Adapting Personal Music Based on Game Play

by

Samuel Max Rossoff

B.Sc. Northwestern University, Evanston IL 2007

A Thesis Submitted in Partial Fullfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

Adapting Personal Music Based on Game Play

by

Samuel Max Rossoff

B.Sc. Northwestern University, Evanston IL 2007

Supervisory Committee

Bruce Gooch (Department of Computer Science)

Supervisor

George Tzanetakis (Department of Computer Science)

Departmental Member

Amy Gooch (Department of Computer Science)

Departmental Member

Supervisory Committee

Bruce Gooch (Department of Computer Science)

Supervisor

George Tzanetakis (Department of Computer Science)

Departmental Member

Amy Gooch (Department of Computer Science)

Departmental Member

Abstract

Music can positively affect game play and help players to understand underlying patterns in the game, or the effects of their actions on the characters [Smith et al. 2008]. Conversely, inappropriate music can have a negative effect on players [Cassidy and MacDonald 2007]. While game makers recognize the effects of music on game play, solutions that provide users with a choice in personal music have not been forthcoming. I designed and evaluated an algorithm for automatically adapting any music track from a personal library so that is plays at the same rate as the user plays the game. I accomplish this without access to the video game's souce code, allowing deployment with any game and no modifications to the system.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Much of society's perception of video games comes from movies and other visual media. When we talk about music in movies it is as a contributing factor to an aesthetic goal. Rarely do we discuss how influential it is toward interactive experiences. Instead music in video games, much like in movies, is often left to the discretion of artistic narrators. These narrators choose tracks to advance the story line of a game or to contribute emotional depth in the games. Unlike movies, a player takes an active role. Often the musical score is added without examining the effect it has on the active role that the player holds. Most games have sound effects associated with them. These sound effects can influence player action or communicate information back to the player; however, they can also detract from the game play experience. Game makers are overlooking an opportunity to help players both enjoy and understand the games. Tuning the parameters of the audio tract of a game to influence player actions and enhance the game experience provides an exciting possibility for improved game creation.

A popular approach to the application of music in video games is to allow players to choose the music they want to hear during the course of a game from a personal music collection. This allows for a degree of customization to help the players better personalize the game. In terms of game play, this results in a "chicken and egg" type problem; players cannot appropriately select music without having already experienced the game. Additionally, typical players are not experienced composers and often lack a conceptual understanding of what they are trying to achieve. Nor does their personal library have the flexibility needed to adapt music titles to game dynamics.

The lack of forethought regarding the reaction of players to audio information can have tangible consequences. Poorly chosen musical scores, by player or artistic designer, will influence players regardless of intent. There is a need for a system that will automatically adjust these audio tracks so that users can listen to their own music while playing their games.

1.1 Data Analysis

While this work is mainly concerned with addressing the issue of synchronizing music with user input, there is another issue, which underlies this problem. To extract meaningful information one must first deal with the problem of identifying periodicity – that is, what things happen and when. While extracting information about frequency is a well-defined problem, identifying when those frequencies occur is a more complex problem. A number of techniques are discussed here in this thesis in detail. However, no single technique adequately solves this problem for all parts of our implementation. As a result, different techniques are used depending on which is most applicable to the part at hand.

1.2 Objective and Strategy

This thesis reports on an algorithm to adapt automatically soundtracks from personal music libraries to video game play. By examining user input during game play one can compute an optimal rhythm for a platform game. I determine the underlying structure of a sound track by automatically estimating the tempo and dominant periodicity of the music. Finally, I adjust the music to the rhythm of the game level to achieve synesthetic game play. This algorithm requires three separate evaluations: user input must be shown to be predictable; user input must be identified as an approximation of the rate of game play; and input types between different genre of video games must be demonstrably different. This evaluation is

necessary to show that the underlying assumptions of my algorithm are true.

Smith et al. [2008] posit that two dimensional platform games can be broken down into subcomponents or “rhythm groups". This theory could provide for an extremely accurate assessment of the underlying “rhythm" of a level; however, currently, no implementations exist. Additionally, one must have to consider the possibility of proprietary code being used for the game in question. As one cannot necessarily know the underlying structure of the game one is forced instead to look at the data we do have: user input. In this thesis I demonstrate that user input is a close approximation of the underlying structure of the game and can be leveraged for the purpose of synchronizing audio. The accuracy of this claim is evaluated in chapter 6.

The algorithm takes a user’s input and translates it into a meaningful beat sequence, which can be synchronized with a predetermined music track from a personal library with an automatically annotated beat structure. This thesis addresses a number of issues related to the implementation of this system. These can be broken down into three major areas:

I Extracting meaningful user input

II Identifying the tempo of the music

III Synchronizing the user input and music

1.3 Outline

The remainder of this thesis is divided into 5 parts. Chapter 2 gives a survey of background work in the area and related disciplines. Chapter 3 presents a method for obtaining meaningful user input related to experimental data. Chapter 4 lays the groundwork for frequency analysis and the application to identifying “beat" or periodicity. Chapter 5 introduces the problem of synchronizing these two inputs, audio and user, and introduces a time scaling

Figure 1.1: *Diagram of Full System*

algorithm for that purpose. Chapter 6 provides an analysis of our underlying assumptions and, finally, the effects of a fully operating system. A diagram of this system can be found in figure 1.1.

Chapter 2

Background and Previous Work

The issue of incorporating Music into Video games is generally reserved as a “trade secret" and rarely included in published academic work. As a result, there are many techniques for dealing with music in relationship to a game, but they are unreported in the academic literature. While this document reports on an original technique, comparison to earlier work is almost impossible, as said work is *hard* to obtain and incomplete. Despite this lack of information exchange, there is a fair amount of work in the relationship between music and the listener that comes to us from the work of psychologists. Additionally, there is what can be known from the reverse engineering of existing products. Finally, there is some published work on the relationship between sound and other tasks in computer science, which maybe relevant. This chapter explores these three areas in an attempt to compare this work with what has come before. However, given the lack of open information in this area, better comparisons may come to light in the future.

2.1 Correlation Between Mood and Music

A large body of literature exists in Psychology and Biology on the effects music can have on the listener. The work described in this thesis is directly related to user interaction, it is important to establish what effects music can have. The impact music can have may appear to be small to those unfamiliar with the field, but a simple glance at previous work demonstrates just how significant an impact music can have on thoughts and actions of its listeners.

Music has long been shown to affect mood and deal with feelings of monotony and boredom [Karageorghis and Terry 1997]. During exercise periods, the addition of music provides for increased work output, a reduction in the perceived exertion, and overall perception of enjoyment. This was especially prominent in cases where subjects were performing at submaximal levels. The addition of music to exercise routines of subjects lead to a dramatic increase in output, up to optimal levels. In addition, music is shown to enhance affective states at both submaximal and optimal levels. This enhancement in quality of experience as well as performance is essential to the interaction.

Matesic and Cromartie [2002] were able to confirm this relationship as well as to show that listening to music decreases lap pace and increases overall performance of untrained runners. While music had an influence on the performance of both trained and untrained athletes, the effect was much larger on the untrained group. Matestic posits that the music may provide a pacing advantage, as the cause for this effect. Again, the case where users are less than optimal shows the largest improvement in performance. The importance of using music as a pacing tool cannot be overstated.

Cram and Duser [2000], were able to show that not only does music have a marked effect on performance. but music affects heart rate response as well. In their study, users who exercised to music with faster tempos showed higher heart rates on average. While this held true for increasing performance, when participants continued to exercise at a standard rate their speeds still increased and their heart rates decreased. This signifies that the music allowed people to adjust more easily to a proper work out and reduced inefficiencies in their performance. Again, users who had less training showed a larger improvement than those without. Finally, the male participants were more affected than the female ones.

This biological response was not constrained to heart rate either; Thayer [Minami et al. 1998] demonstrated that modifying the musical scores for the same visual response can heavily affect the electro dermal response in subjects. Participants were shown a safety film,

a known stressor, with a documentary score, a horror film score, or no music. Not only did participants show significantly more electro dermal response to the horror score, compared to the control, but less reaction when shown the documentary control. The implication of this study is that music can directly affect the viewer's "mood" in a measurable fashion.

2.2 Relationship Between Tasks and Music

The history of Music and Task performance is a fairly contentious one. While earlier studies demonstrate a complimentary or supportive relationship, later work refutes this in favor of music constituting a "distracter." How a distracter affects the participant depends significantly on the characteristics of that person. While there is a good body of work on users, there is less work on modifying music to assist in a user's exercise. Despite its reputation as a distracter, or perhaps because of it, music has worked its way into task performance with lucrative success. What's more, research on the connection between music and task performance is now forthcoming.

Rauscher et al. [1993] show significant improvement at congitive tests from exposure to Mozart. Rauscher argues that music operates as a neuropsychological primer for children, and that playing music before testing increases their performance. This study is, in many ways, the sparking point for controversy as later work was unable to reproduce this effect. While this study is often discredited, it is still historically important as a jumping off point for the discussion of task performance and its relationship to music.

Konecni [1982] argued that because music requires cognative processing, listening to music might impair performance as the additional cognitive load acts as a distracter. Konecni describes this phenomenon in terms of "cognitive context." It is not sufficient to consider only the role that music plays in cognitive activity, one must also consider what additional cognitive demands are being made. To examine this question, a study was conducted where

participants were asked to choose music to listen to while performing a cognitively demanding task. They repeated this choice every 10 seconds. As processing the musical choice required an additional cognitive load, the task in question was almost universally impaired. The result is that asking users to consider the relationship between their musical choice and the activity at hand decreases said activity.

McKelvie and Low [2002], by contrast, demonstrate that in the presence of music there is little or no effect on spatial IQ scores and reading comprehension. McKelvie and Low set out to unravel the earlier work by Rauscher et al. To do this, participants were exposed to music prior to taking an IQ test. Results of this experiment showed no significant difference with or without the exposure. To further cement their findings, a second study was conducted to replicate the earlier work of Rasucher. This study did not show any of the results of the original experiment. This demonstrates that additional music played before or after has no effect on the activity at hand.

Cassidy and MacDonald [2007] show that high arousal music has a demonstrably negative effect, while low arousal music has a smaller, but still negative, effect. While, overall, music does tend to inhibit performance, the music played in most studies typically is unrelated to the task at hand. The study does not compare the effect of music that has been chosen to augment performance in the task. Additionally, on certain cognitive tasks (e. g. Stroop[1935]) music has a positive effect on performance, suggesting that the right music can be beneficial.

In the field of video games, there have been a number of attempts to adapt game play to musical scores. Some of the more famous adaptations are the *RockBand* and *GuitarHero* franchises. Each “level" in a game like this is generated from an original sound track. This structure of game design has become very popular recently. However, it is limited by the media that it will accept. Instead of being able to supply your own sound tracks, tracks must be created and then distributed to users by the official licenser. To combat this, a number

of grass roots projects, such as Dancing Monkeys, have sprung up [O'Keefe and Haines 2009]. Such projects have developed tools to allow users to select tracks from their music library. Holm et al. [2005] provide an excellent survey of tools of this nature. In Holm's work he explores the user selection of not only audio but visual images as well. Finally, Holm implements such an approach in a mobile phone context [Holm et al. 2006]. This work is tangentially related, in that it is mainly concerned with adapting games to music, instead of music to games.

2.3 Music as an Information Platform

The idea of using audio sounds to convey additional information in Computer Science has been around for some time. Early work attempted to simulate audio icons [Blattner et al. 1989]. Synthetic sounds that are often created to express an underlying quality, as in the previous example, are known as Sonification. A good example of Sonification is Chafe and Leistikow [2001], who created a sonification technique for understanding Internet latency. However, instead of using time repetition to display data, a direct mapping between tones was established. While this can still be benefitial because of the human ear's acuity for pitch, it lacks the precision of many other techniques. Kilander and L'onnqvist [2002] attempted to extend peripheral awareness of users through audio-based techniques in a similar manner to our work. Much of their implementation is concerned with the human interactive component; they made important attempts to streamline incoming sounds through packet shaping. While this packet shaping allowed for the separation of sounds (unlike previous work), it did not do so in an intelligent manner. The sounds became spaced, but spacing was unrelated to user interaction.

