TREBALL FINAL DE MASTEAM

Title: Speech/music audio classification for publicity insertion and DRM

Author: Fausto Gil Moreno

Director: Francesc Tarrés Ruiz

Date: February 7th, 2018

Overview

The goal of this project is to develop, implement and optimize an existing method called Continuous Frequency Activation (CFA). The aim is to try to solve the problem that exists when advertising in randomly introduced in TV programmes/films/audio podcasts/etc. that can generate discomfort in the viewer.

The basic idea is to avoid introducing adverts in the middle of a conversation. The final criteria will be selected taking into account metadata of video (change of plane, scene, fade-out, etc.) and audio (voice, music).

To do that, we have developed and algorithm capable of discriminate between music and voice. This algorithm has been developed exclusively for this purpose and does not require base or date training to be trained.

Previously to the creation of the algorithm, different existent methods of discrimination between music and voice have been studied and their pros and cons have been analysed. After performing the study, the method that has been selected is The Continuous Frequency Activation (CFA). CFA is one of the methods with better statistic results and it is not necessary to obtain large data bases for its training.

The implementation of this algorithm has been performed using MATLAB®. Data base have been used in the realization of the trials, using five different musical style: classic music, Blues, electronical music, Jazz and Speech.

The audio files from each different music style have been edited using the software called Audacity®.

After performing all the tests, it can be said that the developed algorithm works correctly and it is able to discern music from voice in a very high percentage of cases (97.55%).

With the results obtained after the trials, it can be said that this method could be used by companies that are involved in the fields of media, television (Antena 3, Telecinco, etc.) and/or audio podcast. The goal is to automatically introduce publicity in audio podcast format at the most appropriate moment.

I want to dedicate this project to all my family and to my supervisor Francesc Tarrés who I consider to be not just a supervisor but a good friend as well.

Also, to my grandfather that although he is not with us anymore I feel he is close to me and will always be with me in my heart.

And specially to my 7 months old daughter Olivia that is the most beautiful and sweet girl of the whole world.

INDEX

INTRODUCTION

This project is about music/voice classification, the different methods in which it can be used and possible applications of this tool.

Classification of sound in categories such as speech or music is widely known as a relevant topic and basic requirement in multimedia recovery systems.

Being able to automatically separate music from speech is a very interesting topic as there are many possibilities and a wide range of fields in which it can be applied.

Because of that, an exhaustive analysis has been performed. Different methods have been used, each of them with specific characteristics, to evaluate how efficient there are in discriminating music from speech and vice versa.

Some of the methods used for classification of music/speech are: (High Zero-Crossing Rater Ratio (HZCRR) and Low Short-Time Rate (LSTER), etc. Each of them have strong and weak points. The first chapter of this paper will analyse in detail each of these methods to gain a better understanding of their main characteristics.

One of the most usual problems when trying to identify speech audio or video file is to be able to separate speech from music. That may happen when trying to identify speech from a musical background or from a film, TV series, etc.

These tools can be applied in a wide and interesting range of situations. Some examples being: voice segmentation to send Speech-to-text, television or radio products, films, TV series, storage and processing. One of the uses of these tools in which radio/tv media could be interested would be to use it to detect the best moments to put advertising, that would result in an increase of quality and rating of a specific company.

Recently, the way in which advertising has been introduced in some of the most popular videos from Youtubers consists on the introduction of advertising at random. The video is temporally stopped at any point for a few seconds and the advert is introduced. In that situation, the viewer may stop watching as he may lose interest. This type of advertising can be especially disruptive as the youtuber explanation can be interrupted at any point. Using the algorithm it would be possible to select the best moment to introduce advertising, so the viewer would probably wait until advertising has finished and will continue watching the video as the advertising has been introduced a moment in which the disruption of the explanation is less.

This tool can also be very useful to recognize moments of silence in an audio file. That will allow us to know if a person is talking or in silence, so the disruption is made in the most appropriate moment. That also, will save a lot of memory when saving audio files.

Another possible application will be to calculate how many minutes a radio station is playing music, so society of authors can claim payment for use of their music.

Also, it could be used to eliminate the musical background in a video and kept just the speech. There are several situations in which this algorithm could be applied.

The aim of this project is to try to solve these problems. In order to do that we will try to implement, adapt and optimize through an existent method called Continuous Frequency Activation (CFA) with the use of the MATLAB® software programming. That has allowed us to perform identifications with a high level of success rate.

Continuous Frequency Activation (CFA) is a tool especially designed for music detection.

Because of that, we have chosen to focus in that tool for this project. We will do an analysis and will do different tests and comparison with other tools previously mentioned, etc.

An extensive explanation of what has already been mentioned in this introduction will be covered in the next chapters. We will start by showing a perspective of the different methods of analysis that can be used for music and speech discrimination, we will evaluate how they work, what are their characteristics and differences and why we finally chose the CFA application.

Then, CFA will be analysed in detail. In order to do that, block diagrams will be evaluated, the most important modules will be identified, and the mathematic analysis will be studied. Sign samples of each of the different stages of analysis will be studied.

In order to do test this method it has been necessary to create a data base, that will be explained in chapter 3.

We will also explain the different development tools that has been used.

In that chapter, we will also explain the different developer tools used in this project.

Finally, we will explain how the different modules have been applied, will discuss results of different tests in percentages, comparison, etc.

We will also evaluate the final conclusions obtained after results from the tests performed and will give some solutions of how to improve the system, how to implement in programming language C++, how difficult it would be, etc.

Now that we have placed ourselves in to a position, we understand what is this project about and we know the problem we want to solve, let's start by looking at the different methods of analysis.

CHAPTER 1. FEATURE BASED STRATEGIES FOR MUSIC/SPEECH CLASSIFICATION

As it has been mentioned at the end of the introduction, in this first chapter we will examine the different methods[1] of analysis used in the classification of audio in the type of music/speech.

1.1. High Zero-Crossing Rate Ratio (HZCRR)

Zero-crossing rate (ZCR) is proved to be useful in characterizing different audio signals. It has been popularly used in speech/music classification algorithms. HZCRR is defined as the ratio of the number of frames whose ZCR are above 1.5-fold average zero-crossing rate in an 1-s window, as

$$HZCRR = \frac{1}{2N} \sum_{n=0}^{N-1} [\mathrm{sgn}(ZCR(n) - 1.5\ avZCR) + 1] \qquad \textbf{(1.1)}$$

Where *n* is the frame index, ZCR(*n*) is the zero-crossing rate at the *n*th frame, *N* is the total number of frames, *av*ZCR is the average ZCR in a 1-s window; and sgn[.] is a sign function, respectively.

In general, speech signals are composed of alternating voiced sounds and unvoiced sounds in the syllable rate, while music signals do not have this kind of structure. Hence, for speech signal, its variation of zero-crossing rates (or HZCRR) will be in general greater than that of music, as shown in Figure 1.1.

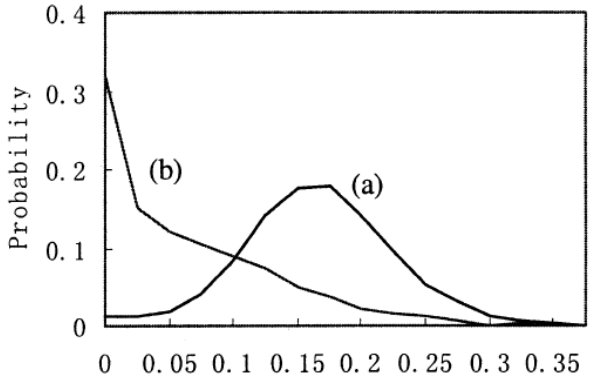


Fig. 1.1 Probability Distribution curves of HZCRR. (a) Speech and (b) music.

The above figure shows the statistics of parameters of speech/music. After evaluation the curves in the figure it can be concluded that it would be possible to use a threshold of 0.1 for discrimination between music and voice.

