# Vorausschauende Online NMF zur Unüberwachten Quellentrennung

# Look-ahead Online NMF for Unsupervised Source Separation

Masterarbeit

Cristian Felipe Martínez Ruiz

Lehrstuhl und Institut für Nachrichtentechnik
RWTH Aachen University
Univ.-Prof. Dr.-Ing. Jens-Rainer Ohm

05.10.2015

# Declaration of Authorship

I hereby declare and confirm that this thesis is entirely the result of my own work. I have only used the stated documents and aids, and also indicated citations correctly.

Aachen, 05.10.2015

# Abstract

Algorithms based on Non-Negative Matrix Factorization (NMF) are commonly used to solve the Blind Source Separation (BSS) problem. The objective of NMF is to split a spectrogram, which is a frequency-time representation of a mixture, in two matrices; the basis matrix and the gain matrix. The most commonly variant used is the iterative NMF algorithm which requires the number of components as well as the full mixture spectrogram. The number of components can be approximated to the number of acoustical events presented in the mixture. Nevertheless, a spectrogram can be quite large, depending on the length of the mixture. A larger mixture introduces high latency and demands larger memory. The main problem is the number of components because it is not usually known beforehand.

Therefore, Online NMF algorithms were developed to avoid these problems. The spectrogram of a mixture is divided in time segments called frames. Thus, instead of iterate over the full spectrogram, the basic online algorithm (ONMF) iterates over frames of the spectrogram to obtain the basis and the gain vectors. Nevertheless, if the basis vectors are estimated with just one frame of the spectrogram, some errors can be introduced due to the fact that a musical note usually requires more than one frame to be developed. In this thesis, the Look-ahead ONMF (LONMF) algorithm is proposed as an improvement of ONMF in order to obtain a better estimation of the basis matrix by taking more frames into account.

# Contents

Chapter 1

# Introduction

## 1.1 Background

In the world of signal processing, the Blind Source Separation (BSS) [1] problem is a commonly encountered problem which tries to estimate sources given a mixture. The BSS problem is also known as the *"cocktail party effect"* [2].

The *"cocktail party effect"* was defined as the phenomenon experienced by a subject when he is trying to have a conversation with one person while there are others speakers at the same time around him.

The Blind Source Separation (BSS) problem is presented in many fields, especially in multisensor systems or in systems which can obtain information from more than one source. The most distinguished fields are: communications with, for example, antenna arrays like Multiple Input Multiple Output (MIMO); audio source separation with music, speech recognition; biomedical applications like electrocardiography, electroencephalography; image processing; monitoring of complex systems and many more [3].

Nowadays there are some algorithms proposed to solve this problem, the most used are: Principal Component Analysis (PCA), Singular Value Decomposition (SVD), Independent Component Analysis (ICA) and Non-Negative Matrix Factorization (NMF). The aim of the PCA [1] procedure is to minimize the number of components from a mix data set losing the least amount of information. This procedure is based on finding correlation between variables presented in the data set using an orthogonal transformation. The goal of the SVD [1] algorithm is to factorize a matrix using the eigenvalues and eigenvectors of two matrices: the products from the input matrix to its conjugate transpose and vice versa. The ICA [1] algorithm is based on two assumptions, the first one is the independence of the source signals and the second one is the non-gaussianity of the sources. Therefore, the target of this algorithm is to achieve independent components with the maximization of the independence of predicted signals. Finally, NMF [1] algorithm was proposed to factorize a non-negative matrix into two non-negative matrices, basis and gain matrices. This algorithm is explained in detail in Chapter 2.

## 1.2 Motivation

The main focus in this thesis is the iterative NMF algorithm proposed by Lee and Seung in 2000 [4] to solve the BSS problem. The iterative NMF algorithm requires as input a time-frequency representation of the mixture called spectrogram ($\boldsymbol{X}$) which is the matrix to be separated into two matrices, the gain matrix and the basis matrix. Nevertheless, a mixture larger in time produces a larger spectrogram $\boldsymbol{X}$ which can introduce latency and demand larger memory. This algorithm also requires the number of components of the mixture as input. The number of components can be approximated as the number of musical notes present in the mixture. In some cases one note can be composed of more than one component like harmonic notes, but this number is not usually known beforehand.

In order to avoid these problems, some Online NMF (ONMF) algorithms were proposed in the master's thesis by Bilal Shuja [5]. The ONMF algorithms do not require the number of components giving it a degree of blindness to the solution of the BSS problem. In this type of algorithms, the spectrogram of a mixture is divided in time segments called frames. On the other hand, the ONMF algorithms iterate over each frame of the spectrogram instead of iterate over the full spectrogram. The main problem of ONMF algorithms is the obtained basis matrix, which requires one or two frames, depending on the algorithm, to learn a new component while a component can be defined with more than three frames. Two of the six Online NMF variants proposed by [5] are explained in detail in Section 2.2, the basic and the best in terms of separation quality.

## 1.3 Scope

As a summary of Section 1.2, the NMF algorithm requires the complete spectrogram and the ONMF algorithms require one or two frames in the component learning procedure. Thus, the scope of this thesis is to provide a new algorithm capable of determinating the number of components of a given mixture and separating it into its sources. The new algorithm has a trade off between the number of frames used to obtain each vector of the basis matrix and the computational effort of the algorithm. This algorithm has a lower latency than NMF and also higher number of frames in the learning components procedure than ONMF in order to achieve better results in terms of separation quality.

## 1.4 Structure of the Thesis

The essence of this thesis is distributed in the following four Chapters and a brief summary of each one is presented in this Section.

In Chapter 2, the fundamentals of this thesis are presented. First of all, the Section 2.1

contains a detailed explanation of the BSS algorithm with the iterative NMF algorithm. Afterwards, in Section 2.2, two different algorithms are presented: the basic Online NMF in Section 2.2.1 and a variant of ONMF which estimates basis vectors using current and past frame in Section 2.2.2. The last two algorithms were proposed by [5].

In Chapter 3, a new NMF based algorithm is proposed which is called Look-ahead Online NMF. The main differences between LONMF and the rest of the algorithms are exposed in the last Section (3.2) of this Chapter with a result comparison of the six seconds mixtures used by [5].

The results obtained by the LONMF algorithm in a larger test set are provided in Chapter 4. More than 250 mixtures were considered for this thesis and a discussion of those results is presented in this Chapter. It should be pointed out that the time length of these mixtures is between 13 and 30 seconds from 10 songs with the purpose to have a stronger test bench than [5].

Chapter 5 concludes this master's thesis and gives advantages and disadvantages of the proposed algorithm compared with NMF and both ONMFs.

Chapter 2

# Fundamentals

## 2.1 Basic Blind Source Separation Algorithm

The basic BSS procedure for Non-negative Matrix Factorization (NMF) based algorithms is exposed in this Section. Basically, the BSS procedure is able to separate a set of sources $s_m(n)$ into a set of estimated sources $\tilde{s}_m(n)$. This procedure is divided in four steps as described as follows.

The input sources $s_m(n)$ are combined as the mixture signal $x(n)$. The mixed signal $x(n)$ is the input required by the next step, the Short-Time Fourier Transform, which output is a time-frequency representation of the input signal called spectrogram $\boldsymbol{X}(k,t)$. As was exposed before, the spectrogram matrix is split in two matrices, the basis $\boldsymbol{B}(k,i)$ and gain $\boldsymbol{G}(i,t)$ matrices by NMF as third step. As the final step, the basis and gain matrices are transformed into the estimated signals $\tilde{s}_m(n)$ by the synthesis and clustering step.

The basic BSS algorithm is summarized in the block diagram shown in Figure 2.1. Each block of the diagram is explained in detail in Sections 2.1.1-2.1.4, as long as its inputs and outputs.

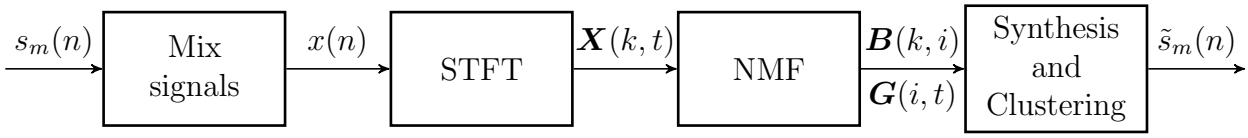


Figure 2.1: Blind Source Separation block diagram.

### 2.1.1 Mixing Model

In audio source separation, there are different types of sources, like instruments with percussive and harmonic notes; human voice; sounds from the nature like ambient noise, wind, rain, animals and many more.

This thesis is focused on musical sources. The algorithms described in Sections 2.1.3 and 2.2 are proposed to separate mixtures of two instruments or a voice and an instrument. Mixtures with $M = 2$ sources are considered.

The mixture signal is assumed to be a linear combination of both sources as is shown in Equation 2.1 for discrete time,

$$x(n) = \sum_{m=1}^{M} s_m(n) \tag{2.1}$$

where, $s_m(n)$ is the input time signal corresponding to source number $m$ and $x(n)$ is the mixture signal.

An exemplary mixture signal is shown in Figure 2.2. This toy example is used along this Section in order to explain in detail the basic BSS algorithm.

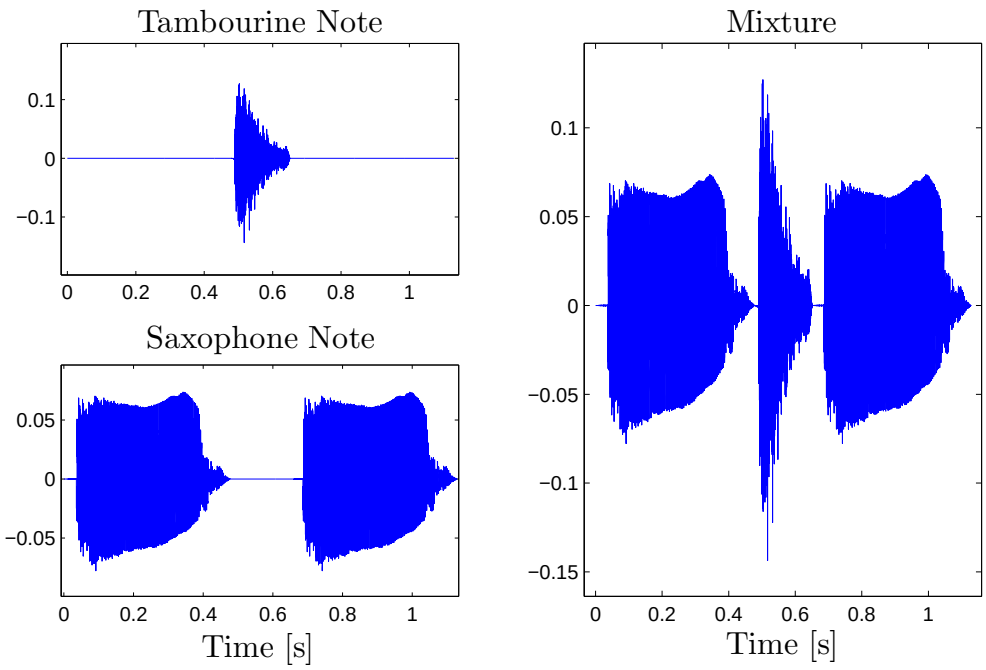


Figure 2.2: Sources and Mixture Signals in time-domain.

In this example, the mixture is composed of a tambourine note and a saxophone note. Therefore, the sources are percussive and harmonic notes respectively.

### 2.1.2 Short-Time Fourier Transform

As was explained in Chapter 1, the NMF algorithm requires a time-frequency representation of the mixture as input which is called spectrogram ($\boldsymbol{X}$). There is a mathematical operation that associates the frequency domain representation to a function of time which is called Fourier Transform. Thus, the Fourier Transform is a tool which obtains the energy distribution through the frequency components presented in a time domain signal. However, it does not obtain information about time.

The Short-Time Fourier Transform (STFT) provides a mixture spectrogram by analyzing the frequency behavior of the signal with a finite time window [6]. The STFT procedure obtains segments from the input signal taken the product between the original signal and a window which moves with time. Then, it applies the Fourier Transform to each segment. Therefore, the Fourier Transform and the product describes the behavior of the signal in the frequency and the time span. This procedure is explained in this Section in detail.

As was exposed before, a finite time window is required. There are many types of windows that can be used for this purpose, like Rectangular, Triangular, Parzen, Welch, Hamming, Hanning, Gaussian and many more. The square root of overlapping *Hanning windows* are used in this thesis. An example of overlapping *Hanning windows* over the input mixture used as toy example is presented in Figure 2.3.

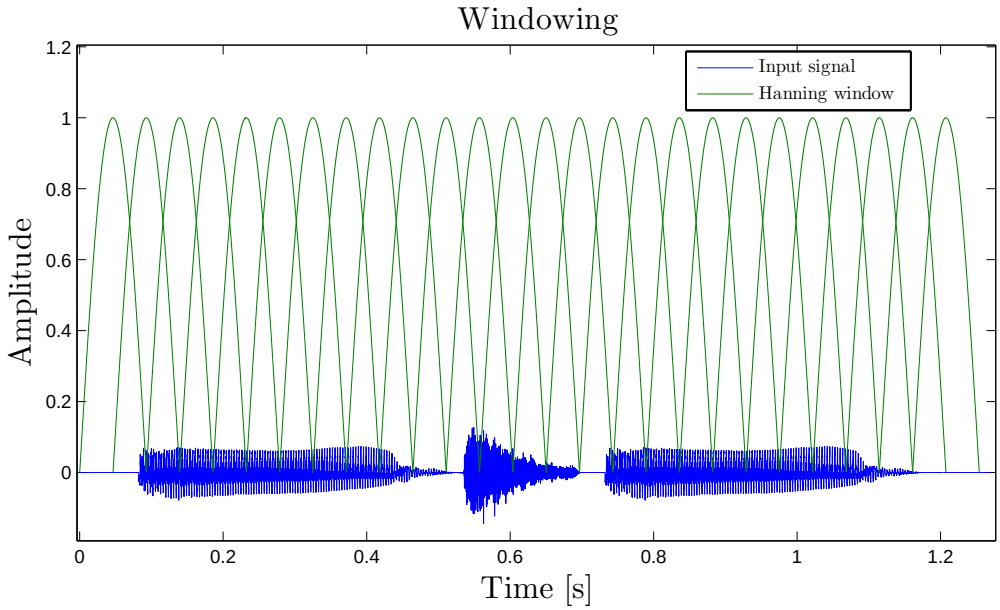


Figure 2.3: Short Time Fourier Transform Windowing.

Afterwards, the input signal $x(n)$ is divided in windowed segments $x_t$ as shown in Equation 2.2,

$$x_t(n) = w(n)x(n + tH), \text{ for } 0 \leq n \leq L - 1 \tag{2.2}$$

where $t$ is the frame index. For this thesis the *Hanning window* length is $L = 2^{12}$ and the hop size is $H = 2^{11}$. As a second step for STFT, the Discrete Fourier Transform DFT of each signal segment $x_t(n)$ is calculated as

$$\underline{\boldsymbol{X}}(k,t) = \sum_{n=0}^{L-1} x_t(n) e^{-j\frac{2\pi k}{L}n} \tag{2.3}$$

where $k$ denotes the frequency index. $\underline{\boldsymbol{X}}(k,t)$ denotes a complex matrix due to the DFT, but the NMF algorithm requires a real non-negative matrix as input. Thus, the input for the NMF based algorithm is the absolute value of $\underline{\boldsymbol{X}}(k,t)$. The spectrogram of the input signal presented in Section 2.1.1 is shown in Figure 2.4.

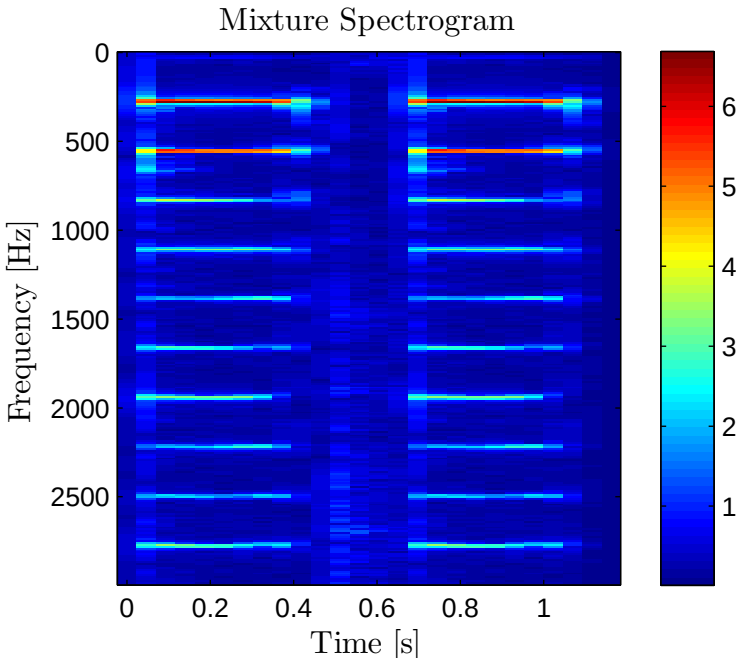


Figure 2.4: Mixture Spectrogram.

