Vibraphone Transcription from Noisy Audio Using Factorization Methods

by

Sonmaz Zehtabi
B.Sc., University of Tehran, 2010

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

in the Department of Computer Science

Vibraphone Transcription from Noisy Audio Using Factorization Methods

by

Sonmaz Zehtabi
B.Sc., University of Tehran, 2010

Supervisory Committee

Dr. George Tzanetakis, Supervisor
(Department of Computer Science)

Dr. Ulrike Stege, Departmental Member
(Department of Computer Science)

Supervisory Committee

Dr. George Tzanetakis, Supervisor
(Department of Computer Science)

Dr. Ulrike Stege, Departmental Member
(Department of Computer Science)

ABSTRACT

This thesis presents a comparison between two factorization techniques – Probabilistic Latent Component Analysis (PLCA) and Non-Negative Least Squares (NNLSQ) – for the problem of detecting note events played by a vibraphone, using a microphone for sound acquisition in the context of live performance. Ambient noise is reduced by using specific dictionary codewords to model the noise.

The results of the factorization are analyzed by two causal onset detection algorithms: a rule-based algorithm and a trained machine learning based classifier. These onset detection algorithms yield decisions on when note events happen. Comparative results are presented, considering a database of vibraphone recordings with different levels of noise, showing the conditions under which the event detection is reliable.

Contents

List of Tables

List of Figures

ACKNOWLEDGMENTS

First I would like to offer my gratitude to my supervisor, Dr George Tzanetakis, who gave me the freedom to choose the research problem that I wanted to work on and supported me thoughout my thesis with his patience and knowledge.

I would also like to thank my colleagues in the MISTIC lab for their support and input, specially Tiago Fernando Tavares without whom this thesis would not have been completed.

Thanks also go to the staff of the department of computer science of University of Victoria for fascilitating the graduation process for me and taking care of the paper work; this allowed me to focus on writing the thesis.

I thank my family for their unconditional love and encouragement during all my studies and for always trusting me and supporting me while I was writing my thesis.

Chapter 1

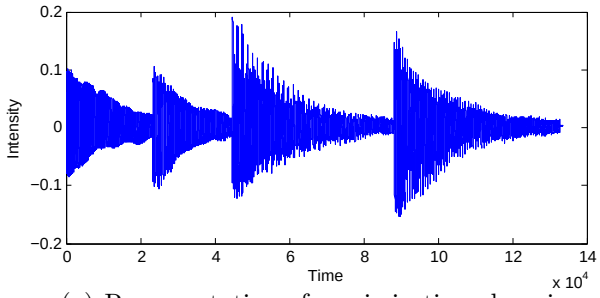
Introduction

1.1 The task of music transcription

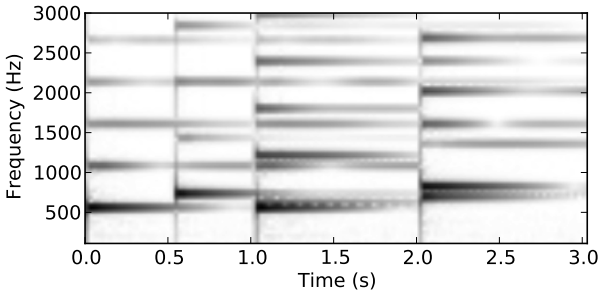
Music transcription can be defined as listening to a piece of music and extracting the symbolic representation, which includes information such as pitch, target instrument, timing, and duration of each individual sound source [31].

The accurate symbolic representation of a song has many useful applications; it is readable by machines and humans, and provides different information than a recording. Creating MIDI output for music compositions that are only available in the form of audio recordings provides tools for musicians to analyze, mix and edit these compositions. It also allows reproducing and modifying the original performance of a given signal. A MIDI representation is a compact form of the audio signal which still retains the important characteristics the signal to a great extent, therefore it is the perfect input for the above-mentioned tasks.

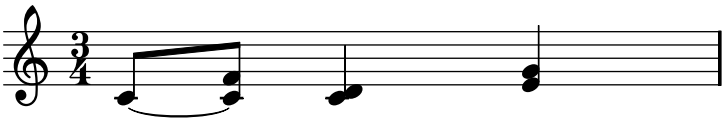
We will use a 3-second long piece of polyphonic music with 6 notes as an example throughout the thesis, to make the concepts clearer. Figure 1.1 depicts the representation of the audio recording of this piece at different stages of the transcription. The input is a waveform, which shows the audio in the time-domain, the time domain signal is then converted to a frequency domain signal, and the output of the system, the transcription result, is represented as a piano roll. The output of the transcription system presented in this thesis is a subset of all the information required to create a music score. These information include notes and their begin and end times; this will be equal to creating a piano roll. In order to create a complete MIDI file, additional steps need to be taken in order to extract information such as speed, etc.



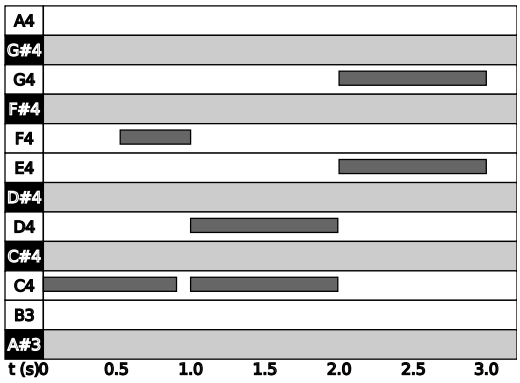
(a) Representation of music in time-domain



(b) Representation of music in frequency domain



(c) Representaion of music in the form of music score



(d) Representaion of music in the form of piano roll

Figure 1.1: Audio representation at different stages of transcription

Music transcription from audio signals is not a trivial task; it usually requires separation of complex sound sources. People without musical education usually are not able to transcribe music and extract high-level music information. Creating a symbolic representation from an audio file often requires expert musicians; transcribing the music that has only one note playing at a certain time is the simplest form of transcription, and people with knowledge of music are usually able to transcribe it. But, the more complex (in terms of the number of the sounds that are playing simultaneously) the music gets, the more difficult it is to transcribe it.

The case of monophonic music is practically solved; several algorithms are designed that transcribe monophonic music accurately and reliably. A summary of these methods can be found in Klapuri and Davy's book [31]. Attempts towards polyphonic music transcription though, have not been quite as successful. No method developed so far has yet come close to the performance of humans in transcribing music.

Moreover, there are strong limitations imposed on the problem in the past. Music transcription research in the past was mostly focused on music that was recorded in an environment with no noise or simply synthesized MIDI. But usually most pieces of music are home recorded or recorded in a concert during a live performance. In the presence of interfering noise, even transcribing monophonic music becomes difficult. The dataset used in this thesis was recorded in a noisy environment, and crowd noise and background music were added in order to imitate the recording of a live performance.

In this thesis, automatic music transcription techniques are employed in order to obtain event data from a specific instrument in the context of a live performance, using only a common microphone as additional hardware. The system presented in this thesis is capable of identifying musical notes played in a live performance, and creating a meaningful representation of the data in the form of a piano roll. This scenario is particularly interesting in contemporary electro-acoustic music, which is a genre that usually involves improvisations and live interaction with digital effect parameters. Although it is possible to build a prepared instrument that is enhanced with digital sensors for each note to be played, that process is generally time consuming, can be costly and can only be applied to a single instrument.

Another neglected aspect of music transcription is causality and real-time performance, which is also necessary for online transcription of live performance. A system is called causal, if the events of the system are only dependent on the current and previous events. If a system transcribes the events of time instant t by analyzing

events happening after time instant t, that system is non-causal. Operations such as finding the maximum value of the signal, or normalizing the input signal of the input signal (the exception is when the system normalizes the signal with factors that are learnt from previously seen data) are examples that will make an algorithm non-causal. Techniques for noise elimination will be discussed as well as a causal algorithm for detecting note onsets.

There are several methodologies investigated for polyphonic music transcription (Section 1.3 provides a summary of these methods). The methods used in this dissertation are from the family of matrix factorization algorithms. In such methods, it is assumed that the digital signal can be expressed as a linear weighted sum of waveforms of individual notes. In the past, these methods have been applied to the polyphonic music transcription problem and encouraging results have been attained. In this thesis, we explore different aspects of these methods and we compare the transcription results of decomposition methods with the transcription results from the state of the art multi-f0 estimation method introduced by Klapuri [30] which won the MIREX music transcription competition in 2006 (details of how the algorithm works are explained in related work, in Chapter 1.3). We show that in the context of noisy audio, the decomposition methods outperform the multi-f0 estimation method. We also provide a comparison between supervised and unsupervised versions of the decomposition algorithms.

In order to accurately detect onsets in polyphonic music, two onset detection algorithms that exploit information from the matrix factorization step are investigated. The first method is a rule-based decision algorithm and the second method is based on support vector machines. These algorithms are designed to have causal properties for the scenario of transcribing recordings from live performance.

1.2 Music terminology

This section briefly defines the concepts that are used throughout the thesis:

Vibraphone

The vibraphone is a percussion instrument with aluminum bars. It is operated by hitting one, two, three or four mallets against the bars. The vibraphone has a sustain pedal; with the pedal down, each note will sound for several seconds but when the

pedal is up, the bars are damped and the notes sound for a shorter period. Figure 1.2 shows a picture of the vibraphone instrument.

Figure 1.2: Vibraphone instrument used in the experiments of this thesis.

Monophonic vs polyphonic

Music signals that have one note sounding at a time are referred to as monophonic, and signals that have several sounds being played simultaneously, are called polyphonic.

Fundamental frequency and harmonics

When a musical instrument is played, the sound we hear consists of several frequencies. The lowest frequency is called fundamental frequency, f_0, and integer multiplies of f_0 are called harmonics. The fundamental frequency is also called first harmonic, $2f_0$ is called second harmonic, and so on.

Pitch

is a perceptual attribute related to the fundamental frequency of a sound; it is ordered from low to high.

Timbre

is the "color" of a sound. It is what differentiates two sounds with the same pitch. For example, if we play the same note on both violin and piano, what makes them sound different is their timbre.

Note

is referred to both a musical symbol and the sound the instrument makes when the symbol is played. In western music, notes are named C, C#,D,D#, etc. 12 notes make up an octave. The notes for the third octave, for example, are called C3, C#3, D3, etc.

Musical Instrument Digital Interface (MIDI)

MIDI is a form of music representation, and the most common symbolic digital music interchange format. It includes information such as MIDI note number, an integer that indicate the pitch of a note, start and end time of a note, and an instrument number.

Onsets

Onsets are defined as the time-locations of all sonic events in a piece of audio and more specifically beginning of notes in the context of music transcription.

1.3 Related Work

There have been numerous attempts to solve music transcription problems and different methodologies have been proposed. These methods can be divided into several categories. This section provides a summary of research done for each category.

Matrix factorization methods

In this thesis, we use two matrix factorization methods: NNLSQ and PLCA. These methods have been used in previous work and in a variety of contexts:

Guo and Zhu [26] use PLCA for transcription of music from several instruments and they do this by first finding the onsets of the original signal and then trying to find the pitch by comparing the intensities of the weights in the frames following the onset frame. For each onset, the weight coefficients of different notes are compared against their sums of magnitude during the 0.15 seconds after the onset and the note with the highest weight coefficient among all the weights is taken as the active note for that onset.

Mysore and Smaragdis [40] proposed an extension to probabilistic latent component analysis for multiple instruments in polyphonic music. The method they propose is unsupervised and makes use of a multi-layered positive deconvolution that is performed to obtain a relative pitch track and timbral signature for each instrument. Since in a constant-Q transform, notes at different pitches appear as shifted versions of the same spectral pattern, Q transform of the instrument can be seen as a convolution of the spectral pattern of that instrument.

Bertin, Badeau, and Richard [7] investigated the behavior of two blind signal decomposition algorithms, non negative matrix factorization (NMF) and non negative K-SVD (NKSVD), in a polyphonic music transcription task and showed that their performances are similar, but in favor of NMF, which is more robust to initialization, choice of the order and is computationally less costly.

Bertin, Badeau, and Vincent [8, 9, 10] propose a Bayesian NMF with harmonicity and temporal continuity constraints which is enforced through an inverse-Gamma Markov chain prior. They also propose a reduction of computational time by initializing the system with an original variant of multiplicative harmonic NMF.

Gillet and Richard [22] focus on drum signals. A complete drum transcription system is described, which combines information from the original music signal and a drum track enhanced version obtained by source separation. They integrate a large set of features, harmonic/noise decomposition, and time/frequency masking, and improve an existing Wiener filtering-based separation method.

Grindley and Ellis[24] extend the non-negative matrix factorization (NMF) algorithm to incorporate constraints on the basis vectors of the solution. In the context of music transcription, this allows us to encode prior knowledge about the space of possible instrument models as a parametric subspace.

Phon-Amnuaisuk [47] also uses non-negative matrix factorization (NMF); the tone model is learned from the training data consisting of the pitches of the desired instrument.

Spectral analysis

There is another family of methods that analyzes the signal in the frequency domain to obtain pitch:

Klapuri [30] proposes a *fundamental frequency (F0)* estimator for polyphonic music signals. The estimator first maps the Fourier spectrum into a F0 salience (strength)

spectrum, by calculating the strength, of a F0 candidate as a weighted sum of the amplitudes of its harmonic partials. From this spectrum the polyphonic pitches are extracted.

Argenti, Neri, and Pantaleo [1] carry out multiple-F0 estimation by means a constant-Q and a bi-dimensional frequency representation, capable of detecting non-linear harmonic interactions, which are typically present in musical audio signals. They estimate onsets by detecting rapid spectral energy variations over time.

Barbancho *et al.* [2] present a system for automatic identification of polyphonic piano recordings. The system divides the piano piece into attack slots which are segmented based on onsets, and then performs a frequency analysis using filter banks and harmonic elimination on each attack slot to detect the pitch.

Bello, Daudet, and Sandler [4] propose a method that groups spectral information in the frequency-domain and use a rule-based framework to deal with the problems of polyphony and harmonicity. The method considers the signal as the linear weighted sum of isolated piano notes. It then estimates the pitch by acquiring an adequate estimation prior training of the isolated notes and analyzing of the signal in frequency and time domain. This hybrid method taks into account the information contained in phase relationships, that are lost when only the magnitude spectra of sounds are analyzed.

Goto [23] proposes a predominant-F0 estimation method that obtains the most predominant F0 supported by harmonics within an intentionally limited frequency range. This method estimates the relative dominance of every possible F0 (represented as a probability density function of the F0) by using MAP (maximum a posteriori probability) estimation and considers the F0's temporal continuity by using a multiple-agent architecture.

Hajimolhosseini, Taban, and Abutalebi [27] propose an algorithm that consists of two main stages: the first stage eliminates the harmonics of the music signal and only passes the fundamental part. The second stage estimates and tracks the fundamental frequency of music signal by means of an Extended Kalman Filter (EKF) frequency tracker.

Kobzantsev, Chazan, and Zeevi [32] apply segmentation of notes in the time domain, estimation of frequency components based on the structure of time segments, extraction of pitches of underlying notes, and tracking of notes to obtain the final music score. A combination of multi-resolution techniques, such as multi-resolution Fourier transform and maximum likelihood frequency estimator, enable them to suc-

cessfully cope with the problems of constant time-frequency resolution and frequency masking.

Lao, Tsoon Tan, and Kam [33] use a two-step strategy: tracks creation and tracks grouping. The scheme utilizes innovative comb-filtering and sharpening steps to produce the desired transcription output in the form of discrete notes with temporal, pitch and amplitude attributes.

Miyamoto *et al.* [36] integrates probabilistic approaches to multi-pitch spectral analysis, rhythm recognition and tempo estimation. In spectral analysis, acoustic energies in spectrogram are clustered into acoustic objects (i.e., music notes) with the method called harmonic-temporal-structured clustering (HTC) utilizing EM algorithm over a structured Gaussian mixture with constraints of harmonic structure and temporal smoothness. After onset and offset timings are found from separated energies of music notes through note power envelope modeling to obtain the piano-roll representation, the rhythm and tempo are simultaneously recognized and estimated in terms of maximum posterior probability given a probabilistic note duration models with HMM (hidden Markov model) and probabilistic "rhythm vocabulary". Variable tempo is also modeled by a smooth analytic curve. Rhythm recognition and tempo estimation is alternately performed to iteratively maximize the joint posterior probability.

Derrien [18] proposes a method consisting of a frame-based expansion of the signal over a multi-scale time-frequency dictionary with a set of logarithmic discrete frequencies. This method, based on the matching pursuit algorithm, provides the same frequency resolution as a constant-Q filter-bank, but with a better time resolution, especially in low frequencies, and an efficient noise rejection.

Ryynanen and Klapuri [52] present a method that uses a multiple-F0 estimator as a front-end and this is followed by acoustic and musicological models. The acoustic modeling consists of separate models for bass notes and rests. The musicological model estimates the key and determines probabilities for the transitions between notes using a conventional bigram or a variable-order Markov model. The transcription is obtained with Viterbi decoding through the note and rest models. In addition, a causal algorithm is presented which allows transcription of streaming audio.