1.2. Low Short-Time Energy Ratio (LSTER)

LSTER is defined as the ratio of the number of frames whose STE are less than 0.5 time of average short-time energy in a 1-s window, as the following:

$$LSTER = \frac{1}{2N} \sum_{n=0}^{N-1} [\mathrm{sgn}(0.5\ avSTE - STE(n)) + 1] \qquad \textbf{(1.2)}$$

Where *N* is the total number of frames, STE (*n*) is the short-time energy at the *n*th frame, and *av*STE is the average STE in a 1-s window. *LSTER* is an effective feature, especially for discriminating speech and music signals. In general, there are more silence frames in speech than in music; as a result, the *LSTER* measure of speech will be much higher than that of music. This can be seen clearly from the probability distribution curves of *LSTER* for speech and music signals, as illustrated in the Figure 1.2. It is shown that *LSTER* value of speech is around 0.15 to 0.5, while that of music is mostly less than 0.15. *LSTER* is a good discriminator between speech and music.

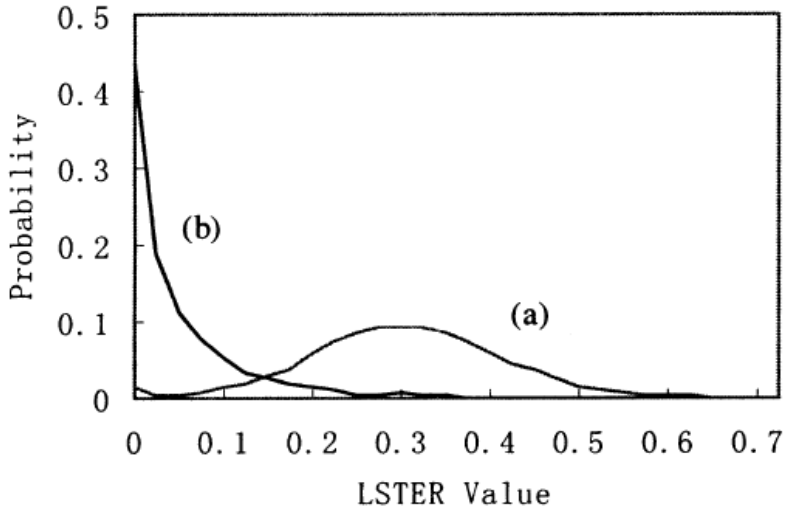


Fig. 1.2 Probability Distribution curves of LSTER. (a) Speech and (b) music.

1.3. Spectrum Flux (SF)

Spectrum flux (SF) is defined as the average variation value of spectrum between the adjacent two frames in a 1-s window

$$SF = \frac{1}{(N-1)(K-1)} \sum_{n=1}^{N-1} \sum_{k=1}^{K-1} \cdot[\log(A(n,\, k) + \delta) - \log(A(n-1,\, k) + \delta)]^2 \quad \textbf{(1.3)}$$

Where A (*n,k*) is the discrete Fourier transform of the *n*th frame of input signal

$$A(n,\, k) = \left| \sum_{m=-\infty}^{\infty} x(m) w(nL - m) e^{-j(2\pi/L)\, km} \right| \quad \textbf{(1.4)}$$

and *x*(*m*) is the original audio data, *w*(*m*) is the window function, *L* is the window length, *K* is the order of DFT, *N* is the total number of frames and δ is a very small value to avoid calculation overflow. SF is a good feature to discriminate speech, environmental sound and music.

Figure 1.3 shows an example of spectrum flux of speech, music and environment sound. The speech segment is from 0 to 200 s, the music segment is from 201 to 350 s and the environment sound is from 351 to 450 s.

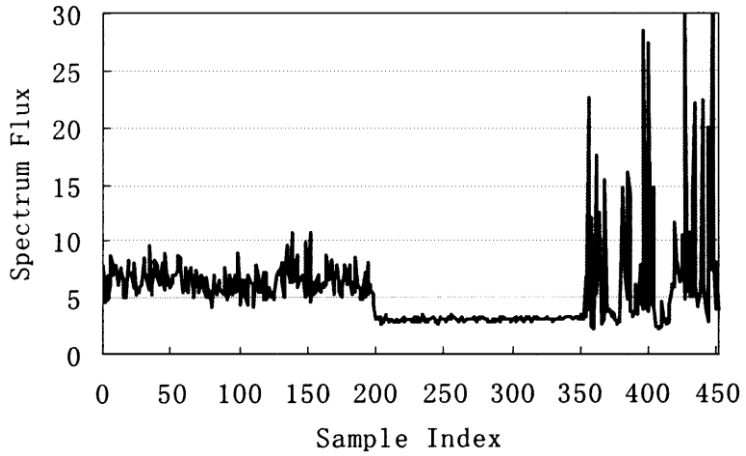


Fig. 1.3 Spectrum flux curve (0–200 s is speech, 201–350 s is music, and 351–450 s is environment sound).

It is important to notice the following:

Music → very predictable → small values

Speech → average values

Noise → abrupt changes in the whole spectrum → big values

1.4. LSP Linear Spectral Pairs

Linear spectral pairs (*LSPs*) are derived from linear predictive coefficients (LPC). Previous researches have shown that *LSP* has explicit difference in each audio class, and that is the cause why they are used in speech recognition. It is also found that *LSP* is more robust in the noisy environment. *K*–L distance is used to measure the *LSP* dissimilarity between two 1-s audio clips

$$D = \frac{1}{2} tr\left[\left(\boldsymbol{C}_{LSP} - \boldsymbol{C}_{SP}\right)\left(\boldsymbol{C}_{SP}^{-1} - \boldsymbol{C}_{LSP}^{-1}\right)\right] + \frac{1}{2} tr\left[\left(\boldsymbol{C}_{SP}^{-1} + \boldsymbol{C}_{LSP}^{-1}\right)\left(\boldsymbol{u}_{LSP} - \boldsymbol{u}_{SP}\right)\left(\boldsymbol{u}_{LSP} - \boldsymbol{u}_{SP}\right)^{T}\right] \quad \textbf{(1.5)}$$

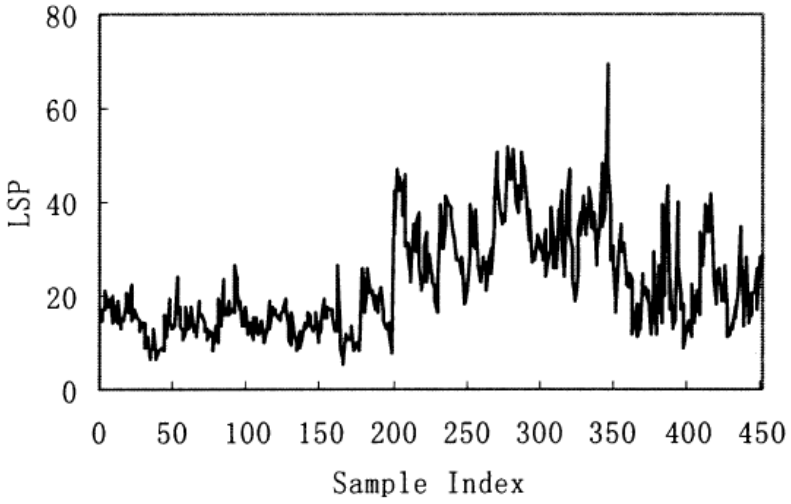


Fig. 1.4 LSP curve (0-200s is speech, 201-350s is music and 351-450s is noisy speech)

Where C_{LSP} and C_{SP} are the estimated *LSP* covariance matrices, u_{LSP} and u_{SP} are the estimated mean vectors, from two audio clips respectively. In real implementation, ten-order *LSP* is extracted from each frame and then the covariance and mean are estimated from each audio clip.

1.5. Band Periodicity (BP)

Band periodicity (*BP*) is defined as the periodicity of a subband. It can be derived by subband correlation analysis. The periodicity property of each subband is represented by the máximum local peak of the normalized correlation function. For example, for a sine wave, its *BP* is 1; but for white noise, its *BP* is 0.

