

Brigham Young University
BYU ScholarsArchive

Theses and Dissertations

2012-07-06

# Methods of Music Classification and Transcription

Jonathan Peter Baker
*Brigham Young University - Provo*

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

 Part of the Mathematics Commons

BYU ScholarsArchive Citation
Baker, Jonathan Peter, "Methods of Music Classification and Transcription" (2012). *Theses and Dissertations*. 3330.
https://scholarsarchive.byu.edu/etd/3330

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

# Methods of Music Classification and Transcription

Jonathan Baker

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Jeffrey Humpherys, Chair
Wayne Barrett
Sum Chow

Department of Mathematics

Brigham Young University

August 2012

Copyright © 2012 Jonathan Baker

All Rights Reserved

ABSTRACT

Methods of Music Classification and Transcription

Jonathan Baker
Department of Mathematics, BYU
Master of Science

We begin with an overview of some signal processing terms and topics relevant to music analysis including facts about human sound perception. We then discuss common objectives of music analysis and existing methods for accomplishing them. We conclude with an introduction to a new method of automatically transcribing a piece of music from a digital audio signal.

Keywords: audio processing, sound perception, automated music transcription

# Contents

**Contents** **iii**

**1 Introduction** **1**

**2 Perception of Sound** **2**

2.1 Chromatic Scale . . . . . 3

2.2 Note Identification . . . . . 4

**3 Tasks in Music Analysis** **4**

3.1 Similarity . . . . . 5

3.2 Classification . . . . . 6

3.3 Transcription . . . . . 8

3.4 Summarization . . . . . 11

**4 Signal Processing** **12**

4.1 Frames and Windows . . . . . 12

4.2 Binning . . . . . 16

4.3 Alternatives to FFT . . . . . 19

**5 Obtaining Music Features** **23**

5.1 Segmentation . . . . . 23

5.2 Cepstrum and Mel-frequency Cepstrum . . . . . 25

5.3 Other Spectral Features . . . . . 28

**6 Optimization-based Transcription** **29**

6.1 Algorithm Setup . . . . . 30

6.2 Training Data . . . . . 33

6.3 Test Data . . . . . 34

**7 Conclusion** **37**

**Bibliography** **39**

# Chapter 1. Introduction

Digital media of all kinds has become readily available all over the world. Personal digital music players are a common sight in many countries and the capacity of these devices is already enormous and still growing. Many consumers enjoy instant access to thousands of songs totaling months of playtime.

Providing access to all this material is a profitable industry so many companies are interested in providing customers with easy ways to search for music they are interested in buying. Additionally, some systems attempt to provide personalized music suggestions based on what songs a consumer has already purchased. Since this can result in additional sales, these recommendation systems are also valuable.

If a user is interested in a specific song, he or she can usually provide "objective metadata" (the song's title, artist, album). Based on these attributes, it is simple to locate the song in a well-designed database. However, a user might want to browse the available songs by specifying "abstract metadata" such as genre, style, tempo or mood. This information is not essential to keeping the song in the database so some systems may not track it. Also, such attributes are subjective. A song may be partially influenced by several musical genres and not belong firmly to any particular one. Such ambiguities need to be resolved before the information can be usefully stored in the database.

The even more complex task of suggesting pieces similar to those already purchased requires significant insight into the nature of the music being offered. Sufficiently organizing the huge libraries controlled by large music providers is an imposing task. According to [24], iTunes has more than 28,000,000 songs totaling 266 years of playtime. Analyzing so many songs will require a great deal of processing and every improvement in the time required to analyze a single song is valuable.

An industrial music analysis algorithm may draw on many sources for its information. Some metadata (such as the names of the song, artist and album) must be provided for the sake of unique, human readable identification of each song. Abstract metadata can be:

- Entered manually by music experts [32]
- Inferred from patterns in data entered manually by consumers [13]
- Derived from attributes of the musical score (a discrete representation of a composition encoding enough information for reproducing a performance) [29]
- Derived from attributes of of the audio content itself

Any of these methods of collecting information could be applied to any of the analytic tasks discussed in Chapter 3. Consumers are perhaps the most difficult source of information to use since they are the least directly under the control of the system. Consumers often apply labels (especially genre) to songs in their libraries in order to make finding purchased content easier, but consumers cannot be expected to provide all the labels the system owners would like to track.

These systems employ different approaches to the item similarity problem. Pandora[1] is engaged in the music genome project in which experts identify attributes of each piece (genre, instrument combinations, time period, etc.) which are then compared across pieces to compute similarity. Amazon[2] uses collaborative filtering: predicting needs through established purchasing patterns ("Customers Who Bought This Item Also Bought..."). Query-by-humming systems (QBH) allow users to search for song by recording an imitation (e.g. by humming). Such systems necessarily depend on the audio content as every query consists only of a music sample.

In this paper, we will focus on explaining attributes of audio content that are useful for measuring song similarity and classification. We will also discuss how these same attributes may be used to automatically transcribe the song being processed.

[1] http://www.pandora.com
[2] http://www.amazon.com

# Chapter 2. Perception of Sound

The intensity and frequency of sounds largely describe our perception of them. It is as though our ears were performing Fourier analysis of the pressure on the ear drum. Especially for sounds we consider musical, the pitch and loudness listeners report are correlated with the the magnitude of the dominant frequencies found by microphones and Fourier analysis.

The intensity of sensations tends to be logarithmically proportional to the measured intensity of the stimulus. Using an integer scale, test subjects report the intensity of odors increases as the log of the concentration of odorous particles [15]. Earthquakes and hurricanes are rated on scales proportional to the log of the disastrous energy output. It should be no surprise that musical pitches are measured as the log of the dominant frequency and that exponentially increasing amplitude is perceived as linearly increasing intensity.

## 2.1 Chromatic Scale

Most western music uses tones from a 12-tone equal-temperament scale. Specifically, each perceived tone will have a frequency from

$$piano_i = 440 \times 2^{(i-49)/12} \tag{2.1}$$

For $i = 49$, this is 440 Hz or A above middle C: the 49th key of a piano. This is often written as A4 as it is the A in the fourth octave of the piano. The numbers in this notation increase on each C meaning middle C is C4 and the note below it is B3. The lowest C on a piano is C1, but the lowest key is A0 with frequency $piano_1 = 27.5$ Hz.

The frequencies represented in a Fourier transform of a sample include frequencies not included in this set. Furthermore, the tone that the human ear perceives is not always the tone with the highest energy. Many frequencies many have significant energy and the combination of these frequencies give each instrument its unique sound. The attributes of this combination aside from the overall loudness and perceived pitch are referred to collectively

as the timbre.

## 2.2 Note Identification

Even the seemingly simple task of discovering what pitches would be perceived by a human at a particular time in a piece is not straightforward. One might think that the perceived frequency would be the frequency with the greatest energy in the signal, that is $r\hat{i}/n$ where $\hat{i} = \text{argmax}_i |\hat{f}_i|$, $r$ is the rate at which samples are taken and $n$ is the length of the signal. It is true that in nearly sinusoidal signals the frequency with the greatest energy dictates the perceived pitch, but in more complex signals, the relationships between many frequencies may determine the pitch. For example, Figure 2.1 shows the energy spectrum of the note A1 played on a Yamaha grand piano. The frequency associated with this note is 55 Hz, but very little energy around 55 Hz is actually present. It is not obvious why the human ear considers this sound so similar to a sinusoid oscillating at 55 Hz.

The authors of [27] explain early misconceptions about this phenomenon. The missing frequency is not created by distortion patterns caused by physics of the ear nor by the frequencies of extreme peaks and troughs in the waveform. It now seems that the brain is trained to associate certain spectral features with pitches. Physical sound-producing systems (like strings and air-chambers in musical instruments) have fundamental frequencies at which they tend to allow vibrations. Vibrations at integer fractions of the fundamental frequency (called harmonics or overtones) are often present as well. The brain, being accustomed to hearing fundamental pitches along with their harmonics, does not even need the fundamental frequency to be physically present. If a sound has the harmonics spaced at a particular frequency, that frequency is often perceived even when little or no energy is present at that frequency as in the piano spectrum in Figure 2.1.

Finding the equally spaced peaks in the energy spectrum can be accomplished through spectral analysis **of the spectrum** which is discussed further in Section 5.2.

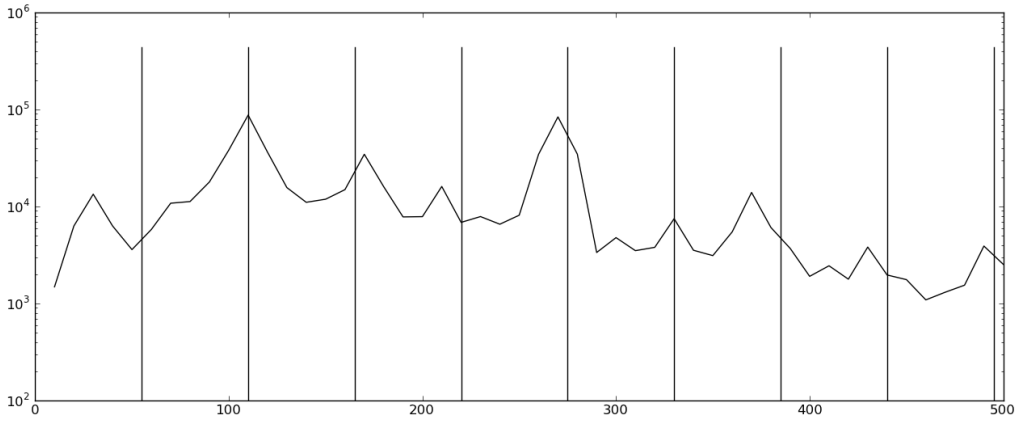



Figure 2.1: Energy spectrum of A1 played by a piano. The fundamental frequency (55 Hz) has little energy, but its harmonics (110, 165, 220, ...) are near prominent peaks. The brain recognizes this spacing and provides the illusion of a 55 Hz signal.

# Chapter 3. Tasks in Music Analysis

In this chapter we discuss important goals for understanding music and the relationships between songs. A system might accomplish one or more of these tasks only as a means of reaching another. For example, one might *transcribe* a song in order to calculate the *similarity* between the song as a whole and intervals within the song in order to find the most appropriate section to *summarize* the song.

## 3.1 Similarity

The degree of similarity between items is subjective and it can be difficult to design an algorithm to mimic the human ability to compare objects. Many systems (including Amazon and Pandora) rely on human input to find similar items, but human time is expensive and many datasets are large enough that designing a fully independent system – that calculates similarities based on automatically-found object attributes – may be more economical.