One of the major problems with adapting music to an interactive setting is that a lot of music, especially western, is linear in nature. Griffin [1998] implies that traditional solutions

to this problem tend to resemble using segments of music to represent specific events and handling interaction between them. This is highly flawed, as handling the interaction oftentimes ruins the intent. For example, when cutting off a segment that is too long, the result is "un-musical." Likewise, reducing the length of segments (to prevent interaction), has less musical value than truncated longer segments. Griffin suggests a solution to this problem by treating the music as a number of layers, some of which are constant, while others fade in and out based on specific events. While Griffin's solution does produce fairly good music, it still makes many poor choices based on the fact that individual MIDI files have to be both simple and universal in nature to compete with the underlying music. Additionally, it can only run on scenarios where a musical score is composed of multiple separate tracks with a predefined underlying layer. It does not, therefore, handle audio files such as the ones in personal audio music collections.

While some of these systems seek to solve the problem of adaptation, they are more often forgone either because they require additional input by the game author or because they do not provide the necessary quality in keeping with the rest of the game. Our system combats this problem by automatically adapting the track dynamically to give the same quality while keeping with the current game play.

Chapter 3

User Event Extraction

It would simplify the problem if a game-adaptive system for personal music could exist in game code where interaction can be observed directly. As source code for most games is generally unavailable, as such, the goal for this the designed system is to interface with an existing game without access to the source code. There are a number of possible alternatives than direct access, and they can be observed by looking at the input structure of an operating system. While an ideal implementation would be cross platform the input structure often depends on the design of the system in question. Given the target of adapting sound to video games, it would make logical sense to target a platform with high proliferation of said media. Although I could consider modern seventh generation consoles as a viable platform, support for running code in parallel is often non-existent. The resulting choice is the Windows platform, specifically vista, for this implementation. A valid implementation could occur at the hardware level, but I will use a software-based implementation for distribution reasons.

3.1 DirectInput

DirectInput is often the device interface of choice for games on windows because it allows for direct access to the device drivers. This relationship between DirectInput and the game makes it necessarily to suppress mouse and keyboard messages from the OS, and further more ignores all settings made by the user in the Control Panel [MSDN]. Additionally, DirectInput does not recognize keyboard character repeat settings. When using buffered data,

each press and release are signified as single events with no repetition in between. This is ideal when compared to using immediate data, where DirectInput would be concerned only with the physical state and thus produce repeated events. Since I am concerned with periodicity in this case, I am more concerned with *when* the user interacts, not that she is still interacting.

3.1.1 Intercepting Events

As DirectInput is an excellent capture point for the data, the implementation will interface with it. Fortunately, Windows, as of 3.2, implements hooks for identifying keyboard and mouse input [Marsh 1993] . A hook is a mechanism for intercepting events and passing them to a filter function, which can be designed by the user. This filter function can then analyze the data before sending it on. In our implementation this analysis will consist of time stamping and pushing the data to a seperate application via a pipe.

The nature of hooks means that these filter functions must exist in a separate dynamically linked library which is then linked back to the library. The net effect is that the filter function can be called from the DLL for every user event without tedious paging of memory. It is important to note, that on a system with multiple cores all hooks will be called in order for a single message, but multiple messages can have their hooks processed asynchronously. To handle this, events need to keep track of their time stamps but not their inter-arrival time. Inter-arrival time needs to be processed outside of the hook. Finally, to guarantee the accuracy of captured keyboard events passing through DirectInput I must use the low level version of keyboard and mouse hooks (WH_KEYBOARD_LL and not WH_KEYBOARD). The application level WH_KEYBOARD is only called when GetMessage or PeekMessage is called which may be at the next cycle of the game loop. By comparison the low level implementation WH_KEYBOARD_LL is called when events are placed in the queue. As I show in section 3.1.2 our hooks have a sampling frequency of 5ms, which is an order of

magnitude faster than most game loops (i. e.usually 16ms). Though it might be interesting to model our data based on how the game perceives it, since I am synchronizing to user input and not the game such an approach beyond the scope of this research. What's more, it means that any source code implementation can have inaccurate analysis on a single core machine.

3.1.2 Accuracy

For receiving accurate measurements of time between events one can record the clock cycles between filter calls by the hook. One can then adjust this value by the number of cycles per second to Beats-per-Minute (BPM), the common unit of musical rhythm. It is worth noting that a machine with variable cycle rate will lead to erroneous data. To avoid having additional computation time of the hook, clock cycles are observed at the beginning of the hook and then passed out of the filter function. When compared with reads from DirectInput, on an Intel(R) Core(TM)2 Duo E6850 @3.00GHz running Windows Vista Ultimate in 32bit mode, I find that the sampling frequency of interarrival time is 5 milliseconds for uniform input (implemented through the WH_JOURNALPLAYBACK hook). As a result, the additional processing time of the filter must be kept below this value. Any recorded interarrival below this frequency can be treated as simultaneous with the previous event.

Finally, because the filter must run in its own thread, the recording program is abstracted out of the audio processing program. In this implementation these two areas are connected via a pipe as they are part of two different executables. To ensure accuracy of timing, all time processing must happen on the filter side of the pipe. As a result, the user event model receives only values corresponding to interarrival times. All arrivals are marked as either keyboard or mouse events. As future work might involve setting up special values to certain kinds of input the key code and mouse position can also be sent. While the hooks do keep accuracy of characters, they do not solve for meta or shift functions. The result is

that special keys (e. g. $SHIFT$ and ALT) are processed separately. While this is desirable, as they constitute different user interaction, the result is that characters and their shifted versions appear the same.

3.2 Establishing a Model

Once user data has been properly identified, statistical information is gathered about the periodicity of user actions. The rate of incoming input is characterized as a distribution with a mean and standard deviation; this is done for each type of input. As a result, new inter arrival times can be identified as being part of the current model, or unlikely. If a number of incoming interarrival times are classified as unlikely $(P < .05)$, the model is considered "broken," as the player has established a new beat rate. The mean inter-arrival time of this model is then compared to a precomputed beat rate. There are two candidate distributions for examination: Power Law and Exponential.

3.2.1 Exponential

An exponential distribution is a continuous probability distribution where events are dictated by a Poisson process, one where events occur independently of one another. If I model the user interaction as a Poisson random variable the resulting distribution of interarrival times should approximate an exponential distribution. This distribution can be expressed using the CDF:

$$1 - e^{-\lambda x} \tag{3.1}$$

where λ is the parameter of the distrubtion, often called the $rateparameter$.

As the user is interacting with the game in response to events originating from the game, and because games often have events generated from a pseudo-random number generator, it is not unreasonable to expect user events to approximate a Poisson random variable. Based

on experimental values events are characterized as being part of an exponential distribution at rates $>$ 120 BPM.

3.2.2 Power Law

A power law distribution, often times referred to as a Pareto distribution after the Italian mathematician of the same name [Pareto 1972], is a relationship where the frequency of events increases at a rate slower than the number of events having that frequency. This relationship is very common in relationships between events that are often times human driven. We can express such a distribution by its CDF:

$$1 - \frac{x_m}{x}^{\alpha} \tag{3.2}$$

with x_m being theminimum possible value of X and α is the parameter of the distribution

One can also consider such a distribution to be one dictated by both a Poisson process and a second process. While one might expect any non-uniform series of events to be Poisson in nature, the effect of having an overarching pattern of a level represents a second variable which may influence our distribution. Based on experimental values, events are characterized as being part of a Power Law distribution at rates less than $<$ 120 BPM.

3.2.3 Fitting a Curve

To select the correct distribution I will need to use a curve-fitting algorithm. For this purpose a least squares approach will be sufficient. Additionally, I have the secondary problem of parameter estimation to find the most likely curve with which to fit.

3.2.3.1 Maximum Likelihood

In order to determine which curve is correct I need to compare the best candidates of both possible models. I will use Maximum Likelihood to produce said candidates from our observed data. The basis for a maximum likelihood estimation is the probability of a given observed value for a parameter space. I will preform this estimate for all data point thus producing an argument that is most likely given our observed data. Because I have an underlying continuous probability density function, parameter estimation becomes easy.

It is worth mentioning that parameter estimation for Power Law functions require an unbiased estimate. I achieve this, I will use a Maximum Likelihood approach. A simpler aproach like linear regression will lead to a highly bias estimate. For the data I fit a power-law distribution on data $x \geq x_{min}$. Thus our estimator equation becomes

$$\hat{\alpha} = 1 + n(\sum_{i=1}^{n} ln\frac{x_i}{x_{min}})^{-1} \tag{3.3}$$

In this implementation I use Matlab and the work of Aaron Clauset to calculate this [Clauset et al. 2007].

3.2.3.2 Total Least Squares

Least squares is a common algorithm for fitting a set of m observations with a model in n unkown parameters such that ($m > n$). Least squares is usually implemented as identifying the difference between a parametrically defined line and observed data points; this difference is known as the residual. By taking the sum of the residuals I am able to compute the probability for a given set of parameters. By identifying the highest probability point I can get the most likely set of parameters that define our curve as the point where the gradient is zero. In the linear case this is merely two parameters. Since our probability curve is not linear, however the same principal applies.

One can define the model function as $y = f(x, \beta)$, where β here represents a set of parameters $(\beta_1, \beta_2, ..., \beta_n)$. Thus our residual is $r_i = y_i - f(x - i, \beta)$ for $i = 1, 2, ..., m$. Finally, I can express the sum as:

$$S = \sum_{i=1}^{m} r_i^2 \tag{3.4}$$

thus the gradient can be defined as

$$\frac{\delta S}{\delta \beta_j} = 2 \sum_i r_i \frac{\delta r_i}{\delta \beta_j} \tag{3.5}$$

Evaluating where this gradient is equal to 0 results in the correct parameter estimation.

3.3 Experimental Model

The most important step in evaluating the design is showing that there is a structure to the data that is targeted for observation. One of the major design components relies on the ability of the system to identify the current pace of the user. As interarrival time of events varies from action to action, the system must be resilient to these small changes while still being able to identify major ones. It becomes necessary to be able to describe how the user interacts with the system. To this end, an experiment to identify and model this interaction has been designed.

3.3.1 Experimental Parameters

For the purpose of identifying this model subjects were recruited from the graduate and undergraduate student body at the University of Victoria. There were 10 participants between the ages of 19 and 27. Initial tests were done on the video games Starcraft and Super Mario World. Both games were administered on a control PC. The PC in question was an Intel(R) Core(TM)2 Duo E6850 @3.00GHz running Windows Vista Ultimate in 32bit mode. All users indicated prior knowledge of these games as well as having played them

before. Users were asked to play a number of games in the manner to which they were accustomed. The system was put in place to record user input but not supply any audio responses. Both keystrokes and mouse clicks were recorded along with the time elapsed since the previous event. The source of the data (mouse vs keyboard) was recorded for further analysis. The interarrival time between events was then calculated and graphed.

3.3.2 Methodology

Because merely taking a histogram of our data presents the problem of which window to use, finding a function to fit this data is instead done through a Cumulative Distribution Function (CDF). Graphing the user data in such a manner shows two obvious trends: first, that data below .16 seconds is linear; and second, that data beyond that point follows an exponential or Pareto distribution. The reasons for the former seems obvious as user interaction below the $\frac{1}{6}$th of a second level becomes a uniform distribution as it is faster than the intelligent response time of the user. This characterizes events such as double clicks. While further examination on reaction time of users may provide more interesting results, such speculation is beyond the scope of this paper. Since such events are uniform in nature they have no influence on the model in question. If I instead examine the range from $\frac{1}{6}$th of a second to ∞, I can fit a function to this equation. Using Mathmatica I can use a least squares approach to fit exponential and Pareto CDFs, derived in MATLAB, to our data (figure 3.1 and 3.2).

