UNIVERSIDAD
POLITECNICA
DE VALENCIA
CAMPUS DE GANDIA

Universität Bremen

Universidad Politécnica de Valencia | Unviersität Bremen

Independent Component Analysis: Blind Source Separation

Bachelor Thesis

Dept. of Communications Engineering

Author: Vicente Boluda Burguete

Supervisor: Ban-Sok Shin

I, Vicente Boluda Burguete, assure that this thesis has been entirely written by me and no other sources than the mentioned have been used.

Contents

1. Introduction & Objectives

1.1. Motivation

Independent Component Analysis (ICA) is a technique used since middle 80s, and due to all its applications, it has been a common research topic. Simplifying the concept, with the ICA technique we can separate multivariate additive signals. Despite that there are other methods to do so, ICA can do it without knowing nothing (or barely nothing) of the signals and context.

Along this thesis the basic algorithm for Independent Component Analysis will be explained. It is called FastICA and was invented by Aapo Hyvärinen as a simply and versatile algorithm with a scheme of fixed-point iterations. This means an algorithm that search the convergence of a vector with iterations, similar to the Newton's method.

This technique is not that simple though, the mathematic and theoretical background is quite complex. But in order to understand how the algorithm works, all of the concepts will be explained step by step.

As has been said before, there are a lot of applications: biomedical, image processing, CDMA communications, etc. But the goal of this report is to talk about how to apply the FastICA algorithm to solve the Blind Source Separation. Specifically to the audio signals separation.

The problem is as follows: we are going to have two speakers and two microphones, each microphone will record a mix of the two sources. The goal is to use ICA to separate the sources.

1.2. Abstract

Part 2

The problem and the scenario are presented. The characteristics of the plot and its main equation will be showed. This is the starting point of the whole process.

Part 3

Independent Component Analysis techniques are quite complex. It is necessary to know the mathematics principles that are used. The measurement of gaussianity and its fundamental relation with ICA is explained.

Part 4

In this section the actual FastICA algorithm is explained. The theoretical part is used to do so, and the section includes the pseudocode and the flow diagram for an easy understanding.

Part 5

After using the algorithm the results must be analyzed. In this part the quality of the separation is analyzed, this is made by comparing the original signals with the signals obtained with the FastICA algorithm.

Part 6

At the end of the document a conclusion can be reached. Also there is a brief explanation about other applications besides the audio separation.

1.3. Notation

x	scalar
$\boldsymbol{x}$	vector
$\boldsymbol{X}$	matrix
$\boldsymbol{x}_i$	the ith vector of the matrix $\boldsymbol{X}$
x_{ij}	the component of the row *i* and the column *j* of the matrix $\boldsymbol{X}$
X	A continue variable

1.4. Objectives of the thesis

Basically, the main objective of this thesis is the study of the basics of Independent Component Analysis and the FastICA. Then this algorithm will be applied as a solution of the Blind Source Separation. To reach the main objective, there are another minority objectives.

- Describe the theory and mathematical background of the Independent Component Analysis basics.
- Describe the scenario and its characteristics.
- Create a virtual scenario of the Cocktail Party Problem and simulate it.
- Check the results and make an analysis of them

2. Problem Description

2.1. Description of the problem

Returning to the goal, the objective is to solve the Cocktail Party Problem. First of all, this concept have to be defined. This concept was used for the first time by Colin Cherry, a cognitive scientist, in 1953. He defines the Cocktail Party Effect, which is the capacity to focus in one specific conversation in a, for example, cocktail party.

So, the Cocktail Party Problem is the situation in which we are in a party with a lot of conversations at the same time and we want to separate those conversations. The human being is able to do that by nature, but we want to solve this with recorded signals.

The separation of the individual sources is known as the Blind Source Separation (BSS). In this context, blind means that we barely know anything about the source signals.

2.2. Characteristics of the scenario

In real life, we can have hundreds of different scenarios with the Cocktail Party Problem. But may be the one described in the paper called '*Convolutive BSS of Short Mixtures by ICA Recursively Regularized Across Frequencies*' is the simplest one.

This scenario consists of a room with two speakers and two microphones. This is, in fact, the simplest scenario we can find. So, at the end we will have two different signals, and each signal will record a different mix of the two speakers. In the Fig.1 we can see a scheme of the room's scenario. Note: the height of the room is also 5 m.

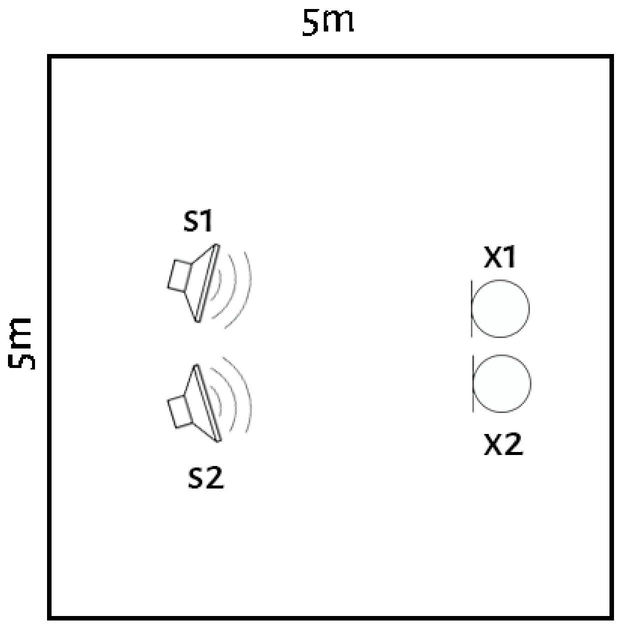


Fig.1: The virtual scenario

As we can see in Fig.1 each microphone records the signals with different amplitude because of the distance. So, the microphone 1 will record the source 1 louder than the microphone 2.

Obviously, the original source signals will not be the same that the signals recorded by the microphones, because the room is acting as a filter. Each room has its impulse response depending on the size, reverberation, etc.

For make the problem simply we are going to suppose a low reverberation time, simulating with a T60 of 300ms. This is because (as is explained later) FastICA have troubles with high reverberations.

2.3. Simulation of the scenario

The simulation of the room will be made with a Matlab package developed, in which is implemented the image-source method (ISM). With this method, the decay of the sound can be calculated with the size and the acoustic characteristics of the room. This package has been developed by Eric A. Lehmann, a Swiss researcher of the Swiss Federal Institute of Technology (ETHZ).

Using this package, the size of the room and the desired reverberation time, we can simulate both microphones. At the end of the simulation we will have a matrix of two vectors, and each vector contain the information of one microphone. These vectors have the audio information sampled with a frequency of 44100 Hz.

These two vectors are the starting point of the ICA algorithm. For now on, we will call the microphone signals $\boldsymbol{X}$, being $\boldsymbol{X}$ a matrix with a size *Number of Samples x 2.*

The number of samples will depend of the length (in seconds) of the audio files.

2.4. System equation of ICA

Now, the main equation of the problem have to be written. Until now we have two microphones with their recorded signals, $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. We also have our incognita, the speaker's signals, called $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$. As we know, $\boldsymbol{x}_i$ are weighted sums of $\boldsymbol{s}_i$ with a coefficients that are dependents on the distance between speakers and microphones. Knowing that, our equation system will be the following:

$$\boldsymbol{x}_1 = a_{11}\boldsymbol{s}_1 + a_{12}\boldsymbol{s}_2 \ (1)$$

$$\boldsymbol{x}_2 = a_{21}\boldsymbol{s}_1 + a_{22}\boldsymbol{s}_2 \ (2)$$

This system can be written as a matrix equation:

$$\boldsymbol{X} = \boldsymbol{AS} \ (3)$$

Of course we only know the matrix $\boldsymbol{X}$, in order to solve the Blind Source Separation, we need to know the matrix $\boldsymbol{A}$, called the mixing matrix. Then, and only then, we will be able to obtain $\boldsymbol{S}$ with the next operation:

$$\boldsymbol{S} = \ \boldsymbol{A}^{-1}\boldsymbol{X} \ (4)$$

For now on, the matrix $\boldsymbol{A}^{-1}$ will be called $\boldsymbol{W}$.

3. Independent Component Analysis

Before start explaining the fundaments of Independent Component Analysis the key idea of the algorithm must be clear. The goal of FastICA is to found the matrix **W** by maximizing the non-*gaussianity*. This is based in the central limit theorem, which said that a sum of any random variables give a normal distribution. So, if the algorithm search for non-*gaussianity* it can find the de-mixing matrix.

3.1. Conditions in ICA

The variables of the Independent Component Analysis have to comply a series of conditions in order to work correctly. In this section we will see the restrictions in ICA.

3.1.1. Statistical Independence

We can say that the main condition of this technique is to work with variables that are statistically independent. This can be obvious, but for a good understand of the theory we must define what statistical independence is.

Imagine two vectors with random coefficients, ***x*** and ***y***. We can approximate an easy definition, saying that, if knowing a value of ***x*** doesn't give us information about ***y*** the vectors are independents. We will have this in the majority of the scenarios, with two different physical measures, with speech and noise, etc. This is why ICA is so powerful, it can be applied in a lot of contexts.

We can approximate a mathematical definition using the probability density functions (pdf). Let's define the probability density of ***x*** and ***y***: $p_x(x)$ and $p_y(y)$. We can be sure that ***x*** and ***y*** are independent if:

$$p_{x,y}(x,y) = p_x(x)p_y(y) \; (5)$$

That is to say, if the joint pdf of the variables are factorable then they are independent.