Machine Learning and AI methods

Machine learning methods are currently used in variety of fields to solve problems by training using existing examples. The Music Information Retrieval community has also used this method in past years to solve the music transcription problem:

Pertusa and Iesta [46] approach transcription through the identification of the pattern of a given instrument in the frequency domain. This is achieved using time-delay neural networks that are fed with the band-grouped spectrogram of a polyphonic monotimbral music recording.

Bruno, Monni, and Nesi [12] use a partial tracking module along with a pre-trained neural network bank with the capability to recognize pitches both of singles notes and of chords of notes. Each neural network has one output that is activated every time a note or a chord is recognized. They use a peak-picking algorithm for onset detection.

Chien and Jeng [14] address the issue of octave detection in automatic transcription of polyphonic music. Pitch detectors for polyphonic music usually fail to detect octaves for lack of information about the timbre of each instrument that appears in the music. They use constant-Q time-frequency analysis along with wavelet transform, to train support vector machine for octave detection.

Gillet and Richard [21, 20] present transcription of drum sequences using audio-visual features. The transcription is performed by support vector machine (SVM) classifiers.

Reis *et al.* [51] present a genetic algorithm approach with harmonic structure evolution for polyphonic music transcription. Music transcription can be addressed as a search space problem where the goal is to find the sequence of notes that best models our audio signal. By taking advantage of the genetic algorithms to explore large search spaces they present a new approach to the music transcription problem.

Constantini, Todisco, and Perfetti [15, 16, 17] propose a method that focuses on note events triggered by events corresponding to the played notes the attack instant, the pitch and the final instant. Onset detection exploits a binary time-frequency representation of the audio signal. Note classification and offset detection are based on constant Q transform (CQT) and support vector machines (SVMs).

Transcription of singing

Several methods have also been proposed for the transcription of the singing or humming voice:

Jiang, Picheny, and Qin [42] present a robust voice-melody transcription system using a speech recognition framework. A cepstrum-based acoustic model is employed and a key-independent 4-gram language model is employed to capture prior probabilities of different melodic sequences.

Lee and Jang [34] describe the construction of a system called i-Ring that can generate a polyphonic ringtone based on a user's humming input. Algorithms used in the system for music transcription and chord generation uses pitch tracking and dynamic programming,

Mesaros, Virtanen, Klapuri [35] propose an approach that includes separating the vocal line from the mixture using a predominant melody transcription system then apply a melody transcription system and then resynthesis the vocals. Within each frame, first they estimate whether a significant melody line is present, and then estimate the MIDI note number of the melody line. For synthesizing, harmonics are generated at integer multiples of the estimated fundamental frequency, and amplitudes and phases are also estimated.

Time-frequency transforms

There are numerous time-frequency transforms studied in the past for the task of music transcription. Researchers believed that if they invent a perfect transform they will solve the problem of music transcription. They did succeed to some extent in solving the monophonic case, but a transform that will solve polyphonic transcription has not been invented. Therefore, the focus of research in this area shifted towards other parts of the transcription system. Some of these transforms are explained below:

Modal transform [29]: adaptively modifies the basis function in order to minimize the bandwidth of each lobe in signal X, so that X[n] (where n is the frame number) can be represented by fewer coefficients in the frequency domain.

Chroma spectrum [3, 39, 44, 45]: it consists of a frequency domain spectrum in which there are only 12 coefficients, each corresponding to a certain musical note (octaves are ignored). In this transform, all energy related to the fundamental frequencies corresponding to a certain note is concentrated to a single coefficient.

Discrete Time Fourier Transform (DFT) [48, 49, 7, 22, 24, 25, 33, 41, 47, 53, 56]: DFT changes a signal from time domain to frequency domain by breaking down the original time-based waveform into a series of sinusoidal terms, each with a unique magnitude, frequency, and phase.

Constant Q transform (CQT) [14, 15, 16, 17, 1, 5, 6, 14, 55, 40]: Introduced by Brown [11], this method is similar to DFT but instead of being linearly spaced, it has logarithmic frequency resolution matching the geometrically spaced notes of the Western music scale.

1.4 Contributions

The major contributions of this dissertation are as follows:

- Employing matrix factorization methods to solve polyphonic music transcription in the context of live vibraphone performance, which requires applying causal algorithms to noisy recordings.
- Showing how the transcription results are improved if matrix factorization methods are employed in a supervised fashion
- Showing that matrix factorization methods outperform state of the art transcription methods in the context of live performance
- Using two onset detection methods in a novel way. The onset detection algorithms in the literature are performed on the original audio; the methods in this thesis however are applied to the activation matrix.

1.5 Thesis organization

The proposed transcription system consists of 3 major components (Figure 1.5): transforms, source separation, and smoothing. Improvement on each of these steps can result in a better transcription.

Transforms from time domain to frequency domain are necessary to change the acoustic signal into an input format for source separation methods. There has been a lot of research conducted on transforms available in the literature (refer to Section 1.3) and Discrete Fourier Transform (DFT) and Constant Q Transform (CQT), are the ones that have proved to be superior to other methods and are most commonly used today. In this thesis both DFT and CQT are used for the transform part. I talk about this step in detail in Chapter 2.

In the second part of the transcription system, we decompose the acoustic spectra using two of the most common algorithms: Probabilistic Latent Component Analysis

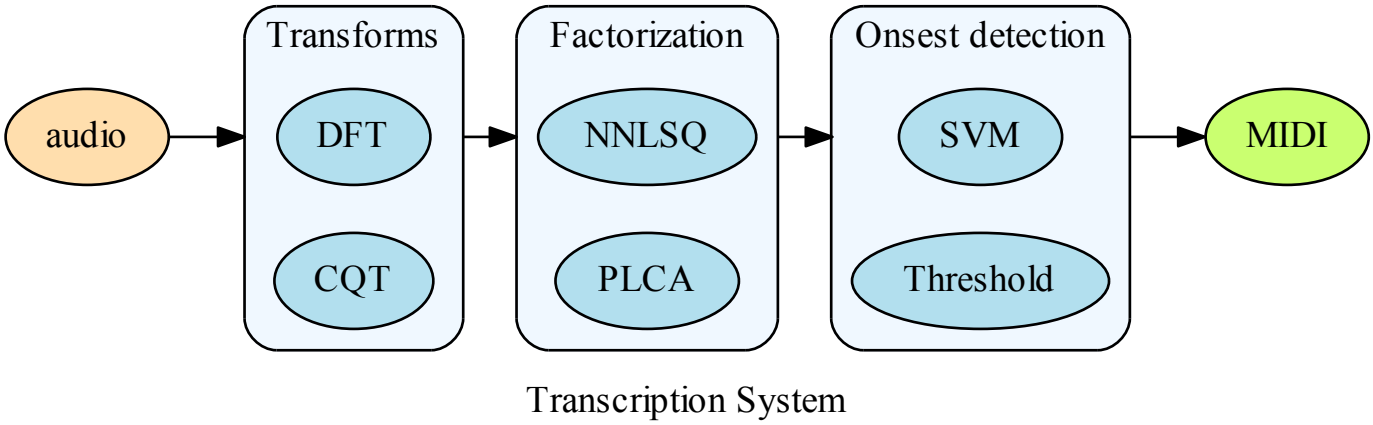


Figure 1.3: Overview of our music transcription system.

(PLCA), and Non-negative Matrix Factorization (NMF). These algorithms will model the audio, and separate its components, i.e. their notes. These methods are explained in detail in Chapter 3.

The method used in this thesis requires both separation of many sound sources and analysis of the content of these sources. In the last part of our transcription analyses the separated components from the previous step in order to find the onset of each note. Two algorithms are implemented for this task *adaptive thresholding*, which simply finds the peaks by setting a few parameters and *Support Vector Machines (SVM)* a machine learning method, which is more sophisticated than thresholding, and requires training prior to onset detection. Chapter 4 of the thesis is devoted to this part of the system.

We evaluate the system using criteria for transcription, such as *precision* and *recall*. In Chapter 5 the performed experiments, as well as the used database, and the obtained results, are discussed. Finally, in Section 6, conclusive remarks are given and the future work is discussed.

Chapter 2

Time-frequency representations

In the physical world, audio is an analog signal. By sampling this continuous signal, a discrete-time signal $x(n)$ is obtained. This discrete-time signal is represented in the shape of a waveform. Music transcription methods can be classified into two categories: time and frequency domain methods. Time domain methods can achieve good results for transcribing monophonic music. The algorithms in this category are based on evaluating the periodicity of the acoustic signal. Moorer [38] uses zero-crossing feature (i.e., the number of times the signal crosses a zero threshold in a time unit) to extract the pitch. Rabiner [50] and Monti and Sandler [37] perform pitch detection algorithm by using the auto-correlation of the signal to calculate its self-similarity over time. Peaks of the autocorrelation indicate the fundamental frequency of the signal, for periodic signals.

To solve the polyphonic transcription problem, frequency domain methods are commonly used to extract information about pitch. These methods take the frequency domain representation of the signal as input. There are numerous time-frequency transforms studied in the past to for the task of music transcription. In this thesis, two of these methods are used to get the frequency representations we need: the Discrete Fourier Transform and the Constant Q Transform. We choose DFT since there is a fast implementation for it called the Fast Fourier Transform (FFT) and the extracted information is straight forward to interpret. We use CQT since it is one of the mostly commonly used transform where source separation approaches are applied.

2.1 Discrete Fourier Transform

The Discrete Fourier Transform or DFT for short is a transform that changes a signal from the time domain to the frequency domain. Equation 2.1 shows how the time-domain signal $x(n)$ is transformed into frequency-domain signal $X(m)$. The Fourier transform accomplishes this by breaking down the original time-based waveform into a series of sinusoidal terms, each with a unique magnitude, frequency, and phase.

DFT takes a finite set of N input vales, that are sampled at a sample rate of f_s - where f_s is the sampling frequency, which must be at least twice the highest frequency component of the original signal- and returns N points associated with the individual analysis frequencies.

Each individual X(m) output value is a correlation between the original signal and a cosine and a sine wave whose frequencies are m complete cycles in the total sample interval of N samples.

$$X(m) = \sum_{n=0}^{N-1} x(n)[cos(2\pi nm/N) - jsin(2\pi nm/N)] \tag{2.1}$$

where m and n are integers from 0 to N - 1, $x(n)$ is the n^{th} sample of the input signal, X(m) is the m^{th} output of the DFT transform.

As you can see in the Equation 2.1, the values for the signal in the frequency domain are complex numbers. Phase and magnitude information is calculated from these complex numbers. Phase information is not useful in the algorithms presented in the thesis, and that is why we only use the magnitude of the DFT.

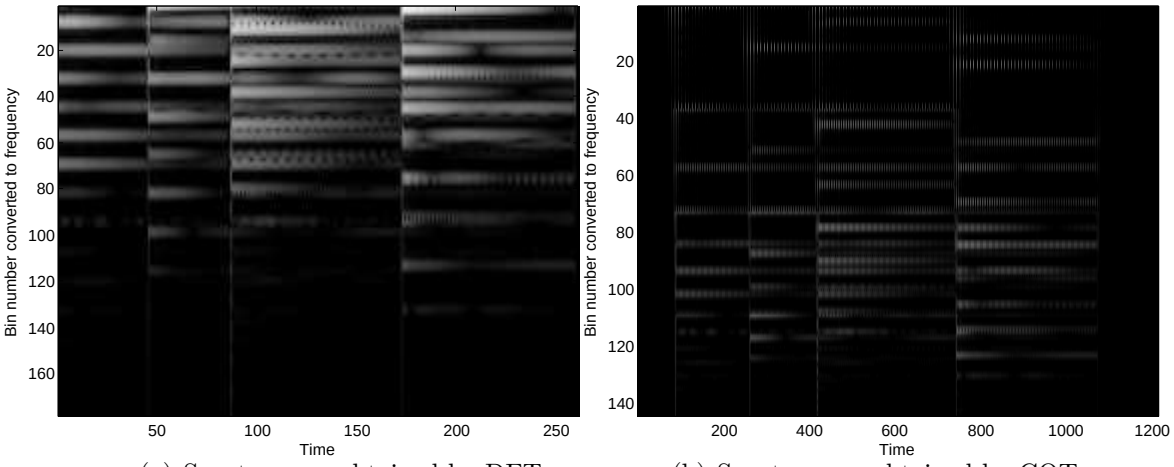
As mentioned earlier, the output of DFT is discrete, but the the frequency values in the original signal are continuous. So if the original signal has a component at some intermediate frequency that doesn't match any of the discrete frequencies analyzed by DFT, this will show up to some extent in all of the N output analysis frequencies. We can reduce this undesired effect by windowing, i.e. multiplying the input signal by a window function before the DFT is performed. The window function used in the experiments for this thesis is called a Hanning window (Equation 2.2).

$$w(n) = 0.5 - 0.5cos\left(\frac{2\pi n}{N}\right) \tag{2.2}$$

2.2 Constant Q Transfrom

The DFT is the most widely used transform, however, the Constant Q Transform (CQT) has several advantages over the DFT that makes it more useful for some applications. For example, in the DFT the frequency scale is linearly divided, whereas a logarithmic scale would be more appropriate for our task since humans perceive pitch as a logarithmic function of frequency. Moreover, in the DFT if we choose a wide window of analysis, the frequency resolution will be good, but the time resolution will be poor, and a narrower window will result in poorer frequency resolution and better time resolution; This is called resolution issue. CQT addresses this issue and fixes it to some degree.

$$X[k] = \frac{1}{N[k]} \sum_{n=0}^{N[k]-1} W[k,n]x[n] \exp\{-j2\pi Qn/N[k]\} \tag{2.3}$$



(a) Spectrogram obtained by DFT (b) Spectrogram obtained by CQT

Figure 2.1: Comparison of two time-frequency transforms: DFT vs CQT

In this transcription system, both these methods are implemented and the comparisons are made in Chapter 5.

Chapter 3

Source separation

The techniques utilized in this chapter rely on the fact that polyphonic audio signals that result from the mixing of several different sources are, in terms of physical measures, the weighted sum of the signals corresponding to each individual source. Also, human perception derived from listening to these sound is essentially the superposition of the sensations triggered when listening to each individual source. Therefore, a reasonable mathematical model for the phenomenon of sound source identification is:

$$\boldsymbol{X} = \boldsymbol{B}\boldsymbol{A}. \tag{3.1}$$

In this model, $\boldsymbol{X}$ is our original signal which is a mixture of several sources. As mentioned earlier, these sources are the basis components that form the signal. The collection of these sources is a dictionary of bases and is represented by matrix $\boldsymbol{B}$. This matrix consists of the sources to be identified and is called basis matrix. Matrix $\boldsymbol{A}$ is a set of weight coefficients that represent how much each source defined in $\boldsymbol{B}$ is active in the mixture signal.

In order to find a stationary representation for the basis matrix $\boldsymbol{B}$, it is important to note that the sensation of pitch is strongly related to the harmonic model:

$$x_h(t) = \sum_{m=1}^{M} G_m \cos(2\pi m f_0 t + \varphi_m), \tag{3.2}$$

where M is the number of harmonics of the signal, G_m is the magnitude of each harmonic (and is related to the timbre of that sound), f_0 is the fundamental frequency of the harmonic series and φ_m is the phase of each sinusoidal component.

According to the model in Expression 3.2, each musical note tends to have a stationary representation when considering the magnitude of its DFT or CQT, i.e. the DFT or CQT do not change much during the course of the note.

Figure 3.1 shows one row of the matrix $\boldsymbol{B}$. This row corresponds to one individual note of vibraphone (F#3). The first peak in the picture corresponds to the first harmonic (also called the fundamental frequency), the second peak corresponds to the second harmonic, and so on (some of the harmonics might not be present for some of the instruments). The relative height of the peaks, i.e. the intensity of harmonics might change from one instrument type to another for the same note, that is the reason why different instruments have different timbre.

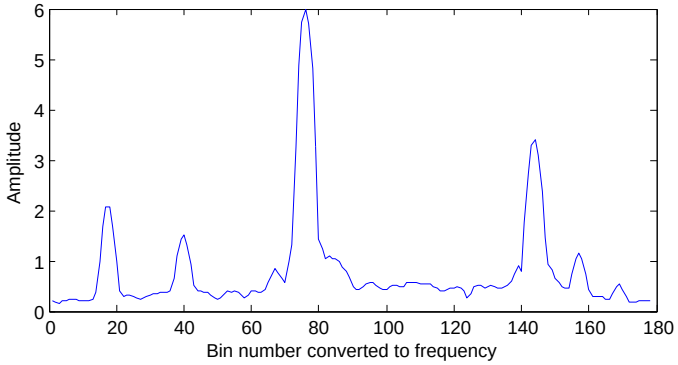


Figure 3.1: One row of matrix B, spectrum for note F#3

When a source i (represent by i^{th} row of $\boldsymbol{B}$) is present at an specific time t, during signal X, we say the source is active at that time. Column i of the $\boldsymbol{A}$ is a time series that shows the activity levels of the source corresponding to $\boldsymbol{A}(i)$

3.0.1 NNLSQ

Several approaches have been designed to deal with the model in Expression 3.1 in the context of automatic music transcription. A commonly used one is the Non-Negative Matrix Factorization (NMF), which aims to obtain both $\boldsymbol{B}$ and $\boldsymbol{A}$ in order to minimize $\|\boldsymbol{X} - \boldsymbol{B}\boldsymbol{A}\|$ with a non-negativity constraint for all elements of $\boldsymbol{B}$ and $\boldsymbol{A}$. Although it is a non-supervised learning technique, experiments [7, 56, 57, 8, 43, 24, 10, 9, 16, 47] show that $\boldsymbol{B}$ usually converges to basis vectors corresponding to notes and $\boldsymbol{A}$ converges to their corresponding activation weights if enough training data is provided.