3.3.3 Fitting Our Data to a Model

Neither equation provides a very good fit to the data. Much of the instability is an attempt to match the shorter duration interarrivals. If, one looks only at events greater than .5 seconds the Pareto Distribution provides an excellent fit to the data, while the Exponential Distribution continues to provide a less than ideal fit. It is important to note that events less

than .5 seconds constitute a significant portion of all events (45%). The most likely cause of this instability is a similar source as that of shorter events. While events below $\frac{1}{6}$th of a second are completely uniform, those from $\frac{1}{6}$th to .5 of a second are a combination of uniform and the later model. As the later model provides a high degree of accuracy past this point it must the underlying structure.

3.4 Implementing Our Model

Given a possible model for icharacterizing the user input, it becomes possible to realize an implementation utilizing the BPM of the user as input into the time scaling modification. As my system has to output audio I can implement our model between $tick()$ calls that output the next frame of audio to the user. Between these successive calls, input is read in from the pipe. This input contains only time stamps due to the fact that successive DirectInput calls are not necessarily processed by our hook in order. As such, the interarrival time needs to then be calculated before it can be inserted into our model. The nature of this model is discussed in section 3.3.

The model of interarrival time indicates the probability that a given sample is likely or not. As it is possible that a single event might occur outside of the model, my implementation requires 3 successive events with in a span of several seconds. Future work is required for identifying a more accurate number of successive events. After the probability of the event is calculated it can then be inserted into the model as it is representative of the user interaction. It is important to note that events are inserted regardless of their probability to establish a correct model.

Finally, the result of this interaction gives a likely approximation of the users rate that can then be compared to the beat rate of the music in question (as detailed in chapter 4). From this a ratio of audio to user can be determined. This value is then combined with

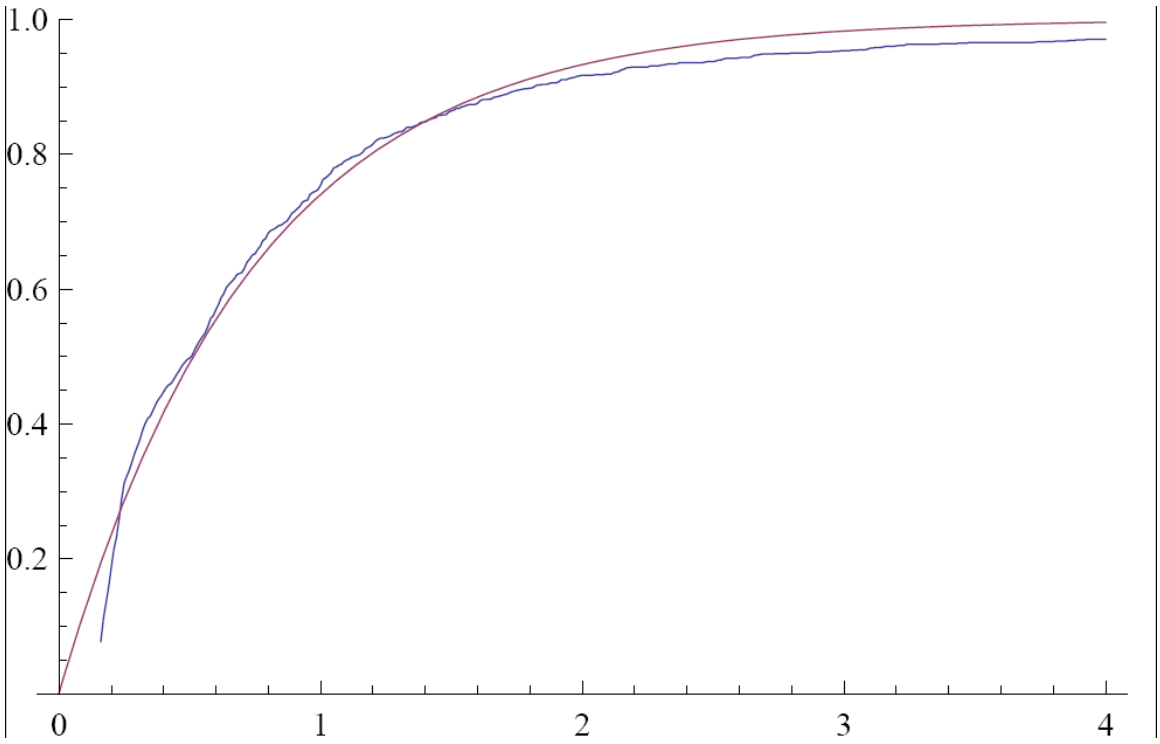

Figure 3.1: *Plot of Estimated Expontential Distribution against Real Data (% of total events vs. thier interarrival time in seconds)*

the sampling rate of the music and inserted into the time stretching algorithm detailed in chapter 5.

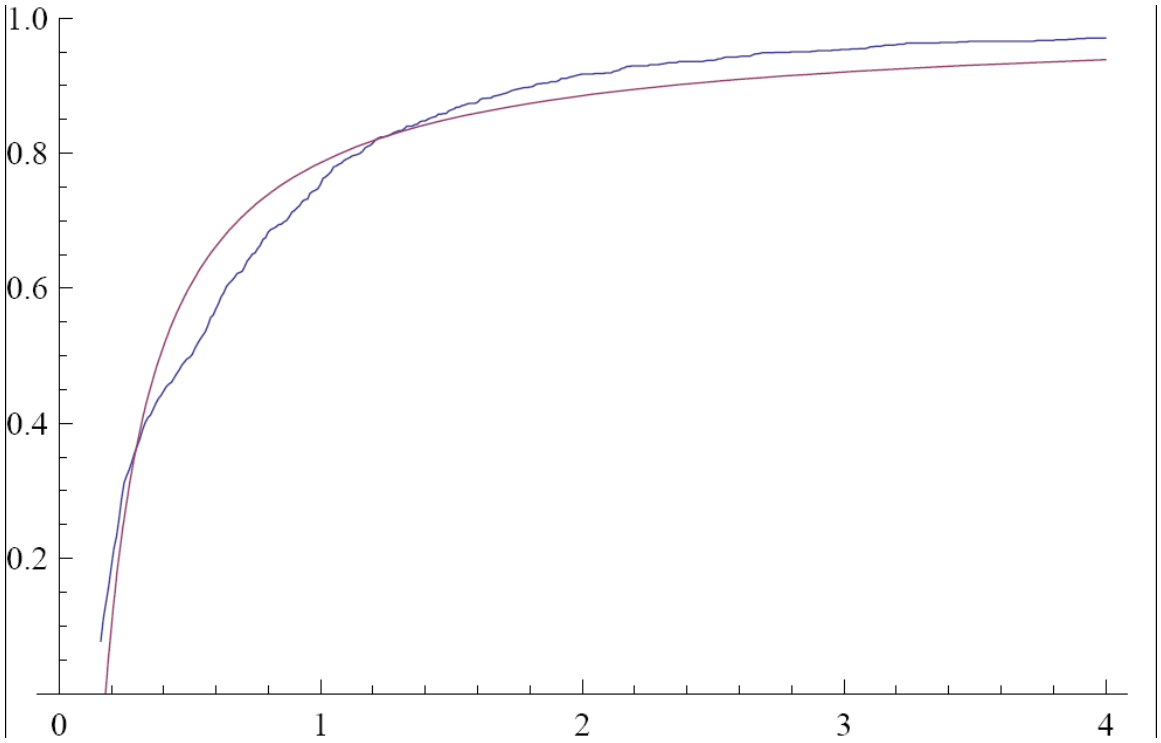

Figure 3.2: *Plot of Estimated Power-law Distribution against Real Data (% of total events vs. thier interarrival time in seconds)*

Chapter 4

Beat Extraction

Identifying tempo and beat structure is a well-defined problem in the field of music information retrieval. Earliest studies involved beat extraction by using subjects tapping or clapping in time [Drake et al. 2000]. Tzanetakis et al. [2002], compare two methods of beat extraction based on beat histograms. These histograms are assembled by identifying the amplitude envelope periodicities of multiple frequency bands. This is accomplished with a standard Discrete Wavelet Transform filter bank and a multiple channel envelope extraction. While this technique does give an accurate picture, some level of information is lost due to imprecision on the part of the performer. The algorithm proposed in this thesis expands on this technique by utilizing graduated non-convexity[Blake and Zisserman 1987].

4.1 Multiresolution Analysis

To preform beat extraction it is necessary to identify which sounds occur at which times. As sounds can be characterized by their frequency it becomes necessary to use frequency analysis to identify the sounds that are contained by the signal. One can consider a signal as existing in the **Time-Domain** where the independent variable is time and the dependent variable is the amplitude at that time. To obtain which sounds are occurring a frequency transform to preform analysis on the frequency must be used. This frequency transform will move us from the **Time-Domain** to the **Frequency-Domain**. Additionally, to identify the periodicity or beats of a given sound I need to not only know which sounds are occurring,

but when they are occurring temporally. As a result, the frequency analysis will have to occur at multiple resolutions. This multiresolution analysis will allow us to identify when frequencies occur, which I can then be corrilated to obtain the periodicity.

4.1.1 Fourier Analysis

Fourier analysis is a technique by which one can identify frequencies given a signal. This analysis can be seen as a shift from **Time-Domain** to the **Frequency-Domain** which is necessary for our signal analysis. To preform this analysis I preform a Fourier Transform. Any measurable function f on the interval $(0, 2\pi)$ can be expressed as having a Fourier series representation:

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx} \tag{4.1}$$

where the constants c_n are the Fourier coefficients of f, which can be formally defined as:

$$c_n = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-inx} dx \tag{4.2}$$

It's worth noting that the Fourier series is a sum of sines and cosines, however they can be related using Euler's formula:

$$e^{2\pi i\theta} = cos(2\pi\theta) + i * sin(2\pi\theta) \tag{4.3}$$

The result of this transfer from sines and cosines to a complex exponent is the reason for the coefficient c_n is needed to preserve amplitudes.

The more compact the function $f(x)$ is the more spread out the associated transform must be. It is not possible to arbitrarily concentrate both a function and its transform. As a result the Fourier transform can only give us information as to which frequencies are present and not when they occur. This Fourier Analysis is therefore only accurate for constant frequency over time. When presented with a signal where the frequency shifts over time, the Fourier analysis will identify all the frequencies as being present.

4.1.2 Short Term Fourier Transform

For beat tracking it is necessary to obtain the temporal information that the Fourier Transform lacks. Although the Fourier Transform doesn't give us time information, we can consider dividing our signal into smaller segments and identifying which frequencies are present in these shorter time slices. To preform this analysis a Short Term Fourier Transform (STFT) can be used. Towards this end it is necessary to create a windowing function ω where the width of ω is equal to the window under which the frequency is constant. If I want to consider the first s seconds of our signal, I can locate the window function at $s = \frac{s}{2}$ and premultiply our signal with the window function resulting in on the first s seconds of the signal being chosen. I can then take the Fourier Transform of this product to detect which frequencies occur in this time slice. I can also sift our window along our signal by some time value t. These changes can be summarized as follows:

$$S(s,x) = \int_t f(s)\omega^*(s-t)e^{-ixs}ds \tag{4.4}$$

Where ω^* is the complex conjugate of our windowing function.

The resulting function characterizes our signal with respect to two parameters, time and frequency, however there is a necessary trade off between the two. By using a narrow window I am able to achieve good time resolution, but receive poor frequency resolution. However, if I increase my window size, I can violate my stationarity condition for frequencies and thus lose time resolution. While my application does not require perfect resolution in both domains, a given window should not have the same resolution on all frequencies.

