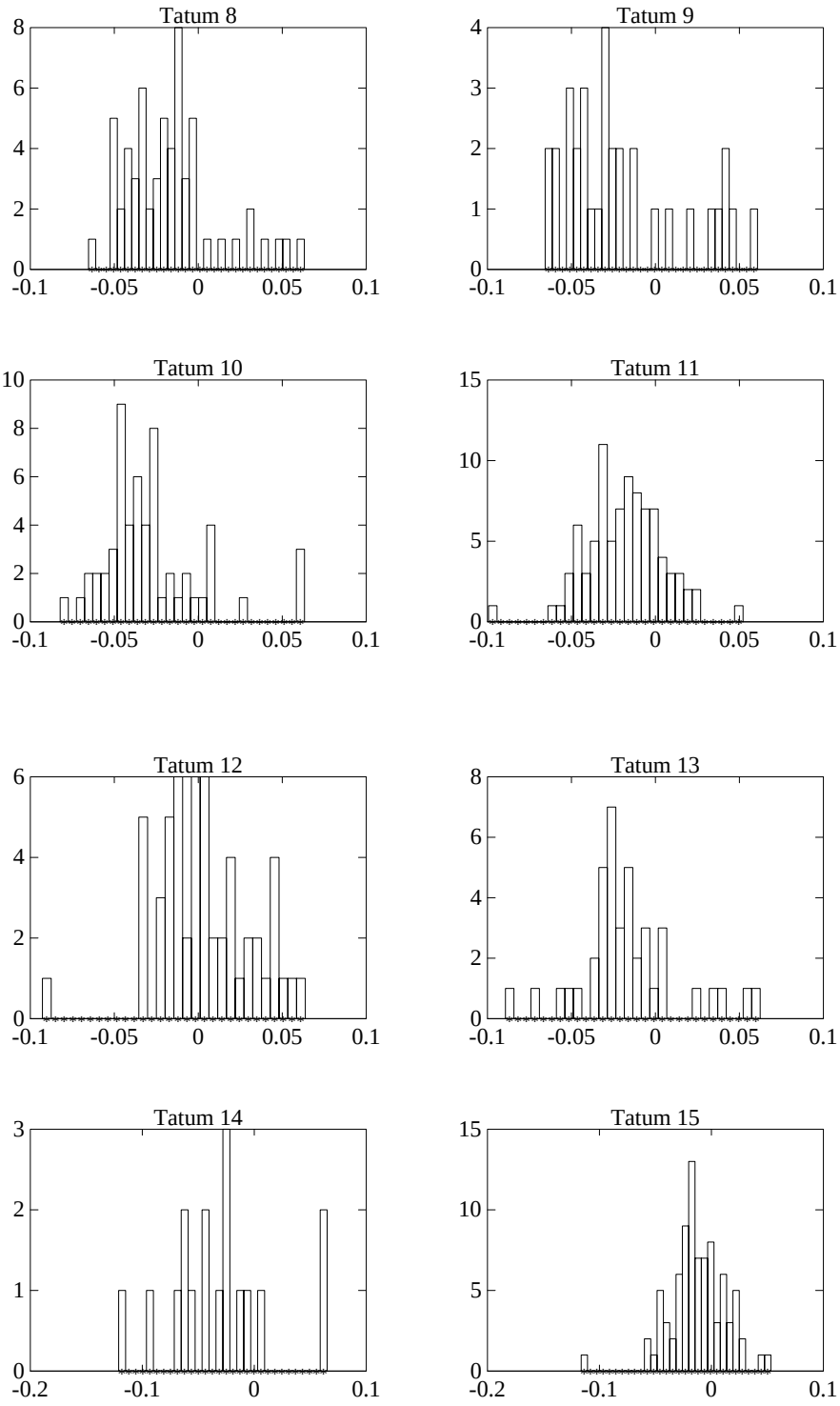


Figure 5-5: Per-Tatum Histogram of Quinto Deviations, tatums 8-15.

in this endeavor, I chose to abandon this approach in search of one that explicitly uses the current context.

5.1.2 Time Series Learning

There are many methods for the learning and analysis of time series. In [Jor89], a recurrent artificial neural network is made to learn, reproduce, and generalize a sequence of actions. Similar work is described in [WZ92, Moz92, SHH88]. In [LO88, Spa88], autoregressive and Bayesian approaches to time sequence learning can be found. Furthermore, many finance theory models have been applied to stock market data with reasonable results.

The deviations and stroke types we have extracted are also time series. The methods above, however, are inadequate for our use because they often have difficulty learning global structure; that is, they are not able to deduce the way in which long-range low frequency information affects short-range high frequency information. In addition, they do not take advantage of potential *a priori* knowledge; they blindly charge ahead trying to learn the series without a pause to consider the generation process. But the main problem is that these approaches learn only a single series. Our problem consists not only of learning both the deviation and stroke type array, but also to understand their inter-relationship. That is, we must understand how the context of the stroke type data influences the deviations. Therefore, I did not pursue the methods discussed above.

5.2 Similar Phrases have Similar Deviations

Figure 5-6 shows two quinto phrases. Phrase A begins on tatum 209 and ends on tatum 216 and phrase B begins on tatum 833 and ends on tatum 840. The phrases are almost the same: they are situated in the same place relative to the measure markers and the stroke types used in them are nearly identical. In fact, these phrases sound practically the same. Note that the deviation patterns are also very similar – phrase B is a bit earlier, but relative to the first tatum, the difference is quite small. That difference is 0.0134, -0.0024, 0.0156, 0.0111, and 0.0105 seconds respectively.[4]

event. The purpose is to make a more expressive sounding phrase. The result is ghastly.

[4]The tempo at phrase A (near tatum 840) is faster than at phrase B (near tatum 210). See Figure 4-2. The deviations are almost all about 10ms smaller in magnitude when the tempo is faster. This suggests that a phrase's deviation is dependent on tempo. Unfortunately, I have deferred this issue for the future. Also see Section 6.1.3.

Fortunately, there are other sets of nearly identical phrases with similar deviations in the quinto performance. Furthermore, in Figure 5-2 we see that the conga performance, primarily a repeated pattern, has matching periodic deviations. The tumbao performance also exhibits this behavior. In general, nearly identical quantized musical phrases seem to have similar corresponding deviations in general. Consequently, we can use the phrase itself to determine an appropriate set of deviations. That is, given a quantized musical phrase and given that we have specific knowledge about the particular musical style, we can determine a set of deviations that can make the phrase sound *expressive*. This fact is extremely relevant to our goal.

5.3 Function Approximation: Learning by Mapping

To produce an expressive rhythmic phrase, we can first find a quantized one and then appropriately deviate each stroke. This sounds quite similar to function approximation.

We assume there exists some mapping

$$f : X \Rightarrow Y,$$

where X is the space of all quantized percussive phrases, and Y is the space of all deviations. Both X and Y are well-defined if we assume that the length in tatums of these percussive phrases is bounded. This is not disconcerting perceptually. Most musical phrases tend not to last more than at most 32 tatums. Our goal is, given a training set of pairs

$$D = \{(x_i, y_i) \in X \times Y\}_{i=1}^{N},$$

where N is the data set size, produce an approximation to f

$$f^* : X \Rightarrow Y.$$

That is, the mapping f^* is an approximation to f based on the training pairs D.

According to Poggio [PG93], producing the mapping f^* will work only if the following three conditions are satisfied:

1. Similar inputs must have similar outputs. This is known as the *smoothness assump-*

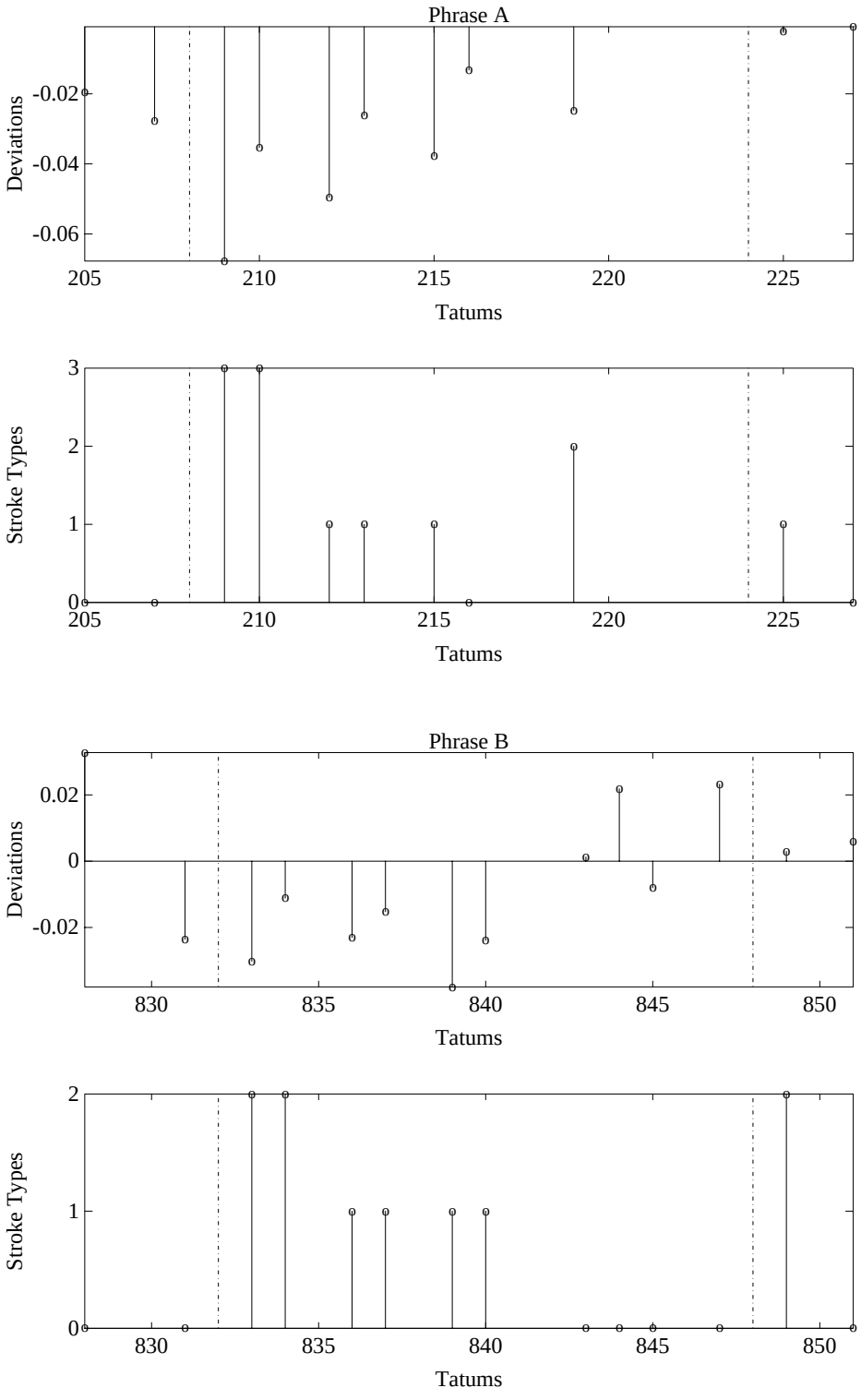


Figure 5-6: Near Identical Quinto Phrases.

tion.

2. The method of producing the approximating map f^* must be powerful enough to resemble the real map f.

3. The data set D must be large. Normally, the number of errors of f^* is inversely dependent to the training set size N.

Condition number 1 is the smoothness assumption. The telephone-number learning example [PG93] clearly illustrates this concept. If the input X is the set of all names and the output Y is the set of all phone numbers listed in a phone directory, then the output space is very jagged. Consider the names "John A. Smith," "John B. Smith," and "John C. Smith." Although the names themselves are very similar (very small steps in the input space), it is extremely *unlikely* that they will have similar phone numbers in the output space (very sharp peaks in the output space). We can say that the mapping is not smooth, or that similar inputs do not have similar outputs. It would therefore be impossible to infer the phone number of "John B. Smith" from knowing the phone numbers of the other two. Section 5.2 demonstrated that similar quantized rhythmic phrases have similar deviations. Therefore, it seems reasonable to assume that we can infer the deviations of a phase knowing only the deviations of other phrases *near* it in the input space.

Condition number 2 requires that the approximating mapping must be powerful enough to represent the real mapping. A powerful approximating strategy, however, always implies a large capability for learning. The results of [VC71] extended in [BEHW89, Hau89] show that the capability (or *VC-dimension*) of a learner is directly correlated with the number of training examples needed to achieve reasonable generalization. It is beyond the scope of this discussion to provide a detailed explanation of this principle (see [HH93] for a nice introduction). Nevertheless, these results obligate condition number 3.