The non-negative LSQ (NNLSQ) algorithm [28] may be applied in each measurement $\boldsymbol{x}$ in order to minimize $\|\boldsymbol{x}_j - \boldsymbol{B}\boldsymbol{a}_j\|$ constrained to $a_j \geq 0$, $\forall j$, thus obtaining

the corresponding weight vector $\boldsymbol{a}$ when a set of basis functions B is provided [41].

The difference between NNLSQ and NMF lies in the fact that NMF derives both $\boldsymbol{B}$ and $\boldsymbol{A}$ given $\boldsymbol{x}$ whereas NNLSQ only obtains $\boldsymbol{A}$ when provided both the basis matrix $\boldsymbol{B}$ and the signal $\boldsymbol{x}$. In this thesis, we will focus on NNLSQ, because the basis matrix can be easily obtained from isolated notes, and no prior training is necessary.

3.0.2 PLCA

PLCA [54] is a statistical technique and it models the relation between a mixture with the individual components that make up the mixture in the following way:

$$P(x) = \sum_{z} P(z) \prod_{j=1}^{N} P(x_j|z)$$

where $P(x)$ is the probability distribution of the random variable x, $P(z)$ is the probability distribution of the latent variable z, and $P(x_j|z)$ is the probability distribution of dimensions of x given the latent variable z. To solve the problem, we aim to find the optimum value for the marginals such that their product most appropriately describes the random variable x.

In the context of audio source separation, since the spectrogram is a two dimensional representation of the recording along the time and frequency axis, we use a two dimensional version of PLCA where $P(x)$ is the spectrogram of the audio recording, z is the note that is active at a certain time. The marginal distributions are $P(t|z)$, and $P(f|z)$ and show the intensity of the latent variable at time and frequency domain respectively:

$$P(x) = \sum_{z} P(z)P(f|z)P(t|z), \tag{3.3}$$

For known instrument types we already have the values for $P(x|f)$ and we only need to obtain the values for $P(x|t)$ which is the information about the activity levels of each source

The estimation of the weights and time vectors is performed using a variant of the EM algorithm. In short this algorithm contains an expectation and a maximization step which we alternate between in an iterative manner until convergence. In the expectation step we estimate the posterior given spectral basis vectors and weights

vectors:

$$R(x,z) = \frac{P(z) \times P(f|z) \times P(t|z)}{\sum_{z'} P(z') \times P(f|z') \times P(t|z')}$$

and in a maximization step we estimate the spectral basis vectors and weights vectors given the posterior

$$P(z) = \int P(x)R(x,z)dx$$

$$P(x_j|z) = \frac{\int ... \int P(x)R(x,z)dx_k, \forall k \neq j}{P(z)}$$

PLCA gives this linear problem a probabilistic interpretation, which is numerically similar to NMF [55].

3.0.3 Separation Example

To illustrate the use of this decomposition method, a simple problem is presented. Assume that we observe a two dimensional random variable composed of three 2-d Gaussians with diagonal covariances:

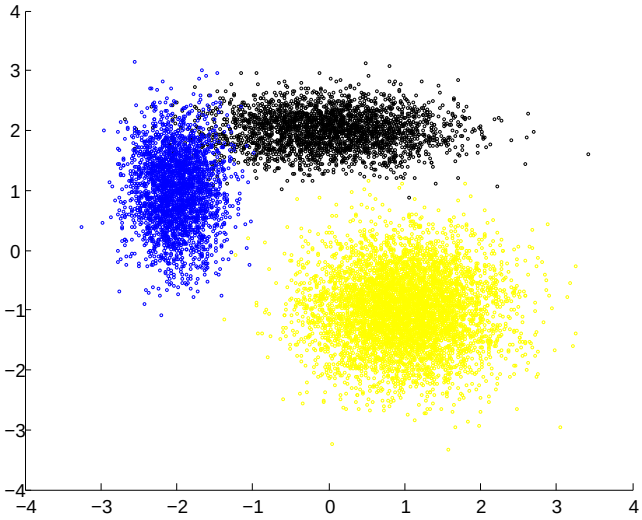
$$x \sim \frac{1}{2}N\left(\begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}0.4 & 0\\0 & 0.4\end{bmatrix}\right) + \frac{1}{4}N\left(\begin{bmatrix}0\\2\end{bmatrix}, \begin{bmatrix}0.7 & 0\\0 & 0.1\end{bmatrix}\right) + \frac{1}{4}N\left(\begin{bmatrix}-2\\1\end{bmatrix}, \begin{bmatrix}0.1 & 0\\0 & 0.4\end{bmatrix}\right)$$

Figure 3.0.3 shows the mixture of Gaussians and how PLCA separates the components. Subfigure 3.2(a) shows each source in a different color, Subfigure 3.2(b) shows how these sources are mixed and finally Subfigure 3.2(c) depicts the mixture of the components separated by PLCA.

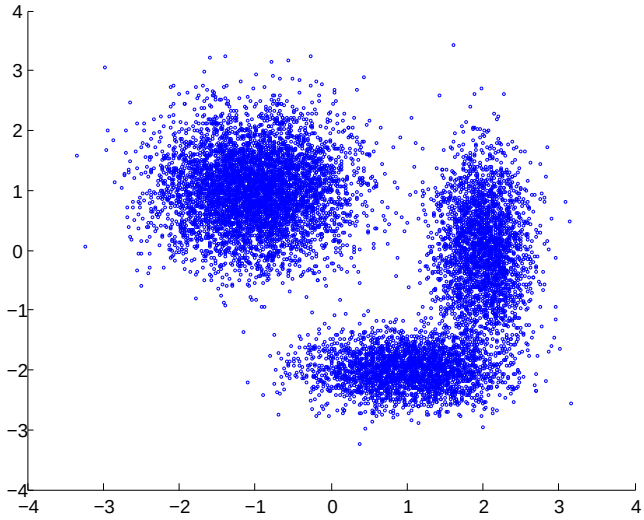
3.0.4 Example with audio

This section shows shows how a factorization method can be used to separate the sources of our 6-note audio example. We have 5 2-dimensional sources that make up the music (a 2-dimensional mixture of the 5 sound sources).

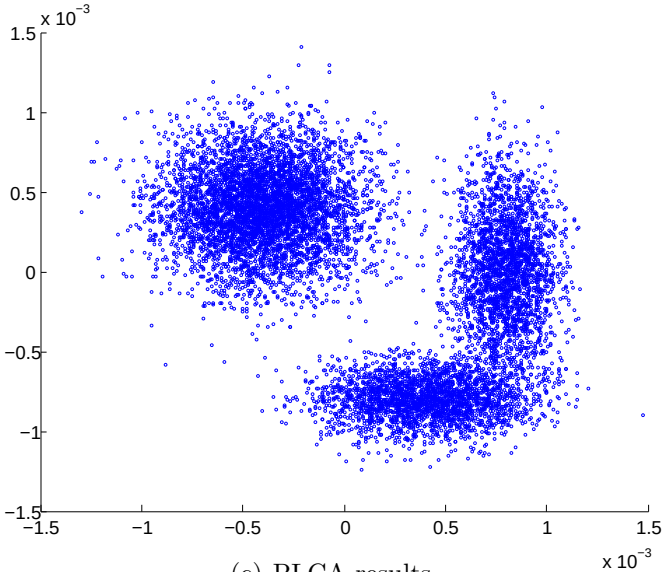
Figure 3.0.4 shows the mixture of sound sources and how PLCA separates the components. Subfigure 3.3(a) shows each of these sources, that are placed in side by side columns, Subfigure 3.3(b) shows how these sources are mixed to make the audio and finally Subfigure 3.3(c) depicts the mixture of the components obtained by PLCA.



(a) Gaussian sources

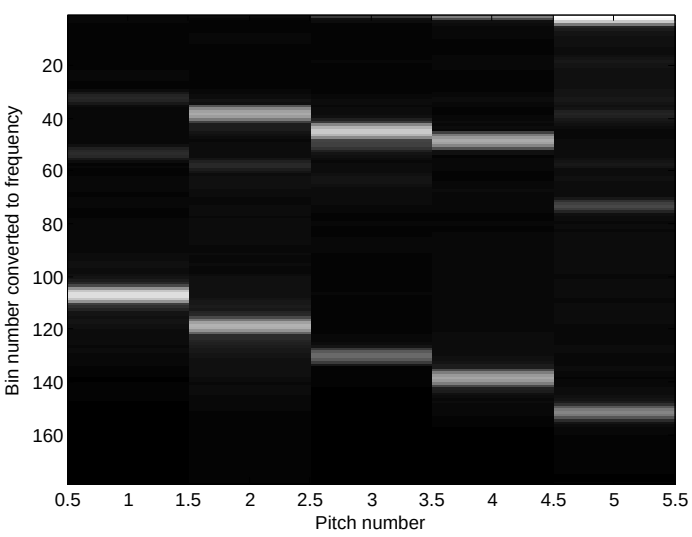


(b) Gaussian mixture

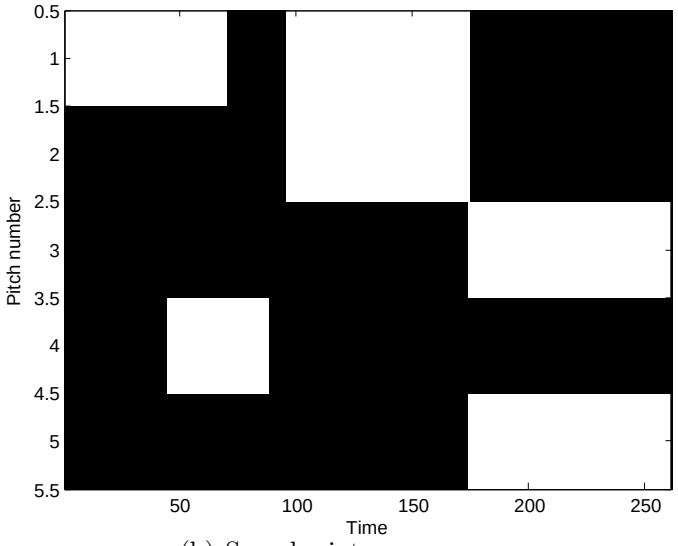


(c) PLCA results

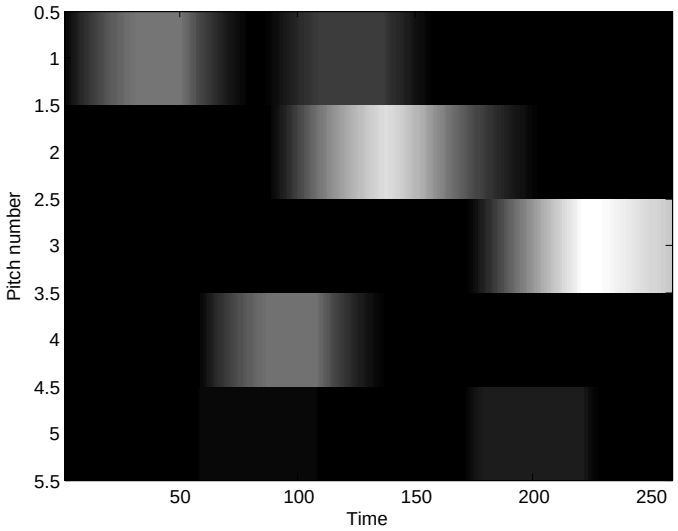
Figure 3.2: Separation of sources in a Gaussian mixtures. Subfigure (b) shows the Gaussian mixture, and subfigure (c) is the result achieved by PLCA which is an approximation of the original mixture.



(a) Sound sources



(b) Sound mixture



(c) PLCA results

Figure 3.3: Separation of sound sources from the audio by PLCA.

You can see from the figure that the result of the separation obtained by PLCA is very similar to the original mixture.

In the PLCA model shown in Expression 3.3, $P(x)$ (3.3(b)) is the spectrogram of the audio recording, z is the note that is active at a certain time, and the marginal distributions $\boldsymbol{A} = P(t|z)$ (Subfigure 3.3(c)) and $\boldsymbol{B} = P(f|z)$ (Subfigure 3.3(a)) represent the intensity of the latent variable in the time domain and the frequency domain, respectively.

This method was originally designed for source separation and in order to make the problem of music transcription fit this model. The problem of extracting the notes from an audio recording is reduced to extracting the notes from the spectrogram of that recording. We assume that we have N sources (N equals the number of the notes an instrument can play) that make up the original audio. The output of a factorization mathod will be N digital signals; each signal has the duration equal to that of the original mixture. The signal for each note will ideally be silent during the whole signal except for the frames where the note is active.

Chapter 4

Onset detection

The final stage is to extract the onset from the activation matrix. An onset of a musical signal is the time instance when a note becomes activated. We derive onsets by analyzing the activation matrix $\boldsymbol{A}$ which is the output of the algorithms described in the previous section. The activation matrix is $N \times M$, where N is the number of pitches and M is the number of frames that make up the length of the original audio.

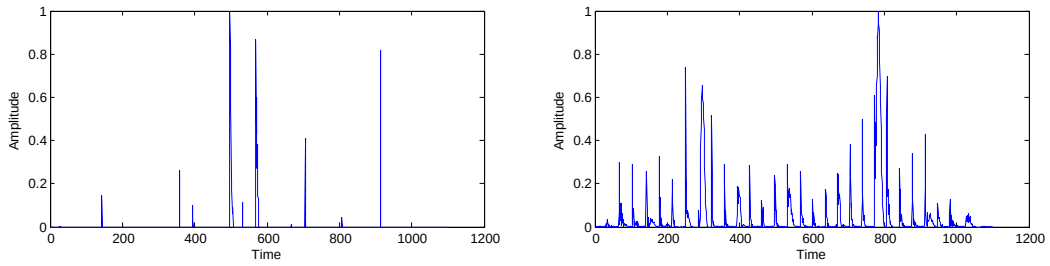
Once the weight matrix $\boldsymbol{A}$ is obtained, it may be used to find discrete events, such as note onsets. In this chapter, two algorithms for onset detection are presented. These algorithms are designed for real-time applications; in the context of live performance, all the steps of the transcription have to be causal, i.e. we cannot use information from future. But before applying the onset detection algorithms, we need a filtering step to make the matrix A a more appropriate input for the onset detection algorithms.

4.1 Filtering and smoothing techniques

Figure 4.1 shows the activation levels of one pitch, one row of the activation matrix, that is played twice during an audio recording.

Subfigure 4.1(a) shows the activation levels obtained by NNLSQ, and Subfigure 4.1(b) depicts the same information obtained by PLCA. In Subfigure 4.1(a), the two peaks that correspond two the position of note onsets are clearly visible, and thus can be extracted with appropriate algorithms (Chapter 4). The activation levels obtained by PLCA appear to be noisier compared to the results of NNLSQ. Although the same peaks are present, the boundaries are not as clear. If we perform the onset

detection algorithm on it, there is a chance that the extra peaks will be detected as onsets and result in false positives, which in turn will reduce the accuracy of the final transcription system. Moreover, since the actual onsets don't have higher values than the surrounding impulses, the onset detection algorithm might miss them, causing false negatives which will also contribute to lower accuracy. To make the information acquired by PLCA more useful, we perform a post-processing step.

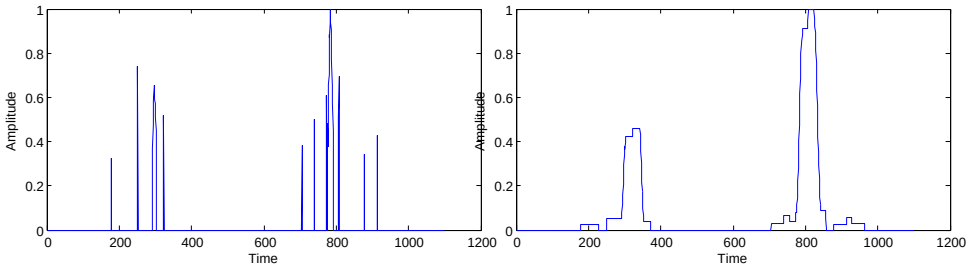


(a) Activation levels of one pitch obtained by NNLSQ

(b) Activation levels of one pitch obtained by PLCA

Figure 4.1: Comparison of activity levels obtained by NNLSQ and PLCA

First, we normalize the activation matrix by dividing it by the maximum value of the matrix (we will skip this step, in the context of live performance transcription, since this will make the algorithm non-causal). After normalization, the first step is assigning 0 to the low values that are clearly not onsets. We set the threshold to be 30% of the maximum level of activity after some trials, and assign zero to anything below the threshold. The result is shown in Figure 4.2(a)



(a) activation levels after normalization

(b) activation levels after moving average

Figure 4.2: The effect of the filtering techniques on the

The next step is a moving average. We set the span of this window to 50 (we pick 50 because it is not too long to shift the peaks so much and not too short that

it doesn't remove impulses). The moving average replaces the values in the window by their average (Figure 4.2(b)). This filter, reduces the number of false negatives by emphasizing the onsets, and reduces the number of false positives by eliminating the impulses.