The numerous ways of describing a song provide many ways to measure their similarity. A pair of songs may belong to similar genres but have different tempos and harmony

complexities. It may be desirable to convert features into a single-dimensional "distance" between pairs of songs. One way of describing the similarity of songs is to measure many of each song's features and then compare the lists of features. These features could be measured from a musical score (possibly generated by automated transcription), directly from audio data or entered manually by experts or users. Most methods of measuring similarity use features calculated from the audio data itself. A list of common features calculated from audio data appears in Chapter 5.

When computing features of a song, it is common to first calculate audio features of short segments of the song. The features of all segments together describe the song. Distances between songs can then be calculated in many ways from their sequences of feature vectors. For example, [3] describes each song with estimates of the means and covariances of the features across all segments. Other methods consider the order of the segments such as [33] and [17] which use Markov models to describes distributions of segment feature vectors conditioned on the previous vector. These distributions form the description of the song used for similarity calculations.

Once the distance between each pair of songs has been calculated, many metric-based techniques are available for identifying related clusters. If the genres of some songs in the dataset are known, each cluster could be named for the majority of known genre labels of the songs it contains.

On the other hand, all genres could be known before similarities are calculated (either by expert labeling or by majority labels of users). In this case, the genre could be one of the features used to calculated the similarity between songs. So, calculated similarities might be used to predict genres or known genres may be used to help measure song similarity.

## 3.2 Classification

In some applications, classifying a piece of music – especially by genre – may be an end goal since it allows organization of music libraries and may be helpful in searching for specific

music and discovering similar songs to purchase. Classification may also be used as an intermediate objective in measuring the similarity between pieces such as being an input to a single-dimension metric (e.g. belonging to the same genre could change the similarity score of two songs by a fixed additive or multiplicative amount).

Genre classification is a difficult task. There are many nuanced sub-genres and even expert musicians will not always agree on how to classify some pieces. Genre identifying algorithms are challenging in part because of this ill-definedness.

While there are dozens of partially overlapping musical genres, only a few basic emotions typically appear in music (anger, joy, sorrow, calm, ...). Emotions tend to be conveyed by features such as volume and tempo which are more objective than culturally defined genres. Hence we might expect it to be easier to assign emotions to a piece of music than a genre. The authors of [28] employed experts to classify songs by six basic emotional labels. They then calculated rhythmic and timbral features of each song. They used some of the songs to train several algorithms which then predicted the moods of the other songs based on their similarity to the songs in the training set. The best algorithm agreed with the experts on 81% of the test songs.

Mood can be used in ways similar to the genre: it may be entered manually (and used as part of a similarity measure) or it may be assigned automatically based on similarity to songs with known moods (and subsequently used to aid searches).

Both genre and mood can be identified with varying reliability by human listeners. Some databases [32] have this information assigned manually by experts. This becomes a static part of the database. However, music collections can be quite large and having experts to carefully examine each piece may be extremely expensive.

Other systems such as Grooveshark[1] rely on users to provide genre information. The system's interface may prompt users to "tag" their songs and subsequently use the most common tag as the genre assumed during searches. This removes the need to pay for expert labelers, but checking user feedback to learn the current classification of each piece will

[1] http://grooveshark.com/

require more computation than establishing a genre at the outset. It also requires that the system designers provide a tagging interface and trust their users to be reliable taggers. Such a system will not be able to list a genre for a piece until at least one user has tagged it which may limit its appearance in search results and delay its first tag.

Systems such as Midomi[2] and the related mobile app SoundHound[3] attempt to find the piece a user is thinking of by comparing songs in the database to a hummed query.

## 3.3 Transcription

Many of the most successful machine learning systems that deal with complicated input first interpret the input in an intermediate format. For example, speech recognition algorithms such as [22] typically often translates raw audio data into discrete sounds (phonemes) which may be compiled into words and finally phrases, each layer of this process being context-sensitive. The comparable choice of representations in music is a musical score in some appropriate electronic format. Such a representation is highly discrete and eases comparison to other songs relative to comparing audio content of songs.

Producing complete transcriptions manually is extremely time consuming – easily requiring many hours of intense concentration to identify each note in a rich harmony. Transcribers typically must listen to portions of a complex song many times, sometimes at a reduced speed in order to clearly hear every note played. Before digital recordings, transcribers could not slow the playback of the music without altering the pitch. Now, special software is designed to play music at reduced speeds (without altering pitch) while simultaneous providing a visual representation of the audio content such as the waveform or energy spectrum.

**3.3.1 Rhythm and Pitch.** The most important features recorded in standard music notation are the pitch of each note and when it occurs – especially when it begins, but also its length. The volume and duration of the note as well as the contributing instrument are

[2]http://www.midomi.com/
[3]http://www.soundhound.com/

typically less important as the melody can be clearly understood even when these features are manipulated. Hence, information on the time and pitch of notes in an audio signal encodes the most important information about the piece as understood by a human and should be useful in simulating the human understanding of musical similarity.

One unique approach to transcription can be found in [25]. The authors begin by finding the energy spectrum of many equal-width frames. These spectral vectors are stored as the columns of a matrix $V$. They apply nonnegative matrix factorization (NMF) – developed in [18] – to obtain an approximate factorization $V \approx WH$ where $W, H$ are nonnegative. In several examples, the columns of $W$ are energy spectra of individual notes and the rows of $H$ are the intensities with which those notes appear across the frames of the sample. To complete the transcription, one would only need to identify the notes represented by the columns of $W$ and determine the temporal positions of specific note events in the rows of $H$.

The above technique is unusual because it does not rely on any features of the original signal beyond the energy spectrum. More standard methods of finding the beginnings of notes are discussed in Section 5.1 and principals of determining pitch in Sections 2.2 and 5.2.

**3.3.2 Instrument Identification.** Recognizing the instruments in a song is less important than identifying notes' pitches, but the choice of instruments is important to a song's identity. In the music genome project [32], the featured instruments of a song are among the attributes recorded. For each song, the project records which of over 200 instrument play-styles are used (such as "electric bass riffs" and "subtle use of the harmonica"). Automatically identifying the instruments in a song has been a difficult challenge that most commercial automatic transcription programs do not handle well.

WIDI[4] is a lightweight product (3.8 MB) that is able to convert audio files to MIDI format (which encodes rhythm and pitch information). Its accuracy is heavily dependent on the user's ability to provide parameters appropriate to the music being analyzed. In particular, the software takes as a parameter the instrument to anticipate in the audio file (or the user

[4]http://www.widisoft.com/

can specify that a rich harmony will be used). This allows the program to attempt the use of appropriate harmonic filters, but it does not attempt to separate the contributions of different instruments.

Celemony[5] is renowned as an industry leader in musical audio processing with the power to retune notes in an audio file even in the presence of other notes. This tool is referred to as Direct Note Access (DNA). However, even this powerful tool carries the disclaimer "Please note that DNA is designed for tracks containing a single polyphonic instrument (a guitar, a piano, ...) and that it divides the material up according to pitch – not instrument." Celemony makes no attempt to distinguish between two different instruments playing the same note. This is often not a problem for artists since the work of individual instruments is often recorded separately. However, a transcription algorithm that takes only a single audio track as input does not enjoy this luxury.

When identifying the instrument that produced a note, one may attempt a match by using prior knowledge of musical features typical of each instrument. For example, [4] uses the MPEG-7 audio descriptors described in [19] to predict which instrument produced a piece of music based on a set of examples of each instrument's sound. They first calculate a descriptor vector for each frame of each training sample. The mean and variance of each MPEG 7 descriptor across the frames of one sample form a feature vector $v_j$. The feature vectors of all samples are then arranged into the matrix $V$.

They use NMF to approximately decomposing $V$ into two relatively low rank nonnegative matrices: a basis $W$ and an encoding $H$ so that $V \approx WH$. The columns of $W$ are an approximate "basis" for the set of feature vectors observed. The nonnegative decomposition $WH$ is not unique and the results of a variety of decomposition methods are compared.

Then, for each test piece, an encoding vector $h$ may be estimated from the piece's feature vector $v$ by, for example, least squares: $h = W^{\dagger}v$. They then predict that the instrument producing the feature vector $v$ is the one that produced the training encoding vector most similar to $h$ as judged by the maximizing the cosine of the angle between them.

[5] http://www.celemony.com/cms/

This process achieved 85% to 95% instrument classification accuracy depending on the matrix factorization method used and the subset of audio features used (fewer features sped computation with little harm to accuracy).

This technique alone may be enough to classify the work of a soloist, but more work will be needed to deal with polyphonic pieces. Depending on the separation between notes, it is possible that a note identification method could be applied first and then this algorithm could be used to predict the instrument being used for each note.

## 3.4 Summarization

Another task that might arise in an application is to find the portion of a song that is most like the rest of the song. For example, iTunes allows customers to listen to clips from some songs before they are purchased. Choosing the preview clip in an unsophisticated way may result in previews that are not representative of the song as a whole. Hearing a haphazard preview may cause customers to fail to recognize the specific song they were searching for or to misunderstand the song's style.

Songs often contains repeated musical phrases. The more a phrase is used, the greater the proportion of the song is spent repeating it. The most-used phrases are good choices for previews since much of the song will be exactly the sound contained in these phrases. Although humans can identify these as portions easily, it would still be expensive to employ sample-choosers to process an industrial music library. As in the classification problem, it could be possible to allow customers to mark suggested previews and use the most-marked 30 second interval. However, while users often voluntarily sort their music libraries by genre (thereby supplying information to the system), there does not appear to be a motivation – except perhaps volunteerism – for users who already own the song to search for the most representative clip. Hence socially distributing this task seems less reasonable than distributing genre identification.

As an alternative, one might attempt to automatically select a preview. For example,

[33] uses a measure of similarity between audio content features to assign classifications to each short interval of a song. Adjacent intervals assigned to the same class are treated as a coherent phrase that might be used as a preview. Any measure of song similarity (including those relying on musical scores) could theoretically be used for this purpose.

# Chapter 4. Signal Processing

## 4.1 Frames and Windows

Understanding a piece of music involves both local and global analysis. Local analysis examines short intervals of the waveform. This simulates the way human memory combines instantaneous sensations into meaningful experiences: the pressures on the eardrum over the course of about 20 milliseconds (ms) allow us to perceive the frequencies in the pressures. Global analysis involves amalgamating local information into larger structures such as melodies made by several notes played one after the other.

For local analysis, it is common to break the waveform signal into short (50–200 ms) intervals called frames. If frames are chosen to be too long, it becomes more likely that the frequencies found in the beginning and end of the frame will be different, that is, the frame contains parts of two different notes. However, frames must not be chosen to be so short that the lowest-frequency components of the signal do not have a full cycle within a single frame.