The percussive note and the harmonic note can be distinguished in the mixture spectrogram shown in Figure 2.4 due to the fact that the vertical structure presented in the spectrogram belongs to the percussive note and the horizontal structure belongs to the harmonic note.

### 2.1.3 Non-negative Matrix Factorization

As shown in Figure 2.1, the next block of the BSS procedure is the NMF. Therefore, the iterative NMF algorithm is explained in detail in this Section and two Online NMF based algorithm will be explained in Section 2.2 as alternatives. The iterative NMF was proposed by Lee and Seung [4] in 2000. The goal of the iterative NMF is to factorize an input non-

negative matrix (spectrogram) into two non-negative matrices, basis and gain matrices as is represented in Equation 2.4,

$$|\underline{\boldsymbol{X}}_{K\times T}| = \boldsymbol{X}_{K\times T} \approx \tilde{\boldsymbol{X}}_{K\times T} = \boldsymbol{B}_{K\times I}\boldsymbol{G}_{I\times T} \tag{2.4}$$

where, the absolute value of the mixture spectrogram is approached as the product of the basis $\boldsymbol{B}$ and the gain $\boldsymbol{G}$ matrices. Therefore, the input spectrogram $\boldsymbol{X}_{K\times T}$ and the estimated spectrogram $\tilde{\boldsymbol{X}}_{K\times T}$ are non-negative matrices which sub-indexes correspond to frequency span ($K$) and time span ($T$). The basis matrix $\boldsymbol{B}_{K\times I}$ and the gain matrix $\boldsymbol{G}_{I\times T}$ are also non-negative matrices which sub-index ($I$) corresponds to the number of components of the audio mixture. $I$ is an user parameter and it must be chosen to be smaller than $K$ and $T$.

This algorithm requires a cost function in order to measure the quality of approximation of the basis and the gain matrices product. Lee and Seung [4] proposed two cost functions in their paper. The first cost function is the square of the Euclidean distance $d_{\text{euc}}$ of two matrices as is shown in Equation 2.5.

$$d_{\text{euc}}[\boldsymbol{X}, \tilde{\boldsymbol{X}}] = ||\boldsymbol{X} - \tilde{\boldsymbol{X}}||^2 \tag{2.5}$$

The lower bound of $d_{\text{euc}}$ is zero and it will be zero only if $\boldsymbol{X} = \tilde{\boldsymbol{X}}$. Another cost function is the divergence $d_{\text{div}}$ of matrix $\boldsymbol{X}$ from matrix $\tilde{\boldsymbol{X}}$ as is presented in Equation 2.6,

$$d_{\text{div}}[\boldsymbol{X}, \tilde{\boldsymbol{X}}] = \sum_{k,t} d_{\text{KL}}(\boldsymbol{X}(k,t)||\tilde{\boldsymbol{X}}(k,t)) \tag{2.6}$$

where the function $d_{\text{KL}}$, which is the Kullback-Leibler divergence, is defined as 2.7:

$$d_{\text{KL}}[p, q] = p\log\left(\frac{p}{q}\right) - p + q \tag{2.7}$$

The iterative multiplicative update rules presented in Equations 2.8 were proposed by Lee and Seung, to minimize the cost function in Equation 2.5. It should be pointed out that for both cost functions the $\boldsymbol{B}$ and the $\boldsymbol{G}$ matrices are updated depending on each other for every iteration. All operations are element wise.

$$\boldsymbol{G} \longleftarrow \boldsymbol{G}. \times \frac{\boldsymbol{B}^T\boldsymbol{X}}{\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{G}} \tag{2.8a}$$

$$\boldsymbol{B} \longleftarrow \boldsymbol{B}. \times \frac{\boldsymbol{X}\boldsymbol{G}^T}{\boldsymbol{B}\boldsymbol{G}\boldsymbol{G}^T} \tag{2.8b}$$

On the other hand, the iterative multiplicative update rules presented in Equations 2.9 were proposed by Lee and Seung, to minimize the divergence with $\tilde{\boldsymbol{X}} = \boldsymbol{BG}$.

$$\boldsymbol{G} \longleftarrow \boldsymbol{G}. \times \frac{\boldsymbol{B}^T \frac{\boldsymbol{X}}{\boldsymbol{BG}}}{\boldsymbol{B}^T \boldsymbol{1}} \tag{2.9a}$$

$$\boldsymbol{B} \longleftarrow \boldsymbol{B}. \times \frac{\frac{\boldsymbol{X}}{\boldsymbol{BG}} \boldsymbol{G}^T}{\boldsymbol{1} \boldsymbol{G}^T} \tag{2.9b}$$

In order to obtain the results of the toy example spectrogram shown in Figure 2.4, the matrices $\boldsymbol{B}$ and $\boldsymbol{G}$ are initialized with random values and both matrices were updated with the divergence update rules as a cost function. The number of iterations $N_{\text{iter}}$ used for this algorithm is set to 300. The same conditions will be used for the rest of the thesis for iterative NMF algorithm. The output of the iterative NMF algorithm is shown as an example in Figure 2.5.

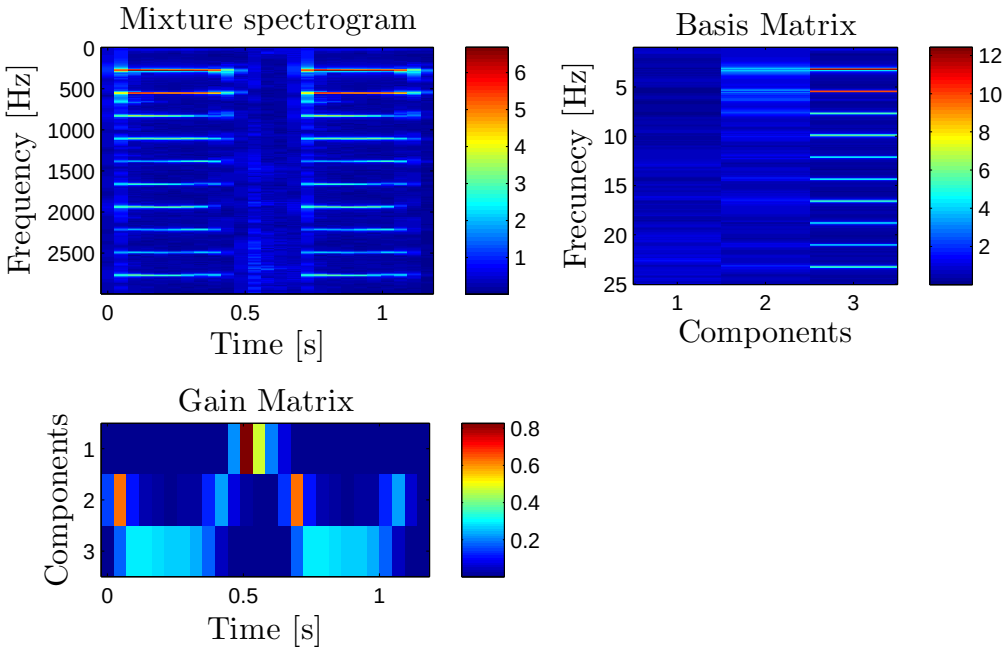


Figure 2.5: Iterative NMF output.

In Figure 2.5, it can be seen that the representation of the percussive note, as the component $I = 1$, is presented in the basis matrix at medium frequencies and in gain matrix around 0.5 seconds. The component $I = 2$ is the representation of the on/off sets of the harmonic note as can be seen at very low frequencies in the basis matrix and in gain matrix in time domain. The third component is the sustained harmonic note as is shown in the gain matrix in time slots 0.1-0.4 seconds and 0.6-1 seconds.

The Iterative NMF pseudo code used in this thesis is presented in Algorithm 2.1 as a summary.

**Algorithm 2.1** Iterative Non-negative Matrix Factorization Algorithm.

**Require:** Number of components: $I$, Iteration number: $N_{\text{iter}}$, spectrogram: $\boldsymbol{X}_{K\times T}$

1: Initialize $\boldsymbol{B}_{K\times I}$ and $\boldsymbol{G}_{I\times T}$ with random values.
2: **for** $i = 1$ to $N_{\text{iter}}$ **do**
3: Update $\boldsymbol{G}$ matrix using 2.9a
4: Update $\boldsymbol{B}$ matrix using 2.9b
5: **end for**
6: **return** $\boldsymbol{G}$ and $\boldsymbol{B}$

The block diagram of the NMF algorithm is presented in the Appendix Chapter in Figure A.1 in order to complement the explanation.

### 2.1.4 Synthesis and clustering

The product between each basis vector times each gain vector obtained by the NMF algorithm provides the spectrogram of each component as presented in Equation 2.10.

$$\boldsymbol{C}_{K\times T|i} = \boldsymbol{B}_{K\times 1}(i)\boldsymbol{G}_{T\times 1}(i) \tag{2.10}$$

where, $\boldsymbol{C}_{K\times T|i}$ is the spectrogram of the $i_{\text{th}}$ component, $\boldsymbol{B}_K(i)$ is the basis vector of the $i_{\text{th}}$ component and $\boldsymbol{G}_T(i)$ is the gain vector of the $i_{\text{th}}$ component.

The Inverse Short-Time Fourier Transform (ISTFT) is used to extract a time domain signal from a complex spectrogram, exactly the opposite procedure as the STFT. A normalization is required in order to check that component spectrograms add up to the mixture spectrogram and in order to obtain the complex spectrogram of each component which is required by the ISTFT. This normalization can be represented mathematically as shown in Equation 2.11.

$$\underline{\tilde{\boldsymbol{C}}}_{K\times T|i} = \underline{\boldsymbol{X}}_{K\times T}\frac{\boldsymbol{C}_{K\times T|i}}{\sum_{i=1}^{I}\boldsymbol{C}_{K\times T|i}} \tag{2.11}$$

where $\underline{\tilde{\boldsymbol{C}}}_{K\times T|i}$ is the estimated spectrogram of the $i_{\text{th}}$ component.

As an example, the spectrogram of each component of the toy example is shown in Figure 2.6.

The next step after the NMF is the synthesis part which transforms the component spectrograms into time domain with the ISTFT. For this procedure is used the same parameters than the used for the STFT.

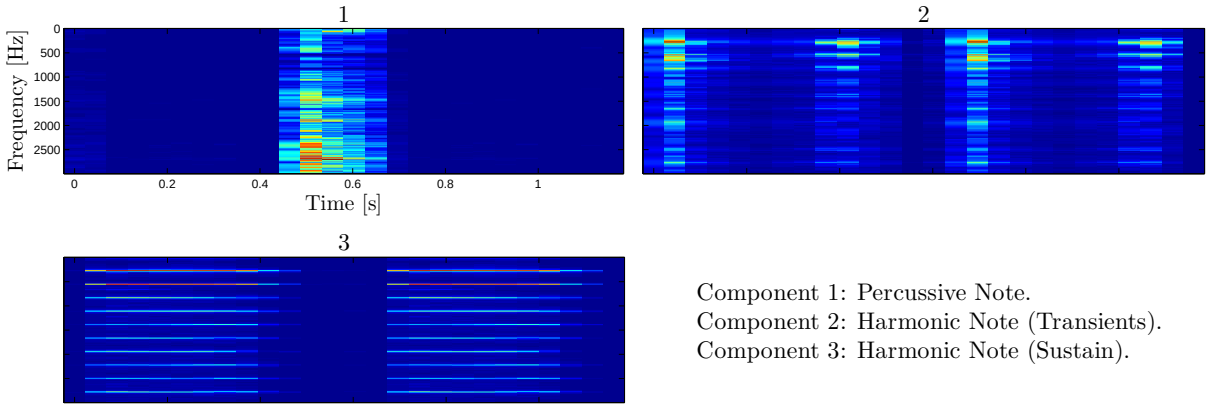


Figure 2.6: Component spectrograms.

Time domain signals for every component of the toy example are shown in Figure 2.7,

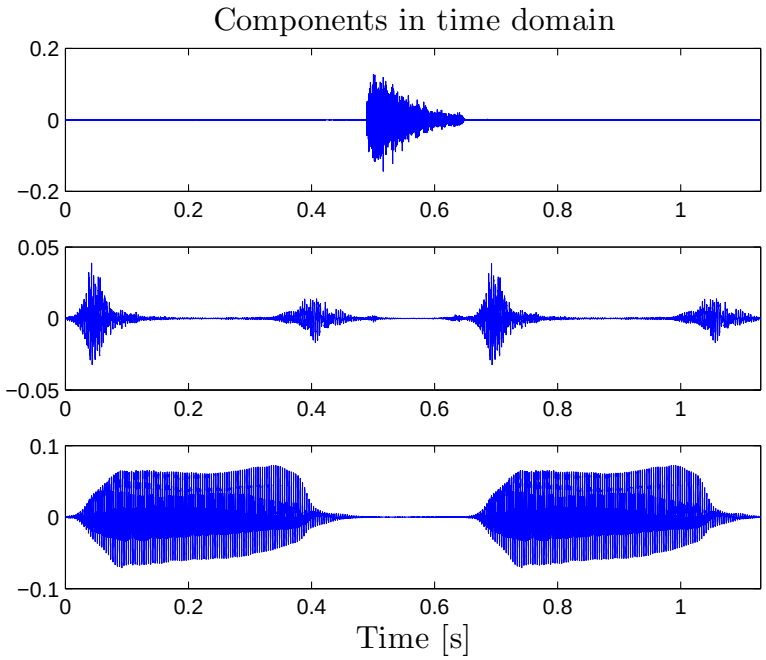


Figure 2.7: Components in time domain.

where, the estimated tambourine note is presented in the top of the Figure, the transients or on/off sets of the estimated saxophone note are plotted in the middle and finally, the sustained part of the estimated saxophone note is shown in the last plot.

Therefore, the structure of each type of note can be described if the Figures 2.6 and 2.7 are compared. The first spectrogram belongs to the first signal in time domain. This is the tambourine note which is a percussive audio signal. Thus, a vertical structure of a spectrogram belongs to a percussive note. The second component belongs to the second signal in time domain, this component belongs to the saxophone note. Nevertheless, the

structure of the spectrogram is vertical. The vertical structure of this component is due to the fact that is the percussive part of the harmonic note. In other words, it is the sound produced when the musician presses and releases the key of the saxophone which produces the harmonic note. This parts of the note are called transients or on/off sets. The third component is described by the third spectrogram which belongs to the last signal in time domain. Its horizontal structure is the principal feature of a sustained harmonic note. Thus, this component belongs to the saxophone note.

Usually, there are more components than sources, this is why a clustering part is required. The clustering is done using a hill climbing method which minimizes least squares between time domain components and sources.

The result after clustering the detected components is shown in Figure 2.8. As can be seen, the estimated sources are quite similar to the originals.

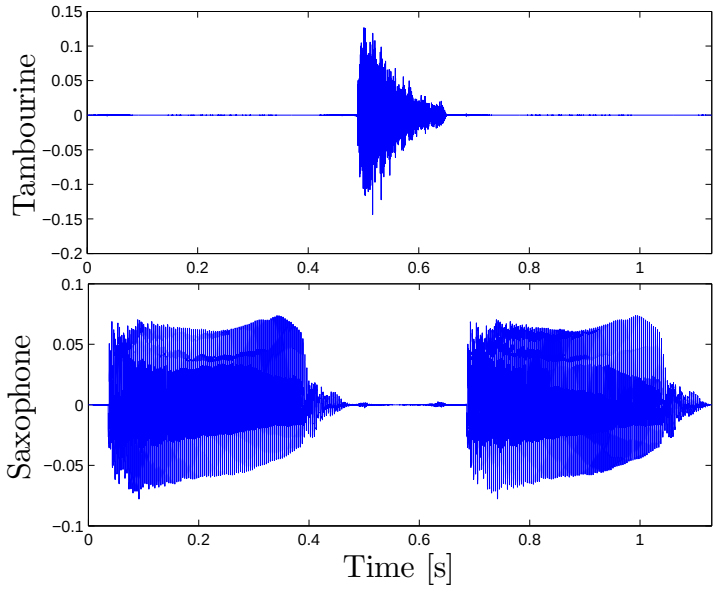


Figure 2.8: Estimated sources.

### 2.1.5 Separation quality

The Signal to Distortion Ratio (SDR) is used in order to estimate the source separation quality between the estimated $\tilde{s}_m(n)$ and the original sources $s_m(n)$. The SDR is calculated for each source $m$ as shown in Formula 2.12.

$$\mathrm{SDR}_m = 10 \log \frac{\sum_n s_m^2(n)}{\sum_n (s_m(n) - \tilde{s}_m(n))^2} \tag{2.12}$$

In order to continue with the toy example, the SDR of the tambourine estimated source is 35.60 dB and the SDR of the saxophone estimated source is 44.76 dB. These values give an idea of the accuracy of the NMF algorithm.

Finally, the total quality of NMF based algorithm is calculated as the mean SDR value of each estimated source. Mathematically, the SDR is represented by Equation 2.13.

$$\mathrm{SDR} = \frac{1}{M}\sum_{m=1}^{M}\mathrm{SDR}_m \tag{2.13}$$

Finally, the SDR of the exemplary mixture is 40.18 dB which is the mean of the obtained values of each estimated source. This value is very high because of this is a simple example.