Assuming the three conditions above can be satisfied, an additional potential problem remains. The information in the data might not be sufficient to uniquely reconstruct the mapping in regions at which data is not available. That is, we can never know with certainty what the appropriate mapping is at locations not covered by D. There are holes in the input and output space in which we must somehow interpolate. Furthermore, consider the rhythmic phrase space X. In practice, it is not possible to obtain a data set that completely spans the input space X by observing only one musical style. There are likely to be

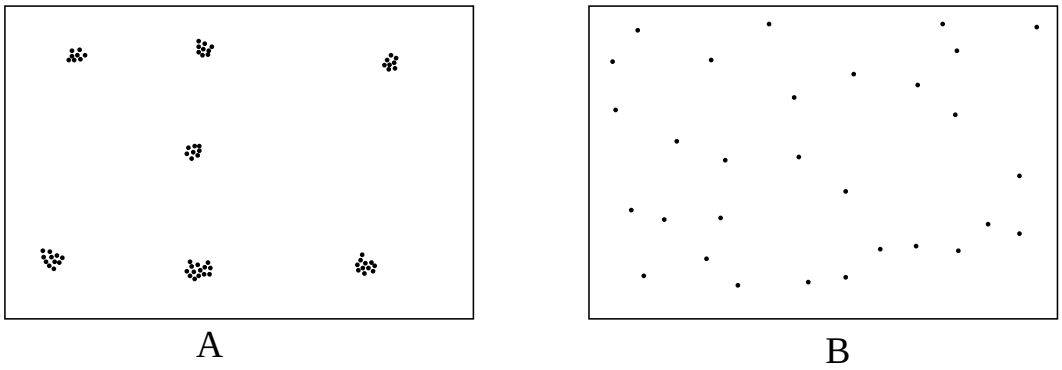


Figure 5-7: Phrase Space.

large regions representable by X that never appear in the musical phrases of a performer of a particular style. Moreover, different musical styles could have different deviations for the same quantized rhythmic phrase, i.e., across musical style boundaries, identical points in X would map to very different deviations. This could even be true of different performers of the same style. Therefore, it seems like we have a problem. If we were to obtain enough data to broadly span the input space X, our data set would contain contradictory $X \times Y$ pairs.

Fortunately, there are ways around this problem. The mapping problem can be thought of as regression, interpolation, splines, etc. The smoothness assumption really means that we have dense data relative to the space, dense enough so that there is no significant variation in the output surface Y between data points. That is, the highest frequency variation in the output space Y should be comparable to the data set spacing. There are two situations in which this condition is achieved:

1. The entire mapping is smooth and we have enough training data to span the entire input space. Therefore, any interpolation between data points will closely approximate any actual point that exists there, and the learner can achieve good generalization (Figure 5-7B).

2. The entire mapping is spiky, and we do not have enough training data to span the entire input space. But, we are interested only in small regions of the input space and we do have enough examples to densely cover those regions. That is, within the small regions, our data set is relatively dense enough so that the mapping (within that region) for all means and purposes becomes smooth. In those regions, an interpolation

between points will not be a gross error. Therefore, it is possible for the learner to achieve good generalization (Figure 5-7A).

We are working only within one musical style and we are not interested in simultaneously obtaining the mapping for phrases not found in that style. Furthermore, we would be satisfied to approximate the deviations of even one expert performer in one style. Therefore, we need only to pick densely packed samples from select regions of the space X. Fortunately, those select regions are exactly the ones that appear in a performance of one style. Therefore, we just need to obtain a large data set from the performer of interest.

When working with the data for the conga (Figure 5-2) and the tumbao, choosing the phrases that constitute the X space is obvious. Each of these drums is repeatedly playing phrases delineated by measure boundaries. The similar phrases are naturally delineated by these measure boundaries. Therefore, the training data may consist of separate measures; x_i is the quantized representation and y_i are the deviations of the i^{th} measure.

For the quinto, choosing the phrases is not obvious. The quinto is improvisational, and does not repeat the same phrase every measure. As we saw above, nearly identical quinto phrases do have similar deviations. We can not, however, simply segment the performance using measure boundaries as with the conga or the tumbao. If we did, we would find that the resulting phrases would not be similar to each other at all. We need a method to determine the similar phrases in the quinto data. The next section presents an algorithm to do just that.

5.4 Similarity Phrase Clustering

The goal of the algorithm is, given a musical quantized score in the form of a sequence of tatum and stroke type pairs, extract the phrases and bundle them into clusters. The phrases in each cluster should be similar. Previous methods of musical pattern detection [MRG85] are not applicable in this context. The approach I take is one of clustering [DH73, And73, JD88]. We want to find the phrases in the piece of music, and simultaneously cluster them into groups of identical or nearly identical sets. Within each cluster, the difference between any two phrases should be minimal.

5.4.1 Clustering Criterion Function

Assume that the vector $\vec{x}$ denotes a phrase from a musical sequence and let $\mathcal{X}_i$ denote one of c clusters of identical or nearly identical phrases. One way of proceeding is to derive a function that judges the quality of a particular clustering; this is called a criterion function. The clustering algorithm can then proceed as an optimization problem; we find the clustering that minimizes the criterion function.

One way of judging a clustering is to use the minimum variance criterion

$$J_e = \sum_{i=1}^{c} \sum_{\vec{x} \in \mathcal{X}_i} \|\vec{x} \Leftrightarrow \vec{m}_i\|^2, \tag{5.1}$$

where the mean vector is

$$\vec{m}_i = \frac{1}{n_i} \sum_{\vec{x} \in \mathcal{X}_i} \vec{x}\,, \quad n_i = |\mathcal{X}_i|.$$

This function is minimized when the intra-cluster phrase distance is as small as possible. There are three problems:

1. How do we segment the musical sequence into phrases?
2. How do we represent a quantized percussive phrase?
3. How do we optimize the criterion function?

5.4.2 Derivation of Phrase Distance $d(\vec{x}, \vec{x}')$

How do we represent variable length percussive phrases with a fixed length vector $\vec{x}$? Furthermore, can we correctly assume that given such a representation, the perceptual distance between two musical phrases corresponds to the Euclidean distance between their two corresponding vectors? Probably not. With a bit of algebra, we may re-write equation 5.1 as follows[DH73]:

$$J_e = \frac{1}{2} \sum_{i=1}^{c} n_i \bar{s}_i,$$

where

$$\bar{s}_i = \frac{1}{{n_i}^2} \sum_{\vec{x} \in \mathcal{X}_i} \sum_{\vec{x}' \in \mathcal{X}_i} \|\vec{x} \Leftrightarrow \vec{x}'\|^2.$$

It is now possible to see that the criterion function is actually an average of the intra-cluster phrase distances. The problem lies, where $\vec{x}$ is mentioned, in the expression $\|\vec{x} \Leftrightarrow \vec{x}'\|^2$. This

quantity is the Euclidean distance between $\vec{x}$ and $\vec{x}'$. If we substitute this distance metric with our own measure as in

$$\bar{s}_i = \frac{1}{{n_i}^2} \sum_{\vec{x} \in \mathcal{X}_i} \sum_{\vec{x}' \in \mathcal{X}_i} d(\vec{x}, \vec{x}'),$$

we have reduced the problem to finding an appropriate distance measure $d(\vec{x}, \vec{x}')$ between percussive rhythmic phrases.

There are several qualities we would like $d(\vec{x}, \vec{x}')$ to possess. First, identical phrases should have a distance of zero. Two quantized percussive phrases are identical only if:

1. Their first strokes have the same relative measure offset.
2. They have the same number of tatums between their bounding strokes.
3. They have the same sequence of stroke types (not counting leading or tailing rests).
4. The inter-onset times between strokes are identical.

Second, nearly identical phrases, phrases slightly violating few of the conditions above, should have very low distance values.

One approach to a distance measure is to discover the set of *features* that constitute all percussive phrases. The features might be contours, patterns of ups and downs, or other shapes. A similarity measure could possibly be found that matches the features of two percussive phrases; the degree of feature commonality would determine the overall similarity [Tve77]. This approach is not suitable, however, for percussive phrases because we do not know *a priori* the appropriate set of features; nor do we want to create a fixed set of features. That would be restricting because a new musical sequence would need its own feature set. Furthermore, this approach is not suitable because percussive phrases come in various lengths, and might consist only of one note on one tatum in one measure. There is no apparent way to obtain features from such a phrase and then compare them to those from one much longer. For the same reason, fixed length vector representations seem unpromising. Therefore, rather than approaching the problem in this way, the distance measure $d(\vec{x}, \vec{x}')$ developed herein compares the two phrases directly without extracting features.

First, the phrases are transformed into a contingency table. Normally, a contingency table is used to represent the co-occurrence between two presumably independent variables

		Var B: Height					
		5'	5.5'	6'	...	12'	Row Totals
	70lb.	n_{11}	n_{12}	n_{13}	...	n_{1q}	$n_{1.}$
	80lb.	n_{21}	n_{22}	n_{23}	...	n_{2q}	$n_{2.}$
Var A:	90lb.	n_{31}	n_{32}	n_{33}	...	n_{3q}	$n_{3.}$
Weight	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...	$\vdots$	$\vdots$
	300lb.	n_{p1}	n_{p2}	n_{p3}	...	n_{pq}	$n_{p.}$
	Col Totals	$n_{.1}$	$n_{.2}$	$n_{.3}$	...	$n_{.q}$	$n_{..}$

Table 5.1: Height versus Weight Contingency Table.

		Var B: Height					
		5'	5.5'	6'	...	12'	Row Totals
	70lb.	f_{11}	f_{12}	f_{13}	...	f_{1q}	$f_{1.}$
	80lb.	f_{21}	f_{22}	f_{23}	...	f_{2q}	$f_{2.}$
Var A:	90lb.	f_{31}	f_{32}	f_{33}	...	f_{3q}	$f_{3.}$
Weight	$\vdots$	$\vdots$	$\vdots$	$\vdots$	...	$\vdots$	$\vdots$
	300lb.	f_{p1}	f_{p2}	f_{p3}	...	f_{pq}	$f_{p.}$
	Col Totals	$f_{.1}$	$f_{.2}$	$f_{.3}$	...	$f_{.q}$	1

Table 5.2: Height versus Weight Relative Frequency Table.

A and B. The table represents $n_{..}$ events. Element n_{ij} in the table is the number of events that fall in both the i^{th} class of variable A and the j^{th} class of variable B. The marginal totals are $n_{i.}$, the number of events that fall in the i^{th} class of variable A, and $n_{.j}$, the number of events that fall in the j^{th} class of variable B. Therefore, we have

$$n_{i.} = \sum_j n_{ij} \quad n_{.j} = \sum_i n_{ij}$$

$$n_{..} = \sum_i n_{i.} = \sum_j n_{.j}$$

For example, class A might be a persons weight, and class B height. An experiment consists of sampling $n_{..}$ events. In Table 5.1, $n_{..}$ peoples' heights and weights are obtained where n_{ij} is the number of people who have weight given in row i and height given in column j.

In contingency table analysis, all entries and the marginal totals are typically divided by $n_{..}$ providing the relative frequency f_{ij} of each event (see Table 5.2). Notice that f_{ij} has

the properties of a joint probability mass function.[5] That is,

$$0 \leq f_{ij} \leq 1,$$

and

$$\sum_i \sum_j f_{ij} = 1.$$

A quantized percussive musical phrase can be represented as a quantized discrete time sequence, discrete in time (the tatum number is the index) and quantized in amplitude (the stroke type). Figure 5-8 shows two musical phrases and their corresponding time sequence representations. These phrases contain only three stroke types; type 0 refers to a rest, type 1 refers to pitch A, and type 2 refers to pitch C.

Each of these rest-padded phrases can be considered a variable in the contingency table and each stroke type in the phrase can be considered a variable class. Furthermore, each tatum can be considered an event. The two phrases must adhere to the same metric form and in particular, they must have the same time signature. The phrases are lined up according to the metric grid, as shown in the figure. The phrase boundaries are determined by the earliest and latest non-rest of both phrases. I assume that a short phrase is equivalent to a long phrase with rest on the ends. So, if one phrase extends past the other, the other is padded with rests. Note in the figure that phrase B has a rest padded onto its beginning, and phrase A has a rest padded onto its end.

A contingency table can thus be constructed that encodes, for each possible pair of stroke types, the number of tatums in which phrase A is one type, and phrase B another. The total number of tatums considered is $n_{..}$. Consequently, the contingency table counts the number of stroke type co-occurrences between two phrases. For example, the contingency table for the phrases given in Figure 5-8 is shown in Table 5.3.

We use $C^{(A,B)}$ to denote the matrix defined by the contingency table for phrases A and B. Notice that, for identical phrases, $C^{(A,B)}$ is zero everywhere except for along the main diagonal.

