Bachelor's Thesis

Real-Time Polyphonic Pitch-Detection for the Harpsichord

Anders Øland (anoe@itu.dk)

Supervisor: Dr. Joseph Kiniry (josr@itu.dk)

Research adviser: Prof. Roger B. Dannenberg (CMU)

...and featuring Prof. Richard Stern (CMU) on the harpsichord

Abstract

In this thesis we study the problem of polyphonic pitch-detection for the harpsichord in real-time. We approach the problem from two angels; spectral analysis and non-negative matrix factorization. The latter method was successfully implemented; using the Euclidean distance with multiplicative updates - for learning a dictionary of notes. Pitch-detection is then performed as a non-negative matrix decomposition using the Beta-divergence, as described by Dessein et al. Furthermore, we show results indicating that squaring the learned note activation vectors, from a good input signal, can be seen to cancel out noise and ease the interpretation task; with some trade-off in the precision of the note duration detection. Lastly, we present a similar novel extension to this method, where the use of two microphones, providing radically different input signals, is seen to provide more accurate durations, while still canceling out noise and ambiguities, simply by multiplying their activation vectors. However, the results we present are not conclusive, they merely suggest some promising results.

Contents

1 Introduction **6**
- 1.1 Problem Statement . . . 6
- 1.2 Motivations . . . 6
- 1.3 The Harpsichord . . . 7
- 1.4 The Recordings & Data . . . 7
- 1.5 Method & Tests . . . 8

2 Music & Data Representation **11**
- 2.1 Music Representation . . . 11
- 2.2 The Features of Music . . . 12

3 Spectral Analysis **18**
- 3.1 Summing Harmonic Amplitudes . . . 18
- 3.2 Heuristic Bottom-Up Approach . . . 18
- 3.3 Harmonic Fingerprints . . . 20

4 Non-Negative Matrix Factorization for Pitch-Detection **23**
- 4.1 Non-Negative Matrix Factorization . . . 23
- 4.2 NMF for Pitch-Detection . . . 26
  - 4.2.1 NMF with the Beta-divergence . . . 27
  - 4.2.2 Non-Negative Decomposition with the Beta-divergence . . . 28
  - 4.2.3 Overview of the System . . . 28
- 4.3 Tests & Analysis . . . 31
  - 4.3.1 Squaring Out the Noise . . . 31
  - 4.3.2 Improving Note Detection by Using Two Microphones . . . 33
- 4.4 Informal Assessment of the Method . . . 38
  - 4.4.1 Comparison with Melodyne . . . 38

5 Conclusion & Future Work **41**

Nomenclature

ADC	Analog-to-Digital Converter
DAC	Digital-To-Analog Converter
DFT	The Discrete Fourier Transform
MIDI	Music Instrument Digital Interface
MIR	Music Information Retrieval
NMF	Non-Negative Matrix Factorization
STFT	Short-Time Fourier Transform

List of Figures

1.1 An illustration of the mechanics of the harpsichord [wikipedia.com]. 8
1.2 The two Neumann KM84's placed in a 110 degree XY position, and the Shure SM57 below pointing towards the mid front, where the picks are placed. 9
1.3 The Audio-Technica AT836 placed in the far end of the instrument, pointing away from the front. 9

2.1 Piano Roll visualization of MIDI data. 12
2.2 Discretizing an audio signal, pre-processed with an anti-aliasing filter (which has no effect in this case). 13
2.3 An example of aliasing; x_a has a frequency of 90 Hz, but when sampled at rate 100 Hz the sampled result, x_b, gets frequency 10 Hz [12]. 13
2.4 Modern staff notation (source: Wikipedia). 14
2.5 Magnitude spectrograms showing the difference between two vowels [8] . . 15
2.6 Spectrograms showing effects of window and step size; on a recording of a C major scale played with a single sinusoid. **Left:** Decreasing window size (top to bottom), fixed step size. **Right:** Decreasing step size, fixed window size. 16

3.1 Klapuri's method on a monophonic chromatic scale played on the harpsichord. The topmost plot shows the target notes, the middle one the estimated f0 saliences, and on the bottom plot the predominant f0 saliences have been enhanced 19
3.2 Klapuri's method on a monophonic chromatic scale played on the harpsichord. The topmost plot shows the target notes, the middle one the estimated f0 saliences, and on the bottom plot the predominant f0 saliences have been enhanced 21
3.3 22

4.1 A figure from Lee & Seung's original article on NMF; showing the parts-based (additive) representation learned by NMF, contrasted with the holistic representations obtained from PCA or vector quantization (PQ) [9]. The bigger squares, ie. matrices, are often refereed to as *dictionaries*. 24
4.2 Schematic view of the system as presented in [4] 29
4.3 A test visualizing the convergence time if the algorithm. 30

4.4 Unfiltered note activations, frame by frame, yielding a pianoroll. The input is a chromatic scale going upwards, covering the entire range (on the plot observed as the line going from the lower left corner to the upper right). . . 32
4.5 Squaring out the noise on the SM57 33
4.6 Squaring out the noise on the AT836 34
4.7 Pianoroll representaions of unprocessed activation vectors from two sources 35
4.8 Pianoroll representaions of unprocessed squared activation vectors from two sources 36
4.9 Pianoroll representaions of unprocessed squared activation vectors from two sources 37
4.10 Comparing our transcription, using the piece-wise multiplication of the activation vectors from the SM57 and the AT836, with MIDI data from another performance of the same piece 39
4.11 Transcription done off-line with Celemony's Melodyne, also using two mic's (SM57 & AT836) 40

About This Thesis

This thesis was written on the sixth and final semester of the Bachelor's Degree in Software Development (BSWU) at The IT University of Copenhagen (ITU). It was written in May 2012 under the supervision of Dr. Joseph Kiniry (ITU). The problem of polyphonic pitch-detection for the harpsichord was proposed by Prof. Roger B. Dannenberg (Carnegie Mellon University), who has also acted as a research adviser. Prof. Richard Stern (Carnegie Mellon University) played the harpsichord, and last but not least Assistant Prof. Riccardo Schulz (Carnegie Mellon University) kindly supplied the recording gear and assisted in the recording session.

Note: the sections on music representation and the theoretic foundation of the general non-negative matrix factorization are adapted from the author's own work, Machine Learning and Its Applications to Music [Anders Øland, December 18, 2011]; available from andersoland.com.

Prerequisites

It is assumed that the reader at least possesses knowledge equivalent to that of a B.Sc. in Software Development. Furthermore, we expect the reader to have access to all the project materials, and have basic skills in linear algebra, probability, MATLAB programming, signal processing, machine learning and spectral analysis.

Structure

The structure of this report is linear. Thus, the understanding of each section may depend on any of its preceding sections.

Mathematical Notation

We will try to maintain a coherent notation, but some ambiguities may occur. Generally, vectors and matrices are written in bold, $\mathbf{x}^T$ denotes the transpose of the vector $\mathbf{x}$, $\mathbf{x}^{(i)}$ denotes the ith example of x from a given data set, $\|\mathbf{x}\|$ is the length of a vector (or the norm of a matrix), $|\mathbf{M}|$ is the determinant of $\mathbf{M}$, and $:=$ denotes value assignments. $A \otimes B$ and $\frac{A}{B}$ are the element-wise multiplication and division, respectively, and $\mathbf{A}^{\cdot p}$ denotes the element-wise power of p.

Enclosed CD

The MATLAB code, test cases, data and plots are included on the enclosed CD. It will also be available from andersoland.com (.../portfolio/thesis).

Chapter 1

Introduction

1.1 Problem Statement

We shall explore different approaches to real-time polyphonic pitch detection for the harpsichord. The goal is to decide which solution is most suited to be used in a computer accompaniment system (to be developed by Rich Stern @ CMU). The task can be thought of as a live transcription of the notes played on a harpsichord - obtained through real-time processing and interpretation of the audio signals captured by one or more microphones.

Performing polyphonic pitch-detection, or multiple f0 estimation, is hard. In fact, a quick search on the topic on scholar.google.com results in several thousand papers and citations - since 1993 alone. The problem has been approached with a wealth of very different techniques; ranging from the quite simple to the more elaborate ones. Yet, a robust general solution still remains to be found. In this thesis we are not trying to solve the general problem. Merely, we shall study possible solutions for a specific instrument, under certain conditions and in a specific setting. This will allow for some useful constraints and priors, which we shall try to utilize. We do not expect to obtain any perfect solution; especially given the real-time constraint. Simply, we wish to provide a pointer in the right direction, and regard this as preliminary work with respect to arriving at a final working solution.

1.2 Motivations

The problem was proposed by Prof. Roger B. Dannenberg, as a first step in developing a accompaniment system for the harpsichord based on real-time score following. Prof. Dannenberg has developed methods for score following, which Prof. Stern wishes to use for the harpsichord - as he is a keen player and owner of several harpsichords.

Music Production & Live Performance

Knowing the pitches in an incoming music audio signal has many useful applications in music production and live performance. For creative uses, one can imagine a wealth of interesting applications where some music software reacts to the incoming notes from some instrument. This could be in the shape of computer generated music, using machine learning, or just simple arpeggios based of the harmonic analysis of a guitar input signal. For music production purposes, this knowledge may be used for adaptive signal processing effects in real-time.

Music Information Retrieval

With the digitalization of music, a new and rapidly growing research area has emerged, called Music Information Retrieval (MIR). This research is focused on the extraction of information from music audio and musical scores. It involves tasks such as music genre classification, music transcription, instrument classification, beat detection, blind instrument separation, to name a few. Extracting such features is becoming increasingly necessary, as online music databases and web-services, not to mention iTunes, grow in popularity.

1.3 The Harpsichord

The harpsichord is a piano-like instrument on which the strings a plucked with plectrums instead of being hammered like on a piano, as seen on figure 1.1. This produces a very distinct tone, which is well-known from Baroque classical music. They come in various sizes, but a generally smaller than pianos. Harpsichords often have multiple choirs of strings in different lengths (typically 8 or 4 feet.) - which can be permuted in different ways; yielding different tones. In this thesis we only work with recordings of one permutations, namely the 8 feet string set.

1.4 The Recordings & Data

The recordings used for this work were made in Rich Stern's living room using high-end professional equipment. They were recorded in the Avid Pro Tools software with PreSonus microphone preamps (Digimax), at a 44.1 kHz sampling frequency with a 16 bit resolution, using four microphones; three different models with very different frequency responses:

- Two Neumann KM84; condenser mic's. Typically used for overhead and ambient recordings.
- One Shure SM57; dynamic mic. Mostly used for close-mic'ing (probably the most common mic in studios and certainly for live use).
- One Audio-Technica AT836; dynamic mic.

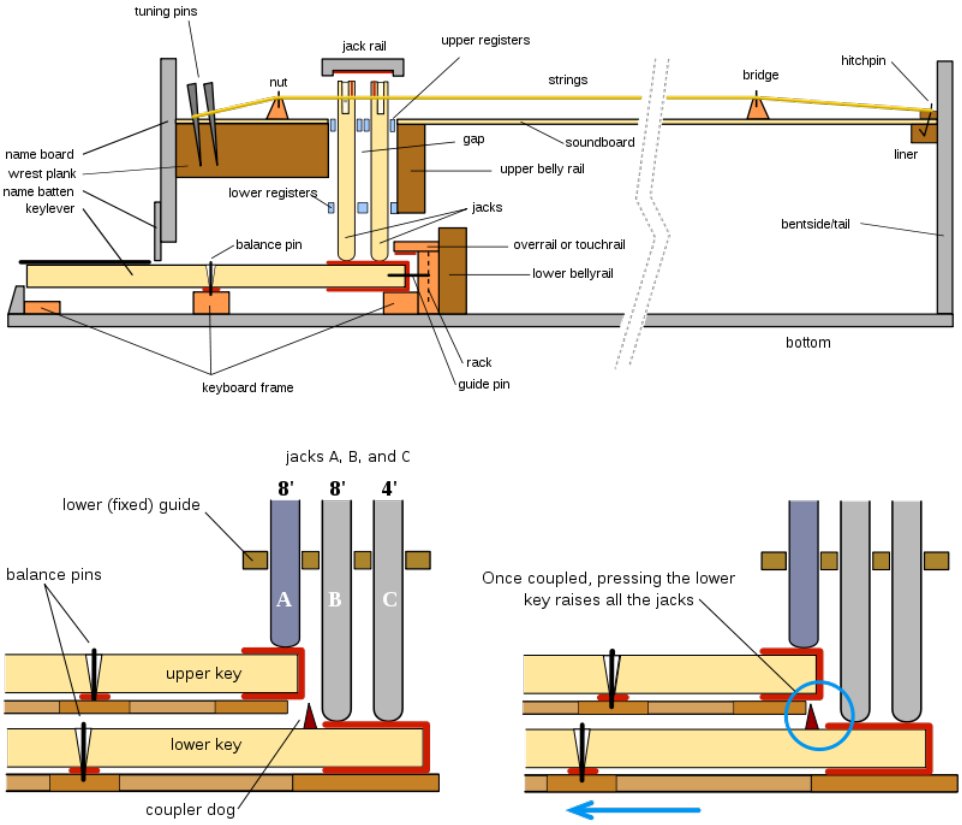

Figure 1.1: An illustration of the mechanics of the harpsichord [wikipedia.com].

