Underdetermined Blind Source Separation for Audio Signals

Der Fakultät für Elektrotechnik und Informationstechnik der Rheinisch-Westfälischen Technischen Hochschule Aachen vorgelegte Dissertation zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften

von
Diplom-Ingenieur
Martin Spiertz
aus Eschweiler

Berichter:
Univ. Prof. Dr.-Ing. Jens-Rainer Ohm
Prof. Dr. rer. nat. Michael Vorländer

Tag der mündlichen Prüfung: 21 Februar 2012
Vorwort

Dieses Buch entstand während meiner Zeit als wissenschaftlicher Angestellter des Instituts für elektrische Nachrichtentechnik. Hier möchte ich die Gelegenheit nutzen um allen zu danken, die mich während dieser Zeit unterstützt haben.

Zuallererst möchte ich mich bei Prof. Ohm für die Unterstützung, das Vertrauen und die Möglichkeit an seinem Institut zu promovieren bedanken. Prof. Vorländer danke ich für das Interesse an meiner Arbeit und die Übernahme des Korreferats.


Desweiteren möchte ich mich auch bei allen Studenten bedanken, durch deren tatkräftige Unterstützung tiefere Einblicke in die Themen meiner Doktorarbeit erst ermöglicht wurden. Stellvertretend nenne ich hier die Arbeiten von Christian Rohlfing und Reimund Dratwa, die mir in der Schlussphase meiner Doktorarbeit viele neue Erkenntnisse brachten.

Contents

Abbreviations .................................................. 1

1 Introduction .................................................. 2
  1.1 Outline .................................................. 3

2 Fundamentals ............................................... 4
  2.1 Conventions .............................................. 4
    2.1.1 Distances and Metrics ............................. 5
    2.1.2 Linear Transforms ................................ 6
  2.2 Time-Frequency Transforms ............................ 7
    2.2.1 Short-Time Fourier Transform ..................... 9
    2.2.2 Logarithmic-Frequency transforms ................. 11
  2.3 Non-Negative Factorization ............................ 13
    2.3.1 Matrix Factorization ............................... 13
    2.3.2 Non-Negative Tensor Factorization ............... 14
    2.3.3 Expectation Maximization for NTF ................ 17
    2.3.4 Estimating the Number of Relevant Non-Negative Components ............................. 18
    2.3.5 Initialization of NTF .............................. 22
    2.3.6 Statistical Motivation of Cost Functions ........ 24
  2.4 Psychoacoustics ....................................... 24
    2.4.1 Logarithmic Behaviour of Human Perception ........ 24
    2.4.2 Spectral Range of Human Listeners ............... 26
    2.4.3 Quality Measures .................................. 27
  2.5 Audio Data ............................................. 29
    2.5.1 Mixing Model ....................................... 31

3 Blind Source Separation: State-Of-The-Art ............. 33
  3.1 Spectrogram Factorization with Supervised Clustering Methods ......................... 33
    3.1.1 Non-Blind Clustering ................................ 33
    3.1.2 Trained Models ..................................... 34
  3.2 Spectrogram Factorization with Unsupervised Clustering Methods ....................... 35
  3.3 Spectrogram Factorization embedding Clustering ........................................ 36
  3.4 Source Separation Frameworks beside Spectrogram Factorization ....................... 37
    3.4.1 Stereo BSS by Attenuation and Delay ............. 37
    3.4.2 BSS Specialized on Certain Types of Mixtures .......... 38
    3.4.3 BSS by Segmentation of Spectrogram Images ........ 38
  3.5 Summary ................................................ 38
4  **Note Separation by Non-Negative Tensor Factorization and Supervised Clustering**  
4.1 Signal Flow for Note Separation  
4.1.1 Time-Frequency Transform  
4.1.2 Note Separation by NTF  
4.1.3 Signal Synthesis  
4.1.4 Reference Clustering  
4.2 Influence of Time-Frequency Transform  
4.2.1 Transform Length and Hop Size for STFT  
4.2.2 Logarithmic Frequency Scaling  
4.2.3 Summary and Conclusions  
4.3 Influence of Factorization Method  
4.3.1 Initialization and Number of Iterations  
4.3.2 Choice of $\beta$ and Model Order  
4.3.3 Mixing Model in the Multichannel Case  
4.4 Summary and Conclusions  

5  **Audio Features**  
5.1 Pre-Processing of Features  
5.1.1 Data Model/Data Normalization  
5.1.2 Dimension Reduction  
5.2 Cepstral Features  
5.2.1 Mel Frequency Cepstrum Coefficients  
5.2.2 Temporal Source-Filter Model  
5.3 Frequency-based Features  
5.3.1 Frequency Features based on Spectrogram $Y$  
5.3.2 Frequency Features based on Matrix $B$  
5.4 Temporal-based Features  
5.5 Spatial-based Features  
5.5.1 Summary  

6  **Unsupervised Note Clustering**  
6.1 Clustering Algorithms  
6.1.1 Partitional Clustering Algorithms  
6.1.2 Hierarchical Clustering  
6.1.3 Initialization of Clustering Algorithms  
6.2 Basic Clustering Algorithms  
6.2.1 Cepstral Clustering  
6.2.2 Clustering all other Features  
6.2.3 Clustering beside Audio Data $A$  
6.3 Summary  

7  **Combination of Clustering Strategies**  
7.1 Pre-Selection of Feature Spaces  
7.2 Combination in Signal Space  
7.3 Combination in Feature Space
7.4 Combination in Cluster Space ................................................. 122
  7.4.1 Adaptation to Dynamic Differences ................................. 123
  7.4.2 Adaptation to Tempo .................................................. 124
  7.4.3 Interpretation of Cluster Reliability ................................. 126
  7.4.4 Summary ................................................................. 127
7.5 Combination Techniques beside Audio Data $A$ ......................... 127
7.6 Convex Fuzzy C-Means with Adaptation to Tempo ....................... 129
7.7 Summary ................................................................. 132

8 Summary and Conclusions .................................................. 134
  8.1 Summary ................................................................. 134
  8.2 Outlook ................................................................. 135

A Detailed Description of Data $A$ ........................................... 136

B Additional Factorization Methods beside Standard NTF ................. 138
  B.1 Multichannel Factorization by Mid-Side Encoding ....................... 138
  B.2 Expectation Maximization vs. Multiplicative Updates ................... 139

C Separation Quality for Single Features .................................. 140

D Usage of Start and Stop Windows in the Context of Blind Source Separation 141

E Applications of Blind Source Separation - Beside Music ............... 143

F Interaction of Time-Frequency Transform and Non-Negative Tensor Factor-
ization ................................................................. 146
  F.1 Adaptive Time-Frequency Processing .................................... 146
    F.1.1 Adaptive Short Time Fourier Transform ......................... 146
    F.1.2 Strategies for Adaption of Time-Frequency Resolution ....... 149
    F.1.3 Summary .......................................................... 152
  F.2 Numerical Sensitivity of the Non-Negative Tensor Factorization .... 152
    F.2.1 Experiment 1: Different Analysis Windows ...................... 153
    F.2.2 Experiment 2: Different Random Initialization ................. 154
    F.2.3 Experiment 3: Zero-Padding .................................... 156
    F.2.4 Comparison of the Experiments .................................. 163
  F.3 Summary .............................................................. 164

G Matlab Code .............................................................. 166

Bibliography .............................................................. 174
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATFR</td>
<td>Adaptive Time Frequency Resolution</td>
</tr>
<tr>
<td>BCT</td>
<td>Box-Cox-transform</td>
</tr>
<tr>
<td>BSS</td>
<td>Blind Source Separation</td>
</tr>
<tr>
<td>CFCM</td>
<td>Convex Fuzzy C-Means</td>
</tr>
<tr>
<td>CFCM-A</td>
<td>Convex Fuzzy C-Means with Adaptive Weightings</td>
</tr>
<tr>
<td>dB</td>
<td>Dezibel</td>
</tr>
<tr>
<td>FCM</td>
<td>Fuzzy C-Means</td>
</tr>
<tr>
<td>ISA</td>
<td>Independent Subspace Analysis</td>
</tr>
<tr>
<td>ISD</td>
<td>Itakura-Saito Distance</td>
</tr>
<tr>
<td>LDA</td>
<td>Linear Discriminant Analysis</td>
</tr>
<tr>
<td>MES</td>
<td>Multi-Expert-System</td>
</tr>
<tr>
<td>MFCC</td>
<td>Mel Frequency Cepstrum Coefficients</td>
</tr>
<tr>
<td>NLM</td>
<td>Non-Linear Mapping</td>
</tr>
<tr>
<td>NMF</td>
<td>Non-Negative Matrix Factorization</td>
</tr>
<tr>
<td>NTF</td>
<td>Non-Negative Tensor Factorization</td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
</tr>
<tr>
<td>pdf</td>
<td>Probability Density Function</td>
</tr>
<tr>
<td>SFM</td>
<td>Source-Filter Model</td>
</tr>
<tr>
<td>STM</td>
<td>Sinus-Transient Modeling</td>
</tr>
<tr>
<td>STFT</td>
<td>Short-Time Fourier Transform</td>
</tr>
<tr>
<td>SVD</td>
<td>Singular Value Decomposition</td>
</tr>
<tr>
<td>WMV</td>
<td>Weighted Majority Voting</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Speech is one of the most important ways of human interaction. All kinds of acoustics and especially music and speech influence the mood and the reaction of humans in a wide range of environments, e.g. in arts, politics, or of course during free-time. The digitalization of a wide range of these auditory data has increased the influence of music and sounds on our daily life. Musicians playing in an ensemble, different speakers talking at the same time, the auditory impressions reach the human ear seldom alone, but usually as a mixture of wanted and unwanted signals.

The human listener is very well trained in detecting different active instruments, or more generally spoken, different active sound sources. The listener is able to concentrate on single sources, and suppress the other sources as disturbing background noise.

This useful ability of the above mentioned listener will be also helpful for automatic music analysis: Many analysis algorithms, like pitch detection or automatic music transcription will benefit from audio streams with only a single active instrument separated by an automatic source separation algorithm. If such a source separation is done without any knowledge of the active instruments, the algorithm is called blind source separation (BSS).

The BSS scenario is called under-determined, if the number of sources is larger than the number of sensors (microphones). If only a single sensor’s signal is at hand, the scenario is called monaural.

Many current BSS algorithms are based on the non-negative matrix factorization or its multi-channel variant, the non-negative tensor factorization (NTF). One shortcoming of the NTF is that it separates only single notes, not complete melodies. One approach for melody separation in literature is to extend the NTF by additional constraints and signal models in order to overcome this disadvantage e.g. by a source-filter model or shift invariance of the spectrum. Other current approaches try to utilize pattern classification techniques for clustering of the single notes. This thesis is about the latter approach. The task of pattern classification is explained as a twofold one in this thesis: Feature extraction and clustering.

There exists already a large bunch of algorithms for note-separation, audio feature extraction, and clustering. Therefore, the contribution of this thesis has to be specified clearly. The main topics of this thesis are:

- A comprehensive comparison of a wide range of pairs of clustering algorithms and audio features.
• The discussion of different combination strategies of different pairs to enhance the separation quality of the proposed BSS algorithm.
• The development of a single clustering algorithm which leads to a separation quality comparable to other state-of-the-art algorithms at significant lower computational complexity.

The additional constraints in the proposed approach are very narrow:
• No specialization or limitation on certain instruments is allowed in order to develop a universally valid model for source separation.
• The algorithms have to handle even monaural (single sensor) mixtures. Spatial information is optional not necessary.
• The computational complexity of feature extraction and clustering shall be small compared to the complexity of the note separation by NTF.
• No external information beside the number of active instruments is necessary for initialization of the algorithm.

Currently, no BSS algorithms exist that fulfill all of these conditions at the same time, which motivates this thesis.

1.1 Outline

This thesis is outlined as follows. Firstly, basic algorithms and definitions are given in Chapter 2. Secondly, a large range of state-of-the-art BSS algorithms are introduced in Chapter 3. Their (dis-)advantages are explained and it is outlined how the algorithms proposed in the later chapters will avoid these disadvantages. After that, the BSS framework used throughout this thesis is outlined in Chapter 4. The main contribution in this thesis is the conversion of separated notes to separated melodies by a clustering of the separated notes. This clustering is done in a supervised way in Chapter 4 in order to define an upper-limit regarding the separation quality for non-supervised clustering algorithms. After these first chapters used for introducing the task of blind source separation, the main topic of this thesis is outlined: Unsupervised note clustering for blind source separation. The general task of clustering or pattern classification is twofold: In Chapter 5, features are defined to describe characteristic attributes of each separated note. In Chapter 6, clustering algorithms are introduced to find similarities between the audio features to classify clusters, which are believed to be identical to the melodies of the active instruments. In this chapter, it will also be shown that simple (or basic) clustering algorithms are not able to reach the upper-limit of separation quality defined by optimal clustering of the separated note events. Therefore, in Chapter 7, the simple clustering algorithms are combined by three different approaches: Combination before clustering, after clustering, and after signal synthesis. It will be shown that the separation quality can be increased significantly by intelligent combination algorithms. This thesis closes with the conclusions and an outlook in Chapter 8.
Chapter 2

Fundamentals

This chapter starts with the introduction of basic ideas and naming conventions in Section 2.1. In Section 2.2, a rough overview over typical time-frequency transforms is given. In Section 2.3, matrix factorization algorithms and some of their strengths and weaknesses are discussed. In Section 2.4, the area of psychoacoustics is explained, as far as it relates to this thesis. Finally, in Section 2.5 the audio samples used in this thesis are introduced.

2.1 Conventions

For better readability of this work, we start with some general definitions.

- Throughout this thesis, variables with underlines corresponds to the complex-valued variables, dropping the underline is equivalent to taking the absolute value.
- This thesis deals only with digital signals already sampled by a sampling frequency $F_s$. Digital/analogue and analogue/digital conversion are beyond the scope of this work.
- Throughout this work, $x * y$ defines the convolution of two signals $x$ and $y$.
- The operator $\leftarrow$ is used as becomes in the context of variable assignment.
- We use a set of standard functions as defined in [1] and [2], with $x, y \in \mathbb{R}$ and $k, t \in \mathbb{N}$:

  \[
  \text{rect}(x) = \begin{cases} 
  1, & \text{for } |x| \leq \frac{1}{2} \\
  0, & \text{otherwise}
  \end{cases} \quad \text{(rectangular function)} \tag{2.1}
  \]

  \[
  \varepsilon(x) = \begin{cases} 
  1, & \text{for } x \geq 0 \\
  0, & \text{otherwise}
  \end{cases} \quad \text{(unit step)} \tag{2.2}
  \]

  \[
  \delta_{x,y} = \begin{cases} 
  1, & \text{if } x = y \\
  0, & \text{otherwise}
  \end{cases} \quad \text{(Kronecker symbol)} \tag{2.3}
  \]

  \[
  \max_t x(k, t) = \lim_{p \to \infty} \left( \sum_t x^p(k, t) \right)^{\frac{1}{p}} \quad \text{(maximum over a given dimension)} \tag{2.4}
  \]
• The first index for matrix indexing corresponds to the rows, the second to the columns. For tensors, the third index numbers the slices of the tensor.

2.1.1 Distances and Metrics

For numerical optimization, the term distance between arbitrary variables \( x \) and \( y \) needs to be defined. According to [3], a mathematical expression for distance grows with \( |x - y| \) and has a single minimum for \( x = y \).

![Graph showing the \( \beta \)-divergence for different values of \( \beta \).](image)

**Figure 2.1**: Visualization of the \( \beta \)-divergence for different values of \( \beta \). Two cases are plotted: \( y \) as a variable, \( x = 1 \) and \( x = 10 \).

**\( \beta \)-Divergence**  The \( \beta \)-divergence is one example for a distance:

\[
\begin{align*}
d_\beta(x, y) &= \begin{cases} 
\frac{x}{y} - \log \frac{x}{y} - 1 & \text{if } \beta = 0 , \\
 x (\log x - \log y) + y - x & \text{if } \beta = 1 , \\
\frac{1}{\beta(\beta-1)} (x^\beta + (\beta - 1) y^\beta - \beta xy^{\beta-1}) & \text{otherwise.}
\end{cases}
\end{align*}
\]

The equations for \( \beta = 0 \) and \( \beta = 1 \) are the results of the identity

\[
\lim_{\beta \to 0} \frac{x^\beta - 1}{\beta} = \log x .
\]

The \( \beta \)-divergence is defined for \( \beta \in \mathbb{R} \). The Euclidean distance (\( \beta = 2 \)), the generalized Kullback-Leibler divergence (\( \beta = 1 \)) as used in [4], and the Itakura-Saito distance (\( \beta = 0 \)) as used in [3] can be expressed by the \( \beta \)-divergence.

In Figure 2.1, the \( \beta \)-divergence is visualized for two cases: \( x = 1 \) and \( x = 10 \). The values for \( y \) are varied around the corresponding \( x \)-value. It can be seen that especially for large amplitudes, smaller values for \( \beta \) result in much smoother enhancement of the divergence.
**Spherical Distance**  The normalized cross-correlation [5] can be used to measure the similarity between two vectors:

\[
\phi_{xy} = \frac{\sum_n x(n) \cdot y(n)}{\sqrt{\sum_n x^2(n)} \sqrt{\sum_n y^2(n)}}. \tag{2.7}
\]

Because of the denominator, \( \phi_{xy} \) is limited to the range \(-1 \leq \phi_{xy} \leq 1\). A spherical distance measure can be derived by

\[
d_{\text{spherical}}(x, y) = 1 - \phi_{xy}, \tag{2.8}
\]

as defined in [6]. The name *spherical* refers to the fact that Equation (2.7) measures the cosine of the angle between two vectors. The normalization shown in Equation (2.8) is necessary to convert the cosine to a distance.

**Metrics**  In [7], a metric has to fulfill four conditions:

- Non-negativity: \( d(x, y) \geq 0 \),
- Reflexivity: \( d(x, y) = 0 \) if and only if \( x = y \),
- Symmetry: \( d(x, y) = d(y, x) \), and
- Triangle inequality: \( d(x, y) + d(y, z) \geq d(x, z) \).

A commonly used metric is the Minkowski metric [7]:

\[
L_p(x, y) = \|x - y\|_p = \left( \sum_n |x(n) - y(n)|^p \right)^{\frac{1}{p}}. \tag{2.9}
\]

Note that Equation (2.9) fulfills the triangle inequality only for \( 1 \leq p \leq \infty \), \( d_\beta \) is only a metric for \( \beta = 2 \). Otherwise \( d_\beta \) is not symmetric. \( d_{\text{spherical}} \) is not a metric, because it does not fulfill the triangle inequality, which can be seen by the following two-dimensional counterexample of

\[
x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad y = \begin{pmatrix} \sqrt{3}/2 \\ 1/2 \end{pmatrix}, \quad \text{and} \quad z = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.10}
\]

**2.1.2 Linear Transforms**

In this thesis, a set of linear transforms according to [5] is used, e.g. the discrete Fourier transform (DFT):

\[
\mathcal{X}(k) = \text{DFT} (x(n)) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi \frac{nk}{N}}, \quad 0 \leq n, k < N, \tag{2.11}
\]

\[
x(n) = \text{IDFT} (\mathcal{X}(k)) = \frac{1}{N} \sum_{n=0}^{N-1} \mathcal{X}(k)e^{j2\pi \frac{nk}{N}}, \quad 0 \leq n, k < N, \tag{2.12}
\]
The scaling factor \( \frac{1}{N} \) used in Equation 2.12 is necessary for perfect reconstruction. Other definitions of the DFT uses a scaling factor of \( \frac{1}{\sqrt{N}} \) for the DFT and the DFT.

Another example for a linear transform is the discrete cosine transform (DCT):

\[
X(k) = \text{DCT}(x(n)) = c \cdot \sum_{n=0}^{N-1} x(n) \cos \left( \frac{k\pi}{N} \left( n + \frac{1}{2} \right) \right), \quad 0 \leq n, k < N, \tag{2.13}
\]

\[
c = \begin{cases} 
\sqrt{\frac{1}{N}}, & \text{if } k = 0 \\
\sqrt{\frac{2}{N}}, & \text{otherwise} 
\end{cases}
\]

The inverse DCT (IDCT) is straightforward and is therefore dropped here.

### 2.2 Time-Frequency Transforms

The time domain representation of a signal \( x(n) \) with time index \( n \) gives no information about its frequency content. To analyse a signal regarding its frequency content, the discrete Fourier transform (DFT) is applied to \( x(n) \), as explained in Equation (2.11). The frequency coefficients \( k \) correspond to analysis frequencies \( f = \frac{k}{N}F_s \), with \( F_s \) being the sampling frequency. The DFT assumes band limited periodical signals:

- The signal to analyse must be zero for frequencies above the Nyquist-frequency \( f_{\text{Nyquist}} = \frac{F_s}{2} \) such that the sampling theorem is satisfied. Otherwise, the discrete sampling in time domain would introduce alias.

- The signal is assumed to be periodical, because it is analysed for discrete frequency coefficients only. If the signal is not periodical, or the periodicity is not an integer-multiple of the signal length \( N \), the DFT of the signal reveals spectral smearing: The peaks in the magnitude spectrum are widened over several frequency bins. This spectral smearing can be explained by the interpretation of the transform length \( N \) as a multiplication with a \text{rect}-function (see also Section 2.2.1) of length \( N \). In Figure 2.2, such a convolution is shown for a signal consisting of three cosines (corresponding to the three peaks in frequency-domain). If the length of one period of the signal is an integer-multiple of \( N \), the periodically-repeated \text{rect}-function becomes a constant value. By this, the convolution in frequency domain simplifies to a convolution with a Dirac, which avoids the additional spectral smearing.

For further details about discrete time processing and/or discrete frequency processing, see also [1] or [8].

It is possible to extend the transform length \( N \) by zero-padding of \( x(n) \). By this, more frequency coefficients are evaluated. In Figure 2.2, \( x(n) \) and the corresponding result of the DFT \( X(k) \) are plotted. The dashed line corresponds to zero-padding, for the solid line no zero-padding is used. It can be seen that no additional information is gained by the artificially increased frequency resolution: Assuming the following spectra evaluated
by the DFT

\[ X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi \frac{n}{N}}, \text{ (without zero-padding)}, \quad (2.14) \]

\[ Y(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi \frac{2n}{2N}}, \text{ (with zero-padding by a factor of 2)}. \quad (2.15) \]

The summation for evaluating \( Y(k) \) can be stopped at index \( N - 1 \) because the rest of the signal \( x(n) \) is filled up with zeros. From these two equations it is easy to see, that by zero-padding with a factor of 2 the following values are identical:

\[ X(k) = Y(2k). \quad (2.16) \]

The values \( Y(2k + 1) \) are evaluated by interpolation in the complex-valued frequency domain, see also frequency sampling in [1].

In a second example regarding zero-padding, two signals with identical magnitude spectra are assumed. In Figure 2.3(a) two signals are plotted that fulfill this condition. If both signals are transformed by the DFT and the phase is discarded for analysis, both signals are identical. On the other hand, if the signals are transformed with zero-padding, the supporting points for the above mentioned interpolation have the same magnitude. The other frequency bins reveal the differences between both signals, as can be seen in Figure 2.3(b). Therefore, these interpolated points encode certain parts of the phase information of the underlying signal. If only the magnitude spectrum is used for analysis, it is possible to utilize information encoded in the phase by zero-padding. In later experiments, this influence of zero-padding is shown.

Equation (2.11) gives no information about the temporal evolution of the frequency content. To interpret information of both the time and the frequency domain, signals can be transformed locally into a frequency domain signal. Such transforms are called time-frequency transforms. In this section, the fundamentals about time-frequency transforms are explained.
2.2 Time-Frequency Transforms

Figure 2.3: Both signals on the left have the same magnitude spectrum, but different phases. On the right, the magnitude spectra of both signals are shown, evaluated with zero-padding. It can be seen that the supporting points for the interpolation operation caused by the zero-padding are identical for both magnitude spectra.

2.2.1 Short-Time Fourier Transform

The time-discrete short-time Fourier transform (STFT) is a well-known example for time-frequency transforms. It is explained in detail in [8]. Here, only the most important aspects will be explained. The signal $x(n)$ is multiplied with an analysis window $w_1(n)$ of length $w_s$ (window size). This windowed signal is transformed into the frequency domain by the DFT. The resulting spectrum forms one column of a matrix $X$. The next column is analysed by shifting the analysis window by a certain hop size $h_s$:

$$X(k, t) = \sum_{n=0}^{w_s-1} x(n + t \cdot h_s)w_1(n)e^{-j2\pi nk w_s}. \quad (2.17)$$

The output $y(n)$ of the inverse STFT of $X(k, t)$ is initialized by $y(n) = 0$. Then for each frame $t$ the result of the inverse discrete DFT is multiplied with a synthesis window $w_2(n)$ and added to $y(n)$ with a certain overlap of neighboring frames. Therefore, this procedure is called overlap-add [8]:

$$y(n + t \cdot h_s) \leftarrow y(n + t \cdot h_s) + w_2(n)\frac{1}{w_s} \sum_{k=0}^{w_s-1} X(k, t)e^{j2\pi \frac{n}{w_s}}. \quad (2.18)$$

Typically used analysis ($w_1$) and synthesis window ($w_2$) are:

- rect window: $w(n) = \text{rect}\left(\frac{n-N/2}{N}\right)$
- sine window: $w(n) = \text{rect}\left(\frac{n-N/2}{N}\right) \cdot \sin\left(\frac{\pi n}{w_s}\right)$
- Blackman window: $w(n) = \text{rect}\left(\frac{n-N/2}{N}\right) \cdot \left(0.42 - 0.5 \cos\left(\frac{2\pi n}{w_s}\right) + 0.08 \cos\left(\frac{4\pi n}{w_s}\right)\right)$
• **Hann window:** \( w(n) = \text{rect} \left( \frac{n-N/2}{N} \right) \cdot \frac{1}{2} \left( 1 - \cos \left( \frac{2\pi n}{w_s} \right) \right) \).

All window functions are equal to zero for \( n < 0 \) and \( n \geq N \) due to the multiplication with the \( \text{rect} \)-function. If not otherwise mentioned, the analysis and the synthesis windows are the same. The condition for perfect reconstruction (\( x(n) = y(n) \)) is:

\[
\sum_{t=-\infty}^{\infty} w_1(n - th_s) \cdot w_2(n - th_s) = \text{const} .
\] (2.19)

If Equation (2.19) is not fulfilled, the STFT and its inverse lead to additional distortions in an audio signal processing framework. Therefore, if the audio signal is not only analysed and possibly modified in the spectrogram domain, but also transformed back into the time domain, perfect reconstruction is a necessary condition for reducing the overall distortion of a system. In order to fulfill Equation (2.19), different windows are used for different quotients \( \frac{w}{h_s} \):

- \( h_s = w_s \): rect window,
- \( h_s = \frac{w_s}{2} \): sine window,
- \( h_s = \frac{w_s}{3} \): Blackman window, and
- \( h_s = \frac{w_s}{4} \): Hann window.

Fast implementations of the discrete Fourier transform exist\(^1\), especially for \( w_s \) being a power of two. Therefore, the transform lengths are restricted to be a power of two in the following. If \( w_s \) is not a power of two, the next power of two, greater than \( w_s \) is chosen as transform length, utilizing zero-padding for adjusting the transform length.

For real-valued input signals, the result of the DFT is complex conjugate symmetric [5]. Therefore, we can drop one half of the spectrum for further processing and reduce the size of \( \mathbf{X} \) to \( K \times T \), with \( K = \frac{w_s}{2} + 1 \). The first coefficient (index 0) of the DFT is always real-valued for real-valued input signals, because it is the DC-coefficient. For even values of \( w_s \), the coefficient with the index \( K = \frac{w_s}{2} \) is also real, because it is the summation of the coefficient itself and its conjugate complex counterpart of the symmetric second half of the spectrum. Therefore, the above mentioned reduction in size results in the same number of input- and output-samples for the DFT: \( w_s \) real-valued samples are transformed into two real-valued coefficients and \( \frac{w_s}{2} - 1 \) complex-valued samples each one corresponding to two real-valued samples (real-part and imaginary-part). Before synthesis, the missing half of the spectrogram can be reconstructed by mirroring the remaining first half of the spectrogram. This procedure helps to reduce computational complexity.

The result \( \mathbf{X} \) of the STFT is called spectrogram. The temporal resolution of the spectrogram is controlled by the hop size \( h_s \), the spectral resolution is controlled by the window size \( w_s \). The frequency resolution \( \Delta f \) and the temporal resolution \( \Delta t \) are given by

\[
\Delta f = \frac{F_s}{w_s} \text{[Hertz]},
\] (2.20)

\[
\Delta t = \frac{h_s}{F_s} \text{[Seconds]} .
\] (2.21)

As explained in [8], the temporal resolution and frequency resolution is constant for the whole spectrogram. In Figure 2.4, the STFT and many typical properties of acoustic

\(^1\)These fast transforms are called Fast Fourier Transform (FFT)
events are shown. In Figure 2.4(a), a harmonic note $x(n)$ is plotted. Additionally, the analysis windows are plotted. The position of the $t$-th analysis window in the left figure corresponds to the temporal position of the $t$-th column of $X(k, t)$ in the right figure. In the following, important terms regarding the spectrogram are explained. For further details, see also [9] or [10, p.406].

Regarding Figure 2.4(b), the temporal evolution of a note can be distinguished in silence (first column), transient (second column), and steady state (remaining columns). In the spectrogram, the transient segment is usually a short segment with a non-sparse spectrum. The steady state segment is usually of longer duration with a sparse spectrum. From this point of view it can be stated that transients are sparse in time domain and non-sparse in frequency domain. Steady state segments are non-sparse in time domain and sparse in frequency domain.

Additionally, the spectrum of a harmonic note can be described by the partials, which are frequency bins typically equally spaced in the spectrum containing nearly all of the spectrum’s energy. The pitch (or fundamental frequency) corresponds to the frequency bin of the first partial.

### 2.2.2 Logarithmic-Frequency transforms

As will be mentioned in Section 2.4.1, the perception and thereby the frequency resolution of humans is logarithmic. Therefore, it is an obvious idea to analyse signals with logarithmically spaced analysis frequencies. For this, a large amount of frequency transforms with logarithmic frequency resolution exists. Examples for such frequency transforms with non-uniform frequency resolution are wavelet-transforms or Gabor transforms, as explained in [8]. Wavelets are commonly used in image analyses and encoding. The high spatial resolution for high frequency content is advantageous for analyzing edges. In audio analysis such abrupt amplitude changes within a small number of samples is very unlikely. Therefore, wavelets are typically not used in audio analysis.
In [11] and [12] another transform with logarithmic frequency resolution is explained: The constant-Q transform. This transform is explained in more details to show general advantages and disadvantages of such transforms.

The basic idea is to analyse the signal only at analysis-frequencies \( f_{\text{analysis}} \) corresponding to a logarithmic scale, e.g. the Bark-scale. For simpler distribution over the frequency axis, the frequency resolution based on the pitches of the MIDI standard as proposed in [13] is discussed here:

\[
f_{\text{analysis}} = 440 \cdot 2^{(p-69)/12},
\]

with \( 21 \leq p \leq 108 \). The frequency resolution depends on the current analysis frequency:

\[
\Delta f = \frac{f_{\text{analysis}}}{Q},
\]

with \( Q \) being a constant value defining the frequency resolution. This relationship between analysis frequency \( f_{\text{analysis}} \) and frequency resolution \( \Delta f \) induces a better time resolution for higher analysis frequencies and a better frequency resolution for lower analysis frequencies.

One common problem of all these logarithmic frequency transforms is the inverse transform and by this the signal synthesis, as mentioned in [14]. Another open point is the handling of the resulting spectrogram: Most signal processing algorithms need a constant time-frequency grid as evaluated by the STFT. Therefore, a toolbox for computing the constant-Q transform and its inverse is introduced in [12]. Additionally, an interpolation scheme is defined to convert the non-uniform sampling of the analysis filters into a constant sampling regarding the temporal axis, thus resulting in a constant time-frequency grid. Unfortunately such interpolated constant-Q spectrograms are much larger than the spectrograms evaluated by the STFT: The time step used for interpolation over the temporal axis of the spectrogram is equal to the smallest hop size of all analysis filters. This interpolation is not necessary for inverting the constant-Q transform. Therefore, the over-completeness introduced by the interpolation leads to much higher computational complexity for the following signal processing steps.

One advantage of log-frequency transforms is the mathematical expression of harmonic spectra corresponding to different pitches \( f_0 \). Spectra corresponding to different values of \( f_0 \) can be expressed by a simple shift operation. Assuming a pure harmonic sound modeled in continuous time and frequency domain by \( N \) harmonically related oscillations:

\[
x(t) = \sum_{n=1}^{N} 2a_n \cos(2\pi n f_0 t) \rightarrow X_{f_0}(f) = \sum_{n=1}^{N} a_n \left( \delta(f - n f_0) + \delta(f + n f_0) \right).
\]

The negative part of the symmetric spectrum is dropped, as described in Section 2.2.1. A logarithmic scaling of the Fourier domain can be expressed by

\[
X_{f_0}(\log(f)) = \sum_{n=1}^{N} a_n \delta (\log(f) - \log(n f_0))
\]

\[
= \sum_{n=1}^{N} a_n \delta (\log(f) - \log(n) - \log(f_0)).
\]
Further assuming a small change in pitch results in a shifted version of $X(\log(f))$:

$$X_{f_0+\Delta f_0}(f) \approx \sum_{n=1}^{N} a_n \delta (\log (f) - \log (n (f_0 + \Delta f_0)))$$

(2.27)

$$= \sum_{n=1}^{N} a_n \delta (\log (f) - \log (n) - \log (f_0 + \Delta f_0))$$

(2.28)

$$= X_{f_0}(f + \Delta f_0).$$

(2.29)

A small change in pitch is usually assumed, because for large changes, the amplitudes $a_n$ will change significantly, as mentioned in [15]. Additionally, this model holds only for a small amount of harmonic oscillations (small $N$). For large $N$ even a small change in $f_0$ results in a large frequency shift.

### 2.3 Non-Negative Factorization

The idea of matrix factorization is basically to factorize a single matrix $X$ into a product of matrices regarding different conditions. The algorithms differ in their cost function and additional constraints used for minimization. Tensor factorizations approximate multi-dimensional tensors by a product of a number of matrices, equal to the number of dimensions in the tensor. Throughout this work, tensors are limited to three dimensions. Firstly, the basic concept of matrix/tensor factorization is explained by two well-known factorization algorithms: singular value decomposition (SVD) and independent subspace analysis (ISA). Both algorithms lack the non-negativity constraint. Instead, the factorizations are based upon orthogonality in the case of the SVD or statistical independence in the case of the ISA. After the introduction of these two algorithms, the non-negative factorization is explained in detail.

#### 2.3.1 Matrix Factorization

**Singular Value Decomposition**

The singular value decomposition (SVD) decomposes the matrix $X$ into a product of three matrices:

$$X = USV^T.$$  

(2.30)

The columns of $U$ and $V$ form an orthonormal basis, and the matrix $S$ is a diagonal matrix with the singular values of matrix $X$ in descending order as diagonal elements [16]. By setting certain singular values to zero, the equality of Equation (2.30) becomes an approximation. Setting always the smallest singular values to zero is equivalent to a least squares approximation of $X$ with a matrix $\tilde{X}$ of a given rank. The rank is given by the number of non-zero singular values. Because matrix $S$ is a diagonal matrix, the matrix $\tilde{U} = US$ is simply the matrix $U$ with
the columns being weighted by the diagonal elements of $S$. By this, the output of the SVD simplifies to a factorization into two matrices:

$$X = \tilde{U}V^T.$$  

(2.31)

The SVD is deterministic, which means only one decomposition according to Equation (2.30) exists for each matrix $X$. It is well-defined for the two-dimensional case. For three-dimensional cases an algorithm called higher-order SVD can be applied. Unfortunately, this extension into a three-dimensional (or even higher order) factorization is not trivial, see also [17].

**Independent Subspace Analysis**

The independent component analysis (ICA) is an algorithm to maximize the statistical independence of signals, see also [18]. The ICA cannot be applied to underdetermined BSS scenarios. Therefore, the independent subspace analysis (ISA) is introduced in [19], which can even separate monaural mixtures. The basic concept is to factorize the spectrogram into a product of two matrices:

$$X \approx BG^T.$$  

(2.32)

This factorization is based on the assumption that either the columns of $B$ or the rows of $G^T$ are statistically independent. The particular columns or rows are made as statistically independent as possible by the ICA. For an intuitive and detailed description of the ISA please refer to [20]. The ISA, applied to the spectrogram of a monaural recording, is the basic factorization algorithm of our earlier work, explained in [21].

**2.3.2 Non-Negative Tensor Factorization**

Non-negative tensor factorization (NTF) is an algorithm to approximate a tensor $X$ of size $K \times T \times C$ by a product of three matrices

$$X(k, t, c) \approx \tilde{X}(k, t, c) = \sum_{i=1}^{I} B(k, i)G(t, i)A(c, i),$$  

(2.33)

where $X$, $B$, $G$, and $A$ are element-wise non-negative. The matrix $B$ is of size $K \times I$, $G$ is of size $T \times I$, and $A$ is of size $C \times I$. By this, $K \cdot T \cdot C$ elements of $X$ are approximated by $I \cdot (K + T + C)$ elements in $B$, $G$, and $A$.

With $C = 1$ and $A(c, i) = 1$, the three dimensional approximation simplifies to the non-negative matrix factorization (NMF). Therefore we will use in the following only the term NTF, even in the two-dimensional case, which is nothing else than the standard NMF\(^2\). The approximation can be done by minimizing a distance function between $X$ and $\tilde{X}$, e.g. the $\beta$-divergence shown in Equation (2.5). Although other distance functions are possible, here only the $\beta$-divergence is considered.

\(^2\)It is worth to mention that standard NMF can be implemented very simple by few matrix multiplications per iteration. The implementation of NTF is more complex compared to the two-dimensional scenario of NMF.
To apply convex optimization methods as mentioned in [22], the cost function needs to be convex. In [2], convexity is defined by non-negativity of the second derivative. The second derivative of the $\beta$-divergence is:

$$\frac{\partial^2 d_\beta(x, y)}{\partial y^2} = (\beta - 1) y^{\beta - 2} - (\beta - 2) x y^{\beta - 3} , \quad (2.34)$$

with $x$ and $y$ being non-negative. Equation (2.5) is convex if $(\beta - 1) y > (\beta - 2) x$: For $\beta = 1$ Equation (2.5) is convex for each $y$, $\beta = 0$ implicates convexity for $y < 2x$. The final cost function to minimize is the sum over the cost function between each element of $X$ and $\tilde{X}$ according to Equation (2.5):

$$d_\beta(X, \tilde{X}) = \sum_{k,t,c} d_\beta(X(k,t,c), \tilde{X}(k,t,c)) . \quad (2.35)$$

As mentioned in [23], this cost function is convex for updating a single matrix $B$, $G$, or $A$, but not for two, or even all three of them. Therefore, minimization is done by alternated updating each of these three matrices. If $I = 1$ or $\beta = 2$ the cost function can be minimized in closed form. In the case of $I = 1$ the update rules become

$$B(k) = \frac{\sum_{t,c} X(k,t,c) (G(t)A(c))^{\beta-1}}{\sum_{t,c} (G(t)A(c))^\beta} , \quad (2.36)$$

$$G(t) = \frac{\sum_{k,c} X(k,t,c) (B(k)A(c))^{\beta-1}}{\sum_{k,c} (B(k)A(c))^\beta} , \quad \text{and} \quad (2.37)$$

$$A(c) = \frac{\sum_{k,t} X(k,t,c) (B(k)G(t))^{\beta-1}}{\sum_{k,t} (B(k)G(t))^\beta} . \quad (2.38)$$

In the case of $\beta = 2$, the updates are defined by the following set of linear equations:

$$\sum_{t,c} G(t,i)A(c,i) \left( \sum_{l=1}^{I} G(t,l)A(c,l)B(k,l) \right) = \sum_{t,c} X(k,t,c)G(t,i)A(c,i) , \quad (2.39)$$

$$\sum_{k,c} B(k,i)A(c,i) \left( \sum_{l=1}^{I} G(t,l)A(c,l)B(k,l) \right) = \sum_{k,c} X(k,t,c)B(k,i)A(c,i) , \quad \text{and} \quad (2.40)$$

$$\sum_{k,t} B(k,i)G(t,i) \left( \sum_{l=1}^{I} G(t,l)A(c,l)B(k,l) \right) = \sum_{k,t} X(k,t,c)B(k,i)G(t,i) . \quad (2.41)$$

With $C = 1$, Equations (2.39)-(2.41) can be solved directly with the SVD explained in Section 2.3.1. In general, the solution of this set of linear equations violates the non-negativity constraints, because the direct solution usually needs a matrix inversion. To guarantee non-negativity, a small positive value replaces all negative elements in matrices $A$, $B$, and $G$. This projection back into the space defined by the non-negativity constraint is also used for minimization by the *method of projected gradients* as explained e.g. in
In all other cases, the current estimates of matrices $B$, $G$, and $A$ are updated with the gradients regarding the cost function combined with a certain step-size. For additional information about such numerical optimization methods, see [2]. The gradients are evaluated by

\[
\frac{\partial d_\beta}{\partial B(k, i)} = \sum_{t, c} \xi_2(k, t, c) G(t, i) A(c, i) - \sum_{t, c} \xi_1(k, t, c) G(t, i) A(c, i), \tag{2.42}
\]

\[
\frac{\partial d_\beta}{\partial G(t, i)} = \sum_{k, c} \xi_2(k, t, c) B(k, i) A(c, i) - \sum_{k, c} \xi_1(k, t, c) B(k, i) A(c, i), \tag{2.43}
\]

\[
\frac{\partial d_\beta}{\partial A(c, i)} = \sum_{k, t} \xi_2(k, t, c) B(k, i) G(t, i) - \sum_{k, t} \xi_1(k, t, c) B(k, i) G(t, i), \tag{2.44}
\]

with $\xi_1(k, t, c) = X(k, t, c) \cdot \tilde{X}^{\beta-2}(k, t, c)$ and $\xi_2 = \tilde{X}^{\beta-1}(k, t, c)$. In [4], multiplicative update rules are proposed to ensure non-negativity of elements. For NTF these update rules are:

\[
B(k, i) \leftarrow B(k, i) \frac{\sum_{t, c} \xi_1(k, t, c) G(t, i) A(c, i)}{\sum_{t, c} \xi_2(k, t, c) G(t, i) A(c, i)}, \tag{2.45}
\]

\[
G(t, i) \leftarrow G(t, i) \frac{\sum_{k, c} \xi_1(k, t, c) B(k, i) A(c, i)}{\sum_{k, c} \xi_2(k, t, c) B(k, i) A(c, i)}, \tag{2.46}
\]

\[
A(c, i) \leftarrow A(c, i) \frac{\sum_{k, t} \xi_1(k, t, c) G(t, i) B(k, i)}{\sum_{k, t} \xi_2(k, t, c) G(t, i) B(k, i)}. \tag{2.47}
\]

These multiplicative update rules correspond to a multiplication with the negative parts of the gradients followed by a division by the positive parts of the gradients, as defined in Equations (2.42)-(2.44). In [3], it is mentioned that convergence is proven for $1 \leq \beta \leq 2$, and convergence can be observed in general for $0 \leq \beta < 1$, but no proof is available.

If not otherwise mentioned, the following three steps ensure numerical stability: Firstly, the columns of tensor $X(k, t, c)$ with $Level(t)$ lower than $-60$ dB below the maximum $Level(t)$ are ignored for NTF:

\[
Level(t) = \max \left[10 \log_{10} \left( X^2(k, t, c) \right) \right] \text{[dB]}, \tag{2.48}
\]

where the max $[\cdot]$ operation corresponds to the maximum over the indices $c$ and $k$. After the NTF, the columns of $G$ being ignored are filled with a small constant value. Secondly, each column of $A$, and $G$ is normalized to unit energy. The columns of $B$ are normalized accordingly for not changing the current approximation. Finally, each element of matrices $A$, $B$, and $G$ is lower bounded by a small constant value ($\sim 10^{-16}$).
2.3 Non-Negative Factorization

Sparseness in the Context of NTF

Sparseness usually addresses matrices with most elements being exactly zero, see also [24] for examples of algorithms dealing with sparse matrices. In the context of NTF, sparseness is defined in a more relaxed way: Most of the elements of a sparse matrix are nearly zero and only a small number of elements are significant (has a non-negligible amplitude). For more information about sparseness in the context of NTF, see also [25] or [26].

On the Similarity between SVD and NTF

The SVD and the NTF are identical under the following conditions:

- The NTF is used to approximate a matrix ($C = 1$).
- The Euclidean distance ($\beta = 2$) is used as the cost function of the NTF.
- The number of non-negative components $I$ is set to the rank of matrix $X$.
- The orthogonality of the rows of matrices $\tilde{U}$ and $V$ in Equation (2.31) is given by non-negative elements: Only one element in each row of both matrices is unequal zero.
- The cost function of the NTF is reduced to zero (global minimum) by the iterative approximation.

The conditions above show the similarity between both matrix factorization algorithms. As mentioned in the following thesis, these conditions are not the typical scenario for the NTF: $I$ is usually set to smaller values than the rank of $X$, the cost function is not reduced to zero to avoid a high number of iterations, and the non-negative components to be factorized are usually not orthogonal.

2.3.3 Expectation Maximization for NTF

Alternatively, the expectation maximization (EM) algorithm can be used for minimization of the cost function of Equation (2.5), see also [27] or [3]. The NTF factorizes $I$ meaningful components $Y_i$ out of the matrix $X$. Several possibilities to interpret these factorized components for further signal processing exists, e.g.

$$Y_i(k, t, c) = B(k, i)G(t, i)A(c, i) \ , \text{ or}$$

$$Y_i(k, t, c) = X(k, t, c) \frac{B(k, i)G(t, i)A(c, i)}{\sum_{l=1}^{I}B(k, l)G(t, l)A(c, l)} .$$

Further details for these interpretations of the factorized components will be given in Section 4.1. If the variant shown in Equation (2.50) is chosen, an EM algorithm can be used for minimization of the given cost function. The basic idea is to interpret the signal $Y_i$ as representative data samples for estimation of parameters $B(k, i)$, $G(t, i)$, and $A(c, i)$. In [3], a detailed description of this method can be found.
2.3.4 Estimating the Number of Relevant Non-Negative Components

Obviously, the cost function of the NTF can be reduced by increasing $I$. Firstly, the number $I$ of non-negative components necessary for perfect reconstruction of matrix $X$ will be explained as an upper limit for the choice of $I$. Secondly, an algorithm for blind estimation of the necessary number of components $I$ is discussed.

Necessary $I$ for Perfect Reconstruction

The rank of a matrix $X$ is the number of rows that are linear independent [2]. It can also be interpreted as the number of entries in matrix $S$ greater than zero, as defined in Equation (2.30). The rank is equivalent to the number of components for a matrix factorization under the condition of perfect reconstruction if no additional constraints like non-negativity applies. In contrast to this, the question of the number of non-negative components in $X$ cannot be solved in a simple way. This number is sometimes also interpreted as the non-negative rank of $X$.

Without loss of generality, the following inequality is assumed: $C < T < K$. Perfect reconstruction with non-negative matrices can be achieved for arbitrary tensors $X$ with $I = C \cdot T$ by initializing all elements of $A$ and $G$ with zeros and afterwards setting

$$A(c, i) = 1, \text{ if } (c - 1) \cdot T < i \leq c \cdot T,$$

$$B(k, (c - 1) \cdot T + t) = X(k, t, c), \text{ and}$$

$$G(t, (c - 1) \cdot T + t) = 1.$$  

(2.51) (2.52) (2.53)

Obviously, there exist tensors that can be factorized with perfect reconstruction with a smaller amount of components $I$. On the other hand it is impossible to factorize arbitrary tensors with $I < C \cdot T$, which can be shown by the following simple example: Assuming a tensor of size $2 \times 3 \times 3$ with the two slices

$$c = 1: \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \text{ and}$$

$$c = 2: \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$  

(2.54) (2.55)

For perfect reconstruction, $I = 6$ components are necessary. The first slice needs the following three components for perfect reconstruction

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} = B_1 \cdot G_1$$

$$= \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$  

(2.56) (2.57)
2.3 Non-Negative Factorization

This is the only factorization leading to perfect reconstruction, if the permutation of the columns and the rows of both matrices is ignored. The second slice can be approximated by

\[
\begin{pmatrix}
0 & 1 & 0 \\
0 & 0 & 1 \\
1 & 0 & 0
\end{pmatrix} = B_2 \cdot G_2
\]

Both matrices $G_1$ and $G_2$ are identical for the factorization of both slices. Obviously, it is impossible to use both matrices of one slice-factorization (use either $B_1$ and $G_1$ or $B_2$ and $G_2$) to reconstruct both slices. Therefore, it can be concluded that $I = C \times T = 6$ components are necessary for perfect reconstruction of this tensor. This simple counterexample shows that it is impossible to factorize arbitrary tensors with less than $I = C \times T = 6$ components.

