Multiple fundamental frequency estimation and instrument recognition using non-negative matrix factorization

Master's thesis

J.M.S. de Wit, student number 3511324
Utrecht University

Supervised by:

W.B. de Haas
R.C. Veltkamp
A. Volk
F. Wiering

February 9, 2012

This one's for you, dad.

Abstract

Non-negative matrix factorization (NMF) is a model-based technique, where a model is built up to reconstruct a musical piece. This model is generated by linearly combining a limited set of known note structures at each point in time. This technique can help us to achieve automatic transcription, where note information is extracted from a musical recording.

For example, if a C and D note are played in the original musical piece, we hope that the matrix factorization process uses the structure of a C and D note to reconstruct this part of the musical piece. If this is indeed the case, because we know that these structures belong to certain notes, we can tell which notes were played in the musical piece - a process known as automatic transcription.

In this project, we examine the performance of NMF for the tasks of estimating the frequencies that are active in a musical piece, which is closely related to the pitch values of notes that are played. We find that it is trivial for several NMF variations to obtain high recall, where correct notes are detected and returned.

This result does however come at the cost of some noise in the returned values, therefore we consider several post-processing steps to reduce the number of incorrect frequencies that are returned. We find that this task is complex and deserves more attention in future research.

The same NMF technique was also applied to the task of instrument recognition, where we attempt to derive the instrument that was used to play a recorded note. Algorithms designed for this task are still challenged by recordings of musical pieces where several instruments and notes are sounding at the same time.

We believe that NMF may be able to extract notes from such a recording that are of high enough quality to be able to derive the instrument from these extracted notes. Results in our experiments were inconclusive but promising.

Combining the results of music information retrieval tasks such as those we have discussed above, and involving more temporal information by thinking in notes instead of individual timesteps, should lead to improved overall performance for all of the tasks that are involved.

Foreword

It is impossible for me to describe the amount of support I have received from the people at Utrecht University that have either supervised or helped this project along in some way; Frans Wiering, Bas de Haas, Remco Veltkamp and Anja Volk. I would like to take this chance to thank Utrecht University for giving me the opportunity to learn so many new things. I have been given the chance, both in the Game & Media Technology course and the thesis project, to learn about many different disciplines and fields of interest.

Many thanks go out to Chunghsin Yeh, Axel Roebel and Niels Bogaards for publishing their markers for separating the individual instrument sounds from the RWC database, a database for which Masataka Goto and his team deserve much credit.

Without the help of Emmanuel Vincent, I would probably still be struggling in trying to generate a basic implementation and test environment. His example source code and e-mail support on datasets have been invaluable. The community and researchers related to music information retrieval are all very open and friendly, a trait that I believe will really improve the quality of research. A lot of work is well documented and the enthousiasm of the authors is very inspiring.

I (literally) would not be here without my mother Ria, my girlfriend Laurie and all of my dearest friends. They have supported me not only through this project but through all of life. I will never forget all the fun times we continue to have. My father, Jan, has always been my inspiration and best friend. Though he passed away several months after I enrolled at Utrecht University, his guidance and love will remain forever. This world has lost a wonderful person way too soon. I hope they are playing some good tunes up there, dad.

Jan de Wit
December 30, 2011

Glossary

adaptive boosting Machine learning technique that combines several weak learners to build a reliable classification system. 44

additive The sound of a note is made up of the addition of all vibrations of air at a number of frequencies. 22

aliasing The effect that vibrations at high frequencies will not be picked up because they fall within two samples, occurs when the sampling rate is too low. 21

amplitude Magnitude of a periodic pattern such as vibrations of air. 18

attack First phase of a note with a quick rise in amplitude and fluctuation in the fundamental frequency. 18

attack-decay-sustain-release (adsr) Model describing the four phases in amplitude development of a note. 18

automatic transcription Extracting all information from a musical piece needed for musicians (or a computer) to fully reproduce the musical piece. 11

bandwidth A certain frequency range. 26

center frequency The middle frequency between the highest and lowest frequency included in a frequency range. 26

cepstral representation A time-frequency transform of a time-frequency transform, providing information on the rate of change in the frequency bands. 48

confusion matrix Method of evaluation that lists all assigned labels in a matrix where the diagonal shows correct assignments. 14

content-based retrieval Using multimedia content as a query to search a database. Example: query-by-humming. 13

convergence Point at which the difference between two iterations of an algorithm will remain stable. 33

cutoff frequency The highest frequency included in the range that is allowed to pass a filter. 26

decay Second phase of a note where the amplitude fades down to the average amplitude of the note. 18

decibel (dB) Relative measure related to the human perception of amplitude. 18

decision boundary In machine learning, it is the (learned) boundary that determines the difference between two classes. 45

distortion measure Measure of the difference between two values, entities or matrices. 33

duration The timespan of a note, defined as the difference between the end time and the start time of the note. 18

Equivalent Rectangular Bandwidth (ERB) Frequency scale based upon research into the frequency response bandwidths of the human auditory system. 28

f-measure Evaluation metric defined as a weighted combination of recall and precision. 14

filter bank A collection of filters, positioned through the frequency range. 25

Fourier series A linear combination of sine waves, each with different amplitude, frequency and phase. Can be used to describe complex signals. 22

frequency Number of times per second that a periodic pattern (such as vibrations of air caused by musical instruments) repeats itself, related to how we perceive sound. 16

fundamental frequency Lowest harmonic frequency that is perceptually related to a certain pitch value. 19

ground truth Values that are determined to be correct, against which the output of an algorithm can be compared to measure its performance. 12

harmonic Frequencies that are whole-number multiples of the fundamental frequency. 19

Hertz (Hz) Measurement of frequency where n Hz means that a pattern repeats n times every second. 16

International Society for Music Information Retrieval (ISMIR) Conference on music information retrieval. 12

machine learning The act of learning machines to learn models for a number of classes from a set of training samples, which can then be used to assign classes to unknown samples. 43

monophonic Only one note will be active at the same time. 11

mother wavelet Basis function for wavelet analysis, is shifted through the time and frequency range. Can be any of a number of functions. 27

Music Information Retrieval (MIR) Field of research that is concerned with retrieving all types of information from music automatically. 11

Music Information Retrieval Evaluation eXchange (MIREX) MIR evaluation session organized by ISMIR in which submissions from various research groups are evaluated on the same dataset. 12

non-negative matrix factorization (NMF) Model-based method to factorize a matrix Y into its components S and A. 31

Nyquist rate Two times the highest frequency that is present in the signal. Sampling at this rate ensures that no frequency information is lost up to half of this Nyquist rate. 20

onset A characteristic peak in amplitude that occurs during the attack phase of a note. 18

overtone Vibrations at a frequency higher than the fundamental frequency. 19

periodic A pattern that repeats itself at a certain frequency and amplitude. 18

pitch Perceptual measure related to how humans perceive frequency, thus allows us to place notes vertically on bars of sheet music. 11, 17

polyphonic Several notes may be active at the same time. 11

precision Evaluation metric defined as the number of correct frequencies relative to the number of estimated frequencies. 14

recall Evaluation metric defined as the number of correct frequencies relative to the number of ground truth frequencies. 14

release Fourth and final phase of a note where the amplitude of the note fades back down to zero. 18

sampling Storing the properties of a signal at a certain interval through time. 20

sinusoid Sine wave that can be shifted in frequency, phase and/or amplitude using parameters. 23

source separation The task of assigning each note to a certain source (instrument). In speech recognition, assigning each speech to a certain speaker. 12

spectral envelope Distribution of the total amplitude of a sound among the frequencies. 19

spectral flux Measure of change in energy at each frequency from successive timesteps. 41

spectrogram Result of a time-frequency transform that contains time, frequency and amplitude information. 24

sustain Third phase of a note where the note is stable, taking up the largest section of its lifespan. 18

temporal envelope Amplitude development of a note through time. 18

timbre All properties that determine instrument character. 20

timestep Step through time in a sampled signal. 21

weak learner Simple classification method in machine learning, will not have reliable performance on its own. 44

List of symbols

Basic properties

e	base of the natural logarithm

Sine waves

ϕ	phase
B	sinusoid amplitude coefficient collection
b	sine wave amplitude coefficient

Fourier analysis

ω	Fourier frequency index, defined as $2\pi f$ where 2π is the natural frequency of sine and cosine
C	Fourier coefficients showing the activity at each frequency
i	complex number $i = \sqrt{-1}$

Time domain

M	number of timesteps
m	timestep index
x	input signal $(1 \times M)$

Frequency domain

$cf(f)$	center frequency of frequency bin f
F	number of frequency bins
f	frequency bin index
$f0$	fundamental frequency

Time-frequency domain

ψ	wavelet function
T	number of timesteps
t	timestep index
X	input signal $(F \times T)$

Matrix factorization

β	choice of beta divergence
A	activations (temporal envelope) of envelopes S $(I \times M)$
E	(harmonic NMF) activations of narrowband spectra N

I	number of spectral envelopes in S
i	(source-filter NMF) index of source
i	index of spectral envelope
j	(source-filter NMF) index of filter
k	(harmonic NMF) number of narrowband spectra in N for certain index i
N	(harmonic NMF) narrowband spectra
P	(harmonic NMF) overtone spectral structures
R	(source-filter NMF) filters
S	spectral envelopes
U	(source-filter NMF) sources
W	(harmonic NMF) activations of overtones in P
Y	NMF reconstruction of time-frequency spectrum

Spectral features (instrument recognition)

a	(spectral flux) exponential decay factor (user-defined)
AE	amplitude envelope, derived by summing the energy of all frequencies
b	(spectral flux) local mean offset (user-defined)
b	octave bin identifier
H	number of harmonics to consider
h	index of harmonic
$NB(b)$	number of frequency bins contained within octave scale bin b
o	index of moment, e.g. $o = 3$ is skewness
X_b	time-frequency spectrum of octave b

Contents

Chapter 1

Introduction

1.1 What is music?

Music is a complex phenomenon that cannot easily be explained using a set of rules or measurements. This is exactly why automatic analysis of music is such an active and interesting area of research. It is possible for us to describe music in a simplified form as events, or notes, that are generated by audio sources such as musical instruments and voices [41]. By changing an audio source or its properties, we can introduce variation in the way these notes sound. For instance, it is possible to influence the sound of a piano by pressing different keys with a certain amount of pressure. Factors such as the recording environment, effects and post-processing also contribute to sound. In this project, we focus solely on instrumental music and do not take percussion into account.

There is a difference between monophonic and polyphonic music. In monophonic music, only one note will be active at any point in time. Think of a melody that is being played by one instrument with no accompanyment, or a person singing in the shower. Polyphonic music can have several notes active at the same time, which increases the complexity of automatic analysis as all these notes together now contribute to the one final sound that we observe. It is possible for the sound of notes in both monophonic and polyphonic music to overlap through time, as it takes a while for a note to completely fade out after a musician stops playing it. During this fade out phase, a new note may have started.

An important property of a note in the context of this paper is its pitch, a measure that allows us to place notes vertically on bars of sheet music. It defines what we know to be for example the difference between a C and a D note. The way this pitch of a sound is determined will be discussed in detail in section 2.1.1.

1.2 Automatic analysis of music

The broad definition of music poses a challenge of making computer systems understand the content of a song. This is similar to how computers are applied to automatically read text, recognize faces, help us park our cars and so on. Less related to music but still a very active area in audio analysis is speech recognition, in which a computer system is able to recognize words that are being said. An example of this technique has recently been included in the YouTube technology, where spoken words from the video are also shown in text at the bottom of the video.

Computers do not have the auditory sense and perception that we possess, therefore the analysis of audio is not as straight-forward to computers as it is to us. To counterbalance this difference, researchers have drawn inspiration from our own hearing system and attempted to recreate this system using models. Alternatively, they have also introduced entirely novel approaches based upon the strengths of computers, such as the ability to analyze big chunks of mathematical data. The third option is to combine aspects of both approaches, taking into account how humans perceive audio to enhance automatic analysis without attempting to recreate the entire hearing system.

The field of Music Information Retrieval (MIR) is concerned with retrieving all types of information from music automatically. The purpose of this information is then mostly to allow and enhance the searching of extensive multimedia datasets. While the results of this project may eventually contribute to this task, at this point the main goal is to retrieve low-level information to aid automatic transcription.

1.3 Automatic transcription

A complete transcription should allow musicians (or a computer) to fully reproduce the musical piece that is being transcribed. The output of a complete transcription therefore is a piece of sheet music for each part or instrument.

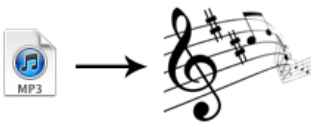

Figure 1.1: Transcription: from audio to musical score

This task of transcription consists of obtaining various components from the complete musical piece, most importantly each individual note and its properties such as duration and the instrument that was used to play the note. This is not a trivial task due to the enormous amount of possible note and instrument combinations. This paper focuses on retrieving both note and instrument information from a musical piece. To obtain a full transcription, information such as the key and beat of the musical piece is also required. While these are not explicitly treated in this paper, the results from the proposed method should provide a foundation for full transcription. Cemgil et al. [4] separate the transcription process into two phases: audio to piano-roll and piano-roll to score. We are focusing on the first phase, where audio is converted into notes with a certain pitch and duration. The second phase then adds information such as tempo, key and rhythm in order to obtain a complete score.

1.3.1 Research context

Most of the current research focuses on a certain component of automatic transcription, such as estimation of the pitch values present in a musical piece, instrument recognition or source separation. With source separation, each note or segment of speech is assigned to a certain source. This allows the identification of instruments in the case of music analysis, or different people in speech analysis.

A well-known conference on retrieving information from music data is the International Society for Music Information Retrieval (ISMIR). Besides providing many papers on the music information retrieval subjects, ISMIR organizes an annual evaluation session in which submissions from various research groups are evaluated on the same dataset. This Music Information Retrieval Evaluation eXchange (MIREX) provides an overview of the current possibilities and performance in retrieving information from music. Tests and datasets are developed for several retrieval tasks, such as key detection, tempo estimation and genre classification.

When applied to monophonic music where there is only one note active at any point in time, the active note at any time can be obtained with a high success rate. An example of a method for this task is YIN [7], which in combination with related techniques is used in the MIREX challenges as a guideline to create ground truth data - values that are determined to be correct, against which the output of a novel method is compared to measure its performance. For polyphonic music, this process is much more challenging because the notes that sound at the same time influence eachother. It is also not known how many notes are sounding at a certain point in time, while for monophonic music we know that this will always be just one note.

1.4 Motivation

Assuming it is possible to get usable transcription results from audio data, these transcriptions can serve many purposes. For example, after obtaining a transcription it will be possible to compare this notation between recordings, or to search a database of audio using pitched input. An example of such a search mechanism is query-by-humming where pitched humming is recorded, the correct pitch values are derived and then used as a search query. This form of content-based retrieval will enable media retrieval even when no textual labels are present in a set of multimedia data and may also improve retrieval performance if there are labels [22].

The transcription results can be used to derive higher-level features regarding the song as a whole, such as genre. Another possibility, if the transcription method performs at real-time, can be automatic generation of accompanyment. Ibáñez in [26] also suggests plagiarism detection as a possible use.

One can also think of applications related to gaming, such as using voice or instrument sounds as input to control a game situation or generating dynamic worlds based on audio data. Games in which the player is expected to match the notes being played using a musical controller as in Guitar Hero have gained popularity over the last few years. Perfect transcription would allow arbitrary songs to be used with these games, as well as the use of regular instruments as input controllers. It would be a refreshing step to use pitched audio to control or even fully generate game content, to somehow translate a song into game content such that each song results in a different experience.

Most importantly, we notice an ever growing amount of attention for music in the digital era, as we can tell from the popularity of various social networks and (online) stores that are based around music. It is interesting to work with multimedia data as it is very extensive and therefore provides endless possibilities for obtaining descriptive information. Research into automatic music analysis combines many fields of interest, such as signal processing, machine learning, music theory and computer science.

1.5 Problem statement

The task of retrieving the notes that were played in a musical piece becomes increasingly challenging as the number of voices, the number of notes that sound at the same time, increases. When trying to retrieve the notes from polyphonic music, many incorrect notes may be discovered as well, due to the harmonic spectral structure of each note. The challenge then lies in distinguishing between notes that were actually used to generate the musical piece and other incorrectly returned notes. We believe that information such as note duration, number of voices and the instrument that was used to play the note can help this extraction of correct notes.

Most research so far has focused on either instrument recognition or pitch estimation. We believe that by combining the results of both tasks it is possible to improve the results of both individual tasks. The combined instrument and pitch information can help us to achieve *source separation*, where each note can be attributed to a certain source, in our case instrument. This in turn may help us to estimate the number of voices and obtain an overview of the instruments that are present in the musical piece, information which can then be used to further refine instrument recognition and pitch estimation performance.

In short, we examine the possibilities and quality of pitch estimation and instrument recognition, focusing on ways that may help to combine the two tasks and improve their performance. We focus on the technique of non-negative matrix factorization, that has its uses in both instrument recognition and pitch estimation. This technique is explained in detail in section 3.2. We attempt to determine the strengths and weaknesses of different implementations of this technique and to maximize its potential for the task of automatic transcription.

1.6 Evaluation

The evaluation of our pitch detection system is done in a way similar to how the MIREX challenges are evaluated. We use the MIREX development dataset containing five single instrument recordings, each thirty seconds in length, then combine these recordings to form mixtures with the number of voices ranging from one to five. Active pitches at each time step of 10 ms are provided in a text file, which is then compared to the output of our system, taking into account the number of correctly returned values as well as incorrect values. It is therefore not only important to return the correct pitch, but also to minimize the amount of incorrect values or noise.

Instrument recognition can be evaluated by using one dataset of isolated notes played by a number of instruments for training and another similar dataset with the same instruments for testing. Both datasets are labeled with the correct instruments, therefore it is possible to see how many samples of each instrument are assigned the correct label. To test performance with polyphonic music, we used the active pitch data from the MIREX development set to extract individual notes from a mixture of five instruments. These pitch values are grouped in columns, one for each instrument, enabling us to label the dataset accordingly. These notes are then used as a testing set and can be evaluated in the same way as the two datasets containing isolated notes.

The evaluation metrics used are identical to those used in the MIREX challenges and in related research. For pitch estimation, we use recall, precision and the f-measure, which are defined as follows:

$$Recall = \frac{number\_correct}{number\_groundtruth} \tag{1.1}$$

$$Precision = \frac{number\_correct}{number\_estimated} \tag{1.2}$$

$$F = 2 \times \frac{precision \times recall}{precision + recall} \tag{1.3}$$

The number of estimated and correct frequencies is sampled at 10 millisecond intervals and then summed up for the entire duration of the musical piece, after which the above metrics can be calculated using this total amount of estimated and correct frequencies.
These metrics are then evaluated at 10 millisecond intervals and combined to obtain the performance of the entire musical piece. Recall indicates how well a method is able to find the correct values from the ground truth. Perfect recall could be obtained by simply returning all pitch values at each point in time, as in that case all pitch values from the ground truth will automatically be present. This is why we also include precision, which shows how many of our estimated pitch values were actually correct. This means that if we now return a large number of pitch values - probably resulting in high recall - we will end up with low precision because many of these pitches will not be correct. The F-measure is a weighted combination of precision and recall.

Instrument recognition is evaluated on a note basis rather than for each timestep. Therefore, instead of using the metrics we use in pitch estimation, we derive a confusion matrix containing the possible instrument label combinations. An example of such matrix:

	Bassoon	Clarinet	Flute	Horn	Oboe	
Bassoon	**10**	2	1	3	1	59%
Clarinet	4	**4**	2	3	2	27%
Flute	1	2	**5**	4	3	33%
Horn	0	2	1	**3**	5	27%
Oboe	2	1	0	3	**6**	50%

Table 1.1: Confusion matrix example

The rows in table 1.1 indicate the ground truth value, so the first row displays all notes that should have been labeled as *bassoon*. The various columns of this first row then show how many bassoon notes were labeled as belonging to that particular instrument. For instance, 2 notes were incorrectly classified as being played by a *clarinet*. This matrix therefore not only shows the performance of the instrument recognition method - the percentage of correct labels in the last column of each row - but also the distribution of incorrect labels. In the final column we

list the number of correctly assigned labels relative to the total number of samples for that particular instrument. The example shows quite a spread across the different instruments, indicating that our hypothetical instrument recognition method may not be able to reliably tell the difference between the instruments.

We believe that these evaluation metrics are able to display the performance of pitch estimation and instrument recognition. These tasks can be considered as low-level tasks, where we are interested in raw performance metrics such as those discussed previously. Hu and Liu in [14] however propose that alternative evaluation methods, most of them focused on the end-user, should be considered. We believe that these methods are especially useful in evaluating a complete system for tasks such as transcription or information retrieval.

1.7 Thesis structure

Before discussing the information retrieval tasks of estimating the fundamental frequencies and instruments in a musical piece, we first provide some background information in chapter 2. We are then able to move on to the first task of estimating the fundamental frequencies present in a musical piece in chapter 3, followed by instrument recognition in chapter 4. Using the theory from these chapters, we define a number of experiments in chapter 5. Using the datasets and evaluation methods defined in chapter 6 we have conducted these experiments and elaborate on the results in chapter 7.

As one would expect, this thesis will not be able to immediately solve the entire challenge of automatic transcription. For this reason our conclusions, future directions and possible applications for this work are listed in chapter 8.

1.8 Contributions

In this thesis project, we have attempted to evaluate the use of the non-negative matrix factorization (NMF) technique, explained in section 3.2, for estimating the fundamental frequency values and instruments present in a musical piece. We investigate the potential performance we can expect from NMF, without considering any post-processing. Furthermore, we try to identify the most challenging aspect of frequency estimation.

We then attempt to apply NMF to the task of instrument recognition. NMF attempts to reconstruct a musical piece using a set of basic frequency structures (spectral envelopes). We investigate whether NMF during this reconstruction process automatically prefers the use of spectral envelopes that belong to the instrument that was indeed used to create the musical piece we are trying to recreate.

To retrieve multiple spectral envelopes from a set of isolated note recordings (training data) that each describe the complete spectral structure of a certain phase or variation of the note, we apply L0 sparse NMF to these recordings. We then explore the effect of using these additional envelopes on fundamental frequency and instrument recognition performance. The influence of the choice of time-frequency scale is also taken into account.

Most instrument recognition methods are based on isolated note recordings. This means that a set of single notes is used to train a model, which is then used to assign instrument labels to a different set of single notes. We attempt to simplify the instrument recognition task for musical pieces where a combination of notes is played, by extracting each note individually from the musical piece using NMF. Although we expect some loss in quality to occur because the notes are extracted from a musical piece containing multiple voices, each of these extracted notes can be used for instrument recognition as if they were isolated recordings.

Chapter 2

Background

Before turning to the task of audio transcription, it is important to present the underlying techniques on which the transcription system is based. Most of these techniques are used in research other than audio analysis as well.

First, the process of recording and storing music is briefly explained. This will provide a better understanding of the structure and limitations of the input data. Following this is an overview of ways to transform this input data into different representations that are able to provide information which is not directly visible in the original input.

2.1 Music

If we simplify the concept of music, we can regard it as a set of notes that is played by a combination of instruments. We exclude the addition of voice and percussion in this project. Every sound we hear is produced by vibrations of air. These vibrations can be caused by several phenomena, such as effects of nature, machines, our voice and musical instruments. The vibrations can be observed by our auditory system, allowing us to hear sound. In the case of musical instruments, the vibrations of air can be caused for example by plucking a string or by blowing air into the instrument.

Besides musical notes, we know and recognize many types of sound such as car engines, knocking on doors and so on. The main difference between these sounds in our daily lives and music is that the vibrations of air in the case of music occur in a pattern that repeats itself at a certain frequency - a number of times per second. Frequency is measured in Hertz (Hz), where 1 Hz means that the pattern repeats once in one second. 2 Hz means that the pattern repeats twice in a second. The frequency of air movement is strongly related to what we perceive as *pitch*. Besides pitch, other important properties that determine how we perceive a sound are amplitude, duration and timbre. By setting and altering these properties we can compose and perform music.

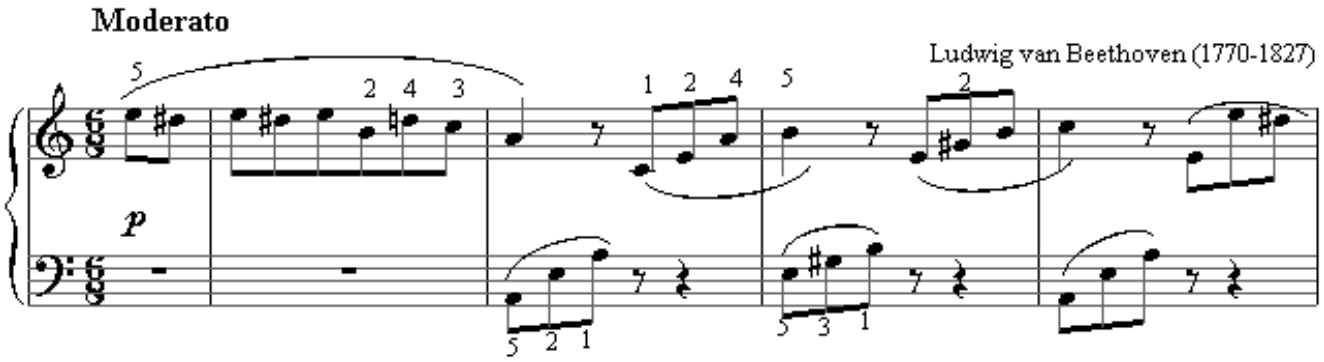

Figure 2.1: First four bars of *Für Elise* by Beethoven

Figure 2.1, a piece of sheet music, is an example of how we can perform music by knowing the properties of a musical piece. Pitch and duration determine the position and appearance of each note. Pitch is displayed on the vertical axis, duration determines the position on the horizontal axis as well as the type of note (half note, quarter

note and so on). The exact duration of a note in time depends not only on the type of note, but also on the beat (displayed in the first bar) and the *tempo* - in this case, *Moderato*. Tempo can also be measured in beats per minute (bpm). Amplitude and timbre (most importantly: choice of instrument) can be annotated and enforced implicitly by distributing different sheet music for different instruments.

2.1.1 Frequency and pitch

Pitch is a perceptual property of sound related to how we as humans hear differences in frequency. The Western tone system uses *octaves*, collections of twelve pitch values. A piano can have 88 keys, starting with the lowest pitch with each key one pitch value higher than the previous one.

While each pitch value is related to a certain frequency, this is not necessarily an exact and stable correspondance. The range of frequencies is much larger than the (usually) 88 different pitch values. Humans are able to hear frequencies in the range of 20-20,000 Hz. This range will vary between individuals and may decrease with age. The range of frequencies that relate to pitch values is smaller, from 27.5 Hz to 4,186.01 Hz.

