MusicBox: Navigating the space of your music

by

Anita Shen Lillie

Bachelor of Science in Mathematical and Computational Science
Stanford University, 2001

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

Master of Science in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2008

Author________________________________
Program in Media Arts and Sciences
August 8, 2008

Certified by___________________________
Tod Machover
Professor of Music and Media
Program in Media Arts and Sciences
Thesis Supervisor

Accepted by___________________________
Deb Roy
Chair, Academic Program in Media Arts and Sciences

MusicBox: Navigating the space of your music

by

Anita Shen Lillie

Abstract

Navigating increasingly large personal music libraries is commonplace. Yet most music browsers do not enable their users to explore their collections in a guided and manipulable fashion, often requiring them to have a specific target in mind. MusicBox is a new music browser that provides this interactive control by mapping a music collection into a two-dimensional space, applying principal components analysis (PCA) to a combination of contextual and content-based features of each of the musical tracks. The resulting map shows similar songs close together and dissimilar songs farther apart. MusicBox is fully interactive and highly flexible: users can add and remove features from the included feature list, with PCA recomputed on the fly to remap the data. MusicBox is also extensible; we invite other music researchers to contribute features to its PCA engine. A small user study has shown that MusicBox helps users to find music in their libraries, to discover new music, and to challenge their assumptions about relationships between types of music.

MusicBox: Navigating the space of your music

Anita Shen Lillie

The following people served as readers for this thesis:

Thesis Reader␣␣

John Maeda
Professor of Media Arts and Sciences
Program in Media Arts and Sciences

Thesis Reader␣␣

Henry Lieberman
Research Scientist
MIT Media Laboratory

Thesis Reader␣␣

Paul Lamere
Principal Investigator
Sun Research Laboratories

Acknowledgments

In completing this thesis, there are many people to whom I am extremely grateful. I am a very lucky person to have them in my life.

Thanks to...

> Tod Machover, who welcomed me into the Lab, and allowed me the freedom to do this work. The Media Lab has been an incredible place to explore, learn, and play.

> Paul Lamere and Henry Lieberman, who gave me tremendous feedback throughout this project.

> Tristan Jehan and Brian Whitman, who have shared so much of their own work with me.

> My mom, who taught me to work hard and stay organized, and my dad, who encouraged me to always ask questions. Thanks for the DNA!

> Jennifer, the closest person to me in the world.

> Ian, who put a lot into this thesis, even if it isn't visible on these pages.

> Craig, with whom I've made an important journey.

> Peter, for whom I have the utmost respect. May you reach your ever-after.

> Adam, who taught me to play Go. I will study hard and come back to crush you.

> Brent, for lifelong friendship.

> Rony, for new friendship.

> My elementary school teachers Mr. Michael Seay and Mrs. Penelope Mincho. You two made a huge difference in my early life, and what you taught me keeps me active and curious, even more so every day.

My thesis buddies: Alyssa, Annina, Jamie, Marcelo, and Nadav. k-up, all the way.

Barry Vercoe, for looking out for me during my time at the Lab.

Kirthi, John, and Turtle, for their home and warm company during my time in Cambridge. I will miss you so much.

And finally, Linda Peterson, the steady rock for all of the graduate students at the Lab. Thanks for your calmness and patience at just the right moments.

Contents

List of Figures

Chapter 1

Introduction

1.1 The problem

Over the past ten years, widespread use of the MP3 audio encoding format and, in turn, the proliferation of peer-to-peer file sharing and online music stores has lead to a dramatic increase in the average size of personal music libraries. In addition to having larger personal music libraries, users have easy access to huge numbers of MP3s online. No longer is your selection limited to that of your local record store, but online retailers such as Amazon.com and iTunes allow you to buy new MP3s from tens of thousands of artists with only the click of a button.

As music library size increases, it becomes correspondingly more difficult to navigate, browse, and find new music. The goal of this thesis is to ameliorate this issue by developing an alternative method of music browsing that is more convenient, enjoyable, and meaningful.

Surprisingly, widely available music browsers have not changed their navigation capabilities to keep up with this trend. Commonly-used music browsers still rely primarily on lists of static textual metadata (e.g. artist/album/song name, genre, release date, track number), and require the user to recall the connection between the music and these pieces of information. They have only limited functionality for personalized navigation based on dynamically-updated metadata, such as play count and track rating. Interfaces like this afford the user little flexibility to organize or search the music in a personal way, nor do they illuminate anything beyond simple relationships between the items in the library.

In addition, music browsers do not provide a way to search the audio content of the music itself, with their "genre" classifier being the only element that allows the user to approximate a search

of what the music sounds like. Even the "genre" classifier is rife with inconsistency and imprecision; "rock" can describe an enormous range of musical style, for example, and indeed there are many audio tracks that cannot be classified into just one genre.

This thesis abandons a reliance on these limiting textual classifiers, and instead presents a more abstract, flexible approach to navigating and exploring your own music library.

1.2   My contribution: MusicBox

MusicBox is a new music browser that organizes a music library's tracks into a 2D space, with track locations determined by a dimensional reduction of characteristic features of the tracks themselves. These "features" can be both content-based (audio) descriptors and contextual descriptors gathered from online sources.

Because MusicBox maps a music collection onto a space, the user experience of this new music browser is to *enter the space of their music*, and to move about in that space to find the music they seek. Similar objects appear closer together, and the user can see clusters of albums, artists, songs, and genres. Unlike mainstream list-based music browsers, this spatial representation of music emphasizes other meaningful relationships between different pieces and types of music, and motivates the user to explore those relationships.

MusicBox's automatically-generated, two-dimensional model of a user's music library is useful on its own because it creates an intuitively organized space that summarizes a much higher dimensional data set. However, one key focus of this project is to also give the user the ability to flexibly manipulate and explore the space, to control and add to what data contributes to the 2D spatialization, and to filter the data based on any component feature. Therefore a bulk of the work in this thesis is directed at building the user interface to enable these sorts of actions.

This research not only provides the groundwork for a new kind of browsing system, but also lays the framework for a larger recommendation system in which the user can find new music that fits their tastes. With access to a very large body of music (like that which might be released by a record company or sold in a music store), MusicBox can show the user their music in the context of music they do not yet listen to. The user can then find music that is similar to music they already like, or, alternatively, find music that fills a "hole" in their collection space. In addition, the static image of one's listening patterns and preferences within the context of a larger, defined set of music can be interpreted as a symbol of musical identity.

The goal of the MusicBox project is to make an understandable space for:

- navigation, exploration, and recommendation

- exploration of unfamiliar music collections (e.g. of a stranger, a friend, or a company)

- showing one's music collection in the context of a larger or separate music collection

MusicBox's core elements are:

- an extensible set of audio and contextual features

- a dimensional reduction of musical data

- an interactive visualization tool

1.3   The MusicBox experience
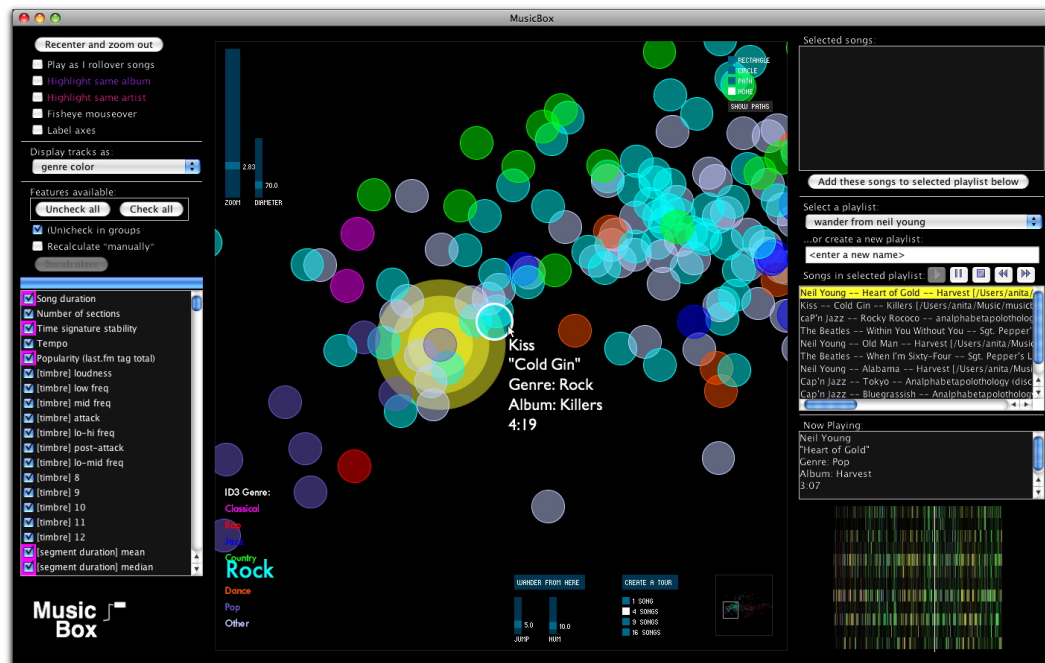
Figure 1-1: Kiss and Neil Young: MusicBox can help you make playlists of even the most unlikely musical neighbors.

One of the first times I sat down to MusicBox, I wanted to find some music that fit my task – I was about to cook dinner – so I decided to find one suitable track, and then create a playlist from that track alone. Simply looking at the genre-coded space, I knew I wanted to stay away from the rap and hip-hop area; I was looking for something more calming, but still with a beat. So I steered away from the hip-hop, rap, and classical, and landed right in the middle of a crowded rock area. The side of this region farthest from the hip-hop area had some softer beats,

and soon I had found Neil Young's "Heart of Gold". This fit my mood well, and I thought it would be a great seed for my playlist.

To create my playlist, I clicked the "Wander from here" tool, selecting a random walk from "Heart of Gold". The second track in the list was by Kiss: "Cold Gin". This surprised me, since I don't think I like Kiss and didn't even know I had any of their music. But, how pleasant a surprise it was, when I listened to the track and found that it had a very classic rock beat that actually went very well with "Heart of Gold", and without all the harsh screaming I would have expected from a Kiss track. If I could manage to get the image of the Kiss band members out of my head, I'd almost say this track had a country twang to it.

The next track MusicBox had selected was by emo band Cap'n Jazz, from their retrospective "Analphabetapolothology". Since it was a slow Cap'n Jazz track, it provided a smooth transition from the first two louder, more energetic selections, winding down to the next track, The Beatles' "Within You Without You". The rest of the playlist had some slow Neil Young (including "Old Man") and more Beatles, and then had some more Cap'n Jazz as a transition back to another Kiss track, "Love Gun".

Not bad. With one click, I had a playlist containing a variety of music, and all transitions between tracks felt smooth. If left to my own devices, I would never have put Cap'n Jazz and Neil Young in the same playlist, but somehow it seemed to work out alright this way. I had also managed to rediscover a track by Kiss, an artist I had forgotten I even had.

———

Later that evening, I tried making another playlist by selecting songs in the immediate vicinity of "Heart of Gold", instead of a path that might venture farther away. The resulting playlist was a very pleasant mix of some more Neil Young, some Willie Nelson, and John Coltrane.

1.4   Thesis structure

This thesis is divided into seven chapters. The first chapter is a short introduction to the motivations and goals for MusicBox. The second chapter provides background on navigating large data sets, centering in on musical data sets and the field of Music Information Retrieval (MIR). Chapter 3 contains an overview of the state-of-the-art in music browsers. Chapter 4 discusses *soundsieve*, one of my early projects at the lab, which visualizes the structure of individual songs. Chapter 5 is devoted to this thesis's core project, MusicBox, outlining its design and implementation. Chapter 6 presents the results of a small user study to evaluate MusicBox, and Chapter 7 closes the thesis by looking forward to a brighter future of music discovery. The two appendices describe algorithms employed by the MusicBox project.

Chapter 2

Background – Terms and Concepts

The enormous multiplication of books in every branch of knowledge is one of the greatest evils of this age; since it presents one of the most serious obstacles to the acquisition of correct information, by throwing in the reader's way piles of lumber in which he must painfully grope for the scraps of useful lumber, peradventure interspersed.
– Edgar Allan Poe

This chapter begins with a broad definition of the problem of searching large data sets, and then focuses more closely on the field of Music Information Retrieval, or the specific case of musical data sets.

2.1 Searching for the indescribable

Anyone who has tried to search a database at one time or another has faced the problem of trying to find something without knowing the most effective words to describe it. You might be searching for a movie whose title you've forgotten, or a paper while you only remember the topic, or a piece of music when you only remember part of the melody. How can you go about finding these things? How can you search the database of the web when you don't have words to describe what you're looking for?

In the field of information retrieval, there is a similar conundrum called the **vocabulary problem** [1]. This phrase specifically refers to the fact that users do not spontaneously choose consistent terms for the same target. Here we extend the vocabulary problem to those situations in which choosing the correct words is not the only problem; in fact, coming up with *any words* to refine the search may be impossible. I call this extension of the vocabulary problem the

vocabulary deficiency problem. It refers to a search where any words may fall short of defining the query, or simply cannot be brought to mind in the first place. The vocabulary deficiency problem is particularly relevant when describing music.

Both the vocabulary problem and the vocabulary deficiency problem allude to Richard Saul Wurman's **information anxiety** [2, page 45]. He defines information anxiety to be exactly the inaccessibility that results from not knowing exactly where or how to look. To Wurman, "the great information age is really an explosion of *non-information*; it is an explosion of data." This vast amount of raw data is not knowledge; in fact, being inundated with it brings about confusion or anxiety in an attempt to make sense of it all. Dealing with information anxiety involves turning data into information, and then getting at the meaning inside of the data in order to understand the patterns within it.

Information anxiety, Wurman offers, can be assuaged by providing different vantage points and organization methods when looking at the same information [2, page 74]. For example, it's helpful to be able to search for businesses in a phone directory based not only on business specialty (e.g. "stationery shop") but by geographic location within your city.

By the same rationale, this thesis aims to approach the large dataset of a personal music library with an application that enables diverse viewpoints and multiple departures for *exploration*. This is particularly appropriate in dealing with music, since it is a good example of data that we tend to have difficulty describing with words.

Systems that focus on analyzing the content of music in order to facilitate searching are called audio-based music information retrieval systems. These systems set out to improve music searching by bridging the **semantic gap in audio content**. That is, they allow a user to search music based on audio content instead of superficial metadata alone.[1]

Semantic queries for music can range from almost non-descriptive (e.g. "songs that sound like this song"), to the descriptive (e.g. "slow songs"), to sophisticated queries for music (e.g. "sad songs that sound like the Pixies"), or query-by-singing, in which the user sings a tune and the computer returns all available recordings of that song [3]. These are the kinds of problems we see addressed in the field of Music Information Retrieval, discussed in the next section.

In this thesis, I use an approach quite like (computer scientist and HCI expert) Ben Schneiderman's **dynamic queries** [4]. Dynamic queries apply the principles of direct manipulation with a visual presentation of query and results; rapid, incremental control of the query; selection by pointing instead of typing; along with immediate and continuous feedback. These visual displays help both novices and experts to formulate complicated queries and comprehend the results. The reason they help users deal with extremely complex data is that humans' highly-developed visual systems can grasp the content of a picture much faster than they can scan and understand text [4, page 75]. Users can recognize spatial configuration and

---

[1] `http://www.sciencedaily.com/releases/2006/01/060106133741.htm`

relationships among elements quickly, recognizing outliers, clusters, and gaps in data sets. Visual interfaces also encourage exploration by revealing relationships and interactions among variables.

I intend to address the vocabulary deficiency problem by using **visualization**. The use of visualization in this project is very much in the spirit of the field of **information visualization**, in that it is dynamic, interactive, and visual, with an emphasis on user understanding and exploration of a large, abstract data set. In using visualization, MusicBox avoids requiring users to focus their musical queries with words, and instead lets a user's interpretation of the map drive his decisions in navigating a musical space.

2.2 Music Information Retrieval

2.2.1 A mountain of music

> *The average 14-year-old can hear more music in a month than someone would have heard in an entire lifetime just 300 years ago.*
> Daniel Levitin, *This Is Your Brain on Music* [5]

Let's look at what has happened to music availability and sales within the past decade. We've seen a rapid growth of the Internet and an audio format, MP3, that paved the way for an explosion of the musical space on the Internet. Since 2001, consumers have shifted away from buying music at traditional music stores and towards digital services and Internet retailers [6]. With peer-to-peer file sharing helping it along, we've seen a dramatic increase in the size of personal music libraries: it is not uncommon for one person to own (and even carry in their pocket!) tens of thousands of songs.

And people are turning to the Web for more. The number of web queries for music had already surpassed those for sexual material by 2001 [3]. People are buying more and more music each year [6], and keeping most of it stored on hard disks at home as storage space gets cheaper.

Music is also being *released* at an amazing rate: In 2007, nearly 80,000 albums were released, up from 30,000 in 2001. And that doesn't even include the several *million* – yes, million – bands on social networking giant `myspace.com`.

With all the musical choice available, what's a user to do? How should one go about finding that next favorite artist? How can we better organize our own growing music libraries?
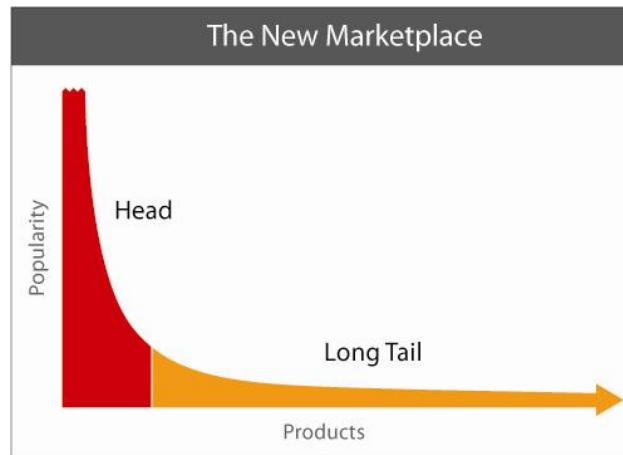
Figure 2-1: The Long Tail: The large number of products in niche markets forms the long tail in the frequency distribution of product sales.

The Long Tail

In the context of business, the phrase "**The Long Tail**" refers to the shift away from a focus on a relatively small number of (mainstream) hits, towards a huge number of unique products sold in much smaller quantities [7]. It's called "The Long Tail" because the large number of products in those niche markets forms the long tail in the frequency distribution of product sales. The quantity of items sold in the tail now rivals that sold in the head. For example, the sum total of books that are only bought once at Internet retailer Amazon.com can easily exceed the sum total of their bestselling books. The phenomenon of The Long Tail has only been made possible in the last decade, as the costs of production and distribution fall.

Music is especially long tail. In 2007, 450,344 of the 570,000 albums sold (80%) were purchased less than 100 times [6].

Since The Long Tail represents such a tremendous amount of potential sales, companies stand to benefit a lot from it, if they can figure out how to tap into it. Chris Anderson, who coined the phrase "The Long Tail" says, "The secret to a thriving Long Tail business can be summarized in two imperatives: 1. Make everything available; 2. Help me find it." [7] The importance of item 2 ("finding it") is paramount: demand for products is fed by giving the customer a range of filters, such as recommendations and rankings, to find their niche products [7]. In this thesis, we'll look at various ways of helping music-lovers find those products.

The User's Dilemma

Music listeners commonly have this task: Sit down in front of the computer and choose something to listen to. This happens to me on a daily basis, and every time I take the same

approach: I start scrolling through my iTunes artist list, imagining what each artist sounds like, and scrolling farther and farther down until I find something that seems like it might fit my mood or activity. If I don't find something, or get tired of scrolling, I put my whole library on shuffle and click "fwd/next track" until something matches. I don't pay attention to genres ("Those are all wrong or weird anyway..."), and I rarely recall the name of an album or a song to bother looking at those lists either. So I'm either engaging in a laborious process of scrolling, or giving up and taking a shooting-around-in-the-dark approach.

Everyone's approach to this problem is different – some carefully maintain playlists for particular tasks or moods, some have a more specific idea of what they would like to listen to, some let web radio stations pick for them. In any of these cases, however, *a smarter system for organizing and presenting music could help tremendously.*

And what about other tasks? Say you'd like to put together a playlist for your friend who is interested in the more "angry music" in your library, or your friend gives you a few hundred new tracks on a thumb drive. How do you get an idea of what's in that collection, or where to start listening to it?

Looking for new music online is another epic task in itself. With the sheer amount of music and music information available, digging one's way through that mountain towards a satisfying musical find can be quite overwhelming.

2.2.2   What is MIR?

I've mentioned **Music Information Retrieval** (**MIR**) a few times in this chapter, so let's discuss what it is.

The field of MIR is a result of the increasing volume of digital music available; its focus is on the new retrieval techniques that that large volume of music necessitates [8]. MIR itself is a large and interdisciplinary field that encompasses music theory, musicology, audio engineering, computer science, law, library science, information science, and business [3].

MIR is challenging not only because the field is so interdisciplinary, but because music itself is complex, subjective, and can be represented in many different ways. Music, by nature, is a form of *expression*, which is realized differently depending on culture, and *experienced* differently by every individual [3]. The subjective nature of music makes it a particularly challenging type of information to deal with.

2.2.3 Goals of MIR

MIR is focused on making the vast body of digital music accessible to anyone [9]. One of its core themes is **discovery**: how consumers discover what they like, and how creators/companies ensure that they get discovered [10, page 125].

Discovery can take many forms from the consumer's perspective: passive, active, and exploratory. We'll talk about each of these in turn.

Passive Discovery: Recommendation

Recommender systems are a type of information filtering that provides users with suggested items of interest. The data that these recommendations are built upon can be actively or passively collected; users may be asked to rate items explicitly, for example, or may have their buying patterns monitored implicitly. Even though the data collection can be active or passive, the recommender system calculates suggestions with no more required input from the user. Therefore, recommender systems are useful in helping users discover items they might not find on their own. This is why I consider recommendation to be a *passive* form of discovery.

Recommender systems generally take two approaches in making their recommendations, depending on whether they focus their recommendations on *similar customers* or *similar items*. Systems that focus on similar customers compare customer ratings, and make recommendations like "Customers who rate this band highly also rate this band highly...". Item-based recommendation systems, on the other hand, require a calculation of item-to-item similarity. This could be done by analyzing what items customers tend to purchase together, as Amazon.com does [11] ("Other customers who bought this item also bought..."), or by looking at the features of the items themselves to see which they have in common.

Both customer-to-customer and Amazon.com's item-to-item comparisons are examples of **collaborative filtering**, which filters data based on analyses of user tastes, however realized. (Item-to-item similarity calculated based on the *content* of an item, though, is not considered to be collaborative filtering.)

Recommender systems can recommend any kind of object, including music. Some popular recommendation systems for music today are: Pandora[2], Last.fm[3], Amazon.com[4], and MyStrands[5]. Pandora provides customized webradio based on item-to-item (track-to-track) similarity; these similarity measures come from a list of several hundred manually-assigned characteristics (e.g. "Aggressive Drumming", "Defiant Lyrics", and "Shifting Beats") [12] along

---

[2] `http://www.pandora.com/`
[3] `http://www.last.fm/`
[4] `http://www.amazon.com/`
[5] `http://www.mystrands.com/`

with some community feedback[6]. Last.fm uses traditional collaborative filtering of user listening data. MyStrands also does collaborative filtering, using users' playlists and play histories[7], then providing many options to customize the recommendations made (for a particular decade or popularity level, for example).

Recall the idea of The Long Tail, discussed in Section 2.2.1. The main challenge in making The Long Tail accessible in recommendation systems is that they primarily rely on user-created data, and items in The Long Tail don't have that data, since there may never have been a previous customer for that niche product (see Section 2.2.4, for more on the "cold-start problem").

As The Long Tail gets longer and fatter, recommender systems must focus on dealing intelligently with that available data. This necessitates integrating automatically-applied, content-based techniques that will be able to recommend items even if no human being has ever listened to them.

Active Discovery: Searching

I use the term "searching" here to refer to a situation in which a user is actively seeking something, with at least an idea of what it is they are looking for. In music search, this could mean seeking a song based on a few of the known lyrics, finding an album released by a particular artist in a particular year, or looking for the title of a song when the melody is known.

There are many tools to help with these kinds of tasks, a few of which I will only mention here. Many music searchers look for the name of a track or artist by typing some of the heard lyrics into Google's[8] search box. Music mega-data websites, like Allmusic.com, provide all kinds of discography information, album art, and song title searches if you have enough information to find it there. And query-by-humming systems like Midomi[9] record a user humming/singing a melody and compare that recording against a database to provide results.

These methods are appropriate for more directed, fairly precise searches. But what if the user doesn't know exactly what they are looking for, whether it exists, or what it sounds like? At this end of the extreme, discovery becomes completely exploratory, which is this thesis's focus and the topic of the next section.

---

[6] `http://tomconrad.blogspot.com/2006/05/big-change-at-pandora.html`
[7] `http://www.mystrands.com/mystrands/whatis/recommendations.vm`
[8] `http://www.google.com/`
[9] `http://www.midomi.com`
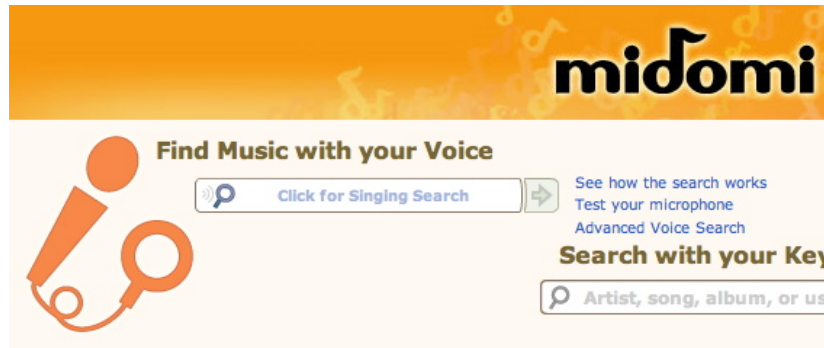
Figure 2-2: Midomi allows users to sing, hum, or whistle to search for music

Exploratory Discovery: Browsing

Music seeking can be motivated by curiosity rather than actual information need. In fact, music enthusiasts often *derive great pleasure* from the information gathering experience alone, without expectations for what they might find. This information-seeking activity can even be addictive [13].

I refer this type of exploratory discovery – deriving pleasure simply from the searching experience, with the hope of serendipitous discovery – as "**browsing**". It may not sound flashy, but browsing is exactly what we do when we want something but don't know what it is. Laplante and Downie, in a 2006 survey of music information-seeking behavior [13], suggest that MIR systems should shift towards offering music enthusiasts a variety of browsing facilities *focused on discovery and novelty*, instead of simply perfecting search algorithms.

While traditional forms of music browsing, like reading reviews, talking to record store employees, and going to concerts, can be tremendous discovery tools, this thesis focuses on how MIR systems can help in the discovery process, both by providing flexibility and by incorporating new types of data. Websites like Allmusic.com give artist biographies along with lists of similar artists, influences, and moods, filled with hyperlinks to allow the user to navigate in many different ways, but that is just the beginning. Chapter 3 discusses a variety of examples of MIR systems that facilitate exploratory discovery in innovative ways. With this thesis, I create a flexible, extensible discovery tool for exploring a musical space from many different vantage points.

2.2.4   Computing similarity

Many MIR systems revolve around computing **audio similarity**, which is a complicated and subjective matter [14]. A system might compare two pieces of music, two artists, or two genres, but it's always about determining how similar they are. If you want to make recommendations,

or show music organized in a customized fashion, there are particular features you can use to characterize the music. This section looks more deeply into how MIR systems calculate similarity.

Types of data

There are two main types of data to consider here: **content-based** (or audio-based) data and **contextual** data.

Music data can come directly from the audio signal, or from a more subjective characterization of that signal. For example, the tempo of a piece can be determined by audio analysis to be an exact number of beats per minute. A human, on the other hand, might characterize a piece with a low tempo as "slow" or even "mellow/sad." Looking at music from both of these vantage points is important when building systems for humans to interact with music.

We call the quantifiable characteristics of the audio signal **content-based descriptors**. These could include low-level descriptors like loudness, timbre, pitch, and time segmentation, as well as higher-level descriptors like rhythm, form, and melody. Researchers have had varying degrees of success in creating applications that can characterize these features automatically [8]. For example, loudness is easy to calculate (it's a simple amplitude measure), but melody can be hard to separate from a polyphonic recording since signal separation is currently unsolved; timbre might be characterizable with numbers (e.g. as a frequency distribution), but instrument identification might be lacking. Picking out these low-level features is an active area of research, and its continuing development means that the data we extract with computer programs is by no means guaranteed to be error free.

Content-based descriptors help us get at *what the music actually sounds like*, which ought to be of some importance. But they are not enough on their own. As Yoshii *et al.* point out in their 2008 paper on hybrid music recommendation, "Similarity in content is only one of many factors characterizing user preferences. [15]" After all, a listener's experience of the music is not an objective process, determined from some simple quantifiable characteristics. The term **contextual descriptors** refers to the listeners' contextual interpretation of an acoustic signal, that is, the words that people use to qualitatively describe musical sound, collected from online reviews, discussion, and tagging of music [16]. They include "everything about the music that is missed in the signal" [17]. Contextual features like ratings and tags are the raw data used in the collaborative filtering techniques described in Section 2.2.3. Some of the more innovative systems, such as The Echo Nest's "Musical Brain"[10], go so far as to "listen" to the online chatter about music in order to draw connections between music and words; this work was pioneered by Brian Whitman in his PhD work at the Media Lab [16].