As can be seen in Equations (2.51)–(2.53), the factorization with perfect reconstruction is a simple repetition of single columns (rows, or slices) multiplied with instances of the identity matrix with or without permuted rows or columns. It is obvious that such a factorization is not useful in the context of BSS or more general in the context of semantic analysis of signals. In the following, alternative ways of adjusting the parameter $I$ will be discussed.

**Blind Adjustment of $I$ for Meaningful Factorization**

The decomposition by the SVD is deterministic as mentioned in Section 2.3.1, but the result of the NTF highly depends on the initialization and the number of components $I$ used for decomposition, as stated in [25].

For factorization methods with non-negativity constraints, the authors of [28] propose an automatic relevance determination. They derive the algorithm for the two-dimensional case (NMF) with $\beta = 1$ (Kullback-Leibler divergence). The extension to the three-dimensional factorization NTF is straightforward and is shown here. The basic idea is to model the entries of $B$, $G$, and $A$ with a probability density function (pdf), namely the half-normal distribution, which is defined as the pdf of the absolute values of a Gaussian distributed random process with zero mean and a certain variance $\gamma$:

\[
p_{B_{(k,i)}}(x) = \sqrt{\frac{2\gamma(i)}{\pi}} \exp\left(-\frac{1}{2} \gamma(i) x^2\right) \cdot \varepsilon(x),
\]

\[
p_{G_{(t,i)}}(x) = \sqrt{\frac{2\gamma(i)}{\pi}} \exp\left(-\frac{1}{2} \gamma(i) x^2\right) \cdot \varepsilon(x),
\]

\[
p_{A_{(c,i)}}(x) = \sqrt{\frac{2\gamma(i)}{\pi}} \exp\left(-\frac{1}{2} \gamma(i) x^2\right) \cdot \varepsilon(x).
\]

The half-normal distribution is chosen because of the non-negativity constraints for $A$, $B$, and $G$. Negative values for these three matrices have a probability of zero. This
The inverse variances $\gamma(i)$ of these half normal distributions are assumed to be Gamma distributed:

$$p(\gamma(i)) = \frac{b^a}{\Gamma(a)}\gamma^{a-1}(i)\exp\left(-\gamma(i)b\right). \tag{2.63}$$

The gamma distribution is parameterized by the form parameters $a = 1$, and $b = 25$, as proposed in [28]. The authors of this paper report that the form parameters $a$ and $b$ have little influence throughout their experiments. They motivate this choice of prior distribution by the Gamma distribution being conjugate to the half-normal pdf.

$$B(k,i) \leftarrow B(k,i) \frac{\sum_{t,c} \xi_1(k,t,c)G(t,i)A(c,i)}{\gamma(i)B(k,i) + \sum_{t,c} \xi_2(k,t,c)G(t,i)A(c,i)}, \tag{2.64}$$

$$G(t,i) \leftarrow G(t,i) \frac{\sum_{k,c} \xi_1(k,t,c)B(k,i)A(c,i)}{\gamma(i)G(t,i) + \sum_{k,c} \xi_2(k,t,c)B(k,i)A(c,i)}, \tag{2.65}$$

$$A(c,i) \leftarrow A(c,i) \frac{\sum_{k,t} \xi_1(k,t,c)B(k,i)G(t,i)}{\gamma(i)A(c,i) + \sum_{k,t} \xi_2(k,t,c)B(k,i)G(t,i)}, \tag{2.66}$$

$$\gamma(i) = \frac{K + T + C + 2a - 2}{\sum_k B^2(k,i) + \sum_t G^2(t,i) + \sum_c A^2(c,i) + 2b}, \tag{2.67}$$

with $\xi_{1,2}$ defined in the same way as for Equations (2.45)–(2.47). As reported in [28], large values $\gamma(i)$ result in negligible factorized components $i$. Therefore the number of non-negative components in $X$ can be estimated by the number of small values in the vector $\gamma$.

This algorithm for blind adjustment of parameter $I$ has two disadvantages: Firstly, $I$ has to be set to large values, to capture all non-negative components of the tensor $X$ safely. This results in a relatively high computational complexity. As mentioned in [29], NMF or NTF algorithms are usually the most complex parts during non-negative signal analysis regarding the computational complexity. Therefore, this algorithm will increase the computational complexity of all algorithms based on non-negative analysis.

Secondly, if the tensor $X$ cannot be described by the tensor factorization model in a compact way, this algorithm tends to very large values for $I$. This effect can be seen in Figure 2.5. In the upper row, the detailed view on a given magnitude spectrogram constructed by the STFT is shown. Additionally, the temporal envelopes of the most dominant partials are shown. In the lower row, the left figure shows the approximation by standard NTF with $I = 1$ which corresponds to the standard analysis model of the NTF: Each single meaningful component (each single note) corresponds to a single non-negative component, see also [3]. The right figure shows the maximum value of vector $\gamma$ for different values of $I$. The dotted line corresponds to the maximum possible value of $\gamma$. Components with $\gamma(i)$ near to this maximum value are discarded. Even for very large $I$, the maximum of $\gamma$ converges to roughly 50% of the maximum possible value. The reason for this convergence behaviour can be seen in Figure 2.5(b): The most dominant partials have similar structure like identical onsets and identical position of maximum. Unfortunately, the form of the decay is not identical. Therefore, a factorization by the NTF into a single semantically meaningful component will fail. The convergence behaviour, shown in Figure 2.5(d), is
2.3 Non-Negative Factorization

Figure 2.5: Upper row: Detail view on a spectrogram of a piano note (Figure 2.5(a)) with the temporal envelopes of the most dominant partials (Figure 2.5(b)). Lower row: The approximation with standard NTF ($I = 1$) (Figure 2.5(c)), and the maximum value of vector $\gamma$ for different starting values of $I$ (Figure 2.5(d)). The dotted line in Figure 2.5(d) corresponds to the maximum possible value of $\gamma$ according to [28].

a violation of the assumption in [28] that redundant components $i$ correspond to values of $\gamma(i)$ nearly equal to the maximum possible value. Even by setting the threshold for discarding components to 50 % of the maximum possible value, roughly 4–5 components remain for approximation of a single note, which is still too high for semantical analysis. From this simple example it can be derived that the automatic relevance detection will tend to higher number of components making the following analysis much more complex, see also Chapter 4 for more details.

Motivated by these disadvantages, blind adjustment of the number of non-negative components $I$ is not used throughout this thesis. Instead, constant values for $I$ are used as a rule of thumb.
2.3.5 Initialization of NTF

Once the number of non-negative components $I$ is set to an appropriate value, the $I$ components need to be initialized. Usually, the matrices $B$ and $G$ are initialized with the absolute values of Gaussian noise [30, 31]. Because of the iterative structure of NTF algorithms, the results of the factorization depend on the initialization. Therefore, a deterministic initialization is desirable to avoid the influence of the initialization on the whole signal analysis. In [23], it is suggested to use random initialization and to apply few iterations of NTF. For better approximation and less influence of initialization, this is repeated several times. The best approximation regarding the used cost-function is chosen as initialization for final NTF. Because of the additional computational complexity, we propose other approaches for initialization. In principle two ideas for initialization exist: Semantic based initialization and data-driven initialization. For both ideas, algorithms will be introduced.

Semantic Based Initialization

If a priori knowledge about the signal that have to be analysed is available, it is possible to initialize the factorization accordingly. In the case of music analysis, a dictionary of notes can be used to initialize the matrix $B$. In [32], the magnitude spectra of the 88 notes corresponding to the 88 keys of the piano are used as initial matrix $B$ in the context of automatic music transcription. The initialization starts with a pitch of $f_0 = 27.5$ Hz. For each note, the pitch is increased by a semitone:

$$f_0 \leftarrow f_0 \cdot 2^{\frac{1}{12}} .$$

(2.68)

A time domain signal is constructed out of the first 20 harmonic oscillations corresponding to the current $f_0$. For the amplitudes of the harmonic oscillations, a logarithmic attenuation of 3 dB/Octave is assumed. After applying the analysis window, the signal is transformed into frequency domain by the DFT. In the case of logarithmic frequency scaling, the matrix $B$ is multiplied with the corresponding mel filter bank $R$, see also Section 2.4.1 for further details about logarithmic scaling of the frequency-axis. A possible initialization for matrix $G$ is

$$G(t, i) = \frac{\sum_k B(k, i) \sum_c X(k, t, c)}{\sum_k B^2(k, i)} .$$

(2.69)

This corresponds to the assumption that only one entry of $G(t, i)$ is greater than zero, for $1 \leq i \leq I$. In this case Equation (2.69) gives the optimal entries for matrix $G$. Therefore this initialization of $G$ is somehow overcomplete, because for a purely additive model induced by the non-negativity of all components, the entries are all too large. According to the sum over channel parameter $c$ in Equation (2.69), matrix entries $A(c, i)$ are set to one.

In general, it is not useful to separate $I = 88$ components out of the mixture signal. On the other hand, for selecting certain components out of the initialized matrix $B$ before the NTF starts, a pitch detection is necessary to decide which components are active in the given mixture. This pitch detection may fail, of course. Therefore, a simpler solution
is proposed. If the number of columns of matrix $B$ is greater than $I$, the component with lowest energy is discarded after each iteration of NTF. By this, the algorithm slowly approximates the correct number of components $I$. As a matter of fact, this is a similar method to that proposed in [28]: Each column of $B$ and $G$ is modeled by a distribution with a given variance. The variance parameters are updated in each iteration. After the NTF converges, the components with lowest variance are discarded.

**Data-Driven Initialization**

In the following, two different data-driven initialization schemes are introduced.

**SVD Based Initialization** In [33], a deterministic initialization based on the SVD is introduced. The pseudo code is shown in Algorithm 1. As mentioned in Section 2.3.1, the extension of the SVD to a three dimensional analysis is not trivial. One possible solution is to concatenate the slices of tensor $X$ and to apply the SVD on each of the three resulting concatenated versions, keeping only matrix $U$ as result for the orthogonal analysis of the three dimensions. Instead of this, the sum over the third dimension is used to transform the tensor $X$ into a matrix for SVD-based initialization throughout this work. According to the summation, matrix $A$ is initialized with ones.

**Algorithm 1 SVD Based Initialization of NTF**

\[
X_m(k, t, c) = \sum_{c=1}^{C} X(k, t, c)
\]

\[
[U, S, V] = \text{svd}(X_m) \text{ according to section 2.3.1}
\]

$\epsilon$ being a small, positive value

for $i = 1$ to $I$

$U_p(k, i) = \max(U(k, i), 0)$, $V_p(t, i) = \max(V(t, i), 0)$

$U_m(k, i) = U_p(k, i) - U(k, i)$, $V_m(t, i) = V_p(t, i) - V(t, i)$

$E_{p,1} = \sqrt{\sum_{k=1}^{K} U_p(k, i)^2}$ and $E_{p,2} = \sqrt{\sum_{t=1}^{T} V_p(t, i)^2}$

$E_{m,1} = \sqrt{\sum_{k=1}^{K} U_m(k, i)^2}$ and $E_{m,2} = \sqrt{\sum_{t=1}^{T} V_m(t, i)^2}$

if $E_{p,1} \cdot E_{p,2} > E_{m,1} \cdot E_{m,2}$ then

$B(k, i) = U_p(k, i) \sqrt{S(i, i)} E_{p,2}$, and $G(t, i) = V_p(t, i) \sqrt{S(i, i)} E_{p,1}$, for all $k$, and $t$

else

$B(k, i) = U_m(k, i) \sqrt{S(i, i)} E_{m,2}$, and $G(t, i) = V_m(t, i) \sqrt{S(i, i)} E_{m,1}$, for all $k$, and $t$

end if

end for

**Iterative Initialization** Another possible initialization works iteratively: The number of active components is set to $I = 1$. The elements of the matrices $B$, $G$, and $A$ are all set to one. Then one iteration of update rules is applied according to Equations (2.45) – (2.47) to adjust the current estimation to the given data matrix $X$. If $I$ is large enough, the algorithm stops. Otherwise it repeats to increment $I$, extending the matrices $B$, $G$, and $A$ with additional columns filled by ones, and performs one single iteration of multiplicative update rules as described in Equations (2.45) – (2.47).
2.3.6 Statistical Motivation of Cost Functions

The cost function in Equation (2.5) can be derived by maximizing the log-likelihood for a given probability density function of the approximation error of the NTF as explained in [28] and [25]. For example, assuming a Poisson distribution $p\left( X(k,t,c) \mid \bar{X}(k,t,c) \right)$ for a given matrix value $X(k,t,c)$ and a given estimated approximation $\bar{X}(k,t,c)$:

$$p(x|\lambda) = e^{-\lambda} \frac{\lambda^x}{\Gamma(x+1)} ,$$  

with $\Gamma$ being the Gamma-function [2]. The log-likelihood for all elements can be written as

$$\log \Pi_{k,t,c} p\left( X(k,t,c) \mid \bar{X}(k,t,c) \right) = \sum_{k,t,c} -\bar{X}(k,t,c) + X(k,t,c) \log \bar{X}(k,t,c) + \text{const} ,$$  

with $\text{const}$ being a term constant with respect to $\bar{X}$. Maximizing the log-likelihood of Equation (2.71) is the same as minimizing the divergence in Equation (2.5) with $\beta = 1$ (Kullback-Leibler divergence).

For $\beta = 2$, a Gaussian distribution is assumed instead of the Poisson distribution, see also [25]. For $\beta = 0$, a statistical motivation is given in [3].

2.4 Psychoacoustics

BSS frameworks can be developed in order to separate a wide range of sources beside human speech and/or musical instruments, e.g. as a pre-processing step for acoustical machine diagnostics. In this case the usage of psychoacoustics does not make sense. On the other side, the focus of this thesis is laid on human speech and musical instruments. Obviously, the human ear is perfectly adopted to the human speech. Additionally, the human skills in source separation are far beyond all current BSS algorithms developed so far. Motivated by this, a closer look is taken on the way humans react on acoustics. Of course, it is not the intention of this section to fully explain the topic of psychoacoustics. Instead, a short overview over relevant aspects of psychoacoustics is given.

2.4.1 Logarithmic Behaviour of Human Perception

Human auditive perception works approximately in a logarithmic manner regarding the frequency and the amplitude. Some popular examples are the mel scale as a logarithmic scaling of the frequency axis (see also [34, p.52]) or the decibel measure (see also [35, p.11]), which is an logarithmic amplitude scaling. The mel scale maps the linear frequency resolution of the Hertz scale onto an approximation of the logarithmic scale based on human perception. The mel scale was first derived by subjective perception tests. In these first experiments, the listeners have to adjust increments in frequency that sounds equally
2.4 Psychoacoustics

Figure 2.6: Weighting functions for mel scale filtering.

spaced to them. The unit *mel* is used for this frequency-scale, which is an abbreviation of melody. Later, this logarithmic behaviour is approximated by this equation:

\[ f_{\text{mel}} = 2595 \log_{10} \left( \frac{f_{\text{Hertz}}}{700} + 1 \right). \]  

(2.72)

Due to the logarithmic frequency resolution a large number of frequency coefficients are concentrated into the output of a single filter in the case of high frequencies, see also Figure 2.6. Beside the psychoacoustical motivation for the logarithmic frequency resolution it can be argued that for higher frequencies typical acoustical signals contain more noise, as will be explained in Section 4.2.2. The influence of the higher amount of noise is reduced by this raw frequency resolution for higher frequencies.

The filters of a mel filter bank are triangular filters with mid-frequencies equally spaced in mel scale, as shown in Picture 2.6. As explained in [14], filtering a magnitude spectrogram \( X \) with a filter bank for logarithmic scaling of frequency axis is the same as a matrix multiplication with a matrix \( R \). \( R \) is of size \( N_{\text{mel}} \times K \). Each row of \( R \) contains one single triangular filter shown in Picture 2.6. The main purpose of \( R \) is to summarize the higher frequencies to single filter bank outputs, motivated by the logarithmic frequency resolution of human listeners. For low frequencies and certain values of \( N_{\text{mel}} \) and \( K \), the content of a single frequency bin of \( X \) can be spread over several filter outputs. This behaviour is usually undesirable. To avoid this frequency spreading for low frequencies, the following modification is applied on the columns of \( R \):

\[
R(n_{\text{mel}}, n_{\text{mel}}) \leftarrow \sum_{n=n_{\text{mel}}}^{N_{\text{mel}}} R(n, n_{\text{mel}})
\]

(2.73)

\[
R(n, n_{\text{mel}}) \leftarrow 0, \text{ for } n_{\text{mel}} < n \leq N_{\text{mel}}.
\]

(2.74)

In general, the reconstruction of spectrogram \( X \) after multiplication with \( R \) is not trivial, because \( R \) is not a square matrix. One possible workaround is to use the pseudo-inverse for reconstruction, as mentioned in [11]. Another idea is to interpret the coefficients in mel domain as excitation signals for basis functions. These basis functions have the same transfer functions as the mel filter used in \( R \). With this assumption, we can multiply the spectrogram in mel domain with the transpose of \( R \) to get a reconstruction in Hertz domain, as suggested in [14]. Mel filters with higher mid frequencies summarize more frequency bins. Therefore, a constant spectrum (e.g. white noise) will not be constant after applying the mel filter bank. To compensate this effect, the mel filter bank \( R \) is
Figure 2.7: Behaviour of matrix $\mathbf{R}$ with (left) and without (right) frequency spreading for lower frequencies.

normalized to $\mathbf{R}_{\text{norm}}$ such that

$$\mathbf{R}_{\text{norm}}^T \mathbf{R}_{\text{norm}} \mathbf{1} = \text{const} \cdot \mathbf{1},$$

(2.75)

with $\mathbf{1}$ being a $K \times 1$ vector filled with ones.

### 2.4.2 Spectral Range of Human Listeners

In [36], a BSS algorithm is applied only on the frequencies that are hearable to humans. This is done in order to reduce the complexity of the permutation and scaling problem of the frequency based independent component analysis algorithms. Beside the reduction of computational complexity, the possible high signal energy at frequencies inaudible to humans may have major impact on the separation quality, which is undesirable.

In the following, the influence of these inaudible frequencies is shown. In Figure 2.8, a single piano note is plotted. In Figure 2.8(b), the decay of this note is shown with more details. For the decay, it can be seen that the wave is oscillating around a mean value of roughly 0.5. Thus, the wave-signal is not zero-mean resulting in a DFT coefficient $|\hat{X}(0)| \gg 0$. This coefficient holds a large amount of signal energy, as can be seen by the following evaluation. The sampled version of the piano note is $x(n)$, $y(n) = x(n) - \frac{1}{N} \sum_{n=1}^{N} x(n)$ is the same signal with zero mean. For the example plotted above, the distance between both signals can be expressed by the SNR:

$$\text{SNR} = 10 \log_{10} \frac{\sum_{n=1}^{N} x(n)^2}{\sum_{n=1}^{N} (x(n) - y(n))^2} = 22.77 \text{ dB}.$$  

(2.76)

The non-zero mean is definitely inaudible to humans but the removal of this offset leads to distortions of 22.77 dB, which cannot be ignored.

In [5], a frequency range of 16 Hz up to 16 kHz is mentioned for the human ear. Therefore, it is suggested to use a high-pass for each signal, to get rid of the spectral components below 16 Hz.

The spectral components above 16 kHz can be erased because of the same arguments:
2.4 Psychoacoustics

Figure 2.8: The time domain signal of a single piano note is shown (left). On the right, the later part of this piano note is shown in detail, to show the offset of the oscillations.

In general, the spectral components above 16 kHz have lower amplitudes, and therefore the signal parts above this limit can be ignored. Other masking effects, like the absolute threshold of hearing or temporal- and frequency-based masking are not applied. For further details about these masking effects and the threshold of hearing, we refer to e.g. [35] or [37]. Especially masking effects have to be ignored for audio analysis: Signal parts not hearable in the mixture can become hearable after BSS, because these signal parts were masked by sources, which are separated out.

2.4.3 Quality Measures

Simple objective quality measures like the SER [30] in magnitude spectrogram domain or the SNR in time domain

\[
\text{SER}_m = 10 \log_{10} \frac{\sum_{k,t,c} S_m^2(k, t, c)}{\sum_{k,t,c} (S_m(k, t, c) - \tilde{S}_m(k, t, c))^2} \quad \text{[dB]},
\]

\[
\text{SNR}_m = 10 \log_{10} \frac{\sum_{n,c} s_m^2(n, c)}{\sum_{n,c} (s_m(n, c) - \tilde{s}_m(n, c))^2} \quad \text{[dB]},
\]

badly reflect human perception of errors [38]. One problem of such global measures is the normalization to the energy of the whole signal. Therefore, time periods with low energy (silent parts for music) have little influence on these measures. To circumvent this problem, [39] suggest to use a segmental SNR evaluated in time domain:

\[
\text{SNR}_\text{local} = 10 \log_{10} \frac{\sum_{n=n_0}^{n_1} s_m^2(n, c)}{\sum_{n=n_0}^{n_1} (s_m(n, c) - \tilde{s}_m(n, c))^2} \quad \text{[dB]},
\]

\[
n_0 \text{ and } n_1 \text{ are the time indices marking the beginning and the end of the local segments. The final objective measure } \text{SNR}_\text{seg} \text{ is the mean value over all SNR}_\text{local} \text{ and all channels.}
\]
evaluated for this signal \( s(n, c) \). Because very low and very large values of \( \text{SNR}_{\text{local}} \) have major influence on the \( \text{SNR}_{\text{seg}} \), the \( \text{SNR}_{\text{local}} \) are clipped to the range \([0, 35]\) dB. These values are motivated by the fact that improvements above a certain threshold (here 35 dB) are not hearable for human listeners. Analogously, the lower threshold induces that quality differences below a given threshold (here 0 dB) are also not hearable for human listeners, because the artifacts are all equally annoying. The given values are suggested by [39]. The \( \text{SNR}_{\text{local}} \) are evaluated on non-overlapping intervals of 20 milliseconds length. In principle, such a local measure is also possible for spectrograms. Unfortunately, spectrograms have the inherent disadvantage of depending on a certain time-frequency resolution, as stated in [40]. To become independent from the time-frequency resolution of the spectrogram, the \( \text{SNR}_{\text{seg}} \) in time domain is preferred here.

Another disadvantage of evaluating the separation quality based on the magnitude spectrogram is the minimization of the cost-function of the NTF. In the case of \( \beta = 2 \), the NTF minimizes the denominator of Equation (2.77). It is reasonable that the SER favors NTF-algorithms minimizing the Euclidean distance (\( \beta = 2 \)). To reduce the influence of this preference the time domain signal shall be used for evaluating the separation quality. Another approach is explained in [41] and [42]. The basic idea is to separate the error signal of a BSS algorithm in parts based on interferences from other sources due to wrong separation, parts based on artifacts introduced by the BSS algorithm, and parts based on noise. Based on this idea, the authors of [42] introduce different quality measures: Source-to-distortion ratio (SDR), source-to-interferences ratio (SIR), source-to-noise ratio (SNR), and source-to-artifact ratio (SAR). In the case of stereo signals, the spatial distortion is measured by source-image-to-spatial-distortion ratio (ISR), as proposed in [42]. These distortion measures can be evaluated by a freely available Matlab toolbox [43]. One disadvantage of this approach is the linear mapping on interferences, artifacts and noise. Linearity is not guaranteed for errors introduced by arbitrary BSS algorithms. Another disadvantage is the relatively high dimension of the quality measure space: Up to four quality measures have to be regarded for comparison of different separation results. On the other hand, the main advantage of these quality measures is the wide acceptance in many publications regarding BSS. By this, the BSS algorithms become comparable.

The basic idea explained in [38] is to learn a linear combination of objective measures that resembles the human perception of similarity in a BSS scenario. The result is a linear combination of the above mentioned measures ISR, SIR, and SAR. Additionally a fourth measure called maximum probability of detection after low-pass filter (MPD, explained in [44]) is used. The weightings for linear combination of these measures are optimized to fit the mean opinion score regarding stereo BSS scenarios. The ISR (measure for spatial error) is most important. Thus, this measurement cannot be applied on monaural scenarios, and is therefore discarded in the following.

For comparison of encoding quality of audio codecs, the ITU suggests the usage of a measure called PEAQ. This measure basically combines a set of formerly used quality measures, each one approximating the human perception of audio quality. Each of these measures utilizes different aspects of human auditory system, e.g. masking or logarithmic frequency resolution. These basic measures are combined by an artificial neural network (ANN) to reduce the output to a single scalar resembling the mean opinion score of human listeners. Freely available implementations exists, e.g in Matlab scripting language. The main field of application for PEAQ is audio encoding with a small amount of distortion.
It can be observed that a large amount of distortion (as present in the case of BSS) leads to constant PEAQ values at the lower edge of the mean opinion score. This makes the PEAQ algorithm not well suited for evaluating the separation quality of typical BSS algorithms, and will not be used in the following.

As a conclusion, the following compromise of different quality measures for different purposes is chosen: The SNR$_{seg}$ is preferred over the measures SDR, SIR, SAR and ISR because it is only a scalar measure which simplifies the comparison of different BSS algorithms. To avoid side-effects of the temporal resolution of 20 milliseconds, additionally the SNR is evaluated. The measures SDR, SIR, SAR, and ISR are used only for comparison with other BSS frameworks.

### 2.5 Audio Data

The question for the optimal data base for scientific simulations is not trivial. For stable training and reliable results, the data shall be as diversified as possible. For reproducible research the whole data shall be freely available. Both postulates seem hard to be fulfilled simultaneously for audio separation. Most audio files are not freely available. Commercial music is usually a mixture of several instruments, e.g. songs played by a whole band. If the single instrument tracks are available the corresponding keyword is *multitrack recording*. At [45], several links to multitrack recordings are provided.

One possibility to produce a large audio data base is the usage of sample based synthesizers, as done in [25]. For most of these synthesizer based data only the process of rendering is described, but the data is not freely available, e.g. in the internet. This is mainly due to the large data size. Another disadvantage for such data is that usually the different instruments do not play in harmony, e.g. a common melody.

The Signal Separation Evaluation Campaign (SiSEC) [46] provides a handful of tasks for audio source separation:

- Underdetermined - speech and music mixtures,
- determined and overdetermined speech and music mixtures,
- professionally produced music recordings, and
- source separation in the presence of real-world background noise.

The most interesting case of these mixtures is the professional produced music scenario, because it assumes a CD recording (a stereo signal sampled with $F_s = 44100$ Hertz) of typical popular instrument combinations, e.g. drums, vocals, and guitar. An older source for audio files for blind source separation is the website *Blind Audio Source Separation evaluation database* (BASS-db) [45]. One common disadvantage of the multitrack recordings mentioned above and the evaluation data of [46] and [45] is the small selection of different instruments: human voices, guitar, drums, bass and keyboard are mostly used. To gather also some classical piece of music we suggest to use the web sources used in [47]. For this evaluation campaign a piece of music from L. van Beethoven’s Variations for String Quartet Op.18 No. 5. is used for fundamental frequency estimation.

Recently, FitzGerald has published the data for his experiments in [48]. It consists of 25 mixtures with $M = 2$ (two different instruments for each mixture).
As a final suggestion, we propose to use the sound quality assessment material from the European Broadcasting Union (EBU). This data base consists of melodies, rhythm, and/or arpeggios of a large range of instruments, even classical ones. Additionally, human sources, like speech and singing are present. The disadvantage of the SQAM data base is that most instruments play different melodies. Therefore harmonic and temporal correlations may be misleading for experiments.

As a conclusion, the following audio data is suggested:

- **Data $A$:** For training of parameters we use a set of 60 signals. They are collected from the SQAM, and from the BASS-dB. Additionally certain signals of the sample based synthesizer GarageBand from Apple are included. With $M = 2$, these 60 signals can be combined to 1770 mixtures of roughly 5 to 20 s length. If not otherwise mentioned the input signals are cut to the length of shortest input signal and the energy of both is normalized. For a detailed description of data $A$, please take a closer look on Appendix A.

- **Data $B$:** The professional produced music recordings from the SiSEC data are the most realistic BSS scenarios: Five stereo recordings with 2-5 sources playing popular music in harmony. Throughout this thesis, the source separation task according to the SiSEC 2010 setting is used for data $B$: It is not necessary to separate all active sources but only a subset of them. The remaining sources are clustered into a background source which is not considered for evaluation of separation quality. For further details which mixture and which sources are used for evaluation of data $B$ please take a closer look on [46] or on Table 6.14 later in this thesis.

- **Data $C$:** The harmonically related sources of the piece of Beethoven are used as audio data, see also [47]. These five instruments are special for two reasons: Firstly, classical instruments are used. Secondly, the mixtures are extraordinary long (roughly 54 seconds). Here, all possible combinations of the five instruments with $M \leq 3$ are used. This results in a total of 20 mixtures.

- **Data $D$:** In [48], a set of 25 mixtures of harmonic instruments without any noisy-, human- or percussive-sources is used for evaluation. The number of active sources is set to $M = 2$ for the whole database $D$.

A final note regarding the strict separation between data used for training and for testing shall be given here: Training a classifier presumes a strict separation between data used for training and data used for evaluation, e.g. in [49], or in [50]. On the other hand, a BSS framework without explicit training is usually applied on a single set of mixtures, without grouping into training and test data. Such experiments are shown e.g. in [27], [48], or [51]. The main reasons for dropping the separation into training and test data are:

- BSS frameworks are usually very sensitive regarding the parameter-settings, e.g. in [27] different analysis lengths for the STFT are used for different sets of audio data.

- BSS frameworks are usually specialized on a subset of all possible audio separation tasks, e.g. the separation of harmonic and percussive components as in [52].

Therefore, the goal of this thesis is to find a BSS framework that can be applied to a large range of audio data with a common set of parameters. Data $A$ has the highest number
of mixtures (1770). The large number of mixtures is the motivation to optimize the BSS framework for data \(A\) and to verify it for data \(B-D\).

### 2.5.1 Mixing Model

In the context of BSS, two different effects are addressed by *mixing model*. Firstly, the mixing model of the single channels of the mixture is explained. Secondly, different mixing models are introduced to derive a stereo- or even multichannel-signal out of the monaural input sources.

#### Single-Channel Mixing Model

Throughout this work the mixture \(x(n, c)\) is defined as a sum over sources to be separated:

\[
x(n, c) = \sum_{m=1}^{M} s_m(n, c).
\]  

\(s_m(n, c)\) is the sample of the input signal \(s_m\) corresponding to source \(m\) at time \(n/F_s\) and sensor \(c\). The real-valued addition in time domain is a complex one in Fourier domain:

\[
X(k, t, c) = \sum_{m=1}^{M} S_m(k, t, c).
\]  

In order to reverse this addition by the additive mixing model of the NTF, an additive mixing model for non-negative values has to be defined. If all sources have equal phase, Equation (2.81) becomes:

\[
X(k, t, c) = \sum_{m=1}^{M} S_m(k, t, c).
\]  

This additive model is often used in the context of BSS, see also [30] or [31]. Of course, the addition with equal phase is not a realistic scenario. The phases of the different sources are usually assumed to be independent and uniformly distributed. In [25], for such a case a squared approximation is suggested:

\[
X^2(k, t, c) \approx \sum_{m=1}^{M} S_m^2(k, t, c).
\]  

Another motivation for such a squared addition is that the values \(S_m(k, t, c)\) are statistically independent. Under this condition, the expectation values of the variances of the single signals are simply added up to form the expectation value of the variance of the mixture\(^4\). The squared amplitudes result in a higher dynamic range of the input values for the NTF. Therefore, it can be assumed that the correct model order (linear, squared, or even something between) has major influence on the parameter \(\beta\) of the \(\beta\)-divergence,

---

\(^4\)If the variances in Equation (2.83) are replaced by the expectation values of the variances, the approximation becomes an equality.
see also Equation (2.5): Generally spoken, $\beta = 2$ is more sensitive to higher amplitudes in $X$ compared to $\beta = 1$, as stated in [25]. On the other hand, $\beta = 0$ is scale invariant, but not convex over the whole range of input values. Therefore, in Chapter 4 we will show simulations regarding the interaction between the model order and the parameter $\beta$.

Multi-Channel Mixing Model

Usually two different multi-channel mixing models are utilized for BSS:

- Instantaneous mixing models, e.g. in [14] or [53], or
- convolutive mixing models, e.g. in [27] or [54].

The convolutive mixing model for a point source can be defined as in [27]:

$$x(n, c) = \sum_m a_c(n) \ast s_{\text{mono}, m}(n).$$

$a_c(n)$ is the impulse response of the transfer path regarding the current recording situation: Recording room and corresponding room impulse response, microphone- and source-position. Additionally, it is also possible to use $a_c(n)$ to approximate artificial sound effects like reverberation or artificial delay. With $a_c(n) = \text{constant}$ the convolutive mixing model simplifies to the instantaneous mixing model. For this model, only level differences for different input signals and sensors are analysed.

The underlying monaural signals $s_{\text{mono}, m}(n)$ can only be estimated up to a constant scaling factor if the BSS scenario is underdetermined (less sensors than sources: $M > C$). Therefore, only the spatial images $s_m(n)$ of the sources defined by

$$x(n, c) = \sum_{m=1}^M s_m(n, c) = \sum_{m=1}^M a_c(n) \ast s_{\text{mono}, m}(n)$$

have to be estimated in the following.

Of course, the convolutive mixing model is more flexible compared to the instantaneous mixing model. One disadvantage of the convolutive mixing model is the higher amount of parameters that have to be fixed by constraints. An additional constrain can be the number of transfer-paths: The number of possible transfer paths is restricted to the number of sources $M$ multiplied with the number of channels $C$, which is much smaller than one transfer path per separated component $i$ and per channel of the mixture $c$. On the other hand, the basic version of the NTF assumes an instantaneous mixing model and in Section 4.3, it is shown that this basic version is sufficient for the BSS framework used in this thesis.
Chapter 3

Blind Source Separation: State-Of-The-Art

There exist already a large number of algorithms for BSS. In this chapter, an overview over the state-of-the-art will be given. The (dis-)advantages of these approaches are made clear, and open problems regarding the note-clustering are pointed out.

Most current approaches for BSS are based on the factorization of spectrograms, e.g. by the ISA or the NTF. All of these approaches suffer the same drawback that the separated components are equivalent to single notes not melodies. Firstly, in Sections 3.1, 3.2, and 3.3 current solutions for this problem are introduced in the context of spectrogram factorization. Finally, in Section 3.4 BSS algorithms beside spectrogram factorization are explained.

3.1 Spectrogram Factorization with Supervised Clustering Methods

Supervised clustering methods are divided into non-blind clustering by knowledge of the input signals and trained models utilizing a restriction to a given number of different sources. For each possible source a model is trained to allow the distinction between the different instruments.

3.1.1 Non-Blind Clustering

The author of [30] proposes to cluster the separated notes with the knowledge of the input signals. Although this is not a blind separation and therefore not applicable in a BSS scenario it is a useful idea to get first insights in the behaviour of the factorization algorithm itself.

In [55], two solutions are proposed for the clustering problem: directed NMF and undirected NMF. For the un-directed NMF the separated notes are clustered by user-interaction\(^1\).

In Chapter 4, BSS with standard NTF and supervised note clustering is explained in

\(^1\)The directed NF is explained in Section 3.1.2.
Figure 3.1: Signal flow of the BSS algorithms based on spectrogram factorization. The necessary time-frequency transform for spectrogram generation and the inverse transform for signal synthesis is dropped for simpler visualization. The indication of the corresponding sections refers to the sections where the corresponding signal flow is described in detail.

detail together with first experiments regarding an appropriate parameter selection. This BSS approach is picked out and explained by first experiments because it is the starting point for the BSS framework proposed in this thesis.

3.1.2 Trained Models

A first obvious idea to cluster the separated sound events is to train some expert system in a supervised way. After that, the trained expert clusters the separated notes blindly. Although the final separation is blindly, this algorithm is called *supervised* because of the needed supervised training step. By this, the algorithm is restricted to instruments represented by the training data. Of course, it is possible to apply a trained classifier to mixtures for which it is not trained. In this case, no guarantee can be given that the separation succeeds or fails.

The *directed* NMF introduced in [55] assumes the availability of training data for each source $m$. An NTF is applied to this set of training data in order to learn a set of basis spectra $B_m$ for each source. After that, the trained basis spectra are concatenated to form the fixed basis for the second NTF. During this second NTF, only the gains $G$ and in the multichannel-case the channel-amplitudes $A$ are adapted to minimize the given cost-function.

The authors of [56] extend this idea: It is shown that it is advantageous to learn the covariances of the activation matrix $G$ for the training data in order to further increase the separation quality.

In [50], a support vector machine is trained to classify either percussive or pitched sounds. The input for the SVM is a set of spectral and temporal based features, e.g. MFCC, spectral flatness, or kurtosis.
In [57], the spectral covariances for male and female speakers are measured. These covariance matrices are modeled by a Gaussian mixture model. This modeling is embedded in the cost function of the NTF to prefer columns of $B$ according to the trained covariance matrices. By this, a set of $I_1$ components corresponding to the female covariance matrix and a set of $I_2$ components corresponding to the covariance matrix of the male speakers are factorized. The corresponding covariance matrix can be interpreted as the clustering information.

One advantage common to all BSS frameworks based on learned models is the fact that after separation each stream can be assigned to a class of instruments according to the detected model. As a disadvantage, the instruments active in the current mixture have to be detected blindly:

Without this instrument classification, it is not clear which trained model has to be used for clustering. This instrument classification has to be applied to the mixture with more than one active instrument. Obviously, instrument classification is harder to solve, if more than a single instrument is active. On the other hand, the interferences from the other instruments are reduced by the BSS regarding the current separated stream. Therefore, it is reasonable that it is simpler to apply the instrument classification to the separated streams instead of the whole mixture.

### 3.2 Spectrogram Factorization with Unsupervised Clustering Methods

To the best of our knowledge, three proposals for unsupervised clustering of acoustical events exist:

In [19], the sources are separated by the ISA, as described in Section 2.3.1. After separation, a distance matrix is evaluated, filled with pairwise distances between separated acoustical events. The symmetric Kullback-Leibler divergence between the histograms of the frequency basis vectors $B$ is used as distance. The clustering for this distance matrix is defined by minimization of the within-cluster compactness and between-cluster homogeneity. The performance of the algorithm is shown for a single mixture: speech and water noise. Such a small set of audio data is generally a problem, because the parameters of the algorithm can be optimized individually for a small set of mixtures to get better separation results.

In [51], a clustering based on the shift-invariant spectra in logarithmic frequency domain is introduced. Shift-invariance of logarithmic spectra is explained in Section 2.2.2. Two disadvantages for this approach exist. Firstly, an inverse transform of the used constant-Q transform is missing. The separation quality is evaluated in log-frequency domain. Second, the clustering algorithm works only for pitched instruments.

In [58], a similar clustering approach is done: A standard NMF is applied to the spectrogram and the resulting frequency-basis vectors stored in matrix $B$ are interpreted as a second spectrogram. This second spectrogram is transformed into logarithmic frequency resolution by a constant-Q transform. After that, the shift-invariance of water noise can be approximated by pink noise, which is also content of audio data $A$. Therefore, this single mixture is also considered in our experiments.
harmonic notes is utilized for clustering: The second spectrogram is factorized by a shift-invariant factorization algorithm as explained in detail in [59]. Similar to [31], the activation tensors (matrices) are interpreted as the clustering decision. These unsupervised note clustering algorithms are based on the fact that each separated component of the factorization algorithm is a single note. In [21] and [60], clustering algorithms are proposed for scenarios with single notes being factorized into different components. As a sparse NTF usually clusters the single frequency-bins of each note together, these clustering algorithms can be interpreted as a pre-processing step applied after NTF and before the final note clustering. In this thesis, it is assumed that the NTF factorizes entire notes. Therefore, a clustering algorithm that forms a single note out of separated harmonic overtones is beyond the scope of this thesis.

### 3.3 Spectrogram Factorization embedding Clustering

As mentioned in [30], one wrong classification in the clustering step can deteriorate the separation quality significantly. To avoid this clustering, it is possible to embed the clustering into the factorization step by extending the underlying note-model of the factorization algorithm.

In [14], extended tensor models are introduced that implement a factorization specialized on either harmonic or percussive instruments. The performance of this algorithm is shown for a test set of 40 mixtures. One major disadvantage of this algorithm is the high computational complexity. Although the authors of [14] did not mention the runtime of their algorithm, the higher computational load can be estimated by the applied factorization methods: The best extended factorization model proposed in [14] (Source-Filter Modeling with Separation of Pitched and Non-pitched Instruments) needs to update seven tensors for each iteration. Each tensor-update involves at least eight tensors multiplied over at least one dimension. Standard NTF as introduced in Section 2.3.2 needs only to update three tensors for each iteration. Each tensor update is done by only four tensor products. This very raw approximation shows that the extended NTF models proposed in [14] are at least four times slower compared to the standard NTF. As mentioned in [29], the computational load of current BSS algorithms is dominated by the time consumption of the factorization algorithm. Therefore, it can be concluded that the extended factorization methods proposed in [14] are very slow compared to approaches with a stand-alone algorithm used for clustering the separated notes.

In [27], a multichannel separation approach is proposed, which utilizes the transfer path between the point sources and the sensors. This transfer path is approximated by a convolution. This convolution is described by a multiplication of the spectrogram columns by a filter corresponding to the current transfer path. As a matter of fact, this approximation by a multiplication is only true for scenarios with short convolution filter. This approximation results in the following NTF model:

\[
X(k, t, c) \approx \sum_{i=1}^{I} A(k, c, i)B(k, i)G(t, i) . \tag{3.1}
\]
3.4 Source Separation Frameworks beside Spectrogram Factorization

The matrix $A(k, c, i)$ encodes the transfer paths from point source $m$ to sensor $c$. The transfer paths are assumed to be sparse: $A(k, c, i) \neq 0$ if and only if component $i$ belongs to source $m$. Thus, a clustering is assumed to be known for this kind of factorization. In [27] different ways of getting this information are suggested: Supervised and unsupervised ones. For the supervised version, the clustering is done manually by user interaction. For the unsupervised version, a preprocessing step evaluates first estimates for all possible transfer paths. Then, these filters are clustered into $M$ different filters for each $c$, thus leading to the sparse structure, necessary for deconvolution and source estimation. Additionally, the authors of [27] introduce a statistical framework plus noise model for evaluating all necessary parameters by the expectation maximization algorithm. The algorithm introduced in [27] is used in the SiSEC 2010 for all audio separation tasks. To make the main disadvantages of this algorithm clear, the professional produced mixture (data $B$) is regarded. The computational complexity is very large (above one hour for a 2.2 GHz CPU). Additionally, user interaction of roughly half an hour is necessary. One additional disadvantage is the restriction that this factorization method is unable to separate monaural mixtures. The main advantage is the ability to deal with convolutive mixtures instead of restricting the algorithm to instantaneous mixtures.

3.4 Source Separation Frameworks beside Spectrogram Factorization

In this section, a rough overview over BSS approaches beside note clustering is given.

3.4.1 Stereo BSS by Attenuation and Delay

The authors of [54] propose a separation framework based on a attenuation and delay model:

$$X(k, t, c) = \sum_{m=1}^{M} A(m, c) S_m(k, t). \quad (3.2)$$

The magnitude matrix $A(m, c)$ encodes the attenuation factors for source $m$ to sensor $c$, the phase matrix $e^{j\phi(m, c)} = \frac{A(m, c)}{A[1, m, c]}$ encodes the according time delay. In order to separate the different sources, each time-frequency cell of each channel except the first one is normalized by

$$C(k, t, c) \leftarrow \frac{X(k, t, c)}{X(k, t, 1)}. \quad (3.3)$$

The resulting coefficients $C(k, t, c)$ are clustered into $M$ clusters according to the $M$ different delays and attenuation factors corresponding to the $M$ distinct point sources. One disadvantage of this model is the assumption of point sources, which is not a sufficient modeling in each case, as mentioned in [27]. Additionally, in [54] it is mentioned that the clustering of the coefficients $C(k, t, c)$ needs certain steps of pre-processing by the user. Finally, this algorithm is not applicable to monaural scenarios.
3.4.2 BSS Specialized on Certain Types of Mixtures

In [52], a separation of harmonic and percussive instruments is shown. The proposed algorithm is based on the observation that percussive sounds can be simplified to vertical lines in the spectrogram, harmonic sounds to horizontal lines, see also Figure 4.4. The median filter is applied to the columns of a spectrogram for separating out percussive sounds. The median filter applied to the rows of the spectrograms separates the harmonic sounds out of the mixture. This simple algorithm is proposed as a preprocessing step in [52]. The effectiveness of this algorithm is shown by remixing commercial music. The main disadvantage of this algorithm is the restriction to the separation of harmonics and percussive attacks. The main advantages are: The algorithm is extremely fast and the output signals are identified as the harmonic or the percussive source, and no further source classification has to be done.

In [61], a separation algorithm is specialized to the separation of the main instrument in a mixture. The main instrument is assumed to be a monophonic and harmonic sound source and is modeled very similar to the approach in [14]: The frequency-basis vector (the columns of matrix $B$ in our framework) are replaced by a source-filter model with a smoothness constraint for the filter-signal. Additionally, the main instrument is restricted to a set of pure harmonic spectra and an optional unvoiced basis spectra. All signal parts not fitting this main instrument model are modeled by a standard non-negative factorization algorithm. The main disadvantage of this algorithm is the restriction to separate the dominant harmonic sound source from the background. Different instruments playing as the background signal can not be separated by this algorithm.

3.4.3 BSS by Segmentation of Spectrogram Images

In [62], it is assumed that blind source separation is possible with binary masking of spectrograms. Additionally, it is assumed that each acoustical event can be described by a segment of the spectrogram image. This segment is assumed to have a closed contour. To segment the spectrogram, the local maxima of the columns of the spectrogram are detected. These maxima are connected over the temporal axis by partial tracking (e.g. adding missing partials by median filtering over the temporal axis). A watershed-algorithm segments the spectrogram with the detected partials as seed-regions, see also [5, p.427] for further details on the watershed algorithm. The separated segments are interpreted each as a binary mask, which are applied to the spectrogram before inverse STFT.

One open problem of this approach is the clustering of these separated segments. In comparison with the NTF, the segmentation leads to much more separated components. Thus, the clustering is much more complex compared to the NTF-based framework used in this thesis. Therefore, this approach is not considered further in this thesis.

3.5 Summary

Of course, only a small number of all existing BSS approaches is explained here. The main focus is on the BSS algorithms most similar to our framework. For each algorithm
the corresponding (dis-)advantages are shown. All approaches beside the clustering based ones has one (or even more) disadvantages compared to the clustering based approaches:

- The algorithm(s) are not able to separate monaural signals because they rely on spatial informations.
- The algorithm(s) result in extremely high computational complexity.
- The algorithm(s) need user-interaction during the separation process.
- The algorithm(s) need the training of instrument specific models or they are restricted to certain mixtures.