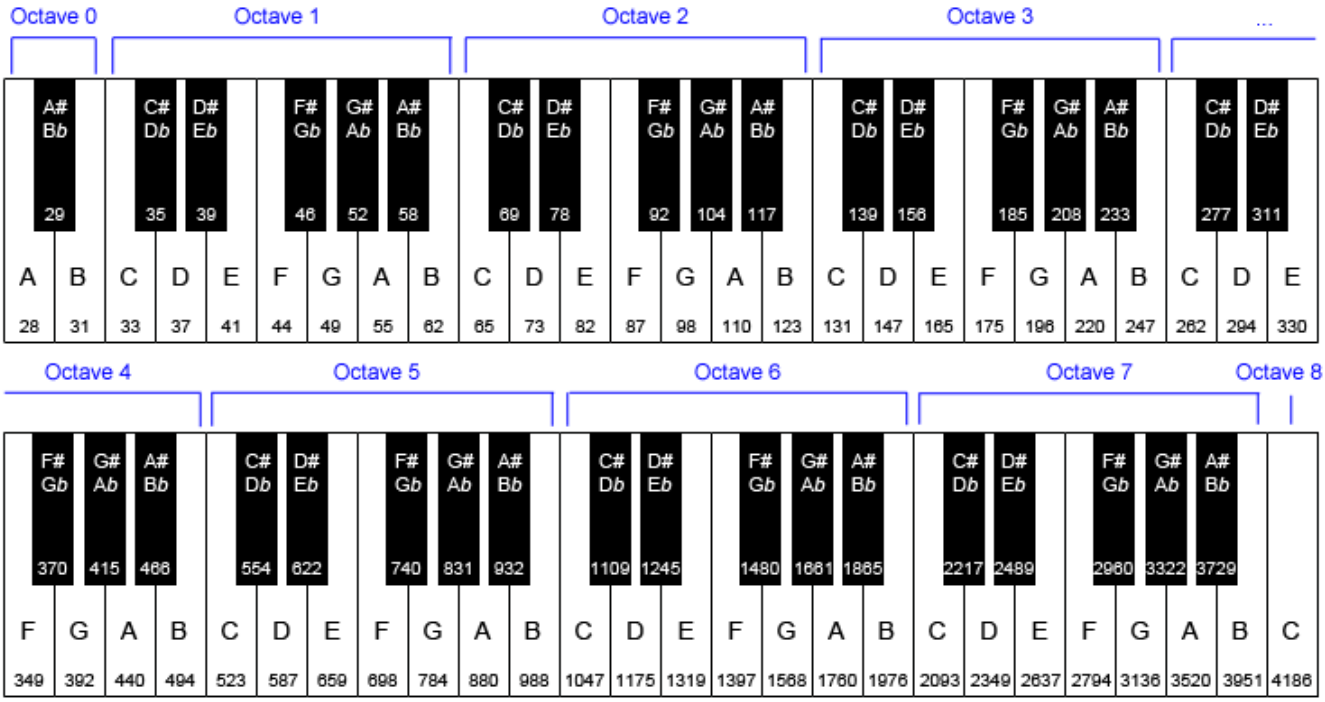

Figure 2.2: All 88 piano keys with their frequency values and octave numbers

Figure 2.2 shows the 88 piano keys with their pitch values ranging from A0 to C8. Displayed below the pitch values is the frequency belonging to that pitch, rounded off to whole numbers. This frequency will not always match the value belonging to a certain pitch exactly, but we are still able to perceive the note as having that pitch value. The relation between pitch and frequency is defined as:

$$frequency = 440 * 2^{((pitch-49)/12)} \tag{2.1}$$

The frequency of a note is related to how fast an instrument is causing the air to vibrate repeatedly. A faster frequency of vibration means a higher pitch.

This vibration pattern can be clarified by comparing it to the behaviour of a sine wave:

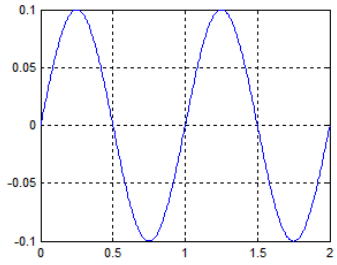

Figure 2.3: Example of a sine wave with frequency 1Hz and amplitude 0.1

The main difference between this sine wave and the vibration pattern caused by an instrument is that a sine wave has no sense of time so it will go on endlessly at some frequency and amplitude - it is periodic. A note played by an instrument has a duration - a start and end time because the instrument player stops playing that note or the vibrations progressively lose amplitude until they can no longer be observed. Upon plucking a guitar string or playing a piano note, the string will not keep vibrating forever. Regardless of this difference - which is important to remember as we proceed - the sine wave provides a visual indication of the repeating pattern of air vibration.

2.1.2 Amplitude

It is possible to apply different amounts of pressure to an instrument, for example by plucking the strings of a guitar with a certain power or blowing certain amounts of air into a trumpet. Harder plucking of strings or blowing more air into a trumpet will result in stronger vibrations of air. The vibrations have a certain amplitude. We perceive this as differences in loudness. Amplitude is measured in decibel (dB), a measure related to the human perception of amplitude. However, the exact loudness is subjective and may vary between people [26]. The amplitude will not be constant as it may first take a while for an instrument to start producing sound and this sound will usually fade away rather than disappear instantly. The changes in amplitude and as a result the lifespan of a note can be described using the attack-decay-sustain-release (adsr) model, describing the four phases of a note. This amplitude lifespan is called the temporal envelope, describing the amplitude changes of a note through time.

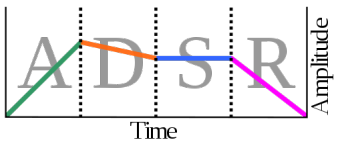

Figure 2.4: ADSR model of time envelope (courtesy of the Wikipedia page on ADSR)

The note starts with an attack, which usually produces sound that is unique to an instrument type. In a piano, this can be when the hammers hit the strings. During this phase the amplitude will quickly rise to what is generally a peak in amplitude - higher than the average amplitude of the note. This peak is known as the onset, shown in figure 2.4 as the border between the attack and decay phases. The attack phase will often be paired with a lot of fluctuation in the frequency of the note. The decay phase is where the amplitude fades down to its average. The frequencies will also stabilize during this phase. Next is the sustain phase in which the note is stable, taking up the largest section of the lifespan of the note. Finally, the amplitude of the note fades back down to zero in the release phase.

2.1.3 Overtones

The sine wave representation of a note is a simplified representation of how a musical note is actually built up. It describes only the fundamental frequency - the frequency which is linked to a pitch value and is closely related to how we perceive notes and how we are able to place them on a musical score.

Notes will generally however cause vibrations at several frequencies - the fundamental and its overtones, resulting in a sound made up of vibrations at multiple frequencies, that is observed as being rich in sound. How the total amplitude of a sound is distributed among the frequencies is very characteristic of a sound. This distribution is known as the spectral envelope.

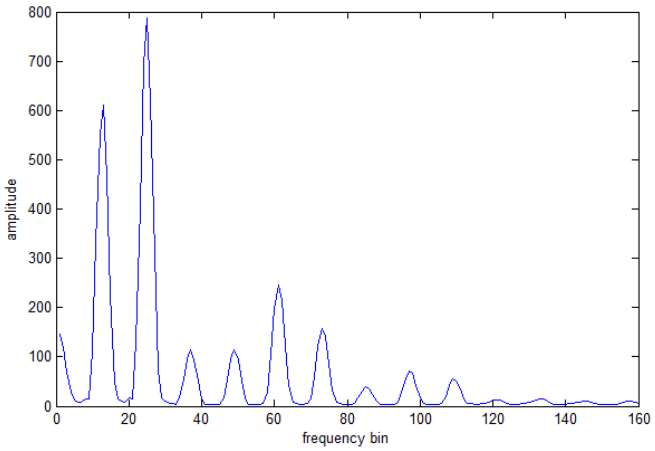

Figure 2.5: Spectral envelope taken from a middle C note recording of a flute

For musical notes with a certain pitch value, the frequencies other than the fundamental will be whole-number multiples of the fundamental frequency. They are known as harmonic frequencies. For example, if we play a middle C note (C of the fourth octave) having a fundamental frequency of $261.63Hz$, it will also cause vibrations at $2 \times 261.63 = 523.26Hz$, $3 \times 261.63 = 784.89Hz$ and so on. Note that the second harmonic belongs to the same note C one octave higher because the frequency doubles. The same increase in octave occurs at the fourth harmonic, the eighth harmonic and so on. For unpitched notes such as those caused by percussive instruments, it is possible for inharmonic overtones to be present in the spectral envelope.

While the relationship between the frequency of a sound and the pitch or tone we perceive is straight-forward if there is only one frequency, the addition of overtones in complex sounds produced by musical instruments can influence the pitch we perceive. For example, in the case of a *missing fundamental*, the fundamental frequency itself is not present in the perceived tone but its harmonics are. In this case, we are still able to detect this fundamental frequency due to the pattern recognition and interpretation that is done by our brain. The effect of the missing fundamental is used on purpose when a certain technology such as a speaker system is unable to produce sound at the fundamental frequency. The illusion of sound being played at a certain frequency is then created by generating the harmonics belonging to the desired fundamental frequency. Effects like the missing fundamental need to be considered when working with fundamental frequency estimation, especially when dealing with situations where the properties and limitations of the various sound sources are unknown.

The addition of overtones adds to the complexity of the task of transcription because we cannot simply examine the fundamental frequency belonging to each pitch value individually. If we did that, we could find activity caused by an overtone of a lower frequency note and treat it as a separate note. Furthermore, when dealing with multiple notes being played at the same time there can be an overlap in the overtones, as one of the notes can have a fundamental frequency at one of the harmonic frequencies of another note.

2.1.4 Timbre

It is possible for us as humans to hear the difference between instruments, even when they are playing the same note. Timbre of musical notes can be regarded as a number of properties such as *brightness* that together determine instrument character. It is not as straight-forward as amplitude or frequency in that it can only have one out of a fixed set of values; the timbre of a sound depends on many different properties. In fact, anything other than amplitude and fundamental frequency can be considered as being part of timbre. For voice, timbre is what enables us to tell the difference between the voices of different people, even if they are saying the same words. The same effect applies to musical instruments, where we are able to tell two instruments apart even if they are playing the exact same note.

As we have briefly pointed out when discussing overtones, instrument character depends heavily on the spectral and temporal envelopes, describing how the sound is built up from its harmonic overtones and how the amplitude of the sound evolves through time. To capture the timbre of a sound, we therefore need to extract a number of properties that apply to the structure of the sound as well as the changes of the sound through time. This is exactly the task that we will focus on in chapter 4 on instrument recognition.

2.1.5 Recording music

Live music can be seen as a continuous stream of information: vibrations of air at certain amplitudes and frequencies. It is impossible to record and store such a continuous stream, therefore this stream is *sampled* at a certain interval and its properties at that time are stored. If this interval is sufficiently small, the original stream can be recreated without much loss in quality, at least to human hearing. This is what happens when live music is recorded so that it can be played back from a computer or CD player.

Acoustic instruments are recorded using microphones, a step that is not needed for electric instruments as they already contain digital information. The microphone converts the vibrations of air into voltage vibrations in an electrical circuit. These rapid voltage changes are then measured using a *digitizer*, which is part of digital recording devices such as sound cards. The sampling process is also done by the digitizer, allowing the resulting data (a big set of discrete numbers) to be saved to a digital file. It is this type of file that will be used as input for our transcription system.

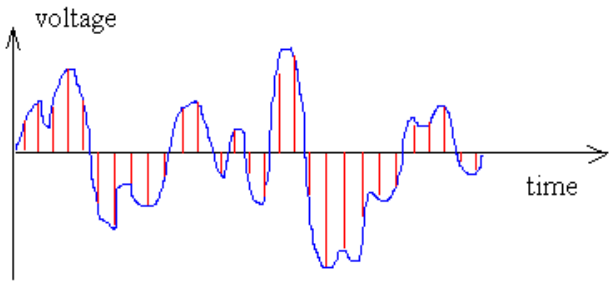

Figure 2.6: Sampling (red) of a continuous stream of voltage changes (blue) (courtesy of Milewski [25])

The sampling rate for audio will generally be at least 44,100 Hz, which means that samples are taken from the continuous audio stream 44,100 times each second. This is because of the Nyquist rate, which is equal to two times the highest frequency that is present in the signal. Sampling at a rate equal to or higher than the Nyquist rate ensures that no information up to half the Nyquist rate is lost. Remember that while the highest fundamental frequency is at 4,186.01 Hz, there will also be overtones at higher frequencies that contribute to the sound. As humans are able to observe up to around 20,000 Hz, sampling at a rate of at least twice this frequency will ensure that the recording will sound like the original performance and that no information will be lost - at least to our human hearing.

If sampling is done at a rate lower than the Nyquist rate, aliasing will occur, which means that vibrations at high frequencies will not be picked up because they fall within two samples. The result is that it is no longer possible to distinguish between those frequencies and lower ones. When playing back a recording with a lower sample rate, it is possible to hear the reduced quality compared to the original.

After sampling, an analog signal will also have to be *quantized* before being able to store it digitally. Quantization is the process of assigning amplitudes to the signal at each sampled timestep. When moving from analog to digital signals, the range of possible amplitudes is limited to the number of bits that are used to store the amplitude values - the *bit depth*. For example, a bit depth of 8 bits results in 256 possible amplitude values. When using a small bit depth, the amplitude range will be limited and subtle amplitude differences will be lost as samples of similar amplitude will be assigned the same value. A large bit depth however also increases the amount of data that will have to be stored, resulting in bigger audio files. In short, there is a trade-off between audio quality and storage requirements.

The sample rate and bit depth together with the number of channels (for example mono or stereo) determine the bit rate:

$$Bitrate = samplerate \times bitdepth \times numberofchannels \tag{2.2}$$

The bit rate is described in bits or kilobits per second, indicating the amount of storage capacity required for one second of audio data.

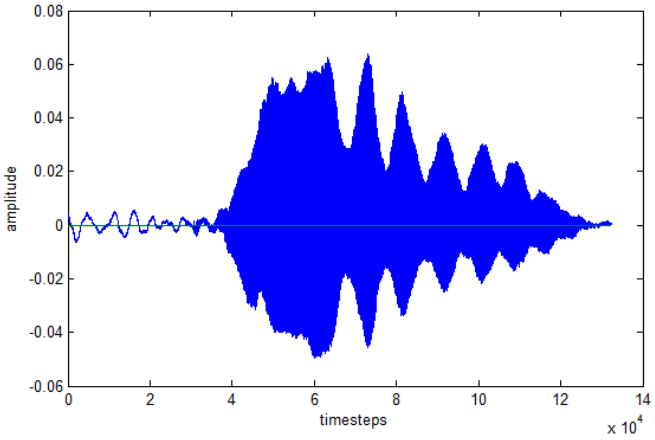

Figure 2.7: A digital representation of a middle C note played by a flute: sampled timesteps on the horizontal axis; amplitudes on the vertical axis

2.2 Time and frequency representations

The previous section has shown that we can examine an audio signal in two different domains. The first is the time domain, shown in figure 2.7, which provides information on the amplitude of the signal at any point in time. The second is the frequency domain, containing the energy or amplitude present in different frequency ranges. Time domain does not explicitly show the frequency structure of the signal, while the frequency domain has no sense of time since it applies to the signal as a whole. If the signal would contain only a single pure tone (no overtones), we would be able to derive the frequency of this tone by looking at the amount of time it takes for the signal to repeat itself in the time domain. Unfortunately, music consists of many of these repeating patterns that add up to create the resulting time domain representation, masking the frequency information in the process.

Music can be seen as an additive process, where the addition of all vibrations of air at a number of frequencies makes up the sound of a note. In turn, the addition of a set of these notes results in music.

For automatic transcription, we need to know the frequencies that make up the signal so that we can tell which pitch values are associated with the spectral envelope. By using information from the frequency domain however, we can only derive that certain frequencies appear *somewhere in the musical piece*. In order to determine when these notes begin and end, information from the time domain is required as well. We need a way to combine the two domains into a single representation, from which we can derive note information.

Before taking the step towards combining time and frequency information, we first examine a way to obtain a frequency domain representation as the digital music file that we use as input only provides us with time domain information.

2.2.1 Fourier series

It is possible to transform a signal from the time domain to the frequency domain by applying a technique based on Fourier's theorem. This theorem states that any periodic signal - a signal that repeats itself at a certain frequency f - can be constructed by using sine waves belonging to the harmonic series of this frequency (f, $2f$, $3f$,...nf). However, if we want to describe audio data using sine waves, we run into the fact that while the frequencies of air vibrations used to generate this audio exhibit periodic behaviour, they are not completely periodic as the vibrations will not go on forever while sine waves do. A simple solution for this difference is to simply assume that the vibrations *do* go on forever, so that a sine wave can be used to describe this behaviour.

We can then use a combination of sine waves, each with different amplitude, frequency and phase, to describe complex signals such as the sound of a note or even an entire musical piece. This combination of sine waves is known as a Fourier series:

$$x(m) = b_1 \sin(2\pi m + \phi_1) + b_2 \sin(4\pi m + \phi_2) + b_3 \sin(6\pi m + \phi_3) + ... + b_n \sin(n2\pi m + \phi_n) \tag{2.3}$$

Where m is the sampled timestep of the input function $x(m)$ and ϕ is the *phase*. 2π is the natural period of a sine wave, resulting in a frequency of 1 Hz at 2π, 2 Hz at 4π and so on. $b_1...b_n$ are the amplitude coefficients. $x(m)$ is the time domain representation of a sound, with an amplitude value for each timestep m. Phase determines the starting point of the sine wave, as shown in figure 2.8.

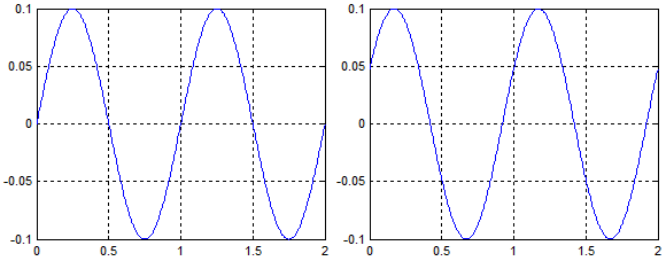

Figure 2.8: The effect of phase. The left sine wave has phase 0; the right wave is the same sine, with phase 0.5

A sine wave such as those shown in equation 2.3, allowed to shift in frequency, phase and/or amplitude is known as a sinusoid. The same combination of sinusoids can also be described using a summation:

$$x(m) = \sum_{f=1}^{F} B_f \sin(2\pi f m + \phi_f) \tag{2.4}$$

Where coefficients B determine the amplitude at frequency f. From this formula we can tell that we are describing our audio data, which is considered as a function x of amplitude values at points in time m, using a linear combination of sine functions with certain frequencies, phases and amplitudes.

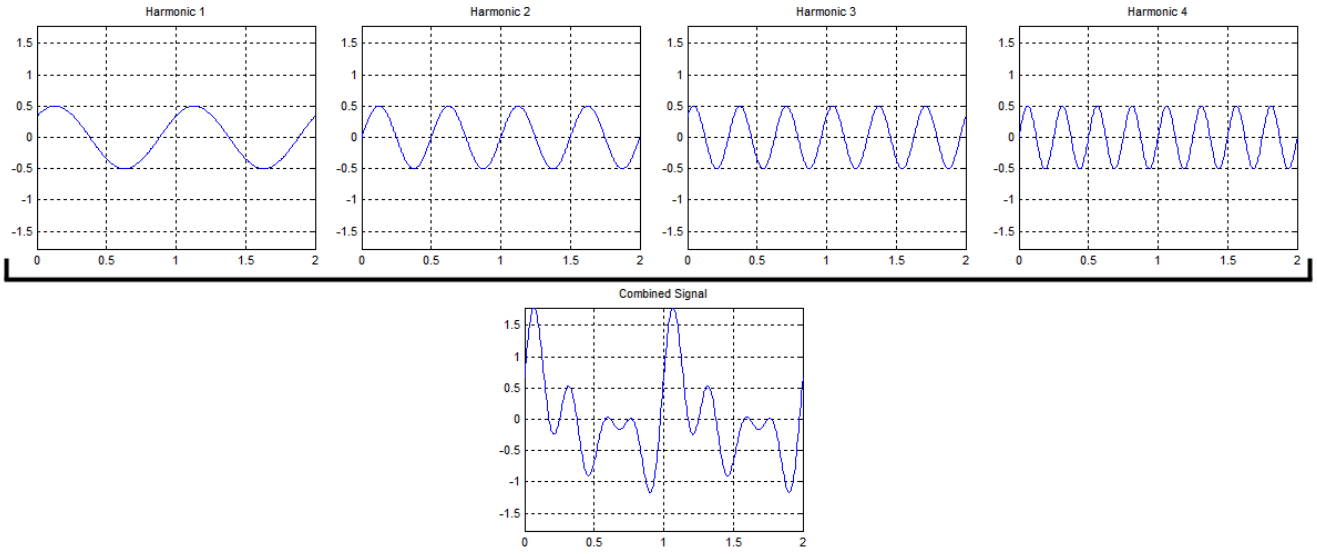

Figure 2.9: A complex function (bottom) reconstructed by superimposing several sinusoids (top)

Figure 2.9 shows how a certain complex function can be reconstructed by combining sinusoids. In this example, the amplitudes of all sine waves are set to 0.5. The frequencies are 1 Hz, 2 Hz, 4 Hz and 8 Hz. The phase of the first and third sines is set to 0.75, the others are set to 0. By increasing the number of sines, more complex functions can be reconstructed. This figure was generated using sinesum2[1].

It is considered easier for analysis to use a cosine to describe the phase instead of ϕ and then to represent this combination of a sine and a cosine using a complex exponential:

$$e^{2\pi i m} = \cos(2\pi m) + i \sin(2\pi m) \tag{2.5}$$

Where $i = \sqrt{-1}$ and e is the base of the natural logarithm. The cosine in this formula is the real part and the sine is the imaginary part. This particular combination is known as *Euler's formula* in complex analysis. This sinusoid is what we can consider a basic function in reconstructing our complex function $x(m)$, which is our audio signal. This is done by linearly combining a number of these sinusoid functions:

$$x(m) = \sum_{f=-F}^{F} C_f e^{2\pi i f m} \tag{2.6}$$

The complex coefficients C_f have a symmetry property which results in the total sum of all the components to be real. This property also changed our summation range to $[-F, F]$. We are interested in these complex coefficients C_f as they encode the amplitude of the sinusoid at each frequency.

[1] http://see.stanford.edu/see/materials/lsoftaee261/assignments.aspx

We can change this formula to regain a positive range of coefficients:

$$x(m) = \frac{1}{F} \sum_{f=1}^{F} C_f e^{2\pi i \frac{f}{F} m} \tag{2.7}$$

2.2.2 Discrete Fourier transform

A Fourier transform can be applied to a continuous or discrete signal. Because in this project we are dealing with discrete information - a number of sampled amplitudes at certain time intervals $x(m)$ - we will examine the discrete Fourier transform (DFT). As stated in the previous section, the coefficients from the Fourier series will provide us with the amplitude at each frequency. So all we need to do to get the transform, is to obtain the coefficients.

Let us review the construction of our audio function $x(m)$ using F sinusoid functions:

$$x(m) = \frac{1}{F} \sum_{f=1}^{F} C_f e^{2\pi i \frac{f}{F} m} \tag{2.8}$$

Obtaining the coefficients C_f can then be done by calculating:

$$C_f = \sum_{m=1}^{M} x(m) e^{-2\pi i f \frac{m}{M}} \tag{2.9}$$

Where C_f is the *energy* at frequency f and $x(m)$ is our time domain signal. Officially, energy is defined as a certain constant times the amplitude squared. However in this thesis with energy we mean the amount of the total amplitude that is explained by a certain frequency range. Note that we now sum over all sampled timesteps to get a frequency representation, where in the case of equation 2.8 we sum over the frequencies to get a time domain representation. Equation 2.8 can be used as the inverse DFT to regain the original time domain audio signal.

These transformations show that converting between the time and frequency domains does not affect the signal physically, it is just a different representation that allows for understanding and manipulation of the signal in the context of the task at hand. Technical implementations of the discrete Fourier transform generally apply a form of fast Fourier transform (FFT) which is more efficient than the regular discrete Fourier transform.

2.2.3 Combining time and frequency

By dividing the entire signal - in our case, a musical piece - into small time windows and applying a frequency transformation to each of these windows, we can form a time-frequency representation of the signal as a whole by combining the results of all frequency transformations. This principle applied using a Fourier transform is called a short-time Fourier transform (STFT):

$$X(f,t) = \sum_{m=1}^{M} x(m) w(m-t) e^{-2\pi i f \frac{m}{M}} \tag{2.10}$$

Note the similarity to equation 2.8, with the addition of a windowing function $w(m-t)$. X now has both a temporal dimension t as well as a frequency dimension f.

The name of the representation $X(f,t)$ resulting from a time-frequency transformation depends on the method applied, but for the sake of simplicity we have opted to call this representation a spectrogram in this work regardless of the method used. When looking at a spectrogram, the spectral envelope is depicted along the vertical axis whereas the time envelope can be found along the horizontal axis. The actual envelope shape is indicated by the variation in amplitude, which is displayed as the color at a certain time-frequency position.

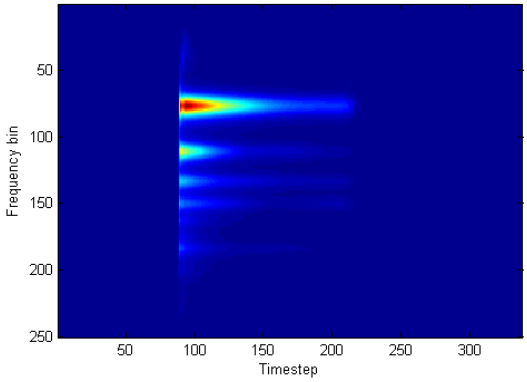

Figure 2.10: Single piano tone in the time-frequency domain. The color value indicates the amplitude at that timestep and frequency bin combination, ranging from low amplitude (dark blue) to high amplitude (red)

The time windows are generally shifted through time with each window partially overlapping the previous one. This avoids spectral leakage, which is when energy is observed at a certain frequency that is not present in the original audio. This is caused by the representation of the signal in a discrete number of samples. The effect of leakage is reduced by using overlapping windows and by picking a window shape that is smooth rather than rectangular, such as the Hanning or Hamming windows.

Window size

Determining the size of the window in which a frequency transformation is performed is an important aspect of a time-frequency transformation. We have already stressed the importance of choosing a sampling rate high enough to retain the important frequencies, those frequencies that belong to pitched music notes and their harmonic overtones.

The window size in a time-frequency transformation determines the trade-off between a high frequency resolution (a small number of frequencies in each bin) and a high time resolution (a small number of time samples in a timestep). Increasing the window size will include periodic signals with a low frequency, which take longer to complete a cycle. A smaller window will have samples at a faster interval and therefore will have a better approximation of the start and end times of a note. If we assume that we have a note that starts at a certain time window of 10ms size, there is a 10ms range in which the exact note start time is located. If the window size is 100ms, the note can have its starting point anywhere in this 100ms window, giving a less accurate indication of the exact starting position.

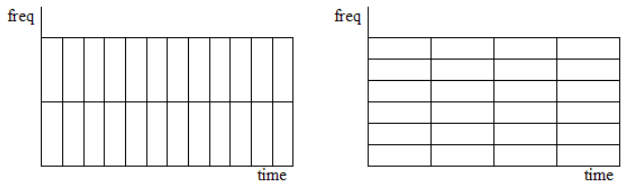

Figure 2.11: Trade-off between high frequency or high temporal resolution. Left: small window size with high temporal and low frequency resolution. Right: large window size with low temporal and high frequency resolution

2.2.4 Filter banks

A time-frequency representation of a signal can also be obtained using filter banks. A filter bank is a collection of filters, in this case *band pass* filters, which each allow only a certain frequency range to pass through. By introducing a set of these filters, each allowing a small frequency range to pass, we obtain the amount of energy present in each of these frequency ranges. The original signal, which spans the entire sampled frequency range, is then decomposed into a number of partial signals that each span a part of this frequency range.

The filters can be positioned linearly, meaning that each filter has the same bandwidth or frequency range size and are placed equally far apart. Alternatively, the bandwidth and spacing of the filters may be allowed to change throughout the frequency range, resulting in a higher level of frequency detail in the filters that have a smaller bandwidth and a lower level of detail in the filters with larger bandwidth. While it is possible to apply small bandwidth filters throughout the frequency range, this is usually not desired as it results in high dimensionality of the frequency axis in the time-frequency spectrogram, reducing performance of analysis using this representation.

In many cases lowering the frequency detail will not significantly decrease the quality of the resulting time-frequency data as long as the detail is high enough to capture the different key frequencies. In our case, this means for example that each fundamental frequency and overtone should be represented by a separate filter so that the difference between these frequencies remains clearly visible in the time-frequency domain. As a result, high frequency detail is especially required in the lower frequency range as the distance between fundamental frequencies in this region is relatively small. The edges of each frequency band are usually aligned to adjacent edges so that the filters line up exactly without overlapping. The bandwidth is defined as:

$$BW = f_{ch} - f_{cl} \tag{2.11}$$

Where f_{ch} is the upper cutoff frequency, the highest frequency included in the range that is allowed to pass the filter and f_{cl} is the lower cutoff frequency. The center frequency c_f is the frequency exactly in the middle between the upper and lower cutoff frequencies. Because the cutoff frequencies and as a result the bandwidth and center frequency can be determined for each filter individually, it is possible to have a different bandwidth for each filter resulting in a non-linear frequency scale.