## 2.2 Online Non-negative Matrix Factorization based algorithms

Two algorithms based on NMF are presented in this Section. That is the basic Online NMF (ONMF) and another Online NMF algorithm (CPONMF) which requires the current and the past frame to estimate basis vectors. These Online algorithms were presented in [5].

### 2.2.1 Basic Online Non-negative Matrix Factorization (ONMF)

The Online NMF algorithm was proposed in order to avoid some iterative NMF problems like high latency and high memory required for long time audio mixtures. The main difference between this algorithm and the NMF algorithm, described in Section 2.1.3, is that iterative NMF uses the full spectrogram for the multiplicative update rules (Eq.2.9), while Online NMF algorithm requires just one frame from the spectrogram. The ONMF algorithm is explained step by step in this Section.

**Initialization**

First of all, it is important to define the $\boldsymbol{V}$ matrix which represents the current frame $t_c$ from spectrogram matrix $\boldsymbol{X}$ as

$$\boldsymbol{V} = \boldsymbol{X}(:, t_c). \tag{2.14}$$

The objective of this algorithm is the estimation of the current frame in each iteration with the product between the basis matrix and the gain vector at the current time

$$\boldsymbol{V}_{K\times 1} \approx \tilde{\boldsymbol{V}}_{K\times 1} = \boldsymbol{B}_{K\times I}\boldsymbol{G}_{I\times 1}(t_c) \tag{2.15}$$

where, subindex 1 refers to time span (in this case just one frame) and $\boldsymbol{G}_{I\times 1}(t_c)$ is the gain vector at the current time $t_c$.

The first step of ONMF algorithm, after fixing the matrix dimensions, is to define initial values to $\boldsymbol{B}$ and $\boldsymbol{G}$ matrices as

$$\boldsymbol{B}_{K\times 1} = \boldsymbol{V}_{K\times 1} \tag{2.16a}$$

$$\boldsymbol{G}_{1\times 1} = \mathbf{1} \tag{2.16b}$$

The basis matrix is initialized with the first frame of the spectrogram and all entries of the gain matrix is set to one. It should be pointed out that in the initialization step the number of components is one, due to the fact that Equation 2.15 must be satisfied.

**Temporal gain and cost calculation**

The ONMF algorithm increments the current time index $t_c$ in the second step and then takes the next frame from the spectrogram (Eq.2.17a), in order to obtain the temporal gain which is calculated as

$$\boldsymbol{V} = \boldsymbol{X}(:, t_c) \tag{2.17a}$$

$$\boldsymbol{G}_{\text{tmp}} = \boldsymbol{G}(:, t_c - 1) \tag{2.17b}$$

$$\boldsymbol{G}_{\text{tmp}} \longleftarrow \boldsymbol{G}_{\text{tmp}}. \times \frac{\boldsymbol{B}^T \frac{\boldsymbol{V}}{\boldsymbol{B}\boldsymbol{G}_{\text{tmp}}}}{\boldsymbol{B}^T\mathbf{1}} \tag{2.17c}$$

The gain calculation requires the gain from the last processed frame as is shown in Equation 2.17b, then it iterates 100 times ($N_{\text{iter}} = 100$) over the update rule given by Equation 2.17c. It should be pointed out that the update rule used in the ONMF algorithm is the same that the update rule of the gain matrix applied in NMF (Eq. 2.9a) but for one frame instead of the full spectrogram. Afterwards, the current frame is estimated as:

$$\tilde{\boldsymbol{V}} = \boldsymbol{B}\boldsymbol{G}_{\text{tmp}} \tag{2.18}$$

As shown in Equation 2.18, the estimated frame is the product between the basis matrix and the calculated temporal gain. At this point, the cost function between $\boldsymbol{V}$ and $\tilde{\boldsymbol{V}}$ is calculated with the Kullback-Leibler divergence as shown in Equation 2.19.

$$d_{\text{div}}(\boldsymbol{V}, \tilde{\boldsymbol{V}}) = \sum_k \boldsymbol{V}(k) \log \frac{\boldsymbol{V}(k)}{\tilde{\boldsymbol{V}}(k)} - \boldsymbol{V}(k) + \tilde{\boldsymbol{V}}(k)] \tag{2.19}$$

Nevertheless, the value obtained in Equation 2.19 is not an absolute term. Thus, a normalization is done in order to be able to compare the divergence with a prefixed threshold $\alpha_{\text{ONMF}}$:

$$d_{\text{rel}}(\boldsymbol{V}, \tilde{\boldsymbol{V}}) = \frac{d_{\text{div}}(\boldsymbol{V}, \tilde{\boldsymbol{V}})}{d_{\text{div}}(\boldsymbol{V}, \boldsymbol{0})} \tag{2.20}$$

**Cost comparison**

If the relative cost function $d_{\text{rel}}$ is below a threshold $\alpha_{\text{ONMF}}$, the algorithm stores the temporal gain calculated in the gain matrix $\boldsymbol{G}$ because the estimated current frame is a good approximation of the original and the $\boldsymbol{B}$ matrix is able to represent it.

If the relative cost function $d_{\text{rel}}$ is above a threshold $\alpha_{\text{ONMF}}$, the $\boldsymbol{B}$ matrix is not able to represent the current frame anymore. Thus, a new basis vector $\boldsymbol{B}_{\text{new}}$ is calculated as the difference between the estimated and the current frame in order to represent the unknown information of the current frame as described in Equation 2.21. That is, the current frame and the estimated frame are quite different. Therefore, the algorithm has detected new component in the audio mixture.

$$\boldsymbol{B}_{\text{new}} = (\boldsymbol{V} - \tilde{\boldsymbol{V}})p \tag{2.21}$$

The operation $()p$ replaces all negative values with zeros in order to keep working with non-negative matrices. At the same time the $\boldsymbol{G}$ matrix has to be updated with the temporal gain and the number of components $I$ has to be incremented as shown in Equations 2.22.

$$\boldsymbol{G}(:, t_c) = \boldsymbol{G}_{\text{tmp}} \tag{2.22a}$$

$$I = I + 1 \tag{2.22b}$$

Due to the new detected component, a new gain vector is stored in the $\boldsymbol{G}$ matrix as shown in Equations 2.23. It should be pointed out that the new basis vector is able to represent just the current frame but not all previous, this is why the new gain vector has all its previous values as zero and the final value as one ($\boldsymbol{G}_{\text{new}} = 1$).

$$\boldsymbol{G}(I, 1 : t_c - 1) = 0 \tag{2.23a}$$

$$\boldsymbol{G}(I, t_c) = \boldsymbol{G}_{\text{new}} \tag{2.23b}$$

Once the basis vector $\boldsymbol{B}_{\text{new}}$ is calculated and the number of components $I$ is incremented, the $\boldsymbol{B}$ matrix is updated as presented in Equation 2.24 in order to include the new detected component.

$$\boldsymbol{B}(:, I) = \boldsymbol{B}_{\text{new}} \tag{2.24}$$

After the cost comparison step, the algorithm checks if the current frame is the last frame from the spectrogram. If not, the algorithm comes back to the temporal gain and cost calculation step.

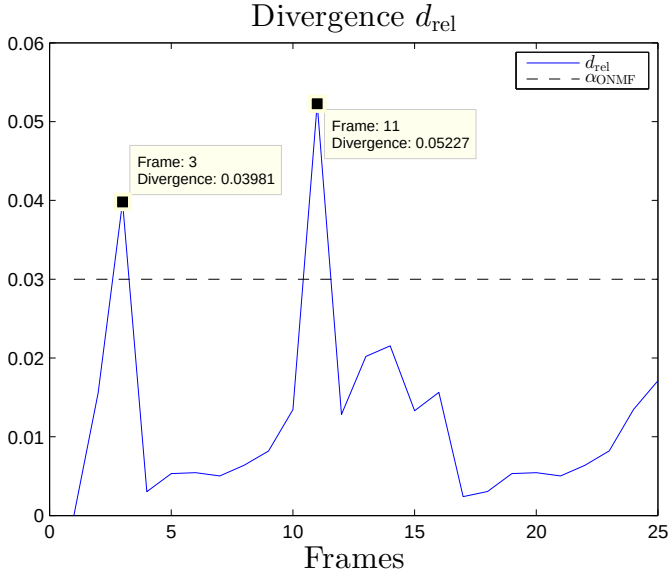


Figure 2.9: Cost function.

As an example, the divergence calculated for each frame of the toy example is shown in Figure 2.9. Therefore, in the cost function results provided by the ONMF algorithm it can be seen that the relative divergence $d_{\text{rel}}$ is above the threshold $\alpha_{\text{ONMF}}$ two times. Thus two components were detected, at the third and the eleventh frame respectively. After each detected component the cost function drops below the threshold again. It should be pointed out that at the end of the algorithm the number of detected components is three due to the component learned in the initialization step.

The Block diagram shown in Figure A.2 is a graphical representation of the algorithm described in this Section. The ONMF pseudo code is presented in Algorithm 2.2 as a summary.

**Algorithm 2.2** Online Non-negative Matrix Factorization Algorithm.

**Require:** Iteration number: $N_{\text{iter}}$, spectrogram: $\boldsymbol{X}_{K\times T}$, Threshold: $\alpha_{\text{ONMF}}$

1: Initialize $I = 1$, $t = 1$, $\boldsymbol{B}$ and $\boldsymbol{G}$ (Eq. 2.16)
2: **for** $t = 2$ to $T$ **do**
3: Update current frame (Eq. 2.17a)
4: Initialize temporal gain as previously calculated gain (Eq. 2.17b)
5: Update temporal gain $N_{\text{iter}}$ times (Eq. 2.17c)
6: Calculate relative cost $d_{\text{rel}}(\boldsymbol{V}, \tilde{\boldsymbol{V}})$ (Eqs. 2.19- 2.20)
7: **if** $d_{\text{rel}} < \alpha_{\text{ONMF}}$ **then**
8: Store calculated temporal gain in $\boldsymbol{G}$
9: **else**
10: Increment $I$ and update $\boldsymbol{B}$ and $\boldsymbol{G}$ matrices (Eqs. 2.21- 2.24)
11: **end if**
12: **end for**
13: **return** $\boldsymbol{G}$ and $\boldsymbol{B}$

### 2.2.2 Current and past ONMF (CPONMF)

A musical note usually requires more than one frame to be developed, for example, transients and sustained parts in an harmonic note. This is the principle of this algorithm.

The ONMF algorithm which takes the current and the past frame to estimate a basis vector, called CPONMF in this thesis for simplicity, is the second algorithm taken from [5].

The CPONMF algorithm is initialized with the same features than ONMF. Equations 2.16.

Afterwards, the CPONMF algorithm takes a new frame in order to calculate the temporal gain $\boldsymbol{G}_{\text{tmp}}$ and calculates the relative divergence $d_{\text{rel}}$ in the same manner as shown in Section 2.2.1, Equations 2.17 to 2.20.

The main difference between ONMF and CPONMF is the cost comparison. If the relative cost function $d_{\text{rel}}$ is below a threshold $\alpha_{\text{CPONMF}}$, the algorithm updates only the $\boldsymbol{G}$ matrix with the temporal gain because the $\boldsymbol{B}$ matrix is able to represent the current frame.

If the relative cost function $d_{\text{rel}}$ is above $\alpha_{\text{CPONMF}}$, the $\boldsymbol{B}$ matrix is not able to represent the current frame anymore. Therefore, the CPONMF algorithm requires the current and the past frame, instead of just the current one as in ONMF, in order to calculate a new basis vector. For this variant a new matrix with unknown information about the current and the past frame is established as $\boldsymbol{U}_{K\times 2}$. The $\boldsymbol{U}$ matrix is calculated as:

$$\boldsymbol{U}(:,1) = (\boldsymbol{X}(:,t_c-1) - \boldsymbol{B}\boldsymbol{G}(:,t_c-1))_p \tag{2.25a}$$

$$\boldsymbol{U}(:,2) = (\boldsymbol{X}(:,t_c) - \boldsymbol{B}\boldsymbol{G}_{\text{tmp}})_p \tag{2.25b}$$

The $\boldsymbol{U}$ matrix describes the differences between the past frame and the estimated past frame in column one and the differences between the current frame and the estimated current frame in column two in order to store all the unknown information of these two frames.

After the definition of $\boldsymbol{U}$, a new problem is presented. The algorithm has to represent $\boldsymbol{U}$ with one new component. Therefore, two new matrices are defined as $\boldsymbol{B}_{\text{new}}$ and $\boldsymbol{G}_{\text{new}}$, which dimensions are $K \times 1$ and $1 \times 2$ respectively. It should be pointed out that both matrices are initialized with random values. After the $\boldsymbol{B}_{\text{new}}$ and the $\boldsymbol{G}_{\text{new}}$ matrices initialization, they are iteratively updated one hundred times ($N_{\text{iter}} = 100$) with the multiplicative rules.

$$\boldsymbol{G}_{\text{new}} \longleftarrow \boldsymbol{G}_{\text{new}}. \times \frac{\boldsymbol{B}_{\text{new}}^T \frac{\boldsymbol{U}}{\boldsymbol{B}_{\text{new}}\boldsymbol{G}_{\text{new}}}}{\boldsymbol{B}_{\text{new}}^T \boldsymbol{1}} \tag{2.26a}$$

$$\boldsymbol{B}_{\text{new}} \longleftarrow \boldsymbol{B}_{\text{new}}. \times \frac{\frac{\boldsymbol{U}}{\boldsymbol{B}_{\text{new}}\boldsymbol{G}_{\text{new}}} \boldsymbol{G}_{\text{new}}^T}{\boldsymbol{1}\boldsymbol{G}_{\text{new}}^T} \tag{2.26b}$$

Iterative multiplication rules presented in Equations 2.26 are quite similar than proposed in Equations 2.9. Thus, this sub-step can be reduced to apply the NMF algorithm to $\boldsymbol{U}$ matrix with one component, that is with $I = 1$. Once $\boldsymbol{B}_{\text{new}}$ and $\boldsymbol{G}_{\text{new}}$ matrices are obtained, $\boldsymbol{B}$ and $\boldsymbol{G}$ matrices are updated as presented in Equations 2.22 to 2.24.

The block diagram shown in Figure A.3 is a graphical representation of the algorithm described in this Section. The CPONMF pseudo code is presented in Algorithm 2.3, this algorithm is quite similar than Algorithm 2.2 but it changes from line nine.

**Algorithm 2.3** ONMF algorithm using the current and the past frame for basis vector estimation

**Require:** Iteration number: $N_{\text{iter}}$, spectrogram: $\boldsymbol{X}_{K \times T}$, Threshold: $\alpha_{\text{CPONMF}}$

1: Initialize $I = 1$, $t = 1$, $\boldsymbol{B}$ and $\boldsymbol{G}$ (Eq. 2.16)
2: **for** $t = 2$ to $T$ **do**
3: Update current frame (Eq. 2.17a)
4: Initialize temporal gain as previously calculated gain (Eq. 2.17b)
5: Update temporal gain $N_{\text{iter}}$ times (Eq. 2.17c)
6: Calculate relative cost $d_{\text{rel}}(\boldsymbol{V}, \tilde{\boldsymbol{V}})$ (Eqs. 2.19- 2.20)
7: **if** $d_{\text{rel}} < \alpha_{\text{CPONMF}}$ **then**
8: Store calculated temporal gain in $\boldsymbol{G}$
9: **else**
10: Calculate $\boldsymbol{U}$ matrix (Eqs. 2.25)
11: Initialize $\boldsymbol{B}_{\text{new}}$ and $\boldsymbol{G}_{\text{new}}$ with random positive values
12: Update $\boldsymbol{B}_{\text{new}}$ and $\boldsymbol{G}_{\text{new}}$ $N_{\text{iter}}$ times (Eqs. 2.26)
13: Increment $I$ and update $\boldsymbol{B}$ and $\boldsymbol{G}$ matrices (Eqs. 2.21- 2.24)
14: **end if**
15: **end for**
16: **return** $\boldsymbol{G}$ and $\boldsymbol{B}$

Chapter 3

# Look-ahead Online Non-negative Matrix Factorization

The main advantage of the iterative NMF algorithm is the accuracy, supported by the fact that is the best algorithm in terms of separation quality as can be seen in Chapter 4. Nevertheless, it has some disadvantages on account of its computational requirements and the fixed number of components. The number of components $I$ is not usually known beforehand, which is a problem in most scenarios, and the full spectrogram introduces high latency and demands lager memory for long time mixtures. In order to overcome these disadvantages, Online NMF algorithms like ONMF and CPONMF were proposed [5], which principal advantage is the convergence speed and low memory consumption due to the use of only one or two frames for basis vector estimation. However, the basis vector estimation is not that accurate as NMF because usually more than two frames are required for a robust estimation.

A new variant of the Online NMF algorithm is presented in this Chapter which is called Look-ahead ONMF (LONMF). LONMF is focused on finding the required number of frames in order to estimate a new component. This procedure is explained in Section 3.1. In Section 3.2, the separation quality of the four mixtures studied by [5] is compared between NMF and the online algorithms.