The clustering algorithms above have either the disadvantage to be tested only on a small set of mixtures, e.g. [19]. Otherwise, they are restricted to certain models, e.g. to pure harmonic mixtures in [51].
Chapter 4

Note Separation by Non-Negative Tensor Factorization and Supervised Clustering

In Chapter 3, a short review of state-of-the-art methods for BSS is given. In this chapter, one of these approaches is picked out and explained in detail: Note separation by Non-Negative Tensor Factorization and Supervised Clustering. This approach is the basis for the unsupervised BSS algorithm introduced later in Chapter 6. Note separation by non-negative tensor factorization (NTF) is proposed by several authors, e.g. in [30], [51], or [14]. We will explain the basic idea of note separation by NTF by the algorithm used in [30]. The signal flow used for evaluation of our algorithmic framework is shown in Figure 4.1.

\[ x(n, c) = \sum_{m=1}^{M} s_m(n, c) \quad x(n, c) \rightarrow \sum_{m=1}^{M} \tilde{s}_m(n, c) \quad \text{solve alignment} \]

\[ \text{evaluate quality} \]

**Figure 4.1:** Signal flow outside the blind source separation algorithm.

The known input signals \( s_m(n, c) \) are added up to the mixture signal, which is separated into the estimated output signals \( \tilde{s}_m(n, c) \) by the blind source separation algorithm. The output signals \( \tilde{s}_m(n, c) \) are aligned to the correct input signals by switching the indices \( m \) of the output signals. This is done with knowledge of the input signals (non-blind) by maximizing the mean SNR between all input and output signals. This non-blind alignment has to be done, because the proposed algorithm separates \( M \) sources without knowledge, which source belongs to which class of input signals, i.e. the algorithm does not know, if a piano or a flute is separated. A final instrument classification is beyond the scope of this thesis.

The signal processing of the proposed BSS algorithm can be summarized basically by the following four blocks: Time-frequency transform (and post processing), note separation by NTF, signal synthesis, and note clustering, see also Figure 4.2.

These signal processing blocks will be explained in detail in Section 4.1. Obviously, each block needs to be adjusted regarding a large set of parameters. To get first insights in


4.1 Signal Flow for Note Separation

The signal flow of the proposed BSS framework is shown in Figure 4.2. Here, all necessary signal processing steps are explained in detail.

### 4.1.1 Time-Frequency Transform

According to Figure 4.2, the multichannel input signal \( x(n, c) \) with time indices \( n \) and channel indices \( c \) is transformed into a three dimensional tensor \( X(k, t, c) \) by an appropriate time-frequency transform. Each slice \( c \) of \( X \) is a spectrogram of the corresponding channel. Time-frequency analysis is usually done by the short-time Fourier transform (STFT), as explained in Section 2.2.1. Other approaches use the constant-Q transform \([51]\), or the modified discrete cosine transform (MDCT) \([63]\). The main advantage of the STFT is the constant time-frequency grid. Log-frequency transforms usually result in a non-uniform sampling of the time-frequency space, as mentioned in Section 2.2.2. The constant grid of the STFT simplifies the factorization step following the time-frequency transform, as will be explained Section 4.1.2. Additionally, the complex-valued output of the STFT can be interpreted in a very simple way: The amplitudes are very sparse for music signals, and the phases encode the local position of the oscillations. Figure 4.3 shows two examples for spectrograms created by the STFT.

As mentioned in Section 2.4.1, the frequency resolution of the human ear is logarithmic: Higher frequencies cannot be resolved in a very sharp way. Therefore, it is an obvious idea to reduce the spectrograms to a logarithmic frequency resolution, too. This reduces the computational complexity of the following analysis, as stated in \([29]\). For this, each slice \( c \) of the tensor \( X \) is filtered by a mel filter bank. This filtering can be described by a multiplication with a matrix \( R_X \), see also Section 2.4.1. Each row of \( R_X \) contains the filter coefficients of one single triangular shaped mel filter, as shown in Figure 2.6. After this filtering, the tensor \( X \) has the reduced dimension \( K_{mel} \times T \times C \). As mentioned in \([14]\), the inversion of this mel filtering can be done by multiplication with the transpose of \( R_X \). For this inversion, the coefficients in log-frequency-domain are interpreted as excitation signals for basis functions corresponding to the mel filters. This logarithmic transform also induces a logarithmic frequency warping. Contrary to

---

**Figure 4.2:** Signal flow of the proposed separation algorithm.

---

the used BSS framework experiments regarding the blocks time-frequency transform are introduced in Section 4.2 followed by experiments regarding the note separation by NTF in Section 4.3 in order to reduce the parameter space for later experiments.
other logarithmic transforms, the constant time frequency grid is kept by this transform allowing analysis by a matrix factorization algorithm.

4.1.2 Note Separation by NTF

According to Figure 4.2, the tensor $X$ is factorized into three matrices $A$, $B$, and $G$. Figure 4.3 gives a rough idea of the general structure of musical notes/acoustical events. Figure 4.3(a) shows a percussive instrument. The short duration of the single sounds results in a broad spectrum, comparable to the constant spectrum of a single impulse in time domain, see also [5]: According to Equation (2.11), a single impulse in $x(n)$ at position $n = 0$ results in the constant spectrum evaluated by the DFT of

$$X(k) = \text{DFT}(x(n)) = 1 \text{ for all } k.$$ (4.1)

Such percussive events can be simplified to vertical lines in the spectrogram. On the other hand, Figure 4.3(b) can be simplified by horizontal lines. This is a typical scenario for harmonic notes played for a long duration. The instrument emits a sound with a basic frequency (pitch). Usually, frequencies that are integer multiples of the pitch also have large amplitudes/significant amount of signal energy, see also [64]. These frequencies do not change very much within the duration of the note. Therefore, spectrograms of harmonic notes look like a set of horizontal lines. Such simplifications of spectrograms can be seen as toy examples, e.g. in [30] and [26].

The NTF can approximate horizontal and vertical structures in a matrix. In Figure 4.4, such a separation is shown. A mixture of castanets and contra bassoon (Figure 4.4(d)) is separated into three components. The contra bassoon plays two consecutive notes at two different pitches. The castanets play the same note at different timestamps. Therefore, a separation into $I = 3$ components leads to a good separation of both notes of the contra bassoon and the single note of the castanets. In Figure 4.4(e), the resulting columns of matrix $B$ are shown. The first and second column correspond to the harmonic spectra of the notes of the contra bassoon. The last column corresponds to the mean spectrum.
4.1 Signal Flow for Note Separation

Figure 4.4: Spectrogram of a mixture of castanets and contra bassoon (left) and the corresponding note separation by NTF (middle and right) in \( I = 3 \) non-negative components. The upper row shows the median filtered spectrogram and the corresponding factorization by NTF. A median filter is applied on each dimension of the spectrogram to avoid suppressing of either the harmonic or the percussive parts. The median filter is applied for better visualization only. The lower row shows the original spectrogram and the corresponding result of the NTF.

of the castanets. In Figure 4.4(f), the corresponding columns of \( G \) can be seen. It is obvious that \( G \) encodes the envelope structure of a single note. Therefore, the first and the second column of \( G \) have significant amplitudes in the temporal segments of the corresponding notes represented by the columns of \( B \). The envelope of the third component shows clearly the rhythm of the castanets.

The simple example above shows the separation of single acoustical events out of the spectrogram of a mixture. Non-negativity ensures herein a purely additive model: For explaining the necessity of the non-negativity constraint, a factorization by the SVD according to Equation (2.31) is assumed. The first three columns of the output matrices are used as decomposition of the spectrogram. This is equivalent to setting \( I = 3 \) for the NTF. As explained in Section 2.3.1, the SVD allows negative entries in the factorized matrices. Using such a factorization method can fail, because a large negative value in reconstruction matrices of component \( i_1 \) can be compensated by a large positive value in component \( i_2 \). In the end, each separated note event has high absolute amplitude/energy at a given time-frequency cell of the spectrogram, where the mixture is completely silent.

\[1\] As mentioned in Section 2.3.2, this compensation of elements by components with different signs usually occurs for ensuring the orthogonality of the basis vectors in the SVD-based factorization in the case
Note Separation by Non-Negative Tensor Factorization and Supervised Clustering

This behaviour is usually unwanted for a factorization into semantical meaningful components.

As mentioned in Section 2.3.2, the number of non-negative components $I$ is critical for factorization. Usually, the number of non-negative components for BSS is set to much smaller values compared to the value of $I$ necessary for perfect reconstruction. Typical sizes for spectrograms are $K = 2049$, $T = 100$, and $C = 1$ in the proposed framework. Therefore, perfect reconstruction is possible with $I \geq 100$. For such sizes of tensors, typical values used for $I$ in literature are:

- $I = 15$ up to $I = 20$ in [27],
- $I = 5$ up to $I = 20$ in [30], or
- $I = 25$ in [31].

For a simple solution, several constant values $I_{\text{const}}$ can be tested. Then, one value for $I$ is chosen. Another solution is proposed by the authors of [27]: For several experiments the number $I$ is set to a value depending on the number of active instruments. As we will see in Section 4.2.2 and Chapter 6, the simpler solution of choosing a single constant value for $I$ is sufficient in our context.

If a dimension reduction is applied after time-frequency transform, e.g. by a mel filter bank, the dimension reduction over the frequency axis of the tensor has to be inverted by a multiplication of matrix $B$ with $R_X^T$:

$$B \leftarrow R_X^T B. \quad (4.2)$$

4.1.3 Signal Synthesis

According to Figure 4.2, signal synthesis basically converts the output of the note factorization back into a separated note event $y_i(n, c)$, with $1 \leq i \leq I$. Firstly, the evaluation of the corresponding spectrograms is explained. The spectrograms of the separated note events contain a magnitude and a phase information:

$$Y_i(k, t, c) = Y_i(k, t, c) e^{j\phi_i(k, t, c)}. \quad (4.3)$$

For both factors, magnitude and phase, several possibilities for evaluation exist. In the following, we will explain the most common methods, but others are possible.

Synthesis of Magnitudes

The most simple way to reconstruct the magnitude of the separated sound event is using the factorized tensors directly:

$$Y_i(k, t, c) = B(k, i) \cdot G(t, i) \cdot A(c, i). \quad (4.4)$$

In [65], [51] and [14], it is proposed to use the separated components as a filter to reconstruct the magnitude information:

$$Y_i(k, t, c) = X(k, t, c) \frac{B(k, i) \cdot G(t, i) \cdot A(c, i)}{\sum_{l=1}^I B(k, l) \cdot G(t, l) \cdot A(c, l)}. \quad (4.5)$$

of overlapping sources. Therefore, it can be argued that this compensation of negative components happens only in the case of sources overlapping in spectrogram domain.
The latter version implies that all separated sources sum up to the mixture signal. Usually, the latter version results in better separation quality. Therefore, we use only this version for reconstruction of separated note events.

This filter can be interpreted in two different ways: Firstly, for each instrument \( m \) and each time-frequency-spatial cell defined by the indices \((k, t, c)\) a weighting factor is defined. These weighting factors sum up to one for all \( M \) instruments. By this, the filter coefficients can be interpreted as the probability of signal \( X(k, t, c) \) belonging to instrument \( m \). A second interpretation is given by the Wiener filter, see also [5]. Assuming two sources, the first source is defined as the signal, the second one as the additive noise. If no linear filtering has to be inverted by the Wiener filter, the optimal filter coefficients are given by

\[
\tilde{S}_m(k, t, c) = X(k, t, c) \frac{S^2_m(k, t, c)}{S^2_1(k, t, c) + S^2_2(k, t, c)}.
\]  

If the NTF approximates the power spectrogram instead of the magnitude spectrogram, the weighting factors in Equation (4.5) and in Equation (4.6) become identical. Therefore, the signal synthesis based on Equation (4.5) is called synthesis by Wiener filtering in the following.

**Synthesis of Phases**

The most common way to reconstruct the phase information is to use the phase of the original mixture:

\[
\exp(j\tilde{\varphi}(k, t, c)) = \frac{X(k, t, c)}{X(k, t, c)}.
\]  

The basic idea of using this phase is the assumption of non-overlapping sources in time-frequency domain, as explained in [54].

If this assumption is not applicable, it is possible to estimate the phase blindly. For simpler description, this idea is explained only for single-channel scenarios in the following. In [65], it is mentioned that a phase estimation algorithm can be applied on the separated magnitude spectrograms. In [66], a phase estimation algorithm is introduced and BSS is mentioned as a possible application. In [67], it is pointed out that phase estimation benefits from good time resolution if the signal is transient, e.g. the castanets. On the other hand, good frequency resolution is necessary for steady state signals, e.g. the double-bass. Unfortunately in a BSS scenario, the existence of both types of signals is possible. In [67], it is proposed to apply a phase estimation to two different spectrograms, one with a good time resolution, and one with a good frequency resolution. In order to achieve significant improvements by adaptive time-frequency processing, the window lengths for STFT has to differ significantly, e.g. \( w_{s,1} = 512 \) for transients and \( w_{s,2} = 2048 \) for steady states. To apply such a phase estimation, we have to evaluate the separation for (at least) two different spectrograms evaluated with these different window lengths. Unfortunately, the choice of the window length \( w_s \) for STFT is not very flexible: Only a small range for \( w_s \) results in good separation quality, as will be shown in Section 4.2.1. This complicates the usage of such phase estimation algorithms with adaptive time-frequency resolution in the context of the proposed BSS algorithm.

It can be assumed that for steady state signals (non-transient signals), phase estimation...
is possible. In the case of non-overlapping sources regarding the time-frequency cells of the spectrogram, this gives no better choice regarding the phase than Equation (4.7). In the case of strongly overlapping sources a phase estimation algorithm may be applied successfully.

Another possible way of reconstructing the complex tensor \( \mathbf{Y} \) is based on the work of [68]. For simpler description the index \( i \) is dropped in the following. Assuming a magnitude spectrogram \( \mathbf{Y}(k,t) \) and an initial (random) phase \( \varphi(k,t) \). Applying the inverse STFT results in a time domain signal

\[
\hat{y}(n) = \text{ISTFT} \left( \mathbf{Y}(k,t)e^{j\varphi(k,t)} \right). \tag{4.8}
\]

The transform back in the spectrogram domain results in a magnitude spectrogram

\[
\tilde{\mathbf{Y}}(k,t) = |\text{STFT} (\hat{y}(n))|. \tag{4.9}
\]

Due to the overlap add procedure during the inverse STFT, \( \tilde{\mathbf{Y}}(k,t) \) usually differs from \( \mathbf{Y}(k,t) \). In [68], the distance

\[
d = \sum_{k,t} \left| \mathbf{Y}(k,t) - \tilde{\mathbf{Y}}(k,t) \right|^2 \tag{4.10}
\]

is minimized by an appropriate adjusting of \( \varphi(k,t) \) in order to find a good phase estimation for the given magnitude spectrogram \( \mathbf{Y}(k,t) \). If the Euclidean distance in Equation (4.10) is reduced to zero, the authors of [68] call this a consistent spectrogram, because it does not change by STFT and its inverse. In this case, the phase estimation is finished. Of course, it cannot be guaranteed that this estimated phase is the correct phase. This can be seen by the following simple counter example: \( \hat{y}(n) \) and \( -\hat{y}(n) \) has an identical magnitude spectrogram and both phases \( e^{j\varphi(k,t)} \) and \( -e^{j\varphi(k,t)} \) lead to a consistent spectrogram.

Contrary to all phase estimation algorithms, not only the phases but also the magnitudes of the spectrogram

\[
\mathbf{Y}(k,t) = \mathbf{Y}(k,t)e^{j\varphi(k,t)} \tag{4.11}
\]

are distorted in a BSS scenario. Therefore, we are not only interested in a good reconstruction of phase leading to consistent spectrograms but we are also interested in using the consistency of spectrograms to get better amplitude estimations.

The matrix \( \mathbf{Y}(k,t) \) is treated as the current estimation of the output signal. The cost function in Equation (4.10) is iteratively minimized by evaluating the corresponding gradient \( \nabla \mathbf{Y} \). The current estimation of spectrogram \( \mathbf{Y} \) is updated by steepest descent:

\[
\mathbf{Y} \leftarrow \mathbf{Y} - \mu \nabla \mathbf{Y}, \tag{4.12}
\]

with \( \mu \) being an appropriate step-size. If the cost function in Equation (4.10) increases by the current step-size, \( \mu \) is divided by two. Otherwise \( \mu \) is multiplied with 1.05. This algorithm is called bold driver algorithm in [25]. The spectrogram \( \mathbf{Y} \) is consistent when the algorithm reduces the cost function below a given threshold.

It can be shown that generally the SNR between input spectrogram \( \mathbf{X} \) and output spectrogram \( \mathbf{Y} \) of a given BSS algorithm increases by this operation. Unfortunately there are two drawbacks for this method:
• The phase correlation between the single channels of a multichannel signal is not guaranteed. This can disturb the source localization after the separation.

• The gain in SNR is relatively small (generally below 0.2 dB) compared to the increasing computational complexity for signal synthesis.

Due to these problems, we use the phase reconstruction as defined in Equation (4.7).

The signal synthesis is finished by inverting the time-frequency transform to evaluate the time domain signals \( y_i(n, c) \) of the separated note events. As mentioned in Section 2.2.2, this is one of the central problems of time-frequency transforms with logarithmically spaced analysis frequencies. Therefore, such time-frequency transforms are not used in this work.

It is worth to mention that first the separated note events are transformed back into time domain, and afterwards clustered into the \( M \) different output signals. This order of operations produces a higher computational complexity compared to clustering followed by signal synthesis, as proposed in [31]. With the signal synthesis followed by the clustering, we are more flexible in evaluating features as basis for the clustering decision, because the time domain signals \( y_i(n, c) \) can be used for feature evaluation, too.

### 4.1.4 Reference Clustering

The last signal processing block in Figure 4.2 is the clustering step: Generally, the number of separated note events \( I \) is greater than the number of the active sources \( M \). Therefore, the separated note events have to be mapped onto the different active instruments. This mapping is defined by a clustering of the \( I \) notes. The melodies played by each instrument can be evaluated by adding up all notes for each of the \( M \) clusters.

In [30], clustering is done with knowledge of the input signals \( s_m(n, c) \). This is called *reference clustering* in the following. It is mentioned that the unsupervised clustering is not trivial, and even one wrong decision can have large influence on the final separation quality. Therefore the author of [30] decides to use the ground truth for clustering. With this, it is possible to evaluate the separation quality of NTF based separation algorithms. Contrary to [30], the reference clustering is not implemented in spectrogram domain, but in time domain after signal synthesis. The motivation for this is to avoid unwanted effects of the time-frequency resolution on the separation measure. The existence of such effects is shown for example in [40].

Clustering is defined by a clustering vector \( \mathbf{a}(i) \), with \( 1 \leq a(i) \leq M \), \( a(i) \in \mathbb{N} \). \( a(i) = m \) corresponds to the information that the \( i \)-th feature sample belongs to cluster \( m \).

With this vector, the estimated output signals \( \tilde{s}_m(n, c) \) can be defined by

\[
\tilde{s}_m(n, c) = \sum_{i=1}^{I} y_i(n, c) \cdot \delta_{m, a(i)} ,
\]

with \( \delta_{xy} \) being the Kronecker symbol explained in Equation (2.3). In this case, a hill-climbing algorithm [69] can be used to find the clustering, which optimizes the mean SNR according to Equation (2.78). The pseudo code for reference clustering can be seen in Algorithm 2. Due to its heuristic nature, this algorithm is not guaranteed to find the global optimum for the SNR. We will see in later experiments that the reference clustering is much better than all blind clustering approaches. Therefore it is sufficient as reference.
Algorithm 2 Reference Clustering

\[
\begin{align*}
\text{initialize } \mathbf{a}_{\text{ref}}(i) &= 0, \text{SNR}_{\text{opt}} = -\infty \text{ and improvement } = \text{true} \\
\text{while } \text{improvement } \text{do} \\
\text{improvement } &= \text{false} \\
\text{for } i = 1 \text{ to } I \text{ do} \\
\quad \text{initialize local clustering } \mathbf{a} = \mathbf{a}_{\text{ref}} \\
\quad \text{for } m = 1 \text{ to } M \text{ do} \\
\quad \quad \text{set } \mathbf{a}(i) = m \text{ and evaluate SNR} \\
\quad \quad \text{if SNR} > \text{SNR}_{\text{opt}} \text{ or } \mathbf{a}_{\text{ref}}(i) = 0 \text{ then} \\
\quad \quad \quad \text{SNR}_{\text{opt}} = \text{SNR}, \mathbf{a}_{\text{ref}}(i) = m, \text{improvement } = \text{true} \\
\quad \quad \text{end if} \\
\quad \text{end for} \\
\text{end for} \\
\text{end while}
\end{align*}
\]

Motivation for Reference Clustering

It is an obvious assumption that for a large test set with thousands of mixtures, it is not possible to define a blind clustering algorithm that can reach the separation quality of the non-blind reference clustering defined in Algorithm 2. On the other hand reference clustering is a useful clustering algorithm for analyzing the signal processing blocks Time-Frequency Transform and Note Separation by NTF shown in Figure 4.2. By using reference clustering the clustering step cannot be a source of failure and the influence of the other signal processing steps can be analyzed. Another reason is the fact, that reference clustering is the upper limit of separation quality regarding different clustering algorithms. Under these conditions, it is a natural guess that the separation quality of a blind clustering algorithm can be improved, if the separation quality of the reference clustering is improved.

Therefore, it is useful to get first insights concerning the influence of the time-frequency transform on the following factorization step.

4.2 Influence of Time-Frequency Transform

If not otherwise mentioned, the following settings are used in the experiments of this chapter: The separation quality is evaluated over a large test set of monaural mixtures of two sources ($M=2$). To compare the effects of time-frequency transforms with different parameters, the spectrogram factorization by NTF uses the following fixed parameters: As mentioned in [30], for magnitude spectrograms $\beta = 1$ performs best. The number of iterations is set to 500. The spectrogram is factorized into $I = 20$ non-negative components. The initialization of the NTF is the iterative initialization. If not otherwise mentioned, separation quality is the mean value over all sources and all mixtures for the SNR and the SNR$_{\text{seg}}$. For $h_s = \frac{w_s}{2}$, the sine window is used as analysis and synthesis window. If $h_s = \frac{w_s}{4}$ the Hann window is used as $w_1$ and $w_2$.

The separation quality is usually measured in terms of SNR$_{\text{seg}}$. As mentioned in Section 2.4.3, the SNR$_{\text{seg}}$ is evaluated over analysis segments of 20 ms length. The measured
separation quality may be influenced by the length of the analysis segments, especially in the case of experiments regarding the time-frequency resolution. For avoiding misleading information, the separation quality is also shown in terms of global SNR.

A very similar comparison is done in [70]. In this paper a so-called oracle-estimator compares the spectrograms of the input sources to construct a nearly optimal binary mask for separating the mixture spectrogram into two (or more) separated sources. This non-blind separation scenario is used to evaluate appealing transform parameters. Contrary to this binary mask, in this work a factorization and reference clustering step is used to learn the effects of different transform parameters on the BSS framework introduced in Section 4.1. As a matter of fact, this is a training of parameters on a given set of audio data (here data $\mathcal{A}$ according to the notation of Section 2.5). This makes sense, if and only if

- the data contains enough mixtures of as many as possible different instruments to be representative for nearly all possible audio mixtures. Additionally, over-fitting of the corresponding parameters is avoided by a large and diverse training set.
- the results gained for this audio data can be verified on other (possibly smaller) sets of mixtures.

In our case, data $\mathcal{A}$ consists of 60 different sources, covering human speakers, different instruments and noise. Throughout this thesis, results will be given for other audio data. Therefore, both conditions can be assumed to be fulfilled and an optimization of parameters for this audio data is legal.

### 4.2.1 Transform Length and Hop Size for STFT

![SNRseg](image1)

**Figure 4.5:** Separation quality in dB over different analysis window lengths $w_s$. The solid lines use no zero padding, the dashed ones use zero padding for doubling the DFT length. The lines without markers use a hop size of $h_s = \frac{w_s}{2}$, the ones with markers use $h_s = \frac{w_s}{4}$.

In a first experiment, the influence of three parameters on the separation quality is investigated: Window size, hop size and transform length (zero-padding). The results can be
seen in Figure 4.5.

In Section 2.2, it is already mentioned that by zero-padding a certain amount of phase-information is encoded in the magnitude spectrogram. This experiment shows that zero-padding has nearly no influence on the separation quality. Therefore, it can be concluded that the magnitude spectrogram without zero-padding is sufficient for the proposed BSS framework. For this reason, zero-padding is not considered in the following. For SNR\textsubscript{seg}, a window size of $w_s = 4096$ with a hop size of $h_s = 1024$ performs best. For SNR, a window size of $w_s = 2048$ and a window size of $w_s = 4096$ gives equal results if the hop size is set to $h_s = 1024$ in both cases. As a conclusion of the both measurements (SNR\textsubscript{seg} and SNR), the best separation quality is achieved with $w_s = 4096$ and $h_s = 1024$ without zero-padding. Therefore, these settings will be used in the following. For a sampling frequency of $F_s = 44100$ samples per second, the optimal hop size corresponds to

$$h_s = \frac{1024}{F_s} \approx 23 \text{ milliseconds}. \quad (4.14)$$

The window size without zero-padding corresponds to a frequency resolution of

$$\Delta f = \frac{F_s}{4096} \approx 10.8 \text{ Hertz}. \quad (4.15)$$

### 4.2.2 Logarithmic Frequency Scaling

![Graph showing separation quality in dB over the number of mel filters $N_{\text{mel}}$ for different values of $I$. With a window size of $w_s = 4096$ no logarithmic scaling is performed in the case of $N_{\text{mel}} = \frac{w_s}{2} + 1 = 2049$.](image)

**Figure 4.6:** Separation quality in dB over the number of mel filters $N_{\text{mel}}$ for different values of $I$. With a window size of $w_s = 4096$ no logarithmic scaling is performed in the case of $N_{\text{mel}} = \frac{w_s}{2} + 1 = 2049$.

In a second experiment, the effects of logarithmic scaling of the frequency axis are examined. For this, a mel filter bank is applied on the tensor $X$ before starting the NTF, as described in Section 2.4.1. The separation quality is shown in Figure 4.6. The mel filter bank can be interpreted as a low-pass filter combined with a sub-sampling over
the frequency axis of the spectrogram. It is reasonable that by dropping of information ($N_{\text{mel}} \leq K = \frac{w}{2} + 1$), the separation quality will decrease. Contrary to this assumption, the separation quality first increases up to a certain point somewhere around $N_{\text{mel}} \approx 0.15 \cdot K \ldots 0.2 \cdot K$. Reducing $N_{\text{mel}}$ even further results in the expected degradation in separation quality. Obviously, the optimal factor of dimension reduction (here: 0.15...0.2) also depends on the sparsity of the underlying spectrograms and the given values are optimal when data $\mathcal{A}$ is used for evaluation. If the spectrograms are even sparser it is a realistic assumption that even smaller values for $N_{\text{mel}}$ can be used for BSS. Firstly, the separation quality for $I = 20$ is examined. The increasing separation quality

![Figure 4.7: The fourth and the fifth harmonic of a single alto saxophone note is plotted. The note is played with a vibrato effect. The three lines correspond to three neighboring frames of the spectrogram.](image)

...can be explained by the suppression of vibrato effects. For visualization of this vibrato-suppression, two single notes of the musical instrument samples of the university of Iowa are mixed together: One cello note, and one note of a saxophone. The latter one is played with a strong vibrato. The NTF factorizes the mixture into $I = 2$ components. If a mel filter bank with $N_{\text{mel}} = 300$ is applied as a preprocessing step, the separation quality increases from SNR = 7.46 dB up to SNR = 15.39 dB. For a detailed analysis, three consecutive frames of the spectrogram of the saxophone note are plotted in Figure 4.7(a). For showing the vibrato-suppression of a mel filter bank, the columns are multiplied with the matrix product $\mathbf{R}^T \mathbf{R}$ for mel filtering and inverse mel filtering. The resulting columns are plotted in Figure 4.7(b). By the mel filtering, the consecutive columns of the spectrogram are low-pass filtered (smoothed) over the frequency axis. Therefore, they become more similar in the case of small changes in pitch, e.g. vibrato. This can be shown also by the correlation coefficient between neighboring columns. The normalized cross correlation coefficient between columns $t_1$ and $t_2$ of the spectrogram is evaluated according to Equation (2.7). Higher values of $\varphi_{t_1,t_2}$ correspond to higher similarity between both columns. In Table 4.1, the correlation coefficients are shown for the three columns plotted in Figure 4.7. It can be seen that the correlation of neighboring columns increases by applying a mel filter bank in the case of vibrato.
Under the assumption that each note is approximated by the product of a single column

\[
N_{\text{mel}} = 2049 \quad \varphi_{12} = 0.90 \quad \varphi_{23} = 0.83 \quad \varphi_{13} = 0.56
\]

\[
N_{\text{mel}} = 300 \quad \varphi_{12} = 0.97 \quad \varphi_{23} = 0.94 \quad \varphi_{13} = 0.83
\]

Table 4.1: Normalized cross correlation \(\varphi_{t_1 t_2}\) between spectrogram columns \(t_1\) and \(t_2\) with and without mel filtering. The spectrogram columns are the same, as plotted in Figure 4.7.

of \(B, G,\) and \(A,\) the columns \(t_1\) and \(t_2\) of this \(i\) note’s spectrogram are nearly identical up to a scaling factor defined by the corresponding entries in matrices \(G\) and \(A.\) Therefore, the columns of the spectrogram of such an approximation by the NTF should have a correlation coefficient of \(\varphi_{t_1 t_2} \approx 1.\) In this case, the single note can be approximated by one component of the NTF. It is obvious that better approximation of a single note by NTF is directly correlated with better separation quality after signal synthesis. Factorization algorithms, like NMF/NTF basically consist of matrix multiplications. As mentioned in [3], the complexity of NMF is in the order of a multiplication of the matrix sizes along all dimensions. Therefore, the mel filter bank increases the separation quality, and gives a large speed-up for the proposed algorithm\(^2\). This speed-up by a logarithmic scaling of frequency axis is also mentioned in [29]. Secondly, the effects of using different values for \(I\) are regarded. In Figure 4.6, the separation quality is plotted for \(I = 15, I = 20,\) and \(I = 25.\) It is obvious that the separation quality increases with increasing number of components \(I.\) Additionally, it is reasonable that the blind clustering (introduced later in Chapter 6) will become more complex in the case of increasing values of \(I.\) \(I = 20\) seems to be a good compromise for the given audio data, because the differences between \(I = 25\) and \(I = 20\) is much smaller than the differences between \(I = 20\) and \(I = 15.\)

4.2.3 Summary and Conclusions

As a result of these first experiments, the influence of the parameters window size and hop size on the separation quality is shown. The advantages of using a mel filter bank, better separation quality and lower computational complexity, are shown. With this knowledge, the next algorithmic step is examined: The spectrogram factorization by the NTF.

\(^2\)One additional note on the decision between \(w_s = 2^{11}\) and \(h_s = 2^{10}\) on one side and \(w_s = 2^{12}\) and \(h_s = 2^{10}\) on the other side: It can be argued that it does not make sense to increase computational complexity by using a hop size of \(h_s = \frac{w_s}{4}\) and to decrease computational complexity by the mel filter bank at the same time. Additionally it seems curious to increase the frequency resolution by choosing a window size of \(w_s = 2^{12}\) instead of \(w_s = 2^{11}\) because after applying the mel filter bank the frequency resolution is identical: \(N_{\text{mel}}.\) The mel filter bank \(R\) is an identity matrix for lower frequencies, because of Equation (2.74). Therefore, the combination of \(w_s = 2^{12}\) and \(h_s = 2^{10}\) leads to identical time resolution as an STFT with \(w_s = 2^{11}\) and \(h_s = 2^{10}.\) After applying the mel filter bank, the frequency resolution for lower frequencies is better in the first case. And the size of the resulting spectrogram \(X\) is identical after applying the mel filter bank.
4.3 Influence of Factorization Method

In Section 2.3.2, the NTF was introduced. To the best of our knowledge, no comparison of different initialization schemes for NTF in the BSS context exists. Therefore, the experiments regarding the NTF and its parameter settings start with the different initialization schemes. Additionally, practical simulations show the trend towards the combination of $\beta = 1$ (generalized Kullback-Leibler divergence) with the magnitude spectrogram. More theoretical approaches rely on the combination of $\beta = 0$ with the power spectrogram, e.g. [27]. Motivated by this, some of the corresponding advantages and disadvantages will be discussed in a second experiment.

4.3.1 Initialization and Number of Iterations

![Graphs](a) SNR$_{seg}$ (b) SNR

**Figure 4.8:** Separation quality in dB for different initialization schemes over the number of iterations.

In a first experiment, the separation quality over the number of iterations is shown for the different initialization schemes. The results are given in Figure 4.8. Random initialization corresponds to the proposal by [23]. The cost function of the NTF is minimized after random initialization with 50 iterations. After ten trials, the version with minimum cost-function is chosen as starting point for the final NTF. Random initialization with only one trial, as used e.g. in [31], is not shown in Figure 4.8 because the performance is worse than the plotted variant according to [23].

It can be seen that the sparse initialization of matrix $B$ by SVD or semantic based initialization results in better separation quality compared to initializations with dense matrices $B$ as in the random case. Semantic based initialization performs better than all other initialization schemes. This can be explained by the correlations between different frequency bins, as explained in [57]. The semantic initialization scheme gives a matrix $B$ where the correlations of different frequency bins correspond to the correlation of pure harmonic notes. Obviously, the influence of the initialization decreases with higher
number of iterations. This is verified by the fact that for a higher number of iterations the differences between the initialization schemes decreases. Additionally, after a couple of hundred iterations, the separation quality for semantic based initialization decreases. This is a side effect, of the decreasing influence of the initialization scheme after a certain amount of iterations. Assuming that the correlation between different frequency-bins of the initialized matrix \( B \) leads to good factorization results, for high number of iterations the influence of the correlation at the beginning of the NTF decreases.

In [57], it is proposed to embed a correlation constraint in the factorization model. The correlation is trained for male and female speakers and these trained models are used for separation of human speakers. It is beyond the scope of this thesis, to specialize the proposed BSS framework to certain sources. Additionally, the BSS framework should also handle non-harmonic sources, e.g. drums. Therefore, the correlation of the frequency-bins is only utilized for initialization but not used as an additional constraint for factorization as in [57].

As a conclusion, it can be stated that even after \( 10^4 \) iterations, the SVD-based initialization cannot reach the separation quality of semantic based initialization. Therefore, in the following the NTF is used with semantic based initialization and 300 iterations, if these parameters are not mentioned otherwise.

### 4.3.2 Choice of \( \beta \) and Model Order

In a second experiment, the interaction between different values \( \beta \) and the mixing model is tested.

**Choice of \( \beta \)** In literature, \( \beta \) is usually set to either \( \beta = 1 \), e.g. in [25] or [14], or \( \beta = 0 \), e.g. in [27] or [71]. The first one corresponds to the Kullback-Leibler divergence usually used as a distance measure between different pdf’s. The latter one is called Itakura-Saito distance (ISD). It is used to approximate the subjective speech quality assessment of humans, as mentioned e.g. in [72]. It is also mentioned there that the ISD can be motivated by measuring the closeness of two spectral envelopes: Finding good coefficients for linear prediction is equivalent to finding a good approximation of the envelope of the given spectrum. Motivated by this, the ISD can be defined by

\[
\text{ISD} \left( X, \tilde{X} \right) = \frac{E_X}{E_{\tilde{X}}} - \log \frac{E_X}{E_{\tilde{X}}} - 1 ,
\]

with \( E_X \) being the energy of the prediction error of a given linear prediction model regarding the spectrogram \( X \). \( E_{\tilde{X}} \) is defined in the same way.

In this thesis, we follow another motivation, introduced in [71]: Firstly, the Itakura-Saito distance is a special case of the \( \beta \)-divergence, which is an arbitrary cost-function usable for measuring the distance between two tensors. Secondly, the special case of \( \beta = 0 \) (ISD) can be motivated by approximation of the power spectrogram \( X^2(k,t,c) \) by a statistical process with zero mean and a variance of \( \sum_{i=1}^{I} B(k,i)G(t,i)A(c,i) \).
**4.3 Influence of Factorization Method**

**Model Order** $p$  The model order $p$ corresponds to the normalization and therefore to the mixing model:

$$X^p(k, t, c) \approx \sum_{i=1}^{I} B(k, i) G(t, i) A(c, i). \quad (4.17)$$

The model order $p$ interpolates smoothly between the linear mixing model shown in Equation (2.82) and the squared mixing model shown in Equation (2.83). Assuming statistical independent phases of input spectrograms $S_m$, the power spectrograms can be added, resulting in a model order $p = 2$, as stated in [25]. Assuming non-overlapping spectrograms $S_m$ the model order can be set to $p = 1$, as done in [30]. From this point of view, $p = 2$ seems to be the more realistic assumption. Unfortunately, $p = 2$ increases the dynamic range of $X^p$ as the input-signal for the NTF. Higher values of $p$ correspond to higher dynamic ranges of $X^p$. As stated in [73], lower values of $\beta$ correspond to lower sensitivity to large dynamic ranges of the input $X^p$.

These assumptions can be verified in Figure 4.9. Large dynamic differences between both input signals $S_1$ and $S_2$ are simulated by adding the signals with a difference of energy of ±12 dB. In these cases, $\beta = 0$ outperforms $\beta = 1$ clearly, due to its insensitivity to absolute values of amplitudes. Additionally, it can be seen that for $\beta = 0$ the choice of model order $p$ is nearly irrelevant. On the other hand, $\beta = 1$ works best with model order $p = 1$. This combination is chosen e.g. in [30] or [51].

As a conclusion: The dynamic differences between input signals are usually unknown and $\beta = 0$ performs much better in the case of large dynamic differences. Therefore, $\beta = 0$ is used in the following if not otherwise mentioned. The model order $p = 1$ implies non-overlapping sources, which can only be assumed for a small number of active sources $M$. For greater $M$, the squared mixing model is more realistic because the sources are assumed to overlap in time frequency domain. Therefore, the model order is set to $p = 2$ in the following.

![Figure 4.9: Separation quality in dB for the Kullback-Leibler divergence ($\beta = 1$) and the Itakura-Saito distance ($\beta = 0$). The model order $p$ varies from the squared mixing model ($p = 2$) to the linear mixing model ($p = 1$).](image-url)
These results are equivalent to the results shown in [48]. In their simulations, the sources are mixed in an identical scheme as in [14]: 2 sources are mixed at equal loudness, 3 sources at nearly equal loudness. For such a scenario, $\beta = 1$ leads to better results compared to $\beta = 0$. As we have shown in our experiments, when large dynamic differences between active sources exist, $\beta = 0$ performs much better.

Another advantage of $\beta = 0$ can be shown by the following example. It is well-known that the harmonics at higher frequencies have different envelopes than the harmonics at lower frequencies, see also [64]. The more sensitive to higher amplitudes a cost function is, the higher the probability that the different envelopes of the single harmonics of a single note are factorized into two or even more factorized components. If the cost function is insensitive to signal energies, it is more likely that a single note is expressed only by a single component, because additionally, more silent regions with lower amplitudes (e.g. noise regions) are factorized by a certain number of components. This behaviour can be observed in [3]: A piece of music consisting of four notes is factorized into $I = 6$ components. For Euclidean-NMF ($\beta = 2$), five harmonic components are extracted, which means that one note is factorized into two components. The same happens for $\beta = 1$. Only for $\beta = 0$, four harmonic components are extracted.

4.3.3 Mixing Model in the Multichannel Case

In Figure 4.10 different multichannel factorization methods are compared. The solid line corresponds to standard NTF. For the dashed line, the spectrograms for each channel $c$ are concatenated along the temporal axis. Therefore, the resulting spectrogram has the dimension $K \times (C \cdot T)$. For the dotted line with markers, the spectrograms are concatenated along the frequency-axis resulting in a spectrogram of dimension $(C \cdot K) \times T$.

As mentioned in Section 2.3.2, the concatenated versions have the advantage of faster computation. The NTF assumes an instantaneous mixing model. The concatenated versions can handle a convolutive model:

- Concatenation of $B$ assumes short impulse responses of the transfer paths between the sound source and the microphone, such that the convolution can be expressed by a multiplication along the frequency-axis of the spectrogram. This results in different frequency basis vectors for each channel $c$.
- Concatenation of $G$ assumes long impulse responses, such that the convolution results in time delays noticeable along the temporal axis of the spectrogram.

The curves show separation quality for different values of $I$. As the factorization by NTF is more compact than the factorization of concatenated matrices, the parameter $I$ cannot be used as x-axis. Instead, the parameter $I$ is scaled to degrees of freedom $\tilde{I}$ as defined in the following equation

$$\tilde{I} = \frac{I (K + T + C)}{K \cdot T \cdot C},$$ (4.18)

The separation quality is evaluated for data $B$. This is the only data in our test environment with realistic mixing effects used by sound engineers. Several effects are obvious: Firstly, the separation quality is very sensitive to the parameter $I$, even in the case of non-blind reference clustering. Secondly, usually the NTF gives best separation quality followed by the concatenation over frequency-axis. Therefore, it can be assumed that the
4.3 Influence of Factorization Method

![Graphs showing separation quality in dB for different factorization models and 5 ≤ I ≤ 30.]

(a) Bearlin - Roads  (b) Tamy - Que Pena Tanto Faz
(c) Another Dreamer - The Ones We Love  (d) Fort Minor - Remember the Name
(e) Ultimate NZ Tour

**Figure 4.10:** Separation quality in dB for different factorization models and 5 ≤ I ≤ 30.
mixing effects can be approximated by an instantaneous mixing model or a convolutive mixing model with short mixing filters. For very short mixing filters, the instantaneous mixing model is sufficient, because for the given window size and hop size these short mixing filters mostly affect the phases, leaving the magnitudes of the spectrogram identical over the different channels up to a given constant corresponding to the instantaneous mixing coefficient.

Beside the three discussed multichannel factorization algorithms in Appendix B two additional algorithms are proposed. They are not discussed in more details because no experimental results are given.

4.4 Summary and Conclusions

The BSS framework proposed so far in this thesis is able to separate monaural mixtures. The algorithm's complexity is relative small compared to the extended factorization models in e.g. [27] or [14].

The necessary time frequency resolution is discussed in a first experiment. After that the advantages of using a logarithmic scaling of the frequency axis is shown: Lower computational complexity and robustness in the case of vibrato.

The most important experiment regarding the factorization shows: The initialization scheme has major influence on the separation quality, which can be explained by better initial correlation between frequency coefficients: The semantic initialization shall be preferred in the case of audio analysis.

The higher robustness of $\beta = 0$ compared to $\beta = 1$ is shown, regarding the separation quality for large dynamic differences between the input signals. The NTF with $\beta = 0$ performs better, especially for large dynamic differences between the input signals.

In a last experiment, it is shown that NTF and NMF with concatenation over frequency axis lead to comparable results with slightly better separation quality for the NTF.

In the following, blind clustering algorithms will be introduced in the remaining chapters to dispose the last remaining disadvantages: Necessary user-interaction and training of instrument specific models. Both disadvantages come from the usage of the reference clustering.
Chapter 5

Audio Features

Human beings usually learn a wide variety of different acoustical sources. Different voices, several instruments, and even a large range of artificial noisy sources, e.g. cars or machines, are examples for the trained database of known acoustical sources. In the following, instrument is a synonym for each kind of acoustical source. The basic idea of note clustering is to evaluate meaningful features in order to distinguish between different instruments. This task is different to learning/training of models for certain types of instruments, which is done e.g. for percussive/harmonic sources in [50] or for speech/music separation in [74]. Contrary to these scenarios, it is assumed that only the number of the active sources ($M$) is known by the algorithm and not the types of active instruments. Therefore, it is only necessary to find $M$ different types of signal features.

In the following, criteria or informations useful for discriminating between different instruments or more generally spoken between different clusters are called features. The abstract audio parameter timbre is usually defined by the difference between two sounds having the same loudness and the same pitch, [35, p.426]. In the same book, the ability of humans to differentiate between several sound sources is explained by a three dimensional feature space:

- **Spectral energy distribution** distinguishes between instruments with band limited spectra and the signal energy being located at the lower frequency range vs. instruments with significantly wider spectra (without band limits) and a non-negligible part of the signal’s energy located at the higher overtones.

- **Static versus dynamic quality** distinguishes between instruments with identical envelopes for the higher overtones and instruments with major changes between the single envelopes of each overtone.

- **Buzz-like but softer attack versus explosive initial attack** distinguishes between instruments regarding the transient phase of the notes and the time necessary for switching from the transient to the steady-state segment of a note.

This three-dimensional feature space is the motivation for using features utilizing not only informations from frequency domain, introduced in Section 5.3, but also features evaluated in the time domain, introduced in Section 5.4.

Additionally, spatial features will be discussed, based on the assumption of point sources. The ability of humans to locate audio sources is discussed e.g. in [35, pp.297]. For BSS, spatial localization is utilized e.g. in [54] and [27]. Motivated by this, spatial features will
be discussed in Section 5.5.
Features for musical instrument classification are proposed e.g. in [75] and [15]. Beside other features, both use mel frequency cepstrum coefficients (MFCC) and in both papers these MFCC outperform all other features clearly. MFCC are based on cepstral analysis, see also [76]. Although the physical unit of the cepstral domain is \( \frac{1}{\text{Hertz}} \), it is usually not assigned to the temporal or the frequency domain of a signal, because non-linear signal processing steps are necessary for cepstral analysis, e.g. the logarithm or a non-linear frequency warping. Therefore, features utilizing any kind of cepstral analysis are assigned to the feature space called cepstral features, introduced in Section 5.2.
In [75], it is proposed to extract audio features separately for the transient region and the steady state region of an instrument. In [77], this is tested for the BSS framework also used in this thesis. The separated components \( y_i(n, c) \) are segmented in regions corresponding to the three states transient, steady-state, and silence. Features are extracted for the transient and the steady state parts of the signals. It cannot be observed that any additional information is gained for our framework by this segmentation. This has mainly three reasons:

- The spectrogram of \( y_i(n, c) \) is evaluated by a Wiener-like filter, see also Equation (4.5). This can be interpreted as a filtering of the mixture’s spectrogram by a filter with a constant spectrum but a time-varying and channel-varying amplitude. Therefore, the spectral information is nearly constant over the whole signal, regardless if the local segment is a transient or a steady-state part. Therefore, the spectral features evaluated for signal \( y_i(n, c) \) can be assumed to be constant.

- The transient region of a signal’s spectrogram is usually less sparse compared to the steady state region, see also Figure 4.3. However, by the filtering mentioned above, a large amount of the additional information in the transient region is lost. Therefore, even the temporal features, e.g. the zero crossing rate of the transient and the steady-state will be nearly identical due to the proposed signal synthesis scheme.

- A third argument for dropping the segmentation is an effect of the NTF shown in [25, p.16]: The harmonical notes of the spectrogram shown there are separated into single components. The transient parts of all notes are recognized as a single pitch-independent component, similar to a virtual percussive instrument. In such a scenario a segmentation of a separated component \( y_i(n, c) \) into transient and steady-state part will not lead to additional information.

For describing the features, the following conventions will be used: \( I \) samples of an \( N \)-dimensional feature are stored in a so-called feature matrix \( F_{xy} \) of size \( N \times I \). The index \( xy \) is used to discriminate between different features. The data \( y_i(n, c) \), \( B(k, i) \), \( G(t, i) \), and \( A(c, i) \) are the only input used for feature extraction for component \( i \). For evaluation of spectral and cepstral features, it is possible to use either \( B(k, i) \) directly, or to transform \( y_i(n, c) \) into a spectrogram tensor \( Y_i(k, t, c) \) and to evaluate the features on this structure. Examples for both cases are given in Section 5.3. In the case of multichannel signals, the features based on \( Y_i(k, t, c) \) are averaged over all channels \( c \) to get the final features used for clustering.

This chapter is structured as follows: In Section 5.1, optional pre-processing steps for the audio features are introduced. After that, the audio features are explained in detail.
5.1 Pre-Processing of Features

In the following, the upper index $\nu$ of feature matrix $F^{(\nu)}_{xy}$ denotes the fact that this feature matrix has passed $\nu$ evaluation steps or pre-processing steps. The distinction between feature evaluation and feature pre-processing is not easy to find. Therefore, in the following the upper index ($\nu$) is used to indicate the number of evaluation steps. The general signal flow for feature evaluation and clustering is shown in Figure 5.1. Usually, the features proposed later in this chapter need several iterations of pre-processing steps. Namely the dimension reduction and the data normalization are explained in detail in the following.