A simple example of a non-linear scale is the *octave scale*, which as the name suggests has a filter for each octave. The frequency ranges could be:

Pitch range	Frequency range	Bandwidth
A1-G2	27.5-100Hz	72.5Hz
G#2-G3	100-200Hz	100Hz
G#3-G4	200-400Hz	200Hz
G#4-G5	400-800Hz	400Hz
G#5-G6	800-1600Hz	800Hz
G#6-G7	1600-3200Hz	1600Hz
G#7-C8	3200-8000Hz	4800Hz

Table 2.1: Octave scale frequency ranges

If we then derive features or information from each filter individually, each octave will have the same weight in this process even though the bandwidth of the higher octaves is much higher. In a set of linearly spaced filters, the higher octaves would span a much larger amount of filters because the frequencies of the notes contained in that octave are much further apart. If the same information is then derived from each of the filters, the higher octaves that have more filters will have a bigger influence than the lower octaves.

Analogous to the wavelet and short-time Fourier transformation, filter banks will also result in a time-frequency representation. The main difference is the domain in which the techniques operate. While the STFT splits the input signal up into time segments, performing Fourier analysis on each segment, the filters of a filter bank divide the signal into frequency segments instead. Wavelets are time-frequency entities as they can be shifted in both directions; by allowing a number of such time-frequency wavelets to describe the original input signal we can obtain time-frequency information.

The same principle of frequency resolution encountered when picking a window size for the Fourier transform applies to choosing the number of band-pass filters and their bandwidths, where a large bandwidth at the lower frequency range may prevent the frequency representation from showing the difference between a A0 note at 27.5 Hz and a A# note at 29.13 Hz. The difference between pitch values will be lost if both frequencies related to these values fall within the same filter.

2.2.5 Wavelets

As an alternative to the short-time Fourier transform, the *wavelet transform* or wavelet decomposition has been proposed to obtain a time-frequency representation. Although the wavelet transform is not used in this thesis project, we would like to briefly explain the technique and compare it to the short-time Fourier transform.

Wavelet analysis is similar to its Fourier counterpart in that it attempts to superimpose certain functions to approximate a signal $x(m)$. In Fourier analysis, sinusoids are used which stretch out to infinity and therefore do not have any aspect of time or duration. This property particularly challenges Fourier transforms when applied to signals containing sharp amplitude spikes. Moreover, we have to resort to techniques such as the STFT to get combined time-frequency information while temporal information is already included in the wavelet transform.

Wavelet analysis uses a mother wavelet, which can be any of a number of functions, instead of sinusoids. This mother wavelet is scaled and transformed similarly to how the sinusoids are controlled by phase, amplitude and frequency in Fourier analysis. Because wavelets *do* have temporal information, they can be contracted for high temporal resolution or stretched for high frequency resolution, similar to changing the window size in STFT. Wavelets however can be placed at different locations in time and in the frequency range to cover the entire spectrogram.

A mother wavelet is also known as a *basis function*, which has the property of being able to reconstruct any signal. This is best illustrated using a comparison with vector theory, where a linear combination of the orthogonal vectors (1, 0) and (0, 1) can be used to describe any position in 2D space. The same principle can be applied to functions, where certain sinusoids or mother wavelets can be combined to approximate any complex signal $x(m)$. These basis functions are orthogonal if their inner product is zero. *Scale-varying* basis functions can be divided into several pieces, where the amount of subdivisions determines the *resolution* or *scale* of the wavelet [11]. The mother wavelet is then positioned using a scaling function, defined as:

$$\psi_{(s,l)}(m) = 2^{-\frac{s}{2}}\psi(2^{-s}m - l) \tag{2.12}$$

Where ψ is the function used as basis, s is a scale index that determines the wavelet width in powers of 2 and l is an integer to position the wavelet in time. The scale s is defined as $\frac{1}{\omega}$ where $\omega = 2\pi f$ is the frequency parameter from STFT. This means that a high scale results in a low frequency. The formula above allows us to construct any number of wavelets from the mother wavelet by changing the scale and position. The basis can be compared to the sinusoid defined in 2.5, a number of which can be superimposed to reconstruct a signal.

To summarize, wavelets and the Fourier transform are similar in various ways, for example how the inverse is generated and how the basis functions are localized in frequency. The main difference is that wavelets are also localized in space while sinusoids are not. This means that the windows that are used in a STFT by default will give the same time-frequency resolution at all points of the spectrum, while the wavelets can be scaled and translated through space to provide a different resolutions. For example, there can be short high-frequency wavelets and gradually longer lower-frequency ones. This will result in high temporal resolution in the upper frequency range and a high frequency resolution in the lower frequency range, which is a realistic choice as the fundamental frequencies are further apart in the upper frequency range allowing a lower frequency resolution. There is an infinite number of possible functions to be used as basis functions for wavelet analysis, similar ones often clustered into families.

There are several different wavelet implementations such as the continuous, fast and discrete wavelet transforms. As the methods used in this project are all based on filter banks and STFT, we will not discuss these implementations in detail.

2.3 Time-frequency scales

We have already briefly seen how the windows of time and frequency in the STFT are equal in size throughout the entire time and frequency ranges. This is known as a *linear* time-frequency scale, as the center frequency of each subsequent window along the frequency range increases linearly. Alternative scales can be used to obtain different types of information. For example, when analyzing a musical piece that has been created by humans and is listened to by humans, it makes sense to examine the musical signal in a time-frequency scale that is based upon the way in which our hearing system perceives this signal.

There is a difference between the amplitude of a sound and what we perceive as the loudness of this sound. If two sounds of similar frequency are observed simultaneously, their combined loudness will be lower than if these sounds would have frequencies that are further apart. In this case, they are said to be within the same *critical band*, which means that they are observed by the same nerve endings in our auditory system. These nerve endings reside within the basilar membrane, where each group responds to a certain frequency range. The membrane therefore helps us to tell the difference between vibrations of different frequencies. The width of the critical band is non-linear and increases in the higher frequency range.

Including this knowledge of the human auditory system and the critical band provides us with the loudness at each frequency range as it is perceived by humans, instead of the objective measured amplitude in the linear case. Because in this particular situation we are examining music with the goal of extracting data that is useful for humans and because this music was generated by humans in the first place, we can obtain a more reliable estimation of the perceived loudness using a time-frequency scale that represents the human auditory system.

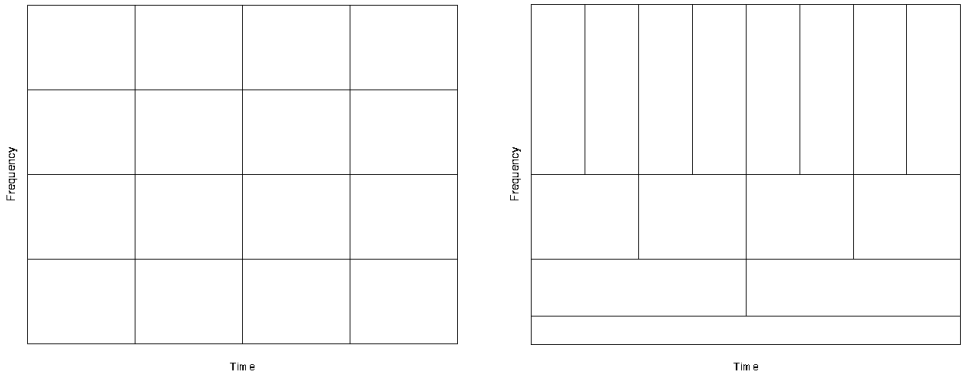

Figure 2.12: Difference between linear and logarithmic time-frequency scales. Left: linear. Right: logarithmic.

As an alternative, *logarithmic* windows have been proposed with a higher frequency resolution in the lower frequency range and a higher temporal resolution in the upper frequency range. This scale is said to be more like human perception, mostly in the higher frequencies, and the nature of frequencies in the musical scale, which are also spaced logarithmically [37]. The frequency difference between two successive pitch values increases for higher pitches, as shown in figure 2.2. The octave scale presented in table 2.1 is another scale that is useful for treating each octave as a single entity.

The Equivalent Rectangular Bandwidth (ERB) scale [15] is based on research into the frequency response bandwidths of the human auditory system. The scale converts between a frequency f and ERBs that express frequencies in terms of how they are observed by human hearing. The idea behind ERBs is that each ERB value corresponds to a certain position on the basilar membrane. The different positions along the membrane can be seen as auditory band-pass filters that are spaced non-linearly with varying bandwidths. The ERB scale attempts to model the frequency ranges and positions of these filters in our basilar membrane. Conversion from Hz to ERB is done using:

$$21.4 \times \log 10(0.00437 * f + 1) \tag{2.13}$$

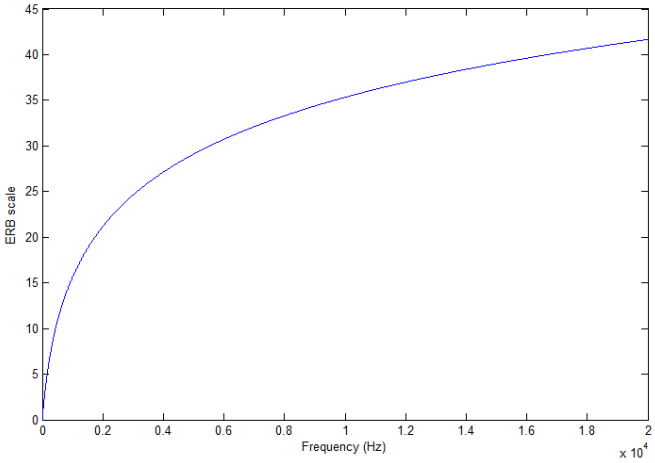

Figure 2.13: ERB scale

To get an ERB scale time-frequency representation, a number of filters can be spaced linearly along the ERB scale, which as seen in figure 2.13 results in high frequency resolution in the lower range and high temporal resolution in the upper range. The bandwidth for each filter can then be derived from their center frequencies:

$$24.7(0.00437f_c + 1) \tag{2.14}$$

Different frequency scales can be implemented using STFT, by keeping the frequency ranges small enough so that they can be combined to form a new scale. Filter banks can be spaced at different intervals to create different scales. Wavelets are derived from a mother wavelet and using the parameters can be positioned individually along the frequency range. The center frequency and bandwidth can therefore be determined separately for each wavelet.

Chapter 3

Multiple fundamental frequency estimation

In this chapter we attempt to highlight the different techniques and methods used to perform fundamental frequency estimation on polyphonic recordings. The non-negative matrix factorization technique in particular is explained in detail, as this technique is used in our work. For an up-to-date overview of current approaches and results, the reader is referred to the MIREX website[1] where results for a number of challenges related to music information retrieval have been posted along with papers describing their technical implementations.

3.1 Overview of frequency estimation techniques

Bay et al. in [2] provide an overview of several techniques for frequency estimation that were submitted to MIREX in 2007 and 2008. They state that all entries work with time-frequency domain information, most of which are based on a Fourier transformation. Two categories of techniques are defined based on how the interaction between multiple sources is handled. The first category is *iterative estimation*, where the frequency that is mostly active is detected and then removed or masked. This can be implemented for example by using comb filters in the time domain, or by using binary masks in the frequency domain. Interaction between sources in this case is handled by first identifying all of the spectral information belonging to one source and then removing this information so that only the structure from the other sources remains. This process of elimination is repeated until a certain termination criterion is met, for example when the amount of energy left after removing the detected notes falls below a certain threshold. A well-known frequency estimation method by Klapuri [19] uses iterative estimation and removal. The main weakness of iterative methods is that one bad elimination can result in poor overall results due to successive errors.

The second category is *joint estimation*, which tries to find a *combination* of frequencies that best fits the musical piece. Yeh [42] reasons that joint estimation techniques should be able to handle the interaction between sources more reliably, though at a higher computational cost than iterative techniques. Joint estimation is achieved by combining the expected spectral envelopes of certain notes and then finding the combination that best matches the envelopes present in the musical piece. This can be achieved either by comparing the envelopes and deriving an error measure, or by removing the expected envelopes from the musical piece and examining the residual.

Frequency candidates are often found by detecting amplitude peaks at certain time-frequency points. Because these peaks can be caused by overtones or noise, constraints or some form of analysis are usually applied to attempt to distinguish between peaks belonging to actual note activity and peaks that are caused by noise.

The paper by Bay et al. [2] concludes that the different approaches and implementations are competitive and that it is therefore difficult to pin-point a method that performs best for frequency estimation. They do suspect methods to benefit from the inclusion of some form of note tracking or other temporal information as well as timbre or instrument tracking. The inclusion of a source detection or separation step appears in the work of research groups

[1] http://www.music-ir.org/mirex/

such as Yeh and Roebel [43]. Yeh in [42] has evaluated the use of monophonic frequency estimation techniques for the task of polyphonic estimation. His evaluation showed that most monophonic methods for frequency estimation do not translate well to the polyphonic case.

A subset of the joint frequency estimation techniques is defined by *model-based* approaches. These approaches use a generated model to describe the audio spectrum. Knowledge of the model then results in knowledge of how the audio spectrum was originally created. An example of such a model is proposed by Kameoka et al. [17], where sound source models are generated using mixtures of Gaussians, a combination of which can be used to represent the spectrogram of the input signal. From this model they can derive not only the fundamental frequencies, but also properties such as onset and duration. A method similar to the Expectation-Maximization (EM) algorithm is used to iteratively improve the frequency estimation and source models.

3.1.1 Challenges in polyphonic music

In [26], Ibáñez describes three complex models that influence polyphonic music and do not occur in monophonic versions. The first is the source model, which states that the number of sources and their properties are unknown. The number of notes at each point in time is therefore also not known, which is why the process of determining the sources is part of several frequency estimation methods. For monophonic music, the number of sources and number of simultaneous notes will always be at most one, so there will either be one active frequency or no frequency at all.

The second model is the noise model, which is present in monophonic music but will be much more complex in the polyphonic case. For this reason, a form of noise suppression or removal can be considered before performing the frequency estimation.

The final model is the interaction between the different sources, where harmonic frequencies may overlap and cause energy from multiple notes in the same frequencies to be combined, increasing the complexity of the task of determining the combination of notes that was used to create the total spectral envelope. While the source model describes how in polyphonic music the number of frequencies we can expect is unknown, the interaction model considers the loss of information due to the interaction of these sources, where several notes sounding at once influence the resulting spectral envelopes.

As the next sections will hopefully show, these models are considered in our choice of the model-based non-negative matrix factorization approach.

3.2 Non-negative matrix factorization

A well-known model-based technique is non-negative matrix factorization (NMF), introduced by Lee and Seung [20]. These model-based techniques attempt to reconstruct a series of measurements, in this case spectral envelopes at sampled points in time, by combining a set of basic spectral envelopes. The envelopes in the set will generally describe a basic structure, belonging for example to a single note. In music containing multiple voices the sampled spectral envelopes will be more complex as they may be generated by several notes sounding at once. However, the idea behind NMF and related techniques is that a combination of basic envelopes from the set can be combined to recreate an envelope that is generated by several notes.

Non-negative matrix factorization is similar to independent component analysis (ICA), the most important difference being the constraints that are imposed on the model. ICA attempts to maximize statistical independence, while NMF is especially suited for data that by definition cannot have negative values. Audio data fits this description due to the additive nature of how music is composed out of a combination of notes, each of these notes consisting of the addition of energy at various harmonic frequencies.

Experiments by Virtanen [39] indicate a superior performance of NMF compared to a technique related to ICA for the task of source separation. Other related techniques include principal component analysis (PCA) and single value decomposition (SVD), the main difference again being the imposed constraints. Furthermore, NMF apparently favors a sparse and localized representation [20]. We focus on NMF due to its proven use in the analysis of audio signals. Besides for the feature extraction task that we are now considering, this family of techniques may

also be used to compress data by reducing its dimensionality and for denoising. Carron on his website[2] attempts to list and categorize the different matrix factorization techniques that are in existence.

The goal of NMF, applied to our frequency estimation task, is to recreate the spectrogram of a musical piece using a combination of spectral envelopes, each describing the structure of a single note. The combination of single note envelopes should, if the same notes from the original musical piece are used, result in the combined spectral envelope present in the musical piece. This means that we now have an estimation of the notes that were played at any point in time because if we take what we expect those notes to look like and combine the resulting spectral envelopes, we are able to obtain yet again the complete time and frequency information of the musical piece. Intuitively, we can compare this to picking up all possible instruments and then playing possible combinations of notes to hear if the resulting sound is similar to what is heard in the original recording.

In fact, the approach of model-based techniques can be compared to how Fourier analysis is used in section 2.2.1 to extract frequency information from the audio signal. In the case of the Fourier transform, sinusoids are combined to match the time domain signal. Because the frequencies of the sine waves are known, we can derive the energy present in each of those frequencies. The same principle applies to model-based approaches, where now we are attempting to reconstruct a time-frequency domain signal rather than a time domain signal. Instead of sinusoids, in the case of NMF we use a set of spectral envelopes.

As an example, we know roughly what a C4 note looks like because we expect a certain activity in its fundamental frequency and the harmonic overtones. We can store this expected spectral shape as one entry in our set of spectral envelopes. Note that these envelopes themselves have no sense of time, just a distribution of energy at certain frequencies. Then, if in the reconstruction process it turns out that this specific spectral envelope is used at a certain point in time, we can conclude that it is possible that a C4 note was active at that time.

The result of activating our spectral envelopes at certain points in time is a spectrogram, where the activations provide the temporal information and the spectral envelopes we defined provide the frequency information. Together they contribute to the amplitude. The resulting spectrogram should be similar to the one obtained from the musical piece after a time-frequency transformation.

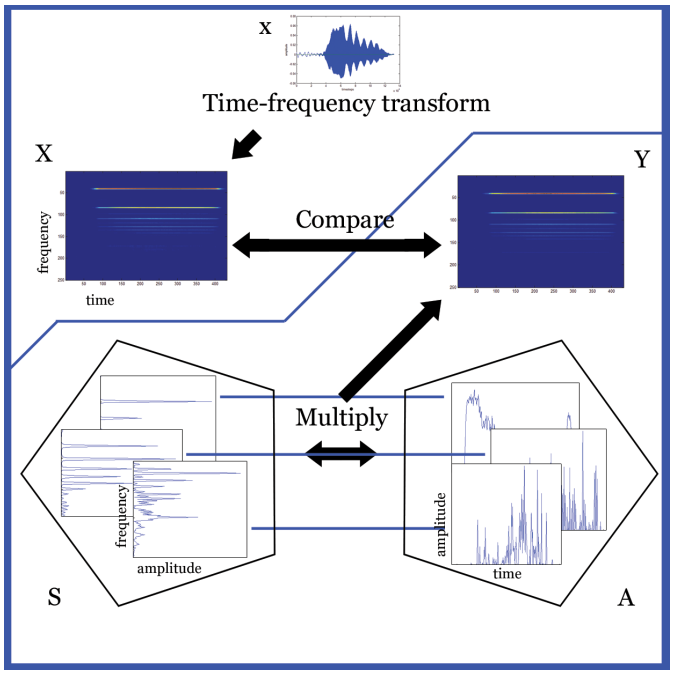

Figure 3.1: Overview of non-negative matrix factorization

[2]https://sites.google.com/site/igorcarron2/matrixfactorizations

In short, non-negative matrix factorization when applied to an audio signal is about reconstructing the spectrogram of this signal by activating a combination of known spectral envelopes at each time interval. These activations can be seen as the temporal envelope of each individual spectral envelope. Because we introduce these basic spectral envelopes ourselves, we can store and use properties of these envelopes such as their pitch values. As we know which envelopes are used in the reconstruction process, we now also know the pitch values that belong to the envelopes, indicating that these pitched notes may have been played in the original musical piece.

The basic factorization process of NMF can be described as:

$$Y = A \times S \tag{3.1}$$

Where Y is the reconstructed spectrogram and A contains the activations of each entry in the set of spectral envelopes S at each timestep. I is the number of spectral envelopes in the set S. For example, if we define one envelope for each of the 88 possible pitch values, then I will be 88 as S contains 88 envelopes. If F is the number of frequency bins and M is the number of timesteps, then Y will have the same dimensionality as the time-frequency representation of the entire musical piece, which is $F \times M$. S will be $F \times I$, describing I spectral envelopes. A contains the activations of each entry in S through time, therefore it is $I \times M$ in size. As the subset of A and S that is actually used for a certain reconstruction (not all envelopes in S are necessarily used) will generally be much smaller than the size of Y, a form of dimensionality reduction takes place.

Upon reconstructing an audio signal using a set of spectral envelopes, we do not know the set of activations - or temporal envelopes - that results in the closest approximation of the original signal. Just imagine the number of possible instrument and note combinations that can be combined to form a musical piece.

3.2.1 Deriving the activations

The high dimensionality of A leaves us with a set of unknown values, which cannot simply be evaluated. Instead, we initialize A with all entries set to unity, then derive the spectrogram Y using known S and assumed A. This spectrogram can be compared to the time-frequency representation of the musical piece, X, to obtain an error measure. Given this error measure, A can be altered after which Y can be constructed and compared to X again. By repeating this process while updating A in a certain way, the difference between Y and X will decrease until it reaches a point of convergence where the difference will remain stable. At this point the reconstruction quality will have reached an optimum.

First, let us examine how Y and X are compared, using what is known as a distortion measure:

$$d(X_{ft}|Y_{ft}) = \frac{1}{\beta(\beta-1)}(X_{ft}^{\beta} + (\beta-1)Y_{ft}^{\beta} - \beta X_{ft}Y_{ft}^{\beta-1}) \tag{3.2}$$

Which results in a scalar value as a measure of the difference between reconstruction Y and original signal X. It can be regarded as the reconstruction error. We have reached convergence once the change in the distortion measure between two consecutive iterations falls below a certain threshold.

β determines the choice within the family of β-divergences, explained in [5]. This family contains a number of different distortion measures, each providing different results. Among the divergences are the Euclidean distance at $\beta = 2$, the Kullback-Leibler divergence ($\beta \to 1$) and Itakura-Saito divergence ($\beta \to 0$). Without describing each of the divergence measures in detail, we would like to point out that each measure defines a way to describe the similarity between, in this case, two spectrograms. In this work we have used $\beta = 0.5$ as this was found to be the optimal value for pitch detection by Vincent (figure 6 in [38]).

This process of deriving the value of an unknown set A by initializing it to certain values and evaluating Y as if A is already correct, then iteratively changing A to improve the quality of Y can be implemented using a method such as estimation-maximization (EM) or, in our case, a multiplicative update rule [38]. The multiplicative rule for updating A is:

$$A_{it} = A_{it}\frac{\sum_{f=1}^{F} S_{if}Y_{ft}^{\beta-2}X_{ft}}{\sum_{f=1}^{F} S_{if}Y_{ft}^{\beta-1}} \tag{3.3}$$

The two equations are applied iteratively. First, A is determined using equation 3.3. Then, it is multiplied with the envelopes in S to obtain a new reconstruction Y. The resulting spectrogram is then compared to X using equation 3.2 to see if a point of convergence is reached. If convergence was not reached, the newly generated Y can then be used again to update A using equation 3.3.

We have now explained how the activations A are obtained. This leaves us with the spectral envelopes contained in S, which up to this point have been assumed to be fixed spectral envelopes. However, these envelopes still need to be constructed.

3.2.2 Deriving the envelopes

There are two different methods for building a set of spectral envelopes. It is possible to define the spectral envelopes before using NMF, for example by deriving them from a set of isolated note recordings. This process is explained in detail in section 3.2.6.

Alternatively, the envelopes can be generated from the musical piece itself in a way similar to how A is determined, during the NMF process. Instead of applying an update rule only to improve A, the same principle will be applied to S so that S iteratively changes to best match the recording as well. The update rule for S is then similar to the one used with A:

$$S_{if} = S_{if} \frac{\sum_{t=1}^{T} A_{it} Y_{ft}^{\beta-2} X_{ft}}{\sum_{t=1}^{T} A_{it} Y_{ft}^{\beta-1}} \tag{3.4}$$

If this update rule for S is used, instead of initializing A with unitary values we now randomly draw values from a unitary distribution for both A and S [38]. Note that the spectral envelopes in S can now take any shape as they are initialized with random values and are allowed to change freely by applying the update rules. For A this effect was acceptable, as we want A to freely explore possible combinations of spectral envelopes because we do not have any knowledge of the number of voices or the notes played a priori.

Because the spectral envelopes are not generated and stored beforehand in this situation, we also need to pick I, the number of envelopes in S. Ideally, we want each envelope to describe exactly the spectral structure of a single note, or at least pitch value. However, we do not know which pitch values and instruments are present in the musical piece. If I is set too high, then each envelope will describe only a small part of a note as several entries in S will be combined to form the final note structure. Because S is updated without constraints, it is also possible for one envelope to describe parts of the structures of two different notes. Setting the number of envelopes I too low will result in one envelope describing more than the structure of a single note, a property that is also not desired because then we still do not know the pitch value of each note.

We cannot simply define 88 envelopes, one for each pitch value, because not all pitch values need to occur in a piece of music. Furthermore, if S is allowed to update freely the frequency of each envelope will have to be estimated after running NMF in order to determine which frequencies are used in the reconstruction. These negative consequences of allowing S to update in the NMF process can be limited by constraining the way the update rules are applied. The next sections will discuss harmonic and sparseness constraints, two examples that guide the spectral envelope generation so that each envelope hopefully describes the spectral structure of a single note.

3.2.3 Sparse NMF

If we set the number of spectral envelopes contained within S, the number I, too high and then let S update freely, then the spectral envelopes may only describe a small part of the envelope of a note. Ideally, we would like each entry in S to describe the complete envelope of a note so that we can tell when this note is active in the musical piece. Introducing sparseness is a way to improve the completeness of the envelopes in S by using as little of these envelopes as possible in the reconstruction process. If it is possible to do the reconstruction with two envelopes from S instead of four, the method will prefer this option. This automatically should mean that the envelopes in S will be constructed to contain as much spectral information as possible.

Variations of NMF that enforce a sparseness constraint onto A stimulate for each timestep the use of as little envelopes as possible. As a result, the spectral envelopes, if S is updated during NMF, are expected to become more *complete*, containing more of the harmonic structure of a note, so that less of these envelopes are needed to reconstruct a part of the spectrogram. Ideally, the spectral envelopes will each start to describe the structure of a single note. It is possible to apply the same sparseness constraint to the envelopes in S, however in our case of harmonic audio signals we generally do not want to constrain S in this way as we want the envelopes to be as complete as possible in describing a note.

Sparse NMF is proposed by Hoyer [13] and is implemented by altering the update rules and enforcing an $L1$ norm by projecting each row of A onto a hyperplane in $L1$. L-norms are measures of length in a certain dimension, where $L2$ for example is the Euclidean norm or magnitude.

The idea behind sparseness is expected to match well with the concept of pitch values and their spectral structures, as ideally we would like to reconstruct the spectrogram of a recording using as little notes as possible, so that notes that are harmonically related to the actual notes of the musical piece, such as the notes that are one octave higher, are not used. Examples of various extensions of NMF, including a sparse version, are provided by Hoyer on his website[3].

$L0$ sparseness

A different type of sparseness constraint is $L0$ sparseness, which is more of a hard constraint than the $L1$ sparseness discussed previously. Where $L1$ sparseness stimulated the use of less spectral envelopes, $L0$ sparseness *enforces* a maximum to the number of envelopes that is allowed to be active at one time. This is done by constraining the amount of allowed non-zero entries in the activations A, a process called *sparse coding*. Obtaining the optimal sparse A that best reconstructs X by sparse coding is NP-hard, therefore an approximate solution is derived. An example of a sparse coding technique provided in source code by Peharz et al. [31] is non-negative matching pursuit (NMP), a modified version of the orthogonal matching pursuit (OMP) algorithm. The sparse coding step is then alternated with the regular NMF update rules to ensure that the distance between the model Y and the input spectrogram X is reduced while maintaining sparseness constraints.