4.2 Wavelet Transforms

The human audio perceptual system does not have excellent resolution for both frequency and time either. Instead it has good time resolution for high frequencies, and good frequency resolution for low frequencies. This implied trade off would require multiple windows in a STFT-based approach. Instead, our application utilizes a Wavelet Transform which encompasses these characteristics. Just as the Fourier Transform is a STFT with a window, so is the STFT a Wavelet transform with a constant window size. Thus one can define the STFT as a special case of a Wavelet transform. The difference between a Wavelet transform and a STFT is that a generalized wavelet can have a windowing function of non-constant size. Instead, a wavelet can be viewed in terms of a scaling function ϕ and a mother wavelet ψ, where the resolution is obtained by updating the scaling function and then scaling the wavelet by the scaling function. The function can be formally defined as:

$$f(x) = \sum_{j,k=-\infty}^{\infty} c_k^j \psi_k^j(x) \tag{4.5}$$

f can be expressed again as a series with analogous coefficients given by:

$$c_k^j = \langle f, \psi_k^j \rangle \tag{4.6}$$

Similarly the integral transformation W_ψ by

$$(W_\psi f)(t,s) := |s|^{-\frac{1}{2}} \int_{-\infty}^{\infty} f(x)\psi(\frac{x-t}{s})dx \tag{4.7}$$

Which changes the coeffecient to

$$c_k^j = (W_\psi f)(\frac{k}{2^j}, \frac{1}{2^j}) \tag{4.8}$$

Where W_ψ can be seen as the "integral wavelet transform" of the wavelet ψ at resolution j who's scale and translate terms are s and t respectively. Note, that here s is a binary dilation (i.e. $s = 2^j$) and t is the dyadic position $b = \frac{k}{2^j}$ [Chui 1992]

4.2.1 Discrete Wavelet Transform

As the application occurs on discrete data to which I wish to use a continuous function it is necessary that I apply concrete mathmatics [Graham et al. 1989] to rectify this situation. Instead of applying a continuous function over the signal, I will instead define a set of matrices that will preform the wavelet transform in an discrete context. I can achieve this by taking the matrices that allow us to go from the father to son scaling functions and mother to daughter wavelets.

For a given scaling function ϕ(x) there must exist some matrix P^j such that I can produce the scaling function at the next level of resolution $j-1$ by taking the product, or:

$$\phi^{j-1}(x) = \phi^j(x)P^j \tag{4.9}$$

Likewise there must be some matrix Q^j such that given a scaling function $\phi(x)$, the product produces the wavelet function $\psi(x)$ at the next level of resolution $j-1$, or:

$$\psi^{j-1}(x) = \phi^j(x)Q^j \tag{4.10}$$

These matrices P^j and Q^j have transforms h^j and g^j expressed formally:

$$A^j = (P^j)^T \tag{4.11}$$

$$B^j = (Q^j)^T \tag{4.12}$$

I can use the rows these matricies A^j and B^j as the filters $\mathbf{g}^j$ and $\mathbf{h}^j$ to analyze the signal data and produce the coeffecients $\mathbf{c}^j$ and $\mathbf{d}^j$ for the wavelet transform through a processes called subband coding. These coefficients will give us translate and scale information about our original signal. In the case of sound this will correspond to approximate location in time and period of a given set of sounds.

4.2.2 Subband Coding

The Discrete Wavelet Transform (DWT) traces its origins back to subband coding in the 1970s [Polikar 1999]. Subband coding is a process by which time-scale representation of a signal is produced through filtering techniques. The signal is passed through high pass filters to obtain the highest grouping and then a series of low pass filters to get lower and lower subbands. If the range of frequencies in the original signal existed from $[0, \pi]$; using the low half band pass filter will reduce this range to $[0, \frac{\pi}{2}]$. Thus, after passing the signal through the half band low pass filter I can reduce the signal by removing half the samples, or subsampling by 2, in accordance to Nyquist's rule. Thus, subsampling scales our signal by a similar amount. The resolution can be described as the amount of information and thus is halved by the filtering process that has removed half the frequencies.

The remaining lower band frequencies lose a portion of their temporal information as a result of reducing the resolution. Because they can afford a loss of temporal resolution to reduce the number of frequencies in that subband, these values can further be decomposed by the same method until the number of samples have been reduce to 1. By comparison the higher band frequencies cannot be further parsed as an additional loss of information would not remove additional information and thus would lose temporal resolution for no gain. The collective result is that higher frequencies maintain better temporal resolution, but have a larger subband; while the lower frequencies lose temporal resolution, but maintain a much smaller frequency subband.

If the original signal is defined by the vector $\mathbf{X}$ with M elements I can express the application of these filters mathematically as

$$c_n^{j-1} = \sum_{i=0}^{\frac{M}{2}} x_i^j * g_{i-2n}^j \tag{4.13}$$

$$d_n^{j-1} = \sum_{i=0}^{\frac{M}{2}} x_i^j * h_{i-2n}^j \tag{4.14}$$

Where h_n^j and g_n^j constitute the high and low half band pass filters for resolution j, respectively, x_n^j the signal on level j, and c_n^{j-1} and d_n^{j-1} are the course and detailed coefficients at the next level after j. It is also worth noting that I am in the discrete case here so d_n^{j-1} constitutes a single signal sample, which is defined as the summation of the filter across all samples. These coefficients can be used to reconstruct the original data by using the matrices for the scaling and wavelet functions. Expressed here:

$$c^j = P^j c^{j-1} + Q^j d^{j-1} \tag{4.15}$$

4.2.3 Wavelet Implementation Used in the System

In our implementation the signal is initially decomposed into octave frequency bands using a multirate filter bank, here implemented as the discrete wavelet transform. The discrete wavelet transform provides high time resolution for high frequencies at the cost of frequency resolution and high frequency resolution for low frequencies at the cost of time resolution as discussed previously. Because this is a similar implementation to the human ear, the DWT provides an excellent tool for this sort of beat extraction when compared to an alternate technique like the short time Fourier Transform (which has uniform time resolution for all frequencies).

As the goal here is beat extraction I will be decomposing the signal into octave frequency subbands, each of which containing half the signals of the next higher frequency subband. The wavelet decomposes the signal into coarse approximation and detail information. The resulting coarse approximation is further decomposed recursively to achieve higher frequency resolution for the lower frequencies. This achieves sucessive highpass and lowpass filtering of the time domain signal while downsampling between steps. The resulting algorithm is pyramidal which has been shown to be fast [Mallat 1989]. The particular wavelet family used for this implementation is the 4 coefficient wavelet family (DAUB4) proposed

by Daubechies [1993]. The actual code can be see in figure 4.1.

PLOTHERE?

4.2.4 Daubechies Wavelets

Daubechies wavelet's have the property of being orthonormal and compact for the infinite real line, which satisfies our requirements for our filterbank. The coefficients of the DAUB4 wavelet family are expressed as follows:

$$p = a = \frac{1}{4\sqrt{2}}(1+\sqrt{3}, 3+\sqrt{3}, 3-\sqrt{3}, 1-\sqrt{3}) \tag{4.16}$$

$$q = b = \frac{1}{4\sqrt{2}}(1-\sqrt{3}, -3+\sqrt{3}, 3+\sqrt{3}, -1-\sqrt{3}) \tag{4.17}$$

Where the p sequence represents the nonzero entries of the columns in our vector **P** and q represents the same entries of the columns in the vector **Q**. Similarly a and b provide the same service for vectors **h** and **g** as rows to maintain the relationship between **P** and **h**. Furthermore these sequences are from a quadrature mirror filter, which means that it is possible create the wavelet sequence from the scaling function sequence by reversing the order of the entries and alternating their signs [Stollnitz et al. 1995].

4.3 Envelope Extraction

Once the data has been separated the time domain relevant amplitude envelope can be extracted for each band. The results can then be run through a simple autocorrelation function:

$$y(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)x(n+k) \tag{4.18}$$

Envelope extraction is a technique used too prepare the data. This is achieved by the bands being initially run through a low pass filter to identify dominant frequencies. Full wave

```
float* process(float* in, int n)
{
        static float c0_ = 0.4829629131445341f;
        static float c1_ = 0.8365163037378079f;
        static float c2_ = 0.2241438680420143f;
        static float c3_ = -0.1294095225512604f;

        if (n < 4) return in;
        nh = n >> 1;

        for (i=0, j=0; j <= n-4; j+=2, i++)
        {
                workspace_[i] = c0_ * in[j] + c1_ * in[j+1] +
                                c2_ * in[j+2] + c3_ * in[j+3];
                workspace_[i+nh] = c3_ * in[j] - c2_ * in[j+1] +
                                c1_ * in[j+2] - c0_ * in[j+3];
        }

        workspace_[i] = c0_ * in[n-2] + c1_ * in[n-1] +
                        c2_ * in[0] + c3_ * in[1];
        workspace_[i+nh] = c3_ * in[n-2] - c2_ * in[n-1] +
                        c1_ * in[0] - c0_ * in[1];

        return workspace;
}
```

Figure 4.1: *An example of the implementation of the DAUB4 wavelet for filtering, where n here represents the current level of resolution defined by* $n = 2^j$

rectification occurs to move our data into the positive domain. Next our data is down sampled to our ideal range and each band is normalized via mean removal. Finally the data is run through the autocorrelation function and the top five periodicities are added to the histogram The music analysis system Marsyas [http://marsyas.sness.net/] was used to implement the algorithms because the framework is naturally designed for synchronous signal processing.

While this beat histogram implementation does provide peaks for beats with greater strength it does not provide more in-depth insight into ranges of beats. If the underlying music has a constant tempo, then the corresponding tempos and dominant periodicities (beats) would show up as impulses spanning single bins of the histogram. Frequently music contains expressive changes in rhythm; therefore, several neighboring histogram bins are affected.

4.4 Graduated Non-Convexity

To distill this range of periodicities into a single value while preserving strength the original code was expanded by applying a technique known as Graduated Non-Convexity. This multiresolution technique is well founded in other computational areas such as computer graphics. As a result, there is some evidence that it can be helpful in this similar circumstance. Analytically, one can express this as an energy minimization function of the form:

$$\sum_{i=1}^{m} \chi_i (f_i - d_i)^2 + \lambda \sum_{i=1}^{m} \sum_{i' \in N} g_\gamma (f_i - f_{i'}) \tag{4.19}$$

Where d_i is the smoothing term and g_γ here represents our blur level going from $g_\gamma^{(n)}$ to $g_\gamma^{(0)}$ as our target, and $g_\gamma^{(n)}$ is sufficiently large to be strictly convex [Blake and Zisserman 1987].

The beat histogram is first blurred by a 1 dimensional Gaussian kernel to establish a pyramid. For the purposes of this implementation, kernels of size 3 were applied starting at a

$\gamma^{(4)}$. Major peaks were then identified on the blurred images. The peaks were then related to other peaks on histograms of higher-level granularity until the bottom of the pyramid was reached. After being identified, the areas around an identified beat rate were then modified with a Haar transform to simulate lateral inhibition. While lateral inhibition may or may not have a perceptual underpinning, such a statement is beyond the scope of this work. Lateral inhibition is necessary for this algorithm to give each group of peaks a deterministic result and to prevent identified peaks from providing secondary influence.

This algorithm was run on a series of music with different beat rates and tempos. Both standard and syncopated beats were successfully identified. A sampling of the results can be seen in Figure 4.2. Figure 4.3 shows a blur factor of $\gamma^{(4)}$. These graphs show that while some beats appear very strong, once I factor in neighbors, they do not retain their absolute strength. It should be noted that this algorithm does not run in real time, and is necessarily precomputed.