When measuring the similarity between two phrases, a stroke type difference on one tatum might matter to a greater or lesser degree than a stroke type difference on another

[5] A discrete joint probability density function.

Figure 5-8: Percussive Phrases as Time Sequences.

		Phrase B			
		0	1	2	
	0	0	3	0	3
Phrase A	1	1	1	1	3
	2	1	0	1	2
		2	4	2	8

Table 5.3: Phrase A versus Phrase B Contingency Table.

Figure 5-9: Two Phrases with Tatum Significance Values.

		Phrase B			
		0	1	2	
A	0	0	$2b_3 + b_1$	0	$2b_3 + b_1$
	1	b_2	b_5	b_2	$b_5 + 2b_2$
	2	b_0	0	b_4	$b_4 + b_0$
		$b_2 + b_0$	$b_5 + 2b_3 + b_1$	$b_4 + b_2$	$b_0 + b_1 + 2b_2 + 2b_3 + b_4 + b_5$

Table 5.4: Phrase A versus Phrase B Tatum Dependent Contingency Table.

tatum. Therefore, we attach a significance to each per-measure tatum, and construct a *tatum dependent contingency table*. For each measure, each tatum has a significance b_i associated with it. The significance is a value, between 0 and 1, that indicates the importance of a stroke type difference. That is, the significance b_i indicates the degree to which a co-occurrence of two stroke types on that tatum is valuated in the contingency table (see Figure 5-9). If there are N tatums per measure, then there are N distinct significance values. A value of 1 indicates that a difference on this tatum is maximally significant and 0 indicates that it is not counted. Each time there is a co-occurrence on the k^{th} per-measure tatum of stroke type i in phrase B and type j in phrase A, the value b_k is added into the i^{th} row and j^{th} column of the tatum dependent contingency table. The result for Figure 5-9 is shown in Table 5.4. Let us denote the tatum dependent contingency table for phrases A and B by $P^{(A,B)}$.

As in regular contingency table analysis, we require a relative frequency measure. If we divide each entry by $\sum b_i$, the sum of the $n_{..}$ tatum significance values spanning the length of the phrases, the result is the *tatum dependent relative frequency matrix* for phrases A and B, $R^{(A,B)}$. That is,

$$R_{ij}^{(A,B)} = \frac{\sum\limits_{i \in Q_{ij}} b_i}{\sum b_i} = \frac{P_{ij}^{(A,B)}}{\sum b_i},$$

		Phrase B			
		0	1	2	
A	0	0	$(2b_3 + b_1)/\sum b_i$	0	$(2b_3 + b_1)/\sum b_i$
	1	$b_2/\sum b_i$	$b_5/\sum b_i$	$b_2/\sum b_i$	$(b_5 + 2b_2)/\sum b_i$
	2	$b_0/\sum b_i$	0	$b_4/\sum b_i$	$(b_4 + b_0)/\sum b_i$
		$(b_2 + b_0)/\sum b_i$	$(b_5 + 2b_3 + b_1)/\sum b_i$	$(b_4 + b_2)/\sum b_i$	1

Table 5.5: Phrase A and B Tatum Dependent Relative Frequency Matrix.

where Q_{ij} is a multi-set of tatum significance indices. The indices in Q_{ij} are those of the per-measure tatums upon which a phrase A stroke of type i and a phrase B stroke of type j both land. An example, for the phrases given in Figure 5-8, is shown in Table 5.5 where $\sum b_i = b_0 + b_1 + 2b_2 + 2b_3 + b_4 + b_5$.

The matrix $R^{(A,B)}$still has the properties of a joint probability mass function. This becomes clear if we consider three things:

1. Clearly, no element in $R^{(A,B)}$can be less than zero.

2. The maximum value of an element in $P^{(A,B)}$ is $\sum b_i$. This occurs when phrases A and B are identical and consist of only one stroke type. Because $R^{(A,B)} = P^{(A,B)}/\sum b_i$, the largest possible value of $R^{(A,B)}$ is 1.

3. By definition, we know that $\sum_{i,j} P_{ij}^{(A,B)} = \sum b_i$, therefore $\sum_{i,j} R_{ij}^{(A,B)} = \sum_{i,j} P_{ij}^{(A,B)}/\sum b_i = 1$.

Once again, our goal is to develop a similarity measure between the two phrases. Notice that the sum of the columns in $R^{(A,B)}$ is the tatum-significance scaled distribution of stroke types for phrase A. Similarly, the sum of the rows is the tatum-significance scaled distribution of stroke types for phrase B. One approach, then, is to use a standard measure of association between variables in a contingency matrix. The chi-square statistical measure [JD88, And73] can be used to test the hypothesis of independence between these two distributions. Specifically, we can test the hypothesis $H_0 : r_{ij} = r_{i.}r_{.j}$ where r_{ij} is the actual value of the i^{th} row and j^{th} column of $R^{(A,B)}$, $r_{i.}r_{.j}$ is the expected value, under the independence assumption, of that row and column, and $r_{i.}$ and $r_{.j}$ are the marginal totals of, respectively, that row and column. If H_0 is found to be probable, the distributions are independent and there is low association. If H_0 is very improbable, the distributions are

likely to be dependent and there is high association. Other association methods are based on optimal class prediction [And73]. These measures test the power of one stroke type in phrase A to predict another stroke type in phrase B. If a high degree of predictability can be found, then we say there is high association. Further association methods are based on the correlation coefficient [HT88], and entropy [PTVF92, CT91].

A problem with the above approaches, however, is that they all test association not similarity. Phrase A is *well associated* with phrase B if, for example, all the rests of phrase A perfectly coincide with any one stroke type in phrase B. But, in that case, the phrases would certainly not be similar. Essentially, the measures mentioned above test how well the matrix resembles the solution to the N-rooks problem, where N is the number of stroke types. We, however, need a measure that tests how well the matrix resembles one that reflects the degree of similarity between stroke types. What follows is such a measure.

Let $\mathbf{S}$ denote a similarity matrix where S_{ij} is a value between 0 and 1. A 0 in row i column j indicates that stroke type i and j are completely dissimilar, whereas a 1 indicates that stroke type i and j are completely identical. The matrix $\mathbf{S}$ can be obtained using data from a perceptual experiment on human subjects (see Section 5.4.4 further describing the results of such an experiment). It is found that $\mathbf{S}$ has the following properties:

- It is symmetric.
- It contains values of 1 along the main diagonal (identical stroke types are identical to each other).
- Off the main diagonal, all values are less than one.

The distance measure on percussive musical phrases is defined by using the joint distribution properties of $R^{(A,B)}$ to measure the probability of dissimilarity. We take the opposite of the expected value, with respect to the tatum dependent relative frequency matrix, of the similarity matrix $\mathbf{S}$. This result is scaled by the difference in lengths between the original phrases. That is, for phrases $\vec{x}$ and $\vec{x}'$,

$$d(\vec{x},\vec{x}') = 1 \Leftrightarrow \Psi(\vec{x},\vec{x}')E^{(\vec{x},\vec{x}')}[\mathbf{S}] =$$

$$1 \Leftrightarrow \Psi(\vec{x},\vec{x}')\sum_{i,j} R_{ij}^{(\vec{x},\vec{x}')}S_{ij},$$

where $\Psi(\vec{x}, \vec{x}')$ is a scaling function defined as

$$\Psi(\vec{x}, \vec{x}') = 1 \Leftrightarrow \frac{\left| \|\vec{x}\| \Leftrightarrow \|\vec{x}'\| \right|}{\|\vec{x}\| + \|\vec{x}'\|},$$

$\|\vec{x}\|$ is the length of the phrase $\vec{x}$ *before rest padding*, $\mathbf{S}$ is the similarity matrix, and where $E^{(\vec{x},\vec{x}')}[\mathbf{S}]$ is the expected value of $\mathbf{S}$ with respect to phrases $\vec{x}$ and $\vec{x}'$

$$E^{(\vec{x},\vec{x}')}[\mathbf{S}] = \sum_{i,j} R_{ij}^{(\vec{x},\vec{x}')} S_{ij}.$$

We also will find it useful to define the similarity measure

$$s(\vec{x}, \vec{x}') = 1 \Leftrightarrow d(\vec{x}, \vec{x}') = \Psi(\vec{x}, \vec{x}') E^{(\vec{x},\vec{x}')}[\mathbf{S}].$$

Notice that $E^{(\vec{x},\vec{x}')}[\mathbf{S}]$ is a similarity measure; $\Psi(\vec{x}, \vec{x}')$ is a penalty for having different length phrases; $s(\vec{x}, \vec{x}')$ is a scaled similarity measure; and $d(\vec{x}, \vec{x}')$ is the opposite of the scaled similarity measure.

Assuming that $\vec{x} \neq \vec{x}'$, the distance measure $d(\vec{x}, \vec{x}')$ has the following properties:

1. $\forall \vec{x}, \vec{x}' : 0 \leq d(\vec{x}, \vec{x}') \leq 1$: Clearly, $s(\vec{x}, \vec{x}') \geq 0$ and $0 \leq \Psi(\vec{x}, \vec{x}') \leq 1$. Furthermore, $R^{(\vec{x},\vec{x}')}$ has the properties of a joint probability mass function, and because $0 \leq S_{ij} \leq 1$ the result follows.

2. $\forall \vec{x}, \vec{x}' : d(\vec{x}, \vec{x}') = d(\vec{x}', \vec{x})$: The similarity matrix $\mathbf{S}$ is symmetric, so $E^{(\vec{x},\vec{x}')}[\mathbf{S}] = E^{(\vec{x}',\vec{x})}[\mathbf{S}]$. Clearly $\Psi(\vec{x}, \vec{x}')$ is a symmetric function. The result follows.

3. $\forall \vec{x}, \vec{x}' : \vec{x} = \vec{x}' \Leftrightarrow d(\vec{x}, \vec{x}') = 0$: *Right to Left*: Because $R^{(\vec{x},\vec{x})}$ contains non-zero values only along the main diagonal, $E^{(\vec{x},\vec{x})}[\mathbf{S}] = 1$. Clearly, $\Psi(\vec{x}, \vec{x})$ is always 1. The result follows. *Left to Right*: If the distance is zero, then $s(\vec{x}, \vec{x}')$ must be 1. $\Psi(\vec{x}, \vec{x}')$ is 1 only when $\vec{x}$ and $\vec{x}'$ have the same length. $E^{(\vec{x}',\vec{x})}$ is 1 only when all off-diagonal elements of $R^{(\vec{x},\vec{x}')}$ are zero, which implies that $\vec{x} = \vec{x}'$.

4. Triangle Inequality is False: $\forall \vec{x}, \vec{y}, \vec{z} : d(\vec{x}, \vec{y}) + d(\vec{y}, \vec{z}) \geq d(\vec{y}, \vec{z})$. Consider three two stroke-type phrases. Each phrase is three tatums long, $\vec{x}_1 = [001]$, $\vec{x}_2 = [010]$, and $\vec{x}_3 = [100]$. Assume the similarity matrix $\mathbf{S}$ has values of 0 everywhere except for the diagonal at which it has only values of 1. Then, $d(\vec{x}_1, \vec{x}_2) = d(\vec{x}_2, \vec{x}_3) = 1$, but

$d(\vec{x}_1, \vec{x}_3) = 2/3$.

It would be strange to expect $d(\vec{x}, \vec{x}')$ to satisfy the triangle inequality. If it did, then this distance measure would be a true distance metric and that would seem counterintuitive. I believe, however, that $d(\vec{x}, \vec{x}')$ has a close correspondence to human judgment of disparity between identical and near-identical percussive phrases.