This constraint is applied to the activations in A, therefore it can also be used in combination with a previously generated set of envelopes. So if S remains fixed and is not updated during the factorization, using $L0$ sparseness we can still ensure that out of those envelopes only a certain amount is active at each timestep. Imagine that we have constructed S so that we have 88 envelopes, each describing one pitch value. If we allow only two of the envelopes to be active at the same time, then only at most two pitch values will be returned at each point in time.

Peharz graciously provides source code and demos for his implementation of $L0$ sparse NMF[4], which has helped in our experiments using this technique.

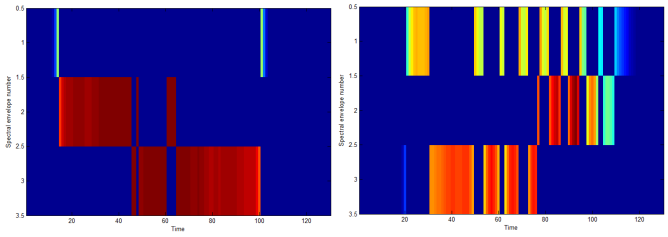

Figure 3.2: Natural modelling in A of note phases using $L0$ sparseness. Left: similar to attack-decay-sustain-release. Right: vibrato effect of a flute

The above figure shows how applying $L0$ sparseness to a single note recording stimulates the use of a separate spectral envelope for each characteristic phase of a note. The number of spectral envelopes is limited to three, where only one of these envelopes is allowed to be active at one time.

[3] http://www.cs.helsinki.fi/u/phoyer/software.html
[4] http://www3.spsc.tugraz.at/people/robert-peharz/

3.2.4 Source-filter model

The source-filter model is introduced to non-negative matrix factorization by Virtanen and Klapuri in [40]. It had been used previously in speech coding and sound synthesis. Virtanen and Klapuri point out that regular NMF requires a spectral envelope for each pitch value of each instrument. Furthermore, there is a large number of free parameters, which as we have pointed out may result in unpredictable envelopes in S. To handle these two disadvantages of the regular, unconstrained NMF, the spectral envelopes are defined as combinations of sources or excitations and filters.

When dealing with musical instrument sounds, the source can be seen as the parts of the instrument that produce sound, such as the strings of a guitar. The filter can be seen as the resonance structure that colors the sound. Source is influenced by pitch while the filter is determined by timbre. Ideally, this method should therefore not only provide information about the pitch values that are active but should also be able to perform source separation.

$$S = U \times R \tag{3.5}$$

Where U are sources and R are filters. The reconstructed spectrogram Y is then generated as:

$$Y = A \times U \times R \tag{3.6}$$

Where A, U and R are all initialized with random positive values and updated using multiplicative update rules as with regular NMF. However, three updates are now required: one for A, U and R. Converting the update rules of Virtanen and Klapuri to our version, which differs slightly from the one used in their work, results in:

$$A_{ijt} = A_{ijt} \frac{\sum_{f=1}^{F} U_{if} R_{jf} Y_{ft}^{\beta-2} X_{ft}}{\sum_{f=1}^{F} U_{if} R_{jf} Y_{ft}^{\beta-1}} \tag{3.7}$$

$$U_{if} = U_{if} \frac{\sum_{t=1}^{M} \sum_{j=1}^{J} A_{ijt} R_{jf} Y_{ft}^{\beta-2} X_{ft}}{\sum_{t=1}^{M} \sum_{j=1}^{J} A_{ijt} R_{jf} Y_{ft}^{\beta-1}} \tag{3.8}$$

$$R_{jf} = R_{jf} \frac{\sum_{t=1}^{M} \sum_{i=1}^{I} A_{ijt} U_{if} Y_{ft}^{\beta-2} X_{ft}}{\sum_{t=1}^{M} \sum_{i=1}^{I} A_{ijt} U_{if} Y_{ft}^{\beta-1}} \tag{3.9}$$

Where i now indicates the index of the source rather than the index of S and j is the index of the filter. Equation 3.7 is identical to what we have seen in the regular NMF update rule, with S replaced by the combination of source and filter.

Virtanen and Klapuri address shifting of envelopes throughout the frequency range and mention that the entire envelope, consisting of both source and filter information, in the regular NMF framework is shifted as a whole. Ideally, only the source would be translated to shift the pitch value of an envelope without influencing the filter characteristics. The source-filter model allows shifting of the source alone.

Heittola, Virtanen and Klapuri have later applied the source-filter model to instrument recognition [12].

3.2.5 Harmonic NMF

Vincent, Bertin and Badeau [38] propose a variation of regular non-negative matrix factorization that enforces harmonic and spectral smoothness constraints. They have been generous in providing the source code for this method[5], which has been useful in learning the details of matrix factorization and the imposed constraints.

Because we want S to contain spectral envelopes of musical notes, we can expect each of these envelopes to have a certain harmonic structure with a fundamental frequency and its overtones. Furthermore, the envelopes should be smooth as this is common to the envelopes of musical notes. These constraints are enforced by extending the way S is generated and updated.

The envelopes in this case are not fixed, they are updated as explained in 3.2.2. However, the uncertainty of choosing the correct number of envelopes I, contained in S, is solved by setting this number to 88, one for each pitch

[5] http://www.irisa.fr:/metiss/members/evincent/software

value, and by initializing each envelope with the spectral structure of one pitch value. For example, envelope and pitch number 40 with a fundamental frequency of 261.63 Hz will be initialized with energy only at the fundamental frequency and a number of harmonic overtone frequencies such as 523.25 Hz, 1046.50 Hz and so on.

The envelopes in S are constructed by defining a number of *narrowband spectra*, each narrowband spectrum describing a spectral envelope. These are then combined to generate the final envelope in S. The principle applied here is similar to how A and S together form time-frequency data, only in this case the narrowband spectra and their activations together form the spectral envelopes S. The equation therefore is similar to how Y is constructed:

$$S = E \times N \tag{3.10}$$

To illustrate, we show how S is initialized for the A3 note at 220 Hz:

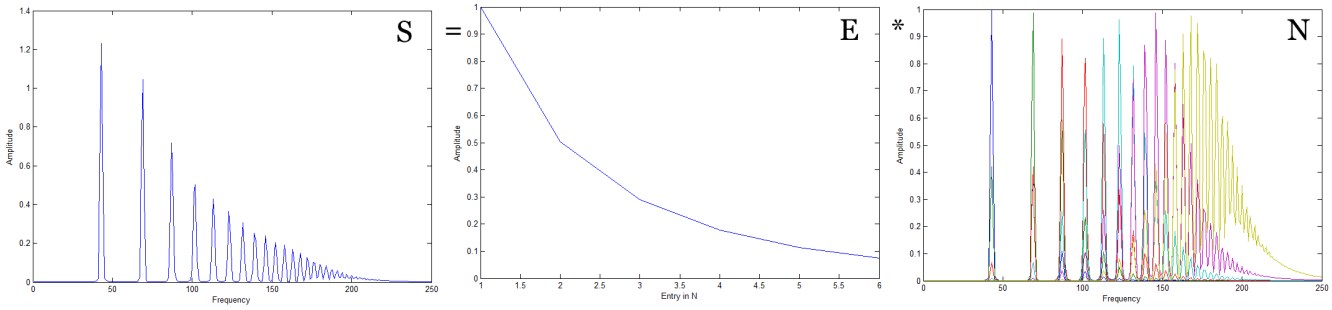

Figure 3.3: Generation of spectral envelope S for note A3 out of activations E of narrowband spectra N

S in this case is $F \times 88$ (note A3 or pitch number 37 is shown in the image). The number of structures in E and N depends on the pitch value. In this case there are 6, so E is 6×88 and N is now $F \times 6$. Each color in N is a different narrowband spectrum.

To define N a set of smaller spectral envelopes, each describing an overtone, are combined with certain weights to form the envelope of a certain pitch value:

$$N = W \times P \tag{3.11}$$

The process of building up N from a structure of overtones P with certain weights W for note A3 looks like:

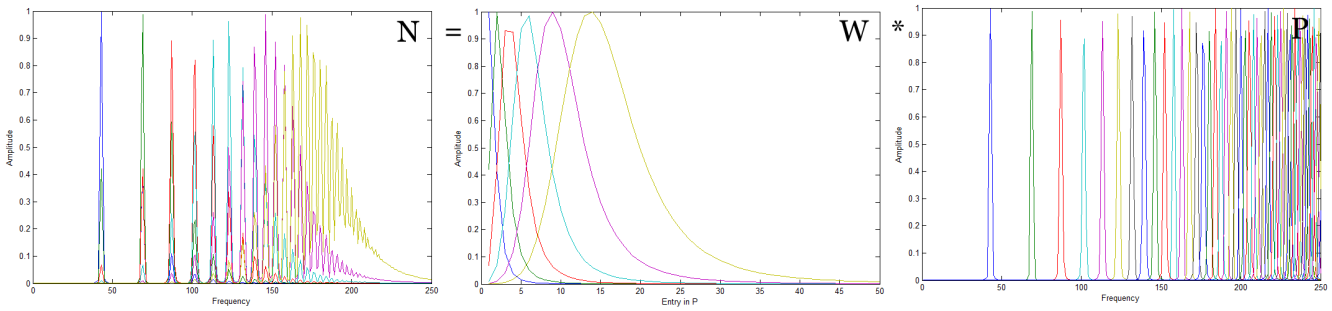

Figure 3.4: Generation of narrowband spectra N for note A3 out of weights W and overtone structures P

The number of entries or overtones in W and P, as in N, vary with pitch value. Each color in P represents a different overtone structure out of a total of in this case 50, where each structure describes one peak at one frequency. The in-between step of using N and E is required because the overtone structures in P only describe single overtones. If we were to generate S directly from P then the number of possible combinations would still be very large, comparable to allowing S to update freely. By first combining overtone structures from P into narrowband spectra in N, the process of generating S is limited to allow only those spectral envelopes that contain energy at a number of harmonic frequencies.

All that is allowed to change during the matrix factorization process is E, containing the weights of the narrowband spectra N that contribute to the envelopes S. Instead of an update rule for A and S, we now have update rules for A and E. The rule for A remains the same (equation 3.3), while the rule for E is similar to this rule as these values can also be seen as activations, only now in the frequency domain on structures N:

$$E_{ik} = E_{ik} \frac{\sum_{f=1}^{F} \sum_{t=1}^{T} A_{it} N_{kf} Y_{ft}^{\beta-2} X_{ft}}{\sum_{f=1}^{F} \sum_{t=1}^{T} A_{it} N_{kf} Y_{ft}^{\beta-1}} \tag{3.12}$$

This equation encodes the process of combining E and N to get S into the update rule. k is the number of narrowband spectra from E belonging to a certain pitch value in S. For example, for note A3 $k = 6$.

After updating E not only Y but also S will have to be recalculated so that A can be updated using the new S. While S is allowed to update based on the spectral structures found in the input signal X, these updates are limited to E and thereby are subject to the harmonic constraints imposed by the structures in N which are in turn based on the overtone structures P. The weights W are generated from a smooth function, enforcing a smoothness constraint.

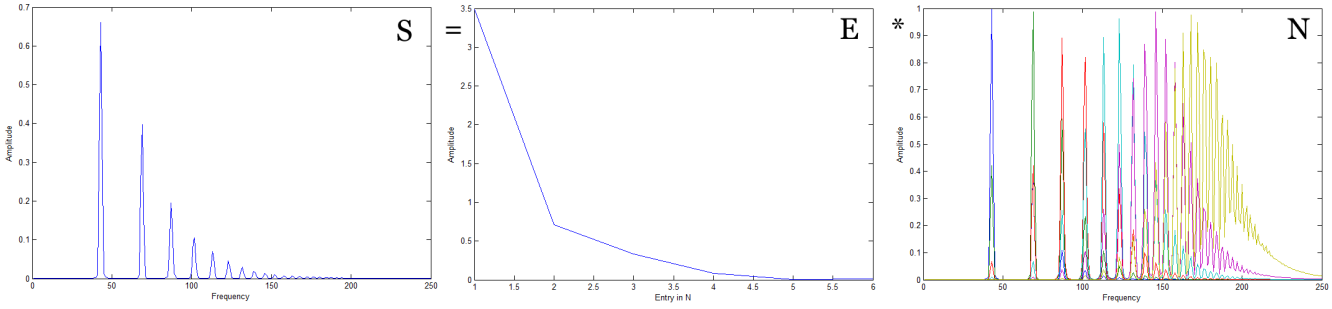

Figure 3.5: Spectral envelope S of note A3, after E has been updated in NMF. N remains the same.

3.2.6 Trained NMF

The spectral envelopes in S can be derived from a set of training data consisting of isolated note recordings, either by taking the frequency distribution of the entire note recording or by applying non-negative matrix factorization to these isolated recordings. Using NMF to obtain the envelope might be more robust to noise as the method will attempt to construct an envelope that best reproduces the note. It is then also possible to derive more than one envelope from a single note to capture its dynamics, for example by using $L0$-constrained NMF as we have shown in section 3.2.3. By learning the spectral envelopes from a set of recordings, harmonic structures are still imposed in a similar fashion to how this is done by the harmonic model in section 3.2.5. However, the constraints imposed by these examples can be regarded more as *soft* constraints where slightly inharmonic frequencies or missing frequencies are allowed as long as they occur in the set used for training.

Because we are dealing with isolated note recordings, the resulting envelopes will all describe the structure of a single pitch value. Therefore, we can use a standard implementation of NMF that updates both S and A without constraints. If we limit the number of allowed envelopes in S to one, this envelope will automatically describe the note. The resulting set of envelopes S can then be used in another NMF framework to recognize notes in musical recordings. Because the pitch value of each training note is known, the resulting pitch value active in the musical piece can be derived as well. The process of training envelopes using NMF is shown in figure 3.6.

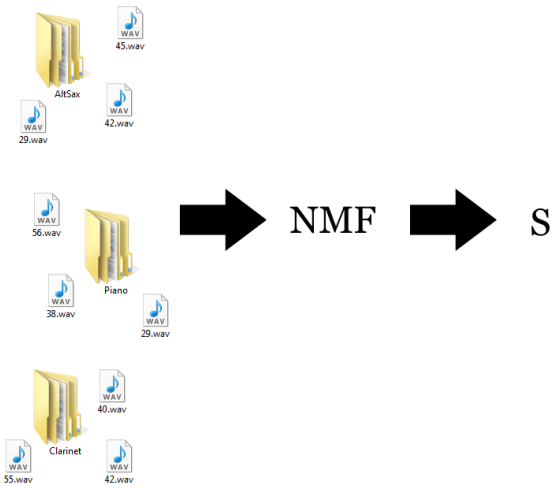

Figure 3.6: The process of learning the spectral envelopes from a set of isolated note recordings.

The only condition we have to satisfy in generating the set of envelopes is that the properties of the time-frequency transform such as the scale and window size are equal to the ones that will be used when later running NMF using the stored spectral envelopes.

3.2.7 Other variations of NMF

Many more variations of non-negative matrix factorization exist. A few examples that we have experimented with and will briefly address here include convolutive NMF and shift-invariant NMF. We will provide some basic information with references to related work so that further experiments in this direction can be pursued if desired.

Convolutive NMF [27] identifies *objects* instead of spectral envelopes. In the case of audio data, these objects can be seen as notes. S therefore now includes not only spectral envelopes but also their development through time as a sequence of several spectral envelopes is stored. The exact number of timesteps that should be stored can be set for all envelopes. This extension of NMF uses update rules that have been altered to work with the higher dimensionality of S.

Our preliminary experiments with S derived from a set of isolated notes indicate that it is difficult to determine the number of consecutive timesteps that should be stored in S because the notes in a musical piece are not necessarily

of similar length to the notes that were used for training. Our best results were obtained using 1 timestep, which reduces to basic NMF. Alternatives to convolutive NMF have been proposed by using a state model such as a hidden Markov model (HMM) to capture the different phases of attack-decay-sustain-release.

To avoid having to train or define a spectral envelope for each pitch value or small variation, it is possible to allow each entry of S to shift a certain number of frequency bins. A spectral envelope can then ben generated from a note with a certain pitch value which is then allowed to shift to adjacent pitch values while NMF is running by moving the envelope along the frequency range. We found this shift-invariant variation of NMF [33] to be computationally inefficient as for each iteration the envelopes in S will have to be shifted after they have been updated. As an alternative we have explored the possibility of training NMF on a subset of all possible pitch values and then shifting the spectral envelopes to cover the pitch values in between. These shifted envelopes are then stored in S along with the actual trained envelopes so that no shifting has to take place during NMF.

3.2.8 Handling polyphonic music

In section 3.1.1, we mentioned three models that influence the performance of frequency estimation in polyphonic music: source model, noise model and interaction model. Because non-negative matrix factorization is able to counter some of the negative consequences of these models, the technique can be considered for the task of frequency estimation. Matrix factorization essentially attempts to separate high-dimensional data, in our case the time-frequency representation of a musical piece, into smaller pieces. These smaller pieces can then be used to obtain information about the sources present in the musical piece. This process has been incorporated into the source-filter model described in section 3.2.4. The task of identifying the different sources of a recording is further explored in our experiments regarding the use of NMF for instrument recognition in section **??**.

Because we are working with the data used for a reconstruction rather than the actual time-frequency representation of the input signal, we can avoid the inclusion of noise by ensuring that the spectral envelopes used in the reconstruction contain as little noise as possible. This can be achieved by learning the envelopes from monophonic, single tone note recordings as we have done in section 3.2.6 or by constraining the shape of the envelopes to a pre-defined set of possible combinations as in the harmonic NMF (section 3.2.5). It is possible to include a separate noise dictionary similar to the spectral envelope dictionary S. This has previously been implemented for speech to isolate and remove noise.

The interaction between sources may result in incorrectly discovered frequencies or correct frequencies that are missed. Because NMF works with frequency structures in the form of spectral envelopes, rather than single frequency values, the effects of the interaction between sources should be reduced. The sparse nature of NMF is then expected to result in the best matching combination of pitched notes to explain the spectral envelope of a musical piece at any point in time.

3.3 Onset detection

There are several ways to implement an onset detection method. The most straight-forward way is to find peaks in the amplitude envelope of a sound, as the onset will generally have a sudden high amplitude compared to the rest of the lifespan of a note. However, similar to frequency estimation, there can be peaks that are incorrectly determined to be onset events due to effects such as noise. For this reason we introduce two methods of onset detection that have been used in this thesis project, one of which is based on frequency behaviour and the other on the spectral flux. Both methods have been implemented in code.

3.3.1 Frequency-based

Zhang and Ras in [44] propose a three-state model for the amplitude envelope of a sound, consisting of the transient state, quasi-steady state and decay state. Their work with timbre analysis focuses on distinguishing between the transient and quasi-steady states.

Instead of examining the amplitude envelope, the behaviour of the fundamental frequency is used to find the transition between the attack (transient state) and steady state, with the moment of transition marked by the onset. During the attack phase, the fundamental frequency is found to be unstable and very different from the steady state. Because the fundamental frequency is expected to stabilize in the steady state phase, Zhang and Ras examine a number of neighboring frames (each frame is 10ms). The time at which all neighboring frames have an identical fundamental frequency then marks the transition between the attack and the steady state. Three different numbers of neighbors are considered depending on the total length of the sound sample. A sound sample with a longer duration is required to have more frames with the same fundamental frequency before it is considered to be stable.

This method relies on knowledge of the fundamental frequency of a sound, so that the frequencies near the fundamental can be examined to determine if the frequency is stable. In the case of isolated notes, we usually know the fundamental frequency or it can be easily derived. For musical recordings, this task is non-trivial, however when working with NMF it is possible to combine all activations of envelopes belonging to a certain pitch value to get a *stream* of all activity at a certain pitch value throughout the entire musical piece.

Using the separation between activations and envelopes - the result of NMF - we are able to apply this form of frequency-based onset detection because we know the fundamental frequency belonging to each stream. However, especially when dealing with polyphonic combinations, unstable frequency behaviour may also occur at harmonic overtones. This could help in detecting the difference between harmonic components and actual different notes, as harmonic components are expected to have similar onset to their fundamental. However, it is also possible for this effect of multiple voices to cause several frequencies to be unstable, resulting in onsets that are not being detected or that have their location shifted as a result of activity at different frequencies.

A second effect that should be taken into account is vibrato, which causes an unstable frequency throughout the entire lifespan of a note, thereby challenging the performance of a frequency-based onset detection method.

3.3.2 Spectral flux

Instead of using peaks in amplitude, it is also possible to find onsets based upon peaks in spectral flux. Spectral flux indicates the change in the energy at each frequency from successive timesteps. This section is based on the information provided in the manual of the SV Mazurka plugin[6]. The basic form of spectral flux is:

$$SF(t) = |X(f,t) - X(f,t-1)|_2 \tag{3.13}$$

Where $|...|_2$ is the L-2 norm or Euclidean distance. Summation, the L-1 norm, can also be used. The equation can be further specified as:

$$SF(t) = (\sum_{f=1}^{F} |X(f,t) - X(f,t-1)|^2)^{1/2} \tag{3.14}$$

[6]http://sv.mazurka.org.uk/MzSpectralFlux/

There exist many variations to this standard spectral flux implementation. Important for onset detection is the positive spectral flux that only shows positive changes in frequency energy by extending the regular implementation with the positive half-wave rectifying function, resulting in an extension of equation 3.13:

$$SF+(t) = |\frac{X(f,t) - X(f,t-1)}{abs(X(f,t) - X(f,t-1))}|_2 \tag{3.15}$$

Spectral flux shows the changes of frequency over time, summed up over all frequencies. The resulting envelope therefore has length $M-1$. Finding peaks in this representation however will still result in incorrectly returned onset values. Some rules are therefore introduced to separate incorrectly returned onsets from correct onset peaks. First, the peak should be a local maximum. There should be no higher flux value in a certain window around the peak. This window in the SV Mazurka implementation is set to 30ms (3 timesteps in our case) in both directions.

The second rule is that the peak should be above an *exponential decay curve*, which is fitted to the spectral flux function. This decay function essentially ensures that onsets that are detected are far enough apart so that a note is able to actually have a steady-state phase. The decay curve is defined as:

$$G(t) = \max(SF(t), a \times G(t-1) + (1-a) \times SF(t)) \tag{3.16}$$

Where a is the exponential decay factor, a parameter that can be set by the user.

The third rule is related to the local mean of the flux function. A peak belonging to an onset is expected to be above the local mean of the flux plus a certain user-defined offset b. The mean is derived from a certain window around the detected peak. The authors of SV Mazurka apply a window of [-90ms, +30ms].

In short, first the spectral flux is calculated from the spectrogram of a sound, after which peaks can be derived from this spectral flux. Many peaks that do not belong to actual onsets may be found, therefore three rules are applied to reduce the number of incorrect peaks. Figure 3.7 shows the spectral flux and the rules applied to part of a single note spectrogram.

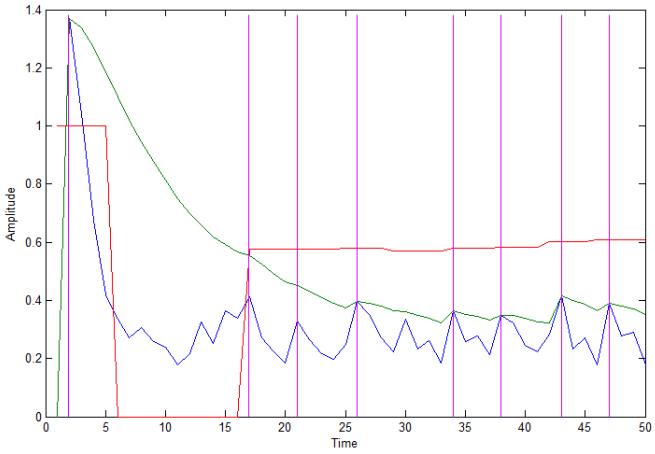

Figure 3.7: Onset detection results. Pink lines indicate peaks; only the first one is an actual onset as it is higher than the decay (green) and local mean (red)

Chapter 4

Instrument recognition

There is an essential difference between the task of instrument recognition and the task of frequency estimation. In frequency estimation, the properties that are used such as amplitude and frequency have a clear definition and are one-dimensional. Frequency can have one out of a set of possible absolute values. When dealing with instrument recognition, we want to capture the *character* or *timbre* of an instrument.

We try to determine features that describe subjective and emotional properties such as brightness. These are not one-dimensional values that can easily be compared. To describe the character of an instrument, several features need to be derived from the spectral and temporal properties. Just like the fundamental frequency, these features are low-level representations. The combination of these features will result in instrument descriptions that are multi-dimensional, requiring some form of classification to assign new instrument samples to known instrument descriptions.

The effect of timbre is said to depend on several properties [1]:

- The range between tonal and noiselike character
- The spectral envelope
- The time envelope
- The changes of spectral envelope and fundamental frequency
- The attack

These properties will play a part in deriving features from the time-frequency representation of notes. The *attack* will generally take up only a small amount of time. It can be very characteristic of an instrument, due to the physical differences between instrument types. While the attack of a piano is caused by hammering of strings, for guitars it will be plucking of strings. During the sustain phase, the amplitude and frequency will remain relatively stable. This phase will take up most of the duration of a musical note.

4.1 Machine learning

The process of recognition for both humans and computers is based on memory and learning. Therefore, for automatic recognition using computers some form of machine learning is required. Machine learning systems are able to learn the properties of different *classes* from a set of examples, after which they are able to assign new samples to a certain class based on the properties of these samples. Non-negative matrix factorization can also be seen as a form of machine learning, where the memory consists of the spectral envelopes in S.

There is a difference between supervised and unsupervised learning. In supervised learning, the samples that are used for training are labeled so that the possible classes are pre-defined and the task is then to assign new samples to the correct class. These labels are not present in unsupervised learning, where the machine learning system is simply provided with a number of samples for training, from which classes need to be derived automatically based

on the differences between the samples.

We have seen the difference between supervised and unsupervised learning in our discussion of NMF, where it is possible to learn the spectral envelopes either from a set of isolated note recordings that are labeled with pitch values (supervised) and from the input sound itself where we have to derive the frequency for each envelope afterwards (unsupervised).

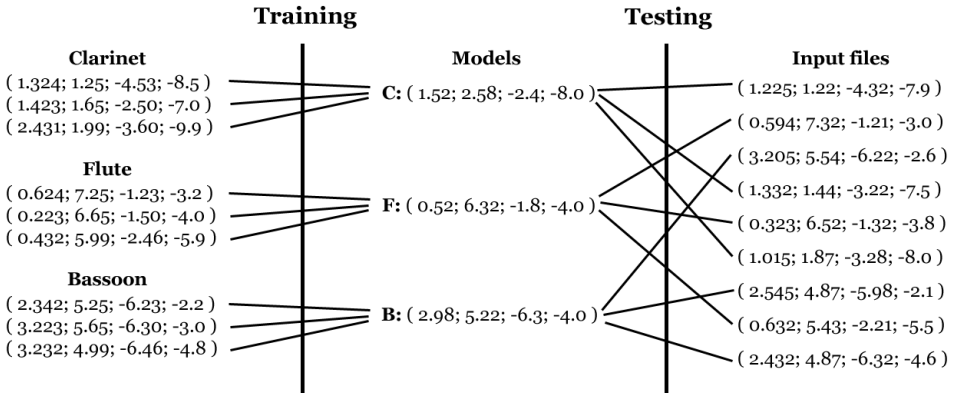

Figure 4.1: Classification process. Left: set of labeled feature sets derived from notes. Middle: models derived from the training set. Right: new input that is assigned a class from the models

In the case of instrument recognition, the machine learning system is provided with a set of instrument descriptions, where each description is generated from a sample note of this instrument or from the input recording itself and consists of a number of features. Each of these descriptions has the correct label, the instrument to which it belongs. The algorithm will then build a model for each of the instruments, determining the most important properties that set one instrument apart from the other. This will give better results than simply calculating the difference between each individual feature, thereby giving all features equal importance. In the unsupervised learning case, the left part of figure 4.1 will look similar to the right part. The training process is then expected to derive the models from this large set of training data.