3.1.2. Non-*Gaussian* distributions

Other restriction of ICA is that the variables must not have *gaussian* distributions. We need to know that ICA uses the information of the high order cumulants of the probability density function. In a *gaussian* function from the second moment, the cumulants are equals to zero.

In statistics, the cumulants gives us an alternative to calculate the statistical moments of any probability distribution. Along this document we will work with the third and fourth moment, which are the skewness and the kurtosis.

3.2. Preprocessing the Independent Components

Before start talking about the actual algorithm and the ICA technique, we must now a few things about how to treat the variables. For an optimal use of ICA, we have to give a preprocessing to our signals.

3.2.1. Centering the variables

Along this thesis, we assume that the independent components have zero mean. This is for simplify the implementation and the theory of the algorithm, but normally this isn't true.

Fortunately, we can force the variables to have zero mean easily. Is as easy as *center* the variables as follows:

$$\boldsymbol{x}' = \boldsymbol{x} - E\{\boldsymbol{x}\} \text{ (6)}$$

Being $\boldsymbol{x}$ the original signal and $\boldsymbol{x}'$ the same signal with zero mean. Of course, this transformation doesn't affect the estimation of the mixing matrix $\boldsymbol{A}$.

3.2.2. Whitening the variables

We have talked about that the independence is necessary but, also a uncorrelated space is needed. We must force a transformation for this to happen, this is typically called whitening or sphering.

Uncorrelatedness is not the same as independence. Two variables are uncorrelated when its covariance is zero

$$cov(\boldsymbol{x},\boldsymbol{y}) = E\{(\boldsymbol{x} - E\{x\})(\boldsymbol{y} - E\{y\})\} = 0 \text{ (7)}$$

Since we will work with a matrix system, we have to make the covariance matrix of the vectors $\boldsymbol{x}_1$and $\boldsymbol{x}_2$ equal to the identity matrix $\boldsymbol{I}$. Mathematically, the definition of the covariance matrix is as it follows

$$\boldsymbol{C}_x = E\{(\boldsymbol{x_1} - E\{\boldsymbol{x}_1\})(\boldsymbol{x}_1 - E\{\boldsymbol{x}_1\})^T\} \text{ (8)}$$

As we say before, the variables should have zero mean, so is obvious that $E\{\boldsymbol{x}_1\} = 0$. According to that, we can say that a variable is uncorrelated if

$$\boldsymbol{C}_x = E\{\boldsymbol{x}_1\boldsymbol{x}_1^T\} = \boldsymbol{I} \ \ (9)$$

Whitening would do that possible, with the following linear transformation

$$\boldsymbol{z}_1 = \boldsymbol{V}\boldsymbol{x}_1 \ (10)$$

Being $\boldsymbol{z}_1$the whitened signal, $\boldsymbol{x}_1$ the original signal (with zero mean) and $\boldsymbol{V}$ a transformation matrix. There are various methods to do so, a good idea is to use Principal Components Analysis (PCA). Due to this thesis is not about PCA, we only will see the basics of this technique. Depending of the field of application PCA can be named in different ways, we will use the eigenvalue decomposition (EVD).

As a starting point we will define $\boldsymbol{E} = (e_1 \ldots e_n)$ as the matrix whose columns are the eigenvectors of $\boldsymbol{C}_x$. And $\boldsymbol{D} = diag(d_1 \ldots d_n)$ the diagonal matrix of the eigenvalues of $\boldsymbol{C}_x$. The eigenvalue decomposition can be always done if the matrix is diagonalizable. It consists in apply the EVD technique in the covariance matrix.

$$\boldsymbol{C}_x = E\{\boldsymbol{x}_1\boldsymbol{x}_1^T\} = \boldsymbol{EDE}^{\boldsymbol{T}} \ (11)$$

Then, the whitening matrix would be the following one

$$\boldsymbol{V} = \boldsymbol{ED}^{-1/2}\boldsymbol{E}^{\boldsymbol{T}} \ (12)$$

The matrix $\boldsymbol{D}^{-1/2}$ can be calculated with $\boldsymbol{D}^{-\mathbf{1}/2} = diag(d_1^{-1/2} \ldots d_n^{-1/2})$.

3.3. Measuring *Nongaussianity*

There are different ways to calculate the ICA model, like Maximum Likelihood Estimation or the Minimization of Mutual Information. These are a very well-studied and used methods, despite that, for the goal of this thesis I will explain ICA by Maximization of *Nongaussianity*. This is the core of the FastICA algorithm.

In this section we will see important definitions for measuring the *gaussianity* of an independent component and the basics of this method.

For the Maximization of *Nongaussianity* we need to measure in some way the *gaussianity* of the independent components. This can be made in different ways, in this section kurtosis and negentropy will be explained. Depending of the method we use, the FastICA algorithm will be different. At the end of the section the choice of the algorithm will be made.

3.3.1. With kurtosis

We need to know how the *gaussianity* of a component is measured. This can be made with kurtosis, which is, a way to measure the form of the probability density function. Kurtosis allows us to view and study the concentration of data around of the average µ. In common words, a high kurtosis means a high concentration of samples surrounding µ. Graphically, this is a distribution with the center forming a spike. Formally, the kurtosis can be expressed as

$$\beta_2 = \frac{\mu_4}{\sigma_4} = \frac{E[(\boldsymbol{x} - E[\boldsymbol{x}])^4}{(E[(\boldsymbol{x} - E[\boldsymbol{x}])^2])^2} \text{ (13)}$$

Where μ_4 is the fourth central moment of the expectation and σ_4 is the square of the standard deviation. Despite this is the formal definition, the kurtosis is commonly expressed as the relation of the fourth cumulant ($\varkappa_4$) and the square of the second cumulant ($\varkappa_2^2$).

$$\gamma_2 = \frac{\varkappa_4}{\varkappa_2^2} = \frac{\mu_4}{\sigma_4} - 3 \text{ (14)}$$

This is equal to the fourth central moment of the expectation divided by the square of the standard deviation minus three. This is because three is the kurtosis value of a normal distribution. Doing this, the coefficient γ_2 will be zero for the normal distribution and then we will have a reference.

So, taking a normal distribution as a reference, any distribution can be classified as leptokurtic, platykurtic or mesokurtic. In a leptokurtic distribution the kurtosis is higher than zero, in a platykurtic is less than zero and mesokurtic means that the distribution is normal.

Leptokurtic	$\beta_2 > 3$	$\gamma_2 > 0$
Mesokurtic	$\beta_2 = 3$	$\gamma_2 = 0$
Platykurtic	$\beta_2 < 3$	$\gamma_2 < 0$

In the Fig. 2 we can see a comparative with different kurtosis.

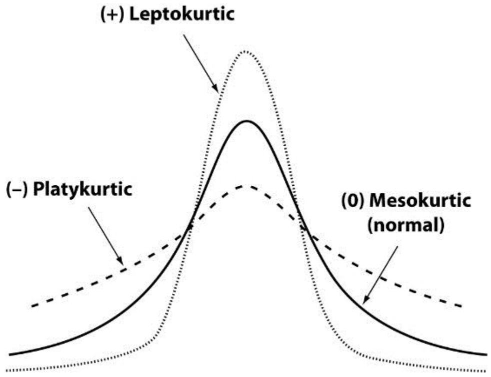


Fig 2: Platykurtic, Leptokurtic and Mesokurtic distributions

For ICA, sometimes, we work with the absolute value of the kurtosis. Because if we only want to know that the distribution is not gaussian, we don't need the sign. If the kurtosis is not null is not gaussian, and the higher the less gaussian is.

Despite this method is easy to implement, may be it is not the best for measuring *Nongaussianity*. It is known that kurtosis doesn't work well with outliers, namely, with irregular distributions or with variables with atypical values. This is why we will study another measure, the negentropy.

3.3.2. With negentropy

We can know how gaussian is a distribution measuring its entropy. This is a basic concept of the information theory, and it means that the more random a component is, the higher is its entropy. The entropy of a discrete variable is defined as follows

$$H(\boldsymbol{x}) = \sum_i P(x_i)I(x_i) = -\sum_i P(x_i)\log_b P(x_i) \ (15)$$

The base of the logarithm can change depending of the unit of the entropy we want. The common value (for computer applications) is two. We can relate entropy of a random vector with its density $p_y(\boldsymbol{\eta})$ with the next equation

$$H(\boldsymbol{x}) = -\int p_y(\boldsymbol{\eta})\log p_y(\boldsymbol{\eta})d\boldsymbol{\eta} \ \ (16)$$

For us, is important to know that a *gaussian* distribution has the highest entropy. Then, it's obvious that we can use entropy for the measuring. Graphically if the entropy is low the probably distribution will have a spike form and vice versa.

The calculation is similar to the kurtosis, we must have a reference to *gaussian* distribution with negentropy. This can be made defining negentropy as

$$J(\boldsymbol{x}) = H(\boldsymbol{x}_{gauss}) - H(\boldsymbol{x}) \ (17)$$

Where $\boldsymbol{x}_{gauss}$ have the same covariance matrix as $\boldsymbol{x}$. With this definition, we reference the variable $\boldsymbol{x}$ to the normal distribution.

Negentropy is a powerful way to measure the *gaussianity*, but the inconvenient is that is more complex than kurtosis. This is because we need to calculate the probability density function of the variables. Fortunately there are approximations for calculate the negentropy.

3.4. Approximating negentropy

In this section methods for approximate negentropy will be studied.