Do the tatum significance values reflect a cultural or stylistic bias? They probably do. In fact, we can compute the significance values using the results of a perceptual experiment. Suppose we ask a set of subjects to rate the dissimilarity, on a scale from 0 to 1, of N pairs of percussive phrases. Suppose further that the i^{th} response is the rating between phrases $\vec{x_i}^A$ and $\vec{x_i}^B$ and that the dissimilarity value is d_i. The entire vector of experimentally derived dissimilarity values obtained from the subjects is denoted by the $N \times 1$ vector $\vec{d}$. Assume there are M tatum significance values we wish to compute and they are represented by the $M \times 1$ vector $\vec{b}$. We want to find the tatum significance values that minimize the difference between the predicted dissimilarity values and those obtained experimentally:

$$\vec{b^*} = \underset{\vec{b}}{\text{argmin}} \left(\vec{d} \Leftrightarrow \vec{d}(\vec{x}^A, \vec{x}^B) \right)^2$$

where $\vec{d}(\vec{x}^A, \vec{x}^B)$ is the $N \times 1$ vector of of predicted dissimilarity values and $\vec{b^*}$ is the vector of optimum tatum significance values. We can perform this minimization by setting the predicted and experimentally derived dissimilarity values equal to each other

$$d_i = 1 \Leftrightarrow \Psi(\vec{x_i}^A, \vec{x_i}^B) E^{(\vec{x_i}^A, \vec{x_i}^B)}[\mathbf{S}] \quad \forall i.$$

This can be represented as a system of N equations with M unknowns

$$(\Psi F)\vec{b} = \mathbf{1} \Leftrightarrow \vec{d}, \tag{5.2}$$

where Ψ is an $N \times N$ diagonal matrix with $\Psi_{ii} = \Psi(\vec{x_i}^A, \vec{x_i}^B)$, and

$$F = \begin{pmatrix} \text{———} F_1 \text{———} \\ \text{———} F_2 \text{———} \\ \vdots \\ \text{———} F_N \text{———} \end{pmatrix},$$

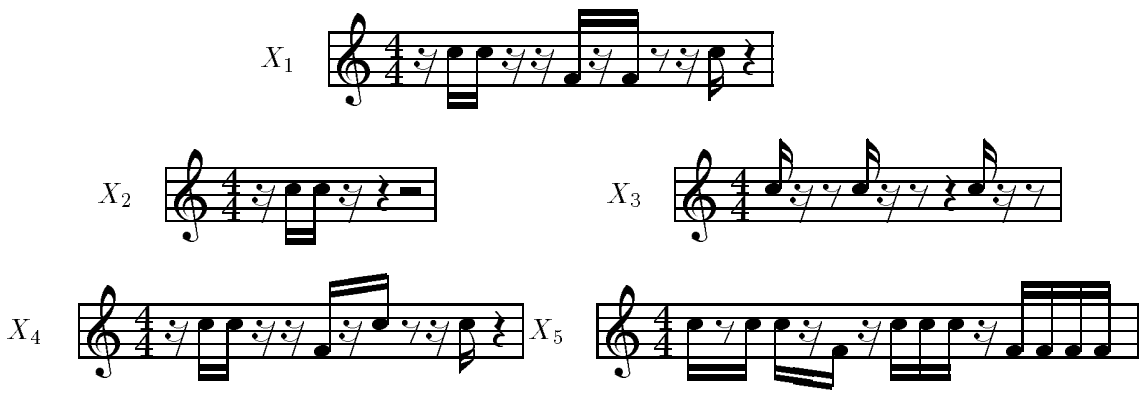


Figure 5-10: Percussive Phrase Examples.

where F_i is a row vector and F_{ij} is the number of tatums with significance b_j existing within the duration of phrases $\vec{x_i}^A$ and $\vec{x_i}^B$. With this definition of F, the predicted dissimilarity values may be given by

$$d(\vec{x_i}^A, \vec{x_i}^B) = 1 \Leftrightarrow \Psi_{ii} F_i \vec{b}.$$

Normally, $M << N$ in Equation 5.2, and $\vec{b}$ can be found by least squares approximation[Str88]

$$\vec{b} = \left((\Psi F)^T (\Psi F)\right)^{-1} (\Psi F)^T (\mathbf{1} \Leftrightarrow \vec{d}).$$

An example follows. Figure 5-10 lists five phrases. Assuming the similarity matrix

$$\mathbf{S} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.5 \\ 0 & 0.5 & 1 \end{pmatrix},$$

and that $\forall i : b_i = 1$, then the values of $d(\vec{x}, \vec{x}')$ for the phrases listed in Figure 5-10 are

$$d(X_1, X_2) = 0.776, \;\; d(X_1, X_3) = 0.647,$$

$$d(X_1, X_4) = 0.045, \;\; d(X_1, X_5) = 1.$$

These distance values seems reasonable, especially considering the contrived similarity matrix and the fact that we have not computed the tatum significance values.

5.4.3 Clustering Algorithm: Divide and Conquer

The next step is the actual clustering. The problem is, find the phrases and clustering that minimizes the expression

$$J_e = \frac{1}{2}\sum_{i=1}^{c}\frac{1}{n_i}\sum_{\vec{x}\in\mathcal{X}_i}\sum_{\vec{x}'\in\mathcal{X}_i}\Big(1 \Leftrightarrow \Psi(\vec{x},\vec{x}')\sum_{i,j} R_{ij}^{(\vec{x},\vec{x}')}S_{ij}\Big). \tag{5.3}$$

Assume that we have N drum strokes. The brute force approach would be to look at all possible ways to form the N strokes into K phrases and all possible ways to cluster the K phrases into c clusters, where K may range from 1 to N and c may range from 1 to K. We would then pick the phrases and clustering with minimum J_e. The number of ways to cluster K phrases into c clusters $S(K,c)$ is very large and given by [JD88, DH73]

$$S(K,c) = \frac{1}{c!}\sum_{i=1}^{c}(\Leftrightarrow 1)^{c-i}\binom{c}{i}(i)^K \approx c^K/c!.$$

And this assumes c is known! The number of ways to form the N strokes into K phrases[6] is $L(N,K) = \binom{N-1}{K-1}$. If K and c are known, the number of ways to form the N strokes into K phrases and cluster the K phrases into clusters is $L(N,K)S(K,c)$. If K is unknown and $c = 1$, the number of ways to form the N strokes into phrases[6] is 2^{N-1}. In our case, however, we do not know either K or c and the number of ways to form the N strokes into phrases and the phrases into clusters is astronomical. Therefore, to avoid a ridiculously intractable algorithm, we must proceed with a heuristic optimization strategy.

Finding the best number of clusters is a fundamental problem in cluster analysis. There are various heuristic clustering strategies given in the literature. One of the most popular techniques is the ISODATA algorithm [BH65, DH73, And73, JD88]. This is an iterative procedure in which clusters are split and merged according to certain guidelines, using K-means along the way. Another method, called hierarchical clustering [DH73, And73, JD88], works bottom-up by first creating one cluster for each sample (phrase in our case) and then merging the clusters that increase J_e the least. There are other methods that test cluster *validity*, i.e., how good is the computed value for c. These methods statistically determine, using Monte Carlo analysis, how unusual (valid) a particular clustering is.[7] In our case,

[6] This is derived in Appendix A.

[7] We also encountered this problem back in Chapter 3.

however, because a sample is a phrase and a phrase is a set of strokes, we need to simultaneously find the phrases and decide how to cluster them, a task that none of the above clustering techniques will perform. We need a multi-stage clustering algorithm, one that simultaneously clusters on two levels. A new algorithm was therefore developed.

The algorithm `Linear Phrase Cluster` divides the sequence of strokes into two halves, solves the problem on each of the two halves, and then merges them back to form the complete solution. It produces a *group* of *clusters*. The final number of clusters is c, as defined in Equation 5.3. Each cluster contains a set of *phrases* that are maximally similar. Each phrase comprises a sequence of stroke types and tatum numbers.

Procedure: `Linear Phrase Cluster`

Input: A sequence of tatum numbers and corresponding stroke types.

Output: A phrase clustering that minimizes J_e.

Step 1: Call `Recurse` with the entire sequence and return the result.

Procedure: `Recurse`

Input: A sequence of tatum numbers and corresponding stroke types.

Output: A phrase clustering that minimizes J_e for the sequence.

Step 1: If the sequence is less than the threshold length, call `baseCluster` and return the resulting group.

Step 2: Split the sequence in half. If possible, the point of division should be situated at a gap between strokes of two or more tatums (thereby, we avoid splitting a phrase in half).

Step 3: Call `Recurse` on the left half.

Step 4: Call `Recurse` on the right half.

Step 5: Call `Merge` on the two halves.

Step 6: Call `Iterative Optimize` on the result.

Step 7: Call `Merge` on the result with itself. This will merge any clusters that either cause a reduction in or only slightly increase J_e (see the definition of `Merge`).

Step 8: Return the result.

Procedure: `baseCluster`

Input: A sequence of tatum numbers and corresponding stroke types.

Output: A phrase clustering obtained by heuristics.

Step 1: Assemble all strokes that are one tatum apart into phrases.

Step 2: Place each phrase into its own cluster.

Step 3: Place all the clusters into one group.

Step 4: Call `Iterative Optimize` on the resulting group.

Step 5: Return the result.

The procedure `baseCluster` uses simple heuristics to obtain its clustering. First, it assumes that the sequence length is small, so there are probably not any similar clusters. Second, it assumes that all adjacent strokes are part of the same phrase (which accounts for step 1). Finally, in case there are similar phrases, it calls `Iterative Optimize` which tries to find them.

Procedure: `Merge`

Input: Two groups to be merged.

Output: A merger of the two input groups.

Step 1: For all unequal pairs of clusters in each group, calculate the difference in J_e between separately representing and merging the two clusters. That is, let $J_e^{(2)}$ be the cost if we keep both clusters and let $J_e^{(1)}$ be the cost if we merge the two clusters. The difference is then $J_e^{(2)} \Leftrightarrow J_e^{(1)}$. Let D_i be the difference for the i^{th} pair of clusters.

Step 2: Sort D_i in increasing order. Create a new empty group.

Step 3: While $D_i \leq 0$, merge the two corresponding clusters and place the result in the new group. Merge some fraction (*mergeFraction*) of the remaining clusters, even if $D_i > 0$. We want to encourage the merging of clusters even if it slightly increases J_e. This step avoids the situation in which J_e is zero but the number of clusters is equal to the number of phrases, clearly an incorrect solution [DH73, page 241].

Step 4: Add any remaining clusters into the new group.

Step 5: Return the new group.

Procedure: `Iterative Optimize`

Input: A group.

Output: A group whose phrases have been manipulated to minimize J_e.

Step 1: Edge Switch: Check the strokes at the edges of each phrase. If switching the edge of a phrase to its neighbor reduces J_e, then do it. The edges of a phrase are defined as the strokes at its left and right ends.

Step 2: Split: Check each phrase. If J_e is reduced by splitting the phrase in half, then do it.

Step 3: Split and Merge: Check each phrase. If J_e is reduced by splitting the phrase in half and merging either the left half with the left neighbor, or the right half with the right neighbor, then do it. Note that the left or right neighbor might be in another cluster.

Step 4: If any changes have taken place and we have not passed a maximum loop count, goto step 1.

Step 5: Return the modified group.

In the procedure `Iterative Optimize`, the edges of a one-tatum phrase are the same. Because we might remove the only edge of a phrase, this step might eliminate phrases.

`Linear Phrase Cluster` would surely be of no use if it was computationally intractable. As we will see, it is not. The following are the costs of each of the procedures:

- Cost of `baseCluster`: because a maximum of N strokes are given to this procedure, and because it is clearly linear in its input size N, this step is $O(N)$.
- Cost of `Merge`: Calculating the difference array is $O(N^2)$; Sorting is $O(N \lg(N))$; Merging is $O(N)$ because a maximum of $N/2$ clusters can be merged. Therefore, this step is $O(N^2)$.
- Cost of `Iterative Optimize`: Each of the optimization stages, Edge Switch, Split, and Split and Merge are clearly $O(N)$. Convergence is not guaranteed however; infinite loops could occur if not for the loop count threshold. But considering the loop threshold as a constant, the cost is a constant times $O(N)$. Therefore, this step is $O(N)$.

The procedure `Recurse` performs four steps: it calls itself with the left and right halves of its input, and calls `Merge` and `Iterative Optimize`. Therefore, the cost is:

$$O(N^2) + O(N) + 2(\text{COSTOF}(\texttt{Recurse}, N/2))$$

This may be expanded into

$$O(N^2) + O(N) + 2(O((N/2)^2) + O(N/2) + 2(O((N/4)^2) + O(N/4) + \ldots$$

$$+O(N'^2) + O(N') + 2(O(N'/2)))\ldots)$$

where $N' = O(N/\lg_2(N))$. This may be reduced to

$$\underbrace{O(N^2) + O(N^2)... + O(N^2)}_{\lg_2(N)},$$

where there are $\lg_2(N)$ terms in the sum. Therefore the complexity of the algorithm is $O(N^2 \lg_2(N))$, far better than the brute force approach. Now the question is, how well does it do?

5.4.4 Clustering Results

I implemented the phrase clustering algorithm and tested it on the quinto data. I obtained the similarity matrix **S** by performing a perceptual experiment on human subjects.[8] The goal of the experiment was to obtain information about perceived similarity between drum strokes. A computer program presented a set of drum stroke pairs to a subject. Each pair was presented using a graphics window that contained eight widgets: two buttons played the drum strokes, five numbered buttons enabled the subject to choose a similarity rating, and a *next* button moved on to the next pair. The pairs were presented in random order. Thirty subjects participated in the study. Each subject produced three similarity matrices, one each for the quinto, the conga, and the tumbao. For each drum, the overall average similarity matrix was calculated, and used as the similarity matrix **S**.

The similarity matrix for the quinto is listed in Appendix B, Figure B-17. Notice

[8] Seth McGinnis, an undergraduate working with me at MIT, was the principle experiment administrator and wrote the computer program to test the subjects. We gratefully acknowledge the support of TVOT for this experiment.