The mic's were positioned at different places inside the instrument, as seen on figures 1.2 and 1.3. The reason for using this setup, which yields two mono and one stereo signal with very different characteristics, was to explore if the tested methods would react differently to different signal - and if this extra information could be used to simplify the problem or improve the results.

1.5 Method & Tests

As this is the first attempt on serious work for the author using MATLAB, a challenge during the process was how to work and perform testing systematically. Hence, a method was only developed through the process, and some tests and results have suffered from this. Much of the MATLAB code consists of snippets that were executed using cell mode. The later work, especially that using non-negative matrix factorization (NMF), was done using numbered test cases where each run was tagged automatically with a serial number (using MATLAB's *now* function.). At each test run, a plot and a CSV file containing the most important variable names and their values were saved - and for the most of them a

Figure 1.2: The two Neumann KM84's placed in a 110 degree XY position, and the Shure SM57 below pointing towards the mid front, where the picks are placed.

Figure 1.3: The Audio-Technica AT836 placed in the far end of the instrument, pointing away from the front.

.mat-file as well (with the data). Should there be any inconsistencies in a few file names - they should be identifiable from their serial number (also included on the plots). Thus, all the results and tests presented here are reproducible with the code and data handed-in on the enclosed CD.

Chapter 2

Music & Data Representation

In this chapter we shall give a brief presentation of some of the features and different representations of music which are relevant to this report.

2.1 Music Representation

We will be using the two most common representations of digital music; Music Instrument Digital Interface (MIDI), and music audio signals.

MIDI

MIDI can simply be thought of as digital music notation. Music is represented as MIDI *messages* distributed through sixteen possible MIDI *channels*. A single note is represented by a *Note-On* message, which is followed by a *Note-Off* message; at the end of the note's duration. The Note-On message contains information about the note's pitch, velocity, and possibly other control parameters, such as vibrato or pitch-bend. In a Digital Audio Workstation (DAW), or a MIDI sequencer, the MIDI data is typically visualized in a so-called *piano roll*, as shown in figure 2.1.

Music Audio Signals

A soundwave may be turned into an electrical current, when captured by a microphone. Such a *continuous-time*, analog signal may be turned into a *discrete-time*, digital signal by an *analog-to-digital converter (ADC)*; a process often referred to as *sampling*. The result is a digital audio file. Conversely, when a digital audio file is played back, it is converted back into and electrical current, through a *digital-to-analog converter (DAC)*. This current activates the membranes on a set of speakers, which then push the air - and re-produce the recorded sound.

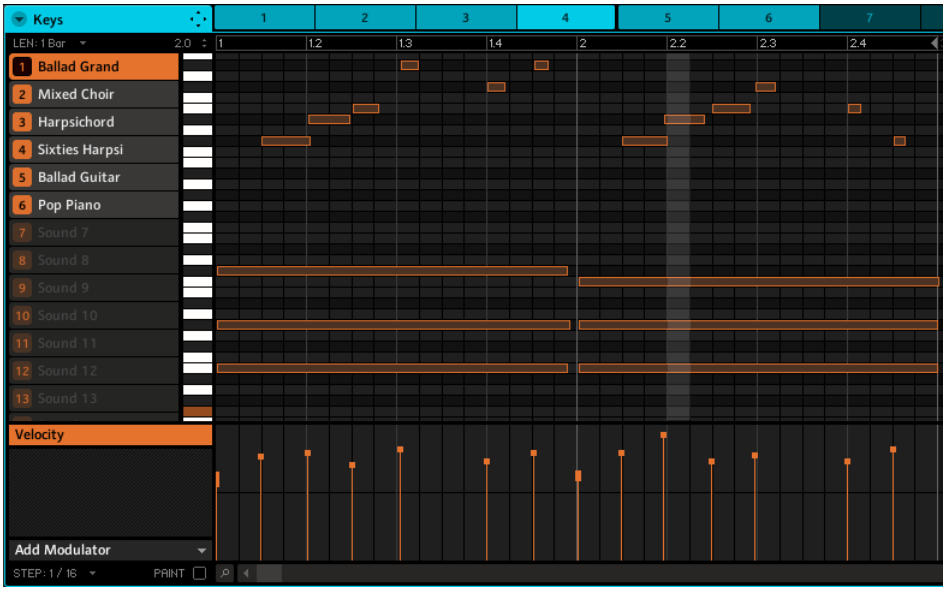

Figure 2.1: Piano Roll visualization of MIDI data.

Discretization & Aliasing

During the ADC process, the audio signal is *discretized* in time and amplitude, as shown on figure 2.2. The voltage of the analog signal is measured, or sampled, at uniform time intervals; called the *sampling period*, T_s. The *sampling frequency* is defined as $f_s = T_s^{-1}$ - typically it is 44.1 kHz (CD quality). The choice of this exact sample rate stems from the fact that, the human ear is capable of detecting frequencies in approximately the range from 20 Hz to 20 kHz. Due to a phenomena called *aliasing*, the sample rate, f_s, must be at least two times the highest frequency present in the signal - in order not to loose information. Formally, for a continuous-time signal *bandlimited* to B Hz: $f_s > 2B$, where $2B$ is commonly known as the *Nyquist frequency* [11, 12] . The problem of aliasing is depicted in figure 2.3. To avoid aliasing an anti-aliasing filter is normally applied as a pre-processing step; essentially a low-pass filter that cuts off all frequencies above the Nyquist frequency.

Each sample, or measurement of the real-valued signal voltage at a given time, is quantized to an integer number representation. On figures 2.2 and 2.3 this would correspond to fitting the second coordinate of the discretized values to some finite number. The maximum size of this integer number determines the resolution on the vertical axis; or voltage measurements. For CD quality a 16 bit integer is used, thus yielding 65,536 possible values.

2.2 The Features of Music

The musical features contained in modern staff notation, as seen on figure 2.4, are: **(a)** Time Signature **(b)** Key Signature & Musical Scales **(c)** Tempo **(d)** Dynamics **(e)** Expression

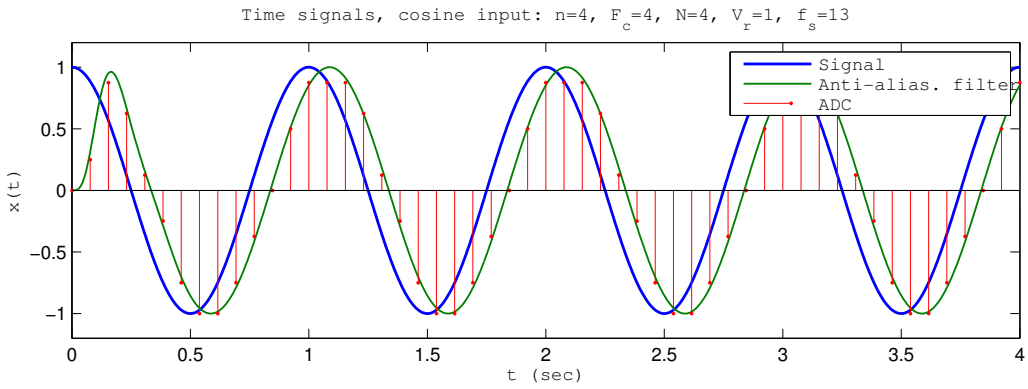

Figure 2.2: Discretizing an audio signal, pre-processed with an anti-aliasing filter (which has no effect in this case).

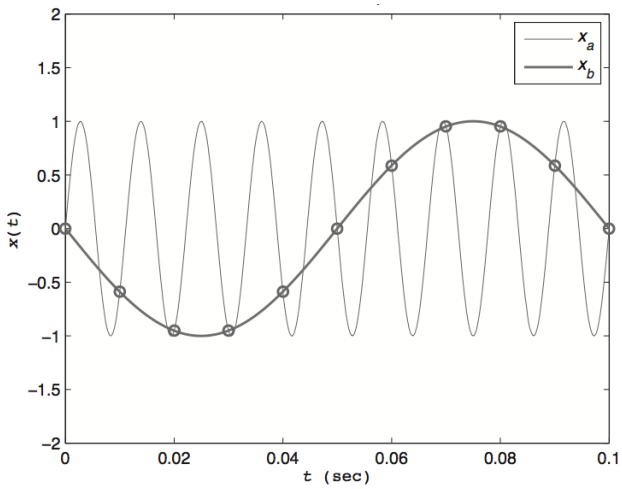

Figure 2.3: An example of aliasing; x_a has a frequency of 90 Hz, but when sampled at rate 100 Hz the sampled result, x_b, gets frequency 10 Hz [12].

(f) Melody **(g)** Rhythm **(h)** Structure (eg. retakes) **(i)** Chords & Harmonies **(j)** Bars & Beats **(k)** Note Pitch & Duration. We will refrain from explaining these, as they are expected to be commonly understood. Of course, all of these features are subject to *artistic interpretation*, and as such an actual performance of the piece, shown in figure 2.4, may very well sound quite different from what Chopin intended.

All of the above features are common in our everyday way of talking about, and describing, music. The same names and concepts are also used in music theory. However, when it comes to music audio signals - or sound - things get a little more difficult. Here we are dealing with the physical phenomena of soundwaves, and how these a processed by the human ear - and the brain. The latter factor is especially intangible and complex to deal with, insofar that features now get more *subjective*. This gives rise to a concept

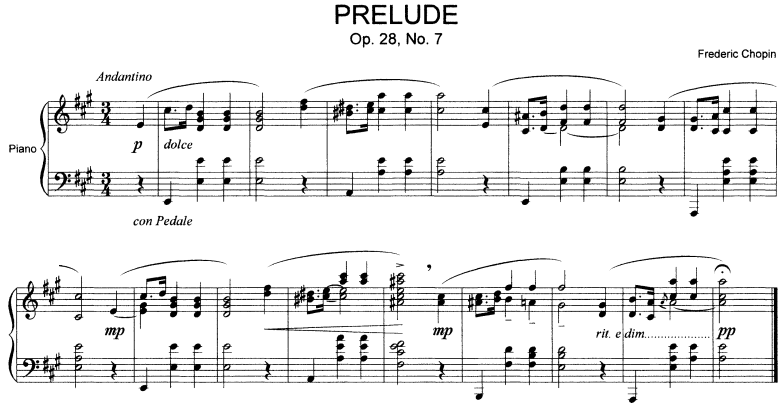

Figure 2.4: Modern staff notation (source: Wikipedia).

known as *psychoacoustics*, which cover features such as timbre, loudness, punch, warmth and localization [1, 10, 11]. Below we shall briefly elaborate on a few of the musical features that are of special interest to this report.

Pitch

It should come as no surprise to anyone, that pitch plays an enormous role in music and sound. The ANSI (1999) definition of pitch says, "that auditory attribute of sound according to which sounds can be ordered on a scale from low to high". What exactly these auditory attributes are, is somewhat more complex to describe than one might think. In this report we shall stick to the definition given in [10]:

> "Frequency is a physical measure of vibrations per second. Pitch is the corresponding perceptual experience of frequency".

For a *periodic signal*, such as a recording of a tone played by a violin, the *frequency* is the reciprocal of the period; eg. 440 Hz for a well-tempered A. Practically all musical instruments produce tones which consist of integer multiples, *partials*, of some fundamental frequency, f_0. The sum of these partials can be expressed as $F(t) = \sum_p a_p \sin(pf_0t + \phi_p)$; where a_p and ϕ_p are the amplitudes and the phases of the P partials, respectively [6, 11].

Timbre

Timbre is a most intangible feature. The ANSI (1999) definition says [10]:

> "Timbre is that attribute of auditory sensation in terms of which a listener can judge that two sounds similarly presented and having the same loudness and pitch are dissimilar. . . . Timbre depends primarily upon the spectrum of the stimulus, but it also depends upon the waveform, the sound pressure,

the frequency location of the spectrum, and the temporal characteristics of the stimulus."

This is the common *definition-by-negation* of timbre, stating merely what timbre is *not*. We can add some more insight to the matter, by relating back to our definition of pitch. Where a tone's *pitch* is the fundamental frequency of its partials (also referred to as *harmonics* or *overtones*), its *timbre* is strongly related to the temporal dynamic evolution of each of its partials; its *spectral* envelope. If a tone consisted only of one fundamental sinusoid, it could be said to sound dull (albeit pure sinusoids make for excellent basses in electronic music). The presence of partials is, in part, what gives a tone its *color* and personality; aka timbre. A visual example is given in figure 2.5.