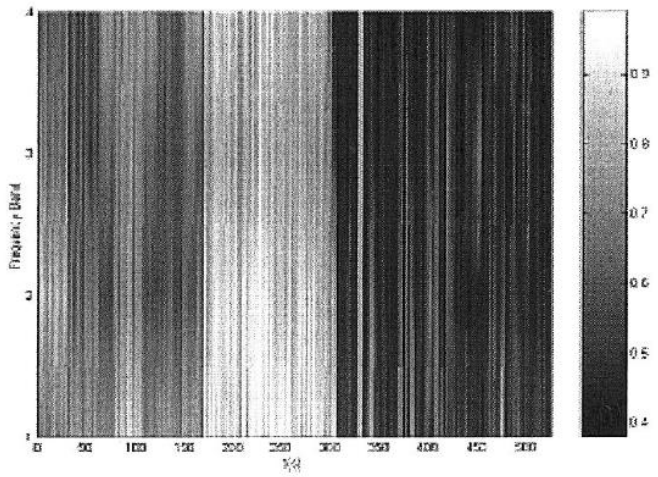


Fig. 1.5 Band periodicity of an example audio segment

The normalized correlation function is calculated from the current frame and previous frame

$$r_{i,j}(k) = \frac{\sum_{m=0}^{M-1} s(m-k)s(m)}{\sqrt{\sum_{m=0}^{M-1} s^2(m-k)}\sqrt{\sum_{m=0}^{M-1} s^2(m)}} \qquad \textbf{(1.6)}$$

Where $r_{i,j}(k)$ is the normalized correlation function; *i* is the band index, and *j* is the frame index. *s*(*n*) is the subband digital signal of current frame and previous frame, when *n*>=0, the data is from the current frame; otherwise, the data is obtained from the previous frame. *M* is the total length of a frame. The maximum local peak as $r_{i,j}(Kp)$ where *Kp* is the index of the maximum local peak, *i* is the band index and *j* is the frame index. That is, $r_{i,j}(Kp)$ is band periodicity of the *i*th sub-band of the *j*th frame. Thus, the band periodicity is calculated as

$$bp_i = \frac{1}{N}\sum_{j=1}^{N} r_{i,j}(k_p) \qquad i = 1, \ldots 4 \qquad \textbf{(1.7)}$$

Where b_{pi} is the band periodicity of *i*th sub-band, *N* is the total frame number in one audio clip. Figure 1.5 shows an example of band periodicity comparison between music and environment sounds. The music segment in the example is

from 0 to 300 s, while the remaining part is environment sounds. The vertical axis represents different frequency sub-bands. It is observed that the band periodicities of music are in general much higher than those of environment sound. This is because music is more harmonic while environment sound is more random. Band Periodicity is an effective feature in music/environment sound discrimination.

To show clearly the discrimination power of this feature, a probability distribution curve of band periodicity in the first subband for environment sound and music is illustrated in the Figure 1.6. From Figure 1.6, it can be obviously seen that the center of bp_1 value of environment sound is around 0.5, while bp_1 value of music is around 0.8, there is a considerable difference between them.

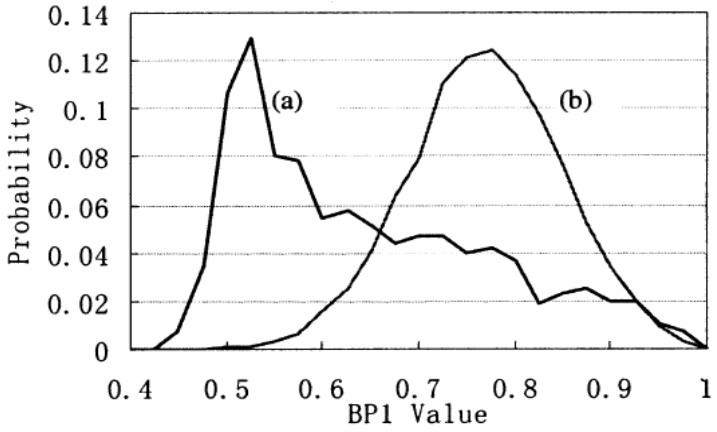


Fig. 1.6 Probability distribution curves of *BP1*. (a) Background sound and (b) music.

The periodicity of the first two bands bp_1 and bp_2 and the sum of the four bands' periodicity, *bp*Sum, are used to discriminate music and environment sound.

1.6. Noise Frame Ratio (NFR)

Noise frame ratio (NFR) is defined as the ratio of noise frames in a given audio clip. A frame is considered as a noise frame if the maximum local peak of its normalized correlation function is lower than a preset threshold. The NFR value of noise-like environment sound is higher than that of music.

Figure 1.7 shows the probability distribution curves of NFR for music and environment sounds from our audio database. For music, almost no NFR value is above 0.3; however, for environment sound, the portion of NFR values that are

higher than 0.3 is much higher. NFR really depends on how noisy the signal is. Data shows some environment sound is more noise-like.

The optimal threshold is the intersection between both curves. That is illustrated in the chart below, although doing an estimation was not a trivial job.

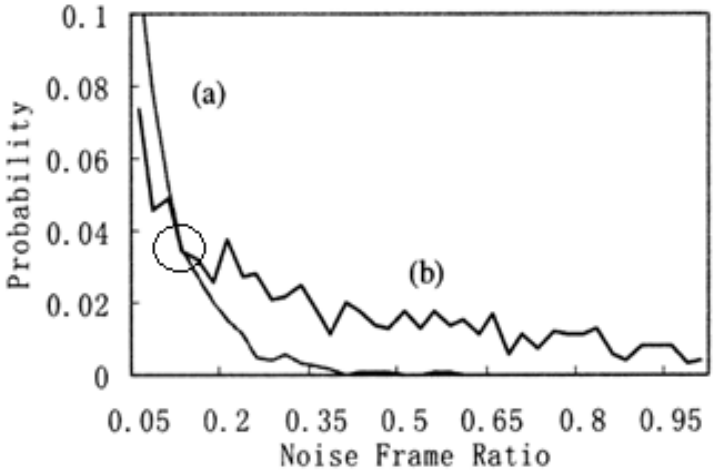


Fig. 1.7 Probability Distribution curves of NFR (a) Music and (b) environament sound

1.7. Mel Frequency Cepstrum Coefficients (MFCC)

Mel Frequency Cepstrum Coefficients are a compact representation of the spectral envelope of a frame. After a non-linear mapping onto the Mel-frequency scale, to better approximate the frequency resolution of the human ear, the envelope of the log-spectrum is compactly represented by the first few coefficients after a DCT compression. MFCCs are well-known for capturing timbral aspects of short audio frames and are widely used for music/speech classification, speaker recognition, etc.

After analysing the different methods and their advantages and disadvantages it has been decided that the method that will be used in the development of this project will be the Continuous Frequency Activation (CFA). This method is the most suitable according to what we want to develop because it has been specifically design for music detection being more precise than other approaches in the identification of music segments in audio transmission. As this project will be focus in the CFA method, it will be studied, and the next chapter will be entirely dedicated to its analysis.

CHAPTER 2. CONTINUOUS FREQUENCY ACTIVATION METHOD IMPLEMENTATION

In general terms, music usually has more stationary parts that speech resulting in perceptible horizontal bars in the representation of the spectrogram of the audio signal. These horizontal bars in the spectrogram are continuous activation of specific frequencies and are usually a consequence of sustained musical tones.

By doing a more exhaustive analysis, focusing in absolute values of the energy of the spectrogram has a counterproductive effect. This is because the horizontal bars can be quite soft and the absolute values of the sounds in the foreground will have a stronger effect.

Figure 2.1 shows the spectrogram of an audio file that contains music and voice alternately. It can be observed that when the bars are in horizontal direction they represent music and when they are vertical direction they represent speech.