Threshold Length	32 tatums (2 measures)
Similarity Matrix	Perceptual Experiment
Tatum Significance Values	$\forall i : b_i = 1$
Merge Fraction	0.5
Loop Count	25

Figure 5-11: Quinto Phrase Cluster Parameters.

that it is almost symmetric. To ensure that presentation order did not effect the similarity rating, the entire matrix of pairs was presented to the subjects. That is, we wanted to verify that, when comparing strokes A and B, the similarity rating was identical regardless of whether stroke A was listed on the left or right. The asymmetries in the matrix are negligible and may be considered experimental error. Figure B-18 shows the matrix standard deviations over all the subjects. The values are quite small which shows there was wide agreement about the similarities. Notice also that the asymmetries in the mean matrix are almost always within one standard deviation of each other.

The algorithm was run with various parameter values on the quinto data; the values from the final run are listed in Table 5-11. I used the distance measure $d(\vec{x}, \vec{x}')$ under the assumption that all tatums were equally significant, i.e., $\forall i : b_i = 1$. Figure B-1 in Appendix B shows the results of the quinto data for the first 450 tatums. The vertical dotted lines show phrase boundaries and the vertical solid lines show the stroke type. Figures B-2 through B-16 show some of the typical clusters in standard musical notation.

Out of 1717 tatums total (about 5 minutes), the algorithm produced 292 clusters of phrases. Most of the clusters were quite small, and many of them contained only one phrase. Although it did produce a clustering with $J_e = 0$, it was not the best possible. There are probably several reasons for this.

- *Insufficient data.* The data in the quinto performance alone does not predominantly contain similar phrases. If we used a much longer performance, or many performances, the similar phrases would become more apparent because they would constitute denser chunks in the sample space. Therefore, with more data, similar phrases would have a better chance of being clustered together. This would, in turn, cause clusters to attract additional similar phrases because the procedure `Merge` finds mergers that increase J_e the least. This is a fundamental problem for cluster analysis. If the data

does not occupy relatively compact and well separated clouds in the sample space, clustering becomes difficult. Insufficient quantities of data can lead to this sparsity.

- *Inadequate assumption for b_i.* All tatums were valuated equal. We do not hold this to be self-evident, however. I expect that obtaining better values for b_i using the method given in Section 5.4.2 would significantly improve the results.

- *The space defined by J_e is difficult.* The criterion function J_e probably has local minima. Moreover, there are many incorrect global minima. That is, the minimum value of J_e is zero, but any clustering with one phrase per cluster will have that minimum. If the sequence has no similar phrases, such a result might be correct. This is not, however, the normal case. Step number 3 in `Merge` and step 7 in `Recurse` was an attempt to avoid this situation. In step 3, even if J_e was increased, we forced the clusters to merge, producing a cluster with more than one phrase. However, it might be additionally beneficial to bias J_e away from clusters with just one phrase.

- *The distance measure.* The distance measure $d(\vec{x}, \vec{x}')$ tends to get large quickly as phrases become dissimilar. If the measure was slightly more tolerant of dissimilar phrases, additional phrases might be clustered together.

I also applied the algorithm to test data containing multiple copies of only two phrases. Although it successfully grouped identical phrases into clusters, many clusters that contained near identical phrases were not merged together. Therefore, the algorithm shows promise, but more parameter fiddling is clearly necessary.

5.5 Neural Network Learning Strategy

This section describes a Neural Network learning strategy for implementing the mapping function $f : X \rightarrow Y$ discussed in Section 5.3. It was designed specifically for learning deviations from quantized rhythmic phrases. The approach is a common one: a multi-layered perceptron (MLP) trained using the gradient descent procedure[HKP91]. The output representation, however, is not.

The input to the network consists of a set of units for each tatum. Each unit in a set encodes a particular stroke type. Each set corresponds to a tatum; a unit in a set is activated if a stroke occurs on the set's tatum and no unit is activated if there is a rest. The

number of tatums in the input layer is determined by the number of tatums in the maximum length phrase. Because a phrase shifted right or left is completely different perceptually, each phrase is lined up according to its measure boundaries when presented as input to the network.

The final network output needs to be a deviation. As the histograms have shown, this is a small real number somewhere between about -0.25 and +0.25 seconds and has what looks like a Gaussian distribution. Any output of the network must also follow a similar distribution; that is, it must have approximately the same mean, variance, and shape. Because we can pre-compute the mean and variance, we can eliminate this burden from the network. Assuming an approximate Gaussian distribution, the probability of a given deviation is given by

$$p = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(d-\mu)^2}{2\sigma^2}},$$

where μ and σ^2 are the computed mean and variance and d is the deviation. The inverse of this function is given by

$$d = \mu \pm \sqrt{\Leftrightarrow 2\sigma^2 \ln(\sigma p\sqrt{2\pi})}. \tag{5.4}$$

Therefore, corresponding to each deviation d, there is a probability p and a sign $+1$ or $\Leftrightarrow 1$. Rather than learning the deviations directly using one MLP, we use two MLP networks. Network A uses logistic output units and learns the probability of a particular deviation, something that is distributed uniformly between 0 and 1. Network B uses hyperbolic tangent outputs and learns the sign of a deviation. After performing a forward pass through the network, the two values are post-processed through Equation 5.4 providing a deviation. This technique should facilitate deviation learning for two reasons: first, we have essentially doubled the amount of deviation training data; network A receives the same probability for two deviations on opposite sides of the mean. Second, the target values are now uniformly distributed and the information about the mean and variance has been removed which is one less thing for the network to learn.

Chapter 6

The Future

This thesis is only a beginning. There are many avenues left to explore and the following section describes several of them.

6.1 Future Work

6.1.1 Multiple Tatum Clocks

Multiple tatum clocks refer to tatums at different rates. This is not tempo variation, however. The tempo might stay fixed while the tatum clock rate changes. Even within an ensemble, different performers might simultaneously perceive different tatum clock rates. There are two (not necessarily mutually exclusive) musical situations in which multiple tatum clocks can occur.

The first may be called *successive tatum clocks*. Some of the phrases in a piece of music seem better represented using a different tatum rate. For example, some phrases more closely match a metric grid containing of a multiple of three tatums per measure rather than a multiple of four. Fast triplet figures are a perfect example. In the Muñequitos recordings, a phrase occasionally sounded more triplet-like and the $L = 16$ assumption produced a few incorrect quantizations. Consequently, the resulting deviations were also incorrect. This occurred rarely, but it could pose problems for future analysis. We therefore need a method to determine if a tatum rate change has occurred.

Perhaps one method could perform the timing analysis described in Chapter 4 with different values of L, the number of tatums per measure. Regions in the performance at

which deviations are smaller for a given L could be evidence for a temporary tatum rate equal to L. Of course, we could not increase L indefinitely because the deviations would eventually approach zero as the clock resolution becomes finer. We would probably choose reasonable tatum rate values, like 16, 24, or 32 tatums per measure – whatever we believe likely in the music. The appropriate tatum rate would probably have the smallest deviations. This *minimum deviation principle*, however, needs to be verified. That is, why should we believe that the best L produces the minimum deviations? This seems akin to the minimum syncopation principle of [LHL84, Lee85] which is not applicable in many musical situations. Therefore, we need more research in this area.

In addition, percussive rhythmic phrases are often ambiguous. In those cases, even humans might not know the tatum rate. We often do not know how to quantize them and therefore find them most difficult to notate using standard musical scripture. When, for example, do phrases begin to sound in three (twenty four tatums per measure) or four (sixteen tatums per measure). We need a test to determine this threshold. I expect that there is a hysteresis effect, in which we maintain one tatum perception until the deviations become very large in magnitude (past some threshold) and then we switch to the other tatum perception. Therefore, at different times, the exact same phrase might be more naturally perceived using different tatum clock rates.

There is a second multiple tatum situation, in which multiple *concurrent tatum clocks* exist in the perception of a music. African music frequently seems to contain two simultaneous tatum clocks; often the clock rates are both 16 and 24 tatums per cycle. The program described in Appendix E describes an interface that can model this situation. We need, however, much more research into this area.

6.1.2 A Drum Machine Project

Ultimately, the research contained herein might be used in the production of a commercial drum machine. There are several changes that I would advise. First and most importantly, the recorded performance should be ideal. If possible, there should be no bleed between tracks. The bleed significantly complicates the attack detection process and solving the problem of bleed removal is irrelevant to obtaining good timing data. In the ideal recording situation, each drummer would reside in a transparent sound-proof isolation booth. The booths should be transparent so the drummers can all see each other. It should be isolated

so there is no inter-track bleed. When recording trap drums, each drum should be monitored separately using a piezo pickup rather than a microphone. Everything that can be done to eliminate bleed should be done. Attack time analysis will then be much easier.

Second, it would be beneficial to test the accuracy of the performers. They should be asked to play the same phrases several times and the deviations for each phrase should be calculated. Then, the deviation variance for identical phrases will provide rough guidelines about the required accuracy of our representation. Furthermore, a small deviation variance will confirm our assumption that the same phrase at the same tempo is always deviated in the same way.

Third, select the reference instrument (defined in Chapter 4) carefully. The reference instrument should be the most solid instrument in the performance. In the Muñequitos recordings, for example, it might have been beneficial to use a combination of the segundo and tumbao. Whatever is chosen, it should be solid and it should define the tempo.

Other suggestions are perhaps more obvious: make sure the performers are well-rested, are in a good mood, and are performing in as natural a setting as possible. All these things will produce a superior performance and, consequently, superior timing data.

6.1.3 Deviations Dependent On Tempo

In Chapter 5, Figure 5-6 suggests that deviations are dependent on tempo. This is not surprising. It seems clear that a performer will use different deviations when playing a piece at different tempos. Unfortunately, I did not inquire into this matter but I would proceed as follows: obtain several performances of the same phrase at different tempos. The timing analysis algorithm can then provide the deviations and we can see how the deviations change with tempo. Perhaps for a given phrase, the deviation is a fixed percentage of tatum duration. In fact, Appendix E describes a drum machine interface that specifies deviations not in seconds but in percentage of tatum duration. Varying the tempo with that program, however, seems to suggest that deviations get smaller relative to the tatum duration as the tempo increases. This makes sense; there is less room to deviate. However, there is much room here for experimentation.

Figure 6-1: Timing Variation on a Metric Hierarchy.

6.1.4 Metric Hierarchy and Deviations and Tempo

Music can be hierarchically divided into separate components based on its metric structure. One simple example of a metric hierarchy can be obtained from the 32 bar jazz AABA form. The form repeats indefinitely. Each repetition is divided into four sections, each section into eight measures, each measure into four beats, and each beat into three tatums. Each piece of music has its own hierarchical structure. In addition, the arrangement may change over time. A jazz tune, for example, might alternatively move from a structured (e.g., AABA) to a free (only measures and tatums) form. Therefore, the hierarchical structure is time-varying; it changes as the piece transpires. In [LJ83] much discussion is devoted to this topic.

It is possible to define both tempo variation and deviations on domains corresponding to levels in a metric hierarchy. So far, we have seen these functions defined only on the lowest level, the tatum level. In general, a function defined on a particular level operates over a time region equal to the duration of that level (e.g., measure, beat, eight measures, etc.) and is band-limited. That is, a function operating on the highest level might last the entire piece and contain only low-frequency energy. A function operating on a low level might last only a measure and contain only high frequency energy. The low-frequency high-level functions may describe global structure whereas the high-frequency low-level functions may describe local phrase specific variation. Figure 6-1 depicts such a situation. Section A constitutes an entire piece and there is one low-frequency timing variation function (tempo variation or deviations) that lasts for that duration. During the first half of the piece, section A1, a higher frequency timing variation function applies. The second half of the piece, section A2, is similar, and on down the hierarchy.

Mathematically, it is feasible to represent tempo variation or deviations as the sum of all the functions on all the levels of the metric hierarchy. This is what I have been doing so far; each tempo variation or deviation function represents all the tempo variation or deviations in the performance. The point of this hierarchical breakdown is to find simple

functions (sinusoids, second or third degree polynomials, etc.) that accurately describe the variation for a particular level. Figures 4-6 in Chapter 4, for example, comprises two main components which might be simpler functions operating at different levels in the metric hierarchy. I believe that a separation based on the metric hierarchy will enable us to find simple functions describing tempo variation and deviations. We can then combine these simple functions during a resynthesis to produce natural-sounding expressive phrases.