---

[10] The Echo Nest's "Musical Brain" reads about music, listens to music, and learns about music trends by scouring the Internet for music-related text (`http://the.echonest.com/company.html`).

Data type	Musical facet	Examples
Content-based	Pitch	melodic contours
	Temporal	tempo, meter... rhythm
	Harmonic	harmony
	Timbral (almost Editorial)	tone color, orchestration
Contextual	Editorial	performance instructions, improvisations
	Textual	lyrics, libretti
	Bibliographic	title, composer, publication date, album art

Table 2.1: Musical facets by data type: Stephen Downie's seven musical facets [3], categorized by type. The line between content-based and contextual categories is blurred; oftentimes the timbral facet can be considered editorial.

Content-based and contextual descriptors each have their own advantages and disadvantages. Content-based descriptors, as I mentioned, may not be easy to extract from the audio signal. They also simply do not characterize any part of the human *experience* of listening to music.

Contextual descriptors, on the other hand, rely on human labeling of music, which is error-prone itself, and relatively time-consuming to apply. Because music is released much faster than we humans can attempt to classify it, contextual data is not easily scalable to be applied to all of music. For example, webradio service Pandora's professional listeners must spend 20-30 minutes per track to classify it according to their several hundred descriptors [10, page 108]. Because of this, Pandora can only afford to classify music that it prioritizes for recommendations, which corresponds to only the most popular music. Therefore they neglect the entire Long Tail of music, even as it becomes more and more accessible.

This neglect of the Long Tail of music is related to the so-called **cold-start problem**. In the process of recommendation, the cold-start problem refers to a sparseness of data available for new or unused items, which limits their recommendability. Collaborative filtering recommenders require a significant amount of data before they can make decent recommendations [18]. When new or unpopular music is released, usage data and ratings are not available, so the music cannot be recommended, even if it is in fact the most appropriate recommendation.

Because of the limited applicability of contextual data, collaborative filtering recommenders tend to recommend the same pieces over and over to a user, and often these pieces are well-known [15]. To a music enthusiast, this can be particularly irksome; it's inherently important to them to discover new artists *before* the music has become popular. Recommender systems that make effective use of *content*-based data can help make up for this shortcoming.

One of the greatest values in contextual data – its subjectivity – is also its fault. Because descriptors like "happy" and "fast" are subjective, their assignment is inconsistent. A particularly popular contextual descriptor, **genre**, is quite limiting: the categorization of one

song is generally limited to only one genre, and that label may be incredibly imprecise (e.g. "rock" music). Social tags, on the other hand, are not limited to a single assignment per song, and therefore can make up for some of what genres lose. Nevertheless, people are notoriously inconsistent about assigning genres and tags [19], so MIR systems must take care when using this kind of data.

Ultimately, *a combination of content-based and contextual descriptors makes the best MIR systems* [19, 20]. Many approaches, as we will see in Chapter 3, use content-based features to organize the music and contextual data to enrich the display of the music (e.g. labeling songs with metadata). But this thesis aims to truly combine both content-based and contextual features, allowing both of them to contribute to the organization of music in a space.

How to calculate similarity

No matter what type of data an MIR system uses, or whether the system is built for browsing or recommendation, the data still has to be combined to create a similarity measure. This can be done explicitly, where similarity is defined between two pieces, or implicitly, where similarity is the result of items having similar feature data. For example, an explicit similarity measure could be created by asking users how similar are pairs of songs or by looking at which bands have overlapping band members. (This is the kind of reasoning used in collaborative filtering, discussed in Section 2.2.3.) Implicit similarity measures, on the other hand, could come from analyzing a body of music for tempo, and finding which pieces have the closest tempos. The explicit similarity approach tends to be more frequently used with contextual data; the implicit approach is used more frequently with content-based data. In this thesis, we use an implicit approach that combines both contextual and content-based data.

Calculating similarity is an inherently tricky process. There is no ground truth: Two people can easily disagree on the similarity of two pieces. (For instance, one might focus more on instrumentation, and the other more on rhythm.) When judging a system's success at characterizing audio similarity, MIR researchers often use more objective tests like genre classification and artist identification. Some of these studies have shown significant subjectivity in genre annotation by humans, and although automatic genre classification has improved over the last decade, it still lags behind human performance on the same task [21]. Therefore, even state-of-the-art machine-assigned music similarity is still rather primitive.

Once similarity measures (or the features from which they are inferred) have been collected for a body of music, researchers combine them into a model for that music by using techniques that emphasize the relationships between the pieces of music. These techniques include dimensional reduction methods like multidimensional scaling and principal components analysis, along with clustering algorithms like self-organizing maps. The technique chosen depends on what kind of data is to be used, as well as the designer's goal in creating the model

(e.g. to see tight musical clusters or to simply order tracks by similarity). We'll consider these methods in more detail in Chapters 3 and 5.

Audio features

In this thesis, I refer to the content-based and contextual descriptors introduced in this chapter simply as "**features**." A feature could be tempo, popularity, rhythm, mood, etc. In the context of MusicBox, a feature is basically a number or a class (e.g. genre, tag) assigned to each track in a music collection.

Sometimes features occur in **feature sets**, which only make sense together. For example, Thomas Lidy's Rhythm Histogram feature [22] comprises 60 numbers, which taken together are the values for bins in a histogram of 'rhythmic energy' per modulation frequency. In a sense, the Rhythm Histogram is 60 features.

Content-based features commonly used for music classification in MIR are (a subset of these are included in the MusicBox project):

- **Tempo** – Speed or pace of a piece of music. In this context, tempo is usually indicated in beats per minute (BPM).

- **Timbre** – A description of "color" or quality of a sound, as distinct from its pitch and intensity [23]. Simply put, timbre is the characteristic that makes two different instruments sound different even when they play a note at the same pitch and intensity. There are a variety of ways to characterize timbre; in this thesis, we focus on a method determined by a principal components analysis. See Section 5.2.1 for more detail.

- **Spectral features** – Statistics describing the power spectral density of a sound sample. These are basically measurements that summarize timbre. Characterizations include spectral centroid (balancing point of the spectrum) and moment (shape of the spectrum) [24].

- **Mel-frequency cepstral coefficients (MFCCs)** – A set of coefficients often used in speech processing to describe the pitch and timbre of a sound. They are derived from an audio selection's power spectrum on the so-called mel scale of frequency (which is specifically tuned to approximate the response of the human auditory system). Because MFCCs have been used extensively in speech research, they have naturally been adopted by the MIR community to describe musical content.

- **FFTs** – Listing this term as a feature is perhaps misleading, because it refers specifically to an algorithm rather than an audio characteristic. However, the term "FFT" is often used interchangeably with the audio characteristic it computes. An FFT is an efficient algorithm for computing the discrete Fourier transform, which transforms a sampled

signal over time into a frequency domain representation of the sound. Therefore, FFTs are used to calculate the frequency spectrum of a sound selection. FFTs are used as a step in extracting many other frequency-related audio features (such as MFCCs above, and the timbre characterization we use in MusicBox).

- **Tonality** – The harmonic effect of being in a particular key [25]. Some recent academic treatments of tonality are presented in Section 4.6.

- **Loudness** – The perceived intensity (/amplitude) of a sound, and variations in that perceived intensity, in an audio selection [26].

- **Rhythm** – Arrangement of musical beats, defined by duration and stress. Rhythm can be thought of as the repetition of a beat or the flow of the music.

- **Pitch** – The perceived frequency of a sound. This is not a simple characterization of the frequency of sound, because the human auditory system can perceive the overall pitch of sound differently from, say, the fundamental frequency of that sound, depending on the overtones present.

- **Harmony** – The simultaneous use of different pitches to create chords. Harmony can also describe the structural progression between chords.

- **Segmentation** – The decomposition of an audio selection into meaningful parts. These segments can be thought of as "sound events", which when applied to a musical sample, often correspond to individual notes. Segmentation at the note level can be based on pitch, timbre, and energy [24]. A "segment" can also refer to a sequence of notes that makes up a section of a piece. Segmentation at this level can be based on pitch, timbre, rhythm, and articulation [27].

- **Duration** – Simply the length of a musical selection, in seconds.

Defining the sound sample

Last but not least, we must define the part of a musical piece characterized by a feature. Do you describe the timbre of a piece as the timbre of its most dominating instrument, or an overall timbre? Do you describe a song's rhythm as a whole, or divide the song into segments within which the rhythm is consistent? Can you succinctly describe an artist's entire career style, or must it be divided by decade, by album, or even track?

One typically chooses between the sound sample at these levels: artist, album, song, or song segment. In this thesis, analysis operates *at the song level* whenever possible. In developing MusicBox, I chose the song rather than a song segment because segmentation is complicated, and users don't usually listen to less than a track at once. I chose the song rather than the album

or artist because many artists' careers and albums are acoustically diverse, so to classify them so simply would be unjust [28, 12]; in addition, more and more music is being bought (and listened to) one song at a time.

Sometimes, data at the song level is limited. For example, many of the contextual descriptors combed from the web, such as moods assigned by Allmusic.com editors, are applied at the artist level. In cases like these, we take what we can get, and use artist assignments to characterize all the artist's songs.

Chapter 3

Related Work – Looking at Music Libraries

In this chapter we look at different ways to visualize and interact with a body of music. We will move from heavily text-dependent interfaces typical of mainstream music browsers, into the realm of music visualized in a space.

3.1 Text-based views

3.1.1 Mainstream music browsers

iTunes, WinAmp, and Windows Media Player are the most commonly used music browsers today. And surprisingly, even though music libraries have changed (grown and have so much more metadata available) so rapidly in the past 10-15 years, these browsers have not changed much at all. What changes we *have* seen have been due to a change in music distribution (e.g. iTunes Music Store), instead of better, more flexible means of interacting with the music that we purchase.

It's clear that these media players, pictured in Figure 3-1, all employ a heavily text-dependent interface. Songs are most often represented as rows of artist name, album name, song name, along with several other pieces of metadata from either the ID3 tags (e.g. genre, track length) or the application itself (e.g. play count, date added, user rating). Users can sort the library based on these metadata, but in many cases can't even filter the library based on the data. What little flexibility is afforded comes in the form of genre classifications, which the user can assign one per track if they wish.

Other features typical of mainstream music browsers include: playlist support, text-based search, and shuffle.
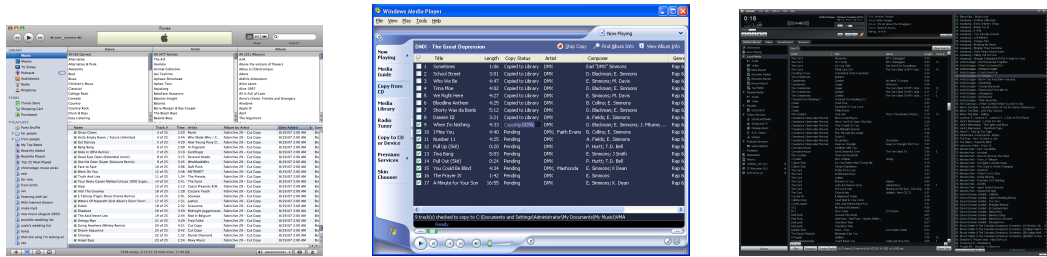
Figure 3-1: Mainstream music browsers. From left to right, iTunes, Windows Media Player, and WinAmp.

Thus, user interactions with music libraries in these browsers is limited to:

- looking at text (the only visual is album art)

- sorting with textual attributes

- filtering data based on textual attributes (or genre)

- creating and playing textual playlists

Text-based interactions like these are appropriate if a user knows what she is looking for, or knows what a piece of music sounds like based on the text that represents it. But that assumption is foolish and impractical, especially as music libraries are getting so large that only the greatest music aficionados would be able to keep track of it all. Music is such an enormously *rich* data set, but here it's been reduced to a few textual terms. The *relationships between the pieces* has been neglected entirely.

I propose that users should be able to search their library based on more flexible text data that they do not have to enter manually, and to search their libraries based on the audio content of the music itself. These categorizations need not be strict: the relationships between tracks in a library is necessarily fuzzy, and the interface should reflect that.

3.1.2 The CUIDADO Project

With the vast amount of music and musical metadata available online, in addition to the profits to be gained from well-made suggestions, **recommendation** has become one of the most active fields of research in MIR. We will see several music recommendation projects in this chapter.

One of the earliest (and largest) undertakings in music recommendation was the CUIDADO Project, funded by Information Society Technologies (IST) for a multi-lab collaboration between
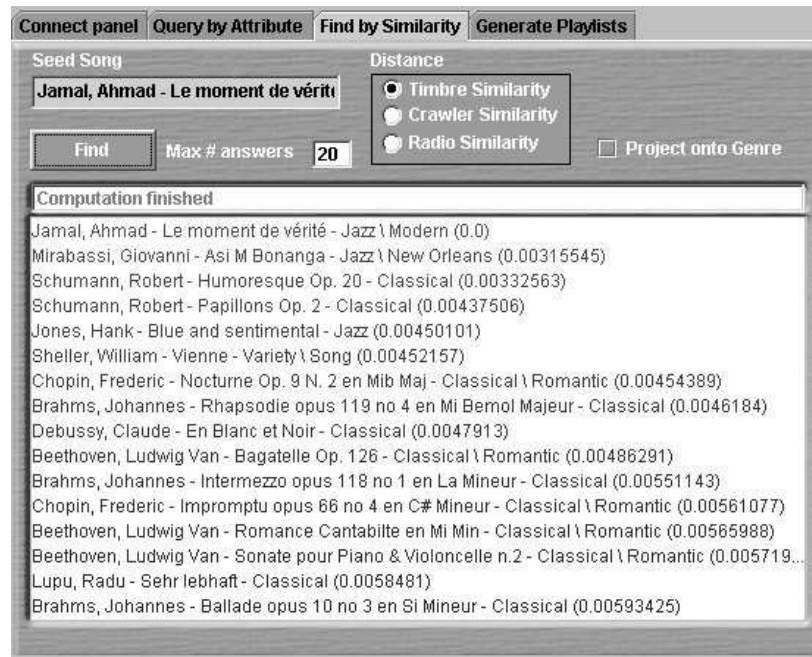
Figure 3-2: CUIDADO Music Browser: "Find by Similarity"

January 2000 and December 2003 [29]. One of the most active participants in this collaboration was Sony Computer Science Laboratories (CSL) in Paris. The project addressed research issues that included feature extraction, and design and implementation of two systems to demonstrate the use of these data to both personal and professional consumers.

The two systems developed were a music browser (the "Music Browser") and a sound production application (the "Sound Palette"). I will focus on the Music Browser here because it is the most applicable to this discussion. "The Music Browser is intended to be the first content-based music management tool for large music catalogues" [30].

The CUIDADO Project developed tools to extract appropriate low-level descriptions of the audio signal, derived high-level information from those descriptions, and combined that data with collaborative filtering data to get artist and track similarity ratings [30]. The Music Browser used these similarity ratings along with extensive editorial metadata to create some very sophisticated descriptor-based library queries (Figure 3-3), artist similarity searches (Figure 3-2), and playlist creation tools (Figure 3-4).

Even though CUIDADO's Music Browser provided highly flexible and powerful search tools, it was still a highly text-based interface. Such an abundance (and variety) of data that is only searchable with text can easily seem overwhelming to the user. Later in this chapter we will see some other approaches that represent artist similarity without depending on text, and that emphasize the relationship (similarity) between tracks more than the underlying metadata itself.

35
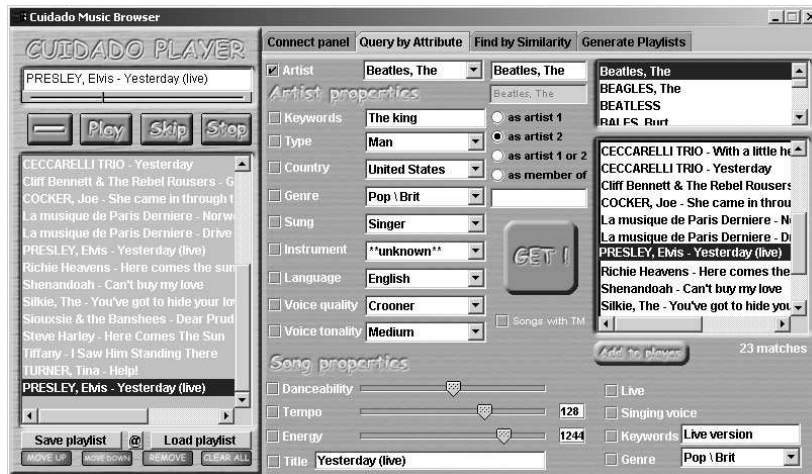
Figure 3-3: CUIDADO Music Browser: Query panel

Figure 3-4: CUIDADO Music Browser: Playlist generation system

3.1.3 MusicLens

MusicLens[1] is a very simple website for searching the `finetunes.net`[2] music database. The interface allows you to search by a set of criteria for which each song in the recommendation database has been manually-assigned. These criteria include tempo, voice (instrumental or not), volume, male/female, age of release, and mood. Visitors set these criteria with a set of sliders, and MusicLens presents the results below, ready to listen and ready to buy.

---

[1] `http://finetunes.musiclens.de/`

[2] `http://www.finetunes.net/` is a major online distributor of independent music in Germany

36

The designers of MusicLens allude to the aforementioned vocabulary problem[3]: "[MusicLens] delivers concrete results even when the search information is vague."

NR.	TITEL	ARTIST	ALBUM	GENRE	PLAY	MORE	GET
1	Hoffentlich	Klee	Unverwundbar	Pop/Electronica	▶	□ » 🗗	SHOP
2	Daily News	Amanda Rogers	Daily News	Pop/Singer/Songw…	▶	□ » 🗗	SHOP
3	enjoy the silence	Dave Gold	Freestyle 80's	Breakbeat	▶	□ » 🗗	SHOP
4	Because The Night	Lollies	Lollies Live	Schlager	▶	□ » 🗗	SHOP
5	Tu eres ajena	Javier Vega	Vamos! Vol.14: Bachata		▶	□ » 🗗	SHOP
6	Sleep	Amanda Rogers	Daily News	Pop/Singer/Songw…	▶	□ » 🗗	SHOP
7	Tonight	Stella	Better Days Sound Great	Pop/Electronica	▶	□ » 🗗	SHOP
8	HAU DOCH AB	MATTHIAS CARRAS	ZÄRTLICHER REBELL		▶	□ » 🗗	SHOP

Figure 3-5: MusicLens

MusicLens is an elegant and relatively flexible interface for buying music, and its success[4] reflects users' need for searching music databases very specifically, and not just at the artist-level, but at the track-level. Still, it hides a lot of the data from the user, so that there is not much room to *explore* (and interact with) other than moving sliders around. Even though searching the MusicLens interface doesn't involve typing textual search terms, users still are only searching the database with text-based terms; they just set them with sliders instead of typing them.

MusicLens is different from most of the others in this chapter because it gives the user such fine control over its recommendations, but it does not visually represent the recommendations or the relationships between them. In addition, it relies on manually-assigned attributes for each track/album, and I'd like to see a move towards automatic attribute assignment.

3.2   Moving away from text: Simple networks with explicit connections

So where does this leave us? How can we make the rich data set of music more interactive, more flexible, and more intuitive for its users?

---

[3] http://www.ddd-system.com/pdf/musiclens_engl.pdf

[4] MusicLens won a journalist-juried competition for innovative music recommendation systems in 2004:
http://www.ddd-system.com/presse/news/?topic=winner_reco

There are many projects in the field of Music Information Retrieval that integrate audio and descriptive data into applications for recommendation, organization, and exploration. These projects build upon existing work that focuses on retrieving important musical characteristics from the audio signal, gathering user-created textual descriptions [16], and collecting users' listening patterns[5]. These data-collection projects have resulted in huge amounts of metadata surrounding the world of music, and it is only growing. *The important problem moving forward is how we will use that data to help us interact better with our music.* More complicated text-based interfaces are clearly not the answer; we need something better.

One way to display the remarkable depth of information available is to depart from text-based lists and present the music collection *in a space*, with distances (and visual representations) reflecting the relationships between (and characteristics of) the elements of the data set. If the visual representation is created well, users can glean much more from these images than they would from text. We can take advantage of that representation to present much richer data sets of music.

The following projects reflect that departure from the text world into the visual world.

3.2.1 LivePlasma

Figure 3-6: LivePlasma

Frederic Vavrille released LivePlasma[6] to the web in 2005. Liveplasma uses the data from Amazon.com's API for item-to-item similarity to create maps of musical artist similarities, with connections between the artists to highlight relationships. These item-to-item similarities are computed by processing which items customers tend to purchase together [11].

---

[5] http://www.audioscrobbler.net/
[6] http://www.liveplasma.com/

Liveplasma users enter the name of an artist, and LivePlasma presents them with a map of similar artists. Users can search, create more maps, discover new artists, as well as save and share their maps.

With LivePlasma, one of the biggest limitations of collaborative filtering algorithms becomes clear: in order to receive recommendations, a user must seed the system with an artist the system already knows about. For a system like Amazon.com, who makes recommendations based on what users are already looking at or buying, this approach makes sense, but what about browsing music when not only are you looking for music with a fresh and new sound, but you cannot easily describe that sound? A system like LivePlasma, which does not have any representation of the audio content of the music itself, cannot make these kinds of recommendations.

3.2.2 Tuneglue
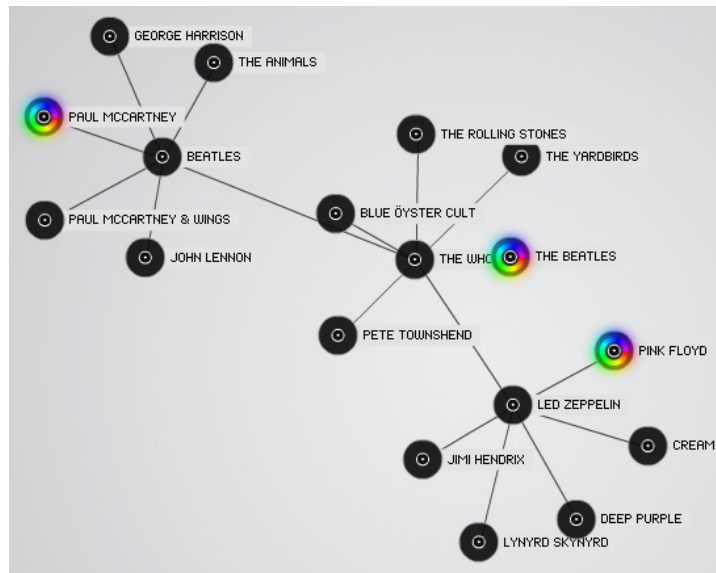
Figure 3-7: TuneGlue's Relationship Explorer [31]

A similar project is Tuneglue [7], which uses data from both Amazon.com (for user purchases) and Last.fm (for user listening history). Tuneglue creates a springy network of artists based on similarity. Users provide a seed, starting artist, and can explore the resulting map by expanding any artist node. Release data is displayed, but not much else.

Tuneglue is limited in much the same way that LivePlasma (3.2.1) is; it is only helpful if you can first seed the system with an artist of interest.

---

[7] http://audiomap.tuneglue.net/

3.2.3 Musicovery

In 2006, Frederic Vavrille (3.2.1) expanded his Liveplasma technology with a service called Musicovery (`http://musicovery.com/`). Musicovery puts the browser online to create interactive, customized webradio. It is an Adobe Flash-based site that still does all of the things that Liveplasma did – displaying similar artists as a network – but has one particularly notable new feature: a pad in which the user can select a *mood* location by which to search the music. Musicovery displays pertinent results in a star-like network of songs, and streams music to accompany.
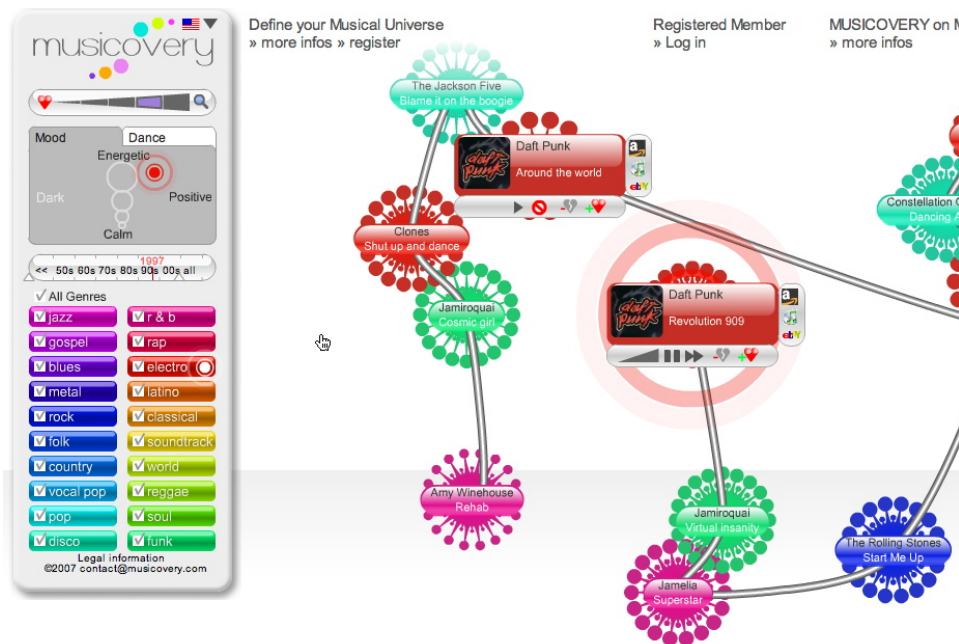
Figure 3-8: Musicovery [32]

Musicovery allows filtering by mood, danceability, popularity, year of release, and genre. It's a good example of the increasing tendency of online music sites and browsers to provide the user a new means by which to search music. The **mood** approach is one of pure simplicity: the underlying engine has classified all tracks by two important qualitative characteristics that can be represented along just two axes. The user simply has to say, "I want something a little energetic, and a lot dark" and the closest artist/tracks are displayed.

Several classic models of mood, such as James Russell's valence-arousal model [33] or Robert Thayer's energy-stress model [34], classify moods based on their location along two axes. In Russell's valence-arousal model, an individual's mood is characterized by pleasure ("valence") and activity ("arousal"), which are (not surprisingly) directly related to the axes used in Musicovery's mood pad.

40

One of the earliest online music recommendation systems, MoodLogic, approached the problem of mood assignment by allowing users to collaboratively profile the mood of their music. The MoodLogic system would then use these mood assignments to create mood-based playlists for its users. Although this user-driven system worked for MoodLogic's first few years, its successors have explored more automatic routes.

The mood of a piece of music can be inferred from simple audio characteristics of the piece, such as mode, tempo, pitch (register), rhythm, harmony, and melody [35]. Although it's not clear how Vavrille achieved the music-to-mood mapping, it's likely that he processed the music, extracting features like tempo, pitch, and rhythm, and automatically mapped the music onto the valence-arousal space. (See [35] for more information on musical mood assignment and the feature extraction necessary to do it.) Because the mood of music can be automatically inferred, and because it provides an intuitive way for users to search for music, it's bursting up as a feature in many new music browsers.

Because Musicovery's mood pad eliminates the requirement that users start exploring with a specific artist of interest, it gives users a springboard into its music space while still dealing nicely with the vocabulary problem. However, it appears that Musicovery's music engine only uses item-to-item similarity based on users' buying patterns; its recommendations could be greatly enhanced with the addition of some content-based descriptors for its music.

3.2.4 Barcelona Music & Audio Technologies (BMAT)

Barcelona Music & Audio Technologies (BMAT) is "a commercial spin-off company of the Music Technology Group (MTG)." MTG is a division of the Universitat Pompeu Fabra, "the world's largest research group in music and audio." [8]

BMAT's discovery tools, like Musicovery (Section 3.2.3), combine moods and audio characteristics to create recommendations for its users. Until early 2008, their website[9] allowed visitors to create artist "Similarity Maps" for popular artists. In the recent commercialization of their recommendation system as an application web service[10], the company seems to have moved away from allowing users to explore artists freely and instead has turned towards interfaces that encourage mood searches (like "party" and "happy") to stream song samples.

BMAT's services combine audio and contextual data (from collaborative filtering). Their "unique recommendation component"[11] creates perceptual and emotional musical descriptions from the audio itself, including inference of genre and mood. This descriptive

---

[8] http://bmat.com/about_us
[9] http://demos.bmat.com/songsphere/ms/
[10] http://www.bmat.com/docs/ella.pdf
[11] http://bmat.com/discover

model enables more complicated searches like "what are the bands from China which sound like Chet Baker" or "I want relaxed jazzy music"[12]. This is exactly the kind of free-form search users need when looking for new music.
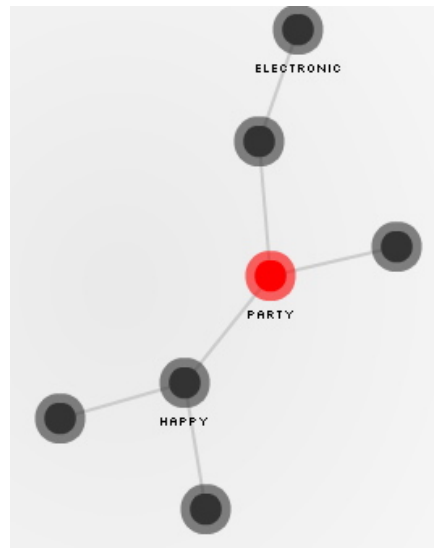
Figure 3-9: BMAT Music Innovation: Discovery tool [36]

3.3 "Geographic" maps

Now that we've seen some simple networks representing musical artist similarities, let's take a look at some more complicated spaces that organize music at the track-level.

