Note Clustering based on 2D Source-Filter Modeling for Underdetermined Blind Source Separation

Martin Spiertz[1], Volker Gnann[1]

[1]*Institut fuer Nachrichtentechnik, RWTH Aachen University, D-52056 Aachen, Germany*

Correspondence should be addressed to Martin Spiertz (`spiertz@ient.rwth-aachen.de`)

ABSTRACT

For blind source separation, the non-negative matrix factorization extracts single notes out of a mixture. These notes can be clustered to form the melodies played by a single instrument. A current approach for clustering utilizes a source filter model to describe the envelope over the first dimension of the spectrogram: the frequency-axis. The novelty of this paper is to extend this approach by a second source-filter model, characterizing the second dimension of a spectrogram: the time-axis. The latter one models the temporal evolution of the energy of one note: an instrument specific envelope is convolved with an activation vector, corresponding to tempo, rhythm, and amplitudes of single note instances. We introduce an unsupervised clustering framework for both models and a simple, yet effective combination strategy. Finally, we show the advantages of our separation algorithm compared with two other state-of-the-art separation frameworks: the separation quality is comparable, but our algorithm needs much less computational load, is independent from other BSS-algorithm as initialization, and works with a unique set of parameters for a wide range of audio data.

1. INTRODUCTION

Blind source separation (BSS) can be used as preprocessing step for many audio processing tasks, e.g. up-/remixing, or instrument recognition/transcription. The separation of single acoustical events can be done by factorization of the mixture spectrogram with the non-negative tensor factorization (NTF), see also [1]. The basic variant of the NTF factorizes only single notes out of the mixture not complete melodies. To separate the melodies played by an instrument, the factorized notes have to be clustered, each cluster corresponding to one instrument. To the best of our knowledge, there exists currently two approaches for clustering. In [2], the features used for clustering are based on the source-filter model (SFM). SFM assumes a harmonic signal (*source*) produced by the vibrating parts of the instrument. This sound is filtered by an instrument specific *filter* to form the final spectrum of the note. Mel-frequency cepstrum coefficients (MFCC) utilizes the SFM, as we will show later.

In [3], the features are based on the harmonic representation (HR). HR assumes, that for one instrument the relative amplitudes of the harmonic partials are constant. Therefore, different pitches can be expressed by shifting a constant spectrum along the logarithmic frequency axis. In [4], it is shown, that SFM outperforms HR regarding the task of instrument classification. One open problem of HR is the restriction to pure harmonic models. Additionally, the necessity of a logarithmic frequency resolution complicates the signal synthesis after separation, as mentioned in [1]. Therefore, the authors of [3] measure separation quality based on the spectrograms evaluated by a Gabor wavelet transform making a comparison with them complicated.

Contrary to the clustering approach, [5] and [1] proposes two state of the art extensions to the basic NTF to factorize whole melodies. In [5], point sources are modeled by a convolutive mixing model for multichannel scenarios. The factorized notes are clustered by the estimated convolutive filters. In [1], the NTF is extended over both dimensions of the spectrogram: the frequency-axis is modelled by an SFM and the time-axis is characterized by an activation vector, encoding the onsets, convolved with the characteristic envelope of a single note instance.

In this paper we will describe this convolution by a SFM over the time-axis of the spectrogram. We introduce a homogeneous clustering framework utilizing the both independent dimensions of the SFM and a simple yet effective rule for combining these both models. The separation quality of our BSS framework is compared with [5] and [6] as state-of-the-art separation algorithms. Our algorithm leads to comparable separation quality with several advantages at our side: lower computational complexity, fixed set of parameters for a large range of audio data, and independency from initial source estimation by other BSS-algorithms.
The paper is structured as follows: In Section 2, we give an outline of the basic BSS algorithm. In Section 3, we describe the two source-filter models and the corresponding clustering algorithms. In Section 4, the performance of the clustering algorithms is evaluated. The paper closes with the conclusions in Section 5.

2. SIGNAL FLOW

In this Section, the signal flow of the proposed algorithm is introduced. In the following, we assume an instantaneous mixture model with M sources $s_m(n,c)$ added up to a mixture signal $x(n,c)$. n is the time index, $1 \leq c \leq C$ is the channel index.