4.2 Thresholding

Figure 4.2 shows the shape of an onset that is taken from the activation matrix. Given an activation matrix, the task of onset detection is defined as finding the indices for which the activation levels of a pitch are similar to the shape shown in Figure 4.2. The thresholding method is a rule-based decision algorithm that yields decisions as to when notes are played (a similar method is used by Foster, Schloss, and Rockmore [19] The algorithm works as follows: First, the activation values for the noise bases, as well as all activation levels below a certain threshold α, are set to zero. After that, all values whose activation level differential are below another threshold β are set to zero. When a non-zero value for the activation value is found, an adaptive threshold value is set to that level multiplied by an overshoot factor γ. The adaptive threshold decays linearly at a known rate θ, and all activation levels below it are ignored. Finally, the system deals with polyphony by assuming that a certain activation level only denotes an onset if it is greater than a ratio ϕ of the sum of all activation values for that frame. After this process, a list of events described by onset and pitch is yielded. We set the values for these parameters by performing a greedy search on a subset of training data for each method of separation.

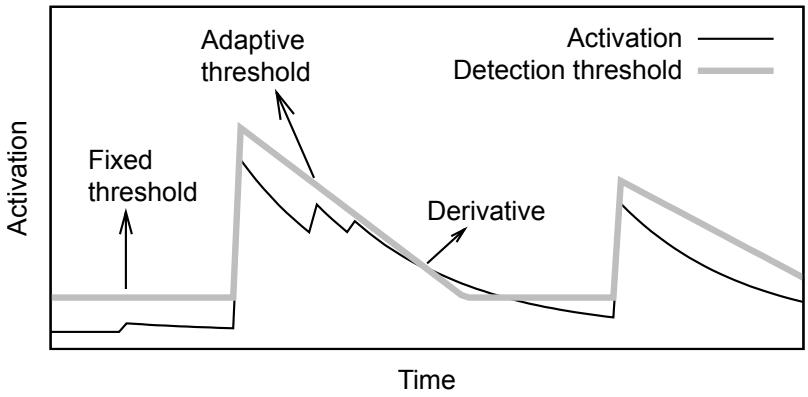


Figure 4.3: Detection of onset using adaptive thresholding.

4.3 Support vector machines

We can define the task of onset detection as finding a mapping between a set of features derived from activation matrix and the onsets of the notes, i.e. learning a function f for the following mapping: $x_i \rightarrow y_i$ (where x_i is a feature vector and y_i the label assigned to the feature vector x_i). This function is chosen in a way that tries to minimize the error on training data, so that it will perform well on a test set. One way to find the best mapping is using Support Vector Machine (SVM). SVM identifies the optimal separating hyperplane (OSH) that maximizes the margin of separation between observations of two classes. The observations that lie closest to the OSH are called support vectors. It can be shown that the solution that maximizes the margin between the hyperplane and the support vectors corrosponds to the best generalization ability.

I used the Sequentional Minimal Optimization (SMO) of Weka Software for my learning system. SMO is an implementation of SVM where the data has nominal class, which is the case in our onset detection.

The dataset for classification consists of l observations each of which consist of a pair: a feature vector $X_i \in R^n, i = 1, ..., l$ and the associated label y_i, given to us by a binary class SVM which will be discussed shortly.

I used a Support Vector Machine (SVM) classifier with a polynomial kernel because of the structure of the problem. Two parameters need to be determined before using polynomial kernels: round-off error and tolerance parameter (P; L). To customize the classifier to our problem we need to find which values for L and P work best for our problem; the difference in classification accuracy between a good pair of (P; L) and a bad one can be huge. Therefore, parameter searching should be done before training the whole model. The values of P and L are set to maximize the precision and recall on a randomly selected subset of the training data. These values are set to (1.0000e-005; 0.1).

For a comprehensive tutorial on Support Vector Machine, we refer the reader to the SVM tutorial by Burges [13].

The training set is initialized using features and labels extracted from activation matrix. Figure 4.4 shows the overview of the process.

Feature vectors

Features for the training set are also extracted from activation matrices of the training data. The feature vector corresponding to frame consists of the following features:

- The activation level at frame t.
- activation levels of frames prior to t, i.e. frames t -m, t - m + 1, ..., t - 1.
- activation levels of frames after the frame t, i.e. frames t + 1, t + 2, t + n.

Therefore, we obtain feature vector of size m+ n + 1. The values for m and n are determined the same way as P and L. Both n and m are set to 30 in the SVM experiments in Chapter 5.

Labels

We extract the labels from MIDI files. A frame t is labeled 1 (note on) if a note is active at time t and 0 otherwise. The SVM is trained on this data, producing an onset detector whose per-frame output is interpreted as a probability (from 0 to 1) of an onset in the frame. Machine learning offers one promising approach, but it is limited by the availability of labeled training data and also the accuracy of the alignment because it's manually labeled) By finding note onsets, we can segment continuous music into discrete note events.

Preprocessing

The preprocessing steps are taken in order to prepare the training samples, so that it is easier to find the underlying patterns. We perform two pre-processing steps:

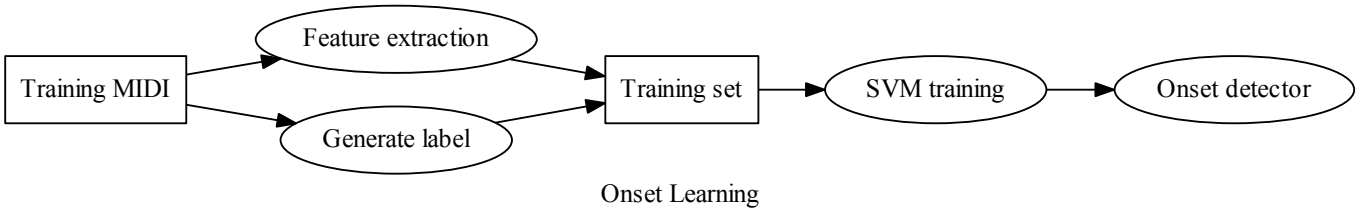


Figure 4.4: Onset detection using the machine learning method SVM

- Normalization: We normalize the attributes so that they have a Gaussian distribution with average 0 and standard deviation of 1. This is done by subtracting the attributes by their average and then dividing them by their standard deviation. After normalization, all the features will be in the same range and therefore they will have the same weight in training the system.

- Resample: Since the number of samples labeled with 1 is fewer than the number of samples labeled with 0, we need to resample the data set to balance the classes, so that we have approximately the same number of instances labeled 0 and 1. This works with removing some of the instances of the class with high number of data points, and repeating some of the instances of the class with lower number of data points.

After performing the onset detection algorithm, we remove the notes that have a duration shorter than 0.1 seconds, because practically notes last longer than this threshold. This step further reduces the number of false positives.

Chapter 5

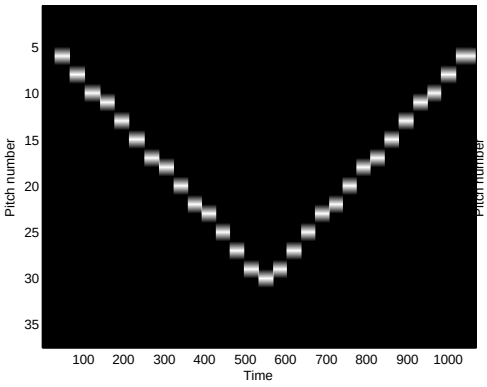
Experiments

5.1 Vibraphone Dataset

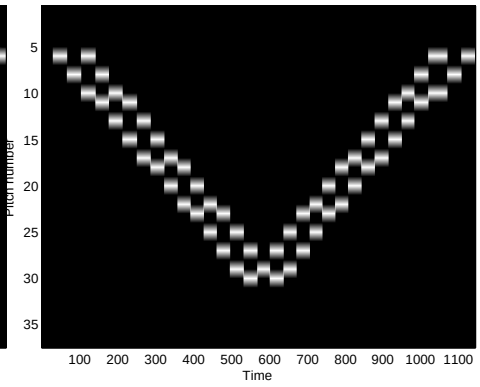
Experiments were conducted using recordings of an acoustic vibraphone surrounded by loudspeakers playing prerecorded music in addition to the vibraphone sounds for the pieces. The loudness of the accompaniment was set to a level suitable for a small concert. Three microphones were used to get audio from the vibraphone: one in the left end, one in the center and one in the right end. The audio data from those three microphones was analyzed independently, in order to determine what is the influence of the microphone positioning in the final results.

Table 5.1: Description of the tracks used in the evaluation.

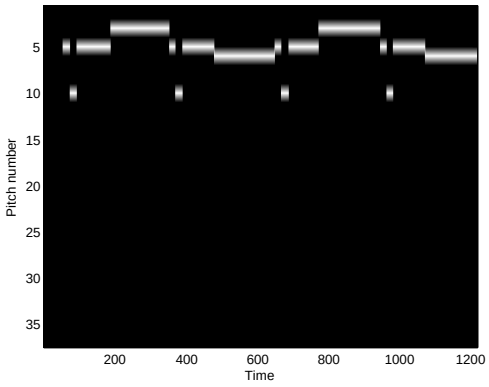
Track	Microphone	Sustain	Ambient music	Other notes	Duration
1	Center	No	No	Piece 1	12 seconds
2	Center	No	Yes	Piece 1	14 seconds
3	Center	Yes	No	Piece 1	21 seconds
4	Center	Yes	Yes	Piece 1	19 seconds
5	Mix	Yes	Yes	Piece 2	14 seconds
6	Center	Yes	Yes	Piece 2	14 seconds
7	Left	Yes	Yes	Piece 2	14 seconds
8	Right	Yes	Yes	Piece 2	14 seconds
9	Center	No	No	Remix (piece 1)	13 seconds
10	Center	No	Yes	Remix (piece 1)	22 seconds
11	Center	Yes	Yes	Remix (piece 1)	22 seconds
12	Center	Yes	Yes	Piece 1, added crowd noise	19 seconds



(a) Ground truth for tracks 1, 2, 3, 4, 10, 11, 12



(b) Ground truth for tracks 9



(c) Ground truth for tracks 5, 6, 7, 8

Figure 5.1: Ground truth for dataset tracks

Polyphony was also considered. Although it is only possible to play two (or four, if the musician holds two mallets in each hand) notes at the same time, by using the sustain pedal the vibraphone is capable of producing several notes at the same time. Some pieces were recorded without using the sustain pedal, and some were recorded with the sustain pedal. Also, in order to increase the amount of data used in the evaluation, some pieces were artificially remixed. Table A.13 describes all tracks used in the evaluation.

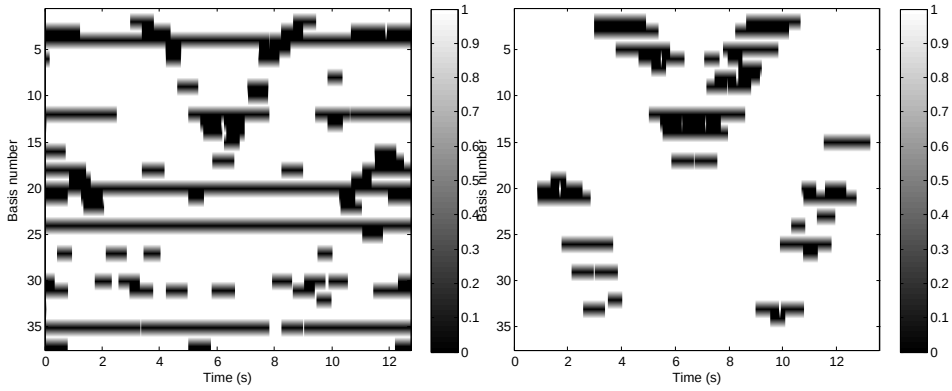
The ground truth transcription for each track was annotated manually. Figure 5.1 shows the ground truth for the tracks of the dataset. The x-axis shows the time flow of the audio, and the y-axis represents the frequency, in the case of vibraphone dataset,

we have 37 frequency bands, one for each vibraphone bar.

5.2 Unsupervised

As mentioned earlier, we are trying to find the activation matrix $\boldsymbol{A}$, given the original signal X. Not knowing the basis matrix $\boldsymbol{B}$, and trying to extract $\boldsymbol{B}$ from data, is called unsupervised. In the case of an unsupervised algorithm, matrix $\boldsymbol{B}$ is initialized randomly and the algorithm optimizes and makes $\boldsymbol{B}$ converge to the pitches.

Figure 5.2 shows the activation matrix for one monophonic and one polyphonic track, obtained by an unsupervised PLCA. The algorithm returns one component for each onset in the original audio. But these components are not the pitches we were looking for, since the algorithm doesn't have enough information to converge to the pitches. The evaluation results are presented in Table 5.2.



(a) Transcription results for a monophonic track

(b) Transcription results for a polyphonic track

Figure 5.2: Transcription results of monophonic and polyphonic recordings using unsupervised PLCA.

The evaluation results in Table 5.2 are not satisfactory. The reason for that is that, although the return bases are at the approximately correct time instance, their fundamental frequency is not related to the pitch in the original signal. Given these reasons, it makes sense to extract the basis matrix from recordings of isolated notes. Having the correct pitches, we can extract the activation matrix $\boldsymbol{A}$ more accurately and get better transcription results. These result will be presented in the following sections.

5.3 Supervised

Like we mentioned in the last section, we will provide the algorithm with the basis matrix $\boldsymbol{B}$; in this section we show that the transcription results are improved significantly.

5.3.1 Obtaining bases

First, it is necessary to obtain a proper basis matrix $\boldsymbol{B}$ that represents the individual sound sources into which the signal is decomposed. After that, a decision algorithm must be executed over the activation matrix $\boldsymbol{A}$ in order to obtain decisions on when and where the mallets hit the vibraphone.

All audio processing in this paper relies on a frame-wise spectrogram representation calculated as follows. The signal is processed in frames of 46 ms with a 23 ms overlap. Each frame is multiplied by a hanning window, zero-padded to twice its length and has its DFT/CQT calculated. Finally, the frequency domain representation is trimmed in order to ignore values outside the frequency band in which the vibraphone typically emits sounds.

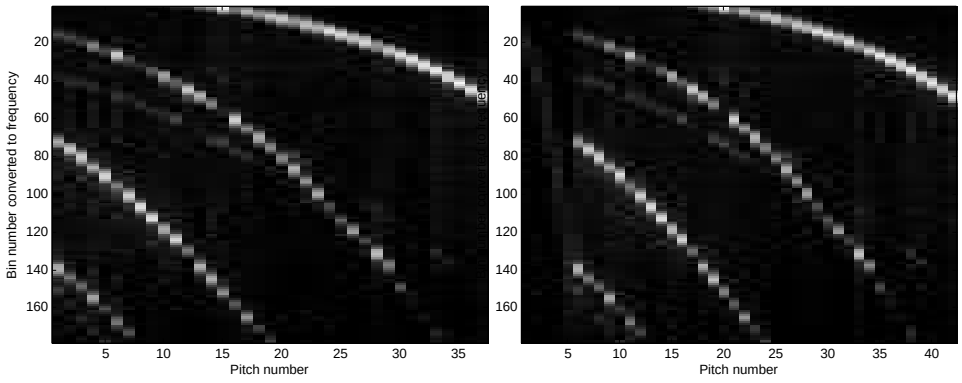
All basis functions are constructed by taking the spectrogram of a recording containing a single note hit and averaging the first few frames. Only these few frames are used because, in the vibraphone sounds, the harmonics decay quickly. Using all data from a particular note would converge to a spectrogram that would be dominated by the fundamental frequency of the series instead of the whole harmonic series. Then, the spectrogram is normalized in order to have unity variance (but not zero mean). The normalization is used in order to ensure that the values in each row of the activation matrix $\boldsymbol{A}$ have the same scale, hence a high value for one row is likely to be a high value for any row.

Table 5.2: Transcription results of monophonic and polyphonic recordings using unsupervised PLCA.

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.0345	0.0116	0.0174	0.0712	0.0246	0.0365
9	0	0	0	0.0029	0.0021	0.0024

5.3.2 Noise Model

A set of additional basis functions, called noise basis functions, are also added to the the basis matrix (or dictionary). Noise basis functions are calculated as triangles in the frequency domain, which overlap by 50% and have center frequencies that start at 20 Hz and increase by one octave from one nose base to the next one. This shape aims to give the system certain degrees of freedom for modeling background noise, specifically non-harmonic sounds, as filtered white noise, so that background noise is less likely to be modeled as a sum of actual notes, that is, less background noise is expected to affect the contents of the activation matrix. Figure 5.3.2 depicts the basis function, before and after adding the noise model.



(a) Basis matrix (b) Basis matrix with noise model

Figure 5.3: Track 1: SVM vs perfect

In the following sections, the result of the experiments using different methods are presented.

