TECHNISCHE
UNIVERSITÄT
WIEN

VIENNA
UNIVERSITY OF
TECHNOLOGY

MASTERARBEIT

Musical Instrument Separation

ausgeführt
am Institut für Softwaretechnik und Interaktive Systeme (E188)
der Technischen Universität Wien

unter der Anleitung von
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Andreas Rauber
und der Betreuung von
Dipl.-Ing. Thomas Lidy

durch
Andrei Grecu,
Matrikelnummer 0125662,
Karmarschgasse 18A/2/6,
1100 Wien, Österreich

Wien, am 15.10.2007

Kurzfassung

Das menschliche Gehirn kann das Problem Instrumente innerhalb eines Musikstückes zu trennen relativ leicht lösen. Für Computer jedoch ist das noch immer ein schwieriges Problem zu dem noch keine zufrieden stellende Lösung gefunden wurde. Unser Ziel ist es deshalb Möglichkeiten zu finden, Musikstücke in Formaten wie z.B. mp3 zu analysieren, die Instrumente mittels verschiedenen Stereomerkmalen und gewissen Annahmen über die Struktur von Musik zu separieren und schließlich die Resultate in mehreren Tonspuren zu speichern.

Unser Beitrag besteht aus drei Algorithmen. Der schablonenbasierte Algorithmus nimmt an, dass die Töne der Instrumente jeweils in ihrer Anzahl über das Musikstück limitiert sind und deshalb wiederholt werden müssen um eine gewisse Klangvielfalt zu erreichen. Diese Redundanz kann ausgenutzt werden um Töne mittels Schablonen zu modellieren. Es wird dabei versucht das Musikstück mit so wenigen Schablonen und Anschlägen wie möglich zu rekonstruieren. Schließlich müssen die Schablonen zu Instrumenten zusammengefasst werden. Als eine Verbesserung dient der zweite Ansatz, wobei wir annehmen dass der Anschlagsvektor nicht unbedingt unter Zuhilfenahme von Relevanzheuristiken gefunden werden muss, sondern dass die Möglichkeit besteht ihn sich iterativ selbst organisieren zu lassen.

Der dritte Ansatz gehört zum Gebiet des Blind Source Separation, wobei er auf Stereomerkmalen im Frequenzspektrum arbeitet wodurch ein leicht durch Histogramme visualisierbarer Merkmalraum entsteht. Unter der Annahme dass Instrumente sich während der Aufführung nicht bewegen, sollte der Merkmalraum Häufungen aufweisen. Durch deren automatische Identifizierung können darunter fallende Frequenzen separiert werden, wodurch alles getrennt werden kann was an der entsprechenden räumlichen Position, liegt. Dieser Ansatz ist jedoch nicht neu in der Literatur. Unsere Verbesserung hierbei ist Frequenzen durch Farben darzustellen, im Gegensatz zu den bisherigen s/w Histogrammen. Zusätzlich clustern wir das Histogramm automatisch durch einen Netzwerk mit radialen Basisfunktionen (RBFN).

Die Evaluierungsergebnisse von zwei von diesen Algorithmen, unter Zuhilfenahme verschiedener Korpora, zeigen erfreuliche Ergebnisse. Ihre Trennschärfe ist in etwa vier Mal höher im Vergleich zur Baseline. Daraus resultierte die Idee für zukünftige Entwicklungen die Konzepte des ersten und dritten Ansatzes in einen neuen Algorithmus zu vereinen.

Abstract

The problem of separating instruments in a musical piece can be easily solved by the human brain. For computers on the other hand this task is still difficult and no general solution exists at the time of this thesis. Our goal is therefore to find some solutions using the limited power of today's computers at its best to analyze a musical performance given in some common format like mp3, separate the instruments using two different stereo cues together with some assumptions about the structure of music and finally save the result into several tracks.

We approached this goal by contributing three separation algorithms where two of them make use of some different properties of music. The first one being a template matching algorithm assumes that instrument tones are only limited in number throughout a song and therefore have to be repeated in order to create diversity. This kind of redundancy can be used by modeling the tones with templates and trying to reconstruct the musical piece with as few templates and onsets as possible, which in turn should lead to a solution where each template matches a tone. The second algorithm is an improvement over the first where we assume that the onset vector does not need to be found using relevance heuristics, but can be let to self-organize iteratively which is why we called it the iterative template matching algorithm (ITM).

The third approach is a blind source separation algorithm using stereo cues in the frequency domain which form an easily visualizable feature space. Assuming that instruments do not move during the performance the resulting histogram visualization will show clusterings. By identifying these clusters we can separate the frequencies falling into them thus separating whatever is at the spatial location corresponding to the cluster region in the histogram. This approach is not new in literature, so our improvements are to use the frequency to generate colours thus adding information to the b/w histograms used before, and to cluster the histogram automatically using a radial basis function network (RBFN).

The evaluation results for two of these algorithms using different corpora look very promising. Their separation performance is about four times higher than a simple baseline used for comparison. As a consequence we then present as an issue for future work, the idea of unifying the concepts of our first and third approach to create a new algorithm.

to my mother

Acknowledgements

I want to thank my father for giving me the opportunity to study at the university, despite the prevailing circumstances.

I want to thank Andreas Rauber for helping me finish this thesis in time.

Finally, thanks go to the people who taught me what is important in life.

Contents

1 Introduction 1

1.1 Motivation 1

1.2 Problem Formulation 1

1.3 Input Restrictions 3

1.4 Overview 3

1.5 Notation and Conventions 4

2 Related Work 5

2.1 Introduction 5

2.2 Blind Source Separation 6

2.3 Harmonic and Sinusoidal Modelling 7

2.4 Model and Feature-Based Approaches 9

2.5 Template-Based Approaches 10

2.6 Speech Separation 12

2.7 Benchmarking in Literature 14

2.8 Summary and Conclusions 20

3 Direct Template Matching 21

3.1 Motivation 21

3.2 Problem Reformulation 22

3.3 Algorithm Overview 25

3.4 The Onset Vector 25
3.5 Main Algorithm 27
3.6 Tone Search 28
3.7 Tone Learning 36
3.8 Fine Tuning 46
3.9 Final Algorithm 51
3.10 Summary and Future Work 54

4 Iterative Template Matching **57**

4.1 Introduction 57
4.2 Overview 57
4.3 Initialization 58
4.4 Main Algorithm 58
4.5 Learning Step 59
4.6 Fine Tuning 66
4.7 Final Algorithm 72
4.8 Summary and Future Work 75

5 Blind Source Separation Approach **79**

5.1 Motivation 79
5.2 Overview 80
5.3 Problem Reformulation 80
5.4 Magnitude-Shift-Frequency Histogram 81
5.5 Algorithm 85
5.6 Fine Tuning 92
5.7 Final Algorithm 96
5.8 Summary and Future Work 98

6 Implementation Details **101**

6.1 Overview . . . 101
6.2 Libraries . . . 101
6.3 Code Structure . . . 103
6.4 Performance Enhancements . . . 103
6.5 Summary and Future Work . . . 107

7 Benchmarking and Evaluation **109**
7.1 Overview . . . 109
7.2 Corpora . . . 110
7.3 Methods and Algorithms . . . 113
7.4 Results . . . 116
7.5 Discussion . . . 120
7.6 Summary and Conclusions . . . 122

8 Conclusions and Future Work **125**

A Corpora **131**

Bibliography **135**

List of Figures

2.1 Changing directional receiving patterns of microphones 16

3.1 Concept of music synthesis using templates 24
3.2 Example of the Lanczos resampling filter for upsampling and phase-shifting . . . 50

List of Tables

2.1 Summary of evaluation methods 19

7.1 Objective evaluation results for the BASS-dB and IS corpus 118

7.2 Summary of the objective evaluation results 118

7.3 Subjective evaluation results for the BASS-dB and IS corpus 120

7.4 Subjective evaluation results for the IS and RWC corpus and the ISMIRgenre collection 121

7.5 Summary of the subjective evaluation results 122

A.1 List of mappings between module file channels and reference tracks 131

A.2 Details of the songs from the BASS-dB and IS corpus 132

A.3 Details of the songs from the ISMIRgenre collection and the RWC corpus 133

Chapter 1

Introduction

1.1 Motivation

Human listeners have the ability to distinguish between a fair amount of instruments in a musical setting. Their performance in real environments is better by a large margin than computational auditory scene analysis (CASA) systems which are still a hot research topic. Research on the human auditory system has revealed a lot of information of the low level processing in the inner ear which led to many algorithms using these findings in order to improve accuracy in several tasks like audio stream segregation, voice and musical instrument separation, pitch estimation and music transcription.

Unfortunately a lot of algorithms published in the domain of voice and musical instrument separation are being evaluated using artificial sound mixtures. This is a problem as in this artificial environment they can even outperform human listeners but on the other hand when being fed with real world mixtures their performance degrades rapidly. Yilmaz and Rickard for example could separate six anechoic mixed voices from two mixtures with acceptable signal to interference ratio (SIR) in [40] which due to the lack of spatial cues would be an impossible task for the human auditory system. However when using two echoic mixtures of only three voices the performance degraded much below the six source result.

Therefore the goal of this thesis is to study how existing algorithms behave in regard to real world recordings and artificial mixtures, to improve them if possible and to design new ones.

1.2 Problem Formulation

We consider music being a mixture of instrument sounds, reverberation and noise. Given a musical piece our aim is now to segregate the instrument sounds possibly suppressing reverberation and noise. Depending on the approaches for solving this problem we can refine the former definition on two levels of abstraction.

1. At a high abstraction level we can say that the instruments are well separated if the resulting pitch and timbre match their original counterpart and the music recomposed from the separated instruments still sounds “right” to the human listener.
2. At the low level we demand that the digitized waveform of the instruments matches the original as well as possible.

Although both formulations might look like being equal their difference lies in weighting errors differently. While for the first one it is acceptable to truncate notes at the end or slip them entirely if they are not important to the perception of the musical piece, it would cause high errors in matching the digitized waveform as in the second formulation. On the other hand, in point two we can tolerate harmonics to be cut or to have an interfering instrument if it approximates the original waveform better than having all harmonics present.

Moving to more concrete terms if we denote the digitized music waveform as $\mathbf{x}_i$ for every input channel i and the original anechoic waveform of the N instruments as $\mathbf{s}_j$ we can write the mixture using scalar mixing parameters $a_{i,j}$ for each instrument j and channel i as

$$\mathbf{x}_i = \sum_{j=1}^{N} (a_{i,j}\mathbf{s}_j + \mathbf{r}_{i,j}) + \epsilon \tag{1.1}$$

where $\mathbf{r}_{i,j}$ denotes the reverberation term and ϵ the noise term. In the algorithms used in this thesis we will not attempt to extract information from the reverberation although it may be possible to do so. Depending on the algorithmic approach the reverberation term will be included either in the instrument waveform or in the noise term leading to a more simplified formula

$$\mathbf{x}_i = \sum_{j=1}^{N} a_{i,j}\mathbf{s}_j + \epsilon \tag{1.2}$$

For the estimated instruments $\hat{\mathbf{s}}_{i,j}$ we can write the estimated mixture $\hat{\mathbf{x}}_\mathbf{i}$ omitting the noise term

$$\hat{\mathbf{x}}_i = \sum_{j=1}^{N} \hat{a}_{i,j}\hat{\mathbf{s}}_j \tag{1.3}$$

With the above notations we can now define the error $\mathbf{e}_i$ simply as

$$\mathbf{e}_i = \mathbf{x}_i - \hat{\mathbf{x}}_i \tag{1.4}$$

We will build our instrument separation algorithms during the remainder of this thesis based on the formulations and the notations given in this section. These include two approaches adhering to the high level goal definition and one approach built to obey the low level formulation.

1.3 Input Restrictions

The input file shall consist of sampled music in uncompressed pulse code modulated (PCM) format. To be more specific the implemented code accepts Microsoft's wave sound file format ending in "wav" and Sun Microsystems' audio file format ending in "au". The input file shall be recorded in stereo or binaural[1] mode with a sampling frequency of at least 11025Hz and a bit depth of at least 8bit per sample. Files with other encodings have to be converted into the two supported formats.

The input shall consist only of the music signal of interest with no leaders or trailers if they do not belong to the music. This last restriction is necessary in order to avoid creation of ghost instruments, which are already hard to cope with.

1.4 Overview

Following this introduction Chapter 2 will discuss related work already reported in literature. We will give an overview of the different approaches together with their strengths and weaknesses. This chapter will also give some background on the algorithms used during this thesis.

In Chapter 3 we will present a new template-based approaches for instrument separation with an improvement of it in Chapter 4. We will give some theory and practical examples leaving more rigorous evaluation for a separate chapter.

Chapter 5 contains a blind source separation approach together with some background information, theory and examples.

Some implementation details will be given in Chapter 6. This includes an explanation of the code structure, needed libraries and usage of the compiled code.

Chapter 7 provides an evaluation of all three approaches. More precisely, this chapter consists of a description of the test system, the data used, the testing procedure and the relevant outcomes in form of quality measures and timings.

---

[1] Binaural recordings are characterized by preserving phase information between channels, compared to stereo which can be mono-downmix compatible meaning that no significant phase difference between both channels may exist.

Finally in Chapter 8 we will summarize the work done during this thesis, draw some conclusions and give an outlook of future work.

1.5 Notation and Conventions

In mathematical formulas small letters set in an italic typeface denote scalars, bold letters denote vectors and bold capitals denote matrices. The indices attached to vectors or scalars begin counting from 1 unless otherwise noted. The symbol $\star$ denotes convolution where the right-hand side argument is time reversed and $\times$ denotes correlation.

In case of deviations the proper interpretation will be stated explicitly in the text.

Chapter 2

Related Work

2.1 Introduction

It is almost a centry ago when Boring [3] published a comprehensive article on how humans localize sound. At that time this was theory based on findings by studying neural cells and human behaviour. Nobody thought about trying to decompose a recorded sound into its parts. This remained that way until computers evolved and recording quality improved.

In 1990 Bregman [4] coined the term *auditory scene analysis* (ASA) which has now become a synonym for research on how the auditory system distiguishes sounds in real environments. Two years later the more common term *computational auditory scene analysis* (CASA) was first used by Brown [5] in his PhD thesis. CASA has since then become a framework for algorithms trying to separate sound sources using models of the human auditory system.

It was not until recently when research on separating musical instruments has begun to evolve. So this domain can be seen as rather new and still developing.

For our purposes we will divide literature relevant to instrument separation into five categories:

- *Blind Source Separation*. This is the class of algorithms making least assumptions on the structural characteristics of the audio signal so they can be applied in a general way.
- *Harmonic and Sinusoidal Modelling*. Simultaneously playing Instruments are assumed to have non-overlapping harmonics in the frequency domain most of the time. Therefore the algorithms listed here have to model each instrument's harmonic structure in order to separate them.
- *Model and Feature-Based Approaches*. Algorithms in this category try to separate instru-

ments by modelling them using a set of features. Some consist of classifiers using the features for discrimination whereas others use the features to model the instruments directly.

- *Template-Based Approaches*. Music is regarded as highly repetitive and structured in these approaches. So instruments are seen as being made up of templates. In order to separate the instruments the corresponding templates have to be learnt.
- *Speech Separation*. Speech is also regarded as an instrument in music. In this category we will gather algorithms specialized on separating speech from music.

We will discuss each item in more detail in its own section. In a separate section we will discuss how algorithms were evaluated in literature. A summary and conclusions will follow at the end of this chapter.

2.2 Blind Source Separation

Blind Source Separation (BSS) is a very large research field. It usually works with generic signals and an abitrary number of input channels. The inputs are regarded as mixtures of sources as in Eq. 1.2. The main goal of blind source separation is to optimize the mixing parameters $a_{i,j}$. The literature depicted here is only a small sample of the publications in the domain of BSS. Only papers concerning core aspects of our work will be discussed here.

As we deal with audio signals we can assume the mixing parameters to have an inter-channel delay and attenuation, as does Masters in his PhD thesis [20]. In the frequency domain the delay is transformed to phase differences and the attuation to magnitude differences. Assuming the sources are mixed with zero phase between the channels Masters further assumes that in the spectral domain a non-zero inter-channel phase difference is a result of two or more sources sharing that frequency bin. If only two sources are involved then the exact magnitudes can be retrieved by splitting the frequency bin in a way that the results have both zero phase difference. He exploits these relations using a bayesian estimator to assign the frequency content to each source and then a weighted substraction approach to separate the magnitudes. The estimator calculates the probabilities of the magnitude belonging to one of the sources from the magniude and phase difference input together with the combined loudness of both channels and the frequency. In order to obtain the distributions needed for the probability estimation, the bayesian estimator has first to be trained on data with known mixing parameters. Masters also considers other approaches for separating the magnitudes like matrix inverse systems and exact approaches, but the weighted substraction approach turns out to be the most promising one.

A downside in Masters' approach is that the phase difference is only used as an indicator for whether two sources overlap on a given frequency bin so the input mixture is limited to signals mixed with zero phase between the channels. For live recordings this is not the case and for studio recordings it is usually also not the case although phase differences may be much smaller than in live recordings.

As our input is music we can make further assumptions about the sources and the mixing parameters. One common assumption is that the sources are sparse, so their amplitude is mostly zero except for some locations in time. This has the advantage that the number of simultaneously active sources may decrease, which reduces the problem complexity.

Bofill [2] further assummes the sources to have a laplacian distribution. He uses magnitude and phase differences between channels but in contrast to Masters he does not assume zero phase mixing, thus making full use of the phase information contained in the signal. He then clusters each frequency component according to the magnitude and phase values. If two sources overlap in the frequency domain then their overlapping frequency bins will have ambiguous phases and magnitudes. He solves the problem of finding the correct magnitudes and phases for each source via second-order cone programming which minimizes the sum of magnitudes of the sources subject to some constraints. According to Masters [20], who also considers this approach, second order cone programming is computationally very demanding with respect to its gains compared to other approaches so the practical value of Bofill's approach is questionable.

Another possibility to separate audio sources is to do a segmentation in the frequency domain labelling each segment which source it belongs to. Hu and Wang [16] acomplish this by using a three step algorithm. First the frequency domain representation is smoothed in order to eliminate false positives. The smoothing is done pyramidally so the result is representation on multiple scales. The onsets and offsets are then detected from coarse to fine scale and matched in order to generate segments for each scale. Finally, the scales are integrated and collapsed to form only one segmentation in the time-frequency plot. The authors state that becaue onsets and offsets are some general cues used by CASA systems they can be used for separating different kinds of sources as for example voice and music, background noise, etc. Unfortunately it is not clear from their paper how labelling of the different sources is done after segmentation. Without correct labelling of each segment separation is not possible.

2.3 Harmonic and Sinusoidal Modelling

As our goal is to separate musical instruments, we can make even further assumptions. Usually, musical instruments have a harmonic structure. An instrument playing a note will resonate on a fundamental frequency and on some integer harmonics. The proportion of the harmonics to each other and to the fundamental frequency makes up for what we perceive as the timbre of an instrument. This fact is used by Zhang [41] as he detects and models the harmonic structure of instruments using some measure of structure evidence and stability. He then uses a pitch estimation algorithm in order to generate a harmonic data set, which is then clustered to obtain the harmonic structure of the instruments. The actual separation is done by predicting the harmonic components in the magnitude spectrum from each instrument and removing the content from the mixture.

In determining the harmonic structure of instruments or notes in general, overlapping harmonics pose a major challenge. There are two problems that have to be solved. At first the overlapping harmonics have to be indentified as such. Then their contribution to every instrument has to be determined. The more instruments are playing simultaneously the more likely they will share some harmonics. With every shared harmonic the probability of correctly estimating their contribution to the instruments they belong to sinks dramatically.

Every and Szymanski [8] tackle the problem of identifying overlapping harmonics by first applying a multipitch estimation algorithm which returns the fundamental frequency for every note and then looking whether the integer harmonics of every note collide with those of other notes. Non-colliding harmonics are separated and removed leaving only the overlapping ones. Usually harmonics will not exactly fall on one frequency so due to the leakage of the frequency transform their energy will spread over some frequency bins. Whenever two harmonics collide then their frequency also won't be exactly the same but due to their collision their energy peaks may merge. As in most cases their frequency and their energy is not the same the resulting peak will become asymmetric. Every and Szymanski separate the collided harmonics by predicting how the peaks may have looked before merging which is possible due to the aforementioned peak asymmetry. Resynthesis of the played notes is done afterwards.

Virtanen takes in [39] a more elaborated approach in separating instruments by their harmonic structure. His method is a form of sinusoidal modelling that is basically a subsumption of Zhang's and Every's approach. In sinusoidal modelling the sources are regarded as being harmonic tones which can be modelled by a sum of cosines with three free parameters - the frequency, the phase and the amplitude. Thus the modelling formula expressed in time domain is

$$x(t) = \sum_{j=1}^{N} \sum_{h=1}^{H} a_{j,h} \cos(2\pi f_{j,h} t + \theta_{j,h}) \tag{2.1}$$

where $f_{j,h}$ is the frequency harmonic h of tone j, $\theta_{j,h}$ the corresponding phase and $a_{j,h}$ the amplitude. Note that Virtanen considers only monaural input therefore only one mixture is available. Optimization of the formula is done by first using an onset detector in order to obtain the tone boundaries. Then a multiple fundamental frequency estimator is used to initialize the fundamental frequencies of each tone. The three free parameters are optimized by a solver which iteratively optimizes one of the three free parameters keeping the others fixed. The solver at the same time resolves overlapped harmonics. In a separate step the notes are assigned to their sources.

2.4 Model and Feature-Based Approaches

Although sinusoidal modeling is an effective method for separating harmonic instruments we are still left with percusive ones like drums and snares. Because of the rather short and noise-like sound of percussive instruments we need to find another way to separate them effectively.

A simple and somehow obvious solution for this problem is to separe the instruments into two groups, with the first being the harmonic instruments and the second the percussive ones. This approach is taken by Helen and Virtanen in [14]. They first decompose the magnitude spectrum of the input signal into two matrices using non-negative matrix factorization (NNMF) where one matrix represents a gain matrix and the second the corresponding spectral components. Due to the nature of NNMF both matrices will be exclusively made up of positive elements so the factorization intuitively will yield physically plausible gains and spectral components.

Now the main point in separating the spectral components into the two groups is the use of a set of features which discriminates between harmonic and percussive instruments. The grouping is eventually done using a classifier, which, following the input of the aforementioned features, yields a decision which group each spectral component belongs to. More specifically the autors use some spectral and temporal features combined with a support vector machine classifier. For more details about the features refer to [14].

The separated spectral components can now be resynthesized to form two audio streams for each group. As the phases of the frequency spectrum are lost during decomposition they now have to be reconstructed in order to complete the resynthesis. Helen and Virtanen use a common trick here: they use the phase of the input signal. Although this is an approximation to the original phase the artifacts generated are minimal.

Another work using features to discriminate between instruments is [33], where Teddy and Lai separate only two instruments, assuming that the type of the first instrument is known (e.g., string). Unfortunately, this is a rather severe limitation as songs usually contain more than two instruments. Teddy's and Lai's work is interesting inasmuch as the authors are borrowing some concepts of their method from the human auditory system.

The first step is pitch estimation which is accomplished by a feature extraction step using an auditory image model and a feature segregation step. The auditory image model consists of several stages which model the signal processing path of the human auditory system, for more details see [24, 26]. The feature segregation step uses a neural network which stores knowledge about the instruments in order to separate their feature vectors. In a following primary feature recognition stage the final feature vector of the first instrument is generated using another neural network. This primary feature vector is also fed into a third neural network to generate the pitch. The pitch is then used to anihilate the fundamental frequency and its harmonics from the mixture which will delete the first instrument leaving only the second one together with the non-harmonic parts of the first one.

2.5 Template-Based Approaches

A distinctive feature of instruments in modern music is their structurally induced repetitiveness. This feature is more pronounced on percussive instruments than on harmonic ones and therefore many algorithms in literature limit themselves on exploiting repetitiveness just for drum tracks.

A common approach for separing highly repetitive instruments is based on using templates which represent typical tones of their associated instruments. The tones described by a template are found using some kind of template matching algorithm. In a second step the template is usually adapted to better fit the tones associated with it. Separation is achieved by resynthesizing the audio signal using only templates belonging to one instrument.

In [43] Zils et al. for example use two templates for their drum track separation. They extract the tracks iteratively using the two templates which are correlated with the mixture and the resulting peaks picked according to some criteria of relevance. In their work the templates are initialized with the impulse response of a lowpass filter and bandpass filter. First they use the lowpass template to detect bass drums then the highpass one to detect snare drums. They refine the templates iteratively until the number of detected peaks of the correlation between the templates and the mix reach a fix point. First they use the bass drum template until the fixpoint is reached then the snare drum template. Separation is achieved by resynthesizing the templates into one mix resulting in the drum track. It is also possible to use either the bass drum or the snare drum template to resynthesize just one of the instruments.

Resembling Zils' approach Benaroya et al. in [1] consider the special case of two sources mixed into one mixture. In contrast to Zils who used the two templates for two drums Benaroya's approach is more generic and thus allows for more kinds of instruments but on the other hand is more restricted because it requires training on an previously separated excerpt of the mixture. Benaroya separates the sources using a bayesian estimator which uses gaussian scaled mixture models (GSMMs) to estimate the spectral amplitudes of each source. The GSMMs are introduced in his paper and described as approximating the distribution of a scaled random variable which has a gaussian distibution instead of a normalized one like the more common gaussian mixture models (GMMs) do. These GSMM were used because a sound with a specific power spectral density (PSD) could be repeated on another time index with a different amplitude which would require a GMM of its own. So the GSMM can be used for all amplitudes of a sound with a specific PSD. Hidden markov models (HMMs) were considered for separation but not used because in that case their computational cost for training them could easily get prohibitive. The bayesian estimator has first to be trained on previously separated excerpt in order to learn the parameters. This reduces the usefullness drastically as usually one has only the mixed sources.

Another way to view the template approach is presented in [38] by Virtanen. He considers the input signal to be a sum of convolutions between onset vectors and their corresponding magnitude spectrum templates. He adapts the templates and the onset vector using an iterative squared error minimization in order to minimize the input reconstruction error. The problem now is that the onset vector should be kept sparse, which means that only few elements shall be non-zero. Sparseness ensures that the final solution is plausible in the sense that the instruments are not hit too often or in too small time intervals. In order to keep the onset vector sparse, Virtanen uses a special cost term which he adds to the reconstruction error term. When reconstructing each instrument by doing the convolution only for its onset vector and template the lost phase spectrum has to be reconstructed, which Virtanen achieves by simply using the phase of the input signal.

A kind of hybride between blind source separation and template based approaches is the work of Plumbley et al. [28]. We recall the input signal being viewed as a weighted sum of sources where the weights represent the mixing parameters. Plumbley approximates the sources by so called atoms or dictionary items which are sparsely present in the mixed signal. He calls the matrix formed by the mixing parameters itself a dictionary. He introduces two approaches for learning the atoms were: time and frequency based ones, where the time approach is sample shift-invariant while the frequency approach is phase-invariant. The invariance is crucial as this means that the dictionary can be kept sparse and thus plausible. Further it means that a shift of the waveform which is represented by an atom within the sampling window will not be mapped onto a new atom, but rather on an already existing one. Concerning time based learning, it produces spikes while frequency based learning produces something like a piano roll which looks like an on/off activation chart. The output of both learning methods synthesizes to notes or part of notes during reconstruction. As Plumbley's goal is just the representation, no assigment of the atoms and encodings to different instruments is given. But with regard to the new sparse representation the remaining work of grouping should be made easier.

A similar approach to Plumbley's but restricted to the frequency domain take Kim and Choi in [18]. They view the signal as being composed of a weighted sum of optimally shifted spectral basis vectors which can be compared to a shiftable version of Plumbley's atoms in the frequency domain. The up or down shift in the frequency domain is done in a way which leads to the best reconstruction approximation. The spectral basis vectors are taken from candidate vectors which are selected to be sparse and to have the smallest reconstruction error after the mixing matrix which they call the encoding matrix because it also contains frequency shift information, was calculated by an overlapping non-negative matrix factorization. The authors achieve separation by resynthesizing only one of the basis vectors. As the basis vectors can not only be gain weighted but also frequency shifted, one basis vector usually is able to represent one instrument. But this works only with instruments whose harmonic structure is very stable through the entire note duration and is also stable for all playable notes. If those both conditions are not fulfilled, more basis vectors are needed for the affected instrument. This poses the problem of how to find out which of the final basis vectors converged to the said instrument. This last problem is common for many template based approaches.

2.6 Speech Separation

Because voice is also considered an instrument in music we will also present some works separating it. Although these algorithms can be considered as special cases of the already mentioned approaches, we have put them into a separate section because their specialization makes them stand out from the general instrument separation algorithms.

We begin with the work of Hu and Wang [15] who generally separate speech from interference. They begin with analyzing the signal by cochlear filtering model with a bank of 128 gammatone filters [25] and subsequent hair cell transduction [21]. This filtering leads to a filtered spectral decomposition of the input signal with frequency bins which contrary to the fast fourier transform (FFT) cover an inequal bandwidth. The resulting frequency bins are called time-frequency (T-F) units and their bandwidth increase with increasing center frequency. T-F units are merged based on temporal continuity and cross-channel correlation. Then, segments are grouped into a foreground and background stream. Pitch is detected from the foreground stream and then the units are labeled according to whether they belong to speech. For T-F units which due to uncertainities were note part of the segmentation more processing is done before labeling. Foreground and background stream labeling is then refined so that finally the foreground stream represents the target speech.

Li and Wang [19] extend the work of Hu and Wang [15] by focusing on singing voices. First they detect the presence of voice using a spectral change detector and a speech/non-speech classificator using mel-frequency cepstrum coefficients MFCC [42] as features. Then they detect the pitch of the voice by analyzing the input signal with a bank of 128 gammatone filters and by computing a normalized correlogram in order to obtain periodicity information. As the peaks indicating periodicity may be misleading they use a trained HMM to decribe the pitch generation process. The output of this stage is a pitch contour. For the final separation of the singing voice a modified version of Hu and Wang's algorithm is used which uses the additional information extracted for a more accurate labelling of the T-F units.

The works discussed previously use only monoaural input signals. Roman et al. on the other hand creates a binaural input signal in [31] by filtering monoaural input sources through head related transfer functions (HRTFs) associated to the direction of incidence. In this way a synthetic mix with several spacial cues is created where the original sources are known beforehand, wich comes in handy for evaluation purposes. Similar to the previous works Roman processes the input signal simulating the auditory periphery using a filerbank that models the cochlear filtering mechanism which is gain-adjusted to account for the direction independent middle-ear transfer. The result is then half-wave rectified and square-root compressed. The interaural time delay (ITD) is then calculated using cross-correlation of the channels, and the interaural intensity difference (IID) is calculated by the log power ratio between the two channels. Furthermore, a cross-correlogram is created to extract further spatial information. A binary mask is then determined, which selects frequency bands with more energy in the target source than the interference. During reconstruction the T-F components belonging to interference are nullified. In order to generate the binary mask, some parameters of the ITD/IID evaluator have to be trained. Fortunately, this training procedure allows using a training set consisting of unrelated audio data to the separation task because only parameters related to the recognition of the direction of the incident signals have to be trained which are not bound to statistical properties of the source like in other works using trainable classifiers. Furthermore, the training needs to be done only once which is a further bonus considering that in other works the algorithms need to be trained before attempting separation on every new input.

Some algorithms, although originally developed for speech separation, can also be used for separating more general audio source and even instruments. An example is the much cited work of Yilmaz and Rickard [40]. The key of their work is the so called W-disjoint orthogonality (WDO) introduced in their paper, which they use for measuring the disjointness of speech signals. As it turns out this measure can be applied to every audio source with harmonic character and therefore their algorithm can be used for these sources. More concretely the WDO exresses how well separated the sources are in the frequency domain, that is, how good an ideal binary mask could separate the sources in that domain. If the sources share only a few frequency bins then their ideal separation can be high while on the other hand if they share many bins their ideal separation will be low. Based on the WDO the authors present the degenerate unmixing estimation technique (DUET) considering that using two mixtures in order to extract more than two sources is a degenerate unmixing case. In order to separate the input they construct a magnitude-delay histogramm from the magnitude difference and the inter-channel delay of the signal spectrum and smooth it. Then they locate the peaks whose locations give the delays and magnitude differences of the sources. Using the parameters found they construct a binary time-frequency mask for every source, selecting only frequencies whose delays and magnitude differences are in the vicinity of the found parameters. Transformation of the selected frequencies in the time domain results in the separated sources. As shown by the authors this algorithm works well for up to six anechoically mixed harmonic sources but the performance degrades rapidly as a real-world example is evaluated where the sources were echoically mixed. In spite of this their algorithm is still usefull because of its general applicability.

2.7 Benchmarking in Literature

Mixing Modes

Evaluating separation results from real recordings is a difficult task. The main problem lies in the fact that in a real environment we do not know how the original sources may have sounded like. Therefore we have no exact measure on how good a separation result is. In order to get an estimate of the performance of the separation algorithms we have to rely on subjective evaluations.

A solution to the problem might be to use an artificial mix where the mixing parameters and the original sources are known. Evaluation can be done using some error measure, ideally a perceptually weighted one where inaudible artifacts are weighted less. This is the most common approach seen in literature. Unfortunately this approach has some severe downsides.

- The first one is that artificial mixing is usually done *anechoically* which is then also called *instantaneous mixing*. This is the simplest form of mixing where the source signals are added, optionally with an associated gain or sample shift. Real environments like rooms, halls, or even outdoors in the open field have some kind of *reverberation*. That is a multitude of small-gain time-decaying echos which are added to the original sound. Although it is possible to simulate reverberation with good software this is usually not done.

  The effect of reverberation on a separating algorithm is that due to the echos, the repeated parts of the input signal will not sound the same. That is due to the interfering echos from the former played notes. So the repetition will be harder to detect and the problem gets worse the shorter the repetition is. Another effect is that the frequency signature of each instrument gets smeared in the frequency domain. This becomes even worse whenever the algorithm relies on features like the magnitude difference and phase shift to locate the sources on stereo or binaural recordings. In that case due to reverberation the two features will become smeared and in the worst case some sources may even become undistinguishable.

Mixing with reverberation on the other hand is called *convolutive* or *echoic mixing* as the source signal is convolved with an *impulse response* of the environment. Here, the echos come up as spikes in the impulse response. Although reverberation degrades the performance of most algorithms there are some special kinds which can deal with it. As each instrument in a musical piece has another position in space it also has another reverberation or impulse response linked to it, which would theoretically be an aid in separating the instruments. Unfortunately, the processing power required for using reverberation information is still pohibitive but we shall note that our brain also uses reverberation as a cue in order to localize sounds.

- Another downside of artificial mixing is that it does not simulate the changing *directional receiving pattern* of the microphones when more than one microphone is used. For low frequencies microphones have a rather omni-directional pattern whereas for higher frequencies they get more directional. Figure 2.1 illustrates this behaviour on the polar pattern diagrams of two microphones. The changing pattern now means that the inter-channel amplitude differences of the sources on low frequencies will be much less than on high frequencies. The amplitude difference due to the different distances to the sources remain unchanged by this. So in conclusion algorithms using artificial mixing do not have to adapt the frequency dependent amplitude differences which leads us to expect their performance to decline when used in real environments.

- Lastly, the third downside is the missing *background* and *registration noise* in the artificial mix. This is usually a less important aspect as background noise is well controlled in music recordings. Registration noise with todays technology can be as low as the quantization noise in the digitization stage. Therefore a noiseless mix may often come close to reality.

One possible partial solution is to record the sources live in an anechoic room. Obviously, the missing reverberation problem still remains, but the other two problems would be solved. This is also a common approach in literature besides software mixing.

A perhaps more interesting solution would be if we could playback some previously digitized sources and record them using two microphones in a regular office room. This should

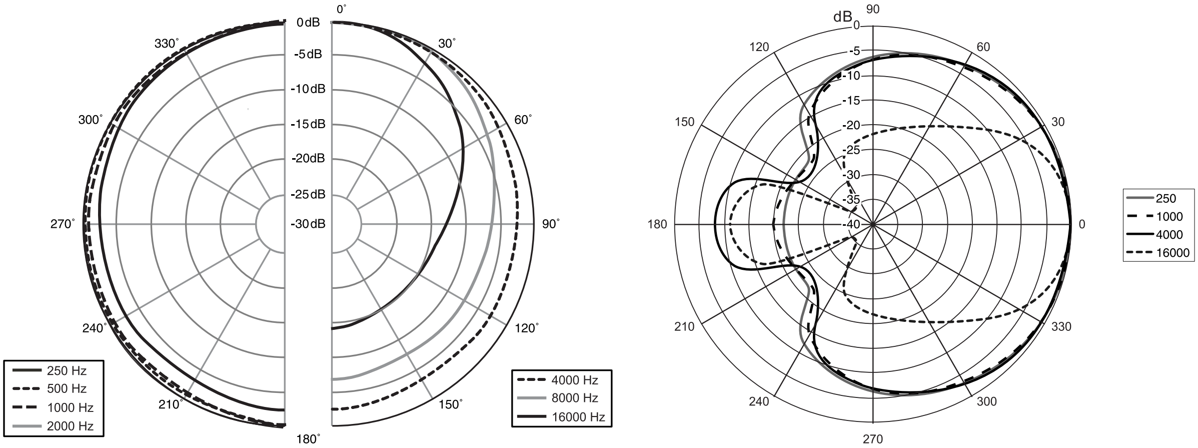

Figure 2.1 The illustration on the left shows the polar pattern diagram of the omni-directional microphone Behringer B-5. The sensitivity of the microphone is normalized to 0dB on the outer ring with decreasing sensitivity values on the inner rings. As we can see the pattern is truly omni-directional at 250Hz and below and gets more and more directional at higher frequencies. Note that the pattern is symmetric but in order to simplify the plot the different frequency patterns were split among the two halves. On the right we see the polar pattern of the Behringer B-1 microphone which is a directional one. Here we notice the same behaviour: the 250Hz curve is less directional than the 16kHz one. *Source: www.behringer.com.*

solve all three problems but could lead to one additional problem: the loudspeaker used will inevitably introduce additional distortion which may not be added to the evaluation error of the separating algorithm. As for the algorithm the distorted loudspeaker output actually is the source it has to separate so we have to either approximate the output of the loudspeaker and account for the distortions in the error measure or calibrate the loudspeaker in order to remove as much of the distortions as possible. The distortions generated by the loudspeaker can be regarded as being composed of the following parts:

- *Phase distortions* on more than one-way speaker systems if they have passive crossover networks. Usually a good speaker system consists of more than one speaker, where each one reproduces only a part of the frequency range. Therefore, the speaker electronics has to split the input into usually two or three frequency bands and delegate each band to the appropiate speaker. There are two kinds of so called crossover networks: the passive and the active ones. The passive network uses condensators and coils to split the frequency range. Unfortunately, this design introduces a phase shift in the output which, although inaudible, may generate a huge error when recorded and compared to the original. This is not the case for active crossovers which use more sophisticated electronics. Unfortunatley, the speaker specification often does not reveal the crossover type used which makes more than one-way speaker systems unsuitable for evaluation. A possible solution is the usage of an single broadband speaker which does not need any crossovers but even here we have some problems like the narrowed frequency range due to its physical limits.

