

UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR



PROYECTO FIN DE CARRERA

REAL-TIME SOUNDTRACK ANALYSIS

Francesco Alfeo

2009

REAL-TIME SOUNDTRACK ANALYSIS
(Análisis en vivo de la banda sonora)

AUTOR: Francesco Alfeo
TUTOR: José M. Martínez

Dpto. de Ingeniería de Telecomunicación
Escuela Politécnica Superior
Universidad Autónoma de Madrid
diciembre de 2009

Summary

The practical realization of the project described in the present document took place in the Telecommunication Engineering Faculty of the Escuela Politecnica Superior (Autónoma University, Madrid), with the supervision of prof. José Marià Martínez Sanchez. This work tries to find a solution to the problem of automatically segment a video using information extracted from the audio track that accompanies it. For this purpose, two classifiers have been designed and projected: the first one is capable to perform this task in a real-time context, while the other performs the same task without worrying about any time constrain. Another objective of the present work is to discover and discuss the differences between these two classifiers in terms of computational complexity, execution velocity and obtained performances.

Both classifiers start from a WAV file (obtained from the starting audio or video file of a generic format) to subdivide the analyzed clip into smaller parts, depending on the identified audio type. At the end of the process, there will be parts belonging to 4 different audio types: voice, music, noise and silence. The proposed classification algorithm is divided into two parts: the first one recognizes and marks the silence frames in the audio clip, while the second one distinguishes the remaining frames marking them as music, noise or voice. Both parts use a series of heuristic conditions based on statistical measures (local maximum and minimum, variance and arithmetic mean) calculated from a series of audio low-level features (energy, Zero Crossing Rate, Root Mean Square, Spectral Energy Distribution in 4 frequency sub bands, spectral Centroid and Roll-Off point). Only in the second part of the algorithm, a certain number of points (stored in a vector) is assigned to each audio type for a certain frame each time one of the above mentioned conditions is satisfied.

Results obtained by both classifiers appear satisfactory: the off-line classifier reaches on average (evaluating 10 different combinations “length of an analysis window, number of analysis windows for frame” - both parameters are set by the user) a classification accuracy equal to 77%, while the real-time one obtains 72% and fully satisfies real-time constraints. These results appear to be very promising as a starting point for future lines of research and development in the field of automated video segmentation using audio features only.

INDEX OF CONTENTS

Chapter 1

INTRODUCTION.....	1
<i>1.1 Motivations and objectives.....</i>	<i>1</i>
<i>1.2 Work structure.....</i>	<i>3</i>

Chapter 2

BACKGROUND.....	4
<i>2.1 Digital Signal Processing: definition and history.....</i>	<i>4</i>
<i>2.2 Automated video segmentation and classification.....</i>	<i>5</i>
<i>2.2.1 Approaches to video classification.....</i>	<i>6</i>
<i>2.2.2 A panoramic on algorithms of video segmentation using video features.....</i>	<i>8</i>
<i>2.3 Audio classification and segmentation.....</i>	<i>10</i>
<i>2.3.1 Why audio content analysis?.....</i>	<i>10</i>
<i>2.3.2 Important contributions in the field of audio classification.....</i>	<i>11</i>
<i>2.3.3 Audio classification algorithms with real-time constraints.....</i>	<i>13</i>
<i>2.4 Integration of audio and visual information for video segmentation.....</i>	<i>15</i>
<i>2.5 Summary.....</i>	<i>17</i>

Chapter 3

WORK DESIGN.....	18
<i>3.1 Introduction.....</i>	<i>18</i>
<i>3.2 Classified audio types: a definition.....</i>	<i>18</i>
<i>3.2.1 Voice.....</i>	<i>19</i>
<i>3.2.2 Music.....</i>	<i>21</i>
<i>3.2.3 Noise.....</i>	<i>22</i>
<i>3.2.4 Silence.....</i>	<i>22</i>

3.3	<i>Low-level features choice</i>	23
3.3.1	Time-domain features.....	24
3.3.1.1	Energy.....	24
3.3.1.2	Zero Crossing Rate.....	25
3.3.2	Frequency-domain features.....	26
3.3.2.1	Energy distribution across frequency bands.....	26
3.3.2.2	Root Mean Square.....	27
3.3.2.3	Spectral Centroid.....	28
3.3.2.4	Roll-off Point.....	29
3.4	<i>An idea for the classification algorithm</i>	29

Chapter 4

WORK DEVELOPMENT	32	
4.1	<i>Chapter organization</i>	32
4.2	<i>Audio file opening and reading</i>	33
4.2.1	Practical implementation.....	35
4.3	<i>Features extraction</i>	36
4.3.1	Off-line classifier.....	36
4.3.2	Real-time classifier.....	38
4.4	<i>Features plotting</i>	40
4.4.1	Audio types behavior as a function of low-level features.....	41
4.4.1.1	Energy.....	42
4.4.1.2	Zero Crossing Rate.....	43
4.4.1.3	Energy distribution across frequency bands.....	44
4.4.1.4	Root Mean Square.....	47
4.4.1.5	Spectral Centroid.....	48
4.4.1.6	Roll-off Point.....	49
4.5	<i>Classification algorithm</i>	50
4.5.1	Heuristic-based conditions.....	51
4.5.2	A points vector for each audio type.....	54
4.5.3	Practical implementation.....	56
4.5.3.1	Off-line classifier.....	56

4.5.3.2	Real-time classifier.....	62
4.6	<i>Classification results plotting</i>	65
4.6.1	Off-line classifier.....	66
4.6.2	Real-time classifier.....	71
4.7	<i>Summary</i>	71

Chapter 5

EVALUATIONS	74
5.1 <i>Preliminary considerations</i>	74
5.1.1 Manual classification and its limits.....	74
5.1.2 Choice of parameters to describe classifiers quality.....	76
5.1.3 Computational time evaluation for real-time classifier.....	81
5.1.4 Data set used to evaluate the work of both classifiers.....	82
5.2 <i>Results evaluation</i>	82
5.2.1 Classification accuracy.....	83
5.2.2 Error Rate.....	86
5.2.3 Relative Error Rate.....	90
5.2.4 Normalization Improvement.....	96
5.2.5 Computational time evaluation.....	98

Chapter 6

CONCLUSIONS AND FUTURE WORK	100
Appendixes	103
<i>A – WAV format description</i>	103
<i>B – MATLAB implementation</i>	105
References	110
Glossary	114

Análisis de la banda sonora en tiempo real.....	116
RESUMEN.....	116
INTRODUCCION.....	117
CONCLUSIONES.....	121
Presupuesto.....	124
Pliego de condiciones.....	125

INDEX OF FIGURES

Figure 2-1: Some of the areas in science and engineering revolutionized by DSP.....	5
Figure 2-2: Saunders' real-time speech/music discriminator: block diagram of processing flow.....	14
Figure 3-1: Human speech apparatus.....	21
Figure 3-2: High-level classifier scheme.....	30
Figure 4-1: Scheme of the format conversion process.....	34
Figure 4-2: Energy function graph: an example.....	42
Figure 4-3: ZCR graph: an example.....	43
Figure 4-4: Sub bands 1 and 2 energy graph: an example.....	44
Figure 4-5: Sub bands 3 and 4 energy graph: an example.....	45
Figure 4-6: RMS graph: an example.....	47
Figure 4-7: Spectral Centroid graph: an example.....	48
Figure 4-8: Roll-off Point graph: an example.....	49
Figure 4-9: An example of normalization applied on a frame labeled as “noise” surrounded by two “music” frames.....	66
Figure 4-10: An example of normalization applied on a group of frames, in which all frames that constitute a minority are labeled as the more frequent audio type (in the case	

of the figure “music”)	67
Figure 4-11: First output file of function <code>OffLineClassifier</code> : “Off line classification results.jpg”	69
Figure 4-12: Second output file of function <code>OffLineClassifier</code> : “Off-line NO NORMALIZED results.jpg”	70
Figure 4-13: Third output file of function <code>OffLineClassifier</code> : “Points noise-speech-music.jpg”	70
Figure 4-14: Flow chart representing all the working steps of the off-line audio classifier.	72
Figure 4-15: Flow chart representing all the working steps of the real-time audio classifier.	73
Figure 5-1: Graph showing how CA changes in the off-line case increasing the number of analysis windows in a frame and keeping the same analysis window length (0,185 s).	85
Figure 5-1: Graph showing how CA changes in the real-time case increasing the number of analysis windows in a frame and keeping the same analysis window length (0,185 s).	86
Figure A-1: Wave string format example.	103
Figure A-2: The canonical WAV file format.	104

INDEX OF TABLES

Table 4-1: Heuristic features used for classification purposes and their practical Matlab implementation.	54
Table 4-2: Example of priority assignment scheme for the various audio types.	56
Table 4-3: List of all heuristic conditions used in music-noise-speech discrimination algorithm.	62
Table 4-4: Heuristic conditions in <code>RealTimeClassifier</code> that were absent or different in <code>OffLineClassifier</code> .	64
Table 5-1: An example of accuracy evaluation referred to the off-line classification of an audio clip of one minute (parameters: 1) number of samples for one window = 8192 → window length = 0.184 s; 2) number of windows for one frame = 10; 3) frame length = 1.84 s).	78

Table 5-2: Main features of the three audio clip employed for the evaluation of both the off-line and the real-time classifier.....	82
Table 5-3: Classification accuracy (CA) evaluated for the three audio clips varying the length of the frame and of the analysis window. NCFOM stands for “No Country For Old Men”, SC stands for “Sin City” and GCBC is “Good Copy Bad Copy”. The three best results obtained for each audio clip are evidenced in bold.....	83
Table 5-4: Extract taken from Table 5-3, that shows how both classifiers improve their performance if using shorter analysis windows. The two exceptions to this rule are evidenced in bold. The two combinations returning the best classification results are underlined.....	84
Table 5-5: Extract taken from Table 5-3, that shows how both classifiers performance is affected in a limited way by the number of windows belonging to one single frame.....	85
Table 5-6: Error Rate (ER) related to speech frames and evaluated for the three audio clips varying the length of the frame and of the analysis window. The two best results obtained for each audio clip are evidenced in bold. The two combinations returning the best classification results are underlined.....	87
Table 5-7: Error Rate (ER) related to music frames and evaluated for the three audio clips varying the length of the frame and of the analysis window. The two best results obtained for each audio clip are evidenced in bold. The two combinations returning the best classification results are underlined.....	88
Table 5-8: Error Rate (ER) related to noise frames and evaluated for the three audio clips varying the length of the frame and of the analysis window. The two best results obtained for each audio clip are evidenced in bold. The two combinations returning the best classification results are underlined.....	89
Table 5-9: Error Rate (ER) related to silence frames and evaluated for the three audio clips varying the length of the frame and of the analysis window. The two best results obtained for each audio clip are evidenced in bold. The two combinations returning the best classification results are underlined.....	90
Table 5-10: Relative Error Rate (RER) related to speech frames and evaluated for the three audio clips varying the length of the frame and of the analysis window. Lowest RER for each combination is evidenced in bold, while the highest is underlined.....	92
Table 5-11: Relative Error Rate (RER) related to music frames and evaluated for the three audio clips varying the length of the frame and of the analysis window. Lowest RER for each combination is evidenced in bold, while the highest is underlined.....	93
Table 5-12: Relative Error Rate (RER) related to noise frames and evaluated for the three audio clips varying the length of the frame and of the analysis window. Lowest RER for each combination is evidenced in bold, while the highest is underlined.....	95

Table 5-13: Relative Error Rate (RER) related to silence frames and evaluated for the three audio clips varying the length of the frame and of the analysis window. Highest RER for each combination is underlined.....	96
Table 5-14: Normalization Improvement (NI) evaluated, for both classifiers and for each of the three analyzed audio clips, varying the length of the frames and of the analysis windows in which they are divided. The three best results obtained for each audio track are evidenced in bold, the three combinations with the highest NI are underlined.....	97
Table 5-15: Computational time needed by real-time classifier to perform classification related to the whole audio track (left column) and to one of its frames (right column; medium value evaluated dividing the parameter shown in the left column for the total number of frames in which the analyzed audio track is subdivided).....	99
Table B-1: MATLAB instructions used for the extraction of each low-level feature related to one analysis window.....	105
Table B-2: Structure of RealTimeClassifier.m. The “for” cycle containing the extraction of the features for each window in one frame is evidenced in bold.....	106
Table B-3: Structure of OffLineClassifier.m. The “if” condition that recognizes a DAT file storing low-level information is evidenced in bold.....	106
Table B-4: Structure of OffLineClassifier.m. The “while” cycle in which grouping of low-level features and evaluation of related heuristic features is evidenced in bold....	107
Table B-5: Structure of OffLineClassifier.m. Structure of both silence identification and music-speech-noise discrimination algorithms is evidenced in bold.....	108
Table B-6: Structure of RealTimeClassifier.m. The “for” cycle performing the classification frame by frame is evidenced in bold.....	109
Table B-7: Normalization algorithm applied on an auxiliary vector (aux).....	109
Table B-8: Scaling algorithm applied on classification results.....	109

1 Introduction

1.1 Motivations and objectives

In the era of Internet and multimedia information, with more and more people using more and more sophisticated digital technologies, it seems to be necessary to introduce new tools and instruments that should permit to handle easily and efficiently this large amount of various information. The trend is, from the first computer to the last telecommunication device (PDAs, mobile phones, Blackberrys, etc.), to unify the flux of different kind of information (text, audio, video) in a unique multimedia stream.

In this situation, video scene analysis and classification are required in many applications, such as information indexing and retrieval in multimedia databases, video editing, etc. Research in this area in the past several years has focused on the use of speech and image information. These include the use of speech recognition and language understanding technology to produce keywords for each video frame or group of frames, the use of image statistics (color histogram, texture descriptor, shape descriptors) for characterizing contents of individual frames, etc. [1].

Only in the last few years some researchers began to focus their attention on the analysis of the accompanying audio signal for the same purposes, finding very encouraging results [2][3]. On a semantic level, for example, the audio in a sport video is very different from that in a news report. Various sport programs also have very different background sounds. Hence good search engines in a site with a lot of video stored in it should permit for search of, for example, all the basketball games, or football ones. On another, but more specific, semantic level, it is possible to find in the middle of a long video all that parts that can be of some interest for a user (e.g. in a football match, all goals or important actions). This task can be accomplished using a good audio classification system that searches for all the parts of the long video that have some acoustical characteristics (e.g. the crowd cheering at a loud volume) strictly linked to a particular fragment of that video and not to others.

Anyway, the best results are obtained when the attention shifts from the separate analysis of audio and video to a joint one that could exploit all the advantages of both aspects. For this reason, a lot of recent frameworks integrate video and audio low-level features [4][5].

This paper's scope is limited to audio analysis, with the objective of extracting some conclusions useful for an upper-level retrieval and indexing system. These conclusions consist essentially of music/speech/noise/silence classification in an audio clip, both in the off-line and in the on-line case. The starting assumption is that, of course, the on-line analysis needs surely to be less sophisticated than the off-line one, but the matter is to see how much real-time requirements constrain the selectivity of the algorithm used to do this segmentation.

This paper wants also to contribute to the advances of the segmentation methods in terms of computational complexity and velocity (especially for the real-time case), because the majority of the frameworks previously proposed obtain optimal results only for simple segmentation cases (for example, only speech/music discrimination). Another contribution of this paper is the fact that it uses a segmentation scheme of the audio signal adaptable to different durations, searching a compromise between analysis precision and segmentation quality.

Surely a good audio segmentation scheme needs a good set of audio features; for this the first step of the process, both in the off-line and the on-line case, is the extraction of these features: in the off-line case these will be Energy, Zero Crossing Rate, Subband Frequency Energy, Root Mean Square, Rolloff Point and Spectral Centroid; in the on-line one only a smaller subset of these features will be used. The second step is an algorithm that, starting from the features extracted, could first recognize and distinguish the silence parts of the clip from the others, and then recognize and segment correctly music, speech, environmental sounds, and noise.

1.2 Work structure

The rest of this Master Thesis is organized in chapters, each of which explains a step of working flow. In the rest of this paragraph there is the name with a short summary of the contents of each chapter.

- In Chapter 2 (“*State of the art*”) there is a brief description of the history of audio digital analysis. It starts with a description of the most important steps in the history of Digital Signal Processing (DSP), then it focuses on the frameworks used in the last few years to segment an audio clip onto smaller parts and classify them in some way.
- Chapter 3 (“*Work design*”) starts with a brief summary of the salient characteristics of each audio type – voice, music, noise and silence – the framework aims at for its classifying purposes. Then, it goes on with the description of all the features extracted from the audio clip useful to find for its further analysis, in relation with the information they give about the type of each audio segment. The chapter ends briefly describing the main idea behind the classification algorithm realization.
- In Chapter 4 (“*Work development*”) all the steps followed during the realization of the project are described and explained in detail, as well as the computational consequences of the choices we made.
- Results and evaluation of the work with its qualities and critic aspects are provided in Chapter 5 (“*Evaluations*”), with comparisons with the results obtained using other frameworks in the various step of the work.
- In the last chapter (6) we summarize the conclusions of the work done, and hypothesize possible lines to follow in the same direction with future works.