2.1. Time-Frequency Representation

Slice c of the three-dimensional tensor $\underline{\mathbf{X}}$ is the spectrogram of $x(n,c)$, evaluated by the short-time Fourier transform (STFT), see also [1]. $\underline{\mathbf{X}}$ is of size $K \times T \times C$, with frequency-bins $1 \leq k \leq K$ and time-bins $1 \leq t \leq T$. In the following, underlined variable $\underline{\mathbf{X}}$ denotes the complex values, $\mathbf{X}$ corresponds to the absolute values. For faster computation, the tensor $\mathbf{X}$ is filtered and downsampled by a mel-filterbank with K_{mel} filters to form a tensor $\mathbf{X}_{\text{mel}}$ with logarithmic frequency-resolution. This filtering can be expressed by a matrix multiplication of each slice of $\mathbf{X}$ with a matrix $\mathbf{R}_{\mathbf{X}}$ of size $K_{\text{mel}} \times K$. Each row of $\mathbf{R}_{\mathbf{X}}$ contains one single mel-filter, see also [1] and [7]. The rows are normalized, such that the L_1-norm of each row/column of $\mathbf{R}_{\mathbf{X}}^T\mathbf{R}_{\mathbf{X}}$ is equal to one. Another advantage of the mel-filterbank is the reduction of vibrato effects. Vibrato is a small change in pitch over time. A change in pitch is equivalent to a shifting over the logarithmic frequency-axis, see also [3]. This can be interpreted, that in linear frequency domain the higher frequencies are shifted more compared to the lower frequencies. The mel-filterbank reduces this non-linear shifting effect to a linear shifting effect, which is advantageous for further analysis, like the non-negative tensor factorization, see also next section.

2.2. Non-Negative Tensor Factorization

Note separation is done as described in [2]: $\mathbf{X}_{\text{mel}}$ is factorized into several acoustical events by the non-negative tensor factorization (NTF). This factorization is done by approximation of $\mathbf{X}_{\text{mel}}$ with three matrices: $\mathbf{B}_{\text{mel}}$ is of size $K_{\text{mel}} \times I$, $\mathbf{G}$ is of size $T \times I$, and $\mathbf{A}$ is of size $C \times I$. I is a user defined parameter, controlling the number of components used for approximation:

$$\mathbf{X}_{\text{mel}}(k,t,c) \approx \widetilde{\mathbf{X}}_{\text{mel}}(k,t,c) = \sum_{i=1}^{I} \mathbf{A}(c,i)\mathbf{B}_{\text{mel}}(k,i)\mathbf{G}(t,i)\ . \tag{1}$$

We apply iteratively the following multiplicative update rules

$$\mathbf{A}(c,i) \leftarrow \mathbf{A}(c,i)\frac{\sum_{k,t}\mathbf{B}_{\text{mel}}(k,i)\mathbf{G}(t,i)\xi(k,t,c)}{\sum_{k,t}\mathbf{B}_{\text{mel}}(k,i)\mathbf{G}(t,i)} \tag{2}$$

$$\mathbf{G}(t,i) \leftarrow \mathbf{G}(t,i)\frac{\sum_{k,c}\mathbf{B}_{\text{mel}}(k,i)\mathbf{A}(c,i)\xi(k,t,c)}{\sum_{k,c}\mathbf{B}_{\text{mel}}(k,i)\mathbf{A}(c,i)} \tag{3}$$

$$\mathbf{B}_{\text{mel}}(k,i) \leftarrow \mathbf{B}_{\text{mel}}(k,i)\frac{\sum_{t,c}\mathbf{G}(t,i)\mathbf{A}(c,i)\xi(k,t,c)}{\sum_{t,c}\mathbf{G}(t,i)\mathbf{A}(c,i)}\ , \tag{4}$$

with $\xi(k,t,c) = \widetilde{\mathbf{X}}_{\text{mel}}^{-1}(k,t,c) \cdot \mathbf{X}_{\text{mel}}(k,t,c)$, to minimize the generalized Kullback-Leibler-divergence

$$d\left(\mathbf{X},\widetilde{\mathbf{X}}\right) = \sum_{k,t,c}\mathbf{X}(k,t,c)\log\frac{\mathbf{X}(k,t,c)}{\widetilde{\mathbf{X}}(k,t,c)} + \widetilde{\mathbf{X}}(k,t,c) - \mathbf{X}(k,t,c)\ . \tag{5}$$

For numerical stability, we normalize each column of $\mathbf{A}$, $\mathbf{B}_{\text{mel}}$, and $\mathbf{G}$ to the same energy after each iteration. In the monaural case ($C=1$), we set all $\mathbf{A}(c,i)=1$ and the NTF-model simplifies to a non-negative matrix factorization (NMF) model, as described in [2]. Additionally, we apply a *noise gate* before the dimension-reduction by the mel-filterbank: $\mathbf{X}_{\max}(t)$ defines the maximum over all channels c and frequencies k of $\mathbf{X}$ for each column t. If $\mathbf{X}_{\max}(t)$ is 60 dB below the overall maximum of $\mathbf{X}$, the corresponding column is dropped. After NTF, the dropped rows of matrix $\mathbf{G}$ are filled with small constant values $\left(< 10^{-15}\right)$.
As in [7], the frequency-vectors $\mathbf{B}$ in linear frequency domain are evaluated by

$$\mathbf{B} = \mathbf{R}_{\mathbf{X}}^{\mathbf{T}}\mathbf{B}_{\text{mel}}\ . \tag{6}$$