- Distortions due to the *spectral ripple* in the frequency response curve and resonances of the loudspeaker or the speaker system. It is very hard to produce one-way speakers which are able to reproduce sounds over the entire audible range and even for the more complex speaker systems it is hard to make their frequency response absolutely flat.
- *Non-linear distorsions and noise.* This is a less important problem as those errors can be kept low with today's technology. Non-linear distortions include harmonic distortions where a loudspeaker may resonate on some harmonic frequencies of the original sound and clipping on loud sounds which is very well audible but can be easily avoided by turning down the volume.

The linear distortions to which we count the phase distortions, the frequency ripple and resonances can be dealt with by measuring impulse response of the speaker system and use it to either calibrate the speaker system or approximate the distorted output. Unfortunately, both solutions require expensive tools for proper calibration which can easily make this approach unfeasible. Still, if done right it is the best approximation of a real world scenario and has the benefit of knowing the original waveform of the sources which in turn enables a precise measure of the capabilities of the algorithms.

A summary of which mixing methods are used by different works is given later in this section.

Error Measures

In order to estimate the distortion created by the separation algorithms we need an error measure which weights the artefacts according to how good they can be heard, and that is easy to implement. In literature there is a plethora of error measures, each with its own benefits but unfortunately this also leads the evaluation to become less comparable as there is no such thing as a standard measure.

We will define here only the *Signal to noise ratio* (SNR) which is also known as the *signal to distortion ratio* (SDR) or *signal to residual ratio* (SRR). It is the most common measure which relates the original signal to the noise energy in the estimated signal. It can be expressed as:

$$SNR = 10\ \log_{10} \frac{\|\mathbf{x}\|^2}{\|\mathbf{x} - \hat{\mathbf{x}}\|^2} \tag{2.2}$$

where $mathbfx$ is the input signal and $mathbf\hat{x}$ its estimate. This measure is zero if the noise and signal energy are equal, and ∞ if the estimation is exact and thus there is no noise term.

The other measures are either very specialized like the *W-disjoint orthogonality* which measures the separation quality only for algorithms using time-frequency masks or are ambiguous as for example the *source to interference ratio* (SIR) where the definition in [40] is not compatible to the one used in [1, 12].

A summary of the measures used can be found on Page 2.1.

Corpora

Although some corpora exist the reviewed works tend to avoid them. Due to this fact the comparability suffers even more because now even the papers using a common error measure become incomparable due to the lack of a common evaluation basis. So in conclusion most papers use homebrew test databases built either from some unspecified commercial CDs or from self-recorded instrument sounds.

For the sake of completness we will give a brief description of the three corpora used more frequently:

- *BASS-dB: the Blind Audio Source Separation evaluation database*, [37]. This music database contains 20 multitrack recordings free for non-commercial use. Parts of it are used by [20].
- *TIMIT Acoustic-Phonetic Continuous Speech Corpus*, [10]. This is a speech database with 630 speakers, each one reading 10 sentences. Parts of it are used by [16, 31, 40].
- Cooke's speech collection, [6]. It consists of 10 speech utterances and 10 intrusions and is used by [15, 19, 31].

So we have two speech corpora and one music corpus. Unfortunately other music corpora have some troubling issues not discussed here therefore they are generally avoided by the research community.

Summary

As we have seen in this section, evaluation is a troublesome topic for blind source separation. We have to first decide whether to use synthetic mixes which allow an objective performance evaluation but without realistic results or we can use real world recordings but lacking the original sources we have to rely on subjective measures. As a third option there is the possibility of recording a playback in a room thus getting the benefits of the knowledge of the original sources and the realistic mixing environment but with the tribute of having to first calibrate the loudspeakers in order to avoid introducing additional distortion sources.

	capacity		mixing		error measures		corpora			
	ch.	src.	anechoic	echoic	SNR	others	BASS-dB	TIMIT	Cooke	others
[20]	2	*				✓	✓1)			✓
[2]	2	*	✓		✓					✓
[16]	1	2				✓		✓1)		
[41]	1	*			✓					✓
[8]	1	*	✓		✓					✓
[39]	1	*	✓		✓					✓
[14]	1	2 2)			✓					✓
[33]	1	2	✓			✓3)				✓
[43]	1	2				✓				✓
[1]	1	2				✓				✓
[38]	1	*				✓				✓
[18]	1	*	✓			✓4)				✓
[15]	1	2			✓5)	✓			✓	
[19]	1	2			✓				✓	
[31]	2	*			✓			✓1)	✓	
[40]	2	*	✓	✓		✓		✓1)		✓

Table 2.1 This is an overview of the papers reviewed in this chapter, grouped according to the section they were reviewed in. ch. represents the number of input channels to the algorithm, src. the maximum number of sources which can be separated where * means that this number is not limited in theory.

1) Only parts of the corpus were used.
2) Algorithm separates sources into 2 groups.
3) Subjective quality assessment.
4) Visualization only, no error measure.
5) Ideal binary masked signal was used as ground truth.

We have also seen that in literature plenty of error measures are being used, thus making results incompatible whenever papers do not use the same measure. To make things worse often homebrew databases were used which further agravates the problem of comparing the published results.

Finally we will give an overview of the evaluation methods used in Table 2.1. The papers are shown in the order they were reviewed in their respective sections. From the table we can see clearly that

- most papers do not use stereo information. This may be good or bad. Using only single tracks raises the compatibility with older, bandwidth limited or poor recorded music but on the other hand it misses valuable cues whenever stereo information is available.
- most papers use anechoic mixtures and many papers do not specify which type of mixture they use – where in that case one can usually assume the mixing is done anechoically.

- few papers use corpora at all and those using BASS-dB or TIMIT, use only parts of them.

2.8 Summary and Conclusions

In this chapter we have reviewed several works relevant to instrument separation. First we have seen some works with algorithms beloging to the domain of blind source separation with the main focus on using auditory cues and thus being generally applicable to any audio source separation task. Then we reviewed papers in the more specialized field of harmonic and sinusoidal modelling where the harmonic structure of instruments was used to differentiate between them. Following that we have seen some template based approaches exploiting the repetitive structure of music. In a section devoted only to speech separation we have seen some specialized algorithms for that task.

Finally we reviewed the benchmarking methods used in literature and have seen that evaluation is a problem in this field. We are confronted with the problem that we either have to choose between realistically mixed signals and no knowledge about the original sources or artificially mixed signals not representing reality well, but where the original sources are available. To overcome this problem we suggested to playback and record the signal using loudspeakers in a room. The problem arising from this technique is that the speakers have to be calibrated in order to not introduce distorsions in the signal path as it would make the separation afterwards much more difficult than it should be. Finally, in order to complicate matters, we saw a phenomenon which may be partially caused by the young nature of this emerging field: the works we reviewed tended to use self-made corpora and error measures thus making them incomparable to each other.

The illustrations and findings in the evaluation section will also be used as a basis for our own benchmarking in Chapter 7.

Chapter 3

Direct Template Matching

3.1 Motivation

We made an interesting observation which naturally leads to a template based approach. Let us start with computer generated music. In order to generate or synthesize a musical piece the computer needs a score and the tones of the instruments belonging to every note in that score. This approach is taken for example by the *musical instrument digital interface* (MIDI) format. When it is used for storing, only the score is saved in a file and the tones have to be provided separately, usually by the playback application. An interesting point here is, that due to lack of space in storing the tones of instruments only few tones of each instrument are really stored and remaining ones are synthesized by interpolating the available ones. If done correctly, music synthesized this way can sound almost indistinguishable to live played instrumental pieces.

If synthesizing using a score and a database of tones for each instrument produces a convincing result there comes the question whether it would not be possible to inverse the process of synthesization. That is to analyze a musical piece and store it in a score and tone-database format. If this can be accomplished then we have found an easy way to separate instruments: we just have to search the tone database for the tones belonging to one instrument and synthesize the score using only them. In top of that we would also have a score which means that further additional analysis besides instrument separation would be possible.

This may now sound easy but it is not and this is the reason why instrument separation or blind audio source separation have become such intensely studied fields in the last years. Still we have some arguments why such an approach intuitively should be successful.

- Computer generated music should intuitively be easy to be reconverted into the score and tone-database format because it was synthesized using that format – even if it was not MIDI. So a possible solution to the analysis problem already exists and that is the original representation. We only have to find an algorithm which can search for and discover that solution or an equivalent one.

- A considerable part of commercial music is at least partially computer generated. Some genres like techno, trance, electro and industrial are entirely made up of computer generated music.
- Even in music pieces played by humans the musicians have to either play a defined score or if they do an improvisation they can only use the tones available on their instruments and will be bound to some rules about harmonicity. These facts should intuitively restrict the search space for a plausible representation thus making the problem more tractable.

We have also thought about when and why a template based approach may fail which resulted in the following arguments.

- Human voice is very hard to synthesize using a score tone-database approach. So we may expect that also analysis resulting in a MIDI like will fail on speech.
- A musical piece is usually also post-processed after synthesizing by applying certain filters and then compressed. The post-processing filters are not necessarily invertible and are hard to be detected and lossy compression like the commonly used mp3 is by definition not fully invertible.
- Simple synthesizing applications always use the same tone for an instrument when generating the same note which eases analysis. However when playing real instruments this observation does not necessarily hold as it might not be possible or desirable to produce the same tone when playing the same note. During analysis this case has to be taken care of which might raise complexity considerably.

As we see success is not guaranteed as the approach may fail. Still we believe that it should be possible. Literature has shown some so called template based approaches more or less successfully solving some simpler subproblems as for example only separating the drums.

3.2 Problem Reformulation

We will base the problem formulation here on Section 1.2 where the basics were described. Now we view the sources as being composed of tones which can occur once ore more often at locations specified by the score for that instrument. So each tone $\hat{\mathbf{s}}_{i,j,k}$ has its associated steering vector $\hat{\mathbf{a}}_{i,j,k}$ which decides when and how loud that tone is played.

The steering vector is as long the entire song duration T in samples and consists of weights $\hat{a}_{i,j,k,t}$ which represent the loudness of the tone if it was onset at time $t = 1, 2, ..., T$. These weights have to be convolved with the time reversed tone in order to result in the audio waveform of the instrument playing just that tone. Formally it can be written as

$$\hat{\mathbf{x}}_{i,j,k} = \hat{\mathbf{a}}_{i,j,k} \star \hat{\mathbf{s}}_{i,j,k} \tag{3.1}$$

where $\hat{\mathbf{x}}_{i,j,k}$ is the resulting waveform of tone k from instrument j in input channel i. The tone vector has a fixed length D for all tones and instruments and represents just the time-domain waveform of the tone. The sum of all convolutions of all tones with all instruments results in the final waveform representation of the resynthesized musical piece

$$\hat{\mathbf{x}}_i = \sum_{j=1}^{N} \sum_{k=1}^{K} \hat{\mathbf{a}}_{i,j,k} \star \hat{\mathbf{s}}_{i,j,k} \tag{3.2}$$

where K is the number of tones per instrument. For an illustration on how the concept works see Figure 3.1.

Now we have the problem that we do not know the correct mapping of the tones to their corresponding instruments. Since solving that problem during the limited time we had for this thesis was not possible we will have to drop the association between the tone and the instrument. Formally, this is accomplished by dropping the index j and redefining K to be the number of tones in total. The simplified Equation 3.2 becomes

$$\hat{\mathbf{x}}_i = \sum_{k=1}^{K} \hat{\mathbf{a}}_{i,k} \star \hat{\mathbf{s}}_{i,k} \tag{3.3}$$

So far this means that we now have a set of tones and their scores but we do not know which instrument they belong to. This formula will be taken as the basis of the template based approaches.

Note that the estimated tone waveform $\hat{\mathbf{s}}_{i,k}$ is always normalized to have unity vector norm $\|\hat{\mathbf{s}}_{i,k}\| = 1$. Normalization is necessary because we have two free parameters which influence each other: the steering vector loudness and tone waveform loudness. If we multiply the steering vector loudness by some constant we have to divide the tone waveform loudness by the same constant in order to keep the overall loudness. Therefore we have to fix one of these parameters so we choose the tone waveform loudness. We use the vector norm as a relative loudness measure which we set to 1 to simplify matters. Note that this can not be viewed as an absolute loudness measure as it depends on the number of samples D per tone. Still if all tones have the same length the normalization makes the loudness weights stored in the steering vector comparable to other tones.

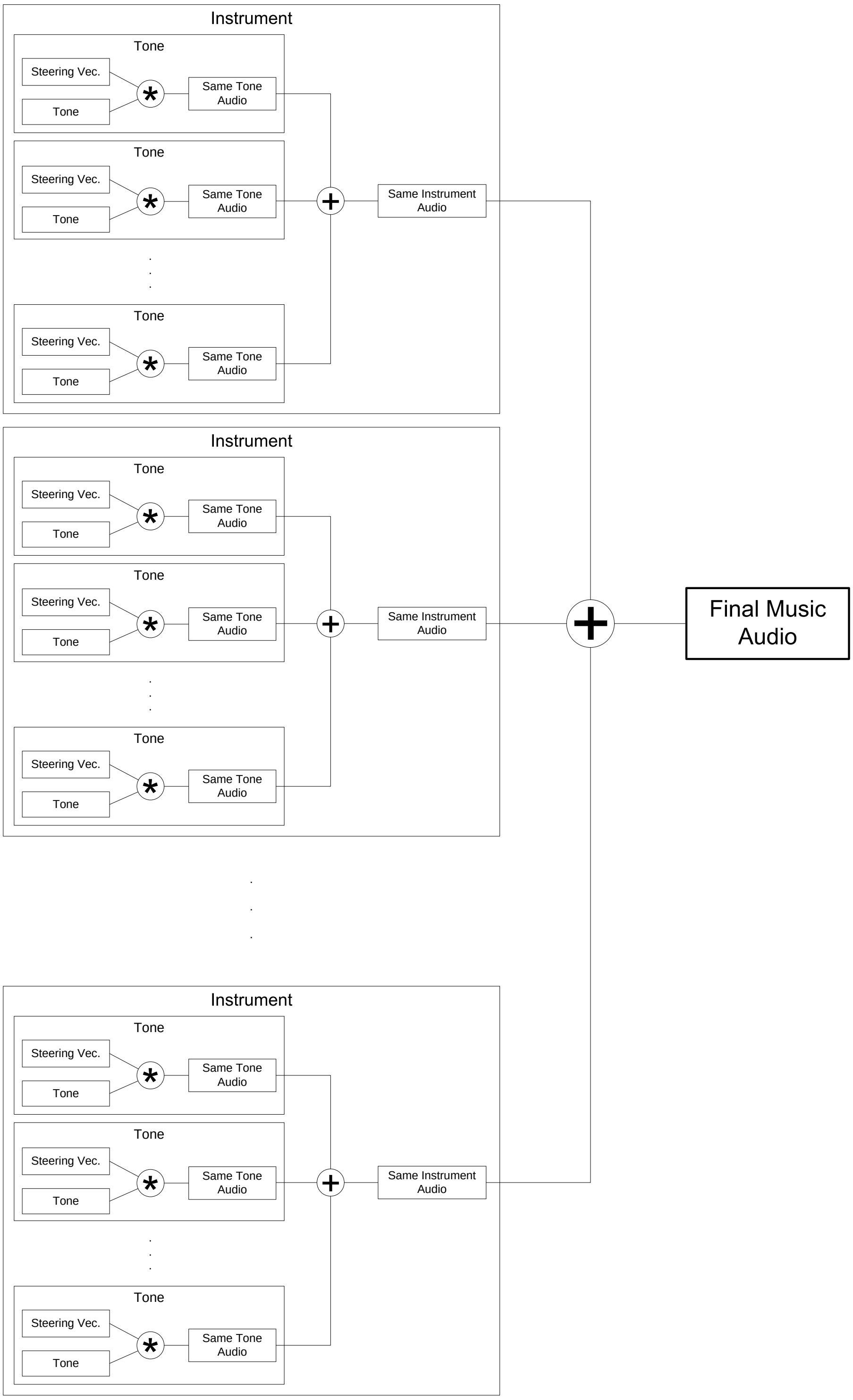

Figure 3.1 Overview of the music synthesis concept using templates. For each tone the steering vectors are convolved with the time-reversed tone waveform to produce an audio signal containing only that specific tone whenever it is played. These signals are summed over each instrument to produce further waveforms containing only that instrument. Our goal ends here but if we want to synthesize or reconstruct the whole musical piece we have to finally mix the instrument waveforms.

3.3 Algorithm Overview

Our first step is to initialize the score or more precisely the onset vectors and the tone waveforms where each combination of onset vector and tone waveform constitute a template. The exact description of the onset vector and its initialization procedure is given in a separate subsection on Page 25.

After the initialization an iterative method is employed to build the templates which is will be called the main algorithm during the rest of this section. An overview of this method is given on in a subsection on Page 27. The iteration comprises of a tone search step described in a subsection on Page 28 where the onset vector is estimated using the available tone waveforms and of a tone learning step beginning on Page 36 where the tone waveforms are adjusted to better match the found onset vector.

Fine tuning steps have a separate subsection beginning at Page 46. These steps are not necessary for the functionality of the algorithm but may provide better quality.

As all the previously described steps are rather detailed and some contain several versions and expansions of their corresponding algorithms a final algorithm subsection is presented on Page 51.

Finally we close the first approach with a summary and future work subsection on Page 54.

3.4 The Onset Vector

We note that the steering vector consists only of a few non-zero values as a single tone will not be played often in a musical piece. Therefore we can store only these non-zero values and save a vast amount of space by doing so. We will call the resulting vector the *onset vector* as its stored non-zero elements now represent the time locations of the tone onsets together with the tone loudness.

The convolution operation can also be simplified as we do only need to consider stored onsets thus we can do the convolution by simply adding a weighted copy of the tone waveform beginning at the location pointed by the onset vector using the stored loudness as the weight.

Before we begin with finding the right onset vectors and tone waveforms we have to do a first and for the beginning rather uncomplicated guess. A first try is to initialize the onset vector with some randomly located onsets with small random loudness weights and the tone waveform with some white noise which is normalized to have a vector norm of 1. Our hope is here that the random templates will converge during the algorithm to their real tone templates they resemble most during the algorithm, but unfortunately this does not work out.

So we try a more promising initialization attempt. Now we will not initialize all templates at the same time but one after the other. More precisely we initialize the first template, do a learning iteration and then initialize the second template from the reconstruction error left by the first template. For the third template we use the reconstruction error caused by the first two templates and so on. This way we can suppress interference from the tones in the input signal covered by the already initialized templates when initializing the new ones.

The initialization of each template is done by first dividing the reconstruction error into windows of the same size as the tone templates. For the first template we use the original input signal instead of the reconstruction error. In the next step we search for the window with the highest energy content using the vector norm as the energy measure and initialize the tone template with the content of that window. In order to normalize it we divide its content by the squared vector norm and use the same value to initialize the onset vector at the same location where the found window begins in the input signal. The rest of the onset vector is initialized with zeros.

The step by step description of the initialization procedure described in the former paragraph is as follows:

- divide the input into slices of size D (where D is usually in the range of $[5000...50000]$) and calculate the loudness λ_m for each slice $m = 1, 2, ..., \lfloor T/D \rfloor$.

  $$\lambda_m = \|\mathbf{x}_{m,i}\| \tag{3.4}$$

  where $\mathbf{x}_{m,i}$ is the vector of slice m.
- find the slice m' with the highest $\lambda_{m'}$.
- initialize the tone waveform $\hat{\mathbf{s}}_{i,k}$ with $\mathbf{x}_{m',i}$.
- initialize the onset vector with $\|\hat{\mathbf{s}}_{i,k}\|^2$ on the time location where the slice m' begins and set the rest of it to zero.
- divide the tone waveform by $\|\hat{\mathbf{s}}_{i,k}\|^2$ in order to normalize it to unity vector norm.

This second approach now concentrates on the parts with the highest residual error thus making a more plausible approach than simple random initialization. Trying to estimate the parts in the musical piece with the highest residual can be interpreted as trying to estimate the loudest and more important parts. The results of this initialization are somehow better but there is still room for improvement. A shortcoming of this approach is, for example, that the algorithm will concentrate on a few loud and hard to estimate portions of the input and leave the rest untouched.

In order to ameliorate this problem we thought of a hybrid between the random initialization and the residual based one. So after computing the reconstruction residual of the preceding templates we do the following to initialize the new template:

- compute the mean loudness λ for the whole input
$$\lambda = \frac{1}{\sqrt{T}} \|\mathbf{x}_i\| \tag{3.5}$$
- divide the input into slices of size D and calculate the mean loudness λ_m for each slice $m = 1, 2, ..., \lfloor T/D \rfloor$.
$$\lambda_m = \frac{1}{\sqrt{D}} \|\mathbf{x}_{m,i}\| \tag{3.6}$$
where $\mathbf{x}_{m,i}$ is the vector of slice m.
- randomly pick a slice m' until $\lambda_{m'} > \lambda$.
- initialize the tone waveform $\hat{\mathbf{s}}_{i,k}$ with $\mathbf{x}_{m',i}$.
- initialize the onset vector with $\|\hat{\mathbf{s}}_{i,k}\|^2$ on the time location where the slice m' begins and set the rest of it to zero.
- divide the tone waveform by $\|\hat{\mathbf{s}}_{i,k}\|^2$ in order to normalize it to unit vector norm.

In words this initialization procedure randomly picks slices from the input signal until one is found whose mean loudness is greater than the mean loudness of the entire piece. The tone waveform is initialized with the found slice and normalized to unit vector norm. Finally, as in the previous approach the onset vector is initialized by the normalization value of the tone waveform and the rest is set to zero.

This initialization procedure also initializes the templates with high-energy residual content but this time the templates are spread more evenly throughout the input. Intuitively, this approach should cover more different tones than the previous one because of the better spread.

3.5 Main Algorithm

Finding the onset vectors and the tone waveforms is done in an iterative manner. This is done because we need to either rely on given tone waveforms while searching for the onset vectors or rely on given onset vectors to find the best matching tone waveforms. Therefore both the onset vector and the tone waveforms are refined during each iteration.

The algorithm is set to stop when a predetermined number of iterations has been reached. We decided for this stopping criterion as the algorithm is very time consuming if viewed on a per iteration basis and we cannot wait till convergence. Furthermore we have observed that convergence may even not be given because as we will see in the next subsection, the tone search step uses some heuristics in order to find a plausible onset vector and these heuristics do not seem to have a fix-point.

Concerning the tone search and learning steps which are part of this main algorithm, we had two options on what kind of input signal to feed them. We could either feed

- *the original signal* $\mathbf{x}_i$. This would cause all templates to extract the onsets and tone waveforms from the same mixture. It has the disadvantage that already correctly identified tones would interfere with the tone search and learning steps for the other templates.
- *the individualized reconstruction error signal* $\mathbf{e}_{i,k}$. The individualized reconstruction error here is simply the reconstruction error of all templates except the template k to which it is fed as the input or more formally

  $$\mathbf{e}_{i,k} = \mathbf{x}_i - \hat{\mathbf{x}}_i + \hat{\mathbf{a}}_{i,k} \star \hat{\mathbf{s}}_{i,k} \tag{3.7}$$

  where $\hat{\mathbf{x}}_i$ is the reconstructed signal as defined in Equation 3.3. This method has the advantage that the correctly identified tones of the other templates will not be present in the input. The disadvantage is that it now includes the errors made by the other templates. Usually the errors made by the other templates should be a smaller problem because as the other templates try to estimate the input signal their error will be smaller than their gain thus we can expect this kind of error to remain small.

So depending on the option chosen we would either extract the tones all at once from the original signal or we would extract tones one by one. We decided for the second one using the individualized reconstruction error as it proved to be more stable and give higher quality results.

We shall note here that the final version of the tone learning step uses the original signal directly as it jointly adjusts all tone waveforms. So in the final main algorithm only the tone search step is performed for each template separately and therefore is able to use the individualized reconstruction error.

3.6 Tone Search

One of the main parts of the algorithm is the search for the templates in the input signal. We have to find the locations in the input signal where the tones waveforms occur. This is accomplished by a template matching algorithm.

Correlation

Our first approach takes the correlation between the input signal and each tone waveform resulting in the correlation vector $\mathbf{r}_{i,k}$ where each element is calculated using

$$r_{i,k,t} = \frac{D \sum_{l=1}^{D} e_{i,k,t+l-1} \hat{s}_{i,k,t+l-1} - \sum_{l=1}^{D} e_{i,k,t+l-1} \sum_{l=1}^{D} \hat{s}_{i,k,t+l-1}}{\sqrt{D \sum_{l=1}^{D} e_{i,k,t+l-1}^2 - \left(\sum_{l=1}^{D} e_{i,k,t+l-1}\right)^2} \sqrt{D \sum_{l=1}^{D} \hat{s}_{i,k,t+l-1}^2 - \left(\sum_{l=1}^{D} \hat{s}_{i,k,t+l-1}\right)^2}} \tag{3.8}$$

with $t = 1..T$. Assuming $\hat{\mathbf{s}}_{i,k}$ and $\mathbf{e}_{i,k}$ have zero mean over any window with a size of D samples we can simplify the above formula to

$$r_{i,k,t} = \frac{\sum_{l=1}^{D} e_{i,k,t+l-1} \hat{s}_{i,k,t+l-1}}{\sqrt{\sum_{l=1}^{D} e_{i,k,t+l-1}^2} \sqrt{\sum_{l=1}^{D} \hat{s}_{i,k,t+l-1}^2}} \tag{3.9}$$

The zero mean condition over a specific time usually holds for audio signals when the window size is sufficiently large because these signals are low-pass filtered in order to remove imperceptible low frequency waves which may cause clipping problems when present. We can further simplify the formula as we note that we have normalized the tone waveform $\hat{\mathbf{s}}_{i,k}$ to have unity vector norm so we get

$$r_{i,k,t} = \frac{\sum_{l=1}^{D} e_{i,k,t+l-1} \hat{s}_{i,k,t+l-1}}{\sqrt{\sum_{l=1}^{D} e_{i,k,t+l-1}^2}} \tag{3.10}$$

Unfortunately this equation can not be solved on today's computers in reasonable time due to its complexity of $\mathcal{O}(TD)$ as the tone size D is in the range of several thousand samples and the size of the musical piece T is in the range of millions of samples.

So we have to find a way to lower the complexity of this formula. Noting that the numerator is an unscaled discrete correlation we can use the FFT for solving it where the FFT algorithm has a complexity of $\mathcal{O}(T \log_2 T)$. Now we can rewrite the formula using our nomenclature for the discrete correlation as

$$\mathbf{r}'_{i,k} = \mathbf{e}_{i,k} \times \hat{\mathbf{s}}_{i,k} \tag{3.11}$$

and

$$r_{i,k,t} = \frac{r'_{i,k,t}}{\sqrt{\sum_{l=1}^{D} e_{i,k,t+l-1}^2}} \tag{3.12}$$

Furthermore we can use a sliding sum algorithm for calculating the denominator. For that we square every input sample and calculate the sliding sum resulting in a temporary vector where each element holds the sum of squares for the needed window size D. Now we apply the square root on each of its elements and get the final denominator vector. The complexity of this algorithm is $\mathcal{O}(T)$ so we finally get a complexity of $\mathcal{O}(T \log_2 T)$ for the entire correlation formula which is an acceptable result.

In the correlation vector obtained so far, each element indicates how much energy content of the D sample sized window following the location of the element is explained by the tone waveform. The elements are 1 whenever the template matches exactly and 0 if there is no correlation. As one might expect there is always some correlation between template and the input therefore $r_{i,k,t}$ is never 0. This may also be partly caused by the zero mean assumption earlier, as the correlation vector is now an approximation.

Peak Generation

The next step in finding the tones in the input signal is to search for the positions where the correlation vector has relatively high values. Our assumption is that the tones are located at positions with high correlation. This is not necessarily always true as the case might occur when two tones annihilate most parts of each other in the input signal by creating beats which will make the correlation between the template associated with each tone to get rather low.

Although we are interested in high correlation values we note that there must be a minimum of time between onsets so we have to find the local peaks of the correlation and ensure that there is enough space between the peaks found.

Finding all the peaks in the correlation vector is easy. We have to apply the discrete derivative opertor Δ on $\mathbf{r}_{i,k}$, which is defined element-wise as

$$\Delta r_{i,k,t} = r_{i,k,t} - r_{i,k,t-1} \tag{3.13}$$

for all $t > 1$. And then we have to find those points where $\Delta r_{i,k,t}$ is zero or changes sign from positive to negative and where $r_{i,k,t}$ is positive. The last condition ensures that we use only positive correlations which are physically meaningful. Negative correlations can not be explained physically as it would mean that the waveform of the instrument would have to be inverted. To our knowledge no real instrument can be played that way and it would also be meaningless because the human ear would not be able to distinguish such a sound from the non-inverted original. We will thus express the peak picking more formally as having a set of indices $\mathcal{T}_{i,k}$ storing time indices t of the correlation vector $\mathbf{r}_{i,k}$ so that

$$\mathcal{T}_{i,k} = \{t \mid r_{i,k,t} > 0 \wedge \Delta r_{i,k,t-1} \geq 0 \wedge \Delta r_{i,k,t} \leq 0\} \tag{3.14}$$

with t ranging from 3 to T. Note that the time indices in $\mathcal{T}_{i,k}$ are not bound to the correlation vector they were taken from. Therefore they can also be used to index other vectors of the same length as for example the onset vector $\hat{\mathbf{a}}_{i,k}$ and the input vector $\mathbf{e}_i$. We will use this property later during the template learning stage.

Peak Picking

After having the set of all peaks there comes the tricky part of choosing the "right" ones. That means we have to filter the peaks using some heuristics.

Our first heuristic is also the simplest one which we mention here for the sake of completeness. First off we define a desired number of onsets to extract which we call θ_a and the minimal distance between the onsets θ_b. For each tone k we do the following:

1. initialize the filtered set $\mathcal{T}'_{i,k}$ to the empty set $\varnothing$.
2. save $\mathcal{T}_{i,k}$ into $\mathcal{T}_{1,i,k}$.
3. do the following until $\mathcal{T}_{i,k}$ is empty:
    - grab the smallest t from $\mathcal{T}_{i,k}$ which we will denote t_0.
    - search for the $r_{i,k,t\in\mathcal{T}_{i,k}}$ with the highest value having $|t - t_0| < \theta_b$. We will call the time index of the result t_1.
    - add t_1 to $\mathcal{T}'_{i,k}$.
    - remove all $t \leq t_1$ from $\mathcal{T}_{i,k}$.
4. go through each element of $\mathcal{T}'_{i,k}$ and check whether there are any t_0 and t_1 with $|t_1 - t_0| < \theta_b$. If any such elements are found then remove the one with the smaller value from $\mathcal{T}'_{i,k}$.
5. if $\left|\mathcal{T}'_{i,k}\right| > \theta_a$
    - multiply θ_b with 3 in order to enlarge the minimal distance and get fewer results next time.
    - restore $\mathcal{T}_{i,k}$ from $\mathcal{T}_{1,i,k}$.
    - start over again.
6. if $\left|\mathcal{T}'_{i,k}\right| < \theta_a$
    - divide θ_b by 2 in order to shrink the minimal distance and get more results next time.
    - restore $\mathcal{T}_{i,k}$ from $\mathcal{T}_{1,i,k}$.
    - start over again.
7. set $\mathcal{T}_{i,k}$ to equal the filtered set $\mathcal{T}'_{i,k}$

The algorithm in words does approximately the following: It iteratively gets the highest correlation element within the minimal onset distance from the beginning and saves it into the new set $\mathcal{T}'_{i,k}$. Then it removes all elements which came before it including itself from the original peak set. This iteration is repeated until the original peak set is emptied. The result of this is not guaranteed to always have the minimum desired distance so we have to check if that condition holds in the new set. If two elements are found violating that condition then we remove the one which has the lower correlation. Now we check whether the desired number of onsets was found and if not then the minimum onset distance is adjusted so that after running the algorithm again using a saved version of the original peak set the number of onsets will come closer to the desired one. We then store the result back in $\mathcal{T}_{i,k}$.

This algorithm is very simple and has a low complexity but also has several drawbacks. First the desired number of onsets to be found is predetermined, which is not realistic as we do not know the number of times a tone is played beforehand. This is a big drawback as it is crucial that we not only find the right positions of the onsets but also do not find too many as each false positive will "pollute" the tone waveform by some degree later in the learning stage. A second drawback, but of lesser importance, is, that even so the algorithm will usually not find the optimum for a fixed minimum onset distance. If we would want to find the optimal set of locations with maximal correlation subject to the minimum onset distance constraint we would have to resort to more complex algorithms. Another drawback is that this algorithm returns onsets almost evenly distributed through the input signal as we specify only a minimum distance between onsets. Usually, between two points with minimum distance there will always be an onset, even if it is weak and around the noise floor. Therefore the algorithm will find onsets even during parts of silence where noise dominates. We thus drop this algorithm and design a replacement.

So we have thought of a second peak picking algorithm which should alleviate some of the problems from the first one. This time we begin with the peaks having the highest correlation and take them if there is no other peak already taken in the neighbourhood having a distance smaller than the minimal one. In detail we do the following:

- initialize the filtered set $\mathcal{T}'_{i,k}$ to the empty set $\varnothing$.
- do iteratively until $\left|\mathcal{T}'_{i,k}\right| = \theta_a$:
  - take the time index t_0 with highest associated r_{i,k,t_0} from $\mathcal{T}_{i,k}$ and remove it from $\mathcal{T}_{i,k}$.
  - if there is no other t_1 in $\mathcal{T}'_{i,k}$ so that $|t_0 - t_1| < \theta_b$ then add it to $\mathcal{T}'_{i,k}$.
- set $\mathcal{T}_{i,k}$ to equal the filtered set $\mathcal{T}'_{i,k}$

Now, though even this algorithm will not find the optimal solution that is the solution with the maximal sum of correlations it will not return evenly distributed onsets anymore. Compared to the former algorithm the minimal distance between onsets has become more important now as if it is chosen too small we may easily get false positives and if we choose it too big we will not get the desired number of onsets. This was not a problem on the former algorithm as it adjusted the minimal distance in order to obtain the desired number of onsets.

As the problem with the predefined number of onsets still remains we have extended this algorithm in order to eliminate this free parameter and let the algorithm decide on the optimal number of onsets. In order to do this a new parameter θ_d with $0 < \theta_d < 1$ is introduced which describes the minimum correlation that must be present for a time index in order to accepted. The algorithm proceeds as follows:

- initialize the filtered set $\mathcal{T}'_{i,k}$ to the empty set $\varnothing$.
- do iteratively until $|\mathcal{T}_{i,k}| = \varnothing$:
    - take the time index t_0 with highest associated r_{i,k,t_0} from $\mathcal{T}_{i,k}$ and remove it from $\mathcal{T}_{i,k}$.
    - if there is no other t_1 in $\mathcal{T}'_{i,k}$ so that $|t_0 - t_1| < \theta_b$ and if $r_{i,k,t_0} > \theta_d$ then add it to $\mathcal{T}'_{i,k}$.

Intuitively θ_d describes the confidence in whether the onset is a true positive. Note that the correlation is normalized therefore an onset occurring during a loud passage must be louder in order to be taken than an onset in a quiet part. This should be intuitively plausible as during a loud passage there may be many fixed low-loudness matches than during a quiet one where there may be even none. The fixed correlation approach should alleviate this problem but unfortunately it creates a new one which we may be familiar with from the first algorithm: during silent periods we may get matches with noise as we only account for the relative correlation above a fixed threshold which may still be given even during such periods. We might overcome this problem by also introducing a free parameter for a low-loudness threshold below which no correlation peak is accepted at all. Unfortunately we ran out of time for further pursuing this as the exact value needs some fine tuning to account for the different music genres.

After some preliminary testing we observed some practical shortcomings of the algorithm:

- in stereo recordings when a peak was detected in one channel no match could be found in the other channel at a nearby time location. This should not be the case but in very old computer generated music files.

- the algorithm often finds matches where there should be none as a result of a low θ_d. This allowed the algorithm to pick some coincidentally good matching templates whenever a piece of the input signal has not been covered by any template thus constructing that part of the input signal by templates which do not match there in reality.
- very few tones were found whenever higher θ_d were used.

It seems that unfortunately there is no ideal value θ_d. Either it is too high and the instruments sound choppy or too low and unknown input signals are reconstructed by some random matching templates.

This led us to an extension which should ameliorate the problems found. The extension does some additional pre-processing by using stereo cues to filter out implausible peaks and at the same time lowers θ_d in order to get more matches through.

The implausible peaks are filtered by the following criteria:

- *constrained sample-shift*. Peaks in one channel having no matching peak in the other channel in the interval of $\pm\theta_e$ samples are deleted. Here $\theta_e \in \mathbb{N}$ is a parameter which can be freely adjusted. Ideally it should represent the maximum sample shift that can occur when using two microphones. It can be calculated by

  $$\theta_e = \frac{d_{\text{mic}}}{c_{\text{air}}} f_{\text{max}} \tag{3.15}$$

  where d_{mic} is the distance between the two microphones in meters, c_{air} the velocity of sound in meters per second and f_{max} is the sampling frequency in Hertz. Unfortunately we usually do not have information about the distance between the microphones so we have to guess. Usually microphones are not placed further apart than the distance between the two ears which is about 25cm but sometimes when a more dramatic effect is desired the microphones can be placed as far as 50cm apart. That means that a if the distance in not known it can be assumed to lie in between 25cm and 50cm.
- *constrained shift deviation*. Assuming that instruments generally are not moved around in the room while being played we can fix their position and thus their time-shift between the two channels. This can be done by first getting the average shift between peaks of all tones belonging to that instrument and then restricting the shift variation consequently filtering out each peak pair whose variation is too high. Unfortunately we have no tone to instrument mapping function yet therefore we have to calculate the average and fix its variation for every tone template in part.
- *constrained loudness difference deviation*. With the same assumption as the point above we can also constrain the normalized loudness difference $\mathring{\lambda}_{k,(t_1,t_2)}$ between the two

channels which we calculate from the unnormalized correlation $r'_{i,k,t}$ defined in Equation 3.11

$$\mathring{\lambda}_{k,(t_1,t_2)} = \frac{r'_{1,k,t_1} - r'_{2,k,t_2}}{r'_{1,k,t_1} + r'_{2,k,t_2}} \tag{3.16}$$

with $t_1 \in \mathcal{T}_{1,k}$ and $t_2 \in \mathcal{T}_{2,k}$. As above we also first get the average normalized loudness difference for each tone in part and then in a second step filter out each peak pair whose difference is varying above a given threshold $\theta_g \in \mathbb{N}$.