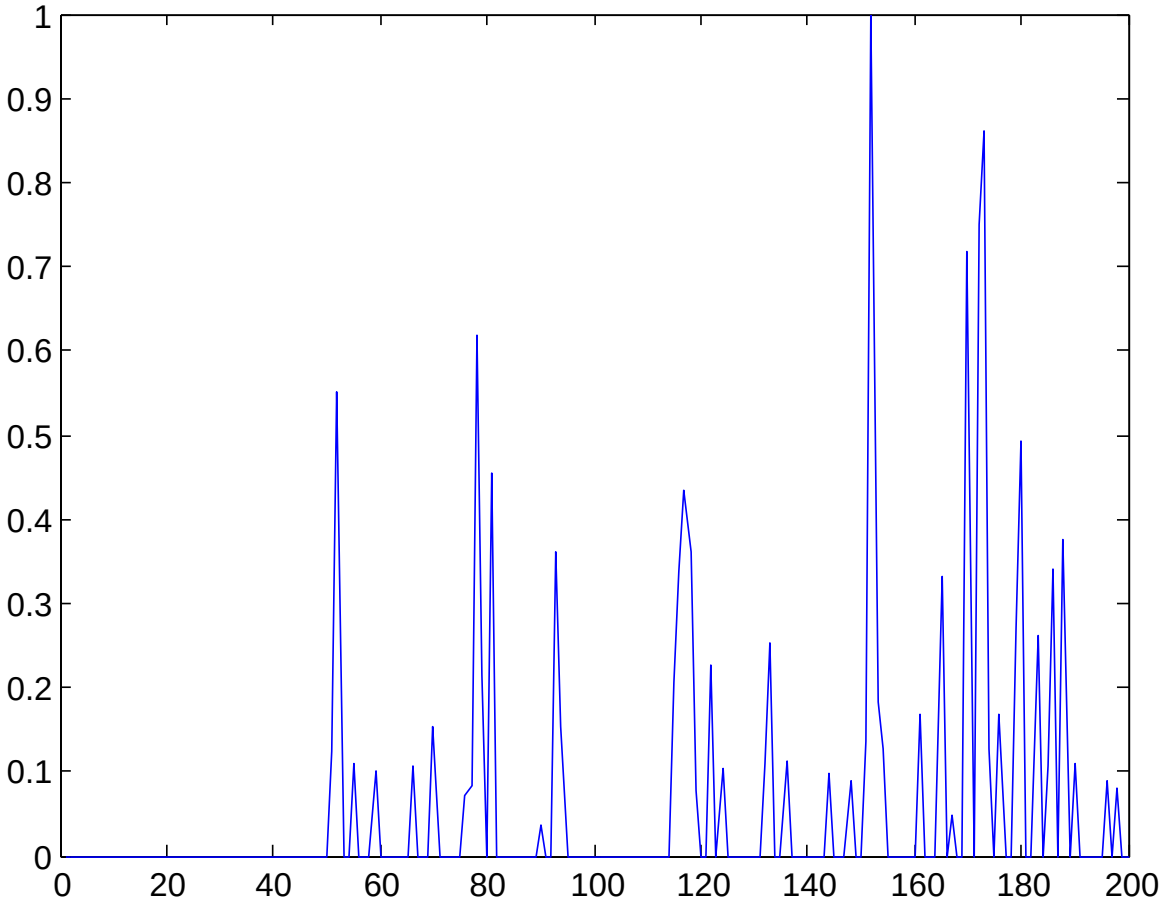

Figure 4.2: *Plot of Beat Histogram*

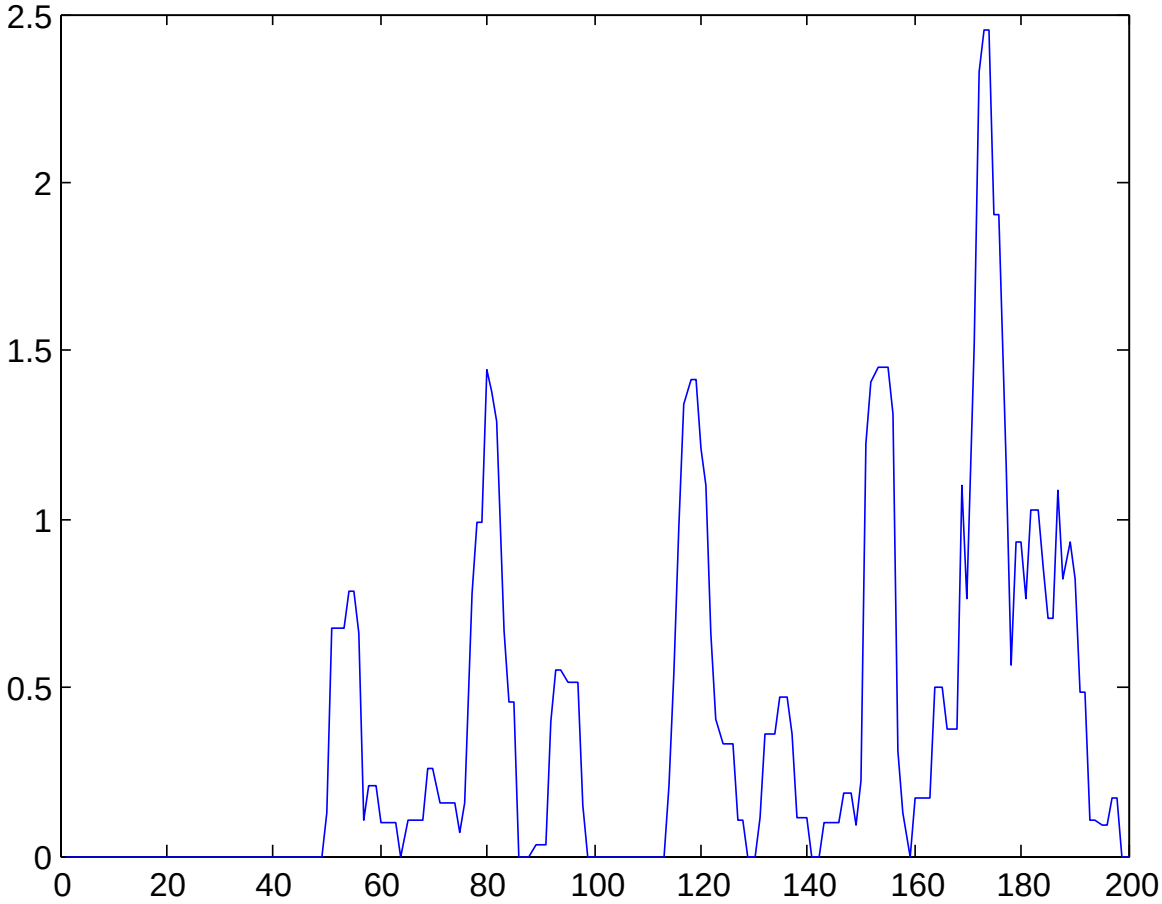

Figure 4.3: *Plot of Beat Histogram with a blurring factor of 4*

Chapter 5

Tempo Adaptation

Given the user beat rate, the nearest beat rate with the highest strength can be identified by our model. As an addendum, I can give greater importance to different kinds of input at this stage (stronger beats to keyboard or mouse as necessary). A ratio is then constructed between the target beat and the user rate. The ratio is used as a key for modifying the rate at which the music is fed to the audio driver, and eventually output. The method for time stretching and shrinking the music is a phasevocoder that allows for changes in the time domain while preserving frequencies.

5.1 Phasevocoder

A vocoder is a time-scaling algorithm that stretches audio samples over a larger or smaller window without causing a shift in the frequencies. The most popular vocoders achieve this dilation by considering overlapping time windows and aligning the “phases" between them. The result is that phase consistency within a given frequency channel is consistent over time, but that phase consistency is not maintained across all the channels in a given time slice. The phasevocoder in question is based on the work of Jean Laroche and Mark Dolson’s[1999].

5.1.1 Time-scale modifications

The phase-vocoder-based time scaling is a modified wavelet transform based upon the short-time Fourier transform (STFT) mentioned in sections 4.2 and 4.1.2 respectively. As the

STFT can be seen as a special case of the Wavelet transform one can consider my approach to be wavelet based, however, it will not have the "nice" characteristics our beat analysis did. The reason for this approach is implementation reasons: the fastest algorithm with little or no perceptible errors. One could theorize a wavelet based approach where the results are stretched in time, but this would require an appropriate change to the scaling function ϕ . It is likely that this approach might produce perceptually superior results at some point in the future; however the current state of the art does not currently produce perceptually superior results. Additionally, such a transform would be computationally more expensive, and as our time dilation need to run in real time, undesirable. As a result our Wavelet Transform will instead be a modified STFT to accomplish time scaling.

5.1.2 Analysis and Synthesis

Recall from Section 4.1.2 that the STFT provides analysis by taking windows of the original signal and then preforming the Fourier transform on each window to determine what frequencies are present. The resulting frequency specturm can later be run through the inverse Fourier transform and the windowed segmentes rejoined back into the original signal. As the purpose of a vocoder is to preserve frequency while manipulating time, no additional changes will be made to the frequency spectrum of any given window. Instead, when the signal is reassembled from the windowed segments the length of the window will be changed. As a result, when the windowed segments are rejoined they contain the same frequencies as before but now occupy more or less time than they did in the original signal. The result is, therefore, the same sounds occuring slower or faster than before. This time scaling approach is summerized in figure 5.1[Sethares 2009].

As I discussed earlier there is a trade off between frequency precision and time when attempting to identify when a sound occurs. In the STFT this trade off is constant across the

frequency domain based on equation 5.1.

$$Resolution = \frac{SamplingRate}{WindowSize} \tag{5.1}$$

As a result, given a sampling rate of 44100Hz and a window size of 1024 our frequency resolution is, at most, 44.1Hz. Thus a signal containing the frequencies 82.4 and 103.8 (low E and G# on a piano making a Major Third), are almost indistinguishable from one another. While taking a large window may seem like a solution to this problem, such an approach would strech frequenies past the time which they occured and thus distort the signal in a way which is undesirable. This seeming flaw in the STFT is what motivated the introduction of the Wavelet transform in section 4.2.

5.1.3 Phase Alignment

While the STFT does have poor frequency resolution (or time resolution with a larger window), it is possible to signficantly improve this resolution by using phase information. This addition seperates a common vocoder from a Phasevocoder. If one were able to obtain the phase information from one frame to the next, it would be possible to improve the frequency estimate. Because the difference between two phases of the same frequency must be a multiple of 2π and the frequency, given by equation 5.2, one can estimate the frequency given two phases and their times, as shown in equation 5.3.

$$f * 2\pi(t_2 - t_1) = \theta_2 - \theta_1 \tag{5.2}$$

$$f = \frac{\theta_2 - \theta_1 + 2\pi n}{2\pi(t_2 - t_1)} \tag{5.3}$$

for given phases θ_1 and θ_2 estimating frequency f with n as an integer.

A more detailed explanation and comparison of Phase Alignment techniques can be found in Appendix A.

5.2 Implementation of Time-Stretching

The actual implimentation of the phase vocoder occurs once again in the Marsyas [http://marsyas.sness.net/] framework. Information about the user interaction rate comes in through a pipe and is modeled in the manner described in chapter 3. Information received from the user model is a ratio representing R^s/R^a, with a being the analysis rate and s the synthesis rate. This Ratio will be used to modify the sampling rate of R^s against the known sampling rate of the audio in question (or R^a). As beat detection will, on occation, determine the harmonic of the underlying beat structure, Ratios greater than 2 or less than $\frac{1}{2}$ will be scaled by the inverse amount. I can then take this ratio and multiply it with the sampling rate to achieve the correct output rate. This is then input into our phasevocoder as our output-sampling rate R^s.