3.4.1. By cumulants

This is the classic method for estimate the negentropy, it is based in the use of higher-order cumulants. The idea is to use an expansion series from the probability density function. With this expansion we will be able to approximate the probability density only with the cumulants. For the explanation of this method we are going to suppose a variable with null mean and variance equals to one. Let's assume too that the pdf of the variable is almost the normal distribution

$$\varphi(\xi) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\xi-\mu}{\sigma}\right)^2} \ (18)$$

And as I said before, µ is equal to zero and σ is one. So the probability density of x is as it follows

$$\varphi(\xi) = \frac{e^{-\frac{\xi^2}{2}}}{\sqrt{2\pi}} \ (19)$$

Now, the expansion we are going to use is named as Gram-Charlier expansion, in honor to Carl Charlier and Jørgen Pedersen Gram. According to them, the polynomials are equal to the derivatives of the probability density function

$$\frac{\delta^i \varphi(\xi)}{\delta \xi^i} = (-1)^i H_i(\xi)\varphi(\xi) \ (20)$$

Where H_i are the Hermite polynomials. It is important to know that these polynomials are orthogonal, so they form an orthonormal system. Then, we can apply the Gram-Charlier expansion to calculate the pdf in a similar way as we would apply a Taylor expansion. Obviously, the expansion has infinite components, but we only are interested in the first three ones.

$$p_x(\xi) \approx \varphi(\xi)\left(1 + \kappa_3(x)\frac{H_3(\xi)}{3!} + \kappa_4(x)\frac{H_4(\xi)}{4!}\right) \ (21)$$

This Taylor similar expansion is possible because the probability distribution of x is near to the normal distribution. As a reminder, the idea is that the *gaussianity* is given by the third and fourth cumulants. We already know the fourth cumulant which is the kurtosis. The third cumulant is called skewness

$$\kappa_3(x) = E\{x^3\} \ (22)$$

$$\kappa_4(x) = E\{x^4\} - 3 \ (23)$$

At this point, we can insert the expansion into the definition of the entropy. Don't confound $H(x)$, the entropy, with $H_i(\xi)$ which are the Hermite polynomials.

$$H(x) \approx -\int p_x(\xi) \log p_x(\xi)\, d\xi \ (24)$$

Is a complex equation, but we have to recall that the cumulants of x are really small, because its pdf is nearly the *gaussian*. Because of that, we can approximate the expansion of above with the following transformation

$$\log(1+\alpha) \approx \frac{\alpha - \alpha^2}{2} \ (25)$$

Applying that, and substituting $p_x(\xi)$ in the definition of the entropy, we have the following approximation

$$H(x) \approx -\int \varphi(\xi)\left(1 + \kappa_3(x)\frac{H_3(\xi)}{3!} + \kappa_4(x)\frac{H_4(\xi)}{4!}\right)\left[\frac{\left(\log\varphi(\xi) + \kappa_3(x)\frac{H_3(\xi)}{3!} + \kappa_4(x)\frac{H_4(\xi)}{4!} - (\kappa_3(x)\frac{H_3(\xi)}{3!} + \kappa_4(x)\frac{H_4(\xi)}{4!})^2\right)}{2}\right] \ (26)$$

This long equation can be simplified using algebra, which gives us the next expression

$$H(x) \approx -\int \varphi(\xi) log \varphi(\xi) d\xi - \frac{\kappa_3(x)^2}{2 \times 3!} - \frac{\kappa_4(x)^2}{2 \times 4!} \ (27)$$

Operating that we can reach a simple definition of negentropy, more computationally simple than the first one (17)

$$J(x) \approx \frac{1}{12}E\{x^3\}^2 + \frac{1}{48}kurt(x)^2 \ (28)$$

We can easily see a problem with this approximation of the negentropy. It uses the kurtosis and, as we saw before, we may have problems with outliers. That's why we will use another approximation for our algorithm. It is possible to calculate negentropy with the expectation. This is what we will see in the next section

3.4.2. By Nonpolynomial Functions

We are going to talk about a method based on the approximation of the maximum entropy. Let's imagine that we have a series of expectations of x

$$\int p(\xi)\, F^i(\xi) d\xi = c_i \text{ ; for } i = 1, \dots, n \quad (29)$$

This expression means that we have calculated the expectations of any function F^i. Normally, these functions aren't polynomials. Due we can't calculate the maximum entropy analytically, we have to approximate the maximum entropy density p_0. For this approximation, we have to make the same assumption as in the previous section, $p(\xi)$ is similar to the *gaussian* density. We need to consider that our variable x has variance equal to one and zero mean as well.

We also will consider that the functions F^i form an orthonormal system, they are orthogonals between each other. Mathematically, the maximum entropy density is defined as

$$p_0(\xi) = Ae^{\left(\sum_i a_i F^i(\xi)\right)} \text{ (30)}$$

Where A and a_i are constants dependents of c_i, to see this we only need to substitute p_0 in the integral of above. Because the density is similar to the *gaussian* model, we can do a similar assumption as in the previous section, a_i are small because the exponential of $p_0(\xi)$ is similar to the *gaussian* exponential. Then, we can apply an approximation of the exponential function and say that, approximately, $p_0(\xi)$ is

$$\hat{p}(\xi) = \varphi(\xi)\left(1 + \sum_{i=1}^{n} c_i F^i(\xi)\right) \text{(31)}$$

Taking $c_i = E\{F^i(\xi)\}$. Now, we can insert $\hat{p}(\xi)$ in the definition of the entropy

$$H(x) \approx -\int p_x(\xi) \log p_x(\xi)\, d\xi \text{ (32)}$$

And taking the same transformation as before (25).

After the Taylor expansion and algebraic operations, we finally reach an approximation for the negentropy.

$$J(x) \approx \frac{1}{2} \sum_{i=1}^{n} E\{F^i(x)\}^2 \text{ (33)}$$

With this expression we have a measure for *nonguassianity*, because $J(x)$ will be zero if x has a gaussian distribution. Now is really important how we choose the calibrating functions F^i. We said before that they must be nonpolynomial functions, but we must take account of other properties.

The expectation of the function should be easy to calculate and strong to outliers. By definition p_0 is integrable, that's why the function should not increase fast. The choice of the functions has been studied by different people, and there are specific functions that work well.

Let's say that we use two functions, G^1 and $G^2$where the first one is odd and the second one is even. So, the odd can measure the asymmetry and the even how gaussian is the distribution. One may thing that this can be made measuring the skewness and the kurtosis but, as we say before, these cumulants are not optimal for measuring negentropy. We can reflex this example transforming the definition of $J(x)$ of above with the functions G^1 and G^2. After mathematic operations:

$$J(x) \approx k_1(E\{G^1(x)\})^2 + k_2(E\{G^2(x)\} - E\{G^2(v)\})^2 (34)$$

Where $k_1$and k_2 are constant values and v is a gaussian variable with gaussian distribution.

And if we only use one function, the approximation of the negentropy will be as in the next expression

$$J(x) \propto [E\{G(x)\}] - E\{G(v)\}]^2 \ (35)$$

The question now is: which functions should we choose? The experts in this field give us recommended functions that have proved to work good. The next examples are quite good for our purpose.

$$G_1(x) = \frac{1}{a_1} \log \cosh a_1 x \ \ (36)$$

$$G_2(x) = -e^{\left(\frac{-x^2}{2}\right)} (37)$$

With $a_1$being an integer constant with possible values from one to two. Normally it takes the value one.

At this point we are able to measuring the *Nongaussianity* of our components with negentropy. But now, we must use this technique to an ICA algorithm.

4. Blind Source Separation with ICA

In this chapter we are going to see the fundamentals of the FastICA algorithm, which is the simplest and most versatile. It works with the measurement of negentropy.

4.1. Fixed-Point Algorithm with negentropy

4.1.1. Estimating the iteration

For an easy explanation of this algorithm, first we are going to suppose that we only apply it to one component. The idea of the FastICA algorithm is to find the direction in which the projection $\boldsymbol{w^T z}$ of the vector $\boldsymbol{w}$ has the maxim *nongaussianity*. This measurement will be done with the negentropy.

We search the *nongaussianity* because when the vector reaches that we have the key to separate the signals. This is because the central limit theorem, the more gaussian the combination is the less separated the signals are.

As the name says, this algorithm is based in iterations for catching the maxim *nongaussianity*. It's similar to the famous Newton's iteration method.

We already have seen that an approximation to negentropy can be calculated with the expectation, in this case, of the projection $\boldsymbol{w^T z}$. We must recall that $\boldsymbol{z}$ is a whitened variable, so the variance of $\boldsymbol{w^T z}$ will be equal to one. This also means that the norm of $\boldsymbol{w}$ must be always one.

The basic fixed-point iteration is the following one.

$$\boldsymbol{w} \leftarrow \gamma E\{\boldsymbol{z} g(\boldsymbol{w}^T \boldsymbol{z})\} \text{ (38)}$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} / \| \boldsymbol{w} \| \quad \text{(39)}$$

Where g is the derivate function of G and $\gamma = E\{G(\boldsymbol{w}^T \boldsymbol{z})\} - E\{G(\boldsymbol{v})\}$ is a constant with $\boldsymbol{v}$ as a gaussian variable. This would be the iteration in the loop until the projection $\boldsymbol{w}^T \boldsymbol{z}$ converges. The constant γ would give us an adaptation quality factor, but this is only used in gradient based algorithms. In our iteration algorithm, the coefficient γ will be omitted because with the normalization it disappears.