The projects in this section build on a geographic metaphor: your music collection can be mapped just like a city/landscape. Neighboring items on the map are acoustically similar pieces of music.

While the interfaces described in the previous section exemplify simplicity in displaying artist relationships and quickly providing users with new (and closed set of) music recommendations, the interfaces described in this section focus more on making users *explore* the relationships between pieces of music and the specific features that characterize them.

These map-like representations encourage more free-form *exploration* of music libraries, and specifically illustrate the *fuzzy relationships* between pieces of music in the library.

---

[12] http://www.bmat.com/docs/ella.pdf

3.3.1 Islands of Music

Elias Pampalk developed the "Islands of Music" visualization of music archives for his Masters Thesis at the Vienna University of Technology. In it, pieces of music are automatically placed on the map according to their audio characteristics [37, 38]. The groupings thus form islands of music which represent musical genres or styles. Within each of these islands are smaller mountains and hills for sub-genres. Genres that are very similar are represented as nearby islands, whereas very different genre islands may be separated by a deep sea.
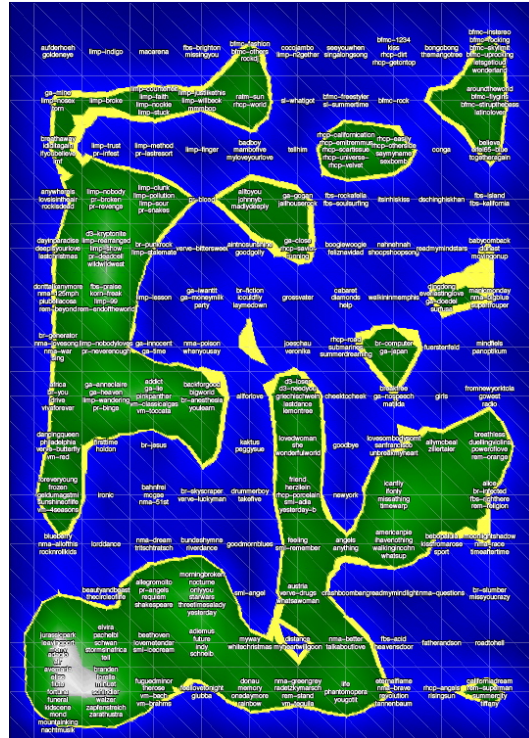
Figure 3-10: Islands of Music

Islands of Music places songs in their locations by using a **self-organizing map (SOM)**. SOMs are used in every project in this section. They are unsupervised neural networks that provide a mapping from a high-dimensional space to a lower-dimensional space (often two dimensions) [39]. The output of a SOM is a mapping that tends to cluster together items with similar feature values. A SOM can be trained on all kinds of feature sets.

Pampalk says that "Islands of Music are intended to support exploration of unknown music collections."[13] As we branch out from simple networks and text-based interfaces into music libraries viewed in a free-form space, we are afforded the opportunity to explore less familiar

---

[13] http://www.ofai.at/~elias.pampalk/

music collections, because these spaces can illuminate the relationships and hierarchies present in the music library.

3.3.2 nepTune

Knees et al. developed a 3-dimensional version of Pampalk's Islands of Music, called nepTune [40]. nepTune also uses a SOM trained on audio similarity to place music on a virtual landscape.

Figure 3-11: nepTune [14]

nepTune has some exciting new features, however. It integrates a surround-sound system with continual playback of the pieces surrounding the user, in order to create an immersive experience much more like virtual reality. As the user moves (flies, rather) through the space, they hear the songs surrounding them, specifically localized to their source. In addition, nepTune incorporates contextual data from the web (like tags and album art) to enrich the space. Addition of contextual data on top of audio-based maps is a theme that will continue throughout this section.

This kind of interface permits and encourages the creative exploration of a body of music, and enables serendipitous discovery within large collections.[15]

3.3.3 PlaySOM

Researchers at the Vienna University of Technology, where Elias Pampalk developed Islands of Music, built on his metaphor with a project called PlaySOM. PlaySOM is much like Pampalk's Islands of Music, as well, yet it adds several modes of interaction that that project was lacking [39]. "The PlaySOM and PocketSOMPlayer applications both enable users to browse collections, select tracks, export playlists as well as listen to the selected songs."[16] PlaySOM has enough features that it can act as a substitute for a traditional media player.

---

[15] http://www.cp.jku.at/projects/nepTune/
[16] http://www.ifs.tuwien.ac.at/mir/playsom.html
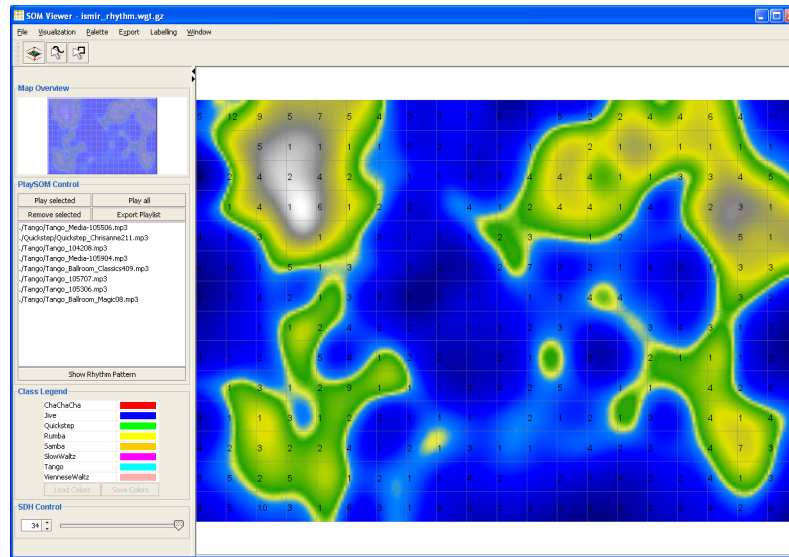
Figure 3-12: PlaySOM

PlaySOM also allows for different visualization modes for musical properties like bass, deviation (variance), and class information (like genre tags) (Figure 3-13). The ability to move between many views helps the user enhance their understanding of the space.

Figure 3-13: PlaySOM alternate views. From left to right: bass, deviation, and class.

The Vienna University of Technology addressed another interesting interface problem with PlaySOM: how much information to show the user at a given location/context. In PlaySOM, the zooming level influences the amount and type of data displayed [39]. These data range from high-level genre information to much more detailed text-based information per track when zoomed in farther.

3.3.4 Globe of Music

The Globe of Music takes an even more literal approach to the geographic metaphor, and uses a Geographic Information System (GIS) to display a music library on a sphere. Song locations are determined by employing a spherical self-organizing map that operates on acoustic feature vectors for each song. [41]

Figure 3-14: Globe of Music

The creators of the Globe of Music state that the model gives users a good impression of the variety of songs in a library (test libraries consisted of about 300 songs). More importantly, their users accepted the new visualization readily and found it *fun* to interact with.

The Globe of Music has several drawbacks. One is that it forces music into evenly-spaced groups on the globe, so that intricate relationships between the musical pieces are hidden from view. In addition, it seems that the Globe's design is not easily expandable to view more than a few hundred songs, and the globe view itself does not give a clear impression of any underlying structure without further user investigation. Moreover, Globe of Music limits the user to spinning the globe about one axis [41], which in this case doesn't necessarily have any real meaning.

3.4   More ambiguous spaces

While the geographic metaphor we saw in the previous section is clearly useful for looking at musical libraries, I'd like to loosen up the metaphor a bit, and move into some more ambiguous spaces. More flexibility means more dimensions, more types of relationships, and perhaps more interesting interactions.

3.4.1   music-map

Gnod's music-map is a good introduction to the idea of user-generated data influencing a spatial organization of music. Gnod (`http://www.gnoosic.com/`) asks every visitor to its site to name three favorite bands. It then uses its previously-collected knowledge from other users to make some new suggestions. When presented with these suggested bands, the visitor can provide

feedback on whether they are appropriate or not ("I like it" / "I don't like it" / "I don't know it"), and the Gnod database (gathers even more data) updates to reflect this feedback.

The "music-map" is an artist similarity map built on top of the user-fed Gnod database. For any artist that Gnod knows about, it will create a map of that artist in relation to other artists it knows about. "The closer two artists are, the greater the probability people will like both artists." [17] Thus, this is an implementation of item-to-item collaborative filtering (where artists are the items).
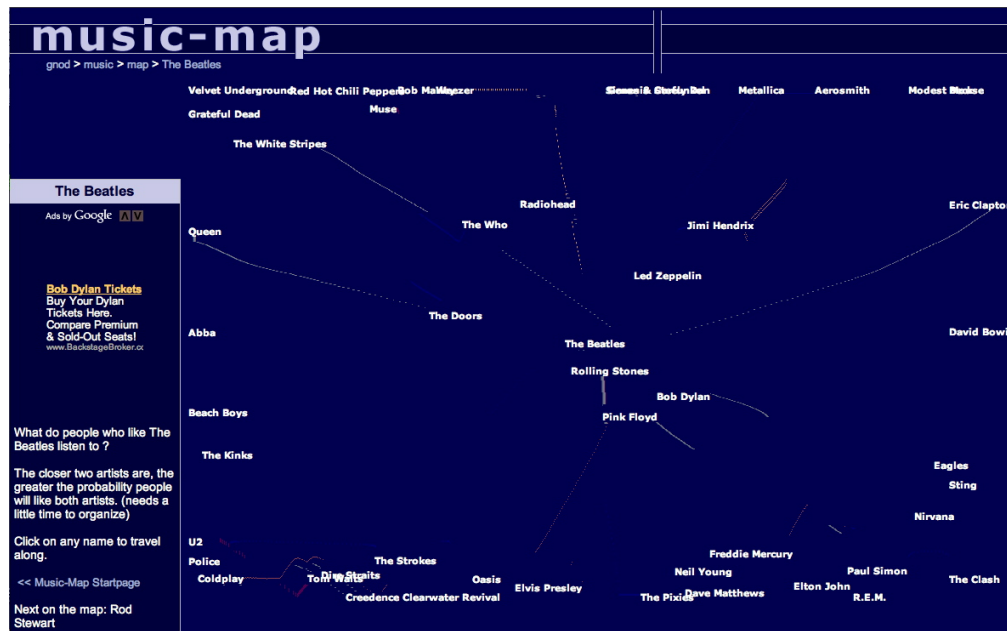
Figure 3-15: Gnod's user-fed "music-map"

Users can click on new artists in the map and explore further, but no sound samples or other data is provided, so the information is a bit limited. As of the time of this writing, Gnod is an active website (`http://www.gnoosic.com/`) collecting preference data from users.

3.4.2 Justin Donaldson / MyStrands

At social recommendation and discovery company MyStrands[18], Indiana University PhD student Justin Donaldson works on new collaborative filtering algorithms and interfaces to display relationships. He maps music into an informative space based on frequency and context analysis of MusicStrand's enormous database of user playlists [42].

---

[17] `http://www.music-map.com/`
[18] `http://www.mystrands.com/`

Figure 3-16: Justin Donaldson's visualizations of artist similarity based on playlist data
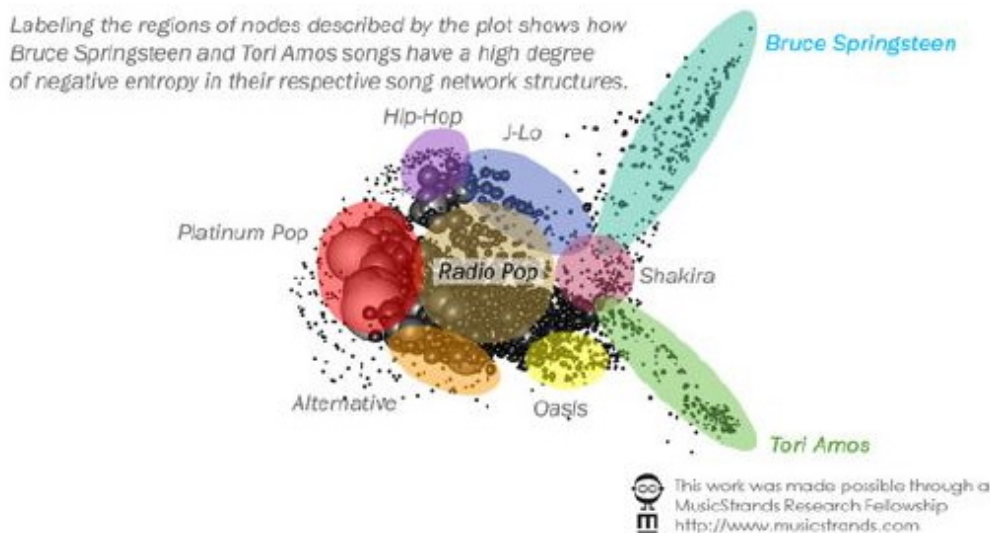
Figure 3-17: Justin Donaldson's work on visualizing graphs of music in 3-D [19]

The methods used, appropriately called "Music Mapping", use visualization to represent closeness between tracks, with that closeness determined by co-occurrence of those tracks on user playlists. Donaldson's goal is ultimately to provide individuals a tool to find new music based on what they already own.[20]

---

[20] http://xavier.informatics.indiana.edu/~jjdonald/ms.php

Donaldson has devised both 2-dimensional and 3-dimensional representations of these spaces. The 2-dimensional spaces are specifically intended for exploring musical recommendations, while the 3-dimensional model has been limited to academic presentations and more generalized conclusions about artist similarity.

Throughout his research, Donaldson emphasizes that we need to look at "exploring music tastes in a more meaningful and open fashion" [42]. This is exactly the kind of sentiment that motivates this thesis work.

3.4.3 MusicRainbow

Researchers Elias Pampalk and Masataka Goto, who have both created their own innovative music browsers (see Section 3.3.1 and [43]), have also worked together to create two beautiful and simple interfaces that use artist similarity to browse music and make recommendations. These are MusicRainbow (this section) and the closely-related MusicSun (Section 3.4.4).

MusicRainbow computes artist similarity from extracted audio features using a traveling salesman algorithm, and places all artists on a "circular rainbow" [44]. Colored arcs reflect the genre of each artist. In addition, Pampalk and Goto enrich the interface with semantic/descriptive words extracted from web pages about the artists. Using a Griffin Powermate knob, users can rotate the MusicRainbow and select artists to listen to (using audio summarization to give users a quick overview of each artist).
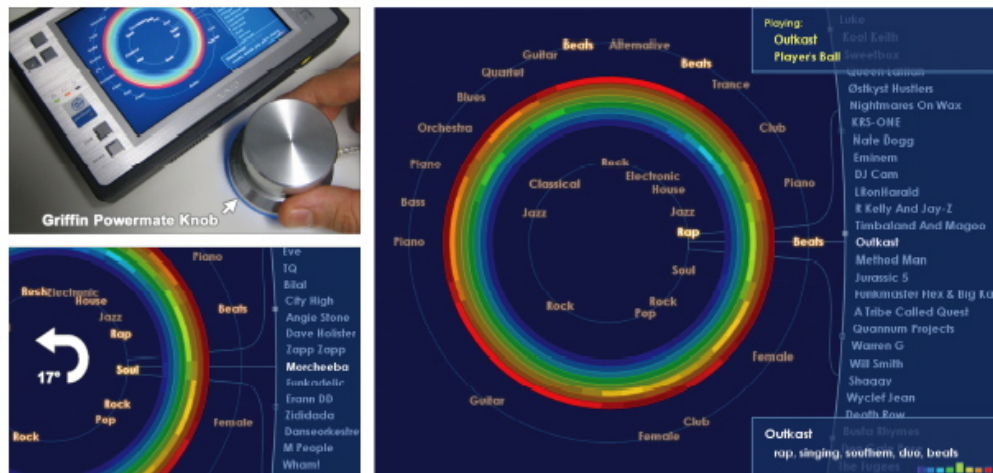

Figure 3-18: MusicRainbow interface [45]

Three important things we note with this project:

- The addition of contextual information gathered from the web to *enrich* user experience, which we've seen in several other projects in this section. After all, interacting with music

is much more than just artist name and genre. Although the authors admit that applying contextual information to the graphical representation was sometimes difficult (e.g. sections labeled with the word "female" included several male artists, because the term is applied broadly to an arc/area instead of to one artist at a time), this is a step towards **combining audio and contextual information** to make the user experience even more rich.

- The use of audio summarization to give users a quick overview of each artist as they browse [44].

- Users browse at the artist-level, but the audio similarity of artists is calculated from track-level information.

We will see all of these issues dealt with in the context of this thesis project.

3.4.4 MusicSun

Pampalk and Goto's MusicSun project is very similar to MusicRainbow in organization style (circular arrangement based on artist similarity), but its goal more heavily emphasizes giving users more choices in focusing their search for new music.

MusicSun combines three different measures of similarity in order to place artists along the circle:

- Audio-based similarity, similar to other algorithms used by projects in this chapter.

- Web-based similarity, computed as tf-idf scores (Appendix B). This means that sociocultural similarity that is high if artist names occur on the same web pages, or if the same words are used on those pages [46].

- Word-based similarity, which allows users to focus in on a descriptive word, like "Swedish", and view artists that most strongly reflect that word.

MusicSun grants the user the flexibility to adjust the impact each of these similarity measures have when creating the combined similarity that controls the display.

As described by Pampalk and Goto, "MusicSun is basically a query-by-example interface" [46]. Like other projects we've already discussed, it is limited in its application primarily because the user must enter an artist name to get any recommendations, and those recommendations are heavily influenced (in the web-similarity measures) by their frequency on the web. In other words, because of the dependence on contextual data from the web, *only popular artists and terms can be searched effectively*. This is a common limitation of applying contextual data to music.

Even so, MusicSun is the first project reviewed here that effectively maps music based on both audio and contextual data.

3.4.5 Search Inside The Music

At Sun Research Laboratories, Paul Lamere's *Search Inside The Music* program maps musical tracks into a 3-dimensional space based on audio similarity. Audio similarity is calculated from features like Mel-frequency Cepstral Coefficients (MFCCs), and projected onto a 3-dimensional space using multi-dimensional scaling (MDS) [47]. Paul Lamere and Douglas Eck hope that this 3D interface provides a much better way of interacting with large music libraries.
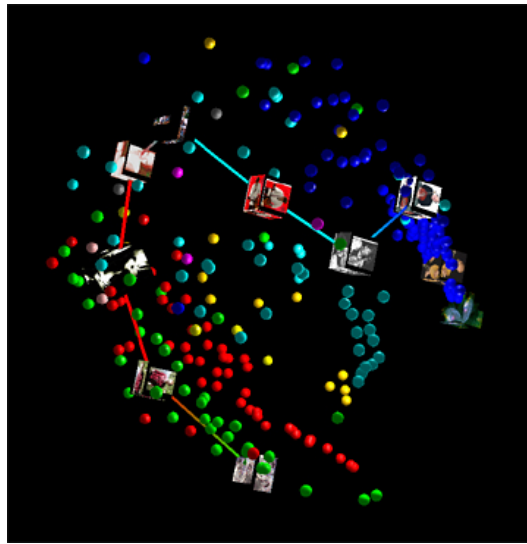
Figure 3-19: Search Inside The Music[21]
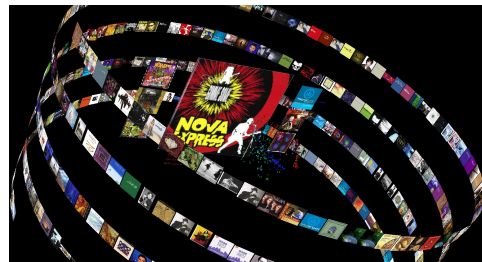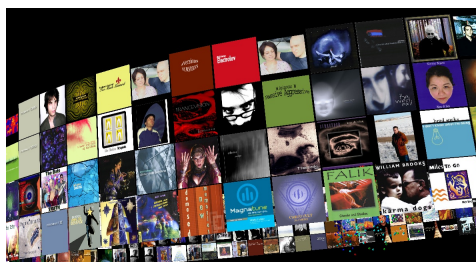
Figure 3-20: Search Inside The Music: Album art views [47]

Search Inside The Music is notable in that it breaks free from a 2-dimensional representation into a space that clearly emphasizes closeness in sound between individual pieces in a music library. It allows users to draw paths through the 3D space to form playlists that have smooth audio transitions (a technique that is also employed in this thesis project). And, unlike the 3D models we saw from Justin Donaldson, its color-by-genre mode instantly gives users a good

impression of the content of a music library without them having heard any of it. All of this underlines the importance of *structure* in a music library, and shows us some ways that users can benefit from features built on top of that structure.

Lamere and Eck also point out that the *experience* of interacting with one's music is paramount, and that we have lost a lot of that experience as we move from the physical world of records, cassette tapes, and compact discs, to the digital world of MP3s. To help bring back this lost experience, *Search Inside The Music* enables many fun album art views (Figure 3-20).

3.5   Controllable spaces / smarter maps

3.5.1   Ishkur's Guide to Electronic Music

Ishkur's Guide to Electronic Music[22] is the most extreme manually-created compendium of music information that we will cover in this chapter. Its goal is to introduce any reader to the styles and attitudes of electronic music.
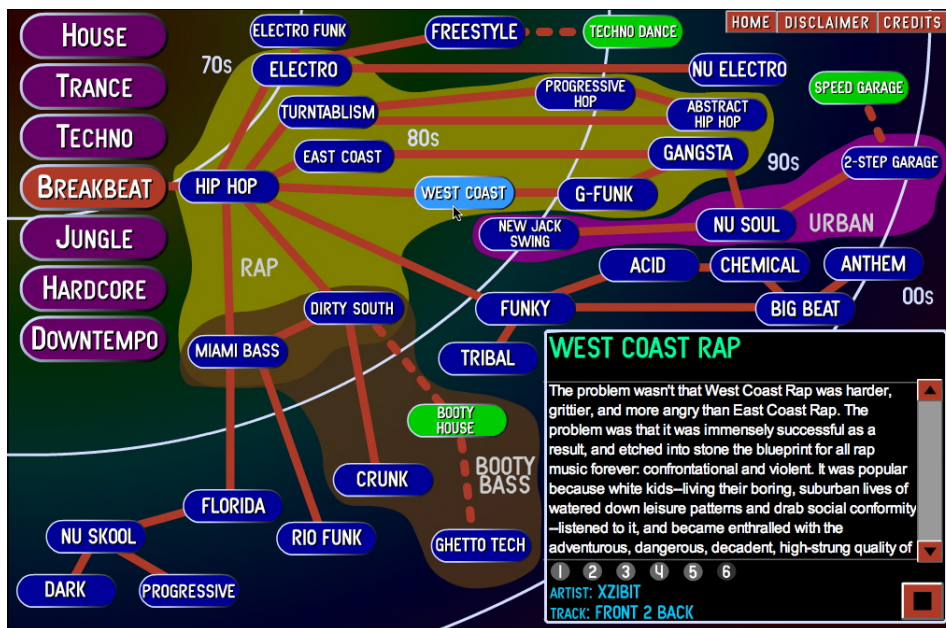
Figure 3-21: Ishkur's Guide to Electronic Music

Ishkur's Guide is basically an explanatory map of electronic music, with hierarchies of sub-genres and sub-sub-genres within them, each accompanied by a succinct description and plenty of sound samples. The Guide's author, an electronic music fanatic named Ishkur, has a sarcastic, sharp sense of humor that permeates his descriptive commentary.

---

[22] http://techno.org/electronic-music-guide/

The amazing maps that make up the Guide encourage a very free-form exploration of the styles of electronic music. The organization scheme varies from page to page, but is generally based on time (year) and style hierarchy presented as a tree crawling across the window area.

While Ishkur's Guide is appropriate and impressively informative for introducing readers to the genres within electronic music, it's clearly subjective and not easily scaleable to a much larger body of music.

3.5.2   Mapping music for portable devices

Fabio Vignoli et al. have developed new music maps specifically designed to be browsed on portable devices [48]. The requirement that the maps be viewed on small screens restricts what data can be shown, so that the mappings are kept simple and clear.
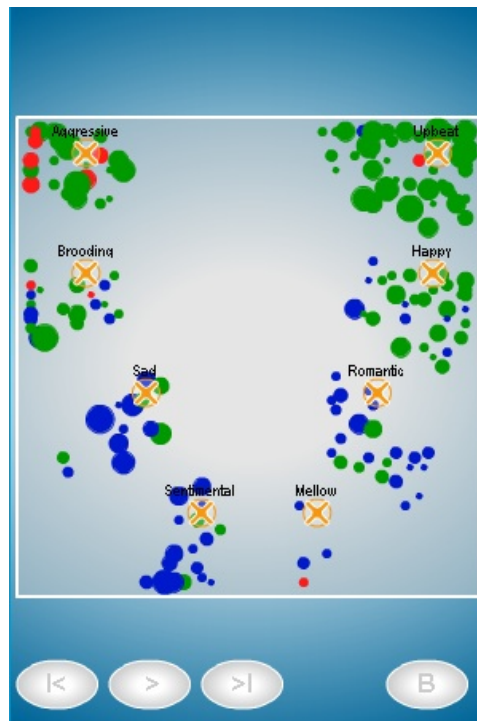
Figure 3-22: Vignoli et al.'s interface for portable devices

Vignoli et al. utilize a new graph-rendering algorithm based on an adapted energy model to create maps of artist similarity based on audio content. They use attributes like mood, genre, year and tempo to label the space and provide context. Users can manipulate the visualization by selecting a color mapping (e.g. year or tempo) and specific attribute magnets (e.g. mood and tempo). Thus, the space is highly customizable and perhaps offers a better overview of an unfamiliar music collection than other algorithms used in the literature [48].

3.5.3 MusicalNodes

The MusicalNodes project also lets users control the organization of a musical space by selecting genre magnets that they can then drag around in the space [49]. MusicalNodes loosens the requirement that we normally have for genre labels: it allows each album to have several genres, if necessary. Allowing multiple genres necessitates a clever way of browsing these genres, since a simple genre listing leaves out any other genre information. MusicalNodes solves this multi-genre problem by displaying songs as blobs surrounding genre-label magnets. Albums can be close to one or more magnets, and rearrangement of the magnets maintains the multi-genre mapping.
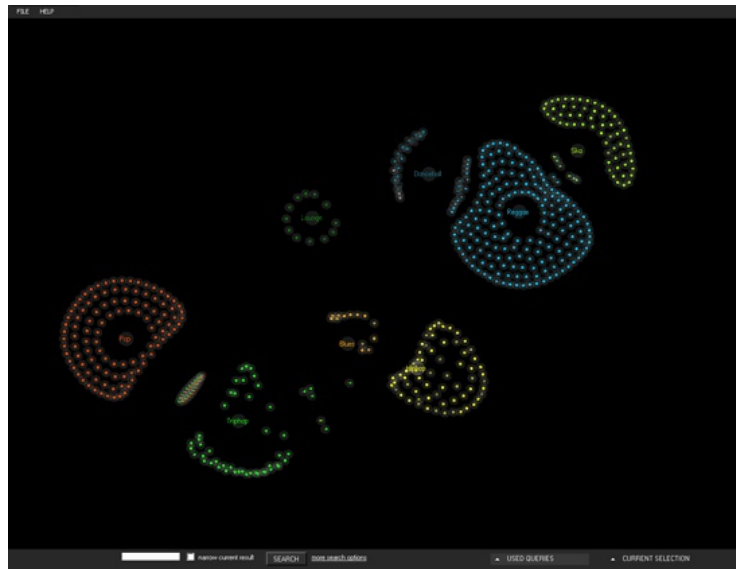
Figure 3-23: MusicalNodes, from `http://waks.nl/index.php?l=nl&x=71`

An interface like MusicalNodes affords a lot of flexibility. Users could even take a light definition of the word 'genre', and invent new terms around which to organize the musical space. The prospects are exciting, but even so, this system requires a lot of its users since it is so dependent on manual classification of music in the library.

3.5.4 Hannes Jentsch

Design student Hannes Jentsch's 2007 thesis[23] puts forth a prototype of a spatially-organized music library. It's an elegant interface, with impressive coordination between the representation of the musical space and the more traditional media player functions (e.g. users drag song circles from the space directly onto playlists that maintain their display icons).