After convergence, the i-th column of $\mathbf{B}$ corresponds to a spectrum for sound event i, the i-th column of $\mathbf{G}$ represents the temporal envelope of the sound event i, and the

i-th column of $\mathbf{A}$ describes the loudness in the C different channels.

Initialization is motivated by [8] and [9]. For harmonic notes, the frequency bins have strong correlation. Therefore, we initialize the matrix $\mathbf{B}_{\text{mel}}$ with 88 artificial harmonic notes corresponding to the 88 keys of the piano. The pitch starts with 27.5 Hz for the first column of $\mathbf{B}$. 20 cosines are added up to form the pitch and the first 19 harmonic overtones of the current pitch. The amplitudes of the cosines decreases with 3 dB/octave. This signal is windowed by the analysis window of the STFT and transformed by the discrete Fourier transform (DFT). The resulting magnitude spectrum is filtered by $\mathbf{R_X}$ and stored in the first column of $\mathbf{B}_{\text{mel}}$. The pitch increases by a half tone step and the next column of $\mathbf{B}_{\text{mel}}$ is evaluated. $\mathbf{A}$ and $\mathbf{G}$ are initialized with ones. After each iteration of NTF, the I_0-th column of $\mathbf{B}_{\text{mel}}$, $\mathbf{G}$, and $\mathbf{A}$ is discarded, as long as the number of columns of $\mathbf{B}_{\text{mel}}$ is larger than I. I_0 is defined by

$$I_0 = \arg\min_i \sum_c \mathbf{A}^2(c,i) \cdot \sum_k \mathbf{B}^2_{\text{mel}}(k,i) \cdot \sum_t \mathbf{G}^2(t,i) \ . \quad (7)$$

2.3. Signal Synthesis

Signal synthesis is done the same way as described in [2]. The separated (complex) tensor for the estimated signal $y_i(n,c)$ is given by the following filter operation:

$$\underline{\mathbf{Y}}_i(k,t,c) = \underline{\mathbf{X}}(k,t,c) \frac{\mathbf{B}(k,i)\mathbf{G}(t,i)\mathbf{A}(c,i)}{\sum_{l=1}^{I} \mathbf{B}(k,l)\mathbf{G}(t,l)\mathbf{A}(c,l)} \ , \quad (8)$$

The channels c of the time domain signals $y_i(n,c)$ are constructed by the inverse STFT for slice c of $\underline{\mathbf{Y}}_i$.

3. SOURCE-FILTER BASED NOTE CLUSTERING

We reconstruct the estimated source signals $\tilde{s}_m(n,c)$ with a clustering operation defined by the vector $\mathbf{a}$:

$$\tilde{s}_m(n,c) = \sum_{i=1}^{I} y_i(n,c)\delta_{m\mathbf{a}(i)} \ , \quad (9)$$

where $\delta_{m\mathbf{a}(i)}$ is the Kronecker symbol, selecting the separated acoustical events belonging to source $\tilde{s}_m(n,c)$:

$$\delta_{xy} = \begin{cases} 1, \text{ if } x = y \\ 0, \text{ otherwise.} \end{cases} \quad (10)$$

$\mathbf{a}$ has I elements, $1 \leq \mathbf{a}(i) \leq M$, $\mathbf{a}(i) \varepsilon \mathbb{N}$. In this Section, the necessary clustering algorithms are introduced.

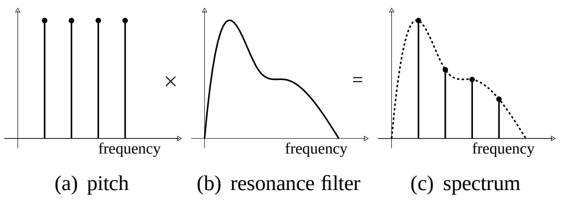

Fig. 1: Source-filter model for frequency domain: description of spectral envelopes.

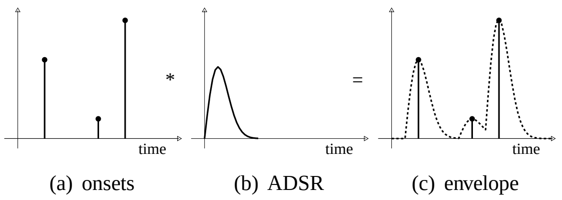

Fig. 2: Source-filter model for time domain: description of tempo, amplitudes, and instrument specific temporal envelopes.