5.1.1 Data Model/ Data Normalization

To each feature space $F^{(\nu)}_{xy}$, a given data model is assumed. The most common assumption for data is the unimodal distribution shown in Figure 5.2(a): Each class can be described sufficiently by a representative centroid, with all other class members located around this centroid due to a certain amount of noise induced by the variances of the class itself. This variance is called *intra class scatter* in the following. The different steps of feature evaluation and pre-processing aim at reducing the intra class scatter as good as possible. Another possible assumption corresponds to the radial data model shown in Figure 5.2(b). For this data model it is assumed that the dimensions of the feature space are strongly correlated. For the radial model, it is reasonable that feature samples with higher amplitudes (located far away from the point of origin in Figure 5.2) are better suited for estimation of the radial cluster center (plotted as dotted lines in the same figure) or in other words: The amplitudes and the normalization of the feature samples can have critical influence on the clustering quality. As mentioned e.g. in [25], the energies of the columns of matrices...
Audio Features

Figure 5.2: Possible samples for a two dimensional feature space $F_{xy}$. The samples are taken from $M = 2$ different sample distributions: Distribution 1 is marked with circles, distribution 2 is marked with squares. Different data models inducing different distance functions necessary for clustering.

$A$, $B$, and $G$ can be exchanged freely. To avoid the preference of a certain dimension of the tensor, the three columns of matrices $A$, $B$, and $G$ corresponding to component $i$ are scaled to equal energy for the context of clustering.

Examples for both distributions, unimodal and radial, and the according data normalizations will be given in this section.

Unimodal Normalization

The standard normalization for unimodal distributed features is to normalize each feature’s dimension to zero mean and unit variance, as proposed e.g. in [5]. In the following, two examples are shown, one with positive influence of normalization, and one with negative influence. Quality of influence is measured by the error-rate of the standard clustering algorithm k-means, which will be explained in detail in Section 6.1.1. To reduce the impact of initialization of the k-means algorithm, the cluster centroids are initialized by the true underlying cluster centroids.

In Figure 5.3, samples of a two-dimensional, unimodal feature are plotted. The feature samples are drawn from two classes. The x-axis has a much larger variance than the y-axis. Therefore, the clustering is mainly influenced by this axis. Normalizing the features to zero mean and unit variance reduces the error rate from 44 % to 1 %.

Normalization of the dimensions of the feature space has smaller positive or even negative influence on the classification rate, if the feature space has strong correlation between different feature’s dimensions. In the example shown in Figure 5.4 the normalization increases the error rate slightly from 44 % to 47 % by normalization of the variances.

The latter example induces that a normalization using the inverse of the covariance matrix of the given samples will avoid the shown negative influence. The Linear Discriminant Analysis (LDA) or the Principal Component Analysis (PCA) utilizes a normalization of the data by the inverse of the covariance matrix. More informations regarding the algorithms LDA or PCA can be found e.g. in [78]. For clustering, the inverse of the covariance matrix of the given feature space can be used for normalization of the input data, e.g. by
Figure 5.3: Influence of normalization for unimodal distributions: If the variances along the different dimensions are significantly different, the normalization can have positive influence on the separation results. Here, a clustering algorithm is applied on the data, and the borderline is plotted in a dashed style. By normalization, the error rate decreases from 44 % to 1 %.

Figure 5.4: Influence of normalization for unimodal distributions: If the variances along the different dimensions are nearly the same or the features have strong correlations, the influence of normalization on the separation results can become negative. In this figure, a clustering algorithm is applied on the data, and the borderline is plotted in a dashed style. By normalization, the error rate increases from 44 % to 47 %.

Spherical Normalization

Another possibility for data normalization is shown in Figure 5.5. The spherical normalization is based on a radial distribution model. For such distributions the main clustering information is the angle of a vector and not its amplitude. For the spherical normalization, each feature sample is normalized to unit length. As mentioned in [6], this feature space
Figure 5.5: Possible samples for a two dimensional feature space $F_{xy}$. The samples are taken from $M = 2$ different sample distributions. In the right Figure, the radial distributed feature samples are normalized to unit variance. For better visualization only a subset of the feature samples is shown.

![Diagram of feature space](image)

Figure 5.6: Interpretation of unimodal and spherical normalization as identical operations applied on the different dimensions of feature matrix $F_{xy}^{(\nu)}$.

needs a spherical distance function, e.g. shown in Equation (2.8).

Another point of view onto the spherical normalization and the unimodal normalization of feature space $F_{xy}^{(\nu)}$ is shown in Figure 5.6. Unimodal normalization is applied on the rows, spherical normalization on the columns of feature matrix $F^{(\nu)}$. However, subtracting the mean is usually not helpful for spherical normalization: Subtracting the mean can arrange samples with nearly identical angle (but different amplitudes) on different sides of the point of origin of the given feature space. This positioning results in the maximum possible distance.

### Statistical Normalization

Beside normalizing the variances along different dimensions, it is possible to normalize features in a statistical way. The Box-Cox transform (BCT), explained in [79], is such a non-linear mapping to handle skewness of the underlying pdf or to get rid of outliers, see
5.1 Pre-Processing of Features

Figure 5.7: In the left figure, a two-dimensional set of feature-samples belonging to \( M = 2 \) classes is shown. With k-means clustering, this feature-set results in 41.5 % wrong classifications. After applying the Box-Cox transform on the first dimension with \( \lambda = 0.1 \) (right figure), the error-rate decreases to 5 %.

also [77]. The BCT is defined by

\[
\mathbf{F}^{(\nu)}_{xy}(n, i) = \text{BCT}_\lambda \left( \mathbf{F}^{(\nu-1)}_{xy}(n, i) \right) = \begin{cases} 
\log \left( \mathbf{F}^{(\nu-1)}_{xy}(n, i) \right), & \text{if } \lambda = 0 \\
\left( \mathbf{F}^{(\nu-1)}_{xy}(n, i) \right)^{\lambda-1}, & \text{otherwise,}
\end{cases}
\]

(5.1)

with \( \lambda \) being a user defined parameter. Assuming non-negative feature samples, \( \lambda > 1 \) increases the dynamic range of the feature samples to be transformed and \( \lambda < 1 \) decreases the dynamic range. From this point of view, the BCT can be interpreted as a compressor or an expander as used in digital speech coding, see also [76].

With an offset +1, the Box-Cox transform maps non-negative features \( \mathbf{F}^{(\nu-1)}_{xy} \) to non-negative values \( \mathbf{F}^{(\nu)}_{xy} \):

\[
\mathbf{F}^{(\nu)}_{xy}(n, i) = \text{BCT}_\lambda \left( \mathbf{F}^{(\nu-1)}_{xy}(n, i) + 1 \right) = \begin{cases} 
\log \left( \mathbf{F}^{(\nu-1)}_{xy}(n, i) + 1 \right), & \text{if } \lambda = 0 \\
\left( \mathbf{F}^{(\nu-1)}_{xy}(n, i) + 1 \right)^{\lambda-1}, & \text{otherwise.}
\end{cases}
\]

(5.2)

5.1.2 Dimension Reduction

In Figure 5.1, it can be seen that in our scenario of note-clustering for BSS, we have only a very small number \( I \) of feature samples. So in the case of multi-dimensional features, this feature space is very sparse. It is hard to estimate the correct underlying pdf’s of the different feature clusters in the case of sparse feature spaces. This problem is often called curse of dimensions, see also [7]. Reducing the number of dimensions can reduce this sparsity.

Linear Dimension Reduction

The simplest way of dimension reduction is feature selection. It is described by forming the pre-processed feature matrix \( \mathbf{F}^{(\nu)}_{xy} \) out of a certain selection of rows of feature matrix
Audio Features

\[ \mathbf{F}^{(\nu-1)}_{xy} \] as shown in Figure 5.8.

A generalization of the feature selection method is the feature weighting. For this, the \( n \)-th feature (\( n \)-th row) \( \mathbf{F}^{(\nu-1)}_{xy}(n, i) \) is weighted by a scalar \( \mathbf{f}_{\text{weight}}(n) \). With \( \mathbf{f}_{\text{weight}}(n) \in \{0, 1\} \), the feature weighting becomes a feature selection. If the feature weighting is generalized to linear feature transforming, the pre-processing can be described by a matrix multiplication:

\[
\mathbf{F}^{(\nu)}_{xy} = \mathbf{R}_{xy} \mathbf{F}^{(\nu-1)}_{xy}. \tag{5.3}
\]

The DCT as explained in Section 2.1.2 is an example for such a linear feature transform, as explained in [5]. Feature weighting is a special case of linear feature transform. In this special case, only the main diagonal of matrix \( \mathbf{R}_{xy} \) is non-zero. Setting whole rows of matrix \( \mathbf{R}_{xy} \) to zero, results in a linear dimension reduction.

In [77], it is shown that other linear dimension reduction algorithms, e.g. the PCA, deteriorate the separation results of the clustering algorithms, because dimension reduction is done along the dimensions with smallest variance, which may be critical as shown in Section 5.1.1 (as a matter of fact, PCA also includes a rotation of the original feature dimensions, but for simplicity of argumentation this rotation can be ignored). To overcome this problem, the LDA can be used, but this algorithm needs knowledge about the feature distributions for each class, which is an information that is not at hand in our scenario.

**Non-Linear Dimension Reduction**

There exists a large amount of algorithms usable for non-linear dimension reduction, e.g. Self Organizing Feature Maps (SOFM), Local Linear Embedding (LLE), or Non-Linear Mapping (NLM). All three are algorithms, used for visualization of high-dimensional data in a two- or three-dimensional feature space. The (dis-) advantages will be discussed shortly and it is motivated, why only the last one (NLM) will be used in the experiments.

**Self Organizing Feature Maps** Self organizing feature maps (SOFM) utilize an artificial neural network for the purpose of dimension reduction. As explained in [7, p.576], SOFM maps a high-dimensional or a non-linear feature space to a map of lower dimension (usually two) by preserving the *neighbourhood*. Neighbourhood is defined by a unimodal window.
function. The choice of this window function is critical, because it has to cover even outliers for stable learning. On the other side, the larger this window, the more relaxed the definition of neighbourhood, which may be a problem during the learning process.

**Local Linear Embedding** Local linear embedding (LLE) is introduced in [80]. The neighbourhood of each feature vector is preserved by a local-linear mapping, i.e each feature vector is approximated by a linear combination of its neighbouring feature vectors. Non-linearity is introduced by considering only the neighbourhood of each feature vector for approximation. After finding a good linear combination a set of feature vectors with lower dimension but similar neighbourhoods is searched. The necessity of defining an appropriate *neighbourhood* makes the LLE less flexible than the last non-linear dimension reduction scheme: The non-linear mapping.

**Non-Linear Mapping** Non-linear mapping (NLM) was introduced by [81]. Assuming a set of \( N \)-dimensional feature vectors \( F_{xy}^{(\nu-1)} \), the NLM tries to represent \( F_{xy}^{(\nu-1)} \) by a set of \( N_1 \)-dimensional feature vectors \( F_{xy}^{(\nu)} \), with \( N > N_1 \). For allowing arbitrary mappings from the \( N \)-dimensional space to the \( N_1 \)-dimensional space, the algorithm minimizes a cost function

\[
E = \frac{1}{\sum_{i<j} d_{ij}^*} \sum_{i<j} \left( \frac{d_{ij}^* - d_{ij}}{d_{ij}} \right)^2,
\]

(5.4)

with \( d_{ij}^* \) being the distances between feature vectors \( i \) and \( j \) in the \( N \)-dimensional domain, and \( d_{ij} \) being the distance in the \( N_1 \)-dimensional domain. Starting with arbitrary (e.g. random) features \( F_{xy}^{(\nu)} \), the NLM iteratively minimizes the cost function in Equation (5.4) by applying Newton’s method for finding local minima of a two times differentiable function. According to [81], the Newton-update is multiplied by a (constant) *magic-factor* of 0.3. In the paper, the algorithm is derived for the Euclidean distance. Here, we will use *steepest descent*, instead of Newton’s method. Additionally, the update is derived for the spherical distance introduced in Section 2.1.1. For steepest descent, only the first derivative is necessary, which simplifies the update rules, see also [24]:

\[
\frac{\partial E}{\partial F_{xy}^{(\nu)}(n_1,i)} = -2 \sum_{i<j} \frac{d_{ij}^* - d_{ij}}{d_{ij}} \cdot \frac{\partial d_{ij}}{\partial F_{xy}^{(\nu)}(n_1,i)},
\]

(5.5)

with

\[
\frac{\partial d_{ij}}{\partial F_{xy}^{(\nu)}(n_1,i)} = \frac{F_{xy}^{(\nu)}(n_1,i) - F_{xy}^{(\nu)}(n_1,j)}{d_{ij}} \text{ (Euclidean distance)},
\]

(5.6)

\[
\frac{\partial d_{ij}}{\partial F_{xy}^{(\nu)}(n_1,i)} = -F_{xy}^{(\nu)}(n_1,j) \text{ (spherical distance)}.\]

(5.7)

Here, Equation (5.7) assumes that the feature vectors are normalized to unit length. As there exists no direct linkage between the input features \( F_{xy}^{(\nu-1)} \) and the output features\footnote{For a deterministic initialization, in [81] it is proposed to select the \( N_1 \) rows of \( F_{xy}^{(\nu-1)} \) with largest variance as initialization for \( F_{xy}^{(\nu)} \).}
Audio Features

It is possible to transform the feature space from unimodal to spherical\(^2\) or to introduce additionally constraints, e.g. non-negativity. This can be an advantage if the used clustering algorithm needs non-negative features. For constraining the output features to non-negative values, negative elements in \(F_{xy}(\nu)\) are set to zero after each update. Convergence cannot be guaranteed by this non-linear changing of values independent from the gradient. However, it can be observed that the cost function described in Equation (5.4) decreases over a large number of iterations.

All three proposed dimension reduction algorithms are non-linear by nature. SOFM needs the definition of a window function defining the neighbourhood. It is not very likely that every set of features introduced in this chapter works well with a unique window function. To avoid the definition of one window function for each feature space, SOFM will not be considered in the following. LLE is locally linear, which is an unwanted restriction for dimension reduction. Therefore, we restrict ourself to the usage of NLM for non-linear dimension reduction in the following.

5.2 Cepstral Features

The source-filter model (SFM) is the standard scenario for cepstral analysis. The SFM will be explained by a simple model of the violin. But it is also applicable to the human voice [76, p.299].

It is assumed that the string of the violin emits a pure harmonic signal \(s(t)\), which is filtered in time domain by the resonance body of the instrument. This can be simplified to a linear time-invariant system with impulse response \(h(t)\). The final output sound of the instrument \(b(t)\) can be described in time- and frequency domain according to [1] by

\[
s(t) * h(t) = b(t) \leftrightarrow S(f) \cdot H(f) = B(f),
\]

with \(\mathcal{F}\) corresponding to the Fourier transform. This multiplication on the right side of Equation (5.8) is shown in Figure 5.9: In frequency domain, the source signal \(S(f)\) is a sparse set of harmonically related peaks. The relative amplitudes are induced by the instrument filter \(H(f)\) thus forming the spectrum of the note \(B(f)\). The notation \(B(f)\) is chosen for consistency with the NTF, for which the spectra of the notes are encoded in the columns of matrix \(\mathbf{B}\).

To separate the source and the filter signal, the logarithm is applied on the absolute values of the modeled spectrum \(B(f)^3\):

\[
\log |B(f)| = \log |S(f)| + \log |H(f)|.
\]

\(^2\)Assuming a feature space with an underlying spherical data model. It is possible that the Euclidean distance has to be preferred for clustering, e.g. because it is faster to evaluate, or because the clustering algorithm is well-defined for the Euclidean distance but not for the spherical distance, e.g. the expectation maximization algorithm. In this case, the distances \(d_{ij}\) are evaluated by the spherical distance. By evaluating the distances \(d_{ij}\) as Euclidean distance, the NLM transforms the underlying data-model from radial to unimodal, see also Figure 5.2.

\(^3\)If the absolute value is not taken, the complex cepstrum is analysed. In this thesis only the real-valued cepstrum is used. For further details about the complex spectrum we refer e.g. to [76]
By the logarithm, the product in frequency domain is transformed into a summation. According to Figure 5.9, a frequency analysis (e.g. a Fourier transform) of the source signal $|S(f)|$ shall reveal more signal energy at higher frequencies compared to the much smoother filter signal $|H(f)|$, which shall reveal somehow low-pass characteristics. This interpretation of the frequency content of both signals combined with the summation in Equation (5.9) leads to separable source and filter signals: The source and the filter signal can be separated by simply applying a low-pass and a high-pass on $\log|B(f)|$, see also [76, pp.313]. This low-pass or high-pass filtering is usually done by a linear transform, e.g. the DCT or the DFT, followed by a feature selection. The frequency transform applied on a spectrum results in a kind of time-domain representation. Because the frequency analysis is applied on the logarithmic magnitude spectrum, the signal domain is called cepstral domain instead of time domain. Throughout this thesis, features based on the analysis of the logarithmic magnitude spectrum are called cepstral features, even if no final transform is applied. This is done, to distinguish these features from other spectral and temporal features.

### 5.2.1 Mel Frequency Cepstrum Coefficients

In [15], it is stated that the mel frequency cepstrum coefficients are strongly related to extraction of the filter signal. The evaluation of the MFCC is described by the following three steps:

- The time domain signal is windowed and the DFT is applied (i.e. applying the STFT).
- The modulus of the DFT is filtered by a mel filter bank and the logarithm is applied to the output.
- The logarithmic mel filter outputs are analysed by the DCT.

Due to the logarithm, the magnitude (or the energy) information of the signal becomes additive and Parseval’s theorem can not be applied any longer to the coefficients: Only the first coefficient of the DCT is strongly related to the signal’s energy. Therefore, it is usually dropped. The coefficients corresponding to the higher frequencies are also dropped, because the MFCC’s are related to the filter signal. According to the SFM used for cepstranal analysis, the source-signal will be located at the DCT-coefficients related to the higher frequencies. As discussed in Section 5.1.2, the dropping of features is interpreted
as a dimension reduction in our framework. Therefore, we start with all MFCC in our framework and discuss the dropping of certain coefficients in the experimental results regarding the dimension reduction.

Because of the evaluation of \( Y_i(k, t, c) \) according to Equation (4.5), the frequency content can be interpreted as nearly constant over the whole signal. Therefore, evaluation of the MFCC on the basis of the STFT of \( y_i(n, c) \) is not necessary. Instead, the evaluation can be simplified to

\[
F^{(0)}_{\text{MFCC}}(k_{\text{mel}}, i) = \sum_k R_{\text{MFCC}}(k_{\text{mel}}, k) B(k, i) \tag{5.10}
\]

\[
F^{(1)}_{\text{MFCC}}(k_{\text{mel}}, i) = 20 \log_{10} \left( f_{\text{MFCC}} F^{(0)}_{\text{MFCC}}(k_{\text{mel}}, i) + 1 \right) \tag{5.11}
\]

\[
F^{(2)}_{\text{MFCC}}(n, i) = \text{DCT} \left( F^{(1)}_{\text{MFCC}}(k_{\text{mel}}, i) \right) \tag{5.12}
\]

The matrix \( R_{\text{MFCC}} \) corresponds to the mel filter bank mentioned above. Each row of \( R_{\text{MFCC}} \) contains one single triangular mel filter, see also Figure 6.3(b) for a simple example for such a matrix.

The DCT in Equation (5.12) transforms the \( i \)-th column of feature matrix \( F^{(1)}_{\text{MFCC}}(k_{\text{mel}}, i) \) into DCT-domain. In principle, every frequency-transform can be used for the last step, e.g. the DFT as proposed in [76]. But in our experiments the best results are obtained with the DCT. The constant offset +1 in Equation (5.11) ensures that non-negative values are mapped onto non-negative values, as already mentioned for the BCT in Section 5.1.1.

This is necessary in the case of using the logarithmic output for a distance function, as can be seen in Figure 5.10. In Figure 5.10(a), the solid line shows an arbitrary signal. The dashed line is the same signal with a small amount of additive noise. In Figure 5.10(b) both signals are plotted after applying the logarithm. In Figure 5.10(c), the logarithm is applied with an offset +1. It can be seen that by adding the offset of +1 the difference between both signals is reduced in logarithmic domain significantly which is important for further (dis-) similarity analysis.

The constant offset results in different effects on values of high or low amplitudes. Therefore, it is useful to scale the dynamic range of the input signal by the factor \( f_{\text{MFCC}} \), such that the logarithmic output is not dominated by the +1 offset.

**Temporal Evolution of MFCC**

A typical extension to the MFCC features is the analysis of their evolution over the temporal axis. A simple description of the temporal behaviour of the MFCC are the \( \Delta \text{MFCC} \) and the \( \Delta \Delta \text{MFCC} \) corresponding to the first and second derivatives over the temporal axis, as used e.g. in [75], [37], and [82]. Again, these features can be thought of being not useful in our scenario, due to the signal synthesis as defined in Equation (4.5). Due to this Wiener-like filtering, the columns \( t \) of \( Y_i(k, t, c) \) can be assumed to be approximately constant up to a scaling factor induced by the multiplication with \( A(c, i) \) and \( G(t, i) \). Therefore, the MFCC evaluated for each column of \( Y_i(k, t, c) \) can be assumed to be approximately constant except the first coefficient, which is the only coefficient influenced by a scaling factor. From this point of view, it is reasonable that the \( \Delta \text{MFCC} \) and the \( \Delta \Delta \text{MFCC} \) contains no additional information compared to the MFCC.
Figure 5.10: In Figure 5.10(a), an arbitrary signal (solid line) and the same signal with a small amount of additive noise (dashed line) is shown. In Figure 5.10(b), both signals are shown after applying the logarithm. For small values of $x$, the difference between $x_1(n)$ and $x_2(n)$ can increase up to 10 dB. The influence of noise is much smaller when using the offset +1, as shown in Figure 5.10(c).

5.2.2 Temporal Source-Filter Model

Motivated by the observation that human listeners not only interpret the spectrum but also the envelope to classify an acoustical event, in [83] several strategies are introduced to utilize the envelopes for clustering in a BSS scenario. It is assumed that each instrument has a characteristic envelope. This assumption is also used for the well-known attack-decay-sustain-release (ADSR) model, used e.g. in [82]. This characteristic ADSR model is convolved with an activation vector corresponding to the informations onsets, rhythm, and amplitudes. An example for such a model is shown in Figure 5.11. One possibility for utilizing such a model in the context of clustering is the factorization of the matrix $G$ by a non-negative matrix deconvolution algorithm, as proposed e.g. in [25], or [84]. This approach has three main disadvantages: Firstly, the computational complexity is relatively high compared to other clustering algorithms. Secondly, several parameters have to be
Audio Features

Figure 5.11: Source-filter model for the time domain: Envelopes of one separated component are described by a convolution of a source signal encoding tempo, rhythm, and amplitudes and an instrument specific temporal envelope.

defined properly, mainly the length of the characteristic envelopes. Finally, the transfer from the factorization to the clustering is not as straightforward as for the k-means or the NMF-clustering. Other approaches, utilized in [83], rely on the segmentation of the columns of the matrix $G$ in two different types of regions: Regions corresponding to the characteristic envelopes and silence. After the segmentation, the characteristic envelopes are clustered into $M$ different clusters. All of these clustering strategies are very sensitive to the initial segmentation.

A more robust and simpler algorithm to analyze the columns of matrix $G$ is based on the SFM, see also [85]. In the introduction of this section, the SFM for the frequency-axis of the tensor $Y_i$ is explained in detail. Here, the SFM for the temporal axis of the tensor $Y_i$ is described shortly. The mathematics of the underlying SFM are the same. Therefore, only the model itself is described here. The instrument characteristic note-envelope is convolved with a vector containing the information tempo, rhythm, and amplitudes. The result of this convolution describes a column of matrix $G$ in a sufficient way.

In [14], or other publications about convolutive NMF/NTF, the instrument specific envelope is utilized for extracting basis functions that are convolved with an activation vector/matrix. Here, we will use the convolution as shown in Figure 5.11 as a basis for a temporal-based SFM. Analogous to the frequency-based SFM, we evaluate the features for the temporal-based SFM by

$$F_{SFM,G}^{(0)}(k, i) = |DFT(G(t, i))|, \ 0 \leq k < T/2 , \ (5.13)$$

$$F_{SFM,G}^{(1)}(k_{filter}, i) = \sum_{k=1}^{T/2} R_{SFM,G}(k_{filter}, k)F_{SFM,G}^{(0)}(k, i) , \text{ and} \ (5.14)$$

$$F_{SFM,G}^{(2)}(k_{filter}, i) = 20 \log_{10} \left( f_{SFM,G}F_{SFM,G}^{(1)}(k_{filter}, i) + 1 \right) . \ (5.15)$$

The DFT used in Equation 5.13 corresponds to the DFT as described in Equation 2.11 with dropping the symmetric second half of the spectrum. Similar to the evaluation of the MFCC, a filter bank is applied by the multiplication with matrix $R_{SFM,G}$. Regarding the frequency based SFM, the number of filters (rows) in matrix $R_{MFCC}$ is set to a significant smaller value than the number of frequency bins (columns). This smoothing of the spectrum is done in order to get rid of the source signal and retaining only the filter signal
(which is assumed to be much smoother than the source signal). On the other hand, the source signal contains important informations like tempo and rhythm in the case of the temporal SFM. Therefore, matrix $R_{SFM,G}$ is a square matrix throughout this thesis. For cepstral analysis (separation of source- and filter-signal), a DFT can be applied on the columns of $F_{SFM,G}^{(2)}$:

$$F_{SFM,G}^{(3)} = \text{DFT} \left( F_{SFM,G}^{(2)} \right).$$  \hspace{1cm} (5.16)

Motivated by [76], the more general frequency-analysis by DFT is chosen for filtering the features $F_{SFM,G}^{(2)}$ instead of the DCT, which is motivated by decorrelation of the different feature-dimensions. As will be discussed in Section 6.2.1, the final frequency analysis is not useful in the context of clustering for the proposed BSS-framework. Therefore, features $F_{SFM,G}^{(2)}$ will be used for clustering in the following.

For deeper insights regarding the temporal SFM, the influence of the signal processing steps defined in Equations (5.13)–(5.15) is shown in Figure 5.12 for three simple examples. For the left figures, it is assumed that the columns of $G$ contain only a single note instance. The difference between the clusters is only defined by the instrument specific envelope, as shown in Figure 5.11(a). In this case, the combination of applying the DFT and taking the absolute value corresponds to converting the signal into a shift invariant feature. The final logarithm compresses the dynamic range of the features for better handling in the clustering algorithm.

In the middle column it is assumed that the envelopes have identical single note instances but different tempo. In this case, the information tempo propagates through the feature evaluation steps, and the two clusters can easily be identified.

For a third example in the right column the distinction between a signal with time-varying pitch and a signal with constant pitch is shown. The corresponding spectrogram is shown in Figure 5.13. The signal with time-varying pitch is separated into several components by the NTF, as can be seen clearly in Figure 5.12(c). The time-varying pitch results in columns of matrix $G$ that are shifted by a certain amount of temporal slots. Therefore, the shift-invariance of the temporal SFM features can be used to distinguish between such signals.

In this section, the two dimensions of a spectrogram are analysed by an SFM as shown in Figure 5.14. The third dimension of tensor $X$ the spatial dimension is not analysed by a SFM because in our scenario no more than $C = 2$ channels (stereo) are analysed. For such a short analysis length, the cepstral analysis does not make sense.

### 5.3 Frequency-based Features

Frequency-based features can be extracted from the spectrogram $Y$, as shown in Section 5.3.1, or directly from the matrix $B$, as shown in Section 5.3.2.

#### 5.3.1 Frequency Features based on Spectrogram $Y$

The basic model of spectra for harmonic audio features is shown in Figure 5.15. Most of the signal’s energy is located at integer multiples of a fundamental frequency, the pitch.
Although it is possible to distinguish between the fundamental frequency and the pitch, throughout this thesis both terms are used synonymously. Therefore, the fundamental frequency is the basic information for evaluation of all harmonic features. Obviously, every harmonic feature is very sensitive to wrong classification of the pitch. The following definitions shall be helpful for describing the harmonic features: In the following, a harmonic spectrum with fundamental frequency (pitch) $f_0$ is assumed. $k_0$ being the discrete frequency-bin corresponding to the pitch, $k_l$ ($1 \leq l \leq L$) being the frequency bin of the $l$-th partial (which is not necessary equal to $(l + 1) \cdot k_0$, see also Figure 5.15). Instead of multiplying the pitch with $(l + 1)$ the surrounding of this position is searched for the maximum to find the exact position of the $l$-th partial. The corresponding amplitudes of the $l$-th partial in spectrogram frame $t$ and each channel $c$ are $Y(k_l, t, c)$. 

**Figure 5.12:** Influence of the evaluation steps shown in Equation (5.13)–(5.15). The left figures show a scenario with two different single note instances and identical tempo (one note instance per column). The middle figures show a scenario with identical single note instances but different tempo. The right figures show the evaluated features for a mixture of one signal with time-varying pitch, and one signal with constant pitch.
5.3 Frequency-based Features

Figure 5.13: Spectrogram of a mixture of a chirp and a harmonic signal with constant pitch.

Figure 5.14: The both proposed schemes for cepstral analysis utilize identical processing steps to analyse either the horizontal axis or the vertical axis of the spectrogram.

As suggested in [86], the number of partials considered for the features based on the fundamental frequency is set to $L = 20$.

**Fundamental Frequency**

The pitch $f_0$ of an instrument is often used as an audio feature, e.g. in [75], or [82]. In this thesis, the YIN algorithm is used for the estimation of the fundamental frequency $F_f^{(0)}(i)$ of channel $i$. For further details of this algorithm, see also [87].
Audio Features

Figure 5.15: Spectrum of a piano note with fundamental frequency at $f_0 = 123$ Hertz. Two things are observable: Firstly, the signal energy usually decreases to higher frequencies, but it is not mandatory that the partial at $f_0$ has the highest amplitude. Secondly, for higher frequencies, the position of the harmonic may differ from the integer multiples of $f_0$, which are marked with dotted lines here.

Average Harmonic Structure

In Figure 5.15, it can be seen that in general, higher frequencies have lower signal energies. For the spectrum of the given piano note, the peak at frequency $f_0$ is not the maximum of the spectrum. In [35, pp.438] harmonic spectra are clustered into three groups: Spectra either have

- nearly constant amplitudes of the harmonic peaks (e.g. the violin),
- decreasing amplitudes of the harmonic peaks (e.g. the flute), or
- the maximum amplitude is not at the position of the pitch (e.g. the oboe).

The same model is used in [86]: The relative heights of the partials are used for source separation. This feature is called average harmonic structure. The underlying model is that the relative amplitudes of the harmonic peaks of the spectrum for a single instrument is constant, regardless of the pitch. This model is also called harmonic representation (HR) in [15].

According to [88], the harmonic amplitudes are normalized to the amplitude of the pitch and scaled logarithmically. With this, the $L$-th dimensional feature HR is evaluated by:

$$F^{(0)}_{\text{HR}}(l, i, t, c) = \max \left( 0, 20 \log_{10} \left( 10^4 \cdot \frac{Y(k_l, t, c)}{Y(k_0, t, c)} \right) \right), \text{ with } 1 \leq l \leq L$$

$$F^{(1)}_{\text{HR}}(l, i) = \frac{1}{T \cdot C} \sum_{t, c} F^{(0)}_{\text{HR}}(l, i, t, c).$$

The multiplicative factor $10^4$ in Equation (5.17) is used for reducing the impact of the $\max (.)$ operation. For further details on the influence of this scaling, see also Figure 5.10.
5.3 Frequency-based Features

Spectral Centroids

In the following, we introduce a set of spectral features, independent from the estimation of \( f_0 \): The spectral centroid. It is nothing else, than the barycenter of the spectrum over a logarithmic frequency-axis. In [75], this barycenter is normalized to the signals pitch, but we have found the non-normalized centroid leading to better separation quality:

\[
F_{SC}^{(0)}(t,c,i) = \frac{\sum_k \log_2 \left( \frac{\text{freq}(k) + 1}{440K} \right) Y(k,t,c)}{\sum_k Y(k,t,c)}, \quad (5.19)
\]

\[
F_{SC}^{(1)}(1,i) = \frac{1}{T \cdot C} \sum_{c,t} F_{SC}^{(0)}(t,c,i) \quad \text{(mean),} \quad (5.20)
\]

\[
F_{SC}^{(1)}(2,i) = \max_{t,c} \left( F_{SC}^{(0)}(t,c,i) \right) \quad \text{(maximum), and} \quad (5.21)
\]

\[
F_{SC}^{(1)}(3,i) = \sqrt{\frac{1}{T \cdot C} \sum_{t,c} \left( F_{SC}^{(0)}(t,c,i) - F_{SC}^{(1)}(1,i) \right)^2} \quad \text{(variance).} \quad (5.22)
\]

The spectral centroid can also be evaluated on the basis of matrix \( B \), but in [75], it is proposed to interpret not only the mean spectral centroid, but also the variance, which is only available, if the spectral centroid is evaluated on the basis of the spectrograms \( Y_i \).

5.3.2 Frequency Features based on Matrix B

Another set of features are evaluated using only matrix \( B \) factorized by the NTF. Obviously, these features are less complex to evaluate regarding the computational load.

Partial Features

In [89] and [82], a set of harmonic features, called partial features, are introduced: Brightness describes the barycenter of the harmonic amplitudes. The tristimulus parameters are introduced analogously to color-spaces. These three tristimulus-parameters summarize the harmonics to a three-dimensional harmonic-space. According to [64] and [82], certain instruments have lower signal energy in the odd harmonics compared to the even harmonics. The odd to even ratio shall distinguish between these types of instruments. The irregularity describes the variations between neighbouring harmonics. By this feature, instruments with nearly constant harmonic amplitudes can be distinguished from instruments with a strong decay over the partial amplitudes. The inharmonicity measures the differences between the exact partial position and the corresponding integer multiple of the pitch. Note that the feature inharmonicity is the only feature of the temporal- and spectral features that can become negative. Therefore, the absolute value is used as a measure of the deviation from a pure harmonic sounds.
These features are evaluated by

\[ F^{(0)}_{\text{partial}}(1, i) = \frac{\sum_{l=0}^{L-1} (l+1) B(k_l, i)}{\sum_{l=0}^{L-1} B(k_l, i)} \text{ (brightness)}, \quad (5.24) \]

\[ F^{(0)}_{\text{partial}}(2, i) = \frac{B(k_0, i)}{\sum_{l=0}^{L} B(k_l, i)} \text{ (tristimulus 1),} \quad (5.25) \]

\[ F^{(0)}_{\text{partial}}(3, i) = \frac{\sum_{l=1}^{3} B(k_l, i)}{\sum_{l=0}^{L} B(k_l, i)} \text{ (tristimulus 2),} \quad (5.26) \]

\[ F^{(0)}_{\text{partial}}(4, i) = \frac{\sum_{l=4}^{L} B(k_l, i)}{\sum_{l=0}^{L} B(k_l, i)} \text{ (tristimulus 3),} \quad (5.27) \]

\[ F^{(0)}_{\text{partial}}(5, i) = \frac{\sum_{l=0}^{L/2} B(k_{2l+1}, i)}{\sum_{l=0}^{L/2} B(k_{2l}, i)} \text{ (odd to even part ratio),} \quad (5.28) \]

\[ F^{(0)}_{\text{partial}}(6, i) = \frac{\sum_{l=0}^{L-1} (B(k_l, i) - B(k_{l+1}, i))^2}{\sum_{l=0}^{L-1} B^2(k_l, i)} \text{ (irregularity), and} \quad (5.29) \]

\[ F^{(0)}_{\text{partial}}(7, i) = \frac{1}{L} \left| \sum_{l=1}^{L} \left( \frac{k_l}{(l+1)k_0} \right)^2 - 1 \right| \text{ (inharmonicity).} \quad (5.30) \]

**Spectral Slope**

The spectral slope is the linear fitting of the double-logarithmically scaled spectrum, see also [82]. The spectral slope can be evaluated by

\[ x(k) = \log_2 \left( \frac{k \cdot F_s}{440 \cdot K} + 1 \right), \quad (5.31) \]

\[ y(k, i) = 20 \log_{10} (B(k, i)) \],

\[ \min \left\{ \sum_k (\mu_1 x(k) + \mu_2 - y(k, i))^2 \right\}, \quad (5.33) \]

with \( \mu_1 \) and \( \mu_2 \) being the parameters of a linear polynomial used for fitting. To minimize Equation (5.33), we solve the following linear system of equations:

\[ \begin{pmatrix} \sum_k x^2(k) & \sum_k x(k) \\ \sum_k x(k) & \sum_k 1 \end{pmatrix} \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} = \begin{pmatrix} \sum_k y(k)x(k) \\ \sum_k y(k) \end{pmatrix}. \quad (5.34) \]

Obviously, the offset \( \mu_2 \) is only a measure of the signal’s energy. Therefore, the relevant feature for the spectral slope is set to

\[ F^{(0)}_{\text{ssl}}(i) = \mu_1. \quad (5.35) \]

**Audio Spectral Flatness**

Audio spectral flatness is a low level descriptor of the MPEG-7 standard as described e.g. in [34]. Basically, it consists of a division of the geometrical mean by the arithmetic mean.
It distinguishes between noisy or harmonic spectra. For this, a set of frequency-bands are defined that are spaced logarithmically over the frequency-axis. The number of frequency-bands is set to $K$, and the frequency limits are defined as $k_\kappa(1)$ (lower limit) and $k_\kappa(2)$ (upper limit), with $1 \leq \kappa \leq K$. For the exact description of $k_\kappa(1)$ and $k_\kappa(2)$ please take a closer look on [34]. With these limits, the audio spectrum flatness is described by

$$
\Delta k_\kappa = k_\kappa(2) - k_\kappa(1),
$$

$$
F_{\text{ASF}}^{(0)}(\kappa, i) = \left( \prod_{k=k_\kappa(1)}^{k_\kappa(2)} B^2(k, i) \right)^{1/\Delta k_\kappa},
$$

$$
F_{\text{ASF}}^{(1)}(i) = \frac{1}{K} \sum_\kappa F_{\text{ASF}}^{(0)}(\kappa, i).
$$

5.4 Temporal-based Features

In [75], a set of temporal features regarding the onsets of note events is proposed for musical instrument recognition. For the reasons mentioned in the introduction of this chapter, no segmentation in transient (or onset) and steady state is done. Additionally, in [77] it is shown that the temporal features like duration of transient and steady state segment, relative slope of steady state, and similar features describing the envelope of the single note events, did not perform very well. Therefore, they are not further taken into account, and more details are given for the more promising features.

Zero Crossing Rate

The feature zero crossing rate (zcr) is a typical feature used in audio signal processing, e.g. in [82]. The zcr measures the instantaneous frequency of a monophonic signal. It can be used as a simple indicator if the signal is more like noise or more harmonic: Generally, noise and high frequency sinusoids have a high zcr. Signals with low frequencies have a small zcr. It is evaluated by

$$
F_{\text{zcr}}^{(0)}(i) = \frac{1}{(N-1) C} \sum_{n,c} \varepsilon (-y_i(n, c) \cdot y_i(n-1, c)),
$$

with $\varepsilon(\cdot)$ defined in Equation (2.2).

Note Activity

The feature note activity (NA) describes the columns of matrix $G$ by two states, note is played and silence, similar to a Hidden Markov Model with two states. As proposed in [83], the distinction between a played note (state 1) and silence (state 0) is done by a threshold defined by the mean value of the corresponding column of matrix $G$

$$
\text{state}(t, i) = \varepsilon \left( G(t, i) - \frac{1}{T} \sum_l G(l, i) \right),
$$
with $\varepsilon(\cdot)$ being defined in Equation (2.2). With this simple state-model, we can describe the note activity by the expected value of the transitions between both states:

$$
F^{(0)}_{\text{NA}}(1, i) = \frac{\sum_t \delta_{1, \text{state}(t, i)} \cdot \delta_{0, \text{state}(t-1, i)}}{\sum_t \delta_{0, \text{state}(t, i)}},
$$

$$
F^{(0)}_{\text{NA}}(2, i) = \frac{\sum_t \delta_{0, \text{state}(t, i)} \cdot \delta_{1, \text{state}(t-1, i)}}{\sum_t \delta_{1, \text{state}(t, i)}},
$$

with $\delta_{x,y}$ being the Kronecker symbol as defined in Equation (2.3). Additionally, we express the tempo (measured in note events per second) of a separated component by

$$
F^{(0)}_{\text{NA}}(3, i) = \frac{F_s}{h_s \cdot T} \sum_t \delta_{1, \text{state}(t, i)} \cdot \delta_{0, \text{state}(t-1, i)}.
$$

### Crest Factor

Crest factor is used e.g. in [75]. It resembles the sparseness of an envelope, and is evaluated by

$$
F^{(0)}_{\text{crest}}(i) = \frac{\max_t G(t, i)}{\sqrt{\frac{1}{T} \sum_t G^2(t, i)}},
$$

with $1 \leq F^{(0)}_{\text{crest}}(i) \leq \sqrt{T}$. For non-negative signals, the minimum crest factor corresponds to a constant signal. The maximum crest factor corresponds to a signal which has only one non-zero element.

### Amplitude Modulation

In [75], the amplitude modulation (AM) in the interval 8–10 Hertz and in the interval 10–40 Hertz are interpreted as useful features for musical instrument recognition. To evaluate the frequency and the energy of this feature, we apply the DFT on the columns of matrix $G$ and evaluate the power spectrum:

$$
F^{(0)}_{\text{AM}}(k, i) = |\text{DFT}(G(t, i))|^2, \text{ with } 0 \leq k < \tilde{T}.
$$

$\tilde{T}$ is the transform length of the DFT, which can be set by the user due to zero-padding. $\tilde{T}$ is assumed to be free of a measurement unit in the following.

The temporal axis of the spectrogram (corresponding to time-index $t$) is sampled with hop size distances, thus the sampling frequency $f_G$ of the columns of matrix $G$ can be described by

$$
F_G = \frac{F_s}{h_s} \text{ (in Hertz)}. \quad (5.46)
$$

To evaluate the amplitude modulation in the above given frequency intervals, we need an integer number of frequency-bins to resemble the smallest interval (2 Hertz) as good as possible. The frequency resolution of the DFT can be described by sampling frequency divided by transform length:

$$
\Delta f_G = \frac{F_G}{\tilde{T}}. \quad (5.47)
$$
Thus, the transform length $\hat{T}$ of the DFT in Equation (5.45) shall be set such that

$$2 \text{ Hertz} \approx n \Delta f_G \Rightarrow \hat{T} \approx \frac{n \cdot F_G}{2 \text{ Hertz}}, \quad (5.48)$$

with $n$ being an integer value. In the following, the frequency-bins corresponding to the frequency intervals mentioned above can be evaluated as shown for the example of 8 Hertz:

$$k_{8 \text{ Hertz}} = \frac{8 \text{ Hertz}}{\Delta f_G}. \quad (5.49)$$

Because of time-discrete signal processing, this value must be rounded. Then, the strength of AM in the interval 8–10 can be evaluated by

$$F_{AM}^{(1)}(1, i) = \frac{\sum_{k=k_{8 \text{ Hertz}}}^{k_{10 \text{ Hertz}}} F_{AM}^{(0)}(k, i)}{\sum_{k=0}^{\hat{T}/2} F_{AM}^{(0)}(k, i)}. \quad (5.50)$$

If there is no local maximum in the given interval, $F_{AM}^{(1)}(1, i)$ is set to zero, because a definition of signal energy in this frequency interval can only be interpreted as AM in the case of a local maximum. $F_{AM}^{(1)}(2, i)$ is evaluated analogously for the frequency interval 10–40 Hertz.

The features $F_{AM}^{(1)}(3, i)$ and $F_{AM}^{(1)}(4, i)$ are the exact frequency positions of the maximum in the corresponding intervals. If no local maximum is present, then the corresponding frequencies are also set to zero.

### 5.5 Spatial-based Features

Analogously to the spectral- and temporal-based features, now a closer look is taken on the third dimension of the tensor $X$: The spatial position of each note. First, the disadvantages (−) followed by the advantages (+) are summarized:

− The spatial position and by this the spatial features can change during recording session, e.g. by surround effects invoked by the sound engineer or by movement of the source.
− The authors of [14] mentioned that the coefficients in matrix $A$ evaluated by the NTF are not stable enough for the purpose of clustering even in the case of using an instantaneous mixing matrix for creating the mixtures.

+ Spatial features are the only information available for clustering in the case of recording a set of sources, being identical regarding their spectral- and temporal features, but being clearly separated in spatial domain. An example is the recording of four human speakers positioned around the microphone position.

### Instantaneous Spatial Model

The simplest spatial feature is defined by the columns of matrix $A$. According to [54], the data model of the columns of matrix $A$ is the radial model, as shown in Figure 5.2(b).
The authors of [54] suggest a clustering of each time-frequency cell in a tensor \( \mathbf{X}(k, t, c) \) by the relative amplitudes and the delay between both channels in the case of stereo signals. First, a closer look on the amplitude feature is taken, which corresponds to an instantaneous mixture model.

The feature used in [54] can be described by

\[
\mathbf{F}_{\text{inst}}^{(0)}(k, t) = \frac{Y(k, t, 2)}{Y(k, t, 1)} . \quad (5.51)
\]

The advantage of this feature is that it works independent from the NTF, and can surpass (theoretically) the separation quality of NTF-based BSS frameworks: The clustering is done for each time-frequency cell of the tensor \( Y(k, t, c) \) and not for each component separated by NTF. The main disadvantage for this feature is that it becomes unstable for small amplitudes in the first channel \((c = 1)\). Therefore, we will not use this feature in the following. Instead, a \( C \) dimensional feature is used for avoiding this numerical problem:

\[
\mathbf{F}_{\text{inst}}^{(1)}(c, i) = \mathbf{A}(c, i) . \quad (5.52)
\]

### Convolutive Model

In [27], the transfer path between a point source and a microphone is modeled by a multiplication over the frequency-axis in spectrogram domain.

The five mixtures of data \( \mathbf{B} \) are the only mixtures recorded under realistic mixing conditions regarding the audio data used in this thesis. With only five mixtures recorded under realistic recording situations no universally valid statement can be given about the validity of the instantaneous mixing model for the NTF. Of course, if echoes with high delays are used for recording, the instantaneous mixing model is unrealistic even for magnitude spectrograms with relative coarse time-resolution as in our scenario. In Figure 4.10, it is shown that the instantaneous mixing model generally works best for data \( \mathbf{B} \). Here, features are introduced to utilize the convolutive mixing models even in a framework with NTF based on instantaneous mixing models.

To utilize convolutive filters for clustering it is possible to extract the convolutive filters according to the point-source model utilized in [27] afterwards. For this, we define a matrix \( \mathbf{\tilde{A}} \) of size \( K \times I \times C \). \( \mathbf{\tilde{A}} \) is initialized by

\[
\mathbf{\tilde{A}}(k, i, c) = \mathbf{A}(i, c) . \quad (5.53)
\]

For the convolutive data model \( \mathbf{\tilde{X}}(k, t, c) \) is defined as

\[
\mathbf{\tilde{X}}(k, t, c) = \sum_{i=1}^{I} \mathbf{\tilde{A}}(k, i, c) \mathbf{B}(k, i) \mathbf{G}(t, i) . \quad (5.54)
\]

Applying the update-rule in Equation (B.5) several times with \( \mathbf{B} \) and \( \mathbf{G} \) being fixed leads to an approximation of the convolutive filters modeling the transfer path between the different sources and the microphones.

These filters can be used as features for clustering. Different scalings and normalizations can be used in combination with these features. Here, only a double-logarithmic scaling
of the frequency- and the amplitude-axis is considered, motivated by the MFCC’s. The final features used for spatial clustering can be described by

\[
F^{(0)}_{\text{conv}}(k, i, c) = \tilde{A}(k, i, c) \quad (5.55)
\]

\[
F^{(1)}_{\text{conv}}(k_{\text{mel}}, i, c) = \sum_k R_{\text{conv}}(k_{\text{mel}}, k) F^{(0)}_{\text{conv}}(k, i, c) \quad (5.56)
\]

\[
F^{(2)}_{\text{conv}}(k_{\text{mel}}, i, c) = 20 \log_{10} \left( f_{\text{conv}} F^{(1)}_{\text{conv}}(k_{\text{mel}}, i, c) \right) \quad (5.57)
\]

If not otherwise mentioned, \( f_{\text{conv}} \) scales matrix \( F^{(2)}_{\text{conv}} \) to a maximum of 60 dB. \( R_{\text{conv}} \) corresponds to the mel filter bank.