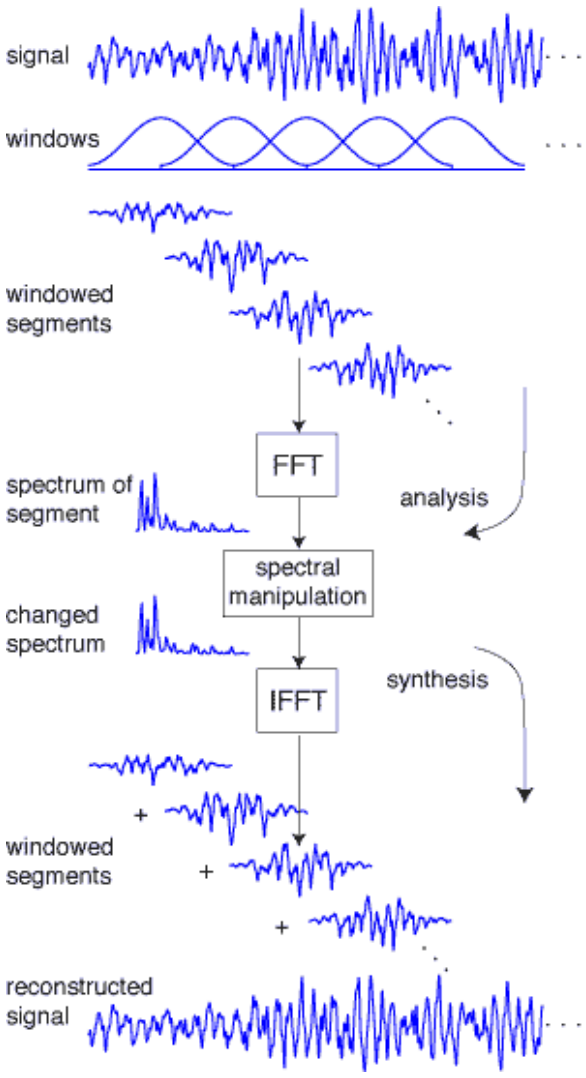

Figure 5.1: *Diagram of STFT*

While the phase vocoder will run in real time, the spacing between the currently playing time slice and the next slice being processes is not taken into account. As a result there is a potential loss of time as the window may now overlap with the currently playing window. This actual change in the output is perceptible if it occurs too often re-

gardless of phase alignment. To compensate for this the output is modified gradually at a rate of 1 to 64 every 8 slices of the current bit rate of music. While this approach is more gradual, it still occurs quickly enough that music is streched in real time. Additionally, if the model changes before the target rate is hit, this gradual change may provide a smoother shift.

Chapter 6

Results and Discussion

While I have discussed and implemented a fully functioning system it is necessary to evaluate our system both as a complete piece of software as well as the individual pieces to investigate whether our underlying assumptions are correct. One of the major assumptions of this work is that video games have an underlying structure. While this has been concluded in previous work [Smith et al. 2008], in this paper I report on the evaluation of the accuracy of this assumption. In the earlier section 3.3 I discussed how one could model the underlying structure, thus implying that there is a structure. In this section I will evaluate if this structure is a) correlated with user experience and b) unique to specific games. Finally, I will report on the evaluation of the fully functioning system.

6.1 Associating with User Experience

In addition to assuming there is an underlying structure, I assume this structure is in some way related to the game in question. It is possible that for any application a single users response may approximate some model. If user interaction with a video game is unique to the game in question (or even games in general), then I should be able to verify this by changing the game in some way. Such an association between a change in user interaction and a change in game play can show a relationship between the two and demonstrate how the experience is unique to the activity at hand.

One way that a game can be changed, without access to the source, is to modify the rate of play. While modern games have fixed play rates, earlier games relied on the processing

speed of the computer in question. As such they sometimes allowed for asynchronous control over the interaction. On such game, Starcraft, provided built in controls for the rate of play so that it could be adapted for older computers. Thus I can modify the rate of play a fixed intervals. My hypothesis then becomes that if the user is attempting to approximate a model then the resulting interarrival time should change proportionally to the change in the rate of play. If, on the other hand, the user merely interacts with the computing devices consistently in this way, I should see no variation.

Users were asked to play a game at two separate speeds (denoted to the user as Slow and Fast). These speeds were the equivalent of 75% and 100% standard game play (thus constituting a 33% speed up). Interaction was recorded from the pipe and saved to files for analysis. Game length was kept consistent across trials at the differing rates. Users were then asked to replay the same level at both speeds and the data was recorded in the previously discussed manner. The order here was assigned randomly and showed no changes in the data. Two data sets were generated per user and the data was analyzed for comparison using a T-test on the means of the trials.

Results indicated a significant difference between the two groups (P(T) $<$ 1%), and that users interacted with the game, on average, 17% faster while maintaining a similar standard deviation. This shift is statistically significant enough for us to indicate that increasing the speed does affect user interaction in the anticipated method. However, this shift is still smaller than the expected 33% increase. This suggests that there is a second factor at play here. While the user does speed up to accommodate the faster pace, there are limits to the speed of human interaction. If I examine only events shorter than .5 seconds (as in the previous subsection) I notice that in addition to having similar distributions, these make up half of all events in both cases, which explains why I see about half the speed up one might expect.

Cassification Matrix				
Genre	RTS	Plat	MMO	FPS
RTS	80%	20%	0%	0%
Platformer	30%	50%	20%	0%
MMO	0%	0%	90%	10%
FPS	0%	17%	17%	67%

Table 6.1: *The Results of Genre Classification*

6.2 Genre Classification

The idea that games have an underlying structure, or design, is not an original idea [Smith et al. 2008]. Demonstrating that this holds for games across genre or in general is. Because I have established that user input is predictable to a degree, and that it is a close approximation of whatever structure exists, it becomes necessary to show that this model is not identical across different games. It is possible that while people will respond to a given game differently based on pace, they will still respond similarly across games. One way to test this is to have users play a variety of different games and see if the input can be used to determine the genre.

To demonstrate this I conducted a pilot study where ten users were asked to play four different games. User input was recorded over the course of normal play. To guarantee the most noticeable results, games were chosen from four different genre: Real Time Strategy (Starcraft^{tm}), First Person Shooter (Left4Dead^{tm}), Massive Multiplayer Online Role Playing Game (World of Warcraft^{tm}), and Platform Leveler (Super Mario World^{tm}). Each genre was run for 10 trials, with the exception of the first person shooter, which received only 6. Data was gathered for each user, and mean and standard deviation of arrival time was calculated for each data sample.

Cassification Matrix			
Genre	RTS	MMO	FPS
RTS	90%	10%	0%
MMO	0%	90%	10%
FPS	0%	17%	83%

Table 6.2: *The Results Without Platformer Data*

To evaluate this data I used a Naive Baysian classifier with a 10 fold cross-validation on the average interarrival time and standard deviation. The implementation used was Weka[http://www.cs.waikato.ac.nz/ml/weka/]. Using a 10 fold cross-validation 72% accuracy was achieved; individual results are tabulated in table 6.1. The most notable standout was the Platformer with 50% accuracy. If I remove that from the data I achieve 88.5% accuracy as shown in table 6.2. This is most likely due to rapid changes in pace between levels, whereas the other genres tend to be consistent across levels. If one examines the data directly one sees that the platformer has a standard deviation from .5 seconds to 2.3 seconds. No other data set has a standard deviation spanning more than a second; moreover it spans the range of all three others (see 6.1).

While more data might still provide higher accuracy, this clearly shows that between these four games, the game being played can be determined purely from the input data. This, in turn, provides strong evidence that the underlying models being approximate by user input are different across these different video games. It is worth noting that identification of game type can be useful for other purposes, but those are beyond the scope of this work.

6.2.1 User Feedback

Finally, there is the concern that our system may have a negative effect on performance as a distractor. Research in the past has demonstrated that music, especially music characterized as “High Arousal" [Cassidy and MacDonald 2007], can have a negative effect. As our system is designed to adapt music to gameplay to reduce negitive effects, we evaluate it in the course of regular usage. For this purpose we conducted a pilot study to observe how well a user preformed under our system. Participants were recruited from the Faculty and Student Body of the University of Victoria Computer Science Department.

Participants were asked to play Super Mario World^{tm} in the manner in which they felt appropriate. If they were unfamiliar with the game they were given instructions on the controls. Participants were randomly assigned to play the game with no music first (Negative Control), with music which was not adapted (Positive Control, or our system (Experimental); this order was randomized for each user. Participants were allowed 5 trials with each condition for a total of 15 trials, where a trial consisted of a death or completion of the level. Data was recorded on how far into the level the users were able to progress in a single life. The data on table 6.3 summerizes the average distance into the level for each of the condidtions. The average distance of all trials was 22.2% which corresponded to approximately a minute and a half of game play.

If we look at the data from this study we can see that Cassidy’s[2007] findings are reaffirmed. In the presence of an additional distractor (Positive Control) performance is inhibited. However, when we examine the application of our current system vs. the negative control we notice that this performance difference is reduced as well as being reversed. Although the sample size is not sufficient to say this trend is certain $(P > .1)$ it is evidence that our system is not as significant a distractor $(P < .1)$.

Preformance	
Condition	Avg. Progress
Silence	24.15%
Unmodified	15.40%
Adapted	26.95%

Table 6.3: *A Comparison of the Preformance between Our System and Control*

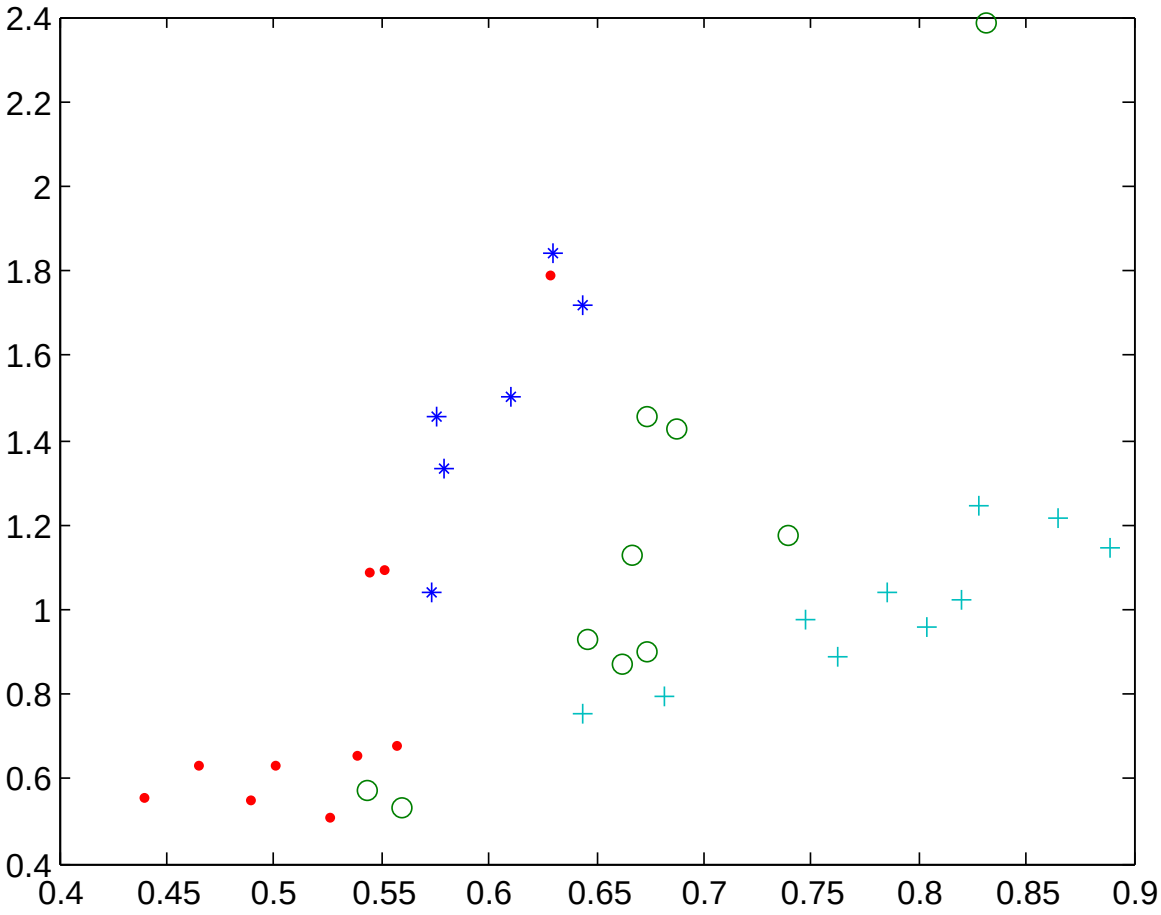

Figure 6.1: *Plot of Std vs Mean for 4 Different Genres, with Green Circle Representing the Platformer*

Chapter 7

Conclusions

This thesis has presented a fully implemented system for synchronizing an arbitrary audio track with active user input. To solve this problem I have broken it down into three component piece and designed an algorithm with this approach. The algorithm identifies user input and models it according to experimental results. The algorithm similarly identifies the periodicity of the audio track. Finally the algorithm synchronizes the input from the first two sections and plays it in real time.