In this section we will focus on a machine learning technique called adaptive boosting. Other classification algorithms include neural networks, Support Vector Machines (SVM) and many more. The choice for adaptive boosting was made because of past experiences with this technique in [8]. A comparison with alternative methods is outside of the scope of this project; machine learning literature and the references from this chapter on instrument recognition should provide an overview of possible alternatives.

4.1.1 Adaptive boosting

Adaptive boosting is one of many techniques that allows classification of data. It is not a classifier by itself; instead it combines the results of several simple classification methods - weak learners. The classification methods used by boosting are allowed to be straight-forward and have relatively low successful classification rates as the combination of a set of such classifiers will lead to classification performance surpassing that of the individual classifiers.

Most of the frameworks that combine several classifiers run these classifiers in parallel. With adaptive boosting however, the individual classifiers are trained in sequence so that the results of the previous classifier influence the training of the current one. Samples that were wrongly classified by the previous classifier can then be given extra attention in the current classification process.

The concept of boosting is best explained using an image taken from the book *Pattern Recognition and Machine Learning* by Bishop [3]:

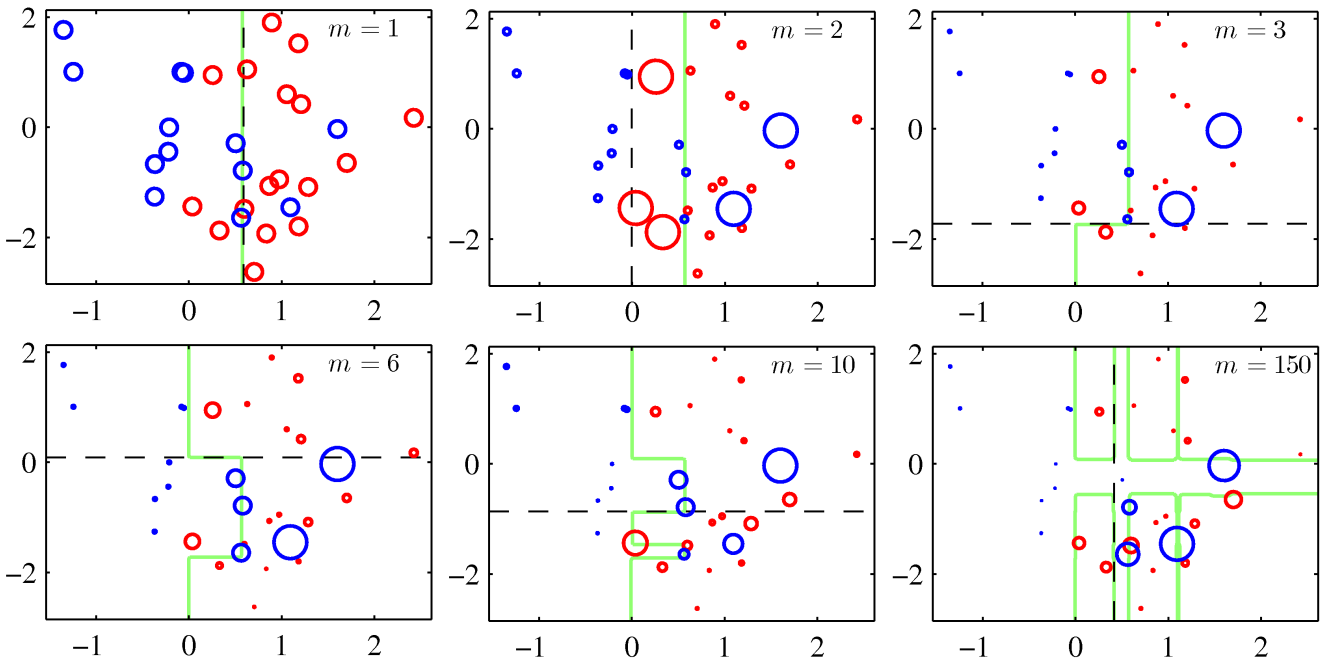

Figure 4.2: Adaptive boosting applied to two classes (red and blue); each image shows the result of adding a new classifier

How the combination of different weak classifiers leads to a strong classifier in adaptive boosting is shown in figure 4.2. It is based on the task of assigning samples - the circles - to one out of two possible classes, in this case red or blue. The radius of each circle (starting from the second image) indicates the weight of that sample in the classification training process. The weight of incorrectly classified samples is increased for the next weak classifier so that they have an increased influence on the placement of the decision boundary. This boundary determines the difference between the two classes. For example, in the first image everything to the left of the boundary is assigned to the blue class while everything on the right is assigned to the red class. The goal is then to determine a boundary where the number of incorrectly classified samples is as little as possible.

The first image shows the result of using one weak classifier and each successive image adds another weak classifier. Green lines indicate the decision boundary which separates the samples between the two classes. The dashed line is the decision boundary of only the latest added weak classifier, therefore in the first image the green and dashed lines are equal. Figure 4.2 is a two-dimensional example of this classification process, where each sample has two properties placing it somewhere in 2D space. In many cases, including instrument features, the number of dimensions for each sample will be higher but the principle of classification remains the same. The goal is still to find a boundary that correctly separates the N-dimensional space so that all samples are assigned to their correct classes. Newly introduced samples should then automatically be located on the correct side of the boundary.

Adaptive boosting is designed to work with two classes. For instrument recognition, where each instrument is defined as a class, we desire the ability to introduce more than these two classes. For this reason we have used MultiBoost, an extension of adaptive boosting that does support more than two classes. It is implemented in C++ and includes four types of weak learners. We have opted to use MultiBoost as an external program by saving our instrument features to a text file of certain format, running the classifier on this text file and then importing the text file containing the results back into our MATLAB implementation. Some alterations are made so that MultiBoost provides for each sample not only the classified instrument but all instruments ranked by the distance of the sample to each instrument. These alterations, along with instructions on how to control MultiBoost, are described in the project source code and in our previous work with MultiBoost [8].

4.2 Overview of instrument recognition techniques

Much of the existing research regarding instrument recognition focuses on isolated notes. A model is trained on a large collection of single note recordings, which is then used to assign instrument labels to similar recordings from a different dataset. Several results using single note recordings are listed in the PhD thesis of Park [28], indicating that a successful recognition rate of around 70-80% should be possible regardless of the classification method used. Neural networks seem to yield particularly high success rates of above 90%, although a comparison between different methods and implementations is often not objective due to differences in datasets used, both in the number of different instruments and the total number of samples used. Non-negative matrix factorization has been used to introduce temporal information to instrument recognition on isolated notes by Tjoa and Liu [35]. They report a success rate of 72.9% for 24 instruments, where there is still some misclassification within instrument families.

It is expected to be a big leap from the single note recognition task to identifying instruments from a polyphonic musical piece. Martins et al. [24] derive a time-frequency template as timbre model before applying PCA to reduce its dimensionality. Performance of matching these timbre models was measured for single note recordings of six instruments, resulting in a 83.3% success rate. Their most challenging instruments for recognition seem to be the violin and clarinet. Classification performance for a polyphonic mixture drops to an average of 65% with two voices, 50% with three voices and 33% with four voices. Performance of other systems that work with polyphonic mixtures instead of isolated notes is similar, with most research publishing recognition rates between 50% and 60%. Leveau et al. [21] provide an overview of the change in performance of their method based on a sparse decomposition into harmonic atoms as more instruments playing simultaneously are introduced. They show a drop in recognition rate of about 30% for each additional instrument. Several methods have obtained improved success rates when including fundamental frequency information in the recognition process. Kitahara et al. in [18] introduce musical context and weights based on the estimated influence of overlapping harmonics on a particular note in order to improve recognition rates from 46.5% to 72.3% for four voices.

We hope that the use of a model-based technique to separate the musical piece into single note events allows us to simplify our instrument recognition task to the single note situation. We can expect the quality of these isolated notes, extracted from a musical piece, to be lower than the quality of isolated recordings. Most extracted notes will have a shorter duration, resulting in a less detailed temporal envelope. Furthermore, part of the harmonic structure will be lost due to the overlaps that occur in the frequency spectrum of polyphonic music. This approach of using a form of source separation and note extraction as a pre-processing step before applying instrument recognition is of particular interest to this project as it combines well with our existing NMF implementation from the fundamental frequency estimation task discussed in chapter 3.

4.3 Features to describe timbre

There are many features that can be derived from the temporal and spectral properties of a music note. By combining a set of these features we attempt to capture the timbre of a note. This timbre description can then be used to compare notes in order to determine which notes originate from the same instrument. We attempt to provide a comprehensive overview of the features that we have implemented for this project and that are commonly applied to the task of instrument recognition.

The Moving Pictures Experts Group (MPEG) have proposed an MPEG-7 standard of sound descriptors based on both time and frequency domains. Though their focus is on audio segment analysis, used for tasks such as genre identification, the features are commonly found in instrument recognition as well [44]. We focus on time-frequency domain features in the following sections as this is the domain in which our information is represented.

Many features apply to the spectral envelope of a note. To involve the temporal aspect, it is possible to use the mean and standard deviation measures of the extracted feature through time. The details of several features can be found in the extensive research that has been done in this area, for example in the work of Peeters [29]. A combination of the features by MPEG en by Peeters is used in this work.

4.3.1 Spectral centroid

The spectral centroid is the center of gravity, the mean frequency, of the spectral envelope at a certain point in time. Its value is therefore a frequency value. From the centroid values through time we can derive the mean centroid and its standard deviation to capture the center of gravity for the entire note duration. Perceptually, it can be regarded as the *brightness* of a sound. The centroid is derived as:

$$SC(t) = \frac{\sum_{f=1}^{F} cf(f) \times X(f,t)}{\sum_{f=1}^{F} X(f,t)} \tag{4.1}$$

Where $cf(f)$ is the center frequency of frequency bin f.

Normalized spectral centroid

The spectral centroid can be normalized so that it does not depend on the pitch of the note by dividing its value by the fundamental frequency:

$$nSC = \frac{SC}{f0} \tag{4.2}$$

4.3.2 Spectral moments

The spectral moments define a number of properties related to the spectral envelope shape of a sound. There exist several different moments, with a general equation:

$$Moment(o,t) = \frac{\sum_{f=1}^{F} (cf(f) - SC(t))^o \times X(f,t)}{\sum_{f=1}^{F} X(f,t)} \tag{4.3}$$

Where o is the moment and $SC(t)$ is the spectral centroid at timestep t. Several moments that can be used as features for instrument recognition are listed below. The moments can be weighted by the power spectrum by changing $X(f,t)$ to $X(f,t)^2$.

Spectral spread

Together with the spectral centroid, this feature roughly describes the spectral envelope shape. It captures the spread of frequency activity around the spectral centroid and is defined by the second moment of the spectral envelope.

Spectral skewness

The difference between the part of the spectral envelope that is below the spectral centroid and the part that is above the spectral centroid in frequency is represented in the spectral skewness measure. It is the third central moment of the spectral envelope, normalized by the second moment:

$$SSk(t) = \frac{Moment(3,t)}{Moment(2,t)^{1.5}} \tag{4.4}$$

Spectral kurtosis

Kurtosis is based on the fourth moment of the spectral envelope. It shows the amount of flatness of the envelope compared to a normal distribution. A high kurtosis means that there is a high peak at the mean of the distribution (the spectral centroid), where the rest of the energy is mostly distributed close to this mean. In this case, a lot of energy is focused near the center of gravity of the spectral envelope. Low kurtosis results in energy that is more spread out throughout the frequency range.

The fourth moment is normalized by the second moment to obtain kurtosis:

$$SK(t) = \frac{Moment(4,t)}{Moment(2,t)^2} - 3 \tag{4.5}$$

4.3.3 Spectral roll-off

The spectral roll-off is the frequency below which a certain percentage of the total energy of the envelope is contained. A percentage of 95% is often used, as it is said to be related to the harmonic/noise cutting frequency [29]. The roll-off can be calculated by summing up the energy at each frequency bin, starting at the lowest bin. When the total energy exceeds 95%, the center frequency of the current bin is the roll-off frequency. This results in the equation:

$$\sum_{f=1}^{f_{ro}} X(f,t) = 0.95 \times \sum_{f=1}^{F} X(f,t) \tag{4.6}$$

Where f_{ro} is the roll-off frequency.

4.3.4 Mel-frequency cepstral coefficients (MFCC)

Cepstral representations are commonly used in speech recognition systems and have recently been applied to other fields of music information retrieval such as instrument recognition. The cepstral representation is defined as the short-time Fourier transform of the logarithm of a spectrogram. It is essentially a time-frequency transformation of a time-frequency transformation, providing information on the rate of change in the frequency bands. The concept is introduced by Davis and Mermelstein [6].

The cepstrum is often converted to a mel frequency scale. The idea behind the mel frequency scale is similar to the ERB scale that we have seen in section 2.3 as it is also based on the way human hearing works, with a linear scale in the lower frequency range and a logarithmic scale in the higher frequency range.

To get the coefficients from the cepstral representation, a discrete cosine transform (DCT) is applied to the cepstrum. The coefficients form a low-dimensional representation of the entire spectrogram, similar to techniques such as linear prediction but apparently more robust to noise [34].

If the number of coefficients is set to CO, the dimensionality of the MFCC is $CO \times M$. To include temporal development while keeping the number of features to a minimum, we can derive the mean or, as experiments have shown to be superior, the median to include temporal information while retaining a dimensionality of CO.

HFCC

While the mel-frequency is perceptually motivated, it does not capture the relationship between frequency and the critical bandwidth of the human auditory system like the ERB scale does [34]. For this reason the human factor cepstral coefficients (HFCC) have been introduced, which are calculated in the same way except for the change in time-frequency scale.

Delta coefficients

To include the development of the cepstral coefficients (either MFCC or HFCC) through time, the delta coefficients can be included as features as well. They are defined as the derivatives of the coefficients through time.

Additionally, some researchers have included the delta delta coefficients, the second order derivatives of the coefficients, as well.

4.3.5 Temporal centroid

Similar to the center of gravity in the spectral envelope, it is also possible to derive a center of gravity from the temporal envelope:

$$TC(t) = \frac{\sum_{f=1}^{F} \sum_{t=1}^{T} X(f,t) \times t}{\sum_{f=1}^{F} \sum_{t=1}^{T} X(f,t)} \tag{4.7}$$

4.3.6 Onset time

The duration of the attack phase of a note is said to be an important feature of an instrument. It can be derived using the onset time, which can be found using an onset detection method. The starting time of the note is also required, which can be retrieved by moving backwards along the amplitude envelope starting at the onset time until the amplitude falls below a certain threshold.

$$OT = t_{onset} - t_{begin} \tag{4.8}$$

4.3.7 Odd-even harmonic ratio

Instruments such as the clarinet have more activity in the odd harmonics, while others such as the trumpet have more activity in the even harmonics. The difference between the energy present in odd and even harmonic frequencies is therefore an important feature for instrument recognition.

$$OER(t) = \sum_{h=1}^{H} \frac{X(f0 \times h \times 2, t)}{X(f0 \times h \times 2 - 1, t)} \tag{4.9}$$

Where $H \times 2$ is the number of harmonics to consider.

4.3.8 Relative harmonic amplitudes

It is also possible to store the relative amplitudes of a number of harmonic frequencies as separate features. Our experiments show that storing the harmonic spectral envelope structure in this way is valid for instrument recognition. The amplitudes at the harmonic frequencies can be stored relative either to the amplitude at the fundamental frequency, to the total amplitude or to the maximum amplitude present in the harmonic spectral envelope. We chose to derive the harmonic amplitudes relative to the amplitude at the fundamental frequency, which is calculated as follows:

$$RHA(t, h) = \frac{X(f0 \times h, t)}{X(f0, t)} \tag{4.10}$$

Where h is the number of the harmonic overtone, $h = 1$ being the fundamental frequency. We take the mean amplitude of each harmonic, so if h is taken from 1 through to 10 then we will have 10 features, a mean for each harmonic through time.

4.3.9 Tristimulus

The tristimulus is strongly related to the relative harmonic amplitudes, however in this case ranges of harmonic overtones are combined into three groups. This concept is said to be similar to the three color attributes of RGB in vision [32]. Three features are extracted, each based on the amplitude of a certain range of harmonics relative to the total amplitude of all harmonics.

$$T1(t) = \frac{X(f0, t)}{\sum_{h=1}^{H} X(f0 \times h, t)} \tag{4.11}$$

$$T2(t) = \frac{\sum_{h=2}^{4} X(f0 \times h, t)}{\sum_{h=1}^{H} X(f0 \times h, t)} \tag{4.12}$$

$$T3(t) = \frac{\sum_{h=5}^{H} X(f0 \times h, t)}{\sum_{h=1}^{H} X(f0 \times h, t)} \tag{4.13}$$

The first group of the tristimulus is based only on the first harmonic frequency (the fundamental), the second group contains the harmonics in the range [2,4] and the last group contains all the others up to harmonic H.

4.3.10 Spectral slope

To capture the rate at which the amplitude between harmonics decreases, we can fit a line to the spectral envelope and derive its slope as a feature. MATLAB supports this polynomial fitting with the command *polyfit*. The slope is defined as:

$$SL(t) = \frac{F\sum_{f=1}^{F} cf(f) \times X(f,t) - \sum_{f=1}^{F} cf(f) \times \sum_{f=1}^{F} X(f,t)}{\sum_{f=1}^{F} X(f,t) \times F\sum_{f=1}^{F} cf(f)^2 - (\sum_{f=1}^{F} cf(f))^2} \tag{4.14}$$

4.3.11 Octave-based spectral contrast (OSC)

Spectral contrast is introduced by Jiang et al. [16] for genre classification. It is implemented using the octave time-frequency scale, where each bin represents the frequency range belonging to a single musical octave. For each octave scale bin the spectral peaks and valleys are derived.

First, the frequency bins within each octave bin are sorted descending by amplitude. The peak value for each octave bin is then defined as:

$$peak(b,t) = log(1/NB(b) \sum_{f=1}^{NB(b)} X_b(f,t)) \tag{4.15}$$

Where b is the octave bin, X_b is the part of the spectrogram that belongs to the octave bin and NB is the number of frequency bins within one octave to include in the peak detection. We have set NB to be:

$$NB(b) = round(.02 \times F_b) \tag{4.16}$$

With F_b the number of frequency bins belonging to octave scale bin b. The valleys are derived in a similar fashion to the peaks, now starting at the bottom of the octave bin as the lowest amplitude values are situated there:

$$valley(b,t) = log(1/NB(b) \sum_{f=1}^{NB(b)} X_b(F_b - f + 1, t) \tag{4.17}$$

The constrast is then defined as the difference between the peaks and valleys.

4.3.12 Amplitude modulation

The temporal envelope of a spectrogram of a note can be derived by summing up all frequency bins, resulting in the total amplitude at each point in time. By smoothing this temporal amplitude envelope and then calculating the difference between the actual envelope and the smoothed version, we can derive the modulation of the amplitude through time. The amount of modulation differs between instruments. Assuming the amplitude envelope is defined as $AE(t) = \sum_{f=1}^{F} X(f,t)$, we can apply smoothing as:

$$AS(t) = \frac{AE(t-2) + 4AE(t-1) + 5AE(t) + 4AE(t+1) + AE(t+2)}{15} \tag{4.18}$$

Modulation is then calculated by taking the difference:

$$AM = AE - AS \tag{4.19}$$

Frequency of amplitude modulation

To extend the feature of amplitude modulation, we can also determine its frequency which is defined as the number of zero crossings of the derivative of the amplitude. The derivative shows the change in amplitude, by then examining the number of zero crossings of this change we know how often a negative change in amplitude is followed by a positive change in amplitude and vice versa. This is similar to how a sine wave repeats its pattern at a certain frequency, where at each cycle the amplitude first drops and then goes back up again. These changes are detected by finding the zero crossings.

Zero crossings are found by examining two pairs of values, in this case two timesteps of the derivative of the amplitude envelope: t and $t-1$. If the envelope at t has a positive value and at $t-1$ is negative then we have found a zero crossing. Likewise, if t is negative and $t-1$ is positive, this also marks a zero crossing.

4.3.13 Spectral flux

The concept of spectral flux is already discussed for the purpose of onset detection in section 3.3.2. The same spectral flux indicating changes in the frequency components over time can be used as a feature for instrument recognition. This is done by calculating the flux from the spectral envelopes of successive timesteps, then storing the mean and standard deviation of the resulting spectral flux envelope.

4.4 Time-frequency scales

Features can be derived from various time-frequency scale representations of a music note. For instance, Livshin and Rodet [23] derive properties such as the spectral centroid and spectral spread from a linear time-frequency scale as well as a *perceptual* scale such as ERB. Each representation defines a different feature type as the perceptual scales are based more on the way humans perceive sound.

Features can also be derived from the *harmonic* envelope, which is a post-processing step rather than a time-frequency scale of its own. It is derived by including only the energy at the fundamental frequency and harmonic overtones, optionally including a small area around these frequencies to incorporate slight frequency vibrations. From this representation, features such as the harmonic spectral centroid can be derived.

4.5 Note phases

The envelope of the lifespan of a sound, captured in a model such as the attack-decay-sustain-release model (ADSR, see section 2.1.2), will provide additional possibilities of distinguishing between instruments. Besides the ability of deriving properties such as attack duration using this model, it is also possible to extract features from each of the phases separately instead of extracting them from the sound as a whole, in order to get more information as the characteristics of the sound may vary greatly between the various phases. Looking at each phase individually could therefore improve the feature quality.

However, by looking at visual representations of individual tones and especially the ones that were extracted from a mixture using NMF, identifying these phases automatically appears to be non-trivial. Peeters in [29] confirms this finding, stating that depending on the instrument used not all phases may even be present to begin with. His suggestion of distinguishing between the attack and steady-state (or rest) phases is exactly what we have concluded as well. Decay in this representation is ignored, whereas sustain and release are combined into the steady-state part.

Alternatively, one could use analysis windows and texture windows described by Tzanetakis and Cook in [36]. Both windows divide a signal or note into smaller segments, extracting features from each segment. Analysis windows (23 ms) are smaller than texture windows (around 1s), resulting in a segment that can be considered stationary. Patterns through time are found using the larger texture windows. By extracting features from several segments rather than the signal as a whole, either using windows or by detecting the note phases, the quality of the feature description is expected to improve.

Chapter 5

Evaluation of non-negative matrix factorization

In the following three chapters we discuss the experiments that we have conducted in order to evaluate the performance of non-negative matrix factorization for the tasks of fundamental frequency estimation and instrument recognition. We start by describing the experiments that we have conducted in the current chapter, focusing on explaining the goal and relevance of these experiments. After this overview, in chapter 6 we provide detailed information on the datasets, algorithm implementations and evaluation techniques that were used in conducting our experiments before listing our findings in chapter 7.

The following table lists the topics that we would like to investigate, followed by the section in which a detailed description of each topic is provided:

Description	Section
Multiple fundamental frequency estimation using NMF	5.1
Impact of training data instrument composition for fundamental frequency estimation	5.2
Isolated note instrument recognition using timbre descriptors	5.3
Instrument recognition using NMF with instrument annotations	5.4
Instrument recognition using timbre descriptors of notes extracted from a musical piece using NMF	5.5
Comparison of time-frequency scales	5.6
Using 3 spectral envelopes per training sample instead of 1	5.7
Introducing our own implementation of NMF	5.8
Using note events instead of single timesteps for fundamental frequency estimation	5.9
Refining note events using post-processing	5.10
Using instrument recognition to enhance fundamental frequency estimation performance	5.11

Table 5.1: List of experiments conducted in this project

The first five sections are experiments with several technical implementations for multiple fundamental frequency estimation and instrument recognition. These sections are followed by two sections that try to determine the influence of parameters such as time-frequency scale. Then, from section 5.8 onward, we introduce our own framework and work towards a combination of frequency estimation and instrument recognition.

5.1 Multiple fundamental frequency estimation using NMF

In our discussion of the non-negative matrix factorization technique (see: section 3.2) we have seen several different variations of this factorization method. The differences in performance of these NMF variations have been evaluated by Vincent et al. in [38].

What we would like to investigate further is the maximal potential performance of non-negative matrix factorization for the task of multiple fundamental frequency estimation. We focus on the ability to obtain the correct fundamental frequencies from a musical piece, while avoiding any forms of post-processing that upon trying to reduce the number of incorrect frequencies may also remove some correct frequencies.

In short, we want to obtain an overview of the potential *recall* that non-negative matrix factorization is able to obtain, without relying heavily on reducing the number of incorrect frequencies that are returned along with the correct ones. Furthermore, we are interested in the differences in potential performance between variations of NMF that generate spectral envelopes from a set of training data and variations such as harmonic NMF by Vincent et al. that do not use training data.

Of course, by simply returning all pitch values at each point in time, any method will be able to obtain good recall. Therefore, we will have to enforce a limit onto the number of frequencies that are considered.

5.2 Impact of training data instrument composition for fundamental frequency estimation

Besides experimenting with non-negative matrix factorization using a trained set of envelopes, we also evaluate the influence of the composition of this training data. Specifically, we examine the quality of the reconstruction when the instruments that are used in the musical piece are not present in the training set.

This experiment is conducted to provide insight into the required size of the dataset used. If every instrument that could possibly occur in any musical piece is required to be present in the dataset, then the use of trained NMF is limited as it will be impossible to generate such a dataset. An increase in the size of the dataset also leads to a decrease in performance as a larger collection of spectral envelopes results in a larger set of activations that needs to be updated iteratively.

5.3 Isolated note instrument recognition using timbre descriptors

In section 4.3, we have discussed features that can be derived from the temporal and spectral information of a note. These features can be used to compare two isolated note recordings in order to determine their similarity in instrument character, or timbre. We already have a set of training notes of which we know the instrument that played them. If we derive features from these notes, we can obtain a description of an instrument against which we can compare any notes of which we do *not* know the instrument.

This process of trying to assign an instrument label to a number of single note recordings by using an instrument model that is derived from a training set of other single note recordings is commonly applied to test the performance of instrument recognition methods. Because the notes used for testing are recorded by a different instrument than those used for training, positive results indicate the ability to extract instrument character regardless of external influences such as the recording environment or the exact model of the instrument used.

5.4 Instrument recognition using NMF with instrument annotations

Besides for the task of fundamental frequency estimation, we would also like to consider non-negative matrix factorization as a technique for instrument recognition. When extracting spectral envelopes from notes in the training set, we expect these envelopes to capture not only the correct pitch value but also the character of the instrument used to record the single note.

If we then use these envelopes to reconstruct a musical piece, we believe that the matrix factorization process will use spectral envelopes that belong to instruments equal to those used to create the musical piece. The underlying thought is that the reconstruction should match the original musical piece more if the same instruments are used in this reconstruction. For example, if the recording contains a piano and a trumpet, then NMF will use spectral envelopes derived from piano and trumpet samples to do the reconstruction. If this is indeed the case we will know the instrument that produced each note, allowing us to group notes by instrument.

5.5 Instrument recognition using timbre descriptors of notes extracted from a musical piece using NMF

When looking at the task of transcription, the input data is an entire musical piece rather than single notes. This is exactly where NMF can help, as it provides us with a way to separate notes from the musical piece. We can do this using the results from the frequency estimation step, which consist of note events that have a certain pitch and duration. The frequency is known because we know which spectral envelopes are used in the reconstruction, and the note duration is marked by the changes in amplitude of this frequency.

Using this information we can simply extract the spectrogram of a note. This spectrogram will not be perfect as it can be influenced by the other notes being played, but the idea is to extract features that are still unique to each instrument even if some information is lost.

As an example, say that the result of the frequency estimation step shows that an $A4$ note (440 Hz) was present between 1 and 3 seconds into the recording. Remember that we build up the reconstructed spectrum Y by multiplying spectral envelopes S with activations through time A. If we take the activations only for the time between 1 and 3 seconds, then multiply them with only the spectral envelopes belonging to A4 notes, we have reconstructed the single A4 note. We can expect the quality of this reconstruction to decrease as the number of notes active at the same time increases.

We can test the performance of instrument recognition by using datasets of isolated note recordings for training similar to what we have done in section 5.3, only now using notes extracted from a recording with several voices as test data.