## 3.1 LONMF Procedure

As explained in Section 2.2, ONMF algorithms are focused on studying the behavior of the divergence of each estimated frame with a threshold $\alpha_{\mathrm{ONMF}}$ as reference value. If the divergence $d_{\mathrm{rel}}$ is above the threshold $\alpha_{\mathrm{ONMF}}$, a new basis vector has to be obtained as the basis matrix $\boldsymbol{B}$ is not able to represent the current frame anymore. Therefore, ONMF algorithms obtain a basis vector based on the current frame (ONMF) or on the current and the past frame (CPONMF). $\alpha_{\mathrm{CPONMF}}$ is the same to $\alpha_{\mathrm{ONMF}}$.

Basically, the LONMF algorithm estimates basis vectors with support of more than two frames if necessary. The basic procedure of LONMF can be motivated with Figure 3.1: If the cost function $d_{\mathrm{rel}}$ is above the threshold $\alpha_{\mathrm{LONMF}}$, which means that the new acoustical event cannot be represented with current NMF model, the LONMF algorithm appends the

next frame to a buffer $\boldsymbol{V}_{\text{buff}}$. This is done until the cost function drops below $\alpha_{\text{LONMF}}$ which means that the acoustical event is over.

Afterwards, the new component is estimated by the NMF algorithm on $\boldsymbol{V}_{\text{buff}}$ with $N_{\text{iter}}$ iterations. Therefore, it obtains the basis vectors which belongs to the buffer. Thus it can detect more than one component at the same time.

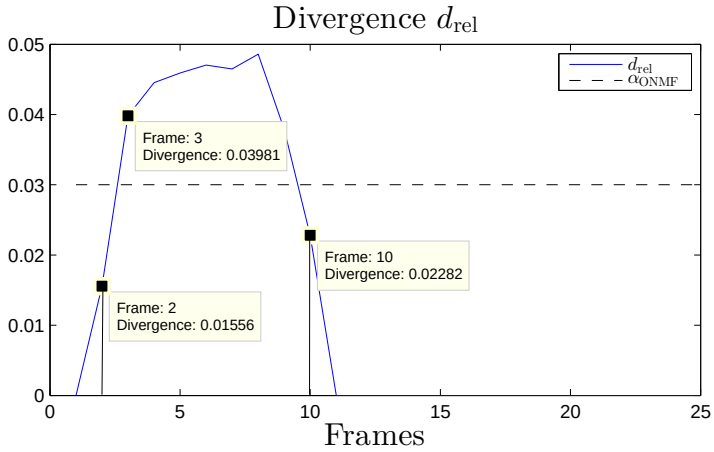


Figure 3.1: Divergence of the first ten frames of the toy example with $t_0 = 2$ and $t_1 = 10$.

The explanation of the algorithm is done in detail in this Section and with the same audio mixture used in Section 2.1 as example, that is an audio mixture with a percussive note and an harmonic note. The input spectrogram used in this example is shown in Figure 3.2 which axis of abscissas is in frames as reference for further Figures presented in this Section.

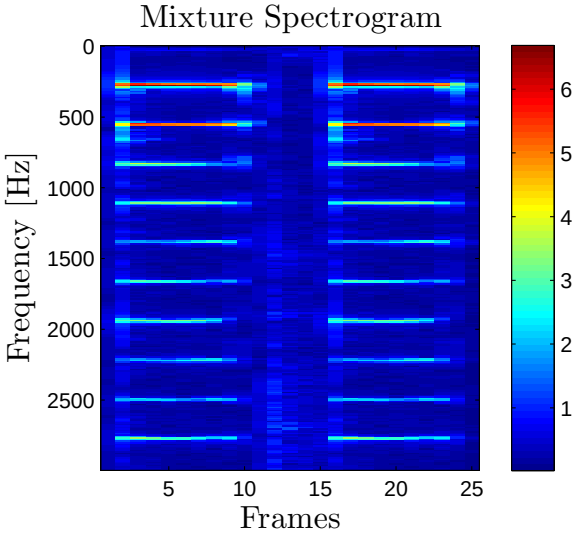


Figure 3.2: Toy example mixture spectrogram.

The LONMF algorithm is initialized with the same matrices as ONMF and CPONMF as shown in Equations 2.16. Afterwards, LONMF takes a new frame in order to calculate the temporal gain $\boldsymbol{G}_{\text{tmp}}$ and it calculates the relative divergence $d_{\text{rel}}$ in the same manner as shown in Equations 2.17 to 2.20 of Section 2.2.1.

If the relative cost function $d_{\text{rel}}$ is below $\alpha_{\text{LONMF}}$, $\boldsymbol{G}$ is updated with the temporal gain calculated with the help of the current frame due to the fact that matrix $\boldsymbol{B}$ is able to represent the current frame perfectly. Otherwise, if the relative cost function $d_{\text{rel}}$ is above a threshold $\alpha_{\text{LONMF}}$, $\boldsymbol{B}$ is not able to represent the current frame anymore. Therefore LONMF needs to update $\boldsymbol{B}$. This procedure is explained as follows.

**Look-ahead basis matrix update**

The aim of this method is to detect the number of frames which belongs to the new component and to update $\boldsymbol{B}$. This is done by storing frames in a buffer $\boldsymbol{V}_{\text{buff}}$. The maximum buffer length is defined by the user with the parameter $T_{\max}$. A new loop is required in order to examine more than one frame. This is why the current time frame index is saved in a new variable called $\tau$ as shown in Equation 3.1a. The first frame stored in the buffer is the previous frame to the current time $t_c$, as presented in Equation 3.1b.

$$\tau = t_c \tag{3.1a}$$

$$t_0 = t_c - 1 \tag{3.1b}$$

Subsequently, the time index $\tau$ is incremented, a new frame is copied out of the spectrogram $\boldsymbol{X}$ (3.2a) and the temporal gain $\boldsymbol{G}_\tau$ is updated by iterating $N_{\text{iter}}$ times as shown in Equation 3.2c. Afterwards the frame $\tilde{\boldsymbol{V}}_\tau$ is estimated as the product between $\boldsymbol{B}$ and $\boldsymbol{G}_\tau$. Equations 3.2 summarize this part of the algorithm.

$$\boldsymbol{V}_\tau = \boldsymbol{X}(:, \tau) \tag{3.2a}$$

$$\boldsymbol{G}_\tau = \boldsymbol{G}(:, \tau - 1) \tag{3.2b}$$

$$\boldsymbol{G}_\tau \longleftarrow \boldsymbol{G}_\tau . \times \frac{\boldsymbol{B}^T \frac{\boldsymbol{V}_\tau}{\boldsymbol{B}\boldsymbol{G}_\tau}}{\boldsymbol{B}^T \mathbf{1}} \tag{3.2c}$$

$$\tilde{\boldsymbol{V}}_\tau = \boldsymbol{B}\boldsymbol{G}_\tau \tag{3.2d}$$

Once the current frame is estimated, the cost relative $d_{\text{rel}}(\boldsymbol{V}_\tau, \tilde{\boldsymbol{V}}_\tau)$ is calculated as exposed in Equations 2.19 and 2.20. The relative cost of the current frame $d_{\text{rel}}(\boldsymbol{V}_\tau, \tilde{\boldsymbol{V}}_\tau)$ is compared again against the threshold. If the relative cost is above the threshold $\alpha_{\text{LONMF}}$, the current frame is appended to the buffer. Subsequently, the current time $\tau$ is incremented again and the procedure described by Equations 3.2 is repeated until the last frame is reached, the number of frames of the buffer is equal to $T_{\max}$ or the cost function is below $\alpha_{\text{LONMF}}$. If one of the last conditions becomes true, the last frame processed is the second index of the buffer, that is $t_1$. As an example, the relative cost function of the first ten frames of the toy example is shown in Figure 3.1.

The LONMF algorithm is able to learn more than one component at the same time, this is why a function which discards corrupted basis vectors is required. The basis vectors which are not able to represent the buffer must be rejected.

$$\boldsymbol{V}_{\text{buff}} = \boldsymbol{X}(:, t_0 : t_1) \tag{3.3}$$

The buffer $\boldsymbol{V}_{\text{buff}}$ has been defined with the frames between the indexes $t_0$ and $t_1$ from the spectrogram as shown in Equation 3.3. Nevertheless, there are some scenarios where the buffer has to be redefined in order to achieve a better basis vector estimation. Usually, the first index $t_0$ does not change because it belongs to the first frame of the buffer which cannot be represented by the basis matrix.

There is one scenario where $t_0$ can be updated, that is when the first component defined in the initialization step is able to represent less than three frames of the spectrogram. This is due to the fact that, as was exposed before, one note usually requires more than two frames to be developed. Then the first component is discarded. This scenario is presented in this example, thus the first index is updated to one and the first component is discarded.

There is one scenario where the second index is updated to obtain a better basis vector estimation, that is when there are minimums in the cost function above $\alpha_{\text{LONMF}}$. In that case, the second index is updated to the maximum-minimum of the cost function, otherwise it does not change. A minimum in the cost function means that the acoustical event presented in the buffer is over. Nevertheless, it does not finish due to the fact that there is a new event present in the next frames or there is a past event which is larger in time. The maximum of all the minimums is taken in order to detect the maximum number of frames with the same buffer. Only the minimums between $\alpha_{\text{LONMF}}$, as lower limit, and two times $\alpha_{\text{LONMF}}$, as upper limit, are considered.

In this toy example there is only one minimum at the frame seven (see Figure 3.1). Therefore the second index is updated to that value and the buffer indexes are now one and seven respectively.

Once, the buffer is defined with $t_0$ and $t_1$, the number of components is incremented and a new vector with random values is appended to the basis matrix $\boldsymbol{B}$. In order to estimate the detected component, the NMF algorithm is applied to the buffer which estimates the corresponding $\boldsymbol{B}$ and $\boldsymbol{G}$ in $N_{\text{iter}}$ iterations. It should be pointed out that only the last random vector appended to the $\boldsymbol{B}$ matrix is updated due to the fact that the previous learned components are good enough to estimate the previous frames.

Afterwards, the algorithm comes back to the main loop and continues the execution until the last frame of the spectrogram is reached. There are some scenarios where the cost function is above the threshold again for frames which belong to the last buffer considered. In this case, the last components learned for this buffer are considered not good enough to represent the buffer and they are discarded. Therefore, the number of components is incremented again,

one basis vector with random values is appended to the $\boldsymbol{B}$ matrix and the NMF algorithm is applied again to the buffer. When this situation occurs, the NMF algorithm only updates the discarded and the last basis vector keeping the previous basis vectors unalterable as was exposed before. Figure A.5 shows the block diagram of the "Look-ahead basis matrix update" procedure as reference.

In Figure 3.3, the divergence and the number of detected components of the toy example is presented. In the toy example, the first index $t_0$ for the first estimated component is one and the first basis vector defined in the initialization step is considered corrupted because it is only able to represent one frame. Thus it is discarded. Therefore, the first time that the divergence is above the threshold, that is $t_c = 3$, the algorithm estimates two components at the same time, as can be seen in Figure 3.3. The first two detected components belong to the saxophone note and the third component belongs to the tambourine note. This can be seen in the graphic of the cost function in Figure 3.3 and it can be compared with the mixture signal. The same conclusion can be achieved by looking the spectrogram of Figure 3.2 due to the structure of the percussive and harmonic notes.

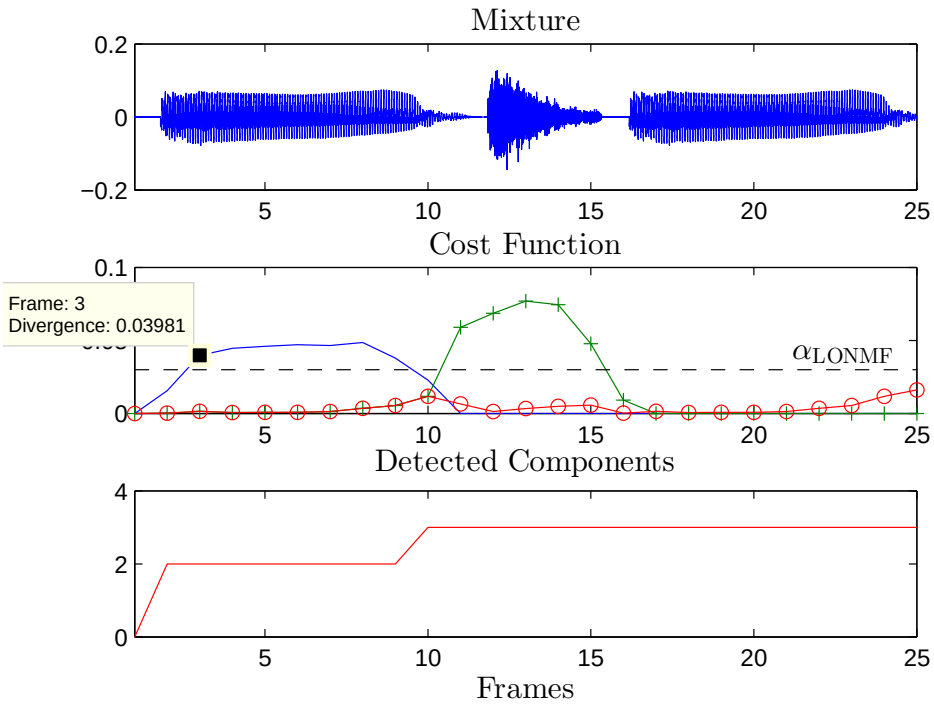


Figure 3.3: Cost function and detected components for the toy example.

Therefore, when the algorithm concludes its execution, the obtained matrices for the toy example, basis and gain, are shown in Figure 3.4. The basis and the gain matrices presented in Figure 3.4 are quite similar to the matrices obtained by the NMF algorithm shown in Figure 2.5. For the LONMF algorithm, the first component belongs to the on/off sets of the harmonic note, as can be distinguished in the gain matrix; the second estimated component belongs to the sustained harmonic note, as can be seen in the gain matrix; and the last estimated component belongs to the percussive note as can be seen in the basis matrix at

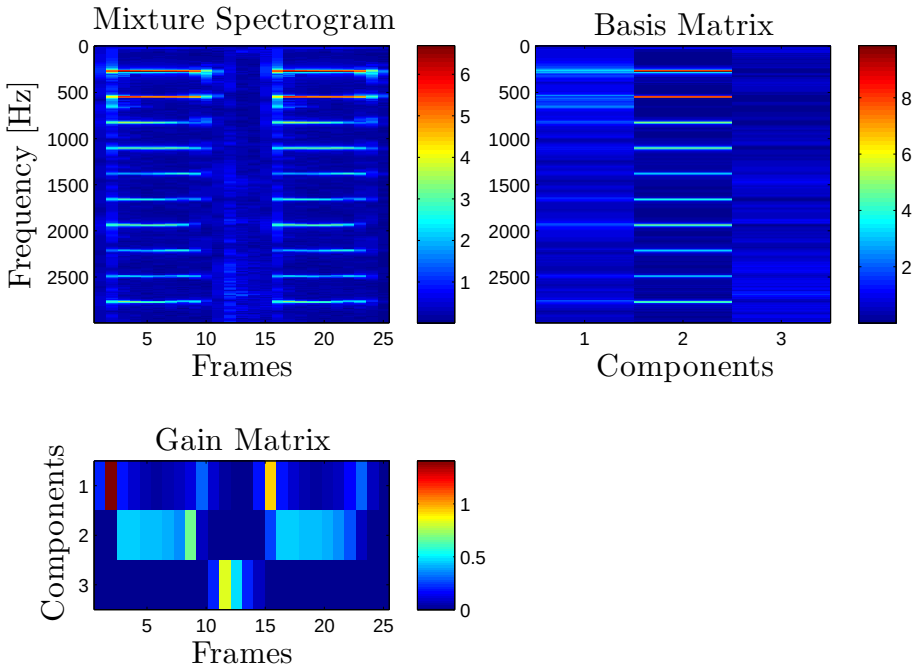


Figure 3.4: The basis and the gain matrices achieved for the toy example.

medium frequencies and in the gain matrix around the thirteenth frame.

This matches with was explained for Figure 3.3. These matrices are the input of the synthesis and clustering block and the spectrogram of each component is shown in Figure 3.5. Where it can be seen that it matches also with the last description due to the horizontal and vertical structure of each component.

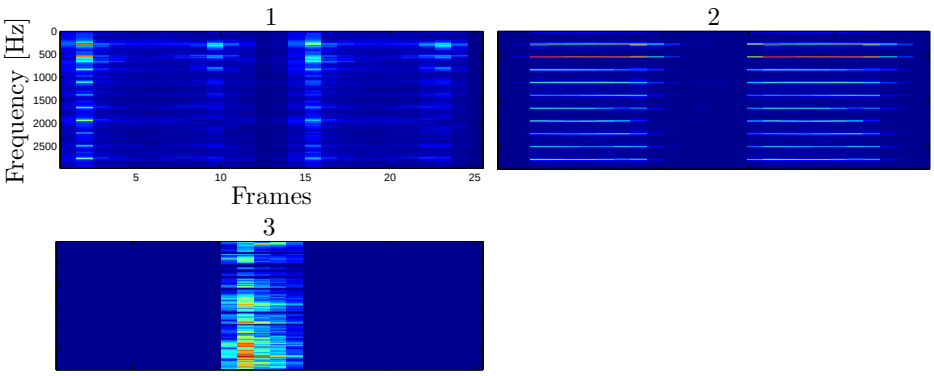


Figure 3.5: Spectrogram of the estimated components of the LONMF algorithm.