- *minimum relative loudness*. At this stage assume that a pair of peaks must have a minimum loudness compared to the total energy in the input signal at the location of the peak in order to be a hit. As above we use the un-normalized correlation $r'_{i,k,t}$ to calculate the relative summed loudness $\tilde{\lambda}_{k,(t_1,t_2)}$ as

  $$\tilde{\lambda}_{k,(t_1,t_2)} = \frac{r'_{1,k,t_1} + r'_{2,k,t_2}}{\sqrt{\sum_{l=1}^{D} e^2_{1,k,t_1+l-1}} + \sqrt{\sum_{l=1}^{D} e^2_{2,k,t_2+l-1}}} \tag{3.17}$$

  with $t_i \in \mathcal{T}_{i,k}$. Now all pairs having $\tilde{\lambda}_{k,t_1} < \theta_h$ where θ_h is a threshold parameter with $0 < \theta_h < 1$ are considered noise and filtered out.

The extended algorithm starting from a set of candidate peaks identified now works as follows

- generate a set $\mathcal{T}_k$ of all time index pairs (t_1, t_2) with $t_1 \in \mathcal{T}_{1,k}$, $t_2 \in \mathcal{T}_{2,k}$ for the two peak candidate lists $\mathcal{T}_{1,k}, \mathcal{T}_{2,k}$ of two stereo channels, where each pair is restricted to have a sample-shift below the treshold θ_e, or more formally

  $$\mathcal{T}_k = \{(t_1, t_2) \mid t_1 \in \mathcal{T}_{1,k},\ t_2 \in \mathcal{T}_{2,k},\ t_1 - \theta_e < t_2 < t_1 + \theta_e\} \tag{3.18}$$

  Note that the time indices may have been duplicated and can now occur in more than one pair. There may now be even more pairs than there were time indices before.

- calculate the mean sample-shift deviation $\bar{z}_k$

  $$\bar{z}_k = \frac{1}{|\mathcal{T}_k|} \sum_{(t_1,t_2)\in\mathcal{T}_k} t_1 - t_2 \tag{3.19}$$

- generate a new set $\mathcal{T}'_k$ containing only those peak pairs from $\mathcal{T}_k$ whose difference between time indices is less than θ_f where $\theta_f \in \mathbb{N}$ is freely adjustable parameter. More formally

  $$\mathcal{T}'_k = \{(t_1, t_2) \mid (t_1, t_2) \in \mathcal{T}_k,\ |t_1 - t_2 - \bar{z}_k| < \theta_f\} \tag{3.20}$$

- calculate the mean loudness difference $\bar{\lambda}_k$

  $$\bar{\lambda}_k = \frac{1}{|\mathcal{T}'_k|} \sum_{(t_1,t_2)\in\mathcal{T}'_k} \mathring{\lambda}_{k,(t_1,t_2)} \tag{3.21}$$

  where $\mathring{\lambda}_{k,(t_1,t_2)}$ is the loudness difference as defined in Equation 3.16.

- constrain the normalized loudness difference $\mathring{\lambda}_{k,(t_1,t_2)}$ with $(t_1, t_2) \in \mathcal{T}'_k$ to have a vary in the range of $[-\theta_g..\theta_g]$ by generating the set $\mathcal{T}''_k$

  $$\mathcal{T}''_k = \left\{(t_1, t_2) \mid (t_1, t_2) \in \mathcal{T}'_k,\ \left|\mathring{\lambda}_{k,(t_1,t_2)} - \bar{\lambda}_k\right| < \theta_g\right\} \tag{3.22}$$

- enforce the minimum loudness θ_h by creating another set $\mathcal{T}'''_k$

  $$\mathcal{T}'''_k = \left\{(t_1, t_2) \mid (t_1, t_2) \in \mathcal{T}''_k,\ \tilde{\lambda}_{k,(t_1,t_2)} \geq \theta_h\right\} \tag{3.23}$$

  where $\tilde{\lambda}_{k,(t_1,t_2)}$ is the relative loudness as defined in Equation 3.17.

- initialize the set $\mathcal{T}_{1,k}$ and $\mathcal{T}_{2,k}$ with the empty set $\varnothing$.

- do iteratively until $|\mathcal{T}'''_k| = \varnothing$:
  - take the index pair (t_1, t_2) with highest associated relative loudness $\tilde{\lambda}_{k,(t_1,t_2)}$ from $\mathcal{T}'''_k$ and remove it from from $\mathcal{T}'''_k$.
  - if there is no other pair (t_3, t_4) in $\mathcal{T}'''_k$ so that $|t_1 - t_3| < \theta_b$ or $|t_2 - t_4| < \theta_b$ then add t_1 to $\mathcal{T}_{1,k}$ and t_2 to $\mathcal{T}_{2,k}$.

This is now the final version of the peak picking algorithm. We have introduced many new parameters for the new pre-processing stage but they can be fixed and used for most musical pieces after some fine tuning.

Although working well this extension also has its pitfalls. It can be observed that the number of peaks found varies strongly for every iteration of the main algorithm, which makes the learning procedure unstable. On some iteration it may be even the case that no single peak is let trough. Regulating these differences needs further investigation in the causes of the variation which due to time constraints is left as part of future work. At this time as a quick solution we set the parameters in a way that enough peaks are let through because the more peaks get out unfiltered the smaller the variance becomes.

3.7 Tone Learning

Having found the onset locations of tone candidates for different instruments, we now have to determine associated loudness weights and the tone waveform. There are two main approaches on how we can do that.

The first one is a direct method which computes the loudness weights and the tone waveform solving linear equations independently for each template. It needs little computational power and is easy to implement but has two main problems which will be discussed in the respective subsection.

The other approach is an iterative method which jointly optimizes the templates, thus being more powerful, which comes at the expense of increased computational cost.

Direct Method

We begin with the loudness weights which we obtain by solving a simple linear equation system.

$$\hat{a}_{i,k,t}\hat{s}_{i,k,l} = e_{i,k,t+l-1} \tag{3.24}$$

with $t \in \mathcal{T}_{i,k}$ and $l = 1, 2, ..., D$. In order to be able to write the above formula in vector notation we introduce a new vector $\mathbf{q}_{i,k,t}$ which is built by copying a window of length D from $\mathbf{e}_{i,k}$ at location t resulting in the definition $\mathbf{q}_{i,k,t} = [e_{i,k,t}, e_{i,k,t+1}, ..., e_{i,k,t+D-1}]^T$. So we get

$$\hat{a}_{i,k,t}\hat{\mathbf{s}}_{i,k} = \mathbf{q}_{i,k,t} \tag{3.25}$$

with $t \in \mathcal{T}_{i,k}$. Assuming that the windows copied from $\mathbf{e}_{i,k}$ do not overlap we can use the Moore-Penrose pseudoinverse [22, 27] to compute an approximation of $\hat{a}_{i,k,t}$ in the least squares sense which results in

$$\hat{a}_{i,k,t} = \hat{\mathbf{s}}_{i,k}^{+}\mathbf{q}_{i,k,t} \tag{3.26}$$

where $\hat{\mathbf{s}}_{i,k}^{+}$ is the Moore-Penrose pseudoinverse of $\hat{\mathbf{s}}_{i,k}$ defined as

$$\hat{\mathbf{s}}_{i,k}^{+} = \left(\hat{\mathbf{s}}_{i,k}^{T}\hat{\mathbf{s}}_{i,k}\right)^{-1}\hat{\mathbf{s}}_{i,k}^{T} \tag{3.27}$$

Because $\hat{\mathbf{s}}_{i,k}$ is a vector the above equation can be reformulated into

$$\hat{\mathbf{s}}_{i,k}^{+} = \frac{\hat{\mathbf{s}}_{i,k}^{T}}{||\hat{\mathbf{s}}_{i,k}||^2} \tag{3.28}$$

Now we can insert the result into Equation 3.26 in order to get

$$\hat{a}_{i,k,t} = \frac{\hat{\mathbf{s}}_{i,k}^{T}\mathbf{q}_{i,k,t}}{||\hat{\mathbf{s}}_{i,k}||^2} \tag{3.29}$$

with $t \in \mathcal{T}_{i,k}$.

Note that the non-overlapping window assumption means that we assume that the same tone will not overlap in time with itself. This assumption is violated if for example some instruments of the same type will start playing the same tone at the same time. This may occur for example in classical music but for other genres this should be seldom the case. The problem here is that overlapping parts of the same tone will result in non-optimal estimates of $\hat{a}_{i,k,t}$.

For example if two loudness weights would overlap over their entire length D then the estimate will result in an optimal loudness for each weight in part but as they overlap completely they will add together to have twice the optimal loudness. Thus the optimal weight for two completely overlapping onsets of the same tone would be only the half of the calculated loudness in that example.

Now that we have the loudness weights of the onsets we can proceed to calculate the tone waveform. This is accomplished by solving Equation 3.25 for $\hat{\mathbf{s}}_{i,k}$. If we regard all $\hat{a}_{i,k,t}$ with $t \in \mathcal{T}_{i,k}$ as elements of a vector then we can use its pseudoinverse in order to solve the equations. This results in the following equation

$$\hat{\mathbf{s}}_{i,k}^T = \left(\hat{a}_{i,k,t\in\mathcal{T}_{i,k}}\right)^+ \mathbf{Q}_{i,k}^T \tag{3.30}$$

where $\mathbf{Q}_{i,k}$ is a matrix whose columns are the vectors $\mathbf{q}_{i,k,t}$ with $t \in \mathcal{T}_{i,k}$ or more formally $\mathbf{Q}_{i,k} = [\mathbf{q}_{i,k,t_1}, \mathbf{q}_{i,k,t_2}, ..., \mathbf{q}_{i,k,t_{\left|\mathcal{T}_{i,k}\right|}}]$ with $t_1, t_2, ..., t_{\left|\mathcal{T}_{i,k}\right|} \in \mathcal{T}_{i,k}$. After some reformulations and simplifications we arrive to

$$\hat{\mathbf{s}}_{i,k} = \frac{\mathbf{Q}_{i,k}\left(\hat{a}_{i,k,t\in\mathcal{T}_{i,k}}\right)}{\left\|\left(\hat{a}_{i,k,t\in\mathcal{T}_{i,k}}\right)\right\|^2} \tag{3.31}$$

So each element in $\hat{\mathbf{s}}_{i,k}$ is calculated as follows

$$\hat{s}_{i,k,l} = \frac{\sum_{t\in\mathcal{T}_{i,k}} \hat{a}_{i,k,t} q_{i,k,t,l}}{\sum_{t\in\mathcal{T}_{i,k}} \hat{a}_{i,k,t}^2} \tag{3.32}$$

with $l = 1, 2, ..., D$.

Now we do the usual normalization of $\hat{\mathbf{s}}_{i,k}$ to unity vector norm and multiply $\hat{\mathbf{a}}_{i,k}$ by the normalization constant in order to preserve loudness information.

Iterative Method

We have designed the iterative method in order to overcome two main problems of the direct method: the non-overlapping window assumption and the independent solving of each template in part. With the iterative method we can now jointly optimize all templates and are able to handle overlapping windows for the same tone.

As this method represents one tone learning step in the main algorithm, all iterations of this method will be embedded in this single learning step. As the main algorithm is iterative as well this will lead to a double-iterative complete algorithm. Unfortunately, double-iterative algorithms are very time consuming so we will have to ensure that this method will converge fast in order to get results in acceptable time.

Now that the method is made up of several iterations we will write the free parameters $\hat{a}_{i,k,t}$, $\hat{s}_{i,k,l}$ and the reconstructed signal $\hat{\mathbf{x}}_i$ as a function of the actual iteration number n which leads to the notation $\hat{a}_{i,k,t}(n)$, $\hat{s}_{i,k,l}(n)$ and $\hat{\mathbf{x}}_i(n)$. With the iteration numbering starting from one we define $\hat{a}_{i,k,t}(1)$ to be the result of the direct method for this parameter and $\hat{s}_{i,k,l}(1)$ to be either the result of a former tone learning step for this tone in the main algorithm or of the initialization as described earlier. We observe that the direct method is still used for computing $\hat{a}_{i,k,t}(1)$ as we need an initial estimate. Note that it is not possible to take this parameter from former iterations of the main, algorithm because due to the peak picking step the time indices in $\mathcal{T}_{i,k}$ and thus the non-zero values of $\hat{\mathbf{a}}_{i,k}$ change abruptly at every iteration.

Our iterative method is based on the gradient descent method which means we will have to define a cost function $\mathscr{C}_i(n)$. We will define this function in the terms of the squared reconstruction error

$$\mathscr{C}_i(n) = \frac{1}{2} \left\| \mathbf{x}_i - \hat{\mathbf{x}}_i(n) \right\|^2 \tag{3.33}$$

where $\hat{\mathbf{x}}_i(n)$ is the reconstructed signal at iteration n as defined in Equation 3.3. Note that for Equation 3.3 we use the free parameters of the actual iteration n.

The free parameters $\hat{s}_{i,k,l}$ and $\hat{a}_{i,k,t}$ are updated by adding the negative gradient defined by the first partial derivative of the cost function and each parameter in part. So we get

$$\hat{a}_{i,k,t}(n+1) = \hat{a}_{i,k,t}(n) - \eta_{\hat{a}} \Delta \hat{a}_{i,k,t}(n) \tag{3.34}$$

and

$$\hat{s}_{i,k,l}(n+1) = \hat{s}_{i,k,l}(n) - \eta_{\hat{s}} \Delta \hat{s}_{i,k,l}(n) \tag{3.35}$$

where $0 < \eta_{\hat{a}} < 1$ and $0 < \eta_{\hat{s}} < 1$ are learning parameters and $\Delta \hat{a}_{i,k,t}(n)$ and $\Delta \hat{s}_{i,k,l}(n)$ are the gradients of the respective parameters which are define by

$$\Delta \hat{a}_{i,k,t}(n) = \frac{\partial \mathscr{C}_i(n)}{\partial \, \hat{a}_{i,k,t}(n)} \tag{3.36}$$

and

$$\Delta \hat{s}_{i,k,l}(n) = \frac{\partial \mathscr{C}_i(n)}{\partial \, \hat{s}_{i,k,l}(n)} \tag{3.37}$$

For the partial derivative of $\hat{a}_{i,k,t}(n)$ we get

$$\frac{\partial \mathscr{C}_i(n)}{\partial \, \hat{a}_{i,k,t}(n)} = \sum_{l=1}^{T-t+1} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n) \right) \frac{\partial \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n) \right)}{\partial \, \hat{a}_{i,k,t}(n)} \tag{3.38}$$

by applying the chain rule and then after solving the last partial derivatives and some simplifications we arrive to

$$\frac{\partial \mathscr{C}_i(n)}{\partial \, \hat{a}_{i,k,t}(n)} = -\sum_{l=1}^{D} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n) \right) \hat{s}_{i,k,l}(n) \tag{3.39}$$

Analogously for the partial derivative of $\hat{s}_{i,k,l}(n)$ we get

$$\frac{\partial \mathscr{C}_i(n)}{\partial \, \hat{s}_{i,k,l}(n)} = \sum_{t=1}^{T} \left(x_{i,t} - \hat{x}_{i,t}(n) \right) \frac{\partial \left(x_{i,t} - \hat{x}_{i,t}(n) \right)}{\partial \, \hat{s}_{i,k,l}(n)} \tag{3.40}$$

by applying the chain rule and then after solving the last partial derivatives and some simplifications we arrive to

$$\frac{\partial \mathscr{C}_i(n)}{\partial \, \hat{s}_{i,k,l}(n)} = -\sum_{t \in \mathcal{T}_{i,k}} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n) \right) \hat{a}_{i,k,t}(n) \tag{3.41}$$

After plugging the results for the partial derivatives into the update formulas we finally get

$$\hat{a}_{i,k,t}(n+1) = \hat{a}_{i,k,t}(n) + \eta_{\hat{a}} \sum_{l=1}^{D} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n) \right) \hat{s}_{i,k,l}(n) \tag{3.42}$$

and

$$\hat{s}_{i,k,l}(n+1) = \hat{s}_{i,k,l}(n) + \eta_{\hat{s}} \sum_{t \in \mathcal{T}_{i,k}} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n) \right) \hat{a}_{i,k,t}(n) \tag{3.43}$$

The learning parameters $\eta_{\hat{a}}$ and $\eta_{\hat{s}}$ shall be chosen small enough so that the algorithm will not diverge. This can usually be done by trying different values in different runs.

After some preliminary testing we found that this algorithm has a too slow convergence behaviour. As we observed afterwards this is a common problem of gradient descent methods and therefore some improvements have been suggested in literature.

We first begin with the simple improvement called the momentum [32, 29]. The problem of the plain gradient descent method is that it often begins to oscillate, thus slowing convergence significantly. The idea behind the momentum is to introduce some kind of inertia in order to damp these oscillations making the algorithm pursue a straighter path to the next minimum. Another side-effect of the momentum is that it gets easier for the algorithm to escape local minima due to its inertia.

The modifications required to introduce the momentum term affect the gradients themselves, which then become

$$\Delta \hat{a}_{i,k,t}(n) = \frac{\partial \mathscr{C}_i(n)}{\partial \, \hat{a}_{i,k,t}(n)} + \alpha_{\hat{a}} \Delta \hat{a}_{i,k,t}(n-1) \tag{3.44}$$

and

$$\Delta\hat{s}_{i,k,l}(n) = \frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{s}_{i,k,l}(n)} + \alpha_{\hat{s}}\Delta\hat{s}_{i,k,l}(n-1) \tag{3.45}$$

where $\alpha_{\hat{a}}$ and $\alpha_{\hat{s}}$ are free momentum parameters in the range between zero and one just like $\eta_{\hat{a}}$ and $\eta_{\hat{s}}$. For the first iteration we define $\Delta\hat{a}_{i,k,t}(0) = 0$ and $\Delta\hat{s}_{i,k,l}(0) = 0$. The final update rules including the momentum term now become

$$\hat{a}_{i,k,t}(n+1) = \hat{a}_{i,k,t}(n) + \eta_{\hat{a}} \sum_{l=1}^{D} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n)\right) \hat{s}_{i,k,l}(n) + \alpha_{\hat{a}}\Delta\hat{a}_{i,k,t}(n-1) \tag{3.46}$$

and

$$\hat{s}_{i,k,l}(n+1) = \hat{s}_{i,k,l}(n) + \eta_{\hat{s}} \sum_{t\in\mathcal{T}_{i,k}} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n)\right) \hat{a}_{i,k,t}(n) + \alpha_{\hat{s}}\Delta\hat{s}_{i,k,l}(n-1) \tag{3.47}$$

Using high values for the momentum parameters speeds up convergence as expected but at the same time the final error levels on a higher value meaning that the algorithm is trapped more easily in local minima while on the other hand when using low values only a small convergence speed-up can be noticed.

This led us to choose a further extension called Super-SAB [34] which is itself an extension of the self-adaptive backpropagation algorithm (SAB) described in [7, 17]. The adaptation is done in a way which speeds up search on plateaus on the error surface where gradients are near zero due to the flatness and slows down on steep areas in order to not shoot over the minimum. More precisely, if the sign of the partial derivate for a parameter changes sign after one iteration then it means that a minimum was overshoot and so the learning rate for that parameter is decreased, the previous update is undone and the new gradient is set to zero in order to not interfere as a momentum term in future updates. On the other hand if the sign remains the same for two consecutive updates then the learning parameter is increased to speed up learning.

The Super-SAB method implies that every parameter has its own learning parameter $\eta_{\hat{a}_{i,k,t}}(n)$ and $\eta_{\hat{s}_{i,k,l}}(n)$ respectively which are adapted at every iteration. The learning parameters can be initialized with any values in the range $(0..1]$ as it does not matter anymore because the parameters will be exponentially increased or decreased until a step forward can be done. However, the learning parameters are not eliminated this way as we now need to specify the constants for increasing the learning rate and decreasing it which we call η_+ and η_- respectively. As a rule of thumb these two new parameters shall not be equal, be greater than one and the increasing parameter shall be smaller than the decreasing one. We found that good choices are $\eta_+ = \sqrt{2}$ and $\eta_- = 1/2$. A smaller η_+ will lead to a slowdown in convergence and a higher η_+ usually makes the gradient change sign very often and thus also slows convergence down but this time through oscillation.

The algorithm for Super-SAB works as follows:

- initialize $\Delta\hat{a}_{i,k,t}(0)$ and $\Delta\hat{s}_{i,k,l}(0)$ with zero.
- do the following till convergence
  - calculate the partial derivative for $a_{i,k,t}(n)$ according to Equation 3.39 and $s_{i,k,l}(n)$ according to Equation 3.41.
  - if
    $$\frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{a}_{i,k,t}(n)} \frac{\partial \mathscr{C}_i(n-1)}{\partial\, \hat{a}_{i,k,t}(n-1)} > 0 \tag{3.48}$$
    calculate the gradient according to Equation 3.44, update $\hat{a}_{i,k,t}$
    $$\hat{a}_{i,k,t}(n+1) = \hat{a}_{i,k,t}(n) - \eta_{\hat{a}_{i,k,t}}(n)\Delta\hat{a}_{i,k,t}(n) \tag{3.49}$$
    and increase the associated learning parameter
    $$\eta_{\hat{a}_{i,k,t}}(n+1) = \eta_{+}\eta_{\hat{a}_{i,k,t}}(n) \tag{3.50}$$
  - if
    $$\frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{a}_{i,k,t}(n)} \frac{\partial \mathscr{C}_i(n-1)}{\partial\, \hat{a}_{i,k,t}(n-1)} < 0 \tag{3.51}$$
    decrease the learning parameter for $\hat{a}_{i,k,t}$
    $$\eta_{\hat{a}_{i,k,t}}(n+1) = \eta_{-}\eta_{\hat{a}_{i,k,t}}(n) \tag{3.52}$$
    and set $\Delta\hat{a}_{i,k,t}(n) = 0$ to avoid interference in the next update in form of a momentum term.
  - if
    $$\frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{s}_{i,k,l}(n)} \frac{\partial \mathscr{C}_i(n-1)}{\partial\, \hat{s}_{i,k,l}(n-1)} > 0 \tag{3.53}$$
    calculate the gradient according to Equation 3.45, update $\hat{s}_{i,k,l}$
    $$\hat{s}_{i,k,l}(n+1) = \hat{s}_{i,k,l}(n) - \eta_{\hat{s}_{i,k,l}}(n)\Delta\hat{s}_{i,k,l}(n) \tag{3.54}$$
    and increase the associated learning parameter
    $$\eta_{\hat{s}_{i,k,l}}(n+1) = \eta_{+}\eta_{\hat{s}_{i,k,l}}(n) \tag{3.55}$$
  - if
    $$\frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{s}_{i,k,l}(n)} \frac{\partial \mathscr{C}_i(n-1)}{\partial\, \hat{s}_{i,k,l}(n-1)} < 0 \tag{3.56}$$
    decrease the learning parameter for $\hat{s}_{i,k,l}$
    $$\eta_{\hat{s}_{i,k,l}}(n+1) = \eta_{-}\eta_{\hat{s}_{i,k,l}}(n) \tag{3.57}$$
    and set $\Delta\hat{s}_{i,k,l}(n) = 0$ to avoid interference in the next update in form of a momentum term.

This approach works now as expected but it turned out to have a significant problem: due to the differing learning speeds of each $\hat{s}_{i,k,l}$ belonging to the same, tone some discrepancies between neighbouring $\hat{s}_{i,k,l}$ occurred which could be heard as noise in the reconstructed waveform. Because the noise is not related to any frequencies occurring in the input signal it becomes disturbing. It is even more disturbing than the usual pollution of the templates caused by lack of separation, because the pollution consists of the tones which are actually played during the musical piece which usually have harmonic character. Unfortunately this noise is unacceptable as the reconstructed input sounds better when using only the simple momentum extension so we had to drop this extension and look for something else.

So we finally came across the Newton method which differs from the simple gradient descent method by taking the curvature information into account. As the speedup may be significant we decided to give it a try.

The curvature information is calculated by the second derivative of the cost function. This means that in our case the update formulas get a second order gradient operator written as Δ^2. The new formulas then become

$$\hat{a}_{i,k,t}(n+1) = \hat{a}_{i,k,t}(n) - \eta_{\hat{a}}(n,m)\frac{\Delta\hat{a}_{i,k,t}(n)}{\Delta^2\hat{a}_{i,k,t}(n)} \tag{3.58}$$

and

$$\hat{s}_{i,k,l}(n+1) = \hat{s}_{i,k,l}(n) - \eta_{\hat{s}}(n,m)\frac{\Delta\hat{s}_{i,k,l}(n)}{\Delta^2\hat{s}_{i,k,l}(n)} \tag{3.59}$$

where the two gradients $\Delta\hat{a}_{i,k,t}(n)$ and $\Delta\hat{s}_{i,k,l}(n)$ are defined in Equation 3.36and 3.37 which do not contain the momentum term and m is the actual refinement iteration which will be discussed later. The second order gradients are defined as

$$\Delta^2\hat{a}_{i,k,t}(n) = \frac{\partial^2 \mathscr{C}_i(n)}{\partial\,\hat{a}^2_{i,k,t}(n)} \tag{3.60}$$

and

$$\Delta^2\hat{s}_{i,k,l}(n) = \frac{\partial^2 \mathscr{C}_i(n)}{\partial\,\hat{s}^2_{i,k,l}(n)} \tag{3.61}$$

For the second order partial derivative of $\hat{a}_{i,k,t}(n)$ we first get

$$\frac{\partial^2 \mathscr{C}_i(n)}{\partial\,\hat{a}^2_{i,k,t}(n)} = -\frac{\partial \sum_{l=1}^{D}\left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n)\right)\hat{s}_{i,k,l}(n)}{\partial\,\hat{a}_{i,k,t}(n)} \tag{3.62}$$

and after solving the remaining partial derivative we get

$$\frac{\partial^2 \mathscr{C}_i(n)}{\partial\,\hat{a}^2_{i,k,t}(n)} = \sum_{l=1}^{D}\hat{s}^2_{i,k,l}(n) \tag{3.63}$$

Analogously for the second order partial derivative of $s_{i,k,l}(n)$ we first get

$$\frac{\partial^2 \mathscr{C}_i(n)}{\partial\, \hat{s}^2_{i,k,l}(n)} = -\frac{\sum_{t\in\mathcal{T}_{i,k}} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n)\right) \hat{a}_{i,k,t}(n)}{\partial\, \hat{s}_{i,k,l}(n)} \tag{3.64}$$

and after solving the remaining partial derivative we get

$$\frac{\partial^2 \mathscr{C}_i(n)}{\partial\, \hat{s}^2_{i,k,l}(n)} = \sum_{t\in\mathcal{T}_{i,k}} \hat{a}^2_{i,k,t}(n) \tag{3.65}$$

Plugging the results into the new update formulas we finally get

$$\hat{a}_{i,k,t}(n+1) = \hat{a}_{i,k,t}(n) + \eta_{\hat{a}}(n,m)\frac{\sum_{l=1}^{D} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n)\right) \hat{s}_{i,k,l}(n)}{\sum_{l=1}^{D} \hat{s}^2_{i,k,l}(n)} \tag{3.66}$$

and

$$\hat{s}_{i,k,l}(n+1) = \hat{s}_{i,k,l}(n) + \eta_{\hat{s}}(n,m)\frac{\sum_{t\in\mathcal{T}_{i,k}} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n)\right) \hat{a}_{i,k,t}(n)}{\sum_{t\in\mathcal{T}_{i,k}} \hat{a}^2_{i,k,t}(n)} \tag{3.67}$$

Note that the second order partial derivative used here is a simple interpretation of the second order gradient. Newton's method in optimization can also be used with the whole Hessian matrix but in order to keep the algorithm simple and fast we decided to take the simple interpretation of the second order partial derivative.

Now the new equation prove to be rather unstable at the beginning therefore requiring very small $\eta_{\hat{a}}(n,m)$ and $\eta_{\hat{s}}(n,m)$. This is counterproductive as we expect a speedup from this extension. Fortunately there is a simple solution to that. We can adapt the learning parameter by doing a fixed number of refinement iterations. During such an iteration we try out how the actual learning parameter affects the cost function basically doing a line search for the best $\eta_{\hat{a}}(n,m)$ and $\eta_{\hat{s}}(n,m)$.

The line search has a simple logic. If the cost function gets smaller than in the former refinement iteration and the learning parameter was increased then the actual learning parameter is increased by a fixed constant η_+. The same happens if the cost function gets bigger and the learning parameter was decreased. Analogously if the cost function gets smaller and the learning parameter was decreased then the actual learning parameter is decreased by a fixed constant η_-. The last possibility with increasing cost function and increasing learning parameter during the former refinement iteration leads to the decrease of the actual learning parameter. Empirically a good choice for η_+ seems to be $\sqrt{2}$ like in the Super-SAB algorithm and $1/2$ for η_+. The initial learning parameters $\eta_{\hat{a}}(n,1)$ and $\eta_{\hat{s}}(n,1)$ are either taken over from the former iteration of the gradient descent method or if it is the first iteration then take the values from the former iteration of the main algorithm. At the very beginning $\eta_{\hat{a}}(1,1)$ and $\eta_{\hat{s}}(1,1)$ are initialized with some small constants in the range $(0..1]$.

The full algorithm for Newton's method works as follows:

- if this is the first iteration of the main algorithm, initialize $\eta_{\hat{a}}(1,1)$ and $\eta_{\hat{s}}(1,1)$ with some small constants in the range $(0..1]$.
- do the following until convergence
  1. initialize $\eta_{\hat{a}}(n,1)$ and $\eta_{\hat{s}}(n,1)$ with the learning parameters of the former iteration or if it is the first one take the values from the former iteration of the main algorithm.
  2. calculate the first and second order gradients for the parameters $\hat{a}_{i,k,t}(n)$ and $\hat{s}_{i,k,l}(n)$ according to Equations 3.36, 3.39, 3.60, 3.63 and 3.37, 3.41, 3.61, 3.7 respectively.
  3. iterate M times using the iteration counter $m = 1..M$
     - update $\hat{a}_{i,k,t}$ according to Equation 3.66.
     - if this is the first refinement iteration
       - if $\mathscr{C}_i(n+1) < \mathscr{C}_i(n)$ increase the learning parameter by η_+

$$\eta_{\hat{a}}(n+1) = \eta_+ \eta_{\hat{a}}(n) \tag{3.68}$$

       - else decrease it by η_-

$$\eta_{\hat{a}}(n+1) = \eta_- \eta_{\hat{a}}(n) \tag{3.69}$$

       and undo the update by setting $\hat{a}_{i,k,t}(n+1) = \hat{a}_{i,k,t}(n)$.
     - else
       - if $\mathscr{C}_i(n+1) > \mathscr{C}_i(n) \wedge \eta_{\hat{s}}(n,m+1) < \eta_{\hat{a}}(n,m)$ or $\mathscr{C}_i(n+1) < \mathscr{C}_i(n) \wedge \eta_{\hat{a}}(n,m+1) > \eta_{\hat{s}}(n,m)$ increase the learning parameter by η_+

$$\eta_{\hat{a}}(n+1) = \eta_+ \eta_{\hat{a}}(n) \tag{3.70}$$

       - else decrease the learning parameter by η_-

$$\eta_{\hat{a}}(n+1) = \eta_- \eta_{\hat{a}}(n) \tag{3.71}$$

       and undo the update by setting $\hat{a}_{i,k,t}(n+1) = \hat{a}_{i,k,t}(n)$.
  4. analogously refine the learning parameter for $\hat{s}_{i,k,l}$ as in step 3 using the appropriate variables and equations.

Due to the common learning parameters for the tone waveform samples $\hat{s}_{i,k,l}$ the noise which could be observed for Super-SAB is now eliminated. Convergence speed is also good therefore we take it for the tone learning step.

Still some further improvements can be done. For example, the onset loudness weights $\hat{a}_{i,k,t}$ do not need to have a common learning parameter so we could extend our implementation of

Newton's method to have a individual learning parameter for each loudness weight which is then adjusted by the Super-SAB algorithm. Furthermore, we could use a more general interpretation of the second order gradient leading to more precise curvature information, and so on.

3.8 Fine Tuning

After finding and learning the templates we see that these two steps may need some fine tuning in order to provide higher quality templates. This fine tuning is not essential as the main two steps will also deliver satisfactory results without it. But as improving the main steps may cost a lot of time we can build some fine tuning steps which quickly improve results.

Template Offsetting

One of the fine tuning steps is to offset the template in such a way that it covers the part of the tone with most of the energy. We thought of this step as we saw that templates often cover the middle or the ending parts of the tones, thus not optimally approximating the note as most of the energy is usually found in the onset part and in the middle. The circumstance of the false coverage can be explained by the inflexibility of the algorithm not being able to move the templates relatively to their match in the input signal once they have specialized on a tone. So even if a template has learnt the tone and can find all its locations during the musical piece it may sound weird because of the absence of the onset.

If we analyze this problem more deeply we see that if the initialization does not lead the template to the onset and middle part of the tone it later specializes on, it is later not possible to move that template relatively to the tone. This is because during the tone finding step we only search for parts in the input signal which match the template. So if the template matches the wrong parts of the tone then the searching step will return only the positions of these parts.

In order to solve this problem we thought of an algorithm which finds the optimal offset according to some criterion so the template can cover the energy rich parts of its associated tone. The algorithm proceeds as follows:

- compute a new template $\hat{\mathrm{s}}'_{i,k}$ using an enlarged window size of $2D$ centred around the original window content and the usual learning step scaled for the new window size.

- slice the new template into $\theta_c \in \mathbb{N}$ slices $\hat{\mathbf{s}}'_{1,i,k}, \hat{\mathbf{s}}'_{2,i,k}, ..., \hat{\mathbf{s}}'_{\theta_c,i,k}$ where θ_c is a free parameter which shall be divisible by 4.
- calculate the loudness weights $a'_{1,i,k,t-D/2}, \hat{a}'_{2,i,k,t-D/2+2D/\theta_c}, \dots, \hat{a}'_{\theta_c,i,k,t-D/2+2(\theta_c-1)D/\theta_c}$ with $t \in \mathcal{T}_{i,k}$ for each slice separately using the usual learning step scaled for the window size of the slice $2D/\theta_c$.
- compute for each slice a total loudness coefficient

  $$\lambda_m = \left\|\hat{\mathbf{s}}'_{m,i,k}\right\| \sum_{t \in \mathcal{T}_{i,k}} \hat{a}'_{m,i,k,t} \tag{3.72}$$

  with $m = 1, 2, ..., \theta_c$.
- using a moving window of $\frac{1}{2}\theta_c$ slices which moves by one slice, calculate the total loudness of each window by summing over all λ_m that are part of that window.
- find the window with the highest total loudness and initialize the tone waveform $\hat{\mathbf{s}}_{i,k}$ with the slices $\hat{\mathbf{s}}'_{m,i,k}$ that are part of that window. Offset the onsets to match the beginning of the window and also update the time indices in $\mathcal{T}_{i,k}$. Finally, recalculate the onset loudness weights using the usual learning step.

We note that θ_c has to be divisible by 4 in order to ensure that also the original template position is considered.

In other words, this algorithm calculates a twice as big template and then searches for a window of the original size which has maximum energy and onset loudness weights in order to reinitialize the template and recalculate the loudness weights to match the new template finally updating $\mathcal{T}_{i,k}$ to reflect the new time indices.

We have also thought about a similar algorithm which is iterative and therefore more expensive in terms of required computational power but is simpler to describe and implement.

- iteratively do the following until convergence
  - compute $\theta_c \in \mathbb{N}$ alternative templates $\hat{\mathbf{s}}'_{1,i,k}, \hat{\mathbf{s}}'_{2,i,k}, \dots, \hat{\mathbf{s}}'_{\theta_c,i,k}$ centred around the original template position. θ_c shall be chosen to be divisible by 4 in order to ensure that also the original template position is considered.
  - compute the corresponding alternative loudness weights $\hat{a}'_{1,i,k,t-D/2}, \hat{a}'_{2,i,k,t-D/2+2D/\theta_c}, \dots, \hat{a}'_{\theta_c,i,k,t-D/2+2(\theta_c-1)D/\theta_c}$ with $t \in \mathcal{T}_{i,k}$.
- search for the alternative m with the highest total loudness

  $$\lambda_m = \sum_{t \in \mathcal{T}_{i,k}} \hat{a}'_{m,i,k,t-D/2+2(m-1)D/\theta_c} \tag{3.73}$$

and take it as the new template. Also update $\mathcal{T}_{i,k}$ to reflect the new offset.

In other words, we now search for alternative templates by learning new templates at positions around the old one and picking the alternative template which has the highest associated total loudness. We then reinitialize the original template with the new one and update the set of indices $\mathcal{T}_{i,k}$ to reflect the changes.

During preliminary testing of the first offsetting algorithm we saw some downsides unfortunately. First, the templates tend to move very often and do rather seldom stand still. This in turn hinders convergence of the main algorithm. Another issue is that the templates begin to collide, specializing on the same tone or forming tone beginning-ending pairs. The latter could be useful on long tones with high energy content but the former is surely a non-desirable result. The collision problem could probably be solved by introducing a penalty term for onset proximity to other tones but unfortunately we were short of time to investigate this further.

Due to the problems above we finally decided to drop the template offsetting, though given more time it might be improved in order to become more useful.

Phase Matching

After the tone search step we get the set of discrete time indices $\mathcal{T}_{i,k}$ as the result. This means that the templates are matched with the input signal at discrete time locations with sample accuracy meaning that the real match may lie in a ±half sample-length neighbourhood. Using a match with that accuracy leads to templates being attenuated in high frequency range. This is caused by the summation in the direct and iterative tone learning method over several occurrences of a template in the time domain which is a main point of the template algorithm in eliminating interference by applying the sum over occurrences of the same tone and making interference cancel itself if the interference is assumed to be some kind of noise and the tone is assumed not to change over all occurrences.