For clustering, the final feature matrix is a concatenation of all \( C \) feature matrices evaluated for each channel \( c \).

### 5.5.1 Summary

In this chapter audio features are introduced. After a first overview over audio features, two important pre-processing steps are introduced: Normalization and dimension reduction. Finally, four classes of audio features are discussed: Cepstral, frequency-based, temporal-based, and spatial-based features.
Chapter 6

Unsupervised Note Clustering

As explained for the separation framework introduced in Chapter 4, the mixture is separated into acoustical events, which have to be clustered into groups according to the active instruments/sound sources. The reference clustering step introduced so far is a non-blind clustering algorithm. Therefore, it cannot be applied in a BSS framework. Instead, for blind source separation, an unsupervised clustering algorithm is necessary.

The audio features discussed in this thesis are introduced in Chapter 5. For final signal synthesis, a clustering vector $a(i)$ is necessary, according to Equation (4.13). In this chapter, the necessary signal processing steps and the corresponding experimental results will be discussed.

This chapter starts with the introduction of unsupervised clustering algorithms, in Section 6.1. After that, in Section 6.2, possible combinations of features and clustering algorithms are used to evaluate the separation quality for blind separation scenarios. The results are summarized in Section 6.3.

6.1 Clustering Algorithms

This section gives an overview over the clustering algorithms used in this thesis. If not otherwise mentioned, the algorithms are explained with more details in [7]. Generally, clustering is the task of grouping a set of multidimensional samples $F(n, i)$ in a certain number of clusters $M$. The $I$ columns of matrix $F$ contain the $N$-dimensional feature-samples. In the following, the term distance function will not be specified to a certain distance function to make the description of the algorithms as general as possible. If a distance function is defined, this is only done for giving a simple example.

As in Section 4.1, clustering is defined by a vector $a$. The single elements of vector $a$ are denoted by the indices $i$, $1 \leq i \leq I$. $a(i) = m$ is equivalent to the information that feature sample $i$ belongs to cluster $m$.

As mentioned in Chapter 4, the number of feature samples is very small in our scenario ($I \leq 25$), compared to the thousands of feature samples used in other clustering algorithms, e.g. in [49]. The small number of features is the result of avoiding a training step on a large set of audio data and to use only the features evaluated for the given mixture for clustering. This induces a set of open problems to the clustering algorithms used in our context:
6.1 Clustering Algorithms

- A statistical interpretation of the sparse feature space is hardly possible.
- Outliers in the feature space will have critical influence.
- The initialization will be critical.

All of these problems will be discussed in the following.

According to the nomenclature in [7], we are mainly interested in Unsupervised Learning and Clustering. In the following, we will introduce a set of clustering algorithms structured as follows:

- **Partitional Clustering Algorithms** try to cluster the whole data at once.
- **Hierarchical Clustering Algorithms** divide the problem of clustering into a set of (possible easier) subproblems. They are divided into top-down algorithms and bottom-up algorithms. Examples for both are discussed below.

Because the number of cluster $M$ is assumed to be known, we concentrate on algorithms with a fixed value for the number of clusters. E.g. the mean shift algorithm introduced in [90] can result in a different number of clusters, and is therefore not used here.

### 6.1.1 Partitional Clustering Algorithms

As mentioned above, partitional clustering algorithms aim at partitioning (clustering) the whole data at once. In the following, a set of partitional clustering algorithms are introduced, and their (dis-)advantages are discussed.

**Expectation Maximization**

The expectation maximization (EM) algorithm is a routine to estimate the parameters of an assumed probability density function (pdf) from a set of random samples. Here, the algorithm is explained from a clustering point of view. Iteratively it applies the two steps

- expectation (evaluate the expected class-affiliation of the samples) and
- maximization (maximize the parameters for each class according to the class-affiliation).

It is proven that the algorithm converges to a certain class-affiliation, depending on the initialization. In the following, the EM algorithm will be explained by a simple example:

For simplicity, a set of one-dimensional samples $F(i)$ is assumed to be given ($N = 1$). The pdf of the samples is modeled by a Gaussian mixture model (GMM) with two Gaussian distributions ($M = 2$) defined by the variances $\sigma^2_m$, the mean values $\mu_m$, and the weights $a_m$:

$$p_f(x) = \sum_{m=1}^{2} a_m \frac{1}{\sqrt{2\pi\sigma^2_m}} e^{-\frac{(x-\mu_m)^2}{2\sigma^2_m}}$$

with $a_1 + a_2 = 1$. The probability that a given class $m$ produces a given sample $F(i)$ is defined by

$$p(m|i) = \frac{a_m \frac{1}{\sqrt{2\pi\sigma^2_m}} e^{-\frac{(F(i)-\mu_m)^2}{2\sigma^2_m}}}{\sum_{l=1}^{2} a_l \frac{1}{\sqrt{2\pi\sigma^2_l}} e^{-\frac{(F(i)-\mu_l)^2}{2\sigma^2_l}}}.$$
Then, the algorithm is defined by:

- **Expectation**: Evaluate $p(m|i)$ for each class $m$ and each sample $i$.
- **Maximization**: Update the parameters $a_m$, $\sigma_m^2$, and $\mu_m$ for each class $m$:

$$
\mu_m = \frac{\sum_i p(m|i)F(i)}{\sum_i p(m|i)} \quad (6.3)
$$
$$
\sigma_m^2 = \frac{\sum_i p(m|i) (F(i) - \mu_m)^2}{\sum_i p(m|i)} \quad (6.4)
$$
$$
a_m = \frac{1}{N} \sum_i p(m|i) . \quad (6.5)
$$

The algorithm can be extended to arbitrary probability density functions in a simple way. The given example and the corresponding formulas are from the tutorial [91]. Because the approximated pdf usually consists of a linear combination of several simple pdf’s, the EM algorithm can be used for clustering. In this case, each of the single pdf’s corresponds to a single cluster. The cluster decision is equivalent to searching a maximum over the $p(m,i)$ for all classes $m$.

**Non-Full-Rank Covariance Matrices** For multivariate pdf’s the variances $\sigma_m^2$ become covariance matrices. The estimation of these covariance matrices can become critical, because they are only of full rank for large $I$. Therefore, the EM can only be applied in a meaningful manner for $I \gg N$. Additionally, the sparse feature space is generally ill-conditioned for estimating pdf’s, because the robust estimation of a given pdf usually needs lots of samples. Therefore, the EM algorithm will not be used in the following, but serves as a motivation for the k-means algorithm.

A possible workaround for the problem of non-invertible covariance matrices would be the evaluation of a *sparse* covariance matrix with only few elements unequal zero: The covariance matrix is evaluated as usual. For sparsification, the main diagonal and the largest values are kept, all other elements are set to zero. Setting no elements on the main diagonal to zero increases the possibility of stable matrix inversion, see also [2]. The small number of non-zero elements ensures stable estimation of the most important attributes of the covariance matrix. Nevertheless, the usage of the covariance matrix of a feature space with only $I \leq 25$ elements is beyond the scope of this thesis.

**Mahalanobis Distance** The Mahalanobis distance is based on the Gaussian model explained for the EM algorithm. Basically, it weights the different dimensions of the feature space by the inverse of the covariance matrix to evaluate the distance. As mentioned above, the covariance matrix and its inverse cannot be utilized in our framework, because our feature space consists of only $I \leq 25$ samples, which is not sufficient to estimate a covariance matrix of full rank. Therefore, the Mahalanobis distance is not used in the context of this thesis. For further details on the Mahalanobis distance please take a closer look into [5].
Dimension Reduction for Stable Covariance Estimation  
As mentioned above, the evaluation of the covariance matrix is only stable in the case of \( I \gg N \) (the number of feature samples have to be significant larger than the number of dimensions in this feature space). For the usage of the expectation maximization algorithm a dimension reduction algorithm can be applied on the feature space, to reduce the dimensionality \( N \) of the feature space to values smaller than \( I \). Unfortunately, it cannot be guaranteed that the loss of information by this dimension reduction is not critical for clustering. Therefore, this experiment is not shown here.

k-Means

The k-means algorithm is a simplified version of the EM algorithm. Usually, an unimodal and symmetric distribution of the feature samples around the mean values (centroids) \( \mu_m \) is assumed. Most commonly, the Gaussian distribution is assumed, but other distributions are also possible. The expectation step is replaced by a hard decision, to which class \( m \) each sample belongs:

\[
a(i) = \arg\min_m d(W(m), F(i)), \tag{6.6}
\]

with \( d(W(m), F(i)) \) being a distance function between the centroid, stored in the \( m \)-th column of \( W \) and the \( i \)-th feature sample stored in the \( i \)-th column of \( F \). The maximization step is simplified to the evaluation of a centroid for each class \( m \). With the hard decision of the expectation step, the maximization step simplifies to evaluation of the centroid only for the samples belonging to class \( m \). If not otherwise mentioned, the centroids are evaluated as

\[
W(n, m) = \frac{1}{\sum_{i=1}^{I} \delta_{m,a(i)}} \sum_{i=1}^{I} F(n, i) \delta_{m,a(i)}. \tag{6.7}
\]

Equivalent to the EM algorithm, convergence is guaranteed: The algorithm converged if vector \( a \) remains unchanged during the last iteration.

Fuzzy c-Means

The EM algorithm assigns each feature with a certain probability to each class \( m \). The k-means algorithm assigns each feature \( i \) hard to a single class \( m \). By this hard decision, the clustering becomes very sensitive to initialization and noisy feature samples. In [92], the hard decision of the k-means algorithm is replaced by a soft decision, which is the reason for naming the algorithm fuzzy. For this, the clustering information \( a(i) \) is replaced by a partition matrix \( H \) of size \( I \times M \). The following Equations are from [93]. The final clustering decision for our BSS-framework can be evaluated by

\[
a(i) = \arg\max_m H(i, m). \tag{6.8}
\]

The cluster centroids are updated by

\[
W(n, m) = \frac{\sum_{i=1}^{I} H(i, m) F(n, i)}{\sum_{i=1}^{I} H(i, m)}, \tag{6.9}
\]
which is nothing else than the mean over each dimension weighted with the corresponding cluster affiliation. The parameter $f > 1$ controls the fuzziness of the clustering. If not otherwise mentioned, throughout this thesis, $f = 2$ is used. The cluster affiliation is updated by

$$H(i, m) = \left( \frac{1}{d(W(m), F(i))} \right)^{2/(f-1)},$$

with $d(x, y)$ being an appropriate distance function as discussed in Section 2.1.1. Finally, the matrix $H$ is normalized such that the cluster affiliation sums up to one for each sample:

$$\sum_{m=1}^{M} H(i, m) = 1.$$  

By this normalization, the entries of the partition matrix can be interpreted as probabilities.

Obviously, for $H(i, m) \in \{0, 1\}$, the fuzzy c-means (FCM) is equivalent to the k-means algorithm. With $f \to 1$, the combination of Equation (6.10) and normalization in Equation (6.11) tends towards this hard clustering. Therefore, for $f \to 1$, the fuzziness becomes smaller, and for $f \to \infty$, the fuzziness increases.

### NTF-Clustering

In [94], the NTF\(^1\) is explained from a clustering point of view. The equivalence between k-means and a symmetric variant of the NTF

$$D(i_1, i_2) \approx \sum_{m=1}^{M} H(i_1, m)H(i_2, m)$$  

is shown, with $D$ being a symmetric distance matrix for a given set of features, e.g. the standard inner-product

$$D = F^T F.$$  

For this variant of the NTF, the columns of $H$ have equal norm.

From this starting point, the authors of [94] explain the similarity between bipartite graph k-means clustering and NTF: Bipartite graph k-means clustering is explained as simultaneously clustering the rows and the columns of a matrix $F$. This is similar to the SVD, where simultaneously a set of output basis vectors (matrix $U$ in Equation (2.30)) and a set of input basis vectors (matrix $V$ in Equation (2.30)) is evaluated. In [94], it is shown that also with non-negativity constraints, the clustering is guaranteed. The non-symmetric standard NTF is used for clustering e.g. in [31]:

$$F(n, i) \approx \sum_{m=1}^{M} W(n, m)H(i, m).$$

Compared to the k-means algorithm, this clustering can be interpreted as follows: The centroids are the $M$ columns of matrix $W$. The matrix $H$ encodes the cluster affiliation.

\(^1\)Although the paper is about the NMF, here the term NTF is used for consistency.
Therefore, the clustering vector $a(i)$ is defined as for the FCM in Equation (6.8).

As a matter of fact, Equation (6.8) is only true, if the columns of $W$ and $H$ are normalized as described in Section 2.3.2: As mentioned in e.g. [71], the columns of $B$ and $G$ are scale-invariant: The following approximations are identical:

$$F(n, i) \approx \sum_{m=1}^{M} W(n, m)H(i, m) = \sum_{m=1}^{M} \left( \lambda_m \overline{W}(n, m) \right) \cdot \left( \frac{1}{\lambda_m} \overline{H}(i, m) \right), \quad (6.15)$$

with $\lambda_m \in \mathbb{R}$ being an arbitrary scaling factor. If no normalization is defined for the columns of both matrices, the clustering decision becomes dependent from the (arbitrary) value $\lambda_m$: If $\lambda_1$ grows to infinity, all feature samples are clustered into cluster $m = 1$. Therefore, normalization is necessary for NTF used in the context of clustering.

**On the Similarity between k-Means and NTF-Clustering** The k-means algorithm assigns each feature vector to the centroid that minimizes the distance between feature vectors $F$ and centroids $W$. This minimization is done for all feature samples $i$ and results in the following cost-function $d_{k\text{-means}}$:

$$d_{k\text{-means}} = \sum_{i=1}^{I} \sum_{m=1}^{M} d_{\beta}(F(i), W(m)) \delta_{m,a(i)}, \quad (6.16)$$

with $\delta_{m,a(i)}$ being the Kronecker symbol defined in Equation (2.3). Replacing the hard clustering of the Kronecker symbol with a soft-clustering by a matrix $H(i, m)$ according to Equations (6.14) and (6.8), reveals the similarity between k-means and NTF-based clustering.

Both algorithms are only similar, if the NTF is used with the Euclidean distance $d_2$ according to Equation (2.5). For other values of $\beta$, the cost function used for clustering becomes asymmetric. The term *asymmetry* clarifies that for the given distance function, the centroids and the feature samples may not be exchanged without changing the distance. Additionally, the evaluation of the centroids cannot be explained by a simple mean-operation as for the k-means algorithm or for the NTF clustering with $\beta = 2$. In [73], this asymmetry is utilized for specializing the NTF-clustering to scenarios with large dynamic differences between the input signals or to scenarios with nearly equal loudness for both input signals. The conclusion of [31] and [73] is that generally the asymmetric cost functions work better, because the loudness differences between input signals is assumed to be unknown.

One major disadvantage of the NTF-clustering is the necessity of non-negative features for clustering. This has to be considered for certain feature processing steps, e.g. the NLM or the BCT.

**On the similarity between NTF-Clustering and Fuzzy C-Means** FCM combines the following advantages of the k-means and the NTF clustering:

- FCM share the property of the NTF-clustering algorithm of soft assignment of the feature samples to the different clusters.
• FCM clustering has the advantage of the usage of symmetric cost-functions.
• FCM is not restricted to non-negative features.

6.1.2 Hierarchical Clustering

Hierarchical clustering algorithms aim at constructing clusters either by a bottom-up or by a top-down scheme.

Bottom-Up

We will explain bottom-up clustering by the algorithm called agglomerative clustering. Agglomerative clustering is initialized by defining each sample as a single cluster. As long as the number of clusters is too large (> M), the pair of clusters with smallest distance is searched. These two clusters are merged together to form a new cluster.

In [7], four schemes are proposed to define the distance \( d \) between clusters \( m_1 \) and \( m_2 \):

- \( d_{\text{min}} \): the minimum distance between all elements of both clusters.
- \( d_{\text{max}} \): the maximum distance between all elements of both clusters.
- \( d_{\text{avg}} \): the average distance between all elements of both clusters.
- \( d_{\text{mean}} \): the distance between the mean vectors of both clusters.

The last distance depends on a proper definition of cluster centroids. This definition can be done in several ways (arithmetical mean, geometrical mean, or others). To avoid this additional degree of freedom, the last distance measure is not considered in this thesis. From the other three, the average-distance definition leads to the best results. Therefore, the experiments with agglomerative clustering are restricted to this cluster distance in the following.

One difference between agglomerative clustering and the proposed partitional clustering algorithms, is the capability of separating two clusters with a non-linear borderline, as shown in Figure 6.1. The agglomerative clustering is capable of partitioning a feature space with a non-linear borderline. K-means, NTF-clustering, and fuzzy c-means clustering result in a linear borderline between the cluster centroids.

Top-Down

The hierarchical clustering, proposed in [31], is an example for a top-down hierarchical clustering algorithm. Because at each iteration step a partitional clustering algorithm is invoked, this type of hierarchical clustering can be interpreted as a meta-clustering. The algorithm starts with all samples belonging to a single cluster. As long as the number of clusters is too small (< M), the algorithm iteratively selects a cluster for further subdivision according to a given criterion. In [31], the cluster corresponding to the highest signal energy among all clusters is chosen for further subdivision. This choice is motivated by the fact that for this cluster the probability is highest that it contains more than one final cluster\(^2\). The selected cluster is clustered into two new clusters, and the algorithm starts

\(^2\)As mentioned in [31], this strategy is based on the assumption of uncorrelated sources. In this case, the energies of the sources are added up leading to the above mentioned hierarchical strategy.
Figure 6.1: Feature samples from \( M = 2 \) classes. In the left figure, the result of the k-means clustering is shown. In the right figure, the result of the agglomerative clustering is shown. For both clusterings the borderlines are plotted for better visualization of the clustering. In the given scenario, the non-linear borderline of the hierarchical clustering leads to less wrong classifications.

again with the iterative partition. Any of these partitions into \emph{sub-clusters} can be done by arbitrary partitional clustering algorithms introduced above.

With the agglomerative clustering a clustering algorithm with non-linear borderline between the clusters is already tested in our framework. The top-down approach is not used in this thesis, because in the case of unknown dynamic differences between the mixed sources the partition of highest energy cluster can lead to classification errors that cannot be compensated in the following (hierarchical) clustering steps.

### 6.1.3 Initialization of Clustering Algorithms

![Signal flow of clustering with different (possible random) initializations](image)

Figure 6.2: Signal flow of clustering with different (possible random) initializations. A criterion evaluated after clustering is used in order to decide, which clustering was successful.

Generally, clustering algorithms are iterative algorithms, thus depending on the initialization of the corresponding set of parameters. As mentioned in [73], for less samples in the
feature space, the algorithms become more sensitive to the initialization procedure. The hierarchical clustering algorithms introduced above have deterministic starting conditions. Therefore, initialization is only critical for the partitional clustering algorithms mentioned above.

If not otherwise mentioned, all partitional clustering algorithms are initialized by random clustering \( a(i) \). Starting from this clustering the iterative clustering algorithm is applied until convergence. The clustering result is judged by an appropriate cost function or clustering criterion. This is repeated several times with different random initializations, and the clustering with best cost function/clustering criterion is chosen as final clustering decision, see also Figure 6.2. As mentioned above, only k-means, fuzzy C-means, and NTF-clustering are used as partitional clustering algorithms in the context of this thesis. For these algorithms, clustering criteria will be discussed.

As proposed in [23], the NTF can be initialized with random values and updated for a few number of iterations. The \( \beta \)-divergence is used as clustering/approximation criterion, and the approximation with smallest \( \beta \)-divergence is chosen:

\[
\text{crit} = d_\beta (F, WH^T) \tag{6.17}
\]

An alternative clustering criterion can be derived by the clustering decision described in Equation (6.8): A better clustering corresponds to nearly orthogonal clustering decisions. Therefore, orthogonality is used as a criterion:

\[
\text{crit} = \frac{1}{M^2} \sum_{m_1=1}^{M} \sum_{m_2=1}^{M} \frac{\sum_{i=1}^{I} H(i, m_1) \cdot H(i, m_2)}{\sqrt{\sum_{i=1}^{I} H^2(i, m_1)} \sqrt{\sum_{i=1}^{I} H^2(i, m_1)}}. \tag{6.18}
\]

The normalization leads to a range of the criterion of \( 0 \leq \text{crit} \leq 1 \). In the experiments, it was observed that this criterion based on cluster-orthogonality leads to much worse separation quality compared to the criterion based on the factorization error. Therefore, for NTF-clustering, only the criterion based on the factorization error is used in the following.

Instead of random initialization of matrices \( W \) and \( V \), the clustering in our context is initialized with a random clustering vector \( a(i) \). For each cluster \( m \), the cluster center (column of \( W \)) is evaluated by an NTF applied only on the columns of \( F \) that belong to the current class. Matrix \( H \) is initialized with ones.

For k-means or fuzzy C-means, a simple clustering criterion is proposed in [7]:

\[
\text{crit} = \sum_{m=1}^{M} \sum_{i_1=1}^{I} \sum_{i_2=1}^{I} \delta_{m, a(i_1)} \delta_{m, a(i_2)} d(F(i_1), F(i_2)), \tag{6.19}
\]

with \( d(x, y) \) being the distance function used for clustering, and \( F(i_{1,2}) \) being the \( i_{1,2} \)-th feature sample. Cluster affiliation is controlled by the Kronecker symbol. Similar to the cost-function of the NTF, small values correspond to better cluster reliability. The main advantage of this cluster reliability is that it only relies on a proper definition of the distance function, and is independent from any kind of data model.

A more complex example for such a criterion to judge the clustering quality is motivated
by the linear discriminant analysis (LDA) and is outlined e.g. in [5] or [7]. Firstly, the mean vectors and the samples without mean are defined by

\[
W(n, m) = \frac{1}{\sum_{i=1}^{I} \delta_{m,a(i)}} \sum_{i=1}^{I} F(n, i) \delta_{m,a(i)} \quad \text{(centroid for class } m) \tag{6.20}
\]

\[
\tilde{W}(n) = \sum_{m=1}^{M} \frac{\sum_{i=1}^{I} \delta_{m,a(i)}}{I} W(n, m) \quad \text{(centroid of the centroids)} \tag{6.21}
\]

\[
tmp_m(n, i) = F(n, i) - W(n, m) \quad \text{(feature samples with zero mean)} \tag{6.22}
\]

\[
tmp(n, m) = W(n, m) - \tilde{W}(n) \quad \text{(mean vectors without sample mean)} \tag{6.23}
\]

Second, the **intra-class scatter** and **inter-class scatter** are defined by

\[
C_m(n_1, n_2) = \sum_{i=1}^{I} \frac{\text{tmp}_m(n_1, i) \cdot \text{tmp}_m(n_2, i) \cdot \delta_{m,a(i)}}{\sum_{i=1}^{I} \delta_{m,a(i)}} \quad \text{(covariance for class } m) \tag{6.24}
\]

\[
C_{mm}(n_1, n_2) = \sum_{m=1}^{M} \frac{\sum_{i=1}^{I} \delta_{m,a(i)}}{I} C_m(n_1, n_2) \quad \text{(mean covariance of all } M \text{ classes)} \tag{6.25}
\]

\[
C_W(n_1, n_2) = \sum_{m=1}^{M} \frac{\sum_{i=1}^{I} \delta_{m,a(i)}}{I} \text{tmp}(n_1, m) \cdot \text{tmp}(n_2, m) \quad \text{(covariance of centroids)} \tag{6.26}
\]

With these covariance matrices, it is possible to define a criterion for the reliability of features for a clustering problem, based on the traces of the covariance-matrices:

\[
crit = \frac{\prod_n C_W(n,n)}{\prod_n C_{mm}(n,n)} \tag{6.27}
\]

Larger values correspond to a better configuration for clustering: Compact clusters, with centroids far away from each other. In [5, p.388], variants of the criterion in Equation (6.27) are proposed, e.g. based on the determinant of \(C_W\) and \(C_{mm}\) instead of the trace. It is observable that all of the variants in [5] lead to very similar results and nearly identical cluster decisions.

Because the definition of covariances, as used for the criterion in Equation (6.27), is sensitive to outliers, and the evaluation of full-rank covariances needs at least as many feature samples as dimensions in feature space, the criterion in Equation (6.27) is not used in the following.

### 6.2 Basic Clustering Algorithms

In this section, the features introduced in Chapter 5 are used as input for the clustering algorithms introduced in Section 6.1. As shown in [73], the mixing levels of the input signals \(s_m(n, c)\) can have major influence on the choice of good parameters for clustering. Therefore, we will evaluate the clustering results on two different mixing scenarios:
Two input signals with equal loudness, and two input signals mixed at ±12 dB dynamic difference:

\[
\text{dynamic difference} = \left| 20 \log_{10} \left( \sum_{n,c} s_1^2(n,c) \right) - 20 \log_{10} \left( \sum_{n,c} s_2^2(n,c) \right) \right|. \tag{6.28}
\]

The discussion of the clustering algorithms start with the cepstral features in Section 6.2.1. After that the clustering results for all other features are present in Section 6.2.2. Finally, the parameters used for clustering data \( \mathcal{A} \) are applied to data \( \mathcal{B} - \mathcal{D} \) in Section 6.2.3.

6.2.1 Cepstral Clustering

Cepstral analysis is the standard algorithm for instrument/speaker recognition or pitch shifting, as shown e.g. in [75], or [15]. On the other hand, cepstral analysis is usually mentioned as reference features for a clustering framework, e.g. in [34] or [31]. Therefore, the discussion of clustering algorithms starts with cepstral features.

Clustering with MFCC

We start with the clustering method proposed in [31]: The columns of matrix \( \mathbf{B} \) are the characteristic spectra of each separated acoustical event. According to Equations (5.10) and (5.11), the columns of \( \mathbf{B} \) are scaled in a double-logarithmic way to evaluate the features for clustering. These features are used for partitional clustering according to Section 6.1.1. Motivated by [31], the decorrelation step by the DCT is dropped for the first experiments. The DCT will be considered later in the context of dimension reduction. For the NTF-clustering a spherical data model has to be assumed. In the first experiment, we want to compare the different data-models. Therefore, the NTF-clustering is not considered in the first experiment.

<table>
<thead>
<tr>
<th>iter</th>
<th>k-means dm = 1</th>
<th>agglomerative dm = 1</th>
<th>FCM dm = 1</th>
<th>k-means dm = 2</th>
<th>agglomerative dm = 2</th>
<th>FCM dm = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7.98</td>
<td>8.25</td>
<td>8.32</td>
<td>7.69</td>
<td>7.79</td>
<td>8.10</td>
</tr>
<tr>
<td></td>
<td>8.07</td>
<td>8.25</td>
<td>8.37</td>
<td></td>
<td>8.01</td>
<td>7.54</td>
</tr>
<tr>
<td>10</td>
<td>8.28</td>
<td>8.25</td>
<td>8.34</td>
<td>8.01</td>
<td>7.79</td>
<td>8.10</td>
</tr>
<tr>
<td></td>
<td>8.31</td>
<td>8.25</td>
<td>8.39</td>
<td></td>
<td>7.56</td>
<td>7.54</td>
</tr>
</tbody>
</table>

Table 6.1: Separation quality for features \( \mathbf{F}_{\text{MFCC}}^{(1)} \): \( dm = 1 \) corresponds to spherical distance, \( dm = 2 \) to Euclidean distance.

Comparison of Data Models, Normalization Methods and Number of Iterations

Firstly, the underlying data-models and their corresponding normalization methods are compared: Unimodal or spherical. For clustering we use the following parameters: The filter bank has 20 filters, \( f_{\text{mfcc}} = \max_{k_{\text{mel}}} \left( \frac{\mathbf{F}_{\text{mfcc}}^{(1)}}{(1000 - 1)^3} \right) \), and all clustering algo-

\[\text{This normalization factor scales the dynamic range of } \mathbf{F}^{(2)} \text{ to } 60 \text{ dB.}\]
6.2 Basic Clustering Algorithms

Algorithms are applied with different number of iterations \( \text{iter} \) to minimize Equation (6.19). To compare the unimodal and the spherical model, we apply the k-means, the agglomerative clustering, and the FCM clustering with Euclidean and spherical distance on the data. Additionally, the influence of the appropriate normalization routine is taken into account. All results are shown in Table 6.1. It can be seen that the spherical model fits the data much better and results in better separation quality regarding all three clustering algorithms. These results are consistent with the results shown in [31]. In this paper, the superiority of the spherical data model is shown. Secondly, it can be seen that normalization has nearly no influence on the separation quality in the case of the spherical distance. For the Euclidean distance, the normalization corrupts the separation quality. Thirdly, it can be observed that a higher value for \( \text{iter} \) did not improve the separation quality significantly. This can be explained by the small number of feature samples \( I \): All criteria used for finding the best clustering are based on the assumption of a dense feature space. In this case, it is possible to evaluate the goodness of fit of the clustering according to the given feature samples. If only a few number of feature samples is present, the criteria are not useful, which is shown by this experiment. Finally, FCM leads to best separation quality, but the differences between the three proposed partitioning clustering algorithms are very small. Obviously, the (possible) non-linear borderline of the agglomerative clustering bears no advantage over the linear borderlines of the fuzzy c-means or the k-means algorithm.

Frequency- and Amplitude-Scaling by the Box-Cox-Transform

In a second experiment, a closer look on the double-logarithmic scaling is taken: The double-logarithmic scaling of spectra is also commonly used in literature for visualization of spectra. See also [64] for several visualizations of spectra in double logarithmic scaling. The question is, if this is the best achievable scaling for clustering in a BSS scenario? The logarithm is a special case of the Box-Cox transform (BCT), see also Equation (5.1). Or otherwise argued: The BCT is a generalization of the logarithm. Therefore, we use the BCT with different parameters \( a_\lambda \) (for scaling the amplitude-axis) and \( f_\lambda \) (for scaling the frequency-axis). The same is proposed e.g. in [95] for spectral analysis, but here, the influence of this scaling parameters on clustering for BSS is investigated.

\( a_\lambda \) modifies Equation (5.11) in the following way:

\[
F_{\text{MFCC}}^{(1)}(k_{\text{mel}}, i) = \text{BCT}_{a_\lambda} \left( f_{\text{MFCC}}(k_{\text{mel}}, i) + 1 \right),
\]

with \( \text{BCT}_{a_\lambda}(x) \) defined in Equation (5.1). \( f_\lambda \) modifies Equation (2.72) in the following way:

\[
f_{\text{mel}} = \text{BCT}_{f_\lambda} \left( f_{\text{Hertz}} \frac{700}{700} + 1 \right).
\]

The normalization constant 2595 in Equation (2.72) can be dropped, because the only condition for the mel filter bank is that the mel filters have identical distance in mel frequency domain. As a matter of fact, Equation (6.30) does not express the mel frequency mapping. For consistency to the definition of the applied filter bank, we use the term \textit{mel} in the following, even if the underlying frequency mapping is done with \( f_\lambda \neq 0 \). If \( f_\lambda \neq 0 \), the corresponding value of \( f_\lambda \) is given. In Figure 6.3, the influence of the BCT on the mel
Unsupervised Note Clustering

The filter bank is shown by four examples. Because of the underlying spherical data model verified in Table 6.1, the NTF-clustering

\[ f_\lambda = -1 \]

\[ f_\lambda = 0 \]

\[ f_\lambda = 1 \]

\[ f_\lambda = 2 \]

**Figure 6.3:** Influence of parameter \( f_\lambda \) on the filter bank \( R_{\text{MFCC}} \). \( f_\lambda = 1 \) corresponds to linear frequency mapping. For \( f_\lambda = 0 \), \( R_{\text{MFCC}} \) is a mel filter bank.

is applied on the features \( F_{\text{MFCC}}^{(1)} \). The corresponding results are shown in Table 6.2. Additionally, the influence of the Box-Cox transform is shown. For \( a_\lambda = f_\lambda = 0 \) the influence of clustering parameter \( \beta \) on the separation quality for different mixing levels can be seen. Smaller \( \beta \) are better suited for clustering mixtures with large dynamic differences, larger \( \beta \) are better for nearly equal loudness, as also discussed in [73]. The reason for this systematic behaviour is the sensitivity of the \( \beta \)-divergence to large amplitudes in the feature matrix for large values of parameter \( \beta \), as shown in Figure 2.1. Applying the Box-Cox transform on the feature space neutralizes this effect by scaling the amplitudes and the frequency-scaling in an appropriate way. Secondly, it can be observed that NTF-clustering, k-means, and agglomerative clustering cannot outperform FCM clustering for this set of audio data. Additionally, the FCM needs no non-negativity constraint for the
features and the distance between the centroids and the features is symmetric\(^4\). Therefore, only the FCM clustering algorithm is considered in the following.

### Influence of Dimension Reduction

Finally, the influence of dimension reduction on this feature space is checked. The evaluation of the MFCC is finished by a DCT according to Equation (5.12). Usually, the coefficients corresponding to higher frequencies are dropped, because of the underlying source-filter model (SFM). Thus dimension reduction is introduced in the evaluation routine of the standard MFCC. The motivation for the DCT is also the decorrelation of the feature space. As shown in Figure 5.2(b), the spherical distance measure is based on correlated features. Therefore, five types of dimension reduction algorithm are applied:

- Non-linear dimension reduction by NLM
- Linear dimension reduction by DFT, Low-pass, and inverse DFT
- Linear dimension reduction by DCT, Low-pass, clustered by spherical distance
- Linear dimension reduction by DCT, Low-pass, clustered by Euclidean distance
- Linear dimension reduction by DCT, Low-pass, normalized by unimodal model and clustered by Euclidean distance

The influence of the dimension reduction is shown in Figure 6.4. For each of the proposed scenarios five different values of the number of filter banks are evaluated: \(K_{\text{mel}} \in \{10, 15, 20, 25, 30\}\). These five values mark the starting points of each of the five lines plotted in each figure.

Best separation quality is achieved with dimension reduction based on the second scheme: DFT–Low-pass–inverse DFT shown in Figure 6.4(b). For this scheme, the separation quality increases up to 8.6 dB. Compared to the basic separation quality without dimension reduction of 8.55 dB, the improvement in separation quality is negligible. The only scenario with significant improvements based on the dimension reduction is the scheme based on the DCT, followed by clustering with the Euclidean distance after normalizing the features according to the unimodal data-model. The maximum of separation quality cannot surpass the scenarios with spherical clustering. Therefore, dimension reduction is not considered for the spectral SFM.

\(^4\)The centroids and the features can be exchanged for evaluating the distance between them as mentioned earlier.

---

<table>
<thead>
<tr>
<th></th>
<th>(f_\lambda = 0, a_\lambda = 0)</th>
<th>(f_\lambda) and (a_\lambda) variable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\pm 12\ dB\ 0\ dB\ mean)</td>
<td>(\pm 12\ dB\ 0\ dB\ mean)</td>
</tr>
<tr>
<td>k-means</td>
<td>7.73 8.23 7.98</td>
<td>-0.50 -0.50 7.91 8.27 8.09</td>
</tr>
<tr>
<td>agglomerative</td>
<td>8.15 8.35 8.25</td>
<td>0.00 0.00 8.15 8.35 8.25</td>
</tr>
<tr>
<td>FCM</td>
<td>8.14 8.51 8.32</td>
<td>-2.00 -0.50 8.29 8.80 8.55</td>
</tr>
<tr>
<td>NTF, (\beta = 1.0)</td>
<td>7.91 7.81 7.86</td>
<td>-1.50 0.00 8.15 8.63 8.39</td>
</tr>
<tr>
<td>NTF, (\beta = 1.5)</td>
<td>7.84 8.14 7.99</td>
<td>-2.00 0.00 8.05 8.82 8.43</td>
</tr>
<tr>
<td>NTF, (\beta = 2.0)</td>
<td>7.63 8.22 7.93</td>
<td>-2.00 -0.50 8.00 8.63 8.32</td>
</tr>
</tbody>
</table>

**Table 6.2:** Separation quality for features \(F^{(1)}_{\text{MFCC}}\) for different parameters of \(a_\lambda\) and \(f_\lambda\).
Figure 6.4: Separation quality for different number of filters $K_{mel}$ for the mel filter bank and different dimension reduction strategies.
Clustering with Temporal SFM

Because of the analogy between the temporal- and the spectral SFM, the same experiments as for the spectral SFM are now applied on the temporal SFM.

\[ y_i(n, c) \]

\[ G_0, \beta=0 \]
\[ G_1, \beta=0 \]
\[ G_1, \beta=1 \]
\[ G_1, \beta=2 \]

Figure 6.5: Direct comparison of time domain signal \( y_i(n, c) \) and the estimated envelopes for one single separated component. \( G_0 \) is the output of the NTF. \( G_1 \) are the outputs of an NTF applied on the separated time domain signal \( y_i(n, c) \) with different values of \( \beta \). It can be seen that due to the scale-invariance for \( \beta = 0 \), the envelopes are not estimated correctly, especially in the region of 2 to 3 seconds.
**Improving the Robustness of Temporal Features**  
First experiments regarding the temporal SFM show that these features are very sensitive to the parameter \( \beta \) used in the BSS framework for separation according to Section 4.3. This can be explained by the scale-invariance of the \( \beta \)-divergence for \( \beta = 0 \): In Figure 2.1, it can be seen that for larger amplitudes of the tensor, the cost function becomes smoother, and therefore insensitive to very crude estimations. This effect is visualized also in Figure 6.5. The more silent temporal segments (first to third second and fourth to fifth second) are only approximated by the NTF in an appropriate way in the case of \( \beta > 0 \).

To avoid this sensitivity and to factorize better envelopes, usually temporal smoothness is introduced as an additional constraint. See also [3] or [25] for further details on temporal smoothness in the context of the NTF. This thesis is only about the basic NTF without any additional constraints. Therefore, a pre-processing of the separated envelopes is proposed: Motivated by this sensitivity, an STFT followed by an NTF (with different values for \( \beta \)) is applied on the separated time domain components \( y_i(n, c) \). By this, the features based on the temporal SFM become independent from the parameter \( \beta \) chosen for separation according to our BSS framework. In [85], this post-processing is not necessary, because the note events are separated with \( \beta = 1 \) leading to smoother approximations of the temporal envelope. The spectrograms \( Y_i(k, t, c) \) for each separated sound event

<table>
<thead>
<tr>
<th>( \beta = 0 )</th>
<th>( \beta = 1 )</th>
<th>( \beta = 2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p = 1 )</td>
<td>5.80</td>
<td>7.13</td>
</tr>
<tr>
<td>( p = 2 )</td>
<td>5.46</td>
<td>7.03</td>
</tr>
</tbody>
</table>

Table 6.3: Separation quality for the temporal SFM for the NTF-parameters regarding the cost function \( (\beta) \) and the mixing model \( p \).

\( y_i(n, c) \) are evaluated with a window size of 4096 and a hop size of 2048 samples. These spectrograms are factorized by an NTF into one component \( (I = 1) \). All of these components are stored in the \( i \)-th columns of matrices \( A_1, B_1, \) and \( G_1 \). This post-processing NTF is applied with \( \beta = \{0, 1, 2\} \) on the magnitude spectrogram \( (p = 1) \) or the power spectrogram \( (p = 2) \). The features for the temporal SFM are evaluated according to Equations (5.13)–(5.15) and clustered by the FCM. The resulting separation quality is shown in Table 6.3. It can be seen that the features \( F_{\text{SFM,G}}^{(2)} \) are very sensitive to the NTF-parameter \( \beta \) used for factorization of the single note events, as can be seen in Figure 6.5, too. For acceptable separation quality, \( \beta \geq 1 \) is necessary. In the following we use \( \beta = 2 \) and \( p = 1 \) for the post-processing NTF, because these parameters lead to the best results.

**Frequency- and Amplitude-Scaling by the Box-Cox-Transform**  
In a second experiment, the influence of the Box-Cox transform on the temporal SFM is evaluated. The resulting separation quality is shown in Table 6.4. Because it is unknown, whether the

---

5Because of the simple solution in the case of \( I = 1 \), only a few number of iterations is sufficient for this NTF (roughly 5 iterations), see also Equations (2.36)-(2.38).

6Remember, for \( \beta = 2 \) and \( I = 1 \), the NTF is equivalent to the SVD followed by dropping all separated channels except the first one with largest variance.
6.2 Basic Clustering Algorithms

<table>
<thead>
<tr>
<th></th>
<th>$f_\lambda$</th>
<th>$a_\lambda$</th>
<th>$\pm 12$ dB</th>
<th>0 dB</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>standard</td>
<td>1.00</td>
<td>0.00</td>
<td>7.38</td>
<td>7.41</td>
<td>7.39</td>
</tr>
<tr>
<td>BCT</td>
<td>2.00</td>
<td>0.00</td>
<td>7.47</td>
<td>7.72</td>
<td>7.60</td>
</tr>
</tbody>
</table>

Table 6.4: Separation quality for features $F_{\text{SFM,G}}^{(2)}$. $f_\lambda = 1$ corresponds to dropping the frequency scaling, $a_\lambda = 0$ corresponds to the logarithmic scaling of amplitudes necessary for standard cepstral analysis.

source or the filter signal is better suited for clustering, the matrix $R_{\text{SFM,G}}$ only scales the frequency-axis by a Box-Cox transform. Contrary to the spectral SFM, the scaling matrix $R_{\text{SFM,G}}$ is a square matrix. Thus it introduces no dimension reduction. Additionally, the amplitude axis is scaled by a Box-Cox transform.

The search range for $f_\lambda$ and $a_\lambda$ is given by $0 \leq f_\lambda \leq 3$ and $-1 \leq a_\lambda \leq 1$. The standard parameters are the same as in [85]: No frequency-warping ($f_\lambda = 1$) and logarithmic amplitude scaling ($a_\lambda = 0$).

As for the spectral SFM, the Box-Cox transform increases the separation quality significantly. Contrary to the spectral SFM, the frequency scaling is done by $f_\lambda > 1$ to increase the separation quality. Such frequency scaling is shown exemplary in Figure 6.3(d). It can be seen that $f_\lambda > 1$ compresses low frequencies and spread high frequencies.

In a third experiment, the temporal SFM is combined with dimension reduction schemes. As shown for the spectral SFM in Figure 6.4, best separation quality is achieved by the DFT in combination with a low-pass filter. Unfortunately, for the temporal SFM no improvements are possible by applying a DFT, a low-pass, and an inverse DFT successively. Therefore, dimension reduction is not discussed any further.

Reducing the Computational Complexity of Envelope Estimation

One open problem is the computational complexity: For each $i$ an additional NTF is necessary. Therefore, a faster version of this algorithm is introduced. Firstly, the envelopes for each channel are estimated by

$$\hat{G}(t, i) = \sqrt{\frac{1}{C'} \sum_{c=1}^{C'} \sum_{n=w_s(t-1)+1}^{w_s t} y_i(n, c)^2}, \quad (6.31)$$

with $w_s$ set to number of samples corresponding to 100 milliseconds. The following evaluation of features $F_{\text{SFM,G}}^{(2)}$ is identical as in Equations (5.13)–(5.15), but based on the new estimated envelopes $\hat{G}$.

In Table 6.5, the evaluation of envelopes with the post-processing NTF and the fast algorithm proposed in Equation (6.31) are compared with each other. The rows corresponding to G are the same as shown in Table 6.4, $\hat{G}$ corresponds to the new envelope estimation algorithm. As mentioned in the beginning of Section 5.1.1, the columns of matrices $A$, $B$, and $G$ are scaled to equal energy for the purpose of clustering. This is advantageous for large dynamic differences, because the cluster centroids of the FCM algorithm are mainly influenced by feature vectors corresponding to higher energies. The new envelopes $\hat{G}$ have the same energies as the signals $y_i(n, c)$ due to the Parseval-theorem. As can be seen in Table 6.5, the fast evaluation of envelopes leads to much worse separation quality for large
Table 6.5: Separation quality for features $F_{SFM,G}^{(2)}$ for the fast evaluation of envelopes $\tilde{G}$ and different scaling parameters $f_\lambda$, and $a_\lambda$.

dynamic differences, and to much better separation quality for nearly equal loudness.

<table>
<thead>
<tr>
<th></th>
<th>$f_\lambda$</th>
<th>$a_\lambda$</th>
<th>$p$</th>
<th>$\pm12\text{ dB}$</th>
<th>$0\text{ dB}$</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>7.38</td>
<td>7.41</td>
<td>7.39</td>
</tr>
<tr>
<td></td>
<td>2.00</td>
<td>0.00</td>
<td>0.00</td>
<td>7.47</td>
<td>7.72</td>
<td>7.60</td>
</tr>
<tr>
<td>G</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>6.55</td>
<td>8.38</td>
<td>7.47</td>
</tr>
<tr>
<td></td>
<td>1.00</td>
<td>-1.00</td>
<td>0.00</td>
<td>6.73</td>
<td>8.31</td>
<td>7.52</td>
</tr>
</tbody>
</table>

Figure 6.6: Separation quality for the temporal SFM based on estimated envelopes $\tilde{G}$ for different normalization parameters $p$ as explained in Equation (6.32).

Normalization of Envelopes Motivated by the results in Table 6.5, the features $F_{SFM,G}^{(2)}$ are normalized by a parameter $p$:

$$F_{SFM,G}^{(2)}(k_{filter}, i) \leftarrow F_{SFM,G}^{(2)}(k_{filter}, i) \cdot \frac{1}{\left( \sum_k \left( F_{SFM,G}^{(2)}(k, i) \right)^2 \right)^p}.$$  (6.32)

By the parameter $p$, the influence on the centroid evaluation of the features corresponding to signal parts with higher energies can be directly controlled. $p = 0$ corresponds to unaltered features without normalization, $p = 0.5$ corresponds to spherical normalization.

The separation quality for $p \neq 0$ can be seen in the last two rows of Table 6.6. The best separation result is achieved with $f_\lambda = 1.0$, $a_\lambda = -0.5$, and $p = 0.3$. As we will see later, the parameters chosen in the last row performs much better for data beside test set $\mathcal{A}$. Motivated by the small deficit in separation quality for data $\mathcal{A}$, the parameters $a_\lambda = -1.0$, $f_\lambda = 1.5$, and $p = 0.3$ are used in the following.

To better visualize the influence of parameter $p$ on the separation quality, the separation quality for different dynamic differences is plotted in Figure 6.6. It can be seen that for large dynamic differences, the normalization to unit energy improves the separation.
Table 6.6: Separation quality for features $F^{(2)}_{SFM,G}$ for the fast evaluation of envelopes $\tilde{G}$ and different scaling parameters $f_\lambda$, $a_\lambda$ and normalization parameters $p$.

<table>
<thead>
<tr>
<th></th>
<th>$f_\lambda$</th>
<th>$a_\lambda$</th>
<th>$p$</th>
<th>$\pm 12 \text{ dB}$</th>
<th>$0 \text{ dB}$</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G$</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>7.38</td>
<td>7.41</td>
<td>7.39</td>
</tr>
<tr>
<td></td>
<td>2.00</td>
<td>0.00</td>
<td>0.00</td>
<td>7.47</td>
<td>7.72</td>
<td>7.60</td>
</tr>
<tr>
<td>$G$</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>6.55</td>
<td>8.38</td>
<td>7.47</td>
</tr>
<tr>
<td></td>
<td>1.00</td>
<td>$-1.00$</td>
<td>0.00</td>
<td>6.73</td>
<td>8.31</td>
<td>7.52</td>
</tr>
<tr>
<td>$G$, normalized features</td>
<td>1.00</td>
<td>$-0.50$</td>
<td>0.30</td>
<td>7.39</td>
<td>8.09</td>
<td>7.74</td>
</tr>
<tr>
<td></td>
<td>1.50</td>
<td>$-1.00$</td>
<td>0.30</td>
<td>7.43</td>
<td>7.98</td>
<td>7.70</td>
</tr>
</tbody>
</table>

quality. For low dynamic differences, the normalization by large values of $p$ reduces the separation quality, because cluster centroids are less influenced by the louder separated components. For unknown dynamic differences, best separation quality is achieved with $p \approx 0.3$.

It is interesting that this scaling by $p \approx 0.3$ corresponds to the scaling of the columns of matrices $A$, $B$, and $G$ to identical energy. By this, the whole signal’s energy is spread over three different vectors (the columns of all three matrices), thus approximating a normalization with $p = 0.3$. Therefore, the only difference between the energy-scaling used for spectral SFM and the temporal SFM is that for spectral SFM, the energy normalization is applied before the first feature evaluation is done, and for the temporal SFM after the last feature evaluation step.