To accomplish these tasks I have utilized a verity of tools each of which help to recognize what events occur and when. Because of the nature of this work there is not a single algorithm with which one can use for all the pieces. Instead each tool is specifically chosen for each of the parts of the algorithm. For modeling user data, it is not sufficient to merely identify beat rate over time, as beat rates will change over the course of the level. As a result, my algorithm identifies a specific model for a series of events and break this model when the user supplies a number of events that are statistically unlikely. For the time stretching algorithm, one needs a very high level of precision of which frequencies occur so that the result does not experience distortion in the frequency domain. To accomplish, this I use a short term Fourier transform for high frequency accuracy. For beat detection, the algorithm one can be relatively assured that beat rate changes less frequently over the course of the audio track. As a result, I can use a wavelet transform to retain better perceptual performance while identifying the beat rate. By combining these tools I construct an algorithm to solve this problem.

In addition to the algorithm, this thesis evaluates the hypothesis that user interaction is a

close approximation of the underlying structure of the game they are playing. To this end I have demonstrated that there is an underlying model which player interaction follows. I have evaluated this model in two different ways to show that the model is not purely descriptive of how a user interacts with games in general, but the specific game in question. Firstly, I showed that users rate of interaction is based on the game's rate of play. Secondly, I showed that this interaction varies across different games and that user interaction alone can be used as a fairly accurate identifier of the game in question. While these alone are sufficient to show that the user interaction is a good approximation of the structure of the game, I additionally demonstrate that modifying the audio with my algorithm does not significantly impair user performance.

7.1 Future Work

While the majority of this thesis is concerned with adapting music to user data, a large portion of our evaluation is concerned with that user data. Being able to draw information out of user interaction is a useful tool; our identifying genre based on user interaction is evidence of this. Future application of this system could includ identifying the specific game the user is playing, the competency of the user playing, or the specific user that is playing, all of which could be of great benefit to our system or the state of the art. Finally, being able to identify which musical track might fit best to the user interaction is a future goal of this system.

Appendix A

Phasvocoder

As phase alignment is a significant portion of the phasevocoder, a complete discussion of the implementation is necessarily included in this work. Because the phasevocoder is not a contribution of this work, the actual discussion of such phase alignment occurs here in the appendix.

A.1 Analysis and Synthesis

During the analysis of signal using a STFT is defined with an analysis hop factor R^a such that given a time-instant t_u^a for successive integer values u, $t_u^a = R^a u$. One can then calculate the foruier transform of this instant t_u^a by taking a windowed portion of the original signal centered about t_u^a. This STFT, denoted by $X_{t_u^a,\Omega_k}$ can be expressed:

$$X_{t_u^a,\Omega_k} = \sum_{n=-\infty}^{\infty} h(n) x_{t_u^a+n} e^{-j\Omega_k n} \tag{A.1}$$

with respect to the original signal x. In this case $h(n)$ is our windowing function which is necessarily compact. Here $\Omega_k = \frac{2\pi k}{N}$ is the center of the frequency channel k where N is the size of the discrete Fourier transform.

As time dilation is the desired result of this process it is possible to use a different synthesis hop factor R^s to give different time instants $t_u^s = R^s u$. One can thus obtain a short-time signal $y_u(n)$ for each of these time-instance via the inverse Fourier transform, denoted $Y_{t_u^s,\Omega_k}$. One

can then multiply each short-time signal with a synthesis window $w(n)$ and sum to produce the output signal y.

$$y_n = \sum_{u=-\infty}^{\infty} w(n - t_u^s) y_{n-t_u^s}^u \tag{A.2}$$

where:

$$y_n^u = \frac{1}{N} \sum_{k=0}^{N-1} Y_{t_u^s, \Omega_k} e^{j\Omega_k n} \tag{A.3}$$

For phasevocoder scaling the STFT is modified slightly. First the scaling factors are different (as mentioned), but also the phase values of the synthesis STFT $Y_{t_u^s, \omega_k}$ are calculated according to a formula. Recall, the input signal can be expressed as the sum of a number I_t of sinusoids with time-varying amplitudes A_t^i and instantaneous frequencies $w^i(t)$. Thus the following must hold true:

$$x_t = \sum_{i=1}^{I_t} A_t^i e^{j\phi_t^i} \tag{A.4}$$

Here ϕ^i represents the instantaneous phase determined by:

$$\phi_t^i = \phi_0^i + \sum_{\tau=0} t w^i(\tau) \tag{A.5}$$

If one wishes to work with a constant modification factor α such that $t_u^s = \alpha t_u^s$, then one can calculate the ideal synthesis phase $\phi_{t_u^s}^s$ of the sinusoid i as:

$$\phi_{t_u^s}^s = \phi_0^s + \alpha(\phi_{t_u^a}^i - \phi_0^i) \tag{A.6}$$

with a starting phase of ϕ_0^s . This produces the correct modification of time-evolution of the sine waves, but leaves the phases uncorrected.

A.2 Phase Alignment

While the STFT will produce values which dictate the frequency at a given time slice, these frequencies are not necessarily have the correct phase. Due to the influence different waves have on each other if they are “in" or “out" of phase, a complete algorithm must identify and correct phases between time slices and frequency channels.

A.2.1 Phase Identification

To identify the phase of the STFT $Y(t_u^a, \omega_k)$ it is necessary to first preform phase unwrapping. Phase unwrapping is a technique that gives us the phase increment between consecutive frames. This is important because the phase between one time slice and the next need to be consistent. The phase increment can then be used to estimate the instantaneous frequency of sinusoids in a given channel channels. The equation to calculate the heterodyned phase increment is as follows:

$$\delta\phi_u^k = \angle X(t_u^a, \Omega_k) - \angle X(t_{u-1}^a, \Omega_k) - R^a\Omega_k \tag{A.7}$$

The instantaneous frequency $\hat{w}_k(t_u^a)$ of the closest sinusoid is determined by the phase increment $\delta\phi_u^k$ and the frequency channel given by Ω_k :

$$\hat{w}_k(t_u^a) = \Omega_k + \frac{1}{R^a}\delta\phi_u^k \tag{A.8}$$

where the principal determination of the increment is taken between $\pm\pi$ because of the cyclical nature of sinusoids. One can think of this as the increment $\delta\phi_u^k$ being the phase shift between the instantaneous frequency $w^k(t_u^a)$ and the frequency channel Ω_k . As phases need to be aligned between time slices, the value $\hat{w}_k(t_u^a)$ can be used to propagate the correct phase alignment between time slices. This relationship can be viewed as the phase of the

STFT $\angle Y(t^s_{u-1}, \Omega_k)$ being incremented by the instantaneous frequency $\hat{w}_k(t^a_u)$ after being scaled by our synthesis factor R^s, or:

$$\angle Y(t^s_u, \Omega_k) = \angle Y(t^s_{u-1}, \Omega_k) + R^s \hat{w}_k(t^a_u) \tag{A.9}$$

As this propagation is restricted to a signal frequency channel and because successive short-time signals are overlapped, the relationship has phase coherence in that channel across time. In order to guarantee good estimates it is recommended to use a standard analysis window of at least 75% . It is also worth noting that phase propagation requires an initial t^s_o . This choice of initial phase will affect the quality of the output signal. While this is the standard technique there are alternatives [Portnoff 1981].

A.2.2 Phase Coherency

As has already been stated the phase-vocoder time-scaling algorithm guarantees phase consistence in a given frequency channel (sometimes referred to as horizontal coherence), but lacks the necessary phase consistency across channels for a given window (this can be thought of as vertical phase coherence). Based on our original definition of the synthesis STFT as $Y_{t^s_u, \Omega_k}$, our algorithm will not necessarily yield a signal who's STFT is close to $Y_{t^s_u, \Omega_k}$ because of this lack of vertical coherence. As a result, there should be perceptible as "beating" in the harmonics.

Given a constant amplitude standing wave there is a simple phase relationship between boardering channels located around this frequency. I. e. if a sinusoid with a constant frequency w_i falls in a channel k with a symmetric analysis window around 0 , it becomes obvious that the channels around k are influenced by this sinusoid:

$$|\Omega_k - w_i| < w_h \tag{A.10}$$

where w_h is the cutoff frequency of the analysis window. Frequencies about this channel will have the same phase as those inside the channel. Given an analysis window $h(n)$ that is more non-zero for $0 \leq n < L$ that is symmetric about its middle point, if the size of the discrete Fourier transform is equal to the analysis window length L, then near by channels will be out of phase by $\pm\pi$. Again, this only works for a standing wave with constant amplitude. For more complex signals no simple phase relationship exists. For example, if I have a sinusoid with a varying frequency, the phases in channel around the instantaneous frequencies are very close to equal, but have no analytical formula that has been developed. As a result one cannot easily check for vertical coherence.

Despite this lack of an easy analytic formula is its possible to derive the error term D^M of STFT from a set of reconstructed signals y_n by checking if the STFT of y_n is close to $Y_{t_u^s,\Omega_k}$ in both amplitude and phase. Griffin and Lim[1984] propose the following formula to determine D^M :

$$D^M = \frac{\sum_{u=P}^{U-P-1}\sum_{k=0}^{N-1}\left(|Z_{t_u^s,\Omega_k}| - |Y_{t_u^s,\Omega_k}|\right)^2}{\sum_{u=P}^{U-P-1}\sum_{k=0}^{N-1}|Y_{t_u^s,\Omega_k}|^2} \tag{A.11}$$

where $Z_{t_u^s,\Omega_k}$ is the STFT of the modified signal set y_n. U here is the number of frames, where I avoid the first and last P frames dues to errors resulting from missing overlapped windows during resynthesis. Ideally, the smaller D^M the more consistency as a D^M of 0 implies that $Z_{t_u^s,\Omega_k} = Y_{t_u^s,\Omega_k}$ for all t_u^s on channels Ω_k.

To understand the effect of consistency I will examine the unwrapping factor at the time analysis time-instant t_i^a given by the difference between the heterodyned phase increment and its principal determination between $\pm\pi$, or:

$$2m_i^k\pi = \delta_p\Phi_i^k - \delta\Phi_i^k \tag{A.12}$$

Because the synthesized phase depends on the analysis phase at a given time-instant and at the origin, any incorrect estimates of the analysis phase will not propagate to subsequent frames as long as the phase-unwrapping factor m_i^k is correct. However, if m_i^k is not correct then all subsequent phases will show that bias.

A.3 Dealing with Phasiness

One can describe this inconsistency between channels as “Phasiness." There are a number of strategies to deal with phasiness, but one of the more interesting ones is avoiding phase unwrapping. To reconstruct purely from the STFT magnitude it becomes necessary to prefore numerous iterations of the STFT analysis/synthesis in order to obtain more accurate STFT and thus reducing the D^M term. While such methods can be guaranteed to converge, there is no assurance that such convergence is the global minimum. Additional, numerous iterations prove to be too computationally intensive for a real time application like this one. At such a point it is almost preferable to investigate a more general wavelet based approach.