3.1. Source-Filter Models

As mentioned in Section 2.2, the i-th column of $\mathbf{B}$ describes the spectrum, the i-th column of $\mathbf{G}$ describes the temporal envelope of factorized note i. In Figure 1, the source-filter model (SFM) for spectra of harmonic notes is shown. A harmonic note is described by a convolution of a pure harmonic source signal with an instrument specific filter. This convolution can be described by a multiplication in frequency domain, thus leading to the specific envelope of a note in the frequency domain. The novelty of this paper is to extend this source-filter model to the time-axis of the spectrogram. This modeling is shown in Figure 2. It is assumed, that a single note of an instrument has a characteristic envelope. This is similar to the assumption of modeling the envelope of a note by the attack-decay-sustain-release (ADSR) model. The envelope of a factorized note can be described by a convolution of the ADSR model with a sparse signal encoding the amplitudes and the onsets of the single instances of the note. This convolution is already utilized for extended factorization methods in [1], but to the best of our knowledge, modeling this convolution with an SFM used for clustering is a novelty.

According to [10], the convolution of source and filter signals in time-domain is expressed by an addition in cepstral domain. Additionally it is mentioned, that source-signals are located at the highpass part in quefrency-domain and filter signals are located at lowpass-part of quefrency-domain. Thus, the filter-signal is the result of applying a lowpass, the source-signal is the result of applying a highpass on the logarithmic magnitude. As a summary, separating the source and the filter signal can be done in three steps:

1. transforming the signal into frequency domain,
2. taking the logarithm of the magnitude spectrum, and
3. applying a linear filter on the logarithmic spectrum.

These three steps are also necessary for evaluating the MFCC, which are also based on the SFM. For MFCC, the discrete cosine transform is used as linear filter. Dropping the highest-frequency coefficients results in a description of the filter-signal. Contrary to the proposed order of signal processing steps, we switch steps 2 and 3, because this leads to much better separation quality. This swapping of processing steps is also used e.g. by [4].
The basic idea of source-filter based note clustering is to transform the columns of matrices **B** and **G** by these three signal processing steps. By applying this for each component i, a set of instrument-specific features are extracted, which can be clustered into M clusters.

3.2. Feature Evaluation

As described in Section 3.1, the spectrum and the envelope of a note can be described by a two-dimensional SFM. Because, these two dimensions of the SFM are independent in spectrogram domain, we will describe their usage in our BSS framework separately. We now introduce a framework for utilizing these two models for clustering in a BSS framework.
First, the feature evaluation for the SFM describing the columns of **B** (SFM$_{\mathbf{B}}$) is explained. The first step can be discarded for the columns of **B**, because the columns of **B** are already in frequency domain. As mentioned in [2], it is useful to scale the matrix **B** with a factor $f_{\mathbf{B}}$ to a given dynamic range before applying the logarithm. The linear filters for the last step are described below.
Second, the feature evaluation for the SFM describing the columns of **G** (SFM$_{\mathbf{G}}$) is described. The columns of **G** are transformed into frequency domain, the dynamic range of the resulting spectrum is scaled by a factor $f_{\mathbf{G}}$, and the logarithm is applied to the resulting magnitude spectra.
The choice of an appropriate linear filter is not straightforward. In [2] a mel filterbank performs the lowpass filtering of the spectrum, which means different behaviour for lower and higher frequencies. It is easy to see in Figure 2 that in the case of envelope modeling, the source signal contains informations about tempo, rhythm and amplitudes. These obviously important informations should not be discarded. Therefore we will derive optimal filter banks in Section 4.1. The final evaluation of the features can be summarized as follows:

$$\mathbf{F_B} = 20\log_{10}\left(f_{\mathbf{B}}\mathbf{R_B}\mathbf{B}+1\right), \tag{11}$$
$$\mathbf{F_G} = 20\log_{10}\left(f_{\mathbf{G}}\mathbf{R_G}\left|\mathscr{F}\left\{\mathbf{G}\right\}\right|+1\right), \tag{12}$$

where $\mathscr{F}\{.\}$ denotes the columnwise DFT with dropping the second (symmetric) half of the spectrum. The number of frequency-bins in **B** and $\mathscr{F}(\mathbf{G})$ is equal, due to zero-padding. $\mathbf{R_{B/G}}$ are matrices with each row containing the filter coefficients corresponding to the appropriate linear filter. The feature matrices $\mathbf{F_{B/G}}$ have I columns and $K_{\text{Feature},\mathbf{B/G}}$ rows, thus representing I feature samples of dimension $K_{\text{Feature},\mathbf{B/G}}$. $K_{\text{Feature},\mathbf{B/G}}$ depends on the number of filters in $\mathbf{R_{B/G}}$. As explained in Section 4.1, $K_{\text{Feature},\mathbf{B}}$ do not have to be equal with $K_{\text{Feature},\mathbf{G}}$.