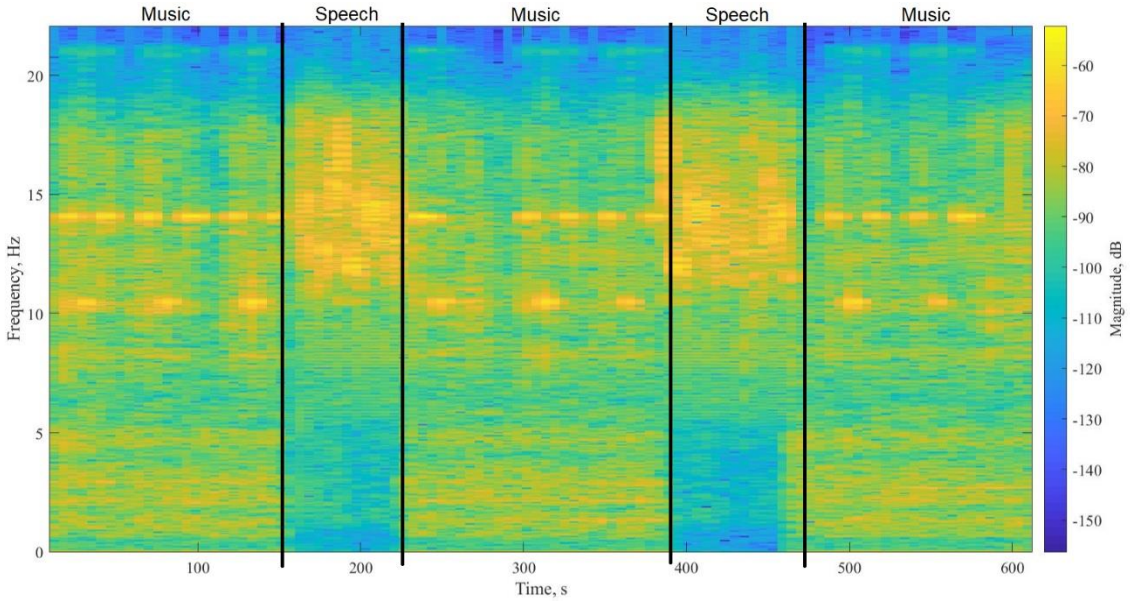


Fig. 2.1 Spectrogram of the signal

This chapter proposes and algorithm to improve the reliability of activation of continuous frequency. This improvement will happen even if there are other audio signals present simultaneously. In the following lines, we will see its development and implementation.

2.1. Development and programming of the algorithm

The following blocks diagram shows the steps in which the algorithm is divided:

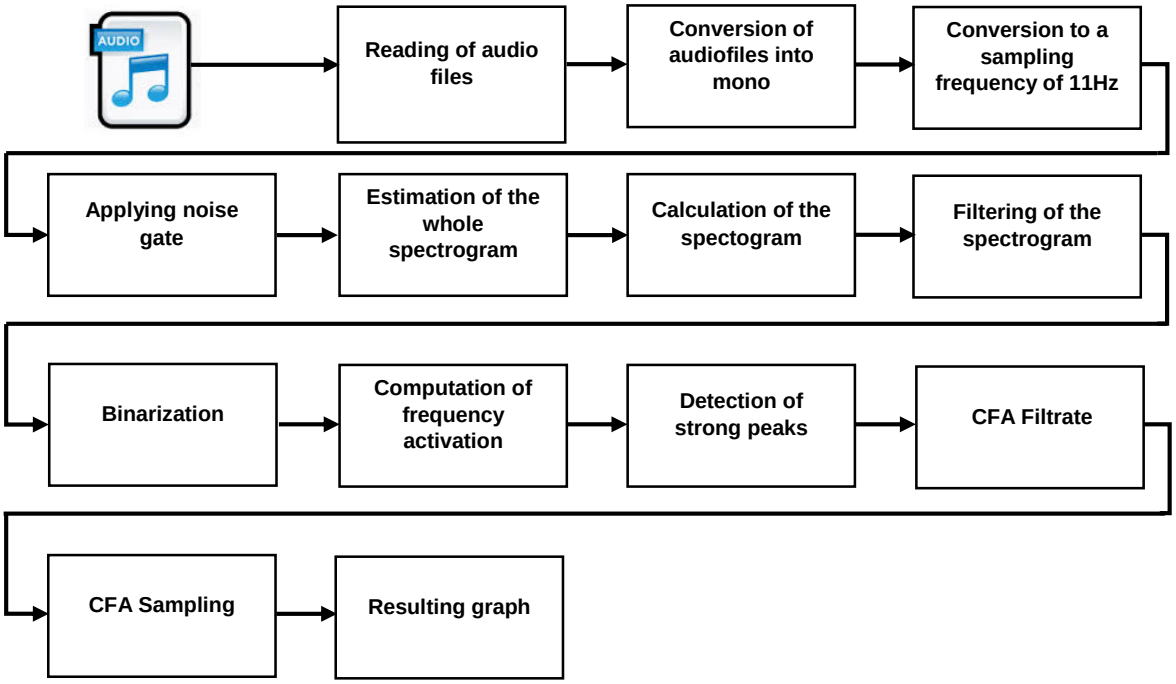


Fig 2.2 Blocks diagram of the algorithm for the analyis of an audio signal

To show this method and to get a better understanding of it, an audio signal sampling of a duration of 106 blocks ('test.wav') has been used. All the graphs of this chapter are representative of this example.

2.1.1. Step one. Reading of audio files.

The first thing that should be done is reading the audio file that is we want to analyse. To do that, the command *audioread* will be used. See the following example:

[ys,fss]=audioread('test.wav');

Where *ys* is the output of array (5526171 files and 2 columns), *fss* is the frequency (44100 Hz) and 'test.wav' is the audio file. It is important to notice that this command is capable of reading .wav, .ogg, .flac, .au, .aiff, .aif, .aifc for all platforms and .mp3, .m4a, .mp4 for Windows® 7 (or later), Macintosh and Linux®.

If the file to be analysed is not in one of these formats, it should be transformed to one of these extensions to prevent problems with *audioread.*

2.1.2. Step two. File conversion from audio into mono.

In this second step, the original file (stereo) will be converted into mono. To be analysed, the file must be in stereo. The useful information to be analysed is duplicated for the two channels and it will be enough with obtaining an average of information of each channel. Considering *ys* as input signal and *y1* as the resulting output signal, the programming would be as follows:

```
y1=0.5*ys(:,1)+0.5*ys(:,2);
```

In our case, the value obtained for *y1* corresponds to a vector of 5526171 positions. This is the length of sampling of the audio signal that has been taking as example.

2.1.3. Step three. Conversion of frequency sampling of 11Hz.

The original signal of 44100 Hz is transformed into a frequency sampling of 11025 Hz using the Nyquist theorem. The value of 11025 Hz is sufficient for determine if the system has detected music or speech, reducing in that way the computational load. To do that, the command *decimate* that can be found at MATLAB® is used. In the next example, we will see how these command works.

```
y = decimate(x,r)
```

In our example, using as input signal the result obtained in the previous step *y1* and using *yd* as the output resulting signal, the programming would be as follows:

```
yd = decimate(y1,4);
```

2.1.4. Step four. Applying a noise gate.

In this step, we will apply the noise gate. Noise gate is a dynamic signal process designed to eliminate sounds during pauses. The function of the noise gate is to avoid the transmission of any signal that do not exceed a predetermined threshold. The programming would be as follows:

```
NoiseGateLevel=0.005;
ygated=yd.*(0.5*(1+sign(abs(yd)-NoiseGateLevel)));
yd=ygated;
```

2.1.5. Step five. Estimation of the whole spectrogram.

In this fifth step we will make the estimation of the whole spectrogram. To make the estimation, we have used the Hann window. In order to generate a Hann window, the following formula should be applied:

$$w(n) = 0.5\left(1 - \cos\left(2\pi \frac{n}{N}\right)\right), \quad 0 \le n \le N \tag{2.1}$$

Where L = N + 1 lenght of the window.