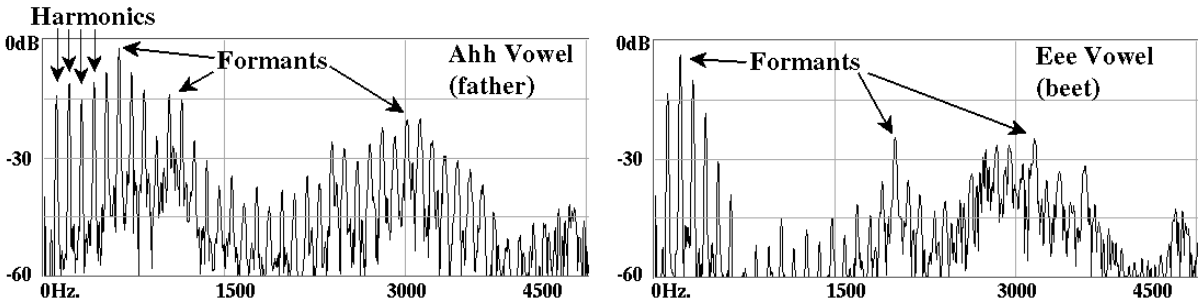

Figure 2.5: Magnitude spectrograms showing the difference between two vowels [8]

Spectral Analysis

To analyze a music audio signal, for instance to find out which notes are present in it, and what their pitches and timbres are, a *spectrum analysis* is typically performed. Visually, this can be done by use of a *spectrogram*, which shows the content, or the *energy*, of different frequencies contained in a signal. A spectrogram is a visualization of a *Fourier transform*, and can take different forms, such as a *magnitude* spectogram (cf. fig. 2.5), or a *time-frequency* spectrogram (cf. fig. 2.6).

Fourier Transform

The Fourier transform is probably *the* most commonly used tool in signal processing. According to the *Fourier Theorem*, any *periodic* phenomena can be modelled as the sum of a - possibly *infinite* - series of sinusoids; with different frequencies, phases and amplitudes [6, 11]. However, periodicity can actually be enforced on *non-periodic* signals, such that the Fourier transform may be applied to them as well. This is very lucky for us, since we are working with finite, and often non-periodic, music audio signals.

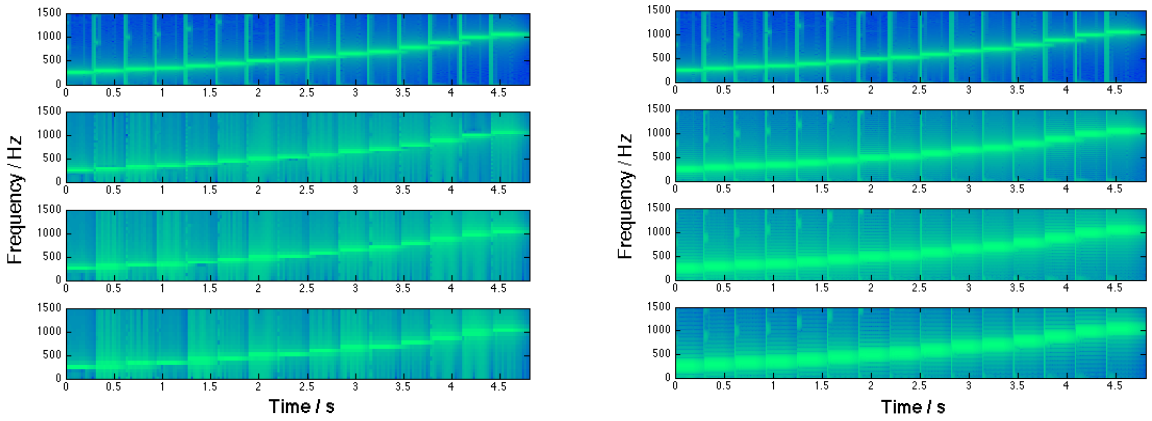

Figure 2.6: Spectrograms showing effects of window and step size; on a recording of a C major scale played with a single sinusoid. **Left:** Decreasing window size (top to bottom), fixed step size. **Right:** Decreasing step size, fixed window size.

For a *continuous* signal, or function, the Fourier transform is defined as:

$$X(f) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi ft} \mathrm{d}t \tag{2.1}$$

$e^{-i2\pi ft}$ is sometimes referred to as the *kernel* function of the Fourier transform, but generally in physics and complex number theory it is called a *phasor*. It relies on *Euler's formula*, $\cos\theta + i\sin\theta = e^{i\theta}$, and can be thought of as a *probe phasor*; a pulse with which the signal, $x(t)$, is probed for its energy at frequency f. This is beautifully done; simply by integrating the probe phasor's *convolution* with $x(t)$, over time. Consequently, the Fourier transform can be said to capture the energy at given frequency in a complex number, $e^{i\theta}$. The size of this number expresses the *energy*, and the imaginary part expresses the *phase*.

Likewise, *The Discrete Fourier Transform (DFT)* is defined as:

$$X(k) = \frac{1}{N} \int_{n=0}^{N-1} x(n) e^{-ik\omega n/N} \mathrm{d}t \tag{2.2}$$

where N is the number of samples, $\omega = 2\pi$, and k indexes the frequencies in the output spectrum [6, 11].

Short-Time Fourier Transform & Windowing

The DFT converts a signal from the time-domain to the frequency-domain. As the DFT can be thought of as taking af "snapshot" of the spectrum at a specific point in time, it alone cannot be used to generate time-frequency spectrograms, such as the ones in figure 2.6. To achieve this, a *sliding window* approach can be used - by applying the DFT to

consecutive windows of the data; that is the *short-time Fourier transform (STFT)*. The size of the window, as well as the size of the steps by which it slides across the data, yield results with different properties. As seen on figure 2.6, there is as trade-off to be considered between the choice of these two sizes; small window sizes give higher temporal resolutions, but decrease the spectral resolution, and vice versa [2]. Lastly, it should be noted that the *shape* of the window, may also be changed depending on the intended application; due to a phenomena known as *spectral leakage* [2]. In this work, we have used MATLAB's *fft* function, which is an implementation of the fast Fourier transform (FFT), and thus runs in lenearithmic time, $O(N * logN)$.

Chapter 3

Spectral Analysis

Although the spectral analysis took the majority of our time, we shall keep this section brief; simply because no results of particular interest where achieved - and to leave more room our next more successful endeavor. The results in this section are not conclusive. Merely, they present some indications and our reasoning for concluding that any relatively simple solution to our problem, will most likely not be found using spectral analysis alone.

3.1 Summing Harmonic Amplitudes

Klapuri has done extensive work on the problem of pitch-detection, and in [7] he presents an interesting and intuitive solution, with which we have performed some experiments. The method is based the summing of harmonic amplitudes, using the intuition that the most predominant fundamental frequency (f0) in a sample will most likely yield the greater sum, with respect to the energy of its harmonic partials. Thus, the method relies heavily on consistent detection of the predominant f0 salience. Once that is found, the energy of its partials is subtracted from the spectrum, and the process is continued iteratively on the residual. All in all a very nice model - which does not work for our data; as seen in figure 3.1. Even with monophonic inputs it fails to detect the predominant f0. Given this result we dropped further work with this method.

3.2 Heuristic Bottom-Up Approach

With offset in Klapuri's model, we tested an idea, where we would try to consistently detect the notes in the lowest octave correctly - only from learning their thresholds. This is theoretically possible, as we know that the notes in the lowest octave would not have any overlapping harmonics. Hence, after detecting a f0, our idea was to perform a similar subtractive method as described by Klapuri, only based solely on the fundamental frequencies of each note - starting from the bottom: detecting and subtracting all the way up to the highest note. This way we would use our prior knowledge of the instrument, ie. by learning the correct thresholds from our recordings of every single note. Here, it should be noted

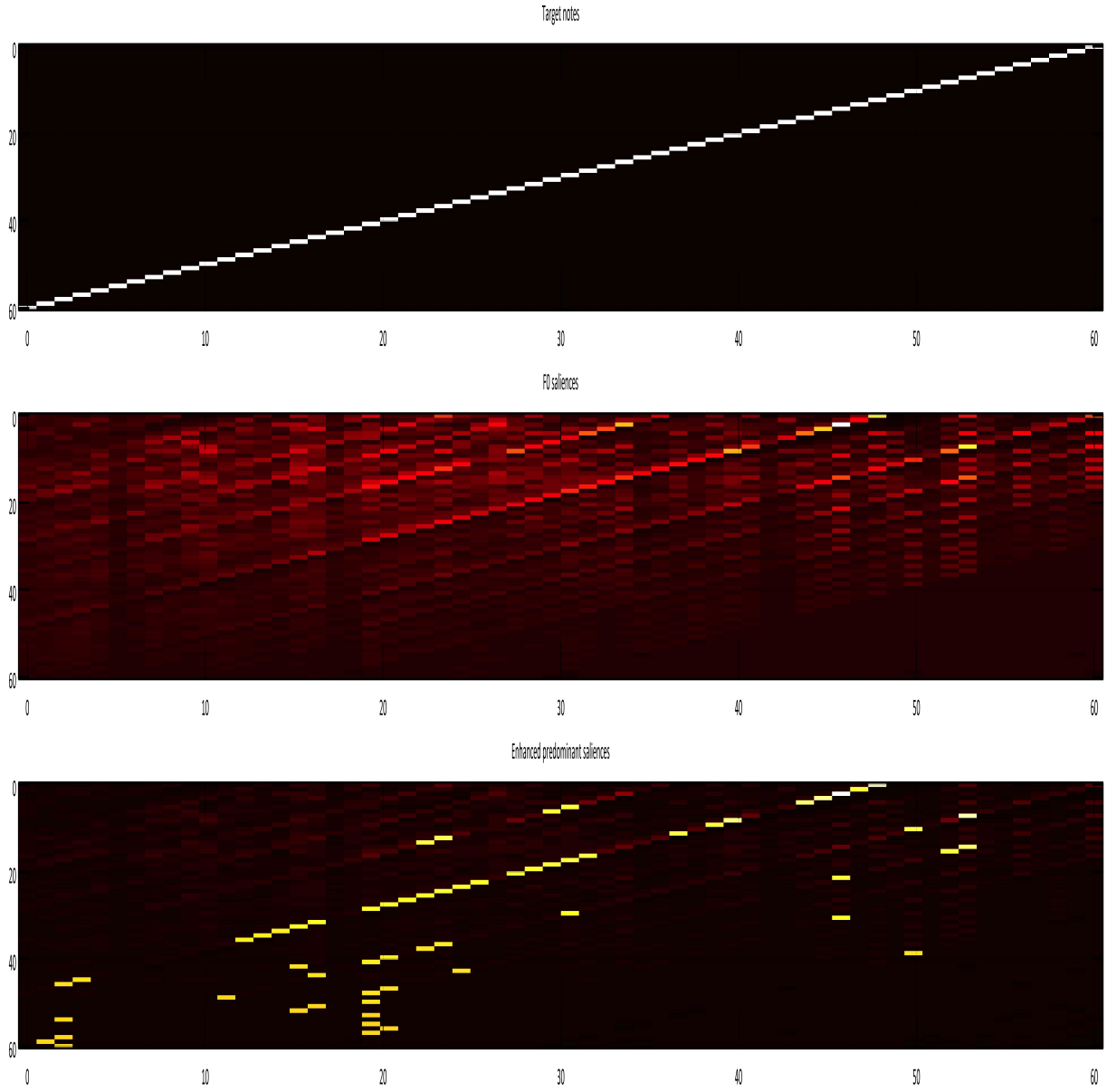

Figure 3.1: Klapuri's method on a monophonic chromatic scale played on the harpsichord. The topmost plot shows the target notes, the middle one the estimated f0 saliences, and on the bottom plot the predominant f0 saliences have been enhanced

that unlike eg. a piano, the harpsichord has no dynamic range - it is fixed. Of course, given that it is a simple mechanic instrument, one should not expect it to produce exactly similar sounding notes for each onset. Still, it could be a useful prior for a solution to the problem. Furthermore, to improve the results even more, we experimented with extracting the harmonic envelopes for each harmonic of each note. This would help us to perform the subtraction step even more precisely than with Klapuri's method, which uses no such priors. However, the idea failed already at detecting the onsets of the lowest octave. The reason is to be found in what we have name the zero-padding problem, depicted in figure 3.2. For a real-time algorithm we need to keep the frame size very small, forcing us to do zero-padding on the incoming signals in order to get a useful frequency resolution with the FFT. This causes an implicit interpolation between the energies at the actual frequency bins, thus distorting the energy measurements at the frequencies in between - rendering our method useless for small frame sizes.

3.3 Harmonic Fingerprints

The last attempt we made on a solution with spectral analysis also tried to use the priors at hand. It is based on the simple idea, that the only non-unique frequencies present in a signal containing one instrument playing in a known tuning and scale are harmonics of the frequencies in that musical scale. With a frame size of 16384 samples this leaves over 8000 potentially unique harmonics - given that they are measured correctly. For each single fundamental this leaves quite a few possible unique harmonic fingerprints that we would then try to learn from the data beforehand. As adding signals in the time domain corresponds to adding their spectra, again theoretically, we should be able to determine a given f0 from one single unique harmonic in a signal without too much noise - even in polyphonic material. Figure 3.3 depicts the situation for a major chord with three notes present at the same time. Alas, once again we hit a brick wall with the imprecision of the FFT at small window sizes. However, have not given up completely on this idea - it just does not seem a probable solution for a real-time system.