The sample accuracy now means that in reality the summation is done over shifted occurrences of the same tone where the shift is at most $\pm 1/2$ sample length. The shift translates in a phase shift in the frequency domain. The shift is small for low frequencies as their duration is much longer than the sample shift thus making the shift being only a small fraction of the duration. But due to the shorter duration of the higher frequencies their phase shift becomes higher topping at a maximum of $\pm 90^\circ$ at the Nyquist frequency. Assuming that the error between the real match and the sample accurate match is equally distributed in the range of $[-1/2$ sample length..$1/2$ sample length$]$ we may run the sum over tone occurrences having their Nyquist frequencies shifted 180° apart thus cancelling themselves despite belonging to the same tone.

This problem is less pronounced on frequencies lower than Nyquist because they can not fully cancel themselves but only be attenuated to a certain degree as their maximum relative phase shift sinks below 180°.

Now a significant error in the reconstructed output is introduced due to the lack of high frequencies and consequently also the separation quality is degraded. This happens even if all other parts of the separation algorithm would give perfect results therefore we have to consider this problem to be very important and have to solve it.

So in order to overcome the attenuation problem we need to render the template match more accurate. The more accurate the match gets the lower the attenuation at the high frequencies will get. There are two solutions for getting a better accuracy.

The first one is to simply upsample the input signal and the template to a higher frequency and work only on the upsampled signals. After the upsampling we will not have any frequencies higher than half the original sampling rate thus we will only have frequencies lower than the half the new sampling rate which will be now less attenuated. Or viewed differently we have moved the Nyquist frequency farther up in the frequency range thus lowering the attenuation on the original frequencies which remained the same. This solution is very easy to implement as it needs only an upsampling filter all other things being unchanged which is the reason we decided to take it into the final version of the algorithm.

For the upsampling filter we take the Lanczos interpolator [35] which is defined as

$$\mathrm{lanczos}(z;\theta_l) = \begin{cases} \frac{\sin(\pi z)}{\pi z}\frac{\sin(\pi \frac{z}{\theta_l})}{\pi \frac{z}{\theta_l}} & |z| < \theta_l \\ 0 & |z| \geq \theta_l \end{cases} \tag{3.74}$$

where $\theta_l \in \mathbb{N}$ is a parameter describing the order of the filter and z is the offset around the actual sample. In words the Lanczos filter is a windowed sinc function where the windows itself is a sinc function with $\theta_l - 1$ sidelobes. The more sidelobes are allowed the better the filter works. We found that a filter with two sidelobes, that is with $\theta_l = 3$, is already enough or our purposes.

In order to do the upsampling the centre of the Lanczos filter has to be positioned at a new raster which will produce the upsampled signal thus involving the use of shifted versions of the filter. To illustrate how this step works let us consider for example an upsampling factor of two like in Figure 3.2. The new sampling raster will include all old positions and new ones which lie in between the old ones. For the old ones we use no shift thus centring the filter around each sample. Applying the filter will produce the old ones as the filter function will have the samples positioned exactly at the zero crossings. For the samples in between the shift will be of half a sample which will result into a non-trivial weighting of the input

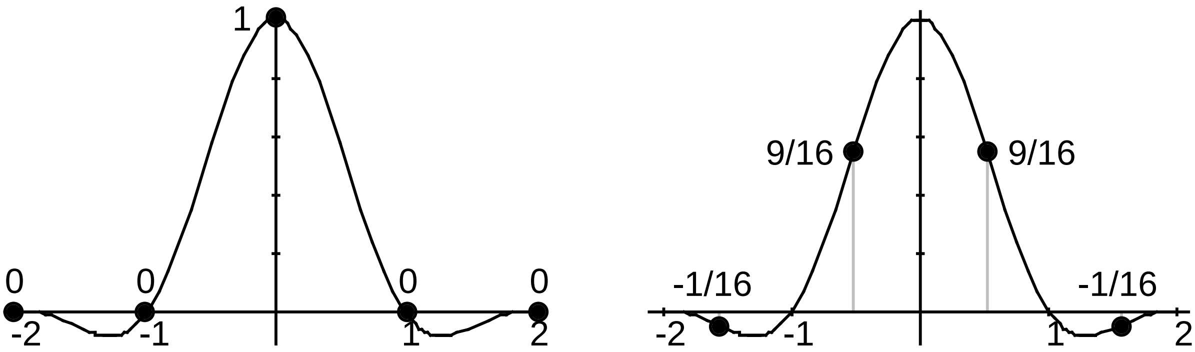

Figure 3.2 The zero sample-offset version of the Lanczos filter with one sidelobe as used for factor 2 upsampling is shown on the right and on the left the 1/2 sample-offset version. Note that the zero offset version only copies the value of the input sample. *Source: [35]*

samples. At this point we shall note that for some upsampling factors it is well possible that the sample weights for some shifts will not sum to 1 resulting into severe interpolation artefacts. The problem is usually handled by normalizing the coefficients for each shift to sum to 1.

The second solution is to better align every match by phase shifting the input signal at that place until it maximally correlates with the template. This solution can be implemented as a refinement of the peak generation step during the tone search. Here we can first generate the peaks and then use a refinement step for each peak in part which searches for a better match by phase shifting the input signal. The best match can be found by a simple line search algorithm as the problem is one-dimensional. Furthermore, because we are searching in a range of only $\pm 1/2$ samples and the maximum frequency representable by the input signal lies at Nyquist there cannot be any other peaks in that range. This guarantees that the search will find the global optimum for the alignment problem.

The phase shifting at the input signal is done using the Lanczos filter analogue to the upsampling described earlier. The filter is now used with a fixed predefined phase shift and the output raster has the same distance between the sample points.

Compared to the first solution approach we can now get an arbitrarily accurate match until we reach the noise floor without increasing the memory requirements too dramatically as we do not have to interpolate and store the whole input signal. Unfortunately, there is also a big downside: there are usually so many peaks found during the first stage that we might easily have to do more computing steps than with the first approach. The unfiltered peaks are often very close by each other so that their distance is only a very small fraction of the template size. So having many peaks makes the matching inefficient compared to first upsampling the input signal and then matching using the fast FFT correlation method. In order to illustrate this point imagine that each second sample has a peak in the correlation function which is a worst case scenario. The complexity of the matching algorithm will be $\mathcal{O}(TD)$ because the number of peaks is proportional to the input size T and the template length D. We note that this is exactly the complexity of the inefficient correlation algorithm described in the Correlation section earlier. But if we upsample the input signal first then the complexity will be reduced to $\mathcal{O}\left(\theta_u T \log_2 \theta_u T\right)$ where θ_u is the upsampling factor. As θ_u is much smaller than the template length we have a substantial advantage in speed when using the first solution. It is this unforgivable rise in computational complexity which drove us away from taking the second solution in the final algorithm.

A theoretical third solution would also exist but we had no time to look closer into it. We could do the entire separation in the frequency domain. Matching in the frequency domain would have a much lower accuracy of window-length only but if we would take the mean over tone occurrences using only the power spectrum and ignoring the phases, no cancellation effects of the frequencies belonging to the same tone would occur. Still, frequencies belonging to other tones are expected to be attenuated towards a noise floor. This means that every other part of the main algorithm would work in the frequency domain too, but it would have taken too much time to implement so we have considered it part of future work.

3.9 Final Algorithm

In this section many versions of the sub-algorithms were presented where some versions did not work out as expected and were excluded from the final algorithm and other were expanded and improved. Therefore we though we should include a complete final version of the algorithm which will be composed of only those sub-algorithms which actually proved to work.

- upsample the input signal $\mathbf{x}_i$ by an upsampling factor of θ_u with a Lanczos filter of order θ_l. Rescale the affected template parameters to fit the new sample frequency.
- initialize all onset vectors $\hat{\mathbf{a}}_{i,k}$ and tone vectors $\hat{\mathbf{s}}_{i,k}$ with zero.
- do iteratively until maximum number of iterations reached
    - calculate the reconstructed signal

$$\hat{\mathbf{x}}_i = \sum_{k=1}^{K} \hat{\mathbf{a}}_{i,k} \star \hat{\mathbf{s}}_{i,k} \tag{3.75}$$

    - do for each template k

- calculate the individualized reconstruction error signal

$$\mathbf{e}_{i,k} = \mathbf{x}_i - \hat{\mathbf{x}}_i + \hat{\mathbf{a}}_{i,k} \star \hat{\mathbf{s}}_{i,k} \tag{3.76}$$

- if this is the first iteration
  - compute the mean loudness λ for the whole input

$$\lambda = \frac{1}{\sqrt{T}} \left\| \mathbf{e}_{i,k} \right\| \tag{3.77}$$

  - divide the input into slices of size D and calculate the mean loudness λ_m for each slice $m = 1, 2, ..., \lfloor T/D \rfloor$.

$$\lambda_m = \frac{1}{\sqrt{D}} \left\| \mathbf{x}_{m,i} \right\| \tag{3.78}$$

    where $\mathbf{x}_{m,i}$ is the vector of slice m.
  - randomly pick a slice m' until $\lambda_{m'} > \lambda$.
  - initialize the tone waveform $\hat{\mathbf{s}}_{i,k}$ with $\mathbf{x}_{m',i}$.
  - initialize the onset vector with $\left\| \hat{\mathbf{s}}_{i,k} \right\|^2$ on the time location where the slice m' begins and set the rest of it to zero.
  - divide the tone waveform by $\left\| \hat{\mathbf{s}}_{i,k} \right\|^2$ in order to normalize it to unity vector norm.
- calculate the unnormalized correlation

$$\mathbf{r}'_{i,k} = \mathbf{e}_{i,k} \times \hat{\mathbf{s}}_{i,k} \tag{3.79}$$

  and the normalized one

$$r_{i,k,t} = \frac{r'_{i,k,t}}{\sqrt{\sum_{l=1}^{D} e^2_{i,k,t+l-1}}} \tag{3.80}$$

- generate the set of time indices

$$\mathcal{T}_{i,k} = \{ t \mid r_{i,k,t} > 0 \land \Delta r_{i,k,t-1} \geq 0 \land \Delta r_{i,k,t} \leq 0 \} \tag{3.81}$$

  where $\Delta r_{i,k,t}$ is defined in Equation 3.13.
- generate a set $\mathcal{T}_k$ of all time index pairs (t_1, t_2) with $t_1 \in \mathcal{T}_{1,k}, t_2 \in \mathcal{T}_{2,k}$ where each pair is restricted to have a sample-shift below the threshold θ_e, or more formally

$$\mathcal{T}_k = \{ (t_1, t_2) \mid t_1 \in \mathcal{T}_{1,k},\ t_2 \in \mathcal{T}_{2,k},\ t_1 - \theta_e < t_2 < t_1 + \theta_e \} \tag{3.82}$$

  Note that the time indices may have been duplicated and can now occur in more than one pair. There may now be even more pairs than there were time indices before.

  - calculate the mean sample-shift deviation $\bar{z}_k$

$$\bar{z}_k = \frac{1}{|\mathcal{T}_k|} \sum_{(t_1,t_2)\in\mathcal{T}_k} t_1 - t_2 \tag{3.83}$$

  - generate a new set $\mathcal{T}'_k$ containing only those peak pairs from $\mathcal{T}_k$ whose difference between time indices is less than θ_f where $\theta_f \in \mathbb{N}$ is freely adjustable parameter. More formally

$$\mathcal{T}'_k = \{(t_1, t_2) \mid (t_1, t_2) \in \mathcal{T}_k,\ |t_1 - t_2 - \bar{z}_k| < \theta_f\} \tag{3.84}$$

  - calculate the mean loudness difference $\bar{\lambda}_k$

$$\bar{\lambda}_k = \frac{1}{|\mathcal{T}'_k|} \sum_{(t_1,t_2)\in\mathcal{T}'_k} \mathring{\lambda}_{k,(t_1,t_2)} \tag{3.85}$$

    where $\mathring{\lambda}_{k,(t_1,t_2)}$ is the loudness difference as defined in Equation 3.16.

  - constrain the normalized loudness difference $\mathring{\lambda}_{k,(t_1,t_2)}$ with $(t_1, t_2) \in \mathcal{T}'_k$ to have a vary in the range of $[-\theta_g..\theta_g]$ by generating the set $\mathcal{T}''_k$

$$\mathcal{T}''_k = \left\{(t_1, t_2) \mid (t_1, t_2) \in \mathcal{T}'_k,\ \left|\mathring{\lambda}_{k,(t_1,t_2)} - \bar{\lambda}_k\right| < \theta_g\right\} \tag{3.86}$$

  - enforce the minimum loudness θ_h by creating another set $\mathcal{T}'''_k$

$$\mathcal{T}'''_k = \left\{(t_1, t_2) \mid (t_1, t_2) \in \mathcal{T}''_k,\ \tilde{\lambda}_{k,(t_1,t_2)} \geq \theta_h\right\} \tag{3.87}$$

    where $\tilde{\lambda}_{k,(t_1,t_2)}$ is the relative loudness as defined in Equation 3.17.

  - initialize the set $\mathcal{T}_{1,k}$ and $\mathcal{T}_{2,k}$ with the empty set $\varnothing$.
  - do iteratively until $|\mathcal{T}'''_k| = \varnothing$:
    - take the index pair (t_1, t_2) with highest associated relative loudness $\tilde{\lambda}_{k,(t_1,t_2)}$ from $\mathcal{T}'''_k$ and remove it from from $\mathcal{T}'''_k$.
    - if there is no other pair (t_3, t_4) in $\mathcal{T}'''_k$ so that $|t_1 - t_3| < \theta_b$ or $|t_2 - t_4| < \theta_b$ then add t_1 to $\mathcal{T}_{1,k}$ and t_2 to $\mathcal{T}_{2,k}$.

- if this is the first iteration initialize $\eta_{\hat{a}}(1, 1)$ and $\eta_{\hat{s}}(1, 1)$ with some small constants in the range $(0..1]$.
- do the following until convergence
  1. initialize $\eta_{\hat{a}}(n, 1)$ and $\eta_{\hat{s}}(n, 1)$ with the learning parameters of the former iteration or if it is the first one take the values from the former iteration of the main algorithm.
  2. calculate the first and second order gradients for the parameters $\hat{a}_{i,k,t}(n)$ and $\hat{s}_{i,k,l}(n)$ according to Equations 3.36, 3.39, 3.60, 3.63 and 3.37, 3.41, 3.61, 3.7 respectively.

3. iterate M times using the iteration counter $m = 1..M$
    - update $\hat{a}_{i,k,t}$ according to Equation 3.66.
    - if this is the first refinement iteration
        - if $\mathscr{C}_i(n+1) < \mathscr{C}_i(n)$ increase the learning parameter by η_+

          $$\eta_{\hat{a}}(n+1) = \eta_+ \eta_{\hat{a}}(n) \tag{3.88}$$

        - else decrease it by η_-

          $$\eta_{\hat{a}}(n+1) = \eta_- \eta_{\hat{a}}(n) \tag{3.89}$$

          and undo the update by setting $\hat{a}_{i,k,t}(n+1) = \hat{a}_{i,k,t}(n)$.
    - else
        - if $\mathscr{C}_i(n+1) > \mathscr{C}_i(n) \wedge \eta_{\hat{s}}(n, m+1) < \eta_{\hat{a}}(n, m)$ or $\mathscr{C}_i(n+1) < \mathscr{C}_i(n) \wedge \eta_{\hat{a}}(n, m+1) > \eta_{\hat{s}}(n, m)$ increase the learning parameter by η_+

          $$\eta_{\hat{a}}(n+1) = \eta_+ \eta_{\hat{a}}(n) \tag{3.90}$$

        - else decrease the learning parameter by η_-

          $$\eta_{\hat{a}}(n+1) = \eta_- \eta_{\hat{a}}(n) \tag{3.91}$$

          and undo the update by setting $\hat{a}_{i,k,t}(n+1) = \hat{a}_{i,k,t}(n)$.
4. analogously refine the learning parameter for $\hat{s}_{i,k,l}$ as in step 3 using the appropriate variables and equations.

- downsample the templates by a factor of θ_u to match the sample rate of original input signal.

3.10 Summary and Future Work

In this section we have presented a template based approach to musical instrument source separation. We have begun with the introduction of the onset vector as a sparse version of the steering vector in order to save space and computation time. Then we discussed the initialization procedure, where we took randomly selected parts of the reconstruction residual whose energy was higher than the average residual energy in order to initialize the tone waveform and the onset vector.

The main algorithm was described as an iterative approach of tone searching and tone learning where the tone searching was fed an individualized reconstruction error for each template in order to minimize false detection rates caused by interference of already recognized tones.

We then discussed the tone search in detail which is made up of several steps with the first being the correlation of the template with the input signal which can be done fast by means of the FFT. Then we talked about peak generation where the positive peaks of the correlation function were indexed by their time location and stored as a set of time indices. This set was then filtered in the peak picking stage in order to keep only the most plausible peak combinations among stereo channels. This was done by taking stereo cues into account and making some assumptions about the stationarity of the instruments during play. We then presented two tone learning methods. A direct one where each tone waveform was calculated separately using the Moore-Penrose pseudoinverse and assuming non-overlapping tone occurrences in the input. The more advanced second method was based on a gradient descent formulation which was extended several times due to its notorious slow convergence leading to a form of Newton's method which also uses second-order information about the error surface.

Some fine-tuning methods were shown. First a template offsetting approach was presented where the content of the template was offset relative to every match in the input signal and newly learnt in order to capture the part of the tone carrying most of its energy. Then the importance of accurate phase matching during tone searching was discussed because the lack of it attenuates high frequencies in the tone waveforms of the templates. The phase matching was described to mean a sub-sample accurate tone search which could be achieved by two approaches. A simple and easy to implement one where the input signal is upsampled and the templates are enlarged to a higher sample frequency and a more sophisticated by computationally intensive way of refining every match in the input signal.

Finally the whole algorithm resulting from taking the versions of the sub-algorithms of each subsection which proved practically usable, was presented.

Now several questions and problems were raised and remained open during this section, which are good candidates for future work.

We begin with the initialization which proved to be an important step in the algorithm as the initial conditions have a high impact on the capability of the templates to find proper tones and their ability to separate them. A more educated guess is needed here which should eliminate the random component completely as the results become unreliable because of the variation introduced by it. Perhaps a clustering in the frequency domain according to some spatial and temporal cues as well as a repetitiveness indicator could prove useful.

The Peak picking could also be more refined. It has big variations in the number of peaks let through in every iteration. This may be caused by starting from zero in every iteration and not using any information from the results of the former iterations. An analysis of how the peaks have changed from the last iteration could give new insights in the behaviour of the algorithm.

The iterative method of tone learning could also benefit from some improvements. The onset vector and the tone waveform could be trained using different methods. Each method could then be better tailored to the vector type it is adjusting. Furthermore, the simple implementation of Newton's method could be extended to use more precise curvature information of the error surface presumably achieving an even faster convergence.

Furthermore, the template offsetting approach which was not working as expected should be more elaborated upon as it is believed to play an important role in separation performance by allowing more flexibility at choosing the best part of a tone in order to represent it as a tone waveform which is not given otherwise.

Another important problem which should be resolved in order to make the approach more complete is that of organizing the individual tones into instruments. While separation may work for tones only, without any higher semantic clustering we have a hard time evaluating the separation quality because we are not used to listening to single tones occurring sporadically the audio signal. A solution for the problem would be some clustering according to the harmonic structure and spatial cues as tones from the same instrument can be assumed to have the same spatial parameters as sample-shift and magnitude difference between channels.

Generally, we could make more use of frequency domain representations in order to get better results for tone searching as well as tone learning because in the frequency domain the tones should be better decorrelated due to their typically high amount of harmonic content.

Lastly we shall note that it usually is also a good idea to analyze and make use of relationships between tones at a higher level in order to minimize template matching errors by taking harmonicity rules into account for example or by looking for repeating patterns in the musical piece.

Chapter 4

Iterative Template Matching

4.1 Introduction

The first approach had some problems especially in finding plausible tone waveforms representing as much of the input signal as possible. So we thought of an approach similar to the first one but allowing more freedom of choosing the tone waveforms by letting the loudness weights of the steering vector self-organize themselves.

We will not discuss algorithms and problems which are in common with the first approach but only reference to the corresponding subsection.

4.2 Overview

The first step in this method is the initialization of the steering vector and tone waveform described on Page 58. As in the first approach we then have an iterative main algorithm consisting of two steps. An overview of the main algorithm and the first step is given on Page 58. In a separate subsection on Page 59 the second step which is learning step is described in more detail.

As the learning step is rather detailed and contains several versions and expansions of its algorithm a final algorithm subsection is presented on Page 72.

Finally we close the first approach with a summary and future work subsection on Page 75.

4.3 Initialization

The initialization plays a very important role in this algorithm, even more important than in the first approach as it now influences the outcome systematically.

We have first tried a random initialization as usual. That is, the steering vector as well as the tone waveform is assigned some Gaussian noise with small amplitude. More precisely, the noise for the steering vector is generated by taking the absolute value of the generated Gaussian noise with mean around zero. After initializing the tone waveform with noise it is then normalized. Initializing this way leads to an outcome of the algorithm which is probably less surprising some noise with small amplitude and amplitude shaped according to the amplitude shape of the original signal. Not having any useful results we then tried another initialization.

The second method initializes the tone waveform with an exact copy of the input signal chosen at a position with relatively high energy and the steering vector with an one at that position and zero elsewhere. The steering vector is then multiplied with the norm of the tone waveform and finally the tone waveform is normalized to unity vector length. Choosing the position was done by randomly picking two locations in the input signal and then choosing the one with the higher energy content which was measured by simply taking the sum of squares of each sample value.

Although this approach worked better we now observed a new phenomenon. The algorithm tended not to change the tone waveform but rather to find a steering vector which could reconstruct the input using the given tone waveform. Therefore the tone waveform remains the one it was initialized with. This phenomenon now has two implications. First the initialization has to be rather accurate and second the learning algorithm has to be improved in order to choose a better suiting tone waveform. Now due to time constraints on this work both issues have been left to be elaborated upon for future work. As for now we use the second method for initialization as it is.

4.4 Main Algorithm

This approach works iteratively, where each iteration is made up of two steps. The first one is a synthesizing step where all steering vectors are convolved with the respective tone waveforms as in Equation 3.3 in order to produce a reconstruction of the input signal $\hat{\mathbf{x}}_i$. This step resembles the tone search step of the first approach except that the tone onsets are not explicitly searched for but are already encoded in the onset vector as a result of the learning step which tries to minimize the reconstruction error using as few non-zero elements in the steering vector as possible. This sparsity property of the steering vector is discussed more thoroughly later in the learning step.

The second step is a learning step where the steering vector and the tone waveform are adjusted in order to minimize the reconstruction error. In order to calculate the reconstruction error it needs the estimate of the input signal generated in the synthesizing step before.

The algorithm stops after a predetermined number of iterations. Other criterions may also be applied as for example when the cost measure of the whole input signal falls below a given threshold or when the decrease of that cost measure falls below a given threshold but the first criterion gives greater control over the time span needed to terminate and thus the total computational expense for a final separation.

In order to get the estimate for each tone in part we only need to convolve the steering vector and tone waveform of the desired template and get an estimated input signal $\hat{\mathbf{x}}_{i,k}$ with

$$\hat{\mathbf{x}}_{i,k}(n) = \hat{\mathbf{a}}_{i,k}(n) \star \hat{\mathbf{s}}_{i,k}(n) \tag{4.1}$$

where only that tone is played. As in the first approach we cannot get an estimation of the sound of the whole instrument as we do not yet have an algorithm to group the tones into instruments, which is left for future work.

4.5 Learning Step

During the learning step the algorithm will adjust the steering vector as well as the tone waveform which will be use used in the next synthesizing step iteration. We note that contrary to the first algorithm the steering vector is now reused in the next iteration and is subject to iterative fine adjustment whereas the onset vector in the first approach was thrown away after learning except when the last iteration was reached.

We use for the weight adaptation of the steering vector and tone waveform a gradient descent algorithm as in the first template-based approach. In order to build the algorithm we first have to define a cost function which in our case will consist of the reconstruction error as Equation 3.33 in the first approach

$$\mathscr{C}_{e,i}(n) = \frac{1}{2} \left\| \mathbf{x}_i - \hat{\mathbf{x}}_i(n) \right\|^2 \tag{4.2}$$

plus an additional cost measure for non-zero steering vector elements as proposed in [38] in order to keep the steering vector sparse

$$\mathscr{C}_{s,i}(n) = \sum_{k=1}^{K} \frac{|\hat{\mathbf{a}}_{i,k}(n)|}{\|\hat{\mathbf{a}}_{i,k}(n)\|} \tag{4.3}$$

where $|\cdot|$ denotes the L_1-norm of the vector and $\|\cdot\|$ the L_2-norm. Finally the new cost function becomes

$$\mathscr{C}_i(n) = \mathscr{C}_{e,i}(n) + \beta(n)\mathscr{C}_{s,i}(n) \tag{4.4}$$

where $\beta(n)$ is a weighting term similar to the one described in [38]. This term should ensure a balance between error minimization and sparseness maximization as the difference between the values of both error functions may be very high, leading to different implicit weighting during minimization if there would be no weighting term correcting this unbalance. Unsurprisingly the optimal value for $\beta(n)$ is very dependent on the input signal characteristic and thus has to be determined for each musical piece in part. A good rule of thumb is to choose $\beta(n)$ in the range

$$0 < \beta(n) < \frac{1}{4}\frac{\mathscr{C}_{s,i}(n)}{\mathscr{C}_{e,i}(n)} \tag{4.5}$$

as suggested in [38]. Note that the range defined in our work is dependent on the values of the cost functions at the actual iteration and thus $\beta(n)$ must be assigned a new value at each iteration. Obviously the suggested range leaves a wide margin for picking the weight parameter making a good guess a challenging task. More tightly defined ranges would make guessing easier and could possibly improve the end result to a great extent which is left at that point as a topic of future work. At the time being we choose

$$\beta(n) = \frac{1}{4}\frac{\mathscr{C}_{s,i}(n)}{\mathscr{C}_{e,i}(n)} \tag{4.6}$$

which seems to work well.

We need the new non-sparseness cost function which penalizes for non-zero elements in the steering vector in order to obtain a vector with only few non-zero elements as in the first approach. If we would not enforce sparseness a trivial solution to the reconstruction vector $\hat{\mathbf{x}}_i$ would be possible. The algorithm could create a copy the original input signal in the steering vector of one template and manipulate the respective tone waveform to contain an one in the first position with the rest set to zero. If this steering vector is then convolved with the tone waveform the reconstruction will be perfect assuming all other templates have either a zero-filled steering vector or zero-filled tone waveform.

So in order to minimize both parts of the total cost function the algorithm will have to find those tone onset positions where the tone waveforms will fit best or looking from another point of view it will have those tone onset positions where the un-normalized correlation between the input signal and the tone waveform is high. This should remind us a little of the first approach where we searched for the onset positions with maximal correlation and loudness among other things and that corresponds to the onset positions with maximal un-normalized correlation.

To continue with the design of the algorithm we will now have to calculate the first partial derivative of the cost function with respect to the steering vector $\hat{a}_{i,k,t}$ and separately with respect to the tone waveform $\hat{s}_{i,k,l}$ in order to obtain the gradient information. Because we want to minimize the cost function we will go in the direction of the negative gradient which means we have to add the negative gradient to each parameter in part.

The main update equations are taken from the first approach

$$\hat{a}_{i,k,t}(n+1) = \hat{a}_{i,k,t}(n) - \eta_{\hat{a}} \Delta \hat{a}_{i,k,t}(n) \tag{4.7}$$

and

$$\hat{s}_{i,k,l}(n+1) = \hat{s}_{i,k,l}(n) - \eta_{\hat{s}} \Delta \hat{s}_{i,k,l}(n) \tag{4.8}$$

where $0 < \eta_{\hat{a}} < 1$ and $0 < \eta_{\hat{s}} < 1$ are learning parameters and $\Delta \hat{a}_{i,k,t}(n)$ and $\Delta \hat{s}_{i,k,l}(n)$ are the gradients of the repsective parameters which are define by

$$\Delta \hat{a}_{i,k,t}(n) = \frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{a}_{i,k,t}(n)} \tag{4.9}$$

and

$$\Delta \hat{s}_{i,k,l}(n) = \frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{s}_{i,k,l}(n)} \tag{4.10}$$

New are the equations of the partial derivatives whose differences come from the fact that we now use the steering vector without assuming it to be sparse. So without an associated index set $\mathcal{T}_{i,k}$ which would show the non-zero values of the steering vector we cannot skip the zero elements in that vector during differentiation. The partial derivative of the steering vector also becomes different due to the new non-sparseness cost function resulting in

$$\begin{aligned} \frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{a}_{i,k,t}(n)} \quad = \quad & \sum_{l=1}^{T-t+1} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n)\right) \frac{\partial \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n)\right)}{\partial\, \hat{a}_{i,k,t}(n)} \\ & + \beta(n) \left(\frac{\operatorname{sgn}\left(\hat{a}_{i,k,t}(n)\right) \left\|\hat{\mathbf{a}}_{i,k}(n)\right\| - \hat{a}_{i,k,t}(n) \frac{\left|\hat{\mathbf{a}}_{i,k}(n)\right|}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|}}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|^2} \right) \end{aligned} \tag{4.11}$$

by applying the chainrule where $\operatorname{sgn}(\,\cdot\,)$ denotes the signum function which is defined as

$$\operatorname{sgn}(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases} \tag{4.12}$$

After solving the last partial derivatives and some simplifications we arrive to

$$\begin{aligned}\frac{\partial \mathscr{C}_i(n)}{\partial \hat{a}_{i,k,t}(n)} \quad = \quad & -\sum_{l=1}^{D} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n)\right) \hat{s}_{i,k,l}(n) \\ & + \beta(n) \left(\frac{\operatorname{sgn}\left(\hat{a}_{i,k,t}(n)\right)}{\|\hat{\mathbf{a}}_{i,k}(n)\|} - \hat{a}_{i,k,t}(n) \frac{|\hat{\mathbf{a}}_{i,k}(n)|}{\|\hat{\mathbf{a}}_{i,k}(n)\|^3} \right) \end{aligned} \tag{4.13}$$

However, now the time index t will not be chosen from an index set indicating the non-zero values, but will sweep through the whole length of the input signal. Analogously for the partial derivative of $\hat{s}_{i,k,l}(n)$ we get

$$\frac{\partial \mathscr{C}_i(n)}{\partial \hat{s}_{i,k,l}(n)} = \sum_{t=1}^{T} \left(x_{i,t} - \hat{x}_{i,t}(n)\right) \frac{\partial \left(x_{i,t} - \hat{x}_{i,t}(n)\right)}{\partial \hat{s}_{i,k,l}(n)} \tag{4.14}$$

by applying the chain rule and then after solving the last partial derivatives and some simplifications we arrive to

$$\frac{\partial \mathscr{C}_i(n)}{\partial \hat{s}_{i,k,l}(n)} = -\sum_{t=1}^{T-l+1} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n)\right) \hat{a}_{i,k,t}(n) \tag{4.15}$$

where we should note the changed summation here which, compared to Equation 3.41, now runs over the whole range of the steering vector.

Now the computational complexity for solving these equations has drastically changed for the worse. The partial derivative for the steering vector uses a sum over D elements and must be evaluated for each sample and of that vector the partial derivative for the tone waveform uses a sum over $T - l + 1$ elements and must be evaluated over each of the D samples of the waveform. This results in a complexity of $\mathcal{O}(TD)$ which means a too high computational burden for today's personal computers as we already discussed in the Tone Search subsection of the first approach. This complexity should look familiar as it was the same for the simple correlation in the said subsection. The solution was found by making use of the fact that correlation can be solved fast using the FFT which had a much lower complexity of $\mathcal{O}(T \log_2 T)$. Now we can speed up the calculations of our partial derivatives the same way. Using the definition of the reconstruction error $\mathbf{e}_i$ given in Equation 1.4 and making it a function of the iteration number n we can rewrite the partial derivative for the steering vector in Equation 4.13 as a correlation in vector form plus the unchanged non-sparseness cost function derivative

$$\frac{\partial \mathscr{C}_i(n)}{\partial \hat{\mathbf{a}}_{i,k}(n)} = -\mathbf{e}_i(n) \times \hat{\mathbf{s}}_{i,k}(n) + \beta(n) \left(\frac{\operatorname{sgn}\left(\hat{\mathbf{a}}_{i,k}(n)\right)}{\|\hat{\mathbf{a}}_{i,k}(n)\|} - \hat{\mathbf{a}}_{i,k}(n) \frac{|\hat{\mathbf{a}}_{i,k}(n)|}{\|\hat{\mathbf{a}}_{i,k}(n)\|^3} \right) \tag{4.16}$$

where the signum function applied to a vector results in a vector where the function is applied to each element in part. We can rewrite the partial derivative for the tone waveform in

Equation 4.15 in the same way resulting in

$$\frac{\partial \mathscr{C}_i(n)}{\partial \, \hat{\mathbf{s}}_{i,k}(n)} = -\mathbf{e}_i(n) \times \hat{\mathbf{a}}_{i,k}(n) \tag{4.17}$$

Now we can solve the correlations using the FFT as in the first approach making the calculations feasible again.

So after plugging the results for the partial derivatives into the update formulas which we now have to rewrite in vector form, we finally get

$$\begin{aligned} \hat{\mathbf{a}}_{i,k}(n+1) &= \hat{\mathbf{a}}_{i,k}(n) + \eta_{\hat{a}} \mathbf{e}_i(n) \times \hat{\mathbf{s}}_{i,k}(n) - \eta_{\hat{a}} \beta(n) \left(\frac{\operatorname{sgn}\left(\hat{\mathbf{a}}_{i,k}(n)\right)}{\|\hat{\mathbf{a}}_{i,k}(n)\|} - \hat{\mathbf{a}}_{i,k}(n) \frac{|\hat{\mathbf{a}}_{i,k}(n)|}{\|\hat{\mathbf{a}}_{i,k}(n)\|^3} \right) \\ \hat{\mathbf{s}}_{i,k}(n+1) &= \hat{\mathbf{s}}_{i,k}(n) + \eta_{\hat{s}} \mathbf{e}_i(n) \times \hat{\mathbf{a}}_{i,k}(n) \end{aligned} \tag{4.18}$$

We shall observe here that the update may cause some elements of the steering vector to become negative. As negative values have no physical meaning as was already pointed out in the tone search subsection in the first approach we simply set all negative elements to zero after every update. As this condition occurs fairly often we get the nice side-effect of enhancing sparseness in the steering vector at no additional cost.

After implementing the above formulas we learned that the gradient descent algorithm described here is highly unstable. In order to avoid divergence we had to keep the learning rate parameters $\eta_{\hat{a}}$ and $\eta_{\hat{s}}$ at very low values resulting in a painfully slow error decay. The unstable behaviour might be explained partly by the vast amount of free variables which are adjusted during every iteration. More precisely there are $K(T+D)$ variables which depending on the length of the input signal and the number of templates may be in the range of some ten to hundreds of millions. The assumption of the gradient descent method that whenever adjusting one parameter all others are constant has now come to be a very rough approximation of the real circumstances. This is then the reason why the learning rate parameters have to kept that small as the assumption holds only in that case because the changes in the variables are then forces to be small making the variables behave almost like constants.

The slow speed and the inherent instability made us search for improvements. So we tried the extensions already described in the Tone Learning subsection of the first approach like momentum, Super-SAB and Newton's method, where none of them worked as expected. Especially Super-SAB and Newton's method were slow due to the divergent behaviour of the algorithm which made Super-SAB back up very often and decreasing the individual learning rates too much and Newton's method needing more refinement iterations in order to find a good common learning rate.

This led us to an algorithm coming from the neural network domain called resilient back propagation (RPROP) as described in [30]. This algorithm may be viewed as an extension to Super-SAB as it very similar to it.

In the RPROP algorithm the error surface is assumed to be very rough having plateaus and steep descents. So if we reach a plateau we need to increase our steps to traverse it more quickly and if we reach a steep descent we have to decrease our steps in order to not miss a minimum. Up to this part Super-SAB is based on the same idea. Now here comes the difference in RPROP. Super-SAB manipulates the learning step size but if we do that we will still not be in full control of our step length in the direction pointed by the gradient as the gradient magnitude also changes. For example if we reach a steep part of the error surface our gradient magnitude may become 10 times higher as before while our learning rate decrease constant is usually set to divide the learning rate by two although in order to not overshoot it should divide by 20 in this case. So in conclusion we may overshoot our target position by a large amount and possibly have to increase learning rate again to come back thus having to perform some additional steps which might be unnecessary. This is the part where RPROP improves speed. It throws away the gradient magnitude and uses only its sign giving full control of the step length on the error surface. It does not use learning rates but uses an estimated gradient which is made up of an estimated magnitude and the sign of the actual gradient. Now the estimated magnitude is changed similarly to the learning rate of Super-SAB. If the gradient sign does not change between two iterations then the estimated gradient magnitude is increased by a multiplicative constant η_+ and if it changes then the magnitude is decreased by η_- and the update in the former equation is undone. We have chosen η_+ and η_- to be 1.1 and $1/2$ respectively whereby η_+ is kept smaller than in the first approach because of the inherent instability of the equation system.