In our case, we will use a window with size sampling of 1024 that corresponds to 100ms approximately. A quarter of the sampling has been used and the overlap obtains has been of seventy-five per cent. In order to obtain that, we will use the *hann* command that can be found at MATLAB®. An example of how this command works is shown in the following lines:

```
w = hann(L)
```

w = hann(L) returns an L-point symmetric Hann window. In our example, the value of w = 1024.

Figure 2.3 is an example of representation of a Hann window of 64 sampling, both in temporal domain (left graph) and spectral domain (right graph).

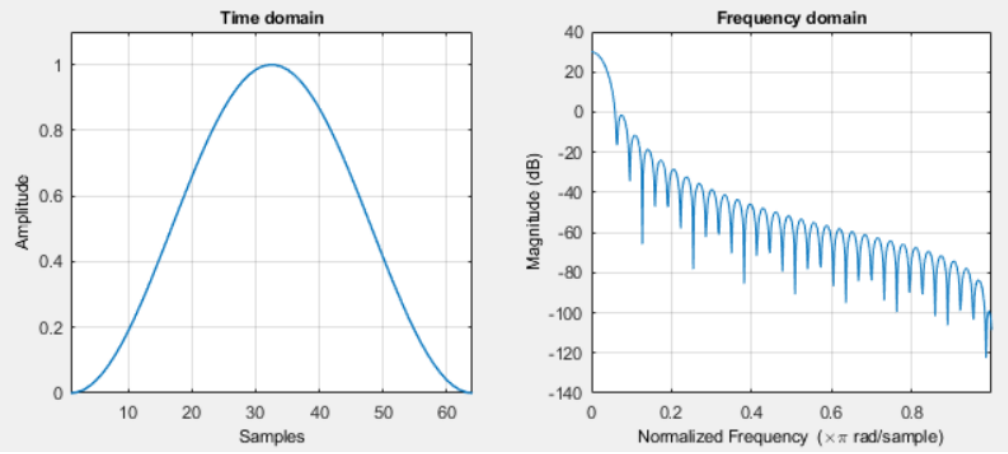


Fig. 2.3 Hann window of 64 sampling

2.1.6. Step six. Spectogram calculation.

To calculate the spectrogram, it is necessary to obtain the absolute value (abs) of Fast Flourier Transform (FFT) of the values previously obtained.

FFT is a fast and efficient algorithm for computing the constituent frequencies of a signal. The magnitude of the FFT gives the peak amplitude of the frequencies contained in a signal.

To do that in MATLAB®, we need to programme a *for* that will iterate between the number of windows. That can be achieved using the a*bs (fft)* command found in MATLAB®. In this way, all the samples are taken at the same time.

2.1.7. Step seven. Spectrogram filtering.

To do the filtering of the spectrogram, the following formula will be applied:

$$X_i^{emph} = X_i - \frac{1}{N} \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} X_{\min(\max(k,1),N)} \qquad \textbf{(2.2)}$$

Where *Xi* corresponds to the energy of each *i-th* frequential component of each sample and *N* is the window of noise extraction. That formula allows us to search maximum and minimum values. This step is useful to emphasize very soft tones from the background music.

2.1.8. Step eight. Binarization.

In this step we will do binarization. In order to do that, we will compare each frequential component with a determined threshold. After several trials, the value of the threshold has been established as t = 0.1. The following formula is used in the realization of this calculation:

$$B_{ij} = \begin{cases} 1 & X_{ij}^{emph} > t \\ 0 & X_{ij}^{emph} \leq t \end{cases} \qquad \textbf{(2.3)}$$

The command that has been used in MATLAB® is the *sign* command.

Specifically, in our algorithm, the value obtain from the *Bij* variable is a matrix formed by 512 rows and 5397 columns. The value of the columns is only valid in this example, in other examples the values will be different.

2.1.9. Step nine. Computation of frequency activation.

In this ninth step, we will do the Computation of Frequency Activation. It consists in averaging filtered and binarized spectrums of the last 100 frames. In that way, the system identifies if it is dealing with music or voice. The resolution obtained was of 2,5 seg.

To do that, the following formula should be used:

$$Activation(i) = \frac{1}{F}\sum_{j=1}^{F} B_{ij} \qquad \textbf{(2.4)}$$

Where *F* is the number of samples of each block, in our example is 100, and *Bij* is the matrix that has been obtained in the previous step after doing the binaraization.

Activation (i) is calculated for each block. The number of blocks will depend on the length of the audio file to be tested. Each block has a Fourier transform of 100 frames that is already filtered, equalized and binarized. With this information, the system will decide if this block of 100 frames relates to music or speech.

Therefore, we will be able to measure the frequency of a component of a specific frequency that is active in a block. Specifically, in our algorithm, the value obtained for the *Activation (i)* variable is a matrix of 512 rows (is the dimension of the FFT) by 106 columns (that depends on the duration of the audio to be tested).

Two graphs can be seen down below. The first one represents blocks without music and the second one represents blocks with music.

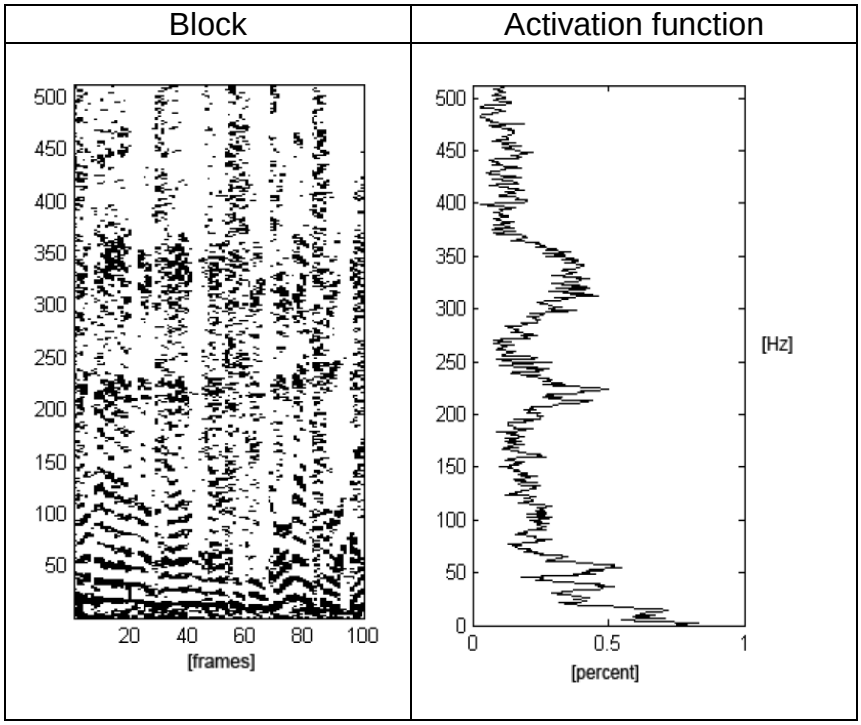


Fig. 2.4 Binarized spectrogram of a block and the corresponding activation function. Contains no music.