We have a problem with this iteration, though. The convergence will be not as good as with the kurtosis method, because the properties of the nonpolynomial functions. That's why we must adapt this iteration. We can modify the equation if we multiply both sides with a, let's say, α factor.

$$\boldsymbol{w} = E\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z})\} \ (40)$$

$$\downarrow$$

$$(1+\alpha)\boldsymbol{w} = E\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z})\} + \alpha\boldsymbol{w} \ (41)$$

The coefficient α is just a factor for regulate and control the convergence, in this thesis it is taking equal to one. Now, we have to find an appropriate coefficient for the algorithm. We are searching the maxima of the expectation $E\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z})\}$ with the condition that the norm of $\boldsymbol{w}$ must be equal to one. According to Lagrange, this maxima is obtained when the Lagrange's gradient is equal to zero.

$$F = E\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z})\} + \alpha\boldsymbol{w} = 0 \ (42)$$

For solve this problem we can use the Newton's iteration method, that will give us the maxima of the Lagrangian gradient. The gradient will be the following one

$$\frac{\delta F}{\delta \boldsymbol{w}} = E\{\boldsymbol{z}\boldsymbol{z}^T g'(\boldsymbol{w}^T\boldsymbol{z})\} + \alpha\boldsymbol{I} \ (43)$$

Note: As a reminder, $\boldsymbol{I}$ is the identity matrix.

Now we can simplify considerably the gradient. We can do the next assumption because the properties of the statistical expectation.

$$E\{\boldsymbol{z}\boldsymbol{z}^T g'(\boldsymbol{w}^T\boldsymbol{z})\} \approx E\{\boldsymbol{z}\boldsymbol{z}^T\}E\{g'(\boldsymbol{w}^T\boldsymbol{z})\} \ (44)$$

And because $\boldsymbol{z}$ is a whitened variable, it's obvious that $E\{\boldsymbol{z}\boldsymbol{z}^T\} = \boldsymbol{I}$. Then we can simplify the expression

$$E\{\boldsymbol{z}\boldsymbol{z}^T g'(\boldsymbol{w}^T\boldsymbol{z})\} \approx E\{g'(\boldsymbol{w}^T\boldsymbol{z})\}\boldsymbol{I} \ (45)$$

So, the gradient becomes the next expression

$$\frac{\delta F}{\delta \boldsymbol{w}} = E\{g'(\boldsymbol{w}^T\boldsymbol{z})\}\boldsymbol{I} + \alpha \ (46)$$

At last, the Newton's iteration of the gradient will be the following one

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \frac{[E\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z})\} + \alpha\boldsymbol{w}]}{E\{g'(\boldsymbol{w}^T\boldsymbol{z})\} + \alpha} \ (47)$$

This is not the final iteration, though. We can simplify the expression by multiplying $E\{g'(\boldsymbol{w}^T\boldsymbol{z})\} + \alpha$ in the both parts. That will give us the final result (after some algebraic operations)

$$\boldsymbol{w} \leftarrow E\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z}) - E\{g'(\boldsymbol{w}^T\boldsymbol{z})\}\boldsymbol{w}\} \ (48)$$

Actually, this is the fixed-point iteration that we will use in the FastICA algorithm.

Now, we have to define the function g. We already know that this function is the derivate of Gin the definition of the negentropy $J(x)$. The derivatives of the suggested functions $G_1$and $G_2$are the followings ones.

$$g_1 = \tanh(a_1 x)\,;\; g_2 = xe^{\frac{-x^2}{2}} \quad (49)$$

And graphically, where g_1 is the blue one and g_2 the red one.

And we also need the second derivate for the iteration algorithm. The result of that give us the following available functions for g'

$$g'_1 = a_1\left(1 - tanh^2(a_1 x)\right) \; (50)$$

$$g'_2 = (1 - x^2)e^{\frac{-x^2}{2}} \; (51)$$

And we can see its graphics in the next image, with g'_1as the blue one and g'_2as the red one.

4.1.2. Algorithm for the iteration

At this point we already know the preprocessing of the signal, how the iteration is and which functions should we use. Then, we can write our algorithm.

Let's recapitulate: first we must center and whiten our recorded signal. A vector can be initialized (e.g. randomly) to find the inverse of the mix matrix. After that the itineration will start, and it won't stop until the vector converges.

For monitor if the vector converges, we can look at the absolute value of the dot-product of the actual vector and the vector of the past iteration.

This process can be draw in form of a flow diagram.

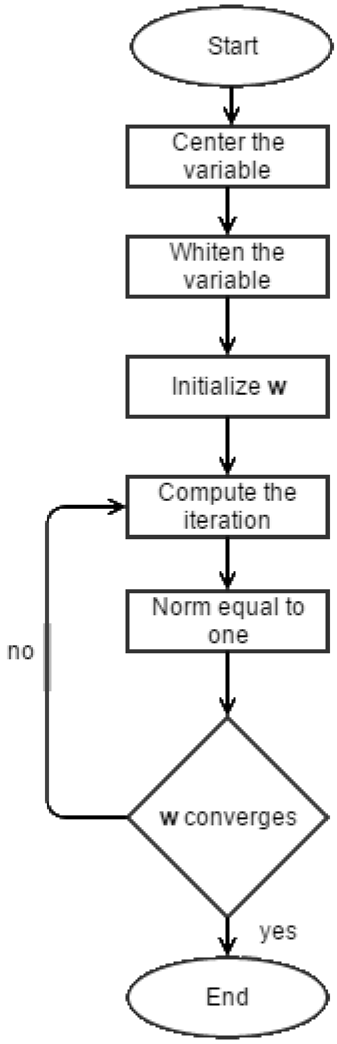


Fig. 3: Flow diagram of the iteration for one component

And the pseudocode of the algorithm is the next one

```
$x \leftarrow x - E\{x\}$
$z \leftarrow Vx$
Initialize w as a random vector
While w doesn’t converge Do
        $w \leftarrow E\{zg(w^T z) - E\{g'(w^T z)\}w\}$
        $w \leftarrow \frac{w}{\|w\|}$
End while
```

Of course, this algorithm only gives us one single component. For reach the goal, we need to calculate the two independent components (the two speakers). We are not able to that with this algorithm. Fortunately, we can use a similar algorithm (using this iteration and negentropy) to solve the problem. This would be studied in the next section.

4.2. Fixed-Point Algorithm for many components

At this point we are able to calculate one single component of any uncorrelated system using negentropy as a measure of *nongaussianity*. But we always will have several components, in our case we have two. We may think in doing the algorithm as many times as components we have, but this will be not correct.

We must remember that the vectors of the un-mixed matrix, $\boldsymbol{W}$, are orthogonal between each others. This is because the space has been whitened, recall that we do that to force the uncorrelatedness between the components. To respect this orthogonality, we must implement the iteration algorithm with an orthogonalization before compute the next iteration.

4.2.1. Orthogonalization with the Gram-Schmidt method

This method was developed by the mathematics Jørgen Pedersen Gram and Erhard Schmidt. They defined an algorithm to build an orthonormal system with independent vectors.

To explain the algorithm mathematically let’s imagine two vectors, $\boldsymbol{v}$ and $\boldsymbol{u}$ with the next projection

$$proj_{\boldsymbol{u}}(\boldsymbol{v}) = \frac{\langle \boldsymbol{v}, \boldsymbol{u} \rangle}{\langle \boldsymbol{u}, \boldsymbol{u} \rangle}\boldsymbol{u} \quad (52)$$

This is the orthogonal projection of $\boldsymbol{v}$ over $\boldsymbol{u}$. Normally we will have a series of vectors $\boldsymbol{v}_1$ ... $\boldsymbol{v}_n$, then (according to Gram and Schmidt) the projections of $\boldsymbol{u}_1$ ... $\boldsymbol{u}_n$ can be calculated as it follows

$$\boldsymbol{u}_1 = \boldsymbol{v}_1 \qquad \boldsymbol{e}_1 = \frac{\boldsymbol{u}_1}{\parallel \boldsymbol{u}_1 \parallel} \quad (53)$$

$$\boldsymbol{u}_2 = \boldsymbol{v}_2 - \frac{\langle \boldsymbol{v}_2, \boldsymbol{u}_1 \rangle}{\langle \boldsymbol{u}_1, \boldsymbol{u}_1 \rangle}\boldsymbol{u}_1 \qquad \boldsymbol{e}_2 = \frac{\boldsymbol{u}_2}{\parallel \boldsymbol{u}_2 \parallel} \quad (54)$$

$$\boldsymbol{u}_3 = \boldsymbol{v}_3 - \frac{\langle \boldsymbol{v}_3, \boldsymbol{u}_1 \rangle}{\langle \boldsymbol{u}_1, \boldsymbol{u}_1 \rangle}\boldsymbol{u}_1 - \frac{\langle \boldsymbol{v}_3, \boldsymbol{u}_2 \rangle}{\langle \boldsymbol{u}_2, \boldsymbol{u}_2 \rangle}\boldsymbol{u}_2 \qquad \boldsymbol{e}_3 = \frac{\boldsymbol{u}_3}{\parallel \boldsymbol{u}_3 \parallel} \quad (55)$$

Where $\boldsymbol{e}$ is the orthonormal vector. Of course, we can generalize this series

$$\boldsymbol{u}_k = \boldsymbol{v}_k - \sum_{j=1}^{k-1} \frac{\langle \boldsymbol{v}_k, \boldsymbol{u}_j \rangle}{\langle \boldsymbol{u}_j, \boldsymbol{u}_j \rangle} \boldsymbol{u}_j \qquad \boldsymbol{e}_k = \frac{\boldsymbol{u}_k}{\parallel \boldsymbol{u}_k \parallel} \quad (56)$$

Now, let's apply this method to our iteration algorithm. In our context, we will have p independent components with their $\boldsymbol{w}_p$ vectors. Imagine that we have calculated any component $\boldsymbol{w}_p$, and the algorithm is calculating the next vector $\boldsymbol{w}_{p+1}$. Then, after each iteration (for the estimation of $\boldsymbol{w}_{p+1}$) the vector has to be orthogonalized. After this iterations, we have the projections $(\boldsymbol{w}_{p+1}^T \boldsymbol{w}_j)\boldsymbol{w}_j$, where j is the previous estimated vector.