---

[23] `http://www.formater.de/`
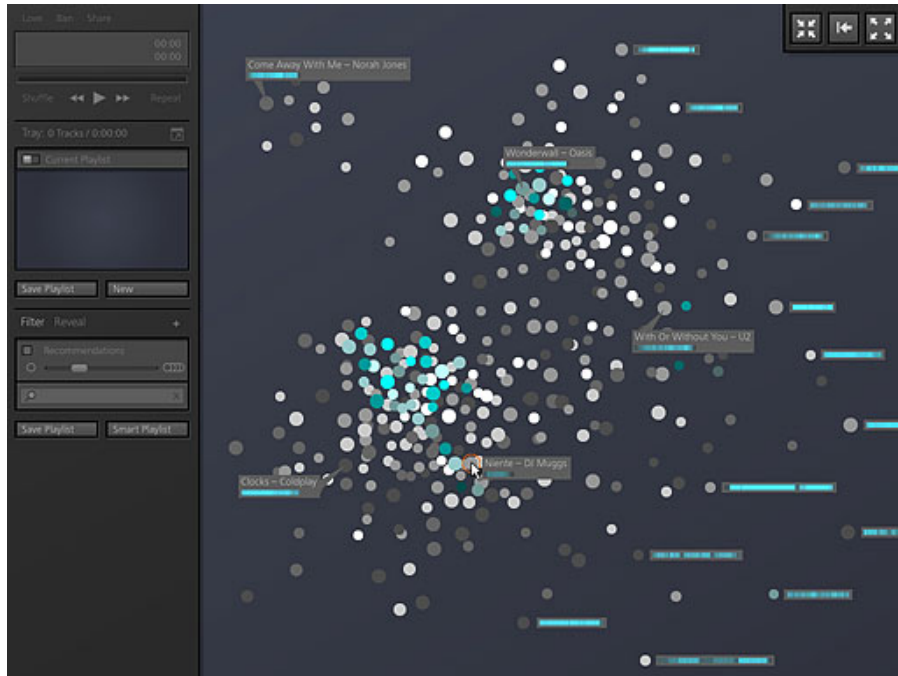
Figure 3-24: Hannes Jentsch's music browser prototype

Jentsch suggests using a SOM to map tracks by musical similarity, most likely because that is a method so commonly used in the literature (Section 3.3). Circle size, brightness, and color reflect track information (e.g. play count). Filters can be set to encourage exploration of unplayed songs in a library. Like other projects in this chapter, Jentsch's interface shows different amounts of information given the user's zoom level.

Because Jentsch is a designer rather than an MIR researcher, his aesthetics are more pleasing and the usability of his (albeit complicated) interface looks to be higher. This thesis project attempts to balance a powerful back-end analysis engine with a decent interface design, in order to provide an interface that is both intuitive and informative.

3.5.5   Visualizing personal music libraries with simple metadata

Now we turn our attention to a music browser clearly designed with information visualization in mind. The approach is much more focused on applying traditional graphing methods to a user's music library in an effort to illuminate the organization possible with semantic metadata.

Torrens et al. [50] specifically target the problem of navigating personal music libraries as they get bigger and bigger (easily having thousands of tracks). The conventional interface of textual lists (e.g. iTunes) can be too limiting when dealing with such large music libraries, and the

authors intend these visualizations to *complement* that interface. Their focus turns to **exploration** and **rediscovery** within one's own music collection.
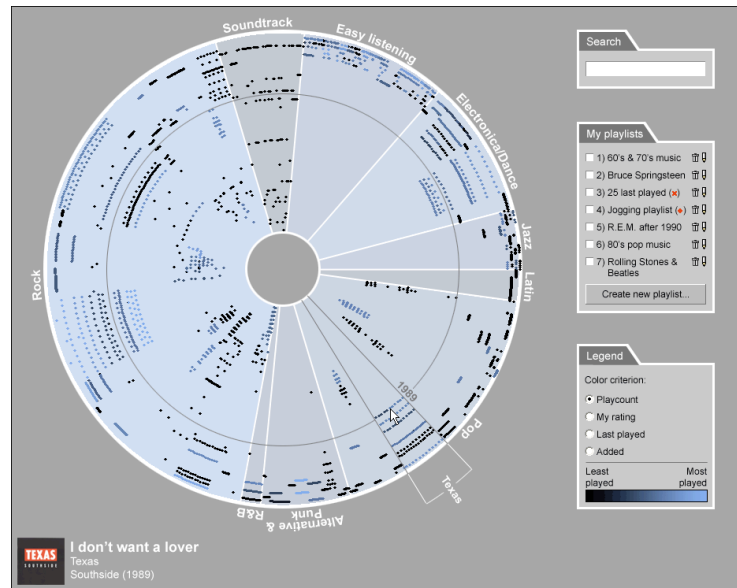
Figure 3-25: Torrens et al.: Visualizing music libraries by using discs

Torrens et al. demonstrate various visualizations – disc, rectangle, and tree-map – to give an informative overview of the contents of a music library and its associated playlists. They build these visualizations from simple music metadata, such as genre, artist, year of release, and play count. These metadata are attributes that many mainstream browsers normally retain anyway.

For example, the disc view, pictured in Figure 3-25, organizes tracks into genres that form slices of the circle, with music released earliest at the center of the disc. This kind of mapping enables the user to easily characterize the genre content of a music library, and to see how these tracks are distributed across that space.

3.6   MusicBox: This thesis in context

Now that I've given some background on related projects, I'd like to describe where MusicBox sits in the context of these projects.

3.6.1   Type of data

In terms of data, MusicBox includes *both content-based and context-based information.* That is, it uses some characteristics of the audio content of songs, in addition to textual information that *describes* the music (e.g. editorial information, user tags, genre class). It's important to use both kinds of data in order to be able to:

- display relationships based on actual audio content, which contextual data ignores

- deal with music that has no contextual data (either because it is new or unpopular)

- capture the cultural significance of contextual data

- treat music at the track-level (instead of the artist/album-level)

3.6.2   Algorithm

The spatially-organized applications described in this chapter use self-organizing maps (SOM), co-occurrence measures, spring-based physical models, and multi-dimensional scaling (MDS) to map their data (tracks/albums/artists) into a space. MusicBox uses principal components analysis (PCA) to take quantitative measures of each piece of music and map it onto a lower-dimensional space.

When PCA is applied to a data set, the resulting dimensions tend to map objects with similar feature values to nearby locations. This means that PCA can give a map of the data that emphasizes similarity between objects. Other methods (e.g. MDS, SOMs) can accomplish the same task, but no significant project has yet analyzed musical libraries with PCA. With MusicBox, we see the effectiveness of using PCA on musical data sets, although further work will be necessary to compare PCA with those other techniques.

3.6.3   Representation

Because MusicBox analyzes its data with PCA, it represents musical tracks as points in a space along axes that are not labelled (Section 5.3.1). The *distances* between tracks are what matter most: similar songs are close together, and different songs are far apart. There are no connections (lines) between the songs because there are no explicit connections between songs' data.

So, although its algorithm is different from other projects in this chapter, MusicBox's representation of artist/track similarity is very similar to what we've seen. It makes sense that MusicBox and other projects all take this approach because users find it intuitive to represent similarity in sound by closeness in space.

3.6.4 Transparency and simplicity

The projects in this chapter vary in how much information they show to the user, and how many choices they require of the user in focusing their queries on the musical space. A simple interface can be more approachable, but it doesn't give the user much control over, or explanation for, its representation. With MusicBox, I opt to make the user interface very flexible and somewhat transparent, giving the user many choices in changing their view of the space. For instance, MusicBox gives users fine-grained control over the included feature set, dynamically remapping the space upon any change. MusicBox also provides an option for labeling axes with their highest contributing features, in order to show the user more of the algorithmic reasoning behind the spatial representation.

Chapter 4

soundsieve

[It is] not so important that there be an absolute mapping between sound and image, which I don't believe there ever can be... but rather that the user feels in watching or using it that the mapping is absolutely present and manifestly felt.
– Golan Levin, describing his project *Number 5* in his talk "Experiments in Art, Technology and Abstract Communication", at the MIT Media Lab on December 10, 2007

This chapter introduces *soundsieve*, one of my early projects at the Media Lab. *soundsieve* is a tool that gives a new view of the structure of music; it inspired me to pursue my main project, MusicBox.

4.1 Motivations

When I came to the Media Lab in 2006, a student named Tristan Jehan had just finished his PhD one year prior in my group. Jehan's work had focused on creating audio analysis tools based on perceptual and structural modeling of music, and then using the results to teach a computer to synthesize music [17].

Jehan's audio analysis tools were impressive, and I was eager to build on what he had left behind. These analysis tools could analyze any MP3 file by breaking it up into perceptually significant units, then quantitatively describing the pitch, timbre, and loudness of each of those units. The result was an XML file filled with numbers.

From an early age, I have loved graphs and information visualization of all types; I have always appreciated the informative power of a good graph. With this new data set, I had *music* as data. How could I make sense of all of those numbers? How could I determine whether the numbers

actually described the music they claimed to have come from? Perhaps I could make a graph of a piece of music, showing how it starts, develops, and concludes, inspired by Charles Joseph Minard's famous graph demonstrating exactly that for Napoleon's ill-fated Russian campaign in 1812[1]. Or perhaps, just as Levin et. al's "The Secret Lives of Numbers"[2] illuminated the hidden structure in our use of numbers, I could uncover some secret life of music.

With these ideas in mind, it was a natural step for me to make images out of those long files of numbers, anticipating that I might be able to find stories in my musical data. And, with that, I started trying to map out the sound inside my music.

4.2   Mapping

Bridging the gap between the audio world and the visual world centers on choosing a mapping between the two. What parts of the audio will be visualized? And how will each of those parts be visually displayed?
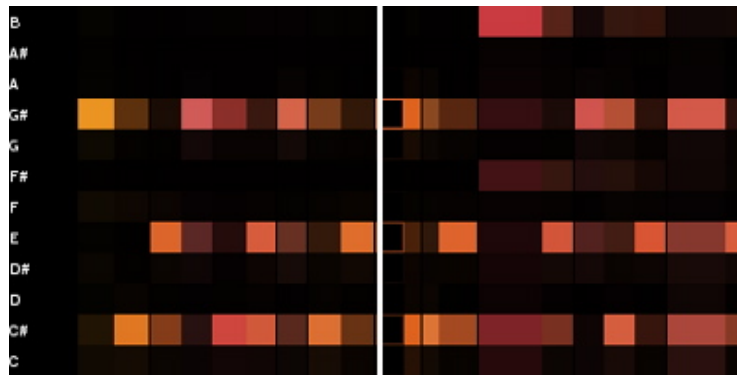
Figure 4-1: Beethoven's "Moonlight" Sonata, up close

The *soundsieve* program visualizes four pieces of data: **time** (on the horizontal axis), **pitch** (on the vertical axis), **loudness** (brightness), and **timbre** (color). These four pieces of data represent much of the basic quantifiable characteristics that distinguish one song from another. *soundsieve* displays this data as a grid of notes over time.

Time is mapped to the horizontal axis, from left-to-right, since that mapping is so frequently used in graphing with time-dependent data that it has become intuitive for most users. The audio analysis breaks up each MP3 into a series of irregularly-sized segments that most closely

---

[1] Edward Tufte's website has more information on Minard's famous graph:
http://www.edwardtufte.com/tufte/posters
[2] The Secret Lives of Numbers is an interactive visualization of Internet popularity of the integers between 0 and 100,000. The interactive applet is available here: http://www.turbulence.org/Works/nums/applet.html. More detailed information on the project can be found at http://www.flong.com/projects/slon/.

correspond to beat onsets/notes (usually about 0.1 to 0.3 seconds long); these become the vertical slices in the image. Wider vertical slices mean longer notes.

Each one of those vertical slices is chopped up into twelve pitches along the vertical axis; these correspond to the twelve semitones in an octave (the chromatic scale). All pitches are compressed into one octave, so notes that are an octave apart appear as the same pitch on the vertical axis. One result of this is that a series of notes increasing a half step at a time can "run off" the top of the visual space and then loop back around to the bottom of the space.

The brightness of each pitch at a given point in time is a direct reflection of its loudness at that point. Darker boxes indicate quieter notes; brighter boxes are louder. Thus, a three-note chord is indicated by three relatively bright boxes against a dark background.

Each vertical slice (equivalent to a time segment) is colored with an RGB color value corresponding to the timbre of the sound during that segment. Jehan's analysis tools describe timbre in terms of a principal components analysis (PCA; see Appendix A) of the frequency distribution during that time slice [17]. *soundsieve* maps the first three principal components directly to R(ed), G(reen), and B(lue) values, respectively. The end result is that timbres characteristic of one type of instrument map onto one region of the color space, and that a very different timbre (i.e. one having a very different frequency distribution) maps to a color that is far away in the RGB color space. Simply put, sounds that sound similar will appear similar in color. (Note that the order of the mapping to R(ed), G(reen), and B(lue) values is arbitrary, and could just have easily been to G, B, and R, respectively, or any other order, instead.)

When choosing the mapping for *soundsieve*, I tried to stay consistent with the intuitions I have gained from reading music and from talking about its content. For example, choosing time along the horizontal axis and pitch on the vertical axis was clearly an intuitive mapping, since that is how music is displayed as notes on a staff. In addition, music is described in terms of tone "brightness" and "color", so I tried to map those facets as literally as possible when moving into the visual realm. All of these direct relationships resulted in a mapping for *soundsieve* that is easy to understand.

Some examples will help to illustrate *soundsieve*'s audio-visual mapping. [3]

4.3   Examples

Figure 4-2 shows *soundsieve*'s image for Beethoven's "Moonlight" Sonata. Pitch names are labeled along the left side, and detectable notes comprise the rest of the image, according to the mapping described in Section 4.2. The image progresses from the beginning of the piece at left, to the end of the piece at right.

---

[3] All of the song images in this chapter are also viewable at:
`http://www.flyingpudding.com/projects/viz_music/`
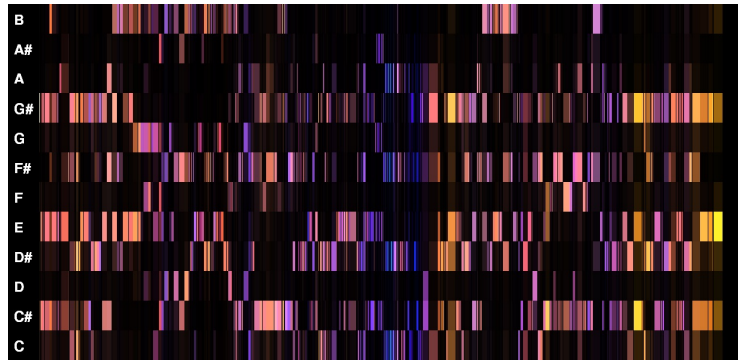
Figure 4-2: Beethoven's "Moonlight" Sonata

Figure 4-1, at the start of the last section, shows a zoomed-in view of the beginning of this same piece. At this level, one can see individual notes being played, creating chords of stacked horizontal bands. The user can track the progression from chord to chord across the piece.

Compare the image for the "Moonlight" Sonata (Figure 4-2) to the images for "Fur Elise" and the "Pathétique Sonata, II" (Figures 4-3 and 4-4). Although the images differ in their placement of notes, the color ranges displayed in these pieces are relatively close to each other. This makes sense, given that all of these pieces are played by a lone piano. The piano has a particular set of timbres that is represented in *soundsieve*'s PCA<->RGB mapping by pinks, peaches, and blues.
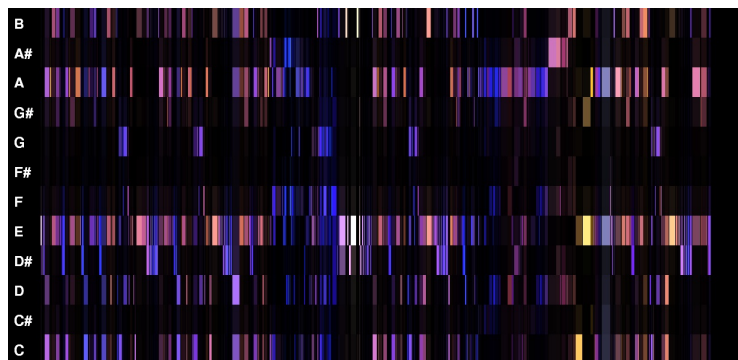
Figure 4-3: Beethoven's "Fur Elise"

In the beginning of Beethoven's "Fur Elise" (Figure 4-5), one can identify the string of E's and E-flats (labeled "D#") that start the very recognizable theme. By looking at the pattern of predominant note names in these pieces, one can almost *see* the key of each piece: the beginning of the "Moonlight Sonata" is in C# minor (note the horizontal bands of notes at C#, E, and G# along the left edge of the piece), and "Fur Elise" is in A minor (note the predominance of A and E with a lack of C).
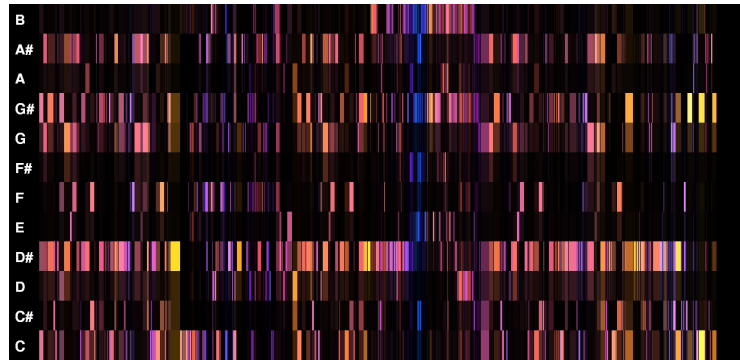
Figure 4-4: Beethoven's Pathétique Sonata, II
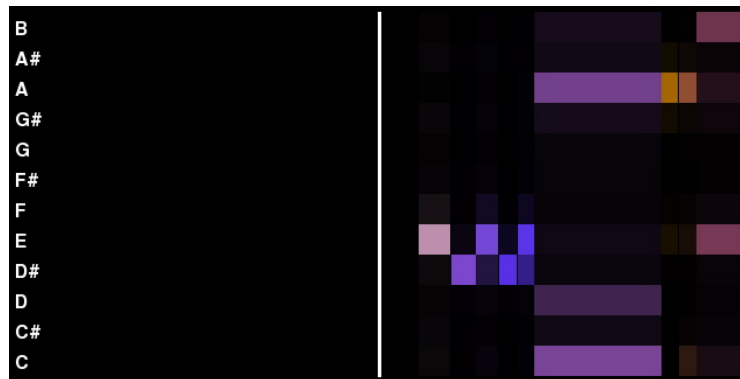
Figure 4-5: The beginning of Beethoven's "Fur Elise," zoomed in.

Let's compare these Beethoven piano pieces to two electronic music tracks by the artist Daft Punk (Figures 4-6 and 4-7).
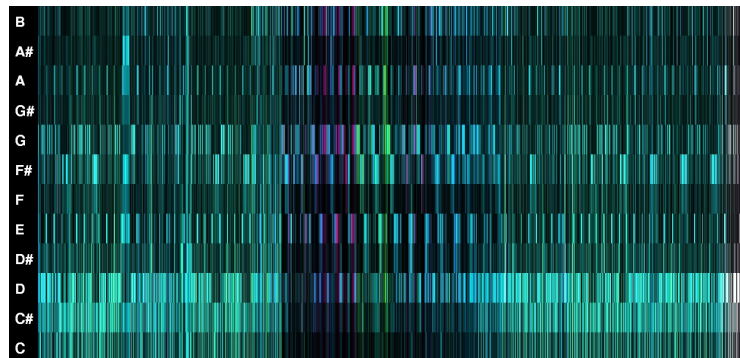
Figure 4-6: Daft Punk's "One more time"

There are several things to note in this comparison. First, the colors visible in the Daft Punk tracks are quite different from those in the Beethoven pieces. In the track pictured in Figure 4-6, the structure is repetitive and we see few timbral changes; what changes we do see are limited to
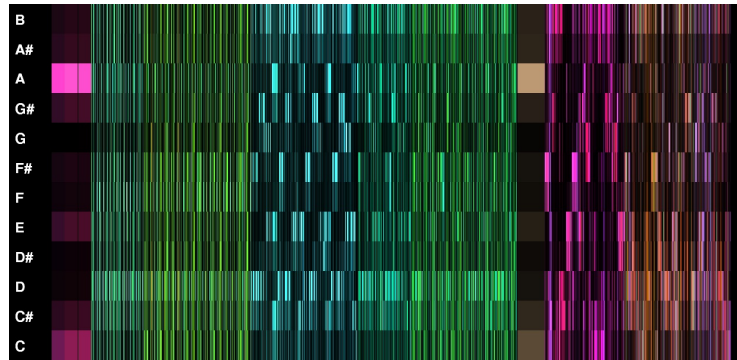
Figure 4-7: Daft Punk's "Aerodynamic"

the middle of the piece. In Figure 4-7, there are marked timbral and structural changes, breaking up the piece into distinct sections. Daft Punk's "Aerodynamic" starts with the gonging of a bell, a sound much more similar to the harmonics of a piano than to the noisy dance beats that follow; the color reflects this similarity.
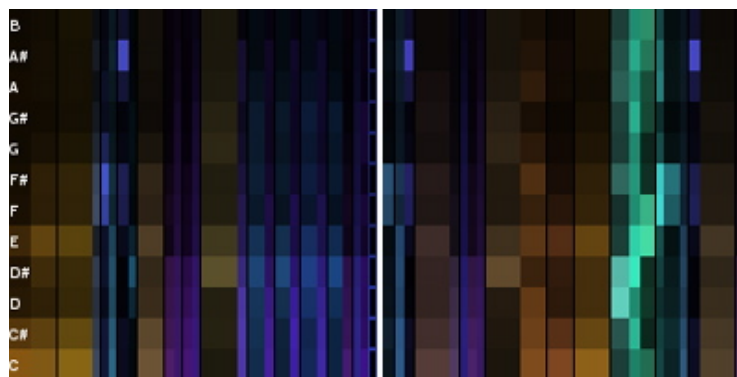
Figure 4-8: Timbre changes during a percussion solo in Daft Punk's "Crescendolls"

Inspection of several other electronic pieces verifies the repetitive structure that we hear (and can now *see*) in these pieces. Electronic pieces have also been observed to have much more sudden changes in timbre (which looks particularly striking at the note level, see Figure 4-8), as well as a much wider variety of timbres. In some other styles of music, we can see that some vocal artists (e.g. Joanna Newsom) cast a distinctive colored shadow over their music whenever they sing.

See Figure 4-9 for other examples of pictures of songs.

Nirvana, *Come As You Are:*
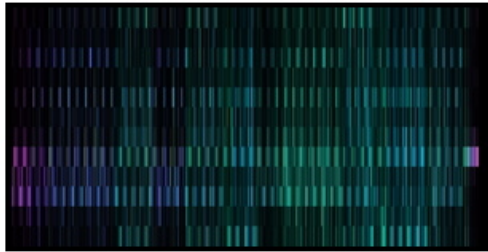
Sonic Youth, *'Cross the Breeze:*
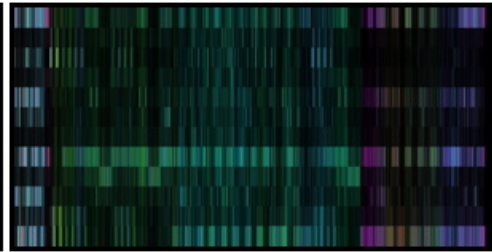
Joanna Newsom, *Bridges and Balloons:*
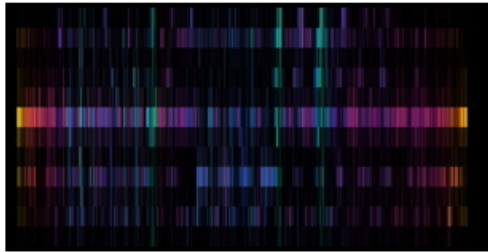
The Flaming Lips, *When Yer Twenty Two:*
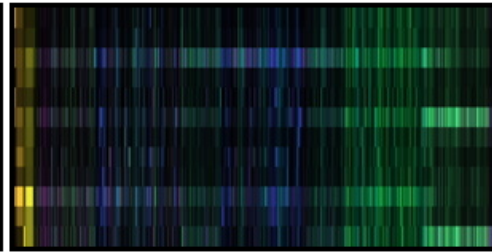
Strauss, *Also sprach Zarathustra*
(theme from "2001: A Space Odyssey"):
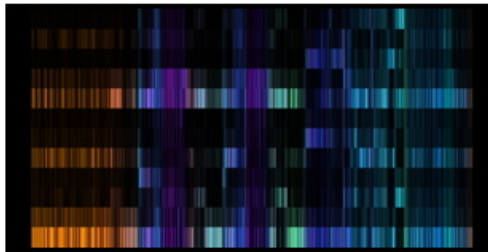
Kraftwerk, *Pocket Calculator:*
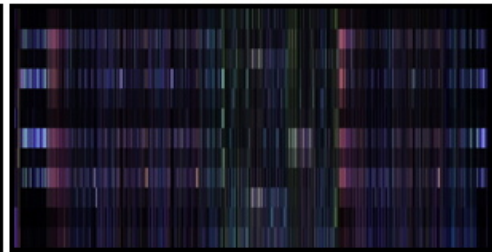
Figure 4-9: A selection of song images from *soundsieve*. Note names have been removed for simplicity.

4.4   The music browser

The core of *soundsieve* is the visualization of each piece's audio structure in one image that becomes a "snapshot" for that piece. However, I extended the functionality of the program to make it a fully-functioning music browser. In this browser, the user selects songs through a traditional media player interface (pictured in Figure 4-10); as each song is selected, *soundsieve* displays the snapshot (at right in the Figure) as a visual preview of the song's audio content. The browser interface makes it easy to compare a wide variety of songs in a library. During song playback, *soundsieve* creates an animation[4] that accompanies the audio (top left in the Figure).

---

[4] A video demonstration of selected song animations in *soundsieve* is available at
`http://www.flyingpudding.com/projects/viz_music/#video`.

The animation maps the current time to the center, demarked by a vertical white line, with notes recently played to its left, and notes immediately to come on its right.
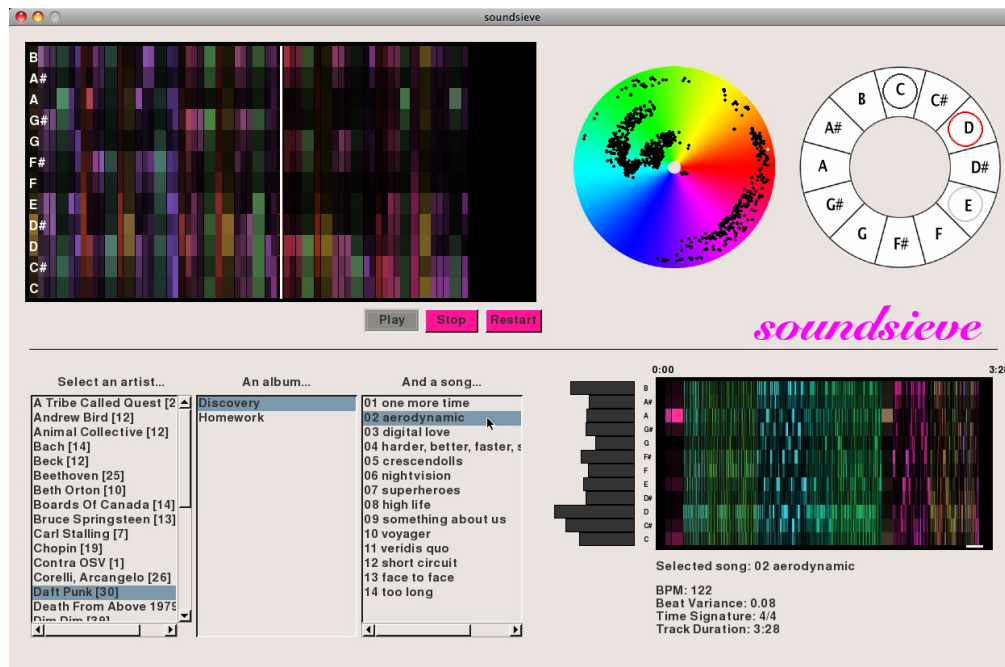
Figure 4-10: Screenshot of *soundsieve*'s full browser interface. Song image is displayed at bottom-right, and animation accompanies playback at upper-left.

soundsieve also employs some visual aids to note changes in timbre and pitch (circles at top right of Figure 4-10). One of these is a color wheel that plots each vertical time slice as a black dot at its corresponding timbre/color. The placement of this dot moves from the center of the circle at the beginning of the piece, to the outside of the circle at the end. In the *soundsieve* screenshot, you can see how this view emphasizes the change in timbre from section to section by noting the clusters of dots moving from one area to another on the color wheel.

The pitch circle simply shows all possible note names, circling in red the note that is most prominent at the current time. The most recent eight notes are also shown, circled in shades of gray that fade to white as they get older.

Along the left side of each song snapshot (just to the right of the text-based controls in Figure 4-10), *soundsieve* also displays the note distribution across the piece; it is simply a histogram on its side. This representation can hint at the key (as described in Section 4.3), or emphasize the lack of one (which you can see in the almost even distribution of notes for the song in Figure 4-10).

soundsieve's animations highlight the connection between the image and the audio. During demonstrations to groups at the Media Lab, visitors were able to understand the audio-visual mapping in the first minute of watching and listening to *soundsieve*.

4.5   Implementation notes

soundsieve was implemented in Python, using Pygame [51] modules to handle drawing rectangles and text to the screen. It creates all of its visualizations from simple MP3 files, which has allowed it to be easily applied to new music libraries.

MP3 files were analyzed using an early version of The Echo Nest's Analyze API[5]. This early version was a stand-alone application that I could call directly from *soundsieve*'s Python code. As The Echo Nest has grown since I started this project, they have replaced this stand-alone application with their Analyze API, which, at the time of this writing, is freely available on their website.

4.6   Other song visualization methods in MIR

There have been many other academic approaches to visualizing the structure of music before *soundsieve*, of course. The one that *soundsieve* most closely resembles is Stephen Malinowski's Music Animation Machine (Figure 4-11).