Suppose the original signal is $x$ (a vector of length $N$) and the frame $f$ (length $n$) is given by $f_i = x_i$ for $a \leq i \leq b$. To say that the Fourier transform of $f$ represents sinusoids of various frequencies is to assume that the signal repeats indefinitely. Most frames will not end near where they began, that is, repeating $f$ causes discontinuities at the edges as on the left of Figure 4.1.

The spectra of signals with discontinuities have many nonzero components, but these do not all correspond to frequencies that appear to be present. The spectra on the right

of Figure 4.1 show that a sinusoidal signal on top (which has only one frequency component) appears to develop more frequencies when a frame is isolated (bottom). This effect is called spectral "leakage" because energy seems to have leaked from the clearly present frequency into neighboring frequencies. Leakage makes signal analysis more difficult since the leaked energy from large peaks can mask smaller peaks nearby. Furthermore, leakage raises the average energy across all frequencies making important peaks seem less significant by comparison.

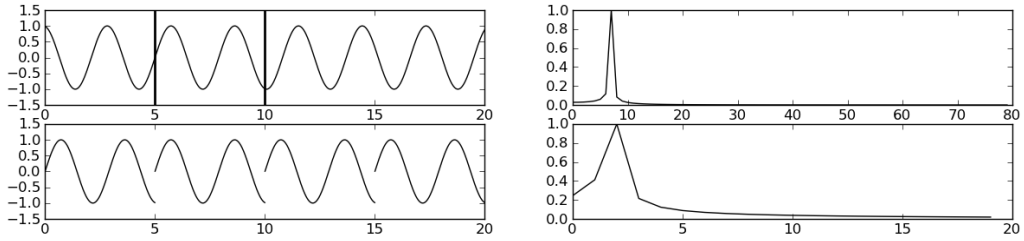


Figure 4.1: Even if the signal $x$ is perfectly sinusoidal, framing can disrupt the continuity at its boundary. The Fourier transform of the sinusoid should have a single significant entry (top left) but framing breaks up the smooth curve (bottom left) causing additional nonzero Fourier coefficients to appear (bottom right).

Leakage can be reduced by smoothing out the discontinuities at the edges of the frame. Multiplying the frame by a continuous "window" function that is (nearly) zero at the edges will make the end of the signal line up with the beginning of its repeat as in the center row of Figure 4.2. Multiplying by a window function in order to modify the spectrum is called "filtering".

To understand this smoothing more precisely, we will discuss its relationship to the convolution theorem:

**Theorem 4.1.** *Suppose* $u = [u_0, u_1, \ldots, u_{n-1}]$ *and* $v = [v_0, v_1, \ldots, v_{n-1}]$ *are signals of equal length. Let*

- $u \odot v = [u_1 v_1, u_2 v_2, \ldots, u_n v_n]$ *be the Hadamard (entry-wise) product*

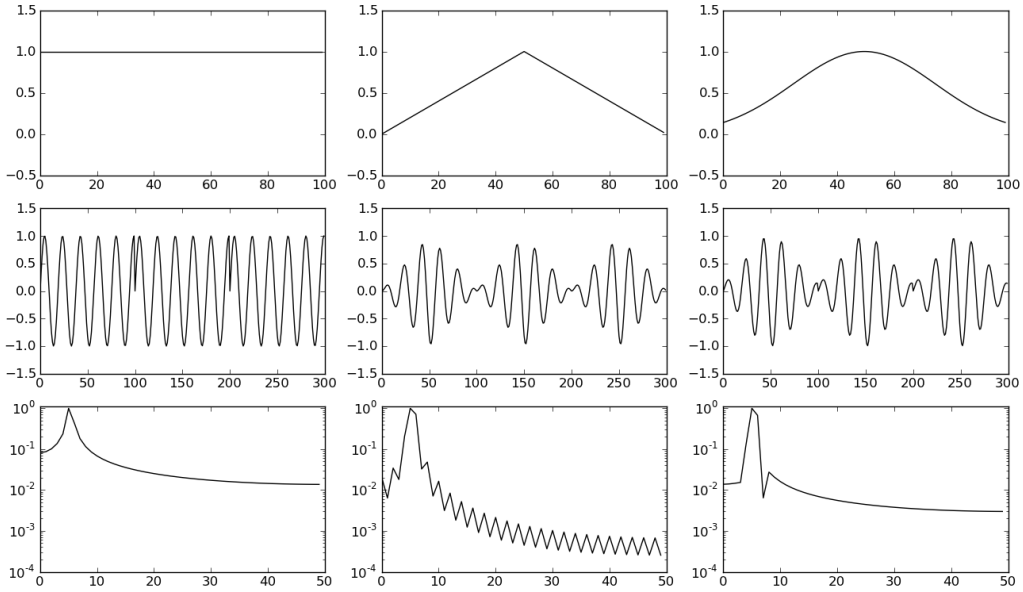


Figure 4.2: Top row: window functions. Center row: product of a sinusoidal frame with the respective windows (the frames are repeated to demonstrate the continuity at when using windows that are nearly zero at the edges). Bottom row: spectra of the respective frames in the center row.

- *$u * v$ denote convolution:*

$$(u * v)_j = \sum_{k=0}^{n-1} u_{(j-k \mod n)} v_k \tag{4.1}$$

- *$\hat{a}$ denotes the discrete Fourier transform of $a$:*

$$(\hat{a})_j = \sum_{k=0}^{n-1} a_k e^{-2i\pi kj/n} \tag{4.2}$$

*Then $\widehat{u \odot v} = \hat{u} * \hat{v}$.*

This theorem is easily proved as in [11].

Now suppose the original signal is $x$ (a vector of length $N$) and the frame $f$ (length $n$) is given by $f_{i-a} = x_i$ for $a \leq i \leq b$. Multiplying $x$ by a the $[a, b]$ interval indicator function

produces a vector related to $f$. Specifically, define the indicator function by

$$R_t = \begin{cases} 1 & \text{if } a \le t \le b \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

so that $x \odot R = \begin{bmatrix} 0 & 0 & \cdots & 0 & f_1 & f_2 & \cdots & f_n & 0 & \cdots & 0 & 0 \end{bmatrix}$ is $f$ padded with zeros (or silence in the context of sound). We will call $f$ a frame and call $x \odot R$ a frame with silence. Assuming $n$ divides $N$, the spectra of $f$ and $x \odot R$ are related by

$$\begin{aligned} |\hat{f}_j| &= \left| \sum_{k=0}^{n-1} f_k \exp(-2i\pi kj/n) \right| \\ &= \left| \sum_{k=a}^{b} x_k \exp(-2i\pi(k-a)j/n) \right| \\ &= \left| \exp(2i\pi a/n) \sum_{k=a}^{b} (x \odot R)_k \exp(-2i\pi kj/n) \right| \\ &= |\exp(2i\pi a/n)| \cdot \left| \sum_{k=0}^{N-1} (x \odot R)_k \exp(-2i\pi k(jN/n)/N) \right| \\ &= 1 \cdot \left| \widehat{(x \odot R)}_{jN/n} \right| = \left| (\hat{x} * \hat{R})_{jN/n} \right| \end{aligned} \tag{4.4}$$

The spectrum of $R$ is not sparse, that is, the entries of $\hat{R}$ are not mostly small (see left column of Figure 4.3). Instead, $\hat{R}$ has a few large peaks and many other frequencies with leaked energy. Consequently, every nonzero coefficient of $\hat{x}$ will leak into other coefficients of the convolution $\hat{x} * \hat{R}$. Since (4.4) shows that the entries of $|\hat{f}|$ are simply $n$ equally spaced entries of $|\hat{x} * \hat{R}|$, $|\hat{f}|$ will typically experience leakage similar to $|\hat{x} * \hat{R}|$.

Leakage can be reduced by using a window function with a Fourier transform more sparse than that of $R$. Examples include triangular functions and truncated Gaussians as in Figure 4.3. Because the spectra of the nonrectangular windows have smaller coefficients away from the main peak, the respective frames with silence (Figure 4.4) and consequently frames without silence (Figure 4.2) have less leakage.

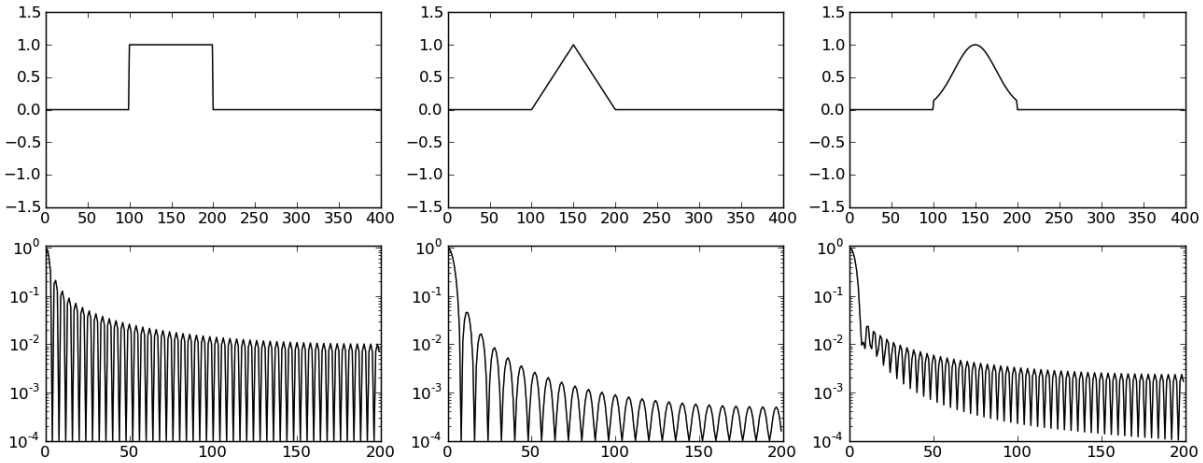


Figure 4.3: Top row: rectangular, triangular and truncated Gaussian window functions respectively. Bottom row: spectra of respective windows on top row. Because the energy in the triangular window decays most quickly with increasing frequency, we expect frames multiplied by triangular functions to have the least leakage.

In some applications, it is preferable to have more total energy leakage if it can be more evenly spread out over the surrounding coefficients. This can cause nearly adjacent spectral peaks to be distinguishable despite the leakage.

Most music analysis begins with analysis of the signal's frames. The results of the frame analyses may be combined or used in other ways, but since the frames are the primary unit of analysis, we may refer to a frame as "the signal."