Now the RPROP algorithm works as follows:

- if this is the first iteration of the main algorithm
  - initialize $\hat{\Delta}\hat{a}_{i,k,t}(0)$ and $\hat{\Delta}\hat{s}_{i,k,l}(0)$ with zero.
- calculate the partial derivative for $\hat{\mathbf{a}}_{i,k}(n)$ according to Equation 4.16 and $\hat{\mathbf{s}}_{i,k}(n)$ according to Equation 4.17.
- do for each time index t
  - if

$$\frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{a}_{i,k,t}(n)} \frac{\partial \mathscr{C}_i(n-1)}{\partial\, \hat{a}_{i,k,t}(n-1)} > 0 \tag{4.19}$$

    update $\hat{a}_{i,k,t}$

$$\hat{a}_{i,k,t}(n+1) = \hat{a}_{i,k,t}(n) - \operatorname{sgn}\left(\Delta\hat{a}_{i,k,t}(n)\right) \hat{\Delta}\hat{a}_{i,k,t}(n) \tag{4.20}$$

    and increase the estimated gradient of $\hat{a}_{i,k,t}$

$$\hat{\Delta}\hat{a}_{i,k,t}(n+1) = \eta_+ \hat{\Delta}\hat{a}_{i,k,t}(n) \tag{4.21}$$

  - if
  $$\frac{\partial \mathscr{C}_i(n)}{\partial \hat{a}_{i,k,t}(n)} \frac{\partial \mathscr{C}_i(n-1)}{\partial \hat{a}_{i,k,t}(n-1)} < 0 \tag{4.22}$$
  restore $\hat{a}_{i,k,t}$
  $$\hat{a}_{i,k,t}(n+1) = \hat{a}_{i,k,t}(n-1) \tag{4.23}$$
  decrease the estimated gradient of $\hat{a}_{i,k,t}$
  $$\hat{\Delta}\hat{a}_{i,k,t}(n+1) = \eta_{-} \hat{\Delta}\hat{a}_{i,k,t}(n) \tag{4.24}$$
  and set $\frac{\partial \mathscr{C}_i(n)}{\partial \hat{a}_{i,k,t}(n)} = 0$ to avoid decreasing the gradient once more

- do for each sample l
  - if
  $$\frac{\partial \mathscr{C}_i(n)}{\partial \hat{s}_{i,k,l}(n)} \frac{\partial \mathscr{C}_i(n-1)}{\partial \hat{s}_{i,k,l}(n-1)} > 0 \tag{4.25}$$
  update $\hat{s}_{i,k,l}$
  $$\hat{s}_{i,k,l}(n+1) = \hat{s}_{i,k,l}(n) - \operatorname{sgn}\left(\Delta\hat{s}_{i,k,l}(n)\right) \hat{\Delta}\hat{s}_{i,k,l}(n) \tag{4.26}$$
  and increase the estimated gradient of $\hat{s}_{i,k,l}$
  $$\hat{\Delta}\hat{s}_{i,k,l}(n+1) = \eta_{+} \hat{\Delta}\hat{s}_{i,k,l}(n) \tag{4.27}$$
  - if
  $$\frac{\partial \mathscr{C}_i(n)}{\partial \hat{s}_{i,k,l}(n)} \frac{\partial \mathscr{C}_i(n-1)}{\partial \hat{s}_{i,k,l}(n-1)} < 0 \tag{4.28}$$
  restore $\hat{s}_{i,k,l}$
  $$\hat{a}_{i,k,l}(n+1) = \hat{s}_{i,k,l}(n-1) \tag{4.29}$$
  decrease the estimated gradient of $\hat{s}_{i,k,l}$
  $$\hat{\Delta}\hat{s}_{i,k,l}(n+1) = \eta_{-} \hat{\Delta}\hat{s}_{i,k,l}(n) \tag{4.30}$$
  and set $\frac{\partial \mathscr{C}_i(n)}{\partial \hat{s}_{i,k,l}(n)} = 0$ to avoid decreasing the gradient once more

This algorithm seems to work very well and is also stable. But like Super-SAB in the first approach it has the problem of introducing noise in the tone waveform due to the different adaptation speed of each sample. This might be solved by updating the tone waveform with another algorithm which keeps one learning rate for all samples but as already mentioned the other methods were much slower and thus were not a real option.

During preliminary testing we also discovered another problem. As musical pieces usually consist of millions of samples the steering vector also consists of that many loudness weights.

Now considering that relatively many templates have to be used the memory consumption grows to critical levels for common personal computer as of 2007 as all weights from all templates, their estimated deltas and their former and actual partial derivatives have to be held in memory. Here we can give as a small example a monaural song 3:20 minutes long, sampled at 44.1kHz which has 8 820 000 samples. As we save all our parameters in floating point numbers each needing 4 bytes of memory space we need 141.120 MBytes of memory to store just one template representing one single tone. We would consume 1.41 GByte by just having 10 templates and 14.1 GByte for 100 templates which would be a more realistic count. For longer songs possibly recorded in stereo we would need even more space.

A solution to the memory problem could be to split one main iteration into subiterations for each template. This means that each template would be updated for a fixed number of subiterations alone. The other templates could be temporarily swapped to the more spacious hard-disk leaving us with just one template in memory at a time. Knowing that the other templates will remain constant during the subiterations we can calculate the reconstructed signal $\hat{\mathbf{x}}_i(n)$ once and update it with whenever the actual template is updated in the subiteration.

We have not implemented this solution yet so we cannot say anything about how the quality of the result would be affected by this strategy as the proposed update is not equivalent to the joint optimization that we have considered until now. But it may be an interesting topic for future work.

4.6 Fine Tuning

We already discussed some problems of the main algorithm including its major steps and provided some suggestions on how to solve them. The algorithm will work that way but quality can be further improved. Therefore we will show some thoughts in this subsection which can usually be quickly implemented and result in an output with better quality.

Non-Sparseness Cost Function

We took the cost function suggested in [38] for the learning step. Unfortunately we had to learn that although it works it has some major problems, mostly of theoretical nature.

$$\mathscr{C}_{s,i}(n) = \sum_{k=1}^{K} \frac{|\hat{\mathbf{a}}_{i,k}(n)|}{\|\hat{\mathbf{a}}_{i,k}(n)\|} \tag{4.31}$$

The first problem is that the global minimum of the cost function is not situated at zero but at one as the maximally sparse vector has just one non-zero element and in this case the L_1 and the L_2 norm are equal. This comes in contrast to the reconstruction error part of the total cost function whose minimum is obviously situated at zero (see the trivial example described at the beginning of the learning step subsection on Page 60). This offset by one caused by each template can sum up depending on how many templates are used. It may be less of concern considering that the ones will vanish during differentiation but it may have an averse effect on the weighting parameter $\beta(n)$ especially when the reconstruction error approaches zero. An obvious solution here is to subtract the ones from the cost function resulting in

$$\mathscr{C}_{s,i}(n) = \sum_{k=1}^{K} \frac{|\hat{\mathbf{a}}_{i,k}(n)|}{\|\hat{\mathbf{a}}_{i,k}(n)\|} - K \tag{4.32}$$

With this out of the way we can examine the more interesting aspects of the cost function. As we know from Equation 4.13 the derivative of $\mathscr{C}_{s,i}(n)$ results in the use of the signum function

$$\begin{aligned} \frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{a}_{i,k,t}(n)} \quad &= \quad -\sum_{l=1}^{D} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n)\right) \hat{s}_{i,k,l}(n) \\ &\quad + \beta(n) \left(\frac{\operatorname{sgn}\left(\hat{a}_{i,k,t}(n)\right)}{\|\hat{\mathbf{a}}_{i,k}(n)\|} - \hat{a}_{i,k,t}(n) \frac{|\hat{\mathbf{a}}_{i,k}(n)|}{\|\hat{\mathbf{a}}_{i,k}(n)\|^3} \right) \end{aligned} \tag{4.33}$$

The problem that arises here is that the signum function is not continuous around zero. This leads to an abrupt change in the gradient for an element of the steering vector whenever it reaches zero or is set to zero because it became negative during the former iteration. As we know abrupt changes are not good in gradient descent algorithms even if the step size is controlled by RPROP as they are often not plausible. For example an element reaching zero will have a more or less constant gradient coming from the non-sparseness cost function consisting of $-1/\,\|\hat{\mathbf{a}}_{i,k}(n)\|$ while the $\hat{a}_{i,k,t}(n)\,|\hat{\mathbf{a}}_{i,k}(n)|\,/\,\|\hat{\mathbf{a}}_{i,k}(n)\|^3$ term reaches zero as expected. Now we would expect the first term to also slowly tend to zero in order to allow some fine adjustments to be done coming from the reconstruction error cost function. We might note here that the said term might be low in practice as the vector norm of $\hat{\mathbf{a}}_{i,k}$ is not normalized unlike the tone waveform. A possible solution here could be the use of the L_2-norm now denoted by $\|\cdot\|_2$ together with the L_3-norm denoted by $\|\cdot\|_3$ resulting in a non-sparseness cost function

$$\mathscr{C}_{s,i}(n) = \sum_{k=1}^{K} \frac{\|\hat{\mathbf{a}}_{i,k}(n)\|_2}{\|\hat{\mathbf{a}}_{i,k}(n)\|_3} - K \tag{4.34}$$

Now taking the partial derivative for the steering vector elements we get

$$\begin{aligned}\frac{\partial \mathscr{C}_i(n)}{\partial \hat{a}_{i,k,t}(n)} &= \sum_{l=1}^{T-t+1} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n)\right) \frac{\partial \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n)\right)}{\partial \hat{a}_{i,k,t}(n)} \\ &\quad + \beta(n) \left(\frac{\hat{a}_{i,k,t}(n) \frac{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_3}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_2} - \hat{a}^2_{i,k,t}(n) \frac{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_2}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_3^2}}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_3^2} \right) \end{aligned} \tag{4.35}$$

by applying the chain rule. After solving the last partial derivatives, some simplifications and reformulating in vector notation we arrive to

$$\begin{aligned}\frac{\partial \mathscr{C}_i(n)}{\partial \hat{a}_{i,k,t}(n)} &= -\mathbf{e}_i(n) \times \hat{\mathbf{s}}_{i,k}(n) \\ &\quad + \beta(n) \left(\frac{\hat{\mathbf{a}}_{i,k,t}(n)}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_2 \left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_3} - \hat{\mathbf{a}}^2_{i,k,t}(n) \frac{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_2}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_3^4} \right) \end{aligned} \tag{4.36}$$

The update formula in our vector notation using correlation then becomes

$$\begin{aligned}\hat{\mathbf{a}}_{i,k}(n+1) &= \hat{\mathbf{a}}_{i,k}(n) + \eta_{\hat{a}} \mathbf{e}_i(n) \times \hat{\mathbf{s}}_{i,k}(n) \\ &\quad - \eta_{\hat{a}} \beta(n) \left(\frac{\hat{\mathbf{a}}_{i,k}(n)}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_2 \left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_3} - \hat{\mathbf{a}}^2_{i,k}(n) \frac{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_2}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_3^4} \right) \end{aligned} \tag{4.37}$$

Analyzing the partial derivative we now see that indeed the gradient generated by the non-sparseness cost function becomes continuously smaller when a steering vector element is about to reach zero. The modifications needed to use RPROP with the above equation should be straightforward.

At this point it should be mentioned that there also are some other cost functions out there in literature as for example $\log\left(1 + \hat{a}^2_{i,k,t}(n)\right)$ presented in [23] whereby this function should be normalized in order to exclude the trivial solution of setting all $\hat{a}_{i,k,t}$ to zero. This would result in

$$\mathscr{C}_{s,i}(n) = \sum_{k=1}^{K} \sum_{t=1}^{T} \frac{\log\left(1 + \hat{a}^2_{i,k,t}(n)\right)}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|} \tag{4.38}$$

and the partial derivative of the total cost function would be

$$\begin{aligned}\frac{\partial \mathscr{C}_i(n)}{\partial \hat{a}_{i,k,t}(n)} &= -\sum_{l=1}^{D} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n)\right) \hat{s}_{i,k,l}(n) \\ &\quad + \beta(n) \hat{a}_{i,k,t}(n) \left(\frac{2}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\| \left(1 + \hat{a}^2_{i,k,t}(n)\right)} - \frac{\log\left(1 + \hat{a}^2_{i,k,t}(n)\right)}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|^3} \right) \end{aligned} \tag{4.39}$$

Obviously the derivative is continuous near zero which is an argument speaking for it but as we did not try it out we can hardly further comment on it. So we leave it as part of future work to investigate more kinds of non-sparseness cost functions and their effects on the quality of the separation.

After providing these minor corrections to the non-sparseness cost function we remain with another objectionable flaw. The cost function reaches its global minimum if the steering vector achieves maximal sparseness. This is the case when all but one element in the vector has a non-zero value. Now in order to be precise we have to say that this is not really our aim as we want the steering vector to contain exactly as many non-zero elements as there are tone occurrences in the input signal and not any fewer. So it is possible that by minimizing the cost function we will miss the optimum which is not what we intended. Unfortunately, we do not know any solution to this as any improvement method would require knowledge about the number of occurrences to expect, which poses a difficult problem as it can at best be estimated. On the other hand if we already knew the number of tone occurrences we probably would not need this algorithm as it may be more efficient to search for the tones explicitly like in the first approach and not let the algorithm self-organize. But to our relief we saw in our preliminary tests that this problem will not be of much concern as the vector sparseness remains usually low and thus we do not even reach the part where we may have too few non-zero elements in the vector.

Lateral Inhibition

The discussion about the non-sparseness cost function was about the number of non-zero elements in the steering vector until now. But nothing was said on how they might be spatially distributed. We have nothing to gain if all non-zero elements would be situated one after the other without any zero elements in between. We want the non-zero elements as they represent onsets to be placed where onsets are most probably in a musical signal. This means that there must be at least some minimum time interval between the onsets representing the duration of the shortest note played in the musical piece. Other constraints would be that these onsets may be more probable at some specific locations derived from the metrum of the song. We will focus here on the minimum distance problem leaving the study of the other constraints as an issue for future work.

So in order to keep a minimum distance between the non-zero elements we thought about introducing some kind of lateral inhibition connections. They should cause that only the element with the highest positive value will remain active and the others be pushed towards zero by it. This idea is based on concepts of artificial neural networks which were copied from real structures of neural networks found in brains. In the nervous system the lateral inhibition weights are intended to remove noise by suppressing activity of weakly excited neurons and letting only the strongly excited ones fire. In our separation algorithm we can also consider the weak onset weights in the steering vector to be noise if there is a strong weight nearby.

Another possible point of view of what lateral inhibition does can be that of sharpening signals like images. Sharpening means here, enhancing the contrast between neighbouring samples. And that is what lateral inhibition is intended to do. It maximizes the difference between neighbouring weights by suppressing the weak ones thus making the difference to the strong ones bigger.

Now after we got to the idea on what the lateral inhibition connections are supposed to do we can move on to a more formal definition. For that we will change the gradient of the elements of the steering vector to simply subtract the sum of neighbouring elements

$$\Delta\hat{a}_{i,k,t}(n) = \frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{a}_{i,k,t}(n)} + \gamma(n) \sum_{o=-\theta_o, o\neq 0}^{\theta_o} \hat{a}_{i,k,t+o}(n) \tag{4.40}$$

where $\theta_o \in \mathbb{N}$ is the range parameter measured in samples and $\gamma(n)$ is a weighting coefficient for striking a balance between the inhibition term and the partial derivative in the gradient.

Analyzing the above equation we see that the sum will be zero for an element if all its surrounding elements are zero. The weighting coefficient now plays the role of not letting the inhibition term get as big as to annihilate the promising strong elements. This coefficient is important for the lateral inhibition to work correctly. Unfortunately as we did not have time to investigate on the initialization of this coefficient we simply set it by trial and error.

As we saw in literature it is also possible to express the lateral inhibition in a more elegant way. We saw that the lateral inhibition is a result of trying to minimize the autocorrelation function of the steering vector. Minimizing the autocorrelation can be done by simple gradient descent which means partially differentiating it by every variable that needs to be updated. The result of the differentiation is then lateral inhibition already discussed. So in conclusion we can extend the non-sparseness cost function with the autocorrelation of the steering vector

$$\mathscr{C}_{s,i}(n) = \sum_{k=1}^{K} \frac{\|\hat{\mathbf{a}}_{i,k}(n)\|_2}{\|\hat{\mathbf{a}}_{i,k}(n)\|_3} - K + \gamma(n)\frac{1}{2} \sum_{k=1}^{K}\sum_{t=1}^{T} \sum_{o=-\theta_o, o\neq 0}^{\theta_o} \hat{a}_{i,k,t}(n)\hat{a}_{i,k,t+o}(n) \tag{4.41}$$

and after a small rearrangement

$$\mathscr{C}_{s,i}(n) = \sum_{k=1}^{K} \left(\frac{\|\hat{\mathbf{a}}_{i,k}(n)\|_2}{\|\hat{\mathbf{a}}_{i,k}(n)\|_3} + \gamma(n)\frac{1}{2} \sum_{t=1}^{T} \sum_{o=-\theta_o, o\neq 0}^{\theta_o} \hat{a}_{i,k,t}(n)\hat{a}_{i,k,t+o}(n) \right) - K \tag{4.42}$$

The scaling term of $1/2$ is used to simplify differentiation. Note that the autocorrelation is not done for the entire signal but on a sliding window basis with window size $2\theta_o$ because we want to restrict lateral inhibition to a plausible range of time instead of having it over the entire signal.

After defining the cost function we can differentiate the total cost function arriving to

$$\begin{aligned}\frac{\partial \mathscr{C}_i(n)}{\partial \hat{a}_{i,k,t}(n)} &= -\sum_{l=1}^{D} \left(x_{i,t+l-1} - \hat{x}_{i,t+l-1}(n)\right) \hat{s}_{i,k,l}(n) \\ &+ \beta(n) \left(\frac{\hat{a}_{i,k,t}(n)}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_2 \left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_3} - \hat{a}^2_{i,k,t}(n) \frac{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_2}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_3^4} \right) \\ &+ \gamma(n) \sum_{o=-\theta_o, o\neq 0}^{\theta_o} \hat{a}_{i,k,t+o}(n) \end{aligned} \tag{4.43}$$

If we want to use our vector notation we first have to introduce a new lateral inhibition weight function $h(z)$ defined as

$$h(z) = \begin{cases} 1 & z \neq 0,\ |z| \leq \theta_o \\ 0 & z = 0 \\ 0 & |z| > \theta_o \end{cases} \tag{4.44}$$

which can be evaluated to the weight vector

$$\mathbf{h} = [h(-\theta_o), h(-\theta_o + 1), ..., h(-1), h(0), h(1), ..., h(\theta_o)]^T \tag{4.45}$$

We will use this weight function in the final formula to introduce a new correlation which can be solved fast by the FFT

$$\begin{aligned}\hat{\mathbf{a}}_{i,k}(n+1) &= \hat{\mathbf{a}}_{i,k}(n) + \eta_{\hat{a}} \mathbf{e}_i(n) \times \hat{\mathbf{s}}_{i,k}(n) \\ &- \eta_{\hat{a}} \beta(n) \left(\frac{\hat{\mathbf{a}}_{i,k}(n)}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_2 \left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_3} - \hat{\mathbf{a}}^2_{i,k}(n) \frac{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_2}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_3^4} \right) \\ &- \eta_{\hat{a}} \gamma(n) \hat{\mathbf{a}}_{i,k}(n) \times \mathbf{h} \end{aligned} \tag{4.46}$$

where the last correlation has to match the offset anchor of the weight vector which is not at the beginning but in the middle where the element $h(0)$ relies.

As we saw there is some fine tuning possible to this weight function $h(z)$. For example we could give the elements of the steering vector nearer to the element whose inhibition value is calculated some higher weights than those farther away. This approach should lead to a smoother transition at the end of window. One possibility would be for example some linear increase from the ends of the window towards the centre where the centre sample remains zero or a Gaussian function with zero centre.

Now as we tested the lateral inhibition a little we did not see much improvement if any in the structure of the steering vector. This may be caused by the fact that the steering vector does not really get as sparse as we would expect it to be. We did not investigate the lateral inhibition further due to lack of time so we decided to drop it despite the efforts we put into this approach. Still we are confident that this fine tuning method would benefit the quality of the separation result due to the reason described earlier that the global non-sparseness cost function does not take spatial arrangement constraints into account. Perhaps this issue will be resolved at some time in future work.

Phase Matching

This algorithm exhibits the same phase matching problems like the first template-based approach. We described this issue in detail in the corresponding phase matching subsection on Page 48f in the first approach. But this time we are left with just one option.

As the algorithm does not search explicitly for the templates but does a self-organized search by the steering vector we cannot do the phase matching by iteratively fine adjusting every candidate match. So we have to resort to up- and downsampling the input signal.

Now as we know we already have limited memory resources so upsampling will aggravate this problem even more. Unfortunately there is nothing we can do about it as we have no other way but to store the steering vector weights in an "upsampled" steering vector. The stability problem also gets worse as the upsampling will introduce new weights in the steering vector as it gets larger thus giving the algorithm even more free parameters which have to be adjusts simultaneously. But as we tried it out we saw that the upsampling helped quality considerably so we opted to choose fewer templates in order to preserve memory and save time, which seemed a good compromise to us.

4.7 Final Algorithm

As for some steps we discussed several possibilities to be solved resulting in more than one version of the step and some improvements were taken out from the final algorithm because of problems encounters we thought we should include a complete final version of the algorithm which will be composed of only those steps and their improvements which actually worked.

- upsample the input signal $\mathbf{x}_i$ by an upsampling factor of θ_u with a Lanczos filter of order θ_l. Rescale the affected template parameters to fit the new sample frequency.
- do for each template k
  - pick randomly two positions in the input signal t_0, t_1

  - choose the position with its highest associated loudness λ_t measured as

$$\lambda_t = \sum_{l=1}^{D} x_{i,t+l}^2 \tag{4.47}$$

    where $t = t_0$ or $t = t_1$.

  - copy the part from the original signal at that position into the tone vector $\hat{\mathbf{s}}_{i,k}$ initialize the steering vector $\hat{\mathbf{a}}_{i,k}$ at that position with one and the rest of it with zeros.
  - divide $\hat{\mathbf{a}}_{i,k}$ by the vector norm of the tone vector
  - normalize the tone vector to unity vector norm.

- do iteratively until maximum number of iterations reached
  - calculate the reconstruction

$$\hat{\mathbf{x}}_i = \sum_{k=1}^{K} \hat{\mathbf{a}}_{i,k} \star \hat{\mathbf{s}}_{i,k} \tag{4.48}$$

  - calculate $\beta(n)$ as being

$$\beta(n) = \frac{\left\|\mathbf{x}_i - \hat{\mathbf{x}}_i(n)\right\|^2}{8\sum_{k=1}^{K} \frac{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_2}{\left\|\hat{\mathbf{a}}_{i,k}(n)\right\|_3} - K} \tag{4.49}$$

  - do for each template k
    - ∗ if this is the first iteration of the main algorithm
      - · initialize $\hat{\Delta}\hat{a}_{i,k,t}(0)$ and $\hat{\Delta}\hat{s}_{i,k,l}(0)$ with zero.
    - ∗ calculate the partial derivative for $\hat{\mathbf{a}}_{i,k}(n)$ according to Equation 4.36 and $\hat{\mathbf{s}}_{i,k}(n)$ according to Equation 4.17.
    - ∗ do for each time index t
      - · if

$$\frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{a}_{i,k,t}(n)} \frac{\partial \mathscr{C}_i(n-1)}{\partial\, \hat{a}_{i,k,t}(n-1)} > 0 \tag{4.50}$$

        update $\hat{a}_{i,k,t}$

$$\hat{a}_{i,k,t}(n+1) = \hat{a}_{i,k,t}(n) - \operatorname{sgn}\left(\Delta\hat{a}_{i,k,t}(n)\right) \hat{\Delta}\hat{a}_{i,k,t}(n) \tag{4.51}$$

        and increase the estimated gradient of $\hat{a}_{i,k,t}$

$$\hat{\Delta}\hat{a}_{i,k,t}(n+1) = \eta_+ \hat{\Delta}\hat{a}_{i,k,t}(n) \tag{4.52}$$

- · if

  $$\frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{a}_{i,k,t}(n)} \frac{\partial \mathscr{C}_i(n-1)}{\partial\, \hat{a}_{i,k,t}(n-1)} < 0 \tag{4.53}$$

  restore $\hat{a}_{i,k,t}$

  $$\hat{a}_{i,k,t}(n+1) = \hat{a}_{i,k,t}(n-1) \tag{4.54}$$

  decrease the estimated gradient of $\hat{a}_{i,k,t}$

  $$\hat{\Delta}\hat{a}_{i,k,t}(n+1) = \eta_{-}\hat{\Delta}\hat{a}_{i,k,t}(n) \tag{4.55}$$

  and set $\frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{a}_{i,k,t}(n)} = 0$ to avoid decreasing the gradient once more

- * do for each sample l
  - · if

    $$\frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{s}_{i,k,l}(n)} \frac{\partial \mathscr{C}_i(n-1)}{\partial\, \hat{s}_{i,k,l}(n-1)} > 0 \tag{4.56}$$

    update $\hat{s}_{i,k,l}$

    $$\hat{s}_{i,k,l}(n+1) = \hat{s}_{i,k,l}(n) - \operatorname{sgn}\left(\Delta\hat{s}_{i,k,l}(n)\right)\hat{\Delta}\hat{s}_{i,k,l}(n) \tag{4.57}$$

    and increase the estimated gradient of $\hat{s}_{i,k,l}$

    $$\hat{\Delta}\hat{s}_{i,k,l}(n+1) = \eta_{+}\hat{\Delta}\hat{s}_{i,k,l}(n) \tag{4.58}$$

  - · if

    $$\frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{s}_{i,k,l}(n)} \frac{\partial \mathscr{C}_i(n-1)}{\partial\, \hat{s}_{i,k,l}(n-1)} < 0 \tag{4.59}$$

    restore $\hat{s}_{i,k,l}$

    $$\hat{a}_{i,k,l}(n+1) = \hat{s}_{i,k,l}(n-1) \tag{4.60}$$

    decrease the estimated gradient of $\hat{s}_{i,k,l}$

    $$\hat{\Delta}\hat{s}_{i,k,l}(n+1) = \eta_{-}\hat{\Delta}\hat{s}_{i,k,l}(n) \tag{4.61}$$

    and set $\frac{\partial \mathscr{C}_i(n)}{\partial\, \hat{s}_{i,k,l}(n)} = 0$ to avoid decreasing the gradient once more

- downsample the templates by a factor of θ_u to match the sample rate of original input signal.

4.8 Summary and Future Work

As the first template-based approach had problems finding plausible tone waveforms due to its explicit search for onsets, we developed and presented an alternative approach which uses self-organizing steering vectors to ameliorate that problem. We have begun with its initialization procedure which copies randomly picked slices from the input signal into the tone waveforms of the templates and sets the weights in each associated steering vector to match the location of the corresponding slice.

The main algorithm was introduced as being an iterative approach consisting of two steps. The first one was the synthesizing step where the signal is reconstructed by a sum of convolutions of the steering vectors with their corresponding tone waveforms. The tone onsets were not being searched after explicitly as in the first approach this time because they were already encoded in the steering vector.

The learning step following the synthesizing stage adjusts the tone waveforms and the steering vectors by a gradient descent method. For that method we introduced a new non-sparseness cost function which should force the algorithm to use few tone onsets in order to reconstruct the input signal, and which should avoid trivial solutions like copying the input signal into the steering vectors. As the algorithm proved very unstable due to its many free parameters and thus converged only slowly even when using the extension discussed in the first template-based approach, we decided to improve the method with RPROP which eliminates the impact of the gradient magnitude by estimating the gradient directly. We then found a memory space problem due to the large amount of data which has to be stored simultaneously, as the steering vector is not sparse anymore and thus cannot be compressed by just indexing it. The problem remained unresolved but we suggested the usage of refinement steps where each template is optimized separately during an iteration thus making it possible to temporarily swap the other templates to the hard disk, freeing up memory.

We also showed some fine tuning methods which are expected to improve the algorithms quality. At first we discussed the non-sparseness cost function and its problems. One of them is that its optimum is not at zero, which distorts the true ratio of the reconstruction error to the non-sparseness cost. The ratio is used for calculating a weighting term keeping the ratio fixed in order to string a balance between both cost functions. The solution to this problem was simple as one only needs to subtract the number of templates from the non-sparseness cost function. Another problem was the discontinuity of its derivative around zero which may be a cause of the instability of the total algorithm. A new non-sparseness cost function was proposed to eliminate that problem. The last problem of that function was that its optimum is found when the steering vector is maximally sparse which is the case whenever this vector is made up of a single non-zero element and all others being zero. Obviously, this does not correspond to the desired minimum as we want the vector to contain as many non-zero elements as there are real onsets of the corresponding tone waveform. We could not come up with a solution to this problem as it would require knowledge about the number of real onsets.

Another fine tuning method we presented was the lateral inhibition which should assure that the non-zero elements of the steering vector keep a minimum distance to each other. The lateral inhibition was accomplished by negative feedback from the elements in a predefined neighbourhood each vector element. We presented an elegant way of adding the lateral inhibition in form of an autocorrelation term in the non-sparseness cost function thus conceptually simplifying the derivation of the update formulas for the steering vector. Unfortunately as this fine tuning method did not work as expected we had to drop it.

We also discussed phase matching which was first introduced in the first approach. This time only we had no per-onset fine tuning option and thus had to take the input signal and template upsampling approach in order to obtain the higher template matching precision. Unfortunately, this aggravates the memory space problem we already mentioned, due to the increased amount of data.

Though we invested some considerable amount of time in the development of this approach it still has much room for improvement. We begin with the initialization procedure which like in the first template-based approach has a random component which should be substituted by some more elaborate deterministic method. Then, we want to mention the calculation of the weighting parameter keeping the two cost functions in balance. This parameter is actually set to make the non-sparseness cost a fourth of the reconstruction error cost. The parameter is important because unbalance of the cost functions will either yield to an almost trivial solution using too many onsets but estimating the input signal very well or it will make the algorithm adjust the templates to represent one slice of the input signal perfectly leading to a maximally sparse vector. So a better estimation of the parameter is expected to greatly improve the separation quality. As the optimal value depends on the structure of the musical piece some data mining using a large set of songs might reveal some rules for better estimation of this parameter.

We already suggested a solution for the memory problem. It remains to be tested how convergence and separation quality is affected by the proposed method using subiterations.

Most work is left to be done in the fine tuning of the algorithm. The non-sparseness cost function is very far from optimal and we believe that it is possible to find a better one. The cost function should ideally indicate how far the steering vector is from the real onset vector. More precisely, the cost function should be built on an estimate of the number of real onsets for each steering vector. Currently this number is always estimated to be one. Then the function should also be able to favour configurations of locations of the non-zero elements which are more probable given some background knowledge on where these onsets are usually situated. For example, the onsets should be synchronized with the metrum of the song which means that they may only lie near some points of a raster derived from that metrum.

Lateral inhibition might constitute a solution, or part of a solution, for the improved non-sparseness cost function. It needs some improvements over the method described in this work as it did not lead to a perceivable enhancement of the separation quality. The lateral inhibition has to be designed so as to not kill the strongest weight in presence of only small surrounding weights in the steering vector, but at the same time suppress weak weights. One improvement was already suggested and that is optimizing the weights of the lateral inhibition weight function, but also other improvements are possible.

Some other improvements could be the inclusion of stereo cues into the non-sparseness cost function in order to better estimate the probable locations of the onsets. This could be done by lateral excitations weights situated between the channels for example.

One last improvement we have thought of would be the exact calculation of the tone waveform. With a given steering vector we can calculate its corresponding tone waveform by deconvolution which can be done fast by means of the FFT. This exact calculation would probably ameliorate the problem we have observed that the tone waveform is hardly changed by the algorithm with the steering vector being the one which is primarily adapted.

Summing up there is much work left to do for this algorithm. But we believe that it has potential as there are many possibilities left for what subalgorithms to incorporate.

Chapter 5

Blind Source Separation Approach

5.1 Motivation

After building the two template-based algorithms we realized that their assumption of repeating tones was somehow restrictive as they do not perform very well on classic or live recorded music. In order to overcome this restriction we decided to seek another substantially different way to separate instruments.

We thought about making an algorithm which is biologically more plausible and decided therefore to focus more on stereo cues as the human auditory system also relies on them for separating sounds. Humans and presumably also other animals are even capable of distinguishing sounds they have never heard before which means that much of the information needed to segregate audio sources must lie in the available stereo cues and in the structure of the sources themselves. We also know that the ear is doing a frequency analysis which means we basically hear frequencies and not amplitudes of some small time slices as it is the case in our former approaches where only the tone waveforms were considered. In the frequency domain the signals are better decorrelated and harmonic tones can be detected more easily due to their typical patterns they generate in the spectrum, which should facilitate our quest of separating musical sound sources which are usually harmonic.

So after some literature work we came across the work of Master [20] and of Yilmaz and Rickard [40] where sources are separated mainly by stereo cues. We decided to build upon their works as they described exactly what we intended to do. So basically the outcome is now a mixture of their works with some improvements where the idea of the main algorithm is taken from [40] and some features taken from [20].

5.2 Overview

We first introduce the concept of the two dimensional magnitude-phase-frequency histogram in Section 5.4 showing the magnitude and phase differences between two channels at different frequencies. Based on the assumption that every instrument has its own spatial location meaning that every instrument has an magnitude ratio and time shift of its own, we develop an algorithm in Section 5.5 which clusters this histogram in order to find accumulations of those differences, and uses this information to separate the instruments in the frequency domain.

Some fine tuning possibilities are then discussed in Section 5.6. Here we focus specifically on how long to choose the FFT window size and how many overlaps to use in order to improve separation quality and suppress musical noise artifacts.

Finally a summary is given and some possibilities for future work are discussed in Section 5.8.

5.3 Problem Reformulation

We will base the problem formulation here on Section 1.2 where the basics were described. Our goal is now to separate the waveforms of the instruments from a musical piece in such a way that they match the real instruments which were played as closely as possible. We will do the separation in the frequency domain because the frequency analysis already does some separation work by decorrelating the signals. The individual frequencies in the spectrum usually originate from one single instrument with some small interference, with some exceptions where the frequencies may overlap, thus we now need only to assign these frequencies to their corresponding instruments.

The frequency picking is usually done via time-frequency masking where a binary mask is created for each instrument which ideally selects only those frequencies in the spectrum where the target instrument has most of the energy. The search for a good mask is now what our algorithm will have to perform.

Moving to more concrete terms we apply an overlapped windowing on the input signal $\mathbf{x}_i$ with window length $\theta_w \in \mathbb{N}$ and $0 < \theta_r < 1$ overlap resulting in

$$W = \left\lceil \frac{T}{\theta_w \left(1 - \theta_r\right)} \right\rceil \tag{5.1}$$

windows. As the windows may need samples from the input signal after time T, we extend the signal with zeroes in order to be able to generate all windows. Afterwards we multiply the content of each window with Hanning-type weights resulting in

$$\mathfrak{x}_{i,w,t_w} = \sin\left(\frac{\pi\left(t_w - 1\right)}{\theta_w}\right) x_{i,t+\theta_w w\left(1-\frac{1}{\theta_r}\right)+t_w-1} \tag{5.2}$$

where t_w is the time index for the window which ranges from 1 to θ_w, and $\mathfrak{x}_{i,w,t_w}$ is the windowed and weighted input signal, and w is the index of the window with $1 \leq w \leq W$.

Now denoting the frequency transformation as $\mathcal{F}\left(\mathfrak{x}_{i,w}\right)$ and the resulting complex spectrum vector $\mathfrak{X}_{i,w}$ we get

$$\mathfrak{X}_{i,w} = \mathcal{F}\left(\mathfrak{x}_{i,w}\right) \tag{5.3}$$

The magnitude of an element of the spectrum is denoted $|\mathfrak{X}_{i,w,f_\theta}|$ where the index f_θ denotes the frequency bin with $1 \leq f_\theta \leq \theta_w/2$. The frequency of each bin f can be calculated as

$$f = f_\theta \frac{F}{\theta_w} \tag{5.4}$$

where F is the sampling frequency. Furthermore we denote the phase as $\angle\left(\mathfrak{X}_{i,w,f_\theta}\right)$ which we measure in radians. As the numbers are complex we denote the real part as $\Re\left(\mathfrak{X}_{i,w,f_\theta}\right)$ and imaginary part as $\Im\left(\mathfrak{X}_{i,w,f_\theta}\right)$.

5.4 Magnitude-Shift-Frequency Histogram

In order to detect the locations of the various instruments we make use of a histogram showing the relationship between inter-channel magnitude ratio and time shift at different frequencies. We note that these two parameters which represent stereo cues correspond roughly to the interaural intensity difference (IID) and the interaural time delay (ITD). This is no one to one correspondence as the IID is a difference while the magnitude ratio is obviously not.

We begin first with the simpler version of the histogram which shows only the magnitude ratio and time shift which resembles the magnitude-shift histogram of Yilmaz and Rickard in [40]. Our histogram uses on the x-axis the magnitude phase

$$\angle\left(|\mathfrak{X}_{1,w,f_\theta}|, |\mathfrak{X}_{2,w,f_\theta}|\right) = \arctan\frac{|\mathfrak{X}_{1,w,f_\theta}|}{|\mathfrak{X}_{2,w,f_\theta}|} \tag{5.5}$$

resulting from the ratio of the magnitudes of the two channels where the magnitude phase values remain in the range of $[0, ..., \pi/2]$ because magnitudes are always positive. We decided to use the angular notation because this way we can represent all ratios as big as they may be within the boundaries of our histogram range. If we would use the ratios directly we would always have some points lying outside the boundaries of the histogram no matter how large they are, as the ratio may theoretically assume values up to infinity. Another reason we use the magnitude phase is that it is hard to properly visualize a ratio as half of the values are in the range of $[0, ..., 1]$ and the other half in $(1, ..., \infty]$. By converting the ratios to radians we obtain a range of $[0, ..., \pi/4]$ and $(\pi/4, ..., \pi/2]$ respectively which results in an evenly distributed plot.

The y-axis shows the time shift $\delta_{f_\theta}\left(\angle\left(\mathcal{X}_{1,w,f_\theta}\right), \angle(\mathcal{X}_{2,w,f_\theta})\right)$ which is calculated from the circular phase difference $\Delta\left(\angle\left(\mathcal{X}_{1,w,f_\theta}\right), \angle(\mathcal{X}_{2,w,f_\theta})\right)$

$$\delta_{f_\theta}\left(\angle\left(\mathcal{X}_{1,w,f_\theta}\right), \angle(\mathcal{X}_{2,w,f_\theta})\right) = \frac{\Delta\left(\angle\left(\mathcal{X}_{1,w,f_\theta}\right), \angle(\mathcal{X}_{2,w,f_\theta})\right)\theta_w}{2\pi f_\theta} \tag{5.6}$$

where $\Delta(\,\cdot\,)$ is the circular difference operator. Circular means here that the difference wraps around π and $-\pi$ due to the periodic nature of phases.

$$\Delta(y, z) = \begin{cases} y - z & |y - z| \leq \pi \\ y - z + \pi & y - z < -\pi \\ y - z - \pi & y - z > \pi \end{cases} \tag{5.7}$$

The circular operator used during time shift calculation now poses a problem whenever the difference wraps around $\pm\pi$ as the resulting time shifts from Equation 5.6 become ambiguous. This may happen with frequencies having a wavelength smaller than a specific threshold which is determined by the spacing of the microphones used for recording. If a sound wave is smaller than double the microphone spacing one can not determine whether a shift is greater than ± one half of the wavelength or not. This comes from the fact that a shift of more than ± one half wavelength causes the phase difference to wrap around and result in a shift of less than ± one half wavelength and so we always get a shift of less than ± one half wavelength from Equation 5.6 even if it is not the case.