Figure 4-11: Music Animation Machine screenshot of Johann Sebastian Bach, WTK Book I, Fugue #4, C# minor, from `http://www.musanim.com/store/`

The Music Animation Machine uses a very similar mapping to *soundsieve*: notes are displayed as colored blocks across time, with pitch arranged vertically, and colors corresponding to musical instrument. The key difference between the Music Animation Machine and *soundsieve* is that the Music Animation Machine is limited to visualizing MIDI file format only. MIDI is a protocol that transmits digital events which signify the pitch, intensity, and instrument to play. It is inherently clean and easier to visualize than real-life recordings. The disadvantage with a reliance on MIDI format is that most of the music we actually listen to each day is assuredly *not*

---

[5] Founded by Tristan Jehan and Brian Whitman, The Echo Nest builds on their Media Lab research to "develop music search, personalization and interactive applications on top of a platform that automatically reads about and listens to the entire world of music" (`http://the.echonest.com/company.html`). For more information on the API itself, see `http://the.echonest.com/analyze/`.

in MIDI, and it is not an easy task to convert a piece to MIDI; it typically takes painstaking work by a well-trained musician to do it correctly.

Another exceptional MIDI visualization comes from Martin Wattenberg, a computer scientist who specializes in information visualization at IBM Research. Wattenberg's "The Shape of Song"[6] highlights repeated passages in music by connecting them with translucent arches (Figure 4-12). The resulting images illuminate the variety of forms present in music. Like the Music Animation Machine, its images are derived from MIDI, however, so it is not easily applied to an arbitrary selection of music.

Figure 4-12: Martin Wattenberg's "The Shape of Song"

Many of the more recent academic approaches to visualizing music focus on *tonality*, which is primarily concerned with the progressions of key through a piece of music. Notable among these are Craig Sapp's "Tonal Landscapes" [52] (Figure 4-13) and MuSA.RT, by Elaine Chew and Alex François [53] (Figure 4-14).
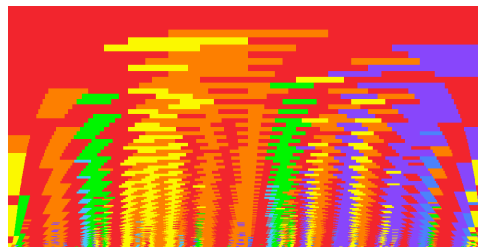
Figure 4-13: Tonal Landscape of Scott Joplin's *Peacherine Rag*

Both "Tonal Landscapes" and MuSA.RT implement key-determination algorithms that take MIDI-like input and hone in on the key iteratively. These images of tonality fit their theme well, choosing to highlight a higher-level characteristic of the music without visualizing the individual notes.

The mapping for Craig Sapp's "Tonal Landscapes" is quite different from that of *soundsieve*. It maps pitch to color, and the images are composed of stacked views at varying levels of detail: the top row of the image is colored by the dominating key of the whole piece, and the bottom

---

[6] http://www.turbulence.org/Works/song/
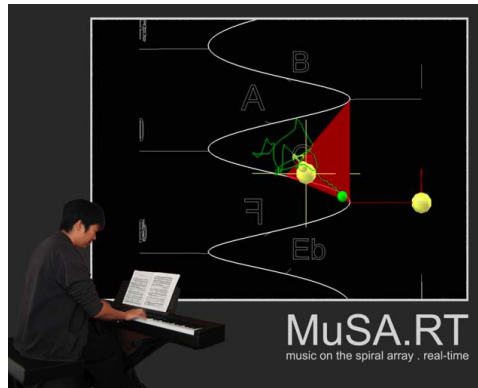
Figure 4-14: Elaine Chew demonstrating the MuSA.RT system

row is colored by key at the note level. This method of visualization provides a nice overview of one piece at a time, highlighting the variety of keys contained.

In contrast, Chew and François developed MuSA.RT as a live interactive demonstration that draws attention to key changes in real-time; it does show both short-term and long-term fluctuations in tonality, but the focus is not to create an overview image of an entire song. Each song is viewed on a 3-dimensional spiral that spins as the notes dance up and down along its length.

At Princeton University, Dmitri Tymoczko has developed a completely novel geometry for viewing musical chord sequences. His work, pictured in Figure 4-15, visually shows the chord symmetries that enable efficient, independent voice leading, which is of particular importance to Western styles of music [54].
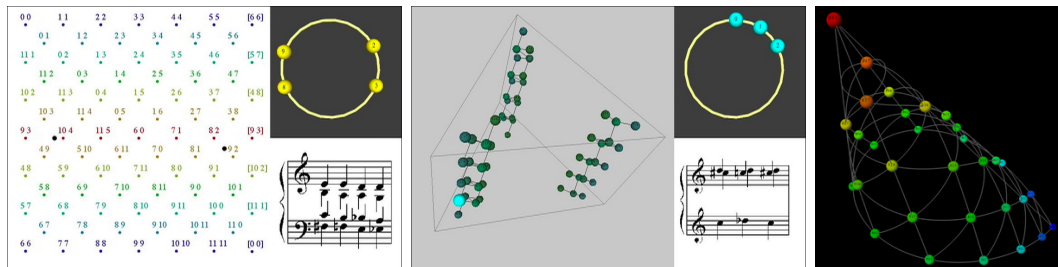
Figure 4-15: A sampling of Dmitri Tymoczko's chord geometries. A chord visualized on a Möbius strip (left); a different chord on an orbifold in three dimensions (center); a chord on a cone (right).

These are just a few of the many diverse academic projects devoted to visualizing the structure inside pieces of music. In the next section and the following chapter, we'll look beyond visualizing individual pieces and into the realm of visualizing entire music collections.

4.7   Extensions and applications: Other uses for *soundsieve*

The version of *soundsieve* described here is quite appropriate for its purpose: to highlight the structure of each piece by use of an image, and to *see* the results from audio analysis algorithms. The images *soundsieve* produces certainly confirm that Jehan's analysis does what it should; in that sense, *soundsieve* acted as a tool to qualitatively test his algorithms.

But *soundsieve* as a music browser is just a start to the possible benefit of displaying music as images. One friend, upon seeing its animations for the first time, remarked that he wished he'd had his guitar with him, because he wanted to use the animations as a guide for accompaniment. Since *soundsieve* shows not only what notes are being played, but what notes are to come in the near future, it can indeed act as such an accompaniment guide, one that would encourage more free-form accompaniment than guitar tabs, for example.

During demonstrations at the Media Lab, some visitors found *soundsieve*'s images so compelling that they said they wanted to see them when browsing for music, either online or in a physical store. Imagine shopping for music on the Apple iTunes Store, finding tracks by a new artist, and using a *soundsieve* image to guide your listening, perhaps motivating you to scrub to a particular area of interest in a song or album. Or you might find yourself leafing through CD cases in a physical store, looking at song images and picking out the albums with repetitive, soft sounds. In situations like these, the *soundsieve* images could help guide you in finding the right track.

Most importantly, though, the images in *soundsieve* were a start to thinking about the higher-level patterns inside of these pieces of music. There were obviously themes, sections, even story arcs, contained in each piece. These were elements of the piece that we could plainly see in the image, but that were not contained in any way in the numerical file that the image was visualizing. These higher-level patterns were perhaps even more important in characterizing the "sound" of a particular piece; picking them out and characterizing them quantitatively would be a major step in teaching computers to listen. So I started thinking about how to generalize these note patterns, and how those patterns could be *used to organize the music library itself*. This was the seed for the MusicBox project.

As we will see in the next chapter, we can take the same types of audio analysis that *soundsieve* depends on and turn the data into a visualization not just for one song at a time, but for an entire music library.

Chapter 5

MusicBox

Music is nothing but unconscious arithmetic.
– Gottfried Wilhelm Leibniz

5.1 Overview

MusicBox is an application that visualizes a music collection by mapping songs onto a
2-dimensional space (Figure 5-1). MusicBox assigns song locations by analyzing many
descriptive features of a song's sound. These features can be *quantitative*, such as the
proportion of high frequency sounds, or *qualitative*, such as characterizing a song as "happy".
MusicBox's organizational algorithm displays songs with similar feature values closer together.
What this means is that users can use their intuition about visual spaces and maps to infer quite
complex conclusions about the relationships between pieces of music, and how they fit together
as a whole to form the music library. Thus, being a visualization allows MusicBox to address the
complexity of a body of music by presenting the music in a free-form, interactive space.

One main advantage of this approach is that it avoids strict classifications of music into genres;
instead, it encourages exploration and discovery. Users move through the space, investigate
patterns of interest, and create playlists by drawing paths or selecting areas in the space.

I stated the goals of the MusicBox project in Section 1.2, but they are worth repeating here. The
goal of the MusicBox project is to make an understandable (and hopefully intuitive) space
for:

- navigation, exploration, and recommendation

- exploration of unfamiliar music collections (e.g. of a stranger, a friend, or a company)
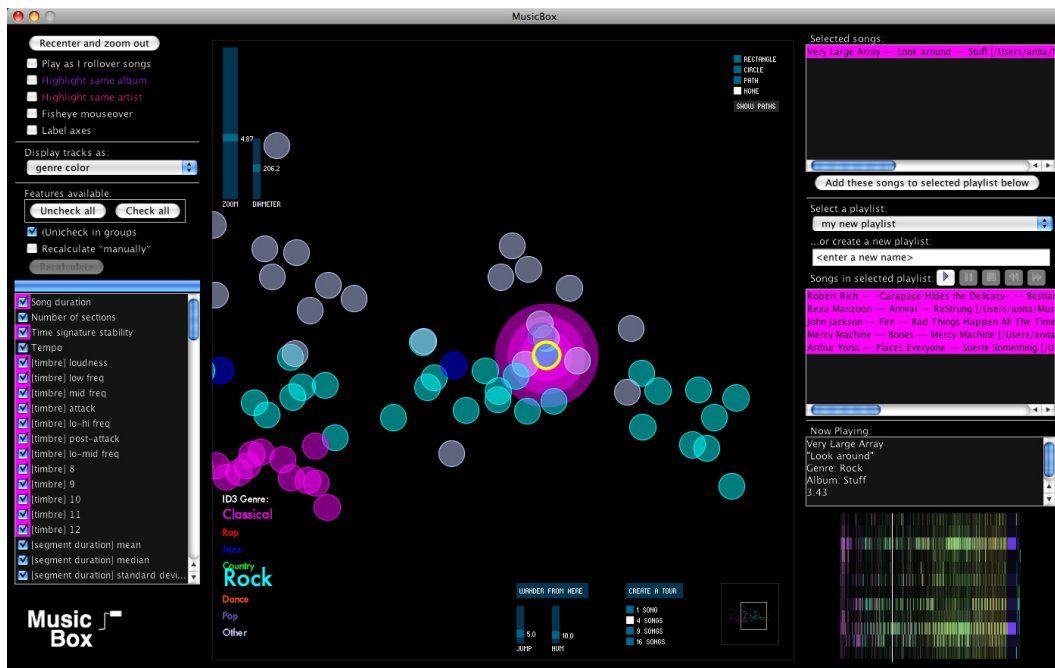
Figure 5-1: Screenshot of MusicBox. The MusicBox application visualizes music in a 2-dimensional space.

- showing one's music collection in the context of a larger or separate music collection

In this chapter we'll look at the main steps MusicBox takes to transform a library of music into an interactive space. Each of these steps becomes a section in this chapter. These steps, outlined in Figure 5-2, are:

1. **Analysis**: the process of extracting features either directly from or associated with the songs in the library. This set of audio and contextual features is extensible; users can write their own feature extraction classes and add them to the MusicBox.

2. **Organization**: reduces the dimensionality of the resulting data set to two dimensions that are more appropriate for visual inspection. MusicBox uses principal components analysis (PCA) to reduce the space.

3. **Visualization**: a graphical user interface that motivates exploration of the reduced space. The display of data is highly customizable, and plenty of traditionally-used music browser tools (e.g. playlist support) are provided.

Every step is automatically completed by the computer, requiring no manual effort on the part of the user. Instead, the user's efforts are expended on *interaction* with the space.
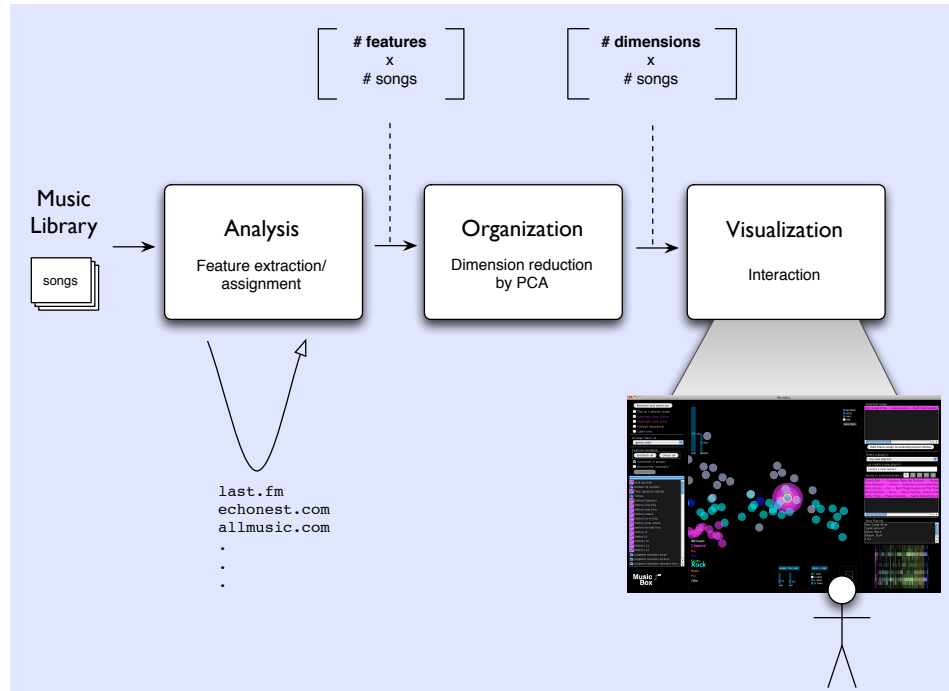
Figure 5-2: Overview of MusicBox. The MusicBox application extracts features for a music library, performs a dimensional reduction of that feature space, and displays the resulting map as an interactive visualization.

By designing MusicBox in this manner, I hope to contribute each of these elements, which set it apart from the related work we saw in Chapter 3:

- visualize music at the song and library level

- demonstrate the value of looking at music in a space

- allow song placement to be determined by both audio features and contextual features

- enable traditional music browser behaviors by interacting with the space

- encourage rediscovery of long-forgotten tracks in one's personal library

- allow deep and plastic user-driven exploration

5.2 Analysis: Feature extraction from MP3s

For each of the MP3 files in the input music collection, MusicBox either extracts features directly or retrieves associated features from an outside source. For example, each file is analyzed locally

to determine its tempo, and user-provided tags describing the track's artist are retrieved from a webservice like Last.fm[1].

Analysis operates on each song instead of on an artist or album, because artists and albums are often so varied in content that grouping them is too limiting. Indeed, one song by Neil Young may seem more "rock" while another may seem more "country". Dealing on a song-by-song basis also makes sense in that it is more and more how consumers *buy* music, as digital music services like the iTunes Music Store and Amazon.com Music gain in popularity. In addition, even our mainstream media software (like iTunes and WinAmp) allows us to easily manipulate lists at the song level. No longer is it necessary to limit oneself to groups of songs at a time.

MusicBox uses the features it extracts and assigns to organize music in a space (see Section 5.3). *It makes the assumption that songs with similar features are similar songs.* This assumption is commonly accepted in the MIR community. (It's also a common rationale used in multivariate statistics.)

Mapping song similarity from feature values is also espoused by the Music Genome Project[2], which started Internet radio service Pandora. The key difference between the approach here and that used for the Pandora music service is that Pandora's features (their "genome") are *manually assigned* by expert listeners. The Music Genome Project staffs people who spend 20-30 minutes classifying each song in the corpus [10, page 108]. Because this manual effort is not scalable, "the Project is continually falling behind the total catalog" [10, page 108], and Pandora can only recommend a limited selection of music.

I designed MusicBox with the requirement that all feature assignment be automatic to avoid this problem of scalability. One implication of this is that MusicBox is limited in the features it is able to assign. For example, there does not yet exist a tool to automatically determine the Pandora traits "cash-obsessed lyrics" or "Brazilian Jazz influences". Indeed, such descriptive and subtle features may result in a better recommendation or organization service. However, I was willing to compromise the ability to use features like this in order to have MusicBox operate in a scalable and equal-opportunity manner.

5.2.1   List of included features

MusicBox uses a wide variety of features, both content-based and contextual, in assessing the quality of each song's sound. I list all of the implemented features here, along with descriptions and attributions as necessary. These features are not intended to represent an ideal feature set. Rather, they are examples of the variety of data available, and serve as a proof of concept that would almost definitely benefit from further expansion and development.

---

[1] For more information on Last.fm's webservices, see
`http://www.last.fm/api` and `http://www.audioscrobbler.net/data/webservices/`.
[2] `http://www.pandora.com/mgp.shtml`

Song duration – simply the length of the track, in seconds [55]. This feature is retrieved using The Echo Nest Analyze API [56]. *[float]*

Number of sections – the number of consistent-sounding partitions of the track. These sections usually correspond to structural units (verse, bridge, solo) [55]. This feature is calculated by The Echo Nest Analyze API [56]. *[integer]*

Time signature stability – a rough estimation of how stable the time signature is throughout the track [55]. This feature is calculated by The Echo Nest Analyze API [56]. *[float]*

Tempo – overall track tempo estimation (BPM) [55]. This feature is calculated by The Echo Nest Analyze API [56]. *[float]*

Timbre – a characterization of timbre, a complex feature. Timbre can be described as what makes two different instruments (e.g. a violin and a piano) sound different even as they play the same pitch at the same loudness [57]. Here, we characterize timbre as a vector of twelve numbers corresponding to the intensity of correlation with a set of basis functions. These basis functions were statistically computed to best describe how timbre varies, and can be interpreted with terms like "low frequencies", "mid frequencies", and "attack" [55]. MusicBox uses a mean measurement for each of these twelve basis functions, averaged across the whole song. This feature is calculated by The Echo Nest Analyze API [56]. *[array of 12 floats]*

Segment duration – statistical metrics (e.g. mean, median, skewness) regarding sound segments in the track. A segment is "a short sound entity (under a second) that is relatively uniform in timbre and harmony. Segments include events such as piano notes and guitar chords. More complex timbres (such as a mix of a bass pluck, cymbal hit, and voice phoneme) can be included here as well" [55]. Segment durations are calculated by The Echo Nest Analyze API [56], and summary statistics are calculated in MusicBox. *[float]*

Genres – used only for music from the Magnatune library[3]. Examples of Magnatune genres are Ambient, Classical, Choral, and Pop. Magnatune assigns genres to each track, and makes these assignments accessible through a song-by-song comma-delimited file, retrieved from `http://magnatune.com/info/api`. Magnatune allows multiple genre assignments per track. *[list of genres is array of Strings, for which each track is assigned a value of 1 (member) or 0 (non-member)]*

Rhythm histogram – a description of rhythm in the audio file, developed by Thomas Lidy and Andreas Rauber at the Vienna University of Technology [22]. The histogram contains 60 bins, which reflect "rhythm energy" per modulation frequency between 0 and 10 Hz. Lidy

---

[3] Magnatune (`http://magnatune.com/`) is a site that works directly with independent musicians world-wide to provide CD-quality tracks with no digital rights management (DRM), at a price the customer chooses.

and Rauber make their analysis available for download[4]; the version used in MusicBox is the Matlab version. *[array of 60 floats]*

Moods – adjectives describing the track, assigned by editors at Allmusic.com. Examples: Angry, Indulgent, and Sentimental. *[list of moods is array of Strings, for which each track is assigned a value of 1 (member) or 0 (non-member)]*

User-assigned tags for artist – most distinguishing tags for the given artist, given by users at Last.fm. MusicBox chooses the top three tags per artist by calculating those with the highest tf-idf scores (Appendix B). *[list of tags is array of Strings, for which each track is assigned a value of 1 (member) or 0 (non-member)]*

Popularity – an estimate of artist popularity, inferred from Last.fm user tags. MusicBox sums raw tag counts to get popularity metric: more tags means more popular. Raw tag counts were provided by Last.fm, via Paul Lamere. *[integer]*

Most of these features were not written by me; rather, I wrapped code that is already publicly-available for either extracting the feature from the audio (e.g. using The Echo Nest Analyze API to extract tempo) or retrieving an assigned contextual value from an online source (e.g. gathering user-assigned tags from Last.fm). The features I did write were segment duration statistics and the popularity feature (which itself is derived from Last.fm data).

MusicBox's feature set is cleanly extensible. MusicBox's code contains an abstract Feature class, from which all of the specific features (listed above) descend. To implement a new feature, one must only extend the basic Feature class and implement one method which returns feature values given a set of tracks. Using abstract classes encourages other developers to wrap and incorporate new features as they feel necessary.

One noticeably missing element to these feature sets is any measure of time-based ordinal data. Ordinal measures, such as chord progression and section-to-section structure, were excluded because it would be too complicated to model them in such a way that dimensional reduction could be applied.

5.2.2 Missing data

For some tracks, feature data cannot be extracted, and this can happen for a variety of reasons. In some instances, MusicBox assigns the tracks empty (0) data values for those features; in other cases, MusicBox removes the problem tracks from the music library entirely (i.e. they will not be viewed in the MusicBox).

Some code fails when dealing with particular tracks that don't have the structure the code expects. For example, Lidy and Rauber's Rhythm Histogram (Section 5.2.1) analysis decodes

---

[4] `http://www.ifs.tuwien.ac.at/mir/audiofeatureextraction.html`

MP3s into WAV files before extracting the histogram. The MP3 decoder expects an MPEG header in the file. If it does not find the header, it cannot process the file, so no rhythm data is collected for that file. Some analysis techniques, like those provided by The Echo Nest, may fail to process a file successfully but provide no explanation of the cause.

Other code fails because the data source does not contain any data for the given track or artist. Some artists, for example, are so obscure that they have no tags given to them by Last.fm users. In addition, some artists are not in the Allmusic.com database, so no moods can be retrieved for them.

Another unexpected lack of data occurs for classical pieces. A Beethoven piano sonata, for example, has the *performer* as the "artist", instead of Beethoven. Last.fm and Allmusic.com might not have data for that performer, so no tags are available for the artist, even though those databases might have data for the composer.

For tracks which failed in analysis, I experimented with giving them the mean values of all the other tracks in the data set. So, if MusicBox encounters a song with an unextractable Rhythm Histogram, it gets a Rhythm Histogram with the mean values of all other Rhythm Histograms in the music collection. However, this causes the unprocessed tracks to always appear together in the visualization space, since they all share the same Rhythm Histogram values. This interferes enough with user understanding of the visual mapping that I decided to remove those tracks entirely.

For other types of failures, it's acceptable to just give tracks empty data for the failed features. For example, if an artist is not found in the Allmusic.com database, there is no need to remove it from the data set. Instead, the failed track simply receives no new mood assignments (getting 0 for all known moods). Since lack of assignment to a mood is indicated by a 0 value anyway, this (0 setting) is not inconsistent with the rest of the data set, and does not result in a skewed visualization space.

To summarize, MusicBox needs to deal with incomplete data sets because some of its resources may not be fail-safe. Depending on the feature, MusicBox may remove the track from its model or give the track empty data for the errant feature.

5.3   Organization: Dimension reduction

Dimension reduction is the process of mapping a multidimensional space into a space of fewer dimensions. MusicBox needs to reduce an n-dimensional space (where n is the number of features) to a 2-dimensional space appropriate for viewing. To do this, MusicBox employs a linear transformation called **principal components analysis** (PCA).

5.3.1 Principal components analysis

Applying PCA to reduce a data set is like looking at a shadow of the data set from the most informative viewpoint. The projection of each data point (in this case, each track) from the higher-dimensional space to its location in the shadow is calculated by combining the data point's feature values according to the techniques described in Appendix A. The reduction mapping attempts to retain as much variance as possible from the original data set. In the MusicBox application, using PCA means this: songs that have similar sets of underlying features in the higher-dimensional space end up close to each other in the lower-dimensional space, and songs that have more different sets of underlying features end up farther apart.

PCA is the main linear technique for dimension reduction. It is linear because the resulting axes (called "principal components") are linear combinations of the underlying features in the higher-dimensional space.

MusicBox could instead have implemented non-linear techniques or clustering methods, such as support vector machines (SVM) or self-organizing maps (SOM). But similarity models based on SVMs are known to result in tightly clustered data sets that are not well-suited for exploration and visualization [47], and many MIR projects have already explored looking at music data with SOMs (see Section 3.3). I also shied away from methods focused on clustering because MusicBox is not ultimately about forcing tracks into musical classifications, but rather about representing the patterns within a musical set (for which PCA is quite appropriate). A similarity-based algorithm, such as multi-dimensional scaling (MDS), could certainly be applied to MusicBox's data set, and is a future extension of this work (Section 7.2).

PCA allows the use of variables measured in dissimilar units, which is absolutely necessary with a varied data set like MusicBox's. In particular, MusicBox can combine quantitative data (its content-based feature values) with more qualitative data (its contextual feature values), as long as the contextual feature values can be translated into a quantitative set. In MusicBox I achieve this translation by representing the qualitative contextual feature values as boolean categories, to which each track is assigned a zero or a one. In the PCA code, all data is mean-centered and normalized before the reduction is applied.

As I mentioned, PCA results in new axes for the lower-dimensional space, called **principal components** (PC). Each of these PCs can be interpreted subjectively as a new, conceptually more coherent feature; for example, a PC that combines tempo and loudness might be interpreted as "energy". (This method of subjective interpretation also applies to MDS [58, page 35].) However, interpreting a PC's meaning is not always so easy (or immediately obvious), and sometimes we must forgo a description or name for an axis and instead settle for only a general feeling for its meaning. I investigate users' interpreted meanings of MusicBox axes in Chapter 6.

PCA gives results as PCs in order of their importance (with respect to their variation) [59]. So we choose a subset of these sized to match the number of dimensions of the output, given our intended use. In MusicBox we look at data in two dimensions, so we select the first two PCs as the axes for our space.

5.3.2   Limitations of PCA

PCA is an optimal linear transformation for retaining as much variance in the original data set as possible. Its appropriateness assumes that the *information* in the data set is represented by its variance, such that the noise in the data will be filtered out to lesser principal components. Data sets must have a high signal-to-noise ratio to make this a reasonable assumption. Fortunately our feature extraction tools are refined enough that this is the case.

PCA can also be computationally intensive when compared to techniques like the discrete cosine transform. Devising more efficient algorithms for computing PCA, such as iterative methods, is an active area of research [60]. However, at the scale at which MusicBox currently operates – on libraries of up to 2000 songs – computational requirements are not a problem.

As I mentioned, PCA compressions often incur a loss of information. This happens not only because variables are often independent, but because in the compression we move to such a lesser-dimensional subspace that even higher PCs (such as the third and fourth PCs) must be dropped. In the implementation of MusicBox, I monitored the amount of variance retained in the two PCs used to map the space, and this was consistently between 50 and 100%, depending on the features included.

Another issue that arose in MusicBox's development occurred when large feature sets (such as timbre, rhythm histogram, or user-assigned tags) would dominate a mapping for a music collection. This happens because PCA weights its component features equally, so feature sets with numerous elements could easily outweigh smaller sets when both were included in the feature list. A good example of this is the "streaking" of albums in some feature arrangements. In this "streaking", tracks that are part of the same album occur in a line across the space because they all have the same values for a large set of features. For instance, Allmusic.com moods are only assigned at the album level instead of the track level, so all tracks on a given album would contain all the same moods; if there are enough mood features, their activation results in a mapping in which one principal component becomes devoted to accounting for their values, which clusters all same-album tracks along that axis. One could compensate for such behavior by adding user-adjustable weightings for features in MusicBox, or by scaling down the normalized feature data proportionally with feature set size.

5.4 Visualization: Enabling interaction

After applying PCA, MusicBox's data set has been reduced to two dimensions, ready to be mapped into a visual space.

5.4.1 Why 2D?

Early versions of the MusicBox application were in 3D. Using three dimensions would have allowed MusicBox to display more information by not reducing the space by another dimension. However, developing a good 3D user interface could be a thesis on its own; keeping the interaction intuitive, keeping the user oriented, not obscuring information, and having the application run efficiently were all concerns. Instead of focusing effort on making a good 3D interface, I chose to make MusicBox a 2D interface, and to spend the saved time making the application more fully-featured, in order to demonstrate the full power of exploring a music collection in a space.