3.3. Clustering by Non-Negative Matrix Factorization

The features evaluated in Section 3.2 need to be clustered for signal synthesis of the separated sources. In [2], it is shown that NMF based clustering outperforms k-Means clustering for this task. [11] show the equivalence of standard NMF ($\mathbf{F} \approx \mathbf{W}\mathbf{V}^T$) and a so called *weighted NMF* for a similarity matrix ($\mathbf{F}_{\text{sim}} = \mathbf{F}^T\mathbf{F} \approx \mathbf{A}\mathbf{B}\mathbf{A}^T$). Because of this equivalence, the features $\mathbf{F_{B/G}}$ are fed independently into a standard NMF which gives us an approximation

$$\mathbf{F_{B/G}}(k,i) \approx \sum_{m=1}^{M}\mathbf{W}(k,m)\mathbf{V}(i,m)\ . \tag{13}$$

The frequency bin k in Equation (13) corresponds to the index of the filterbank output of $\mathbf{R_{B/G}}$. The m-th column of **W** corresponds to the m-th cluster center. The m-th column of **V** corresponds to the m-th connectivity values. Therefore, we define the clustering vector necessary for Equation (8) by

$$\mathbf{a}(i) = \arg\max_{m}\mathbf{V}(i,m)\ . \tag{14}$$

One reason for NMF clustering performing better than k-means is the different definition of cluster centers: Assuming noisy non-negative features, it is obvious that higher values in $\mathbf{F}_{\mathbf{B}/\mathbf{G}}$ have higher *feature to noise ratio* than smaller values. Clustering with NMF utilize these feature vectors by allowing different weightings $\mathbf{V}(i,m)$ for the cluster connectivities and by this paying more attention to higher feature values due to the cost function in Equation (5). In contrast to this, the k-means algorithm needs normalized feature vectors because unimodal distributions around cluster centers are assumed [11]. By this, k-means discards the amplitude information (weighting) between the feature vectors.

3.4. Combination of Clustering Methods

In [3], the two feature spaces based on $\mathbf{B}$ and $\mathbf{G}$ are multiplied, before the feature space is clustered. Unfortunately, we cannot combine our feature matrices $\mathbf{F}_{\mathbf{B}/\mathbf{G}}$ in such a way, because it does not fit to our model. On the other hand, it is not possible to concatenate the feature spaces to a larger feature space. Using the NMF for clustering assumes a certain connectivity of each column of $\mathbf{F}_{\mathbf{B}/\mathbf{G}}$ (feature vector) to each of the m cluster centers. The clustering decision is done only on a direct comparison of these values, as shown in Equation (14). Note that these connectivity values can have large differences for both feature spaces, even if the clustering decision is the same.

Therefore, we propose an alternative scheme to decide which feature space is preferrable for a given mixture. By default we use $\mathbf{F}_\mathbf{B}$ for clustering. We count the *number of note instances* for each column of $\mathbf{G}$ by subtracting the mean value of the corresponding column and counting the zero crossings from negative to positive values. If the mean of all *number of note instances* is greater than a given threshold ϑ, the clustering information *tempo* is more reliable than the clustering decision based on instrument filters. In this case, $\mathbf{F}_\mathbf{G}$ is used for clustering. Therefore $\vartheta = \infty$ corresponds to clustering with features $\mathbf{F}_\mathbf{B}$, $\vartheta = 0$ corresponds to clustering with $\mathbf{F}_\mathbf{G}$. In the following, only the parameter ϑ is given as description, which features are used for clustering.

4. EXPERIMENTAL RESULTS

For evaluation, we use the distortion measure SDR, SIR, and SAR presented in [12]. But for better readability, generally, we use only the overall distortion measure SDR for comparison with [5].

In the following we will use four different sets of audio data: $\mathscr{A}$ correspond to the 60 monaural recordings used in [7]. $\mathscr{A}$ are used only for training purpose. $\mathscr{B}$ are the five mixtures of the professional produced music set used for the SiSEC 2010 ([12]). $\mathscr{C}$ are a subset of the underdetermined speech and music mixtures used for SiSEC. The mixtures with only human speakers are not used, because identical sources are hard to separate by a feature based blind clustering algorithm. For such scenarios, trained models for female and male speakers can lead to good separation results, e.g. presented in [9]. $\mathscr{D}$ are the 25 mixtures used as data for [6].