## 4.2 Binning

The transformed vector $\hat{f}$ is the same length as $f$, that is, there are as many Fourier coefficients as original waveform samples. However, human ears are not sensitive enough to distinguish pitches so finely, and therefore much of the discrete Fourier transform's precision is unnecessary. Many analyses treat groups (or bands) of consecutive frequencies as identical without harming accuracy.

Frequency binning refers to combining bands of frequencies into a single energy coefficient. The binned frequency vector is defined by the weights to apply to each frequency in each

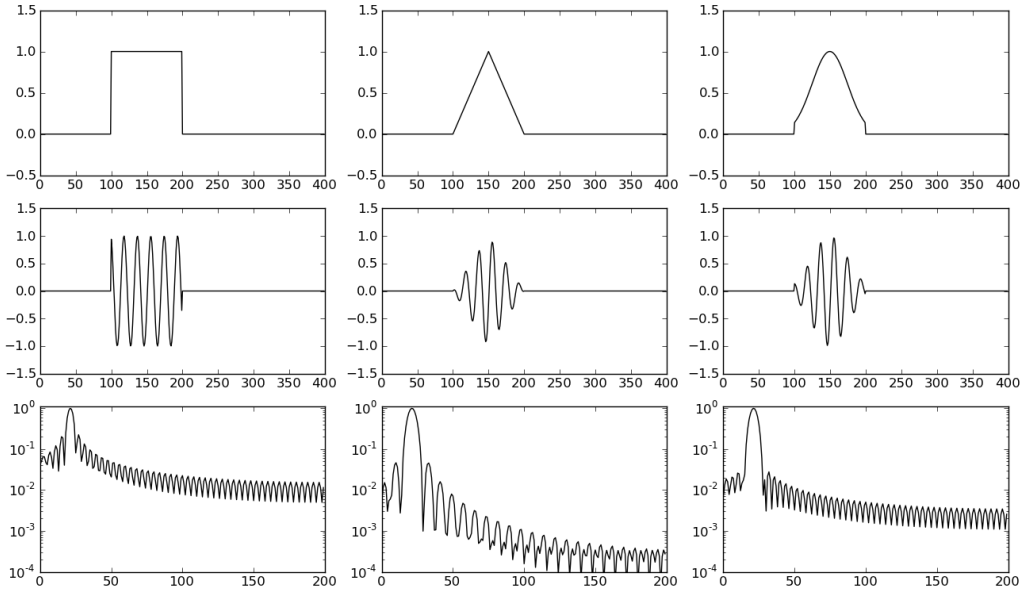


Figure 4.4: Top row: window functions. Center row: frames with silence made by multiplying the full sinusoidal signal by the respective windows in the top row. Bottom row: the spectra of the respective frames with silence in the center row.

bin. If $w_{j,k}$ is the weight applied to frequency $j$ in bin $k$, the bin vector $B$ is given by

$$B_k = \sum_j w_{j,k} |\hat{f}_j| \tag{4.5}$$

If for some $j, k$ it happens that $w_{j,k} = 0$ we may say that frequency $j$ is not in bin $k$. It is common for each frequency to be in only one or two bins.

With proper binning, signals with approximately the same energies in similar frequencies will have similar binned vectors. This is important in music since musicians cannot be expected to always produce precisely the pitch when attempting to produce a particular note. A note played slightly incorrectly (off-key) may move energy into a different Fourier coefficient. Good binning puts the energy from both the correct and incorrect frequencies in the same bin (with similar weight), so there will be little difference between the bin vectors of on-key and off-key performances.

Other advantages of binning are complexity reduction (since one typically uses fewer

bins than Fourier coefficients) and the ability to choose frequencies spaced logarithmically (whereas Fourier coefficients correspond to linearly spaced frequencies).

As an alternative to calculating the weighted sum of all energies, [26] suggests choosing a set of frequency bands $\{b_k\}$ and defining the energy of the band as the maximum energy of any single frequency in the band:

$$B'_k = \max_{j \in b_k}(|\hat{f}_j|) \tag{4.6}$$

They suggest this as a way to avoid producing a spectrogram crowded with many high energy bins in the case of a noisy signal. Pure "noise" is a signal with equal energy in every frequency, and a noisy signal is one where all frequencies have fairly high energy. Important frequencies will be those with energy above the "noise level" that all frequencies share. Since each bin contains the energies from multiple frequencies, the noise energy from each of these frequencies will be included in the bin. The noise energy from many frequencies may dwarf the signal energy and cause important peaks appear to be only random fluctuations. Selecting only the highest energy coefficient to represent each bin avoids this noise compounding problem as the noise energy of only one frequency is included.

Binning the noise energy is more problematic if the noise has limited bandwidth or unequal distribution across frequencies. Bins that receive the most noise energy could be inappropriately marked as significantly more energetic than the other bins. Representing a bin by the maximum energy of a frequency avoids adding many noise energy coefficients and thereby prevents artificial peaks. This makes (4.6) seem like an attractive alternative to binning.

However, if the highest peak in a bin has lost energy to the other coefficients by leakage, ignoring all but the highest coefficient will not reconstitute the energy of that frequency the way that additive binning would. With a single frequency's energy split across several bins, taking only the maximum peak reduces the recorded energy possibly causing a bin to seem insignificant. If the bins are affected by this phenomenon unevenly, the bins with split

energies could be incorrectly treated as insignificant.

## 4.3 Alternatives to FFT

Calculating the discrete Fourier transform (DFT) of a signal directly from the definition (4.2) is generally impractical as it requires $O(n^2)$ operations. Fast Fourier transform (FFT) methods such as the Cooley–Tukey algorithm first described in [6] require only $O(n \log n)$ operations. Cooley–Tukey is the most common approach to computing DFT and implementations are readily available for many programming languages.

We would like to mention some interesting alternatives to FFT techniques that take advantage of music's sparsity in the Fourier domain. In musical signals, only a few Fourier coefficients are usually significant in a single frame. This is partly because of the limited range of frequencies produced by musical instruments. The highest frequencies produced by orchestral instruments is C8 (four octaves above middle–C) which has a frequency of 4186 Hz. Some harmonics may be even higher, but no frequencies are deliberately produced above this level, so the Fourier coefficients for these high frequencies should be small as in Figure 4.5. Lower notes do not have harmonics extending to such high frequencies and thus have even sparser spectra.

The techniques discussed in the next sections randomly start by projecting the signal into lower-dimensional spaces. The chosen projection techniques have been shown to approximately preserve the original information with high probability. Once the low-dimensional projected data is analyzed, it may be reinterpreted in the higher-dimensional Fourier space.

For so called sparse Fourier transform methods, a reduction in computational complexity and time can be achieved in this way. The methods discussed in this paper that are built on Fourier analysis do not require calculating the DFT of any signal longer than frames of approximately 4000 samples. For signals of this size, standard FFT algorithms are not unsatisfactorily slow; if large numbers of songs must be analyzed, minor improvements may prove significant.

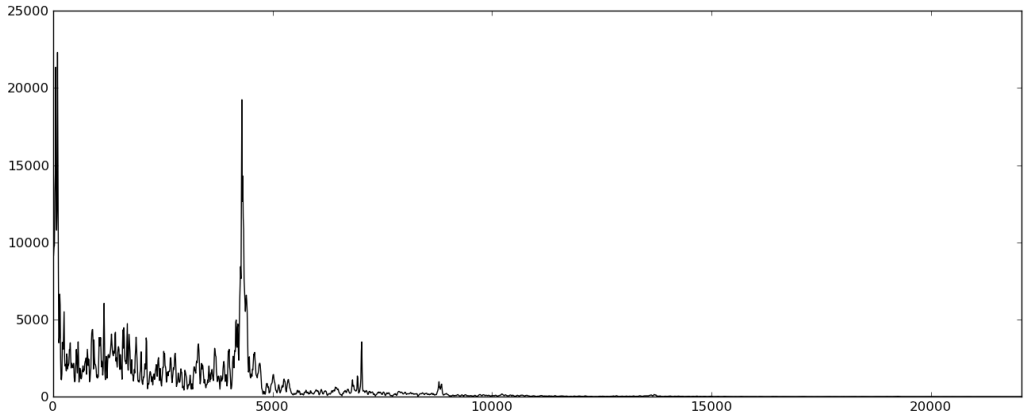



Figure 4.5: The energy spectrum of a piano's highest note, C8. Only a few peaks are noticeable above 5000 Hz. The small coefficients corresponding to high frequencies could be discarded or combined with little loss of information.

Compressed sensing is not a method of speeding up the DFT. In fact it is considerably slower than FFT since it requires solving a linear program. Its strength lies in the number of samples that must be collected. This is unimportant to systems that already have all their songs recorded, but if songs are recorded live on a small device for later analysis when time is not a constraining factor, the reduced number of samples could allow much more audio to be stored with limited memory.

**Sparse Fourier Transform.** Several recent papers demonstrate that Fourier transforms may be approximately computed in less than $O(n \log n)$ time when relatively few of the Fourier coefficients are not (approximately) zero. These algorithms take as a parameter an estimated number of nonzero coefficients $K$ and return $K$ coefficients (with corresponding frequencies) believed to be the $K$ largest after an appropriate number of random trials. Other coefficients are assumed to be zero.

The best-known of these algorithms is the Fast Approximate Discrete Fourier Transform (FADFT) which enjoys a runtime of $O(K^2 \log n)$ as originally presented in [9]. This was later improved to $O(K \log n)$ in [10]. An implementation of FADFT called the Ann Arbor Fast Fourier Transform (AAFFT[1]) is publicly available. An empirical evaluation of the speed and

[1]AAFFT project: http://sourceforge.net/projects/aafftannarborfa/

noise tolerance of several versions of the AAFFT compared to FFT can be found in [14].

A algorithm similar to FADFT with a simpler method of randomization is described in [12]. This algorithm (sFFT for sparse FFT) runs in $O(\log n\sqrt{nK\log n})$ but has a runtime constant small enough that it requires less time than AAFFT even for large enough choices of $n$ and $K$ as to be interesting.

(i) Permute the signal $x$ by indices: $y_i = x_{(\sigma i+\tau \mod n)}$ for some random $\sigma, \tau$ such that $\sigma$ and $n$ are mutually prime

(ii) Split the permuted signal $y$ into equal-width frames (say of length $M$)

(iii) Add the frames together to form the new signal $z$

(iv) Find the Fourier transform of the length-$M$ vector $z$ by a traditional method

(v) Identify the $K$ highest-energy coefficients in the transform $|\hat{z}|$. Call this set $J$.