In Figure 6.7, the final signal flow for evaluating the features of the temporal source-filter model is summarized.

6.2.2 Clustering all other Features

Now, spectral- and temporal-based features are concerned for clustering. For improving the readability of the tables in the following, separation qualities above 6.5 dB are highlighted gray.

Clustering with largest Feature Spaces

<table>
<thead>
<tr>
<th>$F^{(0)}<em>{s</em>{e,1 \set 1}}$ (Spectral features based on $f_0$)</th>
<th>$F^{(0)}<em>{s</em>{e,2 \set 2}}$ (Spectral features independent from $f_0$)</th>
<th>$F^{(0)}<em>{s</em>{e,3 \set 3}}$ (Temporal features)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F^{(0)}_{f_0}$</td>
<td>$F^{(1)}_{SC}$</td>
<td>$F^{(0)}_{zcr}$</td>
</tr>
<tr>
<td>$F^{(1)}_{HR}$</td>
<td>$F^{(0)}_{specSlope}$</td>
<td>$F^{(0)}_{FNA}$</td>
</tr>
<tr>
<td>$F^{(0)}_{ \text{partial}}$</td>
<td>$F^{(1)}_{ASF}$</td>
<td>$F^{(0)}_{crest}$</td>
</tr>
</tbody>
</table>

Table 6.7: Grouping of features according to their evaluation strategy.
Figure 6.7: Signal flow of the final evaluation of the features for the temporal SFM.

In a first experiment, all (non-cepstral) features are clustered into three groups of features as shown in Table 6.7: Spectral features based on $f_0$, spectral features independent from $f_0$, and temporal features. The features of these three types are concatenated to three feature matrices. These feature matrices are clustered by the FCM clustering algorithm. To show the influence of unimodal or spherical clustering, the features are clustered by Euclidean- and spherical distance with and without normalization. This results in four possible clustering algorithms shown in Table 6.8. It can be seen that normalization improves the separation quality in nearly all cases. Best results are gained with spectral features independent from $f_0$.

![Signal flow diagram](image-url)

Table 6.8: Separation quality for maximum size of feature matrices. $d_m = 1$ corresponds to spherical distance, $d_m = 2$ to Euclidean distance. Clustering is done by FCM.

<table>
<thead>
<tr>
<th></th>
<th>no normalization</th>
<th>with normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$d_m = 1$</td>
<td>$d_m = 2$</td>
</tr>
<tr>
<td>$F_{set1}^{(0)}$</td>
<td>5.33</td>
<td>5.12</td>
</tr>
<tr>
<td>$F_{set2}^{(0)}$</td>
<td>4.93</td>
<td>5.27</td>
</tr>
<tr>
<td>$F_{set3}^{(0)}$</td>
<td>5.76</td>
<td>5.05</td>
</tr>
</tbody>
</table>
### Clustering with Fine Granularity

<table>
<thead>
<tr>
<th>Feature</th>
<th>no normalization</th>
<th>with normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dm = 1</td>
<td>dm = 2</td>
</tr>
<tr>
<td>$F_{(0)}$</td>
<td>5.50</td>
<td>5.50</td>
</tr>
<tr>
<td>$F_{(1)}$</td>
<td>6.12 5.89</td>
<td>6.18 5.90</td>
</tr>
<tr>
<td>$F_{HR}$</td>
<td>5.10 4.98</td>
<td>5.04 5.13</td>
</tr>
<tr>
<td>$F_{SC}$</td>
<td>7.05 6.58</td>
<td>7.13 6.48</td>
</tr>
<tr>
<td>$F_{ssl}$</td>
<td>5.27</td>
<td>5.27</td>
</tr>
<tr>
<td>$F_{ASF}$</td>
<td>5.04</td>
<td>5.04</td>
</tr>
<tr>
<td>$F_{AM}$</td>
<td>4.77 4.77</td>
<td>4.77 4.77</td>
</tr>
<tr>
<td>$F_{NA}$</td>
<td>4.95 5.74</td>
<td>4.88 5.75</td>
</tr>
<tr>
<td>$F_{crest}$</td>
<td>4.84</td>
<td>4.84</td>
</tr>
<tr>
<td>$F_{zcr}$</td>
<td>5.93</td>
<td>5.93</td>
</tr>
</tbody>
</table>

**Table 6.9:** Separation quality for medium size of feature matrices. $dm = 1$ corresponds to spherical distance, $dm = 2$ to Euclidean distance.

In a second experiment, the features are concatenated to blocks corresponding to their introduction in Chapter 5. The separation quality is shown in Table 6.9. Best separation quality is achieved with the three features based on the spectral centroid according to Section 5.3.1: Standard deviation, mean value and maximum value over time. The good separation quality induced by this feature set can be explained by the similarity between this feature set and the spectral SFM, which currently leads to best separation quality. The only other feature-sets with noticeable separation quality is the zero crossing rate and the harmonic ratio. Here, especially the zero crossing rate is remarkable, because of the very low complexity necessary for evaluation.

### Clustering Single Features

In a third experiment, each single feature is transformed by the Box-Cox transform (BCT) and clustered with FCM. The spherical distance cannot be applied, because the feature spaces are only one-dimensional. For the same reason, normalization can be ignored, because it has no influence for one-dimensional features. The features are transformed by the BCT with $0 \leq \lambda \leq 2$. $\lambda = 1$ corresponds to the original feature space. Another degree of freedom is introduced by the **offset**. With this **offset** the BCT is described by

$$F_{xy}^{(\nu)}(n, i) = \text{BCT}_\lambda \left( F_{xy}^{(\nu-1)}(n, i) + \text{offset} \right) = \begin{cases} \log (F_{xy}^{(\nu-1)}(n, i) + \text{offset}), & \text{if } \lambda = 0 \\ \left( F_{xy}^{(\nu-1)}(n, i) + \text{offset} \right)^{\frac{1}{\lambda}}, & \text{otherwise.} \end{cases}$$

(6.33)

In Equation (5.1) the **offset** added to the feature before the transform is set to zero, which can result in numerical problems in the case of $\lambda \leq 0$. Motivated by this an **offset** of $+1$ is added before the BCT, as shown in Equation (5.2).
Table 6.10: Separation quality of single features (feature matrices with minimum size). $\lambda$ corresponds to the scaling factor of the used Box-Cox transform and offset is the corresponding data offset to avoid numerical problems, e.g. division by a small absolute value.

<table>
<thead>
<tr>
<th>Feature</th>
<th>$\text{SNR}_\text{seg}$</th>
<th>$\text{SNR}_\text{seg}$</th>
<th>$\lambda_{\text{opt}}$</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F^{(0)}_{f_0}$</td>
<td>5.50</td>
<td>6.93</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F^{(1)}_{HR10}$</td>
<td>5.86</td>
<td>6.06</td>
<td>0.50</td>
<td>0.01</td>
</tr>
<tr>
<td>$F^{(1)}_{HR11}$</td>
<td>5.60</td>
<td>6.02</td>
<td>0.00</td>
<td>0.10</td>
</tr>
<tr>
<td>$F^{(1)}_{HR12}$</td>
<td>5.69</td>
<td>6.11</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>$F^{(1)}_{HR13}$</td>
<td>5.67</td>
<td>6.07</td>
<td>0.00</td>
<td>0.10</td>
</tr>
<tr>
<td>$F^{(1)}_{HR14}$</td>
<td>5.73</td>
<td>6.14</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>$F^{(1)}_{HR15}$</td>
<td>5.62</td>
<td>6.12</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>$F^{(1)}_{HR16}$</td>
<td>5.63</td>
<td>6.15</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>$F^{(1)}_{HR17}$</td>
<td>5.64</td>
<td>6.10</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>$F^{(1)}_{HR18}$</td>
<td>5.64</td>
<td>6.11</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>$F^{(1)}_{HR19}$</td>
<td>5.57</td>
<td>6.08</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>$F^{(0)}_{SC1}$</td>
<td>7.26</td>
<td>7.89</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F^{(0)}_{NA1}$</td>
<td>5.65</td>
<td>6.16</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F^{(0)}_{NA2}$</td>
<td>5.67</td>
<td>6.17</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F^{(0)}_{NA3}$</td>
<td>5.70</td>
<td>6.27</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F^{(0)}_{zcr}$</td>
<td>5.93</td>
<td>7.12</td>
<td>0.00</td>
<td>0.01</td>
</tr>
</tbody>
</table>

On the other hand, an constant offset of +1 has major influence on the features if the amplitude range of the corresponding feature space is limited to small absolute values, e.g. the zero crossing rate, which is by definition limited to the interval $0 \leq F^{(0)}_{zcr} \leq 1$. Therefore, three values are tested as possible offset: $\{10^{-2}, 10^{-1}, 10^0\}$. With this search range for $\lambda$ and for the given values of the parameter offset, it can be shown, if the dynamic range of the features have to be compressed ($\lambda < 1$) or expanded ($\lambda > 1$) before clustering.

For features resulting in a separation quality higher than 6 dB, the results are shown in Table 6.10. For all other features the results are shown in Table C.1 for completeness. For each of the applied clustering algorithms two values for separation quality are given: standard $\text{SNR}_\text{seg}$ without BCT (or equivalent $\lambda = 1$ and offset = 1), and $\text{SNR}_\text{seg}$ for features transformed by the BCT with the given $\lambda_{\text{opt}}$ and offset.

It is surprising that for the harmonic features $F^{(1)}_{HR(1-19)}$, the partials corresponding to the higher frequency orders result in much better separation quality than the lower order partials. This can be explained by the exception handling for higher frequencies. If the frequency of the $n$-th partial exceed half of the sampling frequency, the corresponding amplitude of this partial is set to zero. Therefore, the amount of higher partials that are set to zero can be interpreted as a binary decision about the value of the fundamental frequency: The higher the fundamental frequency, the more higher order partials are set to zero. Therefore, the higher order partials benefit from the separation quality of the fundamental frequency.
6.2 Basic Clustering Algorithms

Considering this side-effect of the fundamental frequency, only three features result in a separation quality above 6 dB: \( F_{f_0}^{(0)} \), \( F_{SC1}^{(1)} \), and \( F_{zcr}^{(0)} \). The pitch has the disadvantage of high computational complexity. Additionally, it cannot be guaranteed that the pitch is detected error-free and improving pitch detection algorithms is beyond the scope of this thesis. The spectral centroid is nearly the same information as utilized for the spectral SFM. On the other hand, the zero crossing-rate is very fast to evaluate and can be assumed of being independent from the cepstral features used so far. All three features are used with an offset of \( 10^{-2} \) and with \( \lambda = 0 \).

In a last experiment, the three features \( F_{f_0}^{(0)} \), \( F_{SC1}^{(1)} \), and \( F_{zcr}^{(0)} \) are concatenated to form a single feature matrix. These feature matrices are called \( F_{set4}^{(0)} - F_{set7}^{(0)} \). The results and the combination of features corresponding to each feature set are shown in Table 6.11. It can be seen that all combinations using the feature spectral centroid do not exceed the separation quality of the spectral centroid as a stand-alone feature. The combination of pitch and zero crossing rate clustered with spherical distances improves the separation quality significantly compared to the pitch or the zero crossing rate used as a stand-alone feature (7.7 dB with spherical distance and normalization compared to 7.12 dB for zero crossing rate and 6.93 dB for pitch). Additionally, it can be observed that the spherical distance measure improves the separation quality compared to the Euclidean distance for all combinations, except for \( F_{set4}^{(0)} \). In the following, feature \( F_{set4}^{(0)} \) is used with Euclidean distance, and features \( F_{set5}^{(0)} - F_{set7}^{(0)} \) are used with spherical distance. All four combinations are normalized according to the chosen data model before clustering.

### Table 6.11: Clustering of combinations of the three best single features

<table>
<thead>
<tr>
<th></th>
<th>( F_{f_0}^{(0)} )</th>
<th>( F_{SC1}^{(1)} )</th>
<th>( F_{zcr}^{(0)} )</th>
<th>( \text{no normalization} )</th>
<th>( \text{with normalization} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F_{set4}^{(0)} )</td>
<td>0.00</td>
<td>1.00</td>
<td>1.00</td>
<td>7.31</td>
<td>7.47</td>
</tr>
<tr>
<td>( F_{set5}^{(0)} )</td>
<td>1.00</td>
<td>0.00</td>
<td>1.00</td>
<td>7.62</td>
<td>7.28</td>
</tr>
<tr>
<td>( F_{set6}^{(0)} )</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>7.82</td>
<td>7.19</td>
</tr>
<tr>
<td>( F_{set7}^{(0)} )</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>7.75</td>
<td>7.43</td>
</tr>
</tbody>
</table>

6.2.3 Clustering beside Audio Data \( \mathcal{A} \)

Now, the experience in clustering gained for data \( \mathcal{A} \) is applied on other data. For data \( \mathcal{B} \), it is shown in Figure 4.10 that the separation quality is not monotonically increasing for increasing degrees of freedom \( I \). Even with the non-blind reference clustering used in Chapter 4, the separation quality collapses for certain values of \( I \). This can be explained by the small amount of mixtures (only 5) for this data-set. Motivated by this, the parameter \( I \) will be considered in the first experiment: \( I \in \{15, 20, 25\} \). For a fair comparison with e.g. [27] and [48], the separation quality is measured additionally with the quality measures proposed in [41]: SDR, SIR, and SAR. Unfortunately,
for data C, the evaluation of these measures is too complex because of the long duration of these mixtures (mixture lengths of more than 50 seconds). Therefore, we evaluate for data C the SNR only, which we have observed to be equal to the SDR of the BSS quality evaluation toolbox. For a more detailed description of these measures, please take a look at Section 2.4.3 or [41].

Cepstral Clustering: Spectral Model

![Figure 6.8](image)

Figure 6.8: Mean separation quality for different sets of audio data. Clustering is done based on the spectral SFM evaluated with $a_\lambda = -0.5$. The separation quality is shown for different scalings of frequency-axis controlled by parameter $f_\lambda$.

In Section 6.2.1, it is shown that the parameter $f_\lambda$ has large impact on the separation quality. In a first experiment, the clustering based on the MFCC is applied with different values of $f_\lambda$ on the data B, C, and D. $a_\lambda$ is fixed to the value $-0.5$. The results for data A are based on experiments presented already before, and are shown here for better
comparison. The results are shown in Figure 6.8 for the quality measure SDR. Firstly, it can be observed that \( I = 25 \) seems to be a proper choice for separation of data \( B-C \). This indicates that the mixtures are either relatively complex as for data \( B \) or they are very long as the mixtures of data \( C \). For data \( D \) (very short mixtures of low complexity), \( I = 20 \) seems to be sufficient. The differences to \( I = 25 \) is relatively small. Therefore, in the following, \( I \) is set to \( I = 25 \) for data \( B-D \) for simplicity.

For data \( B \) and \( D \), it can be verified that the separation quality increases for values \( f_\lambda < 0 \). For data \( A \), \( f_\lambda \leq 0 \) leads to good separation quality. Data \( C \) seems to be robust against changing parameter \( f_\lambda \). Considering all four test sets, best results are achieved with \( f_\lambda = -1.5 \). The conclusion for this experiment is that the separation quality of the spectral SFM can be increased, by setting the parameter \( f_\lambda \) to values below zero. This result can be verified on different sets of audio data.

It should be noted that the curves for data \( B-D \) shown in Figure 6.8 are not as smooth, as for data \( A \) shown in Figure 6.8(a), because of the relatively small number of mixtures for these data sets.

Cepstral Clustering: Temporal Model

In a second experiment, the temporal SFM is used for clustering data \( B-D \). The fast evaluation of envelopes is used, as described in Equation (6.31).

In Table 6.6, separation quality is evaluated for two parameter settings for data \( A \). Regarding the SNR\(_{\text{seg}} \), \( f_\lambda = 1.0 \), and \( a_\lambda = -0.5 \) lead to best separation quality. The other parameter setting (\( f_\lambda = 1.5 \), and \( a_\lambda = -1.0 \)) leads to only a small decrease in separation quality.

In this section it will be shown that \( f_\lambda = 1.5 \), and \( a_\lambda = -1.0 \) lead to best separation quality regarding the SNR. As mentioned earlier, comparison is usually done by the SNR\(_{\text{seg}} \) for data \( A \). For data \( B-D \) the SNR is used for comparison. Additionally, the small decrease in separation quality regarding data \( A \) is compensated by the large gains in separation quality regarding data \( B \).

Therefore, the temporal source-filter model is used with \( f_\lambda = 1.5 \) and \( a_\lambda = -1.0 \) in the following. These parameter settings lead to best separation quality regarding all four used sets of audio data.

Additionally, in Figure 6.9 the influence of proper feature normalization by parameter \( p \) is shown. For all sets of audio mixtures, it can be observed that a normalization with \( p \approx 0.3 \) increases the separation quality compared to the both options no normalization (\( p = 0 \)), and spherical normalization (\( p = 0.5 \)).

Temporal-, Spectral- and Spatial Features

Motivated by the results shown in Section 6.2.2, we restrict the temporal- and spectral feature space to the combinations of the single features pitch, spectral centroid, and zero crossing rate. The definitions of the feature sets \( F^{(0)}_{\text{set4}} \) and \( F^{(0)}_{\text{set7}} \) are given in Table 6.11: These four feature sets consists of a concatenation of the three features pitch, spectral centroid, and zero crossing rate. As shown in Table 6.10, all of these three features have to be scaled by the Box-Cox transform logarithmically (\( \lambda = 0 \)) with an offset of \( 10^{-2} \).
Figure 6.9: Mean separation quality for different sets of audio data over the normalization parameter $p$ for two different frequency-($f_\lambda$) and amplitude-scalings ($a_\lambda$).

<table>
<thead>
<tr>
<th></th>
<th>$F^{(0)}_{\text{set}4}$</th>
<th>$F^{(0)}_{\text{set}5}$</th>
<th>$F^{(0)}_{\text{set}6}$</th>
<th>$F^{(0)}_{\text{set}7}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B$</td>
<td>0.98</td>
<td>0.94</td>
<td>0.81</td>
<td>0.95</td>
</tr>
<tr>
<td>$C$</td>
<td>4.73</td>
<td>4.65</td>
<td>4.96</td>
<td>4.61</td>
</tr>
<tr>
<td>$D$</td>
<td>8.65</td>
<td>9.07</td>
<td>9.25</td>
<td>9.02</td>
</tr>
</tbody>
</table>

Table 6.12: Separation quality for spectral- and temporal-features.

The separation quality for data $B$–$D$ is shown in Table 6.12. $F^{(0)}_{\text{set}4}$ is used with Euclidean distance, all other feature spaces with spherical distance. For data $B$, these features result in a very low separation quality. Analogously to the spectral SFM, for data $C$ and $D$ the used features perform very well.

In Table 6.13, the separation quality for the instantaneous features $F^{(1)}_{\text{inst}}$, introduced in Section 5.5, is shown. Only data $B$ is used for evaluation, because the other data set are monaural. The low separation quality verifies the disadvantages of spatial features men-
6.2 Basic Clustering Algorithms

<table>
<thead>
<tr>
<th>no normalization</th>
<th>with normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>$dm = 1$</td>
<td>$dm = 2$</td>
</tr>
<tr>
<td>$B$</td>
<td>0.47</td>
</tr>
</tbody>
</table>

Table 6.13: Separation quality of the instantaneous features $F_{\text{inst}}^{(1)}$.

mentioned in the introduction of Section 5.5: The spatial features based on the instantaneous mixing model of the NTF are not well suited for clustering.

In the following, the convolutive features based on the estimated filters $\tilde{A}$ are used for clustering, see also Equation (B.5). The FCM algorithm is used for clustering and both data models (spherical and unimodal) are tested with and without normalization. Experiments have shown that a logarithmic frequency-scaling leads to worse separation quality compared to $R_{\text{conv}}$ being the identity matrix.

For $R_{\text{conv}}$ equal to the identity matrix, a frequency-scaling $f_\lambda$ is not applicable. Therefore, the separation quality for the four used clustering algorithms is shown only depending on amplitude scaling parameter $a_\lambda$, in Figure 6.10. Several things can be observed: Firstly, the separation quality of the spatial features is much lower than the features based on the temporal SFM. Secondly, these features shows no clear rank order regarding the separation quality. One reason for this is the evaluation of the convolutive filters $\tilde{A}$ based on the separation by the NTF. The NTF assumes an instantaneous mixing model. It seems that by this limitation of $\tilde{A}$ by the underlying NTF, no additional information can be extracted.

As a conclusion it can be stated that for the used algorithms and audio data, the spatial features are not useful for clustering and are not discussed any further in the following.
Table 6.14: Separation quality for data $B$ and the temporal SFM. Best features/algorithms are highlighted gray. The temporal SFM is evaluated for three different scaling/normalization parameters $a_\lambda$, $f_\lambda$, and $p$. The parameter setting in the leftmost column is optimal for data $A$, the rightmost one is optimal for data $B$. The parameter setting shown in the middle column is a compromise leading to good separation quality for both sets of audio data.

### Comparison with other BSS Frameworks

To finalize this chapter, the algorithms proposed in [27], [48], and [58] are directly compared with the proposed BSS-framework.

For these experiments, the temporal-, spectral-, and spatial-features are dropped. Only cepstral features are considered, because they lead to the best results for all basic clustering algorithms.

In Table 6.14, the algorithm proposed in [27] is compared with our BSS-framework for different features and clustering algorithms. The SDR values for [27] are the official separation results of the SiSEC 2010 discussed in [46]. As mentioned above, only the temporal SFM leads to good separation quality for this set of mixtures. The best parameters for clustering are different compared to data $A$. Therefore, the clustering is shown not only for the parameters optimal for data $A$ ($f_\lambda = 1.0$, $a_\lambda = -0.5$, and $p = 0.3$) and for the parameters optimal for data $B$ ($f_\lambda = 1.5$, $a_\lambda = -1.0$, and $p = 0.1$), but also for a compromise ($f_\lambda = 1.5$, $a_\lambda = -1.0$, and $p = 0.3$) leading to good separation quality for both sets of mixtures. Again, best separation quality for each mixture/input signal is highlighted gray for better visualization.

For the temporal SFM, the separation quality is in the same region as for [27]. Unfortunately, a realistic comparison is hard to obtain, because of the small number of mixtures used for comparison. Beside separation quality, one major advantage of our algorithm is
the relative low computational complexity. In [27], a computational complexity of more than one hour is reported for each mixture, where our algorithm evaluates the separated signals for each mixture in less than five minutes on a comparable PC. Although both algorithms are implemented in Matlab and are far away from being optimized regarding the computational complexity, such a dramatic difference between both evaluation times shows clearly the much smaller computational complexity of our algorithm.

For a second comparison, our algorithm is compared with the results shown in [48]

<table>
<thead>
<tr>
<th></th>
<th>SDR</th>
<th>SIR</th>
<th>SAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>[48]</td>
<td>9.01</td>
<td>24.91</td>
<td>9.52</td>
</tr>
<tr>
<td>[58]</td>
<td>8.94</td>
<td>23.69</td>
<td>9.72</td>
</tr>
<tr>
<td>$f_\lambda = -2.0$</td>
<td>8.84</td>
<td>15.31</td>
<td>14.87</td>
</tr>
<tr>
<td>$f_\lambda = -1.5$</td>
<td>10.18</td>
<td>16.88</td>
<td>15.28</td>
</tr>
<tr>
<td>$f_\lambda = -1.0$</td>
<td>9.93</td>
<td>16.17</td>
<td>14.79</td>
</tr>
</tbody>
</table>

Table 6.15: Separation quality for the spectral SFM evaluated for data $D$. As can be seen in Figure 6.8, for data $A$, $f_\lambda$ has to be set to $f_\lambda = -2.0$. For data $D$ $f_\lambda = -2.0$ leads to very low separation quality. $-1.5 \leq f_\lambda \leq -0.5$ leads to much better separation quality. Therefore, three different settings for $f_\lambda$ are shown here.

[58]. Best separation quality is achieved by the spectral SFM. Motivated by the results for data $A$, parameter $a_\lambda$ is fixed to $-0.5$. Separation quality is given for different values of $f_\lambda$, to show the influence of the frequency-scaling. $f_\lambda = -1.5$ leads to the best separation quality regarding the given search range. This is a slightly varying parameter setting compared to data $A$, as shown in Table 6.2. The parameter $f_\lambda$ is set to $f_\lambda = -1.5$ in the following: This results in a small decline in separation quality regarding data $A$ and much better separation quality regarding data $D$.

It can be observed that our algorithm introduces much less artifacts, and by this results in a better SDR and SAR. On the other hand, our algorithm results in more interferences. The high difference between our algorithm and [58] regarding the SIR can result in misleading interpretations that the algorithm introduced in [58] is superior to our algorithm: Due to the good suppression of interferences of both algorithms (SIR $> 16$ dB), the differences between both algorithms will most probably be not hearable, because the much louder artifacts (SAR $< 16$ dB) will most probably mask the interferences.

### 6.3 Summary

As a conclusion so far, it can be summarized that regarding the proposed BSS-framework

- only a small subset of features and clustering algorithms leads to good separation quality.
- for all audio data cepstral features lead to better separation quality than temporal-, frequency- and spatial-features.
- only three features of the temporal- and spectral-features lead to noticeable separation quality. Combining at least two of them to a concatenated feature space
allows the usage of the spherical distance which results in two advantages: Firstly, the spherical distance generally results in better separation quality for these features. Secondly, for the cepstral features and the temporal- and spectral-features the distance functions become numerical comparable, because the spherical distance is limited to the range of $0 \leq d \leq 2$. This comparability is advantageous if the different clustering algorithms have to be combined, as will be proposed in Chapter 7.

• for spectral SFM, the correlation between frequency-axis scaling parameter $f_\lambda$ and separation quality can be reproduced on other audio data. For temporal SFM, the correlation between normalization parameter $p$ and separation quality can be verified for other audio data.

• the optimal parameter settings for blind clustering algorithms differ for each set of audio mixtures. On the other hand, it is possible to find an appealing compromise leading to good (if not optimal) separation quality for all sets of audio data used in this thesis. The possibility of generalizing the parameter settings, is a necessary condition for motivating the usage of blind clustering algorithms in the context of BSS.

• spatial features perform much worse regarding the separation quality for our framework.

• for different audio data, different features should be preferred: Data $\mathcal{B}$ are better separated with clustering based on temporal SFM, data $\mathcal{C}$ and $\mathcal{D}$ are better separated by features based on spectral SFM.

Especially the last point motivates the following chapter: It is reasonable that the separation quality can only be increased significantly by effectively combining different clustering strategies.

Finally, it is shown that the proposed BSS-framework leads to a separation quality comparable to other state-of-the-art algorithms, e.g. [27] or [58].
Chapter 7

Combination of Clustering Strategies

It can be observed that each clustering algorithm results in good separation quality for a subset of the whole set of audio mixtures. Therefore, it is an obvious idea to combine different clustering strategies. In the following, several algorithms are proposed for combining different clustering strategies.

Firstly, only data $\mathcal{A}$ is considered, because it is the set of audio data with largest number of mixtures. In Section 7.1, the maximum possible gain regarding the separation quality by combining two or three clustering strategies is shown. In Section 7.2, a simple combination strategy is introduced only based on the estimated separated sources $\hat{s}_m(n, c)$. In Section 7.3, several features are combined to a unified feature space before clustering is applied to achieve a combination in feature space. In Section 7.4, the clustering decision of different clustering algorithms is analyzed by so-called multi-expert-systems (MES) in order to improve the overall separation quality. Finally, in Section 7.5, the different combination strategies are applied on data $\mathcal{B}$–$\mathcal{D}$ to verify the experimental results of the preceding sections.

7.1 Pre-Selection of Feature Spaces

<table>
<thead>
<tr>
<th>Feature</th>
<th>SNR</th>
<th>SNR$_{seg}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>reference</td>
<td>11.75</td>
<td>11.90</td>
</tr>
<tr>
<td>random</td>
<td>0.85</td>
<td>4.29</td>
</tr>
<tr>
<td>$F^{(1)}_{\text{MFCC}}$</td>
<td>5.89</td>
<td>8.50</td>
</tr>
<tr>
<td>$F^{(2)}_{\text{SFM,G}}$</td>
<td>5.84</td>
<td>7.70</td>
</tr>
<tr>
<td>$F^{(0)}_{\text{set1}}$</td>
<td>5.64</td>
<td>7.86</td>
</tr>
<tr>
<td>$F^{(0)}_{\text{set5}}$</td>
<td>5.46</td>
<td>7.70</td>
</tr>
<tr>
<td>$F^{(0)}_{\text{set6}}$</td>
<td>5.51</td>
<td>7.84</td>
</tr>
<tr>
<td>$F^{(0)}_{\text{set7}}$</td>
<td>5.58</td>
<td>7.82</td>
</tr>
</tbody>
</table>

Table 7.1: Separation quality of all used clustering strategies. Reference and random clustering are given for showing a lower-limit and upper-limit of reachable clustering quality.

First of all, the separation quality for both used quality measures, SNR and SNR$_{seg}$, are shown for data $\mathcal{A}$ in Table 7.1. Additionally, the upper-limit (reference-clustering as used
in Chapter 4) and the lower-limit (random-clustering) are shown as a starting point of the following experiments.

To get first insights in the possibilities of combining different clustering strategies, the separation qualities of all six blind clustering strategies are combined in a non-blind way: For each mixture at each dynamic difference, the clustering algorithm with maximum separation quality is chosen. The maximum operates on the separation quality and is therefore non-blind. The mean values over the whole test-set are shown in Table 7.2 for the combination of two clustering algorithms, and in Table 7.3 for the combination of three clustering algorithms.

Firstly, it can be observed that the combination of the spectral and the temporal SFM leads to the best results regarding the combination of two clustering algorithms. Therefore, in the following we concentrate on the combination of both cepstral feature spaces in the case of combining two clustering algorithms. Secondly, for the combination of three clustering algorithms, it seems to be regardless, of which third clustering algorithm is combined with the two source-filter models (SFM). There is only one reason against feature space $F_{\text{set1}}$: It has to be clustered by Euclidean distance to improve the separation quality, see also Table 6.11. This can result in numerical problems for combination of several feature spaces, which can be assumed to be simpler in the case of identical distance measures for all used feature spaces. In order to reduce the number of possible combinations, in the following only the combination of $F_{\text{MFCC}}^{(1)}$, $F_{\text{SFM,G}}^{(2)}$, and $F_{\text{set7}}^{(0)}$ is considered in the case of three feature spaces used for combination.

Table 7.2: All possible combinations of two clustering strategies and the resulting separation quality in the case of optimal combination (non-blind maximum operation).
A possible combination strategy for combining different clustering algorithms in the signal space is introduced. This strategy is explained first, because it is the simplest by means of computational complexity and implementation time and effort. This strategy is based on an idea proposed in [96]. It is assumed that we have a set of \( N \) different BSS algorithms, with \( 1 \leq \nu \leq N \). Throughout this thesis, the mixing process is defined as in Equation (2.80). This mixing process combined with each of the \( N \) BSS algorithms can be modeled as a simple addition of an error signal \( e_{m,\nu} \), as shown in Figure 7.1. With \( \sum_{\nu=1}^{N} \alpha_\nu = 1 \), the different outputs of the BSS algorithms can be added up by a simple linear superposition:

\[
\tilde{s}_m(n, c) = \sum_{\nu=1}^{N} \alpha_\nu s_{m,\nu}(n, c), \tag{7.1}
\]

\[
= s_m(n, c) + \sum_{\nu=1}^{N} \alpha_\nu e_{m,\nu}(n, c). \tag{7.2}
\]
Combination of Clustering Strategies

\[ x(n,c) = \sum_m s_m(n,c) \]

BSS_\nu

\[ \hat{s}_{m,\nu}(n,c) \]

\[ s_m(n,c) \]

\[ e_{m,\nu}(n,c) \]

\[ \hat{s}_{m,\nu}(n,c) \]

**Figure 7.1:** Modeling of BSS algorithms by a simple addition of error signals \( e_{m,\nu} \).

With this, the squared error, introduced by this linear superposition, can be defined by

\[
\sum_{n,c} (s_m(n,c) - \hat{s}_m(n,c))^2 = \sum_{n,c} \left( \sum_{\nu=1}^{N} \alpha_\nu e_{m,\nu}(n,c) \right)^2 .
\] (7.3)

For uncorrelated sources, Equation (7.3) becomes

\[
\sum_{n,c} (s_m(n,c) - \hat{s}_m(n,c))^2 = \sum_{\nu=1}^{N} \alpha_\nu^2 \sum_{n,c} e_{m,\nu}(n,c). \] (7.4)

In order to minimize the squared error, the derivation of Equation (7.4) regarding the weighting factors \( \alpha_\nu \) is set to zero. This results in the following set of \( N-1 \) linear equations, which can be solved by matrix inversion:

\[
\sum_{n,c} e_{m,\nu}^2(n,c) = \alpha_\nu \sum_{n,c} e_{m,\nu}(n,c) + \sum_{\nu=1}^{N-1} \alpha_\nu \sum_{n,c} e_{m,\nu}(n,c). \] (7.5)

Therefore, if no other combination strategy is at hand, it is possible to combine different BSS algorithms simply by a linear superposition of their outputs.

For reducing the squared error of the linear superposition, we only need the expectation value of the error introduced by the BSS algorithms. This value can be derived by the SNR corresponding to the BSS algorithm: The mean SNR for a given set of \( L \) mixtures with \( M \) sources can be evaluated by

\[
\text{SNR}_{\text{audio data},\nu} = \frac{1}{L} \sum_{l=1}^{L} \frac{1}{M} \sum_{m=1}^{M} \frac{\sum_{n,c} s_m^2(n,c)}{\sum_{n,c} e_{m,\nu}^2(n,c)}. \] (7.6)

This mean SNR can be measured for each separation algorithm \( \nu \) for a given set of mixtures. Under the assumption of equal or nearly equal loudness, the values \( \sum_{n,c} s_m^2(n,c) \) can be assumed constant, e.g. fixed to the value 1. With this, the values \( \sum_{n,c} e_{m,\nu}^2(n,c) \) necessary for adjusting the weights \( \alpha_\nu \) can be derived out of the \( \text{SNR}_{\text{audio data},\nu} \). In Table
7.4, it will be shown, that even with a brute-force search over a given grid of possible weights $\alpha_\nu$, the separation quality $\text{SNR}_{\text{seg}}$ cannot be increased. Therefore, the smarter\(^1\) approach for adjusting the weights as shown above is not shown in the following.

This linear superposition increases the computational load of the whole system by $N$-times, if the BSS framework is used as a black box. For a combination of different clustering steps, the time-frequency transform and the note separation by NTF has to be done only once. And if not otherwise mentioned, the clustering algorithms can be done in a fraction of the time, necessary for the NTF. Therefore, it can be concluded that the increment in computational complexity can be ignored.

In Table 7.4, the separation quality is shown for linear superposition in time domain. Several things can be observed. Firstly, the alignment problem seems to be critical, because the separation quality with (perfect) non-blind alignment is roughly 0.2–0.3 dB better compared to the blind alignment. Alignment is done in the same way as for quality evaluation, as explained in the introduction of Chapter 4. The only difference between blind and non-blind alignment is the selection of the source-signals used as reference for alignment: Blind alignment uses the signals estimated by the spectral SFM based clustering. Non-blind alignment uses the correct input signals $s_m(n,c)$ for (correct) alignment. This sensitivity to the alignment problem is one disadvantage of the proposed combination in signal space. Secondly, the separation quality can only be increased regarding the SNR. The $\text{SNR}_{\text{seg}}$ cannot be increased by linear-superposition of the separated sources. This can be explained by the global weighting $\alpha_\nu$ for the different sources: The linear superposition of two separated signals increases the separation quality for temporal segments, for which both input signals have very low separation quality, as can be seen in Figure 7.2(b). If one, or both of the input signals have a good separation quality, the linear superposition leads to a separation quality somewhere between both separation qualities, as can be seen in Figure 7.2(c). Therefore, it is not possible to increase the mean $\text{SNR}_{\text{seg}}$ over a large range of mixtures by the linear superposition approach.

\(^1\)In this context, smarter refers to non-brute-force.
The separation quality for the SNR is different. This can be explained by the evaluation of the SNR shown in Equation (2.78). The denominator is primarily influenced by the temporal segments with large errors (e.g. the first segments in Figure 7.2). If the separation quality increases for these segments, the denominator of Equation (2.78) decreases a lot, and by this the overall SNR increases. Because of this different behaviour for both quality measures the weightings in Table 7.4 are chosen in order to maximize the SNR instead of maximizing the SNR\textsubscript{seg}. Therefore, the SNR\textsubscript{seg} is actually below the SNR\textsubscript{seg} of both input signals.

To increase the SNR\textsubscript{seg} it is necessary, to adapt the mixing levels $\alpha_\nu$ locally. A possible strategy for local adaptation is to choose the output signal of the best clustering algorithm as default, i.e. set the corresponding $\alpha_\nu$ to 1. For temporal segments with low separation
quality for all output signals, the different output signals are weighted equally. One open problem for this strategy is, how to detect the temporal segments with low separation quality.

It is reasonable to think of the NTF as a separation algorithm that separates the mixed signals locally in a perfect way, i.e. the different acoustical events are separated each into one single component of the NTF. Therefore, clustering these components can be interpreted as locally weighting of all I separated components. Following these interpretation of the NTF followed by a clustering step, local adaptation is the same as improving the clustering algorithm.

Because of this and the lack of a detection algorithm of segments with low separation quality, local adaptation of the combination in signal space is not considered in this thesis any further.

### 7.3 Combination in Feature Space

In [6], an extension to the k-means algorithm is introduced to combine feature spaces for clustering: The convex k-means algorithm. Assuming a set of L feature spaces $F_l$ of arbitrary dimensions. Further assuming a given clustering $a(i)$, the cluster centroids are evaluated for each of the L feature spaces:

$$ W_l(n,m) = \frac{1}{\sum_{i=1}^{L} \delta_{m,a(i)}} \sum_{i=1}^{L} F_l(n,i) \delta_{m,a(i)}. \quad (7.7) $$

The distance between each cluster centroid and each feature vector for each feature space is defined e.g. by

$$ d_l(m,i) = \sum_n (W_l(n,m) - F_l(n,i))^2 \quad \text{(Euclidean distance), or} \quad (7.8) $$

$$ d_l(m,i) = 1 - \frac{\sum_n W_l(n,m) \cdot F_l(n,i)}{\sqrt{\sum_n W_l^2(n,m)} \cdot \sqrt{\sum_n F_l^2(n,i)}} \quad \text{(spherical distance).} \quad (7.9) $$

Other distance functions are possible, and for each feature space $l$ the choice of the distance function may be different. The final distance between feature $i$ and centroid $m$ is the convex combination of these distances:

$$ d(m,i) = \sum_{l=1}^{L} \alpha_l d_l(m,i), \quad \text{with} \quad \sum_{l=1}^{L} \alpha_l = 1. \quad (7.10) $$

From this, the new clustering is simply defined by

$$ a(i) = \arg \min_m d(m,i), \quad (7.11) $$

and the iteration starts again with evaluating new cluster centroids until the clustering vector does not change by one iteration.

The fuzzy c-means (FCM) algorithm can be simply extended to the convex fuzzy c-means
(CFCM) by the following changes: Hard clustering decision by vector \( \mathbf{a} \) is exchanged by a partition matrix \( \mathbf{H} \) from which final clustering decision can be derived by Equation (6.8). The cluster centroids for each of the \( L \) feature spaces are evaluated by

\[
\mathbf{W}_l(n, m) = \frac{1}{\sum_{i=1}^{L} \mathbf{H}^l(i, m)} \sum_{i=1}^{L} \mathbf{H}^l(i, m) \mathbf{F}_l(n, i),
\]

with \( f \) being the constant fuzzifier set to 2 throughout this thesis. With this centroids, the distances between samples \( i \) and centroids \( m \) are evaluated by

\[
d(m, i) = \left( \sum_{l=1}^{L} \alpha_l d_l(m, i) \right)^{-2/(f-1)}
\]

The partition matrix is updated by

\[
\mathbf{H}(i, m) = \frac{d(m, i)}{\sum_{m=1}^{M} d(m, i)}.
\]

If the partition matrix converges, the algorithm stops, otherwise it starts again by evaluating the new centroids as explained above. Convergence for partition matrix is defined throughout this thesis, if the maximum change in partition matrix \( \mathbf{H} \) is smaller than \( 10^{-6} \).

Now, the CFCM clustering algorithm is used to combine the three feature spaces \( \mathbf{F}_{(1)}^{MFCC} \), \( \mathbf{F}_{(2)}^{SFM, G} \), and \( \mathbf{F}_{(0)}^{set7} \). The optimal weightings \( \alpha_l \) are evaluated by exhaustive search with each weighting \( \alpha_l \) having 11 possible values: \( \alpha_l \in \{0.0, 0.1, 0.2, \ldots, 1.0\} \). The results are shown in Table 7.5. It can be seen that the separation quality increases by the convex combination of two feature spaces (\( N = 2 \)). The combination of three feature spaces cannot increase the SNR\(_{seg} \) any further. Only the SNR is increased by an additional amount smaller than 0.1 dB which can be considered to be irrelevant.

### Table 7.5: Separation quality is shown for CFCM with different number of combined feature spaces \( N \).

<table>
<thead>
<tr>
<th>( N )</th>
<th>( \mathbf{F}_{(1)}^{MFCC} )</th>
<th>( \mathbf{F}_{(2)}^{SFM, G} )</th>
<th>( \mathbf{F}_{(0)}^{set7} )</th>
<th>SNR</th>
<th>SNR(_{seg} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>5.89</td>
<td>8.50</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>5.84</td>
<td>7.70</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>5.58</td>
<td>7.82</td>
</tr>
<tr>
<td>2</td>
<td>0.00</td>
<td>0.20</td>
<td>0.80</td>
<td>6.69</td>
<td>8.42</td>
</tr>
<tr>
<td></td>
<td>0.90</td>
<td>0.00</td>
<td>0.10</td>
<td>5.90</td>
<td>8.50</td>
</tr>
<tr>
<td></td>
<td>0.40</td>
<td>0.60</td>
<td>0.00</td>
<td>6.99</td>
<td>8.99</td>
</tr>
<tr>
<td>3</td>
<td>0.30</td>
<td>0.60</td>
<td>0.10</td>
<td>7.07</td>
<td>8.99</td>
</tr>
</tbody>
</table>

### 7.4 Combination in Cluster Space

A combination in cluster space interprets the vectors \( \mathbf{a}_v(i) \) of the different clustering algorithms which is usually referred as a multi-expert system (MES).
7.4 Combination in Cluster Space

An MES interprets each clustering algorithm as an expert. The MES combines the different clustering vectors by a decision logic to a single output clustering vector for improving the separation quality. The signal flow of a general MES is shown in Figure 7.3. To explain the basic concept of MES, two simple examples are given in the following.

7.4.1 Adaptation to Dynamic Differences

In [73], an MES is introduced based on the observation that NTF-clustering can be optimized to different dynamic differences between the active instruments by simply adjusting the parameter $\beta$ of the $\beta$-divergence used for clustering. Large values of $\beta$ ($\beta \approx 2$) are better suited for nearly equal loudness of the active instruments. Smaller values of $\beta$ ($\beta \approx 1$) generally lead to better separation quality in the case of large dynamic differences. The best value for $\beta$ is selected based on a classifier trained with the AdaBoost algorithm explained in [7]. It is shown that only a small set of features is sufficient for selecting between two different values of $\beta$: $\beta \in \{1, 2\}$, namely statistical independence between the output signals, dynamic differences between the output signals, and temporal evolution of the dynamic differences of the output signals. For more details, take a look at [73].

This MES is not discussed any further in this thesis because of the following reasons:

- FCM is used for clustering in our framework, but it is not possible to adapt this clustering routine the same way as the NTF-clustering: No parameter of the FCM clustering reveals the same sensitivity to the dynamic differences of the input signals, as the parameter $\beta$ used for the NTF-clustering.
- The separation quality of the FCM algorithm is better than the quality of the NTF-clustering regardless the given dynamic differences, as shown in Table 6.2.
- A separate training stage necessary for the AdaBoost is beyond the scope of this thesis: Here, only algorithms without a self-optimizing learning algorithm like AdaBoost are discussed\(^2\).

\(^2\)Of course, each experiment aiming at finding appropriate parameter-settings is equivalent to optimizing
• The choice of the dynamic differences used for testing (in this thesis: ±12 dB and 0 dB, in [73]: ±18 dB, ±12 dB,...,0 dB) is somehow arbitrary. This choice shall reflect real recording situations, but reliable statistics about commonly used dynamic differences in real recording situations are not at hand. Unfortunately, this choice of considered dynamic differences has major influence on the choice of necessary adaptation algorithms. Here, only the two scenarios ±12 dB and 0 dB are considered. For these scenarios, the FCM algorithm without any adaptation to the dynamic differences is sufficient, as can be seen in Table 6.2.

7.4.2 Adaptation to Tempo

Three scenarios for the usage of the temporal SFM are shown in Figure 5.12:

• Instruments with identical tempo but different instrument specific envelopes are clustered by analyzing the filter signal shown in Figure 5.11.
• Instruments with different tempo are clustered by analyzing the source signal shown in Figure 5.11.
• Mixtures with time-varying pitches.

In [83], it is shown that clustering based on the envelopes leads to better separation quality compared to the clustering by the spectral SFM in the case of high tempo of at least one of the active instruments. Motivated by this, an MES is introduced in [85] that decides for each mixture, if the temporal or the spectral SFM is used for clustering. The decision is based on a simple combining rule, which evaluates a single scalar value representing the tempo of the mixture. If the tempo exceeds a given threshold $\vartheta$, the temporal SFM is used for clustering, otherwise, the spectral SFM is used for clustering.

The tempo is measured for each separated note component $i$ by the feature $F^{(0)}_{\text{NA}}(3,i)$, defined in Equation (5.43). As shown in Figure 6.5, the columns of matrix $G$ result in misleading temporal envelopes. Therefore, the tempo is evaluated for both, the matrix $G$ for the first MES, and the matrix $\tilde{G}$ for the second MES:

$$\text{state}(t,i) = \varepsilon \left( \tilde{G}(t,i) - \frac{1}{T} \sum_l \tilde{G}(l,i) \right),$$

$$\tilde{F}^{(0)}_{\text{NA}}(i) = \frac{F_s}{h_s \cdot T} \sum_l \delta_{1,\text{state}(t,i)} \cdot \delta_{0,\text{state}(t-1,i)}.$$ (7.15) (7.16)

The separation quality for both proposed MES is shown in Figure 7.4. For more insights in the behaviour of the proposed algorithm for tempo adaptation, the separation quality for both MES is shown for a large range of threshold $\vartheta$. Again, it can be observed that the envelopes $G$ are better suited for analyzing the underlying instrument specific tempo and characteristic envelopes. The behaviour of both MES is similar: For small values of $\vartheta$ only clustering based on the temporal SFM is chosen. It can be seen that the estimation of the tempo based on $G$ generally leads to larger values regarding the tempo. This can be explained by Figure 6.5: Envelopes, estimated with $\beta = 0$ are not very smooth and have

the set of parameters to the given audio data. Therefore, the parameter-settings are always verified on audio data not used for optimization.
7.4 Combination in Cluster Space

Figure 7.4: MES with adaptation to the tempo of the mixture for different thresholds \( \vartheta \). MES 1 evaluates the tempo based on matrix \( G \), MES 2 uses matrix \( \tilde{G} \).

Many outliers (or peaks). Each of these outliers is interpreted as another note instance. This tendency towards higher tempo is the reason for the maximum of the solid curve being shifted towards higher values of \( \vartheta \). Even for the maximum threshold used in this experiment (\( \vartheta = 3 \)), the solid curve does not converge to the separation quality of the spectral SFM as given e.g. in Table 7.5. This indicates that measuring the tempo based on the matrix \( G \) (MES 1) leads to an unrealistic number of notes per second for a certain amount of mixtures. In Figure 7.5 the mean separation quality for a subset of sources of data \( \mathcal{A} \) is plotted. Firstly, it can be seen that clustering with the temporal SFM is better in the case of human speech, as mentioned above. Especially for steady-state signals like the bass, the spectral SFM performs much better. Secondly, the MES 2 with \( \vartheta = 1.0 \) is a good compromise between both cepstral clustering strategies based on \( F_{MFCC} \) and \( F_{SFM,G} \).
It can be seen in Figure 7.4 that \( \vartheta = 1.0 \) is not the optimal choice for data \( \mathcal{A} \). This sub-optimal choice will be motivated in Section 7.5 by a compromise between different sets of audio data.