Then, the algorithm for the orthogonalization in the FastICA will be the next one

1. $p \leftarrow 1$
2. Initialize $\boldsymbol{w}_p$
3. Apply the iteration algorithm
4. Orthogonalize

$$\boldsymbol{w}_p \leftarrow \boldsymbol{w}_p - \sum_{j=1}^{p-1} (\boldsymbol{w}_p^T \boldsymbol{w}_j)\boldsymbol{w}_j$$

5. $\boldsymbol{w_p} \leftarrow \boldsymbol{w_p} / \parallel \boldsymbol{w_p} \parallel$
6. If $\boldsymbol{w}_p$ hasn't converge, apply the iteration algorithm again
7. $p \leftarrow p + 1$

4.2.2. FastICA Algorithm

We have already said that FastICA is the simplest and versatile algorithm for solve the Blind Source Separation. This method works with the measurement of independence with *nongaussianity*. There are variants of this algorithm, because there are different ways to measure the *gaussianity*.

All the theory and the steps have been explained along this document, but let's make a brief summary. This is our starting point

$$\boldsymbol{X} = \boldsymbol{AS} \ (57)$$

Where the matrix $\boldsymbol{X}$ has n vectors $\boldsymbol{x}_n$. Then we must to center and whiten the data to have a whitened distribution. We will need a loop with many rounds as components we wanted to estimate. In each loop the iteration algorithm will be computed. This algorithm will calculate the vector $\boldsymbol{w}_p$ with iterations until the vector converges. After that the orthogonalization will be done. This process will be repeated until the algorithm have found all the components. At this point we will have the matrix $\boldsymbol{W}$, formed by all the vectors. At the end we can do a simple operation to get back the original signals.

$$\boldsymbol{S} = \boldsymbol{W}\boldsymbol{X} \quad (58)$$

The flow diagram of the FastICA algorithm is the following one

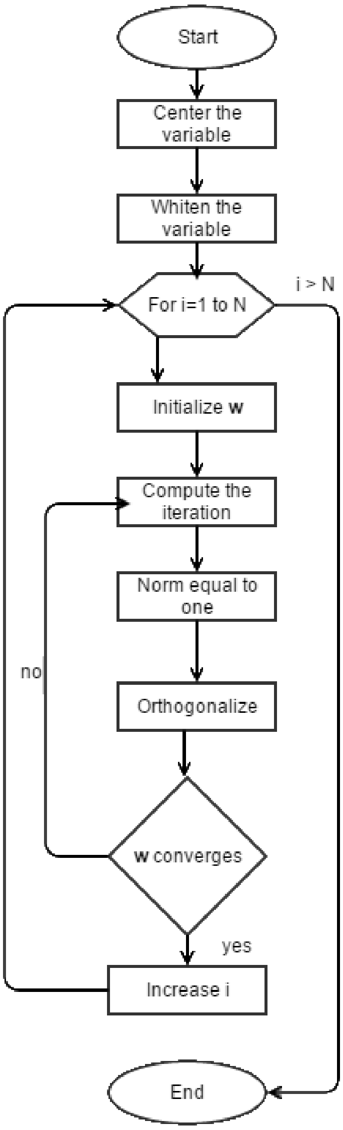
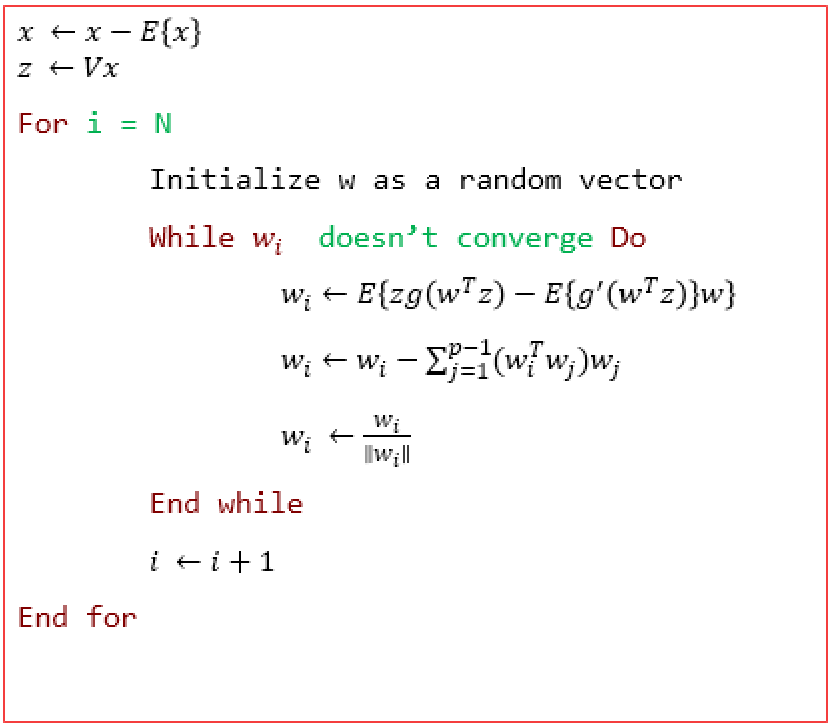


Fig. 4: Flow diagram of the FastICA algorithm for many components

And it can be written with the following pseudocode



```
$x \leftarrow x - E\{x\}$
$z \leftarrow Vx$
For i = N
        Initialize w as a random vector
        While $w_i$  doesn’t converge Do
                $w_i \leftarrow E\{zg(w^T z) - E\{g'(w^T z)\}w\}$
                $w_i \leftarrow w_i - \sum_{j=1}^{p-1}(w_i^T w_j)w_j$
                $w_i \leftarrow \frac{w_i}{\|w_i\|}$
        End while
        $i \leftarrow i + 1$
End for
```

At the end of the computation the original signals will be recovered with the matrix $\boldsymbol{W}$.

5. Analysis & Results

In this section the results will be analyzed after apply the FastICA algorithm to the mixed signals. Then we will be able to compare the recovered signals with the original ones.

5.1. Original Signals

The Cocktail Party Problem is commonly related with speech signals, a lot of people talking at the same time. That’s why speech signals have been selected for the experiments with the FastICA algorithm.

These signals have been subtracted of a radio program called ‘Podcast Reload’, and they are just speech signals. These signals have been cut and treated with Adobe Audition because it allows to see the wave form and its spectrum.

The first signal, $\boldsymbol{s}_1$, is represented in the next wave form.