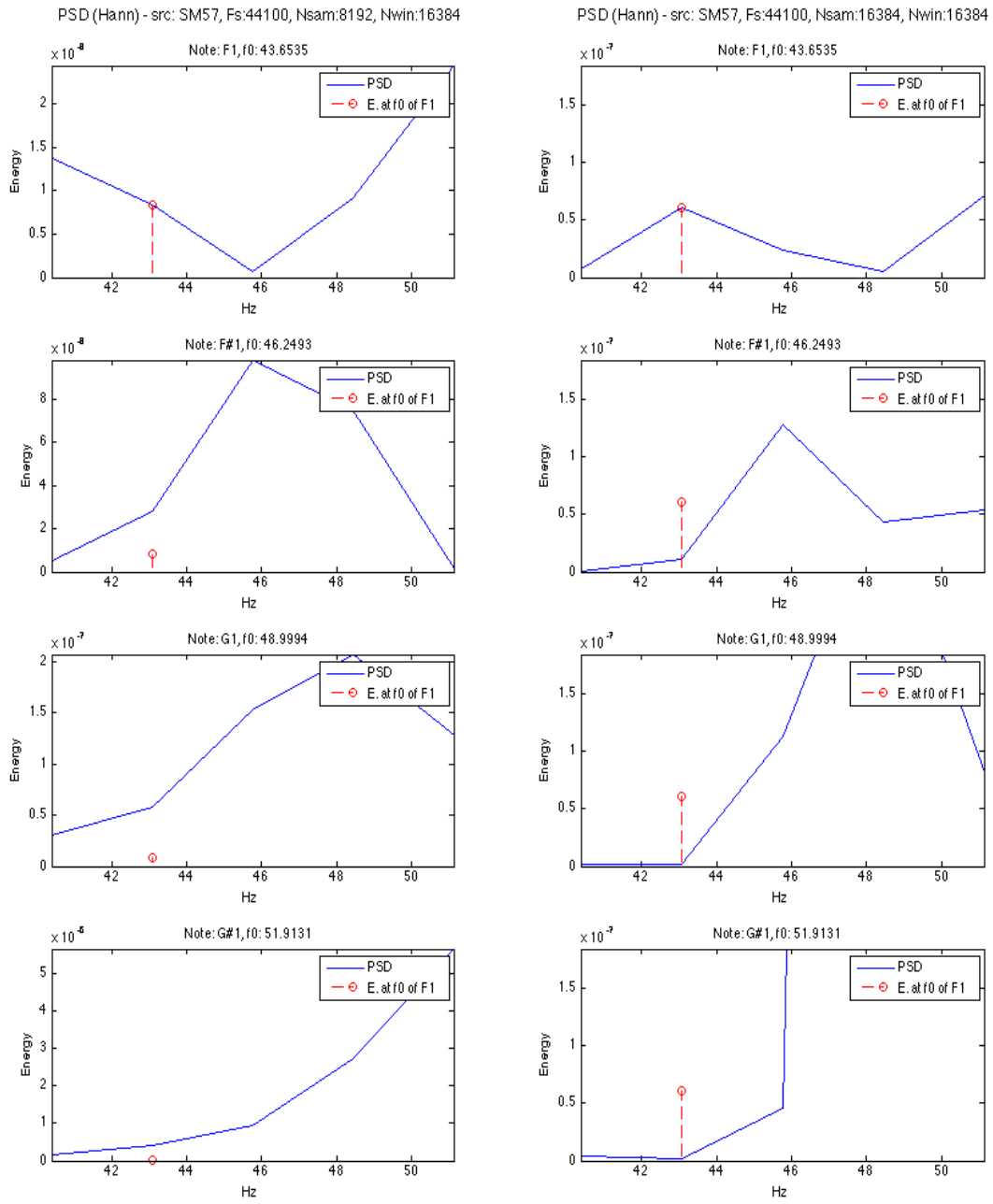

Figure 3.2: Klapuri's method on a monophonic chromatic scale played on the harpsichord. The topmost plot shows the target notes, the middle one the estimated f0 saliences, and on the bottom plot the predominant f0 saliences have been enhanced

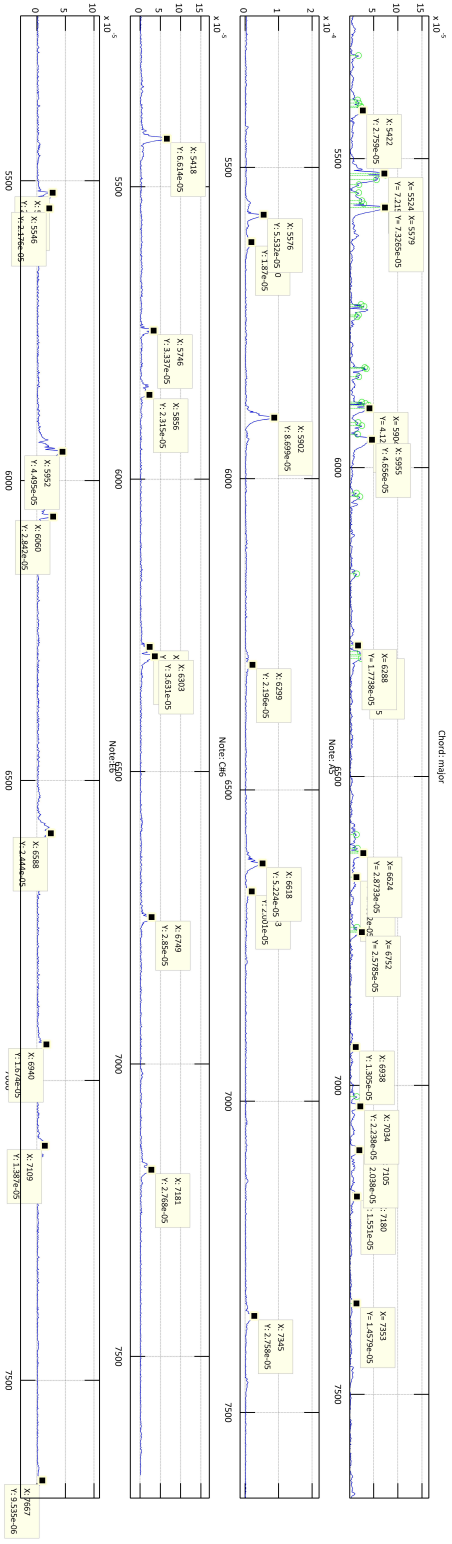

Figure 3.3:

Chapter 4

Non-Negative Matrix Factorization for Pitch-Detection

In this chapter we will start by giving a general introduction to non-negative matrix factorization (NMF), hereafter we will describe the specific method proposed by Dessein et al. [4] on which our implementation is based. Finally, we present our own solution and results.

4.1 Non-Negative Matrix Factorization

Non-negative matrix factorization (NMF) belongs to a category of techniques, that can be categorized as *constrained matrix factorizations* aka *latent variable decompositions*. It was first introduced by Lee and Seung, in an article in Nature in 1999 [9], and has since been extended and modified in various different ways.

Why Non-Negativity?

Many types of signals, such as music audio signals, are inherently non-negative. If we for instance represented an audio sample as amplitude spectra, it would clearly never be negative. The same thing can be said for pixel intensities, occurrence counts, and many more. In a biological context, or a neural network implementation, neither synaptic strengths, or the firing rates of neurons are ever negative [9]. Hence, non-negativity is a very reasonable constraint (and an effective one too). Additionally, it implicitly endows an additive model of the data. This agrees very well with the intuitive notion that eg. a music recording is the sum of a number of sound sources [13]. Figure 4.1 illustrates well, how such "parts-based" representations may look for a data set of facial images.

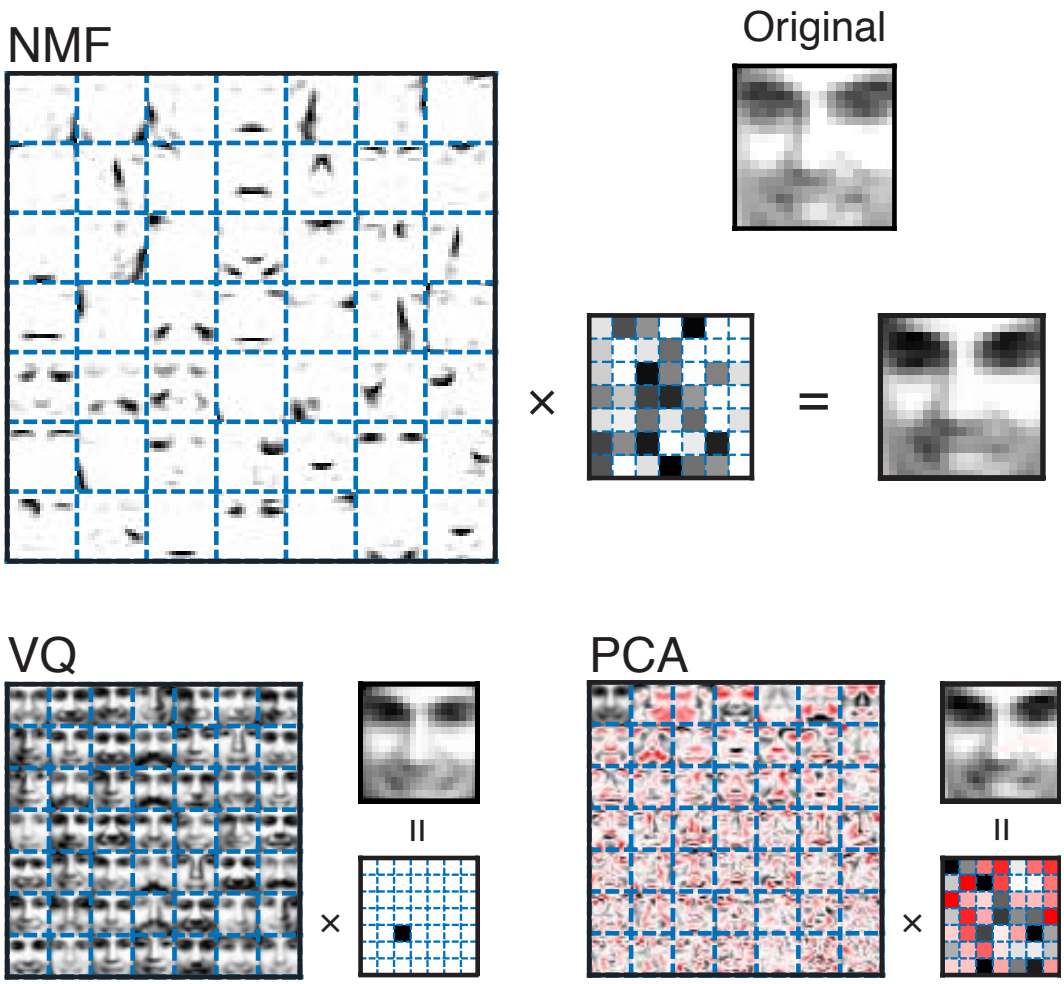

Figure 4.1: A figure from Lee & Seung's original article on NMF; showing the parts-based (additive) representation learned by NMF, contrasted with the holistic representations obtained from PCA or vector quantization (PQ) [9]. The bigger squares, ie. matrices, are often refereed to as *dictionaries*.

Problem Formalization

Given a set of observation vectors $\mathbf{x_t}, 1 \leqslant t \leqslant T$, approximated by this linear, additive mixing model

$$\mathbf{x}_t \approx \mathbf{A}\mathbf{s}_t = \sum_n \mathbf{a}_n s_n \tag{4.1}$$

the goal is to approximate $\mathbf{A}$ and $\mathbf{S}$, such that

$$\mathbf{X} \approx \mathbf{AS} \mid \mathbf{A}, \mathbf{S} \geq 0 \tag{4.2}$$

where $\mathbf{X} \in \mathbb{R}^{P \times T}$ is the *observation matrix* for P sources at T times, $\mathbf{A} \in \mathbb{R}_+^{P \times N}$ is a *mixing matrix* containing the coefficients for N sources to P observations, $\mathbf{S} \in \mathbb{R}_+^{N \times T}$ is a *source*

matrix of T samples of N mixtures. Note that, the source are not assumed to be iid - only non-negative. It is convenient to notice the symmetry,

$$\mathbf{X}^T \approx \mathbf{S}^T \mathbf{A}^T \tag{4.3}$$

meaning that "source" and "mixture" are somewhat arbitrary concepts in this context.

A vast number of approaches and strategies exist for solving the NMF problem. Here we shall present just one simple solution; using gradient descent.