The ambiguity problem can be ameliorated by some heuristics which will be described in more detail in Section 5.6. As for now we will ignore frequencies having a wavelength smaller than double the microphone spacing.

Our histogram has a fixed resolution of $\theta_v \times \theta_v$ bins $v_{a,b}$ where $\theta_v \in \mathbb{N}$ is the number of bins per axis and a, b are indices for the magnitude phase and time shift respectively with values in the range $[1, ..., \theta_v]$. Having a fixed resolution means we have to downsample the data points to that resolution where the data point is defined as a magnitude phase and time shift pair for a frequency bin f_θ. In order to calculate the value of a bin we have to introduce the sampled magnitude phase $\phi_\angle\left(\mathcal{X}_{1,w,f_\theta}, \mathcal{X}_{2,w,f_\theta}\right)$ and time shift $\phi_\delta\left(\mathcal{X}_{1,w,f_\theta}, \mathcal{X}_{2,w,f_\theta}\right)$ with

$$\phi_\angle\left(\mathcal{X}_{1,w,f_\theta}, \mathcal{X}_{2,w,f_\theta}\right) = \left\lfloor \frac{\theta_v - 1}{2\pi}\left(\angle\left(\left|\mathcal{X}_{1,w,f_\theta}\right|, \left|\mathcal{X}_{2,w,f_\theta}\right|\right) + \pi\right)\right\rceil + 1 \tag{5.8}$$

$$\phi_\delta\left(\mathcal{X}_{1,w,f_\theta}, \mathcal{X}_{2,w,f_\theta}\right) = \left\lfloor \frac{\theta_v - 1}{2t_{max}}\left(\delta_{f_\theta}\left(\angle\left(\mathcal{X}_{1,w,f_\theta}\right), \angle(\mathcal{X}_{2,w,f_\theta})\right) + t_{max}\right)\right\rceil + 1 \tag{5.9}$$

where the brackets $[\,\cdot\,]$ denote rounding to the nearest integer and t_{max} the maximum non-ambiguous time shift defined as

$$t_{max} = \frac{d_{mic}F}{c_{air}} \tag{5.10}$$

where c_{air} is the velocity of sound in m/s and d_{mic} the distance between the two microphones in meters. A more detailed explanation of the maximum non-ambiguous time shift and frequency threshold will be given in Section 5.6. The energy of the bins is now computed as a magnitude-weighted, normalized sum over the entire set of windows resulting in

$$v_{a,b} = \frac{\sum\limits_{w=1}^{W}\sum\limits_{f_\theta=1}^{\theta_n} I_{a,b}\left(\phi_\angle\left(\mathfrak{X}_{1,w,f_\theta}\mathfrak{X}_{2,w,f_\theta}\right), \phi_\delta\left(\mathfrak{X}_{1,w,f_\theta}, \mathfrak{X}_{2,w,f_\theta}\right)\right)\left(|\mathfrak{X}_{1,w,f_\theta}| + |\mathfrak{X}_{2,w,f_\theta}|\right)}{\sum\limits_{w=1}^{W}\sum\limits_{f_\theta=1}^{\theta_n}\left(|\mathfrak{X}_{1,w,f_\theta}| + |\mathfrak{X}_{2,w,f_\theta}|\right)} \tag{5.11}$$

where $\theta_n \in \mathbb{N}$ is the cut-off frequency for the histogram with $1 < \theta_n \leq \theta_w$, and $I_{a,b}\left(\,\cdot\,\right)$ is an indicator function defined as

$$I_{a,b}(y,z) = \begin{cases} 1 & a = y \wedge b = z \\ 0 & a \neq y \vee b \neq z \end{cases} \tag{5.12}$$

In words we first sample the magnitude phase and time shift and stretch them to be in the range $[1, ..., \theta_v]$ using the sampling functions $\phi_\angle$ and $\phi\delta$ and then calculate the "intensity" values of each bin by doing a normalized summation over all pairs of magnitude phases and time shifts in all windows and all frequencies in the range limited by θ_n. The normalization is applied in order to make the intensity values independent of the size T of the input signal and its average loudness. The intensities now represent the average energy over the entire input at their magnitude-shift coordinates. Now if an instrument is played in a specific location we expect it to have a constant magnitude phase and time shift across the entire spectrum which should result in a high intensity value at the bin with the sampled coordinates matching these two parameters.

So we should be able to separate that instrument if we are able to find these high intensity spots in the histogram. Their coordinates would then indicate the parameters of their respective instruments we will have to look for in the spectrum. So if we find a pair $(\mathfrak{X}_{1,w,f_\theta}, \mathfrak{X}_{2,w,f_\theta})$ having these parameters we can copy them to the spectrum of the respective instrument.

Unfortunately this procedure is not as easy as it may seem. The frequency bins $\mathfrak{X}_{i,w,f_\theta}$ are not generated solely by one instrument but are "polluted" to a specific degree by the other instruments or by its own echo due to reverberation and harmonic overlaps in the spectrum. Therefore the spots in the histogram become smeared over several histogram bins resulting in high intensity clusters which usually overlap by some degree.

The clusters often overlap or in some situations even get absorbed in other clusters as for example whenever a loud playing instrument is situated near a weaker one, the loud instrument will usually have a bigger cluster which will merge with the smaller cluster of the weak instrument. But in order to not drown the weak instrument during the musical performance, it is either played whenever the stronger one is not or it will have another frequency than the loud instrument.

In the simple magnitude-phase histogram the frequency of an instrument is not taken into account and therefore we would not be able to distinguish both instruments in the above example. As our histogram is made up of greyscale values when visualized, we had the idea to extend it by visualizing the frequency as a colour. In this way we should now be able to distinguish overlapping clusters with different frequencies.

Introducing the frequency in our histogram means that we have to redefine the bins as three dimensional vectors $\mathbf{v}_{a,b}$ where the first element is interpreted as a red, the second as green and the third as blue intensity. Then we will need a frequency to colour mapping function $\psi(f_\theta)$ which maps each frequency to a colour vector where we require the colour vector returned to be of unity length in order to preserve luminosity of the respective colour. Using these definitions we can rewrite Equation 5.11 to include the colour mapping function

$$\mathbf{v}_{a,b} = \frac{\sum_{w=1}^{W} \sum_{f_\theta=1}^{\theta_n} I_{a,b}\left(\phi_\angle\left(\mathfrak{X}_{1,w,f_\theta}\mathfrak{X}_{2,w,f_\theta}\right), \phi_\delta\left(\mathfrak{X}_{1,w,f_\theta}, \mathfrak{X}_{2,w,f_\theta}\right)\right)\psi(f_\theta)\left(|\mathfrak{X}_{1,w,f_\theta}| + |\mathfrak{X}_{2,w,f_\theta}|\right)}{\sum_{w=1}^{W} \sum_{f_\theta=1}^{\theta_n}\left(|\mathfrak{X}_{1,w,f_\theta}| + |\mathfrak{X}_{2,w,f_\theta}|\right)} \tag{5.13}$$

Usually the frequency mapping function is used to generate continuous colour gradations between some predetermined key colours as the frequency parameter increases. We found a good colour mapping to include key colours in that order: red $(1,0,0)^T$, middle yellow $(1/2,1/2,0)^T$, green $(0,1,0)^T$, middle cyan $(0,1/2,1/2)^T$ and blue $(0,0,1)^T$. This mapping follows the colours of the spectrum and uses some common associations as red with low frequencies or bass and blue with high frequencies or trebles.

Looking at the colour mapping above we may observe that the colours may not have the same luminosity at the same values. For example green at its maximum intensity $(0,1,0)^T$ will be brighter than blue at its maximum intensity. We deemed this issue not to be that important to investigate further, so we left as a possible topic for future work whether generating colours of equal brightness will improve results.

As we have seen, the frequency colour mapping in the histogram really works and clusters can be now separated due to their differing centre frequencies where the centre frequency is the average frequency for that cluster. Now the interesting question arises whether extending the histogram bin vectors $\mathbf{v}_{a,b}$ by more dimensions would give better results in the main algorithm which uses this histogram for clustering. This would of course abandon the idea of visualizing the histogram or at least make it more challenging as we cannot simply add more colour channels.

Generally we can view this histogram as a kind of feature space which is used by the clustering algorithms described in the next section. Visualization although being a good idea is not really necessary which means we could render this feature space more complex in order to possibly be able to better separate the frequency bins from each instrument. This would be an interesting topic to be further elaborated upon in future work.

5.5 Algorithm

We have already hinted some of the methods used in the algorithm during the previous section. The algorithm begins with converting the input signal into the spectral domain representation $\mathfrak{X}_{i,w,f_\theta}$ which is obtained using the windowed frequency transform described in Section 5.3. We then calculate the Magnitude-Shift-Frequency histogram described in the previous section in order to obtain the histogram bins $\mathbf{v}_{a,b}$.

The histogram will now be used to obtain a magnitude difference and time shift range for each instrument. More precisely we will use clustering for parting the histogram into areas belonging mostly to one instrument.

Clustering

The goals during clustering are finding the number of instruments N and splitting the histogram into N parts so that each part's associated instrument accounts for most of the energy contained in the histogram bins belonging to that part.

As we have no prior knowledge about how much energy an instrument contributes to each histogram bin, we have to use some heuristics. Here we observe that the energy in the histogram tends to cluster around some values. Assuming that the instruments remain on a fixed location during the musical piece, they will have some fixed magnitude difference and time shift parameters which should show up as high energy points in the histogram. Due to different interference effects the parameters will exhibit some variance leading to the observed clusters. Now by applying some algorithm identifying these clusters and approximating their extent we obtain an area for each instrument in the histogram approximating its parameter range. This result can then be used to separate the instruments in the next stage by assigning each frequency bin pair $(\mathfrak{X}_{1,w,f_\theta}, \mathfrak{X}_{2,w,f_\theta})$ the instrument associated with the histogram area the pair belongs to due to its parameters.

As we saw during development of different clustering algorithms, it is problematic to estimate N, due to overlapping clusters and complex cluster forms. Because we ran out of time to further investigate on solutions to that problem we decided to make N an input parameter which has to be supplied by the user. But as we deem the automatic estimation of the number of instruments to be crucial for usability of the algorithm we see this issue as to be important topic for future work.

Our first clustering algorithm is a slightly modified version of the K-means clustering. The vanilla version of the algorithm works iteratively by first assigning the data points to their nearest cluster centres $\boldsymbol{\mu}_j$ with $1 < j < N$ and

$$\mathcal{M}_{\arg\min_j \left\|\binom{a}{b}-\boldsymbol{\mu}_j\right\|} = \mathcal{M}_j \cup (a,b) \tag{5.14}$$

for all indices (a,b) with $1 < a < \theta_v$ and $1 < b < \theta_v$ where $\mathcal{M}_j$ is the set of histogram bin indices assigned to the cluster j. In the second step it is recalculating the cluster centres as the mean of its assigned data point locations

$$\boldsymbol{\mu}_j = \frac{1}{|\mathcal{M}_j|} \sum_{(a,b)\in\mathcal{M}_j} \begin{pmatrix} a \\ b \end{pmatrix} \tag{5.15}$$

As long data points change their assigned centre after each iteration, the centres will keep moving. Usually the centres will move to locations containing a high data point density until convergence. The algorithm is iterated either until no data point changes its cluster or until a fixed number of iterations is reached. The initialization is done by randomly positioning the cluster centres in the histogram. As the algorithm implicitly tries to minimize the sum of squared distances between the data points and their nearest cluster, which can be written as

$$\mathscr{C} = \sum_{j}^{N} \sum_{(a,b)\in\mathcal{M}_j} \left\| \begin{pmatrix} a \\ b \end{pmatrix} - \boldsymbol{\mu}_j \right\|^2 \tag{5.16}$$

we can use this cost function as a quality measure for the result.

Now the simple K-means clustering will not work for our histogram as only the locations of the histogram bins are accounted for and not the energy content of each bin. But with the simple modification of calculating the cluster centres by the energy-weighted mean of the bin locations instead of the simple mean, the algorithm is able to return usable results. The cluster centers are now calculated as

$$\boldsymbol{\mu}_j = \frac{1}{|\mathcal{M}_j|} \sum_{(a,b)\in\mathcal{M}_j} v_{a,b} \begin{pmatrix} a \\ b \end{pmatrix} \tag{5.17}$$

and consequently the cost function for measuring the quality of the result changes to

$$\mathscr{C} = \sum_{j}^{N} \sum_{(a,b) \in \mathcal{M}_j} v_{a,b} \left\| \begin{pmatrix} a \\ b \end{pmatrix} - \boldsymbol{\mu}_j \right\|^2 \tag{5.18}$$

Basically, the modified clustering algorithm now estimates the energy density in the histogram rather than the bin density which is always constant, thus leading to our desired clustering. We note that we use only the simple magnitude-shift histogram bins $v_{a,b}$ in the equations above. We did not test the algorithm with the colour-extended histogram version.

After implementation of the algorithm we saw that the results were poor and noticed that $\mathscr{C}$ varied drastically on each run. According to literature this outcome is not unexpected as the standard algorithm for computing the K-means clustering is susceptible to local minima and thus the results depend strongly on the initialization. We might have tried to ameliorate this problem by implementing some suggested improvements as doing multiple runs using different initializations or doing some intelligent initialization based on some pre-processing but due to the very weak performance we decided to rather try another algorithm.

We decided to use a radial-basis function network (RBFN) with supervised learning for our second clustering approach. A radial basis function (RBF) has a centroid vector $\boldsymbol{\mu}_j$ where its value peaks at its maximum $\mathbf{w}_j$. The value falls as the input vector - in our case the indices of the histogram bins (a, b) - gets farther away from the RBF centre. The fall off rate in each dimension is controlled by the covariance matrix $\boldsymbol{\Sigma}_j$. Now a RBFN is simply an additive mixture of several RBFs. The network tries to estimate the energy density directly using the following cost function

$$\mathscr{C} = \frac{1}{2} \sum_{a=1}^{\theta_v} \sum_{b=1}^{\theta_v} \left\| \mathbf{v}_{a,b} - \hat{\mathbf{v}}_{a,b} \right\|^2 \tag{5.19}$$

where $\hat{v}_{a,b}$ is the estimated energy at the bin with index (a, b). If we use gaussians of the form of

$$G_{a,b}\left(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j^{-1}\right) = \exp\left(-\frac{1}{2}\left(\begin{pmatrix} a \\ b \end{pmatrix} - \boldsymbol{\mu}_j\right)^T \boldsymbol{\Sigma}_j^{-1} \left(\begin{pmatrix} a \\ b \end{pmatrix} - \boldsymbol{\mu}_j\right)\right) \tag{5.20}$$

for the RBFs, the estimated energy can be calculated as

$$\hat{\mathbf{v}}_{a,b} = \sum_{j=1}^{N} \mathbf{w}_j G_{a,b}\left(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j^{-1}\right) \tag{5.21}$$

where $\boldsymbol{\Sigma}_j^{-1}$ is the inverse of the covariance matrix.

For adjusting the three parameter types $\mathbf{w}_j$, $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j^{-1}$ we use the iterative gradient descent method from [13] which we have extended for multidimensional output in order to account for the colour in the magnitude-shift-frequency histogram. As the algorithm is iterative we will write the three parameter types, the cost function $\mathscr{C}$ and the estimated energy value $\hat{\mathbf{v}}_{a,b}$ as functions of the actual iteration n. Now the update equations using this method are

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) - \eta_w \Delta \mathbf{w}_j(n) \tag{5.22}$$

$$\boldsymbol{\mu}_j(n+1) = \boldsymbol{\mu}_j(n) - \eta_\mu \Delta \boldsymbol{\mu}_j(n) \tag{5.23}$$

$$\boldsymbol{\Sigma}_j^{-1}(n+1) = \boldsymbol{\Sigma}_j^{-1}(n) - \eta_\Sigma \Delta \boldsymbol{\Sigma}_j^{-1}(n) \tag{5.24}$$

where $0 < \eta_w < 1$, $0 < \eta_\mu < 1$ and $0 < \eta_\Sigma < 1$ are learning parameters and $\Delta w_j(n)$, $\Delta\boldsymbol{\mu}_j(n)$ and $\Delta\boldsymbol{\Sigma}_j^{-1}(n)$ are the gradients of the respective parameters which are defined by

$$\Delta \mathbf{w}_j(n) = \frac{\partial \mathscr{C}(n)}{\partial\, \mathbf{w}_j(n)} \tag{5.25}$$

$$\Delta \boldsymbol{\mu}_j(n) = \frac{\partial \mathscr{C}(n)}{\partial\, \boldsymbol{\mu}_j(n)} \tag{5.26}$$

$$\Delta \boldsymbol{\Sigma}_j^{-1}(n) = \frac{\partial \mathscr{C}(n)}{\partial\, \boldsymbol{\Sigma}_j^{-1}(n)} \tag{5.27}$$

We note here that we maintain and update the inverse of the covariance matrix $\boldsymbol{\Sigma}_j$ directly thus we can omit the matrix inversion and save some computational time. For the partial derivatives we first define the error vector $\mathbf{e}_{a,b}(n)$ as

$$\mathbf{e}_{a,b}(n) = \mathbf{v}_{a,b} - \hat{\mathbf{v}}_{a,b}(n) \tag{5.28}$$

and the difference-to-centre vector $\boldsymbol{\delta}_{j,a,b}(n)$ as

$$\boldsymbol{\delta}_{j,a,b}(n) = \begin{pmatrix} a \\ b \end{pmatrix} - \boldsymbol{\mu}_j(n) \tag{5.29}$$

then writing the equations for the partial derivatives themselves as

$$\frac{\partial \mathscr{C}(n)}{\partial\, \mathbf{w}_j(n)} = \sum_{a=1}^{\theta_v} \sum_{b=1}^{\theta_v} \mathbf{e}_{a,b}(n) G_{a,b}\left(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j^{-1}\right) \tag{5.30}$$

$$\frac{\partial \mathscr{C}(n)}{\partial\, \boldsymbol{\mu}_j(n)} = 2\sum_{c=1}^{C} w_{j,c}(n) \sum_{a=1}^{\theta_v} \sum_{b=1}^{\theta_v} e_{a,b,c}(n) G_{a,b}\left(\boldsymbol{\mu}_j(n), \boldsymbol{\Sigma}_j^{-1}(n)\right) \boldsymbol{\Sigma}_j^{-1}(n) \boldsymbol{\delta}_{j,a,b}(n) \tag{5.31}$$

$$\frac{\partial \mathscr{C}(n)}{\partial\, \boldsymbol{\Sigma}_j^{-1}(n)} = -\sum_{c=1}^{C} w_{j,c}(n) \sum_{a=1}^{\theta_v} \sum_{b=1}^{\theta_v} e_{a,b,c}(n) G_{a,b}\left(\boldsymbol{\mu}_j(n), \boldsymbol{\Sigma}_j^{-1}(n)\right) \boldsymbol{\delta}_{j,a,b}(n) \boldsymbol{\delta}_{j,a,b}^T(n) \tag{5.32}$$

where C denotes the number of colour channels which is usually 3 for red, green and blue.

The initialization of the RBFN is rather simple. The centroid vectors $\boldsymbol{\mu}_j(1)$ are randomly distributed across the histogram, the inverse covariance matrices $\boldsymbol{\Sigma}_j^{-1}(1)$ are initialized as

$$\boldsymbol{\Sigma}_j^{-1}(1) = \begin{pmatrix} 0.1 & 0.001 \\ 0.001 & 0.1 \end{pmatrix} \tag{5.33}$$

and the output weights $\mathbf{w}_j(1)$ are assigned $1/N$ to each vector element.

After some preliminary tests we observed that the output weights may become negative for one or more vector elements. This configuration has to be avoided as densities can only be positive. Therefore we clamped all negative weight vector elements to zero after every iteration. Now such configuration with negative weights may happen to lead to a better overall estimation of the energy density but we are interested in the clusters which are represented by the RBFs and not in a good overall estimation of the energy density.

We also made some further observations. In particular we noted that the centroid vectors and the inverse covariance converged very fast to local optima while the output weights adapted steadily. The solution we found for this problem was to adapt only the weight vectors for some iterations and then adapt all parameters for the remaining iterations after this initial start-up. This way we avoided getting distorted gradients for the centroids and the inverse covariance matrix caused by the undeveloped weights during the first iterations thus allowing the weight vectors to settle first and then also adjusting the other two parameters.

There were also two other important problems. The first one was that the centroids sometimes tended to move to coordinates beyond the histogram range of $[1..\theta_v] \times [1..\theta_v]$. Our explanation of this phenomenon was that in a suboptimal configuration the RBFs lying at the borders of the histogram would have to "jump" over the ones round the middle in order to reach their optimal position. But they cannot jump over these RBFs as it would temporarily rise the cost function so they will rather try to get as far away from the RBFs in the centre as possible. If we would have allowed negative weights is might have been possible that this problem would not have occurred as in that case every RBF could contribute to minimize the cost function inside the histogram range of coordinates. As a solution we restricted the centroids to be able to move only inside the coordinate space belonging to the histogram. There might also be other solutions, as for example trying to put the centroid on every possible coordinate inside the range and check whether the cost functions decreases. As the range is discrete and usually small the computational cost should not increase much by doing this. We have not tried this or other alternative solutions as the one we first found also worked well because after some iterations at the border the centroids sometimes got back into more central positions.

The second problem we observed was that sometimes the inverse covariance matrix began to grow to very high values thus making the RBF having a very small spread. This may have the same cause as the first problem as the RBFs are not in their optimal location and thus would have to jump over other RBFs temporarily rising the cost function. But this time the RBF is trapped in the middle of other RBFs and thus the best solution to decrease the cost function would be for the RBF to become very small and specialize on solely one histogram bin. This solution is also unacceptable for us and therefore we have to counter this problem by restricting the inverse covariance matrix to not get larger than $\theta_{\Sigma_{max}}$ on any of its elements. For practical purposes we saw that a value of 10000 for $\theta_{\Sigma_{max}}$ is sufficient for a RBF to spread over small clusters and at the same time not getting as small as one histogram bin.

As the RBFN algorithm had a slow convergence behaviour like all gradient descent methods we decided to improve convergence with RPROP [30] as we did in Section 4.5 in the previous chapter. We will not describe the algorithm in detail once again here but refer to the description in Section 4.5 on Page 63. Some implementation specific details we want to mention here are:

- the estimated gradients are computed element wise for each vector and matrix quantity. That means that we treat the elements of vectors or matrices as being independent.
- that we use different η_+ for the output weight vector elements and the other parameters. For the output weights we use a value of 1.1 and for all other parameters a more prudent value of 1.01. η_- is set to $1/2$ and remains the same for all parameters.

Now after the RBFN was trained, we have to assign a RBF to each histogram bin in order to get our parameter range areas for our instruments. At this point we shall mention that we will not take the different colours of the RBFs into account as there is no possibility to use the colour information during the RBF to bin assignment. The colours generated by the presence of different frequencies were only used to train the RBF as it can better cluster the histogram using the colour information than without it. We will collapse the three colour components into one value by summing over all three responses for each colour channel. The assignment procedure now becomes:

- initialize all sets $\mathcal{M}_j$ of indices of histogram bins assigned to cluster j with $\varnothing$
- do for all histogram bins $\mathbf{v}_{a,b}$
  - get the RBF m with the maximal activation value

$$m = \arg\max_{j} \sum_{c=1}^{C} \mathbf{w}_{j,c} G_{a,b}\left(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j^{-1}\right) \tag{5.34}$$

and set

$$\mathcal{M}_m = \mathcal{M}_m \cup (a, b) \tag{5.35}$$

After the assignment is complete we have enough information to proceed to the separation stage.

Separation

After the histogram has been clustered and each histogram bin has been assigned an instrument, we can proceed by assigning each frequency bin pair $(\mathfrak{X}_{1,w,f_\theta}, \mathfrak{X}_{2,w,f_\theta})$ from the input signal to an instrument depending on which histogram bin it can be mapped to using the sampling Equations 5.8 and 5.9. Now we are able to separate the input signal by copying the frequency bin pairs to the spectrum of their assigned instruments where the spectrum was previously initialized with zeroes.

We obtain the windowed time domain signal for each separated instrument $\mathfrak{x}_{i,j,w}$ we use the inverse of the frequency transform $\mathcal{F}^{-1}$ resulting in

$$\mathfrak{x}_{i,j,w} = \mathcal{F}^{-1}(\mathfrak{X}_{i,j,w}) \tag{5.36}$$

Now we have to undo the overlapped windowing in order to obtain the whole time domain signal. But if we would try to undo the windowing by simply dividing through the Hanning-type weights which we initially used to generate the windows, we would introduce severe artefacts at both ends of the window. Those artefacts were introduced during separation of the frequency bin pairs, and are more or less equally distributed within the window. As the Hanning-type weights decay to 0 at both ends, dividing by these weights would strongly amplify any artefacts introduced. Not dividing by the weights is also not an option because adding the overlapped windows would then result into a signal without constant gain, meaning there will be some audible beat. In addition this would lead to audible cracks and pops due to the non-zero gain at the ends of the windows which would result into discontinuities in the summed signal.

Here we note some interesting properties of Hanning windows

- if the Hanning-type weights, which represent a positive sine wave, are multiplied once more with the window we get a squared sine.
- at 50% overlap for all odd numbered windows the squared sine generated in the previous point behaves like a squared cosine.
- so in conclusion, simply adding 50% overlapped windows which were multiplied once more with the Hanning-type weights will result into a signal reconstruction with constant gain of one. This comes from the fact that for each target sample we will have a sample from a squared sine and a sample from a squared cosine window overlapping which will add up to have unity gain due to the relation $\sin^2 + \cos^2 = 1$.
- each overlap $\theta_r = 1 - 1/2^z$ with $z \in \mathbb{N}$ of squared sine shaped windows will have squared sinus-cosinus pairs which add resulting in a gain of 2^{z-1}.

Multiplying once more with Hanning-type weights reduces the artefacts at the borders to zero due to the zero-gain behaviour of the weights at both ends of the window. This way we are able to solve the border-artefact problem. Now also having a power of two overlap we can reconstruct the unity gain time domain signal for each instrument which is the goal of this separation algorithm.

Summing up, the separation algorithm works as follows

- initialize all frequency bins $\mathfrak{X}_{i,j,w,f_\theta}$ of all instruments to zero.
- do for each frequency bin pair $(\mathfrak{X}_{1,w,f_\theta}, \mathfrak{X}_{2,w,f_\theta})$ of the input signal
  - calculate the sampled magnitude phase a and time shift b

    $$a = \phi_\angle\left(\mathfrak{X}_{1,w,f_\theta}, \mathfrak{X}_{2,w,f_\theta}\right) \tag{5.37}$$

    $$b = \phi_\delta\left(\mathfrak{X}_{1,w,f_\theta}, \mathfrak{X}_{2,w,f_\theta}\right) \tag{5.38}$$

  - update the frequency bins of the instrument assigned to the histogram bin $\mathbf{v}_{a,b}$ with the actual frequency bin pair in the input signal by

    $$\mathfrak{X}_{i,j,w,f_\theta} = \begin{cases} \mathfrak{X}_{i,w,f_\theta} & (a,b) \in \mathcal{M}_j \\ 0 & (a,b) \notin \mathcal{M}_j \end{cases} \tag{5.39}$$

    with $i = 1, 2$ and $j = 1..N$.
- undo the frequency domain transform

  $$\mathfrak{x}_{i,j,w} = \mathcal{F}^{-1}\left(\mathfrak{X}_{i,j,w}\right) \tag{5.40}$$

  for all channels, instruments and windows.
- multiply all samples of all instruments with Hanning-type window weights

  $$\mathfrak{x}'_{i,j,w} = \mathfrak{x}_{i,j,w} \sin\left(\frac{\pi\left(t_w - 1\right)}{\theta_w}\right) \tag{5.41}$$

- merge of each instrument's windows into one final time domain instrument signal $x_{i,j,t}$ by aligning and adding them on a sample by sample basis and then multiplying the result by $2\left(1 - \theta_r\right)$.

5.6 Fine Tuning

After describing the basic algorithm we can move on applying some fine tuning which we expect would raise the quality of the result. The algorithm will also work without the additional fine tuning but the output quality can be usually improved by only a few additional considerations.

Window Size and Overlap

The window size and overlap parameters have an important role in balancing different types of artefacts generated by the separation stage of the algorithm. The artefacts which can be controlled by the two parameters are

- Musical noise. This is a kind of noise which consists of short sine tones or a sum of such. The sound of this noise resembles somehow playing some random notes on a synthesizer. This is a very annoying artefact as it can hardly be filtered by the human brain and thus distracts much more from the signal of the separated instrument than standard white or coloured noise.

  The source of musical noise is the binary and time-independent decision whether a instrument spectrum is assigned a frequency bin pair or whether it remains zero. Therefore it may occur that a frequency bin pair with no other bins in the spectral neighbourhood is assigned to one instrument at for one window. Due to the missing neighbours, the frequency bin pair then sounds like a more or less pure tone. Now if the pair is not assigned to the same instrument for the next window, then the tone will fall silent. If it happens for a longer period of time that pairs without neighbours are assigned for the time span of only one window to an instrument and then changed to another instrument, then the result will sound like musical noise. It is important that the frequency bin pairs assigned to the same instrument have either no neighbours or very weak ones because if this is not the case, it becomes more improbable that the main pair with all its neighbours would not be assigned to the same instrument for the next window.

- Echo, pre-echo and reverberation. All three kinds of artefacts are all produced by the same cause. These three artefacts are better audible at longer window sizes. The cause is that often frequency bin pairs with a low magnitude are more susceptible to be polluted by echo and reverberation of other instruments or get distorted by harmonic sharing. Therefore they are often assigned to the false or to a catch-all instrument which contains only of frequencies bin pairs which could not be attributed to one of the other instruments and may therefore contain content of those instruments. Frequencies with weak magnitudes are usually created by transients in the time domain signal. These weak frequencies which are usually many compared to the strong ones thus sum up to reconstruct a transient. If these frequencies are missing then the transients become wider in time. For longer window lengths even short tones may get audibly widened, thus lasting longer than they should. This widening encountered in longer window lengths is then perceived as echo and reverberation. As the widening happens both forward and backward in time, the forward widening is perceived as pre-echo which is the most annoying of the three artefacts. For very long window lengths the pre-echo

> gets so long that it is perceived as a kind pre-reverberation announcing the tone that will be played at some point in the near future. This is a very strange effect and marks the upper limit of usable window size.
>
> There is also a connection to musical noise here. If for some time instant there are only weak frequencies where a few of them are at the threshold of being assigned to the instrument then the situation will be encountered where tones without neighbours in the spectrum are assigned to the instrument for only one window which is what we described as cause of musical noise.

After describing the artefacts we show a connection of the window length and overlap parameter and the artefact types. Obviously the longer the window length the more echo, pre-echo and reverberation we will encounter as they are best perceptible at longer window lengths. We found empirically that window lengths of about one third to a half of a second lead to an acceptable amount of these artefacts with longer windows having perceivable pre-echo and reverberation. Thus we can say that a window length of half of a second is the practical upper limit. We shall note here that smaller window lengths will decrease spectral resolution and thus separation capability. Therefore we should aim for the longest possible practical window length.

Now contrary to the echoes which have to be held imperceptible or at least keep them from becoming annoying, musical noise can be actually suppressed. This is done via the overlap parameter. The higher the overlap the less musical noise will be present. The reason here is that the spectral change from one window to the other becomes more blurred for each frequency. This way the frequency bin pair assignment becomes less time-independent. The higher overlap also causes the abrupt changes due to the binary nature of the assignment - either the pair is assigned to a specified instrument or it is not - to be smoothed over due to averaging effects. From another point of view it can be said that the high overlap makes the binary assignment to behave like a probability-weighted assignment due to averaging of the different windows consisting frequency pair binary assignments as the many time-independent binary decisions are averaged over to result in a mean which is not binary itself but reflects the probability of the pair to be assigned to that instrument. Assignments based on probabilities are therefore less susceptible to musical noise than binary decisions.

One downside of high overlaps is the increased computational time caused by the extra windows generated from the input signal. As we can use only powers in the calculation of the overlaps we have not much room for fine tuning the overlap parameter. We have decided to use an eight-fold overlap in our implementation, meaning $\theta_r = 87,5\%$ for a good balance between musical noise and computational time.

Time Shift Disambiguation

As we have already mentioned in Section 5.4 whenever the length of a sound wave becomes smaller than double the spacing between the recording microphones, we cannot measure their exact time shift due to ambiguities. This problem cannot be solved exactly as the information needed for disambiguation is not available in the signal, but we can use a heuristic in order to ameliorate that problem.

Moving to concrete terms we first have to determine the frequency threshold f_{amb} from where higher frequencies begin to become ambiguous

$$f_{amb} = \frac{c_{air}}{2d_{mic}} \tag{5.42}$$

where c_{air} is the velocity of sound in m/s and d_{mic} the distance between the two microphones in meters. Now we assuming that phase differences between the channels cluster around locations of instruments we can solve the ambiguity heuristically for all frequencies higher than f_{amb} by generating all possible phase differences for each frequency and picking the one which is situated next to a cluster centre. The possible phase differences can be generated either by adding or subtracting 2π from the phase difference, without wrapping, until the resulting time shift exceeds the range of maximum time shift $[-t_{max}, ..., t_{max}]$ with

$$t_{max} = \frac{F}{2f_{amb}} \tag{5.43}$$

This method assumes that the frequency bins corresponding to each frequency above the threshold contain magnitudes which are contributed from the most part by one instrument, or in other words this heuristic works with unpolluted bins. Those bins which are shared by two or more instruments will have a phase difference not corresponding to any of the clusters so this heuristic will probably assign an inappropriate cluster to such bins.

We shall note here that phase differences of frequencies above f_{amb} do “gradually” become ambiguous. As the wavelength becomes smaller the ambiguous values start at the extremes of the possible time shift interval and move inwards until all values in the interval become ambiguous. This observation is explained by the shrinking maximum non-ambiguous time shift interval for higher frequencies $[-t_{max,f}, ..., t_{max,f}]$ where $t_{max,f}$ which can be written as

$$t_{max,f} = \begin{cases} t_{max}\left(2 - \frac{2t_{max}f_{amb}}{F}\right) & f_{amb} < f < 2f_{amb} \\ 0 & f > 2f_{amb} \end{cases} \tag{5.44}$$

So only values lying in the maximum non-ambiguous time shift interval of f_{amb} but not in the maximum non-ambiguous time shift interval for the specific frequency are ambiguous. The higher the frequency the smaller its own maximum non-ambiguous time shift interval becomes and the more values become ambiguous until above a second frequency threshold all values become ambiguous. This gradually rising ambiguity also continues above that second threshold were we have to count the number of possible values. For some time shift values there will be three other possibilities where the true time shift may be and for some other values only two. This also goes on till a third frequency threshold is reached where all calculated values have three alternatives and the four alternative ambiguities start, and so on.

This gradually rising ambiguity now means that if we calculate a time shift of 0 for a frequency bin pair with $f_{amb} < f < 2f_{amb}$ then we can still be sure that it comes from an instrument playing in the middle of the two recording microphones even though this frequency bin pair will have its frequency above the threshold f_{amb}.

Unfortunately due to time constraints we had no time to implement the disambiguation heuristic so we left it as an issue for future work.

5.7 Final Algorithm

As we have done in the previous chapters we will give a summary of the algorithm we have discussed during the previous sections of this chapter. In this summary we do not include versions or improvements we were not able to implement due to different reasons and therefore the listed algorithm will be the final version which can be found in the source code generated during development of this thesis.

- apply an overlapped windowing on the input signal $\mathbf{x}_i$ with window length θ_w and θ_r overlap.
- multiply the content of each window with Hanning-type weights resulting in

  $$\mathfrak{x}_{i,w,t_w} = \sin\left(\frac{\pi\left(t_w - 1\right)}{\theta_w}\right) x_{i,t+\theta_w w\left(1-\frac{1}{\theta_r}\right)+t_w-1} \tag{5.45}$$

  where t_w is the time index for the window, and $\mathfrak{x}_{i,w,t_w}$ is the windowed and weighted input signal, and w is the index of the window.
- transform each window to obtain its spectrum vector $\mathcal{X}_{i,w}$ from

  $$\mathcal{X}_{i,w} = \mathcal{F}\left(\mathfrak{x}_{i,w}\right) \tag{5.46}$$

- calculate the color vector $\mathbf{v}_{a,b}$ each of the $\theta_v \times \theta_v$ histogram bins according to Equation 5.13.
- initialize the RBFN

  - randomly distribute the centroid vectors $\boldsymbol{\mu}_j(1)$ across the histogram
  - initialize the inverse covariance matrices $\boldsymbol{\Sigma}_j^{-1}(1)$ with

$$\boldsymbol{\Sigma}_j^{-1}(1) = \begin{pmatrix} 0.1 & 0.001 \\ 0.001 & 0.1 \end{pmatrix} \tag{5.47}$$

  - set each element of the output weight vector $\mathbf{w}_j(1)$ to $1/N$, for all weight vectors.

- do iteratively, using the iteration counter n
  - update the parameters $\boldsymbol{\mu}_j(n)$, $\boldsymbol{\Sigma}_j^{-1}(1)$ and $\mathbf{w}_j(1)$ using the RPROP method first presented in Section 4.5 and discussed in Section 5.5 for details concerning the separation algorithm of this chapter. For calculating the estimated gradients use the partial derivatives in the Equations 5.30, 5.31 and 5.32. Concerning the adaptation parameters η_+ and η_- use a value of 1.1 for η_+ of the output weights and for all other parameters 1.01. Use η_- with $1/2$ for all parameters.
  - clamp all negative weights to zero.
  - restrict the centres to lie within the coordinate range $[\theta_v..\theta_v]$ of the histogram.
  - clamp all the elements of the inverse covariance matrix $\boldsymbol{\Sigma}_j^{-1}(n)$ being bigger than $\theta_{\Sigma_{max}}$ to $\theta_{\Sigma_{max}}$. Where a good value for $\theta_{\Sigma_{max}}$ is 10000.

- initialize all sets $\mathcal{M}_j$ of indices of histogram bins assigned to cluster j with $\varnothing$

- do for all histogram bins $\mathbf{v}_{a,b}$
  - get the RBF m with the maximal activation value

$$m = \arg\max_j \sum_{c=1}^{C} \mathbf{w}_{j,c} G_{a,b}\left(\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j^{-1}\right) \tag{5.48}$$

    and set

$$\mathcal{M}_m = \mathcal{M}_m \cup (a, b) \tag{5.49}$$

- initialize all frequency bins $\mathfrak{X}_{i,j,w,f_\theta}$ of all instruments to zero.