A.3.1 Loose Phase Locking

If computation time is a major constraint, one of the simplest solutions is a Loose Phase Locking algorithm. Puckette[1995] motivates this algorithm by the observation that a constant amplitude and frequency sinusoid in a given channel should have similar analysis phases in nearby channels. As a result one can lock channels to nearby phases if that channel is a maximum of the Fourier transform magnitude. The result is a bidding system where each channel bids against its neighbors and the channel with the highest bid gets to keep its phase while the losing channels adopt it. Thus channels around a peak are "phase locked" to the phase of that peak.

I can express this as the complex number:

$$Y_{t_u^s,\Omega_k} + Y_{t_u^s,\Omega_{k-1}} + Y_{t_u^s,\Omega_{k+1}} \tag{A.13}$$

Where if channel k is the maximum its phase remains unchanged as a result of $Y_{t_u^s,\Omega_{k-1}}$ and $Y_{t_u^s,\Omega_{k+1}}$ being much lower in amplitude. However, if the channel k is not the maximum, its phase will be determined by $Y_{t_u^s,\Omega_{k+1}}$ who's magnitude is larger.

While this algorithm is very fast due to the small number of additional operation per channel, it does not prove a significant reduction is phasiness is the general case.

A.3.2 Identity Phase Locking

Loose phase locking was efficient because it does not explicitly calculate the underlying structure of the signal. If efficiency is not the major concern than a more complete sum-of-sinusoids model will allow for accurate reconstruction of the underlying structure to develop vertical phase coherence. Good phase coherence can be achieved by identifying peak channels and calculating the associated phase. Phases of near by channels can then be locked to the phase of the peak channel. Loose Phase locking can be reconstructed as an application of this concept. One can further expand on this concept by constraining the phase difference between successive channels around a peak to have the same phase differences as the original analysis Fourier transform. Thus for a given frequency channel k and a near by peak channel kl the following must hold true:

$$\angle Y_{t_u^s,\Omega_k} = \angle Y_{t_u^s,\Omega_{kl}} + \angle X_{t_u^a,\Omega_k} - \angle X_{t_u^a,\Omega_{kl}} \tag{A.14}$$

Despite the additional computation necessary to calculate the phase much of these calculations can be amortized over the current system. As the phase unwrapping need only occur at the peaks, the phase unwrapping constraint $R_a w_h < \pi$ can be relaxed. The net benefit is an input overlap of 50% is possible. As the phasevocoder itself usually requires an overlap

1. For each STFT frame, identify peaks
2. For each peak, calculate the instantaneous frequency using horizontal phase unwrapping
3. Calculate rotation angle θ with the phasor $\zeta = e^{j}\theta$
4. Apply rotation to all channels around the peak
5. Repeat

Figure A.1: *Algorithm for Identity Phase Locking*

of at least 75% identity phase locking actually reduces the computational cost by allowing a relaxation of the window. The additional cost can further be minimized by calculating the angle of rotation as a single complex multiply:

$$\theta = \angle Y_{t_u^s,\Omega_{kl}} - \angle Y_{t_u^s,\Omega_{kl}} \tag{A.15}$$

$$Y_{t_u^s,\Omega_k} = \zeta X_{t_u^s,\Omega_k} \tag{A.16}$$

where ζ is the phasor $\zeta = e^{j}\theta$.

A.3.3 Scaled Phase Locking

There is an implicit relationship between a peak at frame $u-1$ and a peak at frame u going frome channel k_0 to k_1. This relationship means that the phase unwrapping equation should be based on $\angle X_{t_u^s,\Omega_{k_1}} - \angle X_{t_u^s,\Omega_{k_0}}$ instead of $\angle X_{t_u^s,\Omega_{k_1}} - \angle X_{t_u^s,\Omega_{k_1}}$. As a result phase propagation can be fully expressed as:

$$\angle Y_{t_u^s,\Omega_{k_1}} = \angle Y_{t_u^s,\Omega_{k_0}} + R_s \hat{w}_{k_1}(t_u^a) \tag{A.17}$$

where the phase increment $R_s \hat{w}_{k_1}(t_u^a)$ is based on $\angle Y_{t_u^s,\Omega_{k_0}}$ instead of $\angle Y_{t_u^s,\Omega_{k_1}}$.

While this relationship is useful, there is the additional problem of associating the peak at frame $u-1$ to the peak in frame u. One way to identify the peak is to select the peak of the region in which the channel belonged in the previous frame $u-1$. Neighboring channels can thus be syncronized to this peak. The generalized equation for identity phase locking from the previous section can be expressed as:

$$\angle Y_{t_u^s,\Omega_k} = \angle Y_{t_u^s,\Omega_{kl}} + \beta(\angle X_{t_u^a,\Omega_k} - \angle X_{t_u^a,\Omega_{kl}}) \tag{A.18}$$

where β is the scaling factor. For identity phase locking $\beta = 1$. Various implimentations have shown that it is possible in practice to achieve better phase coherence by setting β between 1 and α. Values with a low amount of phasiness have been found to be approximately $\beta \approx \frac{2}{3} + \frac{\alpha}{3}$ where α is the constant with which the unwrapping factor is multiplied [Laroche and Dolson 1999]. It is worth mentioning that the phases for the analysis STFT must be unwrapped prior to applying the scaling factor or risk $2\beta\pi$ jumps in the synthesis.

Bibliography

[Blake and Zisserman 1987]BLAKE, A., AND ZISSERMAN, A. 1987. *Visual Reconstruction*. MIT Press.

[Blattner et al. 1989]BLATTNER, M. M., SUMIKAWA, D. A., AND GREENBERG, R. M. 1989. Earcons and icons: Their structure and common design principles (abstract only). *SIGCHI Bull. 21*, 1, 123–124.

[Cassidy and MacDonald 2007]CASSIDY, G., AND MACDONALD, R. A. 2007. The effect of background music and background noise on the task performance of introverts and extraverts. *Psychology of Music 35*, 3, 517–537.

[Chafe and Leistikow 2001]CHAFE, C., AND LEISTIKOW, R. 2001. Levels of temporal resolution in sonification of network performance. In *Auditory Display*, International Community for Auditory Display.

[Chui 1992]CHUI, C. K. 1992. *An introduction to wavelets*. Academic Press Professional, Inc., San Diego, CA, USA.

[Clauset et al. 2007]CLAUSET, A., SHALIZI, C. R., AND NEWMAN, M. E. J., 2007. Power-law distributions in empirical data.

[Cram and Duser 2000]CRAM, S. M. J. H. D., AND DUSER, B. L. V. 2000. Effect of music tempo on heart rate and perceived exertion during rest, exercise, and recovery. *Research Quarterly for Exercise and Sport*.

[Daubechies 1993]DAUBECHIES, I. 1993. Orthonormal bases of compactly supported wavelets ii: variations on a theme. *SIAM J. Math. Anal. 24*, 2, 499–519.

[Drake et al. 2000]DRAKE, C., PENEL, A., AND BIGAND, E. 2000. Tapping in time with mechanically and expressively performed music. *Music Perception 1*, 18, 1–23.

[Graham et al. 1989]GRAHAM, R. L., KNUTH, D. E., AND PATASHNIK, O. 1989. *Concrete mathematics: a foundation for computer science*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Griffin and Lim 1984]GRIFFIN, AND LIM. 1984. Signal estimation from modified short-term fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing ASSP-32*, 2.

[Griffin 1998]GRIFFIN, D. S. 1998. Musical techniques for interactivity. *Gamasutra 2*, 18, 3–5.

[Holm et al. 2005]HOLM, J., HAVUKAINEN, K., AND ARRASVUORI, J. 2005. Personalizing game content using audio-visual media. In *ACE '05: Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, ACM, New York, NY, USA, 298–301.

[Holm et al. 2006]HOLM, J., ARRASVUORI, J., AND HAVUKAINEN, K. 2006. Modifying game content with personal media. In *ACE '06: Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, ACM, New York, NY, USA, 86.

[http://marsyas.sness.net/]HTTP://MARSYAS.SNESS.NET/. Marsyas: Musical analysis, retrieval and synthesis for audio signals.

[http://www.cs.waikato.ac.nz/ml/weka/]HTTP://WWW.CS.WAIKATO.AC.NZ/ML/WEKA/. Weka 3: Data mining software in java.

[Karageorghis and Terry 1997]KARAGEORGHIS, C. I., AND TERRY, P. C. 1997. The psychophysical effects of music in sport and exercise: a review. *Journal of Sport Behavior (JSB) 20*, 1, 54 – 68.

[Kilander and L'onnqvist 2002]KILANDER, F., AND L'ONNQVIST, P. 2002. A whisper in the woodsŮ an ambient soundscape for peripheral awareness of remote processes. *International Conference on Auditory Display*.

[Konecni 1982]KONECNI, V. J. 1982. Social interaction and musical preference. *The Psychology of Music*, 497–516.

[Laroche and Dolson 1999]LAROCHE, J., AND DOLSON, M. 1999. Improved phase vocoder time-scale modification of audio. *IEEE Transactions on Speach and Audio Processing 7*, 3.

[Mallat 1989]MALLAT, S. G. 1989. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11*, 674–693.

[Marsh 1993]MARSH, K. 1993. Win 32 hooks. *MSDN*.

[Matesic and Cromartie 2002]MATESIC, B. C., AND CROMARTIE, F. 2002. Effects music has on lap pace, heart rate and perceived exertion rate during a 20-minute self-paced run. *The Sport Journal*.

[McKelvie and Low 2002]MCKELVIE, P., AND LOW, J. 2002. Listening to mozart does not improve children's spatial ability: Final curtains for the mozart effect. *British journal of developmental psychology 20*, 2, 241–258.

[Minami et al. 1998]MINAMI, K., AKUTSU, A., HAMADA, H., AND TONOMURA, Y. 1998. Video handling with music and speech detection. *IEEE MultiMedia 5*, 3, 17–25.

[MSDN]MSDN. Understanding directinput. http://msdn.microsoft.com/en-us/library/ee418998(VS.85).aspx.

[O'Keefe and Haines 2009]O'KEEFE, K., AND HAINES, E., 2009. Dancing monkeys. http://monket.net/dancing-monkeys-v2/Main_Page.

[Pareto 1972]PARETO, V., 1972. Manual of political economy. Translation.

[Polikar 1999]POLIKAR, R., 1999. The wavelet tutorial. http://users.rowan.edu/ polikar/WAVELETS/WTtutorial.html.

[Portnoff 1981]PORTNOFF. 1981. Time-scale modifications of speech based on short-time fourier analysis. *IEEE Transactions on Acoustics, Speech, and Signal Processing 29*, 3.

[Puckette 1995]PUCKETTE, M. S. 1995. Phase-locked vocoder. *in Proceedings IEEE ASSP Worksop on application of signal processing to audio and acoustics*.

[Rauscher et al. 1993]RAUSCHER, F. H., SHAW, G. L., AND KY, C. N. 1993. Music and spatial task-performance. *Nature 365*, 611.

[Sethares 2009]SETHARES, W. A., 2009. A phase vocoder in matlab. http://eceserv0.ece.wisc.edu/ sethares/vocoders/phasevocoder.html.

[Smith et al. 2008]SMITH, G., CHA, M., AND WHITEHEAD, J. 2008. A framework for analysis of 2d platform levels. In *Sandbox Symposium*, ACM SIGGRAPH.

[Stollnitz et al. 1995]STOLLNITZ, E. J., DEROSE, T. D., AND SALESIN, D. H. 1995. Wavelets for computer graphics: a primer. 2. *Computer Graphics and Applications, IEEE 15*, 4, 75–85.

[Stroop 1935]STROOP, J. R. 1935. Studies of interference in serial verbal reactions. *Journal of Experimental Psychology*, 18.

[Tzanetakis et al. 2002]TZANETAKIS, G., ESSL, G., AND COOK, P. 2002. Human perception and computer extraction of musical beat strength. *5th Int. Conference on Digital Audio Effects*.