An NMF Algorithm

As our *divergence measure* we will use the sum-of-squares aka the *Euclidian distance*[3]:

$$J_E = \frac{1}{2} \|\mathbf{X} - \mathbf{AS}\|^2 = \frac{1}{2} \sum_{pt} (x_{pt} - [\mathbf{AS}]_{pt})^2 = \frac{1}{2} Tr((\mathbf{X} - \mathbf{AS})^T (\mathbf{X} - \mathbf{AS})) \tag{4.4}$$

where $[\mathbf{M}]_{pt}$ denotes the (p,t)-th element of the matrix $\mathbf{M}$, and $Tr(\cdot)$ is the sum of the trace. The right-most expression for J_E is included because we will later utilize the fact that, the squared norm equals the sum of the trace. Our gradient descent update rule should be

$$s_{nt} := s_{nt} - \eta_{nt} \frac{\partial J_E}{\partial s_{nt}} \tag{4.5}$$

the learning rate parameter η_{nt} may have individual values for each pair (n,t).

Differentiating (4.4) with respect to $\partial \mathbf{S}$, and expressing it as the sum of the trace, gives us

$$\begin{aligned} \partial J_E &= -Tr((\mathbf{X} - \mathbf{AS})^T \mathbf{A} \partial \mathbf{S}) && (4.6)\\ &= -Tr((\mathbf{A}^T\mathbf{X} - \mathbf{A}^T\mathbf{AS})^T \partial S) && (4.7)\\ &= -\sum_{nt} [\mathbf{A}^T\mathbf{X} - \mathbf{A}^T\mathbf{AS}]_{nt} \partial s_{nt} && (4.8)\\ & && (4.9) \end{aligned}$$

leading to,

$$\frac{\partial J_E}{\partial s_{nt}} = [\mathbf{A}^T\mathbf{X} - \mathbf{A}^T\mathbf{AS}]_{nt} = -([\mathbf{A}^T\mathbf{X}]_{nt} - [\mathbf{A}^T\mathbf{AS}]_{nt}). \tag{4.10}$$

And finally, by substituting (4.10) into (4.5) we obtain the following update rules,

$$\begin{aligned} s_{nt} &:= [s_{nt} + \eta_{nt}([\mathbf{A}^T\mathbf{X}]_{nt} - [\mathbf{A}^T\mathbf{AS}]_{nt})]_+ && (4.11)\\ a_{pn} &:= [a_{pn} + \eta_{pn}([\mathbf{XS}^T]_{pn} - [\mathbf{ASS}^T]_{pn})]_+ && (4.12) \end{aligned}$$

due to the symmetry property mentioned in (4.3). The non-negativity is guaranteed by $[\cdot]_+$, which denotes the rectification function; $[s]_+ = max(0, s)$.

Given these *two* update rules, we can now apply *alternating* gradient descent [3]. This method treats the optimization as two subproblems; hence the two update rules. The idea is, on each iteration to solve the problem for A, while keeping S fixed, and vice versa; as described below [13]:

Definition 1

Alternating Gradient Descent:

```
Repeat until convergence {
```

$$\mathbf{A} := arg\min_{A \geq 0} J_E(\mathbf{X}; \mathbf{A}, \mathbf{S})$$

$$\mathbf{S} := arg\min_{S \geq 0} J_E(\mathbf{X}; \mathbf{A}, \mathbf{S})$$

```
}
```

This method has several advantages to regular gradient descent. Firstly, it can be shown to converge in linear time to a local solution, under certain conditions. Next, it is often better at avoiding local optima, and offers the interesting option of choosing different cost functions, or even algorithms, for the two subproblems. Lastly, the joint NMF problem is non-convex, but with alternating gradient descent we may choose cost functions that are convex for the individual subproblems [13].

For the NMF algorithm to work, the matrices **A** & **S** must be initialized with random non-negative values. Even still, this may be done in a more or less effective way. Various methods have been suggested for initialization, that will lead to faster convergence [13]. Moreover, it should be mentioned that the solutions yielded by NMF are not unique. Thus, it may be desirable to compare results given by different initializations of the algorithm.

4.2 NMF for Pitch-Detection

Non-negative matrix factorization (NMF) has been widely used in music applications. In the MIR world, it has been applied to eg. music transcription, classification, feature extraction, and source separation - with quite promising results. Sha and Saul have used NMF to estimate fundamental frequencies of simultaneous acoustic sources; Smaragdis and Brown applied it to polyphonic pitch detection, by learning the spectral profiles of each note; Wang and Plumbley used NMF to decompose audio signals into components,

that are grouped to form individual audio sources; and the list continues.

In this thesis, we use the method given by Dessein et al. in 2010 [4], which is based on a novel use of the β-divergence, and presents a simple and computationally efficient update rule, which can be applied in real-time.

4.2.1 NMF with the Beta-divergence

The Beta-divergence is quite different from the Euclidean distance, insofar that it does not measure distances, as such; it is asymmetrical and generally does not satisfy the triangle inequality. It belongs to a parametric family of distortion functions, and is defined as follows:

Definition 2

For any $\beta \in \Re$ and any points $x, y \in \Re_{++}$, the β-divergence from x to y is

$$d_\beta(x|y) = \frac{1}{\beta(\beta-1)}\left(x^\beta + (\beta-1)y^\beta - \beta x y^{\beta-1}\right) \tag{4.13}$$

Some special cases arise when taking the limits in (4.13) $\beta = 0$ and $\beta = 1$, which lead to the well-known Itakura-Saito and Kullback-Liebler divergences, respectively. For $\beta = 2$ the β-divergence specializes to the Euclidean distance (4.4):

$$d_{\beta=2}(x|y) = d_E(x|y) = \frac{1}{2}(x-y)^2 \tag{4.14}$$

An important property, relevant to the current implementation, is the scaling property of the β-divergence:

$$d_\beta(\lambda x|\lambda y) = \lambda^\beta d_\beta(x|y) \tag{4.15}$$

where $\lambda \in \Re_{++}$.

From the scalar divergence in (4.13) a matrix divergence can be constructed by summing the element-wise divergences. The problem can then be described as minimizing this divergence function:

$$\mathcal{D}_\beta(\mathbf{V}|\mathbf{WH}) = \sum_{i,\,j} d_\beta(v_{ij} \,|\, [\mathbf{WH}]_{ij}) \tag{4.16}$$

which again is exactly equal to (4.4) when $\beta = 2$.

4.2.2 Non-Negative Decomposition with the Beta-divergence

In the method described in [4] the actual pitch-detection is performed on-line after first having learned a dictionary of note templates using the standard NMF with Euclidean distance; off-line. This is done by performing a non-negative decomposition with the β-divergence using gradient descent. If $\mathbf{W}$ is the learned dictionary (or basis) onto which we wish to decompose (project) the incoming music signal, the problem is equivalent to minimizing the following cost function:

$$\mathcal{D}_\beta(\mathbf{v}|\mathbf{Wh}) = \sum_i d_\beta(v_i \,|\, [\mathbf{Wh}]_i) \tag{4.17}$$

As $\mathbf{W}$ is fixed, the algorithm boils down updating $\mathbf{v}$ iteratively until convergence, using this update rule:

$$\mathbf{h} \leftarrow \mathbf{h} \otimes \frac{\mathbf{W}^T\big((\mathbf{Wh})^{\cdot\beta-2} \otimes \mathbf{v}\big)}{\mathbf{W}^T(\mathbf{Wh})^{\cdot\beta-1}} \tag{4.18}$$

which can be rewritten as

$$\mathbf{h} \leftarrow \mathbf{h} \otimes \frac{\big(\mathbf{W} \otimes (\mathbf{ve}^T)\big)^T(\mathbf{Wh})^{\cdot\beta-2}}{\mathbf{W}^T(\mathbf{Wh})^{\cdot\beta-1}} \tag{4.19}$$

where $\mathbf{e}$ is a vector of ones. Given that the dictionary $\mathbf{W}$ is fixed, and that $\mathbf{v}$ is the incoming signal, this formulation reduces the computational cost, as the term $(\mathbf{W} \otimes (\mathbf{ve})^T)^T$ need only be computed once per input frame; yielding the method very suitable for real-time decomposition.

Due to the scaling property in (4.15), in this context, the β-parameter can be interpreted as controlling the relative weight given to high and low energy frequencies, such that $\beta > 0$ emphasize higher energy components, and conversely for $\beta < 0$. A means to avoiding the common f0 detection problem of octave and harmonic errors, is then to set this parameter to a suitable value; Dessein et al. propose a value of 0.5, which also according to our own experiments is a good choice.

4.2.3 Overview of the System

In MATLAB the whole thing was implemented in less than four hours - which stand in dire contrast to our very time-consuming and pretty fruitless tampering with spectral analysis methods. An overview of the system is displayed in figure 4.2, and the MATLAB code for the decomposition looks like this[1]:

[1] file: bs_test0301_nmf.m

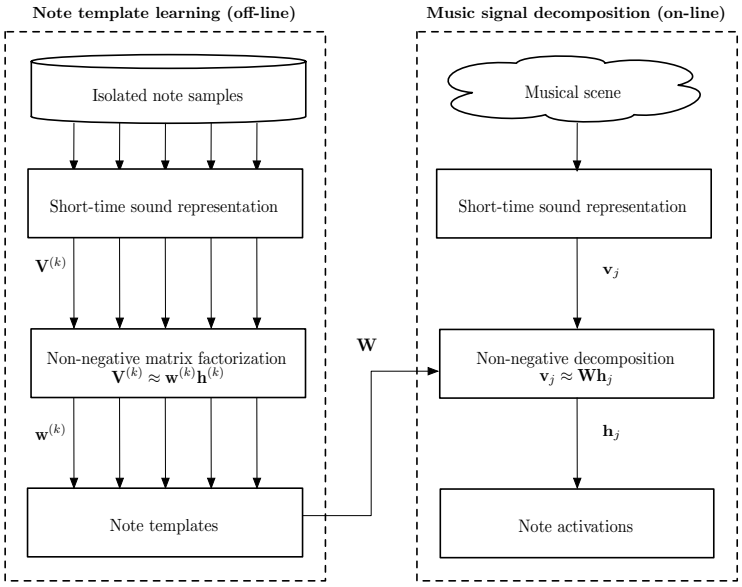

Figure 4.2: Schematic view of the system as presented in [4]

```
init = randn(length(NOTES),1);
while ~isDone(hmfr) && frame < N_frames
  v = step(hmfr);               % read audio
  step(hap, v);                 % play audio

  v = 2*abs(fft(v.*win,N_win))/N_sample;  % transform to frequency domain
  v = v(1:N_win/2+1);
  % v = bs_normalize(v,'peak');       % max normalize

  % Apply update rule for Beta-divergence as described by Dessein et al.
  h = init;                   % initialize note activation vector
  K = (W.*(v*ones(length(NOTES),1)'))'; % the constant term
  for i = 1:30
    h = h .* ((K*(W*h).^(beta-2)) ./ (W'*(W*h).^(beta-1))); % update h
  end
  PROLL(:,frame) = h;
  frame = 1+frame;
end
```

where **W** is the dictionary matrix of note templates learned beforehand using Graham Grindlay's NMFlib [5]. The dictionary is obtained by feeding spectrograms of recordings of each note on the harpsichord to the NMF algorithm, keeping the matrix **W** while disregarding the other matrix from the factorization, **H**. As can be seen above, for this first version of the algorithm we did not even bother to check for convergence; and it stil worked like a charm. In fact, from our tests, the algorithm is seen to converge very fast, and usable results typically come after only about five iterations, see figure 4.3.

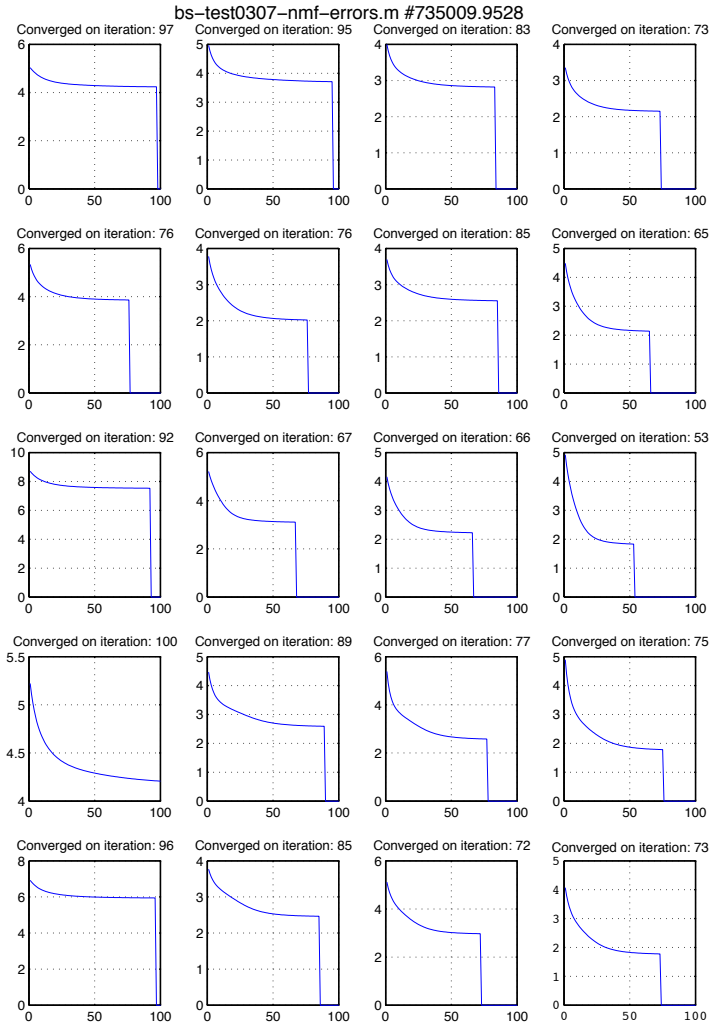

Figure 4.3: A test visualizing the convergence time if the algorithm.