5.3.3 Evaluation Metrics

We use three metrics used to evaluate the system:precision, recall, and f-measure as define below:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives}$$

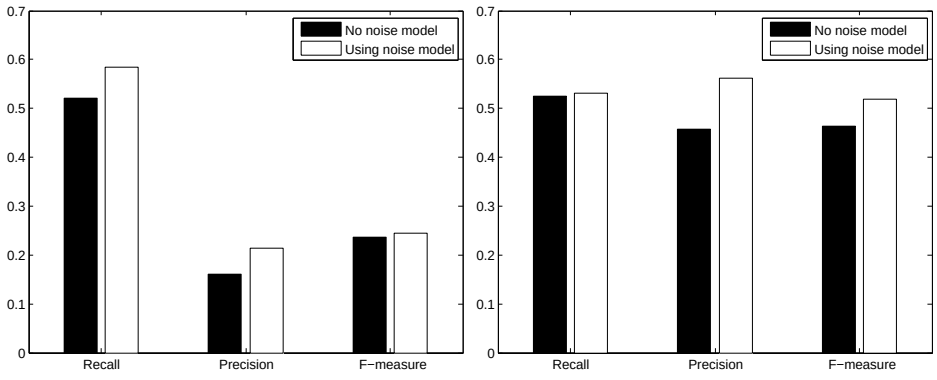
$$F - measure = 2.\frac{precision.recall}{precision + recall}$$

In order to have an accurate transcription system, we need both precision and recall to be high. Having only high precision, or high recall is not desirable; for example a transcription system that returns every possible note at every time frame will have 100% recall but every poor precision. On the other hand a transcription system that transcribes part of the audio correctly, and doesn't return any notes for the rest of the audio will have a 100% precision, but very low recall. Therefore, we need to consider both precision and recall, as well as their a combination of them (their harmonic mean) to evaluate the system.

In Chapter 5, all the bar graphs show the average of precision, recall, and f-measure over all tracks of the dataset. Details of the metrics for each track are presented in tables in Chapter A.

5.4 Noise model

As mentioned earlier, we use noise model to reduce the impact of background music and noisy environment. The experiments show that the transcription results are improved by using a noise model. Figure 5.9 compares the result of the transcription with Table 5.4 and without Table 5.3 using a noise model for NNLSQ. As can be seen, the results are consistently improved. Therefore, we use a noise model for the experiments from now on.

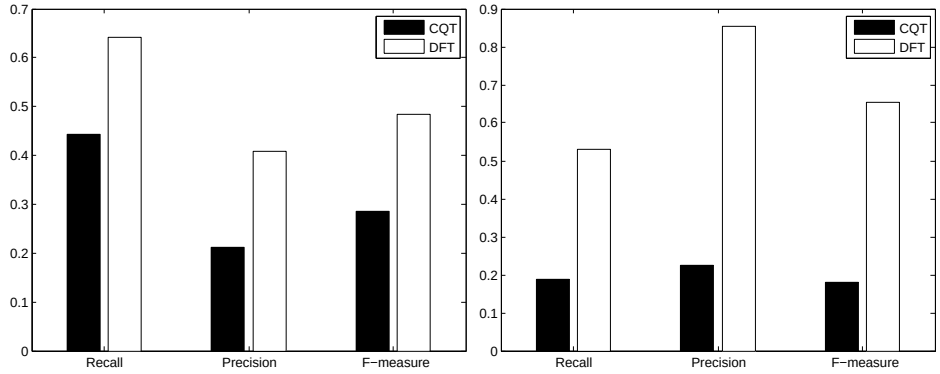


(a) Noise model for PLCA (b) Noise model for NNLSQ

Figure 5.4: Transcription results with and without using noise model.

5.5 CQT vs DFT

In this section we show the results of the transcription using two different time-frequency transforms: CQT and DFT. Figure 5.5 compares the two methods.



(a) DFT vs CQT for PLCA (b) DFT vs CQT for NNLSQ

Figure 5.5: Transcription results using CQT vs DFT.

As you can see in the figure, when the audio is recorded in a noisy environment, DFT performs better than CQT for both methods. Therefore, we use DFT to transform the signal from time-domain to frequency domain. Tables 5.5 and 5.6 show the results for NNLSQ. (Refer to appendix for more detailed tables.)

Table 5.3: Transcription results for NNLSQ no noise model

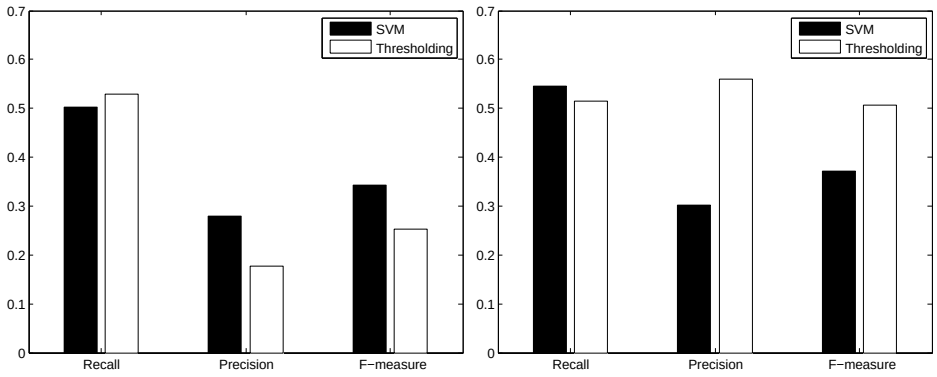
Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.5517	0.6667	0.6038	0.5333	0.6609	0.5902
9	0.5517	0.6275	0.5872	0.5067	0.6039	0.5510

Table 5.4: Transcription results for NNLSQ using noise model

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.5862	0.8500	0.6939	0.5651	0.8829	0.6891
9	0.5517	0.8649	0.6737	0.5176	0.8394	0.6404

5.6 SVM vs Thresholding

We explored two onset detection methods for our system that are causal and therefore can be used in the context of live performance: thresholding, and SVM. Figure 5.6 compares the results of the transcriptions when these two methods are employed.



(a) SVM onset detection for PLCA (b) SVM onset detection for NNLSQ

Figure 5.6: Transcription results of using Thresholding vs SVM.

As you can see in Figure 5.6, using SVM works better with PLCA, whereas thresholding works better for NNLSQ, which has a smoother output. From this results, we can conclude that, although SVM is a more sophisticated method, it doesn't perform substantially better, and, at some situations, it works even worse than the simple thresholding method. Plus, training a classifier for onset detection is a difficult task that requires several pre-processing steps. On the other hand, after finding the cor-

Table 5.5: Transcription results using NNLSQ and DFT

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.5862	0.8500	0.6939	0.5651	0.8829	0.6891
9	0.5517	0.8649	0.6737	0.5176	0.8394	0.6404

Table 5.6: Transcription results using NNLSQ and CQT

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.1724	0.0505	0.0781	0.2309	0.4129	0.3193
9	0.1207	0.0365	0.0560	0.2313	0.3972	0.3021

rect parameters for the kernel of SVM once, we can run it on any data, but in case of thresholding, we have to look for the appropriate threshold, each time the scaling and the range of the data changes. Tables 5.8 and 5.7 compare the results of the transcription using SVM and thresholding for onset detection.

Table 5.7: Transcription using NNLSQ with thresholding

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.5517	0.8421	0.6667	0.5314	0.8764	0.6616
9	0.5345	0.8611	0.6596	0.5033	0.8396	0.6294

Table 5.8: Transcription using NNLSQ with SVM

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.9310	0.6000	0.7297	0.8885	0.5866	0.7067
9	0.5000	0.6170	0.5524	0.4605	0.5735	0.5108

5.7 Sustain versus no sustain

5.8 Upper bound

As a last experiment, we present an upper bound for the transcription system with removing the false positives to achieve a higher precision. We obtain this upper bound by multiplying the activation matrix resulted from decomposition methods, by ground truth. This operation will remove all the false positives and we can have an estimate of how much room there is for improving the system. The evaluation metrics of the upper bound system is presented in Figure 5.8. Figure 5.8 shows the transcription result of upper bound system versus the ground truth for track 2 of the dataset.

5.9 Multi-f0 estimation method

We find the transcription of music that is recorded with background noise, very challenging. We believe that this system using decomposition methods, performs

better than state of the art transcription systems in this context. In this section a comparison between our system, and a state of the art multi-f0 estimation system introduced by Klapuri [30] is made. The results are presented in Figure 5.9.

As you can see in the figure, methods presented in this thesis work considerably better in the presence of noise and methods used in the past that are only tested on recordings with no environment noise will fail to transcribe noisy audio signals. The results are presented in Table 5.9.

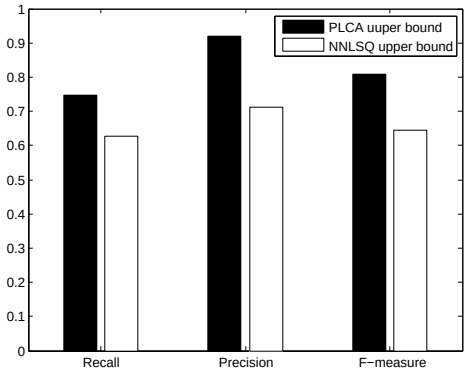
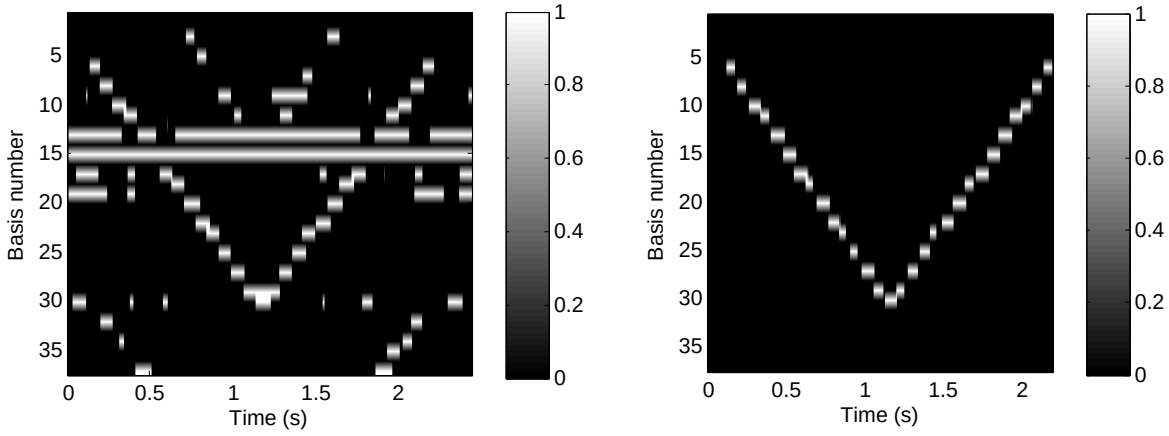


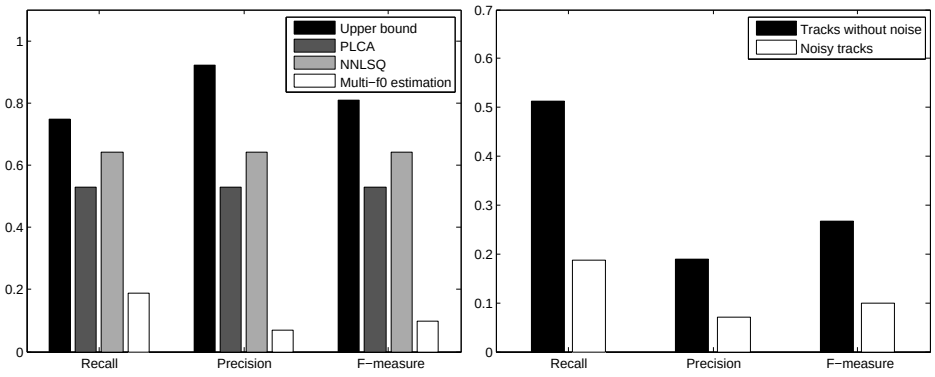
Figure 5.7: Upper bound for transcription system.



(a) SVM

(b) Upper bound

Figure 5.8: Transcription results versus ground truth for track 2.



(a) Multi-f0 vs NNLSQ vs PLCA

(b) Multi-f0 method for tracks with and without noise

Figure 5.9: (a) Transcription results comparing our system with multi-f0 estimation method. Black bars show the results for NNLSQ, gray bars represent PLCA, and multi-f0 estimation method is represented by white bars. (b) Transcription results applying multi-f0 method to the data with background noise vs data with no background noise.

Table 5.9: Transcription results using the multi-f0 estimation method

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.6552	0.1638	0.2621	0.6120	0.1569	0.2497
9	0.3621	0.2000	0.2577	0.4219	0.2350	0.3018

Chapter 6

Conclusion

This thesis presented a comparison of two different techniques – PLCA and NNLSQ – for the problem of detecting note events of a vibraphone, in the context of a live performance. Two causal algorithm for onset detection are also presented that perform onset detection on activation matrix (unlike most algorithms that perform this task on audio) and may be implemented as a real-time process. The performance of the transcription system is evaluated considering several different use scenarios, with two different kinds of ambient noise (accompaniment music and a recording of a crowd talking).

The results show that the special noise-related basis can improve the overall performance of the system. The system has shown better performance in low-noise situations, even in the presence of high levels of polyphony.

The results of experiments also indicate that the non-negative factorization methods are a suitable solution to the transcription of noisy recordings and they outperform the state of the art multi-f0 estimation method for transcription.

We also observed that using isolated notes to extract the basis functions improves the transcription results significantly.

In low-noise cases, the proposed system is shown to yield symbol data with around 60% accuracy, which degrades to around 50% when considering accompaniment music cases and to around 20% in highly noisy environments. This shows that, although it is possible to use the proposed method under certain constrains, there is still need for further research in order to increase the usability.

6.1 Future Work

We found the problem of vibraphone transcription from noisy audio very challenging and there are still several unsolved issues that need further research. The following is a list of some of these issues:

- Finding the necessary conditions to make the unsupervised algorithm (in terms of settings and the necessary amount of data for training) converge to the pitches.
- Creating the confusion matrix from the values for true positives, false positives, etc. The confusion matrix will help us identify the type of errors that are more frequent and focus on removing those types of errors.
- Source separation evaluation: all the evaluation metrics used in the thesis only evaluate the transcription of the music, it would be interesting to investigate source separation metrics such as SDR (Signal Distortion Ratio)
- Extending the system to support several instruments of different types, such as woodwind instruments that have different onset shapes.
- Creating a better noise model (a database for noise for instance) that will improve the accuracy for highly noisy environments

Improvements might also be achieved with regard to its usability. During training, it would be interesting for the program to automatically obtain basis models for all notes from a subset of all notes, as well as suitable values for the parameters necessary for the event detection stage. That would allow greater flexibility to the system. Also, a better noise removal algorithm might be useful, since the results show that the performance of the algorithm is highly dependent on the amount of noise in the input signal.

Bibliography

[1] F. Argenti, P. Nesi, and G. Pantaleo. Automatic transcription of polyphonic music based on the constant-q bispectral analysis. *Audio, Speech, and Language Processing, IEEE Transactions on*, 1(99):1, 2010.

[2] I. Barbancho, A.M. Barbancho, A. Jurado, and L.J. Tardn. Transcription of piano recordings. *Applied Acoustics*, 65(12):1261 – 1287, 2004. Musical Acoustics.

[3] M.A. Bartsch and G.H. Wakefield. Audio thumbnailing of popular music using chroma-based representations. *Multimedia, IEEE Transactions on*, 7(1):96–104, 2005.

[4] J.P. Bello, L. Daudet, and M.B. Sandler. Automatic piano transcription using frequency and time-domain information. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(6):2242 –2251, nov. 2006.

[5] E. Benetos and S. Dixon. Multiple-instrument polyphonic music transcription using a convolutive probabilistics model. In *In 8th Sound and Music Computing Conf.*, pages 19–24, July 2011.

[6] E. Benetos and S. Dixon. Polyphonic music transcription using note onset and offset detection. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 37 –40, may 2011.

[7] N. Bertin, R. Badeau, and G. Richard. Blind signal decompositions for automatic transcription of polyphonic music: Nmf and k-svd on the benchmark. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 1, pages I–65 –I–68, april 2007.

[8] N. Bertin, R. Badeau, and E. Vincent. Fast bayesian nmf algorithms enforcing harmonicity and temporal continuity in polyphonic music transcription. In

Applications of Signal Processing to Audio and Acoustics, 2009. WASPAA '09. IEEE Workshop on, pages 29 –32, oct. 2009.

[9] N. Bertin, R. Badeau, and E. Vincent. Enforcing harmonicity and smoothness in bayesian non-negative matrix factorization applied to polyphonic music transcription. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(3):538 –549, march 2010.

[10] N. Bertin, C. Fevotte, and R. Badeau. A tempering approach for itakura-saito non-negative matrix factorization. with application to music transcription. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 1545 –1548, april 2009.

[11] Judith C Brown. Calculation of a constant q spectral transform. *Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

[12] I. Bruno, S.L. Monni, and P. Nesi. Automatic music transcription supporting different instruments. In *Web Delivering of Music, 2003. 2003 WEDELMUSIC. Proceedings. Third International Conference on*, pages 37 – 44, sept. 2003.