- do for each frequency bin pair $(\mathfrak{X}_{1,w,f_\theta}, \mathfrak{X}_{2,w,f_\theta})$ of the input signal
  - calculate the sampled magnitude phase a and time shift b

$$a = \phi_\angle\left(\mathfrak{X}_{1,w,f_\theta}, \mathfrak{X}_{2,w,f_\theta}\right) \tag{5.50}$$

$$b = \phi_\delta\left(\mathfrak{X}_{1,w,f_\theta}, \mathfrak{X}_{2,w,f_\theta}\right) \tag{5.51}$$

  - update the frequency bins of the instrument assigned to the histogram bin $\mathbf{v}_{a,b}$ with the actual frequency bin pair in the input signal by

$$\mathfrak{X}_{i,j,w,f_\theta} = \begin{cases} \mathfrak{X}_{i,w,f_\theta} & (a,b) \in \mathcal{M}_j \\ 0 & (a,b) \notin \mathcal{M}_j \end{cases} \tag{5.52}$$

    with $i = 1, 2$ and $j = 1..N$.

- undo the frequency domain transform

$$\mathfrak{x}_{i,j,w} = \mathcal{F}^{-1}\left(\mathfrak{X}_{i,j,w}\right) \tag{5.53}$$

  for all channels, instruments and windows.

- multiply all samples of all instruments with Hanning-type window weights

$$\mathfrak{x}'_{i,j,w} = \mathfrak{x}_{i,j,w} \sin\left(\frac{\pi\left(t_w - 1\right)}{\theta_w}\right) \tag{5.54}$$

- merge of each instrument's windows into one final time domain instrument signal $x_{i,j,t}$ by aligning and adding them on a sample by sample basis and then multiplying the result by $2\left(1 - \theta_r\right)$.

5.8 Summary and Future Work

In this chapter we developed a blind source separation algorithm as an alternative for the two template-based methods in the previous chapters. This alternative was thought to be biologically more plausible as it now would make use of two stereo cues - IID and ITD - which are also processed by the human auditory system. Furthermore we decided for this algorithm to work on a spectral domain representation as the frequency transform would better decorrelate the signal sources.

The main idea of this algorithm is to use a magnitude-shift-frequency histogram which shows the distribution of the magnitude ratio and time shift parameters for each frequency bin representing roughly the two stereo cues IID and ITD respectively, together with the frequency of each bin visualized as a colour. We use the magnitude phase rather than the ratio directly when calculating the histogram because it distributes the ratios more evenly and limits them to a finite range.

We discussed the main problem when generating the histogram which are the ambiguous time shifts of frequencies higher than a certain threshold frequency which itself depends on the spacing between the recording microphones used to generate the stereo input signal.

Due to the ambiguity we had to set the maximal frequency shown in the histogram to the threshold frequency to get plausible results.

The main algorithm for separating the instruments from the input signal now uses the histogram to estimate the spatial position of the instruments which is translated to a specific magnitude phase and time shift. The spatial location is estimated using a clustering approach where a RBF network with a fixed number of RBFs is trained to adapt to the intensities and colours of the histogram bins where the number of RBFs is fixed to the number of clusters which have to be supplied by the user. After training the histogram is parted by assigning each histogram bin to the cluster represented by the RBF which has its highest activation for that bin.

The actual separation is now achieved by is then splitting and copying the spectrum of input signal into the spectra of the different instruments which were previously initialized with all zeroes. The splitting is done on the basis to which histogram bin and thus cluster the frequency bin pair consisting of frequency bins from both channels at the same time and spectral position, is assigned to. Finally the spectrum of each instrument is transformed back to time domain resulting in the waveform representation of each estimated instrument which is what we searched for.

We also discussed some fine tuning methods for improving the quality of the algorithm. The first topic was a deeper insight into which kinds of artefacts were generated by the separation algorithm and how they could be best suppressed and how the they could be suppressed by adjusting the window size and overlap during the windowed frequency transform and its inverse. We introduced the notion of musical noise which is used in literature for the especially disturbing distortion characterized by its pure tones which sound like being played erratically by some synthesizer and thus is sometimes not perceived as noise but rather as a kind of interference. We explained that musical noise can be best fought by choosing a high window overlap as it averages over the binary decisions during separation. The second kind of noise being added echoes, reverberation and pre-echoes which were caused by loss of weak frequencies and thus widening of transients could be reduced by using shorter window length where we also had a practical limit for the smallest length as it would reduce frequency resolution too much which would impair separation performance and could also cause musical noise.

The other topic was an attempt for disambiguation of the time shifts for higher frequencies. We proposed there a heuristic where we would try every possible time shift in the range of the maximum possible time shifts and take the shift which would be next to a cluster. We also noted there that the ambiguities rise gradually when the frequency threshold where the ambiguities begin is reached and thus there are portions in the shift interval where the calculated time shifts are still non-ambiguous although the frequency is higher than the threshold.

This is true till a higher second threshold where all shifts are ambiguous. This attempt was left to be further elaborated upon in some future work.

Another topic which should be investigated in some future work is how the histogram could be further extended to include either more spatial cues or other parameters capable of discriminating between instruments. We view this histogram as a kind of feature space rather than a visualization. So by adding some new features we would possibly destroy the possibility to visualize this feature space but we would expect to be able to obtain better results by doing this. New features would mean that the trained RBF would have more input dimensions and could possibly better approximate the clusters formed by each instrument thus leading to better separation decisions during assignment of the frequency bin pairs to their corresponding instruments.

A potential improvement which should be considered for future work is the automatic recognition of the number of clusters in the histogram. Actually this number has to be supplied by the user who in turn must somehow either “see” from the histogram how many clusters an input signal may have or count the different instruments by careful listening which may have to be repeated to in order get a good estimate. The automatisation will not be trivial as there may be clusters at different resolutions and often clusters overlap thus making the estimate less precise. Fortunately we saw that even when providing a wrong number of clusters the performance does not degrade too rapidly as to require very precise approximations of that number.

We should also mention here that it may be advantageous if some heuristics could be added to split frequency bins which are shared by more than one instrument. This can often be the case in musical pieces as we consider in this thesis and thus might raise the separation quality considerably. Furthermore also some heuristics for signal separation an frequency bin splitting based on repetitiveness of signals like in our first two approaches could be considered as a further improvement as is often done lately in blind signal separation literature.

Summing up we have come with a working blind source separation algorithm and also presented some quality improvements but there is still much work to do. We believe that this algorithm can be drastically improved by finding new and better features for the histogram of rather for the feature space and by implementing a good disambiguation algorithm.

Chapter 6

Implementation Details

6.1 Overview

In this chapter we will give a short overview of the implementation of the three separation algorithms discussed in this thesis. Reading this chapter may prove helpful when considering the speed figures in the evaluation chapter which will come next.

The code was written in C++ using Microsoft's Visual C++ (MSVC) 2003 as the programming environment. The three algorithms were implemented in three separate programs each having its own MSVC project. Each application was designed to run on the command line in order to keep the overhead down which would be incurred by a graphical user interface (GUI).

The three programs are all separate versions of the base program we called INEX which stands for INstrument EXtractor.

We will give some details on the libraries we used in Section 6.2, then we will give an overview of the code structure in Section 6.3. Some discussion on performance enhancements especially on multithreading and the more theoretic topic of the optimal FFT size in regard to speed for convolutions and correlations will be given in Section 6.4.

Finally we will summarize the content of this chapter in Section 6.5 and will also explain some improvements which could be realized in future work.

6.2 Libraries

We did not have to code the whole program from ground up as libraries with pre-coded routines for often used high-level functionality already exist thus allowing us to have more time to code the main algorithms.

The first library we used was Marsyas [36] version 0.1 which is an open source audio analysis framework. Important to us was the capability of reading and writing audio files with different extensions like reading *.wav, *.au and *.mp3 and writing *.wav and *.au. The library is rather big and complex because it can do much more than reading and writing audio files therefore we have separated the code for the formats we needed and added it to our projects. We then implemented an own wrapper class to interface with the separated library code where the wrapper also included some additional functionality we needed. On some occasions we found out that the wave file format reading and writing code was either strange or buggy so we had to debug it at some points before we could definitively use it. We should mention here that at the time of writing a new version of this library is available. As the separated code from version 0.1 had all the functionality we needed, we felt it was not necessary to upgrade especially because this would have implied that we would have to take the time to do another separation of the code from the new library.

For the widely used fast Fourier transform (FFT) we took the fftw[1] library [9] version 3.1.2 which is open source and is known for being fast due to its many optimizations in the FFT algorithm and also due to its capability of adapting the algorithms to the computer it is ran on in order to achieve maximum speed. We shall note here that the FFT is very a time consuming task because the number of operations it needs is bound by $\mathcal{O}(N \log_2 N)$. So with large windows with a length of 2^{14} samples for example it needs at least $14 \cdot 2^{14}$ operations. Now as we use the FFT extensively for all three algorithms either for frequency domain transformation, correlation or convolution, having a fast implementation can speed things up considerably.

While we were using the fftw we noticed that we could not get working real to complex transforms (i.e., the time domain to frequency domain transform) for window sizes above 2^{20} as the resulting spectra were completely off. This turned out to not be a problem since as we will see in Section 6.4 the optimal FFT size for convolution and correlation is a smaller than that number. We were running into this anomaly as we tried to use long window lengths for the blind source separation based algorithm which in this case did not create any meaningful histograms. Still this posed no problem to us as we already mention in Section 5.6 a window length of half a second is the practical maximum due to noise considerations. The half second translates to about 2^{14} samples for a sampling rate of 44.1kHz which is well under the window size where the anomaly starts.

As MSVC 2003 has problems with getopt we had to use an external library. We decided to go for GetPot[2] which implements the functionality of getopt using object oriented code. According to the website in recognition to getopt, the name GetPot was chosen as an anagram.

---

[1] `http://www.fftw.org`

[2] `http://getpot.sourceforge.net`

6.3 Code Structure

All three algorithms were implemented using the same structure so we will need only to explain it once.

There are three C++ files where each one represents one component:

- DMX.cpp - Implements the DeMiXing class which represents the separation algorithm. Here we use the fftw library for calculating the FFT and its inverse.
- SoundFile.cpp - This is the interface class to our code fragment taken from Marsyas. For the two template based algorithms this includes the up- and downsampling functionality mentioned in Section 3.8.
- INEX.cpp - This file contains the command line parsing where we use GetPot and the main program which uses the SoundFile interface to Marsyas in order to read and write sound files and calls the separation algorithm DMX with the appropiate input.

6.4 Performance Enhancements

Multithreading

Although the code was written on a dual-core Pentium we did not use threads in the demixing class nor did we use the threading functionality of the fftw library. Using all cores on a multi-core processor would in the best case divide the time needed for running the algorithm or the FFT by the number of cores which in our case would be two. Implementing multithreading capability would have needed some additional time especially for debugging and because we were short of time we simply skipped it. In this case the speed given in the evaluation results can be considered to be about double as much as the computer may have been capable of if both cores were used concurrently.

Theoretically all time consuming sub-algorithms of the three separation methods presented in this thesis can be parallelized using threads. Especially the FFT, Newton's method, RPROP, Lanczos up- and downsampling and histogram calculation can be split up in several parts with only a small computational overhead.

As a shortcut for making our code multithreading capable we have used the Intel compiler for doing some automatic parallelization but it could split up only very simple loops. It also did not improve the fftw at all when trying to recompile it, rather it worsened performance so we had to turn off the optimizations for it - this is an issue which is mentioned on the website of the fftw. So, in conclusion, parallelization has still to be taken care of when the program is designed and coded and can not be added by the compiler.

32 bit versus 64 bit

A performance aspect of our implementation which we came across while we ran our experiments was that our code was written and compiled on a 32 bit machine which induces that though having 3 GB or available physical memory we could use at most 2 GB thereof in the best case. This issue is caused by the addressing capability of a 32 bit memory pointer. The detrimental effect of not being able to use the physical memory is that we had to consider unloading temporarily some arrays from memory on to the hard disk in order to free up some process memory which not only induced some coding overhead but also slows down the algorithm.

Here we should note that we are working with stereo audio files consisting of millions of samples, each sample occupying in memory 4 bytes. Now as we need some more arrays for storing intermediate results, or for pre-computing some time-intensive operations, gigabytes of physical memory are consumed rather fast.

So in some future work it may be an interesting issue to consider whether an upgrade to a 64 bit application would be worthwile, especially as it would then require to be run on a 64 bit machine which would break compatibility to the more widespread 32 bit architecture. Therefore it must be weighted whether a 64 bit implementation would have more benefits by being much faster due to the ability to use the whole physical memory or more losses due to compatibility problems.

fftw Wisdom

The fftw adapts its algorithms to each computer it is run on by estimating the behaviour of the algorithms or by performing several FFTs with different internal configurations and choosing the fastest one. The latter method generates faster algorithms but takes a very long time to complete which is impractical to be run each time the program is started. The estimation method on the other hand is very fast but usually chooses suboptimal configurations for the algorithms thus making them slower. We observed a difference in the algorithms by up to a factor of two in speed between the two adaptation methods.

Now the fftw has the possibility to save the parameters it found which according to its website work well only for the computer they were generated on. These parameters are then called collectively wisdom and can be saved to the hard disk once they were found so the adaptation need to be done only once for every computer. Interestingly the optimal parameters may also vary for the same computer between different user sessions on the operating system due to memory and CPU load. So loading wisdom from file does not necessarily mean that the parameters will be optimal for the time the program is ran.

Unfortunately again due to time constraints we did not look further into the wisdom mechanism of the fftw and how to optimally use it. So this topic may be considered worth some further investigation in some future work.

Considerations of FFT Size

While the size of the FFT in the blind source separation algorithm matters due to the desired high frequency resolution and the noise effects, we can choose it freely for the other two algorithms. More precisely when using the FFT for correlation and convolution we can aim to choose a FFT size which will need the least time to calculate.

From now on we will discuss the optimal size of the FFT for correlation which is the same as for convolution because correlation and convolution differ only by the time reversal of one of the input signals.

So in order to do a non-cyclic correlation we need to choose a suitable FFT size $\theta_{\mathcal{F}}$. An obvious but naïve solution would be to choose the size as the size of the first input signal T_A plus the size of the second input signal T_B. We then pad both signals with zeroes until they reach the length $T_A + T_B$. We note here that we need to pad the bigger signal which we choose to be A, with the length of B in order to ensure that the resulting correlation will be non-cyclic. Now although the naïve solution will work, it is not optimal in terms of speed.

We shall note here that fast algorithms for the FFT can be used if $\theta_{\mathcal{F}}$ can be factored into small prime numbers. The fftw will also employ algorithms with a complexity of $\mathcal{O}(N \log N)$ if this is not the case but they are not fast in absolute terms. The fastest speeds can be achieved if the $\theta_{\mathcal{F}}$ is a power of two and therefore we should aim for such a $\theta_{\mathcal{F}}$ in our search for the optimal FFT size.

Now the logical improvement for choosing $\theta_{\mathcal{F}}$ is to have it be the next power of two which is bigger than $T_A + T_B$ and pad A and B with zeroes to fill up the extra space. Assuming $T_A = 2^{23} - 2^{14}$ samples which is equivalent to about $3 : 10$ minutes at a 44.1kHz sample rate and $T_B = 2^{14}$ samples which equals 0.37 seconds, we can calculate the approximate number of abstract FFT operations as

$$3\,(T_A + T_B)\log_2(T_A + T_B) = 3 \cdot 2^{23}\log_2 2^{23} = 3 * 23 * 2^{23} = 578\,813\,952 \tag{6.1}$$

where we had to multiply the outcome with three as we have to transform signal A and signal B and have to transform the result back in the time domain. We do not count the number of complex multiplications in the spectrum which represent the correlation operation itself, as they do not correspond to abstract FFT operations but we have to keep in mind that they exist and will change with further improvements discussed in the next paragraphs. Note that in our example the result has the same number of operations as the naïve method but in general the naïve method will need fewer abstract operations. The difference will usually be that if $\theta_{\mathcal{F}}$ is not a power of two then one abstract operation will in average need more concrete operations and therefore the naïve solution will not necessarily be faster than the this power of two solution. There is also another aspect of this: zeroes consume much less computational time (i.e., CPU cycles) than non-zero entries. So padding a signal with zeroes will not slow down the calculation speed too much.

We observe here that the correlation can now be split up in smaller segments. This means that we can do the FFT for smaller sizes thus reducing the number of abstract operations and speeding up the calculations. A second benefit of smaller segments is that we need to calculate the FFT of B only once and can reuse it for each segment. The minimum segment size is $2T_B$ if we are to use powers of two. In order to calculate the segments correctly we have to introduce an overlap before the end of the segment instead of the zero padding. This will increase the total size of the samples needed to be transformed. Because the overlapping is exactly T_B the smallest possible segmentation will increase the number of samples which need to be processed by about two.

Now having the information about the smallest segment size and the fixed required overlap we see that we have to minimize the function $\mathscr{O}\left(W; T_A, T_B\right)$ of abstract operations for segment size W

$$\mathscr{O}\left(W; T_A, T_B\right) = 2N_W W \log_2 W + W \log_2 W \tag{6.2}$$

or equivalently

$$\mathscr{O}\left(W; T_A, T_B\right) = \left(2N_W + 1\right) W \log_2 W \tag{6.3}$$

by choosing the right segment size W, where N_W is the number of segments. In the following we will minimize this function for our example in order to show how many abstract FFT operations can be saved but usually the segment size must be determined by trial and error on each computer in part as there are also other factors as for example the processor cache, the number of complex multiplications in the frequency domain in order to produce the correlation, and so on.

We note that the number of segments N_W is a function of W, T_A and T_B, defined as

$$N_W = \left\lceil \frac{T_A + T_B}{\left(W - T_B\right)} \right\rceil \tag{6.4}$$

where W is rounded up to the next power of two through the ceiling operator $\lceil\cdot\rceil$. So

$\mathscr{O}(W; T_A, T_B)$ then becomes

$$\mathscr{O}(W; T_A, T_B) = \left(2\left\lceil\frac{T_A + T_B}{(W - T_B)}\right\rceil + 1\right) W \log_2 W \tag{6.5}$$

Using the values from our example this function has one minimum at $N_W = 74$ with $W = 2^{17}$ resulting in $332\,005\,376$ abstract operations. This is only 57.4% of the operations of the non-segmented method. Note that this result is a worst case reduction because the example was made to be comparable with the naïve method. If we were to choose $T_A = 2^{23} - 2^{14} + 1$ instead of $T_A = 2^{23} - 2^{14}$ we would have had 1 207 959 552 compared to 332 005 376 abstract operations for the segmented method which would be a reduction to 27.5 percent. This would now be a best case reduction compared to the non-segmented method.

As we see, doing correlation and convolution segment-wise using the FFT can reduce the number of abstract operations needed from one half to one fourth of a non-segmented method. Considering that for smaller FFT sizes the processor cache can be better reused then the speed gain for segmented calculation may be even higher.

6.5 Summary and Future Work

In this chapter we explained the libraries we used during the development of the code. More precisely we talked about Marsyas where we separated the code we needed and used it through a wrapper class. Then for the fast Fourier transform we used the fftw library which is known to be a fast implementation of the transform. And lastly, due to problems with the getopt function in MSVC 2003 we mentioned the use of GetPot as a worthy replacement.

After a brief overview of the code structure of the three implementations, each containing three files we arrived to the topic of performance enhancements. The benefits of enhancing the implementation with multithreading capability which is expected to cut processing time in half for a dual core processor, were discussed. Though not being implemented at the time of writing, this enhancement should be considered for future work especially as multi-core processors are becoming mainstream.

During evaluation we came across the problem of our implementations being able to address only 2 GB of physical memory due to their 32 bit addressing. As our algorithms work with big amounts of data originating from audio files and also need to store the resulting variables, they fill up the whole addressable memory space rather fast. Upgrading to 64 bit which would enlarge the addressable amount of memory but would at the same time break compatibility to 32 bit machines which are widespread by the moment is left as an issue to be carefully thought upon in some future work.

Another possible performance improvment using fftw wisdom was discussed where wisdom is the capability of the library to save internal configuration parameters which result in fast transforms. As finding such parameters is a time consuming task saving it for future use is an option to be considered. This capability is not used in the current implementation and therefore was left for some future work.

The more theoretical aspect of optimal FFT size in regard to execution speed was elaborated upon in the next part. Following the idea that a convolution or correlation using the FFT can be done segment-wise, we can opt to choose a favorable segment size in regard to speed. Here we saw that segments with lengths being powers of two were suitable for this purpose due to the availability of fast implementations for these special sizes. Furthermore keeping the segments short compared to the length of the input signal not only reduces the number of abstract operations needed for the FFT but also increases cache reusability.

Chapter 7

Benchmarking and Evaluation

7.1 Overview

Now we have come to the point where we have to show that the algorithms we have come up with, are indeed capable of separating instruments as they should. As we cannot formally prove their capability we will empirically show their efficiency by running them on well known music collections and on our own corpus of publicly available music files and measuring their outcome either subjectively by a rating system or objectively by an error measure in order to generate scores which should be comparable between our algorithms and possibly between further publications in this field.

We will continue this chapter with a discussion on the corpora used and the songs we selected in Section 7.2. Here we will also describe how we generated our own corpus called the IS corpus and what types of recordings it contains.

In Section 7.3 we will explain our testing methods, the scoring system, the error measure as well as the algorithms used for comparing the output of our implementation with the given reference tracks. Furthermore we will also introduce a base-line in order to include a simpler algorithm in our comparison.

The results will be published in Section 7.4 continuing with a discussion in Section 7.5 where we will interpret the results more deeply.

Finally we will give a summary and draw some conclusions in Section 7.6.

7.2 Corpora

Existing Corpora

We used songs from three existing corpora and our own corpus for evaluation. The existing corpora are in detail the

- *BASS-dB* [37], the Blind Audio Source Separation dataBase. Although this database is the most interesting one, it is no longer actively maintained at its website cited in the reference. Still, due to the Creative Commons (CC) license, some of the songs can be found and downloaded legally in the Web either at the Internet Archive[1] or on their respective websites (see Appendix A for further details). This is the reason we decided to use what was left of the database. The interesting part here is that due to the BASS-dB being especially made for blind source separation, the songs were selected to also include their reference tracks before the final mixdown. Thus using these songs allows us to compare the output of our algorithms to the reference tracks using an objective error measure.
- *ISMIRgenre* collection, containing full songs from different genres. We included two songs from each genre in our benchmarks making a total of twelve titles.
- *RWC* [11], containing full songs, copyright cleared for research usage. Ten titles were selected for our benchmarks from this database.

Summing up we have 25 songs from existing corpora, counting the three songs we could take from the BASS-dB. As a sidenote we should mention that we found more than three songs belonging to the BASS-dB but the other songs did not have their tracks aligned and therefore needed some complicated mixing and stereo panning in order to obtain the outcome intended by the author, which determined us to exclude them from testing.

Instrument Separation Corpus

Now, while these corpora may be representative for music available today, we felt that we also needed to include some songs where our algorithms are expected to achieve good results which lead us to build an additional corpus of our own, the IS corpus.

First we decided to use some module files, also known for their .MOD ending. Unlike WAVE files they do not contain the waveform of the song but notes similar to the MIDI format. Additionally they also contain the sound of the corresponding instruments which makes

---

[1] `http://www.archive.org`

these files have the same rendered output on different computers, unlike MIDI where the sounds of the instruments are stored externally and thus may vary depending on which hardware or software they are played on.

An advantage of module files is that their content is organized in channels which can be rendered separately using editing software like the free ModPlug Tracker[2]. This means that potentially every module file can be decomposed into its component tracks and be rendered to common WAVE or AU files thus enabling us to build reference tracks for our benchmarks. As these tracks would not vary depending on the computer played upon, the decomposition is repeatable and thus usable for our purposes.

Now as good as it may sound, the tracks in the module files do rather seldom represent one instrument or a group of instruments and therefore not every module file is suitable for benchmarking. The main problem we encoutered was that instruments often change the tracks they are played in. On the other side if we separe an instrument with our algorithms we usually generate a track with only that instrument playing or as for the direct template matching algorithm, we generate a track for every separated tone. So if we select one track as the reference we will miss parts of the instrument which changes tracks, a situation which would inevitably lead to a high error value in both tracks although the corresponding instruments may have been separated perfectly by the algorithm. A possible solution would be to add the affected tracks together but we noticed that the instruments change tracks in such a way that at the end all tracks would have to be added together. So our quest in building a corpus using modules was to find files where the instruments would either not change tracks at all or change only tracks in a way that if these tracks were added together there would still remain a number of indepentent tracks at the end.

We found about four freely available modules fullfilling the criteria for having the possibility to create some independent tracks out of them. We examined the independence by listening to the tracks, no formal parser was built, so there may be some sections which slipped our attention. Not using a formal parser which guarantees that the tracks are independent, assuming the saved sounds are not from the same instrument - and even in that case a heuristic could be used to check for that condition - means that the error is limited by the missing parts of the instruments which cause deviations from the reference even though the instrument may have been separated correctly by the algorithm.

Following the module files we searched for more songs in the usual wave format with their individual tracks available so that we have more than three songs we can objectively evaluate our algorithms on. We looked out for songs being published under the Creative Commons license so that others may be able to obtain them easily by being able to download them freely. This resulted in four additional songs, all of them obtained through the Internet Archive.

---

[2] `http://www.modplug.com`

We then also looked for music recorded in binaural mode for benchmarking the histogram based blind source separation approach. Although binaural and stereo may be used synonymously for describing a two channel audio file there is usually a difference between these recording modes. Recordings which are labelled as ″stereo´´ are usually made to be mono downmix compatible. In that way, downmixing both channels to one single channel will not change the tone colour of the audio signal due to phase cancellation phenomena which is important for music in order to be receivable even by analog shortwave receivers, if broadcasted. Additionally some analog stereo to multiple channel decoding techniques also make use of the mono-compatibility in order to generate an additional center channel out of the both stereophonic ones.

Now mono-compatible downmixes are not very usefull for the HSBSS as the only stereo information is given by the magnitude differences in the signal which translate to only one dimension in the magnitude-shift-frequency histogram thus limiting its separation capabilities considerably. Now usually most genres of modern music are recorded in a mono-compatible stereo except for classical music where precise localization of the instruments is desirable, which can only be conveyed by also preserving the phase information.

So we had to choose to either search for more classical music or to search for binaural music. The term ″binaural´´ is usually used to indicate that the recording was made to sound in the way the listener must have perceived if he would have been there at the recording site. This often implies using microphones modelling the human ear and head which may look strange if used in operas for example and does not allow using separate microphones for singers and instruments as downmixing to two channels while still pertaining the binaural cues would not be easy. Unfortunatelly due to these facts also because they have to be listened to while wearing headphones in order to reveal the full panorama where this circumstance is seen as a nuissance by most listeners, such recordings are rather rare.

Still we were able to find some freely downloadable binaural demo recordings (for the links see AppendixA). As recording quality is a major problem with these recordings we could pick only four out of them which sounded good.

Summed up we now have 12 musical pieces for the IS corpus: 4 binaural recordings which make a subset of the corpus called IS-B, 4 module files in the subset IS-M and 4 waveform files in the subset IS-R. There could have been more but we were also restricted by the running time of the algorithm implementations and by the amounts of data these algorithms generated. We worked on uncompressed music where possible and also generated uncompressed files before measuring errors for the objective tests. As a hard disk rapidly fills up during such a testing procedure we eventually had to compress the resulting tracks using lossy compression, but we did that only after generating the error values. This way we did our tests using the uncompressed files and then archived the resulting tracks using lossy compression.

We should mention here that the files forming the IS corpus itself are not archived using lossy compression except for the binaural recordings which were lossy compressed from the beginning.

7.3 Methods and Algorithms

Testing Conditions

The benchmarks were done on a Core 2 Duo 2.13 GHz machine with 3 GB of DDR2-667MHz RAM - where only 1 GB could be used by the algorithm as mentioned in Chapter 6. Only one core of the processor was used as the implementations did not support multi-threading.

For the subjective listening tests a Creative X-Fi XtremeMusic sound card was used. Playback was done on headphones. SoundForge XP 4.0 was used as the player so that the test subjects could also see the envelope of the signal while playing.

Baseline

In order to have some simpler algorithm which should provide the lower bound on the performance measure we decided to use a baseline which makes use of the structure of the often used multichannel to stereo encodings. The algorithm works as follows for each sample pair $(x_{0,j}, x_{1,j})^T$:

$$\begin{pmatrix} x'_{0,j} \\ x'_{1,j} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x_{0,j} \\ x_{1,j} \end{pmatrix} \tag{7.1}$$

Thus $x'_{0,j}$ represents the monaural or sum channel and $x'_{1,j}$ the difference channel. We note that applying the transformation matrix once again without the $1/2$ scaling coefficient results in the original signal.

As the difference channel usually has much less energy than the sum channel we decided to scale the difference channel by 2 for the subjective tests in order to compensate for the reduced loudness which may have been perceived as a signal quality problem by the test listeners. For the objective tests we used the original formula.

This baseline now is expected to work well for multichannel coded stereo recordings as it mimics a very simple decoder. For binaural recordings there should be no perceivable separation as the instruments spread through the magnitude-shift space instead of concentrating around the middle point which stands for zero time shift and equal magnitude.

Subjective Evaluation

For this benchmarking technique we need to define the criteria according to which the resulted tracks should be evaluated. During preliminary listening we realized that it was not possible to use only one quality measure. We rather needed to describe two qualities of the separated signal:

1. the *grade of separation*, "S´´. Usually the algorithms separate whole groups of instruments. So this measure now should tell us about how much the instruments not belonging to the separated group are suppressed and at the same time how small this group is. The range of this measure is set to be between 0 and 5 with zero meaning that the group contains all instruments or that no instruments were suppressed which basically means there was no separation and 5 means that all but one instrument were suppressed and that the suppressed instruments are cannot be recognized as such but rather as some noise. The values 2 and 3 mean that the instrument group contains about half of the playing instruments and the others are suppressed percetibly where 2 stands for weaker suppression and 3 for a better more perceptible suppression. Now 1 means that the group contains many instruments compared to the total number of instruments and that the other instruments are only weakly suppressed. Finally the score of 4 means that the group of instruments is small, tending towards one instrument and that the other instruments are very well suppressed.

   We did not split this criterion further into group size and suppression quality because they correlate. Tracks with weakly suppressed instruments tend to be perceived as having selected a wider group of instruments than tracks with strongly suppressed ones. As counting the instruments in a group only by listening is usually a difficult task as the songs in the corpora we used have plenty of them playing at the same time, we regarded it as a better choice to tie the group size with the supression power.

2. its *perceived quality*, "Q´´. Now even if one instrument was separated perfectly in a resultant track with the other instruments being imperceptible the signal may still have a bad quality as the algorithm may also have suppressed some parts in the frequency range from the selected instrument. So the perceived quality of the track may be low although the separation may be good. This subjective score can also take up values from 0 to 5 with 0 meaning the instruments in the resultant track are cannot be identified and 5 meaning that the instrument is separated with a very good quality with maybe some barely perceptible residual noise. 1 should mean that the selected instrument or instrument group is barely identifiable, 2 to 3 meaning that the instrument group has annoying degradations with 3 being less annoying. Finally 4 is thought to stand for a

track whose selected group has some perceptible degradation but the sound quality is pleasing to the extent allowed by the content of the song.

Now there may be some extreme cases like 0/0 (grade of separation/perceived quality) which should be interpreted that the track contains no data and 0/5 that there nothing was separed. The combination 5/0 should not exist as it is not possible to say how well an instrument was separated if the signal is degraded so much that the selected instrument or group cannot be identified anymore. Now, finally 5/5 should stand for perfect separation and this is what we aim for with our algorithms.

The scores discussed until now are given to each track separately, where we form a mean for each score type over all testing subjects. Coming to the two average scores for each song we came to the problem whether we should calculate the mean over all separated tracks with valid data e.g., the tracks not having a 0/0 score or rather over all tracks generated. Here we decided to build the mean over all generated tracks as we think it should not matter whether many tracks were separated without good separation power or quality or only two tracks with good separation power and quality even if there should be more.

We should note here that we did not use the direct template matching algorithm for subjective evaluation as due to its lack of the tone clustering stage which should gather together all tones corresponding to one instrument. We got about 40 resulting Tracks per song. These tracks would have to be clustered and mixed by hand into some fewer tracks before evaluation and it was too much effort for the restricted time frame and ressources we had at our disposal.

Objective evaluation

The objective evaluation is somehow easier to explain and perform as in this case we have the reference tracks available and thus only need to perform a measurement of the error between reference and result.

The error measure we chose is the signal to noise ratio (SNR) which is expressed in dB according to Formula 2.2. Here we will use a version adapted to stereo input

$$SNR = 10\ \log_{10} \frac{\sum_{i=1,2} \|\mathbf{x}_i\|^2}{\sum_{i=1,2} \|\mathbf{x}_i - \hat{\mathbf{x}}_\mathbf{i}\|^2} \tag{7.2}$$

where $\hat{\mathbf{x}}_\mathbf{i}$ is the i^{th} channel of the input track. The SNR is zero if the noise and signal energy are equal and ∞ if the estimation is exact and thus there is no noise term.

Before being able to measure the difference between estimate and reference we first have to match the output of the algorithm to the reference as the output is not generated in the order the references are listed. Furthermore the references may contain groups of instruments which the algorithm may have further separated.

So in order to match the outputs to their references we coded an algorithm searching for the best match between every output and the reference where each output may be matched only to one reference. If more outputs match the same reference best then they are added together and the SNR is calculated thereafter. This algorithm is applied to the HSBSS, the baseline and to the direct template matching algorithm. Especially for the latter we have the advantage that the 40 instrument tone tracks are grouped together and then compared to the reference after being summed correspondigly. This way we are given a best case approximation of the missing clustering stage. Unfortunately as this approximation uses information contained in the references file in order to cluster the instruments we will always obtain better results than with a real clustering stage and therefore our results for the direct template matching algorithm can be regarded only as a guide and not as a reference when compared to other algorithms. Using a real clustering stage with all other parts of the algorithm being the same, can only produce lower or at best the same results.

While calculating the mean value for the SNR we saw two possibilities on how to do this

1. calculate the arithmetic mean of SNR values.
2. concatenate the references and separately concatenate their matching outputs to form one signal for the references and one for the matching outputs. Calculate the SNR on these two signals. The result can be considered the mean SNR over all references. Still, due to the logarithm and the fraction this is not the same as the arithmetic mean of the SNR values.

We finally opted for 1 as it produces more intuitive results. 2 has the problem that if there is one well separated track and two badly separated ones then this well separated track will not count much. It is hard to give an example here because 2 is also dependent on the energy and the length of the signals being concatenated. So choosing 1 is expected to give more intuitive results.

7.4 Results

For testing, we used the direct template matching approach, the HSBSS algorithm and the baseline. The iterative self-organized template matching implementation did not to converge in reasonable time and therefore was not included in the following tables. We decided that stopping the algorithm before it reached convergence would produce results which would not be representative of its capabilities. As the results may be interesting, an evaluation for this algorithm may be done in some future work.

We start with the objective evaluation results given in Table 7.1. The evaluated songs come from the BASS-dB and from the parts of our corpus where we have the reference tracks or could generate them, as it was possible with the module files.

Looking at the number of reference tracks covered by each algorithm we see that the DTM approach clearly separated the most tracks for the module file part of the IS corpus. As this algorithm can be thought as the inversion of a module file playback, it is very natural for it to be able to perform better than the HSBSS. Interestingly though this is not the case for the separation performance itself where the HSBSS is able to reach the highest signal to noise ratios for both, the best separated track out from a song and for the mean separation quality. On the top of that it is also faster than the DTM by a factor of two.

For the BASS-dB examples we see that the HSBSS covers most of the reference tracks with best SN ratios as well. The exception is the reference part of the IS corpus where the DTM has the better track coverage and SN ratios. As the songs do no differ that much from the BASS-dB, this result is rather interesting.

Concerning the baseline, it has the lowest SNR as expected and also the lowest overall reference track coverage. The only advantage the baseline has and which is not shown in Table 7.1 is its speed. It needs only about one second to generate the two resulting tracks which is two orders of magnitude faster than any of the two other algorithms.

A summary of the means over all three groups of songs can be seen in Table 7.2. The DTM seems to have a little better mean performance but we must keep in mind that it has a best case approximation of its missing clustering stage. Considering that the HSBSS needs less computational time for achieving a separation nearly as good as the actual DTM we can say that the HSBSS has a better performance than the DTM.

After having presented the objective evaluation we now shall come to the subjective one which is much more interesting as it shows the performance perceived by humans.

Looking at Table 7.3 which shows the same songs as Table 7.1 for the subjective evaluation we can truly see the performance difference between the HSBSS and the baseline which was not so obvious during objective evaluation. It is best visible in the mean separation score ″S´´ which never goes beyound 1.