4.3 Tests & Analysis

The amount of testing we performed for this method are too many to fully cover in this report. They are all included on the CD, and the test cases have numbers 0301 and up. We shall instead start by summing up a few tendencies we have observed.

- A suitable value for β is 0.5, as suggested in [4]. See test 0304.
- As with spectral analysis, the low notes seem to be more difficult to detect. That is, the lower range of the notes look significantly more noisy than the mid to high ranges. See figure 4.4 and tests 0302-0304.
- Overall the two condenser microphones, Neumann KM84, seem to provide less useful data. One could speculate whether this is connected to their general ambient characteristic; as opposed to the SM57 and the AT836 which are both dynamic mic's, and thus take in less of the room ambience and (possibly) noisy surroundings; one could also describe the data from them as sounding more diffused.
- The effects of the random initializations are *very* detectable, as the algorithm can produce quite different results on consecutive runs. For this reason we seeded MATLAB's random generator on each test, to yield more comparable results.
- Setting a threshold on the dictionary learning error seems to have no detectable influence. Similarly, for the matrix decomposition, which generally converges very fast - yielding very string results with max. iteration set to 15.
- No noticeable difference in results was observed between 44.1 kHz and 22 kHz sample frequencies.
- With long frame sizes, eg. 186 ms, all the mic's seem to give good results. However, severe ambiguities are still seen, and noise (observed as wrong note detected with low energy) is present when no threshold filtering is applied.
- For short frame sizes, the unfiltered piano-rolls from single microphones look *very* noisy. In fact, it seems implausible, in our context, that they would be very useful.
- Changing the hop-size for the spectrograms that are fed to the dictionary NMF algorithm, does not seem to be very important for the learned dictionaries.

4.3.1 Squaring Out the Noise

In [4] Dessein et al. use thresholds to filter out false note detections, and mention the need for doing more post-processing of the activaion vectors in order to get better results. As we can see in figure 4.5, the results from a single source - using a short frame size - can look rather messy. However, we have applied a very simple, but quite effective method for cleaning it up, so to speak: the element-wise power of two on the activation vector, $\mathbf{h}^{\cdot 2}$

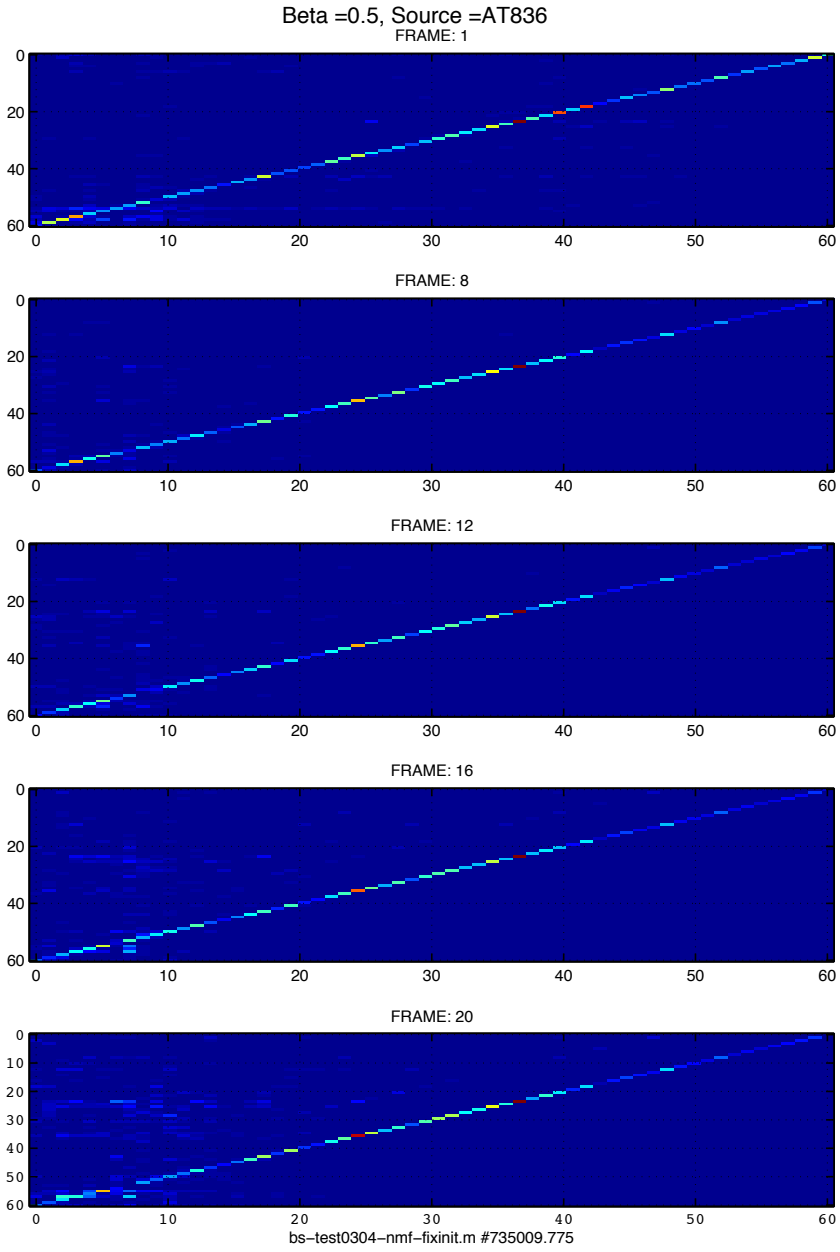

Figure 4.4: Unfiltered note activations, frame by frame, yielding a pianoroll. The input is a chromatic scale going upwards, covering the entire range (on the plot observed as the line going from the lower left corner to the upper right).

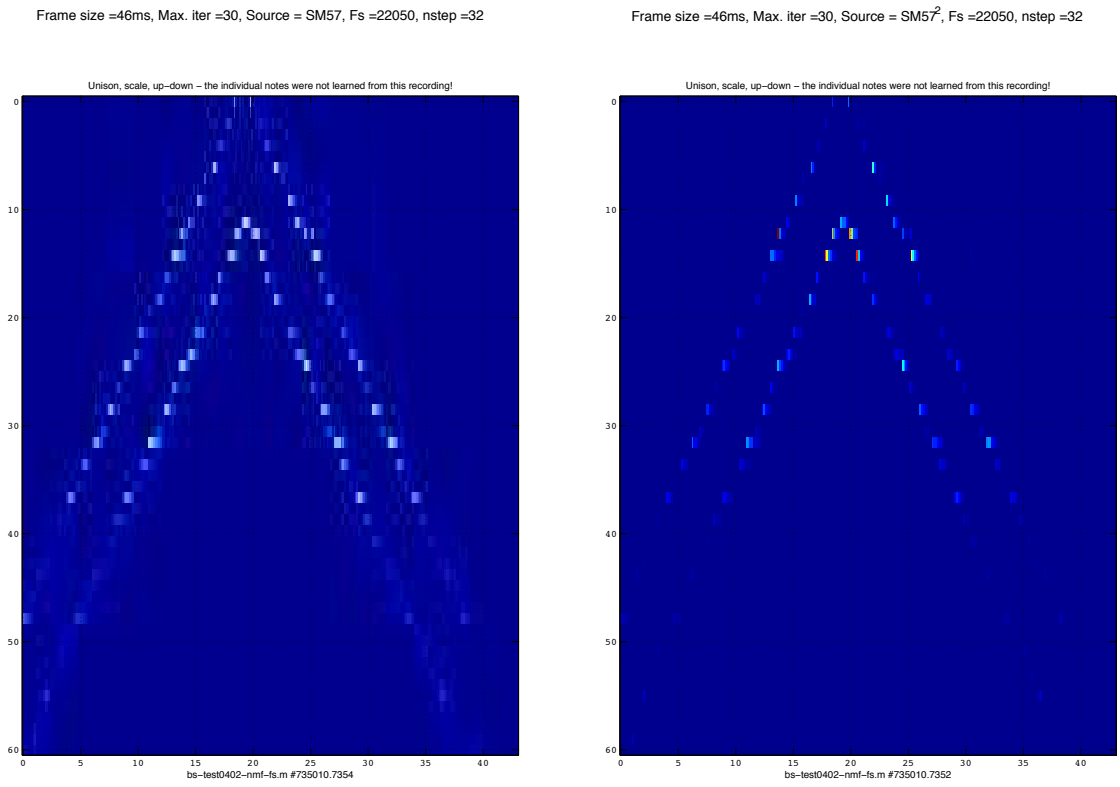

Figure 4.5: Squaring out the noise on the SM57

Figures 4.5 and 4.6 show the unprocessed outputs from the algorithm on a fast up-down scale played in unison. As otherwise seen so often with pitch-detection, we observe that there does not seem to be any clear octave errors. Squaring out the noise this way seems to work quite well, which we shall also see in the following. Since the note tails typically contain less energy, there is, quite logically, at tendency towards note durations being shortened by this method. However, in many applications, such as our own, the durations do not matter all that much; we are more concerned about the onsets.

4.3.2 Improving Note Detection by Using Two Microphones

Following the same line of thinking, instead of squaring the activation vector from one signal, multiplying the vectors from two *different* signals is an obvious idea; $\mathbf{h_1} \otimes \mathbf{h_2}$ The logic being, that whatever the two signals, or vectors, agree on is more likely to be correct; and will also be enhanced by the element-wise multiplication. Conversely, if the two signals (microphones, placements and whatever influences the signal) are truly different, then the activations appearing as noise on the pianorolls are likely to be radically different between the two - yielding a bunch of multiplications between one number which is not zero (the noise) and one that is; $0 \approx a*b$, for $a > 0$ and $b \approx 0$. Now, let us see how the idea works in practice.