[13] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[14] Yu-Ren Chien and Shyh-Kang Jeng. An automatic transcription system with octave detection. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 2, pages II–1865, 2002.

[15] G. Costantini, M. Todisco, and R. Perfetti. On the use of memory for detecting musical notes in polyphonic piano music. In *Circuit Theory and Design, 2009. ECCTD 2009. European Conference on*, pages 806 –809, aug. 2009.

[16] G. Costantini, M. Todisco, R. Perfetti, R. Basili, and D. Casali. Svm based transcription system with short-term memory oriented to polyphonic piano music. In *MELECON 2010 - 2010 15th IEEE Mediterranean Electrotechnical Conference*, pages 196 –201, april 2010.

[17] Giovanni Costantini, Renzo Perfetti, and Massimiliano Todisco. Event based transcription system for polyphonic piano music. *Signal Processing*, 89(9):1798 – 1811, 2009.

[18] O. Derrien. Multi-scale frame-based analysis of audio signals for musical transcription using a dictionary of chromatic waveforms. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 5, page V, may 2006.

[19] S. Foster, W.A. Schloss, and A.J. Rockmore. Toward an intelligent editor of digital audio: Signal processing methods. *Computer Music Journal*, 6(1):42–51, 1982.

[20] O. Gillet and G. Richard. Automatic transcription of drum loops. In *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, volume 4, pages iv–269 – iv–272 vol.4, may 2004.

[21] O. Gillet and G. Richard. Automatic transcription of drum sequences using audiovisual features. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, volume 3, pages iii/205 – iii/208 Vol. 3, march 2005.

[22] O. Gillet and G. Richard. Transcription and separation of drum signals from polyphonic music. *Audio, Speech, and Language Processing, IEEE Transactions on*, 16(3):529 –540, march 2008.

[23] Masataka Goto. A real-time music-scene-description system: predominant-f0 estimation for detecting melody and bass lines in real-world audio signals. *Speech Communication*, 43(4):311 – 329, 2004. Special Issue on the Recognition and Organization of Real-World Sound.

[24] G. Grindlay and D. Ellis. Multi-voice polyphonic music transcription using eigeninstruments. In *Applications of Signal Processing to Audio and Acoustics, 2009. WASPAA '09. IEEE Workshop on*, pages 53 –56, oct. 2009.

[25] G. Grindlay and D.P.W. Ellis. A probabilistic subspace model for multi-instrument polyphonic transcription. In *11th Int. Society for Music Information Retrieval Conf*, pages 21–26, 2010.

[26] Y. Guo and M. Zhu. Audio source separation by probabilistic latent component analysis.

[27] H. Hajimolahoseini, M.R. Taban, and H.R. Abutalebi. Automatic transcription of music signal using harmonic elimination method. In *Telecommunications, 2008. IST 2008. International Symposium on*, pages 559 –563, aug. 2008.

[28] Richard J. Hanson and Charles L. Lawson. *Solving least squares problems*. Philadelphia, 1995.

[29] R. Keren, Y.Y. Zeevi, and D. Chazan. Multiresolution time-frequency analysis of polyphonic music. In *Time-Frequency and Time-Scale Analysis, 1998. Proceedings of the IEEE-SP International Symposium on*, pages 565 –568, oct 1998.

[30] A. Klapuri. Multiple fundamental frequency estimation by summing harmonic amplitudes. In *Proc. ISMIR*, volume 6, pages 216–221, 2006.

[31] Anssi Klapuri and Manuel Davy, editors. *Signal Processing Methods for Music Transcription*. Springer, New York, 2006.

[32] A. Kobzantsev, D. Chazan, and Y. Zeevi. Automatic transcription of piano polyphonic music. In *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, pages 414 – 418, sept. 2005.

[33] Weilun Lao, Ek Tsoon Tan, and A.H. Kam. Computationally inexpensive and effective scheme for automatic transcription of polyphonic music. In *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on*, volume 3, pages 1775 – 1778 Vol.3, june 2004.

[34] Hong-Ru Lee and J.-S.R. Jang. i-ring: a system for humming transcription and chord generation. In *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on*, volume 2, pages 1031 –1034 Vol.2, june 2004.

[35] A. Mesaros, T. Virtanen, and A. Klapuri. Singer identification in polyphonic music using vocal separation and pattern recognition methods. In *Proc. ISMIR*, pages 375–378, 2007.

[36] K. Miyamoto, H. Kameoka, H. Takeda, T. Nishimoto, and S. Sagayama. Probabilistic approach to automatic music transcription from audio signals. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 2, pages II–697 –II–700, april 2007.

[37] G. Monti and M. Sandler. Monophonic transcription with autocorrelation. In *Proceedings of the COST G-6 Conference on digital audio effects (DAFX-00), Verona, Italy*, 2000.

[38] J.A. Moorer. On the segmentation and analysis of continuous musical sound by digital computer. 1975.

[39] M. Muller, F. Kurth, and M. Clausen. Chroma-based statistical audio features for audio matching. In *Applications of Signal Processing to Audio and Acoustics, 2005. IEEE Workshop on*, pages 275–278. IEEE, 2005.

[40] G.J. Mysore and P. Smaragdis. Relative pitch estimation of multiple instruments. In *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*, pages 313 –316, april 2009.

[41] Bernhard Niedermayer. Non-negative matrix division for the automatic transcription of polyphonic music. In *Proceedings of the ISMIR*, 2008.

[42] Dan ning Jiang, M. Picheny, and Yong Qin. Voice-melody transcription under a speech recognition framework. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–617 –IV–620, april 2007.

[43] Paul D. O'Grady and Scott T. Rickard. Automatic hexaphonic guitar transcription using non-negative constraints. In *Signals and Systems Conference (ISSC 2009), IET Irish*, pages 1 –6, june 2009.

[44] L. Oudre, Y. Grenier, and C. Fevotte. Chord recognition using measures of fit, chord templates and filtering methods. In *Applications of Signal Processing to Audio and Acoustics, 2009. WASPAA '09. IEEE Workshop on*, pages 9 –12, oct. 2009.

[45] L. Oudre, Y. Grenier, and C. Fevotte. Chord recognition by fitting rescaled chroma vectors to chord templates. *Audio, Speech, and Language Processing, IEEE Transactions on*, 19(7):2222 –2233, sept. 2011.

[46] Antonio Pertusa and Jos M. Iesta. Polyphonic monotimbral music transcription using dynamic networks. *Pattern Recognition Letters*, 26(12):1809 – 1818, 2005. Artificial Neural Networks in Pattern Recognition.

[47] S. Phon-Amnuaisuk. Transcribing bach chorales using non-negative matrix factorisation. In *Audio Language and Image Processing (ICALIP), 2010 International Conference on*, pages 688 –693, nov. 2010.

[48] Martin Piszczalski and Bernard A. Galler. Automatic music transcription. *Computer Music Journal*, 4(1):24–31, 1977.

[49] M. Privosnik and M. Marolt. A system for automatic transcription of music based on multiple-agents architecture. In *Electrotechnical Conference, 1998. MELECON 98., 9th Mediterranean*, volume 1, pages 169 –172 vol.1, may 1998.

[50] L. Rabiner. On the use of autocorrelation analysis for pitch detection. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 25(1):24–33, 1977.

[51] G. Reis, N. Fonseca, F. Fernandez, and A. Ferreira. A genetic algorithm approach with harmonic structure evolution for polyphonic music transcription. In *Signal Processing and Information Technology, 2008. ISSPIT 2008. IEEE International Symposium on*, pages 491 –496, dec. 2008.

[52] M. Ryynanen and A. Klapuri. Automatic bass line transcription from streaming polyphonic audio. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–1437 –IV–1440, april 2007.

[53] P. Smaragdis and J.C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on.*, pages 177 – 180, oct. 2003.

[54] P. Smaragdis and B. Raj. Shift-invariant probabilistic latent component analysis. *Journal of Machine Learning Research*, 2007.

[55] P. Smaragdis, B. Raj, and M. Shashanka. Sparse and shift-invariant feature extraction from non-negative data. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 2069 –2072, 31 2008-april 4 2008.

[56] S. Sophea and S. Phon-Amnuaisuk. Determining a suitable desired factors for nonnegative matrix factorization in polyphonic music transcription. In *Information Technology Convergence, 2007. ISITC 2007. International Symposium on*, pages 166 –170, nov. 2007.

[57] E. Vincent, N. Berlin, and R. Badeau. Harmonic and inharmonic nonnegative matrix factorization for polyphonic pitch transcription. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 109 –112, 31 2008-april 4 2008.

Appendix A

Additional Information

A.1 Noise model

as mentioned earlier, we use noise model to reduce the impact of background music and noisy environment. The experiments show that the transcription results are improved by using a noise model. Table A.1 shows the results when no noise model is used, and Table A.2 shows the results when the noise is used running NNLSQ. Table A.3 shows the results when no noise model is used, and Table A.4. as you can see the results are consistently improved.

Table A.1: Transcription results for NNLSQ no noise model

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.5517	0.6667	0.6038	0.5333	0.6609	0.5902
2	0.6207	0.6667	0.6429	0.5925	0.6880	0.6367
3	0.7241	0.4667	0.5676	0.6819	0.4710	0.5572
4	0.5862	0.3778	0.4595	0.5335	0.4216	0.4710
5	0.8750	0.6667	0.7568	0.3432	0.6585	0.4513
6	0.8750	0.6087	0.7179	0.3458	0.5939	0.4371
7	0.3125	0.1786	0.2273	0.1254	0.1503	0.1367
8	0.9375	0.6250	0.7500	0.3610	0.6192	0.4561
9	0.5517	0.6275	0.5872	0.5067	0.6039	0.5510
10	0.2931	0.2787	0.2857	0.5533	0.2847	0.3760
11	0.2586	0.2308	0.2439	0.4891	0.2479	0.3290
12	0.4828	0.0707	0.1233	0.4552	0.0953	0.1576
Mean	0.5891	0.4554	0.4972	0.4601	0.4579	0.4292

Table A.2: Transcription results for NNLSQ using noise model

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.5862	0.8500	0.6939	0.5651	0.8829	0.6891
2	0.6552	0.8636	0.7451	0.6247	0.8573	0.7227
3	0.6552	0.6129	0.6333	0.6180	0.6733	0.6445
4	0.6207	0.5294	0.5714	0.5909	0.6247	0.6073
5	0.9375	0.6818	0.7895	0.3610	0.6916	0.4744
6	0.8750	0.6667	0.7568	0.3297	0.6336	0.4337
7	0.2500	0.1600	0.1951	0.1068	0.1520	0.1254
8	0.9375	0.6522	0.7692	0.3610	0.7332	0.4838
9	0.5517	0.8649	0.6737	0.5176	0.8394	0.6404
10	0.3276	0.3333	0.3304	0.6427	0.3568	0.4588
11	0.2241	0.2549	0.2385	0.4169	0.2848	0.3384
12	0.4828	0.1094	0.1783	0.5030	0.1544	0.2363
Mean	0.5920	0.5483	0.5479	0.4698	0.5737	0.4879

Table A.3: Transcription results for PLCA no noise model

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.9655	0.4444	0.6087	0.8641	0.4112	0.5573
2	0.6897	0.1818	0.2878	0.6541	0.1844	0.2877
3	0.5517	0.2581	0.3516	0.5369	0.2750	0.3637
4	0.6207	0.1651	0.2609	0.5936	0.1838	0.2807
5	0.6250	0.0625	0.1136	0.2119	0.0481	0.0785
6	0.6875	0.0775	0.1392	0.2432	0.0615	0.0982
7	0.3750	0.0375	0.0682	0.1661	0.0375	0.0612
8	0.6875	0.0833	0.1486	0.2432	0.0656	0.1033
9	0.5862	0.2595	0.3598	0.6590	0.2964	0.4089
10	0.3276	0.1776	0.2303	0.6126	0.1874	0.2871
11	0.2414	0.1138	0.1547	0.4186	0.1120	0.1767
12	0.5172	0.0679	0.1200	0.4413	0.0719	0.1237

Table A.4: Transcription results for PLCA using noise model

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.9655	0.5385	0.6914	0.8622	0.4936	0.6278
2	0.5862	0.1771	0.2720	0.5345	0.1761	0.2649
3	0.5862	0.3091	0.4048	0.5314	0.3010	0.3843
4	0.5517	0.1702	0.2602	0.5405	0.1904	0.2816
5	0.6250	0.0621	0.1130	0.2119	0.0478	0.0781
6	0.6250	0.0813	0.1439	0.2119	0.0586	0.0918
7	0.4375	0.0543	0.0966	0.1653	0.0428	0.0680
8	0.6250	0.0909	0.1587	0.2127	0.0660	0.1008
9	0.6207	0.3103	0.4138	0.6943	0.3508	0.4661
10	0.2931	0.1753	0.2194	0.5871	0.1939	0.2915
11	0.2414	0.1120	0.1530	0.4404	0.1168	0.1846
12	0.5172	0.0661	0.1172	0.4648	0.0745	0.1285

A.2 SVM vs Thresholding

Following tables show the results of expreiments using SVM and thresholding. The result of onset detection are slightly better when SVM is used for PLCA, but for NNLSQ thresholding outperforms SVM. PLCA-svm NNLSQ-thresh NNLSQ-svm

Table A.5: Transcription PLCA for threshold

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.9655	0.5091	0.6667	0.8622	0.4661	0.6051
2	0.6897	0.1942	0.3030	0.6440	0.1929	0.2969
3	0.6207	0.3158	0.4186	0.5779	0.3217	0.4133
4	0.6207	0.1698	0.2667	0.5709	0.1860	0.2806
5	0.6250	0.0870	0.1527	0.1975	0.0607	0.0928
6	0.6875	0.0775	0.1392	0.2280	0.0568	0.0909
7	0.4375	0.0429	0.0782	0.2178	0.0469	0.0772
8	0.6875	0.0840	0.1497	0.2280	0.0599	0.0949
9	0.5862	0.3036	0.4000	0.6371	0.3337	0.4380
10	0.3621	0.1909	0.2500	0.6636	0.1921	0.2980
11	0.2241	0.1083	0.1461	0.3890	0.1076	0.1686
12	0.5172	0.0615	0.1099	0.4708	0.0697	0.1214

Table A.6: Transcription PLCA for svm

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.9310	0.4426	0.6000	0.8604	0.4188	0.5634
2	0.9310	0.3803	0.5400	0.8519	0.3633	0.5094
3	0.4483	0.2500	0.3210	0.5014	0.2944	0.3710
4	0.5517	0.2963	0.3855	0.6127	0.3625	0.4555
5	0.5625	0.2571	0.3529	0.2466	0.2313	0.2387
6	0.5625	0.4286	0.4865	0.2364	0.3695	0.2884
7	0.3750	0.2727	0.3158	0.1847	0.2763	0.2214
8	0.6875	0.2200	0.3333	0.2390	0.1570	0.1895
9	0.3793	0.3729	0.3761	0.3995	0.3956	0.3975
10	0.3276	0.2209	0.2639	0.6035	0.2144	0.3164
11	0.1552	0.0841	0.1091	0.2715	0.0812	0.1250
12	0.5862	0.1491	0.2378	0.5683	0.1595	0.2491

Table A.7: Transcription NNLSQ for threshold

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.5517	0.8421	0.6667	0.5314	0.8764	0.6616
2	0.5862	0.8500	0.6939	0.5593	0.8876	0.6862
3	0.6207	0.5806	0.6000	0.5871	0.6619	0.6222
4	0.5862	0.4722	0.5231	0.5283	0.4848	0.5056
5	0.9375	0.7895	0.8571	0.3610	0.7370	0.4846
6	0.8750	0.7000	0.7778	0.3297	0.6336	0.4337
7	0.3750	0.2308	0.2857	0.1534	0.2018	0.1743
8	0.9375	0.6818	0.7895	0.3610	0.7345	0.4841
9	0.5345	0.8611	0.6596	0.5033	0.8396	0.6294
10	0.2759	0.3019	0.2883	0.5205	0.3179	0.3947
11	0.2241	0.2453	0.2342	0.4282	0.2842	0.3417
12	0.5172	0.1034	0.1724	0.4743	0.1267	0.2000

Table A.8: Transcription PLCA for threshold

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.9310	0.6000	0.7297	0.8885	0.5866	0.7067
2	0.7931	0.3333	0.4694	0.7029	0.3082	0.4285
3	0.7931	0.4510	0.5750	0.7211	0.4318	0.5401
4	0.6207	0.2308	0.3364	0.5387	0.2208	0.3133
5	0.5625	0.2500	0.3462	0.1780	0.1624	0.1698
6	0.5625	0.2045	0.3000	0.1780	0.1328	0.1521
7	0.5625	0.1406	0.2250	0.1754	0.0899	0.1189
8	0.5625	0.1731	0.2647	0.1780	0.1127	0.1380
9	0.5000	0.6170	0.5524	0.4605	0.5735	0.5108
10	0.3966	0.3485	0.3710	0.7475	0.3458	0.4729
11	0.2241	0.2321	0.2281	0.4204	0.2403	0.3058
12	0.7241	0.2234	0.3415	0.6493	0.2208	0.3295