The estimated signals are obtained in the synthesis and clustering block. The ISTFT transforms the spectrogram of the estimated components (Figure 3.5) to the time domain as was exposed in Section 2.1.4. Then, the estimated components are classified into each source.

The SDR of each estimated source obtained by each algorithm is presented in Table 3.1. The SDR values are shown in decibels (dBs) and it can be seen that the estimated sources

| Algorithm | Tambourine Note | Saxophone Note | Full Mixture |
|---|---|---|---|
| **NMF** | 35.60 | 44.76 | 40.18 |
| **ONMF** | 14.30 | 23.45 | 18.87 |
| **CPONMF** | 14.47 | 23.63 | 19.05 |
| **LONMF** | 33.25 | 42.41 | 37.83 |

Table 3.1: Toy example SDR comparison

by the LONMF algorithm is quite accurate compared with the NMF algorithm.

Therefore, for this example, LONMF achieves 18.95 dB and 18.78 dB more for each note than ONMF and CPONMF respectively. Nevertheless, the NMF algorithm achieves 2.35 dB more for each note than LONMF.

As a summary the pseudo-code of the LONMF algorithm is presented in Algorithm 3.1. The first six lines are the same than Algorithm 2.2, afterwards the LONMF algorithm includes the "Look-ahead basis matrix update" procedure.

**Algorithm 3.1** Look-ahead Online Non-negative Matrix Factorization Algorithm.

**Require:** Iteration number: $N_{\text{iter}}$, spectrogram: $\boldsymbol{X}_{K\times T}$, Threshold: $\alpha_{\text{LONMF}}$, $T_{\max}$
1: Initialize $I = 1$, $t = 1$, $\boldsymbol{B}$ and $\boldsymbol{G}$ (Eq. 2.16)
2: **for** $t = 2$ to $T$ **do**
3: Calculate temporal gain $\boldsymbol{G}_{\text{tmp}}$ and relative cost $d_{\text{rel}}(\boldsymbol{V}, \tilde{\boldsymbol{V}})$ (Eqs. 2.17a- 2.20)
4: **if** $d_{\text{rel}} < \alpha_{\text{LONMF}}$ **then**
5: Store calculated temporal gain in $\boldsymbol{G}$
6: **else**
7: Save current time in $\tau$ and set $t_0$ (Eqs. 3.1)
8: **while** $\boldsymbol{B}$ matrix is not updated **do**
9: increment $\tau$ and calculate temporal gain $\boldsymbol{G}_\tau$ (Eqs. 3.2)
10: Calculate relative cost $d_{\text{rel}}(\boldsymbol{V}_\tau, \tilde{\boldsymbol{V}}_\tau)$ (Eqs. 2.19- 2.20)
11: **if** $d_{\text{rel}}(\boldsymbol{V}_\tau, \tilde{\boldsymbol{V}}_\tau) \geq \alpha_{\text{LONMF}}$ or $\tau < T_{\max}$ or $\tau < T$ **then**
12: Append frame to the buffer
13: **else**
14: Discard $\boldsymbol{B}$ vectors and set $t_1$
15: Call NMF algorithm with: $\boldsymbol{V}_{\text{buff}} = \boldsymbol{X}(:, t_0 : t_1)$, $\boldsymbol{B}$, $\boldsymbol{G}$ and $N_{\text{iter}}$
16: Update $I$ and $\boldsymbol{B}$ and $\boldsymbol{G}$ matrices
17: **end if**
18: **end while**
19: **end if**
20: **end for**
21: **return** $\boldsymbol{G}$ and $\boldsymbol{B}$

The LONMF algorithm can be divided in blocks as shown in the principal block diagram in Figure A.4 as reference.

### Improvements of Look-ahead basis matrix update

Two improvements of the LONMF algorithm were reached while LONMF was tested with the four audio mixtures considered by [5].

The first improvement was applied to the procedure which calculates the second index $t_1$ of the buffer $\boldsymbol{V}_{\text{buff}}$. This procedure calculates the maximum of all the minimums presented in the cost function of $\boldsymbol{V}_{\text{buff}}$. There are some scenarios where the cost function variates as noise. Therefore, the end of the acoustical event is not reached. This problem can be seen in Figure 3.6.

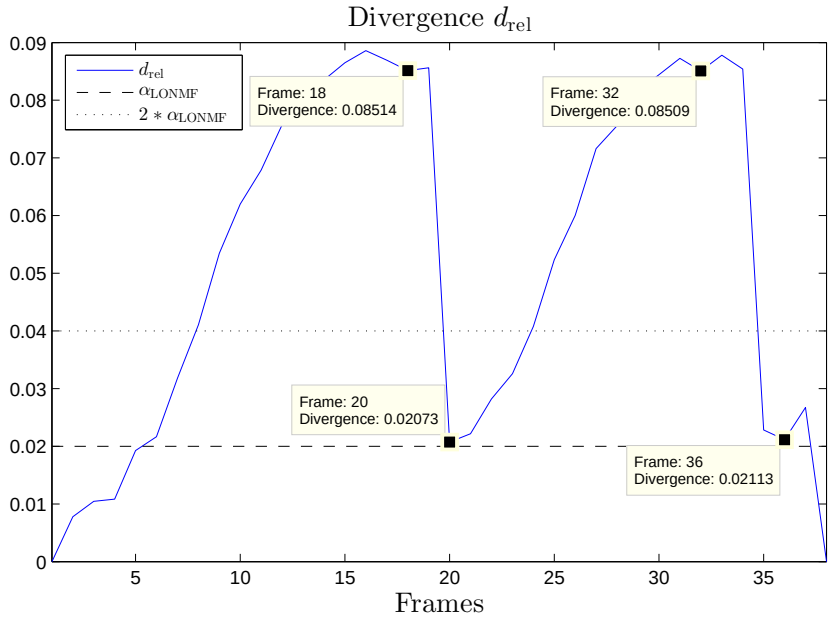


Figure 3.6: Maximum-minimum example.

There are four minimums in the cost function as can be seen in Figure 3.6. The frame which has the maximum minimum is frame 18. Nevertheless, minimums like that can introduce some errors in the basis estimation because usually this is not the end of the component. These minimums are provided by the variance of the divergence. Therefore, in order to avoid that type of minimums, an upper bound was established as the double of the threshold $\alpha_{\text{LONMF}}$.

The second improvement was introduced while one mixture was being separated. This mixture has a note with high amplitude and it is played while the other source is silent. In this type of scenarios, the divergence undergoes a great increase. This considerable change in the cost function can introduce some errors in the basis estimation. This error is because, usually, the previous frame belongs to the acoustical event and is not considered.

Therefore, if the change in the cost function between two frames is larger than $\alpha_{\text{LONMF}}$ a big increase is considered. In that case the first index $t_0$ of $\boldsymbol{V}_{\text{buff}}$ is decreased if possible. An example of this special case is shown in Figure 3.7.

In this example $t_0$ is updated from 54 to 53. This is done in order to consider the frame which is the beginning of the new acoustical event.

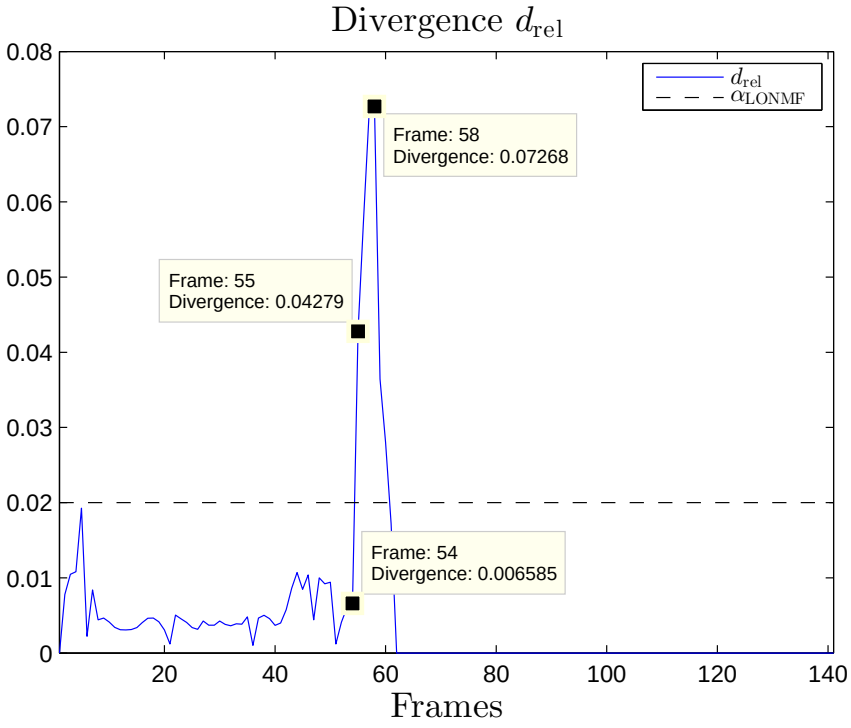


Figure 3.7: High increment of the divergence.

This type of acoustical events are considered as special cases because the difference between the estimated and the current frame is quite big. In addition, the learned components are not able to represent this buffer. Therefore, if there is considered one frame more for a new basis estimation, better separation quality can be achieved.

## 3.2 Comparison

Once the four NMF based algorithms were explained in detail, the comparison of the four audio mixtures considered in [5] is presented in this Section. The audio mixtures are combinations of two harmonic and two percussive sources of six seconds each. Therefore, the instruments played for every percussive signal are a tambourine and a drum respectively; for every harmonic signal are a trumpet and a saxophone respectively as is shown in the spectrograms of the sources, in Figure 3.8. The harmonic sources can be recognized due to their horizontal structure and the percussive sources due to their vertical structure.

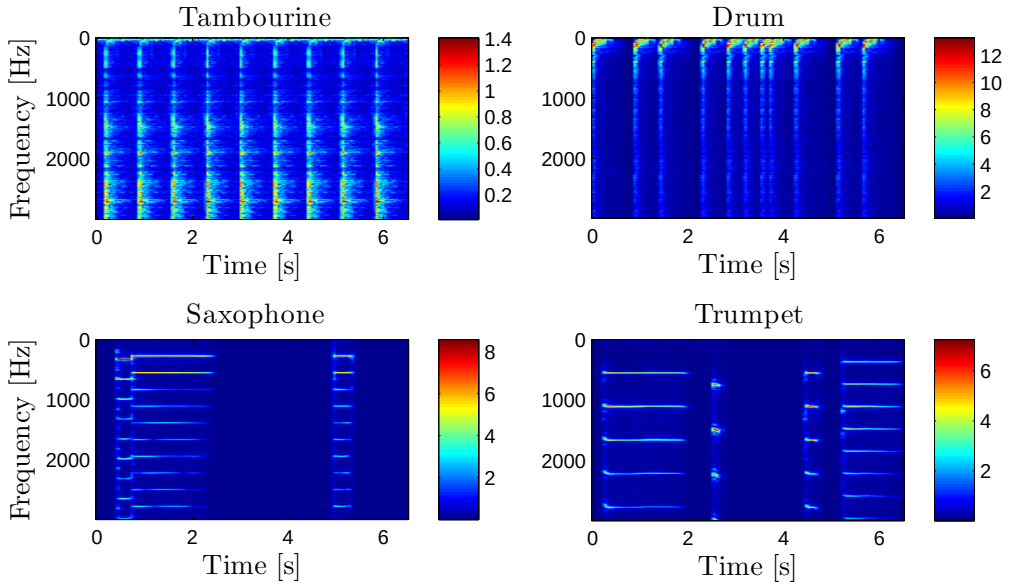


Figure 3.8: Spectrograms of source signals.

In order to facilitate the comparison, the axis of abscissas of the spectrograms are presented in time because of the rest of the Figures of this Section have the same axis.

The four mixtures considered by [5] are composed of one harmonic and one percussive source. That is a linear combination of every percussive note with each harmonic note. The input spectrogram of each mixture is shown in Figure 3.9.

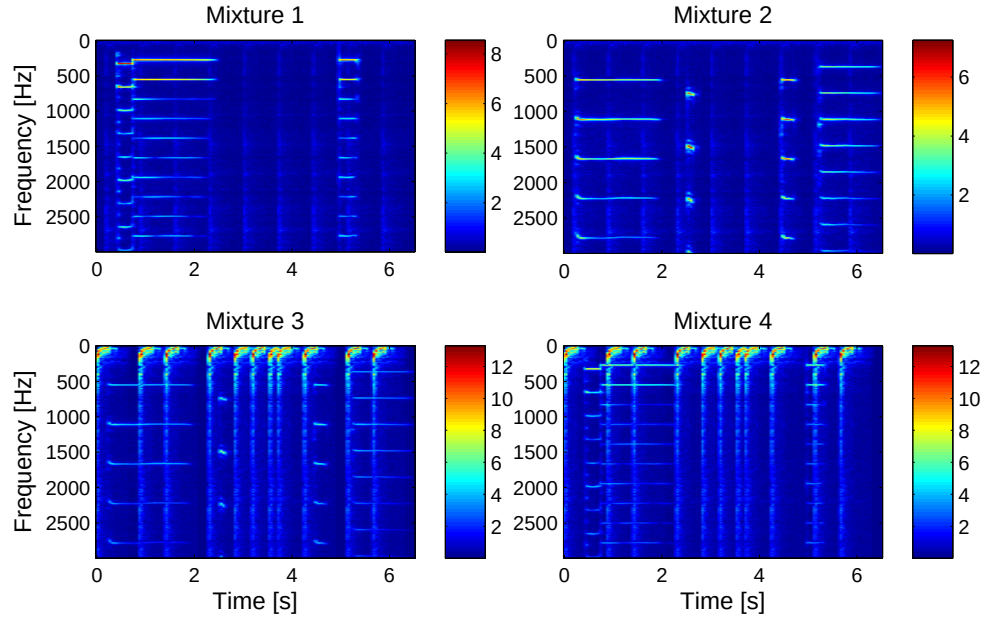


Figure 3.9: Spectrogram of each mixture.

It can be seen that in mixture one, the energy of the harmonic signal is larger than the percussive signal due to the fact that the intensity is higher in the horizontal structure of the spectrogram. The same fact is presented in the spectrogram of the mixture two. It also can be seen in Figure 3.8 because of the energy is more concentrated on the harmonic sources compared with the tambourine source. The duration over time of each note in the harmonic sources is larger, while the tambourine note appears more times in the mixture but its duration is shorter.

On the other hand, the energy of the percussive signal, the drum source, is larger than the harmonic signals presented in mixtures three and four. The intensity of the percussive signal is larger than the intensity of the harmonic sources. Additionally, the drum notes are replied with more frequency between two and four seconds, as can be seen in Figure 3.8. Therefore, the energy depends on the amplitude and the duration of each note presented in the signal, not in the type of source.

The mixtures are separated by all the algorithms explained in this thesis. The best SDR results are achieved for the mixture one and the worst SDR results are achieved for the mixture three. These mixtures are studied in detail in Sections 3.2.1 and 3.2.2 respectively.

### 3.2.1 Mixture with the best SDR values

All the NMF based algorithms achieve the best results for the mixture 1. The spectrogram of the estimated sources by each algorithm are shown in Figure 3.10 in order to analyze them.

If this Figure is compared with the spectrogram of the original sources (Figure 3.9), it can be seen that the estimated spectrograms are good enough. Thus, the source separation achieved by all the algorithms has high quality.

It also can be seen that the tambourine source estimated by ONMF has very little parts of the saxophone signal around the frames 10 and 110. This is due to the fact that the acoustical events presented in the harmonic source have begun at that frames and they are not fully developed. The estimated basis vectors for both frames did not take in account any previous or next frame and they cannot represent these frames correctly. This is why part of the energy of the harmonic source is presented in the percussive source.

| | **NMF** | **ONMF** | **CPONMF** | **LONMF** |
|---|---|---|---|---|
| **Tambourine** | 25.76 | 20.76 | 21.94 | 24.07 |
| **Saxophone** | 30.57 | 25.57 | 26.75 | 28.88 |

Table 3.2: SDR of each estimated source by the four algorithms. Mixture 1.