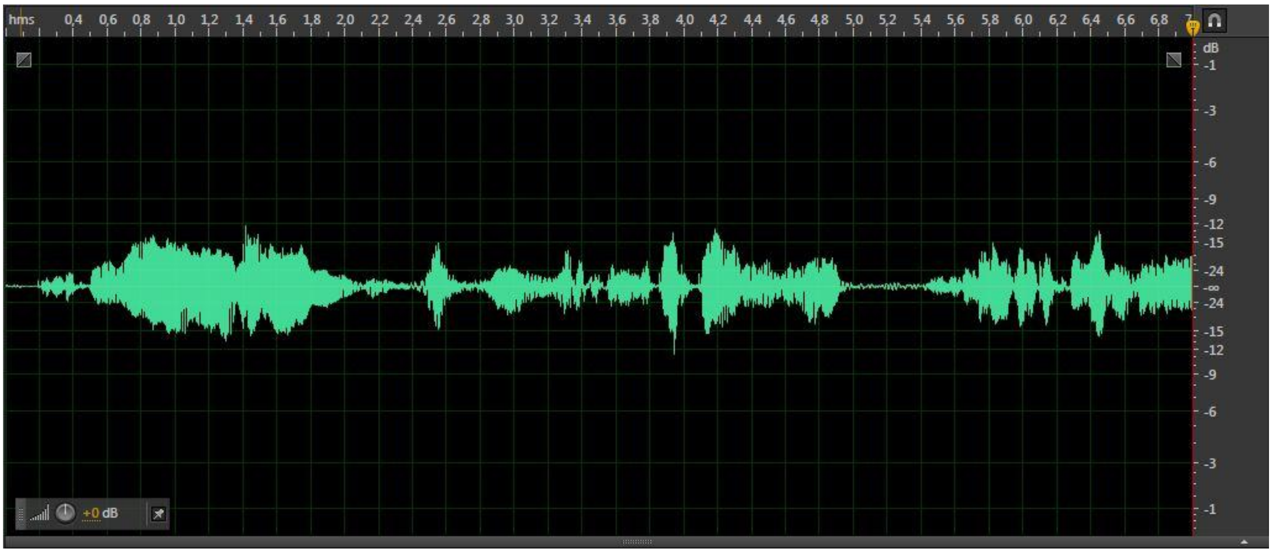


Fig. 5: Wave form of the original signal 1

And its spectrum is the following one

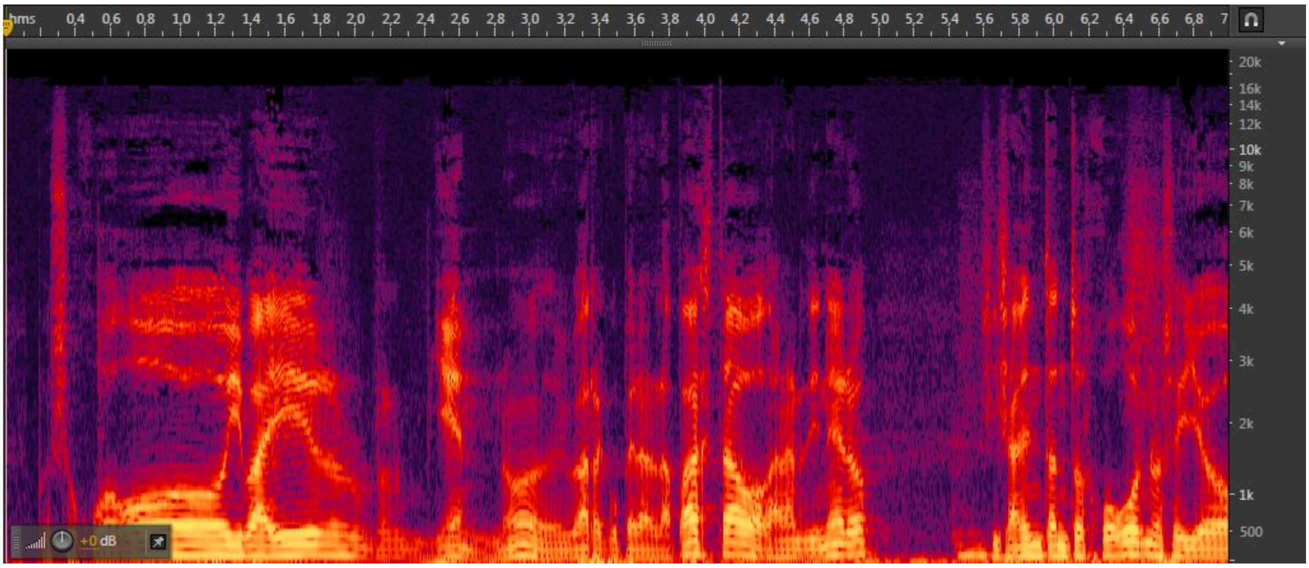


Fig. 6: Spectrum of the signal

As we can see the energy is focused around the low and middle frequencies, despite that the human voice is rich in harmonics that can reach the 3000 or 3500 Hz. Of course, this will depend if a woman or a man is talking. Specifically, this wave form belongs to an adult man.

Secondly we have the other original signal $\boldsymbol{s}_2$ with its wave form

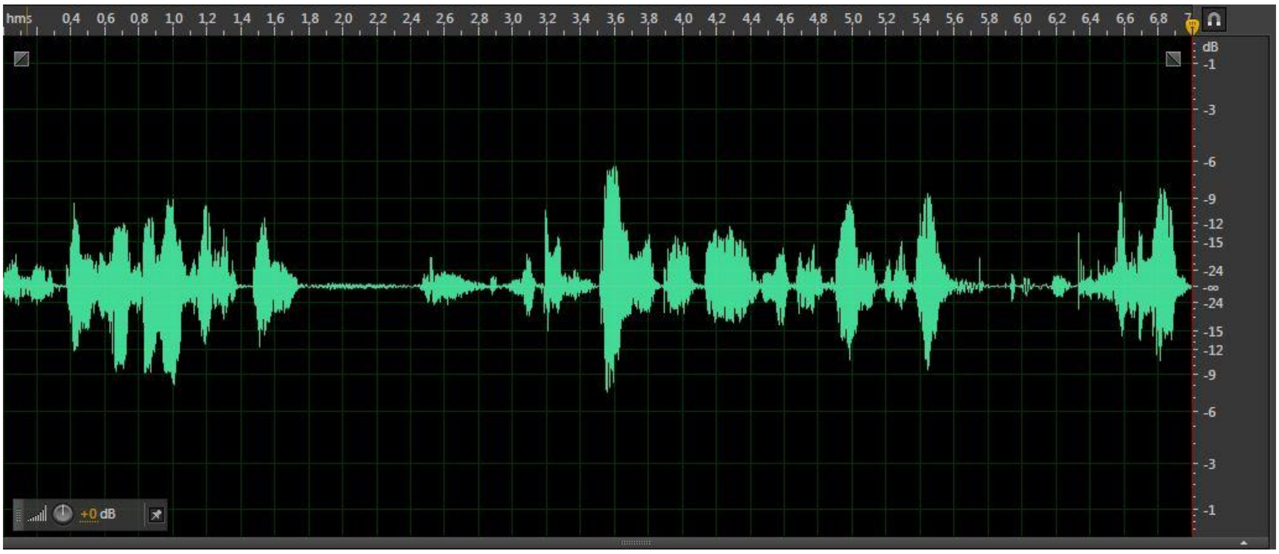


Fig. 7: Wave form of the original signal 2

And its spectrum can be visualized in the next figure.

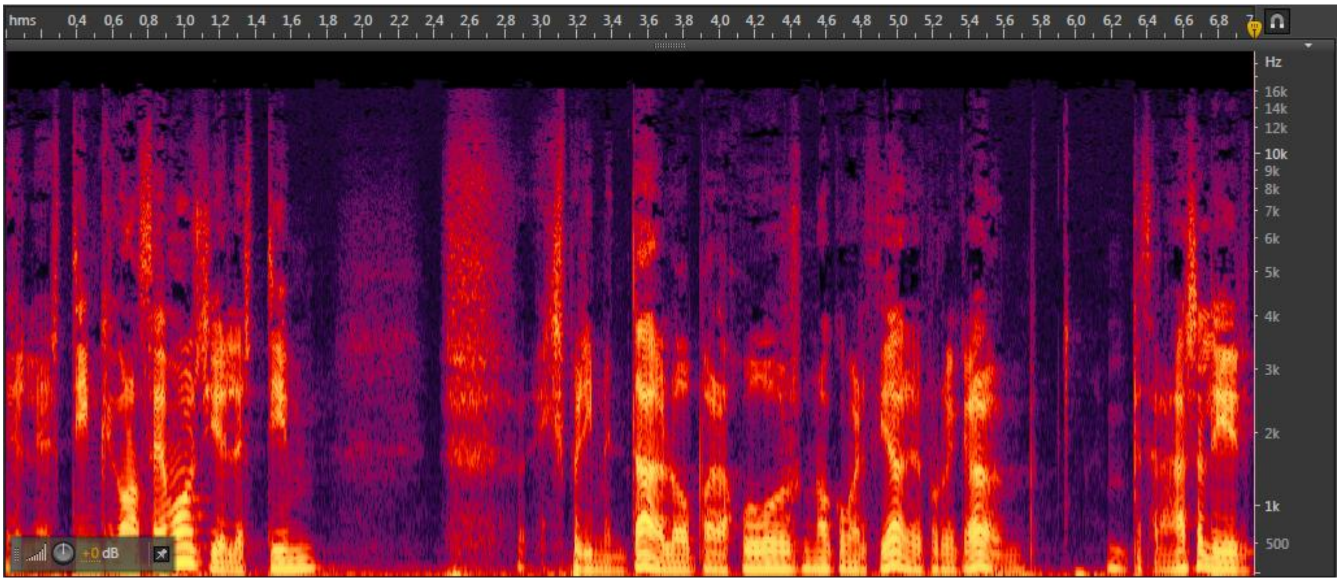


Fig. 7: Spectrum of the signal

Both spectrums are quite similar because they are just speech signal, without music or loud noise.

5.2. Simulation of the room

With the signals selected, the acoustical parameters of the room can be simulated. As has been said before, the simulation can be made with a specific Matlab package. It has a series of functions that allow us to simulate the speakers, the microphones and the frequency response of the room.

The function obtains the frequency response of the room with the parameters of the volume, area and absorption coefficient. Also, it takes account of the distance and inclination of the speakers and the microphones.

It has been chosen a low reverberation time because FastICA is not robust versus high reverberations. So, the reverberation time of the virtual room is 300 ms.

At the end of the process we will have the microphone signals, which are the mix of the two speakers. All the signals are monophonic in a way that an audio signal equals to a vector in the Matlab workspace.

We can see the wave form of the mix signals in the following figures.