2 Background

2.1 Digital Signal Processing: definition and history

DSP is concerned with the representation of the signals by a sequence of numbers or symbols and the processing of these signals. DSP is the mathematics, the algorithms, and the techniques used to manipulate these signals after they have been converted into a digital form.

Since the goal of DSP is usually to measure or filter continuous real-world analog signals, the first step is usually to convert the signal from an analog to a digital form, by using an ADC (Analog-to-Digital Converter). Often, the required output signal is another analog output signal, which requires a DAC (Digital-To-Analog Converter). Even if this process is more complex than analog processing and has a discrete value range, the stability of digital signal processing thanks to error detection and correction and being less vulnerable to noise makes it advantageous over analog signal processing for many, though not all, applications [6].

The roots of DSP are in the 1960s and 1970s when digital computers first became available. Computers were expensive during this era, and DSP was limited to only a few critical applications. Pioneering efforts were made in four key areas:

- *radar and sonar*, where national security was at risk;
- *oil exploration*, where large amounts of money could be made;
- *space exploration*, where the data are irreplaceable;
- *medical imaging*, where lives could be saved.

The personal computer revolution of the 1980s and 1990s caused DSP to explode with new applications. Rather than being motivated by military and government needs, DSP was suddenly driven by the commercial marketplace. Anyone who thought they

could make money in the rapidly expanding field was suddenly a DSP vendor. DSP reached the public in such products as: mobile telephones, compact disc players, and electronic voice mail. Figure 2-1 illustrates a few of these varied applications.

This technological revolution occurred from the top-down. In the early 1980s, DSP was taught as a graduate level course in electrical engineering. A decade later, DSP had become a standard part of the undergraduate curriculum. Today, DSP is a basic skill needed by scientists and engineers in many fields. As an analogy, DSP can be compared to a previous technological revolution: electronics. While still the realm of electrical engineering, nearly every scientist and engineer has some background in basic circuit design. Without it, they would be lost in the technological world. DSP has the same future [7].

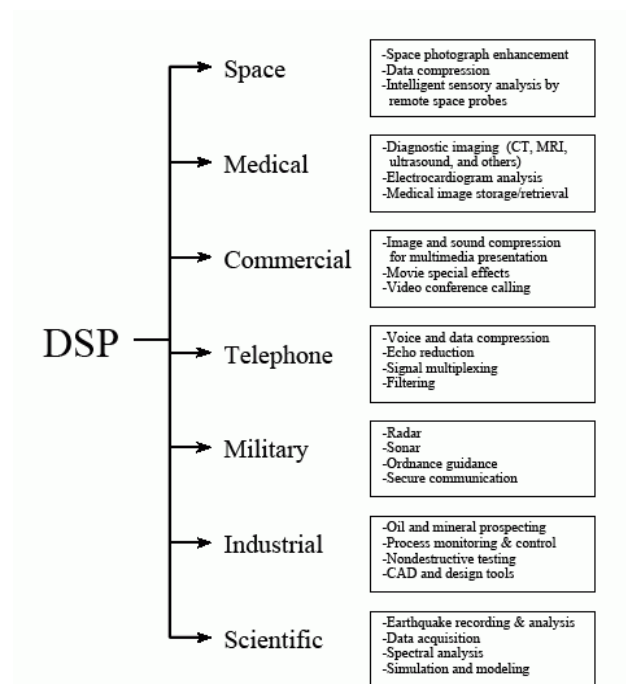


Figure 2-1: Some of the areas in science and engineering revolutionized by DSP

2.2 Automated video segmentation and classification

In the field of DSP, automated analysis, classification and segmentation of a video clip is an operation whose importance is increasing, especially in the last few years.

That's because digital video collections are growing rapidly in both the professional and consumer environment, and are characterized by a steadily increasing capacity and content variety. In this sense, the Internet is surely multiplying the impact of this phenomenon among common people. A site like YouTube, for example, gives people the opportunity to upload their videos and share them with the world. The 2008 Presidential Election has been utilizing YouTube as a medium to promote, support, and publicize the presidential candidates, and this is the most powerful way to say how important video sharing is and, at the same time, about the possibility the user has to choose what to watch, and when.

The task of automated segmentation, indexing, and retrieval of audiovisual data has also important applications in professional media production, education, entertainment, surveillance, and so on. For example, let's take the case of a television that has in its database a vast amount of films and other kinds of audiovisual material. If these data can be properly segmented and indexed, it will facilitate the retrieval of desired video segments for the editing of a documentary or an advertisement video clip. Another example is given by audiovisual libraries or family entertainment applications, in which it would be convenient for users if they are able to retrieve and watch video segments of their interest. Nowadays, the most used way to help people search among these huge video collections is based on text (especially *tags*) but, as everybody can experience, this is an easy but not so effective way to categorize video, used a lot of times to make people visit some advertising videos or others in which the user wasn't really interested in. On the other hand, as the volume of the available material becomes huge, manual segmentation and indexing is impossible. So, transferring the search and retrieval tasks to automated systems becomes crucial to be able to efficiently handle stored video volumes. The development of such systems is based on the algorithms for video content analysis [8].

2.2.1 Approaches to video classification

The description starts presenting the three main directions relatively to video content

analysis. There are those techniques that are example-based queries, others that attempt to summarize video and, finally, there are approaches that attempt to identify content of the video placing it into one of a set of previously defined categories [9].

In the example-based case, often termed *query by example* or *retrieval by example*, the system is presented with one or more examples of the type of video sequence required. The system then searches for videos with similar attributes. Approaches to query by example can vary in complexity of the similarity measure between the examples and the retrieved video sequences and functionality of the user interface. Early examples were applied to limited domains such as satellite imagery using formative similarity of roads and river networks by detecting edges. Bouthemy and Fablet [10] presented approaches that used basic formative motion measures to retrieve clips with similar motion attributes. Later, approaches made deliberate attempts to measure the cognitive similarity; for example Dori [11] modeled cognitive similarity using a visual object-process diagram scoring the similarity using object attributes: color, location and lighting. A commercial system that employed many measures of similarity was presented by Flickner [12]. It was called QBIC (Query By Image and video Content) system. Chang [13] presented an approach to this task that supported space-temporal queries, and a survey of content-based video retrieval was presented by Yongsheng and Ming [14], mainly focusing on processing in MPEG compressed video domain.

Video summarizing attempts to capture semantic content of a video and present a general highlight of the video in a shorter period of time. The abstraction of video is related to video retrieval as a full system would allow the browsing of a retrieved video. There exist certain difference between video (content) summarization and video skimming, although often these two expressions are used interchangeably in literature. While the result of the former is often represented by certain well-designed Graphics Interface depicted in key frames, in a mosaic form, or a kind of importance measures etc., the latter is almost certain to be in the form of a shorter semantics-preserved video file [15]. In this sense, one of the most important desired applications is real-time "adaptive video streaming". Aigrain [16] suggested in his paper that video skimming is

more successful in limited domains such as education or news videos; namely those with very explicit speech or text (closed caption) contents.

The third application drive of video classification is *video indexing or labeling* and differs from retrieval in that this approach attempts to generalize and make models for the classifications. However there may be similarity measures that are comparable between the two different approaches. This type of classification, where a given video is assigned to one of a pre-defined set of classes, has received more attention in the literature. Arguably the topic is now sufficiently mature to make comparisons of recognition accuracies across different modes, features and classifiers. Important practical issues that influence accuracies, can be summarized in terms of the classes themselves, and their intra- and inter-class variations. In other words the number of classes together with the specific classes (video genre) under test are likely to greatly influence classification accuracy. Another practical factor in assessment is the type of testing, whether it is closed or open-set identification or whether it is the verification of a claimed identity label (a yes/no decision). Closed-set testing, though often not explicitly stated, seems to be the most common in the current literature. Where closed-set testing means that the test video sequences are by design a member of one of the defined classes.

2.2.2 A panoramic on algorithms of video segmentation using video features

The first works dealing with the problem of video segmenting focused principally on scene cut segmentation using visual cues. Some of these works scoped on shot cut or gradual transition detection, others were relative to key frame selection.

Most of the existing methods dealing with *shot cut detection* use some inter-frame difference metric. A frame pair where this difference is greater than a predefined threshold is deemed to contain a shot cut. The simplest way to quantify the difference between two frames is to compare the intensity values of corresponding pixels. If the

mean absolute change in the intensity value of the pixels is greater than a threshold, a shot cut is deemed to be present between two frames [17]. A potential problem with this approach is its sensitivity to camera and object motion. In order to overcome this problem, methods were proposed that compare global features of each frame. The average intensity measure takes the average value for each RGB component in the current frame and compares it with the values obtained for the previous and successive frames [18]. A weakness of the global-level comparisons is that they can miss changes in the spatial distribution between two different shots. For this, Zhang proposed the comparison of corresponding regions (blocks) in two successive frames [19]; the blocks were compared on the basis of second-order statistical characteristics of their intensity values using the likelihood ratio. Another framework has been suggested that incorporate a block-matching process to obtain an inter-frame similarity measure based on motion [20]. The last feature we see used for the detection of shot cuts is edge features. Zabih proposed a method to detect shot cuts by checking the spatial distribution of exiting and entering edge pixels, known as the edge change ratio [21].

Less work than on shot cut detection has been reported on the *detection of gradual transitions* and accurate detection can still be considered an unsolved problem. Unlike shot cuts, the inter-frame difference during a gradual transition is small. For this reason, it can be difficult to distinguish between changes caused by camera and object motion and those caused by a gradual transition. As a result, detecting all of the gradual transitions often results in many false detections. One of the first attempts for detecting gradual transitions was the twin-comparison technique proposed by Zhang which compares the histogram difference with two thresholds. A lower threshold was used to detect small differences that occur for the duration of the gradual transition, while a higher threshold was used in the detection of shot cuts and gradual transitions [19]. During a dissolve, the edges of objects gradually disappear while the edges of new objects gradually become apparent. During a fade-out the edges gradually disappear, whilst during a fade-in edge features gradually emerge. This is exploited by the edge change ratio used to detect shot cuts, which was extended to detect gradual transitions as well [21]. Another method for detecting gradual transitions is to analyze the temporal

behavior of the variance of the pixel intensities in each frame. It can be shown that the variance curve of an ideal dissolve has a parabolic shape. Thus, detecting dissolves becomes a problem of detecting this pattern within the variance time series. Given an ideal dissolve the first order derivative before and after a dissolve should be zero and a positive constant during a dissolve. Alattar proposed to detect the boundaries of a dissolve by detecting two large spikes in the second-order difference of this curve [22].

The main approach to automate the video indexing process is to select representative key frames from shots or scenes to generate a video abstract (or storyboard). The collection of key frames can then be employed for further characterization and subsequent queries on the video data. An initial approach to *key frame selection* was proposed in which the first frame of each shot was selected as a key frame [23]. Zhang suggested extracting key frames using similarity measures (such as color histograms or moments) similar to those used in shot cut detection [24]. An alternative approach to find the optimal set of key frames such that the frames are maximally distinct and individually carry the most information was proposed by Vermaak [25]. Instead of using a distance criterion, Wolf used the optical flow to identify local minima of motion in a shot to identify key frames [26].

2.3 Audio classification and segmentation

In the last years intensive studies employing different features and methods have been conducted in the field of automated audio content analysis. Before reporting the history and the more important contributions in this field (see paragraph 2.3.2), an important question will be answered: how can audio analysis contribute to the advance of video segmentation techniques?

2.3.1 Why audio content analysis?

The most important (and obvious) reason why it's useful to study audio channel for

video segmentation is that even a very basic discrimination speech/music/environmental noise/silence (that is also the aim of the present project) can help improving scene segmentation, especially if combined with visual based segmentation techniques. In fact, there is more than one work showing that combining audio and visual analysis for video segmentation improves performance (for a brief analysis of some of these techniques see paragraph 2.4) [27] [28].

Other motivations for audio segmentation/classification include:

- 1) giving more reliable data (speech only) to the ASR, which minimizes word error rates, out of vocabulary and computations on non-speech data;
- 2) giving meaningful descriptors such as music, speech and so on, in conformity with the MPEG7 standard;
- 3) improving audio coding, by decreasing the bit rate for silence segments, already extracted reliably;
- 4) extracting speech it is possible to apply speaker recognition techniques for identifying and tracking specific speakers in a video, which is a very important descriptor for content indexing.

2.3.2 Important contributions in the field of audio classification

Even if this field seems full of opportunities in a quantity of practical applications, serious researches (apart some isolated works) began only in the late 90s, prove of this being the fact that are considered classical studies that were made in that years. Among these, it's worth citing the work of Foote [2], that made a system able to retrieve audio documents using acoustic similarity to a query sound. The measure of this similarity was based on statistics derived from a supervised vector quantizer; so, as the author emphasized, this model was purely data-driven and not based at all on perceptual criteria. The same pioneer emphasis can be found in another "classic" of this field, the

work of Wold, Blum and Wheaton [3], which underline the correspondence existing between the psychoacoustic characteristics of a sound (like pitch, loudness and brightness) and other more measurable ones (short-time Fourier spectra, centroid and RMS). In the same year (1996), Pfeiffer was presenting her theoretical framework and application of automatic audio content analysis using some perceptual features in which she analyzed music and recognize some sounds indicative of violence like shots, explosions and cries [29].

Many other works have been conducted to enhance audio classification algorithms. In [30], audio recordings were classified into speech, silence, laughter and non-speech sounds, to segment discussion recordings in meetings. In this framework speech was represented by sequences of feature vectors, each of which provided a short term characterization of the signal, with a HMM inducing probability densities over the sequences mentioned above. Srinivasan proposed an approach to detect and classify audio that consisted of mixed classes such as combinations of speech and music together with environment sound. The reported accuracy of classification was more than 80% [31].

Then, the work on audio for video segmentation purposes went on, and the first attempts to categorize the growing quantity of methods about it began. A first (and one of the most important) of these categorizations is probably the distinction between supervised and unsupervised techniques. *Supervised* approaches (such as [2] and [32]) use a group of a-priori known audio classes and audio segmentation is performed via a classification task, by assigning audio frames in the respective classes. *Unsupervised* techniques (like [33]) treat audio segmentation as a hypothesis test by detecting changes in the audio signal, given a specific observation sequence.

Another manner to summarize the various algorithms of audio classification/segmentation proposed in the last years is to distinguish them depending on the approach proposed in each work [34]. Using this classification scheme we have:

- a) *Model-based approaches*, where a model is created for each audio class, such as GMM [29]. These models are based on low level features in general, such as

cepstrum or MFCCs. Then, after a training phase, audio can be classified as one of these classes, with advantage of classifying and segmenting at the same time. The principal problem of this approach is the time needed for the training phase.

b) *Metric-based segmentation*, where acoustic vectors are computed using distances between neighboring windows [35]. Different acoustic features can be used to create the acoustic vectors, like auto-regressive Gaussian model parameters or DFT parameters. The advantage of this approach is that there is no need for prior knowledge on audio classes, and it can be applied successfully for real time segmentation. The most important disadvantage is that this segmentation doesn't give information about audio segments.

c) *Rule-based approaches*: in these approaches rules are created describing each class [36]. These rules are usually based on both high and low level features.

d) *Decoder-based approaches*, in which HMM of the speech recognition system is used. HMM are trained to give the class of the audio signal.

2.3.3 Audio classification algorithms with real-time constrains

It's been already underlined the fact that the field of audio content analysis for video segmentation purposes is limited and more recent than video one. Automatic video classification using audio features in a real-time context is even smaller. There are only a few methods in this field, and they are even more recent than the generic audio context study ones.

The pioneer in this sense is surely Saunders, that in 1996 presented a real-time speech/music classifier based essentially on ZCR and STE for radio broadcast [37]. The paper reported that the accuracy rate could achieve 98% when a window size of 2.4 seconds was used. The characteristic that makes it suitable for real-time applications is that it avoids the need of an FFT processor to compute spectral features, because this method uses only time-domain features (time-domain features and spectral features will

be analyzed in detail in paragraph 3.3). A block diagram of the processing flow of this method can be seen in figure 2-2.