5.4.2 A tour of the user interface
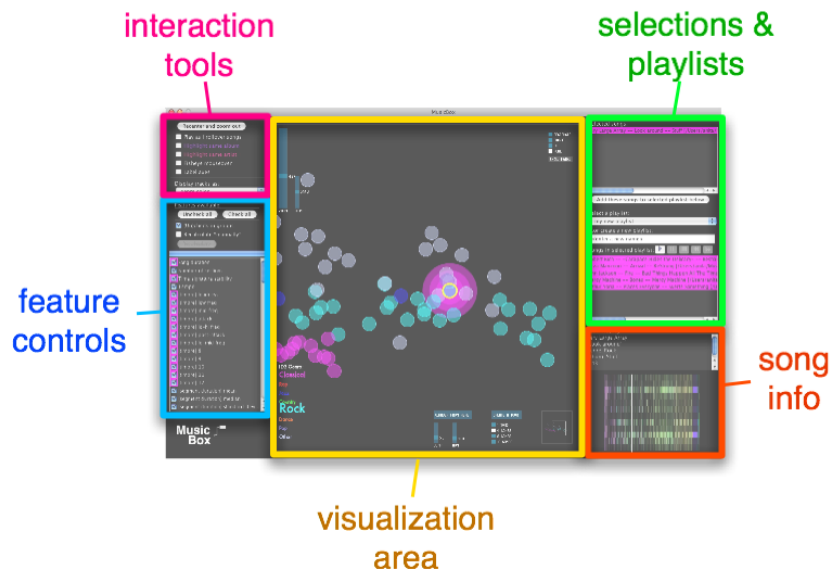
Figure 5-3: User interface elements. The MusicBox interface is broken up into separate areas, each focusing on different tasks.

Figure 5-3 shows the MusicBox interface broken up into several regions of interaction. The main visualization area, showing the music collection as an interactive map, is in the center of the

application window. To the left of the visualization area is a set of tools that affect user interaction with the visualization space, and a set of very fine-tuned feature controls to change the mapping of the space. Along the right are a selection box, playlist controls, and an area displaying detailed information about the current song.

Visualization area

The largest area of the MusicBox application is used for visualization of the music library. Each song is represented as a circle in the space, with its location determined by its feature values. Because of the way PCA reduces the space, songs that have similar values for the included feature set are mapped close together, and songs with more different feature values are mapped farther apart. *Therefore, the guiding intuition in navigating the music space is that songs that are close together sound similar, while songs that are farther apart sound more different.*
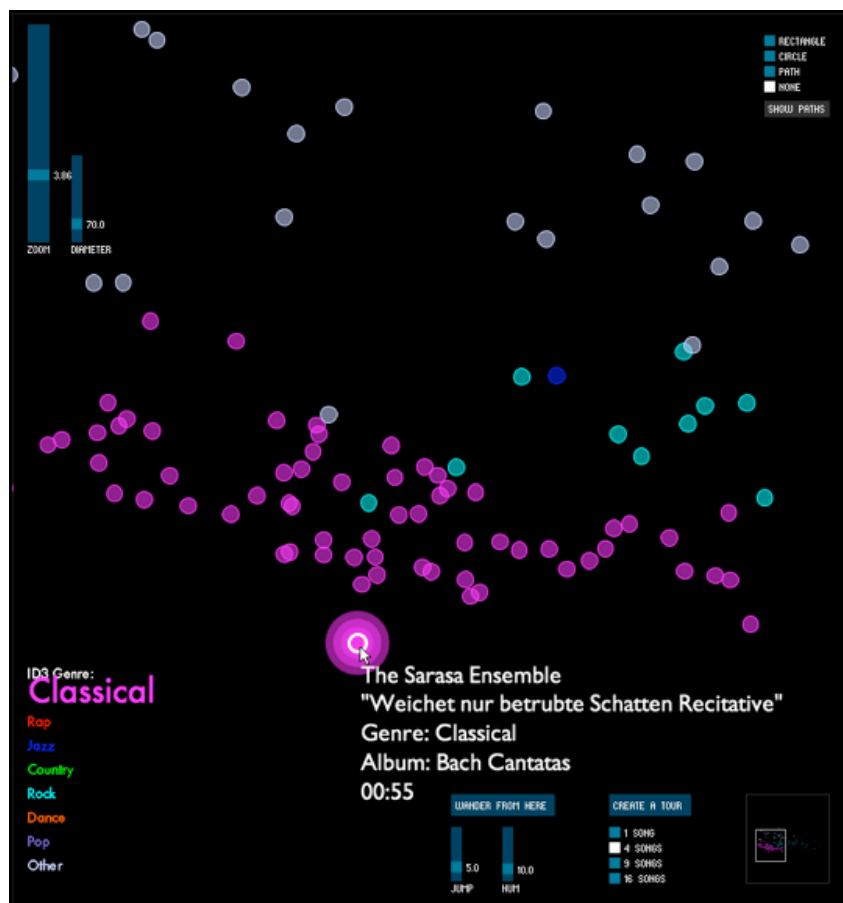
Figure 5-4: The visualization area. Songs are displayed as circles arranged according to musical content.

Users interact with the visualization like a map. They can zoom in and out with the mouse scroll wheel or with the zoom slider in the top left of the visualization area. Users can pan by dragging the space, and re-center by double-clicking. To keep the user aware of his current viewpoint, a mini-view of the music map is always displayed in the bottom-right, with the current viewable area outlined in white. Panning is also possible by clicking and dragging this outlined area in the mini-view.

Moving the mouse cursor over a song circle brings up the song's metadata: artist name, track title, genre, album, and track length. Clicking once on a song circle starts MP3 playback of the track.

In the default display mode, songs are shown as filled circles color-coded by genre (as defined in the ID3 tag for each audio file). A genre legend is drawn in the bottom-left of the visualization area to illustrate the genre-color mapping. MusicBox displays these genre names as a tag cloud, with the most prevalent genres being displayed in the largest font. The size of each genre tag is updated as the set of viewable tracks changes. The genre list and color mapping is hard-coded: this selection of genres was made in an attempt to indicate the most basic genre set: Classical, Rap, Rock, Pop, Country, etc. Color-coding songs by genre acts as a bridge for users to start using MusicBox from a familiar vantage point (genre), at the same time demonstrating the inherent inaccuracy of these genre labels.

Alongside the zoom slider at the top-left is another smaller slider that allows the user to control the size of the song circles. Users can diminish song size if they desire non-overlapping songs in crowded areas, or enlarge song size if they desire more song image detail while in other, non-genre display modes (see "Interaction tools" subsection).

Also contained in the visualization area are selection, shuffle, and tour functions (described in the next section).

Special features: Area selection, playlist paths, tours, and smart shuffles

Because MusicBox displays a music collection in a space, it has some special features that take specific advantage of organizing music in this way.

One obvious set of features is a group of selection tools. MusicBox has a path-select tool that allows a user to draw a path through the space, selecting all the songs that intersect with that path (Figure 5-5). This feature is especially useful for creating playlists with smooth musical transitions, since you can draw a line starting at a classical piece, for example, and meander over to the hip-hop area through some rock; MusicBox will select tracks in order along this path, and because of the inherent organization to the space, each new track will take one small step closer to the end point.

Figure 5-5: Path selection tool, as the user defines the selection (left) and the resulting selection (right). The user can select songs along a path through the space.

And what about clusters of songs? Say you've found a song you like, and would like to grab all the songs within a particular distance of that song. For this, you can draw a circle centered on that song, dragging out until an appropriate distance (Figure 5-6), and the resulting selection is easily made into a playlist. A rectangle-select tool is also provided in order to select subregions of the space (e.g. "the left section here that has a lot of rap music").
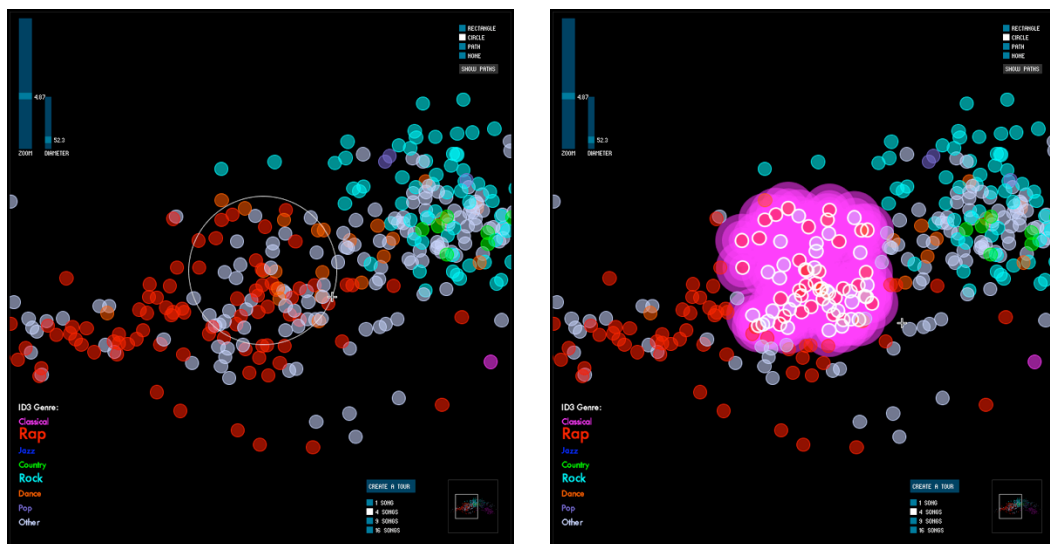
Figure 5-6: Circle selection tool, as the user defines the selection (left) and the resulting selection (right). The user can select songs within a particular distance of a given song.

MusicBox's "Create a tour" feature (Figure 5-7) is for finding one's way in an unfamiliar library.

The user selects a tour size (between one and sixteen tracks), and the feature selects that many tracks evenly dispersed throughout the viewable space. Creating a small tour from a library of two hundred songs you've never heard will pick out songs from the extremes of the library's sound. The "Create a tour" feature is applied to the *viewable area*, so users can home in on a particular region of the space and create a tour of just that area, if desired. It works by breaking up the viewable space into a grid and finding the track closest to the center of each box in that grid.
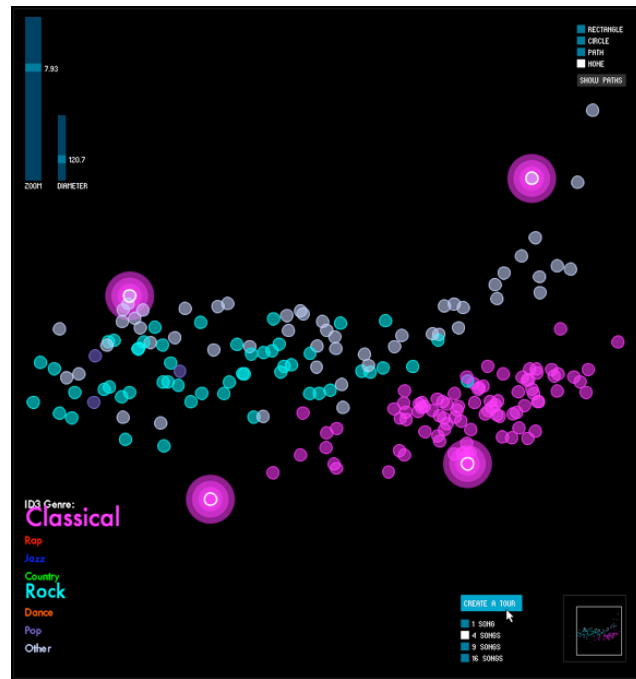
Figure 5-7: Creating a tour. A subset of songs is selected to demonstrate the variety of sound available amongst the viewable tracks.

One of the greatest drawbacks of the shuffle features in mainstream music browsers today is that they are completely random, which can result in shuffles with abrupt, sometimes jarring, transitions between tracks. iTunes allows users to specify how random their shuffles are, but this specification only applies to the likelihood of playing the same artist/album/group in consecutive tracks, giving no attention to the actual content of the tracks. Since MusicBox organizes music based on audio content, it can use the map to create smarter shuffles that have some degree of randomness, but that avoid these same jarring transitions.

A feature that demonstrates this kind of smart shuffle is the "Wander from here" feature. "Wander from here" creates a random walk from the currently playing track. The user specifies the length of the walk, and the distance by which the walk can jump from one track to the next. Because the direction MusicBox takes in making each of these steps is randomly chosen, many different random walks can be created starting at the same song. Figure 5-8 shows two wander

paths starting at the same song, each created with a different jump size. (Tracks are not allowed to repeat in any of these "wander paths".)
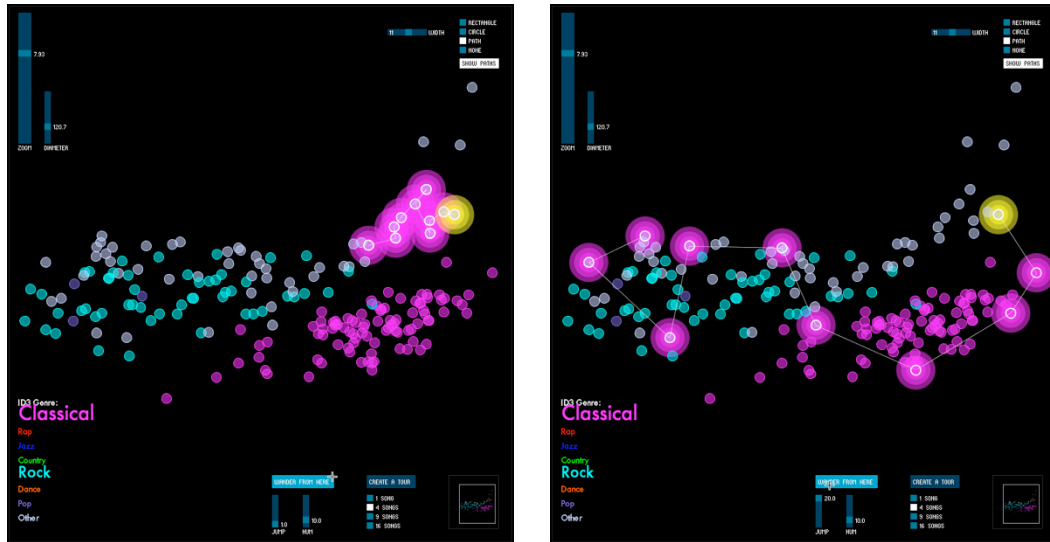
Figure 5-8: Smart shuffle tool, called "Wander from here". Random walks with the same start song (in yellow), using a small jump size (left) and a large jump size (right).

The resulting selection from any of these tools is a list of tracks that is placed in the selection box at the top-right of the whole application window (see "Selections & playlists").

Interaction tools

At the top-left of the MusicBox application window is a set of interaction tools (Figure 5-9). The title "interaction tools" here is vague, since these tools do a lot of different things to change the way a user interacts with the visualization area and aid them in making sense of the space. I will just describe each of the options in turn.

The "Recenter and zoom out" button is basically a "Reset" that brings the user back to where they were when the program started. This is helpful in situations in which a user finds themselves lost in an empty region of the space, for example.

"Play as I rollover songs" is a checkbox that starts playback of the corresponding MP3 the moment the mouse cursor moves over it. The default behavior is that track metadata is displayed with mouse rollover, but songs do not play unless the user clicks on the circle. With "Play as I rollover songs", interaction with the space is efficient (but noisier), enabling a much more immediate interaction with the space. (This feature is not on by default for performance reasons.)
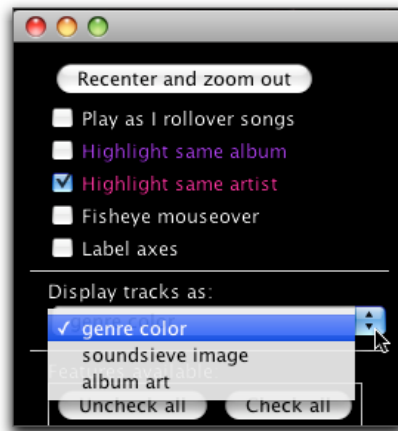
Figure 5-9: Interaction tools, a collection of tools that can help the user better understand the spatial organization of the map.
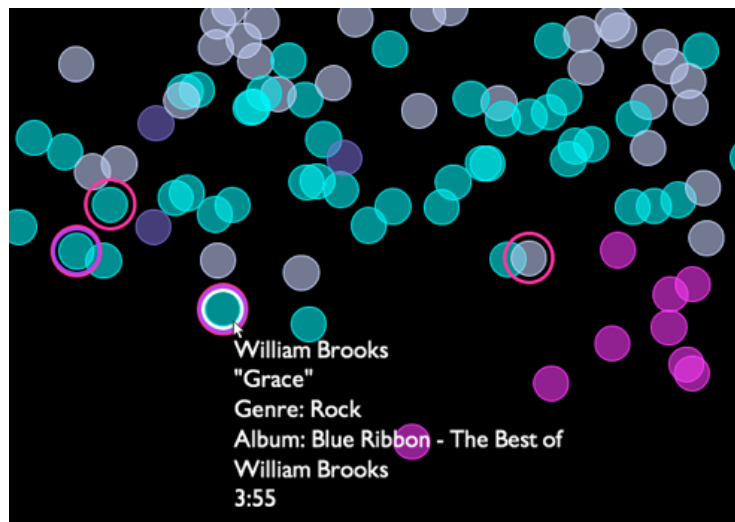
Figure 5-10: Highlight same artist/album. As the user mouses over a track, other tracks by the same artist (and/or on the same album) are outlined in a bright color. In this example, both "highlight same artist" (magenta) and "highlight same album" (purple) are turned on, so we can see that artist William Brooks has four tracks in this space, two of which are on the same album.

"Highlight same artist/album" augments the visualization area by outlining other tracks in the visualization area (Figure 5-10). As the user mouses over a track, other tracks by the same artist (and/or on the same album) are outlined in a bright color. This is useful in giving users an idea of the expanse of an artist's music in the space. One might, for example, find that one artist's tracks are tightly clustered and that another's lie across a variety of subregions of the space. The feature can also help when looking for other music by the same artist, since MusicBox does not (yet) have a text-based search.

86

Sometimes the visualization area can contain regions that are crowded with tracks, because there are many similar-sounding tracks given the current mapping. (I saw this happen frequently in the rock-like areas of my test libraries.) In order to deal with that crowded data a bit better, MusicBox has an option for a "Fisheye mouseover", which acts as a magnifying glass when mousing over tracks in the space and literally pushes crowding songs out of the way. The fisheye view operates smoothly, but still needs some work to enhance its usability.

The "label axes" checkbox, when enabled, causes the axes in the visualization area to be labeled with the underlying audio features that contribute most to each. In PCA terms, this lists the top features that contribute to the variance represented with the first two principal components. Three features are listed for each direction on each axis, for a total of twelve features listed. For example, if the first principal component (x-axis) is a linear combination to which tempo, song duration, and time signature stability contribute most positively, those three features will label the positive side of the x-axis. Correspondingly, the features that contribute most negatively to that PC label the left side of the x-axis. Figure 5-11 shows a screenshot of the visualization area with the "label axes" feature turned on.
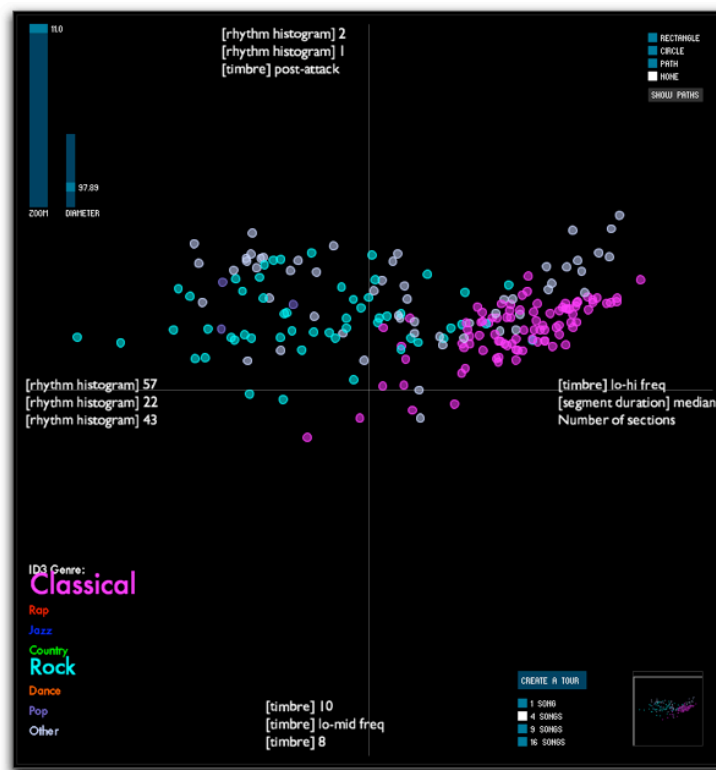
Figure 5-11: Label axes. This option displays the underlying audio features that contribute to each axis definition.

Interpreting these axis labels can be tricky. As I mentioned before, describing their meaning is a bit of an art, and sometimes cannot be captured succinctly. In general, MusicBox should not

force the user to attempt to interpret the axes defining the space. Rather, it is the distances between songs that should impress the meaning of the space on the user. Therefore, the "label axes" option is provided only for the most inquisitive, analytical user.
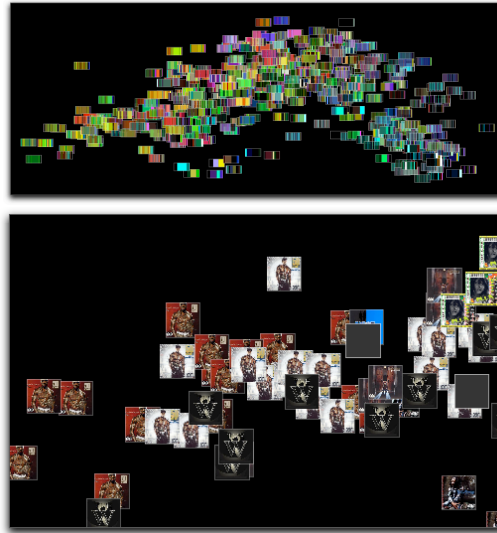
Figure 5-12: Display modes. "soundsieve image" mode shows simplified *soundsieve* images (top); "Album art" mode shows album art (bottom).

The final set of features in this part of the interface determines how each song is displayed in the visualization area. The display options are located in a drop-down list labeled "Display tracks as:" at the bottom of Figure 5-9. By default, each song is a filled circle color-coded by genre, which is the first of three options ("genre color"). The other two options are "soundsieve image" and "album art", pictured in Figure 5-12. "Album art" displays each song as a small image of the album of which it is a part. The "soundsieve image" mode displays each song as a summarized thumbnail of the more intricate *soundsieve* image a user would see in that project (Chapter 4). These simplified images show the song broken up into sections (like chorus and verse), each section filled with its respective timbre color (same as in *soundsieve*). Seeing these *soundsieve* images can give the user a sense of the organization of timbre content across the space, as well as a sense for the busy-ness of a piece, which can be inferred from the number and length of the song sections.

Feature controls

MusicBox's mapping is all about the features. Every song in the space has a value for each of these features. If we were to remove any feature, the song mapping would change. Feature controls give users the power to see how that mapping would change. This is particularly important since users have different opinions about which features are important, and the same user might be interested in one set of features on one day, and a different set on another.
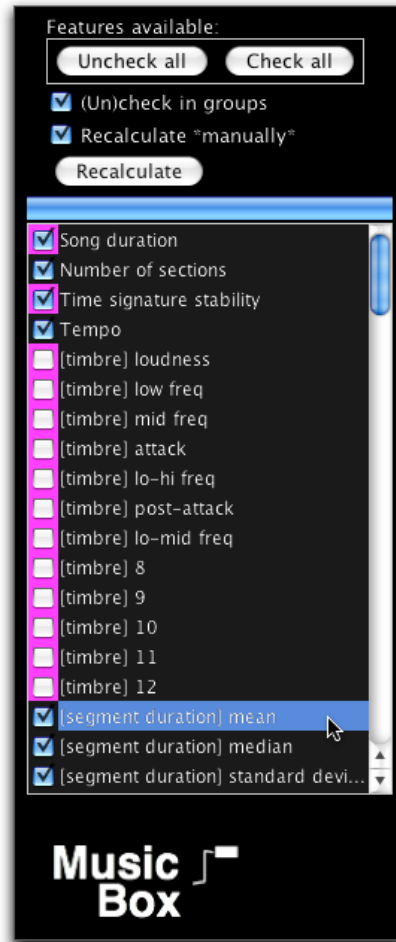
Figure 5-13: Feature controls. Users can adjust which features contribute to the song mapping. Activating or inactivating features changes the shape of the space; users watch songs move to their new locations.

The first mapping a user sees in MusicBox is one that comes from a PCA of all possible features. But users can adjust which features are included in the PCA by selecting or deselecting them in the feature list. Each time the feature set changes, MusicBox performs a new PCA to remap the whole space, and animates the shift of each song from its old location to its new location. Animation is very important in giving the user a sense of continuity as the space changes shape.

Because some features make the most sense when taken together (e.g. timbre mean values across a set of basis functions, or the separate bins of a rhythm histogram), MusicBox lets the user "(Un)check in groups" to add or remove feature sets as a unit. For example, in the feature list in Figure 5-13, all of the timbre mean values have been removed from the MusicBox.

Giving the user such precise control over the features used in the MusicBox allows for some
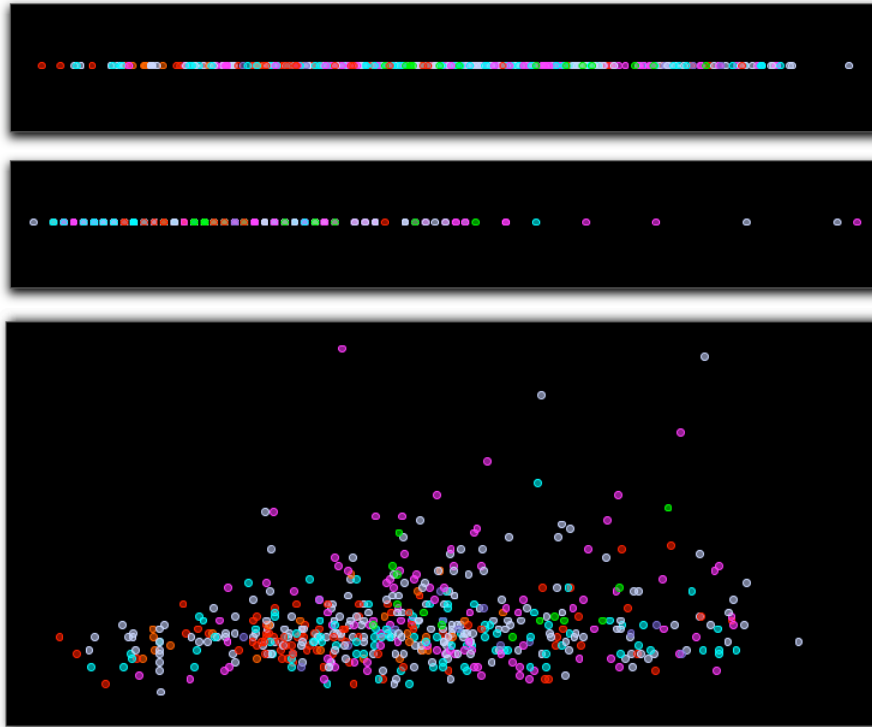
Figure 5-14: Maps using only one or two features. The top image is a music collection viewed according to only one (float-valued) feature. The second image is the same music collection mapped by an integer-valued feature. The larger image is the music collection mapped with both features at the same time.

interesting music maps. Want to see all of your music library mapped by tempo alone? Just click "Uncheck all" and then add the tempo feature. The resulting graph might look like the one in Figure 5-14. Mapping based only on an integer-valued feature (such as "Number of sections") would look like the middle image in Figure 5-14. Putting the two together results in the map at the bottom of the figure.

When the feature set is restricted to just one or two features, no PCA is necessary because the number of dimensions does not need to be reduced. This means that a music map made from just one or two features displays all songs directly along those features as axes. So it's very easy for a user to make simple graphs showing the distribution of their music collection across selected variables.

Being able to select subsets of features and remap the MusicBox space is a key part of the MusicBox experience; it's essential in encouraging users to explore the relationships between songs and their features. By providing this flexibility, MusicBox allows users to explore their music libraries at a depth that no other music browser permits.

Selections & playlists

Whenever a selection is made in the visualization area (using circle, rectangle, path, or single-selects), MusicBox displays the resulting list of songs in the selection box at the top-right of the application window (Figure 5-15). Users can play the songs in this list by double-clicking on the song titles. In addition, any selection of songs (single or multiple) in the list highlights those songs in the visualization area. This way, the user always maintains a connection between the text-based lists and the music space.
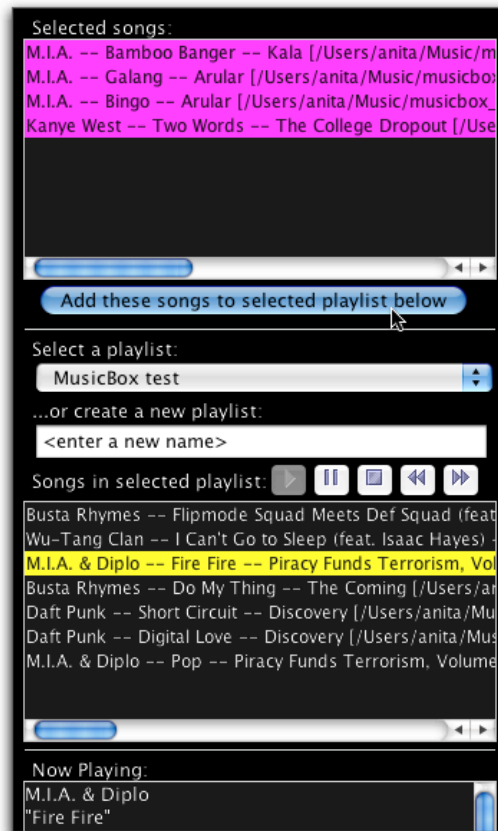
Figure 5-15: Selection box and playlist controls. Results from all selection modes (circle, rectangle, path, or single-selects) are displayed in the selection box at top. Items can be reviewed, reordered, and added to the selected playlist. Basic playlist playback is provided.