The window-size for STFT is 80 milliseconds with an overlap of 50 %. For analysis and synthesis the square root of *Hann* window is used. The next power of two is used as transformation length for DFT. For NTF we use 400 iterations, $\beta = 1$, and $I = 15$. $\mathbf{R}_\mathbf{X}$ has 500 mel filters. For clustering, $\vartheta = 1.6$ leads to good separation quality, and $f_\mathbf{B}$ and $f_\mathbf{G}$ scales the feature space to 50 dB dynamic range.

4.1. Derivation of Parameters

For derivation of good filter banks $\mathbf{R}_{\mathbf{B}/\mathbf{G}}$, we use $\mathscr{A}$ as training data. We derive $\mathbf{R}_\mathbf{B}$ by the following *brute force* algorithm:

1. Initialize the rows of $\mathbf{R}_\mathbf{B}$ with uniformly spaced gaussian filters, see also Figure 3(a).
2. For each filter: try to merge two neighboring filters.
3. For each filter: try to split into two filters.
4. For each filter: try to move filter center up or down one frequency bin.
5. Repeat steps 2-4 until no more improvements are possible, with respect to mean SDR on the whole test set.

All rows of $\mathbf{R}_\mathbf{B}$ are scaled to unit energy. $\mathbf{R}_\mathbf{G}$ is derived accordingly. In Figure 3(b), the final filterbank $\mathbf{R}_\mathbf{B}$ is shown. The logarithmic scaling of frequency axis seems to lead to meaningful features, regarding the task of clustering separated notes. Therefore we use in the following a standard mel-filterbank with 25 mel filters as $\mathbf{R}_\mathbf{B}$. Figure 3(c) shows no logarithmic frequency mapping for $\mathbf{R}_\mathbf{G}$. Better frequency resolution is necessary only for certain middle frequencies. Therefore, we use an uniform lowpass filterbank as linear filterbank $\mathbf{R}_\mathbf{G}$.

To increase separation quality further, we evaluate the influence of the number of lowpass filters in $\mathbf{R}_\mathbf{G}$ on the separation quality. Table 1 shows increasing separation quality for increasing number of filters. Therefore we use in the following the identity matrix as $\mathbf{R}_\mathbf{G}$ (this corresponds to skipping the linear filter for $\mathbf{F}_\mathbf{G}$).

	Bearlin			Tamy		Another Dreamer			Fort Minor				Ultimate NZ Tour		
	vocals	bass	piano	guitar	vocals	vocals	drums	guitar	vocals	drums	bass	claps	vocals	drums	bass
[5]						-0.70	0.90	1.80	3.20	2.00	4.20	2.00	**2.60**	4.20	0.70
$\vartheta=\infty$	**3.11**	**4.67**	-0.17	0.24	0.48	1.03	-0.29	0.71	0.91	0.86	4.79	-0.76	0.14	2.67	1.31
$\vartheta=1.6$	2.60	2.95	-2.80	**9.87**	**10.11**	**2.62**	**4.46**	**3.75**	**3.34**	**2.57**	**5.90**	**4.64**	-0.49	**4.23**	**2.00**
$\vartheta=0$	2.60	2.95	-2.80	9.87	10.11	2.62	4.46	3.75	3.34	2.57	5.90	4.64	-0.49	4.23	2.00

Table 2: SDR in dB for clustering test set $\mathscr{B}$ with different values of ϑ. For [5], the given values are reported by [12].

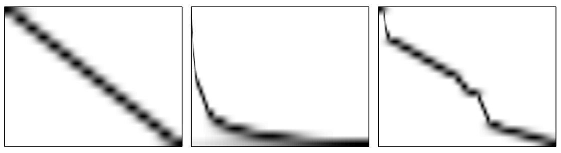
(a) Initial filterbank (b) $\mathbf{R_B}$ (c) $\mathbf{R_G}$

Fig. 3: Initial and optimal filterbanks for spectral- and envelope-based SFM.

n_{filter}	2^5	2^6	2^7	2^8	2^{10}	2^{11}
SDR [dB]	4.95	5.04	5.07	5.10	5.12	5.13

Table 1: Mean SDR over audio data $\mathscr{A}$ for different number of equal spaced lowpass filters in $\mathbf{R_G}$.