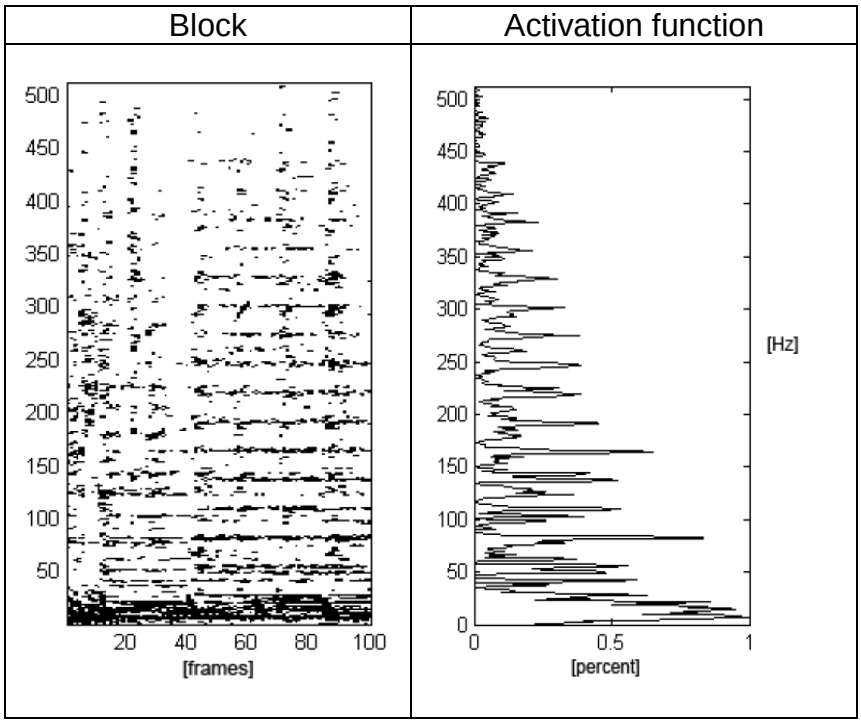


Fig. 2.5 Binarized spectrogram of a block and the corresponding activation function. Music is present.

2.1.10. Step ten. Strong peak detection.

To detect strong peaks, we have programme a function in MATLAB®. This function goes through the signal and detects all its peaks. The strong picks provide us useful information as they are good indicators of presence of music.

2.1.11. Step eleven. CFA filtration.

In order to do the filtration, we will use the values obtained in the previous step.

We will take the values of all peaks detected and we will sort them in descending order. Once that has been done, we will obtained a determined numerical value for each block, that value quantifies the presence of components of constant frequency inside the current audio segment.

In our example, we will store all this information in the variable CFAW. CFAW is the variable vector lenght that depends on the duration of the audiofile to be tested.

It is important to highlight that in the blocks that contains music, the resulting value should be higher than in the blocks in which there is no music.

2.1.12. Step twelve. CFA sampling.

In order to do the sampling, we will compared from *i = 1* to *q value* (106 position vector) the *CFAW* with *TH. CFAW* is the result of the CFA filtering, it is a vector of 106 positions. *TH* is the comparison threshold, in our example and after numerous trials, the value of *TH = 1.*

As a result, and after doing the necessary calculations, a signal that corresponds to the resulting vector will be obtained. If the value is higher than 1, the system will identify it as music, by contrast, if the value is lower than 1, the system will identify it as speech. If the value equals 0, the system will identify it as moment of silence in the audio file.

2.1.13. Step thirteen. Resulting graph.

In the last step of our algorithm, we will compare the original audio file with the result obtained after the process.

The result obtained after all the process can be seen down below.

The graph on the left corresponds to the original audio signal. This signal is formed by music and speech alternatively. Looking at the graph on the right, it can be observed that the areas of signal that are above the determined threshold correspond to music, while the areas that are below correspond to speech.

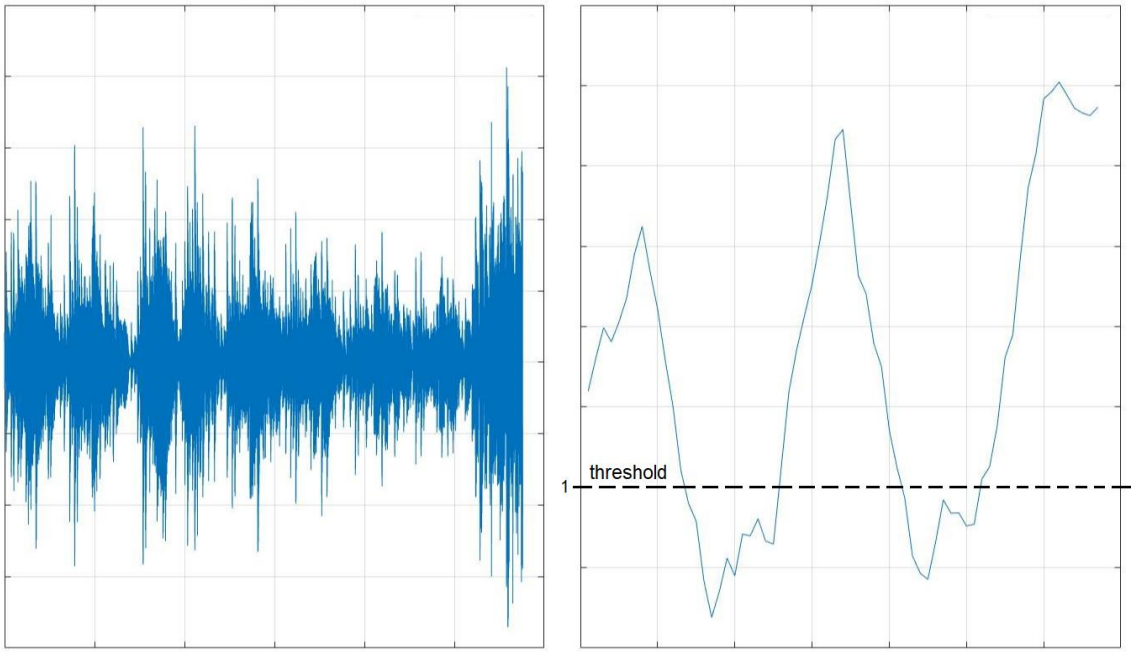


Fig. 2.6 Original audio signal and final resulting graph

As previously said, all these steps have been programmed through the use of the MATLAB® software, several tests has been performed with the help of this software. The results obtained, and their interpretation will be analysed in chapter 4 of this project, in that chapter, we will also talk about the audio data base used in the realization of the tests, construction, decisions, etc.

The next chapter will be focused in the tools that have been use in the analysis and the development of this project.

CHAPTER 3. SOFTWARE DEVELOPMENT TOOLS

3.1. Matlab®.

MATLAB® (stands for MATrix LABoratory, laboratory of arrays) is a tool for integrated development environment (IDE) with own programming language (language M). It is available for platforms Unix, Windows, Mac OS X and GNU/Linux.

Some of their basic features are: manipulation of arrays, representation of data and performance, algorithm implementation, user interface creation (GUI) and communication with other programmes in different languages and with other hardware devices.

MATLAB® has been used for programming, testing and evaluating all the process of the designed algorithm. It is a software relatively easy to use that contributes to a quite fast and intuitive programming with acceptable results.

3.2. Audacity®.

In order to edit/convert/cut/combine/record, etc, the audio file needed for our database we have use a program called Audacity®.

Audacity® is a very easy to use and intuitive program and therefore ideal for our requirement in this project. Another advantage of this program is that it supports a wide range of different audio file formats that allow us to work with a wide range of audio files in different formats.

Audacity is a free computer application that can be use for recording and audio editing. It is distribuited under GPL license and it s the most widespread audio editor in GNU/Linux systems.

In the development of this project, its has been fundamental the use of a device with a great potential in graphics. The reason being that the graphic simulator of MATLAB® requires a minimum of power in order to obtain visible results in a relatively short period of time.

Specially, when several tests need to be performed and the time of each result is being added to the previous and so on.

3.3. Analysis processor.

The device that has been used in the devolpment of this project is last generation ASUS® laptop that offers a great performance in MATLAB® simulation.

This great perfomance can be achieved thanks to its outstanding specifications: windows 10, 7th processor gernation Intel® Core™ i7 and dedicated charts GeForce® GTX™ 960M, all of them of great relevance in the development and testing of the studied algorithm.

CHAPTER 4. RESULTS

Before analysing the results obtained after the trials, we will talk about the data base that has been used in the realization of these trials.

4.1. Audio database.

A data base or data bank is a group of data that belongs to a same context and are systematically stored for its later use.

To test this method and perform the maximum trial possible, there has been necessary to create a data base.

Creating a data base is a very slow and demanding task, as it is not only necessary to find audio flies, it is also necessary to edit them and build new audio files during the process. Although it is not a difficult process it is both slow and laborious.