Once songs have been reviewed (and reordered if necessary), they can be added to the selected playlist. The playlist area (bottom of Figure 5-15) has the same playback functionality as the selection list above it, along with some simple playlist controls to advance from track to track. When left alone, a playing playlist will just play through all of its tracks in order.

MusicBox's playlist creation tools are fairly limited: They only provide the ability to play back and reorder songs. Functionality such as adding and deleting from a song list has not yet been

added.

Song information

Any time a user plays a song in MusicBox, the bottom-right of the application window displays information about the track (Figure 5-16). This includes the textual metadata for the song (i.e. artist, track title, etc.), but also displays the *soundsieve* image of the song. As the song plays, a vertical white bar moves across the image to indicate the current playback location. One of the most urgent upcoming additions to the MusicBox interface is the ability to jump to a particular point in a track by clicking on that location in the *soundsieve* image.

Figure 5-16: Song information. Metadata and *soundsieve* image for the current (or most recent) song are displayed. A white vertical line moves across the *soundsieve* image during playback, indicating the current location in the song.

5.5   The best new things you can do in MusicBox

MusicBox's arrangement of music in a space allows it to do some sophisticated things. These actions would be impossible without its system of organization for the music library. Additionally, the visualization of this organization model provides the user a context for the decisions they make while using the specialized features.

Some features that were mentioned in this chapter that are specifically made possible by MusicBox's organization are:

- **Tours for unfamiliar libraries** – The space gives a structure to the library from which MusicBox can select musically different tracks as a representation of the diversity of music

present in the library. The inherent structure also serves as a guide to the user when exploring a new music library.

- **Smart shuffle** – MusicBox's "Wander from here" feature takes a random walk through the space, moving gradually from one type of music to another, avoiding the sometimes abrupt and unpleasant transitions one might get when randomly shuffling a music library. Users can adjust the smoothness of these musical transitions.

- **Playlists as paths and areas** – Because similar musical pieces are located close together, a user can select subregions of the space that will naturally be cohesive. And, just as "Wander from here" defines a random walk through the space, users can employ the path selection tool to draw their own path through the space.

Other features that make MusicBox stand out when compared with other music browsers are:

- **Dynamic feature selections** – MusicBox invites users to manipulate the included feature set. Users can turn features (and feature sets) on and off, and MusicBox instantly remaps the space accordingly. This fine-grained control enables users to investigate underlying patterns more closely. It also increases the transparency of the MusicBox system.

- **Song-level visualizations** – As users explore the space, *soundsieve* visualizations give them an instant impression of song content. As we will see in Chapter 6, the incorporation of these images adds more context to the space, and helps drive users' exploration.

One thing that MusicBox succeeds in demonstrating is the limited helpfulness of genre labels. Genres are by nature fuzzy categorizations; in MusicBox the user literally sees how unfocused these categorizations may be. One can also see how some genre assignments are simply inappropriate (e.g. a "rock" track that is really more correctly described as "ambient"), and how MusicBox is able to compensate for missing genre tags by placing unlabeled tracks in the neighborhood of their musical counterparts. Because MusicBox departs from categorizing music into strict lists (as we see in mainstream music browsers) and has a way of displaying more fuzzy relationships, it deals better with a complex data set like a music library.

MusicBox can also help a user rediscover forgotten music in their own libraries. Say a friend gave you an album years ago that you never really took the time to listen to, but it just happens to have a lot of the same audio characteristics as your favorite punk album. An interface like iTunes keeps you ignorant of this forgotten album, and unless you have the genre set to "Punk", you would quite likely never find it again. MusicBox, on the other hand, will display this album right next to your favorite punk album, since it is so similar in sound; drawing attention to it so that you just might listen to it next time you're "in the area."

MusicBox's potential to help a user explore an unknown library, make playlists, and rediscover forgotten music are all things I attempt to evaluate in the next chapter.

5.6   Implementation notes

MusicBox is written in Java 6, and uses Processing[5] for drawing the visuals. Its dependencies are written in Python (for creating the cached *soundsieve* thumbnails) and Perl (for uploading and retrieving analysis result files from The Echo Nest). Audio analysis depends on The Echo Nest Analyze API [56] and Rhythm Histogram Matlab code from the Vienna University of Technology[6]. Jonathan Hilliker created the icons used for the playlist playback buttons.[7]

---

[5] `http://processing.org/`
[6] `http://www.ifs.tuwien.ac.at/mir/audiofeatureextraction.html`
[7] `http://www.codetoad.com/java_mp3_player.asp`

Chapter 6

Evaluation and Discussion

Visualization is the path to madness.
– Annina Rüst

6.1   Evaluation design

The MusicBox project aims to demonstrate these things:

1. There is value in visualizing music in a space.

2. In navigating that space, the user develops a mental model of its organization, and trusts that model.

3. Users trust the mental model enough that they can use it to make a routine task easier than it would be in a mainstream music browser.

4. A map like MusicBox's can help users discover forgotten or unfamiliar music in a library.

In order to test each of these items, I designed a small user study that touches upon each of them. Briefly put, I spent an hour each with ten participants, spending about 20 minutes each on three things: giving a tour of the MusicBox interface, having the user explore an unfamiliar library in MusicBox to determine its content, and having the user create playlists in MusicBox and iTunes. I gave each subject a questionnaire, the results of which are aggregated in the chart in Figure 6-1.

To address item 1 above, I directly asked users whether they thought there was value in visualizing music in a space. For item 2, users were asked to describe their mental map of the space, and to comment on how they used this mental map to guide their actions in the

MusicBox software. For Item 3, the user study contains an A-to-B comparison of MusicBox and mainstream music browser iTunes in the task of creating a playlist; after users completed the playlist creation tasks, I asked them whether they thought MusicBox made the tasks easier or harder. And finally, to test whether MusicBox helps users rediscover music in a library, I compared the playlists they created in MusicBox and iTunes to see if either one consistently included more unknown tracks.

This evaluation process was fairly qualitative, and should not be interpreted as a rigorous study. Even so, the results are promising.

6.2   Evaluation results

6.2.1   Visualizing music in a space

The most important point I wanted to make with MusicBox is that there is indeed value in visualizing music in a space. Spatially representing songs emphasizes the similarities between songs and the fuzziness inherent in categorization of music by genre, artist, or even album.

When asked directly how they felt about using a spatially-organized music collection ("I think the concept of visualizing music in a space is valuable"), users overwhelmingly (9 out of 10) expressed strong agreement. This was the most positive result from the whole questionnaire, and shows strong support for the value of visualizing music in a space.

Users also agreed that simple genre labels in traditional music applications are insufficient when navigating a music library. As one user noted, after making the transition from MusicBox to iTunes, "There's a difference between 'genre' and *what [a song] sounds like*... genres don't really mean that much." Another user said that the grouping of the genres, which could also show where they overlap, was preferable in MusicBox.

More importantly, users got a sense of *structure* in the visual space, which we will see in more depth in the following sections. "I really liked the way [MusicBox] made similarities between songs obvious," one user wrote in their comments. Another user wrote that they liked MusicBox because it "arranges data more usably and intuitively than traditional interfaces." MusicBox's visual representation was also easier for some users to manage when compared to interfaces that don't take advantage of our visual acuity: "[MusicBox] is more intuitive, visually intuitive." With this I saw the true reward in creating the MusicBox application: that users can easily create *an intuition* for the space.

6.2.2   Users' mental models

To examine the kinds of mental models users created while navigating the music space, I asked users to draw a map of the space on a piece of paper after exploring an unfamiliar library (of approximately 200 songs) for ten minutes. The task was open-ended; subjects could draw and label the map with as little or as much detail as they chose.

All users drew a map and expressed little difficulty in completing that task. Results from the questionnaire back up users' confidence in the organization of the space, and the ease with which they were able to discover it.

Of course, all of the user maps were different, some of them remarkably so, as each user had a different mental representation of the space. One user described one area as "punky-trance" and another as a "breathy peninsula" (which happened to be filled with ambient music and didgeridoo); other users placed more emphasis on the underlying audio characteristics of the songs, using phrases like "fast, harsher timbres" and "slower, calm beat". These kinds of differences in describing music are unsurprising; we know that people use markedly different words and ideas to describe music, and in fact, it is a premise of this thesis that that difference is why it is particularly hard to find and organize music. *By avoiding the use of and requirement for words to describe music, instead simply presenting a map, MusicBox lets the user decide how they will interpret the map and infer meaning from it.*

Ease in creating a model

How easy was it for users to create their mental models of the music space? The results for this question ("I found it easy to discover the organization of the MusicBox space") were not as overwhelmingly positive as the responses for the value of music in a space. This less enthusiastic (although positive) reaction could certainly be due to users' unfamiliarity with the MusicBox interface (they answered this question early in the 1-hour session), but is also likely influenced by the limitations of some of the features in MusicBox, along with the inherent complexity in making sense of a diverse music library.

One user expressed that, although the general layout of the space was clear, "discovering the subspaces is still mostly manual." Another user expressed a similar but more general concern: "I couldn't be listening to a song on the left and using only that, tell you what a song on the right would sound like." This concern is understandable, and it's unrealistic to expect that this will ever be untrue; devising a mapping for music that is so intuitive that seeing a song in its place will ever substitute for having heard it is impossible. Even if discovering the content and structure of the subspaces is manual, MusicBox doesn't pose any more difficulty than a user would already have in any of the other music browsers described in this thesis.

Some users, despite no axes being drawn or labeled in the space, attempted to derive a definition for them. One user wrote, "I was trying to discern some parameters of the space. There are amorphous regions with some clear relationships, but it is difficult to articulate what that relationship is." Another user commented, "I know there is a pattern, that is easy. I had difficulty trying to verbalize what that pattern is." This is exactly the difficulty that I would expect in trying to describe the axes of the space; even looking at the contribution of the various features to the principal components often makes this no clearer. However, the first user went on to say that, "I don't think that is necessarily a bad thing for general exploration once I become more familiar with the space." Thus, the user implies that, even though articulating a clear definition for the axes was difficult, he can eventually get a more general feeling for the organization of the space.

One user had a more negative response to the interface: "The navigation was difficult and not 'logically' laid out / labeled like a Windows program." It's true that MusicBox is not a polished piece of software, and that the interface itself can hinder a close understanding of the space. Even so, only one user made a comment to this effect.

Trust in the model

Once users came up with a mental model for the space, how much trust did they place in it? I will try to gauge users' level of trust by the actions they took and the assumptions they made while navigating the space.

Most of the users took clear actions that resulted directly from assumptions they made about MusicBox's mapping. For example, upon finding an artist or song they liked, all users looked in the immediate vicinity of that song to find other songs they would like. Some users also exhibited the complementary behavior: they stayed away from entire regions of the space because those regions contained an artist they knew they didn't like (this happened consistently in response to finding anything by the Backstreet Boys), or a genre they didn't identify with. This pattern held true even when MusicBox's mapping pointed out exceptions; for example, users would ignore a rock track in the classical section altogether, even though it was clearly marked as rock. (In the test libraries, a rock track in an area that is predominantly classical always contained sounds that made it sound more "classical" than "rock".)

One user told me that he had an "intuitive understanding and felt comfortable navigating in the space." He was so confident in that navigation that he said he had formed an *expectation* for what he might find in a particular area.

Half of the users (5 of 10) trusted the *tools* in MusicBox enough to let them guide their exploration of the space. For example, all of these subjects used the "Wander from here" tool at some point in order to give them a plan of attack for their listening. Some would click the "Wander..." button until they found a path that seemed to agree with where they had hoped to

"go", and then listened through each track in the resulting path. During the playlist creation task, three of these users, having found seed songs that they liked, used the "Wander..." tool to get more songs like that song. When I asked one of the subjects why he didn't just use the circle or rectangle selection tools to retrieve songs nearby, he said that he "didn't want to limit [himself] to such similar-sounding songs [as he would get with those more focused tools]." Since this user is even drawing distinctions between close areas in the space, he places an impressive amount of trust in the organization of the system.

One very important observation is that users expressed they would have more trust in the system if they had more control. Four of the subjects told me that they felt they could get a better understanding of the space if they could look at their own music, since they are intimately more familiar with it than the test libraries. Two users specifically asked to be able to control the organization of the space: to be able to increase the importance of a particular feature, or to be able to move a song if they disagreed with its default placement. Therefore, a future addition to the MusicBox interface is a set of feature influence controls along with a user-adjust mode that allows users to directly relocate songs to their liking.

6.2.3   Using mental models to make routine tasks easier

Now we'll look at how subjects used their mental models of the space to create a playlist. Can this routine (for a music browser) task be made easier if users have a model of the musical space?

The questionnaire results showed that 7 out of 10 subjects felt that creating the playlist was *easier* in MusicBox than it was in iTunes, even with MusicBox's limited playlist creation tools. When given the task of making a playlist in iTunes, one user, who was unfamiliar with most of the music in the library, sighed and said, "This is going to be ridiculously bad compared to the other [MusicBox]... You'd have to know the music before you could make a playlist." Other users with the exact same problem had similar things to say: "See, I wish I had some indication of... tempo, sound... *something*. This tells me absolutely nothing about either of those; I'm making inferences solely on genre and duration of the piece... I don't know where I want to be." "[Making a playlist in MusicBox] was really easy. Making the iTunes playlist was difficult and unpleasant because I had only the titles and names to go by. I ended up clicking at random until I found something similar." Harking back to the trust in MusicBox's organization, one user wrote, "I was happier making the playlist in MusicBox because I could be more confident that my selections would work."

Of course, these comments would have been very different if users had been working with their own music libraries; a lot of these problems are exacerbated when the music is unfamiliar. "Not knowing most of what this is, it's daunting to go through an entire list of songs and artists." This

phenomenon emphasizes MusicBox's value as a discovery tool, since it gives users an intuition for navigating a large body of unknown music.

As for the content of the resulting playlists (as opposed to the process of creating them), 6 of 10 subjects were *happier* with their resulting playlist in MusicBox than iTunes. This doesn't show a particularly strong favorability for MusicBox, but at least it shows that perhaps it would be a worthy competitor to a program like iTunes.

The most important thing we glean from assessing users' reactions to creating playlists in MusicBox and iTunes is this: There is clearly a need for new methods beyond what exists in mainstream music browser technology. Every user in the study agreed (albeit two of them with caveats), that they would consider using MusicBox in place of their current media player.

6.2.4   (Re)Discovery

Several users were struck by the fact that they were *noticing more music* in MusicBox than in iTunes, specifically that they were more frequently listening to (and adding to playlists) unfamiliar songs while using the MusicBox software. Users almost completely ignored unknown tracks while using iTunes because there was nothing encouraging them to pay attention to those tracks. Nor was there any easy way to make sense of those tracks: "I'm just going to look at the songs I know because there's not really an easy way of doing it otherwise."

In order to test whether users paid more attention to unfamiliar tracks in iTunes or MusicBox, I needed a source of unfamiliar tracks. I sought out the Magnatune music database[1], which is a set of about 8000 DRM-free songs by independent musicians. Only one of the subjects claimed any familiarity with the non-classical Magnatune tracks, and his familiarity was limited to a minority of the artists included. (Classical tracks from the Magnatune library were often familiar, because they are performances of pieces by well-known composers like Johann Sebastian Bach.)

The test library used for the playlist task was a collection of just over 500 songs, about half of which were from Magnatunes. The other half of the tracks were by artists that were more likely to be familiar to the test subjects, such as The Beatles, Diana Ross, and Kanye West.

After each evaluation session, I reviewed the user's playlists. Twice as many (8 out of 10) subjects included Magnatune tracks in their MusicBox list than did in their iTunes list. Of the four users who included Magnatune tracks in their iTunes lists, two of these had started their list with a classical track (the only other classical pieces in the library were also Magnatune tracks), and the other two started with Magnatune tracks. So it's very clear that subjects paid unfamiliar tracks more attention when working in the MusicBox interface than they did in the iTunes interface.

---

[1] `http://magnatune.com/`

I also received consistent comments that MusicBox was helping users find new music. At least half of the subjects said something similar to this comment: "[MusicBox] caused me to play songs (and enjoy them) which I would never have thought to select by title alone." One user even suggested that MusicBox be built into a specialized recommendation interface, in which users suggest songs they like to MusicBox, and MusicBox gives back recommendations gathered from the locations of songs in its space. Another user noted that, for the large areas in the space filled with unfamiliar music, the interface "might help me figure out what I *do* like in those places."

In the same way that MusicBox can help users find completely new music, it can also help them *re*discover forgotten music in their libraries, much like I was able to find the one pleasing Kiss track amongst my Neil Young tracks (Section 1.3). If there is an album in your library that you haven't listened to in a long time, but MusicBox shows it next to your favorite Beatles album, you might just give it another listen. With a traditional text-based music browser, you'd likely never have noticed it again. And, if you forget what artist that was after listening, not to worry; it will still be there the next time you look at The Beatles.

6.3   Details of the evaluation process

Each 1-hour evaluation session followed this structure:

1. (20 min) Introduction to MusicBox with library of 140 unknown songs

2. (10 min) Open-ended task ("Task A"): Explore a library of 200 mostly unknown songs, and draw a map of the contents.

3. (14 min) Focused task ("Task B"): From a library of 500 songs, choose two songs (4 min) and then create a playlist:

    (a) (5 min) In iTunes, starting with one of the songs

    (b) (5 min) In MusicBox, starting with the second song

    The order of these two tasks (iTunes and MusicBox) and the seed song for each were determined by coin tosses.

Times listed above do not include time spent answering paper questions.

All but one subject were familiar with using iTunes. The media players used by study participants were iTunes (5 users), Windows Media Player (3 users), and Winamp (2 users). (One subject uses two pieces of software on a regular basis.) The subject unfamiliar with iTunes does not use any media player software.

In the focused task ("Task B"), subjects used MusicBox first to select two songs that would serve as seeds for their playlists. I required that they choose songs in dissimilar areas of the MusicBox space so that knowledge from creating one list in MusicBox would not carry over to the creation of the next list in iTunes, and vice versa. With the benefit of hindsight, I now know that I should have adjusted the design of this task; giving users time to explore the MusicBox space while selecting seed songs allowed them to gain more familiarity with the space, and that may have helped them in the MusicBox portion of that task.

Statement:		Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
After task A...	I am confident there was an organization to the MusicBox space.				●●●●●	●●●●●
	I found it easy to discover the organization of the MusicBox space.			●●●	●●●●●●	●
After task B...	I was confident there was an organization to the MusicBox space.				●●●●●●	●●●●
	Comparing the experience of creating the playlists, the task was easier in MusicBox than iTunes.*		●	●	●●	●●●●●
	Comparing the results in my two playlists, I was happier with my results in MusicBox than iTunes.			●●●⊖	●	●●●●●
	I think the transitions between tracks were smoother in the MusicBox playlist than the iTunes playlist.**		●	●	●	●●●●●●
	I think the tracks were a more cohesive set in the MusicBox playlist than the iTunes playlist.			●	●●●	●●●●●●
Final questions	I think the concept of visualizing music in a space is valuable.				●	●●●●●●●●●
	The space was easy to navigate.		●		●●●●●●●	●●
	I want to use a piece of software like this more.				●●●●	●●●●●●
	I would consider using this software in place of my current media browser/ player.			●●	●⊖⊖	●●●●●

⊖ = user gave answer with caveats: "I'd need the time to listen to pieces in their entirety to say whether or not I was 'happier' with the playlist in one or the other."
⊖ = user gave answer with caveats: "I don't use a media player." and "I would use this software *in addition to* my current media player."
* Only 9 results included. One user said the two conditions (MusicBox and iTunes) were too hard to compare.
** Only 9 results included. One user said that the question was not applicable since they were not trying to make smooth transitions between tracks.

Figure 6-1: Questionnaire results. Each user's response is represented with a filled circle.

6.4   Discussion

The MusicBox interface was well-received. In addition to several users telling me that MusicBox "looks cool", subjects volunteered consistent positive comments about interacting with the space. They said that the space helped them to remember which songs were which; they could return to a song by remembering where it was rather than what it was called. In addition, users' trust in the organization of the space led them to choose regions of interest and to stay focused within them for some time.

Many of the subjects commented on the usefulness of seeing *soundsieve* images when they played songs. They indicated that the images drew their attention to particular parts of the song, or helped them know when they had listened to enough of a track to get a good idea of its content. Almost every subject requested the ability to jump to a particular point in a track by clicking on a place in the image, and commented that the inability to do so was almost painful. This is a feature I intend to add to MusicBox in the very near future.

One subject used the *soundsieve* visualization mode, in which each song is displayed in the space as a summarized *soundsieve* image (Figure 5-12), to help him discriminate between crowded tracks in the space. Just as other subjects had relied on their memory for track locations, this subject combined images with those locations to make the track representation more unique, and therefore the memory more solid.

Two users paid a surprising amount of attention to *non-musical* elements of the MusicBox interface. One of them stated, "The names are interesting; I like looking at them in the list." MusicBox is less helpful for this particular preference, since it only lists track names when tracks have been selected in the space, and it doesn't have the space to clearly show a long list of track names. Another user, self-described as "very visual", looked closely at the *soundsieve* images when selecting each track, and in creating playlists looked for *images* that were pleasing instead of sounds that were pleasing. While it's interesting that this subject was motivated so strongly by the *soundsieve* images, this tactic doesn't take advantage of the MusicBox interface because it ignores the audio-visual mapping that makes MusicBox so powerful.

Although all users were given an introduction to the many features of the MusicBox interface (such as "Create a tour", selecting areas, and drawing paths), only a few of them used those features while creating their playlists. 8 of 10 subjects picked and added each song one at a time instead of using MusicBox's selection tools. Subjects exhibited this point-and-click behavior even though the features to which they had been introduced were specifically designed to do the same task automatically. For example, subjects would click out their own path, track by track, rather than drawing a path that followed the same trajectory. Assumedly this is because MusicBox's playlist editing tools are not fully developed and the individual point-and-click strategy gives users the most control over what goes in their playlists. It's also possible that users unfamiliarity with the interface (each feature had been introduced to them in about 2-3 minutes at the beginning of the hour session) caused them to simply not think to use the features in the first place. A more lengthy user study, perhaps over the course of days or weeks instead of one hour, would help shed some light on whether these features would eventually be accepted by users. And, despite users' low usage of MusicBox's tools, user behavior demonstrates that the intended purpose of these tools is well-guided.

Subjects who did use MusicBox's tools, on the other hand, placed a lot of trust in them from the start, seemingly convinced that the organization of the space made sense enough to do that. One user, in a rush to finish his playlist, went so far as to create a playlist using the path selection

tool, only choosing the start and end tracks, and just left it at that without having listened to the selected tracks.

One assumption users consistently made was that an artist's tracks are consistent with each other in sound; that is, it is appropriate to play any two of an artist's tracks together. Even when MusicBox showed two of an artist's tracks in very different locations in the space, users would listen to both of them when trying to come up with a cohesive set of tracks. This behavior provides an interesting counterpoint to how willingly users paid attention to unfamiliar tracks in MusicBox if they were located in a region of interest. The comparison seems to indicate that people have expectations about the meaning of song placement in the space, but that they can abandon those expectations when an old assumption is allowed to come to the fore. My hope is that innovative, provocative interfaces will encourage users to question their old assumptions and truly begin to think about their music in a new way.

Having addressed the validity of MusicBox's organization scheme, we must also discuss the problems that can occur when a system's model conflicts with the user's mental model. We cannot expect that a mathematically-created model will always make sense to any given user, and should provide them with the tools to adjust the model or to make a model of their own. MusicBox does not currently have these tools, but they will be implemented in the next revision.

Of particular interest is the complementary reaction: that users, when presented a conflict with their own mental model, attempted *to explain this discrepancy by adjusting their own model.* In the 1-hour sessions, a few users stumbled upon neighboring tracks that they did not expect would have similar sounds, such as Daft Punk's "Night Vision" being placed amongst classical tracks. Users expressed interest in this juxtaposition, and listened to the Daft Punk track. Instead of proclaiming that the "Night Vision" placement was in error, they would verbally express that it had soft, orchestral-frequency sounds, with a smooth beat that made the track not unlike classical music. In other cases, users would click on two songs close together and their immediate reaction was one of skepticism. ("See, I would not put these songs together. They sound pretty different.") However, these users would usually pause at this point, listen for a little while longer, and then conclude that, in fact, the two songs really do sound similar, just in a way they had not previously noticed. What we're seeing is that users' mental models for music are plastic; MusicBox's model can challenge them, and they can challenge it.

From all of these results, it seems that MusicBox has differing value depending on who is using it. In David Jennings's book *Net, Blogs and Rock'n'Roll*, he delineates four types of music fans: Savants (whose lives are defined by their relationship with music), Enthusiasts (for whom music is a key part of life but who are not defined by it), Casuals (who possess a passive interest in music), and Indifferents (who don't care much about music at all) [10, page 30]. Although MusicBox was primarily developed with the true Savant in mind, it can help all but the Indifferent user. Casual listeners can find value in using it as an explorative tool for learning

about musical realms they know little about, Enthusiasts can use it for discovering new artists and making playlists, and Savants can use MusicBox's more advanced feature-manipulation tools and song visualizations to truly personalize representations of their music libraries.

MusicBox helps users make better sense of their music collections, and perhaps to discover new artists given a bigger musical space. Even so, MusicBox does not necessarily help a user find *good* music. After all, defining "good" music is likely much more complicated than simple contextual and content-based descriptors could account for. The human connection with music is an age-old enigma that may never be "solved" by an approach like this one, but perhaps MusicBox takes us one step closer to making sense of it.

Chapter 7

Present and Future

7.1   Conclusions

With the proliferation of connected devices and easy file distribution, one can imagine that in the near future we will have every piece of music ever recorded available on demand. In the digital music world, this idea is known as the "celestial jukebox", a big cloud in the ether to which any of our devices may connect and retrieve any piece of music with the push of a button. The term "jukebox" clearly refers to music as its digital medium, but the phrase has grown to apply to all entertainment content.

Right now we are entering the time when this "celestial jukebox" is imminently possible. No longer does the difficulty lie in creating the technology to enable the jukebox; *now we focus on how to navigate that jukebox*. We will have *billions* of songs available, but how will we find our next favorite track?

MusicBox helps us to make sense of the immense, complex music cloud by giving us an intelligible context for exploration. As we saw, it combines both content-based and contextual descriptors for music to organize a music collection in a space that illuminates its structure. It can guide us in exploring an unfamiliar music collection, or give us a new way to envision the composition of our own music libraries. Moreover, MusicBox gives the user flexibility to control their interaction with the musical space, to manipulate its mapping in order to glean more information about its patterns. We also saw that MusicBox's representation of music encourages discovery, providing a solid framework within which recommendations can be made.

At MusicBox's core are three things: an extensible set of descriptive data, a mapping of music onto a visual space, and a highly-customizable, interactive visualization tool. All of these combine to create a new and valuable way to interact with a music library. In addition, each of

these steps is automatic, so that MusicBox does not exclude the long tail of music. Including all music at an equal level is of vital importance as that tail gets longer and fatter.

In its current implementation, MusicBox demonstrates the variety of musical descriptors it can contain; the existing feature set works well, but is just the beginning. We invite other MIR researchers to contribute descriptive features to MusicBox's PCA engine in an effort to improve its view of the vast musical space.

MusicBox also shows us a new way to interact with data that breaks free from the limitations of word-based queries and descriptions by approaching the music search problem visually. Its non-linguistic format can not only serve musical content better – because of the inherent difficulty in describing musical content with words – but can make the world of music accessible to a wider, international audience by removing the reliance on a specific linguistic basis for searching it.

7.2   Future Work

How can MusicBox be an even better tool for navigating the celestial jukebox? Here we briefly touch on some planned improvements to the MusicBox interface, and introduce new applications for it.

7.2.1   Improvements

Here are some planned additions to the MusicBox interface:

Turnkey system for library analysis – MusicBox's process for analyzing a music library should be streamlined and distributable.

Filters – Users can use filters to refine the viewable data, with the ability to "vote away" certain artists or tracks so that they no longer clutter the space.

Specialized recommendation tools – MusicBox provides a good framework for discovery of new music, but its interface could be improved with some tools made specifically for providing recommendations. For example, a "More like this" button would be beneficial.

New selection tools – Multiple point-and-select capability. MusicBox's playlist management can also be improved by expanding playlist editing capability.

New playback tools – Users should be able to jump to any point in a song by clicking on the corresponding location in its *soundsieve* image.

Automatic audio tours of the space – MusicBox already creates playlists to give users a tour of an audio space. It could also automatically create audio mixes of those lists as a more meaningful result of those tours.

Text-based search – Users should be able to search their libraries with text if they know what they are looking for.

Customization – Give users the ability to save favorite perspectives on the space, allow them to manually remap the space, and to adjust the amount that each feature contributes to the mapping.

New mapping strategies – MusicBox would ideally have an extensible set of mapping engines in just the same way as it has an extensible set of audio features. Multidimensional scaling (MDS) is one mapping strategy that would be incorporated.

Comparing libraries – MusicBox should make it easy to compare libraries from different sources.

Dynamic updates – MusicBox can reflect the dynamic online world of music. It could use RSS feeds to create a constant flow of new music into a library, automatically analyzing the available MP3s and placing them into their computed locations.