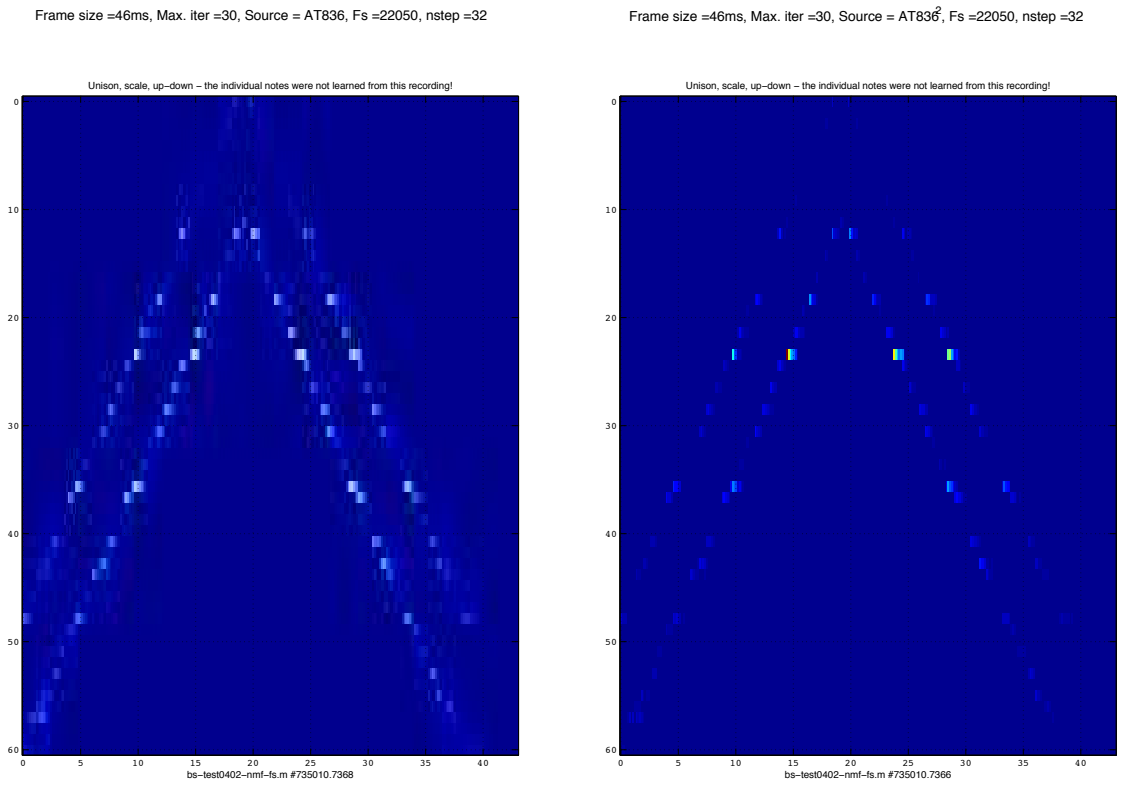

Figure 4.6: Squaring out the noise on the AT836

Figure 4.7 show the results of a transcription performed in real-time on 30 seconds of a piece by J.S. Bach, cf. test 0501. The frame size is 23 ms and the sampling frequency is 22 kHz - no post-processing was applied. The test ran under MATLAB on a Macbook Air, 1.8 GHz Intel Core i7, with 4 GB RAM. As we can clearly see, the music is in there, but it is really crowded by noise. It should be noted though, that these plots display all detections very clearly, as to enhance the noise. The plots in figure 4.8 present a *much* more clear and readable result; obtained by simply squaring out the noise. Most of the noise has canceled out, and it actually looks like music. The SM57 signal on the left generally looks more noisy, but the AT836 signal has more false or missing note-detections and many durations look too short. In figure 4.9 we have kept the right-hand side plot from figure 4.8, of the AT836 squared, and compared it to the pianoroll obtained from multiplying the two individual sources' activation vectors on each frame. At first glance the two plots do not look all that different, but upon further investigation it seems that the multiplied note activations yield a better result; as pointed out with the ellipses on the plots. The multiplied activation vectors yield three very weak false notes detections, and miss the low G note, as highlighted. The $AT836^{.2}$ generally displays too short durations, but also misses some musically important details, such as falsely detecting a note trill in the last bar; it too misses the low G note.

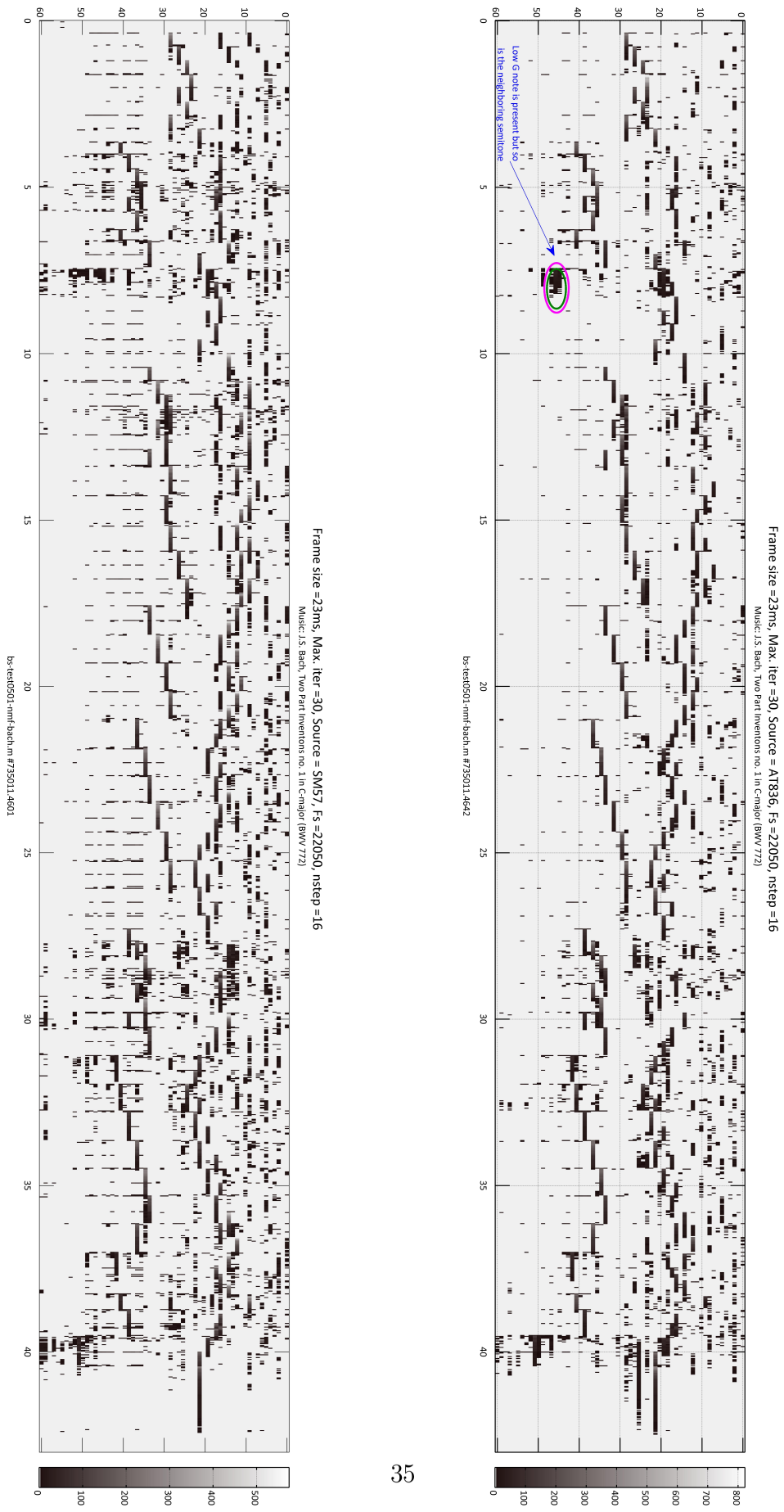

Figure 4.7: Pianoroll representaions of unprocessed activation vectors from two sources

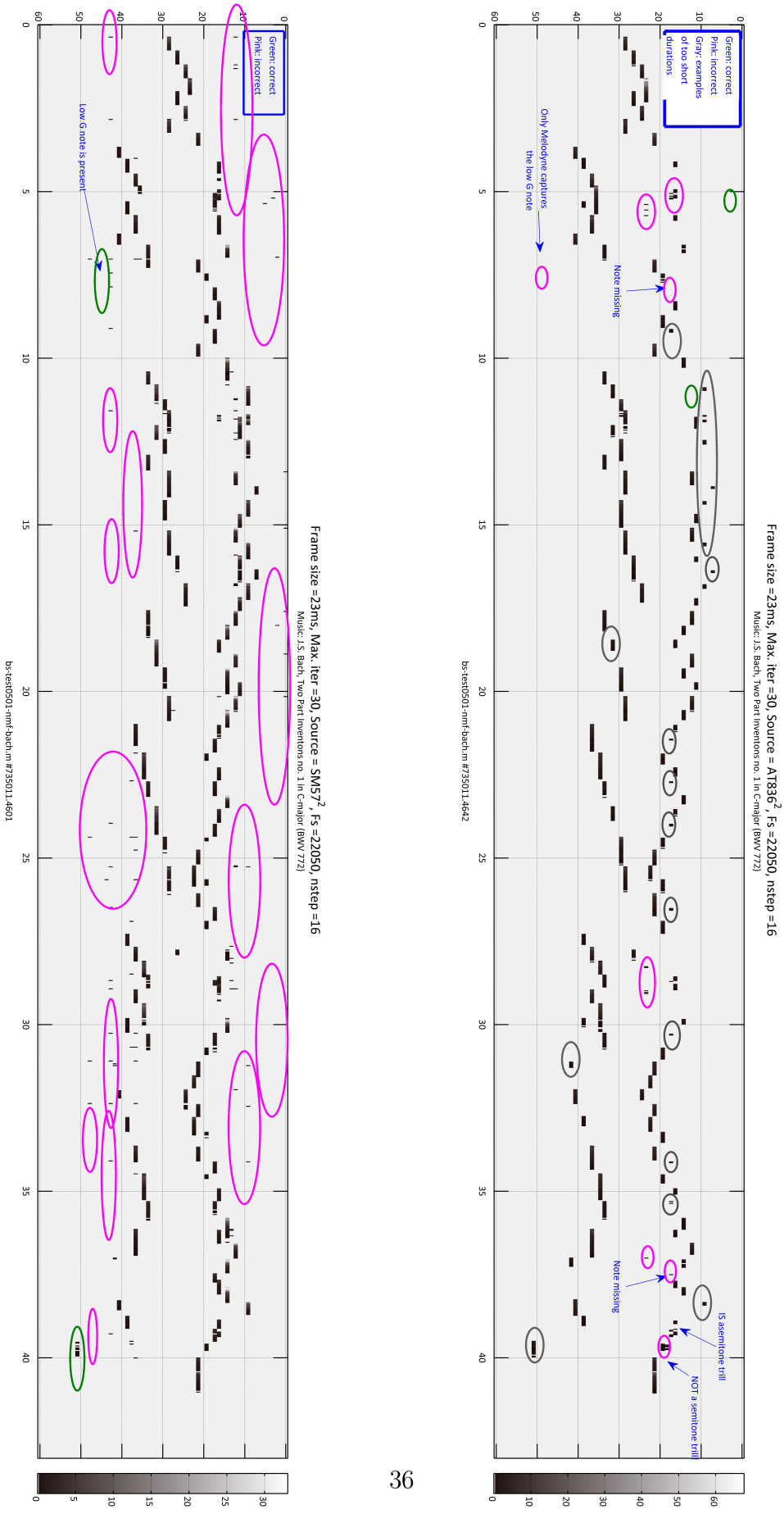

Figure 4.8: Pianoroll representaions of unprocessed squared activation vectors from two sources

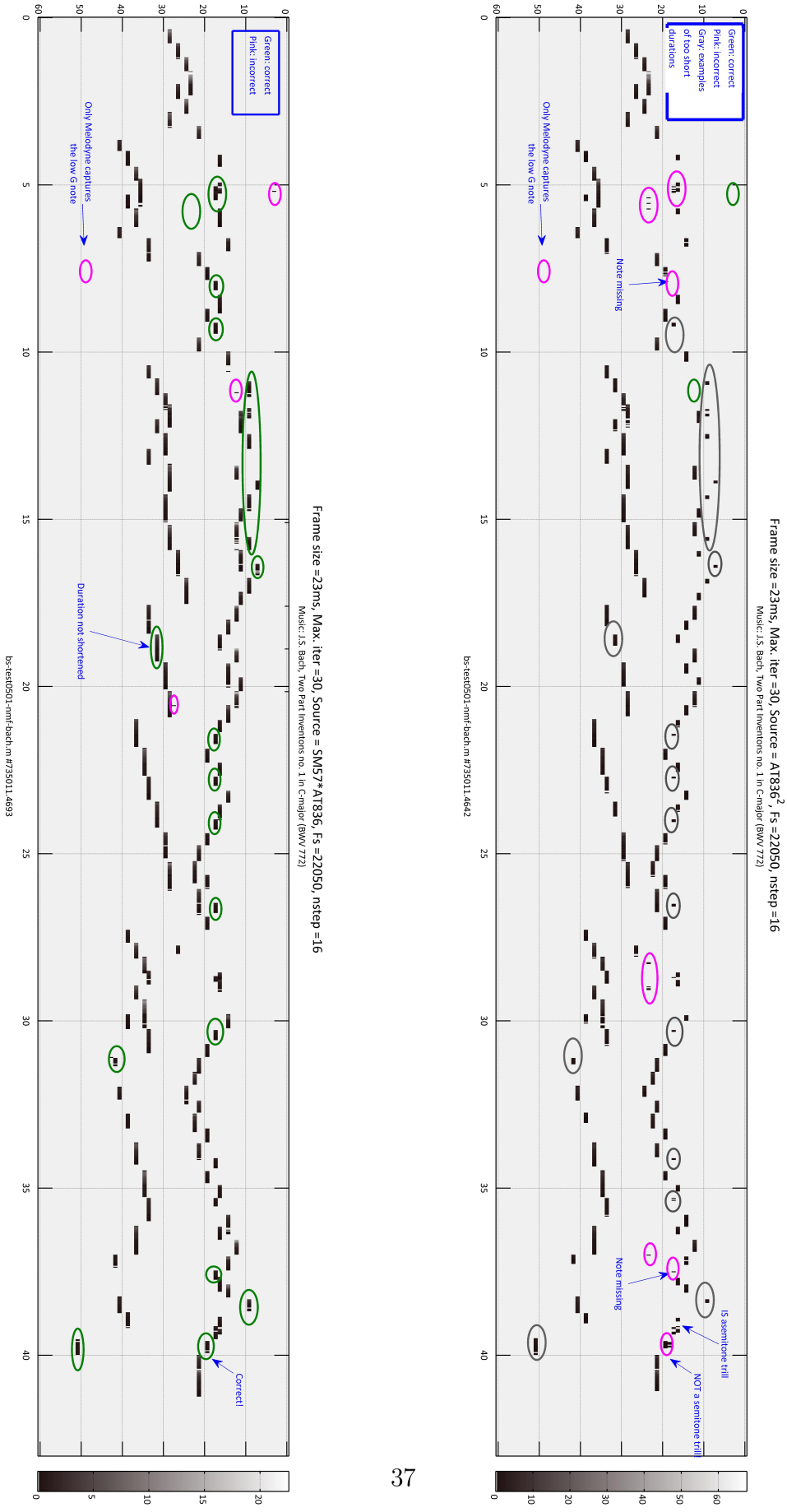

Figure 4.9: Pianoroll representaions of unprocessed squared activation vectors from two sources

4.4 Informal Assessment of the Method

As our time spent on spectral analysis did not yield any viable solutions, and we do not have MIDI data for any of our harpsichord recordings, it is not really possible to perform a proper quantitative assessment of the method; we shall leave that for future work. However, we have done an informal assessment by comparing our transcription of the Bach piece with MIDI data from another performance of the same piece, as well as compared it to a transcription done off-line using the state-of-the-art proprietary software; Celemony's Melodyne.