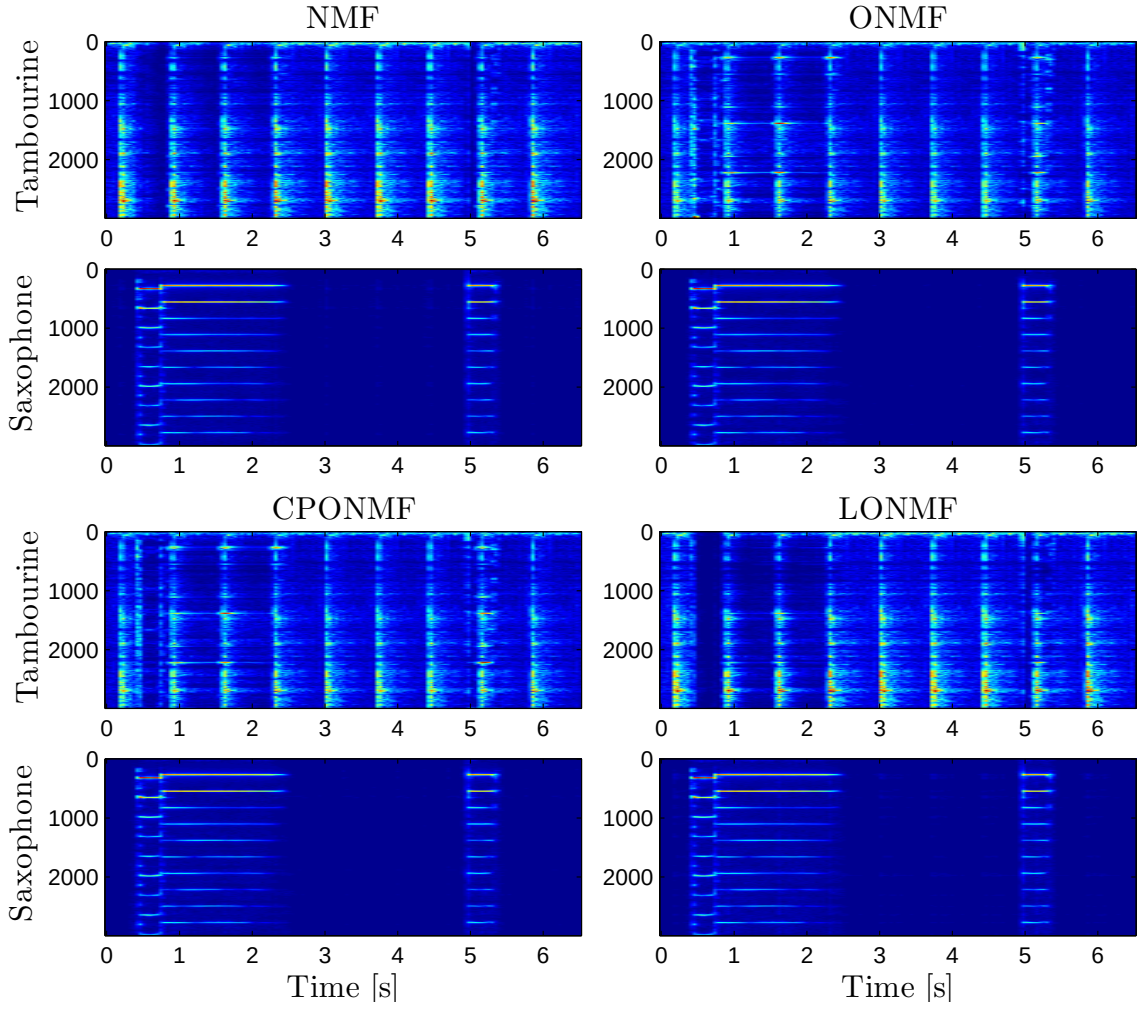


Figure 3.10: Estimated sources of mixture 1.

Nevertheless, as can be checked in Table 3.2, the estimated signals of the saxophone source have better results than the estimated signals of the tambourine. This is due to the higher energy level presented in the harmonic signal and the sustained harmonic notes which makes more easy their estimation. In other words, there are not too many changes in the cost function and the algorithms can distinguish easily which frame belongs to each component.

Looking at the SDR values of each algorithm, the LONMF algorithm has the best results for online source separation algorithms while the SDR values of the NMF algorithm cannot be achieved. However, the SDR values provided by LONMF are close to them. Therefore, this is a good example of a mixture that can be easily separated.

Looking at the SDR values of online NMF algorithms, LONMF has achieved 2.13 dB more than CPONMF and 3.31 dB more than ONMF.

### 3.2.2 Mixture with the worst SDR values

The spectrogram of each estimated source is shown in Figure 3.11. All of the estimated sources have parts of each other. Even the trumpet source estimated by NMF have some little parts of the drum source. Nevertheless, the drum source was really good estimated if it is compared with Figure 3.8. The ONMF algorithm has obtained the worst estimation of the sources, due to the fact that it does not consider past or future frames for basis vectors estimation as was explained in Section 3.2.1. CPONMF has estimated the sources better than ONMF but parts of the trumpet signal take part in the drum estimated signal. This is because both sources were presented in the same frame and they were synthesized in the same component. The LONMF algorithm has done a good estimation of the drum and the trumpet signal. However, few parts of the percussive signal are presented in the harmonic signal.

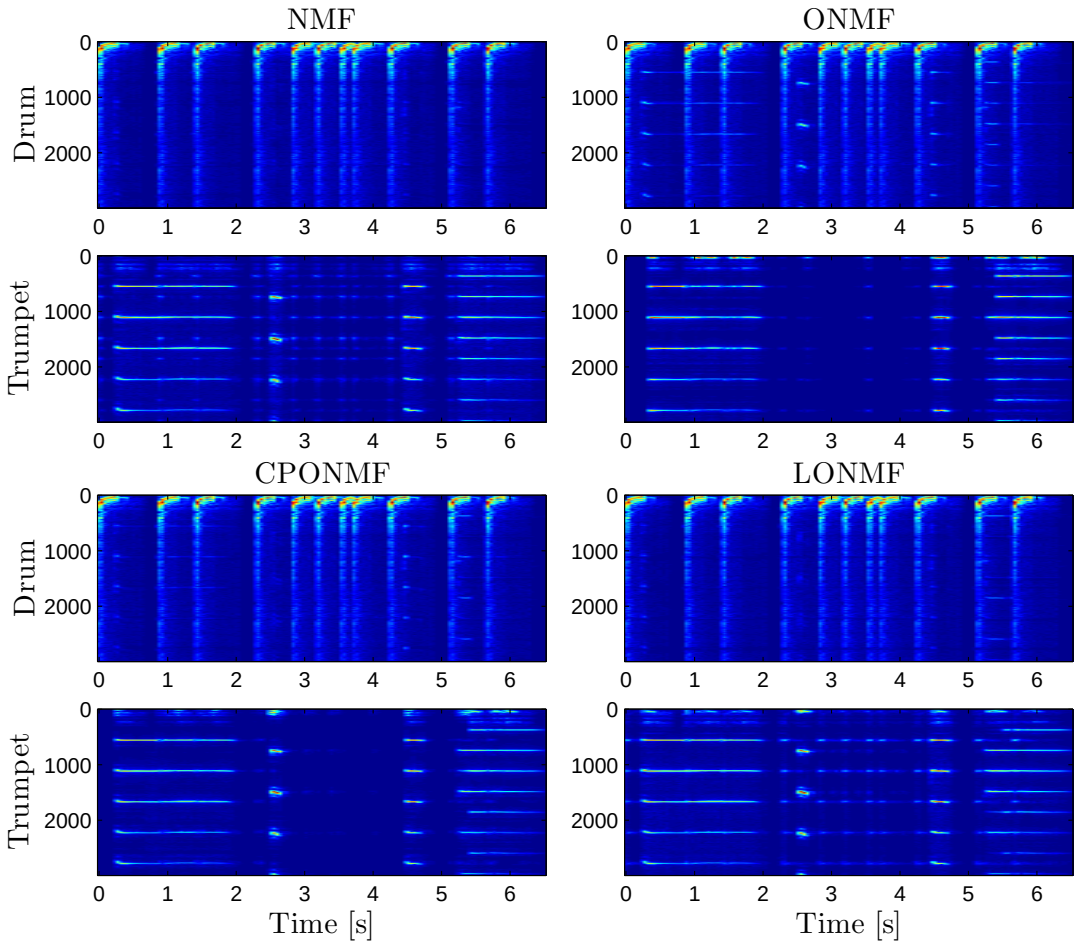


Figure 3.11: Estimated sources of mixture 3.

Table 3.3 summarizes the SDR values obtained by each algorithm for this mixture. The SDR values of the estimated harmonic signal are worse than the SDR values of the drum signal due to the higher energy of the percussive source. Nevertheless, in this mixture, the best SDR values are achieved by NMF and again the closest SDR values are achieved by the

LONMF algorithm, with almost 4.5 dB less for each signal. On the other hand, LONMF has achieved 1.29 dB more than CPONMF and 6.18 dB more than ONMF.

| | **NMF** | **ONMF** | **CPONMF** | **LONMF** |
|---|---|---|---|---|
| **Drum** | 28.45 | 18.12 | 23.01 | 24.30 |
| **Trumpet** | 12.86 | 2.53 | 7.42 | 8.71 |

Table 3.3: SDR of each estimated source by the four algorithms. Mixture 3.

### 3.2.3 Separation quality of each mixture

The total SDR values obtained by all the algorithms are shown in Table 3.4 for each mixture. As was discussed in previous Sections (3.2.1 - 3.2.2), the best algorithm in terms of separation quality is the NMF algorithm. Focused on the online algorithms, the algorithm which provides the best SDR values for all the mixtures is the LONMF algorithm with the exception of the SDR value of the fourth mixture which is 0.09 dB lower than the ONMF algorithm.

| **Mixture** | **NMF** | | **ONMF** | | **CPONMF** | | **LONMF** | |
|---|---|---|---|---|---|---|---|---|
| | **SDR** | **I** | **SDR** | **I** | **SDR** | **I** | **SDR** | **I** |
| **1** | 28.16 | 7 | 23.16 | 7 | 24.34 | 6 | 26.48 | 4 |
| **2** | 19.18 | 5 | 10.67 | 5 | 15.26 | 5 | 18.72 | 5 |
| **3** | 20.65 | 8 | 10.32 | 8 | 15.21 | 9 | 16.50 | 6 |
| **4** | 19.01 | 7 | 18.10 | 7 | 17.87 | 7 | 18.01 | 5 |

Table 3.4: Total SDR obtained by the four algorithms.

In this case, the LONMF algorithm has detected less components than the other online algorithms and better SDR values are achieved. The NMF algorithm was applied with the same number of detected components by the ONMF algorithm. If the NMF algorithm is applied to the same mixtures with the same number of detected components by the LONMF algorithm, the obtained SDR values are still better than LONMF but worse than the values shown in Table 3.4.

Therefore, it can be concluded that for these four mixtures the LONMF algorithm provides the best results in terms of separation quality for the online source separation algorithms presented in this thesis.

In terms of simulation time, the NMF algorithm takes around 9.6 seconds to separate each mixture, due to the number of iterations $N_{\text{iter}}$ which was set to 300. The online algorithms

take around 3.2 second to separate the same mixtures due to the fact that $N_{\text{iter}}$ was set to 100 for ONMF and CPONMF and it was set to 20 for LONMF. If $N_{\text{iter}}$ on the NMF algorithm is reduced to 20 as the LONMF algorithm, the SDR values obtained are worse than the obtained by LONMF for three of the four mixtures. The SDR value is just 0.4 dB better for the other mixture. Therefore, the online algorithms converge more quickly than the NMF algorithm. The computer used for this time measurement is a laptop equipped with a 2.5GHz quad-core Intel® Core™ i5-2450M processor, with 4GB of RAM and the OS is Windows® 7 premium. The simulation was done in MATLAB® version 8.2.0.701(R2013b).

Chapter 4

# Results

A test set composed of 257 mixtures is considered in this Chapter in order to evaluate the LONMF algorithm. The mixtures under test were taken from the Signal Separation Evaluation Campaign website [7]. The test set is composed of audio mixtures from ten songs. The sources are recordings from guitars, drums, basses, saxophones, harps, pianos, voices and more.

In this thesis, each audio mixture is composed of two sources from the same song. The time duration depends on the source. Thus two sources from different songs are not mixed. In the end, 257 mixtures from 70 different sources are studied. The number of sources and mixtures of each song is shown in Table 4.1.

| **Author - Song** | **Sources** | **Mixtures** | **Time** [s] |
|---|---|---|---|
| Bearlin - Roads | 10 | 45 | 14 |
| Tamy - Que Pena Tanto Faz | 2 | 1 | 13 |
| Another Dreamer - The Ones We Love | 3 | 3 | 25 |
| Fort Minor - Remember the Name | 6 | 15 | 24 |
| Ultimate NZ Tour | 5 | 10 | 18 |
| Glen Philips - The Spirit of Shackleton | 8 | 28 | 22 |
| Nine Inch Nails - The Good Soldier | 7 | 21 | 21 |
| Shannon Hurley - Sunrise | 8 | 28 | 23 |
| Jims Big Ego - Mix tape | 8 | 28 | 20 |
| Vieux Farka Touré - Ana | 13 | 78 | 30 |
| **Total** | **70** | **257** | **-** |

Table 4.1: Sources considered in the test set.

The separation quality obtained by the LONMF algorithm is compared with NMF, ONMF and CPONMF in this Chapter. Two quality measurements are used to compare the algorithms. The first one is the Signal-to-Distortion Ratio (SDR) described in Section 2.1.5. The second one is the segmental SDR which is the same that the segmental SNR used by [8].

There are some input parameters required for each algorithm. The number of components used for the NMF algorithm is a fixed number, $I = 20$, for each mixture due to the fact that they are not known beforehand. The number of iterations $N_{\text{iter}}$ is set to 300. The online algorithms proposed by [5] require more parameters like the number of iterations, the threshold and the seed which generates random numbers. For both algorithms, ONMF and CPONMF, $N_{\text{iter}}$ is set to 100, $\alpha$ is set to 0.02 and the seed is set to one as was presented in [5] as default parameters for all online NMF based algorithms. The parameters used by NMF, ONMF, CPONMF and LONMF for the STFT procedure are the same as described in Section 2.1.

The evaluation of the LONMF algorithm was done for different threshold values $\alpha_{\text{LONMF}} \in [0.00, 0.05]$ with a step of 0.001 in order to find the best range of threshold values. The threshold $\alpha_{\text{LONMF}}$ is the main parameter of LONMF because of it influences the separation quality and also the execution time. This evaluation was done for each mixture with ten different seeds to generate random values to initialize basis vectors as exposed in Chapter 3. The number of iterations $N_{\text{iter}}$ was set to 25 and the maximum number of frames in the buffer $T_{\max}$ was set to three seconds which is 65 frames.

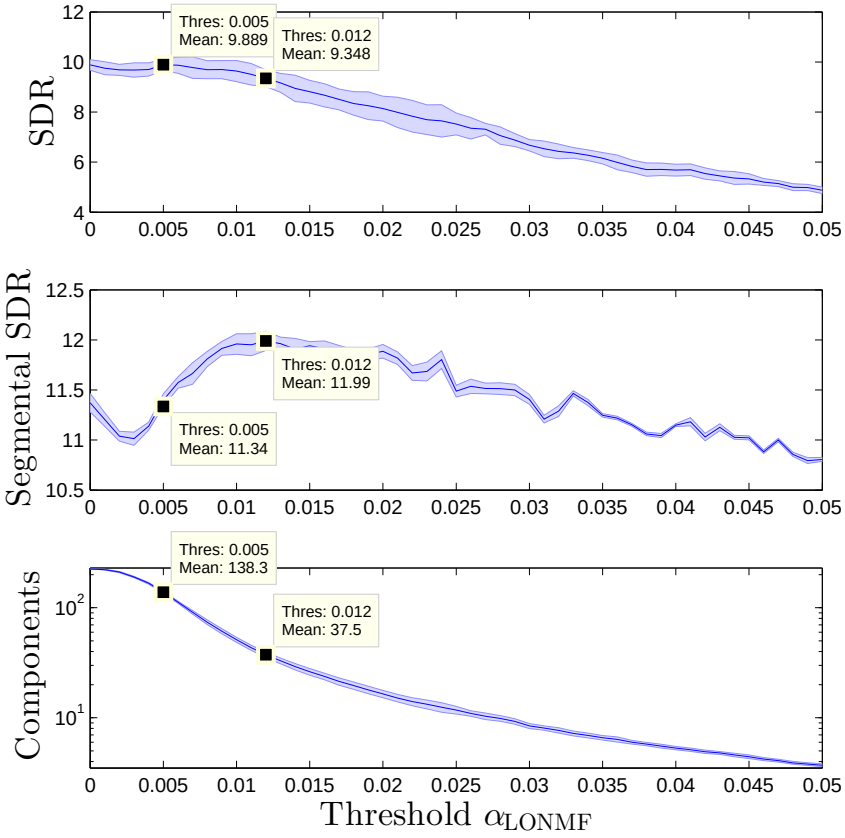


Figure 4.1: Mean Values obtained by LONMF. $N_{\text{iter}} = 25$, $T_{\max} = 3$ s.

The first simulation produced the main values of SDR, segmental SDR and detected components shown in Figure 4.1. The standard deviation of each mean value is also shown.

Therefore, the maximum SDR value achieved by LONMF is 9.889 dB. Nevertheless, this SDR value is obtained for $\alpha_{\text{LONMF}}$ set to 0.005. This low threshold value produce a high memory consumption due to the fact that LONMF has to obtain more components to represent each buffer (138.3 in mean). It should be pointed out that if $\alpha_{\text{LONMF}}$ is extremely low, LONMF has to use more basis vectors to represent the buffer. Then, LONMF requires more components to overcome $\alpha_{\text{LONMF}}$. The maximum segmental SDR value is 11.99 dB and is achieved for $\alpha_{\text{LONMF}}$ set to 0.012. With this threshold, LONMF achieves 37.5 components in mean which is almost one hundred components less than the number of components achieved for $\alpha_{\text{LONMF}}$ set to 0.05.