### 7.4.3 Interpretation of Cluster Reliability

Another possibility to define an MES is to define a cluster-reliability and combine these cluster reliabilities by a decision logic, as proposed e.g. in [97].

A simple reliability measure for the FCM algorithm is the maximum of each column of the partition matrix:

\[
\psi_\nu(i) = \max_m H_\nu(m, i) - \frac{1}{M}.
\]  

(7.17)

The offset \( \frac{1}{M} \) is subtracted to set the minimum of the reliability to zero. Additionally, each classifier (expert) can be weighted by a constant reliability measure \( \beta_\nu \). This is a useful option, if the different experts have major differences regarding their mean separation quality. Dropping this classifier weighting is equivalent to setting \( \beta_\nu = \frac{1}{N} \). The final clustering decision is called weighted majority voting (WMV) in [97] and can be defined by

\[
a(i) = \arg \max_m \sum_{\nu=1}^{N} \beta_\nu \cdot \psi_\nu(i) \cdot \delta_{m, a_\nu(i)}.
\]  

(7.18)

In simple words: The WMV sums up all reliabilities of all classifiers that have decided for a given cluster \( m \). After \( M \) summations, the cluster with highest reliability wins.

As a pre-processing step, it is necessary to align the clustering decisions before combining the cluster decisions \( a_\nu(i) \). This is done by evaluating the estimated output signals for each classifier \( \nu \). The classifier \( \nu_{\text{align}} \) with highest reliability is chosen as reference for alignment:

\[
\nu_{\text{align}} = \arg \max_\nu \sum_{i=1}^{I} \psi_\nu(i).
\]  

(7.19)

All other clustering decisions of the other classifiers are aligned to the source signals defined by classifier \( \nu_{\text{align}} \) according to the alignment strategy explained in the introduction of Chapter 4.

One possible advantage of such an MES is the possibility to decide between \( N \) different classifiers not only for whole classifications \( a_\nu \) but on a feature sample basis: Each single feature sample \( i \) can be assigned to a new cluster independent from the assignments for all other feature samples. By this, reliable parts of two or more clustering vectors \( a_\nu \) are mixed together to form a new clustering vector.

In Table 7.6, the separation quality for different MES are shown. The selection of the number of feature spaces to be used (\( N \)) is controlled by the weighting factors \( \beta_\nu \). \( \beta_\nu = 0 \) corresponds to dropping feature space \( \nu \).

The MES based on the cluster reliability results in nearly identical separation quality as the MES based on the adaptation to the tempo of the sources. For the MES based on the WMV, a combination of all three features is necessary to reach this separation quality. In contrast to this, the MES based on tempo adaptation gives the same results with only two feature spaces. This can be interpreted in two ways:
7.5 Combination Techniques beside Audio Data

Similar to Chapter 6, the clustering techniques proposed in this chapter are applied on data \( B-D \). Again the separation quality is measured by the SNR instead of the \( \text{SNR}_{\text{seg}} \) for better comparison with other experiments. It can be seen in Tables 7.4 and 7.5 and Figure 7.4 that it is sufficient to combine only two feature spaces, i.e. the spectral and the temporal SFM. Only the multi-expert-system based on cluster reliability (see also Table 7.6), benefits from the usage of all three feature spaces. Motivated by this, only the results for the combination of the both SFM are shown in Figure 7.6. Several observations can be made:

Firstly, data \( B-D \) prefer either the temporal SFM or the spectral SFM as already mentioned in Section 6.2.3.

Secondly, nearly all combination strategies perform best at the extreme points. This results in choosing only one of the both SFM for each data set. The only exceptions to this are: The CFCM applied to data \( C \) and the linear superposition applied to data \( B \) and \( C \). For these three exceptions, the combination of two clustering strategies leads to a

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
N & F_{\text{MFCF}}^{(1)} & F_{\text{SFM,G}}^{(2)} & F_{\text{set}}^{(0)} & \text{SNR} & \text{SNR}_{\text{seg}} \\
\hline
1 & \beta_1 = 1.00 & \beta_2 = 0.00 & \beta_3 = 0.00 & 5.89 & 8.50 \\
 & 0.00 & 1.00 & 0.00 & 5.84 & 7.70 \\
 & 0.00 & 0.00 & 1.00 & 5.58 & 7.82 \\
\hline
2 & \beta_1 = 0.00 & \beta_2 = 0.60 & \beta_3 = 0.40 & 6.41 & 8.13 \\
 & 0.70 & 0.00 & 0.30 & 6.14 & 8.54 \\
 & 0.60 & 0.40 & 0.00 & 6.38 & 8.63 \\
\hline
3 & \beta_1 = 0.50 & \beta_2 = 0.40 & \beta_3 = 0.10 & 6.69 & 8.72 \\
\hline
\end{array}
\]

Table 7.6: Separation quality is shown for an MES based on the cluster reliability with different number of combined feature spaces \( N \).

- The MES based on WMV is capable of combining even more than two clustering algorithms. Therefore, it is reasonable that the separation quality can be increased by combining even more than three clustering algorithms.
- The MES based on tempo adaptation is faster, because only two clustering algorithms need to be combined to reach the given separation quality of roughly 8.8 dB.

7.4.4 Summary

In this section, combination strategies in the cluster space are introduced and their advantages and disadvantages are outlined. Experimental results for a tempo-based adaptation and an interpretation of the cluster reliability are shown. In the following, only the tempo-based adaptation is discussed any further due to the fact that it leads to better separation quality than the analysis of the cluster reliability.

7.5 Combination Techniques beside Audio Data
Figure 7.6: Combination of spectral and temporal SFM for data $\mathcal{B}$ (solid line), $\mathcal{C}$ (dashed line), and $\mathcal{D}$ (solid line with markers).

separation quality exceeding the qualities of both single clustering strategies used for the combination. For all other combinations, the weighting factor ($\alpha_1$, $\beta_1$, or $\vartheta$) leads to a smooth fading between both clustering strategies. The main reasons for this are the small number of mixtures in each test set beside data $\mathcal{A}$ and the specialization to either pure harmonic instruments for data $\mathcal{C}$ and $\mathcal{D}$ or more rhythmic-based music in data $\mathcal{B}$.

Finally, only the temporal adaptation shown in Figure 7.6(a) results in a single parameter suitable for a large range of audio data: $\vartheta = 1.0$. This is a small deviation to the optimal threshold $\vartheta = 0.9$ shown in Figure 7.4. As can be seen there, even with $\vartheta = 1.0$ good separation quality is possible for data $\mathcal{A}$. Therefore, $\vartheta = 1.0$ is chosen in the following as a good compromise for different sets of audio data.
7.6 Convex Fuzzy C-Means with Adaptation to Tempo

In Section 7.3, the CFCM algorithm is explained. It is shown that this clustering algorithm leads to the best separation quality for data $\mathcal{A}$. As shown in Section 7.5, beside data $\mathcal{A}$ the MES based on tempo-adaptation performs most robust. The MES based on tempo-adaptation can be interpreted as a CFCM clustering with two feature spaces ($F_{\text{MFCC}}$ und $F_{\text{SFM,G}}$) and a single weighting $\alpha_1$ switching between $\alpha_1 = 0$ (high tempo) or $\alpha_1 = 1$ (low tempo). From this point of view it is an obvious idea to combine the tempo-based MES with the CFCM algorithm. For this, the weighting $\alpha_1$ becomes a function of the criterion tempo.

In Section 7.4.2, it is already mentioned that mixtures with lower tempo are better

![Images showing optimal choice for weighting parameter $\alpha_1$ regarding the criterion tempo.](image)

**Figure 7.7:** Optimal choice for weighting parameter $\alpha_1$ regarding the criterion tempo. Figure 7.7(a) shows the optimal $\alpha_1$ without any restriction, Figure 7.7(b) shows the optimal $\alpha_1$ with smoothness constrains, and Figure 7.7(c) shows the optimal $\alpha_1$ if $\alpha_1$ is restricted to be monotonic decreasing for increasing criterion tempo.

separated with spectral clustering and mixtures with higher tempo are better separated
with temporal clustering. To get further insights in the dependencies between the \textit{tempo} and a good choice of $\alpha_1$, data $\mathcal{A}$ is clustered with a CFCM algorithm with the spectral and the temporal SFM and weighting factors $\alpha_1 = \frac{n}{100}$, $0 \leq \alpha_1 \leq 1$, $n \in \mathbb{N}$. This results in 101 clusterings.

In Figure 7.7, three non-blind decision schemes are used to determine a good choice for $\alpha_1$ depending on the tempo:

In Figure 7.7(a), the choice of $\alpha_1$ depends only on the separation quality $\text{SNR}_{\text{seg}}$: This results in the best separation quality. Obviously, no dependencies between the parameter \textit{tempo} and the optimal $\alpha_1$ can be observed.

In Figure 7.7(b), a \textit{smoothness} constraint is used to reduce the noise in the measurement of the optimal $\alpha_1$: The mixtures are sorted by increasing \textit{tempo} and bundled to packages of 150 mixtures. For each of these bundles, an optimal $\alpha_1$ is selected according to the $\text{SNR}_{\text{seg}}$ for these mixtures.

Motivated by the observation in Section 7.4.2, it is an obvious idea to restrict the $\alpha_1$ to be a monotonic decreasing function of the \textit{tempo}. To get this optimal path through all 101 clusterings a trellis-graph is used: The 101 clusterings are the states for each time-step, the different time-steps corresponds to all mixtures sorted according to increasing \textit{tempo}, and the criterion to be maximized is the $\text{SNR}_{\text{seg}}$. For each time-step only state-transitions corresponding to constant or decreasing $\alpha_1$ are allowed. See also [24] or [98] for more details on the trellis-graph and its usage to optimize the path through a set of states.

The non-blind choice corresponding to Figure 7.7(a) results in a separation quality of $\text{SNR}_{\text{seg}} = 10.36$ dB, the smooth decision scheme in Figure 7.7(b) results in an $\text{SNR}_{\text{seg}} = 9.22$ dB, and the choice shown in Figure 7.7(c) in a separation quality of $\text{SNR}_{\text{seg}} = 9.17$ dB. The difference between the smooth and the non-increasing path is negligible. The non-increasing restriction is preferred for two reasons: Firstly, both models (smooth weighting and non-increasing weighting) lead to nearly identical results. Secondly, the set of mixtures beside data $\mathcal{A}$ prefers hard switching of the weighting-factor $\alpha_1$ between $\alpha_1 = 0$ and $\alpha_1 = 1$, as shown in Figure 7.6. Therefore, it can be argued that the non-increasing model can be applied to a larger range of mixtures than the smooth model.

Both mean separation qualities are still higher than the current best blind clustering algorithm, CFCM with a constant $\alpha_1$ resulting in a separation quality of $\text{SNR}_{\text{seg}} = 8.99$ dB, see also Table 7.5.

Motivated by this, the curve in Figure 7.7(c) can be simplified to the following properties:

- For larger \textit{tempo}, $\alpha_1$ converges to zero. By this, only $\mathbf{F}_{\text{SFM,G}}^{(2)}$ is used for clustering.
- For smaller \textit{tempo} $\alpha_1$ converges to one. By this, only $\mathbf{F}_{\text{MFCC}}^{(1)}$ is used for clustering.
- For a \textit{tempo} between 0.5 and 1.0 a value for $\alpha_1$ somewhere in the range $0.2 \leq \alpha_1 \leq 0.5$ seems to lead to good separation quality.

Therefore, it is proposed to approximate the curve of Figure 7.7(c) by a sum of two arctangents:

$$\alpha_1(t) = \frac{1}{2} + \frac{1 - b}{\pi} \arctan (c_1 \cdot (T_1 - t)) + \frac{b}{\pi} \arctan (c_2 \cdot (T_2 - t)),$$  \hspace{1cm} (7.20)

with $t = \text{tempo}$. The parameters $c_1$ and $c_2$ control the width of the crossover between two different values at the positions $T_1$ and $T_2$. The meaning of the other parameters is visualized in Figure 7.8.
7.6 Convex Fuzzy C-Means with Adaptation to Tempo

CFCM uses a concatenated feature space of $\mathbf{F}^{(1)}_{\text{MFCC}}$ and $\mathbf{F}^{(2)}_{\text{SFM,G}}$. For this clustering algorithm, the weighting factor is set to $\alpha_1 = 0.4$ according to the results shown in Table 7.5. Clustering algorithm MES corresponds to the multi-expert-system based on the tempo-adaptation introduced in Section 7.4.2 with $\vartheta = 1.0$. CFCM-A corresponds to the convex fuzzy $c$-means algorithm with weighting $\alpha_1$ being evaluated according to Equation (7.20). Good separation quality is achieved with the following parameters used for Equation (7.20): $b = 0.5$, $c_1 = 20$, $c_2 = 5$, $T_1 = 0.5$, and $T_2 = 1.3$. The resulting separation quality is shown in Tables 7.7 and 7.8.

To show the effectiveness of the CFCM-A algorithm, the two best clustering algorithms

<table>
<thead>
<tr>
<th></th>
<th>$\mathbf{F}^{(1)}_{\text{MFCC}}$</th>
<th>$\mathbf{F}^{(2)}_{\text{SFM,G}}$</th>
<th>CFCM</th>
<th>MES</th>
<th>CFCM-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>5.91</td>
<td>5.82</td>
<td>6.99</td>
<td>6.61</td>
<td>6.96</td>
</tr>
<tr>
<td>B</td>
<td>1.35</td>
<td>3.53</td>
<td>2.60</td>
<td>3.53</td>
<td>2.95</td>
</tr>
<tr>
<td>C</td>
<td>5.55</td>
<td>3.50</td>
<td>5.58</td>
<td>5.60</td>
<td>5.63</td>
</tr>
<tr>
<td>D</td>
<td>10.67</td>
<td>4.88</td>
<td>9.78</td>
<td>9.36</td>
<td>9.86</td>
</tr>
</tbody>
</table>

Table 7.7: Comparison of the different clustering algorithms based on the SNR.

<table>
<thead>
<tr>
<th></th>
<th>$\mathbf{F}^{(1)}_{\text{MFCC}}$</th>
<th>$\mathbf{F}^{(2)}_{\text{SFM,G}}$</th>
<th>CFCM</th>
<th>MES</th>
<th>CFCM-A</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>8.50</td>
<td>7.70</td>
<td>8.99</td>
<td>8.74</td>
<td>9.00</td>
</tr>
<tr>
<td>B</td>
<td>2.87</td>
<td>3.90</td>
<td>3.92</td>
<td>3.90</td>
<td>3.97</td>
</tr>
<tr>
<td>C</td>
<td>9.28</td>
<td>6.93</td>
<td>9.18</td>
<td>9.06</td>
<td>9.40</td>
</tr>
</tbody>
</table>

Table 7.8: Comparison of the different clustering algorithms based on the SNR$_{\text{seg}}$.

for each set of mixtures are marked bold. It can be seen that only for data $\mathcal{B}$ and the measurement by SNR, the CFCM-A do not belong to the two best clustering algorithms. Additionally, it can be seen that CFCM-A performs better than CFCM with constant weights. The only exception is data $\mathcal{A}$ evaluated with the SNR: In this case, nearly equal
separation quality is achieved by both algorithms. Therefore, it can be concluded that CFCM-A combines the good separation quality of the CFCM with the robustness of the MES based on tempo-adaptation in an appropriate way.

To summarize the currently best clustering algorithm CFCM-A, all necessary processing steps are visualized in Figure 7.9.

![Figure 7.9: Signal flow of the CFCM-A algorithm.](image)

**7.7 Summary**

In this chapter, several strategies are explained for combining different basic clustering algorithms already introduced in Chapter 6. Basically, these combination strategies differ in the point in the signal flow where they are applied:

- Before the clustering by combination in feature space according to Section 7.3.
- After the clustering by interpretation of the clustering decision according to Section 7.4.
- After signal synthesis by linear superposition of the separated sources according to Section 7.2.

By experiments it can be shown that two approaches are most interesting: Combination after the clustering by adaptation based on the tempo of the mixture and combination in
feature space by the CFCM algorithm. The results of the first algorithm are very robust: It leads to good separation quality for a wide range of audio data. The second algorithm leads to best separation quality regarding data $\mathcal{A}$, which can be considered most trustworthy because data $\mathcal{A}$ contains several hundreds of mixtures. On the other hand, this second algorithm cannot be applied on all sets of data ($\mathcal{A} \rightarrow \mathcal{D}$) with a single set of parameters, thus being less robust.

As shown in Section 7.6, the hard decision between two clustering algorithms based on the feature tempo can be interpreted as a CFCM clustering with weightings for the single feature spaces being set either to 0 or to 1. Therefore, the CFCM-A algorithm is introduced, which utilizes adaptive weightings controlled by the feature \textit{tempo}. This clustering algorithm is as robust as the standard temporal-adaptation but leads to significantly better separation quality.
Chapter 8

Summary and Conclusions

8.1 Summary

Blind source separation is a topic of ongoing interest either as a pre-processing step for arbitrary audio analysis frameworks or for re-/up-mixing of audio streams. Many state-of-the-art algorithms are based on the non-negative tensor factorization (NTF). This thesis addresses one short-coming of the NTF: It separates only notes but not whole melodies consisting of several (different) notes of one single instrument.

State-of-the-art algorithms typically address this problem by extensions of the NTF with appropriate instrument models. These extensions usually results in a high computational complexity. Another approach is to extract characteristic audio features useful for clustering the single separated notes into melodies. In this thesis, the latter approach is discussed. After a short introduction of the necessary mathematics a brief overview is given over current techniques used for blind source separation. The basic framework used in this thesis is explained in more details. After this, first experiments are done to get first insights in useful parameter settings regarding the time-frequency processing and the note separation.

After that, the main part of the thesis starts with the introduction of useful audio features, unsupervised clustering algorithms and their strengths and weaknesses. Good pairs of audio features and clustering algorithms are shown by experiments. Finally, three strategies for combining different clustering algorithms are introduced to increase the separation quality further: Combination before clustering, after clustering, and after signal synthesis.

The best clustering algorithm uses a weighted combination before clustering with weights adapted to the tempo of the mixture.

The clustering algorithms discussed in this thesis fulfill the following requirements.

- They can be used unsupervised. No interaction of humans is necessary up to the signal synthesis step.
- Their robustness is tested on different sets of mixtures to assure the parameters to be as universally valid as possible.
- All clustering algorithms can be evaluated in a fraction of the time necessary for the NTF. Thus, their computational complexity can be ignored and the overall blind source separation algorithm becomes very fast compared to separation frameworks based on factorization methods with embedded models.
Additionally, the proposed blind source separation framework has the following advantages: It is able to handle even monaural signals because no spatial information is used during the signal processing.

### 8.2 Outlook

So far it is already clear what the proposed algorithms can do. On the other hand, there are some open points:

- The algorithms classify streams, **not** instruments. A final classification which stream belongs to instrument 1 and which stream to instrument 2 (in the case of two active instruments) is an open problem.

- The algorithms are not able to count the sources. The number of active instruments has to be provided by the user.

Additionally, it is obvious that each clustering algorithm can only be tuned up to a certain hit-rate for a large range of audio mixtures. Perfect clustering can only be possible for a small number of mixtures. This is the main reason for using the mean SNR or the mean SNR_{seg} over a large number of mixtures as a measure of the separation quality. It is an open point to improve the used clustering algorithms for further increasing the hit-rate.

Another current issue is the question if the separation quality of the blind clustering algorithms proposed here, can be improved by exchanging the basic NTF with the more complex variant: The non-negative tensor deconvolution algorithm, explained e.g. in [25]. It is reasonable that such a deconvolution is advantageous for speech and other signals with fast changes in the spectrum. On the other hand, it is shown in this thesis that all algorithms based on non-negative factorization and or non-negative deconvolution are based on a strong compression regarding the degrees of freedom. Therefore, it is not clear if the increasing number of degrees of freedom by allowing deconvolution instead of factorization is advantageous or not.

In [27], it is proposed to initialize the given BSS algorithm for certain scenarios by the separated sources of another BSS framework. Such mutual initialization is possible with a large range of other BSS frameworks: It is possible to use the proposed BSS framework for initialization of other BSS frameworks, e.g. [27] or [54]. On the other hand, it is possible to initialize the clustering used in this thesis by the source signals separated by arbitrary other BSS frameworks: For this, the factorized components \( y_i(n,c) \) can be mapped onto the separated sources used for initialization e.g. by reference clustering. This mapping can be used for initialization of the clustering algorithms.

One additional open point is an objective quality measure with good fitting of the mean opinion score to enable better comparison between the different separation strategies.
Appendix A

Detailed Description of Data $\mathcal{A}$

In Table A.1, the filenames of data $\mathcal{A}$ are listed to give a more detailed overview over the content of data $\mathcal{A}$. The filenames are organized as follows:
The description starts with the type of source and the number of this source. The possible file types are: H for human sources, IE for electronic and popular music, IP for percussive instruments, IS for string instruments, IW for wind instruments, and finally N for noise. The second part of the filename represents the origin of the wave-file: EBU for the sound quality assessment material (SQAM) [99], BassDB for the blind audio source separation site [45], GB for the samples used by a sample based synthesizer of Apple, and finally FHMDetmold for samples provided by the Fachhochschule für Musik - Detmold. The remaining name of the audio file is the original filename of the corresponding database.
<table>
<thead>
<tr>
<th>No.</th>
<th>Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>H01_EBU_Soprano_tec_sqam_44bw_cmc6-12467.wav</td>
</tr>
<tr>
<td>2</td>
<td>H02_EBU_Alto_tec_sqam_56bw_cmc6-12468.wav</td>
</tr>
<tr>
<td>3</td>
<td>H03_EBU_Tenor_tec_sqam_55bw_cmc6-12469.wav</td>
</tr>
<tr>
<td>4</td>
<td>H04_EBU_Bass_tec_sqam_57bw_cmc6-12470.wav</td>
</tr>
<tr>
<td>5</td>
<td>H05_EBU_FemaleEnglish_tec_sqam_58bw_cmc6-12472.wav</td>
</tr>
<tr>
<td>6</td>
<td>H06_EBU_MaleEnglish_tec_sqam_59bw_cmc6-12473.wav</td>
</tr>
<tr>
<td>7</td>
<td>H07_EBU_FemaleFrench_tec_sqam_57bw_cmc6-12474.wav</td>
</tr>
<tr>
<td>8</td>
<td>H08_EBU_MaleFrench_tec_sqam_55bw_cmc6-12475.wav</td>
</tr>
<tr>
<td>9</td>
<td>IE01_BassDB_EBass.wav</td>
</tr>
<tr>
<td>10</td>
<td>IE02_BassDB_EGuitar.wav</td>
</tr>
<tr>
<td>11</td>
<td>IE03_BassDB_Keyboard.wav</td>
</tr>
<tr>
<td>12</td>
<td>IE04_BassDB_Drums.wav</td>
</tr>
<tr>
<td>13</td>
<td>IE05_GB_Deep Pad 02.wav</td>
</tr>
<tr>
<td>14</td>
<td>IE06_GB_Blue Jazz Organ Riff 02.wav</td>
</tr>
<tr>
<td>15</td>
<td>IP01_EBU_BassDrum_tec_sqam_57bw_cmc6-12526.wav</td>
</tr>
<tr>
<td>16</td>
<td>IP02_EBU_Castanets_tec_sqam_58bw_cmc6-12521.wav</td>
</tr>
<tr>
<td>17</td>
<td>IP03_EBU_Celesta_tec_sqam_59bw_cmc6-12519.wav</td>
</tr>
<tr>
<td>18</td>
<td>IP04_EBU_Claves_tec_sqam_37bw_cmc6-12518.wav</td>
</tr>
<tr>
<td>19</td>
<td>IP05_EBU_Cymbals_tec_sqam_58bw_cmc6-12517.wav</td>
</tr>
<tr>
<td>20</td>
<td>IP06_EBU_Cymbals_tec_sqam_59bw_cmc6-12516.wav</td>
</tr>
<tr>
<td>21</td>
<td>IP07_EBU_Gong_tec_sqam_56bw_cmc6-12535.wav</td>
</tr>
<tr>
<td>22</td>
<td>IP08_EBU_KettleDrums_tec_sqam_38bw_cmc6-12522.wav</td>
</tr>
<tr>
<td>23</td>
<td>IP09_EBU_KettleDrums_tec_sqam_39bw_cmc6-12523.wav</td>
</tr>
<tr>
<td>24</td>
<td>IP10_EBU_SideDrum_tec_sqam_40bw_cmc6-12524.wav</td>
</tr>
<tr>
<td>25</td>
<td>IP11_EBU_SideDrumSnare_tec_sqam_41bw_cmc6-12525.wav</td>
</tr>
<tr>
<td>26</td>
<td>IP12_EBU_Triangle_tec_sqam_42bw_cmc6-12532.wav</td>
</tr>
<tr>
<td>27</td>
<td>IP13_EBU_TubularBells_tec_sqam_43bw_cmc6-12537.wav</td>
</tr>
<tr>
<td>28</td>
<td>IP14_EBU_Vibraphone_tec_sqam_44bw_cmc6-12542.wav</td>
</tr>
<tr>
<td>29</td>
<td>IP15_EBU_Xylophone_tec_sqam_43bw_cmc6-12540.wav</td>
</tr>
<tr>
<td>30</td>
<td>IP16_GB_Conga Groove 03.wav</td>
</tr>
<tr>
<td>31</td>
<td>IS01_EBU_Double-bass_tec_sqam_11bw_cmc6-12494.wav</td>
</tr>
<tr>
<td>32</td>
<td>IS02_EBU_Guitar_Sarasate.wav</td>
</tr>
<tr>
<td>33</td>
<td>IS03_EBU_Harp_tec_sqam_25bw_cmc6-12518.wav</td>
</tr>
<tr>
<td>34</td>
<td>IS04_EBU_Harpichord_tec_sqam_26bw_cmc6-12549.wav</td>
</tr>
<tr>
<td>35</td>
<td>IS05_EBU_Piano_tec_sqam_27bw_cmc6-12545.wav</td>
</tr>
<tr>
<td>36</td>
<td>IS06_EBU_Violin_tec_sqam_28bw_cmc6-12540.wav</td>
</tr>
<tr>
<td>37</td>
<td>IS07_EBU_Violoncello_tec_sqam_29bw_cmc6-12542.wav</td>
</tr>
<tr>
<td>38</td>
<td>IS08_EBU_Viola_tec_sqam_30bw_cmc6-12540.wav</td>
</tr>
<tr>
<td>39</td>
<td>IS09_GB_Eastern Santoor 01.wav</td>
</tr>
<tr>
<td>40</td>
<td>IS10_GB_Exotic Sarod 01.wav</td>
</tr>
<tr>
<td>41</td>
<td>IS11_GB_Folk Mandolin 07.wav</td>
</tr>
<tr>
<td>42</td>
<td>IS12_GB_Middle Eastern Oud 03.wav</td>
</tr>
<tr>
<td>43</td>
<td>IS13_GB_Nordic Keyed Fiddle 02.wav</td>
</tr>
<tr>
<td>44</td>
<td>IW01_EBU_Accordion_tec_sqam_42bw_cmc6-12463.wav</td>
</tr>
<tr>
<td>45</td>
<td>IW02_EBU_Accordion_tec_sqam_43bw_cmc6-12464.wav</td>
</tr>
<tr>
<td>46</td>
<td>IW03_EBU_Accordion_tec_sqam_44bw_cmc6-12465.wav</td>
</tr>
<tr>
<td>47</td>
<td>IW04_EBU_Accordion_tec_sqam_45bw_cmc6-12466.wav</td>
</tr>
<tr>
<td>48</td>
<td>IW05_EBU_Accordion_tec_sqam_46bw_cmc6-12467.wav</td>
</tr>
<tr>
<td>49</td>
<td>IW06_EBU_Accordion_tec_sqam_47bw_cmc6-12468.wav</td>
</tr>
<tr>
<td>50</td>
<td>IW07_EBU_Accordion_tec_sqam_48bw_cmc6-12469.wav</td>
</tr>
<tr>
<td>51</td>
<td>IW08_EBU_Accordion_tec_sqam_49bw_cmc6-12470.wav</td>
</tr>
<tr>
<td>52</td>
<td>IW09_EBU_Accordion_tec_sqam_50bw_cmc6-12471.wav</td>
</tr>
<tr>
<td>53</td>
<td>IW10_EBU_Accordion_tec_sqam_51bw_cmc6-12472.wav</td>
</tr>
<tr>
<td>54</td>
<td>IW11_EBU_Accordion_tec_sqam_52bw_cmc6-12473.wav</td>
</tr>
<tr>
<td>55</td>
<td>IW12_EBU_Accordion_tec_sqam_53bw_cmc6-12474.wav</td>
</tr>
<tr>
<td>56</td>
<td>IW13_EBU_Accordion_tec_sqam_54bw_cmc6-12475.wav</td>
</tr>
<tr>
<td>57</td>
<td>IW14_EBU_Accordion_tec_sqam_55bw_cmc6-12476.wav</td>
</tr>
<tr>
<td>58</td>
<td>IW15_EBU_Accordion_tec_sqam_56bw_cmc6-12477.wav</td>
</tr>
<tr>
<td>59</td>
<td>IW16_GB_Blues Harmonica 01.wav</td>
</tr>
<tr>
<td>60</td>
<td>N01_EBU_PinkNoise_tec_sqam_52bw_cmc6-12481.wav</td>
</tr>
</tbody>
</table>

Table A.1: Detailed description of data A by its filenames.
Appendix B

Additional Factorization Methods beside Standard NTF

B.1 Multichannel Factorization by Mid-Side Encoding

Mid-side encoding is a technique to encode stereo informations. It is based on the assumption that both stereo signals are very similar. The underlying assumption is that an instantaneous model is used for mixing. In the case of convolutive mixing, the pure mid-side encoding model is no longer true. Of course, for amplitude spectrograms or power spectrograms short delays are not critical.

Assuming an instantaneous mixing model, the both channels are transformed into a mid-signal $x_m(n, c)$ and a side-signal $x_s(n, c)$ e.g. by

$$x_m(n) = x(n, 1) + x(n, 2), \quad (B.1)$$
$$x_s(n) = x(n, 1) - x(n, 2). \quad (B.2)$$

This encoding can be inverted at decoder side in a lossless way.

Motivated by this encoding technique, the multi-channel tensor $X(k, t, c)$ can be transformed into a mid-side encoded tensor $V(k, t, c)$ by

$$V(k, t, 1) = \sum_{c=1}^{C} X(k, t, c), \quad (B.3)$$
$$V(k, t, c) = X(k, t, c) - X(k, t, 1), \quad \text{for } c > 1. \quad (B.4)$$

The matrices $B$ and $G$ are used for the approximation of the mid-signal $V(k, t, 1)$. By this, the faster NMF can be used instead of the NTF.

For the spatial information encoded in matrix $A$ two possible approximations will be discussed:

- A matrix $A(c, i)$ is used for encoding the instantaneous mixing model as given in Equation (2.33).
- A tensor $A(k, i, c)$ is used for encoding the convolutive mixing model $X(k, t, c) \approx \sum_{i=1}^{I} A(k, c, i) B(k, i) G(t, i)$. In this convolutive scenario, the factorization of matrix $V(k, t, 1)$ by the matrices $B$ and $G$ resolves the ambiguity between matrix $B$ and tensor $A$. 
The multiplicative update rules for $A$ according to the $\beta$-divergence are in the latter case:

$$A(k, i, c) \leftarrow A(k, i, c) \frac{\sum_t \xi_1(k, t, c) G(t, i) B(k, i)}{\sum_t \xi_2(k, t, c) G(t, i) B(k, i)}, \quad (B.5)$$

with $\xi_1(k, t, c)$ and $\xi_2(k, t, c)$ defined as e.g. in Equation (2.44).

Unfortunately, these approaches lead to worse separation quality compared to the multi-channel factorization methods compared in Figure 4.10. Therefore, they are not considered any further.

### B.2 Expectation Maximization vs. Multiplicative Updates

In addition to the experiments mentioned in Section 4.3.3, the multiplicative update rules are compared to the expectation maximization algorithm introduced in [3]. A more complex model for the EM-based NTF is derived in [27]: It is shown that the EM-version of this algorithm leads to increasing separation quality for increasing number of iterations. Contrary to this, in [27] it is reported that the multiplicative update version of their algorithm is not stable, thus leading to decreasing separation quality for a number of iterations above a given optimum. This effect can also be verified for the basic NTF combined with the semantic initialization as shown in Figure 4.8. On the other hand, in our experiments the EM-based NTF leads to worse separation quality compared to the multiplicative update rules. Therefore, the EM-variant of the NTF is not considered in this thesis.
Appendix C

Separation Quality for Single Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>$\text{SNR}_{\text{seg}}$</th>
<th>$\text{SNR}_{\text{seg}}$</th>
<th>$\lambda_{\text{opt}}$</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{\text{HR1}}^{(1)}$</td>
<td>5.58</td>
<td>5.58</td>
<td>1.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{HR2}}^{(1)}$</td>
<td>5.46</td>
<td>5.46</td>
<td>1.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{HR3}}^{(1)}$</td>
<td>5.68</td>
<td>5.68</td>
<td>1.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{HR4}}^{(1)}$</td>
<td>5.60</td>
<td>5.65</td>
<td>0.50</td>
<td>1.00</td>
</tr>
<tr>
<td>$F_{\text{HR5}}^{(1)}$</td>
<td>5.72</td>
<td>5.74</td>
<td>0.50</td>
<td>0.10</td>
</tr>
<tr>
<td>$F_{\text{HR6}}^{(1)}$</td>
<td>5.73</td>
<td>5.83</td>
<td>0.50</td>
<td>0.10</td>
</tr>
<tr>
<td>$F_{\text{HR7}}^{(1)}$</td>
<td>5.68</td>
<td>5.83</td>
<td>0.50</td>
<td>1.00</td>
</tr>
<tr>
<td>$F_{\text{HR8}}^{(1)}$</td>
<td>5.69</td>
<td>5.86</td>
<td>0.50</td>
<td>0.10</td>
</tr>
<tr>
<td>$F_{\text{HR9}}^{(1)}$</td>
<td>5.72</td>
<td>5.91</td>
<td>0.50</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{partial1}}^{(0)}$</td>
<td>4.92</td>
<td>5.14</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{partial2}}^{(0)}$</td>
<td>5.08</td>
<td>5.15</td>
<td>2.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{partial3}}^{(0)}$</td>
<td>4.99</td>
<td>5.33</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{partial4}}^{(0)}$</td>
<td>4.79</td>
<td>5.29</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{partial5}}^{(0)}$</td>
<td>4.99</td>
<td>5.56</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>$F_{\text{partial6}}^{(0)}$</td>
<td>4.97</td>
<td>4.97</td>
<td>1.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{partial7}}^{(0)}$</td>
<td>4.91</td>
<td>5.02</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{SC2}}^{(1)}$</td>
<td>5.74</td>
<td>5.92</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{SC3}}^{(1)}$</td>
<td>5.22</td>
<td>5.40</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{SC2I}}^{(0)}$</td>
<td>5.27</td>
<td>5.27</td>
<td>1.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{ASF}}^{(1)}$</td>
<td>5.04</td>
<td>5.28</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{AM1}}^{(1)}$</td>
<td>4.77</td>
<td>4.77</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{AM3}}^{(1)}$</td>
<td>4.77</td>
<td>4.77</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{AM2}}^{(1)}$</td>
<td>4.77</td>
<td>4.77</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{AM4}}^{(1)}$</td>
<td>4.77</td>
<td>4.77</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>$F_{\text{crest}}^{(0)}$</td>
<td>4.84</td>
<td>5.02</td>
<td>0.00</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Table C.1: Separation quality of single features (feature matrices with minimum size). Here, only features resulting in a separation quality smaller than 6 dB are shown.
Appendix D

Usage of Start and Stop Windows in the Context of Blind Source Separation

Typical analysis windows are symmetric, e.g. the Hann or the Blackman window. Asymmetric analysis windows are used e.g. by the Fourier-t-transform [100] to adapt the time-frequency resolution of the transform to the resolution of the human ear. Another example for the usage of asymmetric analysis windows is the adaptive time-frequency resolution visualized in Figure F.1.

To verify that the usage of start and stop windows did not lead to additional distortions in our BSS framework the following experiment is done:

All mixtures of data $\mathcal{A}$ are separated by the NTF and clustered with the reference clustering. Time-frequency analysis is done with only long windows, only short windows, switching windows from start to stop window periodically starting with the long flank and starting with the short flank. The separation quality for these four cases is shown in Table D.1. It can be seen that asymmetric analysis and synthesis windows perform as good as only long or only short windows. It can be concluded that the asymmetry of the analysis windows did not lead to additional distortions.

Figure D.1: Window switching scheme for evaluating the influence of start and stop windows on the proposed BSS framework.
Table D.1: Separation quality for different dynamic differences of the input signals and different
time-frequency resolutions: Long corresponds to a sine window with a window size
of 4096 samples and a hop size of 2048 samples. The short case uses the same
window with half the window size and hop size as for the long case. Long → short
and short → long corresponds to a time-frequency transform, which uses only start
and stop windows as shown in Figure D.1.
Appendix E

Applications of Blind Source Separation
- Beside Music

So far only music and related sources like human speech are considered. Beside music and related sources other possible applications of source separation arise e.g. in the area of vehicle noise. In [101], a large bunch of BSS algorithms, e.g. ISA, NTF, and PCA are applied on the task of separating the wind noise from the noise induced by the tires. Both types of noises can be considered as steady state in the case of constant speed of the car used for measurement and in the case of constant bottom covering of the road. As can be seen in Figure 4.4, the NTF benefits from steady state segments of the signal. It is necessary that the different components of the signal have different temporal envelopes which can be interpreted as at least two different states in the spectrogram. This restriction is also true for the ISA.

Based on this observation it seems to be necessary that something changes the spectra or the temporal envelopes during the measurement of the two noises: Wind and tire noise. These changes can include but are not limited to:

- A change in the bottom cover of the road will change the noise of the tires but the noise of the wind will be unchanged if the speed of the car remains constant.
- A switching of the tires followed by a second measurement will lead to (nearly) identical wind noise but different tire noise.

Concatenating the spectrograms of both measurements shall reveal a change in the spectral structure at the time-position of the change in the measurement-scenario.

In [53], an algorithm is introduced to utilize such abrupt changes in the spectrogram by a score-informed sparseness for the matrix $G$ of the NTF: In Figure E.1(a), three random spectra are plotted as a toy example for the

- noise of the wind at constant speed,
- noise of the tires at measurement 1, and
- noise of the tires at measurement 2.

It is not important to know why the spectra changes or how the spectra differs. Only the timestamp of the change has to be known. The corresponding envelopes are plotted in Figure E.1(b): The switching on/off of the single tire-noises can be seen clearly in the middle of the temporal axis.
Applications of Blind Source Separation - Beside Music

Figure E.1: Spectra and their corresponding envelopes during the measurement (toy example).

(a) input spectra  
(b) input envelopes

Figure E.2: Spectrogram of the mixture and the initialization of matrix $G$ (toy example).

(a) spectrogram of the mixture  
(b) initial envelopes

In Figure E.2(a), the resulting spectrograms observed during both measurements are concatenated and plotted. The change in spectrum after time-position $t = 10$ is obviously. The correct separated spectra are not such obviously. An NTF with non-informed initialization will surely fail in separation of the different noisy signals. The main reason for this is the massive overlapping of the three sources in the spectrogram.

In Figure E.2(b), the initialization of matrix $G$ according to the score-informed sparseness algorithm introduced in [53] is plotted. The only information necessary for this initialization is the length of the signal segments, or equivalent, the time-slot to switch the first tire-noise off and to switch the second tire-noise on. Of course, this information is easy to get.

In Figure E.3, the corresponding separation results are shown after applying the NTF. Due to the multiplicative update rules, the elements of matrix $G$ initialized with zeros...
remain zero during the iterative optimization by the NTF. By this, the spectra of the
different noise-types can be extracted.
Of course, this is only a toy example used for visualization. To show the effectiveness or
the lack of effectiveness of this algorithm real measurements under realistic circumstances
are necessary. Nevertheless, the possible solution of the extreme hard problem of sepa-
rating two different types of noise is outlined here to show a possible application for the
score-informed sparseness algorithm and to show a possible solution to the problem of
separating two different types of noise.

**Figure E.3:** Spectra and their corresponding envelopes estimated by the NTF (toy example).
Appendix F

Interaction of Time-Frequency Transform and Non-Negative Tensor Factorization

In Chapter 4, the time-frequency processing by the STFT and the note factorization by the NTF is considered each on its own. In this chapter, a closer look onto the interaction between both signal processing steps is taken. In Section F.1, different possibilities for adaptive time-frequency transforms are introduced. Several objectives are discussed to decide which time-frequency resolution has to be chosen and their corresponding disadvantages are explained. In Section F.2, three experiments are introduced to explain the open problems that have to be solved before the time-frequency resolution is adapted to the mixture. This chapter is closed in Section F.3 with a summary why adaptive time-frequency transforms must fail to improve the separation quality of the proposed BSS framework.

F.1 Adaptive Time-Frequency Processing

In this section, the basic ideas of adaptive time-frequency processing are introduced. In audio coding, e.g. advanced audio coding (AAC), adaptive time-frequency resolution (ATFR) is used for increasing audio quality in transient audio segments. Longer analysis windows result in better frequency resolution at the cost of worse time resolution. In encoding, this can result e.g. in temporal smearing of transient regions. Motivated by the increasing quality in the case of audio encoding, several algorithms are tested for adapting the time-frequency resolution of the STFT to the needs of the NTF, e.g. in [102], [103], or [104].

F.1.1 Adaptive Short Time Fourier Transform

In the following, three algorithms for adaptive time-frequency resolution are introduced and their possible shortcomings are discussed.
Adaptivity by Asymmetric Windows  Usually, the parameter window size of the analysis and synthesis windows is switched for changing the time-frequency resolution of the STFT. One possibility for switching the window size is the usage of start- and stop windows as shown in Figure F.1. By this approach, the hop size is changed, when the time-frequency resolution is changed. This results in a kind of zoom-in effect for the local spectrogram (more columns are evaluated).

One possible disadvantage of this ATFR is the non-uniform sampling of the temporal axis of the spectrogram. In Chapter 5, features will be introduced that analyse the temporal axis of the spectrogram, e.g. the envelope analysis in Section 5.2.2. This analysis becomes more complex by using variable hop size.

Adaptivity by Symmetric Windows  Another approach for ATFR is proposed in [105]: The hop size of the analysis remains constant. Analysis windows of different lengths, but of identical type, are used. This adaptive scheme is shown in Figure F.2. It is used e.g. in [102] and [104].

One disadvantage of this ATFR approach is the violation of the condition for perfect reconstruction as defined in Equation (2.19). Therefore, it is necessary to divide the separated signals after the inverse STFT by the sum over the applied analysis and synthesis windows. This cumulated sum is shown as a solid line in Figure F.2. The non-constant function can be interpreted as paying more attention on the samples that corresponds to regions with peaks in the summation over all analysis windows. This effect is usually unwanted, because it is not guaranteed that these segments are more important for the analysis by the NTF. Additionally, the separation quality SNR_{seg} assigns equal importance to each temporal segment.

Adaptivity with Symmetric Windows and Non Constant Overlap-Add  A third approach for ATFR is shown in Figure F.3. By this approach asymmetric analysis windows and non-constant overlap-add are avoided. Therefore, analysis windows are constructed in the following way: The short analysis window is a standard sine window. The long
Figure F.2: Adaptive time-frequency analysis by the scheme proposed in [105]. The single analysis windows are plotted in a dashed style, the sum over all analysis windows is plotted in a solid style.

Figure F.3: Symmetric analysis windows for ATFR with constant overlap-add.

analysis window has the same flanks as the sine window. In the middle section the window is lengthened by a constant value (here: 1). By this, the analysis windows are symmetric and the constant overlap-add condition is fulfilled. Again, an analysis of the temporal axis of the spectrogram becomes difficult because of the non-uniform sampling of the temporal axis.
F.1.2 Strategies for Adaption of Time-Frequency Resolution

In the following, three different approaches for using the above mentioned adaptive analysis windows in an BSS framework.

Adaptivity based on Transients

A first approach on ATFR is discussed in [102] and [106]. The basic idea is to detect transient segments by an appropriate detection algorithm. Then, for transient segments the time resolution is increased and (of course) the frequency resolution is reduced. Two possible transient-detectors are implemented:

- Phase deviation according to [9]: For transient segments, the second derivative of phase over time is assumed to be non-constant.
- Energy concentration according to [107]: Harmonic segments are assumed to have sparse spectra, but for transients, the energy is distributed over much more frequency-bins.

The high-frequency content, according to [108], is also considered, but lead to slightly worse results compared to the energy concentration measure. The transient detection algorithms are applied to a spectrogram with constant time-frequency resolution. Here, the long analysis windows are used as standard resolution. The output of both transient detectors is a single scalar \( \varphi(t) \) for each column \( t \) of the spectrogram \( X \). The pdf of these scalar values is assumed to be a normal distribution. The mean value \( \mu \) and the standard deviation \( \sigma \) are estimated for each mixture. The final detection output is normalized to the measured distribution:

\[
\text{transient} = \begin{cases} 
\text{true}, & \text{if } \varphi(t) > \mu + \theta \cdot \sigma \\
\text{false}, & \text{otherwise,} 
\end{cases}
\]  

with \( \theta \) being a user defined parameter to control the sensitivity of the detection function.

Temporal segments that are classified as transient are analysed by short analysis windows, according to the schemes shown in Figure F.1 or F.2. Best results are achieved for the phase-deviation measure, the ATFR scheme shown in Figure F.1, and a longer window size of 2^{12} samples and a shorter window size of 2^{11} samples. With this configuration, the SNR can be increased by roughly 0.2 dB. Two things make the direct comparison with our results difficult: Firstly, the data \( \mathcal{A} \) contains 60 input signals contrary to the 40 input signals used in [106]. Secondly, the quality measure has switched to \( \text{SNR}_\text{seg} \), because it is assumed to have a higher correlation to human perception of audio quality.

Applying the proposed algorithm to our new (larger) test data results in no gain in separation quality compared to the scenario with constant time-frequency resolution. This may have several reasons. It can be observed that all algorithms adapting the time-frequency resolution are very sensitive to the amount of data in the test set. For a small number of mixtures, it is very simple to train a good classifier for detecting regions that have to be analysed with short windows in order to increase the separation quality significantly by ATFR. Unfortunately it can be observed that such a classifier cannot be generalized to other data without major impact on the separation quality due to over-fitting the classifier. Another open problem of this transient-based approach is that the processing steps
NTF and clustering are not considered by the ATFR. Additionally, up to now it is unclear, whether the NTF benefits from better time resolution in transient regions.

**Adaptivity using BSS as a Black Box**

![Figure F.4: The signal processing blocks NTF and clustering are modeled as a black box for simpler evaluation of the effects of these blocks on the optimal ATFR scheme.](image)

A second approach used in [103], is based on the observation that the separation quality can be increased significantly, by switching the time-frequency resolution in certain segments that have nothing to do with transient/steady-state decision. Additionally, to consider the influence of the NTF and the clustering step more efficiently, this approach assumes the signal blocks NTF and clustering as a black box, as shown in Figure F.4. The mixture signal \( x(n, c) \) is segmented into blocks of length \( 0.33 \leq T_0 \leq 4 \) seconds. Each segment is transformed with a single constant window size: \( w_s = 4096 \) or \( w_s = 2048 \). By this approach, start and stop windows are only used at the borders of each segment to minimize the influence of these asymmetric windows. Depending on the length of \( x(n, c) \) and the chosen block-length the mixture is segmented into \( N_0 \) temporal segments. Thus, \( 2^{N_0} \) different adaptive STFTs are possible in this framework. Each of these \( 2^{N_0} \) spectrograms is fed in the BSS algorithm and the resulting separation quality is measured.