Figure 4.10 shows what we believe to be a very accurate result. The topmost plot is the multiplied activation vectors; this time plotted with a different colormap that does not show the very weak onsets. The low G note is of course still missing, but other than that we haven't found any wrong notes; the discrepancies between the two are due to Richard Stern playing the piece a little differently than the MIDI performance.

4.4.1 Comparison with Melodyne

Celemony's Melodyne is generally regarded as the state-of-the-art in commercial software for polyphonic pitch-detection. It is known to perform extremely well, and was a near revolution in music software, when it was released a few years ago. Hence, it is interesting to see how it does with our harpsichord recordings; figure 4.11 shows the result. Interestingly, Melodyne does not have any problems detecting the low G note that our algorithm did not capture. Melodyne also performs significantly better when fed the stereo wav file with the SM57 and the AT836 signals - giving us some clue that it too utilizes the extra information to solidify its predictions. Surprisingly though, the software cannot capture the fast semitone trills that are so characteristic for harpsichord music; the NMF method does that quite consistently. Furthermore, Melodyne misses more notes and has more false note detections. So the conclusion is, that *in this particular case* the NMF algorithm - with the element-wise multiplied activation vectors from two different signals - actually outperforms the state-of-the-art proprietary pitch-detection software. In addition this tells us something about the complexity of the problem, and how significant priors can simplify it; that is, we knew the instrument, the mic's and each note template in advance.

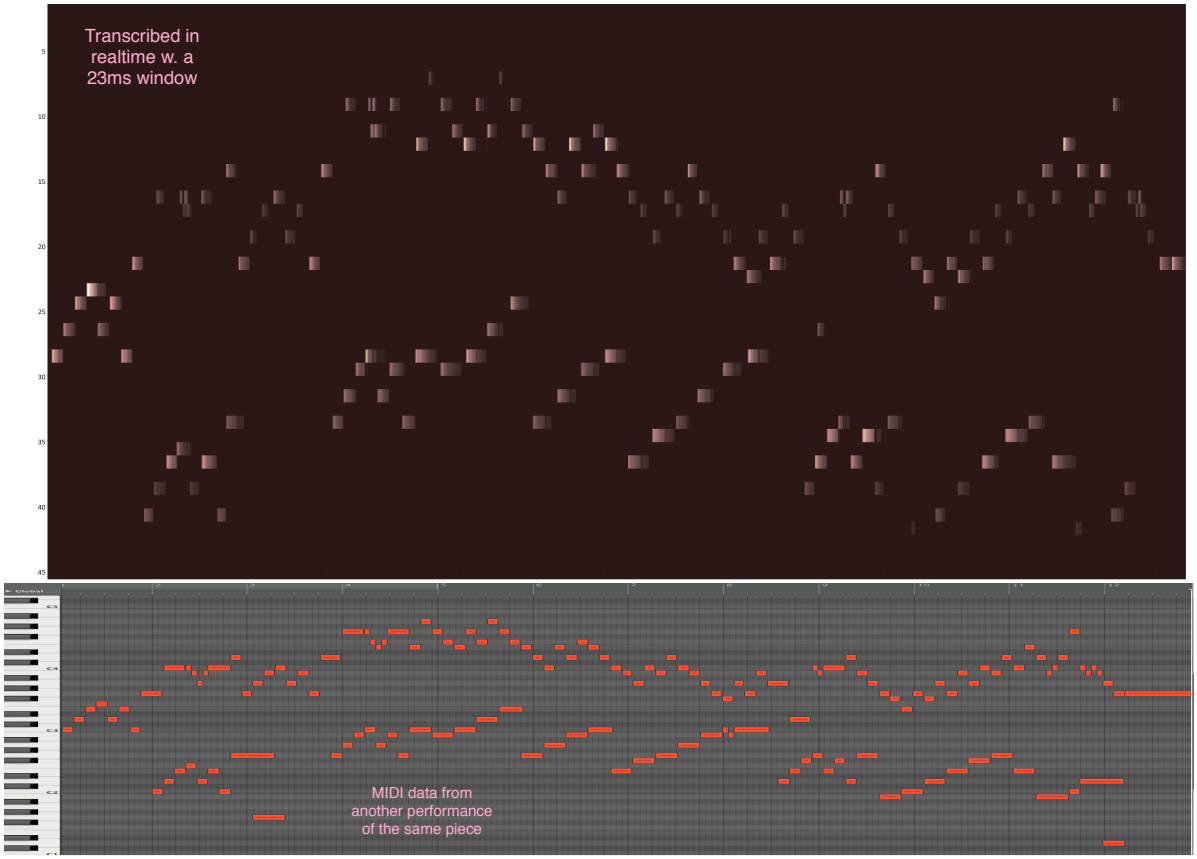

Figure 4.10: Comparing our transcription, using the piece-wise multiplication of the activation vectors from the SM57 and the AT836, with MIDI data from another performance of the same piece

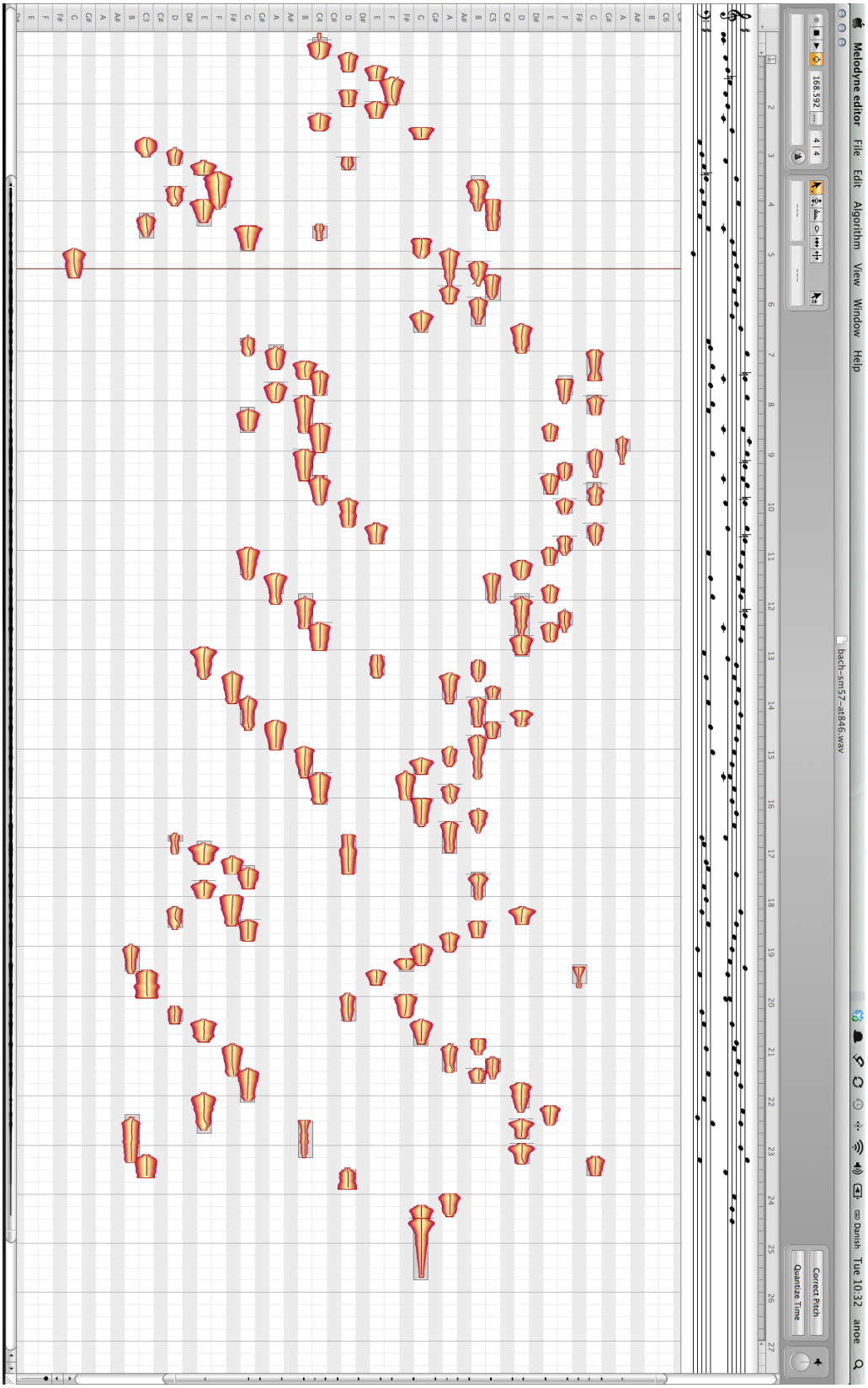

Figure 4.11: Transcription done off-line with Celemony's Melodyne, also using two mic's (SM57 & AT836)

Chapter 5

Conclusion & Future Work

We set out to study the problem of polyphonic pitch-detection for the harpsichord in real-time. The goal was to establish some measure of certainty towards which direction might lead to a robust final solution. A wealth of other techniques could just as well have been included from the vast amount of literature on the subject, given unlimited time and pages. However, within the current scope, we have achieved our goal; and with quite promising results to boot.

Dessein et al. report that the matrix decomposition has a tendency to shorten the note durations. Our results confirm this, insofar that squaring the activation vectors for a single signal significantly shortens the durations - indicating that the activations in the note tails are weak. This means that, as seen in [4], some amount of post-processing of the decompositions is needed in order to get proper durations. We have observed however, that by simply multiplying (element-wise) the activation vectors from two different signals, we seem to both cancel out noise and ambiguities - as well as get much better note duration detection. This approach then reduces, or even eliminates, the need for post-processing.

The main task we have ahead of us is to perform a proper quantitative assessment of the NMF algorithm discussed. If this approach does not end up being the actual solution it can serve as a rough baseline for comparison to other methods. Should it turn out, that in fact the use of two microphones, with our method for getting better data from them, quantitatively does a better job than can be done with a single microphone - this would, to our knowledge, be a novel and nice publishable result.

Bibliography

[1] Sunil Bharitkar and Chris Kyriakakis. *Immersive audio signal processing*. Springer, 2006.

[2] Richard Boulanger and Victor Lazzarini. *The Audio Programming Book*. MIT Press, 2011.

[3] P Comon and C Jutten. *Handbook of Blind Source Separation: Independent Component Analysis and Applications*, volume 35 of *Academic Press*. Academic Press, 2010.

[4] Arnaud Dessein, Arshia Cont, Guillaume Lemaitre, and Ircam Cnrs Umr. REAL-TIME POLYPHONIC MUSIC TRANSCRIPTION WITH NON-NEGATIVE MATRIX FACTORIZATION AND BETA-DIVERGENCE. *Information Retrieval*, (Ismir):489–494, 2010.

[5] Graham Grindlay. nmflib, 2010.

[6] J F James. *A Student's Guide to Fourier Transforms*. Cambridge University Press, 2002.

[7] Anssi Klapuri. Multiple fundamental frequency estimation by summing harmonic amplitudes. In Kjell Lemström, Adam Tindale, and Roger Dannenberg, editors, *Proc ISMIR*, pages 216–221. Citeseer, 2006.

[8] Princeton Sound Lab. http://soundlab.cs.princeton.edu/learning/tutorials/SoundVoice/add5.htm.

[9] D D Lee and H S Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[10] Garet h Loy. *Musimathics, vol. 1*, volume 1. 2006.

[11] Gareth Loy. *Musimathics: The Mathematical Foundations of Music, vol. 2*, volume 61. The MIT Press, 2011.

[12] Robert J Schilling and Sandra L Harris. *Introduction to Digital Signal Processing using MATLAB*. CENGAGE Learning, 2nd edition, 2011.

[13] Mikkel N Schmidt. *Single-channel source separation using non-negative matrix factorization*. PhD thesis, 2008.