(vi) Adjust the large coefficients and corresponding frequencies by reversing the permutation:

  (a) define $h(j) = round\left(\frac{\sigma j M}{n}\right)$

  (b) the frequencies to report are $\{j/n : h(j) \in J\}$

  (c) the corresponding coefficients are $\hat{x}'_j = \hat{z}_{h(j)} e^{2i\pi\tau j/n}$ (where $i$ is the imaginary unit, not an index).

(vii) Report these coefficients and frequencies.

As the authors of [12] prove, repeating this process causes the frequencies most often reported as having high energy will be those of largest magnitude in the actual Fourier transform $\hat{x}$. The estimated coefficients for these frequencies may be chosen based on the reported coefficients (such as the median of the reports with real and imaginary medians calculated separately).

The key to this algorithm is the ability to recover information from the randomized and combined frames. The authors prove that $\hat{z}_j$ is the sum of all $\hat{x}_i$ such that $h(i) = j$. Since $\hat{x}$ is assumed to be sparse, few of its entries are significant, so for each $j$, it reasonably likely that only one significant $\hat{x}_i$ has $h(i) = j$ and so $z_{h(i)}$ is likely to be approximately equal to that significant $\hat{x}_i$.

The advantage of this algorithm is that the only Fourier transform calculated directly is that of $z$ which has length $M < N$. If $M$ is chosen to be much less than $N$, even the task of repeating this process many times for better coefficient estimates may be worth this shortcut.

**Compressed Sensing.** The Nyquist–Shannon sampling theorem shows that if the rate of sampling ($r$) is at least twice the frequency of the highest frequency component of a signal, the signal is uniquely identified by the digitized signal. Humans can hear sinusoidal sounds with frequencies up to about 20,000 Hz, so any sound detectable by a human should be able to be reproduced from samples taken 40,000 times per second. Not coincidentally, most audio recordings contain 44,100 samples per second.

The theorem is tight because if frequencies any higher than $r/2$ are allowed, there are signals with high frequency components that will give the exact same digital samples as signals with only lower frequencies. However, if the signal is known to satisfy additional restrictions, then the Nyquist criterion may be subverted without sacrificing uniqueness of the signal.

Compressed sensing deals with exploiting an assumption of sparsity to perfectly reconstruct a signal with fewer samples than the Nyquist theorem dictates. In other words, it is only necessary for all of the signal frequencies to be lower than $r/2$ if the signal is has many frequency components significantly different from zero (i.e. the signal is similar to pure noise). If only a few frequencies are meaningfully present, it is as though these frequencies – although possibly higher than $r/2$ and scattered throughout a wide frequency band – can be "compressed" into a low frequency band which does not require so much sampling.

As discussed in Section 4.3, musical signals have relatively sparse Fourier transforms so they are good candidates for compressed sampling.

# Chapter 5. Obtaining Music Features

In this section, we discuss features of audio data that are useful for describing music. First, each song has feature vectors calculated for each short segment of time. These segments are most often frames, but could be note-events instead as described in Section 5.1.

The means and covariances of the feature vectors across segments are often used as features to describe the entire song for similarity measures and classification. The author of [2] criticizes this approach which loses information about the order in which the features were observed in the song.

When attempting to transcribe the song, the order of the segments is essential. An algorithm may attempt to determine what note or notes are present in the segment. Repeating this process to all segments provides a record of what notes were played when. This is a musical score.

Analysis of what notes appear in each segment need not be independent of time, that is, one may use knowledge of notes played in earlier segments when predicting additional segments' notes. Several authors ([23],[30]) have used hidden Markov models to partially base the predicted notes of one segment on those of neighboring segments.

## 5.1 Segmentation

Frames are chosen to be short relative to the song's changes in pitch, but longer segments could be used if there is some assurance that the pitch (and other spectral features) will change little from the beginning to the end of the segment. Most instruments produce fairly discrete notes (rather than sliding from one sound to another) during which the pitch remains constant and we can hope that other spectral features will also change little. This allows us

to segment songs by note (or "event") instead of frame-by-frame.

Segmenting by events has several advantages. Many notes last at least 0.5 seconds which is longer than the standard 0.05–0.2 frame. This means there are typically far fewer notes than frames in a song, so it will be much cheaper to analyze relationships between features of events than between features of frames. Also, if the events of a song have been located, an important part of transcription has already been accomplished – it only remains to identify the note (or notes) within the event and optionally the instruments involved.

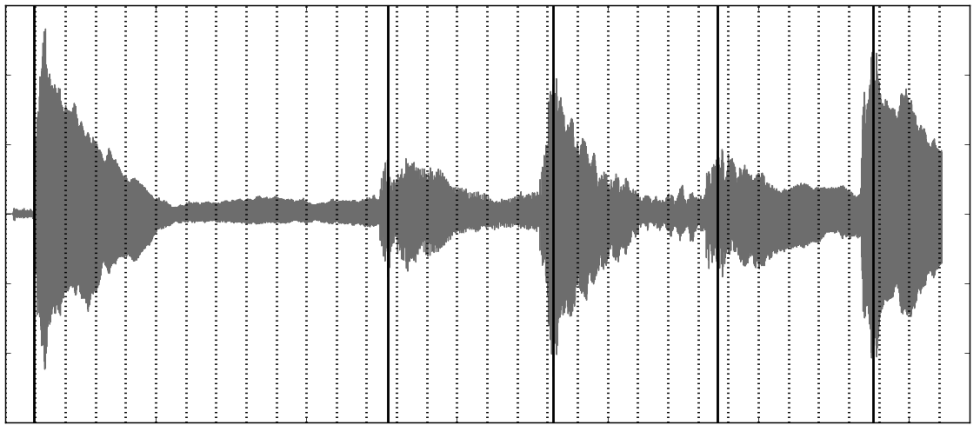


Figure 5.1: The waveform of a an instrument playing five notes. Vertical lines demonstrate two methods of audio segmentation: dashed lines show equal-width frames and solid lines mark approximate event onsets (the intervals between successive solid lines are events). An algorithm might apply either of these segmentation methods and then calculate feature vectors on each segment.

One method of identifying events is to define an "onset detection function" to be calculated on each frame as an estimate of the probability that the frame contains onset of an event. The most important part of locating events in time is to determine when the event begins (or the event's "onset"). An event can reasonably be assumed to end when the next event begins (a note that persists into an event triggered by the start of another note will have to be detected twice in this case). Many different criteria could be used to decide what changes in spectral features indicate the beginning of an event.

A variety of onset detection functions may be formulated. In [7] several methods are compared with the most successful being "spectral flux" onset detection. This is the total

increase in energy across all Fourier coefficients. For the frame $f$ with preceding frame $f'$, this is

$$\sum_{i=1}^{n}(|\hat{f}_i| - |\hat{f}'_i|)^+ \tag{5.1}$$

where $x^+$ denotes the positive part $\frac{x+|x|}{2}$.

Similarly, [31] suggests using the mel-frequency flux (see (5.4)) for onset detection:

$$\sum_{k}(MEL_k(f) - MEL_k(f'))^+ \tag{5.2}$$

Once an onset detection function is calculated, some criterion should be selected for determining how large the function value must be to trigger the prediction of an onset. The author of [31], suggests that an event onset be marked each time a frame's onset function value is greater than the mean of the frames in the preceding 120 ms and greater than all frames within 60 ms.

## 5.2 Cepstrum and Mel-frequency Cepstrum

The cepstrum and mel-frequency cepstrum (MFC) are audio features that attempt to consolidate frequency information based on knowledge about harmonics. As discussed in Section 2.2, regularly spaced peaks in spectral energy contribute to the sense of pitch. By finding the most prominent patterns in the harmonics, we may be able to predict perceived pitches even when the fundamental frequency has harmonics with higher energy or is missing entirely.

The cepstrum of the signal $f$ is typically stated as

$$C = \left|\widehat{\log(|\hat{f}|^2)}\right|^2 \tag{5.3}$$

although the internal squaring only changes the result by a factor of 4. The log reflects the need to exponentially increase energy of a given frequency in order to make its perceived

intensity increase linearly.

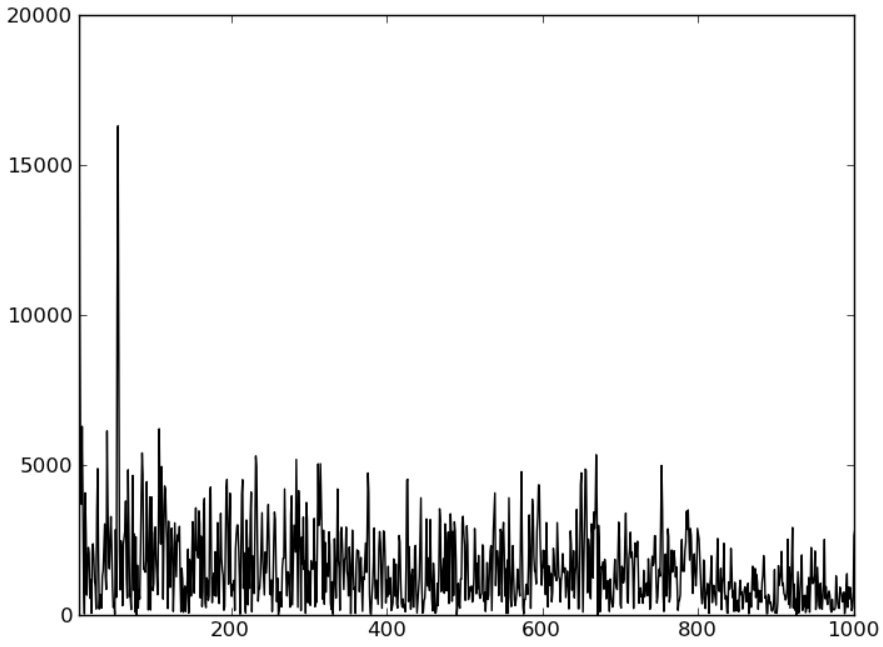



Figure 5.2: The cepstrum of the piano note whose spectrum is shown in Figure 2.1. The peak at 55 indicates the presence of strong 55 Hz harmonics which may be perceived as a pitch of 55 Hz even if the signal contains no energy at this frequency.

The word "cepstrum" was made by reversing the first four letters of "spectrum". Similarly – just as the cepstrum is the spectrum of the (log of the) spectrum – the frequencies of the spectrum are "quefrencies" and filtering the spectrum is "liftering".