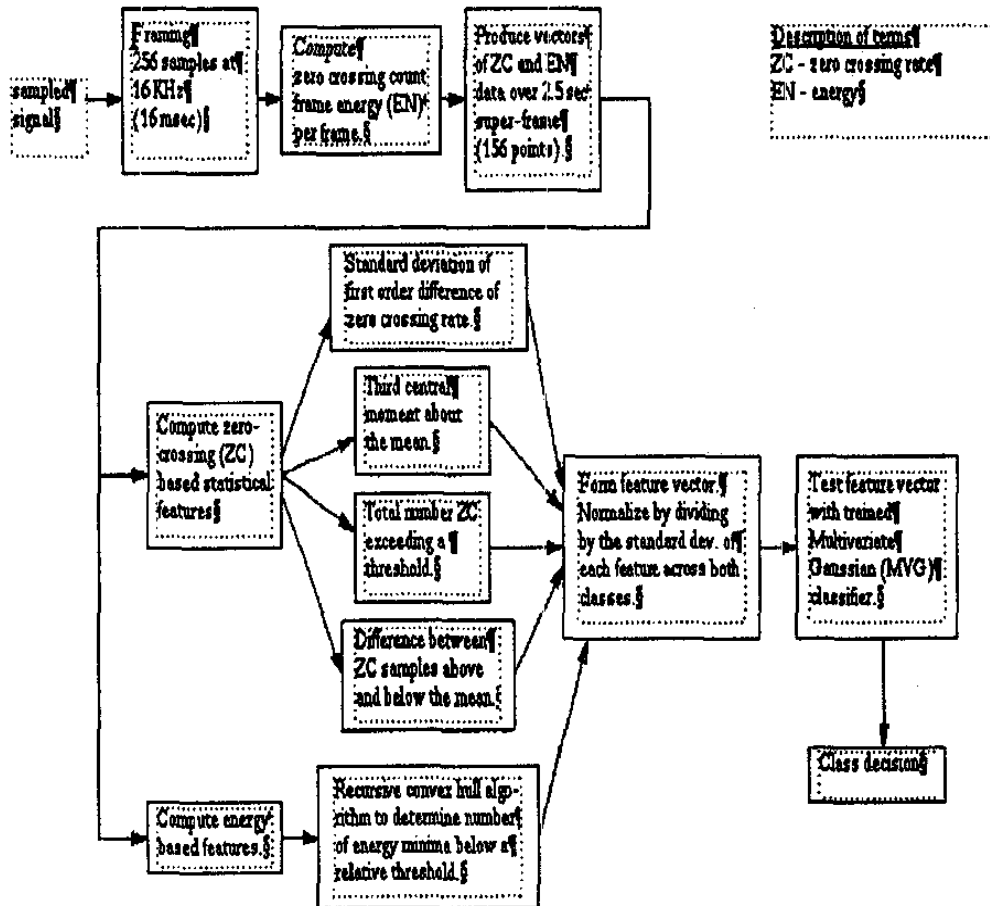


Figure 2-2: Saunders' real-time speech/music discriminator: block diagram of processing flow

In 1997 Scheirer and Slaney [38] introduced more features (13) for audio real-time classification and performed experiments with different classification models including GMM and KNN. An interesting thing of this model is that it used for the first time some variance features (5) justifying this choice with the assumption that “if a feature has the property that it gives very different values for voiced and unvoiced speech, but remains relatively constant within a window of musical sound, then the variance of that feature will be a better discriminator than the feature itself”. When using a window of 2.4 s, the reported error rate was of 1.4%.

In 2001, Zhang [39] used energy, ZCR and fundamental frequency as features and

speech/music segmentation and classification was achieved by means of a procedure based on heuristic rules. A similar approach, but using RMS instead of fundamental frequency was proposed in [40].

Ajmera [41] choose a different approach, in which a ANN trained on clean speech only was used as a channel model at the output of which the entropy and “dynamism” were measured every 10 ms. These features were then integrated over time through a 2-state (speech and non-speech) HMM with minimum duration constraints on each HMM state. Casagrande [42] adapted an AdaBoost-based image processing algorithm to the task of predicting whether an audio signal contains speech or music. Using FFT and no built-in prior knowledge of signal structure he obtained an accuracy of 88% on frames sampled at 20ms intervals. In the same year Muñoz [43] introduced a low complexity approach for speech/music discrimination, which exploited only one simple feature, called WLPC-SC, using a three-component GMM classifier.

The last contribution cited is that of Giannakopoulos [44], that in 2008 presented a speech/music discriminator for radio recordings. The main idea of his approach is that “if speech/music discrimination is treated as a segmentation problem (where each segment is labeled as either speech or music), then each of the segments can be the result of a segment (region) growing technique, where one starts from small regions (segments) and keeps expanding them as long as certain criteria are fulfilled”. The only feature this scheme adopts is a variant of the spectral entropy.

2.4 Integration of audio and visual information for video segmentation

An important trend for video segmentation and indexing is to combine audio and visual information in a single framework. This idea is more recent than the others, but probably has the most brilliant future because the results that obtained until now are significantly good in terms of accuracy and there is room for further development. For this reason it's worth mentioning some works proposed in this field.

In the method proposed by Huang [45] a small set of audio features was used including the silence ratio, the speech ratio and the subband energy ratio (extracted from the volume distribution), the pitch contour and the frequency domain. These features were combined with the color and motion information to detect scene and shot breaks. In the approach presented by Naphade [46], subband audio data and color histograms of one video segment were integrated to form a “Multi-ject,” and two variations of the HMM were used to index Multi-jects. Experimental results of detecting the events of “explosion” and “waterfall” were reported. In the approach of Boreczky and Wilcox [47], color histogram differences, cepstral coefficients of audio data and motion vectors were used together with a HMM approach to segment video into regions defined by shots, shot boundaries and camera movement within shots.

Two years later (2000) Chang and Sundaram [4] discussed their model, in which they proceeded separately to the segmentation of the video and audio data into scenes. The audio segmentation algorithm determined the correlations amongst the envelopes of audio features, while the video one determined the correlations among shot key-frames. Then, they fused the resulting segments using a kNN algorithm that was further refined using a time-alignment distribution derived from the ground truth. The algorithm, tested on the first hour of a commercial film (a difficult data set), achieved a scene segmentation accuracy of 84%. Meanwhile, Jang, Lin and Zhang [5] were presenting their audio aided video scene segmentation scheme. This model consisted of an audio segmentation and classification algorithm that segmented an audio stream into speech (further segmented into parts of different speakers), music, environment sounds and silence segments. The shots within an audio segment were grouped together and marked as related. Then, color correlation analysis between shots was performed and an “expanding video grouping” algorithm was applied, such that shots whose objects or background were closely correlated were grouped. In this way, a sequence of shots was grouped into a scene only when both visual content correlation analysis and audio segmentation detected a common scene boundary.

Another contribution in the field of the integration of video and audio features for video segmentation add the textual analysis is a system developed to help home users in

browsing broadcast news video programs. This system, created by Zhang [48], performs at the bottom level image analysis to segment a video sequence into shots. Above this level is the “group of shots” one. These groups are extracted by grouping process using both visual and audio information. At the top level (“story” level), video and audio features are combined to detect anchorperson in the program as the indicator of story boundaries, while NLP classifies these stories into pre-defined categories based on text information associated with each news item, and associate stories of each news item linked with the news story.

2.5 Summary

There have been many algorithms proposed for video segmentation, most of which use visual information. Those using audio information accomplish this task using audio information alone or in conjunction with video features, but only a few of these (as seen in paragraph 2.3.3) are able to respect real-time constrains. This suggests that an algorithm is required able to comply with real-time requirements, and this is the principal aim of the present Master Thesis. The second aim of this work is to analyze the structural differences between this algorithm and another one that, too, could be able to perform video classification using only audio features, but without the objective to work well in real-time. This is important to discover the differences between the two approaches in terms of computational complexity (and, subsequently, velocity), classification accuracy and adaptability to different situations.

3 Work design

3.1 Introduction

This chapter is fully dedicated to the explanation of the preparation process of the present work and to the motivations that drove the author to do a choice instead of another *before* the beginning of the work.

Subsection 3.2 gives a summary of the salient properties of each audio type in which audio will be classified: speech, music, noise and silence.

Subsection 3.3 consists principally of a description of all the low level features chosen to help audio classification in this work.

Subsection 3.4 describes the basic idea for the implementation of the classification algorithm.

3.2 Classified audio types: a definition

Classifying an audio file means first of all deciding in what and how many categories we want to split this file. Simple classifiers (especially those working in real-time) often use a speech/music discriminator. Others classifiers (a little more sophisticated) usually segment an audio file into four categories: speech, music, noise and silence. In these four categories it's possible to find a lot of subcategories; this solution is usually used in the case of more specific applications. For example, music can be further divided in pure music, harmonic sounds, melody, and so on; speech can be classified in male and female voice, or in a more specific way, each different speaker in an audio clip can be identified and distinguished from the others; noise can be divided into environmental sounds or hamming noise (caused by, for example, the way the file was registered). Environmental sounds can be further classified into a lot of categories, such as cheer,

applause, gun-shot, explosion, ring, police siren and so on. Furthermore, some systems need the capacity to discriminate between more complex sounds, the so-called hybrid ones. In a movie, for example, there are a lot of scenes in which someone talks over background music, or environmental noise.

The more types of sound a classifier identifies, the better this classifier is. Theoretically, a perfect classifier should be able to categorize an audio clip into the totality of existent categories and subcategories. Obviously this can be done, but it's incredibly expensive from a computational point of view. So, each application, depending on its purposes, is more or less specialized in one field or another.

In the present work, the choice has been that to develop a classifier able to distinguish between human voice, music, environmental noise and silence parts. A classifier able to distinguish only music and speech parts is a solution used a lot of times in past frameworks, especially those that had to work with real-time constrains. A classifier working well in these constrains and slicing an audio file into the four categories above mentioned would be a step forward in this direction.

In the next four subparagraph there is a brief description of each audio type, starting from human voice, going on with music and noise and finishing with silence.

3.2.1 Voice

The vocal signal is constituted by an acoustic pressure wave created by movements of the body parts delegated at the sounds production. These body parts (see also figure 3-1) are:

- *diaphragm* (dome shaped muscle positioned in the lower part of the ribcage);
- *lungs*;
- *windpipe*;

- *larynx* (organ that produces the voice because it contains the vocal chords);
- *pharynx cavity* (throat);
- *oral cavity* (that changes its shape according to the position of lips, jaw and tongue);
- *nasal cavity*.

The sounds emitted using this apparatus can be classified according to various criteria, each considering a different aspect concerning the emission phenomenon. According to the tonality, vocal sounds can be classified in vowels and consonants:

1) *vowels* (or voiced sounds) are mainly characterized by their high power value, due to the fact that they are produced by the vibration of the vocal chords with the vocal tract not contracted. The frequency of this vibration corresponds to the fundamental frequency of the voice signal and determines the pitch. This pitch is of 50 - 200 Hz for a man, and 120 – 500 Hz for a woman. We are talking about a harmonic sound, so it is characterized by a central frequency (the above mentioned pitch frequency) whose power is maximum, and a series of harmonics whose power lapses as we move away from the central one. At these frequencies (the so-called *overtones*) the vocal tract acts like a resonant filter. A little part of the energy is produced by air turbulences in the vocal tract. Vowels' duration is very regular, and the majority of their power is concentrated at low frequencies;

2) *consonants* (or unvoiced sounds) are sounds generated without the vibration of the vocal chords. Their unique source is the air turbulences produced in the vocal tract. The power of consonants is quite uniformly distributed at all frequencies, so that the power spectrum of the correspondent signal seems like that of noise.

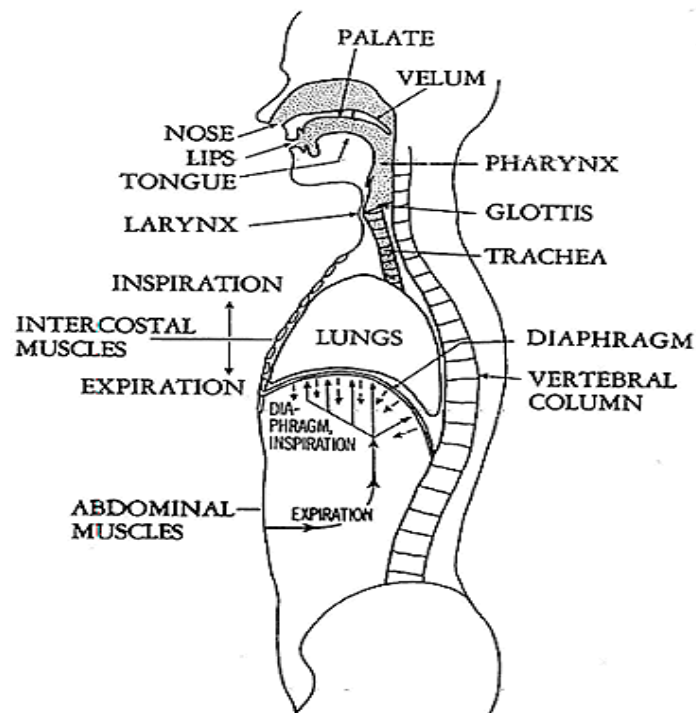


Figure 3-1: Human speech apparatus

The upper limit of the voice signal's bandwidth is around 8 kHz. Furthermore, the voice pattern tends to have a quite regular structure, in which a fragment can be formed by some words, and these are in turn divided into consonants and vowels. Vowels regions can be easily identified for their higher energy levels, alternating with regions characterized by lower energy levels, corresponding to consonants.

3.2.2 Music

The principal feature of musical sounds, that distinguish them from the vocal ones, is their periodicity. Music sounds are also characterized by a more or less regular rhythm, thing that can be noticed studying its amplitude variation in the time domain.

Music spectral analysis reveals the presence of tunes. They consist in the presence of a series of simultaneous frequencies during an extended period of time.

The majority of energy in a musical signal is concentrated at low frequencies.

Looking at the energy distribution, it's easy to notice that music is not characterized by the presence of peaks (like the vowels ones of voice) and presents little variations over time.

The main difference with a vocal signal is that, especially when we consider “loud” music, a part of the harmonics of this signal goes over 8 kHz, sometimes passing the human hearing limit of 20 kHz. This is for example the case of trumpets and other brass music instruments. On the other hand, guitar and piano have narrow frequency ranges. Therefore energy distribution is not enough to categorize all music. Another difference is the fact that music rarely has periods of complete silence.

3.2.3 Noise

Noise is probably the audio class classifiable with more difficulty because it has a random spectra, that means that occupies equally almost all the frequency bins.

While the Pure Random Noise has no peak frequency bins, less intense random noise (the so-called brown noise) may be characterized by some small peaks.

Another peculiarity of noise is the presence of many short discords between different frequency tracks.

3.2.4 Silence

Silence is surely the easier audio category to identify, because it is characterized by an energy level extremely low. Samples' value should be ideally zero, but most of the times the presence of some background noise makes it higher than this value.

3.3 Low-level features choice

The objective of the present work, as explained above, is to classify an audio file into four fundamental categories: music, speech, noise and silence. To do so, it is first necessary to read an audio file and to read and save in some way the waveform enclosed in it (see paragraph 4.2 for details).

Having the waveform, next step consists in extracting from it some low-level features. The choice of these features is a very important step in the whole process; choosing the right features can result in effortlessly good classification results, while choosing the wrong ones (or too less, or even too many) is certainly a complication for the next steps, whatever kind of classification algorithm we want to use. Unfortunately, literature cannot help a lot in this sense because this choice depends a lot on the purpose of the application. For this reason, finding useful features for a better subsequent classification is especially a matter of intuition and (a little) of luck. The benefit of a feature can be seen only having a graph showing its variations along the audio file. Only at this point you can decide whether to hold that feature or to focus on others. Besides, a feature can be useful to identify an audio type (such as voice) and not others, or the differences between two particular different audio types (such as voice and music). Summarizing, the features must have these two important properties:

- *selectivity*: they have to suffer significant modifications passing from an audio type to another;
- *controlled computational complexity*: signal processing time required for extracting each feature must be proportional to the amount of information that feature provides.

Audio low-level features can be divided into two big groups: *time-domain* features are those extracted directly “playing” in some way with the digital values of the waveform, while the *frequency-domain* ones are obtained starting from the Fourier transform of the digitalized audio signal, which shifts the field of analysis from a temporal to a frequency meaning.

Besides presenting each of the features that will be used in the rest of the work, in the following paragraphs there will be a brief description of their behavior with respect to each audio type.

3.3.1 Time-domain features

3.3.1.1 Energy

The energy function offers a representation of the audio signal variation during the time line. Its value is in direct ratio to the samples amplitude, and it has the following expression:

$$E[n] = \sum_m (x[m]w[n - m])^2$$

In this formula $x(m)$ is the m -th sample of the audio file and $w(m)$ is a rectangular window of length N and height $1/N$:

$$w(m) = \begin{cases} 1 / N, & \text{when } 0 \leq m \leq N - 1 \\ 0, & \text{otherwise} \end{cases}$$

With ZCR (see paragraph 3.3.1.2), the energy function is one of the most used time-domain features in audio analysis. Its behavior in relation with the different audio types can be summarized as follows:

- *silence* is characterized by energy values very small, ideally zero. The higher the SNR, the better one can distinguish between silences and speech pauses on one hand, and audible sounds on the other hand;
- *vocal* sounds: consonants have a function energy value considerably lower than vowels;
- variations of the energy function along the time line can make clear the presence of any rhythmic or periodic sound (often *musical* sound).

3.3.1.2 Zero Crossing Rate

Zero Crossing Rate (*ZCR*) measures how many times the audio signal crosses the zero axis in a determined period of time. This definition means that the *ZCR* value increases by one whenever two consecutive audio samples have a different sign. Its usefulness lies in the fact that it is a simple measure of the frequency content of the audio signal, because audio intervals characterized by high frequencies will also have a high *ZCR* value and vice versa. In other words, it represents an average of spectral energy distribution.