7.2.2   Extending MusicBox to a larger musical space

In this thesis, we've seen that MusicBox is an effective tool for small, personal music libraries. However, there is significant work in extending MusicBox to accommodate much larger data sets with upwards of a million songs.

Therefore, applying MusicBox to a data set like the celestial jukebox is a big step, and will focus on three concerns: the value of content-based data, the extension of PCA to large data sets, and the challenges faced when creating a user interface for large data sets. I'll discuss each of these in turn, so that we can see what is necessary to develop MusicBox into an access point for a large body of music.

Value of content-based data

Because of the aforementioned cold-start problem (Section 2.2.3), a system that intends to deal with the Long Tail of music cannot rely on listener-created data, because there may not yet be any listeners to create that data. So the system must be able to operate sufficiently on content-based data alone, which is most often extracted automatically from the audio files.

How useful is MusicBox when contextual data is removed? In the user study for this thesis (Chapter 6), most of the features used were content-based. This was because users were interacting with libraries containing a sizable proportion of Magnatune tracks, for which there was little or no Last.fm and Allmusic.com data; therefore I simply removed these contextual feature sets from the MusicBox for the course of the study. The only contextual features included in the study were Magnatune genres. Removing the Magnatune genre feature set from the data set changes the shape of the space, but only slightly; the space retains the same shape and composition without them. The PCA also retains more of the variance from the original data when these contextual features are removed, because the contextual features here are more independent from other features than are content-based features. That is, the 2-dimensional representation loses less data when contextual features are excluded.

The fact that MusicBox's musical space is still organized in a meaningful way when contextual features are removed means that MusicBox is useful even without them. And although further work would be necessary to more quantitatively evaluate the residual usefulness of the space after contextual features are removed, I don't expect the spatial structure to be significantly less informative, even with the simple set of content-based features currently implemented. What MusicBox would lose from lacking contextual data would be in flexibility: Clearly, users would not be able to filter and remap the space based on those features if they were not available. However, this sacrifice is one we must make when our data sets are necessarily less full.

Extending PCA to large data sets

In its current implementation, MusicBox uses a Java Matrix Package (JAMA) to perform PCA. I chose to use JAMA for MusicBox because it is freely available, and because it is easy to incorporate into one's code; therefore it was not a chore to build PCA into the MusicBox. However, JAMA is by no means the most efficient package to compute PCA. Even other Java libraries, such as ojAlgo[1], can compute SVDs on "tall" data sets (more rows than columns) like MusicBox's at rates ten times faster than JAMA's. Additionally, code written in other programming languages may offer even more speed: My colleagues in the Software Agents group[2] at the Media Lab run SVDs on data sets whose dimensions approach one million in just seconds, using a C library called SVDLIBC[3] running on desktop servers.

There are also alternative approaches which permit on-the-fly remapping behavior even while interacting with a large data set. One could engineer a large-scale recommendation system in such a way that PCA remapping could be cached and additions to the music library could be accommodated without a complete re-computation of the PCA. For instance, when building a recommendation system possessing a limited set of feature combinations, the system could

---

[1] `http://ojalgo.org/matrix_compare.html`
[2] `http://agents.media.mit.edu/`
[3] `http://tedlab.mit.edu/~dr/svdlibc/`

pre-compute and cache the PCA results for all of the (or, at least the most popular) feature set combinations. As new music is added to the large database between caches, these songs' locations can be estimated by computing each PC's linear combination using each song's feature list. This allows new songs to be mapped into the space to their approximate locations until the next full PCA can be run. A system like this could rebuild itself every night, running a PCA over its entire contents for several different feature combinations, and could still accept new songs during the day if necessary.

Interface issues when visualizing large data sets

There are many relevant interface concerns when designing for large data sets, only a few of which I will mention here. This is a whole area of research from which a large-scale version of MusicBox could benefit.

Some of the main features that would help in dealing with a large data set are:

- **Maintaining users' sense of context.** Users should be able to view a large data set at many different levels, in order to get a good overall picture of its structure, along with local detail in small subregions of the space. Presenting multiple views at the same time is a good approach; an example project that accomplishes this multi-level view particularly well is The Secret Lives of Numbers[4].

- **Displaying different levels of detail depending on what data is viewed.** At different zoom levels, the interface should display different amounts of data. For example, displaying artist names is not practical if the user is viewing 3000 songs at once. Summarization techniques, such as drawing clusters of individual points as a cohesive cloud, can also help.

- **Providing data filters.** Filters that limit the amount of data viewed can be immensely beneficial. In MusicBox, these filters could correspond to feature values, popularity, genres, artists, or could even randomly resample the space. All of these would unclutter the visualization and help the user focus on just what they are interested in.

- **Taking advantage of visual features.** Visual features such as hue, orientation, form, intensity, and motion of objects help users more easily discern patterns in data [61]. MusicBox currently uses color and motion to some extent, but other visual cues could certainly be incorporated.

- **Providing aids to navigate locally-crowded areas.** Even at highly magnified zoom levels, data objects can be so crowded that they obscure each other and are difficult to navigate. Tools like MusicBox's fisheye lens (page 85) can help.

---

[4] `http://www.turbulence.org/Works/nums/prepare.html`

7.2.3 Example applications

I hope to make a simplified version of MusicBox for the Apple iPod/iPhone. With a platform designed to enable such easy map navigation and music playback, the iPhone begs for an innovative spatial music player like MusicBox. Users' music libraries would be analyzed and stored on their personal computers, then synced to their iPod/iPhone. The MusicBox player on the iPod/iPhone would simply visualize the space and allow the user to move about inside it in order to find music of interest. Simple shuffle and filtering features would be included. To take even greater advantage of the connectedness of the iPhone, the view of a user's personal library could also be enhanced by setting it against a backdrop of the iTunes Music Store, showing relevant recommendations and encouraging discovery with instant reward.

MusicBox can also provide a way to navigate a *social* music space. A social network like MySpace, for example, could benefit greatly from having a geography by which to navigate its artists. Thus, MusicBox can act as a gateway to an ever-expanding social music experience.

7.2.4 Looking forward

MusicBox affords us a novel perspective on the world of music. As increasing amounts of music are recorded each day, and as more and more of it becomes accessible through ubiquitous networks, MusicBox's clean, sophisticated music map can help make the task of finding one's way in that huge space more palatable. In contrast to text-based interfaces whose lists only get longer and more daunting with increased library size, MusicBox gets smarter with more data. The more music it incorporates, the better context it can provide and the better recommendations it can make. It promises to change our relationship with music and to help us explore and discover far beyond what we have already found.

In an age in which the celestial jukebox is fast becoming a reality, we need a fresh approach to interact with its vast contents. MusicBox can act as our pilot, deftly guiding us through this complex space.

Appendix A

Principal Components Analysis

Principal components analysis (PCA) is a multivariate statistical technique whose primary application is to reduce the dimensionality of a data set. Therefore, it can be thought of as a summarization technique for data. After PCA has been applied, the data set can be read in terms of a smaller number of variables called principal components (PC).

I'll introduce PCA here with a simple example of dimensionality reduction. Then I'll show how PCA is mathematically computed, using singular vector decomposition (SVD). This Appendix aims to provide enough information to allow a novice to implement PCA in code.

A.1   A simple, but important, example

Our simple example will use a small data set of human weight and height measurements, shown in Table A.1. We will reduce this data set from two dimensions to one dimension.

Height (in.)	Weight (lb.)
62	125
63	128
64	145
67	147
68	160
70	189
71	200
73	202

Table A.1: Sample data set. Height and weight of 8 human subjects.

If we were to plot the sample data in two dimensions, we'd get a graph like the one in Figure A-1. The scatter plot shows each data point as a combination of two values, the one on the x-axis
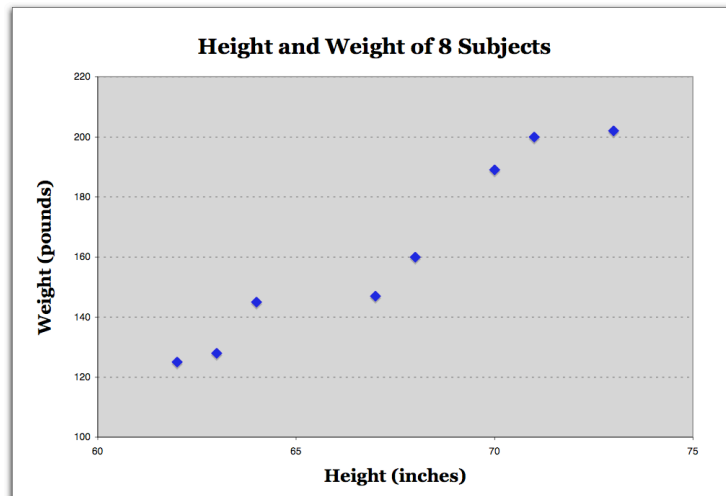
Figure A-1: A scatter plot of the sample data. This representation shows each data point as a combination of two values, the one on the x-axis (height), and the one on the y-axis (weight).
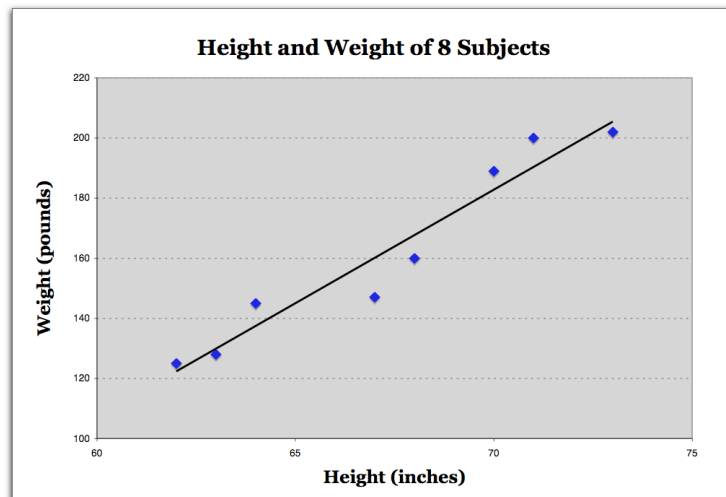
Figure A-2: Scatter plot with best-fit line. The best-fit line becomes our new axis.

(height), and the one on the y-axis (weight). This is a fine way to look at the data, but we're going to see how we can summarize it, so that each data point is represented with just one value instead of two.

Because height and weight appear to be related in some way (linearly, in fact) – weight increases as height increases – we could imagine *making a new axis* along a linear best-fit line through the existing data.

Figure A-2 shows the scatter plot with the best-fit line determined by linear regression. Now that we have our new axis, we can *project our original data* onto it by dropping a perpendicular line

114

Figure A-3: Scatter plot with data dropped to best-fit line. We are projecting the old data set onto the new axis.
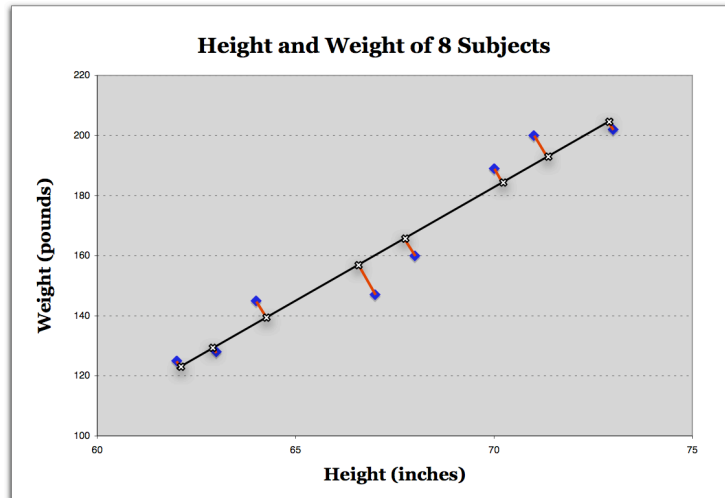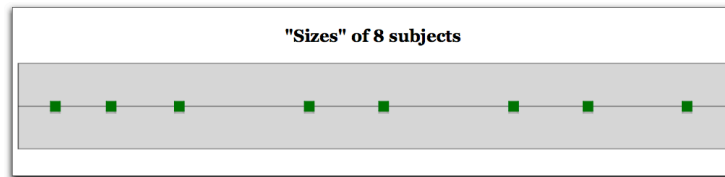

Figure A-4: New data projection. We are now viewing the original data set projected onto the lower dimensional space (in this case, just one axis). This new axis (or "PC") can be interpreted as a "size" axis.

from each point to the new axis (Figure A-3). Once we've projected the data, we can look at it along just the one new axis, as pictured in Figure A-4. If we were to try to interpret what this new axis represents, we might called it "size". Along this new "size" axis, we see smaller people on the left, and larger people on the right.

In the process of projecting the original data onto this smaller set of axes (in this case, onto just one axis), we are *losing some data*; that is, we are losing the distance that each of those points lay from the best-fit line. This is the sacrifice we make in moving to a lower number of dimensions. For a data set that does not have a linear relationship amongst the variables (that is, a situation in which a best-fit line doesn't even make sense), using PCA to reduce dimensionality can result in a loss of data at an unacceptable level. If this is the case, another dimensionality reduction method should be chosen (some of which are mentioned in Section 5.3.1).

In order to *not* lose that extra data, we could compute a second axis – a line perpendicular to the first one – along which we could measure these distances. However, doing that would bring our data set back to two dimensions, and therefore we have not achieved anything in reducing the dimensionality of the original data set.

PCA outputs an ordered set of principal components (PC), the number of which equals the dimensionality of the original data set. Each of these PCs is an axis for the new space. In our simple example, the original data set has two dimensions, so the PCA outputs two PCs, the first of which corresponds to the "size" axis, representing most of the variance in the data; the second axis is the one we dropped.

In a more realistic example of PCA, our original data set would have a much higher number of dimensions, M, from the tens to hundreds. A PCA gives that same number of PCs (M) out; from those, we choose the first L that we would like to retain. Choosing the value of L is slightly subjective, and usually involves using a scree plot to find the point where adding further dimensions yields a proportionately smaller gain in information (see [59, page 41] for an example). The value of L also depends on how the new, projected data set will be interpreted: if the data is to be visualized, selecting more than three dimensions makes little sense. In this thesis project, we always use just the first two PCs, because MusicBox is designed to visualize a 2-dimensional data space.

In summary, PCA finds an ordered set of new axes through the data space in such a way that each new axis holds information that is a combination of the original axes (with the first few hopefully holding much *more* information than the same number of original axes). In PCA, these axes are always linear combinations of the underlying axes, and are computed to be independent of one another (this is like being perpendicular in the higher-dimensional space).

In slightly more mathematical terms, PCA starts with finding a basis to span the original space. The user selects a subset of that basis which defines the perspective through which they will view the reduced data set. Once the basis has been computed and the first L PCs selected, PCA projects the old data onto the new PCs. For each new axis, one can also calculate the amount of variation in the original data set that it accounts for. This measurement helps give a sense of how much of the variance is retained from the original data set when selecting fewer axes (i.e. how much information is lost in moving to the lower dimensional space).

A.2   Computing PCA

Now that we have a better idea of what PCA does through a simple example, I'll present the math necessary to make these computations and projections.

One way to compute PCA is by factorizing the input matrix into its singular value decomposition (SVD).

SVD is defined as follows:

For an m-by-n matrix $\mathbf{X}$, there exists a factorization of the form

$$\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}} \tag{A.1}$$

where $\mathbf{U}$ is an m-by-n matrix, $\mathbf{S}$ is a diagonal n-by-n matrix, conventionally ordered high-to-low from upper-left to lower-right, and $\mathbf{V}^{\mathrm{T}}$ is also an n-by-n matrix [62]. The diagonal values in $\mathbf{S}$ are called "singular values", which are the square root of the eigenvalues of $(\mathbf{X}\mathbf{X}^{\mathrm{T}})$.

This factorization is akin to the simple example in the last section where we determined best-fit lines for the original data; here those best-fit lines are the basis vectors contained as columns of $\mathbf{V}^{\mathrm{T}}$. The columns of $\mathbf{V}^{\mathrm{T}}$ are the ordered principal components of the original data set.

There are many computational methods used to decompose a matrix into its SVD; I won't go into these methods here, but both [59] and [62] discuss the topic.

Once we have our SVD, we reduce the number of dimensions by reducing the number of columns of $\mathbf{V}^{\mathrm{T}}$. If we choose to use the first L principal components, we remove all but the first L columns of $\mathbf{V}^{\mathrm{T}}$. We'll call that reduced L-by-n matrix $\mathbf{W}$.

Now that we have our PCs chosen, we project the original data into the new subspace by computing

$$\mathbf{Y} = (\mathbf{W}\mathbf{X}^{\mathrm{T}})^{\mathrm{T}} \tag{A.2}$$

In equation (A.2), since $\mathbf{W}$ is an L-by-n matrix and $\mathbf{X}^{\mathrm{T}}$ is an n-by-m matrix, the transpose gives a new m-by-L matrix $\mathbf{Y}$ which defines the m rows of $\mathbf{X}$ in the newly-reduced space.

For a more technical, deep, but clear description of PCA, try [59]. [62] contains a nice example of PCA applied to a biological data set.

A.3   Implementation of PCA in MusicBox

To perform PCA, MusicBox computes the SVD of its input data set using a Java matrix package called JAMA[1].

MusicBox's input data set is an m-by-n matrix with one row per musical track, and one column per descriptive feature (giving m tracks and n features). (MusicBox test libraries usually contained between 500 and 1500 tracks, and between 150 and 250 features.) Before the SVD is computed, each value in the matrix is mean-centered by subtracting the mean of all the track values for the corresponding feature, then normalized by dividing by the standard deviation of all the track values for the corresponding feature.

---

[1] `http://math.nist.gov/javanumerics/jama/`

Decomposing the mean-centered, normalized input matrix into its SVD gives $\mathbf{V}^{\text{T}}$. MusicBox selects just the first two columns of $\mathbf{V}^{\text{T}}$, multiplying the result (still using JAMA) by the transpose of the input matrix $\mathbf{X}$. Just as we saw in equation (A.2), the transpose of the resulting matrix is $\mathbf{Y}$.

In MusicBox, $\mathbf{Y}$ is an m-by-2 matrix, so now we have each of the m musical tracks in two dimensions. MusicBox plots each musical track in a scatter plot using these two dimensions.

Appendix B

Calculating tf-idf scores

tf-idf is used in information retrieval to measure the importance of a word in distinguishing one document from others in a set of documents. "tf-idf" is an abbreviation for "term frequency times inverse document frequency" [63]. Therefore it directly measures term importance as the frequency of a term in a document divided by the proportion of documents that contain that term.

Variations of *tf-idf* are commonly used in search engine ranking schemes in order to determine a document's relevance given a user's query. That is, if a user enters a term, the search engine would want to return documents with high *tf-idf* weights for that term. The documents with high *tf-idf* weights would tend to have a higher frequency of the given term than other documents, and these weights are higher still if the term is less frequent in the entire document set.

Calculating *tf-idf* is straightforward. The term frequency (*tf*) is the number of times the term appears in the document, divided by the number of total words in the document. The document frequency (*df*) is the number of documents that contain the term, divided by the total number of documents. The final *tf-idf* weight is calculated by dividing *tf* by *df*.

Here's a simple example. Suppose you want to calculate the *tf-idf* weight for "chocolate" for a document in a collection. Say the document has 1000 words, 5 of which are "chocolate". This gives us a *tf* of 5/1000 = 0.005. Now we look at the collection of documents to determine the *df*. Say the collection contains 10,000 documents, 7 of which contain the word "chocolate". This gives a *df* of 7/10000 = 0.0007. Our *tf-idf* weight is simply *tf* divided by *df*: 0.005/0.0007 = 7.14.

MusicBox uses *tf-idf* scores to determine which user-created tags most clearly distinguish an artist from others in the MusicBox library. "Term frequency" here refers to tag frequency for an artist, and "document frequency" refers to the proportion of artists who have ever been given

that tag. Using *tf-idf*, MusicBox is able to determine, for example, that "motown" is a more effective descriptor of Diana Ross than "rock".

Bibliography

[1] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, 1987.

[2] Richard Saul Wurman. *Information Anxiety*. Doubleday Publishing, 1989.

[3] J. Stephen Downie. *Music information retrieval*, chapter 7, pages 295–340. Information Today Books, 2003.

[4] Ben Schneiderman. Dynamic queries for visual information seeking. *IEEE Software*, 11(6):70–77, 1994.

[5] Daniel J. Levitin. *This Is Your Brain on Music: The Science of a Human Obsession*. Dutton Adult, 2006.

[6] The Nielsen Company. Nielsen SoundScan: State of the Industry 2007-2008. `http://www.narm.com/2008Conv/StateoftheIndustry.pdf`, 2008.

[7] C. Anderson. *The Long Tail*. Random House Business, 2006.

[8] Joe Futrelle and J. Stephen Downie. Interdisciplinary communities and research issues in music information retrieval. *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, 2002.

[9] J. Stephen Downie. Toward the scientific evaluation of music information retrieval systems. *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, 2003.

[10] David Jennings. *Net, Blogs and Rock 'n' Roll: How Digital Discovery Works and What it Means for Consumers*. Nicholas Brealey Publishing, 2007.

[11] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

[12] Peter Kirn. Pandora's founder on decoding taste and promoting indie music. `http://createdigitalmusic.com/2007/03/16/pandoras-founder-on-decoding-taste-and-promoting-indie-music/`, March 2007.

[13] Audrey Laplante and J. Stephen Downie. Everyday life music information-seeking behaviour of young adults. *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, pages 381–382, 2006.

[14] J. Foote. An overview of audio information retrieval. *Multimedia Systems 7, Springer-Verlag*, pages 2–10, 1999.

[15] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H.G. Okuno. An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):435–447, February 2008.

[16] Brian Whitman. *Learning the Meaning of Music*. PhD thesis, Massachusetts Institute of Technology, MA, USA, June 2005.

[17] Tristan Jehan. *Creating music by listening*. PhD thesis, Massachusetts Institute of Technology, MA, USA, 2005.

[18] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. In *Neural Information Processing Systems Conference (NIPS)*, 2007.

[19] Cory McKay and Ichiro Fujinaga. Musical genre classification: Is it worth pursuing and how can it be improved? *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, pages 101–106, 2006.

[20] Kris West, Stephen Cox, and Paul Lamere. Incorporating machine-learning into music similarity estimation. In *AMCMM '06: Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 89–96, New York, NY, USA, 2006. ACM.

[21] Stefaan Lippens, Jean-Pierre Martens, Tom De Mulder, and George Tzanetakis. A comparison of human and automatic musical genre classification. In *Proceedings of the 2004 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2004)*, pages 233–236, Montreal, May 2004. IEEE.

[22] Thomas Lidy and Andreas Rauber. Evaluation of feature extractors and psycho-acoustic transformations for music genre classification. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, pages 34–41, London, UK, September 11-15, 2005.

[23] David Wessel. Timbre space as a musical control structure. *Computer Music Journal*, 3:45–52, 1979.

[24] G. Tzanetakis and P. Cook. MARSYAS: A framework for audio analysis. In *Organised Sound*. Cambridge University Press, 2000.

[25] *The Columbia Encyclopedia*. Columbia University Press, Sixth Edition.

[26] Max Mathews. What is loudness? In *Music, cognition, and computerized sound: an introduction to psychoacoustics*, pages 71–78. MIT Press, Cambridge, MA, USA, 1999.

[27] David S. Lefkowitz and Kristin Taavola. Segmentation in music: Generalizing a piece-sensitive approach. *Journal of Music Theory*, 44(1):171–229, 2000.

[28] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Semantic annotation and retrieval of music and sound effects. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):467–476, February 2008.

[29] F. Pachet. Metadata for music and sounds: The Cuidado Project. In *Proceedings of the CBMI Workshop*, University of Brescia, September 2001.

[30] Hugues Vinet, Perfecto Herrera, and Francois Pachet. The Cuidado Project. In *Proceedings of the 3rd International Symposium on Music Information Retrieval*. IRCAM, October 2002.

[31] TuneGlue – Relationship Explorer. `http://audiomap.tuneglue.net/`, June 2008.

[32] Musicovery : interactive webradio. `http://musicovery.com/`, June 2008.

[33] Eshkol Rafaeli, Gregory M. Rogers, and William Revelle. Affective Synchrony: Individual Differences in Mixed Emotions. *Pers Soc Psychol Bull*, 33(7):915–932, 2007.

[34] R. E. Thayer. *The Biopsychology of Mood and Arousal*. Oxford University Press, 1989.

[35] Owen Meyers. A mood-based music classification and exploration system. Master's thesis, Massachusetts Institute of Technology, MA, USA, June 2007.

[36] BMAT - Barcelona Music & Audio Technologies. `http://bmat.com/discover`, June 2008.

[37] Elias Pampalk. Islands of Music: Analysis, organization, and visualization of music archives. Master's thesis, Vienna University of Technology, December 2001.

[38] E. Pampalk, A. Rauber, and D. Merkl. Content-based organization and visualization of music archives. In *Proceedings of the ACM Multimedia*, pages 570–579, Juan les Pins, France, December 1-6, 2002. ACM.

[39] Robert Neumayer, Michael Dittenbach, and Andreas Rauber. PlaySOM and PocketSOMPlayer: Alternative interfaces to large music collections. In *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, pages 618–623, London, UK, September 11-15, 2005.

[40] Peter Knees, Markus Schedl, Tim Pohle, and Gerhard Widmer. An Innovative Three-Dimensional User Interface for Exploring Music Collections Enriched with Meta-Information from the Web. In *Proceedings of the ACM Multimedia 2006 (MM'06)*, Santa Barbara, California, USA, October 23-26, 2006.

[41] Stefan Leitich and Martin Topf. Globe of Music - Music library visualization using GeoSOM. In *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, Vienna, Austria, September 2007. Österreichische Computer Gesellschaft (OCG).

[42] Justin Donaldson. A small movie of "music mapping". `http://labs.strands.com/interns/SonyCatalog.mov`.

[43] Masataka Goto and Takayuki Goto. Musicream: New music playback interface for streaming, sticking, sorting, and recalling musical pieces. *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR 2005)*, pages 404–411, 2005.

[44] Elias Pampalk and Masataka Goto. MusicRainbow: A new user interface to discover artists using audio-based similarity and web-based labeling. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, 2006.

[45] Elias Pampalk and Masataka Goto. MusicRainbow: A new user interface to discover artists using audio-based similarity and web-based labeling (poster). `http://staff.aist.go.jp/m.goto/MusicRainbow/pam_ismir06_musicRainbow_poster.pdf`, Presented at the ISMIR International Conference on Music Information Retrieval, October 8-12, 2006, Victoria, Canada.

[46] Elias Pampalk and Masataka Goto. MusicSun: A new approach to artist recommendation. *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[47] P. Lamere and D. Eck. Using 3D visualizations to explore and discover music. *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[48] Fabio Vignoli, Rob van Gulik, and Huub van de Wetering. Mapping music in the palm of your hand, explore and discover your collection. *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, 2004.

[49] Lisa Dalhuijsen and Lieven van Velthoven. MusicalNodes: The visual music library. `http://waks.nl/anderemedia/MusicalNodes_paper.pdf`, June 2008.

[50] Marc Torrens, Patrick Hertzog, and Josep Lluís Arcos. Visualizing and exploring personal music libraries. *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004)*, 2004.

[51] pygame - python game development. `http://pygame.org/`, May 2007.

[52] Craig Sapp. Tonal landscape gallery. `http://ccrma.stanford.edu/~craig/keyscape/`, May 2008.

[53] Elaine Chew and Alexandre R. J. Francois. Interactive multi-scale visualizations of tonal evolution in MuSA.RT Opus 2. *Computers in Entertainment (CIE)*, 3(4):1–16, 2005.

[54] Dmitri Tymoczko. The Geometry of Musical Chords. *Science*, 313(5783):72–74, 2006.

[55] The Echo Nest. XML Tag Reference. `http://developer.echonest.com/docs/analyze/xml`, July 2008.

[56] The Echo Nest. The Echo Nest Analyze API. `http://the.echonest.com/analyze/`, July 2008.

[57] Nicolas Scaringella, Giorgio Zoia, and Daniel Mlynek. Automatic genre classification of music content: A survey. *Signal Processing Magazine, IEEE*, March 2006.

[58] Joseph B. Kruskal and Myron Wish. *Multidimensional Scaling*. Number 07-011 in Sage University Paper series on Quantitative Applications in the Social Sciences. Sage Publications, 1978.

[59] George H. Dunteman. *Principal Components Analysis*. Number 07-069 in Sage University Paper series on Quantitative Applications in the Social Sciences. Sage Publications, 1989.

[60] Y. Li, J. Xu, L. Morphett, and R. Jacobs. An integrated algorithm of incremental and robust PCA. *Proceedings of IEEE International Conference on Image Processing*, 2003.

[61] C. Healey. *Effective Visualization of Large Multidimensional Datasets*. PhD thesis, The University of British Columbia, Vancouver, B.C., 1996.

[62] Michael E. Wall, Andreas Rechtsteiner, and Luis M. Rocha. Singular value decomposition and principal component analysis. In *A Practical Approach to Microarray Data Analysis*, chapter 5. Kluwer Academic Publishers, March 2003.

[63] Chris Buckley. The importance of proper weighting methods. In *HLT '93: Proceedings of the workshop on Human Language Technology*, pages 349–352, Morristown, NJ, USA, 1993. Association for Computational Linguistics.