6.1.5 Higher Order Similarity Matrices

In Section 5.4.2, the similarity matrix measured only the similarity between isolated percussive strokes. As was mentioned in Section 3.2.4, different strokes might sound identical when heard out of context. The similarity matrix in Chapter 5 (and listed in Figure B-17), however, tested the similarity of strokes only out of context. Context is important and incorporating it into the distance measure should produce more realistic values. Let's say there are N stroke types. A first-order similarity matrix (Figure B-17) provides a similarity for each of the $N(N-1)/2$ pairs of strokes. A second order similarity matrix, however, provides the similarity of two strokes also considering the strokes that preceded them. The preceding strokes provide a context. Each stroke may be preceded by one of N other strokes, so this matrix contains a value for each of the $N^2(N^2-1)/2$ possibilities. This idea could be extended to even higher order similarity matrices. In all cases, the similarity matrices may be used in the distance measure defined in Chapter 5 as long as the contingency tables also considered the preceding strokes.

6.1.6 Additional Points

- The timing analysis algorithms in Chapter 4 use low-pass filters to remove high-frequency variation from a performance. It is assumed that the deviations constitute this high-frequency variation. We need a method to determine the stop-band cutoff frequency of the low-pass filter so this algorithm can be applied to a variety of different musical forms.

- Ametric musical phrases should be studied and better understood.

- Deviations could prove useful for determining the style or flavor of a piece of music. The style is the type of music, and the flavor is the category within a type of music

such as "excited", "sad", "complex", etc. The same set of quantized phrases could be performed within a variety of styles and flavors and the resulting deviations (extracted using the timing analysis algorithm) could be examined. The deviations might then indicate from which style the music originated and could be used in a flavor oriented drum machine interface such as [Mat93].

- *Flams* are two or more very rapid successive drum strokes. They are much faster than the tatum rate and I believe they are often perceived as one stroke. Nevertheless, the inter-onset times of the flams should be studied. Deviations will appropriately model this situation.

6.2 *Finis*

The main goal of this thesis was to represent and reproduce expressive timing in percussive musical rhythm – that is, to design algorithms that computers can use to produce expressive sounding rhythmic phrases. I believe I have been successful in this endeavor and have begun to quantitatively describe one of the many elusive human behaviors. Music is one of the most important means of expression, and rhythm is probably the most important means of attaining musical expression. We have herein begun to produce expressive sounding rhythmic phrases using a computer. We have yet, however, to achieve the level of Star Trek's *Data*, the level at which computers can combine styles and be creative successfully. And if computers do not soon progress, they might suffer a dreadful fate – they might become what is referred to in the following:

He ain't got rhythm.
So no one's with 'im.
He's the loneliest man in town.[1]

[1] An Irving Berlin tune. Sung by Billie Holiday with Teddy Wilson and his Orchestra, 1937.

Appendix A

Derivation of $L(N, K)$

Herein we derive the formula $L(N, K)$, the number of ways to form N strokes into K phrases. The strokes all live on a line and the phrases must all consist of contiguous sequential line segments.

We can consider the domain as a sequence of numbers on the number line. That is, we have the numbers 1 through N, and we want to find the number of ways to cluster those N numbers into K clusters, where the clusters must consist of a contiguous sequence of numbers.

Clearly, $L(N, K)$ is defined only for $N > 0$ (there must be something to cluster) and $K \leq N$ (we can not ask to cluster N numbers into more than N clusters). Figure A-1 shows a derivation tree for $L(N, K)$. The left-most branch is the case when just the number 1 is contained in one cluster. The remaining problem is then to cluster the remaining numbers,

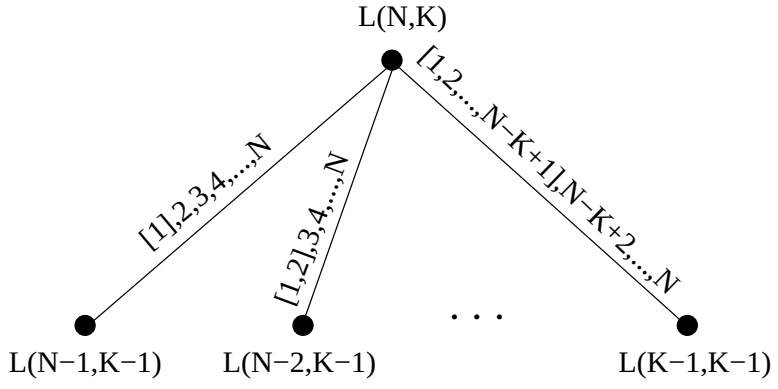


Figure A-1: $L(N, K)$ Derivation Tree.

2 through N, into K-1 clusters. The second branch then shows the case when numbers 1 and 2 are contained in one cluster. The remaining problem then is to cluster the remaining numbers, 3 through N, into K-1 clusters. This proceeds until we ask for the minimum number of numbers (K-1) to be clustered into K-1 clusters. This implies the recurrence relationship

$$L(N,K) = \sum_{i=K-1}^{N-1} L(i,K-1)$$

where $\forall i, L(i,1) = 1$ and $L(K,K) = 1$ are the base values. If we generate this recurrence relationship in a table, we get the following:

		K						
		1	2	3	4	5	6	7
N	1	1						
	2	1	1					
	3	1	2	1				
	4	1	3	3	1			
	5	1	4	6	4	1		
	6	1	5	10	10	5	1	
	7				...			

Therefore, this generates Pascal's triangle. Looking carefully at the table, we see that

$$L(N,K) = \binom{N-1}{K-1}.$$

This simple answer begs for an intuitive derivation of $L(N,K)$. An intuitive one follows: We have N sequential strokes, thus there are N-1 *stroke gaps* between samples. We want K clusters. For a given clustering, there are K-1 *cluster gaps* between clusters, each cluster gap must correspond to some sample gap. The number of ways of choosing K-1 stroke gaps out of a total of N-1 is exactly $\binom{N-1}{K-1}$, and this is identical to the number of ways of clustering the N strokes into K clusters.

The number of ways to cluster N strokes into K clusters where K is unknown follows[Bey87, page 66]:

$$\sum_{i=1}^{N} L(N,i) = \sum_{i=1}^{N} \binom{N-1}{i-1} =$$

$$= \sum_{i=0}^{N-1} \binom{N-1}{i} = 2^{N-1}.$$

Appendix B

Phrase Cluster Results: Quinto Data

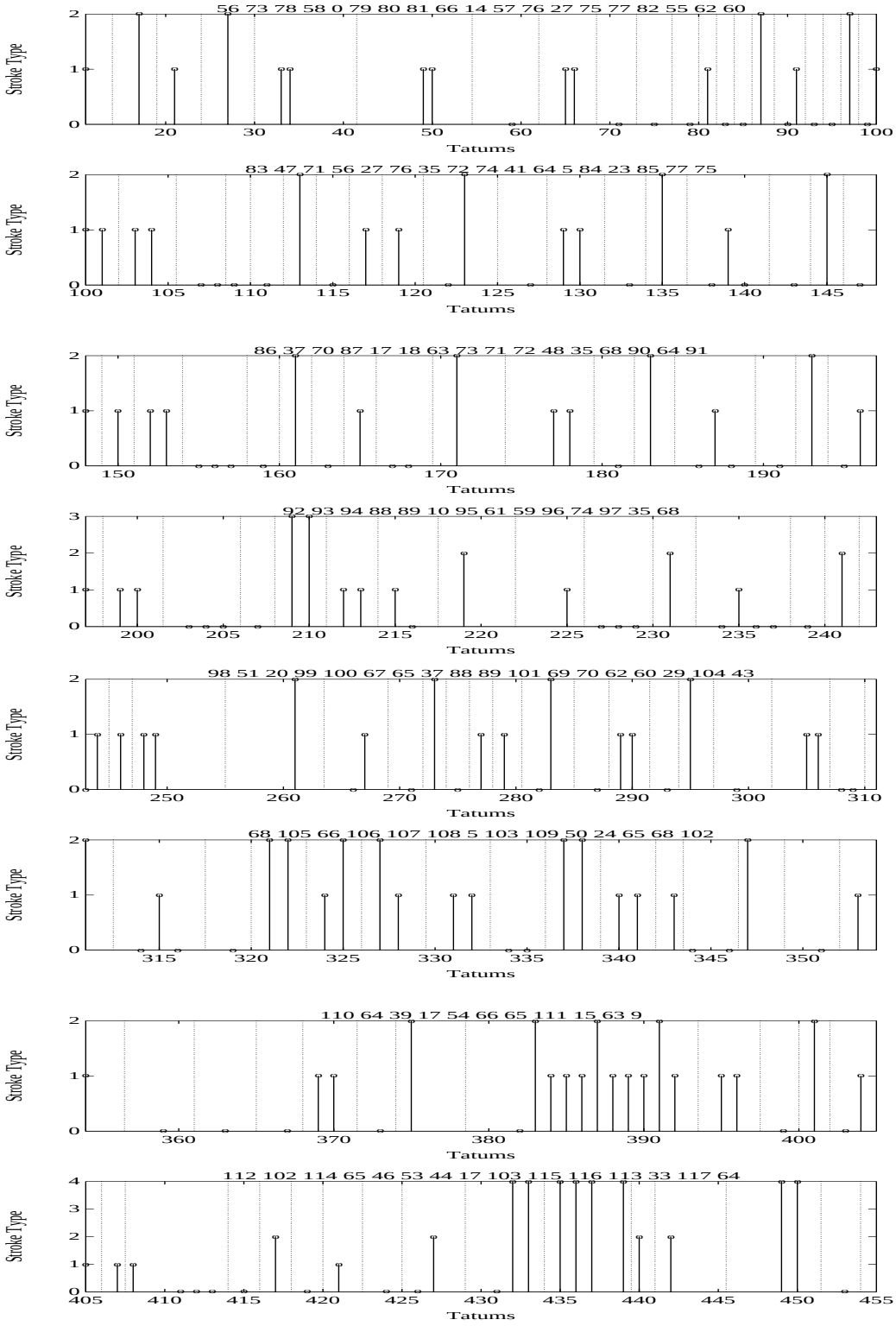


Figure B-1: Quinto Phrase Cluster Results: First 455 Tatums.

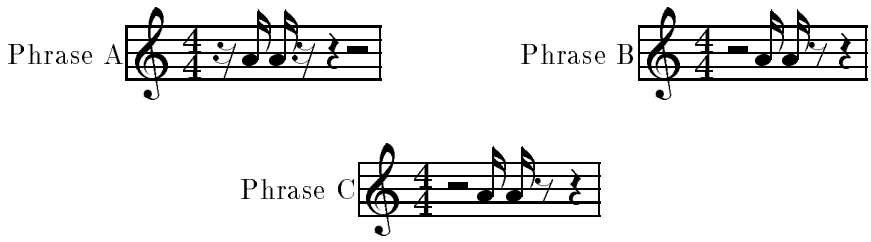


Figure B-2: Cluster Number 0.

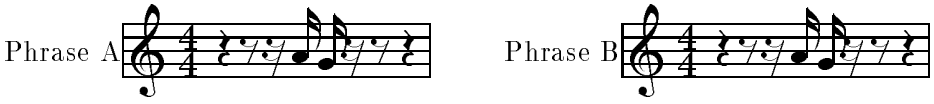


Figure B-3: Cluster Number 1.

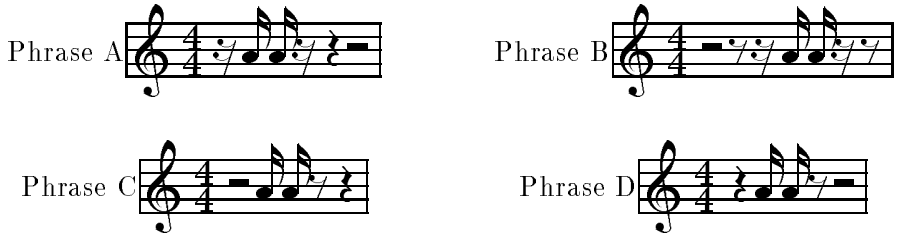


Figure B-4: Cluster Number 5.

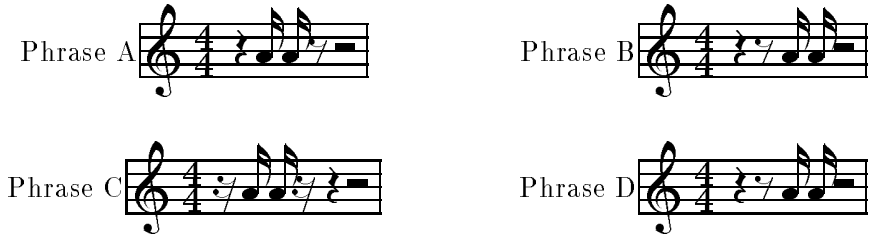


Figure B-5: Cluster Number 10.

Figure B-6: Cluster Number 11.

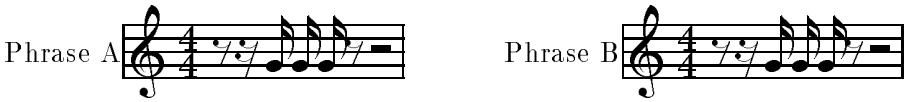


Figure B-7: Cluster Number 13.