Note that for $\alpha_{\text{LONMF}}$ set to zero, LONMF has to estimate perfectly each buffer with the maximum number of frames allowed, that is the third part of the length $\boldsymbol{V}_{\text{buff}}$. For $\alpha_{\text{LONMF}}$ set to infinity, each component achieved is good enough to represent $\boldsymbol{V}_{\text{buff}}$. Therefore the separation quality is extremely low.

Therefore, the optimum threshold $\alpha_{\text{LONMF}}$ range for LONMF is from 0.015 to 0.020. If the threshold decreases, better separation quality is achieved. Nevertheless, the number of detected components is higher and the latency, the memory and the execution time are increased. Thus, there is a trade off in the choice of $\alpha_{\text{LONMF}}$ between the convergence speed and the separation quality. On the other hand, the number of detected components is not that significant for this threshold range compared with lower $\alpha_{\text{LONMF}}$ values.

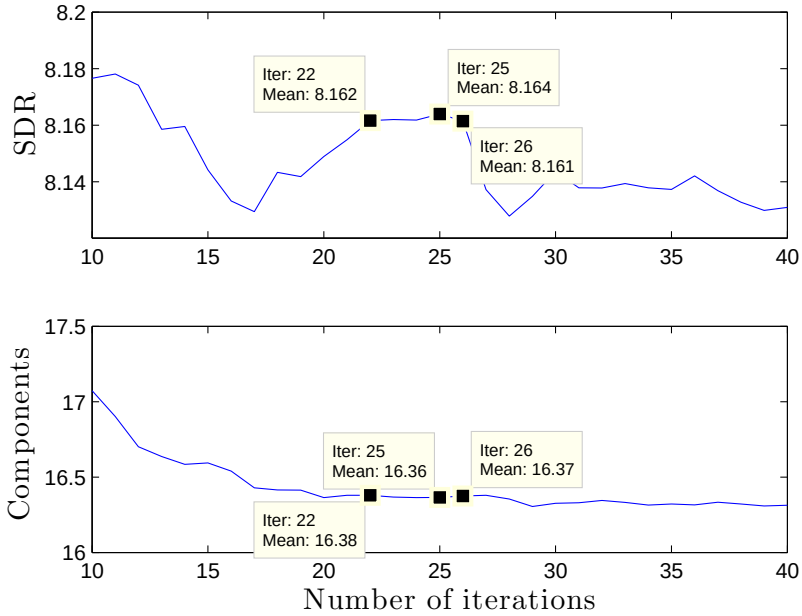


Figure 4.2: Mean SDR and $I$ values vs. Number of Iterations. $\alpha_{\text{LONMF}} = 0.02$, $T_{\max} = 3$ s.

In another simulation, the influence of the number of iterations on the separation quality is

evaluated. $N_{\text{iter}}$ was evaluated for the range [10, 40].

In Figure 4.2, the mean SDR and $I$ values are presented. The best separation quality is achieved for $N_{\text{iter}}$ set to 11. Nevertheless, the number of detected components and the memory consumption are increased. Therefore, it can be seen that the optimum number of iterations is between 22 and 26 in order to obtain a better separation quality. The best separation quality is done for $N_{\text{iter}}$ equals to 25. It should be pointed out that this simulation was done with $\alpha_{\text{LONMF}}$ set to 0.02, the same threshold as ONMF and CPONMF, and ten seeds for all the mixtures.

Note that for $N_{\text{iter}}$ set to zero, LONMF has to estimate the buffer $\boldsymbol{V}_{\text{buff}}$ without iterations. Thus, $\boldsymbol{V}_{\text{buff}}$ is estimated with basis vectors initialized with random values. Therefore, the number of detected components is increased and also the memory consumption. For $N_{\text{iter}}$ set to infinity, the estimation of $\boldsymbol{V}_{\text{buff}}$ is better and the number of detected components will be lower. Nevertheless, the execution time is increased as well.

The last evaluation was done in order to find the maximum number of frames in the buffer $T_{\text{max}}$ which provides better separation quality. In Figure 4.3, the mean values of SDR and detected components are shown.

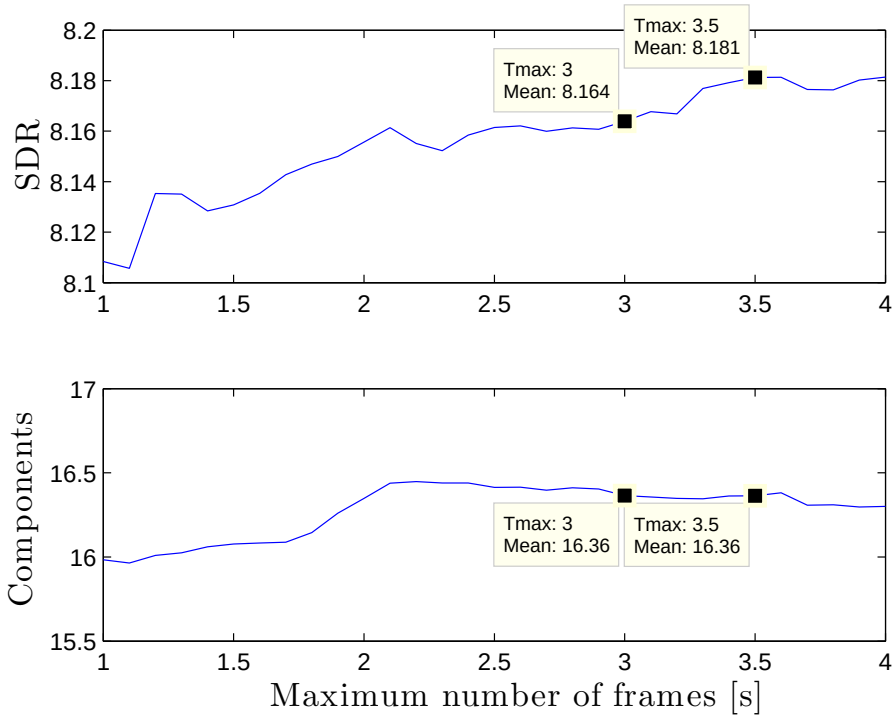


Figure 4.3: Mean SDR and $I$ values vs. Maximum number of frames. $\alpha_{\text{LONMF}} = 0.02$, $N_{\text{iter}} = 25$.

If $T_{\text{max}}$ is increased to 3.5 seconds (77 frames), the SDR mean value is increased also and the mean $I$ value remains constant. Therefore, in order to increase the SDR values a bit, $T_{\text{max}}$ can be increased. It should be pointed out that the execution time is increased a few.

The number of frames which belongs to $T_{\max}$ is defined by Equation 4.1.

$$F_{\max} = \left\lfloor \frac{T_{\max} F_s + 1 + h_s}{h_s} \right\rfloor + 1 \tag{4.1}$$

Note that for $T_{\max}$ set to zero seconds, the maximum number of frames for $\boldsymbol{V}_{\text{buff}}$ is two. Therefore, the behavior of LONMF is close to CPONMF with one the difference, the initialization of the basis and gain vectors is with random values for LONMF and with the unknown information for CPONMF as explained in Section 2.2.2. If $T_{\max}$ is set towards infinity, LONMF fills the buffer with the rest of frames of the mixture until the end. Therefore the execution time, the memory consumption and the latency are increased.

In order to compare the results of each algorithm, the mean SDR, segmental SDR and the number of components obtained for all the mixtures is shown in Table 4.2. It should be pointed out that the threshold is set to 0.02 for all the online algorithms.

| | **NMF** | **ONMF** | **CPONMF** | **LONMF** |
|---|---|---|---|---|
| **SDR** | 11.89 | 8.11 | 8.19 | 8.18 |
| **Segmental SDR** | 12.61 | 11.93 | 12.01 | 11.89 |
| **I** | 20 | 19.86 | 19.81 | 16.36 |

Table 4.2: Mean Values obtained by NMF, ONMF and CPONMF in test set.

The SDR values obtained by the online NMF algorithms are almost the same. Nevertheless looking at the number of detected components, the LONMF achieves less components in mean with the same threshold than ONMF and CPONMF. Which means that the LONMF algorithm is able to obtain almost the same separation quality with less memory consumption and latency.

In order to compare the execution time of all the algorithms, a simulation was done in the same computer. It was done with the optimum parameters obtained for LONMF. That is: the threshold $\alpha_{\text{LONMF}}$ was set to 0.02 ,to make the results comparable; the number of iterations $N_{\text{iter}}$ was set to 25 and the maximum number of frames $T_{\max}$ was set to 77 (3.5 seconds). It should be pointed out that this simulation was done for ten mixtures selected randomly between the 257 mixtures and a hundred of times. The used computer is a laptop equipped with a 2.5GHz quad-core Intel® Core™ i5-2450M processor, with 4GB of RAM and the OS is Windows® 7 premium. The simulation was done in MATLAB® version 8.2.0.701(R2013b).

Therefore, the LONMF algorithm can obtain almost the same SDR values in mean than CPONMF for less time. This is due to the fact that the number of iterations is lower and it can detect more than one component with the same buffer. On the other hand, if the threshold is decreased between 0.015 and 0.018, a better estimation is achieved. Between

| | **NMF** | **CPONMF** | **LONMF** |
|---|---|---|---|
| **Time** [s] | 47.08 | 18.41 | 14.58 |
| **SDR** | 8.38 | 4.03 | 4.32 |
| **Segmental SDR** | 10.89 | 4.74 | 5.06 |
| $\alpha$ | - | 0.02 | 0.02 |
| $N_{\mathbf{iter}}$ | - | 100 | 25 |
| $T_{\mathbf{max}}$ | - | - | 3.5 |

Table 4.3: Mean time obtained by NMF, ONMF and CPONMF in test set.

0.6 dB and 0.14 dB compared with CPONMF as can be seen in Figure 4.1. Nevertheless, the mean time is increased between 8.7 and 3.42 seconds for mixture.

It should be pointed out that, it has been demonstrated that the best algorithm in separation quality is the NMF algorithm and the online algorithms are faster than NMF.

Chapter 5

# Conclusion

The Blind Source Separation (BSS) problem is presented in many systems which obtain information from more than one source. Therefore, an algorithm able to solve this problem is required in the signal processing world. This thesis is focused on unsupervised audio source separation.

The algorithms based on Non-negative Matrix Factorization (NMF) are the most extended in order to solve the BSS problem. The basic NMF algorithm was proposed by Lee and Seung [4] as the iterative NMF. The iterative NMF algorithm is able to solve the unsupervised source separation problem in most scenarios with great separation quality. Basically, the goal of NMF is to factorize an input non-negative matrix into the product of two non-negative matrices called basis and gain matrices. The basis matrix contains the components which are presented in the mixture. The gain matrix contains the projection of each component in the time of the mixture. In audio source separation, the non-negative matrix used as input is a frequency-time representation of the mixture called spectrogram. Therefore, a mixture larger in time produces a larger spectrogram.

Nevertheless, NMF has some disadvantages on account of its computational requirements, the fixed number of components $I$ which is a parameter defined by the user. The number of components is the number of acoustical events presented in the mixture which can be approximated to the number of notes. However, some notes are composed of more than one component and this number is not usually known beforehand. Additionally, NMF requires the complete spectrogram of the mixture which demands larger memory and introduces high latency for larger mixtures. Thus, this algorithm is not able for real time scenarios.

In order to overcome the disadvantages of NMF, Online algorithms based on NMF were proposed [5]. The basic online NMF (ONMF) algorithm does not require a fixed number of components giving it a degree of blindness to the solution of the BSS problem. ONMF divides the input spectrogram in time slots called frames. Therefore, the aim of the basic ONMF algorithm is to estimate each frame with the product of a basis and a gain vectors. In order to evaluate the accuracy of the estimated frame against the current frame, both are compared using the Kullback-Leibler divergence as a cost function. Then, the divergence is compared against a prefixed threshold.

This comparison is the fundamental point of this algorithm because the estimation of the

new basis vector is decided in this step of the ONMF procedure. If a new acoustical event is presented, the basis matrix is not able to represent it. Therefore, the divergence will be above the threshold and a new component is detected as the difference of both frames. It should be pointed out that the difference has to maintain positive values, the algorithm is based on non-negative matrices, then the negative values are set to zero and some information is missing with this procedure. This is why some variants of ONMF were presented in [5].

The best algorithm in terms of separation quality presented in [5] is a variant of ONMF. This algorithm requires the current and the past frame for a basis vector estimation, in this thesis is called Current and Past ONMF (CPONMF). When an acoustical event cannot be represented by the basis matrix, CPONMF calculates the difference between two real and two estimated frames, that is the current and the past frame. This difference is defined as the unknown information matrix. As ONMF, the unknown information must be a non-negative matrix. Afterwards, CPONMF applies NMF with the unknown information matrix in order to obtain the basis matrix. There are some missed information with this procedure but the basis estimation is better than ONMF.

Thus, the online algorithms decrease the execution time, the memory consumption and the latency compared with NMF due to the fact that both algorithms use one or two frames instead of the full spectrogram. Nevertheless, some acoustical events require more than two frames to be fully developed. Therefore, the use of just one or two frames can produce some errors in the basis vector estimation. This principle has motivated the algorithm presented in this thesis, the Look-ahead Online NMF (LONMF) algorithm.

The goal of the LONMF algorithm is to provide a better basis vector estimation with the help of more than two frames if necessary. Basically, LONMF estimates the number of frames which belongs to an acoustical event and then estimates its components. If the basis matrix is not able to represent an acoustical event, LONMF appends the frames of the event in a buffer. Then LONMF applies NMF over the buffer in order to estimate its components. Therefore, this algorithm estimates more than one component with the same buffer. This happens when the buffer has more than one note or the note in the buffer is composed of more than one component as the harmonic notes.

LONMF was evaluated with a test set composed of 257 mixtures of different time length. As presented in Chapter 4, the SDR values obtained by NMF are unreachable in most cases because of is the best algorithm in terms of separation quality. LONMF provides almost the same separation quality than the other online NMF algorithms for the same threshold. Nevertheless, the execution time is lower for LONMF. This means that for a lower threshold, the estimation is better due to the fact that the number of detected components are closer to the real. However, as exposed before the execution time and the memory consumption increases.

The main disadvantage of LONMF is that in some cases, a considered buffer cannot be separated properly even with the maximum number of components allowed. The maximum number of components permitted was set to the third part of the size of the buffer. This definition was done due to a trade off between quality estimation and time. When that

maximum number is reached, LONMF continues the execution with the next frame after the buffer and store the last basis vectors detected to the basis matrix. Therefore, the last basis vectors detected are not a good estimation of the buffer. Then, here is a point of future research. It is important to estimate better the size of the buffer or use another type of cost functions in order to obtain better separation quality.

This disadvantage can be resolved if the number of iterations is increased. Nevertheless, the execution time and the memory consumption are increased.

Appendix A

# Non-negative Matrix Factorization Algorithms

The block diagram of the NMF algorithm is presented in Figure A.1.

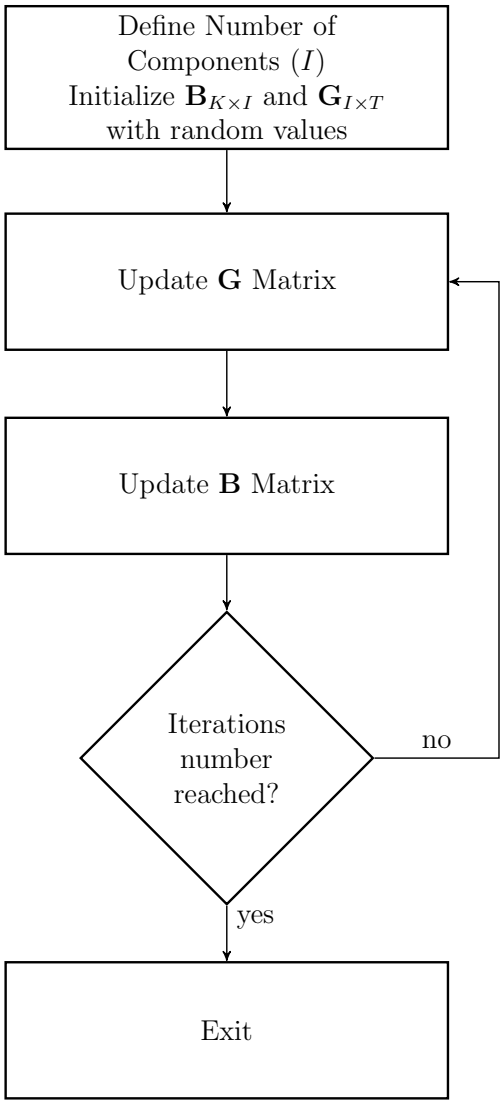


Figure A.1: NMF Block diagram.

The Block diagram shown in Figure A.2 is a graphical representation of the basic ONMF algorithm described in Section 2.2.1.