A mel-frequency scale is any sequence of frequencies (such as the chromatic piano scale given by (2.1)) with consecutive frequencies having a constant ratio. Each frequency in a mel-scale has a mel-index (such as $i$ in (2.1)). In the notation of (2.1), negative mel-indices are valid (corresponding to low frequencies), but mel-scales are usually shifted so that the mel-index is positive integer for the range of common musical frequencies. Mel-scales are also called scales of "equal temperament" since the log-nature of human pitch perception causes the notes in such a scale to seem equally spaced.

Binning can be used to convert the linearly spaced frequency coefficients of a Fourier transform to a log-spaced mel-scale where each bin value is the energy of a mel-frequency. In

the notation of (4.5), the weight applied to coefficient $j$ for the bin with mel-index $k$ is $w_{j,k}$. The weights $w_{j,k}$ can be chosen in any desired way, but the triangular weighting schemes shown in Figure 5.3 are popular. For the experiments in Chapter 6, we will instead use a "rectangular" binning scheme where a constant number of coefficients are in each bin and each coefficient has uniform weight (as described in (6.5)).

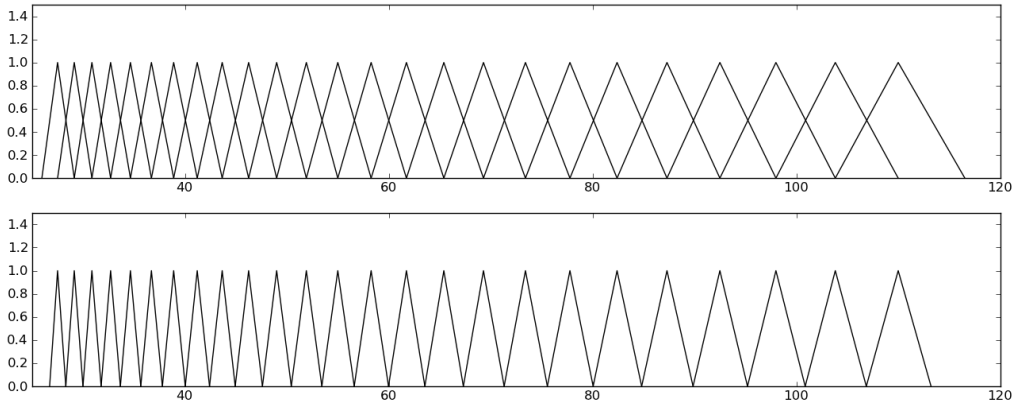



Figure 5.3: Two schemes for binning according to the same log-spaced mel-scale: with 50% overlap (top) and no overlap (bottom). Each triangle is the graph of a column of $w$ in (4.5), that is, a graph of $w_{j,k}$ as a function of $j$

Several authors have suggested scales that are not quote logarithmic such as a scale that is linear in low frequencies and logarithmic above 1000 Hz (as in [16]), a scale that imposes positivity on the mel-index ([8] and [21]) and the "Bark" scale in [34] which makes creative use of trigonometric functions to achieve a log-like function.

The MFC is similar to the cepstrum but looks for periodicity in the mel-frequency coefficients rather than the Fourier coefficients. This is sensible in a musical setting since harmonics are exponential in the frequency domain. The MFC typically employs a discrete cosine transform (DCT) instead of a second Fourier transform:

$$MEL_k = \sum_{j \in \text{bin } k} w_{j,k} |\hat{f}_j| \tag{5.4}$$

$$MFC = DCT(MEL)$$

## 5.3 Other Spectral Features

If the signal in one frame is given by $f$ and the next frame is $f'$, the following features are recommended by [31] to describe $f$:

- **Flux**: The total change in spectral energy across frequencies

$$\|\hat{f} - \hat{f}'\|_1$$

- **Euclidean Volume**: The loudness of this frame

$$\|\hat{f}\|_2$$

- **Spectral Flatness**: A standard measure of similarity of the frame to white noise (higher is more disorderly)

$$\frac{\exp\left(1/n \sum_{i=1}^{n} \log(|\hat{f}_i|)\right)}{\|\hat{f}\|_1/n}$$

- **Entropy**: An alternative measure of noisiness

$$\sum_{i=1}^{n} p_i \log(p_i)$$

  where

$$p_i = \frac{|\hat{f}_i|}{\|\hat{f}_i\|_1}$$

- **Centroid**:

$$\frac{\sum_{i=1}^{n} i|\hat{f}_i|}{\|\hat{f}\|_1}$$

- **Sample Variance, Skewness and Excess Kurtosis**: These are standard statistical

descriptions of the distribution of the spectrum If the $k$th central moment is given by

$$m_k = \sum_{i=1}^{n} (|\hat{f}_i| - centroid)^k / n$$

Then these features are

$$\begin{aligned} Variance &= m_2 \\ Skewness &= \frac{m_3}{m_2^{3/2}} \\ Excess_Kurtosis &= \frac{m_4}{m_2^2} - 3 \end{aligned}$$

Additionally, [33] and [4] include the **zero-crossing rate**:

$$\frac{|\{i : f_i \cdot f_{i+1} < 0\}|}{n}$$

# Chapter 6. Optimization-based Transcription

In Section 3.3 we discussed methods of automatic music transcription. Identifying notes typically involves mapping dominant frequencies and quefrencies to the chromatic scale. The instrument used to produce the note might be identified by comparing audio features of the song to the features of training samples.

Identifying instruments through comparison to training samples can create a problem. Training samples typically contain multiple notes, and notes in different ranges can have significantly different timbre. For example, the physical properties of piano strings used for very high and low notes result in harmonics that are not quite integral fractions of the fundamental frequency. This causes the lowest piano notes to have a metallic buzz and the

high notes to have a hollow pinging sound. Midrange notes do not have this problem. Soprano clarinets have three distinct timbres: the low, woody chalumeau; the bright, midrange clarion; and the high, shrill altissimo. These differences in timbre affect most of the audio features discussed in Chapter 5 and these differences may prevent extreme notes from being matched to the feature of the training set.

## 6.1 Algorithm Setup

In this experiment we represent an instrument by a collection of feature vectors each taken from samples of the instrument playing a single note. Any frame in a test signal containing a particular instrument playing a specific note should be similar to the corresponding training vector.

It is possible that a similar result could be obtained by using many play samples that emphasize the various registers of the instrument's range. Such samples would exhibit a similar variety of timbres to our samples of individual notes. If a frame's features are similar to the low-register piano sample, it probably contains a piano playing a note in the lower part of its range. However, using individual note samples provides greater interpretability: if a frame's features are similar to the features of a piano playing B3, we can predict that a piano was playing B3 at that time.

To transcribe a song, we first obtain a single feature vector $w_i$ for each instrument/note pair $i$ in the training data. We then obtain a sequence of frame feature vectors $v_1, v_2, \ldots v_m$ for the song that is to be transcribed. We would like to determine which of the training feature vector(s) $w_i$ each frame's vector $v_j$ is most like. We will allow the possibility of several notes being played concurrently by the same or different instruments, so rather than simply finding a training feature vector $w_i$ that is similar to $h_j$, we will look for a combination of training feature vectors that approximates the test frame $j$'s vector. If we only choose features (such as the spectrum) that are linear in the waveform, then the features of notes

are additive and the problem becomes finding $h_j$ such that

$$v_j = Wh_j \tag{6.1}$$

Each entry $h_{i,j}$ of the vector $h_j$ will represent the energy input of the instrument/note combination $i$ predicted to be present in the test sample.

If there are more training samples than features, (6.1) is an underdetermined problem with infinitely many exact solutions. Many criterions could be used to select a particular choice of $h_j$, but because of the sparsity of musical data in Fourier space (discussed in Section 4.3) we will attempt to find the sparsest solution.

Truly finding the sparsest solution $h_j$ was been shown to be an NP-hard problem in [20]. Consequently, the problem is prohibitively expensive to solve over a large number of dimensions. This is essentially because one must iterate over all subsets of entries of $h_j$ and attempt to find a solution whose support is contained by that set of entries. For a piano, any combination of its 88 notes could be heard at any given time. Even if it is known that only 3 notes will be played at a time (i.e. there is a solution $h_j$ to (6.1) with only 3 nonzero entries), there are 109,736 subsets to check. Using a larger basis $W$ with more notes and instruments and allowing the possibility of more simultaneous notes creates an enormous number of subsets of instrument/note combinations to try.

Since a direct search for the sparsest solution is unreasonable, we turn to an approximation. For reasons that are well-documented (such as in [5]), minimizing the $\ell_1$ norm of the choice vector often finds sparse solutions. Remembering that an instrument cannot play negatively suggests a nonnegativity constraint on the problem. Using these facts turns (6.1)

into a linear program

$$\begin{aligned} \text{for each frame } j : \min_{h_j} & \|h_j\|_1 \\ \text{s.t. } & Wh_j = v_j \\ & h_j \geq 0 \end{aligned} \tag{6.2}$$

However, this problem is not feasible in general. If $v_j$ lies outside the positive cone formed by the columns of $W$, no positive combination of $W$'s columns can reach $v_j$ (such as when $W_{i,k} > W_{j,k}$ for all $k$ but $v_i < v_j$). This appears to be a practical problem as well as a theoretical one since this condition arose in our experiments. Consequently we look to altered versions of (6.2).

If we replace the highly restrictive equality constraint with a penalty for error, every $h_j \geq 0$ is a candidate so the problem is feasible:

$$\begin{aligned} \text{for each frame } j : \min_{h_j} & \|Wh_j - v_j\|_1 + \gamma\|h_j\|_1 \\ \text{s.t. } & h_j \geq 0 \end{aligned} \tag{6.3}$$

Increasing $\gamma$ emphasizes the $\ell_1$ norm (and consequently the sparsity) of $h_j$. Lower values of $\gamma$ increase the relative penalty for error thereby emphasizing the accuracy of the solution.

The solution's sparsity is achieved by the $\|h_j\|_1$ term in (6.3) but the choice of the $\ell_1$ norm for the error term is rather arbitrary. We could instead use the sum of squared errors to create the comparable quadratic program:

$$\begin{aligned} \text{for each frame } j : \min_{h_j} & \|Wh_j - v_j\|_2^2 + \gamma\|h_j\|_1 \\ \text{s.t. } & v_j \geq 0 \end{aligned} \tag{6.4}$$

## 6.2 Training Data

For these experiments, we used the Real World Computing Musical Instrument Sound Database[1] which contains recordings of many instruments playing a single note at a time. Each note in the instruments' ranges is played in a variety of styles (staccato, trilling, with pedal, etc.) and with a variety of brands of instruments.