The mathematical function that calculates *ZCR* for a determined time interval is the following:

$$ZCR[n] = \sum_{m=-\infty}^{m=\infty} |\text{sgn}[x(m)] - \text{sgn}[x(m-1)]|w(n-m)$$

Here, $x(m)$ is the m -th sample of the audio file and $w(m)$ is a rectangular window of length N and height $1 / 2N$:

$$w(n) = \begin{cases} 1 / 2N, & \text{when } 0 \leq m \leq N - 1 \\ 0, & \text{otherwise} \end{cases} \quad \text{sgn}[x(m)] = \begin{cases} +1, & \text{when } x(m) \geq 0 \\ -1, & \text{otherwise} \end{cases}$$

In the following will be explained *ZCR* peculiarities for different audio types:

- *vocal* sounds: consonants, and in particular fricative sounds, are characterized by high *ZCR* values because their energy is concentrated at high frequencies. Vowels, instead, have low *ZCR* due to the fact that most of their energy is concentrated at frequencies lower than 3 kHz;
- *musical* sounds have an irregular *ZCR*, but with some properties that make them easily distinguishable from voice ones: lower average value and smaller variation range.
- *noise sounds* have a high *ZCR* value. This behavior is justified by the fact that noise energy is spread over the frequency spectra, so with an higher component in

high frequencies. The same consideration is good for *silences*, which often contain only a little noise component.

3.3.2 Frequency-domain features

Frequency-domain features extraction requires performing previously the Fourier Transform of the audio signal. Using this mathematical instrument is necessary to convert the audio samples directly extracted from the file (that correspond to its variation during the time line) into their amplitude in the frequency domain. The practical result of this operation is a continuous graph in the case of an analog signal, and a histogram if the signal we are studying is digital. If we are in the latter case, the possibility is to use Discrete Fourier Transform (DFT) or Fast Fourier Transform (FFT). DFT mathematical expression is:

$$X(m) = DFT\{x(n)\} = \sum_{n=0}^{N-1} x(n) e^{-j2\pi mn/N}, \quad m = 0, 1, 2, \dots, N-1$$

Here, $x(n)$ is the audio sample in the time domain, N is the number of points of DFT and m is a positive natural index whose value ranges from 0 to $N-1$. An algorithmic implementation of this expression requires an elevated number of operations (in the order of N^2). A FFT algorithm reduces this complexity; the most famous of the existent FFT algorithms is the Cooley-Tukey one, that transforms a digital signal using a number of operations in the order of $N \cdot \log N$, with the constraint that the number of points must be a power of two [49].

3.3.2.1 Energy distribution across frequency bands

The audio signal energy distribution across different frequency bands allows the understanding of the energy content of the signal and it's useful to understand which is the frequency limit of each part of the signal. Calculation is quite simple, and consists only in splitting the audio frequency content into smaller parts, each part corresponding

to a predefined band.

The choice of the upper and lower limit of each band strongly depends on the audio types' characteristics. As said in paragraph 3.2.2, vocal signals are limited at a lowest frequency (8000 Hz) with respect to the musical ones (top limited at 16000 Hz, even if a great part of their energy is at low frequencies) and to the noise ones (whose energy distribution is spread over the whole frequency spectra). Starting from these considerations, a choice was made to divide the audio signal into the following four frequency bands:

- *band* b_1 : frequencies below 1000 Hz;
- *band* b_2 : frequencies between 1000 and 8000 Hz;
- *band* b_3 : frequencies between 8000 and 16000 Hz;
- *band* b_4 : frequencies above 16000 Hz and under the limit imposed by the fundamental sampling theorem (Nyquist theorem): the sample rate divided by 2. So, if the sample rate (as in most cases is) is 44100 Hz, the upper limit of the b_4 band will be 22050 Hz.

The mathematical expression for the calculation of energy distribution across different frequency bands is the following:

$$Energy\ band_n = \frac{\sum_k X(k)^2}{\sum_l X(l)^2}$$

Here, k goes from the lowest to the highest frequency of the n -th band, while l goes from the lowest to the highest frequency of the whole spectra. Dividing for $\sum X(l)^2$ means normalizing the energy of the n -th band for the total energy spread among all the frequency bands.

3.3.2.2 Root Mean Square

The Root Mean Square (RMS), also known as quadratic mean, is a statistical

measure of the magnitude of a varying quantity. It is especially useful when variables are positive and negative, and it can be calculated using the following expression:

$$RMS = \sqrt{\frac{1}{N} \sum_{n=1}^N X_n^2}$$

Here, X_n is the n -th value of the frequency amplitude, and N is the length of the analysis window.

RMS behavior for the four different audio types is the following:

- *silences* are characterized by a very low RMS level (ideally zero) . As the energy function, this feature can be used to easily distinguish between this audio type and the others;
- *vowels* and consonants are characterized, respectively, by a very high and a very low RMS value. This gives speech a RMS curve shape that makes vocal sounds quite distinguishable from the other kinds of sound;
- *noise* has often a littler variation range compared to other audio types, especially speech.

3.3.2.3 Spectral Centroid

Spectral centroid (SC) of an audio fragment corresponds to the average of its frequency content. For this reason, its value is simply a frequency expressed in Hz whose value depends on the audio type and on its dominant spectral features.

For an audio fragment of a determined length (corresponding necessarily to the number of FFT points), the equation with which it is possible to calculate the value of this audio feature is:

$$C = \frac{\sum_k kX[k]}{\sum_k X[k]}$$

In this formula $X[k]$ is the FFT value associated with the frequency k .

The behavior of the spectral centroid for the different audio types is the following:

- *consonants*, being unvoiced sounds and having higher frequency components, are characterized by a SC much higher than *vowels*. As a consequence, it's logical hypothesizing an higher variance of the vocal signal with respect to all the other audio types;
- *musical* sounds will be presumably characterized by centroid values with smaller variance than voice ones. Often these values are also higher than other audio types;
- *noise* sounds should have generally very high centroid values.

3.3.2.4 Roll-off point

This parameter (that can be indicated also with the abbreviation RP) is the measure of the frequency (in Hz) under which 95% of the audio signal energy is concentrated. For each audio frame, it can be calculated using the following equation:

$$\sum_{k < RP} X(k) = 0.95 \sum_k X(k)$$

Here, $X[k]$ are the values of the FFT applied to the audio signal. Its behavior is very similar to that of the spectral centroid for all the audio types. In particular, *music* sounds will generally have a higher roll-off point than voice ones; this is due to the fact that voice is upper limited at 8000 Hz, while music have a wider range extended up to 16000 Hz. Variance of *voice* sounds roll-off point is higher than other audio types for the great differences existing between vowels and consonants (consonants are characterized by a roll-off much higher than vowels for the same reason explained for the spectral centroid).

3.4 An idea for the classification algorithm

After extracting all the features, it is possible to proceed to the classification.

In the work design phase, only a generic scheme for the classification algorithm has been projected (see figure 3-2). Fundamentally, the algorithm will consist of two main parts:

- a first part for the silence extraction, that often is very simple and can be based on a very small set of features;
- a second part delegated to the discrimination between noise, music and speech fragments of the audio clip. This distinction constitutes the core of the entire work, and the way of performing it is the most important choice to do dealing with the problem of audio classification. The fundamental idea for this step is to use a vector for each audio type, each element of the vector containing the number of points obtained by an audio type for a determined frame. The higher the number of points obtained by an audio type for one frame, the higher the probability for the same frame to be classified as that specific audio type.

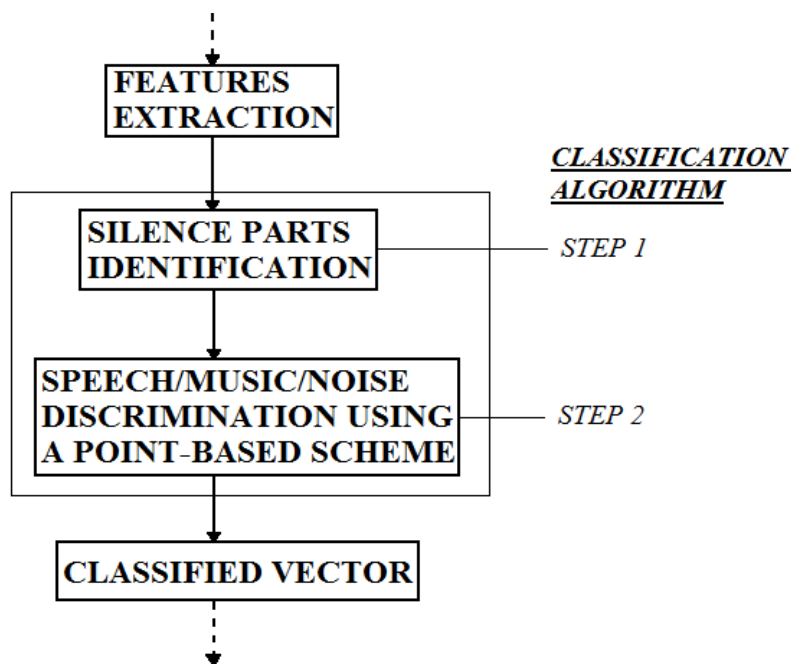


Figure 3-2: High-level classifier scheme

So, the classification algorithm will assign, on the basis of the features previously extracted, a certain number of points for noise, speech and music for each frame in which the audio clip will be divided. Each frame will be subsequently assigned to the audio type receiving the higher number of points. But which will be the link between

feature values extracted for one frame and the number of points assigned to each audio type for that frame? The idea in this sense is to apply a series of conditions to that values, based on heuristic considerations that, if satisfied, can add or subtract points for all or some of the audio types regarding the frame those values belong to. Then, it would be possible to apply a normalization algorithm on the obtained results; in this way, it would be avoided the occurrence of a frame classified in a certain type in the middle of a group of frames all classified as another audio type.

Only a brief description of the ideas underlying the practical realization of the project has just been realized. For the details regarding the classification algorithm and the conditions applied on each frame and for each low-level feature extracted from the audio clip, see chapter 4.

Before ending the chapter, it is worth mentioning the fact that a points-based audio classification has never been proposed in the past literature. Of course, using this new method represents a risk in case of poor results; otherwise it could constitute a help in the field of automatic audio classification and a huge possibility for further research. The results of this new approach will be analyzed in chapter 5.

4 Work development

4.1 Chapter organization

This chapter is surely the “core” of the present document. Here a presentation is proposed of all the steps that brought to the realization of the project. Each following paragraph corresponds to one of these steps, in the order they were carried out. Furthermore, as the project consisted of the realization of a “normal” audio classifier and another able to work well with real-time constraints, more than one paragraph of this chapter will be divided into sub paragraphs, two of them often (if there are consistent differences in the scheme used) related to one of the two parts of the project. Below the contents of each paragraph in this chapter are quickly described:

- paragraph 4.2 deals with all the problems related to the extraction of the samples from an audio file, audio format choice and its motivations;
- in paragraph 4.3 is described the way audio low-level features are extracted;
- paragraph 4.4 consists essentially in a comment (with the help of graphs) of the results obtained plotting previously extracted features, and thus gives some headlines for a better understanding of the subsequent classification step;
- in paragraph 4.5 there is a detailed description of the algorithm used for the classification. Two sub paragraphs are dedicated to the two main concepts on which the algorithm is based: the utilization of heuristic conditions and point vectors. Attention is paid also to the explanation of the differences existing between off-line classifier and the real-time one;
- paragraph 4.6 fundamentally explains the process thanks to which are created the graphs illustrating classification results;
- last paragraph (4.7) simply consists of the flow charts summarizing the way both off-line and real-time classifiers work.

4.2 Audio file opening and reading

The first thing to do when working on digital audio analysis is to wisely choose the *format* of the input audio file. A lot of audio formats exist, with more or less specific purposes and domains; most of them have been introduced in the market in the very last years. A brief classification is necessary before going on with the description of the work flow.

There are three major groups of audio file formats classifying them according to their *compression scheme*:

1. *uncompressed* audio formats, such as WAV, AIFF and Au file format;
2. formats with *lossless* compression; this means that their compression algorithm allows the exact original audio data to be reconstructed from the compressed one. Some of the audio formats belonging to this category are: FLAC, ALAC, WavPack (filename extension .wv) and Monkey's Audio (filename extension .ape);
3. formats with *lossy* compression, where compressing data and then decompressing it retrieves data that are slightly different from the original but still useful for a lot of applications. Compression algorithms used by these formats are characterized by a compression ratio a lot higher than lossless ones. The most used audio formats belonging to this category are: MP3, Vorbis (filename extension .ogg), WMA, ATRAC and AAC.

Being the purpose of this project very general, format of the input audio file should be preferably uncompressed. This choice gives the possibility to adopt it in a scheme like that shown in figure 4-1. Converting an audio file from a compressed format to an uncompressed one (and vice versa) is not a problem, existing on the market a variety of audio format converters, usually very fast and completely free. The most popular of them are Any Audio Converter, AVS Audio Converter, Switch Audio File Converter. Choosing an uncompressed audio format is not a limitation even in the case you want to extract the soundtrack from a video (such as a movie, a documentary, a football match and so on), because there are also in this field a lot of free and well-working programs

capable to extract from a video only the audio with a wide choice of formats, sample rates, number of channels and so on. A lot of products accomplishing this task exist on the market such as FormatFactory, Quick Media Converter or MediaCoder.

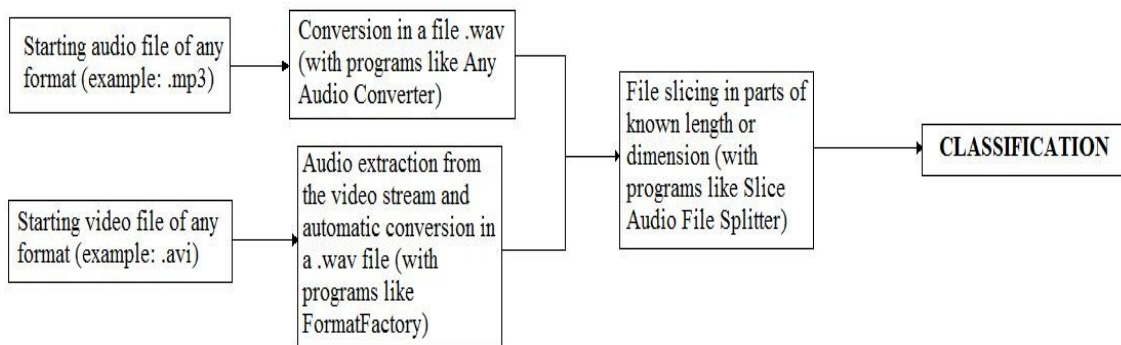


Figure 4-1: Scheme of the format conversion process

The audio format of the file that has to be classified should be also widely used and preferably free. For these reasons the choice fell on the *WAV* format, used by Windows and for this reason surely the most popular between the existing uncompressed audio file formats (for a detailed description of WAV format, see Appendix A).

Another important thing to do to prepare the field for further work is that of slicing the obtained .wav file into smaller parts. This operation is necessary not only for the obvious reason of handling files of a length or dimension set by the user, but especially because WAV files have an upper limit in dimension (4 GB) they cannot exceed. This is because WAV format uses a 32-bit unsigned integer to record the file size header. This task can be accomplished by more than one program, one of these is named Slice Audio File Splitter, very simple and effective.

Finally, using both WAV format and programs capable to obtain it from other audio formats or from video ones (for the realization of this project were used, respectively, Any Audio Converter and FormatFactory) and to slice an audio file into smaller parts (for this purpose Slice Audio File Splitter was used) surely constitutes a good starting point to create a big database of files ready to be opened, read and classified.

4.2.1 Practical implementation

Having ready the file .wav of the desired length or dimension, the next step is to prepare it for classification. To do so, it is necessary to extract its data and header information (some of which will be very useful for the next steps of the project).

The simpler way to proceed in this sense would be that of using a predefined function; Matlab offers one whose name is *wavread* that automatically opens a WAV file and extracts its data and information from its header such as the sampling frequency and the number of bits on which each sample is represented. Unfortunately, it cannot work at all if it finds an error in one of the header fields of the file; this case is not so rare, especially when programs are used to convert an audio or video file of any format into a .wav one.

For this reason, both for the normal and for the real time classifier it has been necessary to create something able to accomplish the same task of *wavread* avoiding crashes due to eventual header errors. This has been done using C++ language, more efficient of Matlab when dealing with a lot of exceptions.

Two functions have been projected to open and read efficiently a WAV file. Both are stored in a file .cpp whose name is *OpenWAV*:

1. *OpenFile* is a short function that takes in input the name of the WAV file to open (put by the user in the command prompt) and opens it using the function *open* included in the *fstream* C++ library. *OpenFile* returns to the main function an integer value equal to -1 if file opening failed for some reason (file does not exist, access to it is forbidden, etc.). The reason is explained to the user in the command prompt correspondingly to the value assumed by variable *errno* (standard C++ library *errno.h*).
2. If file opening is successful, program goes on passing to the function *ReadFile* the name of the WAV file to read and the name of the .dat file in which putting the values taken from the WAV. *ReadFile* basically uses functions *read* (library *fstream.h*) to read all the header information (that user can see being printed on the

command prompt), *lseek* (library *io.h*) to move across the WAV data that have to be read and function *fprintf* to print on a .dat file all the waveform values stored in the WAV. In *ReadFile* all the exceptions are handled, included that of having some extra fields in the WAV header. Another important thing to say about this function is that it stores in the first position of the file .dat the sample rate of the WAV file (information extracted directly from one of the header fields – SampleRate in the *fmt* subchunk). This operation is necessary because the sampling frequency will be used by other programs using a different programming language (MATLAB) to calculate the duration of the audio file and to transform a measure expressed in seconds to one expressed in samples.