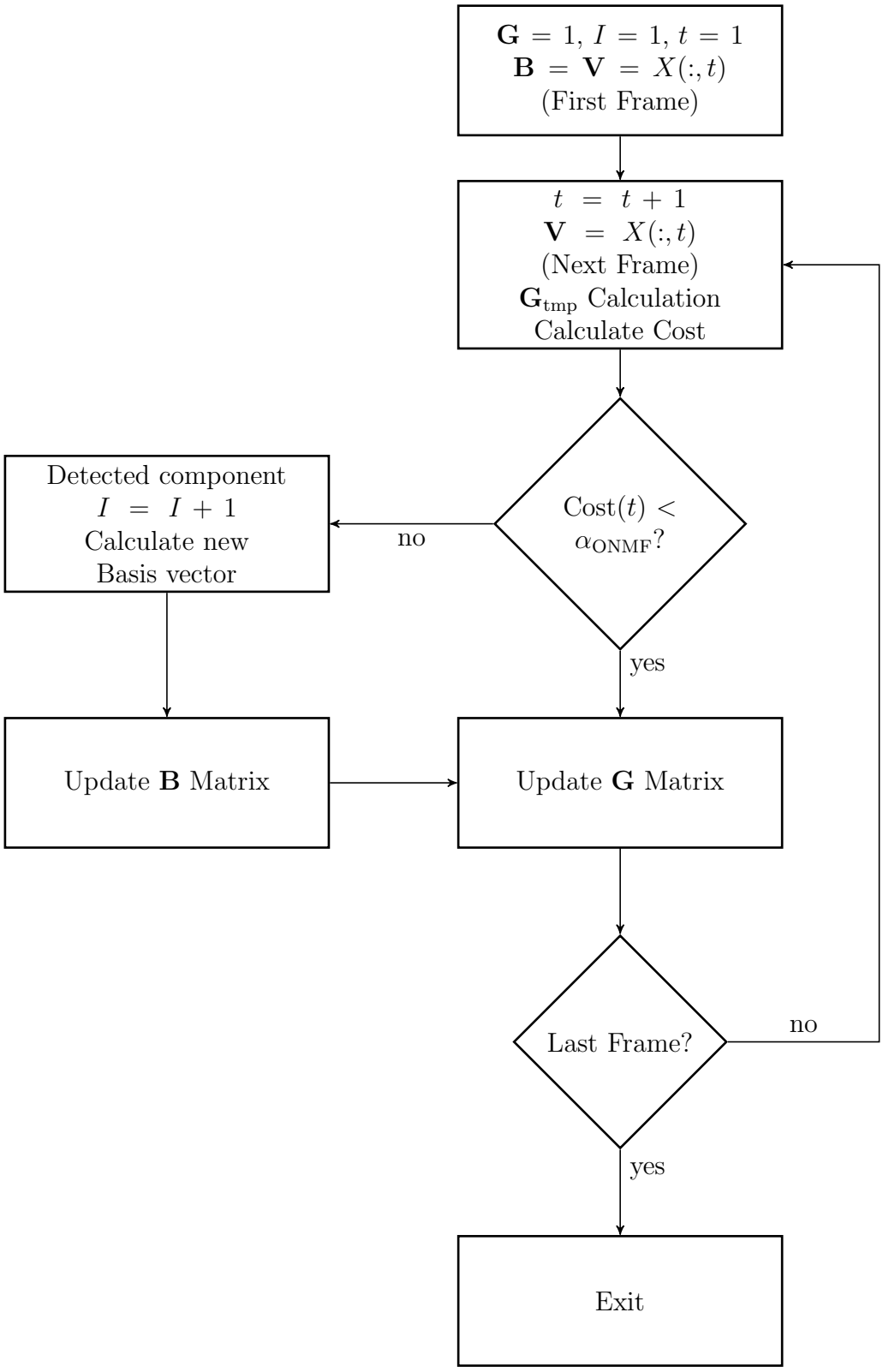


Figure A.2: ONMF Block diagram.

The Block diagram shown in Figure A.3 is a graphical representation of the CPONMF algorithm described in Section 2.2.2.

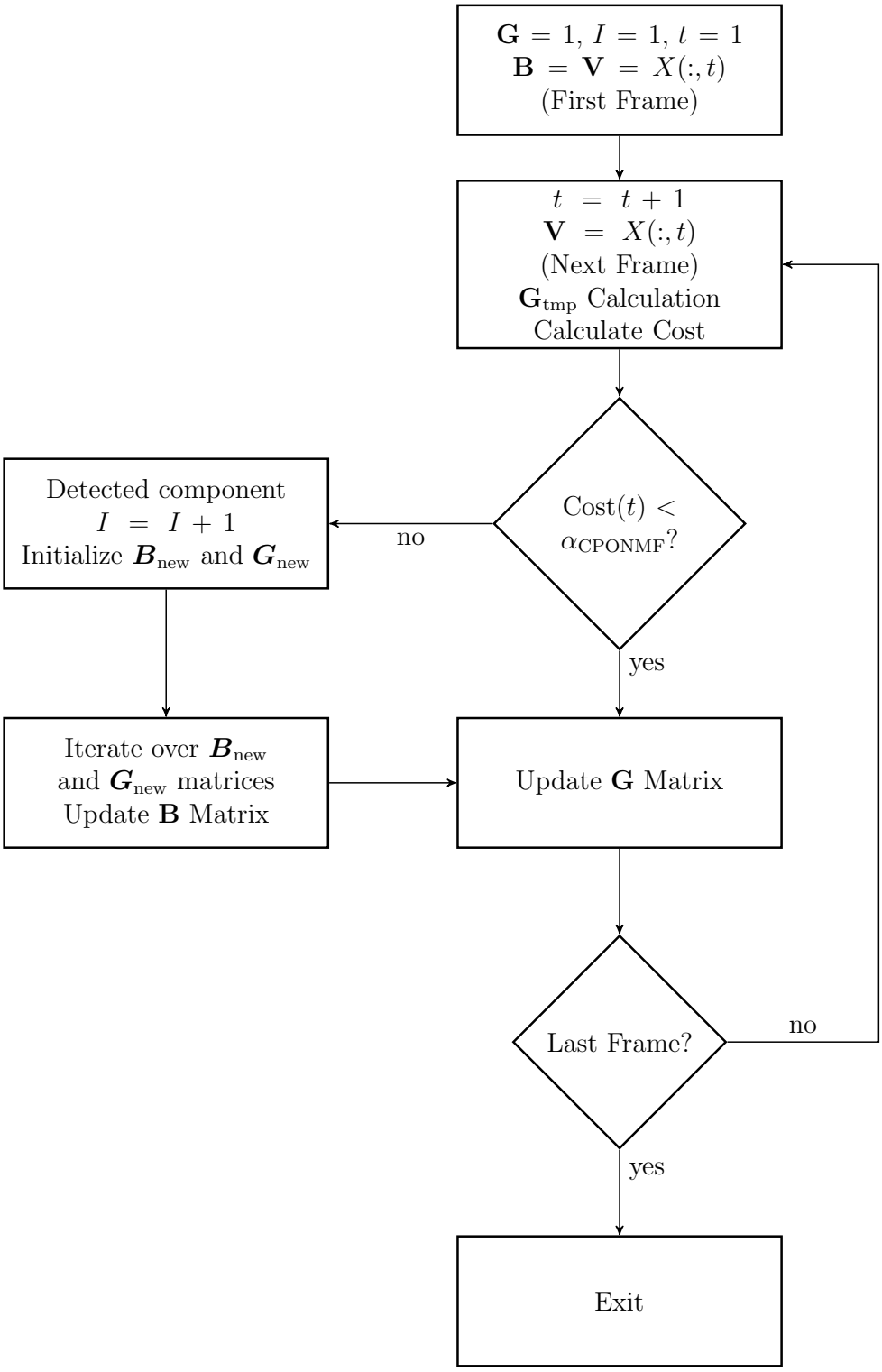


Figure A.3: CPONMF Block diagram.

The LONMF algorithm explained in Section 3.1 is sumarized in the Block diagram shown in Figure A.4.

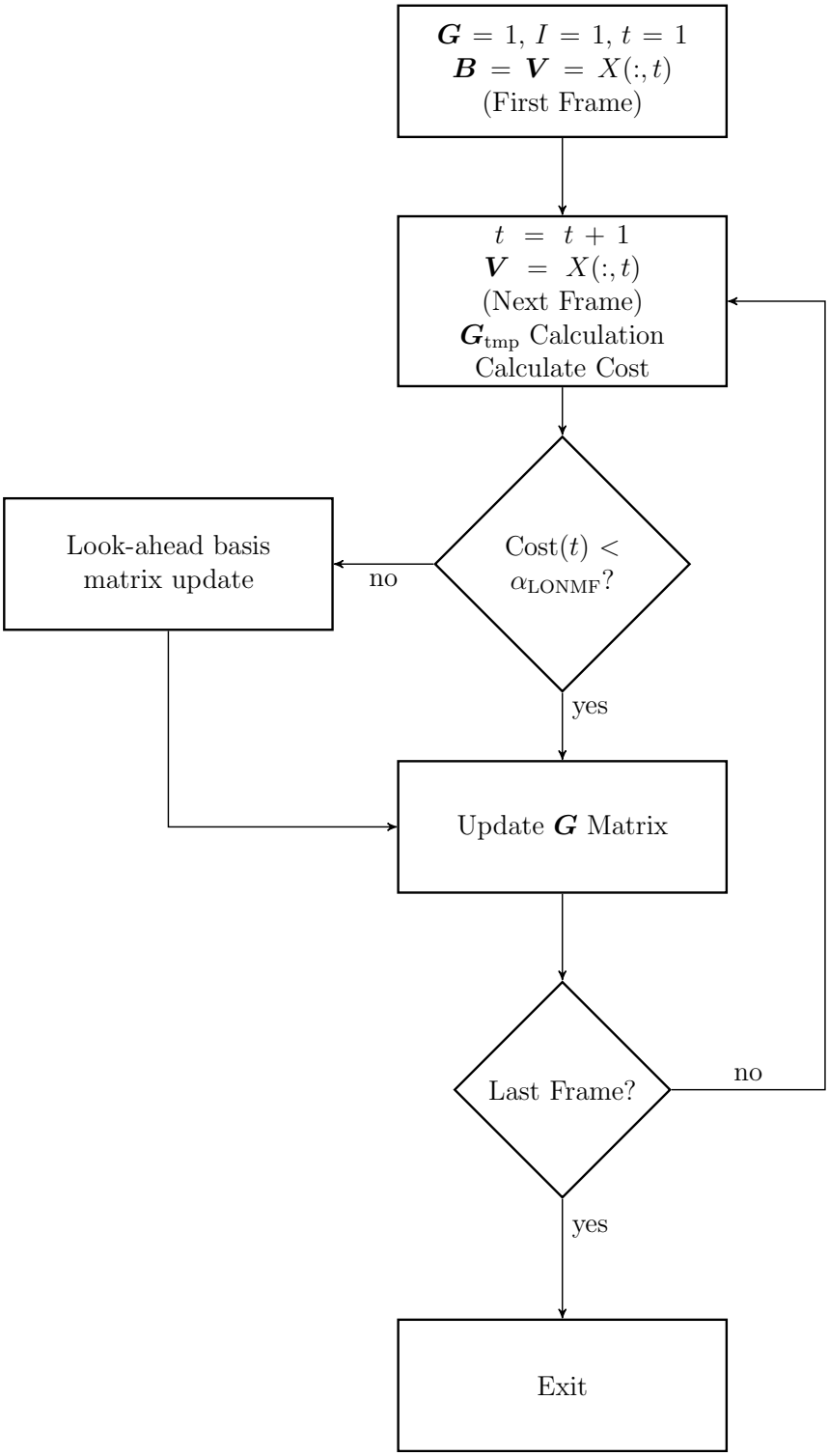


Figure A.4: LONMF Principal Block diagram.

The block diagram of the "Look-ahead basis matrix update" procedure is shown in Figure A.5

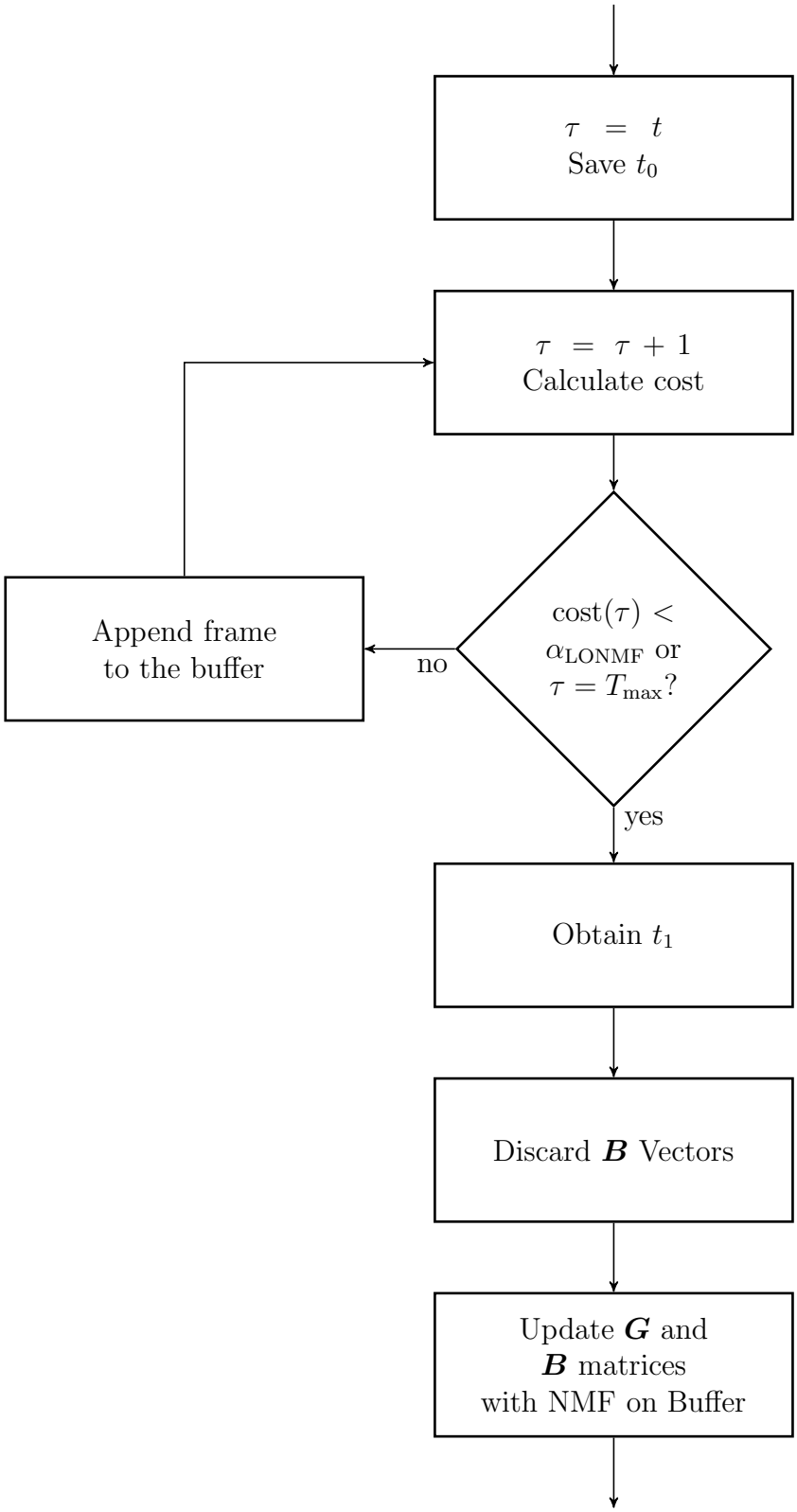


Figure A.5: LONMF Update $\boldsymbol{B}$ Matrix Block diagram.

# List of Figures

# List of Tables

# Bibliography

[1] V. Zarzoso and A.K. Nandi. “Blind Source Separation”. English. In: *Blind Estimation Using Higher-Order Statistics*. Edited by AsokeKumar Nandi. Springer US, 1999, pages 167–252. ISBN: 978-1-4419-5078-9. DOI: 10.1007/978-1-4757-2985-6_4 (cited on page 1).

[2] E. Colin Cherry. “Some Experiments on the Recognition of Speech, with One and with Two Ears”. In: *The Journal of the Acoustical Society of America* (May 5, 1953) (cited on page 1).

[3] Pierre Comon and Christian Jutten. *Handbook of blind source separation: independent component analysis and applications*. English. 1st. Amsterdam [etc.]: Elsevier, 2010. ISBN: 9780123747266 (cited on page 1).

[4] Daniel D. Lee and H. Sebastian Seung. “Algorithms for Non-negative Matrix Factorization”. In: *In NIPS*. MIT Press, 2000, pages 556–562 (cited on pages 2, 8, 9, 43).

[5] Bilal Shuja. “Low Latency Online NMF for Unsupervised Source Separation”. Master’s thesis. Institut für Nachrichtentechnik (IENT), RWTH Aachen University, 2014 (cited on pages 2, 3, 14, 18, 21, 28–30, 38, 43, 44).

[6] J. Ohm. *Multimedia Communication Technology: Representation,Transmission and Identification of Multimedia Signals*. Signals and Communication Technology. Springer Berlin Heidelberg, 2003. ISBN: 9783540012498 (cited on page 7).

[7] *Professionally produced music recordings*. Sept. 28, 2015. URL: https://sisec.wiki.irisa.fr/tiki-index.php?page=Professionally+produced+music+recordings (cited on page 37).

[8] Martin Spiertz. *Underdetermined Blind Source Separation for Audio Signals*. Shaker, 2012. ISBN: 978-3-8440-1174-6 (cited on page 37).