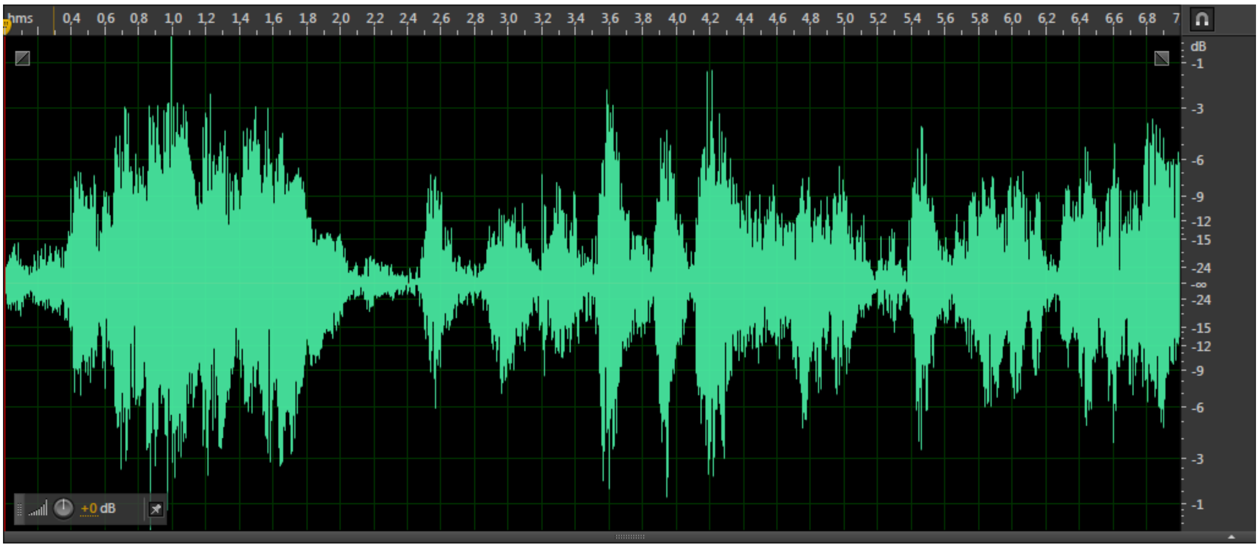


Fig. 8: Signal of the microphone 1

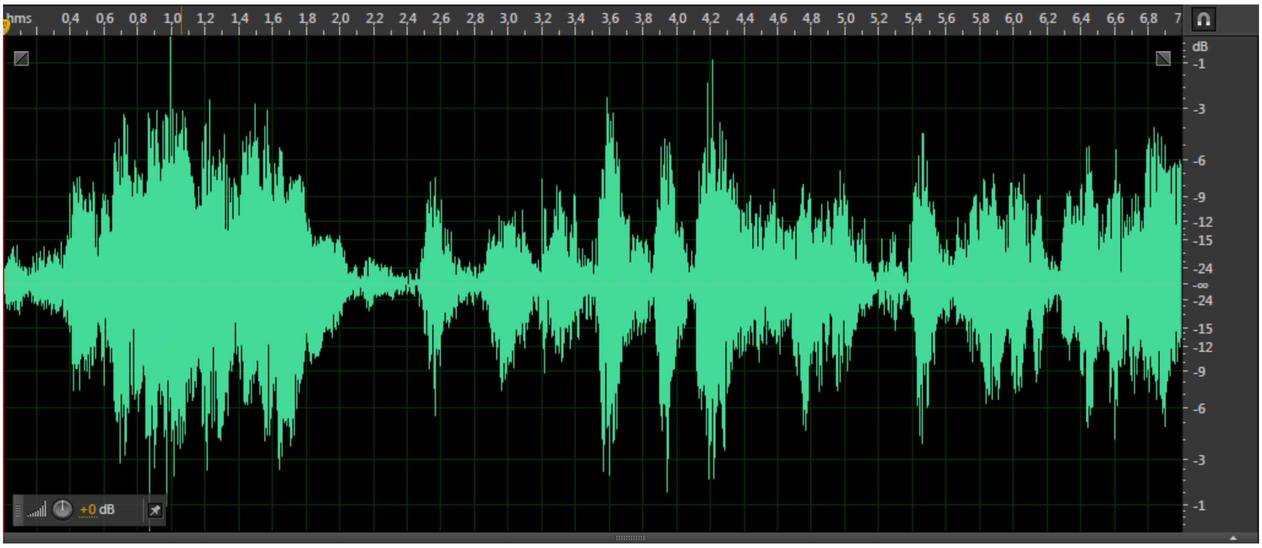


Fig. 9: Signal of the microphone 2

Graphically the signals are quite similar because the microphones are near. These two signals will be the inputs for the FastICA algorithm. And after the execution the mixing matrix and the de-mixed signals will be available.

The results are detailed in the next section.

5.3. Recovered signals

In this section the quality of the recovered signals will be studied. With the mixed signals in the Matlab workspace, the algorithm has been run and we can analyze the characteristics of the obtained signals and compare them with the original ones.

In the next plot the wave form of the recovered signal s1 can be seen.

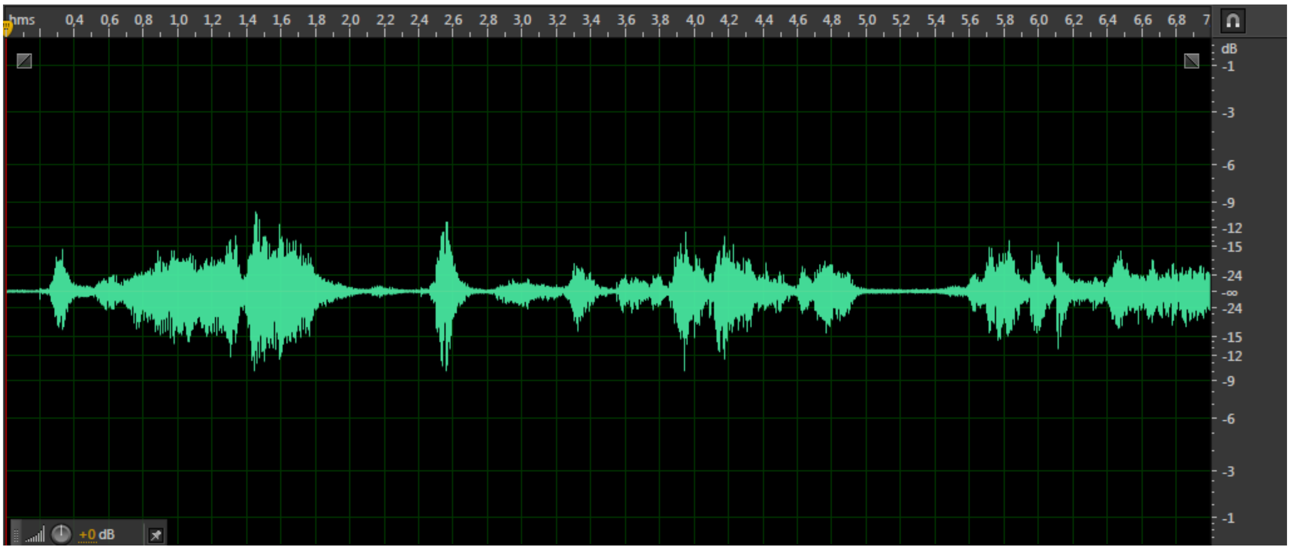


Fig. 10: Wave form of the recovered signal S1

Visually, the wave form is almost the same as the original. Despite that, the signal doesn't sound equal because the problems of the algorithm with reverberation. This can be checked by looking at its spectrum.

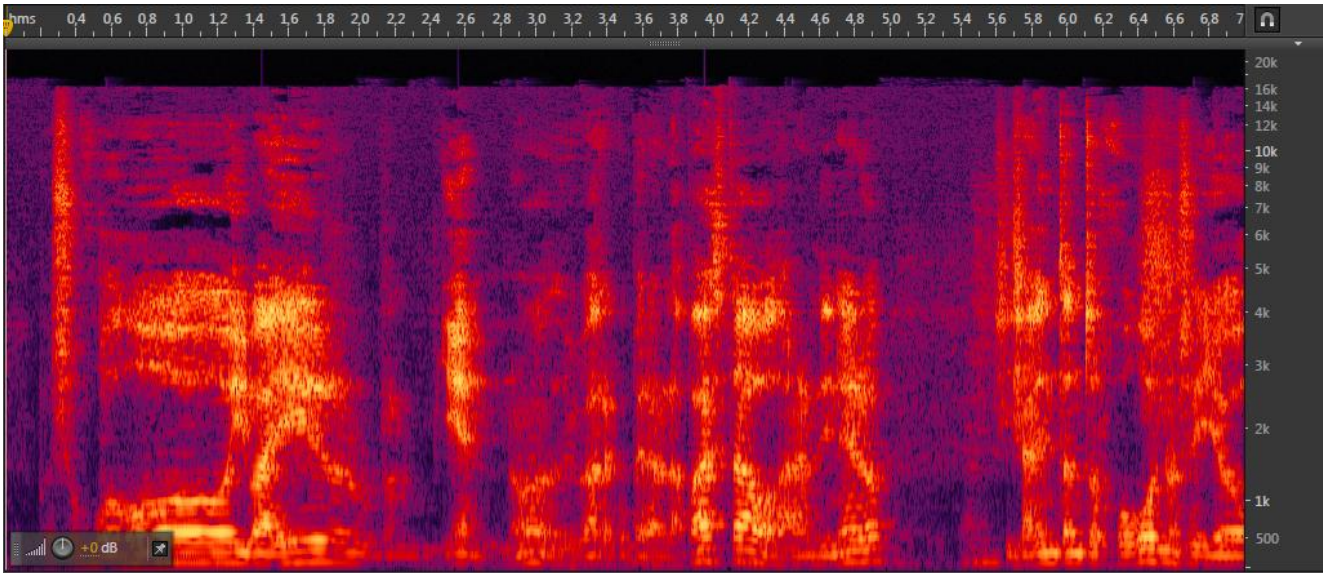


Fig. 11: Spectrum of the recovered signal S1

If we compare it with the Figure. 6 (the spectrum of the original signal s1) we can see tiny differences. In the original spectrum the energy was only focused in the middle frequencies, while in the recovered signal there are much energy in all the frequencies. This is because the noise that has been generated due the reverberation of the room.

In common words, the new signal sounds like hear a voice through a metal pipe. It's a normal side effect in almost all the noise reduction methods.

Next, we will check the second recovered signal.