The program works only if WAV files sampled on one channel (mono) are used. This does not constitute a limitation to the scope of the program, for the fact that programs like those used (for example, FormatFactory) to convert a video or an audio file of a generic format into a WAV one, give user the possibility to choose the number of the channels of the WAV file.

4.3 Features extraction

Once the WAV file has been opened and read, it is necessary to create something able to extract the low-level features from the .dat file where the waveform has been stored. In the following two paragraphs are described the different proceedings adopted to accomplish this task in the off-line and in the real-time case.

4.3.1 Off-line classifier

The off-line classifier doesn't have time constraints. For this reason, it is possible for its realization to perform the extraction of the features all in once and before the beginning of the classification step. To do so, a M-file (MATLAB script file) has been projected, consisting of a single function named *FeaturesExtraction* (file

“FeaturesExtraction.m”).

FeaturesExtraction takes in input two variables that user has to put in a MATLAB ambient:

1. the name of the .dat file in which the waveform is stored (variable *filename*);
2. the number of the samples belonging to one analysis window (variable *WindowSamples*).

After reading the content of the .dat file, and storing it in a variable called *data*, the function calculates some necessary parameters such as the number of the analysis windows in which the audio clip will be divided (*NumberOfWindows*). Then, it begins the proper feature extraction. A *for* cycle that goes from 1 to *NumberOfWindows* starts, and for each round of this cycle the low-level features referred to a window are calculated. Following the order the program follows to compute them, these are: Zero Crossing Rate, Energy, Spectral Centroid, Root Mean Square, Frequency Subband Energy and Rolloff Point. A description of all these features can be found in paragraph 3.3, MATLAB instructions used to evaluate each feature for one window are listed in table B-1 (Appendix B), while graphs showing the behavior of each feature for different audio types can be seen in paragraph 4.4.1.

To compute the frequency features, it is necessary to perform the FFT of the data contained in a single window. For this target, the standard Matlab function *FFT* is used. This function takes in input the waveform values of a window and their number, and returns the complex FFT values. The modulus of the first half of them (for the sampling theorem, only the first half of the FFT values is useful for further calculations) constitutes the basis starting from which all frequency-based low-level features are computed.

After computing all low-level features for each analysis window constituting the audio clip, the function put them orderly in a series of files .dat created by the function itself. Each file contains all the values assumed by one feature along all the length of the audio clip. Their name is in the form “FEATURENAMEwavfilename.dat”. For

example, a file containing the ZCR values computed for the file “abc.wav” will be named “ZCRabc.dat”. For the Frequency Sub band Energy four files are created, one for each sub band in which the frequency spectra is divided.

4.3.2 Real-time classifier

The MATLAB function that in the real-time case performs both the extraction of low-level features and the classification of the audio clip into silence, music, noise and speech is called *RealTimeClassifier* (file “RealTimeClassifier.m”). This function takes in input three variables whose value is set by the user:

1. the name of the file .dat containing the waveform previously read from the WAV audio file (variable *filename*);
2. length in windows of an analysis frame (variable *FrameLength*);
3. length in seconds of a window, the subunit of a frame needed for the extraction of the low-level features (variable *WindowLengthInSeconds*).

The latter of these three variables is used by the program only to calculate *WindowSamples*, that corresponds to the number of WAV samples belonging to a single window ($WindowSamples = round(SampleRate * WindowLengthInSeconds)$). The difference with the off-line case is that there user putted directly *WindowSamples*, while for the real-time classifier a different approach has been used only to make more intuitive the effective length in seconds of one frame; in fact, this can be obtained simply multiplying the length of one frame in windows for the length of one window in seconds ($FrameLengthInSeconds = FrameLength * WindowLengthInSeconds$).

Features extraction scheme adopted for the real-time classifier will be described analyzing the *differences existing between this case and the off-line one*. These differences are fundamentally two:

- the utilization of less low-level features with respect to the off-line case. In fact, *Root Mean Square* and *sub band 4* (that corresponds to the frequencies above 16000 Hz) *energy distribution have not been considered* in real-time classification

process.

- the *extraction of low-level features is interleaved with the classification for every frame*; on the contrary, in the off-line classifier the extraction of the same features is performed all in once before the beginning of the classification step. The obvious explanation of this different approach lies in the fact that real-time classifier has to emulate the situation of a video in streaming. In this case, the approach used in the off-line case would be completely ineffective for the real-time one, because it would be able to return the first results only after some seconds from the beginning of the execution of the audio clip.

RMS has been excluded because during the classification step only its normalized values are used. Normalizing means dividing each value belonging to the vector related to one generic low-level feature for another value: this can be, for example, the mean applied on that feature or the maximum value that the same feature reaches during the audio clip. This operation is very useful when dealing with features, such as temporal energy or RMS, which amplitude is strongly dependent on the volume of the audio clip. A normalized measure gives the possibility to apply a condition being more confident about the probability it has to effectively work on a wide range of audio clips recorded at different volume levels. Unfortunately, normalization means also almost doubling the computational time for each feature calculation, and this is why Root Mean Square has been excluded from the group of low-level features calculated and then used for the real-time classification.

Sub band 4 energy distribution is not used at all in the real-time classifier because, as it will be better explained in paragraph 4.4.1.3, it doesn't provide a sufficient amount of information useful for the subsequent classification step, the word “sufficient” being used here with the following meaning: computational costs related to sub band 4 energy calculation are higher than benefits that this feature provides for the classification results.

In practice, the system regulating the extraction of the low-level features for every window and the classification for every frame of windows is composed of two nested *for* cycles, using the scheme shown in table B-2 (Appendix B).

The rest of function *RealTimeClassifier* will be analyzed in paragraph 4.5.3. There it will be described the functioning of the classification scheme applied on every frame and the variables the function returns to the MATLAB prompt for further usage.

4.4 Features plotting

Once audio low-level features are extracted and before searching for reliable classifying rules to apply on them, it is necessary to graphically see in some way how these features vary along the time line. This has been done realizing another MATLAB function whose name is *ReadAndPlot* (file “ReadAndPlot.m”).

Before describing in detail this function, it is important to emphasize the fact that it works only in the off-line case and not in the real-time one. There are two reasons justifying this choice:

1. being the proceeding used to evaluate audio features the same both in the off-line and in the real-time classifier and using real-time classifier only a smaller subset of features, it did not seem necessary to create twice the same graphs;
2. real-time classifier works with strict time constraints; adding to the classifier function something able to memorize the frame-related features in an external file, or even plotting them in a graph and make an output of it are I/O operations highly time consuming in computational terms.

ReadAndPlot does not take input variables, but using the MATLAB construct `dir('*.*')` it is able to read all the files `.dat` in the folder in which the function works. In this way, it can read all the features file and the file containing the waveform directly extracted from WAV file. Then the function creates a graph for each feature (and for the waveform) making a massive use of the standard MATLAB function *plot*. Each graph is then formatted using some other MATLAB functions such as *xlabel*, *ylabel*, *legend* and *title*. The last thing this function do before passing back the control to MATLAB main page is saving the obtained graphs in files `.jpg` named automatically by the function as “namedatfeaturefile_image.jpg” (example: “ZCRabc_image.jpg”).

4.4.1 Audio types behavior as a function of low-level features

In this paragraph the results will be analyzed of the function *ReadAndPlot* applied on a WAV file. In this way it will be possible to see how low-level features vary passing from an audio type to another. This analysis is the last necessary step to do before passing to a comprehensive description of both off-line and real-time classifiers' functioning.

The audio clip used for the considerations that will be presented in this paragraph is an extract from the film “Sin City”. It lasts 40 seconds and it has been chosen because it is characterized by a first part (approximately 18 seconds) of music, a little pause of noise and the remaining part of speech. Furthermore, audio of this clip is quite clean for the almost total absence of noise due to a bad registration quality. For these reasons it constitutes a good starting point to explain the differences existing between music, speech and noise audio parts watching only the features extracted from a particular clip.

Of course, considerations made in the following chapters cannot be generalized to all the audio clips, because each has a different quality level. The quality of an audio clip depends on a lot of factors, including the quality of the registration and the related SNR: bad quality audio files will be characterized by a lower SNR than high quality ones, making it more difficult to correctly classify the file. Also volume level influences a lot values related to some low-level features along the clip. Another consideration that must be taken into account is that most of the times, if the audio clip is extracted from a film or a TV program, more than one audio type can be simultaneously present in one generic moment. This aspect will be further analyzed in paragraph 4.4.1.1.

4.4.1.1 *Energy*

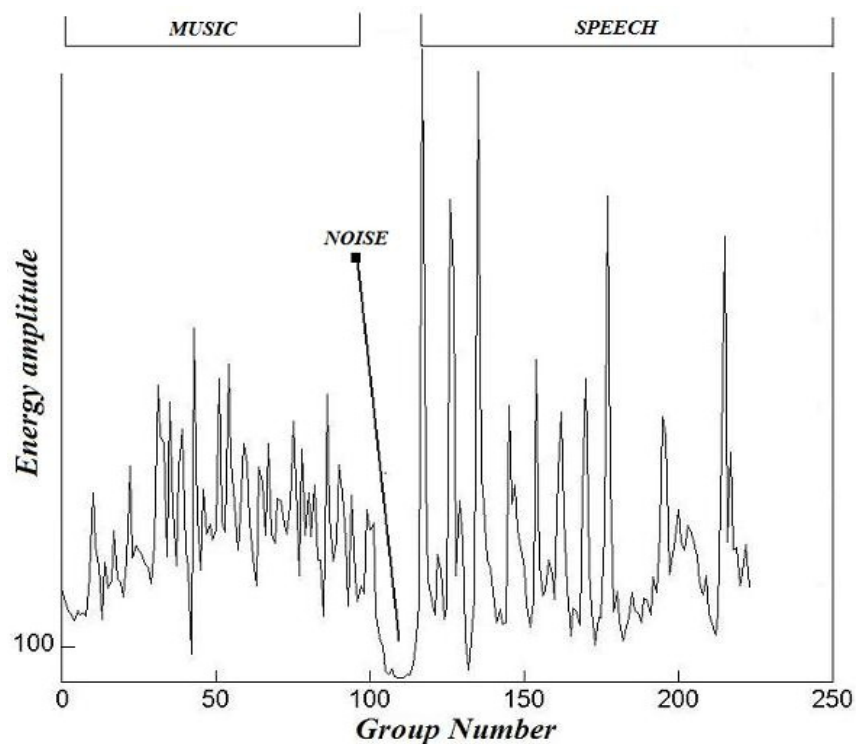


Figure 4-2: Energy function graph: an example

The figure above represents the graph of the energy function. From it, it is possible to see that generally music has a distribution more uniform than speech. Local peaks of musical sounds are quite regularly spaced, the distance between local maximum and minimum being not so big with respect to vocal sounds. This is due to the fact that musical sounds are harmonic and characterized by a mostly regular rhythm vocal segments often lack of.

On the other hand, speech segments appear to be characterized by a big amplitude difference for the energy. This is due to the fact that vowels have a very high energy content, while consonants and pauses not at all. If the audio signal has not noise or background music during a speech segment, or it is not characterized by distortion due to a bad quality of the registration, consonants sounds should have an energy value very near to zero. Of course an audio clip completely clean from noise and music is very rare to find: apart from the possibility of a bad registration quality, very often we deal with songs, that are naturally a mix of voice (that of the singer) and music, or films, that most

of the times are characterized by an almost continuous background music (more or less loud) and noises, or TV programs, often full of music parts and, almost ever, noise ones (let's think about a football match or a naturalistic documentary). This is also the case of the forty-seconds clip whose energy function is depicted in figure 4-2. For example, around the group 200 there is a quite big period of silence surrounding the local pitch determined by the vowel sound. In reality, this period is also characterized by background noise and music at a low volume, a fact that makes almost constantly energy amplitude value stay above the level of 100.

4.4.1.2 Zero Crossing Rate

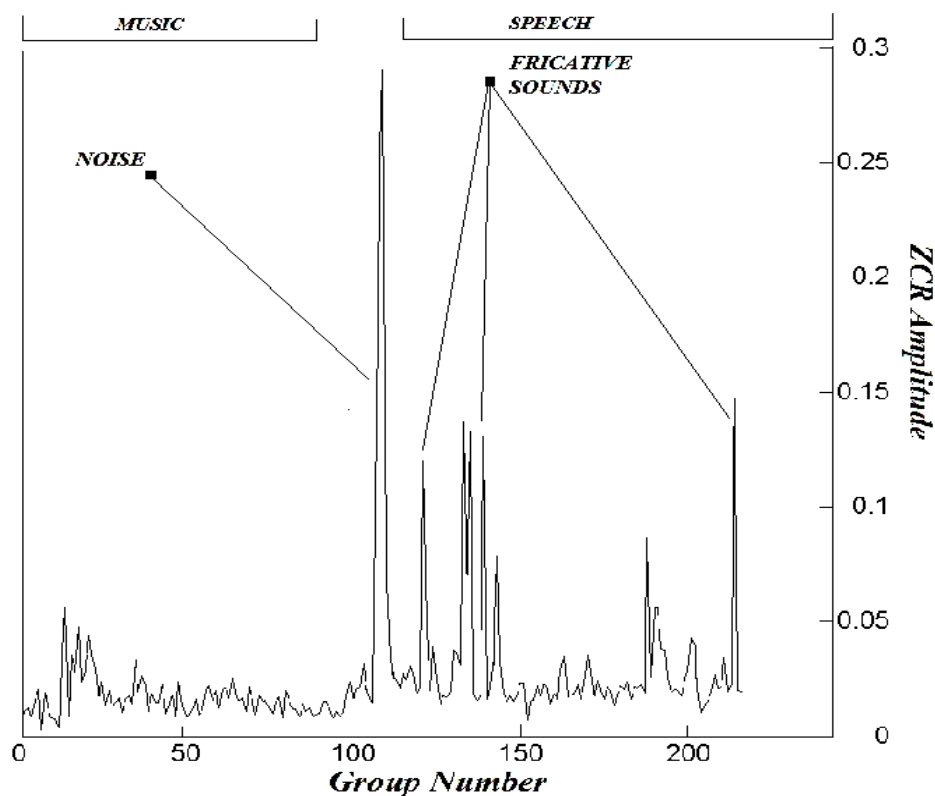


Figure 4-3: ZCR graph: an example

Figure 4-3 is a graphical representation of Zero Crossing Rate behavior. At a glance, it is possible to see how generally speech has some high peaks that music doesn't have. For example, here in the music part ZCR never comes through the limit of 0.1 and rarely reaches 0.05, limit that speech passes every time there is a *fricative sound* (like

\f), \s\ or \z\). This is justified by the fact that ZCR is a temporal feature, but acts like a “frequency measurer” and here lies most of its usefulness. For this reason, every time audio signal is strongly limited at low frequencies ZCR is also low, while it boosts up when signal is characterized by a consistent amount of high frequencies. The former situation happens with vocal sounds and some types of music like that produced by a drum, while the latter is the case of fricative sounds in speech and, most but not all the times, of noise.

As a consequence of this, in this situation noise has the highest peak (reaching almost 0.3) and this can be used positively for discriminating it from the other two audio types during the classification process. Due to its peaks, speech variance should be generally much higher than music one. It's important to notice even here that in absence of fricative sounds (the situation of groups around 175) speech is almost indistinguishable from music. This fact can constitute a difficulty when trying to discriminate this kind of speech fragments from music ones during the classification process.

4.4.1.3 Energy distribution across frequency bands

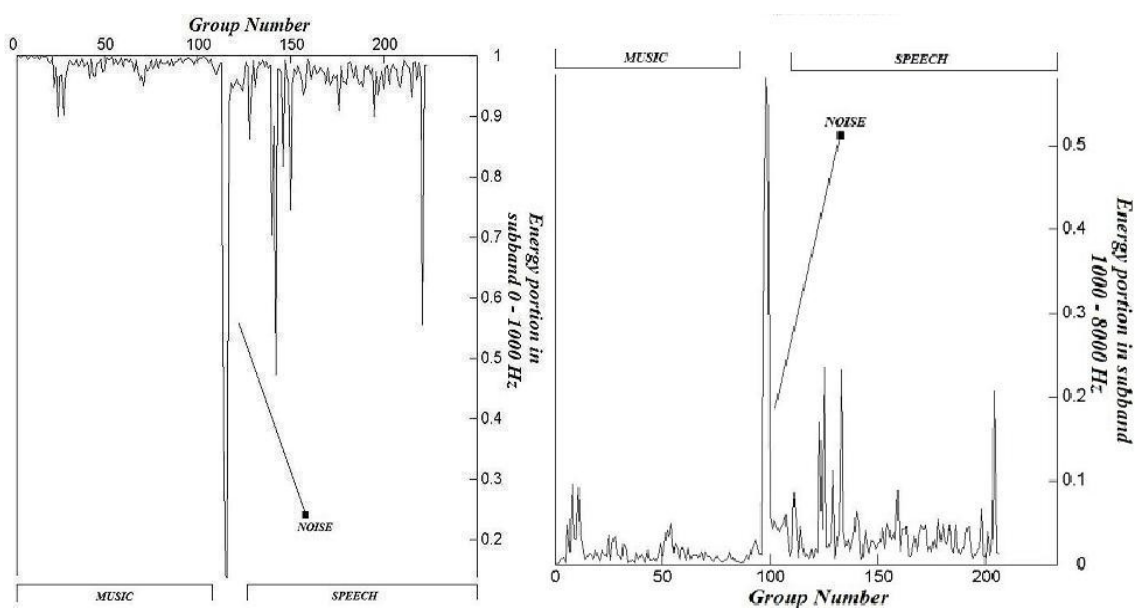


Figure 4-4 (left): Sub band 1 (0-1000 Hz) energy graph: an example

Figure 4-4 (right): Sub band 2 (1000-8000 Hz) energy graph: an example

Figure 4-4 shows energy frequency distribution related to sub bands 1 (from 0 to 1000 Hz) and 2 (from 1000 to 8000 Hz). The first obvious consideration to do is that, as expected, both musical and speech parts concentrate much of their energy in the sub band 1, with values almost ever above 0,9. But there is also a significant difference between these two audio types; in correspondence of pauses or fricative sounds the spectral content of speech parts becomes very similar to that of noise, with an higher percentage of energy lying at higher frequencies. So, variance for sub band 1 distribution related to speech parts is expected to be higher than that of music ones.