5.6 Comparison of time-frequency scales

After exploring the potential of non-negative matrix factorization in the previous experiments, we would like to find out if this performance is related to the choice of a certain time-frequency scale. As we have seen in section 2.2.3, the Fourier transform by default results in a linear time-frequency scale. This means that the distances and sizes of consecutive frequency bins is always equal. The equivalent rectangular bandwidth (ERB) scale introduced in section 2.3 is commonly used in music information retrieval tasks as it attempts to describe amplitudes and frequency relationships in a way that is similar to how humans observe sound.

While there are many tasks within the field of music information retrieval that may benefit from a frequency response that matches that of humans, we are especially interested in the difference between the linear and ERB scales for the tasks of multiple fundamental frequency estimation, NMF-based instrument recognition and instrument recognition based on timbre features.

5.7 Using 3 spectral envelopes per training sample instead of 1

It is possible to apply non-negative matrix factorization to isolated note recordings in order to extract a spectral envelope from the recording that best describes the frequency structure of the particular note present in the recording. This is done by setting I, the number of spectral envelopes contained in S, to 1. This ensures that only one envelope will be present in S, which can then be stored and used together with envelopes generated from other isolated note recordings to perform NMF on a musical piece.

By setting I to a number higher than 1, it is possible to derive more than a single envelope from the same note recording. However, each envelope may then start to describe only part of the spectral structure of a note. In this case, multiple envelopes are activated simultaneously to reconstruct the note recording. Ideally, we would like each envelope by itself to describe the spectral envelope of an entire note, which means that only one envelope is active at each point in time. Each of the envelopes is then used to describe a different phase or variation within the note so that the note can be reconstructed with more detail, for example capturing effects such as vibrato or the characteristic spectral envelope at the attack phase of the note.

As we have seen in section 3.2.3, by using L0-sparse non-negative matrix factorization we may achieve this effect by allowing only one of the spectral envelopes to be active at any point in time. We are interested in the influence of deriving more than one spectral envelope for each note on the quality of the reconstruction of a musical piece. We evaluate the influence of using more than one spectral envelope per note for the tasks of multiple fundamental frequency estimation and instrument recognition using NMF.

5.8 Introducing our own implementation of NMF

After conducting several experiments using existing NMF techniques, we would like to introduce our own NMF implementation. We base this implementation on known research, example code and libraries that are publicly available so that future research is possible without depending on the work of others. We expect the results of this new method to be similar to the results that we have obtained in our previous experiments.

5.9 Using note events instead of single timesteps for fundamental frequency estimation

Taking into account the activations of the spectral envelopes of various pitch values through time, we are able to introduce a form of temporal continuity by working with note events rather than individual timesteps. Onset detection plays a part in generating these note events from the pitch activations through time, as it can be used to determine the characteristic onset of a note. These onset times can be used to derive the start and end times of notes by moving through time in both directions from the onset location to a point where the amplitude stabilizes. While evaluation of the onset detection method itself is outside of the scope of this project, its performance will become apparent in the quality of the generated notes.

By defining notes, we hope to rule out noisy incorrect frequencies that only show up at single or a few timesteps, leaving the correct frequencies intact. The result will then be a higher precision in the frequency estimation task with a minimal loss in recall. For the task of automatically transcribing a musical piece, what we are interested in are in fact notes that have a certain pitch value and duration rather than the exact frequencies that are active at small intervals.

5.10 Refining note events using post-processing

In the previous experiment, we have defined notes using an onset detection algorithm. Because we do not expect this onset detection method to return only the correct notes, we consider several post-processing steps in this follow-up experiment.

For instance, some detected notes may have a duration that is too short to be observed by human listeners. It is also possible for notes to have an amplitude that is too low to be observed. We can examine both the absolute amplitude value of the note as it is observed. However, it is also possible to consider the amplitude of a note relative to the other notes that are active at a certain point in time. If a very loud note is already active, it might prevent us from hearing a note that would have been audible with no other notes active.

5.11 Using instrument recognition to enhance fundamental frequency estimation performance

Our goal is to improve both instrument recognition and frequency estimation by estimating the instruments present in the musical piece and the number of notes these instruments will generally play at the same time. If an instrument has been playing one note at a time for 80% of a musical piece, we can expect the instrument to be able to play only one note at a time and therefore have only one note at a time for the remainder of the piece. If a piano shows up but only for one or two notes throughout an entire recording, we may assume that the instrument used was not a piano, but a violin as that instrument had been playing already throughout the musical piece.

We believe that providing as much basic information as possible about a musical piece will allow the development of systems that combine this information with heuristics and models based on musical knowledge to improve the results of music analysis. After removing incorrect notes in the previous steps, we now attempt to derive the instruments and the number of notes each of these instruments is expected to play at one time. By enforcing these limits onto the number of active frequencies that are allowed, we hope to raise precision without sacrificing recall.

Chapter 6

Experiment outline

6.1 Datasets

Dataset name	Description
TrainedVERB	Using single note recordings with the ERB scale by Vincent et al., 1 envelope per recording
TrainedVERB3	Using single note recordings with the ERB scale by Vincent et al., 3 envelopes per recording
TrainedGERB	Using single note recordings with the ERB scale by Ellis, 1 envelope per recording
TrainedGLin	Using single note recordings with the linear scale by Ellis, 1 envelope per recording
MIREX	MIREX 2007 Development dataset

Table 6.1: Datasets used in the experiments of the following chapters

6.1.1 Training datasets

The first four datasets in 6.1 are all used for training in order to derive spectral envelopes S for use with NMF. Each of the datasets uses the same collection of isolated note recordings, which are obtained from the Real World Computing (RWC) dataset [10] as well as the University of Iowa music instrument samples set[1]. The exact samples that were used are provided as an appendix with this thesis report. In total, eleven different instruments were extracted from both datasets. We included only the *mezzo forte* variation of each instrument recording as we expected this style of playing to best match the style that is present in the musical pieces used for testing.

Both the Real World Computing and the University of Iowa datasets consist of instrument recordings containing multiple notes being played by a certain instrument. In order to derive spectral envelopes from each note, we have separated these recordings into smaller ones, each containing only one note. For the RWC dataset, this was done using onset markers provided by Chunghsin Yeh[2]. The recordings from the University of Iowa dataset were separated using the *Sound eXchange (SoX)* command line utility[3] to split each recording into segments.

To get from isolated note recordings to a set of spectral envelopes, we use a standard non-constrained NMF that is randomly initialized and then learns spectral envelopes for each note recording. These spectral envelopes are then stored in S. The exact process of obtaining spectral envelopes from note recordings is further explained in section 3.2.6. Upon using a certain dataset for applying NMF to a musical piece, we must ensure that the time-frequency transform that is applied to the musical piece is identical to the one used to generate the training dataset that is used.

[1] http://theremin.music.uiowa.edu/MIS.html
[2] http://anasynth.ircam.fr/home/english/media/synthesized-polyphonic-music-database
[3] http://sox.sourceforge.net/

TrainedVERB

The first dataset is generated using the time-frequency transform implementation by Vincent et al. [38]. The resulting time-frequency representation has an ERB scale. One spectral envelope is derived from each isolated note recording.

The resulting dataset is 250×888 in size, where each of the 888 notes is represented using one spectral envelope and each spectral envelope contains 250 frequency bins. Note that because the spectral envelopes are represented in the ERB scale, the number of frequencies contained within each frequency bin is not equal throughout the frequency range.

TrainedVERB3

TrainedVERB3 is generated using the same time-frequency transform as *TrainedVERB*. However, instead of deriving one spectral envelope for each note recording we now derive at most three. This is done by applying L0-sparse NMF (explained in section 3.2.3) to ensure that each spectral envelope still describes the entire spectral structure of the note.

For some notes we expect less than three spectral envelopes to suffice in describing its different phases. Although we have manually fixed the number of allowed spectral envelopes to be generated in S to three, the L0 sparse NMF method will activate only one or two of these envelopes if no more are needed. This is why we introduce an amplitude threshold for A. The threshold is set to 33% of the maximum activation value in A. This means that the spectral envelope that has the most activity will always be included, as its maximum activation will always be 100% of the overall maximum in A. Any spectral envelope that never crosses this threshold throughout the note recording is not included in the resulting training dataset, reducing the risk of introducing random or incorrect spectral envelopes into the *TrainedVERB3* dataset.

Because L0 sparse NMF has a risk of convergence to a local minimum, we run the method five times for each note recording and then take the median to obtain the spectral envelope combination that the method will converge to most often.

TrainedGLin and TrainedGERB

TrainedGERB, like *TrainedVERB*, only derives one spectral envelope for each note recording. The difference between the two methods lies in the technical implementation of the time-frequency transform that is used. *TrainedGERB* uses the gammatonegram library by Ellis [9]. This library generates a spectrogram with an ERB scale by first performing a Fourier transform that results in a linear scale spectrogram and then applying a matrix transformation to turn the linear scale into an ERB scale.

TrainedGLin is identical to *TrainedGERB*, now excluding the final step that changes the spectrogram of linear scale into a spectrogram of ERB scale. *TrainedGLin* is therefore the linear scale representation that is transformed into the ERB scale in *TrainedGERB*.

6.1.2 Testing: MIREX dataset

The MIREX 2007 development dataset is the only set that for each file contains a combination of notes rather than one single note. The set consists of five isolated real world recordings of: clarinet, flute, bassoon, horn and oboe. In order to reproduce the test results of Vincent, who has used this dataset in a comparison of several fundamental frequency estimation methods, we have extracted the first thirty seconds of each recording.

Ground truth was provided separately for each recording, where the fundamental frequencies are listed at 10 millisecond intervals. Because each recording contains only one voice, there is only one fundamental frequency at each interval for each ground truth file.

The ground truth files and instrument recordings were then combined to form musical pieces with a certain maximum number of voices, ranging from 1 to 5 voices. The following combinations were constructed:

- Clarinet
- Clarinet and flute
- Clarinet, flute and bassoon
- Clarinet, flute, bassoon and horn
- Clarinet, flute, bassoon, horn and oboe

These combinations result in five input files with corresponding ground truth, with the maximum number of voices ranging from one to five. We have ensured that the five instruments contained within the MIREX dataset are present in the training datasets discussed above as well.

6.2 Technical implementations

Two different technical NMF implementations are used in our experiments, where we have introduced several variations of both implementations.

Implementation	Description
VincentOrig	Original harmonically constrained NMF with amplitude threshold by Vincent et al.
VincentLim	*VincentOrig* without amplitude threshold, instead returns a fixed number of frequencies
TrainedLim	*VincentLim* using training data instead of harmonic constraints
WitLim	Alternative to *TrainedLim* developed for this project
WitFinal	*WitLim* with post-processing applied to improve the precision

Table 6.2: Technical implementations (systems) used in the experiments of the following chapters

6.2.1 VincentOrig

VincentOrig is the harmonically constrained non-negative matrix factorization discussed in section 3.2.5. It applies non-negative matrix factorization using a β-divergence as distance measure where β is set to 0.5. The spectral envelopes in S are generated using a set of basic spectral structures, each containing energy at a number of harmonic frequencies. This ensures that all entries of S describe the harmonic spectral envelope characteristic of a note. 88 envelopes are defined, one for each pitch value. Because the spectral envelopes are defined and fixed, no training data is used in this method. After applying NMF to reconstruct the spectrogram of the musical piece, active pitch values at each timestep are derived by considering the amount of activity for each pitch value in A. These activity values are thresholded to obtain the active pitch values.

6.2.2 VincentLim

VincentLim is identical to *VincentOrig* without the threshold on the amount of activity in A. Instead, the pitch values for each timestep are sorted according to their activity in A so that we can extract the N most active pitch values, where N can be set to any value. For example, by setting $N = 4$ we obtain the four most active pitch values at each time step. This version of the method by Vincent et al. can be used to evaluate the ability for harmonic NMF to obtain the correct fundamental frequencies given an estimation of the number of voices present in the song. Since there is no threshold of activity or amplitude, fundamental frequencies of low amplitude will be returned instead if no actual fundamental frequencies are present. Therefore, this alternative can be considered to maximize the potential *recall* at the cost of an expected drop in *precision*. Because post-processing is removed, this method can be used in a comparison with alternatives that also do not have post-processing, such as *TrainedLim* and *WitLim*.

6.2.3 TrainedLim

For *TrainedLim*, we have taken the exact implementation of *VincentLim* and removed the construction of S from basic spectral structures. Instead, we generate S from a training set containing isolated note recordings by applying NMF to each note recording. This generated S is then used when applying *TrainedLim* NMF to a musical piece, where S is fixed and therefore not allowed to update. This is the first method that uses training data.

6.2.4 WitLim

WitLim is a method developed specifically for this project. It is a similar implementation to *TrainedLim* without using the code provided by Vincent et al. so that *WitLim* can be used in future research without depending on the work of others.

Analogous to *TrainedLim*, the *WitLim* method also uses spectral envelopes generated from a training dataset for S. The results of running *TrainedLim* and *WitLim* with the same dataset for training are therefore expected to be similar.

6.2.5 WitFinal

WitFinal is based upon *WitLim*, where the fixed limit on the number of frequencies to return is removed and instead replaced by post-processing steps and an estimation of the number of voices. It provides a trade-off between recall and precision, similar to how this is done in *VincentOrig*.

6.3 Overview of experiments

Before discussing our approach for each experiment in detail, the following table lists the experiments discussed in chapter 5, expanded with the NMF implementations and datasets that we have used in conducting the experiments:

Description	Section	Systems	Datasets
Multiple fundamental frequency estimation using NMF	5.1	VincentOrig VincentLim TrainedLim	TrainedVERB
Impact of training data instrument composition for fundamental frequency estimation	5.2	TrainedLim	TrainedVERB (subsets)
Isolated note instrument recognition using timbre descriptors	5.3	MultiBoost	TrainedVERB (subsets)
Instrument recognition using NMF with instrument annotations	5.4	TrainedLim	TrainedVERB
Instrument recognition using timbre descriptors of notes extracted from a musical piece using NMF	5.5	TrainedLim MultiBoost	TrainedVERB
Comparison of time-frequency scales	5.6	TrainedLim	TrainedVERB TrainedGLin TrainedGERB
Using 3 spectral envelopes per training sample instead of 1	5.7	TrainedLim	TrainedVERB TrainedVERB3
Introducing our own implementation of NMF	5.8	WitLim TrainedLim	TrainedGLin TrainedVERB
Using note events instead of single timesteps for fundamental frequency estimation	5.9	WitLim	TrainedGLin
Refining note events using post-processing	5.10	WitLim	TrainedGLin
Using instrument recognition to enhance fundamental frequency estimation performance	5.11	WitFinal	TrainedGLin

Table 6.3: Experiments conducted in this project

6.4 Multiple fundamental frequency estimation using NMF

In this first test we try to maximize the potential performance of non-negative matrix factorization techniques, to see how well non-negative matrix factorization is suited for the task of multiple fundamental frequency estimation.

Multiple fundamental frequency estimation is conducted on the *MIREX* dataset, where the frequency estimation performance is measured for all five musical pieces in this dataset.

As a reference method, we use *VincentOrig*, which is a complete implementation that uses post-processing to reduce the number of incorrectly returned frequencies, thereby raising precision, though at a cost of some correct frequencies that are also removed. To avoid this trade-off between recall and precision, we then compare two methods that do not include any post-processing to remove incorrect frequencies: *VincentLim* and *TrainedLim*. Out of the three methods used in this experiment, *TrainedLim* is the only method that uses a training set to create the spectral envelopes S. To train *TrainedLim*, we used the *TrainedVERB* dataset.

The results of this experiment are evaluated by deriving the precision, recall and F-measure using the output of each method and the ground-truth data from the MIREX dataset. We expect both *VincentLim* and *TrainedLim* to obtain a higher recall than the reference method *VincentOrig*, though at a cost of a loss in precision. Precision in this experiment is higher than it would be for arbitrary input as the number of frequencies to return is fixed manually, based on knowledge of the number of voices present in each musical piece from the MIREX dataset.

6.4.1 List of thresholds and parameters

Several parameters were set for this experiment, as shown in the table below. The window size and Beta-divergence values were chosen based on experiments by Vincent et al. in [38]. Although the referenced work lists 23 ms as the window size, the related code examples use 20 ms therefore we have adopted the same value for our research.

Parameter	Value	Implementation
Window size	20 ms	All
Beta-divergence	0.5	All
Local convergence threshold	0.005	All
Global convergence threshold	0.01	All
Number of returned frequencies	Number of voices + 4	VincentLim, TrainedLim
Time-frequency scale	ERB (filterbank)	All

Table 6.4: Parameters for multiple fundamental frequency estimation using NMF

6.5 Impact of training data instrument composition for fundamental frequency estimation

This experiment is identical to the previous experiment, with the exception that the *TrainedVERB* dataset is now split up into two subsets:

- **TrainedVERBcfbho:** contains exactly the five instrument used in the MIREX dataset
- **TrainedVERBother:** contains the six other instruments that are in *TrainedVERB*

We evaluate the fundamental frequency estimation performance and expect this performance to drop for *TrainedVERBother*, where the instruments that are used in the *MIREX* musical pieces are not present.

The method used for this experiment is *TrainedLim*, where the parameters are equal to those listed in table 6.4.

6.6 Isolated note instrument recognition using timbre descriptors

In this experiment, we do not use non-negative matrix factorization. Timbre features can be extracted directly from spectrograms of notes, therefore all we need to do is apply a time-frequency transform to all note recordings in our training dataset in order to extract the desired timbre features.

In this case, we use *TrainedVERB*. Instead of deriving spectral envelopes using NMF, we now only obtain the spectrogram for each recording within the dataset. Remember that the *TrainedVERB* dataset consists of combined samples taken from the RWC and University of Iowa datasets. We now split these samples so that we have one subset containing the RWC samples and one subset containing the University of Iowa samples. This ensures that each instrument is represented in both datasets, while the recording environment and exact instrument model used are different. By then training on one subset and testing using the other, we can tell how well the instrument recognition method is able to capture the character of each instrument.

The timbre features each consist of one or several numbers. The exact meaning of these numbers is irrelevant for a classification method such as MultiBoost, all that matters is that the numbers are far enough apart for the classification method to derive a model for each instrument. Each recording supplied for testing is then assigned a certain instrument class based on the similarity between the descriptor of this recording and each of the instrument models.

In short, in order to evaluate instrument recognition performance using MultiBoost and timbre descriptors, we apply a time-frequency transform to all notes within two different datasets and put the resulting descriptors (collections of values) into a text file for each dataset along with the known instrument label. We now have two text files, one for each dataset. Then, we train MultiBoost on one text file, after which it will generate a model for each instrument within this text file. Finally, we can provide the second text file to MultiBoost and it will assign an instrument label to each of the entries in this second text file.

Because we know the correct instrument label for each recording of each dataset, we can compare the labels that were assigned by MultiBoost to the correct labels that we have manually assigned.

To measure the instrument recognition performance, we derive a confusion matrix which shows for each instrument the number of correctly assigned notes as well as the number of incorrectly assigned notes, grouped by instrument. We briefly revisit the example from section 1.6:

	Bassoon	Clarinet	Flute	Horn	Oboe	
Bassoon	**10**	2	1	3	1	59%
Clarinet	4	**4**	2	3	2	27%
Flute	1	2	**5**	4	3	33%
Horn	0	2	1	**3**	5	27%
Oboe	2	1	0	3	**6**	50%

Table 6.5: Confusion matrix example

The bold numbers in this matrix are notes that are assigned to the correct instrument. The other numbers indicate incorrect labels, allowing us to see which instruments cause the most confusion. The percentages in the last column indicate the number of successful classifications.

6.6.1 List of thresholds and parameters

Experimenting with timbre features is a time-consuming process as there are many possible combinations of features that can be used. Ideally, one could set up a system that derives the importance of each feature individually as in [23] and [30]. The features that were ultimately chosen for our experiments have been determined by manual experiments, therefore we expect there to be room for improvement.

Parameter	Value	Implementation
Window size	20 ms	All
Time-frequency scale	ERB (filterbank)	All
Number of iterations for training	100	MultiBoost
Timbre features	Tristimulus, relative harmonics, odd/even ratio, spectral centroid, spectral spread, kurtosis, HFCC	TrainedVERB
Note phase	Attack and steady-state (decay-sustain-release)	All but kurtosis
Note phase	steady-state (decay-sustain-release)	kurtosis
Number of harmonics considered	10	Odd/even ratio

Table 6.6: Parameters for instrument recognition using timbre descriptors

6.7 Instrument recognition using NMF with instrument annotations

By including the instrument that was used to create each note recording in the training set, we know which instruments were ultimately used to do the reconstruction of a song. In the training process, we store the instrument from which the spectral envelope was derived in the same way as how we store the pitch value. As an example, imagine that we have a recording of a clarinet that is playing a $C4$ note (pitch value: 40). We then name the recording 40.wav and place it in the clarinet folder so that our training data generation code stores this information with the spectral envelope.

If we then apply *TrainedLim* to a musical piece, we can derive the activity of a certain instrument at a certain pitch and timestep by summing up the activity in A of all spectral envelopes that belong to that pitch value at that timestep. In short, our instrument activity is stored in a 3-dimensional matrix of size $88 \times number of instruments \times number of timesteps$. If we then want to know which instrument had the most activity at the $C4$ note, 2 seconds into the musical piece, we can find this by summing up the activity in $A(2seconds)$ of all spectral envelopes that describe a $C4$ note.

We need to sum up over multiple envelopes because it is possible for more envelopes to describe the same note and instrument as we are combining note recordings from the RWC and Iowa datasets (so most instrument and pitch combinations will exist at least twice). It is also possible to derive more than one envelope for each recording as is the case with *TrainVERB3*.

Similar to the previous experiment, we again use a confusion matrix to evaluate the instrument recognition results. Each entry in the confusion matrix is a musical note, while our active instrument data is measured for each timestep. To group this instrument data into notes without relying on frequency estimation performance, we use the ground truth data supplied with the MIREX dataset.

For example, if the ground truth states that a $C4$ note is active from 1 second into the musical piece until 2 seconds into the musical piece, then we consider the active instruments during this time period at note $C4$ and sum up all the activity through time. The instrument that has the most total activity for the duration of the note is considered to be the instrument that was mostly used to recreate this note. The note is then assigned to this instrument in the confusion matrix.

6.7.1 List of thresholds and parameters

Parameter	Value	Implementation
Window size	20 ms	All
Beta-divergence	0.5	All
Local convergence threshold	0.005	All
Global convergence threshold	0.01	All
Musical piece used	MIREX set, number of voices 5	All
Time-frequency scale	ERB (filterbank)	All

Table 6.7: Parameters for NMF-based instrument recognition

6.8 Instrument recognition using timbre descriptors of notes extracted from a musical piece using NMF

This experiment is similar to what we have done in section 6.6. We use the same dataset for training, namely the spectrograms from *TrainedVERB*. For this experiment however, the database is not divided into two subsets, we instead use the whole set for training.

For testing, we extract spectrograms of single notes from a musical piece containing at most five voices. This musical piece is taken from the *MIREX* dataset. NMF allows us to extract specific parts of a musical piece, as A contains activations of each spectral envelope in S at each point in time. For the spectral envelopes in S, we know the pitch values. So if we then know that a $C4$ note was active in the musical piece between 1 and 3 seconds into the recording, we can take all spectral envelopes from S that belong to pitch value $C4$ and multiply them with the activations in A for the time between 1 and 3 seconds. This results in a spectrogram that contains only those 2 seconds of a $C4$ note. We can then store these spectrograms and use them as input for our MultiBoost instrument recognition model, trained on *TrainedVERB*.

To remove the dependency on frequency estimation performance, we extract the individual notes by using the ground truth data from the *MIREX* dataset. The list of frequencies contained within this ground truth allows us to define notes with a certain start and end time, by considering the activity of each frequency over time.

The main goal of this experiment is to determine whether we can use NMF to separate a musical piece into its individual notes, which can then be used for instrument recognition. Because the notes are not recorded individually and are instead extracted from a musical piece, we expect some loss in quality to occur. This experiment will show if it is still possible to tell the difference between instruments from these imperfect notes.

6.8.1 List of thresholds and parameters

Parameter	Value	Implementation
Window size	20 ms	All
Time-frequency scale	ERB (filterbank)	All
Number of iterations for training	100	MultiBoost
Timbre features	Tristimulus, relative harmonics, odd/even ratio, spectral centroid, spectral spread, kurtosis, HFCC	TrainedVERB
Note phase	Attack and steady-state (decay-sustain-release)	All but kurtosis
Note phase	steady-state (decay-sustain-release)	kurtosis
Number of harmonics considered	10	Odd/even ratio

Table 6.8: Parameters for instrument recognition with notes extracted from a musical piece

6.9 Comparison of time-frequency scales

Because the generation of spectral envelopes in *VincentOrig* and *VincentLim* is based upon the specific ERB scale implementation used in this methods, we could not simply convert these methods to a linear scale for fundamental frequency estimation. *VincentLim* is therefore provided merely as a reference.

For our fundamental frequency comparison using different time-frequency scales, we use *TrainedLim*, with *TrainedVERB* as its training data. As alternatives, we consider the same *TrainedLim* technique with two different training sets: *TrainedGLin* with a linear scale and *TrainedGERB* with an alternative ERB scale. The *MIREX* musical pieces are used for testing.

The results are measured in the same way is in our previous experiment (see: section 6.4), where we derive the recall, precision and F-measure from the output of each method and the ground-truth data of the *MIREX* dataset. Our emphasis is still on recall as we still manually limit the number of returned frequencies.

Besides experimenting with fundamental frequency estimation, we also explore the differences when using a linear scale for the task of instrument recognition, using either the NMF implementation explored in section 6.7 or the timbre feature based variation discussed in section 6.6. The only variation we have introduced in this case is the time-frequency scale.

6.9.1 List of thresholds and parameters

As in the previous experiment, we base our window size on the findings by Vincent et al. For time-frequency scales other than the ERB scale, Vincent et al. suggest a 46 ms window size as "it is close to the effective time resolution of the ERB filterbank at the fundamental frequency corresponding to the average observed pitch".

For each window size variation the same window size used in the NMF method was used in generating the training set.

Parameter	Value	Implementation
Window size	20 ms	VincentLim, TrainedVERB
Window size	46 ms	TrainedGERB, TrainedGLin
Window hop	23 ms	TrainedGERB, TrainedGLin
Beta-divergence	0.5	All
Local convergence threshold	0.005	All
Global convergence threshold	0.01	All
Number of returned frequencies	Number of voices + 4	All
Time-frequency scale	ERB (filterbank)	TrainedVERB
Time-frequency scale	Linear	TrainedGLin
Time-frequency scale	ERB (from linear)	TrainedGERB
Timbre features	Tristimulus, relative harmonics, odd/even ratio, spectral centroid, spectral spread, kurtosis, **HFCC**	TrainedVERB
Timbre features	Tristimulus, relative harmonics, odd/even ratio, spectral centroid, spectral spread, kurtosis	TrainedGLin
Note phase	Attack and steady-state (decay-sustain-release)	All but kurtosis
Note phase	steady-state (decay-sustain-release)	kurtosis
Number of harmonics considered	10	Odd/even ratio

Table 6.9: Parameters for comparing time-frequency scales

6.10 Using 3 spectral envelopes per training sample instead of 1

We have discussed the procedure of generating a collection S that contains more than one spectral envelope for each training note in section 5.7. We expect the range of possible spectral envelopes that can be described using this bigger S to increase, hopefully resulting in a reconstruction spectrogram that more closely resembles the spectrogram of the musical piece.

To test this assumption for frequency estimation, we run *TrainedLim* using two different training sets: *TrainedVERB* and *TrainedVERB3*. As in the previous experiments, the *MIREX* set is used as input and the results from *TrainedLim* are compared with the MIREX ground truth data.

We also evaluate the difference in NMF-based instrument recognition performance, by repeating the experiments from section 6.7 using *TrainedVERB3*.