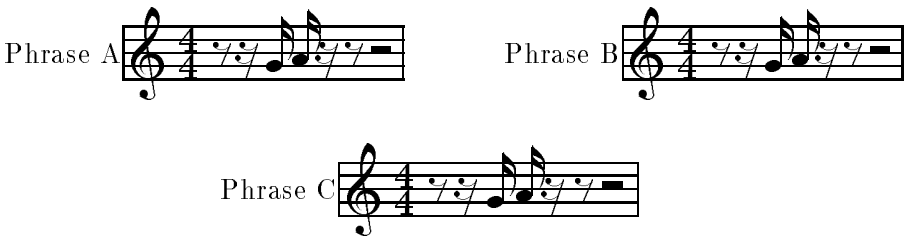


Figure B-8: Cluster Number 19.

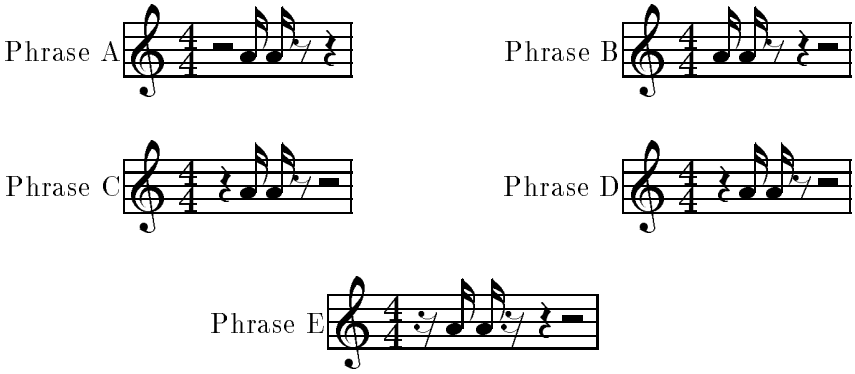


Figure B-9: Cluster Number 20.

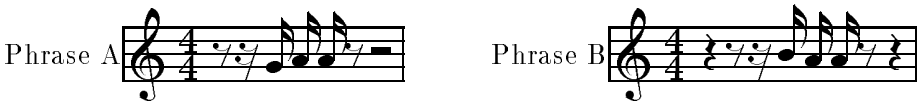


Figure B-10: Cluster Number 38.

Figure B-11: Cluster Number 42.

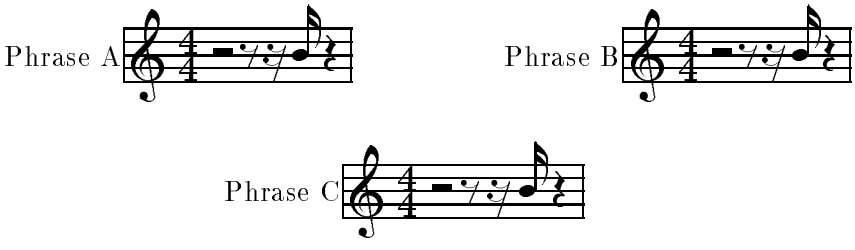


Figure B-12: Cluster Number 58.

Figure B-13: Cluster Number 126.

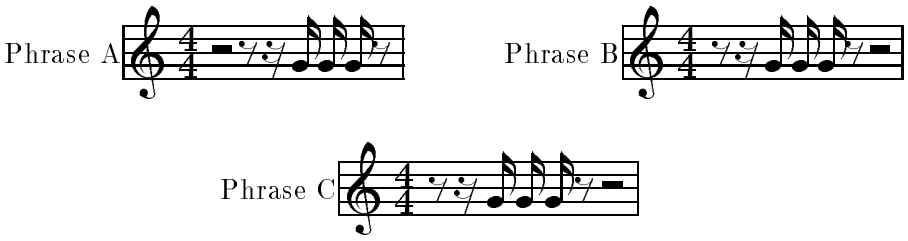


Figure B-14: Cluster Number 184.

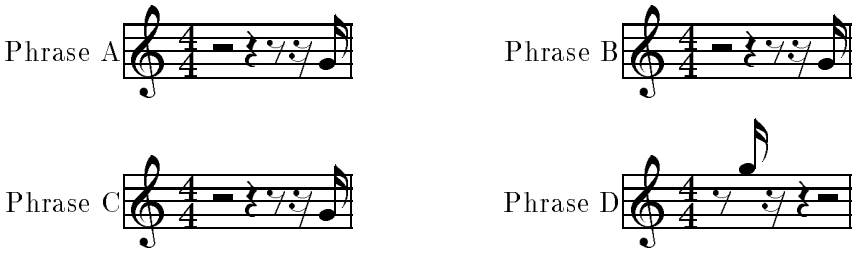


Figure B-15: Cluster Number 187.

Figure B-16: Cluster Number 239.

1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	1.00	0.35	0.54	0.47	0.42	0.36	0.69	0.35
0.00	0.36	1.00	0.31	0.42	0.67	0.52	0.35	0.70
0.00	0.56	0.31	1.00	0.73	0.43	0.49	0.64	0.39
0.00	0.52	0.42	0.72	1.00	0.57	0.63	0.57	0.67
0.00	0.42	0.59	0.40	0.59	1.00	0.81	0.54	0.56
0.00	0.32	0.55	0.45	0.64	0.84	1.00	0.46	0.56
0.00	0.64	0.34	0.63	0.56	0.55	0.48	0.99	0.44
0.00	0.29	0.66	0.40	0.63	0.60	0.52	0.43	1.00

Figure B-17: Quinto Mean Stroke Similarity Matrix. 8 Stroke Types, 1 Rest.

0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.03	0.03	0.04	0.02	0.02	0.02	0.02
0.00	0.03	0.00	0.01	0.02	0.03	0.04	0.02	0.02
0.00	0.03	0.01	0.00	0.02	0.04	0.04	0.03	0.03
0.00	0.03	0.02	0.02	0.00	0.03	0.03	0.03	0.02
0.00	0.02	0.03	0.03	0.03	0.00	0.01	0.03	0.04
0.00	0.03	0.04	0.03	0.02	0.01	0.00	0.03	0.03
0.00	0.03	0.02	0.03	0.03	0.03	0.04	0.00	0.03
0.00	0.02	0.02	0.04	0.02	0.03	0.04	0.04	0.00

Figure B-18: Quinto Drum Stroke Standard Deviations.

Appendix C

Tape Contents

Two sets of musical examples are contained on a tape that accompanies this thesis. Each set contains two sections: 1) a section of short sound segments and 2) a section of long sound segments from which the short segments were extracted. Csound[Ver] was used for all syntheses.

We first hear musical examples from the Los Muñequitos de Matanzas recording. In the first section, each segment is about 30 seconds long:

1. Recording of the segundo attack detection example shown in Figure 3-8.
2. Original two microphone room recording.
3. Off the board, recording.
4. Off the board, recording. Vocals not included.
5. Direct synthesis by triggering samples of the performance at the appropriate time (Section 4.3.2, Example 1).
6. Quantized with the tempo equal to the average (Section 4.3.2, Example 2).
7. Quantized including the original tempo variation in the performance (Section 4.3.2, Example 3).
8. Quantized with the original deviations in the performance and average tempo (Section 4.3.2, Example 4).
9. Quantized with random Gaussian deviations and average tempo (Section 4.3.2, Example 5).

10. Quantized with per-measure tatum random Gaussian deviations and average tempo (Section 4.3.2, Example 6).

The second section consists of the full four minute examples. The descriptions are the same as above.

The third section contains 20 second examples from a different performance. Here, the timing analysis and data extraction methods were applied to a performance given by myself and C.K. Ladzekpo[Lad89], a master drummer from Ghana, Africa.

1. Original Recording.
2. Direct synthesis by triggering samples of the performance.
3. Quantized with the tempo equal to the average.
4. Quantized with the original performance tempo variation.
5. Quantized with the original deviations (average tempo).
6. Quantized with both the original deviations and tempo variation.
7. Quantized with double the deviations in the original performance, original tempo.
8. Quantized with the original performance deviations negated, original tempo.

The fourth section consists of the full four minute examples of the Ladzekpo performance. The descriptions are the same as above.

The taped examples demonstrate two things: 1) The original performance's expressivity is captured by the synthesis. Therefore, when working with a representation controlling only attack times, it is possible to produce expressive music. 2) The deviations, not the tempo variation, is crucial to determining expressivity in these percussive performances. Without the deviations, the syntheses sound lifeless and cold, regardless of the tempo variation. With the deviations, they sound warm and alive. These claims were confirmed by C.K. Ladzekpo himself in an experiment in which he was given no prior knowledge.

Appendix D

Source Code

The postscript form of this thesis, the compressed sound examples, and all the source code is or will be[1] available via anonymous ftp from one of the following locations:

- `media-lab.media.mit.edu:pub/bilmes-thesis`
- `cecelia.media.mit.edu:pub/bilmes-thesis`
- `ftp.icsi.berkeley.edu:pub/bilmes-thesis`

The source code was all written in C++. Therefore, to take advantage of the code, you will need a C++ compiler. I used GNU C++ version 2.4.5, a free compiler available from the Free Software Foundation. Unfortunately, the numerical recipes code[PTVF92] is not included because it would be in violation of copyright restrictions.

[1] Or was once.

Appendix E

xited: A New Drum Machine Interface

Drum machines and music sequencers should start providing advanced facilities for experimenting with deviations. While waiting for this to occur, we[1] have developed a deviation experimentation program called `xited` (pronounced "excited" for eXperimental Interactive Tatum-Editor of Deviations, see Figure E-1). Currently, `xited` runs on SGI IRIS Indigo workstations.

The program consists of a control panel and any number of pattern windows. The control panel controls global tempo in units of *normal-tatums* per minute, starting and stopping, and other miscellany.

The pattern windows determine the score. Each pattern window consists of a rectangular grid of toggle buttons (of any size), an additional row of sliders, and a duration value. A pattern window's grid represents a repeatedly-played percussive phrase. The rows correspond to drum samples or voices and the columns correspond to *pattern-tatums*. If a toggle is set for row i and column j, then voice i will be triggered during pattern-tatum j. Each column also has a corresponding deviation slider. The slider for pattern-tatum j determines, in percentage of pattern-tatum, the amount of time to shift all voices set to play on that pattern-tatum.

A pattern window also contains a duration in units of normal-tatums. Therefore, different patterns, with identical absolute durations, might have different numbers of pattern-

[1] Jeff Foley, an undergraduate working with me at MIT, has been the main implementor of this program.

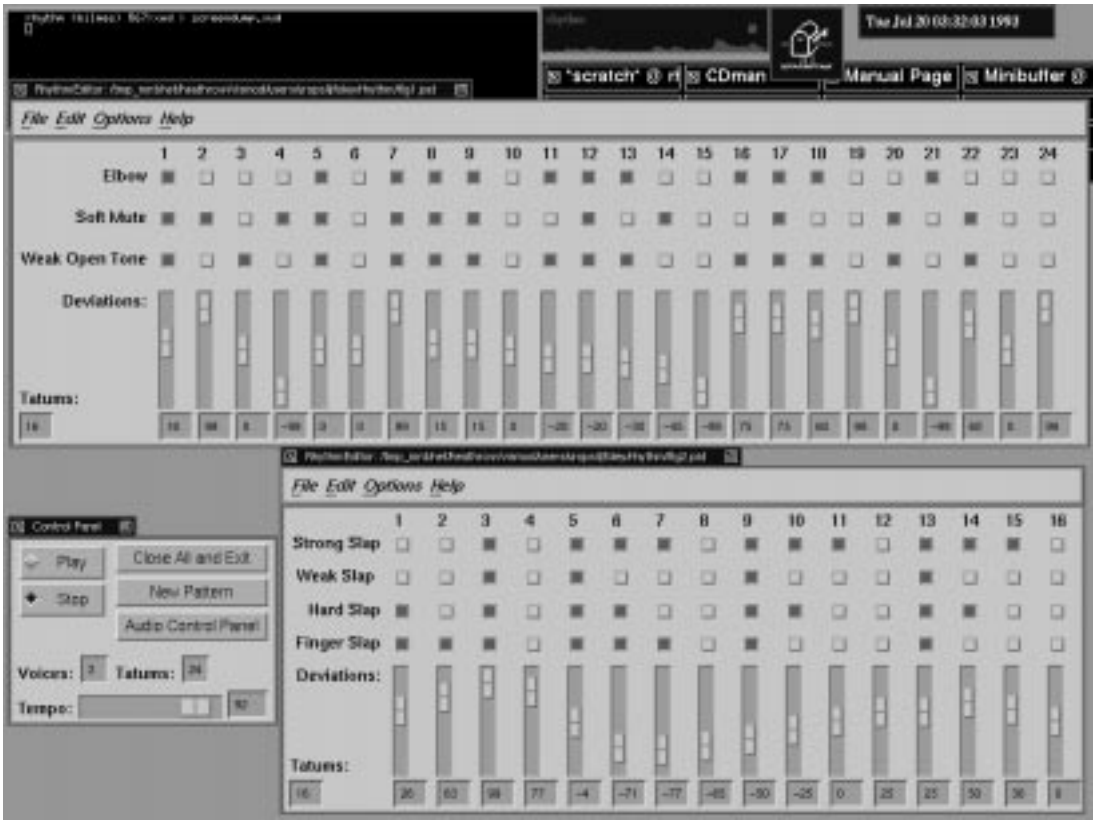


Figure E-1: Graphical Deviation Program `xited` in Action.

tatums. This can be used to express polyrhythms and multi-tatum ethnic music. For example, in Figure E-1, the top pattern has a duration of 16 normal-tatums and contains 24 pattern-tatums. The bottom pattern has a duration of 16 normal-tatums and contains 16 pattern-tatums. This example encodes, in a sense, the feeling of multiple concurrent tatums that is heard in African or Afro-Cuban music.