4.2. Comparison of Clustering Algorithms with Extended Factorization Models

First, we compare our clustering algorithm with the separation framework of [5]. For this, we evaluate the separation quality for data $\mathscr{B}$. The results are shown in Table 2. It can be seen, that only for the *Bearlin*-song clustering with $\vartheta=\infty$ outperforms clustering with $\vartheta=0$. For data $\mathscr{B}$, the threshold $\vartheta=1.6$ is equivalent with using $\vartheta=0$. Additionally, it can be seen, that clustering with $\vartheta=1.6$ outperforms [5] for nearly all sources. This induces, that for data $\mathscr{B}$, the assumption of point sources combined with a convolutive mixing model used in [5] is not superior to the assumption of an instantaneous mixing model with long analysis windows for STFT as used in our algorithm.

For further comparison with [5], the separation quality is evaluated for data $\mathscr{C}$. Two things are observable: First, the effectiveness of the simple decision rule based on the tempo of the separated notes can be seen. $\vartheta=1.6$ leads to better separation quality compared with $\vartheta=0$ and $\vartheta=\infty$ in nearly all cases. Second, it can be seen, that the assumption of point sources in [5] works much better for these type of mixtures. Most interestingly, only for the convolutive and the live recorded mixtures our algorithm leads sometimes to better results. For instantaneous mixtures, [5] leads to better results.

	instantaneous mixtures					
	nodrums			wdrums		
	s_1	s_2	s_3	s_1	s_2	s_3
[5]	**27.15**	**7.61**	**21.41**	**0.95**	**4.66**	**29.89**
$\vartheta=\infty$	15.13	0.49	2.99	-1.18	-11.16	5.00
$\vartheta=1.6$	15.13	0.49	2.99	-1.18	-1.93	15.07
$\vartheta=0$	7.17	-3.63	-2.55	-1.18	-1.93	15.07
	live recorded mixtures					
	nodrums			wdrums		
	s_1	s_2	s_3	s_1	s_2	s_3
[5]	4.57	**7.34**	-4.51	**4.96**	**5.55**	**8.03**
$\vartheta=\infty$	9.29	0.15	-0.12	1.59	0.25	0.80
$\vartheta=1.6$	**9.29**	0.15	$\mathbf{-0.12}$	1.72	0.84	6.51
$\vartheta=0$	3.95	-3.40	-0.66	1.72	0.84	6.51
	convolutive mixtures					
	nodrums			wdrums		
	s_1	s_2	s_3	s_1	s_2	s_3
[5]	**4.18**	1.02	-1.80	3.93	-12.46	**1.32**
$\vartheta=\infty$	0.16	2.71	-0.98	4.51	-2.08	0.07
$\vartheta=1.6$	0.16	**2.71**	-0.98	**4.51**	$\mathbf{-2.08}$	0.07
$\vartheta=0$	1.74	0.53	$\mathbf{-0.94}$	3.98	-6.36	0.78

Table 3: SDR in dB for clustering test set $\mathscr{C}$ with different values of ϑ. For [5], the given values are reported on the corresponding demo page. s_m are the sources to separate.

To summarize the comparison with [5], our algorithm results in higher separation quality for data $\mathscr{B}$, their algorithm performs better for data $\mathscr{C}$. But our algorithm has several advantages besides the separation quality: First, our algorithm is much faster. The authors of [5] reported a computational load of more than one hour on a 2.2 GHz CPU for each song. A Matlab implementation of our algorithm needs slightly less than 3 minutes per song on an AMD Opteron 2214 processor with 2.2 GHz. Both implementations may not be optimal and the architectures

	SDR	SIR	SAR
[6]	9.01	24.91	9.52
$\vartheta = \infty$	9.80	15.05	15.91
$\vartheta = 1.6$	9.80	15.05	15.91
$\vartheta = 0$	3.80	9.11	13.88

Table 4: Mean separation quality in dB for clustering test set $\mathscr{D}$ with different values of ϑ. For [6], the separation quality is reported on the corresponding website.

may differ, but a speedup by this factor shows the lower computational complexity of our algorithm clearly. Second, we use a homogeneous set of parameters, independently from the used data and the corresponding sampling frequencies of the input mixtures. [5] reports, that the time-frequency resolution and the number of components to factorize (similar to I for our algorithm) is tuned for each dataset. Third, our algorithm needs no initialization by other BSS frameworks and is therefore much simpler to implement.