6.10.1 List of thresholds and parameters

Parameter	Value	Implementation
Window size	20 ms	All
Beta-divergence	0.5	All
Local convergence threshold	0.005	All
Global convergence threshold	0.01	All
Number of returned frequencies	Number of voices + 4	All
Time-frequency scale	ERB (filterbank)	All

Table 6.10: Parameters for evaluating the number of spectral envelopes per note

6.11 Introducing our own implementation of NMF

We test the performance of the *WitLim* implementation by comparing it to *TrainedLim*. Both methods use training data and should apply the same basic unconstrained NMF technique. A comparison is done using two different datasets: *TrainedVERB* and *TrainedGLin*. Tests are conducted on the MIREX dataset, where we run the same frequency estimation task that we have applied in the previous experiments.

In this experiment, obtaining frequency estimation results using *WitLim* that are equal to or better than *TrainedLim* means that our own implementation is ready to be used in future research. We expect this to indeed be the case, therefore in the following experiments we will extend the *WitLim* method to include temporal information in the form of notes.

6.11.1 List of thresholds and parameters

Parameter	Value	Implementation
Window size	20 ms	TrainedVERB
Window size	46 ms	TrainedGLin
Window hop	23 ms	TrainedGLin
Beta-divergence	0.5	All
Local convergence threshold	0.005	All
Global convergence threshold	0.01	All
Number of returned frequencies	Number of voices + 4	All
Time-frequency scale	ERB (filterbank)	TrainedVERB
Time-frequency scale	Linear	TrainedGLin

Table 6.11: Parameters for the comparison of *WitLim* and *TrainedLim*

6.12 Using note events instead of single timesteps for fundamental frequency estimation

Because we are now moving towards a complete transcription system, this is where we start using our own method, *WitLim*, for our experiments. We use *TrainedGLin* for training data.

In order to extract note events, we need to apply onset detection. The challenge of such an onset detection system lies in the choice of its threshold parameters. If these thresholds are set too low, thresholds are detected where no note is played. If they are set too high then we risk missing note onsets, resulting in the removal of correct frequencies.

Onset detection is applied to the spectrogram belonging to each pitch value separately. This means that for each possible note (out of 88), we get a list of onset locations through time. This is possible because the factorization from NMF allows us to multiply only the spectral envelopes that belong to a certain pitch value by their activations to get the spectrogram of just that pitch value. This is similar to how we have obtained separate notes from a musical piece in section 6.8.

To get from a list of onset points to start and end times of notes, we examine the relative amplitude envelope of each pitch value. We can generate this relative envelope by dividing each amplitude value by the overall maximum amplitude of that particular pitch value. If we then take the derivative of this envelope, we get the rate of change in amplitude through time. By then moving back in time from an onset point, we can obtain the nearest point where the amplitude stabilizes - the derivative is below a certain threshold, which is likely to be the start time of the note. The same process can be applied in the other direction to find the end time of the note.

Our first experiments with note events returned a fixed number of most active notes, similar to how our previous experiments returned the active frequencies. However, this drastically reduced recall as slight variations in frequency were no longer included. Therefore, in these experiments we have opted to again return the most active frequencies, but each frequency will have to be within two pitch values of a note that was detected.

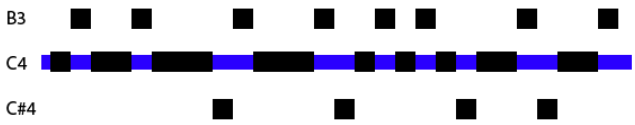

Figure 6.1: Assigning frequency variations (black) to a note event (blue)

6.12.1 List of thresholds and parameters

The parameters for onset detection, a and b, determine the trade-off between returning many onset locations, some of which will be incorrect or divide a longer note into smaller notes, and returning little onset locations of which most will be correct but where some correct notes may be missed.

Parameter	Value	Implementation
Window size	46 ms	All
Window hop	23 ms	All
Beta-divergence	0.5	All
Local convergence threshold	0.005	All
Global convergence threshold	0.01	All
Number of returned frequencies	Number of voices + 4	All
Time-frequency scale	Linear	All
Exponential decay a (onset detection)	0.9	All
Local mean offset b (onset detection)	0.05	All
Note start/end relative amplitude threshold	0.1	All

Table 6.12: Parameters for deriving notes

6.13 Refining note events using post-processing

The following experiments are conducted using the *WitLim* implementation and the *TrainedGLin* dataset.

Upon defining note events in our previous experiment, besides retrieving the correct notes we have also obtained some notes that belong to noise and should not be included in the final output. We can attempt to remove these incorrect notes by imposing certain requirements onto each potential note, for example expecting a minimum duration or amplitude.

When generating note events from the timesteps as we have done in the previous experiment, we can store the duration, frequency and amplitude with each note because all these values are known. We already retrieved the frequency for each timestep using NMF, where the activations in A can be summed up for that particular timestep and frequency to provide an indication of the amplitude. The duration is based on the start and end times of the note, which are retrieved by using onset detection. We define the amplitude of a note by summing up the amplitudes at its frequency for each timestep that is within the note duration. We then divide this sum by the number of timesteps that are within the duration of the note, resulting in the average amplitude for each timestep. This allows a comparison of amplitudes between notes, so that a threshold can be applied.

Besides duration and amplitude thresholds, we also examine the relative amplitude at each timestep. When two notes are sounding at the same time, it is possible that one of the notes cannot be heard simply because of the presence of the other note, even though the note would have been loud enough to be observed if it had been played alone. To incorporate this phenomenon into our framework, we examine the amplitude of each active frequency relative to the frequency with the most amplitude at each timestep. This is done for each timestep instead of each note event as there is the possibility of only parts of notes overlapping, which should not result in complete removal of one note as it may be audible at a later point in time.

6.13.1 List of thresholds and parameters

Parameter	Value	Implementation
Window size	46 ms	All
Window hop	23 ms	All
Beta-divergence	0.5	All
Local convergence threshold	0.005	All
Global convergence threshold	0.01	All
Number of returned frequencies	Number of voices + 4	All
Time-frequency scale	Linear	All
Exponential decay a (onset detection)	0.9	All
Local mean offset b (onset detection)	0.05	All
Note start/end relative amplitude threshold	0.1	All
Duration threshold	50 ms, 100 ms 200 ms	All
Absolute amplitude per timestep threshold	5, 10, 15, 20	All
Relative amplitude threshold	30%, 40% of maximum amplitude	All

Table 6.13: Parameters for refining notes

6.14 Using instrument recognition to enhance fundamental frequency estimation performance

We continue our tests using *WitLim* and *TrainedGLin*, including the post-processing steps from the previous experiment. We now move towards *WitFinal*, where the manually fixed number of returned frequencies is replaced by an estimation of the number of voices that are expected to be active. We base this estimation on prior knowledge of the instruments present in the song and the number of notes each instrument will generally play at the same time.

Because we do not want to depend on the results of the instrument recognition method of section 6.8, we train MultiBoost on the ground truth data that in section 6.8 was used as test data. This means that the instrument recognition step is performed using a model that is trained on notes extracted from the same musical piece that is now used for testing. Therefore, the instrument recognition results should be (near) perfect, allowing us to test the benefit of having instrument data as prior knowledge for frequency estimation.

Instruments were determined to be present in the musical piece if their total energy made up at least 10% of all energy present in the musical piece. The number of frequencies originating from each instrument at each timestep was stored, from which we took the number that was mostly found as the number of voices for that instrument. So if most timesteps had two frequencies of a flute active, then we assume that the flute had a maximum number of voices of two.

6.14.1 List of thresholds and parameters

Parameter	Value	Implementation
Window size	46 ms	All
Window hop	23 ms	All
Beta-divergence	0.5	All
Local convergence threshold	0.005	All
Global convergence threshold	0.01	All
Number of returned frequencies	Number of voices + 4	All
Time-frequency scale	Linear	All
Exponential decay a (onset detection)	0.9	All
Local mean offset b (onset detection)	0.05	All
Note start/end relative amplitude threshold	0.1	All
Duration threshold	100 ms	All
Absolute amplitude per timestep threshold	15	All
Relative amplitude threshold	30% of maximum amplitude	All
Required instrument energy	10% of musical piece energy	All

Table 6.14: Parameters for combining instrument recognition and frequency estimation

Chapter 7

Results

7.1 Multiple fundamental frequency estimation using NMF

With this test, we would like to evaluate the performance of using NMF as a multiple fundamental frequency detection method. By using *VincentOrig* as a reference, which uses post-processing to improve precision, we can get an impression of the amount of recall that is sacrificed by this method to obtain a higher precision.

Note that the precision for *VincentLim* and *TrainedLim* is not realistic as the number of expected frequencies is fixed. Our main focus however is on the recall.

Max voices	1			2			3			4			5		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
VincentOrig	**94**	40	56	**93**	58	71	**81**	46	58	**76**	58	66	**64**	63	64
VincentLim	**96**	18	30	**97**	28	44	**86**	31	45	**85**	33	48	**76**	34	47
TrainedLim	**98**	18	30	**97**	28	44	**89**	32	47	**89**	35	50	**82**	36	50

Table 7.1: Frequency estimation results on musical recordings with 1-5 voices. Results are listed as: recall/precision/ F-measure in percentages

For musical pieces with only few voices, there is no substantial difference between *VincentLim* and the trained version *TrainedLim*, which is impressive considering the small amount of 88 spectral envelopes used in *VincentLim*. The harmonic constraints apparently succeed in creating a complete spectral envelope for each pitch value.

However, when including more voices the difference between the harmonically constrained *VincentLim* and the alternative based on training data, *TrainedLim*, increases. In this case, the training set *TrainedVERB* that was used includes all instruments that are present in the musical pieces contained within the *MIREX* test set. Regardless of the method used, we can conclude that NMF can indeed be applied to the task of multiple fundamental frequency estimation.

7.2 Impact of training data instrument composition for fundamental frequency estimation

To evaluate the extent to which the the frequency estimation performance depends on the instruments used for generating the spectral envelopes, we created two subsets from the *TrainedVERB* set: *TrainedVERBcfbho* containing exactly those five instruments present in the test data, *TrainedVERBother* containing the remaining six instruments from *TrainedVERB*. Note that some instruments in the second group belong to the same instrument families as those in the first.

	1			2			3			4			5		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
TrainedVERB (TrainedLim)	**98**	18	30	**97**	28	44	**89**	32	47	**89**	35	50	**82**	36	50
TrainedVERBcfbho	**99**	18	30	**98**	28	44	**90**	32	47	**90**	35	50	**84**	37	51
TrainedVERBother	**98**	18	30	**96**	28	43	**87**	31	46	**86**	34	48	**78**	35	48

Table 7.2: Frequency estimation results on musical recordings with 1-5 voices with *TrainedVERBcfbho* containing the instruments of the musical piece and *TrainedVERBother* containing other instruments. Results are listed as: recall/precision/ F-measure in percentages

It seems that the presence in the training set of the same instruments used to create the musical piece does improve the frequency estimation, however this difference is not as big as one would probably expect it to be. Remember that only five different instruments (with two separate recordings of each instrument) were used to generate the datasets. We expect the dataset, given enough samples of each instrument *type* to be able to generate the characteristic spectral envelope of instruments not present in the dataset by combining the envelopes of several trained instruments. Note that the performance of the dataset containing instruments different than those used for the musical piece is still close to the harmonic NMF implementation listed in table 7.1.

7.3 Isolated note instrument recognition using timbre descriptors

We test classification performance using a number of features described in section 6.6 on sets of isolated notes. Two subsets of the *TrainedVERB* dataset are used, one containing only the samples from the RWC dataset and the other containing the samples from the University of Iowa. Both subsets contain the same eleven instruments. We first use one subset for training and the other for testing, then vice versa.

The following confusion matrix shows the results when using the RWC samples for training and the University of Iowa samples for testing:

	Alt sax	Bass	Bassoon	Cello	Clarinet	Flute	Horn	Oboe	Piano	Tuba	Violin	
Alt sax	**3**	0	3	0	0	22	0	0	0	0	4	9%
Bass	0	**21**	0	2	0	0	0	0	0	0	0	91%
Bassoon	0	0	**41**	0	0	0	0	0	0	0	0	100%
Cello	0	20	6	**5**	0	0	0	0	0	0	0	16%
Clarinet	0	3	0	0	**8**	28	0	1	2	0	4	19%
Flute	0	0	0	0	2	**33**	0	0	0	0	0	85%
Horn	0	1	3	0	0	0	**28**	0	0	0	0	87%
Oboe	1	1	0	0	3	10	0	**20**	0	0	0	57%
Piano	0	52	0	9	0	0	1	0	**8**	16	0	9%
Tuba	0	0	0	35	0	0	0	0	0	**2**	0	5%
Violin	11	0	0	10	1	0	1	0	1	0	**0**	0%

Table 7.3: Instrument recognition confusion matrix, using RWC as training set and Iowa as testing. Each entry is a note classification

We then swap the two datasets and use the University of Iowa samples for training, while the RWC samples are used for testing:

	Alt sax	Bass	Bassoon	Cello	Clarinet	Flute	Horn	Oboe	Piano	Tuba	Violin	
Alt sax	**16**	0	0	2	0	7	1	0	0	0	7	48%
Bass	0	**19**	0	3	6	0	0	0	8	0	0	53%
Bassoon	3	1	**32**	1	0	1	1	0	0	0	0	82%
Cello	0	20	0	**4**	10	2	0	0	3	7	0	9%
Clarinet	0	0	0	0	**37**	3	0	0	0	0	0	92%
Flute	1	0	0	0	22	**8**	1	5	0	0	0	22%
Horn	0	0	8	5	0	0	**23**	0	1	0	0	62%
Oboe	0	0	0	0	19	0	0	**15**	0	0	0	44%
Piano	1	4	6	11	53	2	0	4	**2**	0	5	2%
Tuba	0	0	7	0	0	0	0	0	23	**0**	0	0%
Violin	31	0	0	1	5	9	0	0	0	0	**0**	0%

Table 7.4: Instrument recognition confusion matrix, using Iowa as training set and RWC as testing. Each entry is a note classification

While performance is competitive for some instruments, the classifier still struggles with others. For example the bassoon and horn perform well with both dataset combinations. The performance differs greatly after switching the datasets, indicating that either the feature set or the number of samples is not sufficient to correctly describe the unique properties of each instrument. Performance of using the RWC subset as training data and the University of Iowa subset is superior to the reversed scenario.

7.4 Instrument recognition using NMF with instrument annotations

In the following experiment, we use *TrainedVERB*. We consider the entire note duration and find the instrument that has the most energy in the pitch value belonging to the note:

	Alt sax	Bass	Bassoon	Cello	Clarinet	Flute	Horn	Oboe	Piano	Tuba	Violin	
Bassoon	0	1	**139**	3	0	0	18	0	0	50	0	69%
Clarinet	0	0	71	4	**10**	0	96	0	10	18	0	5%
Flute	0	0	0	0	255	**61**	1	44	0	0	0	17%
Horn	0	0	0	0	0	0	**27**	0	0	4	0	87%
Oboe	0	0	0	0	14	3	154	**42**	0	0	0	20%

Table 7.5: Confusion matrix for NMF-based instrument recognition using ERB scale with training set of 1 envelope for each sample. Each entry is a note classification

It appears from these results that the NMF process is not guaranteed to prefer the spectral envelopes that are learned from the same instruments that were used in the musical piece to reconstruct the time-frequency representation of this recording, although we can see a tendency towards instruments that have similar character. We believe that timbre is too complex to just assume that the factorization process will automatically pick those envelopes belonging to the same instrument when it is possible to do a similar or possibly better reconstruction using a combination of envelopes from several instruments.

Regardless, this experiment shows that in reconstructing a musical piece, in many cases the original instrument or a similar instrument is used. The performance approaches that of other methods for polyphonic music (discussed in section 4.2), although it is surpassed by recent techniques involving musical knowledge or an estimation of overlapping energy. To obtain a competitive instrument recognition performance some form of further analysis will be required beyond simply taking the instrument whose spectral envelopes were mostly used for the reconstruction.

For this method of instrument recognition to work, each instrument present in the musical piece needs to be present in the training data as well. This increase in training data will affect the performance of the factorization process, while a smaller dataset may provide similar pitch estimation performance.

7.5 Instrument recognition using timbre descriptors of notes extracted from a musical piece using NMF

This experiment is similar to what we have done using isolated note recordings. However, while we still train MultiBoost on isolated note recordings we now test the instrument recognition using notes extracted from a musical piece, to see how well the classifier is able to match instruments regardless of the recording environments used. This means we are working with imperfect data, as the notes used for testing are extracted from a musical piece in which a combination of notes may be played at the same time.

	Alt sax	Bass	Bassoon	Cello	Clarinet	Flute	Horn	Oboe	Piano	Tuba	Violin	
Bassoon	0	0	**73**	1	0	1	30	0	21	24	0	49%
Clarinet	2	5	13	9	**5**	15	65	0	6	8	1	4%
Flute	0	0	1	0	83	**100**	0	59	0	0	1	41%
Horn	0	0	2	0	0	2	**10**	0	1	1	0	62%
Oboe	2	0	3	0	7	50	13	**38**	1	0	8	31%

Table 7.6: Instrument recognition confusion matrix, using Iowa and RWC as training set and notes extracted using NMF as testing. Each entry is a note classification

Unfortunately, we were unable to obtain results similar to what we have seen in other research. We suspect that this is caused by the quality of our implementation of extracting and combining the features as well as the smaller size of our datasets. We regret that due to time constraints we were unable to further investigate and solve these low success rates.

The results are interesting because we expect the quality of such extracted notes to be much worse than the quality of isolated recordings, which should result in a lot of confusion in trying to determine the correct instrument. The results discussed in this section indicate that it should be possible to recover the instrument used to create notes, even if these notes were mixed in a musical piece and then extracted using NMF.

7.6 Comparison of time-frequency scales

We now move to an evaluation of the influence of different time-frequency scales on multiple fundamental frequency estimation and instrument recognition results.

7.6.1 Fundamental frequency estimation

We compare the linear time-frequency scale to two different ERB implementations. The table below lists the different training sets used in our evaluation. Note that the first entry, *TrainedVERB*, is identical to the *TrainedLim* that we have seen in section 7.1. It is named differently because we are now comparing datasets instead of technical implementations.

Max voices	1			2			3			4			5		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
VincentLim	**96**	18	30	**97**	28	44	**86**	31	45	**85**	33	48	**76**	34	47
TrainedVERB (TrainedLim)	**98**	18	30	**97**	28	44	**89**	32	47	**89**	35	50	**82**	36	50
TrainedGLin	**96**	18	30	**97**	28	43	**84**	30	44	**85**	33	47	**80**	36	49
TrainedGERB	**80**	15	25	**85**	25	38	**66**	24	35	**65**	25	36	**59**	26	36

Table 7.7: Multiple fundamental frequency estimation with different time-frequency scales. Results are listed as: recall/precision/ F-measure in percentages

Table 7.7 shows that the linear scale performs slightly worse than the ERB implementation by Vincent (*TrainedVERB*). Note that the dimensionality of the linear scale is higher due to the larger window and hop sizes. The linear scale contains 1025 frequency bins, whereas *TrainedVERB* contains 250.

The alternative ERB implementation by Ellis (*TrainedGERB*) seems to perform considerably worse than both *TrainedGLin* and *TrainedVERB*. The output of *TrainedGERB* contained many frequencies that were one pitch value higher or lower than the correct frequencies. We suspect that this is due to the conversion from linear to ERB scale used in this method. If the linear scale does not have sufficient detail - small frequency bins - then information will be lost upon converting this scale to ERB. We have attempted to run *TrainedGERB* with a smaller window and hop size of 20 ms and 10 ms respectively to match the parameters of *TrainedVERB*. However, this had a negative impact on the measured results.

Note that by deriving the recall, precision and F-measure we do not directly measure the amplitudes that are assigned to the returned frequencies. It is possible, for instance, that the difference in amplitude between the correct and incorrect frequencies is bigger in *TrainedVERB* than in *TrainedGLin*. This means that the correct frequencies will appear higher on the list of returned frequencies, making it less likely for correct frequencies to be removed.

By manually examining the outputs of *TrainedVERB* and *TrainedGLin*, we see that the correct frequencies do seem to appear higher on the amplitude list of *TrainedVERB*. The relative amplitudes of returned frequencies is something we would like to consider in future experiments as a different measure of frequency estimation performance.

7.6.2 NMF-based instrument recognition

We have evaluated the performance of instrument recognition using a linear time-frequency transform. *TrainedGERB* was not considered in this section due to its relatively poor performance in the fundamental frequency estimation task. To allow a comparison with *TrainedVERB*, we first repeat the confusion matrix from section 7.4:

	Alt sax	Bass	Bassoon	Cello	Clarinet	Flute	Horn	Oboe	Piano	Tuba	Violin	
Bassoon	0	1	**139**	3	0	0	18	0	0	50	0	69%
Clarinet	0	0	71	4	**10**	0	96	0	10	18	0	5%
Flute	0	0	0	0	255	**61**	1	44	0	0	0	17%
Horn	0	0	0	0	0	0	**27**	0	0	4	0	87%
Oboe	0	0	0	0	14	3	154	**42**	0	0	0	20%

Table 7.8: Confusion matrix for NMF-based instrument recognition using ERB scale with training set of 1 envelope for each sample. Each entry is a note classification

We now run the same test using *TrainedGLin*:

	Alt sax	Bass	Bassoon	Cello	Clarinet	Flute	Horn	Oboe	Piano	Tuba	Violin	
Bassoon	0	2	**122**	0	0	0	2	0	0	75	0	61%
Clarinet	0	0	69	0	**22**	0	72	0	0	36	0	11%
Flute	0	0	0	0	255	**104**	1	1	0	0	0	29%
Horn	0	0	0	0	0	0	**22**	0	0	9	0	71%
Oboe	0	0	10	0	14	3	129	**57**	0	0	0	27%

Table 7.9: Confusion matrix for NMF-based instrument recognition using linear scale with training set of 1 envelope for each sample. Each entry is a note classification

The results for both methods are similar, although the incorrect instrument classifications seem to be less spread out across the different instruments in the linear scale. Overall, it seems that the linear scale performs slightly better as well. The percentage of correctly classified *horn* samples is over 15% lower, however the number of horn samples present in the musical piece is so little that a few incorrect classifications have a big impact on the success rate.

7.6.3 Instrument recognition using timbre descriptors

To see if the linear and ERB scales indeed describe different timbre properties, we use *TrainedVERB* and *TrainedGLin* to perform instrument recognition with timbre features. We have already done this experiment using *TrainedVERB* in section 7.3, where the dataset was split into two subsets: one containing only the University of Iowa recordings and one containing only the RWC dataset samples.

As a reference point, our results of training on the RWC subset of *TrainedVERB* and testing with the University of Iowa subset of *TrainedVERB*, as obtained in section 7.3, were:

	Alt sax	Bass	Bassoon	Cello	Clarinet	Flute	Horn	Oboe	Piano	Tuba	Violin	
Alt sax	**3**	0	3	0	0	22	0	0	0	0	4	9%
Bass	0	**21**	0	2	0	0	0	0	0	0	0	91%
Bassoon	0	0	**41**	0	0	0	0	0	0	0	0	100%
Cello	0	20	6	**5**	0	0	0	0	0	0	0	16%
Clarinet	0	3	0	0	**8**	28	0	1	2	0	4	19%
Flute	0	0	0	0	2	**33**	0	0	0	0	0	85%
Horn	0	1	3	0	0	0	**28**	0	0	0	0	87%
Oboe	1	1	0	0	3	10	0	**20**	0	0	0	57%
Piano	0	52	0	9	0	0	1	0	**8**	16	0	9%
Tuba	0	0	0	35	0	0	0	0	0	**2**	0	5%
Violin	11	0	0	10	1	0	1	0	1	0	**0**	0%

Table 7.10: Instrument recognition confusion matrix, using *TrainedVERB* with RWC as training set and Iowa as testing. Each entry is a note classification

The results of running the same experiment with *TrainedGLin* are:

	Alt sax	Bass	Bassoon	Cello	Clarinet	Flute	Horn	Oboe	Piano	Tuba	Violin	
Alt sax	**6**	0	0	5	1	6	0	2	6	0	6	19%
Bass	0	**12**	3	0	0	0	0	0	8	0	0	52%
Bassoon	0	0	**26**	0	0	0	3	0	12	0	0	63%
Cello	0	0	2	**0**	0	3	0	0	26	0	0	0%
Clarinet	1	0	0	1	**6**	24	0	2	2	0	6	14%
Flute	1	0	0	0	2	**24**	0	0	1	0	11	62%
Horn	0	0	6	0	0	0	**25**	0	1	0	0	78%
Oboe	6	0	1	0	0	18	0	**4**	0	0	6	11%
Piano	0	1	2	0	2	19	29	0	**29**	0	4	34%
Tuba	15	1	0	19	0	1	0	0	1	**0**	0	0%
Violin	3	0	0	0	0	2	0	1	2	0	**16**	67%

Table 7.11: Instrument recognition confusion matrix, using *TrainedGLin* with RWC as training set and Iowa as testing. Each entry is a note classification

We have done the same test using the Iowa subset for training and the RWC subset for testing. We first list our results using *TrainedVERB*:

	Alt sax	Bass	Bassoon	Cello	Clarinet	Flute	Horn	Oboe	Piano	Tuba	Violin	
Alt sax	**16**	0	0	2	0	7	1	0	0	0	7	48%
Bass	0	**19**	0	3	6	0	0	0	8	0	0	53%
Bassoon	3	1	**32**	1	0	1	1	0	0	0	0	82%
Cello	0	20	0	**4**	10	2	0	0	3	7	0	9%
Clarinet	0	0	0	0	**37**	3	0	0	0	0	0	92%
Flute	1	0	0	0	22	**8**	1	5	0	0	0	22%
Horn	0	0	8	5	0	0	**23**	0	1	0	0	62%
Oboe	0	0	0	0	19	0	0	**15**	0	0	0	44%
Piano	1	4	6	11	53	2	0	4	**2**	0	5	2%
Tuba	0	0	7	0	0	0	0	0	23	**0**	0	0%
Violin	31	0	0	1	5	9	0	0	0	0	**0**	0%

Table 7.12: Instrument recognition confusion matrix, using *TrainedVERB* with Iowa as training set and RWC as testing. Each entry is a note classification

And using *TrainedGLin*:

	Alt sax	Bass	Bassoon	Cello	Clarinet	Flute	Horn	Oboe	Piano	Tuba	Violin	
Alt sax	**8**	0	0	1	0	3	0	12	0	6	3	24%
Bass	0	**23**	0	3	1	5	0	0	0	4	0	64%
Bassoon	1	3	**22**	5	2	0	1	0	3	0	2	56%
Cello	5	16	0	**1**	6	4	0	1	0	8	5	2%
Clarinet	6	0	0	4	**26**	1	0	1	0	0	2	65%
Flute	2	0	1	0	11	**14**	0	4	1	0	4	38%
Horn	2	0	3	0	3	0	**28**	0	1	0	0	76%
Oboe	1	0	0	0	9	0	2	**20**	0	0	2	59%
Piano	19	9	7	15	25	5	2	0	**0**	3	3	0%
Tuba	1	15	2	0	0	0	10	0	2	**0**	0	0%
Violin	3	0	0	3	11	3	0	3	1	0	**22**	48%

Table 7.13: Instrument recognition confusion matrix, using *TrainedGLin* with Iowa as training set and RWC as testing. Each entry is a note classification

The results when using a linear scale show a strong tendency towards the plucked instruments. To test whether combining the two scales results in better classification results, we have combined the timbre descriptors from the linear and ERB tests and ran the classification process on this combined set. While the results were better than those obtained using the linear scale alone, they were worse than the results using the ERB scale.

7.7 Using 3 spectral envelopes per training sample instead of 1

Similar to our experiments in section 7.6, we evaluate the impact on both frequency estimation and NMF-based instrument recognition of generating three spectral envelopes for each isolated note sample in the training set instead of one.

7.7.1 Frequency estimation

The following results were obtained by running NMF on the *MIREX* dataset using two different sets for training: *TrainedVERB* where a single envelope was derived from each note recording and *TrainedVERB3* where three envelopes were taken.