Another consideration is that sub band 2 spectral content doesn't give a lot of useful information for audio classification because it doesn't add anything to the information given analyzing sub band 1 and 3 only. For this reason, *the spectral energy amount of a generic audio clip related to sub band 2 hasn't been extracted neither for the off-line classifier nor for the real-time one.*

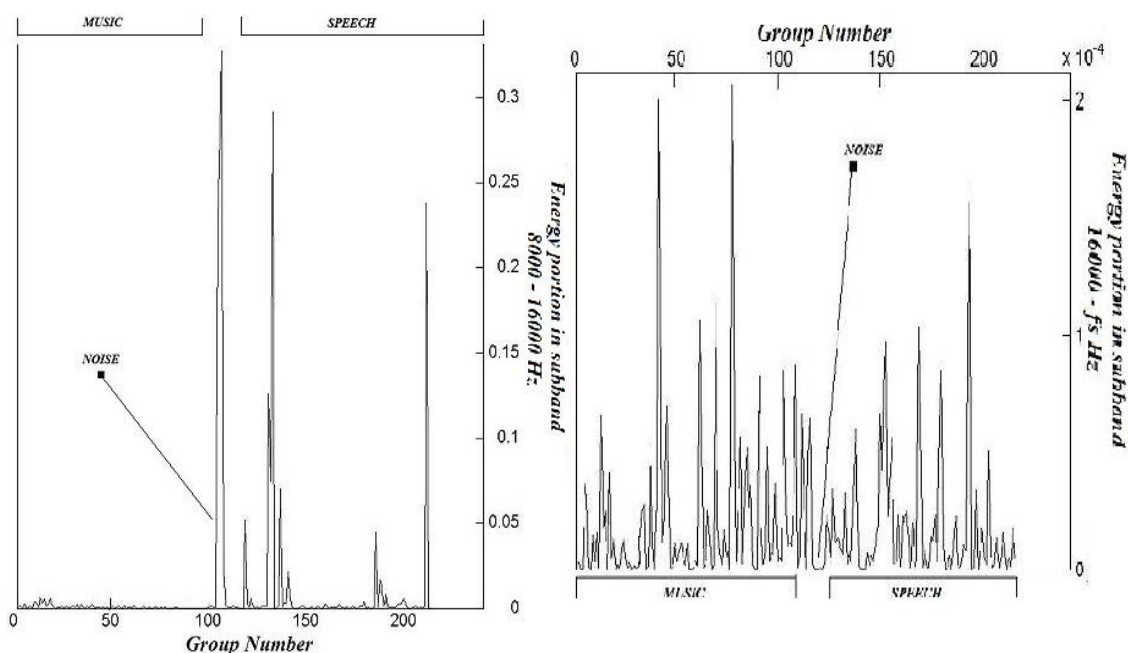


Figure 4-5 (left): Sub band 3 (8000-16000 Hz) energy graph: an example

Figure 4-5 (right): Sub band 4 (16000- f_s Hz) energy graph: an example

The contribution to the overall energy given by sub band 3 (from 8000 to 16000 Hz) is in general lower than that of the two sub bands previously analyzed, as it is possible to see looking at the left graph of figure 4-5. The study of this sub band substantially

confirms considerations made for sub band 1 about how spectral energy distribution have a strong variation in correspondence of: 1) fricative sounds or pauses during a speech part; 2) noise. So, using sub band 3 energy content represents a good way to discriminate between speech and noise on one side, and music on another. It's worth mentioning, however, that certain types of musical sounds (such as cymbals) are characterized by an significant contribution of frequencies between 8000 and 16000 Hz. Also in this case, they are often quite easily recognizable from the other audio types because of their lower variance. This aspect will also be taken into account during the classification process.

Sub band 4 (from 16000 Hz to the half of the sampling frequency) contains a quantity of spectral energy that often can be negligible. This is the case also of the audio clip whose sub band 4 energy distribution is represented in figure 4-5. Furthermore, it does not give a precise idea of differences existing between music and speech. However, there are some circumstances in movies or songs (frequently at the very beginning of a movie) in which some high frequency noises are used like “beep” or similar. In this cases, sub band 4 energy boosts up reaching value higher than 0.1. This situation happens so rarely that *has not been considered a sufficient condition to use sub band 4 energy distribution in the real-time classifier*, also because the same property can be also noticed watching the graph related to Spectral Centroid or Roll-Off Point.

4.4.1.4 Root Mean Square

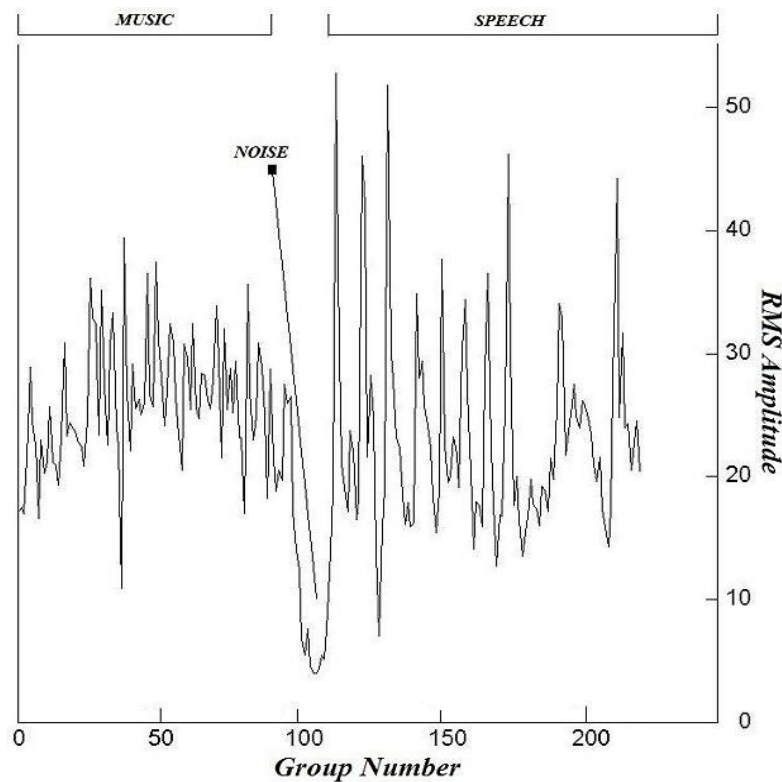


Figure 4-6: RMS graph: an example

Figure 4-6 represents Root Mean Square behavior along the 40-seconds audio clip.

From the graph, it is possible to make the following observations:

- noise is very often characterized by a very low RMS, but in particular situations (such as a gun-shot or a slammed door) it can have a RMS much higher than the other two audio types;
- music has a very regular RMS. Like for other features, its variance is smaller than that of speech parts;
- speech has some peaks in correspondence of vowels (higher energy content). Consonants and pauses are characterized by a much lower RMS.

Apart from these aspects, it is important to say that RMS is (with energy) the feature that is affected more by changes in the volume level. So, it can be dangerous to use for classification purposes properties like local maximum, local minimum or *delta* (distance) existing between them. For this reason it is probably better to focus the attention on normalized measures related to *RMS*. That is why this feature *has not been*

employed in the real-time classifier.

4.4.1.5 Spectral Centroid

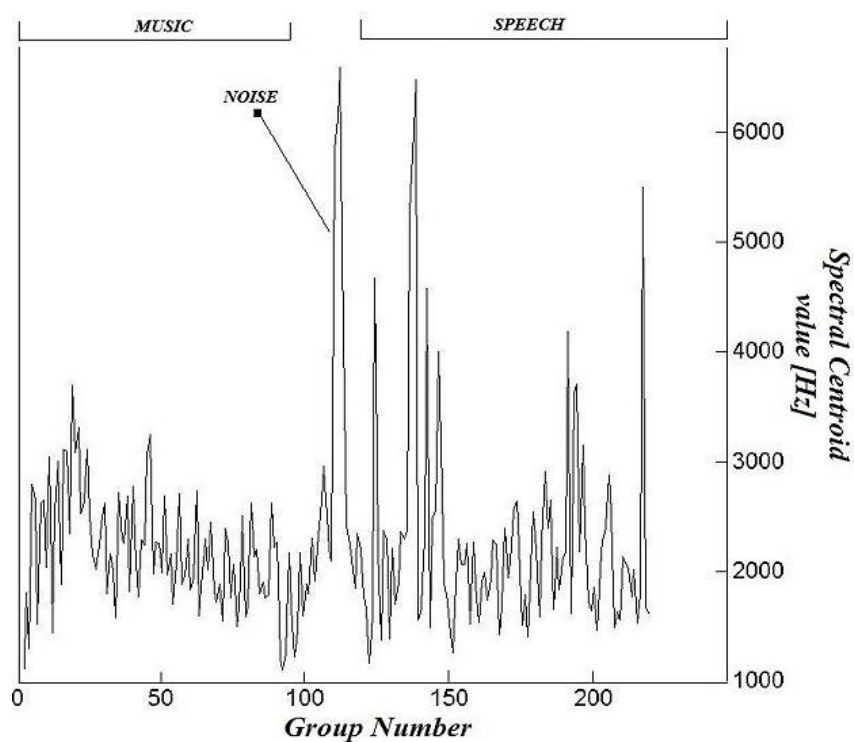


Figure 4-7: Spectral Centroid graph: an example

Figure 4-7 provides a graphical representation of the trend of Spectral Centroid along the analyzed 40-seconds audio clip. As it is possible to see from the figure, there are high Centroid values (often higher than 4000 Hz) in correspondence of noise fragments, pauses and fricative sounds in speech fragments. This can be easily explained with the fact that this kind of sounds have principally a noise component, whose frequency spectra is spread over almost all the frequency range, this resulting in an higher percentage of energy concentrated at higher frequencies than the other audio types. On the other hand, music lacks almost completely of very high peaks, and the distance between the existing small peaks in music fragments is quite regular. Speech, being less harmonic and having a more irregular rhythm than music, doesn't follow the same trend. For all this reasons, variance related to speech fragments is usually a lot higher than that

related to music or noise ones. This feature will be used in the subsequent classification step.

Another thing that cannot be seen directly from figure 4-10, but that often happens in correspondence of high frequency noise, is the fact that Spectral Centroid can reach a local maximum that can be higher than 7000 or even 8000 Hz. In this case the correspondent frame has an high probability to be a noise one. With lower maximum values it is difficult to establish if the maximum corresponds to a noise fragment or to a pause or a fricative sound belonging to a speech fragment.

4.4.1.6 Roll-off point

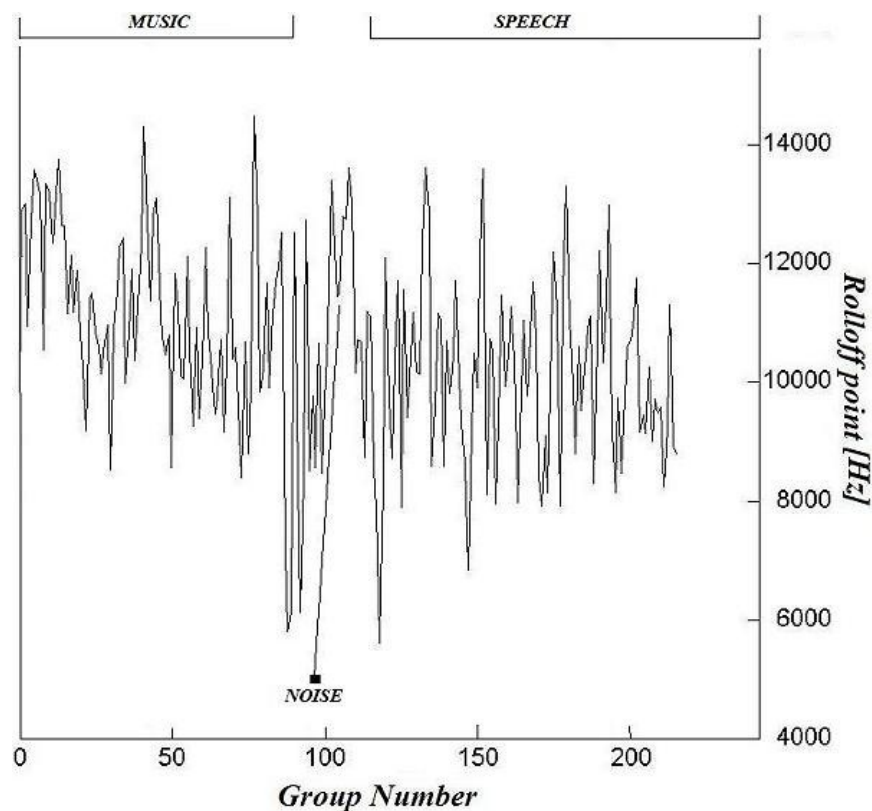


Figure 4-8: Roll-off point graph: an example

Roll-off Point has a behavior that is quite similar to that of the Spectral Centroid, as can be seen looking at Figure 4-8 above. Speech parts are characterized here by high values in correspondence of fricative sounds or pauses, lower ones in correspondence of

vowels and other consonant sounds. As for Spectral Centroid, noise is often characterized by high Roll-off values. The main difference with Centroid lies in music parts, whose variance in this case doesn't appear to be a lot lower than speech ones. Moreover, local maximum and especially mean for each group are often even higher than those related to noise or speech fragments. However, this situation can be somewhat different when dealing with some particular kinds of musical genres (for example, if drum or bass are the predominating instruments Roll-off Point would fall down at lower frequencies).

All the above considerations can be used in the following way during the classification step: all fragments having a quite high mean can be treated as musical parts; the same can be done when dealing with fragments whose local maximum is higher than a predefined threshold. This method could not work for all kinds of music genres, but could be very helpful for the discrimination between music and the other audio types in a lot of cases.

4.5 Classification algorithm

At this point low-level features have been extracted and eventually (only in the off-line classifier) plotted. It is time to pass the control of the processing flow to the classification algorithm. The algorithm developed in the present work is based on two main concepts that are fundamental both for the off-line classifier and the real-time one; these are:

- 1) *heuristic-based conditions*;
- 2) *points vector*.

Before describing in detail the functioning of the classification algorithm underlining the differences existing between the off-line and the real-time case (paragraph 4.5.3), these two concepts will be analyzed and described in the following two sub paragraphs (4.5.1 and 4.5.2).

4.5.1 Heuristic-based conditions

The idea of using conditions based on the shape assumed by the curves related to each low-level feature extracted from the audio clip is not original. As explained in paragraph 2.3, especially in the field of audio classification with real-time constraints, more than one attempt was made following this direction (for example see [39] and [40]).

The originality of this project in this sense lies in the fact that here more than two or three low-level features are used; on the contrary, for example in the work of Zhang ([39]) only ZCR, energy and fundamental frequency were used to obtain a reliable speech/music discriminator. The higher number of features is not a problem at all for the off-line classifier, where there are not strict limits of time in which the classifier has to return its results; could be a problem in the real-time one, but, as it will be better explained in the next paragraph, this is avoided using a classification scheme very “soft” in terms of computational time required.

There are some heuristic conditions used in the present work; these can all be classified into two major groups:

- *no normalized heuristic features*: these features are used both in the off-line and in the real-time classifier. Their values are used directly in the classification conditions;
- *normalized heuristic features*: these features (as already mentioned in paragraph 4.3.2) are used only in the off-line classifier due to the time constraints of the real-time one. In this work, the heuristic features, taken directly from a low-level audio feature, are divided for the mean of the whole correspondent feature vector. Results of this division are used in the classification conditions.