The data base must be complete and must represent the problem that we want to resolve. Also, it must be manually labelled and therefore generating a ground truth with which it will be possible to validate the consistency of the method, in other words, its probability or error or success.

This data base must have a variety of different musical and voice styles enough quantity of information for obtaining reliable results after doing the trials.

Taking this into account, a total of 750 audio file has been tested, of which 600 were music and 150 were speech. That is the equivalent of saying that 175 minutes of music from different musical styles and 44 minutes of speech has been tested.

As it can be observed, the sampling music space is wider than the sampling speech space, that is because several musical styles have been analysed.

The following chart shows a summary of all the information that has been tested:

Chart 4.1. Tested information in the realization of the trials.

Music genre	Number of audio files tested	File length
Classical	150	15-20 seg
Blues	150	15-20 seg
Electronic	150	15-20 seg
Jazz	150	15-20 seg
Speech	150	15-20 seg

An scheme of our data base and the process needed to obtaining results is shown in the figure below:

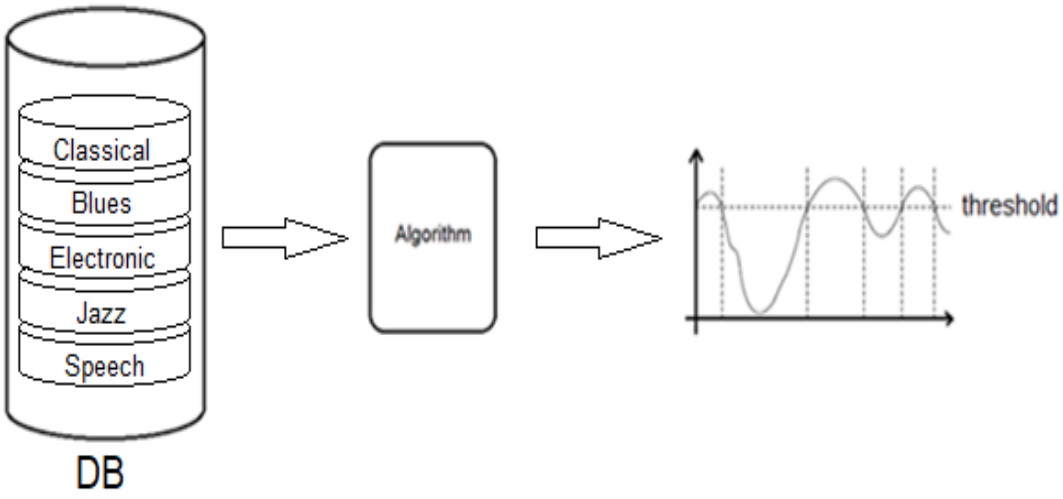


Fig. 4.1 Scheme of data base used in he realization of the trials

After examining the data base, the information that has been tested and the process to follow, we will study the results obtained after the tests.

4.2. Results.

In the initial test, silences were consciously inserted in audio files at specific moments to verify that the time was consistent. That allowed us to monitor if the algorithm was able to detect these silences correctly. As the main objective of this system is to detect moments in which there is music with absence of voice and after confirmation that the algorithm could detect moments of silence correctly, files of mixed audio that alternate music and speech consecutively were used.

Charts with the breakdown results for musical style are down below. But first, it necessary to define:

- True positive (TP): number of musical events correctly identified
- False negative (FN): number of musical events wrongly identified
- False positive (FP): number of voice events wrongly idenfied
- True negative (TP): number of voice events correctly identified

Chart 4.2. Results obtained according to music genre.

Genre	File tested	True Positive	Percentage of True Positive	False Negative	Percentage of False Negative
Classical	150	148	98.66%	2	1.33%
Blues	150	145	96.66%	5	3.33%
Electronic	150	144	96%	6	4%
Jazz	150	145	96.66%	5	3.33%

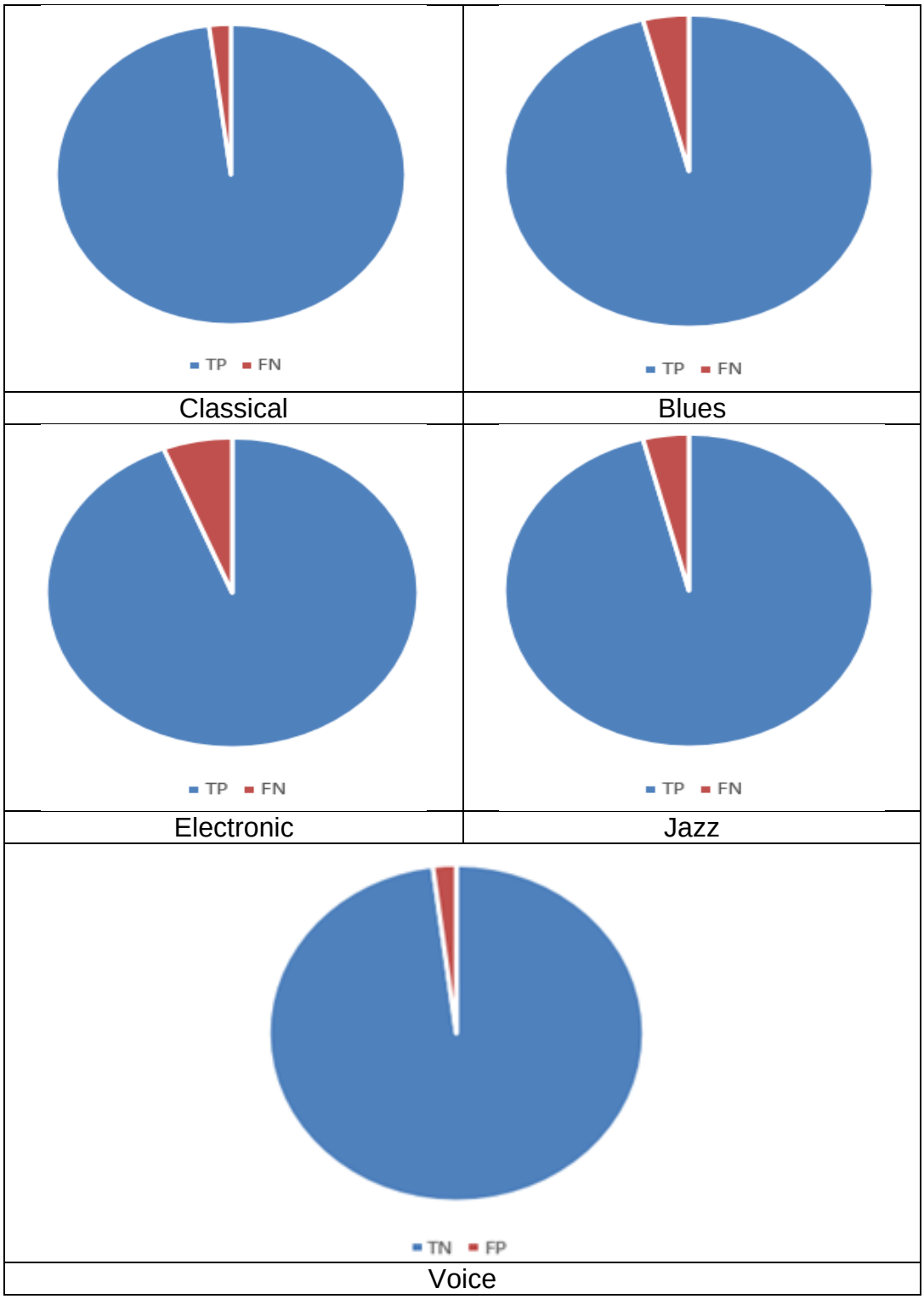
The results obtained with the voice audio file are as follow:

Chart 4.3. Results obtained in voice files.

Genre	Number of file tested	False Positive	Percentage of False Positive	True Negative	Percentage of True Negative
Voice	150	2	1.33%	148	98.66%

The following chart show the percentage of success and error represented by music genre and voice:

Chart 4.4. Results of percentages success/mistake by music genre and voice.