Max voices	1			2			3			4			5		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
TrainedVERB	**98**	18	30	**97**	28	44	**89**	32	47	**89**	35	50	**82**	36	50
TrainedVERB3	**97**	15	26	**97**	28	44	**88**	31	46	**88**	34	49	**81**	36	50

Table 7.14: Difference between generating training data using one or three envelopes per training note. Results are listed as: recall/precision/ F-measure in percentages

Table 7.14 shows no difference between using one or three envelopes. It seems that by deriving more spectral envelopes we are unable to provide a more complete description of the character of a note, at least for the goal of frequency estimation. Furthermore, we now believe that by including the spectral structure of different note phases we might actually introduce more uncertainty, as most of the added spectral envelopes will describe phenomena such as vibrato and characteristic attack phases, which both have unstable frequency behaviour. The added envelopes may therefore describe a frequency other than the fundamental that is assigned to the note.

7.7.2 NMF-based instrument recognition

We now consider using *TrainedVERB3* for the task of NMF-based instrument recognition. As a reference, we use our results from section 7.4, where we have used *TrainedVERB* for training:

	Alt sax	Bass	Bassoon	Cello	Clarinet	Flute	Horn	Oboe	Piano	Tuba	Violin	
Bassoon	0	1	**139**	3	0	0	18	0	0	50	0	69%
Clarinet	0	0	71	4	**10**	0	96	0	10	18	0	5%
Flute	0	0	0	0	255	**61**	1	44	0	0	0	17%
Horn	0	0	0	0	0	0	**27**	0	0	4	0	87%
Oboe	0	0	0	0	14	3	154	**42**	0	0	0	20%

Table 7.15: Confusion matrix for NMF-based instrument recognition using ERB scale with training set of 1 envelope for each sample. Each entry is a note classification

If we then train on *TrainedVERB3*, we get the following results:

	Alt sax	Bass	Bassoon	Cello	Clarinet	Flute	Horn	Oboe	Piano	Tuba	Violin	
Bassoon	0	0	**138**	5	3	0	29	0	0	26	0	69%
Clarinet	12	0	55	4	**42**	0	58	0	0	18	10	21%
Flute	0	0	0	0	179	**137**	1	44	0	0	0	38%
Horn	0	0	0	0	4	0	**23**	0	0	4	0	74%
Oboe	0	0	0	0	15	4	152	**42**	0	0	0	20%

Table 7.16: Confusion matrix for NMF-based instrument recognition using ERB scale with training set of at most 3 envelopes for each sample. Each entry is a note classification

Using more spectral envelopes to describe the character of a note according to this experiment should lead to an improvement in the instrument recognition task, most importantly for the flute and clarinet. For the other instruments there is no improvement but also not a big decrease in instrument recognition performance. It seems that while the contribution of adding more spectral envelopes per note is limited for fundamental frequency estimation, it does seem to better capture the character of the instruments.

7.8 Introducing our own implementation of NMF

We now evaluate the fundamental frequency estimation performance of *WitLim* by comparing it to *TrainedLim* using two different datasets: *TrainedVERB* and *TrainedGLin*.

Max voices	1			2			3			4			5		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
TrainedLim + TrainedVERB	**98**	18	30	**97**	28	44	**89**	32	47	**89**	35	50	**82**	36	50
WitLim + TrainedVERB	**99**	18	30	**98**	28	44	**90**	32	47	**90**	35	50	**82**	37	51
TrainedLim + TrainedGLin	**96**	18	30	**97**	28	43	**84**	30	44	**85**	33	47	**80**	36	49
WitLim + TrainedGLin	**96**	17	29	**97**	28	44	**86**	31	45	**85**	33	48	**81**	36	50

Table 7.17: Comparing the developed *WitLim* framework to the existing *TrainedLim* implementation. Results are listed as: recall/precision/ F-measure in percentages

Table 7.17 shows that our own implementation of non-negative matrix factorization has nearly identical results to *TrainedLim*. We are unsure of the exact cause of this slight difference in performance. However, as the performance of *WitLim* matches that of *TrainedLim* we can now use it for further research and to develop a complete transcription system in the following sections. Because the time-frequency transform used for *TrainedGLin* is based on a library that can be freely distributed, we can use this dataset in combination with *WitLim* to create our own framework.

7.9 Using note events instead of single timesteps for fundamental frequency estimation

We now focus not only on recall but also on precision and the F-measure, as we attempt to increase the precision while maintaining a high recall.

Max voices	1			2			3			4			5		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
TrainedGLin	96	17	29	97	28	44	86	31	45	85	33	48	81	36	50
TrainedGLin + notes	94	25	39	96	34	50	84	34	49	85	37	51	79	40	53

Table 7.18: Working with note events instead of active frequencies. Results are listed as: recall/precision/ F-measure

We notice a raise in precision at little cost in recall. Taking the step of defining notes now allows us to examine these notes as a whole, instead of defining

7.10 Refining note events using post-processing

The following subsections describe the results of removing incorrect notes in order to raise precision. In the results, *TrainedGLin + notes* is the result of the previous section, where note events have been extracted from the frequency estimation results. Tests are conducted in the same way as in the previous frequency experiments, on the *MIREX* set.

7.10.1 Duration threshold

The first step in reducing incorrect frequencies is to remove notes that have a very short duration. These notes will generally not be audible as being individual notes. For this experiment, we examine the musical piece with five voices and apply a number of thresholds for the expected duration of notes:

Max voices	1			2			3			4			5		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
No threshold (TrainedGLin + notes)	94	25	39	96	34	50	84	34	49	85	37	51	79	40	53
50 ms threshold	94	25	40	95	37	53	84	37	50	84	38	53	78	41	54
100 ms threshold	93	27	41	95	38	55	83	38	52	83	41	55	77	44	56
200 ms threshold	87	29	44	84	43	57	70	44	54	72	46	56	63	49	55

Table 7.19: After removing notes with short duration. Results are listed as: recall/precision/ F-measure

It seems that we can remove notes that last up to 100 ms without sacrificing a lot of recall. Upon moving to 200 ms, we lose a lot of recall and our F-measure decreases compared to the 100 ms threshold.

7.10.2 Amplitude threshold

The thresholds that are applied in this experiment are absolute values so they depend on the choice of time-frequency transform and the range of amplitude values that are as a result present in the spectrogram. The amplitudes are averages per timestep so that the duration of a note does not affect the threshold.

Max voices	1			2			3			4			5		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
No threshold (TrainedGLin + notes)	94	25	39	96	34	50	84	34	49	85	37	51	79	40	53
Amplitude 5	93	42	58	95	41	57	84	37	52	84	40	54	78	42	55
Amplitude 10	92	54	68	94	48	64	83	42	56	84	44	58	77	46	58
Amplitude 15	91	59	71	94	51	66	83	46	59	83	48	60	76	49	60
Amplitude 20	90	62	73	92	52	67	81	50	62	82	51	67	73	52	61

Table 7.20: Removing low average amplitude notes. Results are listed as: recall/precision/ F-measure

We see that at a threshold of 15, we get the best trade-off between decrease in recall and increase in precision.

7.10.3 Relative amplitude

We now consider relative amplitude, where we take the instantaneous amplitude (at a certain timestep) at a certain frequency and divide it by the frequency that at that same timestep has the most amplitude. Note that the values in the following table are in percentages relative to the maximum amplitude at that timestep:

Max voices	1			2			3			4			5		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
No threshold (TrainedGLin + notes)	94	25	39	96	34	50	84	34	49	85	37	51	79	40	53
Threshold 3% of maximum amplitude	94	32	48	95	41	58	84	36	51	84	39	53	78	42	55
Threshold 4% of maximum amplitude	94	35	51	95	44	60	83	38	52	83	41	55	77	43	55

Table 7.21: Removing low relative amplitude frequencies. Values are in percentages relative to the highest amplitude active at each timestep. Results are listed as: recall/precision/ F-measure

In moving from 30% to 40%, we sacrifice one percent of recall for one percent of precision. We value recall more as we can hopefully improve precision using the other thresholds, therefore we consider 30% to be the optimal threshold.

7.11 Using instrument recognition to enhance fundamental frequency estimation performance

The following results were obtained using *WitFinal*, where post-processing has been applied to improve the precision of *WitLim*. Furthermore, in *WitFinal* there is no longer a fixed limit to the number of returned frequencies as the number of voices is estimated from instrument activity. To calculate instrument activity, we use a model trained on the same musical piece that was used for testing. Therefore, the performance of *WitFinal* will degrade on other musical pieces as it is overfitted to this particular piece. *VincentOrig* is provided as reference.

Max voices	1			2			3			4			5		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
VincentOrig	94	40	56	93	58	71	81	46	58	76	58	66	64	63	64
WitLim	96	17	29	97	28	44	86	31	45	85	33	48	81	36	50
WitFinal	86	89	85	87	86	86	72	74	73	70	67	68	63	68	65

Table 7.22: Frequency estimation results on musical recordings with 1-5 voices and polyphony estimation based on perfect instrument recognition. Results are listed as: recall/precision/ F-measure in percentages

In the case of 4 and 5 voices, the estimation of the number of voices was incorrect. This is probably due to *horn* notes being misclassified as *clarinet*. Therefore, with 4 voices the horn was not detected and the number of allowed notes for clarinet was instead set to two. In the musical piece containing at most 5 voices, all instruments were detected however the clarinet was still allowed two simultaneous notes which results in a loss in precision. It is possible to increase recall at the cost of precision by changing the thresholds that we have discussed in the previous section. For example, by removing the relative amplitude threshold, in many cases the trade-off was equal: 1% of precision for 1% of recall.

This section does not provide a fair comparison as our method requires perfect instrument recognition and is expected to overfit to the MIREX development set. However, we do believe that it shows the influence of estimating the number of voices on the overall performance.

Chapter 8

Conclusions and future work

8.1 Evaluation of non-negative matrix factorization

We have investigated the use of non-negative matrix factorization for the estimation of fundamental frequencies from a musical piece containing multiple voices and for instrument recognition.

8.1.1 Fundamental frequency estimation

Two forms of NMF were considered: the harmonic constrained version proposed by Vincent et al. [38] and a version based on isolated note recordings as training data. We focused on maximizing recall, the ability to retrieve the correct frequencies, and found that both variations of NMF perform well at this task. The main challenge is then posed by removing only the incorrectly retrieved frequencies.

Performance between the harmonic NMF and trained NMF is similar, while harmonic NMF uses less spectral envelopes. Because it is infeasible to train NMF on every possible instrument, we measured the performance after training on a set that did not contain the same instruments present in the musical piece. While performance did decrease, we believe that by introducing enough variety in instrument classes it is possible to capture the characters of all instruments by combining several of the trained instruments.

There was very little difference in performance when using more than one spectral envelope for each training sample. In fact, performance dropped slightly while the increase in the size of the dictionary containing spectral envelopes reduced the speed of the method.

8.1.2 NMF-based instrument recognition

In 6.7, we have used NMF to retrieve the instruments for notes that were played in a musical piece similar to how pitch values are derived. The amount of correct instrument classifications is impressive considering NMF is free to pick envelopes from each instrument and to combine these envelopes. However, besides introducing some form of sparseness on the allowed number of instruments to be used, it might prove to be difficult to find other methods to improve upon these results. Instead of using the instrument that has the most activity throughout the note duration, one might consider other measures of instrument activity or focus solely on a certain phase of the lifespan of a note.

Using NMF to perform instrument classification requires samples of each instrument to be present in the training set from which spectral envelopes are generated. This drastically increases the size of dictionary S, which serves only the instrument recognition as we have seen that frequency estimation performance does not suffer much when the instruments used in the musical piece are not actually used in training. This increase in S to cover a big set of instruments will lead to performance issues.

In short, we were impressed by the automatic tendency of NMF to use the correct instruments for reconstruction even if the recordings are different, however we fear that growth potential is limited.

8.1.3 NMF as separation technique

We have attempted to use NMF as a pre-processing step in section 6.8, where single notes were extracted from a musical piece using the reconstruction by NMF. The spectrograms of these notes can then be used as if they were isolated note recordings, meaning that a large number of existing instrument recognition techniques can be applied.

Although our results were inconclusive, we believe this was due to our choice of timbre features. More experiments with the combination of NMF and instrument recognition techniques should result in better instrument recognition rates.

Using NMF to extract note spectrograms should outperform techniques that are applied directly to the entire musical piece, because the reconstructed spectrograms are based on a set of known spectral envelopes that contain little noise and are derived from isolated recordings.

8.1.4 NMF summary

To summarize, we believe that NMF can successfully be applied to the task of frequency estimation and as a pre-processing step for instrument recognition. The matrix factorization process is able to reduce the effect of the three models (see: section 3.1.1) that influence musical pieces that contain multiple voices.

In future work we would like to focus more on measuring and comparing the reconstruction quality, to measure exactly the contribution of NMF and the loss in quality that occurs when we extract notes from a musical piece using NMF compared to isolated recordings.

8.2 Time-frequency scales

Our tests indicate little difference in frequency estimation performance between the ERB and linear time-frequency scales. However, the filter bank ERB implementation by Vincent et al. outperformed the gammatonegram using matrix multiplication on a linear scale by Ellis. We were unable to determine exactly the cause of this difference in performance, though we expect that it has something to do with the smaller available frequency range when converting from a linear scale to an ERB scale, whereas filters can be spaced exactly to return the correct frequency ranges. Furthermore, there seems to be some form of frequency grouping in the method by Vincent et al. that might reduce the amount of energy present at adjacent ERB bins.

We would like to further investigate the differences in amplitude distribution among frequencies between the time-frequency scales. A larger difference in amplitude between correct and incorrect frequencies will make it easier to remove the incorrect ones.

For both NMF-based instrument recognition and instrument recognition using timbre features, using an ERB scale resulted in better recognition rates than using a linear scale. It seems that the ERB scale is useful for describing instrument character or timbre. Related work suggests that a combination of linear and ERB scales may improve instrument recognition performance as both scales describe different properties of timbre.

8.3 Instrument recognition using timbre features

Our research concerning timbre feature-based instrument recognition unfortunately was limited due to time constraints. In hindsight, we should have included a method to determine the quality and importance of each individual feature to obtain the most compatible feature set. Researching both frequency estimation and instrument recognition might have been too much, although we have learned a lot from both experiments.

We have introduced several interesting concepts regarding the use of non-negative matrix factorization for instrument recognition. Most importantly, the process of NMF seems to enable us to extract from an instrumental recording the single notes used to create this musical piece without losing the characteristics of the instrument used to generate the note. Our classifier was able to correctly tell apart the different instruments if similar extracted samples were used for training. Unfortunately, we were unable to extend this classification to a training set of isolated notes from different recordings than the polyphonic musical recording. We were also unable to obtain competitive results for isolated note recordings, which can be caused either by the limited dataset size or by our choice and implementation of the different timbre features.

We feel that an extensive analysis is needed on the effect of extracting notes from a polyphonic mixture, in order to quantify exactly the amount of quality loss that is inflicted onto the extracted notes. Some timbre features might be more sensitive to this loss of information than others, eventually resulting in a set of features that may identify the difference between instruments even if for instance some harmonic data is lost.

The process of instrument recognition may greatly benefit from deriving global instrument statistics from a musical piece. We have seen that a success rate of around 70-80% should definitely be attainable. However, this is tested on a single note basis. By first estimating the instrument of each note in a musical piece, then analyzing their contribution to the recording it would be possible to do a second correction that changes some of the misclassified notes to the correct instruments based on the limited possibilities imposed by the instrument combination expected to be present in the musical piece.

In order to generalize the instrument recognition method to a larger set of instruments so that it will work on any musical piece as input, we would like to explore the ability to learn instrument classes from the input data itself. Imagine that we have extracted single notes from a musical piece using non-negative matrix factorization, if we could then derive features from each note and group features that are similar, we might be able to define classes from these features without knowing exactly which instruments they are. Simply knowing that there are a certain number of different instruments and which notes belong to each instrument is enough to improve transcription and to perform source separation.

This is similar to how the harmonic constrained NMF has managed to obtain performance comparable to trained NMF without depending on a training set. If the set of timbre features extracted from notes is complete enough so that notes from the same instrument are similar and notes from other instruments are different, it will be possible to determine the number of different instruments present in a musical piece and to assign each note to such an instrument automatically.

8.4 Note events

Using onset detection and amplitude behaviour, together with the factorization results of NMF, we were able to extract note events from the list of returned frequencies per timestep. We believe that by looking at notes instead of frequencies for each timestep, it is possible to include a temporal aspect in our post-processing steps. This should result in more confidence upon removing notes that we believe to be incorrect, as there is a motivation behind removing a certain note.

The influence of reliable onset detection, or any other form of prior information for that matter, should therefore not be underestimated when attempting to perform frequency estimation. NMF can help by dividing the spectrogram of the musical piece into smaller segments, allowing us to perform onset detection for each pitch value.

8.5 Combining instrument information and frequency estimation

We have to remember that running NMF on a musical piece will not result in perfect note extraction. The correct notes will be joined by a number of incorrect notes that are noise or that belong to harmonic components or slight variations in frequency of the correct notes. We hoped that these incorrect notes are prone to have the same instrument properties as the note to which they belong, therefore providing more information that can help to distinguish between correctly and incorrectly returned notes. For example, if two notes with similar onset time are exactly one octave apart, it is possible for one of them to be a harmonic of the other but it is also possible for them to be two separate notes. If both notes appear to originate from the same instrument then it is much more likely that there is indeed only one note really being played. If the notes are determined to be played by different instruments, we can use this information to avoid writing off a correct note as being a harmonic overtone. Unfortunately, this was not the case as the harmonic notes in many cases would be classified as a different instrument.

The main contribution of the knowledge of instrument data is the ability to derive global instrument statistics that apply to the entire musical piece. This allows identification of the instruments that are expected to be present in the musical piece, information which can be used to estimate the number of voices present in the recording. We have found that knowledge of the number of voices and the instruments present in a recording contributes greatly to determining the correct frequencies.

The way the total energy at a certain point in time is divided among either frequencies or notes may provide information on the number of voices. For example, it is possible for one frequency to have around 40% of the total amplitude if the number of voices is two. If the number of voices is one, the relative amplitude of the fundamental frequency of this voice is expected to be higher. This effect becomes more reliable if harmonic frequencies are detected and combined with the fundamental frequency they belong to.

We fear that the method used in this case to determine the number of voices of each instrument is too unreliable. If an instrument were to play two notes at only few points in the musical piece, it may be determined to have a only one voice. Instead of strictly enforcing the number of voices as we have done, using this polyphony estimation merely as an indication for deletion of incorrect frequencies rather than immediately removing all frequencies that exceed the number of voices might improve performance.

8.6 Datasets

The methods proposed in this paper might have been overfitted to the MIREX development set. Therefore, generation of a bigger dataset for testing the transcription performance is recommended. Although other research groups have worked with similar dataset sizes, we believe that the number of notes for each instrument for the task of instrument recognition should be larger as well. Most of our experiments still show a lot of uncertainty which may be caused by a dataset that fails to fully describe each instrument.

8.7 Performance

When using a set of spectral envelopes derived from a set of training note recordings, the computational requirements and memory footprint depend heavily on the size of the training set. After all, more recordings means more envelopes and thus more data to update iteratively in the NMF process. The harmonic constrained NMF is expected to require more iterations to converge as both A and E need to be updated, however the size of the data in this case is much smaller as there are only 88 entries in S, and a few entries for each S in E.

A big decrease in performance is due to the use of both linear and ERB frequency scales in order to improve instrument recognition. We have found that increasing the number of frequency bins in each time-frequency representation may also drastically increase the memory footprint and the amount of processing time required.

The advantage of using a classifier such as MultiBoost is that it derives a model from the training data, which may take a lot of time to generate but will only need to happen once as long as the features used remain the same. Matching a set of new notes is then done almost instantly, even when the number of notes runs in the thousands. This process is fast because we extract a set of features from the note rather than using the entire time-frequency representation of each note as input.

8.8 L0 sparse NMF

While using more than one envelope for each note in the training set did not improve our frequency estimation results, we did see an improvement in NMF-based instrument recognition. Unfortunately, we were unable to measure the effect of using these added envelopes on the quality of the reconstructed spectrogram.

We have also briefly examined L0 sparse NMF as a method of estimating the number of voices present in a musical piece. Remember that L0 sparseness allows only a certain number of spectral envelopes to be active at one time. We introduced another layer in the NMF process to group all envelopes that belong to a certain pitch value, resulting in 88 entries in S that were each built up from a number of envelopes. L0 sparse NMF was then applied to these 88 entries, to allow only a certain number of pitch values to be active at the same time. By increasing and decreasing this number of allowed pitch values, we hoped to find the correct frequencies along with an estimation of the number of voices to retain a high precision.

Unfortunately, this method of estimating the number of voices appeared to be too unreliable. We hoped that by gradually increasing the number of allowed pitch values, we would keep getting more correct values until a harmonic frequency or a frequency with low amplitude showed up indicating that we had retrieved all correct pitch values. However, in some cases upon increasing the number of allowed pitch values by one the resulting pitch values would not be the same pitch values returned previously with one additional new frequency. It is also not always the case that the correct frequencies have the highest amplitude by themselves, as they could simply have relatively little amplitude compared to even the harmonics of another frequency or their amplitude can be spread out over a number of harmonic components.

Appendix B: List of instrument samples

From both datasets, the mezzo forte variation was used. We list the instruments and the pitch range that was used.

Real World Computing (RWC) dataset

Alt saxophone: 29-61
Bassoon: 14-52
Cello: 16-61
Clarinet: 30-69
Contrabass: 8-43
Flute: 40-76
Horn: 21-57
Oboe: 38-71
Piano: 1-88
Tuba: 9-38
Violin: 35-80

University of Iowa dataset

Alt saxophone: 29-60
Bassoon: 14-54
Cello: 16-46
Clarinet: 35-76
Contrabass: 8-30
Flute: 39-77
Horn: 14-27 and 40-57
Oboe: 38-72
Piano: 3-88
Tuba: 4-40
Violin: 35-58

Bibliography

[1] AbsoluteAstronomy.com. Absolute astronomy: Timbre. `http://www.absoluteastronomy.com/topics/Timbre`.

[2] M. Bay, A.F. Ehmann, and J.S. Downie. Evaluation of multiple-F0 estimation and tracking systems. *10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, 2009.

[3] C.M. Bishop. *Pattern recognition and machine learning*, pages 657–663. Springer, sixth edition, 2006.

[4] A.T. Cemgil, B. Kappen, and D. Barber. Generative model based polyphonic music transcription. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, October 2003.

[5] A. Cichocki, R. Zdunek, and S.-I. Amari. Csiszars divergences for non-negative matrix factorization: Family of new algorithms. *Independent Component Analysis and Blind Signal Separation, lecture notes in computer science*, 3889:32–39, 2006.

[6] S.B. Davis and P. Mermelstein. Comparison of parametric representation for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech and Signal Processing*, August 1980.

[7] A. de Cheveigné and H. Kawahara. YIN, a fundamental frequency estimator for speech and music. *Journal of the Acoustical Society of America*, 111(4):1917–1930, April 2002.

[8] J. de Wit. Emotion classification from facial images (experimentation project report). 2011.

[9] D.P.W. Ellis. ğammatone-like spectrograms; web resource. `http://www.ee.columbia.edu/~dpwe/resources/matlab/gammatonegram/`, 2009.

[10] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Music genre database and musical instrument sound database. *Proceedings of the 4th International Conference on Music Information Retrieval*, pages 229–230, October 2003.

[11] A. Graps. An introduction to wavelets.

[12] T. Heittola, A. Klapuri, and T. Virtanen. Musical instrument recognition in polyphonic audio using source-filter model for sound separation. *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, 2009.

[13] P.O. Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of Machine Learning Research*, 5:1457–1469, 2004.

[14] X. Hu and J. Liu. Evaluation of music information retrieval: towards a user-centered approach. *HCIR 2010*, August 2010.

[15] J.O. Smith III and J.S. Abel. The bark and erb bilinear transform. *IEEE Transactions on Speech and Audio Processing*, December 1999.

[16] D.N. Jiang, L. Lu, H.J. Zhang, J.H. Tao, and L.H. Cai. Music type classification by spectral contrast feature. *Proceedings of Intelligent Computation in Manufacturing Engineering*, 113-6, 2002.

[17] H. Kameoka. Statistical approach to multipitch analysis (PhD thesis). 2007.

[18] T. Kitahara, M. Goto, K. Komatani, T. Ogata, and H.G. Okuno. Instrument identification in polyphonic music: feature weighting to minimize influence of sound overlaps. *Journal on Advances in Signal Processing*, 2007.

[19] A. Klapuri. A perceptually motivated multiple-F0 estimation method. *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 291–294, October 2005.

[20] D.D. Lee and H. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature 401*, pages 788–791, 1999.

[21] P. Leveau, D. Sodoyer, and L. Daudet. Automatic instrument recognition in a polyphonic mixture using sparse representations. *International Society for Music Information Retrieval Conference (ISMIR)*, 2007.

[22] M.S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: state of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications*, February 2006.

[23] A.A. Livshin and X. Rodet. Musical instrument identification in continuous recordings. *Proceedings of the 7th International Conference on Digital Audio Effects*, October 2004.

[24] L.G. Martins, J.J. Burred, G. Tzanetakis, and M. Lagrange. Polyphonic instrument recognition using spectral clustering. *International Society for Music Information Retrieval Conference (ISMIR)*, pages 213–218, 2007.

[25] B. Milewski. The fourier transform. `http://www.relisoft.com/science/physics/sound.html`.

[26] A.P. Ibá nez. Computationally efficient methods for polyphonic music transcription (PhD thesis). 2010.

[27] P.D. O'Grady and B.A. Pearlmutter. Convolutive non-negative matrix factorization with sparseness constraint. *International Workshop on Machine Learning for Signal Processing*, pages 427–432, 2006.

[28] T.H. Park. Towards automatic musical instrument timbre recognition (PhD thesis). November 2004.

[29] G. Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. 2004.

[30] G. Peeters and X. Rodet. Automatically selecting signal descriptors for sound classification. 2002.

[31] R. Peharz, M. Stark, and F. Pernkopf. Sparse nonnegative matrix factorization using $\ell 0$-constraints. *IEEE Workshop on Machine Learning for Signal Processing*, pages 83–88, 2010.

[32] H. Pollard and E. Jansson. A tristimulus method for the specification of timbre. *Acustica*, 51:162–171, 1982.

[33] V.K. Potluru, S.M. Plis, and V.D. Calhoun. Sparse shift-invariant nmf. *IEEE Southwest Symposium on Image analysis and interpretation*, pages 69–72, 2008.

[34] M.D. Skowronski and J.G. Harris. Exploiting independent filter bandwidth of human factor cepstral coefficients in automatic speech recognition. *Acoustical Society of America*, pages 1774–1780, 2004.

[35] S.K. Tjoa and K.J.R. Liu. Musical instrument recognition using biologically inspired filtering of temporal dictionary atoms. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 435–440, Utrecht, Netherlands, August 2010.

[36] G. Tzanetakis and P. Cook. Musical genre identification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, July 2002.

[37] M. van Dyke Kotvis. An adaptive time-frequency distribution with applications for audio signal separation (research project). 1997.

[38] E. Vincent, N. Bertin, and R. Badeau. Adaptive harmonic spectral decomposition for multiple pitch estimation. *IEEE Transactions on Audio, Speech and Language Processing*, 18:528–537, 2010.

[39] T. Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE Transactions on Audio, Speech and Language Processing*, 15(3):1066–1074, March 2007.

[40] T. Virtanen and A. Klapuri. Analysis of polyphonic audio using source-filter model and non-negative matrix factorization. *Advances in Models for Acoustic Processing, Neural Information Processing Systems Workshop*, 2006.

[41] S. Wood. Non-negative matrix decomposition approaches to frequency domain analysis of music audio signals (MSc thesis). December 2010.

[42] C. Yeh. Multiple fundamental frequency estimation of polyphonic recordings (PhD thesis). June 2008.

[43] C. Yeh and A. Roebel. Multiple-F0 estimation for MIREX 2010. *Music Information Retrieval Evaluation eXchange*, 2010.

[44] X. Zhang and Z.W. Ras. Differentiated harmonic feature analysis on music information retrieval for instrument recognition. *IEEE International Conference on Granular Computing*, 2006.