The basis matrix $W$ is formed by the following process

(i) For each note in each instrument's range

  (a) For each play style, volume and instrument brand

    i. For each 0.1 second frame $f$ (overlapping by half with previous frame)

      A. Multiply $f$ by a triangular window function

      B. Obtain the half of the energy spectrum corresponding to positive phase:

$$|\hat{f}_i| \text{ for } i \leq n/2$$

      C. Obtain the bin vector $B$ of the half-spectrum according to equations (6.5) and (2.1).

    ii. Average the spectrum and feature vectors across frames

  (b) Average the features vectors across styles, volumes, brands

We used a mel-scale binning scheme with bin centers at the frequencies of the 12-note chromatic scale. In the notation of (2.1) and (4.5) this is

$$\text{For each bin } j \text{ and frequency index } i : w_{i,j} = \begin{cases} 1 & \text{if } ri/n \text{ is one of the 4 Fourier frequencies} \\ & \quad \text{closest to } piano_j \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

[1] http://staff.aist.go.jp/m.goto/RWC-MDB/rwc-mdb-i.html

## 6.3 Test Data

To compare the effectiveness of (6.3) and (6.4), we recorded the sound of a piano performing a portion of the Russian folk song "Korobeiniki." Specifically, the first four measures were played with only the melody (top notes in Figure 6.1) and again with the melody and harmony (all notes in Figure 6.1).

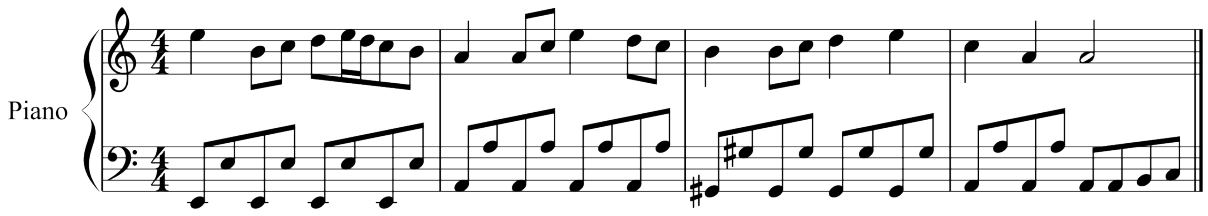



Figure 6.1: The first 4 measures of "Korobeiniki"

The same framing and binning procedure used on the training data was applied to the recording. Since the mel-scale binning used corresponds to the chromatic scale, the general shape of the music is discernible in the binned data shown in Figures 6.2. The bins corresponding to each struck note is visible, but the overtones' bins are visible as well. Treated as a transcription, the binned spectral data suggests that the piano keys corresponding to each of the melody's overtones should be struck. This is an interesting but unsatisfactory transcription.

We used the CVXOPT python package [1] to solve (6.3) and (6.4) with various values of $\gamma$. Finding an optimal value of $\gamma$ would require an objective measure of the quality of the transcriptions produced by various $\gamma$ values. This would be difficult if perfect transcriptions are not available for comparison. Instead, one could vary $\gamma$ and compare the results of experiments such as this:

(i) transcribe a collection of music (with known genres) by (6.3) and (6.4) with a particular value of $\gamma$

(ii) measure similarity between songs based on the transcriptions as in [29]

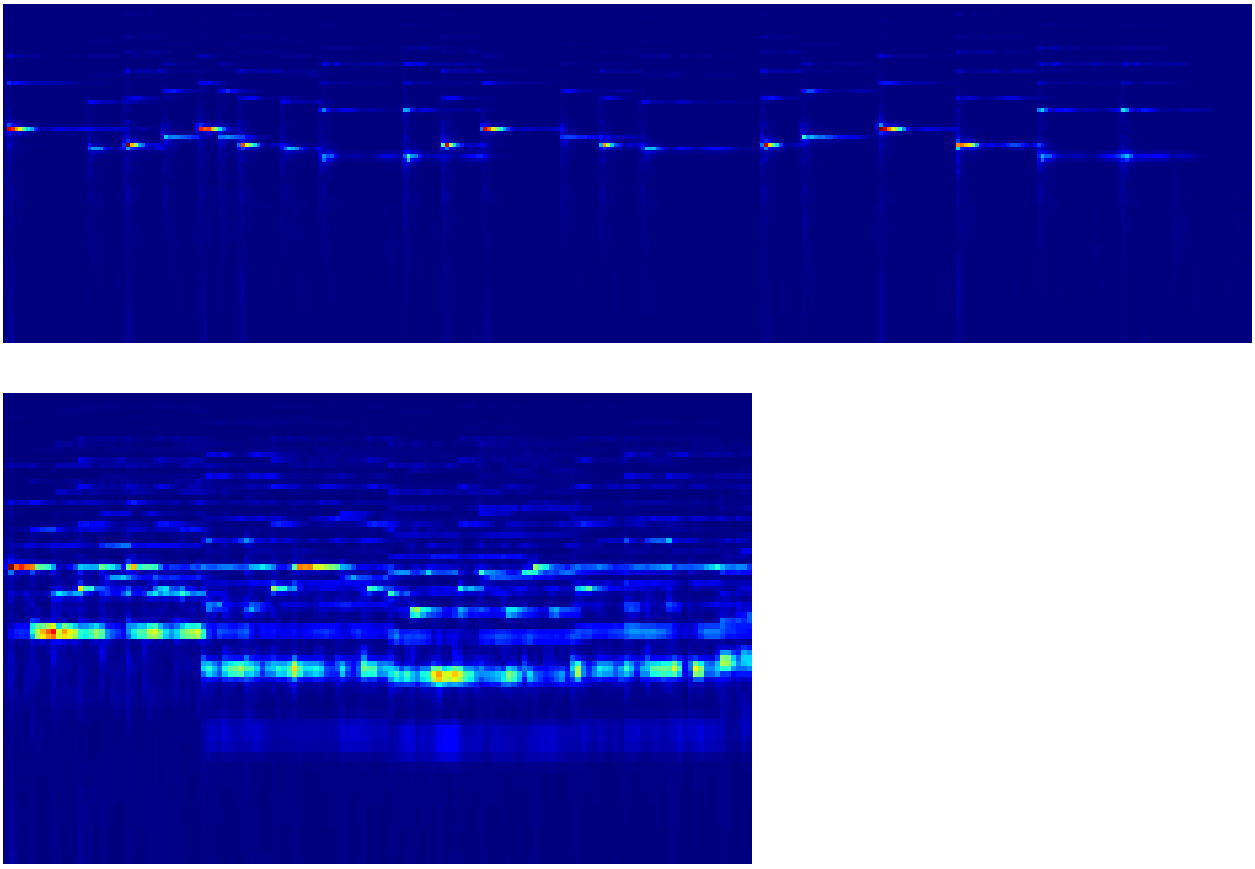


Figure 6.2: Binned frequency data $V$ for the melody (top) and the melody with harmony (bottom). The height corresponds to the mel-index of the frequency bin (higher frequencies are positioned higher) and the horizontal position corresponds to the frame position in the song. In the top melody matrix, each note in the melody is visible as the high-energy coefficients across the center of the matrix. The note's overtones are visible as the higher-frequency lines above the melody lines. In the bottom harmony matrix, the melody is still visible but difficult to distinguish from the overtones of the harmony.

(iii) classify the songs according to their transcription-similarity to a few examples

(iv) measure the accuracy of the classifications

We instead relied on a visual inspection of the transcribed matrices (as in Figures 6.3 and 6.4). By coincidence, $\gamma$ value of 0.1 seemed appropriate for both (6.3) and (6.4): higher values produced extremely sparse solutions that lacked many important notes while lower values produced solutions with so many visible overtones that it was not possible to distinguish the struck notes.

Solving (6.3) produces the transcribed solution matrices $H$ gives in Figure 6.3. The

melody's overtones are somewhat visible and the low notes of the harmony in Figure 6.3 are better-resolved than in the original binned data in Figure 6.2. If we interpret the matrix as a transcription, the significant overtones are taken to be notes actually struck by the pianist. Since this was not actually the case, we would prefer a solution where the actually played notes account for the frequencies of the overtones and the overtone lines do not appear in the transcription.

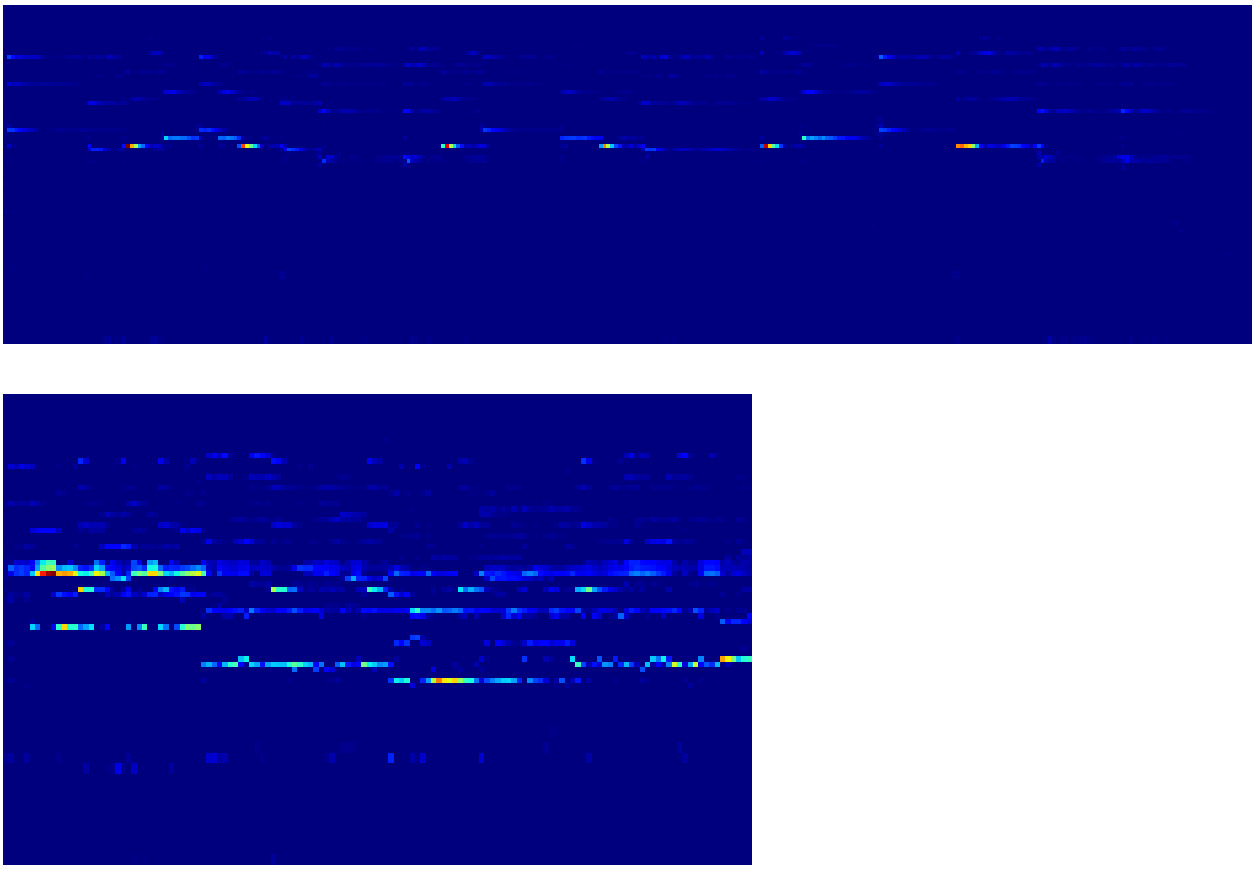


Figure 6.3: Melody (top) and melody with harmony (bottom) as transcribed by (6.3) with $\gamma = 0.1$

Compare Figure 6.3 to the solutions of (6.4) shown in Figure 6.4. Figure 6.4 shows very few overtones and – in the melody-only solution – shows most of the correct notes strongly. Unfortunately some actually played notes fail to appear in this solution. The sixteenth notes (the fifth and sixth notes of the melody) do not appear in the melody-only solution and hardly any of the melody is visible in the transcription that includes the harmony (bottom). This

solution to (6.4) has provided greater sparsity than the solution to (6.3) but at the expense of important note information being discarded along with the unimportant overtones.