The advantage of this method resides in its few parameters, there is only one threshold of decision and therefore the main objective has been to test the performance of the method and select the best threshold value.

The precision, recall and accuracy values of the system can be obtained through the results of the previous charts. To do that, we will use the following formulas:

Precision

$$Precision = \frac{\sum \mathrm{TP}}{\sum \mathrm{TP} + \sum \mathrm{FP}} \quad \textbf{(4.1)}$$

Precision = ∑TP / (∑TP+∑FP) = (730) / (730+2) = **99.73%**

Recall

$$Recall = \frac{\sum \mathrm{TP}}{\sum \mathrm{TP} + \sum \mathrm{FN}} \quad \textbf{(4.2)}$$

Recall = ∑TP / (∑TP+∑FN) = (730) / (730+20) = **97.33%**

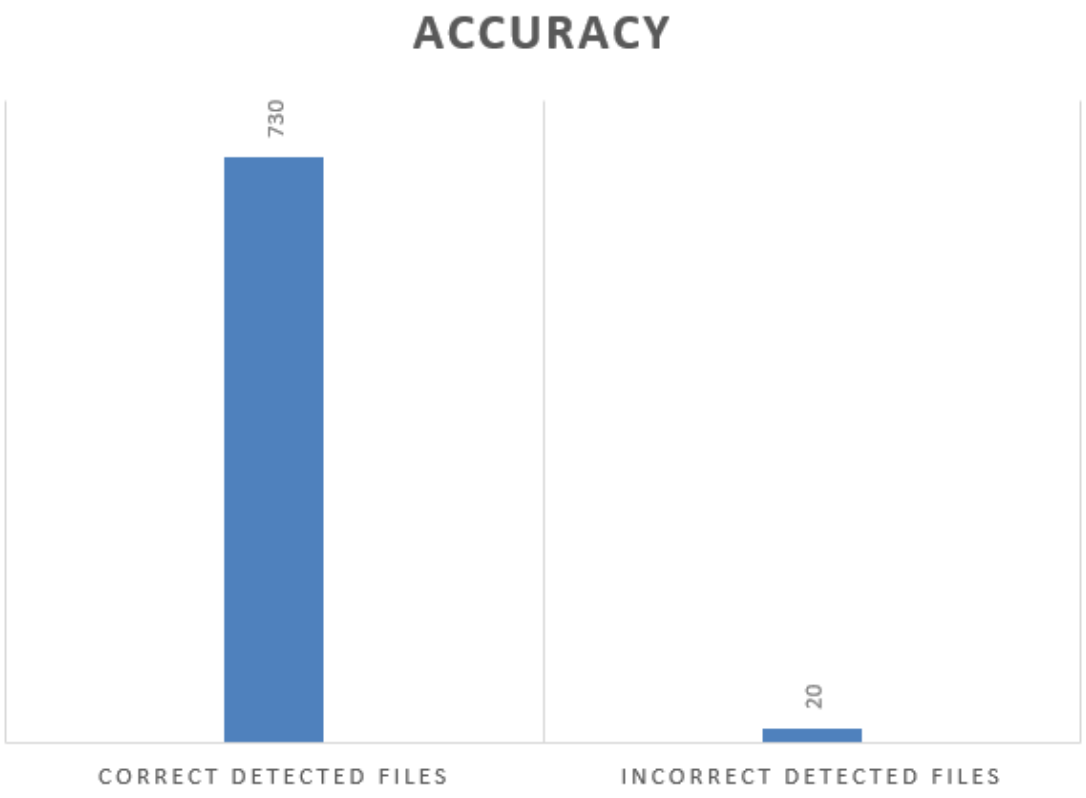
Accuracy

$$Accuracy = \frac{\sum \mathrm{TP} + \sum \mathrm{TN}}{(\sum TP + \sum TN + \sum FP + \sum FN} \quad \textbf{(4.3)}$$

Accuracy = (∑TP+∑TN) / (∑TP+∑TN+∑FP+∑FN) = (878) / (900) = **97.55%**

The next graph shows the system accuracy after testing 750 audio files:

Chart 4.5. Correct detected files vs. Incorrect detected files.



Analysing the results, it can be said that the percentages of error seem quite acceptable. The system can discern musical events from voice very precisely. Specifically, with a 97.55% of fiability.

As any other system, improvements could be made. One option for improving the final statistics would be to apply morphological filters (erosion, dilatation).

CHAPTER 5. CONCLUSIONS

Now that the project has been analysed, it's time to discuss the conclusions obtained after reviewing the results.

In the first place, it can be observed that through the algorithm that has been developed, implemented and optimized, it is possible to distinguish if an audio file contains speechless sections, this is very important because that was the main goal of the project. Also, as it has been previously seen, the error rate in voice/silence discrimination was very low (2.45%).

Therefore, it can be said that the algorithm is performing according to the function it was created for and provides a quite high level of satisfaction (97.55%). Also, this algorithm is able to solve several common problems that exits in the field of music/speech classification.

Therefore, it can be said that this algorithm can be applied for the purpose it was created for, introducing advertising at the most appropriate moment in an video/audio file (film, tv series, etc).

Although, the focus of the application of the algorithm in this project has been voice/audio file discrimination it can be also used in many other application, as it has been previously mentioned in this report.

This versatility provides a very interesting appeal to be applied in many other fields.

It is a closed process, exclusively developed to detect specfic characteristics of the musical signal. There is no training needed and does not require data bases for the training, only for realization of tests. This, simplifies the design of the data base to a large degree.

Another very important thing in this project is the tool that has been used to create/programme/implement this algorithm, as previously mentioned, the name of this tool is MATLAB®.

While working in this project, there has been moments in which I had the necessity of being self-taught and look for information about commands, instructions, etc.

That has given my the oportunity of extending my knowledge of this program and learn how to use it fluently.

At this point, several lines of development can be proposed. In the following lines, we will explain what this future development lines consist on.

1. In order to improved music detection it could be possible to use structural aspects and rhythm properties of music; this will allow to detect background music in an audio/video file. The rate of detection will be positively increased.

2. A possible improvement would be the use of methods that detect the characteristics that has been explained in the first chapter and later algorithm of machine learning as SVM (Support Vector Machine) are applied, although for that, it would be necessary to use data bases.

3. Another option would be the use of Deep learning algorithm. In this case, very large data bases would be required, but that won't suppose a problem as the system will automatically learn the best features to perform the classification.

4. Another improvement when using C++ programming is that, as we already know, computational calculation in this language is much faster than using MATLAB® software. That will provide extra speeding computational calculation and therefore a faster processing system and programme discrimination.

Generally speaking, these would be some of the measures to take as future lines in the continuation of the development of this project.

It is important to consider again the relevance that this algorithm can have for some companies dedicated to TV (Antena 3, Telecinco, etc). In most cases, advertising and insertion of adverts while playing tv programmes/films/tv series, is something quite bothering for the viewer when it is randomly done (as it being done currently) without taking into account any of the aspects that have been seen in the development, implementation and optimization of this algorithm.

BIBLIOGRAPHY

[1] Lu, Zhang, Jiang - 2002 - Content analysis for audio classification and segmentation

[2] Seyerlehner et al. - 2007 - Automatic Music Detection in Television Productions

[3] Ewald Wieser, Matthias Husinsky - 2014 - SPEECH MUSIC DISCRIMINATION IN A LARGE DATABASE OF RADIO BROADCASTS FROM THE WILD

[4] Ghosal et al. - 2012 - Music Classification based on MFCC Variants and Amplitude Variation Pattern A Hierarchical Approach

[5] Tzanetakis, Cook - 2002 - Musical genre classification of audio signals

[6] McLoughlin I. Applied Speech and Audio Processing with Matlab Examples (CUP,2009)(en)(206s)

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4322669/

https://cba.fro.at

http://lamusicaclasica.com

https://www.ivoox.com

http://freemusicarchive.org

http://musicaq.biz