A.3 CQT vs DFT

This section shows the transcription results where CQT is used instea

Table A.9: Transcription using PLCA and CQT

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.3793	0.1528	0.2178	0.6267	0.2616	0.3691
9	0.2414	0.1321	0.1707	0.5285	0.2991	0.3820

Table A.10: Transcription using NNLSQ and CQT

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	0.1724	0.0505	0.0781	0.2309	0.4129	0.3193
9	0.1207	0.0365	0.0560	0.2313	0.3972	0.3021

A.3.1 Perfect activation matrix

Table A.11: Transcription results for perfect activation matrix, PLCA,svm, using noise

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	1.0000	1.0000	1.0000	0.8866	0.9079	0.8971
2	0.9310	0.9643	0.9474	0.8868	0.9602	0.9220
3	0.9655	1.0000	0.9825	0.8870	0.9682	0.9258
4	0.9310	0.9310	0.9310	0.8738	0.9635	0.9165
5	0.8125	1.0000	0.8966	0.2966	0.7625	0.4271
6	0.8750	1.0000	0.9333	0.3271	0.7798	0.4609
7	0.6875	0.9167	0.7857	0.2771	0.7823	0.4093
8	0.8750	1.0000	0.9333	0.3288	0.7807	0.4627
9	0.8966	0.9123	0.9043	0.8914	0.9145	0.9028
10	0.3793	0.8462	0.5238	0.7639	0.8991	0.8260
11	0.4828	0.9655	0.6437	0.8738	0.9635	0.9165
12	0.9655	0.9333	0.9492	0.8695	0.9267	0.8972

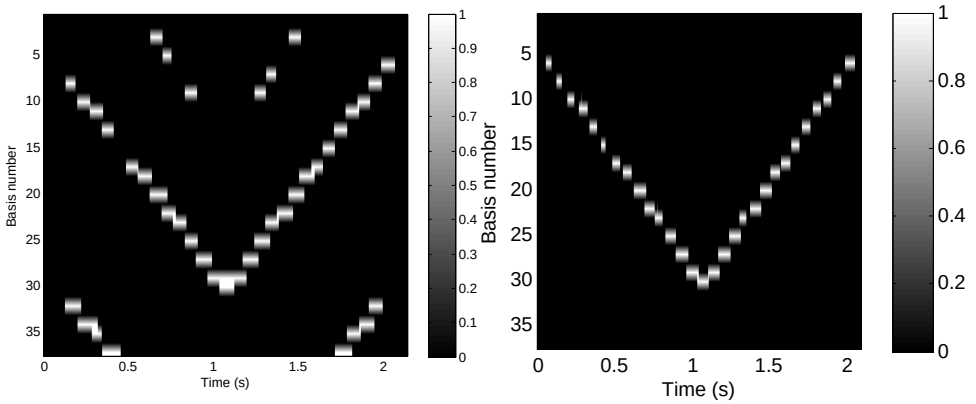
A.4 Multi-f0 estimation

Table A.12: Transcription results for NNLSQ using noise model,svm, perfect

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE
1	1.0000	0.9063	0.9508	0.8547	0.8157	0.8348
2	1.0000	0.8788	0.9355	0.8353	0.7889	0.8114
3	0.8966	0.5306	0.6667	0.7675	0.4790	0.5898
4	0.9655	1.0000	0.9825	0.7615	0.8689	0.8117
5	0.4375	0.6364	0.5185	0.2119	0.6313	0.3173
6	0.5000	0.8000	0.6154	0.1814	0.5944	0.2779
7	0.5000	0.7273	0.5926	0.1873	0.5581	0.2805
8	0.3750	0.4286	0.4000	0.1237	0.2903	0.1735
9	0.6724	0.8125	0.7358	0.6010	0.7320	0.6600
10	0.4655	0.6923	0.5567	0.7958	0.6236	0.6992
11	0.4655	0.7714	0.5806	0.7433	0.6778	0.7090
12	0.9655	1.0000	0.9825	0.7607	0.8671	0.8104

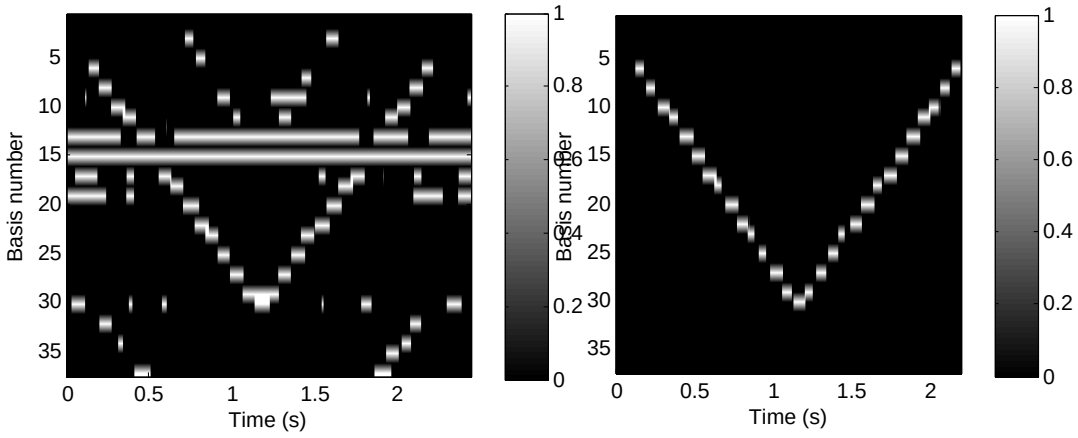
Table A.13: Transcription results using the multi-f0 estimation method

Track	Precision-SE	Recall-SE	F-SE	Precision-FE	Recall-FE	F-FE	
1	0.6552	0.1638	0.2621	0.6120	0.1569	0.2497	
2	0.2759	0.0889	0.1345	0.3431	0.1154	0.1728	0.0946
3	0.2414	0.1273	0.1667	0.3191	0.1773	0.2279	0.1286
4	0.0690	0.0270	0.0388	0.1393	0.0602	0.0840	0.0439
5	0.0625	0.0075	0.0133	0.0186	0.0046	0.0073	0.0037
6	0.0625	0.0084	0.0148	0.0153	0.0042	0.0066	0.0033
7	0	0	0	0.0169	0.0072	0.0101	0.0051
8	0.1875	0.0203	0.0366	0.0432	0.0096	0.0157	0.0079
9	0.3621	0.2000	0.2577	0.4219	0.2350	0.3018	0.1777
10	0.0517	0.0385	0.0441	0.1550	0.0607	0.0872	0.0456
11	0.0517	0.0417	0.0462	0.1419	0.0630	0.0872	0.0456
12	0.1034	0.0229	0.0375	0.1584	0.0387	0.0621	0.0321