Each pattern window maintains a counter. When the `<PLAY>` button is pressed, the counters are isochronously incremented modulo their pattern-tatum length. When the counter reaches a particular tatum, any voices scheduled for that tatum are appropriately shifted and triggered. Deviations, toggles, and pattern durations may all be adjusted during playback.

`xited` is thus a novel drum machine user interface. A similar such interface could be used by music sequencers, or eventually, by commercial drum machines. In Chapter 5, an algorithm is defined that creates a mapping between quantized musical patterns and sets of deviations. This algorithm will be eventually incorporated into `xited`. `xited` provides the ability to experiment with deviations and to determine the best sounding deviations for a drum pattern. Indeed, some very interesting rhythmic effects may be attained with `xited` by varying deviations and pattern durations. Yet, they must be heard to be fully appreciated.

Bibliography

[AB84] Mark S. Aldenderfer and Roger K. Blashfield. *Cluster Analysis*. Quantitative Applications in the Social Sciences. Sara Miller McCune, Sage Publications, 1984.

[AB91] David P. Anderson and Jeff A. Bilmes. Concurrent real-time music in C++. In *USENIX C++ Conference Proceedings*, Washington D.C., 1991.

[AB92] David P. Anderson and Jeff A. Bilmes. Mood: A concurrent C++-based music language. In *ICMC Conference Proceedings*, San Jose, CA, 1992.

[ACD87] Phipps Arabie, J. Douglas Carroll, and Wayne S. DeSarbo. *Three-way Scaling and Clustering*. Quantitative Applications in the Social Sciences. Sara Miller McCune, Sage Publications, 1987.

[AK89] David P. Anderson and Ron Kuivila. Continuous abstractions for discrete event languages. *Computer Music Journal 13(3)*, 1989.

[AK91] David P. Anderson and Ron Kuivila. Formula: A programming language for expressive computer music. *IEEE Computer*, 24(7), 1991.

[And73] Michael R. Anderberg. *Cluster Analysis for Applications*. Academic Press, INC., 111 Fifth Avenue, New York, N.Y. 10003, 1973.

[BEHW89] A. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4), 1989.

[Bey87] William H. Beyer, editor. *CRC Standard Mathematical Tables, 28th Edition*. CRC Press, Inc., 1987.

[BH65] G.H. Ball and D.J. Hall. Isodata, a novel method of data analysis and pattern classification. Technical Report 699616, Stanford Research Institute Technical Report, Menlo Park, CA, 1965.

[Bij73] E. J. Bijnen. *Cluster Analysis: Surver and Evaluation of Techniques.* Tilburg University Press, The Netherlands, 1973.

[BP91] Judith C. Brown and Miller S. Puckette. An efficient algorithm for the calculation of a constant Q transform. *J. Acoust. Soc. Am.*, 92(5), November 1991.

[Bre90] Albert S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound.* MIT Press, 1990.

[Bro91] Judith C. Brown. Calculation of a constant Q spectral transform. *J. Acoust. Soc. Am.*, 89, 1991.

[CG91] Gail A. Carpenter and Stephen Grossberg, editors. *Self-Organizing Neural Networks.* The MIT Press, 1991.

[Cli77] Manfred Clines. *Sentics, The Touch of Emotion.* Doubleday Anchor, New York, 1977.

[CRMR+84] John Chowning, Loren Rush, Bernard Mont-Reynaud, Chris Chafe, W. Andrew Schloss, and Julies Smith. Intelligent systems for the analysis of digitized acoustic signals: Final report. Technical Report STAN-M-15, CCRMA: Stanford University, January 1984.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory.* John Wiley & Sons, Inc., 1991.

[DH73] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* John Wiley and Sons, Inc., 1973.

[DH92] Peter Desain and Henkjan Honing. *Music, Mind and Machine: Studies in Computer Music, Music Cognition, and Artificial Intelligence.* Thesis Publishers: Amsterdam, 1992.

[DHJO87] Edwin Diday, Chikio Hayashi, Michel Jambu, and Noboru Ohsumi. *Recent Developments in Clustering and Data Analysis*. Academic Press, Inc., March 1987. Proceedings of the Japanese-French Scientific Seminar.

[DMR87] Roger B. Dannenberg and Bernard Mont-Reynaud. Following an improvisation in real-time. In *Proceedings of the ICMC*, Urbana, Illinois, 1987.

[Ell92a] Daniel Patrick Whittlesey Ellis. A perceptual representation of audio. Master's thesis, MIT, Department of EECS, 1992.

[Ell92b] Daniel Patrick Whittlesey Ellis. Timescale modifications and wavelet representations. In *Proceedings of the ICMC*, San Jose, CA, 1992.

[FERes] Jacob Feldman, David Epstein, and Whitman Richards. Force dynamics of tempo change in music. *Music Perception, forthcoming*, in pres.

[Fra83] Paul Fraisse. Rhythm and tempo. In Diana Deutch, editor, *The Psychology of Music*. Academic Press, 1983.

[Gor84] John William Gordon. *Perception of Attack Transients in Musical Tones*. PhD thesis, CCRMA: Department of Music, Stanford University, May 1984.

[Han89] Stephen Handel. *Listening: An Introduction to the Perception of Auditory Events*. The MIT Press, 1989.

[Hau89] D. Haussler. Generalizing the PAC model: Sample size bounds from metric dimension-based uniform convergence results. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 1989.

[HH93] Don R. Hush and Bill G. Horne. Progress in supervised neural networks. *IEEE Signal Processing Magazine*, 10(1), January 1993.

[HKP91] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation*. Allan M. Wylde, 1991.

[HT88] Robert V. Hogg and Elliot A. Tanis. *Probability and Statistical Inference, 3rd Edition*. Macmillan Publishing Company, 1988.

[Jac89] Leland B. Jackson. *Digital Filters and Signal Processing, Second Edition.* Kluwer Academic Publishers, 1989.

[Jaf85] David Jaffe. Ensemble timing in computer music. *CMJ*, 9(4):38–48, 1985.

[JD88] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, Advanced Reference Series, Englewood Cliffs, New Jersey 07632, 1988.

[Jor89] Michael Jordan. Serial order: A parallel, distributed processing approach. In J. L. Elman and D. E. Rumelhart, editors, *Advances in Connectionist Theory: Speech.* Hillsdale: Erlbaum, 1989.

[KP88] Nelson Yuan-Sheng Kiang and William T. Peake. Physics and physiology of hearing. In *Stevens' Handbook of Experimental Psychology: Second Edition*, chapter 5. John Wiley and Sons, 1988.

[Lad89] C.K. Ladzekpo. Foundation course in African dance drumming. Unpublished Manuscript, U.C. Berkeley, Department of Music, 1989.

[Lee85] Christopher S. Lee. The rhythmic interpretation of simple musical sequences: Towards a perceptual model. In Peter Howell, Ian Cross, and Robert West, editors, *Musical Structure and Cognition.* Academic Press, 1985.

[Lee91] Christopher S. Lee. The perception of metrical structure: Experimental evidence and a model. In Peter Howell, Robert West, and Ian Cross, editors, *Representing Musical Structure.* Academic Press, 1991.

[LHL84] H.C. Longuet-Higgins and C.S. Lee. The rhythmic interpretation of monophonic music. *Music Perception*, 1(4):424–441, Summer 1984.

[LJ83] Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music.* MIT Press, 1983.

[LO88] Jae S. Lim and Alan V. Oppenheim. *Advanced Topics in Signal Processing.* Prentice-Hall, Inc., 1988.

[Mak75] John Makhoul. Linear prediction: A tutorial review. *Proc. IEEE*, 63:561–580, April 1975.

[Mar82] David Marr. *Vision.* W.H. Freeman and Company, San Francisco, CA, 1982.

[Mat93] Fumi Matsumoto. Using simple controls to manipulate complex objects: Application to the drum-boy interactive percussion system. Master's thesis, MIT, Department of Media Arts and Sciences, 1993.

[Mel91] David K. Mellinger. *Event Formation and Separation in Musical Sound.* PhD thesis, CCRMA, Stanford University, 1991.

[Mey56] Leonard B. Meyer. *Emotion and Meaning in Music.* The University of Chicago Press, 1956.

[Min81] Marvin Minsky. Music, mind, and meaning. *Computer Music Journal*, 5(3):28–44, Fall 1981.

[Moo77] James A. Moorer. On the transcription of musical sound by computer. *Computer Music Journal*, pages 32–38, November 1977.

[Moz92] Michael C. Mozer. Induction of multiscale temporal structure. In *Neural Information Processing Systems 4*, 1992.

[MRG85] Bernard Mont-Reynaud and Mark Goldstein. On finding rhythmic patterns in musical lines. In *Proceedings of the ICMC*, pages 391–397, 1985.

[OW89] Alan V. Oppenheim and Ronald W.Schafer. *Discrete-Time Signal Processing.* Prentice-Hall, Inc., 1989.

[Pal88] Caroline Palmer. *Timing in Skilled Music Performance.* PhD thesis, Cornell University, Department of Psychology, 1988.

[PG93] Tomaso Poggio and Federico Gerosi. Learning, approximation, and networks. Chapters 1 and 2 of class notes from a forthcoming book. MIT course 9.520., Spring 1993.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C, 2nd Edition.* Cambridge University Press, 1992.

[Rep90] Bruno H. Repp. Patterns of expressive timing in performances of a beethoven minuet by nineteen famous pianists. *J. Acoust. Soc. Am.*, 88:622–641, 1990.

[Ros92] David F. Rosenthal. *Machine Rhythm: Computer Emulation of Human Rhythm Perception.* PhD thesis, MIT Media Lab, Massachusetts Institute of Technology, 1992.

[Sch85] Andy Schloss. *On The Automatic Transcription of Percussive Music – From Acoustic Signal to High-Level Analysis.* PhD thesis, Stanford University, CCRMA, 1985.

[SHH88] W.S. Stornetta, T. Hogg, and B.A. Huberman. A dynamical approach to temporal pattern processing. In *Neural Information Processing Systems.* American Institute of Physics, 1988.

[Spa88] James C. Spall, editor. *Bayesian Analysis of Time Series and Dynamic Models.* Marcel Dekker, Inc., 1988.

[Str88] Gilbert Strang. *Linear Algebra and its Applications, Third Edition.* Saunders College Publishing, 1988.

[The89] Charles W. Therrien. *Decision Estimation and Classification: An Introduction to Pattern Recognition and Related Topics.* John Wiley and Sons, Inc., 1989.

[Tve77] Amos Tversky. Features of similarity. *Psychological Review*, 84(4):327–352, July 1977.

[VC71] V.N. Vapnik and A.YA. Chervonenkis. Theory of probability and its applications. *Theory of Probability and its applications*, 16(2):264–280, 1971.

[Ver] Barry Vercoe. *Csound: A Manual for the Audio Processing System and Supporting Programs with Tutorials.* MIT Media Laboratory, Cambridge MA.

[Ver84] Barry Vercoe. The synthetic performer in the context of live performance. In *Proceedings of the ICMC*, pages 199–200, IRCAM Paris, France, October 1984.

[WBS87] David Wessel, David Bristow, and Zack Settel. Control of phrasing and articulation in synthesis. In *Proceedings of the ICMC*, Urbana, Illinois, 1987.

[WZ92] Ronald J. Williams and David Zipser. Gradient-based learning algorithms for recurrent networks. In Y. Chauvin and D. E. Rumelhart, editors, *Back-*

propagation: Theory, Architectures and Applications. Hillsdale, NJ: Erlbaum, 1992.