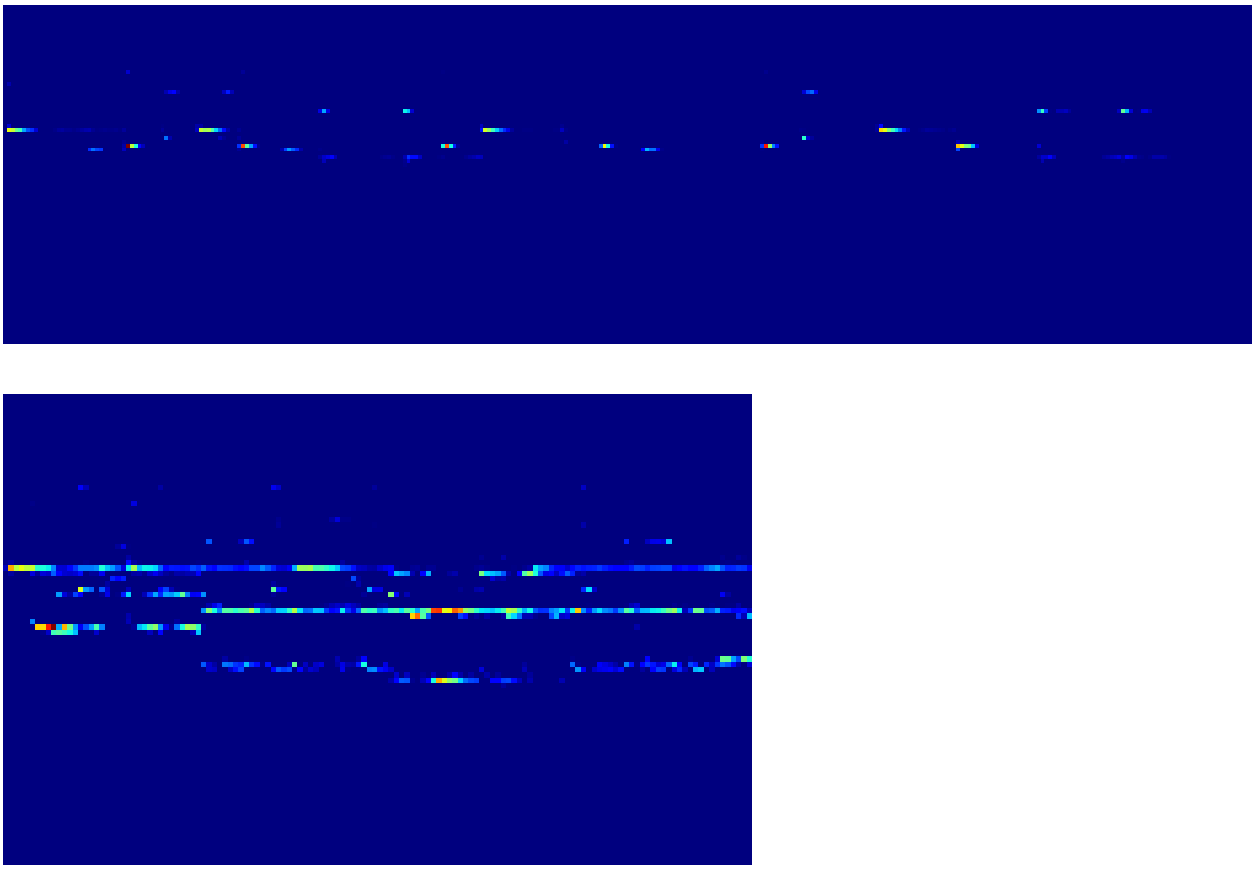


Figure 6.4: Melody (top) and melody with harmony (bottom) as transcribed by (6.4) with $\gamma = 0.1$

# Chapter 7. Conclusion

Music analysis is a complex process. When the audio content is analyzed directly, there are many ways to overcome the obstacles that arise and different combinations of these solutions provide different results. To help understand the advantages of analysis techniques, we have discussed the human perception of music and methods of summarizing the musical experience in feature vectors. We have discussed the relationships between important goals that can be reached by music analysis and given examples of other authors' methods of

accomplishing specific tasks with this interrelated system. To address the problem of timbral variations across instruments' ranges, we have introduced a system of optimization-based transcription that avoids averaging instrument features across multiple notes. We hope that the introductory material has been instructive and that our original algorithm will inspire additional research in music analysis.

# Bibliography


[1] Martin Andersen, Joachim Dahl, and Lieven Vandenberghe. Cvxopt 1.1.5. http://abel.ee.ucla.edu/cvxopt/index.html, 2012.

[2] Jean J. Aucouturier. *Ten experiments on the modeling of polyphonic timbre*. PhD thesis, University of Paris 6, France, 2006.

[3] Jean J. Aucouturier, F. Pachet, and M. Sandler. The way it sounds: Timbre models for analysis and retrieval of polyphonic music signals. *IEEE Transactions of Multimedia*, No. 6, 2005.

[4] Emmanouil Benetos, Margarita Kotti, and Constantine Kotropoulos. Musical instrument classification using non-negative matrix factorization algorithms and subset feature selection. *IEEE*, pages 221–224, 2006.

[5] E.J. Candés, J. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52, 2006.

[6] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex series. *Mathematical Computation*, 19, 1965.

[7] S. Dixon. Onset detection revisited. In *9th International Conference on Digital Audio Effects*, 2007.

[8] Gunnar Fant. *Speech Sounds and Features*. MIT Press, 1973.

[9] A. Gilbert, S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss. Near-optimal sparse Fourier estimation via sampling. *Association for Computing Machinery Symposium on Theory of Computing*, 2002.

[10] A. Gilbert, S. Muthukrishnan, and M. Strauss. Improved time bounds for near-optimal sparse Fourier representations. *Society for Optics and Photonics*, 2005.

[11] Richard W. Hamming. *Digital Filters*. Courier Dover, 1997.

[12] Haitham Hassanieh, Piotr Indyk, Dina Katabi, and Eric Price. Simple and practical algorithm for sparse Fourier transform. In *Proceedings of the Twenty-Third Annual Association for Computing Machinery–Society for Industrial and Applied Mathematics Symposium on Discrete Algorithms*, SODA 2012, pages 1183–1194. Society for Industrial and Applied Mathematics, 2012.

[13] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Conference on Research and Development in Information Retrieval*, 1999.

[14] M.A. Iwen, A. Gilbert, and M. Strauss. Empirical evaluation of a sub-linear time sparse dft algorithm. *Communications in Mathematical Sciences*, 5, 2007.

[15] J. Jiang, P. Coffey, and B. Toohey. Improvement of odor intensity measurement using dynamic olfactometry. *Journal of the Air and Waste Management Association*, 56, 2006.

[16] W. Koenig. *A new frequency scale for acoustic measurements*, volume 27. Bell Telephone Laboratory Record, 1949.

[17] Thibault Langlois and Gonçalo Marques. A music classification method based on timbral features. In *ISMIR*, 2009.

[18] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401, 1999.

[19] José M. Martnez. Mpeg-7 overview. http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm, 2004.

[20] B. K. Natarajan. Sparse approximate solutions to linear systems. *Society for Industrial and Applied Mathematics Journal on Computing*, 24, 1995.

[21] Douglas O'Shaughnessy. *Speech communication: human and machine*. Addison-Wesley, 1987.

[22] Mari Ostendorf and Salim Roukos. A stochastic segment model for phoneme-based continuous speech recognition. *IEEE Transactions on Acoustics, Speechm and Signal Processing*, 37, No. 12, 1989.

[23] Christopher Raphael. Automatic transcription of piano music.

[24] Paul Resnikoff. Drowning? The iTunes store now has 28 million songs... *Digital Music News*, April 25, 2012.

[25] Paris Smaragdis and Judith C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2003.

[26] A. M. Stark and M. D. Plumbley. Real-time chord recognition for live performance. In *Proceedings of International Computer Music Conference*, 2009.

[27] Peter M. Todd and D. Gareth Loy. *Music and Connectionism*. MIT Press, 1991.

[28] Konstantinos Trohidis, Grigorios Tsoumakas, George Kalliris, and Ioannis Vlahavas. Multi-label classification of music into emotions. In *ISMIR*, 2008.

[29] Rainer Typke. *Music Retrieval based on Melodic Similarity*. PhD thesis, Utrecht University, Netherlands, 2007.

[30] Emmanuel Vincent and Xavier Rodet. Music transcription with ISA and HMM.

[31] Kris West. *Novel techniques for Audio Music Classification and Search*. PhD thesis, University of East Anglia, 2008.

[32] Tim Westergren. The Music Genome Project. http://www.pandora.com/about/mgp.

[33] Changsheng Xu, Namunu C. Maddage, and Xi Shao. Automatic music classification and summarization. *IEEE transactions on speech and audio processing*, 13, No. 3, 2005.

[34] E. Zwicker and E. Terhardt. Analytical expressions for critical-band rate and critical bandwidth as a function of frequency. *Journal of the Acoustical Society of America*, 68, 1980.