Here we should also note that we had to make the algorithms comparable. As the baseline always creates two tracks instead of the desired number which was four or higher depending on whether we had the reference tracks or not, we considered all the tracks up to the desired number to have a 0 separation and 0 quality score as they were never created and

	Tracks Reference	Tracks DTM	Tracks HSBSS	Tracks BL	Max. SNR DTM	Max. SNR HSBSS	Max. SNR BL	Mean SNR DTM	Mean SNR HSBSS	Mean SNR BL	Time DTM	Time HSBSS
BASS T1	4	2	**3**	2	**3.00dB**	2.95dB	1.38dB	0.96dB	**1.13dB**	0.60dB	612s	**180s**
BASS T2	5	**2**	**2**	**2**	1.29dB	**8.33dB**	1.17dB	0.28dB	**3.22dB**	0.34dB	580s	**257s**
BASS T3	6	3	**4**	1	4.19dB	**4.72dB**	0.80dB	1.07dB	**1.10dB**	0.13dB	253s	**159s**
Mean	5.0	2.3	**3.0**	1.7	2.83dB	**5.33dB**	1.12dB	0.77dB	**1.82dB**	0.36dB	482s	**199s**
IS-R T1	7	**3**	**3**	1	**3.62dB**	1.39dB	0.18dB	**1.00dB**	0.53dB	0.03dB	474s	**359s**
IS-R T2	5	**3**	2	1	**4.45dB**	0.00dB	0.00dB	**3.10dB**	-0.58dB	-0.30dB	878s	**152s**
IS-R T3	10	**4**	3	1	**6.22dB**	5.09dB	3.00dB	**0.87dB**	0.51dB	0.30dB	400s	**568s**
IS-R T4	9	**3**	**3**	2	**5.80dB**	0.04dB	0.00dB	**0.53dB**	-0.36dB	-0.57dB	786s	**808s**
Mean	7.8	**3.3**	2.8	1.3	**5.02dB**	1.63dB	0.80dB	**1.38dB**	0.03dB	-0.14dB	635s	**472s**
IS-M T1	4	**3**	2	2	0.92dB	**4.03dB**	0.67dB	0.24dB	**1.31dB**	-0.02dB	181s	**73s**
IS-M T2	6	**4**	3	2	4.70dB	**9.92dB**	3.16dB	1.09dB	**1.43dB**	0.49dB	582s	**295s**
IS-M T3	5	**5**	4	1	1.89dB	**2.90dB**	0.00dB	0.31dB	**0.32dB**	-0.75dB	218s	**108s**
IS-M T4	7	**4**	2	2	**5.78dB**	5.10dB	4.43dB	**1.08dB**	0.83dB	0.72dB	386s	**256s**
Mean	5.5	**4.0**	2.8	1.8	3.32dB	**5.49dB**	2.07dB	0.64dB	**0.97dB**	0.44dB	342s	**183s**

Table 7.1 BASS-dB and IS corpus. Results of the objective evaluation with reference tracks. For the title names and their authors please see Appendix A.

	Tracks Reference	Tracks DTM	Tracks HSBSS	Tracks BL	Max. SNR DTM	Max. SNR HSBSS	Max. SNR BL	Mean SNR DTM	Mean SNR HSBSS	Mean SNR BL	Time DTM	Time HSBSS
BASS	5.0	2.3	**3.0**	1.7	2.83dB	**5.33dB**	1.12dB	0.77dB	**1.82dB**	0.36dB	482s	**199s**
IS-R	7.8	**3.3**	2.8	1.3	**5.02dB**	1.63dB	0.80dB	**1.38dB**	0.03dB	-0.14dB	635s	**472s**
IS-M	5.5	**4.0**	2.8	1.8	3.32dB	**5.49dB**	2.07dB	0.64dB	**0.97dB**	0.44dB	342s	**183s**
W-Mean$^{1)}$	6.6	**2.9**	**2.9**	1.5	**4.08dB**	3.22dB	0.94dB	**1.12dB**	0.80dB	0.07dB	569s	**355s**
W-Mean	6.2	**3.3**	2.9	1.6	3.80dB	**4.04dB**	1.35dB	**0.94dB**	0.86dB	0.21dB	487s	**292s**

Table 7.2 Summary of the objective evaluation results. The weighted means are calculated using a weight proportional to the number of titles in each corpus. $^{1)}$ Without IS-Module.

thus could be assmuned to be zero. Therefore only if the baseline would be able to split up the two tracks perfectly that is with half of the instruments in one and the other half in the other part and with perfect quality, then it would be on par with the HSBSS. It now can be argued that this comparison would be now unfair for the baseline as two tracks with perfect separation and quality would be better than four with average separation and quality but in practice the sum channel of the baseline did rarely separe anything. Here we should recall the mono-compatibility, people should still be able to listen to the song on a simple monaural player which presumes that nothing in the song is lost or hidden and thus ″separated´´ for this channel. So the case where both channels of the baseline would show best separation performance and quality cannot not occur in practice so the addition of zeroed tracks to the baseline can be regarded as fair.

Back to the results we may notice in Table 7.3 as well as in Table 7.4 that the standard deviations are high at times. This is caused by the fact that we had only few test subjects and thus the means do not converge as the subjects are biased on how to interpret separation and quality despite of our efforts to clearly define it. Still even with this high standard deviation the results are interpretable.

What we may also notice is that for the baseline quality is less of a problem that separation performance. As the algorithm is very simple and there is no dependence between the samples in time no artefacts can be created. Only timbre and tonality of the song as a whole can be unfavorably changed but compared to the HSBSS it cannot create musical noise for example.

Looking for the maximum separation we will also see that the baseline is not that bad when compared to the mean separation in Table 7.5 which summarizes the result over all corpora and collections.

If we look at the variance of the weighted mean in the subjective evaluation summary we will also observe that it is higher for the baseline than the HSBSS. This could be interepreted as if the output of the baseline is in general more controversial. Especially if it is good to have a big group of instruments with few well suppressed instruments and good remaining signal quality or a small group of instruments, ideally only one instrument with a less than perfect quality.

Finally we can see in Table 7.5 that the mean overall separation performance is scored a little weaker than average while overall quality is scored average for the HSBSS. So if improvements are to be thought of in some future work it would be probably best if they would target at the separation performance.

	Tracks Reference	Tracks HSBSS	Max. S HSBSS	Max. S BL	Mean S HSBSS	Mean S BL	Mean Q HSBSS	Mean Q BL	Time HSBSS
BASS T1	4	3	1.5	**2.0**	**1.1±1.6**	0.5±0.7	1.6±0.5	**2.5±0.0**	180s
BASS T2	5	2	**4.0**	0.0	**2.7±1.6**	0.0±0.0	**2.6±0.0**	1.9±0.1	257s
BASS T3	6	4	**2.5**	1.5	**2.0±1.2**	0.3±0.4	**2.9±0.4**	1.6±0.1	159s
Mean	5.0	3.0	**2.7**	1.2	**1.9±0.4**	0.3±0.4	**2.4±0.3**	2.0±0.1	199s
IS-R T1	7	3	**5.0**	4.0	**1.7±0.2**	0.6±0.2	**1.8±0.1**	1.3±0.2	359s
IS-R T2	5	2	0.0	**2.5**	0.0±0.0	**0.8±1.2**	1.7±0.0	**2.7±0.9**	152s
IS-R T3	10	3	**4.0**	3.0	**1.0±0.3**	0.3±0.3	**1.2±0.0**	1.0±0.1	568s
IS-R T4	9	3	**3.5**	3.0	**1.6±0.4**	0.3±0.3	**1.3±0.0**	0.9±0.2	808s
Mean	7.8	2.8	**3.1**	**3.1**	**1.1±0.0**	0.5±0.5	**1.5±0.0**	**1.5±0.4**	472s
IS-M T1	4	2	**5.0**	1.0	**3.5±0.0**	0.3±0.4	**3.0±0.7**	2.1±0.5	73s
IS-M T2	6	3	**4.0**	3.5	**1.6±0.1**	0.7±0.0	**1.3±0.2**	1.2±0.7	295s
IS-M T3	5	4	**3.0**	0.0	**2.1±0.7**	0.0±0.0	**2.7±0.1**	1.8±0.3	108s
IS-M T4	7	2	**4.5**	3.0	**1.6±0.5**	0.5±0.1	**1.5±0.5**	1.1±0.4	256s
Mean	5.5	2.8	**4.1**	1.9	**2.2±0.3**	0.4±0.1	**2.1±0.0**	1.6±0.5	183s

Table 7.3 BASS-dB and IS corpus. Subjective evaluation with the same titles as for the objective evaluation. "S´´ stands for the grade of separation and "Q´´ for the remaining signal quality. For the title names and their authors please see Appendix A.

7.5 Discussion

The results show that the direct template matching algorithm and the histogram based blind separation method have about equal separation strength. Now there is also another thing of concern: the performance on the objective evaluation is very unevenly distributed among the BASS-dB, IS-R and IS-M. The results vary widely. We think that this problem comes from using to few songs per corpus. Unfortunately for the BASS-dB we could not do much more and we also lacked the time to extend our IS corpus to contain more songs. So as for future work it would be very important to create a corpus with a lot more titles so that we can rule out coincidence while interpreting the results. So for example we cannot explain why the HSBSS has such a bad performance on the IS-R corpus compared to the BASS-dB by other means than with coincidence.

So we need a better corpus. The main requierements for it would be to contain as many songs

	Tracks Reference	Tracks HSBSS	Max. S HSBSS	Max. S BL	Mean S HSBSS	Mean S BL	Mean Q HSBSS	Mean Q BL	Time HSBSS
IS-B T1	4	2.5	**2.0**	1.5	**1.0±1.1**	0.5±0.0	1.6±0.2	**2.0±0.7**	161s
IS-B T2	4	3.5	**3.5**	1.0	**2.1±0.5**	0.3±0.0	**2.9±0.2**	2.3±0.4	171s
IS-B T3	4	3.5	**4.5**	0.0	**2.0±1.1**	0.0±0.0	**3.3±0.4**	2.1±0.5	168s
IS-B T4	4	3.5	**4.0**	1.0	**2.4±0.2**	0.3±0.4	**2.6±0.5**	2.4±0.2	46s
Mean	4.0	3.3	**3.5**	0.9	**1.9±0.6**	0.3±0.1	**2.6±0.3**	2.2±0.4	137s
ISMIR-G T1	4	3.0	**5.0**	0.0	**2.0±0.7**	0.0±0.0	**2.5±0.4**	**2.5±0.0**	150s
ISMIR-G T2	4	3.5	**3.0**	0.0	**2.8±1.1**	0.0±0.0	2.4±0.2	**2.5±0.0**	175s
ISMIR-G T3	4	3.0	**4.0**	2.0	**1.8±0.4**	0.5±0.4	1.9±0.2	**2.3±0.4**	497s
ISMIR-G T4	4	3.0	**4.0**	2.5	**2.4±0.2**	0.9±0.2	**2.4±0.2**	1.4±0.5	472s
ISMIR-G T5	4	2.0	**2.5**	1.0	**1.5±2.1**	0.4±0.5	**1.6±0.5**	1.3±0.0	179s
ISMIR-G T6	4	4.0	**3.5**	**3.5**	**3.0±0.4**	1.1±0.2	**2.6±0.2**	2.1±0.5	248s
ISMIR-G T7	4	3.5	**3.5**	3.0	**2.4±0.2**	0.9±0.5	**3.0±0.4**	2.0±0.7	224s
ISMIR-G T8	4	3.5	**3.0**	**3.0**	**2.3±1.4**	1.0±0.0	**2.6±0.9**	1.4±0.2	779s
ISMIR-G T9	4	3.0	**4.0**	3.5	**2.4±0.5**	0.9±0.5	**2.6±0.9**	2.3±0.4	399s
ISMIR-G T10	4	2.5	**4.5**	3.5	**2.5±0.0**	0.9±0.5	**2.3±0.4**	2.1±0.5	244s
ISMIR-G T11	4	3.5	**3.5**	2.0	**2.9±0.2**	0.6±0.2	**3.0±0.4**	2.0±0.4	262s
ISMIR-G T12	4	3.5	**3.5**	0.0	**2.4±0.9**	0.0±0.0	**2.1±0.9**	2.0±0.7	328s
Mean	4.0	3.2	**3.7**	2.0	**2.3±0.6**	0.6±0.1	**2.4±0.2**	2.0±0.4	330s
RWC T1	4	3.0	**2.5**	**2.5**	**1.6±0.5**	0.6±0.9	**2.8±0.4**	2.3±0.4	301s
RWC T2	4	3.0	**4.0**	2.5	**2.3±1.1**	0.6±0.9	**2.8±0.4**	2.5±0.0	302s
RWC T3	4	3.0	**3.5**	2.0	**2.3±1.1**	0.5±0.7	**3.0±0.7**	2.4±0.2	246s
RWC T4	4	3.5	**4.5**	3.0	**2.8±0.4**	0.8±0.7	**3.3±0.0**	2.4±0.2	336s
RWC T5	4	3.0	**4.0**	3.5	**2.3±1.1**	0.9±0.5	**3.3±0.7**	2.4±0.2	268s
RWC T6	4	3.0	**3.5**	2.5	**1.5±0.0**	0.6±0.5	**2.5±0.4**	2.3±0.4	401s
RWC T7	4	3.5	**3.5**	2.5	**1.8±1.1**	0.6±0.5	**2.6±0.2**	2.3±0.4	362s
RWC T8	4	2.5	**3.0**	**3.0**	**2.0±0.4**	0.8±0.7	**2.3±0.4**	**2.3±0.4**	264s
RWC T9	4	2.0	**3.5**	2.0	**1.6±1.2**	0.5±0.7	**2.3±0.4**	**2.3±0.4**	413s
RWC T10	4	4.0	**3.5**	1.5	**2.8±0.7**	0.4±0.5	**3.1±0.2**	2.5±0.0	231s
Mean	4.0	3.1	**3.6**	2.5	**2.1±0.5**	0.6±0.7	**2.8±0.2**	2.3±0.2	312s

Table 7.4 IS and RWC corpus, ISMIRgenre collection. Subjective evaluation of the three corpora. ”S´´ stands for the grade of separation and ”Q´´ for the remaining signal quality. For the title names and their authors please see Appendix A.

	Tracks Reference	Tracks HSBSS	Max. S HSBSS	Max. S BL	Mean S HSBSS	Mean S BL	Mean Q HSBSS	Mean Q BL	Time HSBSS
BASS	5.0	3.0	**2.7**	1.2	**1.9±0.4**	0.3±0.4	**2.4±0.3**	2.0±0.1	199s
IS-B	4.0	3.3	**3.5**	0.9	**1.9±0.6**	0.3±0.1	**2.6±0.3**	2.2±0.4	137s
IS-R	7.8	2.8	**3.1**	**3.1**	**1.1±0.0**	0.5±0.5	**1.5±0.0**	**1.5±0.4**	472s
IS-M	5.5	2.8	**4.1**	1.9	**2.2±0.3**	0.4±0.1	**2.1±0.0**	1.6±0.5	183s
ISMIR-G	4.0	3.2	**3.5**	1.7	**2.3±0.6**	0.6±0.1	**2.4±0.2**	2.0±0.4	330s
RWC	4.0	3.1	**3.7**	2.0	**2.1±0.5**	0.6±0.7	**2.8±0.2**	2.3±0.2	312s
W-Mean	4.65	3.08	**3.53**	2.05	**2.04±0.07**	0.50±0.30	**2.40±0.05**	2.00±0.32	293s

Table 7.5 Summary of the subjective evaluation results. The weighted means are calculated using a weight proportional to the number of titles in each corpus.

with reference tracks or binaural recorded songs as possible. It would be perfect if it could also contain binaural recorded songs together with their reference tracks which are rather difficult to obtain because binaural mixing and at the same time preserving signal quality of the mixdown is known not to be an easy task. On the other hand it may be argued that binaural recorded titles are a rarity in practice and therefore an algorithm tuned to work with binaural songs would rather be of academic interest rather than practical. Still, musical instrument separation is a difficult task which is not satisfactorily solved by the time being and therefore even being able to separate binaural songs might constitute an important achievement.

Regarding the subjective evaluation results we might say that the HSBSS comes out to be very promising at the moment and deserves more attention in future as there are a vast number of improvements which could be applied to this algorithm. An interesting thought here would be to unify the concepts of template matching and histogram based blind source separation. It should be expected to produce superior results to what we have seen so far in our experiments.

7.6 Summary and Conclusions

At first we gave an overview over the corpora used in our benchmarking. We pointed out that the titles from the BASS-dB are very well suited for testing instrument separation algorithms as they contain their reference tracks before mixdown. The existing corpora lacked more titles of this type or binaural recorded songs which would suit the HSBSS algorithm best as they would also contain much phase information which mono downmix compatible stereo recordings lack. Therefore we decided to build our own corpus called the Instrument Separation corpus or shortly the IS corpus.

Our corpus was made to contain four binaural recorded songs and four titles having their reference tracks available. Additionally we also included some module files which are known by their typical .MOD extension as they proved to be easily splittable into independent tracks if we could find a mapping of the channels in the module file to tracks with the constraint that the instruments do not spread or wander across tracks. Furthermore as the direct template matching method is practically the inversion of playing a module we thought they could be well suited for objectively evaluating the DTM.

We then described the testing conditions for the evaluation as well as the criteria for subjective and objective evaluation. For the subjective evaluation we indrocuded the grade of separation score and the perceived quality score of the resultant signal after separation while for the objective testing we elaborated on the SNR error measure for stereo audio files. Furthermore we also described a simple baseline algorithm to be used as a reference point for evaluation.

After presenting the evaluation results we discussed the outcome and how evaluation could be improved for future work. This includes the expansion of the IS corpus to contain more titles as well as a new kind of title which is recorded binaurally and additionaly has its reference tracks available. Furthermore the idea was presented to unify the DTM and HSBSS concepts to create a hybrid which is expected to perform better than its base ideas separately.

Concluding this section we want to say that although the SNR figures show rather low SN ratios with 0.94dB in average where we would expect about 8-10dB, they should not be discouraging as the subjective evaluation shows that the HSBSS algorithm performs average. Additionally it also compares favorably to the baseline. So we may conclude that the algorithms which are partly new and partly improvements over existing methods, have shown their benefits and strengths. Still the weak point of this evaluation was our inability to get hold of some implementations of related algorithms in the field of blind audio source separation which surely would have made the results more interesting and would have given a better grasp of the performance of our algorithms. So we should strive harder to obtain working implementations of other algorithms for comparison purposes in some future work.

Chapter 8

Conclusions and Future Work

During this thesis we have presented three algorithms for separating sounds of instruments from musical recordings.

We started with the general problem formulation in Chapter 1. Then we presented the related work in Chapter 2 reviewing general purpose blind source separation methods using auditory cues for discriminating between sources as well as more specialized algorithms using harmonic and sinusoidal modelling in order to estimate the harminic structure of instruments. We also reviewed some template based approaches which brought us to the idea of the iterative template matching algorithm in Chapter 4. Continuing the related work chapter, we reviewed the benchmarking methods found in literature together with some of their problems as for example the artificial mixed sound titles which do not modell well reality. This was a problem we also stumbled across in our evaluation chapter later on as we could not find binaural files with reference tracks but only their more popular artificially mixed counterparts.

In Chapter 3 we introduced our first template based approach. The Direct Template Matching (DTM) algorithm is thought to separate repeating tones by averaging over their repetitions in order to eliminate other tones potentially playing at the same time. We begin the development of the algorithm with the introduction of the onset vector containing all onsets of a single tone and the initialization procedure. Following the main algorithm which iterated over all instruments we discussed the tone search procedure which searched the correlation surface for peaks representing tone onsets which were subsequently filtered to keep only the most plausible ones. The tone learning step then adapted the template of the tone to match the found onsets through a form of Newton's method. Subsequently some refinement methods were presented like shifting the template relative to its onsets in order to cover more of the tone energy or another refinement being phase matching by subsample accurate tuning of the onset in order to prevent phase deviations leading the destructive interference in the high frequency waves of the tone template which then would exhibit an undesired high frequency damping during the learning phase.

Due to some problems in the direct approach concering the ability to find plausible tone onsets we created a second approach called iterative template matching (ITM) in Chapter 4. The idea for improvement was to create an self-organizing steering vector which is the continuous form of the onset vector. As with the first approach we began its development with an initialization procedure. The iterative main algorithm which followed, consisted of a synthesizing step where the output signal was generated and a learning step using a gradient descent procedure called RPROP to adapt the templates and the steering vector where the steering vector was constrained to be kept sparse which ideally should make it approach its discrete version, the onset vector. Some refinements were then presented with the first being some modifications of the non-sparseness cost function used for the gradient descent algorithm. Further proposals for refinement were a lateral inhibition technique in order to assure minimum distance between spikes of the steering vector which did not work at first and thus needed to be further elaborated upon. Finally phase matching by upsampling the input signal was also discussed as an option for improvement like in the DTM algorithm.

As a third option for separation, we decided to move to the domain of blind source separation which lead us to the histogram based blind source separation approach (HSBSS) in Chapter 5. This algorithm works on an input feature space which can be easily visualized as a histogram with its axes representing magnitude phase and time shift between the audio channels while the intensities represent the average loudness of the frequencies having these features. The first two features resemble the two cues used by the human auditory system, the interaural intensity difference (IID) and the interaural time difference (ITD), a fact which makes this approach more biologically plausible. A limiting problem we stumbled upon in that chapter was now that frequencies above a given treshold have ambiguous time shifts causing spreading of otherwise well localizable points in the histogram which limits the usable frequency range. In order to split the signal into components representing instruments a RBF network was trained to approximate the intensity distribution across the histogram. The neurons of the trained network were then used to create decision boundaries for feature value regions in the histogram. Each region had its associated class. Splitting was done by copying each frequency falling into one of these regions to the previously zero initialized spectrum of the region's class. The resulting instrument or group of instruments was the time-transformed spectrum of each class. Following the algorithm description we then presented some improvements for suppressing artefacts through adjusting the window size of the fourier transform to minimize the pre-echo effect and using of overlaps in order to reduce musical noise.

In Chapter 6 we described some details of our implementations of the three algorithms. More precisely we specified the libraries we used which were Marsyas, fftw and GetPot and identified Microsoft Visual Studio 2003 as our programming environment. After outlining the code structure we then discussed the problem of 32 bit code architecture applications which can only address 2 GB of physical memory with the solutions of swapping large arrays temporarily to the hard disk or porting the code 64 bit which would break compatibility to many actual machines and operating systems. The theoretical aspect of FFT size on speed was then discussed showing that a convolution can be split up in some smaller convolutions needing less operations and making better use of the processor cache thus increasing performance.

The benchmarking and evaluation of our algorithms was done in Chapter 7. Following an overview of the existing corpora and which parts of them we used, an own corpus called Instrument Separation (IS) corpus was introduced. We built it to contain binaural recorded songs as well as freely available titles with reference tracks. We also included module files which resemble the MIDI format and additionally include the instrument samples in the file, which is not the case for MIDI where the samples are stored as a common database in the playback device. In the following sections we then described the testing conditions and defined the scoring criteria for the subjective evaluation as well as the error measure for the objective evaluation part which was taken to be the signal to noise ratio (SNR). The results were then presented which showed the DTM and the HSBSS to score closely in terms of dB SNR with only the baseline being off. Due to the missing tone to instrument mapping part of the direct template matching approach we had to use an idealized clustering using information from the reference files and thus could not effectively compare the DTM and HSBSS. As the DTM was expected to show higher scores due to its idealized clustering part we believe that the HSBSS has a better performance especially as it is double as fast. The following subjective evaluation showed a wide gap between the separation capability of the HSBSS and the baseline while this gap was much narrower for the signal quality of the separated result.

The algorithms described during this thesis, left a lot of work to do. This also happened due to tight time constraints for the thesis. Bringing the algorithms and their impelementation to maturity would have lasted more than one semestre so something had to be kept unfinished or not be done at all. In order to compensate for these shortcomings we have given ideas for future work in every chapter, especially in the summary section. Here we will now give a summary of much of these ideas.

The first algorithm has a very simple initialization procedure using randomness. As initialization is very important to this algorithm a more educated guess as for example with some frequency-domain clustering or using a repetition indicator could prove useful. Furthermore due to high variations in the onset filtering algorithm a some more research on how to stabilize it in order to generate a more predictable and stable amount of peaks might improve results. Concerning the learning method a more precise curvature information could be used than our implementation of Newton's algorithm which might raise convergence speed. The template offsetting method should be looked closer into as it does not work as desired. Still we consider the offsetting to be a good tool for moving a template to cover a higher energy point of a tone. Not overlooked shall be the missing part of organizing tones into instruments as its absence proved to be a problem for evaluation and for practical use. Also worth mentioning is that some or most of the work could be moved into the frequency domain where instruments should be better decorrelated from the beginning. And lastly it is always a good idea make use of higher semantic structures of music and thus relationships between tones.

There are also plenty of improvements possible for the iterative template matching algorithm. For example a better non-sparseness cost function having its minimum located at a point nearer the real number of onsets present in the music instead of just one onset, could improve results considerably. Then a memory problem of storing all steering vectors when a reasonably number of them are used should also be looked at in some future work. It is possible to swap most of the arrays to the hard disk and do some local optimizations on the one found in memory but that could alter the convergence behaviour and the final result of the algorithm in terms of quality. The lateral inhibition for keeping onsets from the same tone apart should also be improved in order to work as actually there is no other method implemented for keeping very closely spaced onsets apart. The non-sparseness cost function could include information from inter-channel correlation. For example it could favour onsets if they are stereo aligned or penalize them if they are misaligned. Lastly we should mention that the template vector part could be learned exactly rather than using some kind of gradient descent and in that way possibly speeding up convergence.

The HSBSS algorithm is also in need for some improvements. For example a time shift disambiguation heuristic which tries all possible locations of a frequency sample and chooses the one for which that frequency seems to be a harmonic, could be implemented. Minimizing the smearing effect is expected to improve results of the RBF clustering stage. Furthermore we could also try to extend the histogram by adding new features and thus dimensions. As the histogram may be viewed as an input space it does not necessarily need to be visualizable if it better serves the purpose of discriminating between instruments. Another practial improvement would be to be able to estimate the number of clusters automatically. Actually it has to be supplied by the user which in turn usually just looks at the histogram and tentatively listens to some parts of the song. Still another improvement to be considered would be some heuristic to be able to separate harmonics sharing the same frequency bin. This is rather complicated task but it may improve results qualitatively.

A lot of improvements can also be done at the implementation level. For example using multiple parallel threads in order to be able to use all cores of a modern multicore machine. Then we can also use library specific optimizations like the wisdom mechanism of the fftw in order to calculate and store parameters which lead to much faster FFTs as these transforms are used widely in all three separation algorithms.

Finally there is room for improvement even for the benchmarks. The IS corpus was rather small and therefore some results could not be interpreted with certainity as the fluctuations might also be caused by coincidence. It would be interesting to have some binaural files with their reference tracks. As it seems, best chances to get some are to make them on one's own as for example using multiple speakers with multiple microphones and possibly doing multiple recordings runs. Furthermore we should mention that we had no code of another program in the field of musical instrument separation or blind audio source separation to test with which is a great loss for the benchmarks. This should be a priority for future evaluations as it would make the performance better comparable to other existing algorithms.

Closing this chapter we should also mention that we had the idea of unifying the concepts of template matching and histogram base blind source separation into one algorithm. It might result into a powerful combination.

Appendix A

Corpora

In the following we will give the details of the corpora and collections used. Note that the songs we use from the BASS-dB, ISMIRgenre and RWC are only a subset of the much bigger corpus. Our own corpus, the Instrument Separation or IS corpus was defined in Section 7.2 and is fully used in this thesis.

We should also note here that the BASS-dB seems to be no longer maintained at [37] and therefore we give here direct links to where to find the songs belonging to this corpus.

For the module file part of the IS corpus, the mapping between channels in the module file and reference tracks is available in Table A. This mapping was used when generating the reference tracks in order to insure minimal instrument wandering between them. Please note that one module file channel corresponds to a pair of audio file channels after being rendered. Rendering was done at a sampling rate of 44.1kHz.

Abbrev.	Track 1	Track 2	Track 3	Track 4	Track 5	Track 6	Track 7
IS-M T1	CH1	CH2	CH3	CH4			
IS-M T2	CH1+CH2	CH3	CH4+CH5+CH6+CH7	CH8	CH9+CH10	CH11	
IS-M T3	CH1+CH4	CH2	CH3	CH5+CH6	CH7+CH8		
IS-M T4	CH1	CH2+CH3	CH4	CH5+CH6	CH7+CH8	CH9	CH10

Table A.1 Mappings between module file channels and reference tracks. For more details to each module file please see the next tables.

Abbrev.	Song Title	Corpus	Availability	Ref. Tracks	Binaural
BASS T1	Espi Twelve - No More Work	BASS-dB	www.archive.org	4	
BASS T2	Espi Twelve - Sun Under Shadows	BASS-dB	www.archive.org	5	
BASS T3	Mister Mouse - Natmin	BASS-dB	www.archive.org	6	
IS-B T1	Camden Catholic High School Jazz Band	IS	www.soundprofessionals.com		✓
IS-B T2	Bill Weaver Composition - ERIN	IS	www.sonicstudios.com		✓
IS-B T3	Jim Dukey, REC Eng./Musician - Hayden Cello #5	IS	www.sonicstudios.com		✓
IS-B T4	Maynard Ferguson	IS	www.soundprofessionals.com		✓
IS-M T1	heatbeat of rebels - Kikka-Walz	IS	modarchive.org	4	
IS-M T2	Cybelius - Radio 10091	IS	amp.dascene.net	6	
IS-M T3	Purple Motion of Future Crew - Satellite One	IS	amp.dascene.net	5	
IS-M T4	Cybelius - Smokater	IS	amp.dascene.net	7	
IS-R T1	everamzah - Lightning Room	IS	www.archive.org	7	
IS-R T2	everamzah - To My Ear	IS	www.archive.org	5	
IS-R T3	Mister Mouse - Seven Years of Sorrow	IS	www.archive.org	10	
IS-R T4	PsychoVoyager - Above It All	IS	www.archive.org	9	

Table A.2 BASS-dB and IS corpus. Details on the songs from both corpora, ordered according to the abbreviation used in this thesis.

Abbrev.	Song Title	Corpus	Availability	Ref. Tracks	Binaural
ISMIR-G T1	La Riche and Co. - Allegro, Sonata for Violoncell	ISMIRgenre			
ISMIR-G T2	classical artist 1 album 2 track 2	ISMIRgenre			
ISMIR-G T3	Belief Systems - Blue-Tinted Sunglasses	ISMIRgenre			
ISMIR-G T4	Strojovna 07 - Palacinka	ISMIRgenre			
ISMIR-G T5	Jag - My Momma Told Me	ISMIRgenre			
ISMIR-G T6	Drop Trio - Invisible Pants	ISMIRgenre			
ISMIR-G T7	Electric Frankenstein - Born Wild	ISMIRgenre			
ISMIR-G T8	Seismic Anamoly - Ten Million Tears	ISMIRgenre			
ISMIR-G T9	The West Exit - Artificial	ISMIRgenre			
ISMIR-G T10	Norine Braun - Hanna to Hollywood	ISMIRgenre			
ISMIR-G T11	Solace - Circle (5-8, 6-8, 7-8)	ISMIRgenre			
ISMIR-G T12	world artist 112 album 1 track 2	ISMIRgenre			
RWC T1	01 - Unknown Artist - Track01	RWC	staff.aist.go.jp/m.goto/RWC-MDB		
RWC T2	02 - Unknown Artist - Track02	RWC	staff.aist.go.jp/m.goto/RWC-MDB		
RWC T3	08 - Unknown Artist - Track08	RWC	staff.aist.go.jp/m.goto/RWC-MDB		
RWC T4	09 - Unknown Artist - Track09	RWC	staff.aist.go.jp/m.goto/RWC-MDB		
RWC T5	12 - Unknown Artist - Track12	RWC	staff.aist.go.jp/m.goto/RWC-MDB		
RWC T6	19 - Unknown Artist - Track03	RWC	staff.aist.go.jp/m.goto/RWC-MDB		
RWC T7	21 - Unknown Artist - Track05	RWC	staff.aist.go.jp/m.goto/RWC-MDB		
RWC T8	30 - Unknown Artist - Track14	RWC	staff.aist.go.jp/m.goto/RWC-MDB		
RWC T9	33 - Unknown Artist - Track01	RWC	staff.aist.go.jp/m.goto/RWC-MDB		
RWC T10	35 - Unknown Artist - Track03	RWC	staff.aist.go.jp/m.goto/RWC-MDB		

Table A.3 ISMIRgenre collection and RWC corpus. Details on the songs from both corpora, ordered according to the abbreviation used in this thesis.

Bibliography

[1] L. Benaroya, F. Bimbot, and R. Gribonval. Audio source separation with a single sensor. *IEEE Transactions on Audio, Speech and Language Processing*, 14(1):191–199, 2006.

[2] P. Bofill. Underdetermined blind separation of delayed sound sources in the frequency domain. *Neurocomputing*, 55(3-4):627–641, 2003.

[3] E. G. Boring. Auditory theory with special reference to intensity, volume, and localization. *American Journal of Psychology*, 37(2):157–188, 1926.

[4] A. S. Bregman. *Auditory Scene Analysis: Perceptual Organization of Sound*. MIT Press, Cambridge, MA, USA, 1990.

[5] G. J. Brown. *Computational Auditory Scene Analysis: A Representational Approach*. PhD thesis, University of Sheffield, Sheffield, UK, 1992.

[6] M. Cooke. *Modeling Auditory Processing and Organization*. Cambridge University Press, Cambridge, UK, 1993.

[7] M. R. Devos and G. A. Orban. Self learning backpropagation. In *Proceedings of the NeuroNimes*, pages 469–476, 1988.

[8] M. R. Every and J. E. Szymanski. Separation of synchronous pitched notes by spectral filtering of harmonics. *IEEE Transactions on Audio, Speech and Language Processing*, 14(5):1845–1856, 2006.

[9] M. Frigo and S. G. Johnson. The design and implementation of fftw3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Invited paper, Special Issue on Program Generation, Optimization, and Platform Adaptation.

[10] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, N. L. Dahlgren, and V. Zue. TIMIT acoustic-phonetic continuous speech corpus. Linguistic Data Consortium, Philadelphia, 1993.

[11] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC music database: Popular, classical, and jazz music databases. In *Proceedings of the 3rd International Conference on Music Information Retrieval (ISMIR 2002)*, pages 287–288, 2002.

[12] R. Gribonval, L. Benaroya, E. Vincent, and C. Févotte. Proposals for performance measurement in source separation. In *Proceedings of the 4th International Symposium on Independent Component Analysis and Blind Signal Separation (ICA 2003)*, pages 763–768, 2003.

[13] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1994.

[14] M. Helen and T. Virtanen. Separation of drums from polyphonic music using non-negative matrix factorization and support vector machine. In *Proceedings of the 13th European Signal Processing Conference (EUSIPCO 2005)*, 2005.

[15] G. Hu and D. L. Wang. Monaural speech separation. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS 2002)*, volume 15, pages 1221–1228, Cambridge, MA, USA, 2002. MIT Press.

[16] G. Hu and D. L. Wang. Auditory segmentation based on event detection. In *Proceedings of the ISCA Tutorial and Research Workshop on Statistical and Perceptual Audio Processing (SAPA 2004)*, 2004.

[17] R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1(4):295–307, 1988.

[18] M. Kim and S. Choi. On spectral basis selection for single channel polyphonic music separation. In W. Duch, J. Kacprzyk, E. Oja, and S. Zadrozny, editors, *Proceedings of the International Conference on Artificial Neural Networks (ICANN 2005)*, volume 3697 of *Lecture Notes in Computer Science*, pages 157–162. Springer, 2005.

[19] Y. Li and D. L. Wang. Singing voice separation from monaural recordings. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR 2006)*, pages 176–179, 2006.

[20] A. S. Master. *Stereo Music Source Separation via Bayesian Modeling*. PhD thesis, Stanford University, Stanford, CA, USA, 2006.

[21] R. Meddis. Simulation of auditory–neural transduction: Further studies. *The Journal of the Acoustical Society of America*, 83:1056–1063, 1988.

[22] E. H. Moore. On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society*, 26:394–395, 1920.

[23] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

[24] R. D. Patterson. Auditory images: How complex sounds are represented in the auditory system. *Journal of Acoustical Society of Japan (E)*, 21(4):183–190, 2000.

[25] R. D. Patterson, J. Nimmo-Smith, J. Holdsworth, and P. Rice. An efficient auditory filterbank based on the gammatone function. APU Report 2341, MRC, Applied Psychology Unit, Cambridge, UK, 1988.

[26] R. D. Patterson, K. Robinson, J. Holdsworth, D. McKeown, C. Zhang, and M. Allerhand. Complex sounds and auditory images. In Y. Cazals, L. Demany, and K. Horner, editors, *Auditory Physiology and Perception: Proceedings of the 9th International Symposium on Hearing (ISH 1991)*, volume 83 of *Advances in the Biosciences*, pages 429–446, Oxford, England, 1992. Pergamon Press.

[27] R. Penrose. A generalized inverse for matrices. *Proceedings of the Cambridge Philosophical Society*, 51:406–413, 1955.

[28] M. D. Plumbley, S. A. Abdallah, T. Blumensath, and M. E. Davies. Sparse representations of polyphonic music. *Signal Processing*, 86(3):417–431, 2006.

[29] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999.

[30] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In *Proceedings of the IEEE International Conference on Neural Networks (ICNN 1993)*, volume 1, pages 586–591, 1993.

[31] N. Roman, D. L. Wang, and G. J. Brown. Speech segregation based on sound localization. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2001)*, volume 4, pages 2861–2866, 2001.

[32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, et al., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[33] S. D. Teddy and E. M. K. Lai. Model-based approach to separating instrumental music from single channel recordings. In *Proceedings of the 8th International Conference on Control, Automation, Robotics and Vision (ICARCV 2004)*, volume 3, pages 1808–1813, 2004.

[34] T. Tollenaere. Supersab: Fast adaptive back propagation with good scaling properties. *Neural Networks*, 3(5):561–573, 1990.

[35] K. Turkowski. Filters for common resampling tasks. In A. Glassner, editor, *Graphics Gems I*, pages 147–165. Academic Press, San Diego, CA, USA, 1990.

[36] G. Tzanetakis and P. Cook. Marsyas: A framework for audio analysis. *Organized Sound*, 4(3):169–175, 2000.

[37] E. Vincent, R. Gribonval, C. Févotte, and al. *BASS-dB: the Blind Audio Source Separation Evaluation Database.* URL: `http://www.irisa.fr/metiss/BASS-dB/`.

[38] T. Virtanen. Separation of sound sources by convolutive sparse coding. In *Proceedings of the ISCA Tutorial and Research Workshop on Statistical and Perceptual Audio Processing (SAPA 2004)*, 2004.

[39] T. Virtanen. *Sound Source Separation in Monaural Music Signals*. PhD thesis, Tampere University of Technology, Finland, 2006.

[40] O. Yilmaz and S. Rickard. Blind separation of speech mixtures via time-frequency masking. *IEEE Transactions on Signal Processing*, 52(7):1830–1847, 2004.

[41] Y. G. Zhang and C. S. Zhang. Separation of music signals by harmonic structure modeling. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems (NIPS 2005)*, volume 18, pages 1619–1626, Cambridge, MA, USA, 2005. MIT Press.

[42] F. Zheng, G. Zhang, and Z. Song. Comparison of different implementations of MFCC. *Journal of Computer Science and Technology*, 16(6):582–589, 2001.

[43] A. Zils, F. Pachet, O. Delerue, and F. Gouyon. Automatic extraction of drum tracks from polyphonic music signals. In *Proceedings of the 2nd International Conference on Web Delivering of Music (WEDELMUSIC 2002)*, pages 179–183, 2002.