Finally, we compare our algorithm with [6]. In Table 4, the separation quality for our algorithm and the algorithm introduced in [6] is shown for data $\mathscr{D}$. Contrary to data $\mathscr{B}$, $\vartheta = \infty$ outperforms clustering with $\vartheta = 0$ clearly for $\mathscr{D}$. This can be explained by the pure harmonic content of data $\mathscr{D}$. Similar to the results in Table 2, clustering with $\vartheta = 1.6$ results in good separation quality. It can be seen, that the algorithm proposed in [6] leads to lower distortions by interferences (SIR) but to higher distortions by artifacts (SAR). Therefore it can be assumed, that their algorithm is better suited as preprocessing step for automatic music analysis. But our algorithm works better for remixing or upmixing scenarios, where artifacts are critical. In [6], no information about computational complexity is given. Instead, we will analyse the number of necessary operations for each iteration. The extended NTF-model proposed in [6] needs to update 7 tensors, with each single update consisting of at least 8 tensor products over two dimensions. The basic NTF needs only to update 3 tensors, with each update consisting of 4 tensor products over one dimension. Dropping the element-wise multiplications and divisions for update, it can be deduced, that the extended NTF model of [6] needs at least 4 times more operations compared to the basic NTF used in our algorithm. The clustering step can be done in a fraction of the time, necessary for the basic NTF. Therefore, it is reasonable, that our algorithm is at least 3 times faster compared to [6].

5. CONCLUSIONS

In this paper, we use the NTF for note separation and cluster the separated notes to form the estimated output signals. For this, we extend the one-dimensional source-filter model for spectra used so far to a two-dimensional one, describing the spectra and the envelopes of instruments. This new dimenion of the source-filter model describes the temporal evolution of signal energy by characteristic envelopes of instruments convolved with an activation vector acountable for tempo, rhythm and amplitudes of notes. We suggest an identical processing framework based on cepstral analysis for both independent dimensions of the model. Linear-filtering schemes for cepstral analysis are derived. Finally, we introduce a simple heuristic to combine both dimensions of the source-filter model in a clustering framework. Compared to the clustering algorithm proposed in [3], we are even able to cluster mixtures containing non-harmonic instruments. In contrast to [5], we are able to separate even monaural mixtures. Finally, we compare our algorithm with two state-of-the-art separation frameworks: [5] and [6]. Beside comparable separation quality, our algorithm has three advantages: Lower computational complexity, no external initialization is necessary, and a homogeneous set of parameters can be used for a large range of audio data.

6. REFERENCES

[1] D. FitzGerald, M. Cranitch, and E. Coyle, “Extended nonnegative tensor factorisation models for musical sound source separation,” *Computational Intelligence and Neuroscience*, 2008.

[2] M. Spiertz and V. Gnann, “Source-filter based clustering for monaural blind source separation,” in *Proc. Int. Conference of Digital Audio Effects DAFx* , 2009.

[3] K. Murao, M. Nakano, Y. Kitano, N. Ono, and S. Sagayama, “Monophonic instrument sound segregation by clustering nmf components based on basis similarity and gain disjointness,” in *Proc. Int. Conf. on Music Information Retrieval ISMIR ’*, 2010.

[4] A.B. Nielsen, S. Sigurdsson, L.K. Hansen, and J. Arenas-Garcia, “On the relevance of spectral features for instrument classification,” in *Proc. IEEE Int. Conference on Acoustic Speech and Signal Processing ICASSP ’*, Apr. 2007, vol. 2, pp. 485–488.

[5] A. Ozerov and C. Févotte, "Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 550–563, 2010, `http://www.irisa.fr/metiss/ozerov/demos.html`.

[6] D. FitzGerald, M. Cranitch, and E. Coyle, "Musical source separation using generalised non-negative tensor factorisation models," in *Workshop on Music and Machine Learning, International Conference on Machine Learning*, 2008, `http://eleceng.dit.ie/derryfitzgerald/index.php?uid=489&menu_id=52`.

[7] M. Spiertz and V. Gnann, "Beta divergence for clustering in monaural blind source separation," in *128th AES Convention*, London, UK, May 2010.

[8] S.A. Raczynski, Ono N., and S. Sagayama, "Multipitch analysis with harmonic nonnegative matrix approximation," in *Proc. Int. Conf. on Music Information Retrieval ISMIR '07*, Sept. 2007, pp. 381–386.

[9] T. Virtanen, "Spectral covariance in prior distributions of non-negative matrix factorization based speech separation," in *Proc. European Signal Processing Conference EUSIPCO '09*, Glasgow, Scotland, Aug. 2009, pp. 1933–1937.

[10] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[11] C. Ding, X. He, and H. D. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering," in *Proc. SIAM Data Mining Conf*, 2005, pp. 606–610.

[12] S. Araki, A. Ozerov, V. Gowreesunker, H. Sawada, F. Theis, G. Nolte, D. Lutter, and N. Duong, "The 2010 Signal Separation Evaluation Campaign (SiSEC2010): - Audio source separation -," in *Proc. Int. Conf. on Latent Variable Analysis and Signal Separation '10*, 2010.