The statistical features listed below are all related to a group of analysis windows (or frame) whose length is previously set, both in the off-line and in the real-time classifier, by the user:

1. *arithmetic mean*: this feature can be useful all the times a low-level feature

assumes values that are particularly high or low in correspondence of a determined audio type. In Matlab it can be easily computed using the standard function *mean* which calculates the arithmetic mean of its unique argument;

2. *local maximum*: this feature can be useful if a low-level feature assumes values that are particularly high in some points. In Matlab it can be computed using the standard function *max* that calculates the maximum of its unique argument (that in our case is a series of values that the low-level feature in exam assumes during a certain frame). Local maximum results particularly useful if it is combined with the arithmetic mean in the same classification condition (example: *if max(frame) > 2 * mean(frame) ...*);

3. *local minimum*: this feature can be useful if a low-level feature assumes values that are particularly low in some points. In Matlab it can be computed using the standard function *min* that calculates the minimum of its unique argument;

4. *variance*: this feature is very useful when dealing with low-level features characterized by a very high variation rate for certain audio types (usually speech has an higher variance than music or noise). In Matlab it can be computed using the standard function *var*, which calculates the variance of its unique argument;

5. *normalized arithmetic mean*: for the utility of this heuristic feature the same considerations are valid made for the arithmetic mean. It can be calculated applying Matlab function *mean* on the previously normalized values referred to a single frame. Its use is preferred when dealing with low-level features whose amplitude can vary considerably depending on volume level (for example, energy or Root Mean Square);

6. *normalized variance*: it can be calculated applying Matlab function *var* on the previously normalized values referred to a single frame. This heuristic feature is very useful when dealing with low-level features whose variance is considerably different between an audio type and another and whose amplitude is affected by

changes in volume level.

For Matlab implementation of the heuristic features used in the project, the correspondences between each of them and the low-level features on which they have been applied, and the mathematical equation that describes them, see Table 4-1 below.

Heuristic feature name	Mathematical equation	MATLAB implementation	Low-level features in which the heuristic is employed
<u>ARITHMETIC MEAN</u>	$\bar{X} = \frac{\sum_{i=1}^n X_i}{n}$ <p>n is the length of the frame (in windows), i is the window index</p>	<i>mean</i> (<i>frame_vector</i>)	<ul style="list-style-type: none"> •Energy •Zero Crossing Rate •B1 sub band energy distribution •B3 sub band energy distribution •Roll-off Point •Root Mean Square
<u>LOCAL MAXIMUM</u>	$M = \max\{X_1, X_2, \dots, X_n\}$ <p>n is the length of the frame (in windows)</p>	<i>max</i> (<i>frame_vector</i>)	<ul style="list-style-type: none"> •Energy •Zero Crossing Rate •B3 sub band energy distribution •B4 sub band energy distribution •Roll-off Point •Spectral Centroid
<u>LOCAL MINIMUM</u>	$m = \min\{X_1, X_2, \dots, X_n\}$ <p>n is the length of the frame (in windows)</p>	<i>min</i> (<i>frame_vector</i>)	<ul style="list-style-type: none"> •Energy •B3 sub band energy distribution •Roll-off Point
<u>VARIANCE</u>	$S^2 = \frac{\sum_{i=1}^n (X_i - \bar{X})^2}{n}$ <p>n is the length of the frame (in windows), i is the window index, \bar{X} is the arithmetic mean related to the current frame only</p>	<i>var</i> (<i>frame_vector</i>)	<ul style="list-style-type: none"> •Zero Crossing Rate •Spectral Centroid
<u>NORMALIZED ARITHMETIC MEAN</u>	$\bar{X}_{norm} = \frac{\sum_{i=1}^n X_i}{n * X_{clip}}$ <p>n is the length of the frame (in windows), i is the window index, \bar{X}_{clip} is the</p>	<i>mean</i> (<i>normalized_frame_vector</i>) <i>normalized_frame_vector</i> =	<ul style="list-style-type: none"> •Energy

	arithmetic mean calculated considering all the audio clip	frame_vector / mean (frame_vector)	
<u>NORMALIZED VARIANCE</u>	$S_{norm}^2 = \frac{\sum_{i=1}^n X_i}{n * \overline{X}_{clip}}$ <p>n is the length of the frame (in windows), i is the window index, \overline{X}_{clip} is the arithmetic mean calculated considering all the audio clip</p>	var (normalized_frame_vector) normalized_frame_vector = frame_vector / mean (frame_vector)	<ul style="list-style-type: none"> •Energy •Root Mean Square

Table 4-1: Heuristic features used for classification purposes and their practical Matlab implementation

4.5.2 A point vector for each audio type

Probably the originality of the whole project lies mostly in the idea of employing a point vector for the classification choice. Except silences (whose identification algorithm will be explained in Paragraph 4.5.3.1) all the other audio types – music, speech and noise – are discriminated each other using this system.

But in what does consist precisely an audio classification algorithm based on point vectors? The simpler and faster answer to this question could be the following: *an audio classification algorithm based on point vectors is an algorithm that operates a discrimination between different audio types choosing for each frame the one that receives more points as the audio type for that frame; these points are assigned using conditions based on some feature of the audio file.* This feature could be a mathematical instrument or a statistical model: in this project heuristic features directly extracted from low-level features of the audio file are used.

A first consideration about this method can be about its *computational simplicity*, unlike other mathematical or statistical instruments that often have been used in past researches for audio classification purposes (let's think for example at the Gaussian Mixture Model or at the k-Nearest Neighbor algorithm). It has been this simplicity that allowed the utilization of low-level frequency-domain features also in the real-time

classifier in the present work. This is a very important point, independently from the obtained results that can be improved adding or modifying heuristic conditions applied over the low-level features (however, the actual results obtained by this method will be discussed in Chapter 5).

The second thing one can say is about the practical realization of this method. Discussion about this point will not be too long, because it is sufficient to say that a vector must be created for each audio type on which this method should be applied (or alternatively a matrix whose number of rows should be equal to the number of audio types one wants to discriminate). The length of each of these vectors must be equal to the number of frames in which the audio clip is divided. For example, if the audio clip is one minute long, each frame lasts 2 seconds and we want to discriminate between four audio types using this method, we should have 4 points vectors of 30 elements each.

A last consideration regards eventual cases of equal points obtained by two or more features for the same audio frame. The better is designed the algorithm the less these cases should be, but it is almost impossible to achieve results in which it never happens to have two audio types being characterized by the same number of points for the same frame. So, it is necessary to set a *priority level* for each audio type to handle these situations. In the present project, this problem has been solved assigning the highest priority to the audio type that, potentially, could receive the lowest points score if all conditions concerning it would be satisfied. For a better understanding of this concept, see Table 4-4 below (the conditions and the vector names used in the table are only a hypothetical example: they do not correspond to those used in the present project).

	<i>Points assigned to SPEECH [k]</i>	<i>Points assigned to NOISE [k]</i>	<i>Points assigned to MUSIC [k]</i>
Condition 1 - If <i>maximum</i> of k-th frame Energy is higher than threshold 1	2	-2	-1
Condition 2 - If <i>variance</i> of k-th frame Spectral Centroid is lower than threshold 2	1	-2	2
Condition 2 - If <i>mean</i> of k-th frame Root Mean Square is lower than threshold 3	0	3	0
<i>Potential points that can be assigned to each</i>	3	-1	1

<i>audio type if all conditions are satisfied</i>			
ASSIGNED PRIORITY	<u>3</u>	<u>1</u>	<u>2</u>

Table 4-2: Example of priority assignment scheme for the various audio types

4.5.3 Practical implementation

4.5.3.1 Off-line classifier

All classification tasks in the off-line case are accomplished by Matlab function *OffLineClassifier* (file “OffLineClassifier.m”). This function takes in input the number of windows that have to be grouped together in one single frame; this number is expressed by a variable whose name is *GROUPLN* and whose value is set by the user while calling the function *OffLineClassifier* in Matlab command prompt. *OffLineClassifier* returns in output some graphs (in format .jpg) illustrating the classification results. For more information about the plotting of these results see Paragraph 4.6.1; here only the classification part related to this function will be described.

After the declaration of some global variables the classification process starts. Before going on with a more detailed description of this process, it's worth saying that it can be divided into two main parts (as anticipated in paragraph 3.4):

1. the *first part* is quite short and simple, and *consists essentially in recognition of eventual silence parts* in the audio clip;
2. *in the second part point vectors are used to discriminate between noise, music and speech parts* in the audio clip.

Both parts function using heuristic conditions applied over low-level features, that at this point are stored in a series of DAT files (one for each extracted feature). To read information stored in the desired DAT file, the same Matlab constructs are used both in the first and in the second part of the classification process. In practice, a *for* cycle is used to move through all the DAT files of the working directory, and each time the desired feature file is opened to read its content by means of standard Matlab functions

strncmpi (that recognizes the DAT file) and *load* (that loads in an array all data contained in the file whose name is indicated by its unique argument). An example of this implementation is shown in table B-3 (Appendix B); there is also explained how function *strncmpi* works.

After loading information stored in low-level feature desired file, the next step is grouping the extracted information in frames whose length is determined by the value assumed by variable *GROUPLLEN*. This operation is performed in a *for* cycle inserted in a *while* cycle in which desired heuristic features for each frame are computed. On these heuristics some conditions will be subsequently applied. The implementation of the grouping and heuristic features calculation just described can be found in Table B-4 (Appendix B).

Having extracted from each frame the desired heuristic features, everything is ready to apply some conditions on these results and to begin the proper classification process. Here there is the first difference between the silence identification algorithm and the one discriminating between the other three audio types.

In fact, in the former a system is used that classifies possible silence fragments frame-by-frame, and before the end of the process a frame comparison is performed that selects only frames that satisfied all conditions applied on them as silence.

The *second algorithm is based on point vectors and discriminates only between frames that have not been classified as silence by the first one*. The reason of this choice is the fact that there is a very small probability of misclassification errors between silences and all the other audio types; this because silence has some heuristic features that assume values very distinguishable from the others. The algorithm discriminating between music, noise and speech audio fragments cannot achieve so good results, and so if silence detection algorithm classifies some fragments as silence, there is no reason at all to modify this result with those provided by the music-speech-noise discrimination one. Apart from this aspect, the most important differences between the two algorithms lie in the fact that the second one works using point vectors and all-in-once (in the sense that heuristic features are applied – out of the *while* cycle that can be seen in Table B-4 –

on a vector storing all the heuristics related to a particular low-level feature for all the frames of the audio clip), while the first worked using a comparative scheme and frame-by-frame. The best way to explain this concept is probably a table once again: Matlab implementation related to both the above mentioned algorithms can be found in Table B-5 (Appendix B).

The silence identification algorithm illustrated in Table 4.7 is the very model of the one used in the project. It simply applies on each features frame two heuristic conditions:

- the arithmetic mean related to an energy frame has to be smaller than a threshold fixed at value 5. As it is possible to see also from Figure 4-2, this is a very small value for energy function, usually noise, speech or music fragments in an audio clip being characterized by an arithmetic mean higher than 100. Only noise sometimes can create misclassification problems, being its energy very small too;
- the arithmetic mean related to a RMS frame has to be smaller than a threshold fixed at value 0,2. This condition is a lot more conservative than the first one; in fact, usually all the audio types except silence have an arithmetic mean related to RMS higher than 5.

If both conditions are satisfied for a determined frame, the value of the position corresponding to the number of that frame is set to 1 in vector *CLASSIFICATION_GLOBAL*, whose total length is equal to the number of frames in the audio clip. At the end of the classification process, each position of this vector storing value “1” will be considered as a silence frame.

The music-noise-speech discrimination algorithm has a lot more conditions with respect to the previous one. For this reason they are not reported in Table B-5; a list of them, with the reasons that led to their utilization, can be found in Table 4-3 below. Table B-5 shows only a model of the algorithm, using only one low-level feature (energy in the example) and one statistical feature (local minimum in the example). Once all desired heuristic conditions have been applied on all low-level features, each of the three point vectors (named *points_SPEECH*, *points_MUSIC* and *points_NOISE*) will

have a positive or negative, high or low value for each vector position (corresponding to a frame) depending on how many heuristic conditions have been satisfied or not.

<u>Heuristic Condition</u>	<u>Points assigned for each audio type</u>	<u>Motivations</u>
<p><u>If local minimum</u> related to <i>sub band 3</i> ([8000 – 16000] Hz) energy distribution is <u>higher</u> than value...</p> <p>1) 0.003 2) 0.002 3) 0.001</p>	<p>1) <i>NOISE</i> >>> + 3 <i>MUSIC</i> >>> + 3 <i>SPEECH</i> >>> - 3</p> <p>2) <i>NOISE</i> >>> + 2 <i>MUSIC</i> >>> + 2 <i>SPEECH</i> >>> - 2</p> <p>3) <i>NOISE</i> >>> + 1 <i>MUSIC</i> >>> + 1 <i>SPEECH</i> >>> - 1</p>	<p>An interesting property of sub band 3 energy distribution is the fact that it can have quite high mean values in correspondence of certain types of noise and some kinds of musical instruments. Anyway, mean is not sufficient to describe this phenomena, because fricative sounds and pauses in speech give rise to very high values for this feature, but these are very near to zero in correspondence of vowels. This fact has been used to differentiate between speech on one hand, music and noise on the other, using a condition based on local minimum for each frame.</p>
<p><u>If local maximum</u> related to <i>sub band 4</i> ([16000 – $f_s/2$] Hz) energy distribution is <u>higher</u> than value 0.05</p>	<p><i>NOISE</i> >>> + 3 <i>MUSIC</i> >>> - 3 <i>SPEECH</i> >>> - 3</p>	<p>Sub band 4 energy distribution value is often very small (in the order of 10^{-4}). Only rarely, with high frequency noises, its value passes the limit of 0.05.</p>
<p><u>If local maximum</u> related to <i>Spectral Centroid</i> is <u>higher</u> than value...</p> <p>1) 8000 Hz 2) 7000 Hz</p>	<p>1) <i>NOISE</i> >>> + 2 <i>MUSIC</i> >>> - 2 <i>SPEECH</i> >>> - 2</p> <p>2) <i>NOISE</i> >>> + 1 <i>MUSIC</i> >>> - 1 <i>SPEECH</i> >>> - 1</p>	<p>Spectral Centroid reaches usually high values in correspondence of music, and very high ones in correspondence of fricative sounds and pauses in speech; anyway peaks of 7000/8000 Hz can be reached only quite rarely by some kind of noise sounds.</p>
<p><u>If variance</u> related to <i>Spectral Centroid</i> is <u>higher</u> than value...</p> <p>1) 1000000 2) 500000</p>	<p>1) <i>NOISE</i> >>> - 1 <i>MUSIC</i> >>> - 2 <i>SPEECH</i> >>> + 2</p> <p>2) <i>NOISE</i> >> - 0.5 <i>MUSIC</i> >>> - 1 <i>SPEECH</i> >>> + 1</p>	<p>Condition 1 and, more often, condition 2 are satisfied by speech fragments, and almost never by fragments of the other two audio types. This because fricative sounds and pauses are characterized by high Spectral Centroid levels for their noise components, while vowels have a value a lot lower.</p>
<p><u>If variance</u> related to <i>Spectral Centroid</i> is <u>lower</u> than value 50000</p>	<p><i>NOISE</i> >>> + 1 <i>MUSIC</i> >>> + 1 <i>SPEECH</i> >>> - 1</p>	<p>Variance in speech fragments has an higher value with respect to noise and music fragments, whose Spectral Centroid values (as it can be seen also</p>

		in Figure 4-7) are often more uniformly distributed, generally having an high value for noise and medium ones for music parts.
<p>If <i>local minimum</i> related to <i>Energy</i> is <u>higher</u> than value....</p> <p>1) 200</p> <p>2) 100</p>	<p>1) <i>NOISE</i> >>> - 2 <i>MUSIC</i> >>> + 2 <i>SPEECH</i> >>> - 2</p> <p>2) <i>NOISE</i> >>> + 0.5 <i>MUSIC</i> >>> + 1 <i>SPEECH</i> >>> - 1</p>	<p>Almost all kinds of music are characterized by the fact that their energy does never have very low energy values. This constitutes an opportunity to differentiate this audio type from speech, whose consonant sounds and pauses have a noise-like behavior being characterized by a very low energy level.</p>
<p>If <i>normalized arithmetic mean</i> related to <i>Energy</i> is <u>higher</u> than value....</p> <p>1) 1.5</p> <p>2) 1</p> <p>.... <u>and</u> <i>normalized variance</i> related to <i>Energy</i> is <u>lower</u> than value....</p> <p>1) 0.1</p> <p>2) 0.25</p>	<p>1) <i>NOISE</i> >>> - 2 <i>MUSIC</i> >>> + 2 <i>SPEECH</i> >>> - 2</p> <p>2) <i>NOISE</i> >>> - 1 <i>MUSIC</i> >>> + 1 <i>SPEECH</i> >>> - 1</p>	<p>As it can be seen also in figure 4-2, music sounds are characterized by often quite high energy values with variations smaller than speech ones. This consideration is exploited practically by these two heuristic condition, in which are labeled as “music” those audio parts whose arithmetic mean is higher than a predetermined threshold and whose variance is lower than another one.</p>
<p>If <i>normalized arithmetic mean</i> related to <i>Energy</i> is <u>lower</u> than value....</p> <p>1) 0.5</p> <p>2) 1</p> <p>.... <u>and</u> <i>normalized variance</i> related to <i>Energy</i> is <u>higher</u> than value....</p> <p>1) 0.75</p> <p>2) 0.4</p>	<p>1) <i>NOISE</i> >>> - 2 <i>MUSIC</i> >>> - 2 <i>SPEECH</i> >>> + 2</p> <p>2) <i>NOISE</i> >>> - 1 <i>MUSIC</i> >>> - 1 <i>SPEECH</i> >>> + 1</p>	<p>These two conditions are the opposite of the ones above: in practice, audio parts characterized by an high energy variance and, at the same time, a not so high arithmetic mean, are considered to have a good probability to be speech parts. It is important to consider that both these and the conditions above assign a negative number of points to noise: this because both have one condition (an energy higher than a threshold in one case, a variance higher than another threshold in the other) that is almost impossible to be satisfied by noise parts.</p>
<p>If <i>normalized variance</i> related to <i>Energy</i> is <u>lower</u> than value...</p> <p>1) 0.005</p> <p>2) 0.01</p>	<p>1) <i>NOISE</i> >>> + 2 <i>MUSIC</i> >>> - 2 <i>SPEECH</i> >>> - 2</p> <p>2) <i>NOISE</i> >>> + 1 <i>MUSIC</i> >>> - 1 <i>SPEECH</i> >>> - 1</p>	<p>Values of energy variance particularly low for a frame are often sign of a noise part in the audio clip.</p>