The AdaBoost algorithm is used for training a classifier that decides for each segment if it has to be transformed by a long window size or a short one. For this, a set of features is evaluated for each segment, e.g. the classical transient detection features like phase deviation, high frequency content, and energy compactness. Contrary to [102], additionally a set of MPEG-7 low-level descriptors (e.g. audio spectrum flatness and audio power) and other audio features (e.g. MFCC) are used to train the classifier. For a detailed description of all used features, please refer to [103].

The main results of this approach are:

- Even with a global switching of the window size parameter (using only long or only short windows for the whole mixture) can lead to a gain > 0.5 dB in separation quality.

- The larger the number of mixtures used for training, the lower the separation quality of the adaptive approach based on the trained classifiers. Therefore, it can be concluded that the classification of useful time-frequency resolution resulting in high separation quality is not trivial. The classifiers trained by this approach cannot be applied to large sets of audio data, and therefore can be considered as over-fitted.

Because of the last point, this approach is not considered any further in this thesis.
Adaptivity by Sinusoid Modeling

Sinusoid modeling or sinusoid plus residual modeling (or sinusoid plus transient modeling – STM) is a parametric audio coding technique. According to [10], the basic idea is to encode the $R$ sinusoids of a signal by a small set of parameters (instantaneous amplitude $a_r(n,c)$ and phase $\theta_r(n,c)$):

$$x(n,c) \approx x_s(n,c) = \sum_{r=1}^{R} a_r(n,c) \cos(\theta_r(n,c)).$$  \hfill (F.2)

The residual of this approximation will be called *transient signal* $x_t(n,c)$ in the following:

$$x(n,c) = x_s(n,c) + x_t(n,c).$$  \hfill (F.3)

The general idea of adaptive time-frequency resolution by STM is shown in Figure F.5.

![Figure F.5: Adaptive time-frequency resolution by sinus-transient modeling as a pre-processing step.](image)

The mixture $x(n,c)$ is separated into an approximation of the harmonic parts of the signal $x_s(n,c)$ and the corresponding transient signal according to Equation (F.3). The harmonic signal is separated into $I_1$ components $y_{i,1}(n,c)$ by the blind source separation algorithm BSS$_1$, the transient signal is separated into $I_2$ components $y_{i,2}(n,c)$ by BSS$_2$. Adaptive time-frequency resolution is obtained by using different parameters for window size and
hop size for both BSS algorithms $\text{BSS}_1$ and $\text{BSS}_2$. The reference clustering introduced in Algorithm 2 is used as clustering algorithm for better comparison of the adaptive approach with the standard BSS algorithm with constant time-frequency resolution.

Two STM approaches are tested:

- A peak picking applied on the spectrogram followed by parabolic interpolation of the frequency peaks for better frequency resolution \cite[p.410]{10} (\textit{sinemodel.m}).
- A median filtering applied on the spectrogram, as proposed in \cite{52}.

Both algorithms are restricted by the time-frequency resolution of the underlying STFT. The advantage of the first algorithm is the improved time-frequency resolution by a parabolic interpolation. The second algorithm has the advantage that the horizontal and vertical median filtering is well adapted to the following NTF, which can only extract horizontal and vertical structures out of the spectrogram, see also Figure 4.4 for more details on this effect of the NTF.

Unfortunately, this approach did not improve the separation quality compared to the standard BSS algorithm. This can be explained by the mixture of the castanets and the double-bass. An STM should separate these both instruments as good as possible. Dropping the BSS algorithms and setting

\[
\tilde{s}_1(n,c) = x_s(n,c), \quad \text{and} \\
\tilde{s}_2(n,c) = x_t(n,c),
\]

leads to an SNR of 15.51 dB. This is a very low separation quality compared to the 32.75 dB reached with our separation algorithm with constant time-frequency resolution. Therefore, in the following the adaptivity by STM is not considered any further.

\subsection*{F.1.3 Summary}

In this section, a set of algorithms is introduced for adapting the time-frequency resolution to the needs of the proposed BSS algorithm. These algorithms lead to a small average gain ($\approx 0.2$ dB) in separation quality under the condition of well defined parameters and a small number of mixtures in the test set, see also \cite{102} or \cite{103} for further details. Unfortunately, it is up to now unclear if a higher time resolution in transient segments is beneficial for the proposed BSS algorithm.

\section*{F.2 Numerical Sensitivity of the Non-Negative Tensor Factorization}

In this section three experiments are introduced to show the following facts:

- It is possible to gain roughly 0.6 dB by using an appropriate analysis window for the non-adaptive STFT (see also Section F.2.1). By choosing the analysis window in a correct way for each single mixture, the time-frequency resolution is adapted globally to the mixture.
It is also possible to reach the same gain by correctly choosing between different (possible random) initializations of matrices $G$ and $A$ (see also Section F.2.2). Additionally, the same improvements in separation quality can be reached by simply zero-padding the mixture (see also Section F.2.3). This can be interpreted as moving the analysis windows of the STFT by a small amount of samples.

F.2.1 Experiment 1: Different Analysis Windows

In a first experiment, the time-frequency resolution is adapted to each mixture globally: Each mixture is transformed by the STFT using the four different analysis windows introduced in Section 2.2.1: $\text{rect (} w_s = h_s \text{)}, \text{ sine (} w_s = 2h_s \text{)}, \text{ Blackman (} w_s = 3h_s \text{)},$ and $\text{Hann (} w_s = 4h_s \text{)}$. For better comparability with the experiments regarding the offset and the random initialization introduced later, the four proposed analysis windows are complemented by the four non-integer quotients $\frac{w_s}{h_s} = \left( \frac{19}{16}, \frac{22}{16}, \frac{25}{16}, \frac{28}{16} \right)$. In these cases, analysis- and synthesis windows are of the type Tukey window. This family of window functions can be interpreted as a $\text{rect}$ window combined with the flanks of a Hann window. In our case, we use the flanks of the sine window to enable usage of identical analysis- and synthesis windows. Two sampled Tukey windows as used in our framework are plotted in Figure F.6.

The hop size is fixed to 1024 samples and the transform length of the DFT is fixed to 4096 samples due to zero-padding. Using different window sizes but constant hop size and constant DFT-length is also proposed in [105]. Changing the time-frequency resolution in a global manner has the advantage of avoiding the non-constant overlap-add of the analysis windows shown in Figure F.2.

The separation quality for the eight different analysis windows and their corresponding window sizes is shown in Table F.1. Similar to the results shown in Figure 4.5, window sizes in the range $2048 \leq w_s \leq 4096$ perform best. As mentioned earlier, if we choose for each mixture the analysis window that results in the best separation quality, the SNR

![Figure F.6: Tukey-like windows for interpolation between the rect- and the sine window. The hop size is 1024 samples. Such analysis windows are also shown in Figure F.3.](image-url)
Table F.1: Separation quality for data $A$ and reference clustering (non-blind): The quality is evaluated for different values of the window size $w_s$. For evaluating the separation quality of the last row ($\text{max}$), for each mixture the window size leading to the maximum separation quality is chosen.

<table>
<thead>
<tr>
<th>time-frequency resolution</th>
<th>SNR</th>
<th>SNR$_{seg}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w_s = 1 \cdot h_s$</td>
<td>10.03</td>
<td>9.24</td>
</tr>
<tr>
<td>$w_s = \frac{10}{3} \cdot h_s$</td>
<td>10.68</td>
<td>10.26</td>
</tr>
<tr>
<td>$w_s = \frac{15}{7} \cdot h_s$</td>
<td>10.93</td>
<td>10.63</td>
</tr>
<tr>
<td>$w_s = \frac{20}{9} \cdot h_s$</td>
<td>11.10</td>
<td>10.91</td>
</tr>
<tr>
<td>$w_s = \frac{25}{11} \cdot h_s$</td>
<td>11.25</td>
<td>11.10</td>
</tr>
<tr>
<td>$w_s = 2 \cdot h_s$</td>
<td>11.37</td>
<td>11.33</td>
</tr>
<tr>
<td>$w_s = 3 \cdot h_s$</td>
<td>11.66</td>
<td>11.70</td>
</tr>
<tr>
<td>$w_s = 4 \cdot h_s$</td>
<td>11.80</td>
<td>11.93</td>
</tr>
<tr>
<td>max</td>
<td>12.77</td>
<td>12.55</td>
</tr>
</tbody>
</table>

Improves by roughly 0.6 dB and the SNR$_{seg}$ by roughly 0.9 dB compared to the separation quality of the best analysis window.

F.2.2 Experiment 2: Different Random Initialization

In a second experiment, the influence of different random initializations of $A$ and $G$ for the NTF is tested. The Matrix $B$ is initialized as described in Section 4.3, because of the superior separation quality for this initialization scheme.

Similar to experiment 1, the separation quality over data $A$ remains nearly constant.

<table>
<thead>
<tr>
<th>initialization</th>
<th>SNR</th>
<th>SNR$_{seg}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>deterministic</td>
<td>11.80</td>
<td>11.93</td>
</tr>
<tr>
<td>random 1</td>
<td>11.68</td>
<td>11.82</td>
</tr>
<tr>
<td>random 2</td>
<td>11.75</td>
<td>11.85</td>
</tr>
<tr>
<td>random 3</td>
<td>11.70</td>
<td>11.82</td>
</tr>
<tr>
<td>random 4</td>
<td>11.70</td>
<td>11.83</td>
</tr>
<tr>
<td>random 5</td>
<td>11.71</td>
<td>11.83</td>
</tr>
<tr>
<td>random 6</td>
<td>11.72</td>
<td>11.86</td>
</tr>
<tr>
<td>random 7</td>
<td>11.68</td>
<td>11.82</td>
</tr>
<tr>
<td>max</td>
<td>13.07</td>
<td>12.94</td>
</tr>
</tbody>
</table>

Table F.2: Separation quality for data $A$ and reference clustering (non-blind): The quality is evaluated for different random initializations. For evaluating the separation quality of the last row ($\text{max}$), for each mixture the offset leading to the maximum separation quality is chosen.

Choosing the optimal initialization leads to an increment of more than 1 dB.

This experiment indicates that local minima of the $\beta$-divergence are an open problem in the proposed BSS framework: If the NTF does not converge to a fixed-point it is impossible
to find out, whether possible improvements regarding the separation quality result from adapting the time-frequency transform, or simply from finding a better minima. In this context it is important to remember that a lower value of final $\beta$-divergence is only slightly correlated with higher separation quality.

As mentioned in [23], it is not guaranteed that the NTF finds the global minimum of the cost-function. This can also be verified in Table F.2. The authors of [23] propose the initialization of matrices $A$, $B$, and $G$ in the following way: After random initialization a small number of iterations of the NTF is applied. This is repeated several times, and the matrices $A$, $B$, and $G$ corresponding to the smallest resulting cost-function are kept. After this initialization procedure, these three matrices are used for a final number of initializations.

Therefore, it can be argued that the variances regarding the separation quality in this experiment can be minimized simply by increasing the computational complexity. From this point of view it is only mentioned that the different initialization has such a significant influence on the separation quality that each advantage of an adaptive time-frequency resolution can be erased or turned into decreasing separation quality if the initialization is not properly chosen, or the NTF does not converge to the global minimum (which is the default case).

Another possibility to avoid local minima is proposed in [71]: It is mentioned that the non-convexity of the Itakura-Saito distance $d_{\beta=0}(x, y)$ for $y > 2x$ leads to higher sensitivity regarding local minima. To avoid these local minima, in [71] it is suggested to start with higher values for $\beta$, e.g. $\beta = 2$. After a given number of iterations, the parameter $\beta$ is changed smoothly to $\beta = 0$. Unfortunately, it cannot be guaranteed that for higher values of $\beta$ the number of local minima decreases in our scenario. Additionally, another experiment is performed for investigating the influence of the non-convexity of the $\beta$-divergence in Section F.2.3. As shown there, a symmetric variant of the $\beta$-divergence (defined in Equation (F.6)) leads to similar separation quality as the standard $\beta$-divergence. From this point it can be derived that the non-convexity of the cost function seems not to be a problem for the BSS framework proposed in this thesis.

### Choosing an Optimal Initialization based on Blind Decision

Three features are evaluated in order to find blindly an initialization resulting in a good separation quality:

- $\beta$-divergence of the approximation by the NTF. The lower the feature, the better the separation quality.
- Cross-correlation between logarithmic input $X_{\log}(k, t, c) = \log X(k, t, c)$ and output $\hat{X}_{\log}(k, t, c) = \log \hat{X}(k, t, c)$ of the NTF according to Equation (2.7): $\varphi_{X_{\log}\hat{X}_{\log}}$. The higher the feature, the better the separation quality.
- Estimated segmental SNR: $\frac{1}{Tc} \sum_{tc} \log_{10} \frac{\sum_k X^2(k,t,c)}{\sum_k (X(k,t,c) - \hat{X}(k,t,c))^2}$. The higher the feature, the better the separation quality.

Depending on these three features a simple classifier decides which initialization has to be chosen for final evaluation of separation quality. The separation quality increases by roughly 0.1 dB for such a blind classifier, which is far less than the maximum possible
Figure F.7: Separation quality $\text{SNR}_{\text{seg}}$ for different degrees of freedom: Blind decision between different (random) initializations based on the three features $\beta$-divergence, cross-correlation between input and output of the NTF, and estimated $\text{SNR}_{\text{seg}}$.

gain of roughly 1 dB. This small gain does not legitimize the much higher computational complexity.

F.2.3 Experiment 3: Zero-Padding

<table>
<thead>
<tr>
<th>offset</th>
<th>SNR</th>
<th>$\text{SNR}_{\text{seg}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>11.80</td>
<td>11.93</td>
</tr>
<tr>
<td>128</td>
<td>11.79</td>
<td>11.94</td>
</tr>
<tr>
<td>256</td>
<td>11.79</td>
<td>11.93</td>
</tr>
<tr>
<td>384</td>
<td>11.79</td>
<td>11.92</td>
</tr>
<tr>
<td>512</td>
<td>11.81</td>
<td>11.93</td>
</tr>
<tr>
<td>640</td>
<td>11.80</td>
<td>11.93</td>
</tr>
<tr>
<td>768</td>
<td>11.81</td>
<td>11.93</td>
</tr>
<tr>
<td>896</td>
<td>11.78</td>
<td>11.91</td>
</tr>
<tr>
<td>max</td>
<td>12.74</td>
<td>12.73</td>
</tr>
</tbody>
</table>

Table F.3: Separation quality for data $\mathcal{A}$ and reference clustering (non-blind): The quality is evaluated for different values of the offset (number of zeros used for zero-padding). For evaluating the separation quality of the last row (max), for each mixture the offset leading to the maximum separation quality is chosen.

Finally, the influence of a small shift of all analysis windows is discussed in a third experiment. It is reasonable that this shift results in a very small change in the spectrogram $X(k, t, c)$. It is achieved by zero-padding the mixture signal $x(n, c)$ with a given number $n_0$ of zeros inserted before the first sample. This zero-padding moves the analysis windows of the STFT a little bit. The number $T$ of analysis frames is kept constant by appending $w_a - n_0$ zeros at the end of the signal. It is reasonable that the separation quality remains
nearly unchanged by this small shift. For a large number of mixtures this is true, as can be seen in Table F.3. For single mixture, the influence of this offset can become critical, as can be seen by the last row in Table F.3: Choosing the optimal offset regarding the separation quality leads to a significant higher separation quality.

For a more detailed point of view, the mixture celesta + glockenspiel is analysed: In Figure F.8, the mixture’s spectrogram and the corresponding separation quality is shown. The large difference in SNR\textsubscript{local} between the 6-th and the 8-th second can be seen clearly.

**Figure F.8:** The spectrogram of the mixture celesta + glockenspiel and the corresponding separation quality for different offsets \( n_0 \) is shown. The large difference in SNR\textsubscript{local} between the 6-th and the 8-th second can be seen clearly.

The mean SNR\textsubscript{seg} increases from 8.0 dB up to 12.1 dB by changing the offset from \( n_0 = 512 \) to \( n_0 = 640 \). Due to Figure F.8(b), this rise in separation quality can be assigned to the temporal region between the 6-th and the 8-th second. Taking a closer look on the spectrogram, this temporal segment is a steady-state region with nearly constant spectrum.

Motivated by the nearly constant spectrum of the region of interest, the mean spectrum is plotted for this temporal segment in Figure F.9. In Figure F.9(a), it can be seen clearly that the approximation by the NTF leads to a large amount of signal energy around the frequency of 8172 Hertz although the mixture has no signal energy at this frequency. Changing the offset to \( n_0 = 640 \) samples results in the correct estimated signal energy for this frequency. The tendency of the Itakura-Saito distance (\( \beta \)-divergence with \( \beta = 0 \)) to estimate the signal energy higher than necessary is also visualized in Figure 2.1: For \( \beta < 2 \), the \( \beta \)-divergence is asymmetric with higher values for the case \( y < x \) than for the case \( y > x \). Therefore, the energy of the estimated signal \( y \) tends to higher values compared to the original signal’s energy.

The influence of zero-padding with different values for \( n_0 \) can be observed also for data \( \mathcal{B} \). In Figure F.10, the influence of the zero-padding by \( n_0 \) zeros is shown for two mixtures. It can be seen that the differences between best separation quality and worst separation quality is roughly 0.5 dB which cannot be ignored.
Interaction of Time-Frequency Transform and Non-Negative Tensor Factorization

Figure F.9: Comparison of the mean spectrum of the mixture and the approximation by the NTF with an offset $n_0 = 512$ (left) and an offset $n_0 = 640$ (right).

Figure F.10: Separation quality for different offsets $n_0$ for two mixtures out of data $B$. The chaotic influence of the offset can be seen clearly. Factorization is done with $I = 25$.

Choosing an Optimal Offset based on Blind Decision

In a first approach, the differences in separation quality for different offsets are utilized to improve the overall separation quality. The same classifiers as mentioned in the experiments regarding the initialization are used for automatic detection of a good offset. The results can be seen in Figure F.11. Even with these simple classifiers it is possible to improve the separation quality by roughly 0.1 dB. Unfortunately, this improvement is much smaller than the maximum possible gain of roughly 1 dB shown in Table F.3 but needs eight times higher computational complexity. Again, this small gain in separation quality is not worth the much higher computational complexity.
Reducing the Influence of the Offset

In the following, three approaches are shown for reducing the influence of the offset: Symmetric Itakura-Saito distance, increasing the number of components $I$, and influence of the convergence of the $\beta$-divergence.

**Symmetric/Convex Itakura-Saito-Distance** As mentioned in Section 2.3.2, the Itakura-Saito-Distance $d_\beta(x, y)$ is not convex for $y \geq 2x$. Firstly, the influence of this non-convexity is shown by the following experiment:

As mentioned in Section 2.1.1, the $\beta$-divergence is not symmetric for $\beta \neq 2$. A simple solution to make arbitrary distance measures symmetric is to use the arithmetic mean of

$$d_{\text{symmetric}, \beta}(x, y) = \frac{1}{2} (d_\beta(x, y) + d_\beta(y, x)) \quad .$$

(F.6)

This method is used for the Kullback-Leibler divergence ($d_{\beta=1}$) e.g. in [73] or [78]. For $\beta = 0$ the symmetric variant is defined by

$$d_{\text{symmetric}, \beta=0}(x, y) = \frac{1}{2} \left( \frac{x}{y} + \frac{y}{x} \right) - 1 \quad .$$

(F.7)

The first two derivatives regarding the variable $y$ are

$$\frac{\partial d_{\text{symmetric}, \beta=0}}{\partial y} = \frac{1}{x} - \frac{x}{y^2} \quad ,$$

(F.8)

$$\frac{\partial^2 d_{\text{symmetric}, \beta=0}}{\partial y^2} = \frac{2x}{y^3} \quad .$$

(F.9)

As can be seen in Equation (F.9), $d_{\text{symmetric}, \beta=0}$ is convex for all $y$ in the case of non-negative $x$ and $y$. A non-negative tensor factorization can be derived by evaluation of
Interaction of Time-Frequency Transform and Non-Negative Tensor Factorization

the gradients and defining multiplicative update rules in the same way as in Equations (2.45)–(2.47). For the symmetric variant, \( \xi_1(k, t, c) \) and \( \xi_2(k, t, c) \) are defined as follows:

\[
\xi_1(k, t, c) = \frac{X(k, t, c)}{X^2(k, t, c)}, \quad (F.10)
\]

\[
\xi_2(k, t, c) = \frac{1}{X(k, t, c)}. \quad (F.11)
\]

Beside the convexity, one additional advantage of the symmetric variant is a slightly lower computational complexity: During the iterative minimization by the multiplicative updates \( \xi_2(k, t, c) \) remains constant and is evaluated only once at the beginning of the algorithm.

The separation quality of the symmetric \( \beta \)-divergence can be seen in Table F.4. Interestingly, the symmetric \( \beta \)-divergence leads to the same results as the asymmetric \( \beta \)-divergence. Unfortunately, the variance regarding the separation quality increases as can be seen for the increased maximum value in the last row of Table F.4. For this reason, the symmetric \( \beta \)-divergence is not used in the following. The slightly lower computational complexity due to constant \( \xi_2 \) and the approximately equal separation quality can be of interest for applications with limited evaluation time.

<table>
<thead>
<tr>
<th>Offset</th>
<th>( d_{\beta=0} ) SNR</th>
<th>( d_{\text{symmetric, } \beta=0} ) SNR</th>
<th>( d_{\text{symmetric, } \beta=0} ) SNRseg</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>11.80</td>
<td>11.93</td>
<td>11.73</td>
</tr>
<tr>
<td>128</td>
<td>11.79</td>
<td>11.94</td>
<td>11.73</td>
</tr>
<tr>
<td>256</td>
<td>11.79</td>
<td>11.93</td>
<td>11.74</td>
</tr>
<tr>
<td>384</td>
<td>11.79</td>
<td>11.92</td>
<td>11.75</td>
</tr>
<tr>
<td>512</td>
<td>11.81</td>
<td>11.93</td>
<td>11.75</td>
</tr>
<tr>
<td>640</td>
<td>11.80</td>
<td>11.93</td>
<td>11.73</td>
</tr>
<tr>
<td>768</td>
<td>11.81</td>
<td>11.93</td>
<td>11.72</td>
</tr>
<tr>
<td>896</td>
<td>11.78</td>
<td>11.91</td>
<td>11.70</td>
</tr>
<tr>
<td>Max</td>
<td>12.74</td>
<td>12.73</td>
<td>12.90</td>
</tr>
</tbody>
</table>

Table F.4: Separation quality for data \( \mathcal{A} \) and reference clustering (non-blind): The quality is evaluated for different values of the offset (number of zeros used for zero-padding). For evaluating the separation quality of the last row (\( \text{max} \)), for each mixture the offset leading to the maximum separation quality is chosen. \( d_{\beta=0} \) corresponds to the Itakura-Saito distance as defined in Equation (2.5) for \( \beta = 0 \). \( d_{\text{symmetric, } \beta=0} \) corresponds to the symmetric variant explained in this section.


elements original signal and approximated signal. The summation over all $K \times T \times C$ elements of the tensors $X$ and $\tilde{X}$ leads to a distance function which prefers a small number of elements with large differences between the original and the approximated tensor.

Figure F.12: SNR$_{seg}$ for different number of components $I$ used for the factorization by the NTF for the mixture Tamy - Que pena tanto faz. The larger $I$ the smaller is the difference in separation quality between the highest possible separation quality and the smallest one. Reference clustering is used to avoid misleading results due to clustering errors.

**Increasing the Number of Components $I$** In Figure F.12, the separation quality for different number of components $I$ to factorize is shown for one mixture of data $\mathcal{B}$. Minimum is the minimum separation quality over all 8 different offsets, maximum is the corresponding maximum, and $n_0 = 0$ is shown as the default case without using offsets. It can be seen clearly that for smaller values of $I$, the influence of the offset becomes more critical.

This result can be verified also for data $\mathcal{A}$. In Figure F.13, it can be seen that the difference between the maximum and the minimum separation quality decreases with increasing number of components $I$. On one hand, larger $I$ results in smaller influence of the offset $n_0$. On the other hand, the complexity of the blind clustering algorithm becomes higher for larger $I$. This has two reasons:

- The clustering algorithm groups $I$ feature samples into $M$ clusters. This results in a total of $M^I$ clustering possibilities$^1$. The higher the degree of freedom, the harder is the correct clustering.

- Obviously, for large $I$ the NTF is not longer forced to factorize a single note into a single component $i$, because there are too much parameters for the NTF to approximate the tensor. This effect is also shown e.g. in [3] by factorizing a piece with the NTF based on the $\beta$-divergence with $\beta \in \{0, 1, 2\}$: For $\beta \geq 1$, the transient regions of all notes are factorized into a separate component and not factorized together.

$^1$The real degree of freedom is smaller, because after clustering and signal synthesis the correct indices of each estimated source is determined not blind to solve the alignment problem.
with the corresponding notes.

By this, an additional problem arises for the clustering algorithms: Not only the single notes have to be clustered but also the single segments (transient and steady state) of a note or the single harmonics of a note have to be clustered together.

Considering both disadvantages, $I = 20$ seems to be a good compromise between reducing the influence of the offset $n_0$ and keeping the clustering as simple as possible.

**Influence of Convergence of the $\beta$-divergence** The results of experiment 2 suggests that the cost-function of the NTF has many local minima. These local minima are the reason for the variance regarding the separation quality in experiment 2 (different random initializations). Additionally, in Figure 4.8 it can be seen that with 300 iterations the NTF usually did not converge to a fixed point. Otherwise, after 300 iterations the separation quality shall not change any more.

Motivated by these observations, the variances in separation quality caused by different offsets is evaluated for different number of iterations done by the NTF. The results are shown in Figure F.14. Interestingly, the higher number of iterations the higher the difference between maximum and minimum of possible separation qualities. Two things can be concluded: Firstly, the semantic based initialization works pretty well for audio data because only 100 iterations are sufficient for appealing separation quality. Secondly, it is not necessary to reach a fixed point of the $\beta$-divergence for good separation quality. Contrary to Figure 4.8, the separation quality does not decrease for smaller number of iterations. The only differences between the experiment responsible for Figure 4.8 and the experiment applied here are $I$ is set to 25 and $\beta = 0$. 

Reducing the number of iterations induces the question, if it is possible to find an initialization algorithm for our BSS framework resulting in even higher separation quality but such experiments are beyond the scope of this work.

F.2.4 Comparison of the Experiments

In Figure F.15, the three above mentioned experiments are summarized. The parameter degree of freedom corresponds to the number of trials to separate a mixture, either with different zero-paddings, different random initializations of the NTF, or with different analysis windows for STFT. The decision, which trial’s output to use for evaluation of the separation quality is done in a non-blind manner: E.g. for four degrees of freedom for the parameter offset, each mixture is separated four times with offsets $n_0 = 0, 256, 512, 768$.
samples. The mixture separated with highest separation quality is chosen for evaluation. Adapting the time-frequency resolution in a global way is not superior to the variations with different onsets or different random initializations regarding the separation quality. The non-blind decision used for evaluation of the data shown in Figure F.15 is not appli-

![Figure F.16: Separation quality SNR\textsubscript{seg} for different degrees of freedom: Blind decision.](image)
cable in a BSS scenario. Therefore, a set of features is evaluated for each mixture. These features are used for classifying which trial has to be used as final separation result. The three most important features are the above mentioned $\beta$-divergence, the cross-correlation between the logarithmic input and output of the NTF, and the estimated $\text{SNR}_{\text{seg}}$. All three features are described in Section F.2.2. The other features used for blind decision are typical features used for transient detection, e.g. phase deviation, high-frequency-content, and different forms of energy concentration. Unfortunately, only the first three features lead to satisfying results. Therefore, the other features are not described in detail.

The order of the different approaches is similar to the non-blind decision: Varying the zero-padding of the mixture and using different random initializations for NTF leads to better separation quality than adapting the time-frequency resolution of the spectrogram. The fact that typical transient-detectors like phase-deviation, high-frequency-content and compactness of energy are also used for selection of an appropriate separation trial indicates that adaption of time-frequency resolution cannot be solved in a sufficient way only by transient detection.

### F.3 Summary

In this chapter, different algorithms for adapting the time-frequency resolution to the needs of the BSS framework are introduced. The disadvantages of different adaption schemes are discussed. After that, three experiments are introduced: Different time-frequency resolution, different random initialization, and different zero-padding of the mixture. These three experiments show the sensitivity of the NTF regarding even smallest changes in the input data. It is shown that these three experiments can influence the
separation quality by roughly 1 dB each. It is pointed out that experiments about adaptive time-frequency resolution are obsolete as long as the NTF is such sensitive to even smallest changes of the input signals: Someone can never be sure that possible increases in separation quality are based on an arbitrary adaptive time-frequency transform or on such side-effects like different offsets due to different starting positions of the analysis windows.

Blind decision rules are introduced to improve the separation quality by roughly 0.1 dB at eight times higher computational complexity. These blind decision rules are only applicable to the experiments regarding the different initializations or the different zero-paddings. For the different time-frequency resolutions no unsupervised decision rule is found. This fact induces the high level of difficulty for finding a good adaptive time-frequency transform for BSS, too. Therefore, a constant time frequency resolution and a standard NTF with the parameters discussed in Chapter 4 is used in the following.
Appendix G

Matlab Code

In the following, the Matlab code for the CFCM-A in the proposed BSS-framework is given. This code comes without any warranty. In this thesis, it is shown in several sections that the proposed framework generally leads to good separation results. For a single mixture, it is impossible to predict the separation quality. This code is to be intended to give a good starting position for beginners in the area of BSS. Or it can be used as an initialization algorithm for other BSS-frameworks, as suggested in Section 8.2. The used audio samples are from the website [45].

```matlab
function UnderdeterminedBlindSourceSeparation()
    [s1,Fs,bits]=wavread('src_1.wav');
    s2,Fs,bits=wavread('src_2.wav');
    s3,Fs,bits=wavread('src_4.wav');
    s4,Fs,bits=wavread('src_5.wav');
    s_in = cat(3,s1,s2,s3,s4);
    x = sum(s_in,3);
    M = size(s_in,3);
    y = BlindSourceSeparation(x,Fs,M);
    [perm,SNR] = solve_alignment(s_in,y)
    y = y(:,perm);
    NumSamples = round(20/1000*Fs);
    SNRseg = zeros(1,M);
    for m=1:M
        for c=1:size(y,2)
            idx1=1;
            idx2=NumSamples;
            while (idx1<=size(s_in,1))
                SNRlocal=10*log10(sum(s_in(idx1:idx2,1,m).^2)/... //
                sum((s_in(idx1:idx2,1,m)-y(idx1:idx2,1,m)).^2));
                SNRlocal=max(SNRlocal,0);
                SNRlocal=min(SNRlocal,.35);
                SNRseg(m)=SNRseg(m)+SNRlocal/size(y,2)*(idx2-idx1+1)/size(s_in,1);
                idx1=idx1+1;
                idx2=min(idx1+NumSamples-1,size(s_in,1));
            end
        end
        wavwrite(y(:,perm,m),Fs,bits,sprintf('src_est%d.wav',m));
    end
    SNRseg
```

---

```matlab
function [y]=BlindSourceSeparation(mix,Fs,M)
% STFT + NTF parameters
ws = 2^12;
hs = 2^10;
w = hann(ws,'periodic')*sqrt(2/3);
NyquistIndex = round(ws/2)+1;
MaxIter = 300;
I = 25;
NMel = 400;
beta = 0;
```

---
MixingModel = 2;

% evaluating time frequency transform
fprintf('perform STFT...');
tmp = STFT(mix(:,1), ws, hs, w);
MIX = zeros(size(tmp,1), size(tmp,2), size(mix,2));
MIX(:,1,:) = tmp;
for channel=2:size(mix,2)
   MIX(:,channel,:) = STFT(mix(:,channel), ws, hs, w);
end
MIX = MIX(1:NyquistIndex,:,:);
MIX_tmp = abs(MIX);
disp('done!');

fprintf('apply Noise Gate: ');

dBThreshold = -60;
tmp = 10*log10(MIX^2);
tmp = tmp - max(tmp(:));
tmp = max(tmp(:,[1,3]), 1);
NoiseGateMask = tmp > dBThreshold;
MIX_tmp = MIX_tmp(:,NoiseGateMask,:);
fprintf('%d column(s) removed
', sum(NoiseGateMask==0));

disp('done!');

fprintf('perform dimension reduction...');
if (NMel<NyquistIndex)
   MelFiltMat=CreateFilterBank(size(MIX_tmp,1)*2-2,NMel,Fs);
tmp=MelFiltMat' * MelFiltMat;
tmp=sum(tmp,2); % sum over frequencies
W=sparse(1:length(tmp),1:length(tmp),1./sqrt(tmp));
MelFiltMat=W*MelFiltMat';
tmp=zeros(NMel, size(MIX_tmp,2), size(MIX_tmp,3));
for channel=1:size(MIX_tmp,3)
   tmp(:,channel) = MelFiltMat' * MIX_tmp(:,channel);
end
MIX_tmp=tmp;
else
   MelFiltMat=sparse(1:size(MIX_tmp,1),1:size(MIX_tmp,1),1);
end
disp('done!');

fprintf('perform NTF...');
% semantic initialization
B = ones(size(MIX_tmp,1), 88);
G = ones(size(MIX_tmp,2), 88);
A = ones(size(MIX_tmp,3), 88);
f_0 = 27.5; % starting pitch
damping = -3; % 3 dB/Octave
t = (0/ws-1)/Fs;
for i=1:size(B,2)
x=zeros(ws,1);
   for n=1:20
      f=n*f_0;
      if (f<Fs/2)
         x=x+10^-((damping*log2(1))/20)*cos(2*pi*f*t);
      end
   end
   x = fft(x.*w); % dft + analysis window
   B(:,i) = MelFiltMat' * abs(x(1:size(MelFiltMat,1)));
   B(:,i) = B(:,i)/sqrt(sum(B(:,i).^2)); % normalization
   f_0 = f_0 * (2^(1/12)); % next note
end
tmp1=sum(MIX_tmp,3);
for i=1:size(G,2)
   for t=1:size(G,1)
      G(t,i)=sum(B(:,i).*tmp1(:,t));
   end
end
[B,G,A]=NTF(MIX_tmp.^MixingModel, I, B, G, A, MaxIter, beta);
disp('done!');
fprintf('perform inverse dimension reduction...');
if(size(G,1)<length(NoiseGateMask))
tmp = zeros(length(NoiseGateMask),size(G,2));
tmp(NoiseGateMask,:) = G;
G = tmp;
end
B = MelFiltMat*B;
Xi = eps*ones(size(MIX,1),size(G,1),size(A,1),size(B,2));
for i=1:size(B,2)
tmp=B(:,i)*G(:,i)';
for c=1:size(A,1)
Xi(:,c,i)=A(c,i)*tmp;
end
end
disp('done!');
fprintf('note reconstruction...');
Xi = Xi+eps;
weighting = MIX./sum(Xi,4);
sources_all = zeros(size(MIX,1),size(MIX,2),size(B,2));
for i=1:size(B,2)
Xest = Xi(:, :, i) .* weighting;
for channel=1:size(Xest,3)
tmp = Xest(:, :, channel);
tmp = ISTFT([tmp conj(tmp(1:end-1))],ws,hs,w);
sources_all(:,channel,i)=tmp(1:size(sources_all,1),1);
end
end
disp('done!');

function [X]=STFT(x,ws,hs,w)
x = [randn(ws,1)*eps;x;randn(ws,1)*eps];
NumOfCols = floor((length(x)-ws)/hs)+1;
X = zeros(ws,NumOfCols);
for col=1:NumOfCols
X(:,col) = fft(x((col-1)*hs+1:(col-1)*hs+w),w);
end

function [x]=ISTFT(X,ws,hs,w)
x=zeros((size(X,2)-1)*hs+ws,1);
X=ifft(X,'symmetric');
for col=1:size(X,2)
x((col-1)*hs+1:(col-1)*hs+ws,1) = x((col-1)*hs+1:(col-1)*hs+ws,1) + X(:,col).*w;
end
x = x(ws+1:end,1);

function [MelFiltMat]=CreateFilterBank(fftlen,N,Fs,f_lambda)
if nargin<4 f_lambda=0; end
M = round(fftlen/2)+1;
melmax = f2m(round(Fs/2),f_lambda);
s = m2f((0:N-1)*melmax/(N-1),f_lambda);
k = fftlen*s/Fs+1;
d = diff(k);
c = floor(k);
F = zeros(N,M);
\begin{verbatim}
F(1,:) = (k(2)-(1:M))/d(1);
for iter = 2:N-1
    F(iter,:) = ((1:c(iter))-k(iter-1))/d(iter-1), ...
                (k(iter+1)-(c(iter)+1:M))/d(iter);
end
F(N,:) = ((1:M)-k(N-1))/d(N-1);
MelFiltMat = max(F,0);
% avoids frequency spreading for low frequencies
for col = 1:size(MelFiltMat,1)
    MelFiltMat(col,col) = MelFiltMat(col,col) + sum(MelFiltMat(col+1:end,col));
    MelFiltMat(col+1:end,col) = 0;
end

function y=f2m(x,lambda) % frequency to mel transform
y=BoxCoxTrafo(x./700+1,lambda);

function y=m2f(x,lambda) % mel to frequency transform
[tmp, y]=BoxCoxTrafo(x,lambda);
y=700*(y-1);

function [z,x]=BoxCoxTrafo(y,lambda)
if (lambda==0)
z=log(y); x=exp(y);
else
    z=(y.^lambda-1)/lambda; x=(lambda*y+1).^(1/lambda);
end

function [B,G,A]=NTF(X, I, B,G,A, MaxIter, beta)
for iter = 1:MaxIter
    A = update_A(X, A, B,G, beta);
    G = update_G(X, A, B,G, beta);
    B = update_B(X, A, B,G, beta);
    % normalization for numerical stability
    E1=sqrt(sum(A.^2,1));
    E2=sqrt(sum(B.^2,1));
    E3=sqrt(sum(G.^2,1));
    E = (E1.*E2.*E3).^(1/3);
    for i = 1:size(B,2)
        A(:,i) = max(A(:,i)/E1(i)*E(i),eps);
        B(:,i) = max(B(:,i)/E2(i)*E(i),eps);
        G(:,i) = max(G(:,i)/E3(i)*E(i),eps);
    end
    % adjust size of B,G,A
    if (size(B,2)>1)
        E = sum(B.^2).*sum(A.^2).*sum(G.^2);
        [val, idx] = min(E);
        B(:,idx) = [];
        G(:,idx) = [];
        A(:,idx) = [];
    end
end

function [A]=update_A(X, A, B,G, beta)
if (size(X,3)>1)
    Xest = eval_model(A, B,G);
    [X, Xest] = adapt_xi1_xi2(X, Xest, beta);
    enumerator = zeros(size(A));
    denominator = enumerator;
    tmp1 = B';
    for c = 1:size(A,1)
        tmp2 = X(:,c)*G;
        tmp3 = Xest(:,c)*G;
        for i = 1:size(A,2)
            numerator(c,i) = tmp1(i,:)*tmp2(:,i);
        end
    end
end
\end{verbatim}
Matlab Code

denominator (c, i) = tmp1(i, :) * tmp3(:, i);
end
end
A = A.*(enumerator + eps) ./ (denominator + eps);
end

function [B] = update_B(X, A, B, G, beta)
Xest = eval_model(A, B, G);
[X, Xest] = adapt_xi1_xi2(X, Xest, beta);
if (size(X, 3) > 1)
  % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
  % % % NTF: multi channel case
  enumerator = zeros(size(B));
denominator = enumerator;
for c = 1:size(X, 3)
  tmp1 = X(:, c) * G;
  tmp2 = Xest(:, c) * G;
  for i = 1:size(B, 2)
    enumerator(:, i) = enumerator(:, i) + A(c, i) * tmp1(:, i);
    denominator(:, i) = denominator(:, i) + A(c, i) * tmp2(:, i);
  end
end
else
  % % % NMF: single channel case
  enumerator = X * G;
denominator = Xest * G;
end
B = B.*(enumerator + eps) ./ (denominator + eps);

Xest = eval_model(A, B, G);
[X, Xest] = adapt_xi1_xi2(X, Xest, beta);
if (size(X, 3) > 1)
  % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % % %
  % % % NTF: multi channel case
  enumerator = zeros(size(G));
denominator = enumerator;
for c = 1:size(X, 3)
  tmp1 = X(:, c)';
  tmp2 = Xest(:, c)';
  for i = 1:size(G, 2)
    enumerator(:, i) = enumerator(:, i) + A(c, i) * (tmp1 * B(:, i));
    denominator(:, i) = denominator(:, i) + A(c, i) * (tmp2 * B(:, i));
  end
end
else
  % % % NMF: single channel case
  enumerator = X' * B;
denominator = Xest'*B;
end
G = G.*(enumerator + eps) ./ (denominator + eps);

function [X, Xest] = adapt_xi1_xi2(X, Xest, beta)
if (beta == 1)
  X = (X + eps) ./ (Xest + eps);
  Xest = ones(size(Xest));
elseif (beta == 0)
  Xest = (1 + eps) ./ (Xest + eps);
  X = X .* (Xest.^2);
elseif (beta == 2)
  % do nothing
else
  tmp = exp((beta - 1)*log(Xest + eps));
  X = (X .* tmp + eps) ./ (Xest + eps);
  Xest = tmp;
end
function [X]=eval_model(A,B,G)
if(size(A,1)>1)
    %% multi channel case
    X=zeros(size(B,1),size(G,1),size(A,1));
    for channel=1:size(X,3)
        tmp = B.*repmat(A(channel,:),size(B,1),1);
        X(:, :, channel) = tmp*G';
    end
else
    %% single channel case
    X=B*G';
end

function [genom]=FeatureBasedClustering(sourcesall,B,M,FS)
    % standard features
    SFMB = 60;
    NMel = 20;
    a_lambda = -0.5;
    f_lambda = -1.5;
    F1 = CepstralAnalysis(B,f_lambda,a_lambda,f_scale,NMel,FS);
    SFMG hs = round((100/1000*FS);
    ws = hs;
    Gnew = zeros(round((size(sourcesall,1)-ws)/hs+1),size(sourcesall,3));
    for i=1:size(sourcesall,3)
        for row=1:size(Gnew,1)
            idx1 = (row-1)*hs+1;
            idx2 = min((row-1)*hs+ws,size(sourcesall,1));
            tmp = sourcesall(idx1:idx2,:); i);
            Gnew(row,i) = sqrt(sum(tmp.^2)/numel(tmp));
        end
    end
    f_scale = 10;
    fftlen = max(2^12,size(Gnew,1));
    NI = round(fftlen/2+1);
    X = fft(Gnew,fftlen);
    X = abs(X(1:NI,:));
    NMel = NI;
    a_lambda = -1;
    f_lambda = 1.5;
    p = 0.3;
    F2 = CepstralAnalysis(X,f_lambda,a_lambda,f_scale,NMel,FS);
    for col=1:size(F2,2)
        F2(:,col)=F2(:,col)/(sum(F2(:,col).^2).^p);
    end
    %% eval tempo
    TonInstanzen=zeros(size(Gnew,2));
    for i=1:size(Gnew,2)
        Gi=[0;Gnew(:,i)>mean(Gnew(:,i))); 0];
        TonInstanzen(i)=sum(diff(Gi)>0);
    end
    TonInstanzen = TonInstanzen*FS/(hs*size(Gnew,1));
    crit = mean(TonInstanzen);
    %% adjust weighting
    T1 = 0.5; T2 = 1.3; b = 0.5; c1 = 20; c2 = 5;
    a1 = (1-b)*atan((T1-crit)*c1/pi+0.5)+b*(atan((T2-crit)*c2)/pi+0.5);
    a2 = 1-a1;
    %% convex fuzzy c-means
    F=[F1;F2];
    [C,U,genom]=ConvexFuzzyCMeans(F,M,[size(F1,1),size(F2,1)],[a1,a2],[1,1]);
end

function [F]=CepstralAnalysis(X,f_lambda,a_lambda,f_scale,NMel,FS)
R = CreateFilterBank(2*size(X,1)-2,NMel,FS,f_lambda);
F = fftshift(R.*X); % frequency warping
F = F/max(F(:))*(10^((f_scale/20)-1)); % normalization
F = BoxCoxTrafo(F+1,a_lambda); % amplitude scaling
function [C, U, genom] =Convex_fuzzy_cmeans(F,M,featdim,weights,distmode)

f = 2.0; % Fuzzifier
MAXITER = 100;
MINIMPROVEMENT = 1e-6;
% Initialize centroids and partition matrix
U = 1e-6*ones(M,size(F,2));
genom=mod(1:size(F,2),M)+1;
for n = 1:size(F,2)
    U(mod(genom(1,n),M)+1,n)=1-1e-6;
end
% FCM loop
for iter=1:MAXITER
    % Step 1: Update centroids
    C=zeros(size(F,1),M);
    for featurespace=1:length(featdim)
        idx1=sum(featdim(1:featurespace-1))+1;
        idx2=sum(featdim(1:featurespace));
        for m=1:M
            tmp1=0;
            for n=1:size(F,2)
                tmp2=U(m,n)ˆf;
                tmp1=tmp1+tmp2;
            end
            C(idx1:idx2,m)=C(idx1:idx2,m)/tmp1;
        end
    end
    % Step 2: Update partition matrix
    Uold = U;
    D = zeros(size(U));
    for m = 1:M
        for n = 1:size(F,2)
            d=0;
            for featurespace=1:length(featdim)
                idx1 = sum(featdim(1:featurespace-1))+1;
                idx2 = sum(featdim(1:featurespace));
                d = d+weights(featurespace)∗sdist(F(idx1:idx2,n),C(idx1:idx2,m),distmode(featurespace));
            end
            D(m,n) = dˆ(-2/(f-1));
        end
    end
    for n=1:size(F,2)
        U(:,n)=D(:,n)/sum(D(:,n));
    end
    % Evaluate improvement
    dU = abs(U - Uold);
    obj_fcn = max(dU);
    if (obj_fcn < MINIMPROVEMENT)
        break
    end
    [val,genom]=max(U);
    % This case should never occur (empty cluster encountered)
    if(length(unique(genom))<M)
        disp('Less clusters than necessary!');
        for m=1:size(U,1)
            if(sum(genom==m)<eps)
                [val,idx]=max(U(m,:));
                genom(idx)=m;
            end
        end
    end
end

function [d]=sdist(x,y,mode)
if (nargin < 4) p=2; end % Euclidean distance
if (nargin < 3) mode=0; end
if (mode==1)
    % spherical distance
tmp1=sqrt(sum(x.^2));
tmp2=sqrt(sum(y.^2));
if (max(tmp1,tmp2)<10^-10)
    d=0; % both zero: minimum distance
elseif (min(tmp1,tmp2)<10^-10)
    d=2; % one of them zero: maximum distance
else
    x=x/tmp1;
y=y/tmp2;
d=1-sum(x.*y);
end
else % Minkowski metric
    d=sum(abs(x-y).^p)^(1/p);
end

function [permi,SNR_opt]=solve_alignment(sim,sim_est)
fprintf(‘solve alignment...’);
M = size(sim,3);
Mest = size(sim_est,3);
P = perms(1:Mest);
E1 = zeros(1,M);
E2 = zeros(M);
for m1=1:M
    E1(m1) = sum(sum(sim(:, :, m1).^2));
    for m2=1:Mest
        E2(m1,m2) = sum(sum((sim(:, :, m1) - sim_est(:, :, m2)).^2));
    end
end
SNR_opt = -realmax;
SNR = zeros(1,M);
for row=1:size(P,1)
    for m=1:M
        SNR(m)=10*log10(E1(m)/E2(m,P(row,m)));
    end
    if (sum(SNR)>sum(SNR_opt))
        SNR_opt = SNR;
        permi=P(row,:);
    end
end
disp(’done!’);
Bibliography


Lebenslauf

Dipl. Ing. Martin Spiertz

1986 – 1990 Katholische Grundschule Eschweiler-Röhe
1990 – 1999 Heilig-Geist-Gymnasium Broichweiden
1999 – 2000 Wehrdienst
2000 – 2006 Studium der Elektrotechnik an der RWTH Aachen
Fachrichtung Nachrichtentechnik
Juni 2006 Diplom
2006 – 2011 Wissenschaftlicher Mitarbeiter am
Institut für elektrische Nachrichtentechnik, RWTH Aachen
seit August 2011 Angestellter bei der HEAD Acoustics GmbH