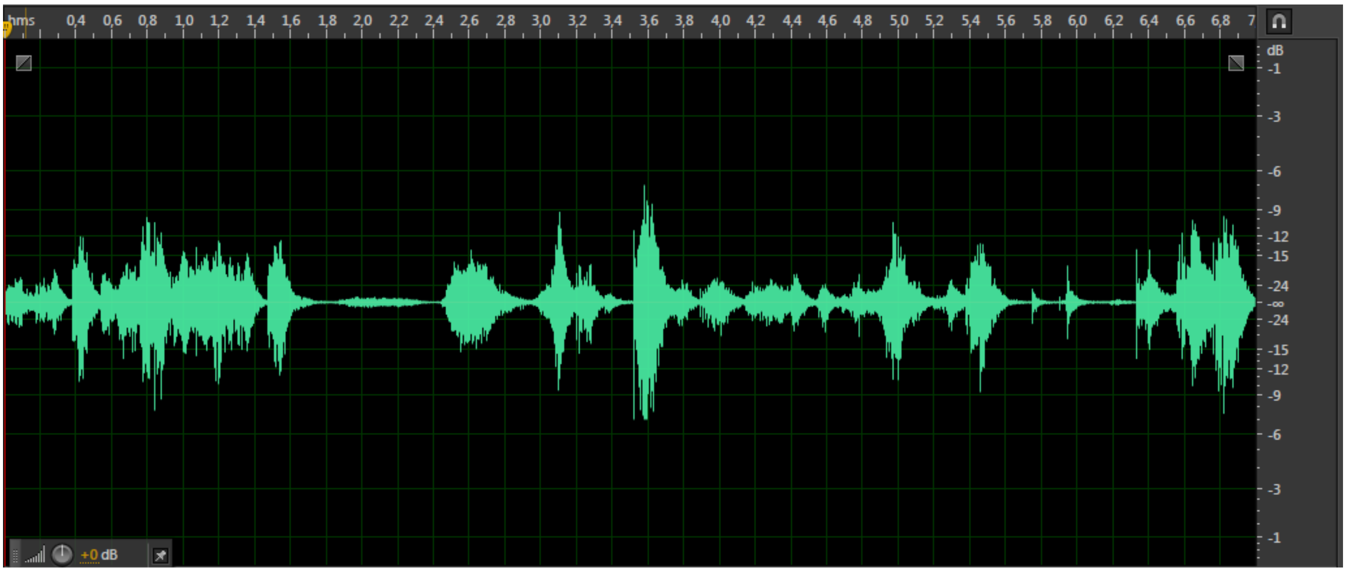


Fig. 12: Wave form of the recovered signal S2

As happen with the signal 1, the wave form of the signal 2 is practically the same as its original. But if we look at the spectrum, the same effect as in the previous signal will be found.

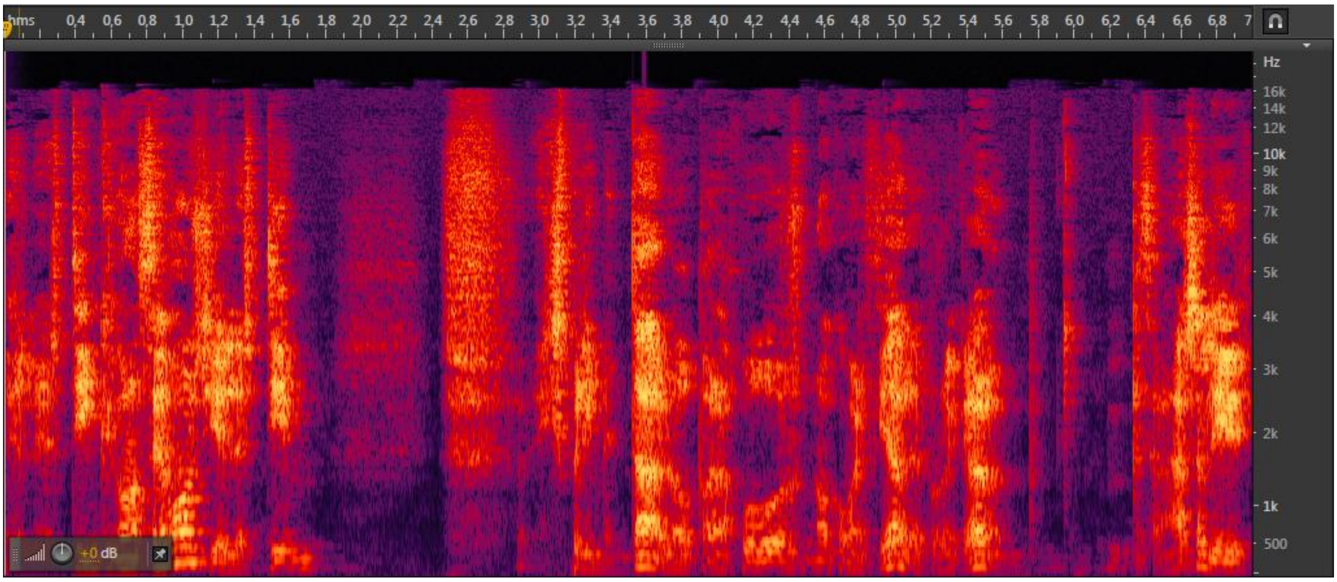


Fig. 13: Spectrum of the recovered signal S2

There is more energy than in the original signal, once again because the reverberation. Despite this phenomenon the results are quite good if we listen to them, the separation is complete, there is nothing mixed in the recovered signals.

6. Conclusion & Future Work

6.1. Conclusion

In this thesis the FastICA algorithm (using negentropy) and the results with a virtual scenario have been studied in detail. From those results we can approximate some conclusions.

The first assumption is about the FastICA algorithm. As it has been said before, it is the simplest algorithm to perform the Blind Source Separation. The relation between simplicity and quality is quite good. Most of the others ICA algorithms are computationally more complicated, but the result is of course better.

Possibly the best thing about the ICA techniques is its versatility. Unlike other methods, with ICA any kind of signals can be computed. The only requirement that the signals must have is statistical independence. The type and characteristics of them don’t matter. This is why there are a lot of applications and fields of study related with Independent Component Analysis.

6.2. Other Applications

Until now only the audio separation has been explained, but the amount of applications is huge. The brain imaging applications are really common too. Normally, the researchers apply ICA to electroencephalography and magnetoencephalography (EEG and MEG). In this kind of biomedical techniques a lot of mixed signals are measured and there are needed methods like ICA.

Telecommunications is also a common field of application. Specifically the CDMA communications may work with Blind Source Separation. CDMA are normally use for radio broadcasting or for mobile communications, among other applications.

There are other minority applications such as financial applications or face recognition.

6.3. Future Work

At this point is clear that FastICA is the basic algorithm, but if we want better results and advanced techniques other algorithms are needed. There is a lot of future work for the investigation and implementation of ICA algorithms. The next step will be study the same scenario of this thesis but taking account of the noise.

The noisy ICA is actually the model that we would find in real-life scenarios. All the measurements have some kind of noise: because the physical noise on the microphones, the background noise of the room, etc. That's why is important to study theoretically and mathematically the noisy ICA.

The simplest solution to this consist in using noise reduction techniques before apply the ICA algorithm. But this isn't always possible, because sometimes the noise is too loud or invulnerable to noise reduction techniques. Then a new model for ICA is needed, and this is quite more complicated that the one which has been explained in this thesis.

Also, there are another kind of approaches to Independent Component Analysis. To continue this field of study, the techniques that work in the frequency domain must be studied. There are methods such as the one described in the paper '*Convolutive BSS of Short Mixtures by ICA Recursively Regularized Across Frequencies*'. The method that is explained here is really robust versus reverberation and noise, is a great algorithm for Blind Source Separation. Of course the complexity is high, because it works with Fourier analysis in frequency domain.

Bibliography

E. Colin Cherry, *Some Experiments on the Recognition of Speech, with One and with Two Ears,* The Journal of the Acoustical Society of America, volume 25, number 5; September 1953.

Francesto Nesta, Piergiorgio Svaizer, Maurizio Omologo, *Convolutive BSS of Short Mixtures by ICA Recursively Regularized Across Frequencies*, IEEE Transactions on Audio, Speech, and Language Processing, Vol. 19, number 3; March 2011

E. Lehmann and A. Johansson, *Prediction of energy decay in room impulse responses simulated with an image-source model*, Journal of the Acoustical Society of America, vol. 124(1); July 2008.

www.eric-lehmann.com

Aapo Hyvärinen, Juha Karhunen, Erkki Oja, *Introduction*, Independent Component Analysis, chapter 1 ; 2001

Aapo Hyvärinen, Juha Karhunen, Erkki Oja, *What is Independent Component Analysis?* Independent Component Analysis, chapter 7 ; 2001

D.N. Joanes, C.A. Gill, *Comparing Measures of Sample Skewness and Kurtosis*, Journal of the Royal Statistical Society, Vol. 47, issue 1 ; January 2002

Aapo Hyvärinen, Juha Karhunen, Erkki Oja, *ICA by Maximization of Nongaussianity*, Independent Component Analysis, chapter 8 ; 2001

Claude E. Shannon, *A Mathematical Theory of Communication*, Bell System Technical Journal, Vol. 27 ; October 1948

Aapo Hyvärinen, Juha Karhunen, Erkki Oja, *Information Theory*, Independent Component Analysis, chapter 5 ; 2001

Ward Cheney, David Kincaid, *Inner-Product Vector Spaces*, Linear Algebra: Theory and Applications, chapter 7 ; 2009

Vicente Domingo Estruch Fuster, Valentín Gregori, Almanzor Sapena Piera, *Variables Aleatorias*, Lecciones de Estadística, chapter 4 ; 2010