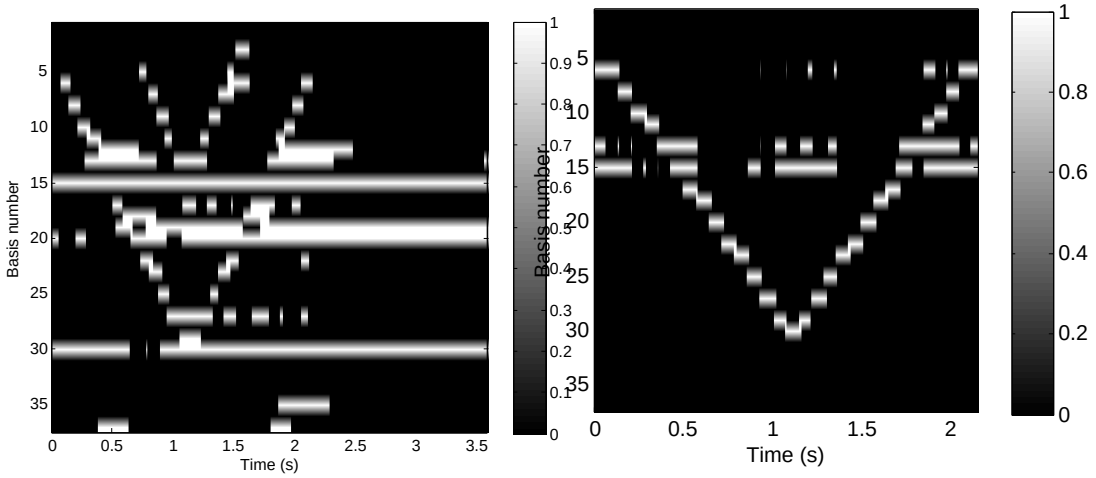
(a) SVM (b) Upper bound

Figure A.1: Track 1: SVM vs perfect



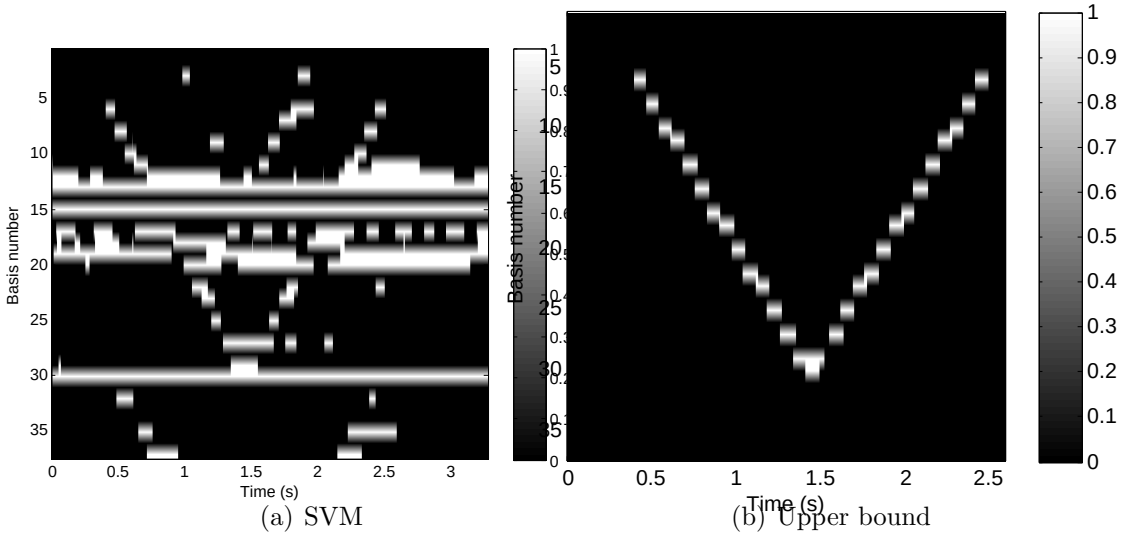
(a) SVM (b) Upper bound

Figure A.2: Track 2: SVM vs perfect



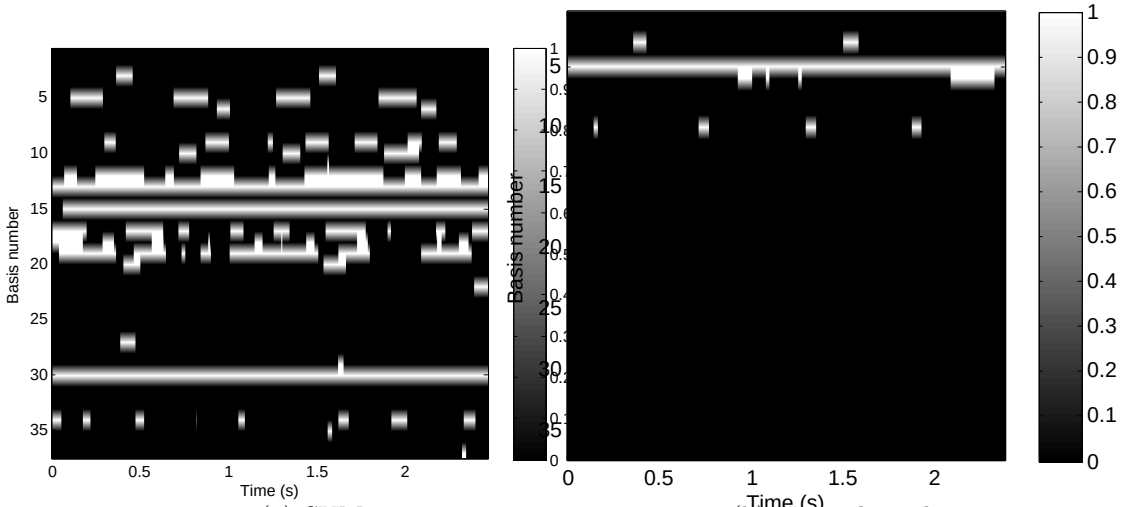
(a) SVM (b) Upper bound

Figure A.3: Track 3: SVM vs perfect



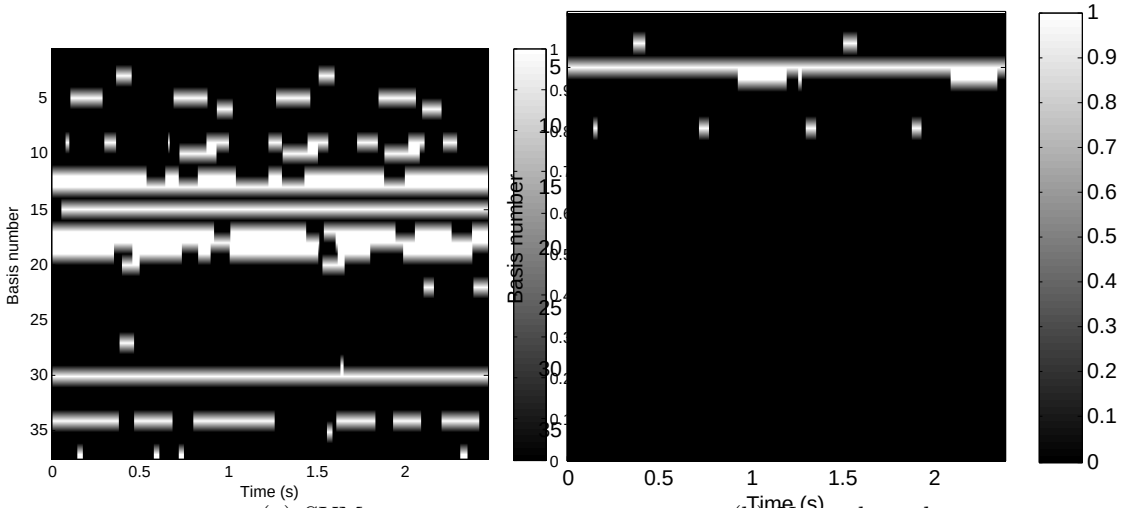
(a) SVM (b) Upper bound

Figure A.4: Track 4: SVM vs perfect



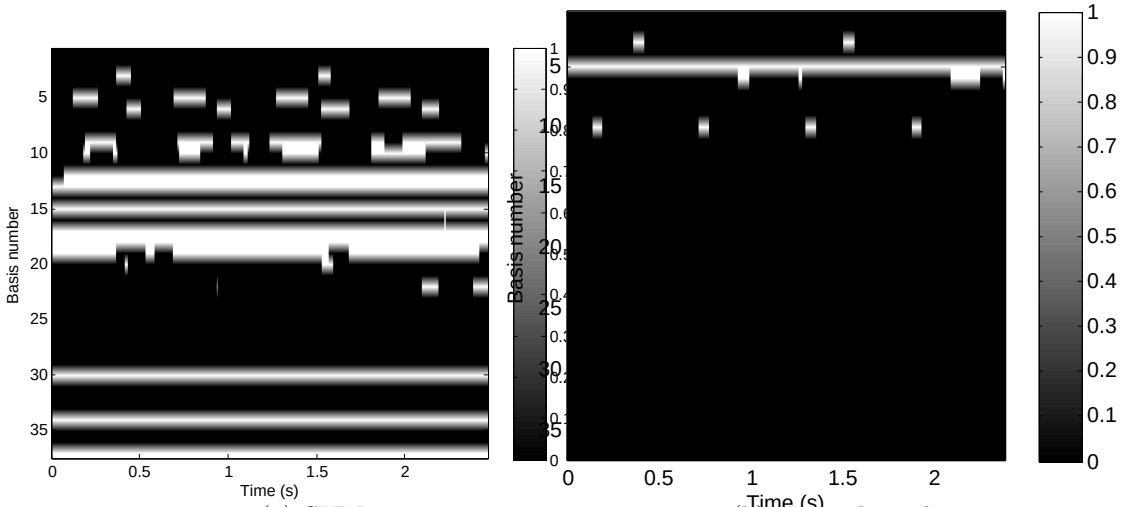
(a) SVM (b) Upper bound

Figure A.5: Track 5: SVM vs perfect



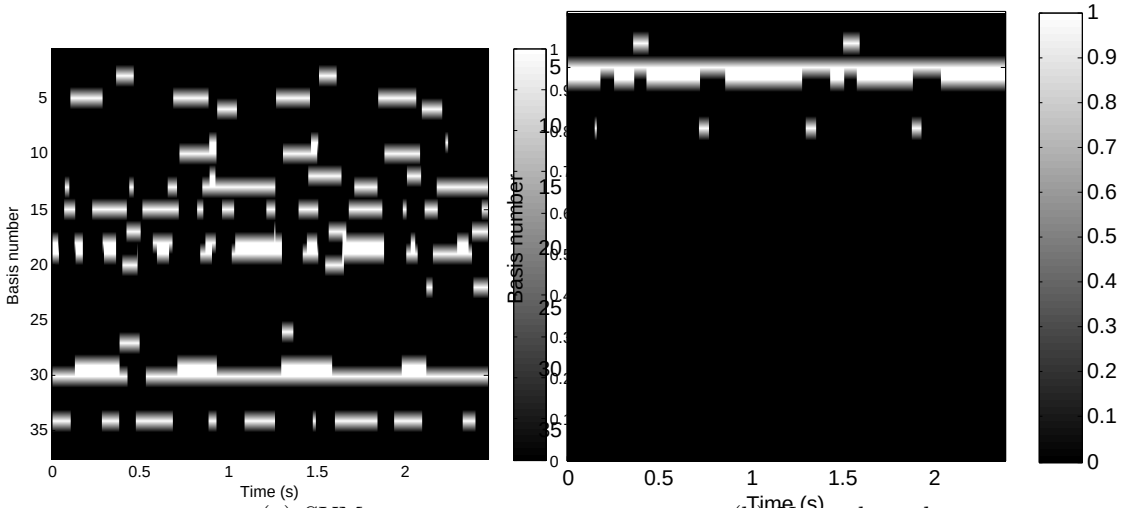
(a) SVM (b) Upper bound

Figure A.6: Track 6: SVM vs perfect



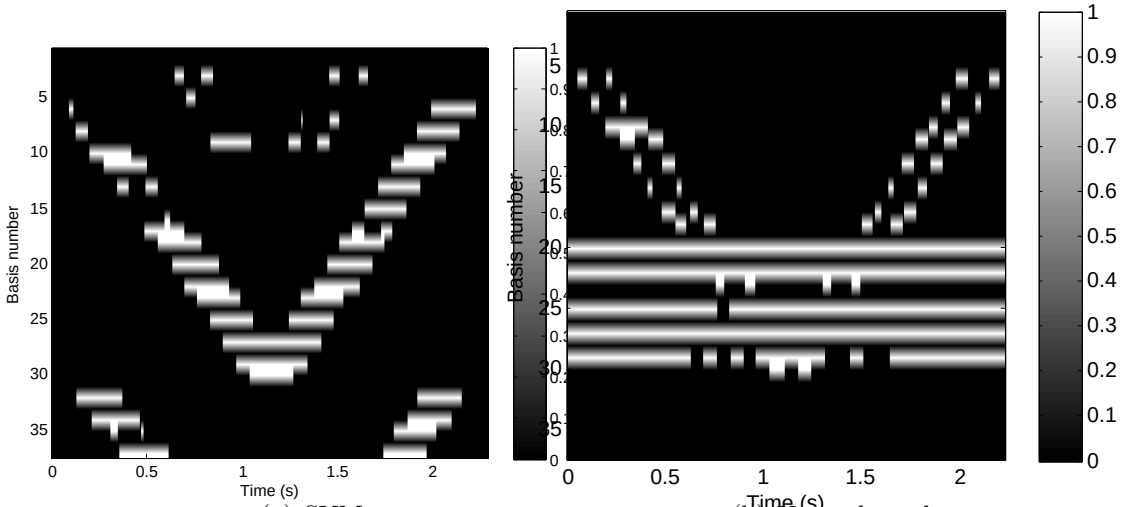
(a) SVM (b) Upper bound

Figure A.7: Track 7: SVM vs perfect



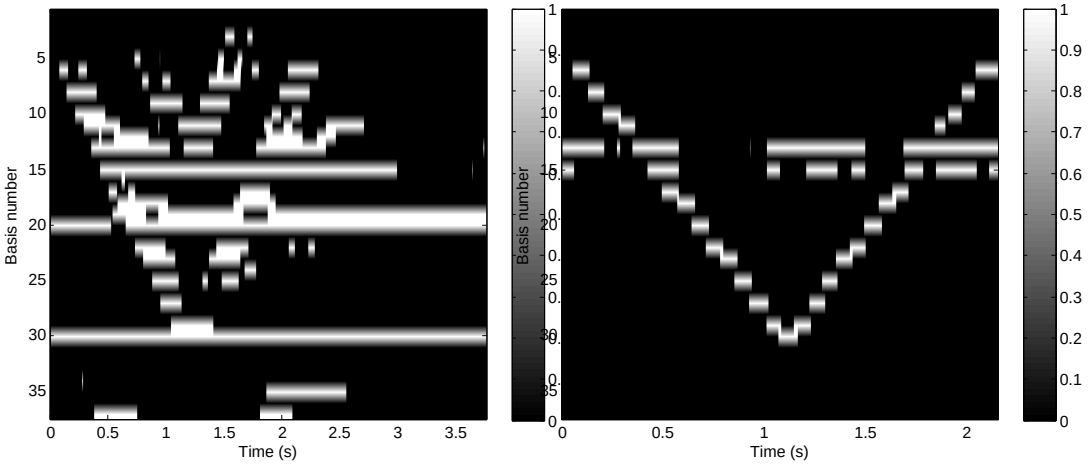
(a) SVM (b) Upper bound

Figure A.8: Track 8: SVM vs perfect



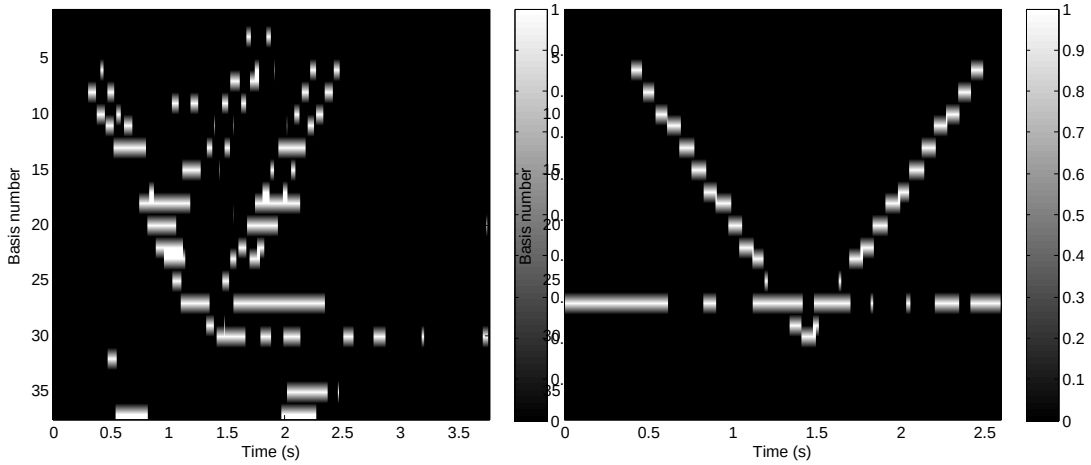
(a) SVM (b) Upper bound

Figure A.9: Track 9: SVM vs perfect



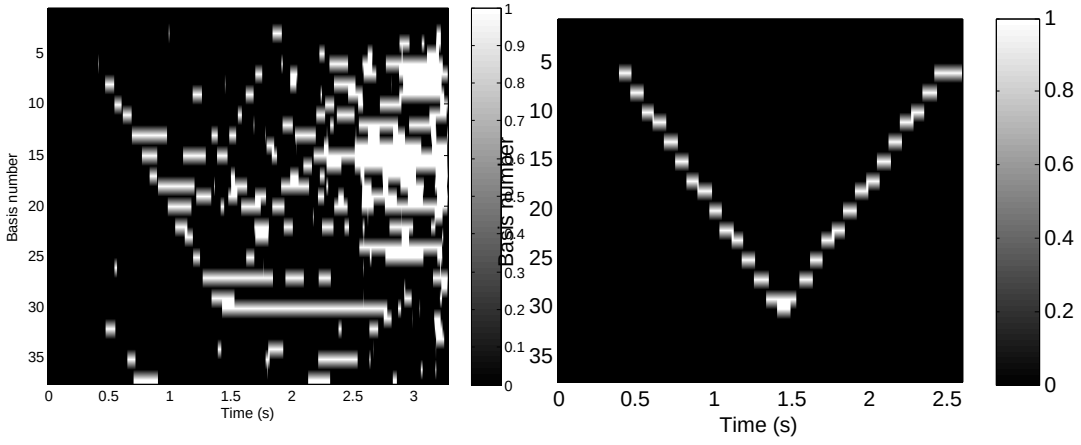
(a) SVM (b) Upper bound

Figure A.10: Track 10: SVM vs perfect



(a) SVM (b) Upper bound

Figure A.11: Track 11: SVM vs perfect



(a) SVM (b) Upper bound

Figure A.12: Track 12: SVM vs perfect

A.5 PLCA tests

A.5.1 Run settings

minPitch = 1; maxPitch = 37; separationMethod = 'PLCA'; event detection method = 'threshhold'; Thresholding parameters: minValue = 0.6; minDelta = 0.4; decayN = 0.99;extraTh = 1.3; minTh = 0.2; energyRatio = 0.05; spectrogram method = 'dft';

In this study, SVM machine learning outperformed a hand-tuned detection system on our test data, indicating that our semisupervised learning approach is successful.(feature extraction) You07polyphonicmusic[change]

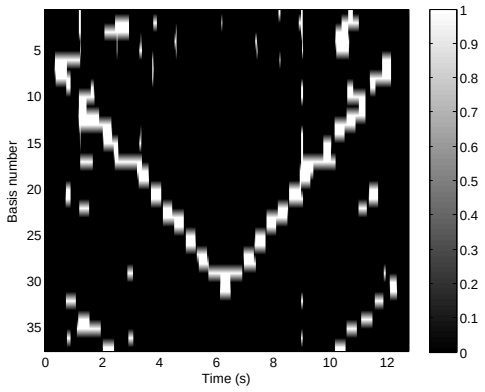


Figure A.13: Track 1: Multi-f0 estimation

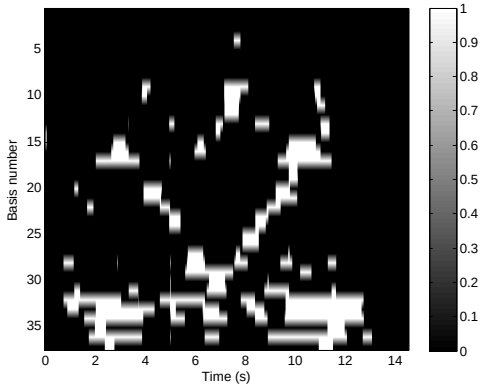


Figure A.14: Track 2: Multi-f0 estimation

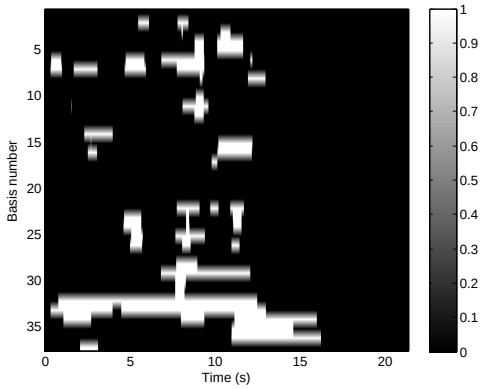


Figure A.15: Track 3: Multi-f0 estimation

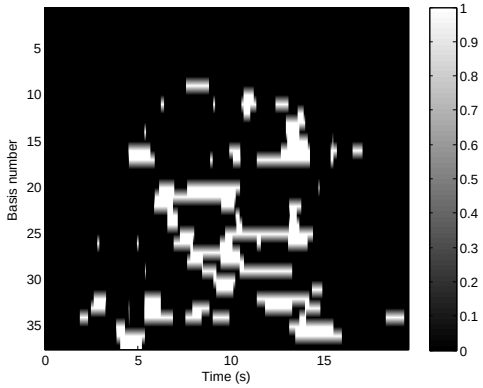


Figure A.16: Track 4: Multi-f0 estimation

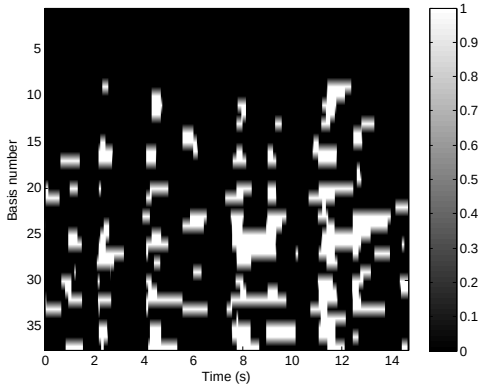


Figure A.17: Track 5: Multi-f0 estimation

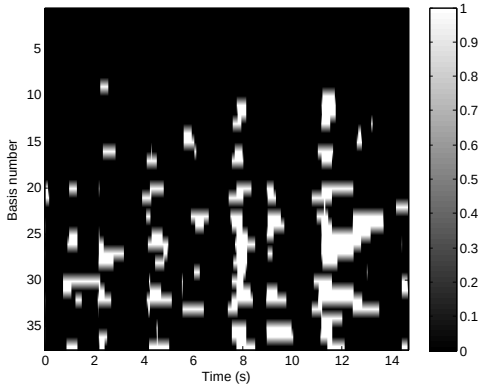


Figure A.18: Track 6: Multi-f0 estimation

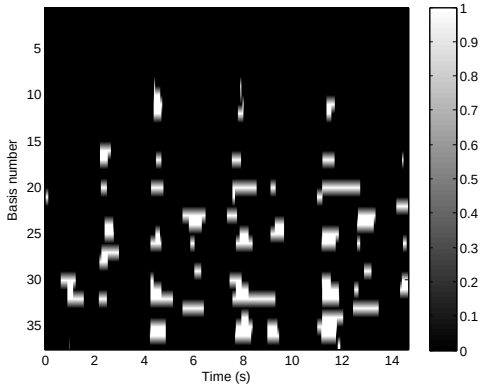


Figure A.19: Track 7: Multi-f0 estimation

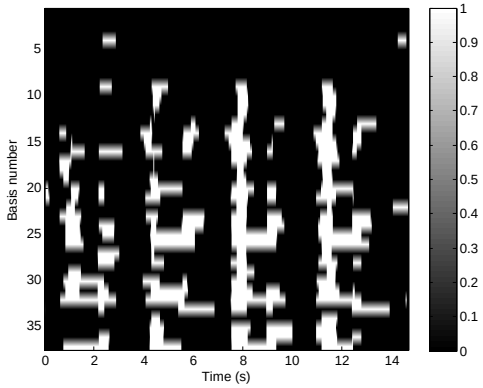


Figure A.20: Track 8: Multi-f0 estimation

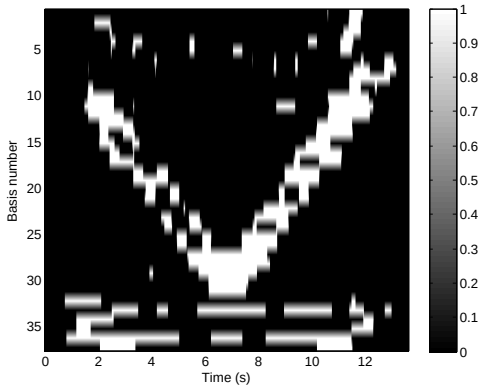


Figure A.21: Track 9: Multi-f0 estimation

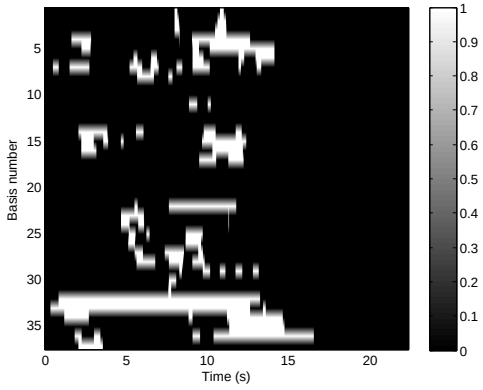


Figure A.22: Track 10: Multi-f0 estimation

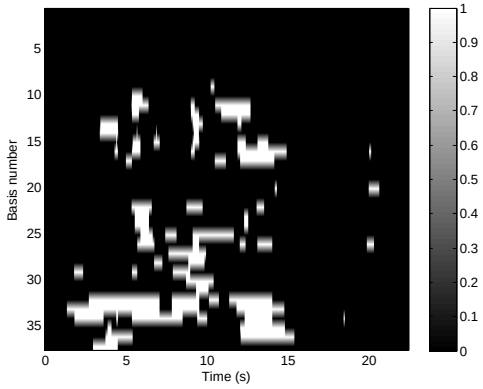


Figure A.23: Track 11: Multi-f0 estimation

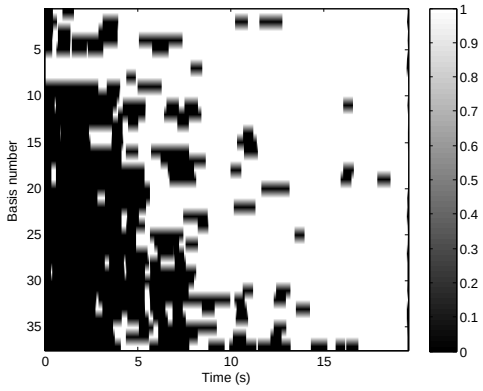


Figure A.24: Track 12: Multi-f0 estimation