<p><u>If local maximum</u> related to <u>Energy</u> is <u>higher</u> than....</p> <p>1) 5 times the <i>arithmetic mean</i></p> <p>2) 4 times the <i>arithmetic mean</i></p> <p>3) 3 times the <i>arithmetic mean</i></p>	<p>1) <i>MUSIC >>> - 3</i> <i>SPEECH >>> + 3</i></p> <p>2) <i>MUSIC >>> - 2</i> <i>SPEECH >>> + 2</i></p> <p>3) <i>MUSIC >>> - 1</i> <i>SPEECH >>> + 1</i></p>	<p>These three conditions exploit the fact that energy curve related to speech parts is characterized by peaks of a small width (corresponding to vowel sounds) and low values in correspondence of pauses and consonant sounds. This led to the consideration that often in speech parts the local maximum for a frame could be much higher than the arithmetic mean for the same frame. This condition doesn't subtract points to noise because it can sometimes satisfy this condition; this because generally energy values related to noise parts are so low that even a small improvement can lead to a local maximum 5 or 10 times higher than the arithmetic mean.</p>
<p><u>If normalized variance</u> related to <u>RMS</u> is <u>lower</u> than value....</p> <p>1) 0.001</p> <p>2) 0.05</p>	<p>1) <i>NOISE >>> + 2</i> <i>MUSIC >>> + 1</i> <i>SPEECH >>> - 2</i></p> <p>2) <i>NOISE >>> + 1</i> <i>MUSIC >>> + 1</i> <i>SPEECH >>> - 1</i></p>	<p>Often RMS variance (and normalized variance) values are particularly low for music parts and, especially, noise ones.</p>
<p><u>If normalized variance</u> related to <u>RMS</u> is <u>higher</u> than value 0.25</p>	<p><i>NOISE >>> - 1</i> <i>MUSIC >>> - 1</i> <i>SPEECH >>> + 1</i></p>	<p>On the other hand, as can be seen also in Figure 4-4, speech is characterized by a lot higher variance with respect to the other two audio types.</p>
<p><u>If local maximum</u> related to <u>Roll-off Point</u> is <u>higher</u> than value 16000 Hz</p>	<p><i>NOISE >>> + 3</i> <i>MUSIC >>> - 1.5</i> <i>SPEECH >>> - 1.5</i></p>	<p>This condition is verified very rarely; when this happens, it means that we are dealing with particularly high frequency noises.</p>
<p><u>If local minimum</u> related to <u>Roll-off Point</u> is <u>lower</u> than value 4000 Hz</p>	<p><i>NOISE >>> - 0.75</i> <i>MUSIC >>> - 0.75</i> <i>SPEECH >>> + 1.5</i></p>	<p>Roll-off point falls at very low frequency levels only in correspondence of vowel sounds related to audio parts.</p>
<p><u>If arithmetic mean</u> related to <u>Roll-off Point</u> is <u>higher</u> than value 10000 Hz</p>	<p><i>NOISE >>> + 1</i> <i>MUSIC >>> + 1</i> <i>SPEECH >>> - 0.5</i></p>	<p>On the other hand, Roll-off point is particularly high in correspondence of silences or fricative sounds in speech parts, or in music and noise parts. So arithmetic mean will be higher for noise or music frames then for speech ones.</p>
<p><u>If local maximum</u> related to <u>ZCR</u> is <u>higher</u> than....</p>	<p>1) <i>NOISE >>> - 2</i> <i>MUSIC >>> - 2</i></p>	<p>For ZCR are valid the same considerations made for local</p>

1) 3 times the <i>arithmetic mean</i>	<i>SPEECH >>> + 2</i>	maximum condition related to energy. The only difference is that here negative points can be assigned to noise parts, because ZCR values for noise are not particularly high and so it is difficult to happen this condition to be verified for noise parts.
2) 2 times the <i>arithmetic mean</i>	2) <i>NOISE >>> - 1</i> <i>MUSIC >>> - 1</i> <i>SPEECH >>> + 1</i>	
If <i>variance</i> related to ZCR is <u>lower</u> than value....	1) <i>NOISE >>> + 2</i> <i>MUSIC >>> + 2</i> <i>SPEECH >>> - 2</i>	As it can be seen also from Figure 4-3, ZCR variance is usually higher for speech parts in an audio clip, than for noise or music ones. For this reason the present heuristic condition is applied on each frame, to discriminate speech fragments from music and noise ones.
1) 0.0001	2) <i>NOISE >>> + 1</i> <i>MUSIC >>> + 1</i> <i>SPEECH >>> - 1</i>	
2) 0.0005		

Table 4-3: List of all heuristic conditions used in music-noise-speech discrimination algorithm

The last thing to do at this point, is to compare frame by frame the values obtained by each audio type. Each frame is classified as the audio type that has scored the highest number of points. In case two audio types have obtained an equal number of points for the same frame, a choice is made based on the priority assignment scheme described in Paragraph 4.5.2. In *OfflineClassifier* the highest priority is assigned to music, then speech and finally noise, that has the lowest priority.

The final result of the execution of the two algorithms just described is the vector *CLASSIFICATION_GLOBAL* above mentioned. Each value it can store (an integer ranging from 1 to 4) corresponds to a specific audio type: “1” is silence, “2” noise, “3” corresponds to music and “4” to speech. In Paragraph 4.6.1 will be described the last part of function *OfflineClassifier*, the one that performs the plotting of these results and of the three points vectors.

4.5.3.2 Real-time classifier

As already mentioned in paragraph 4.3.2, classification algorithms in the real-time classifier are located in the same Matlab function that performs low-level features extraction (*RealTimeClassifier* in file “RealTimeClassifier.m”). Classification task is

performed in the outer of the two *for* cycles characterizing this function, as it is possible to see also from Table B-6 (Appendix B).

As done in paragraph 4.3.2 for low-level features extraction, classification scheme used in the real-time classifier will be described analyzing the *differences* existing between this classifier and the off-line one.

The first important difference between the two classifiers deals with their own nature. In fact, real-time classifier has to provide frame-by-frame results; it cannot wait for the end of the low-level feature extraction process for the whole audio clip (thing that happens for the off-line classifier) to begin its work. For this reason, each time low-level features are extracted for a frame, the control of function *RealTimeClassifier* passes from the inner to the outer *for* cycle, that applies both silence detection and music-noise-speech discrimination algorithms on calculated features. At the end of the classification step, control returns to the inner *for* cycle for the extraction of low-level features related to subsequent frame, and this process goes on until the end of the audio clip.

The second difference lies in the number of heuristic conditions applied in the two classifiers. This number is lower for the real-time one. This for two reasons:

1. *real-time classifier doesn't use heuristic features related to RMS and sub band 4 (that goes from 16000 Hz to the half of the sampling frequency) frequency energy distribution.* The reason of this double exclusion, as it has already been explained in paragraph 4.4.1, is different for the two features: RMS is helpful only if its values are normalized, but normalizing is computationally too expensive with respect to the amount of classification information that is possible to obtain extracting this feature; sub band 4 energy distribution gives information that can be obtained also from other low-level features;
2. *all conditions based on normalized heuristic features (used in the off-line classifier) have not been used in the classifier we are currently analyzing;* this for their high computational (and subsequently time) costs.

The result of this difference for silence detection algorithm is that in this classifier it

is composed by only one condition (that related to energy) but more strict than its homologous used in the off-line classifier (see Table 4-10). For music-noise-speech discrimination algorithm there are 10 conditions used in the off-line classifier missing in the real-time one and three new conditions (see Table 4-10 for details) introduced in the latter to try to replace these exclusions without making computational time rise too much.

<u>Heuristic Condition</u>	<u>Algorithm in which condition is used</u>	<u>Motivations</u>
If <i>arithmetic mean</i> related to <i>Energy</i> is <u>lower</u> than value 0.05	Silence detection	In this condition (used also by off-line classifier) threshold has been diminished to compensate the absence of a condition based on RMS.
If <i>variance</i> related to <i>Energy</i> is lower than value 10000 <u>and</u> 1) <i>arithmetic mean</i> is <u>lower</u> than value 10 2) <i>arithmetic mean</i> is <u>higher</u> than value 10	Music-noise-speech discrimination POINTS ASSIGNED 1) <i>NOISE</i> >>> + 1 2) <i>MUSIC</i> >>> + 1	These two conditions are used to make a distinction between noise and music audio parts; they are based on the consideration that both these audio types are often characterized by a small variance related to energy, but that often music has also an higher level of energy (and so an higher arithmetic mean) with respect to noise, that is often characterized by a very low energy.
If <i>variance</i> related to <i>ZCR</i> is lower than value 0.0005	Music-noise-speech discrimination POINTS ASSIGNED <i>NOISE</i> >>> + 0.5 <i>MUSIC</i> >>> + 1 <i>SPEECH</i> >>> - 0.5	This condition on ZCR variance is less strict than the other two already present in the off-line classifier. It is based on the consideration that values between 0.0001 and 0.0005 of this heuristic feature are more frequent for music than for noise: another way to make a distinction between these two audio types.

Table 4-4: Heuristic conditions in RealTimeClassifier that were absent or different in OffLineClassifier

A third difference existing between the classification methods used in the off-line and in the real-time case is the changing of the priority level assigned to music, noise and speech. In real-time classifier, in the case there are two or three audio types being characterized by the same number of points for the same frame, the highest priority is assigned to speech (*CLASSIFICATION_RESULT* set to “4”) and then music (“3”). *CLASSIFICATION_RESULT* is set to “2” (noise) only if this audio type has obtained a

number of points higher than the other two for the current audio frame.

At the end of the classification process for all the frames in the audio clip, function *RealTimeClassifier* terminates his tasks. It returns four variables as a result; these will be used by function *PlotResults* (see paragraph 4.6.2 for details about this function). These variables are:

1. *CLASSIFICATION_RESULT*, as its name suggests, is the vector storing the classification results for each frame of the audio clip;
2. *points* is a matrix storing the points obtained by music, speech and noise for each frame. So, its number of columns is equal to the number of frames in which the audio clip had been previously divided; its three rows store, respectively, points related to music, speech and noise for each frame;
3. *DURATION* is a variable indicating the length in seconds of the audio clip. This had been calculated at the beginning of function *RealTimeClassifier* (before the two nested *for* cycles) dividing the number of samples (L) of the audio clip for the sample rate of the WAV file (variable *SampleRate*);
4. *NumberOfFrames* is obviously the number of the frames in which the audio clip has been divided, used also as a termination condition for the outer *for* cycle.

4.6 Classification results plotting

The last step of the present project consists in the plotting of the obtained classification results. This is not a necessary step for the project in the sense that both off-line and real-time classifiers work well also without the plotting of the results they return. However, it is almost essential to have a graph summarizing this results if you want to analyze how well the classifier work.

In this paragraph it will be only described how the aim of plotting the results returned by the classification algorithm has been achieved in terms of Matlab programming. A

complete discussion over the qualitative aspects related to the graphs (and so to the classification algorithm too) will be provided in chapter 5.

4.6.1 Off-line classifier

Plotting of the classification results (stored in vector *CLASSIFICATION_GLOBAL*) is performed in the off-line classifier by the same function that contains the classification algorithm (*OffLineClassifier*). Before beginning the proper plotting, this function executes two preliminary operations.

The first thing this part of the program does is the *normalization of vector CLASSIFICATION_GLOBAL*: here the action “normalizing” has the meaning of eliminating the eventual discontinuities the vector may have. This concept needs to be analyzed a little more in profundity.

The reasoning that led to the choice of performing this kind of operation is the fact that very often a frame classified as an audio type positioned in the middle of two or more others classified as another audio type (see also Figure 4-11) may be the result of a classification error or, anyway, it can be considered useless for the aim of the classifier to identify a part in a clip being of one particular audio type whose length is shorter than a predetermined limit.

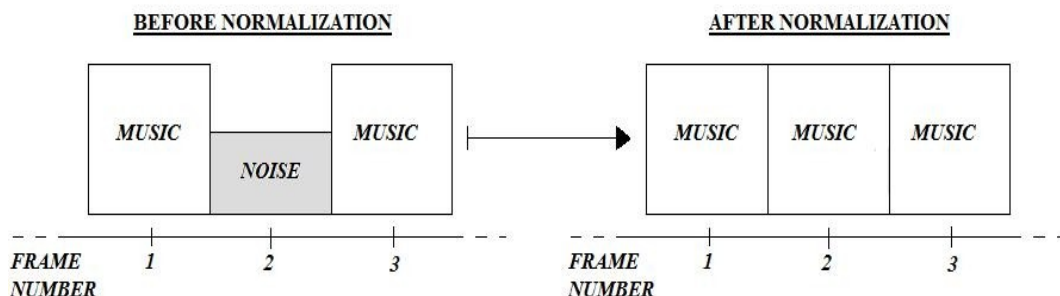
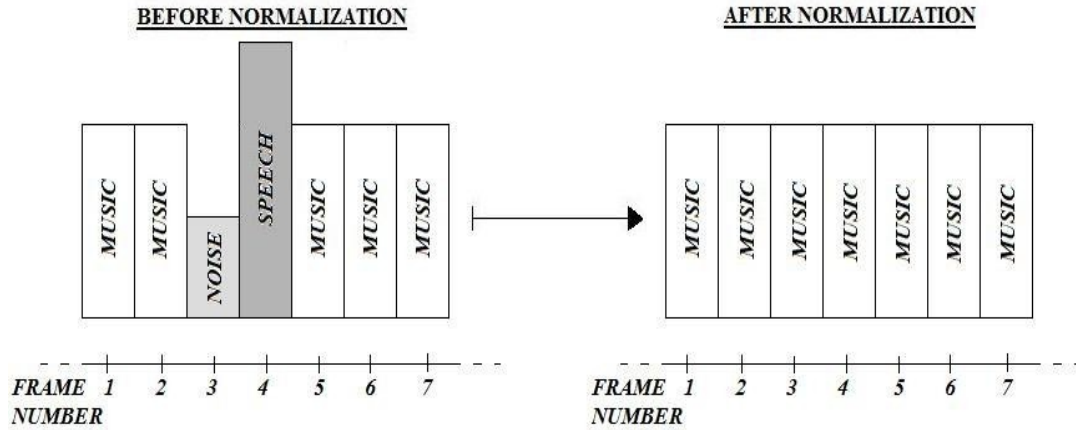


Figure 4-9: An example of normalization applied on a frame labeled as “noise” surrounded by two “music” frames

In reality the same argumentations could be extended also to other and more complex situations, for example considering a big part of or even the whole audio clip, in which

the classifier returns a majority of frames classified as an audio type, and a minority classified as one or more other audio types (see figure 4-9 for an example). A situation



of this kind could be also the result of one or more classification errors, but anyway in an application that searches for all the parts of a clip of a determined audio type, usually is not interesting to find parts whose length is of a few seconds only.

Figure 4-10: An example of normalization applied on a group of frames, in which all frames that constitute a minority are labeled as the more frequent audio type (in this case “music”)

Having the present project the generic aim to classify an audio clip the better (and faster in the case of the real-time classifier) as possible in a simple way, here has been designed and created only a simple normalization algorithm, that only solves simpler kind of classification conflicts (that exemplified in Figure 4-10) and for the rest leaves classification results unmodified. To do so, an auxiliary vector was created (named *aux*) at the beginning equal to *CLASSIFICATION_GLOBAL*; on this vector was subsequently applied a simple normalization algorithm as that shown in Table B-7 (Appendix B).

The second operation the program executes before beginning the creation of the graphs illustrating classification results is a *scaling of the values on x axis*. Vectors *aux* and *CLASSIFICATION_GLOBAL*, that before this operation were represented in a vector whose position index corresponded to the frame number whatever was its length, are rescaled so that the new index is equivalent to one second of the audio clip. This is done to allow a more intuitive representation of the classification results, whose graph in this way can be easily compared to an eventual summary of the audio clip listing all the parts (with the related audio type) composing it. Results of the operation of rescaling are

put in two other vectors named *CLASSIFICATION_FINAL* and *CLASSIFICATION_NORMALIZED*.

To perform this operation it has been necessary to introduce a variable (named *step*) representing the scaling step used to appropriately modify the index of the new vectors created. The proceeding used to perform the scaling of both the normalized classification results vector and the no normalized one is shown in Table B-8 (see Appendix B).

Once normalization and scaling of the classification results have been performed, the function can begin the proper plotting operations. Each audio type is associated in the graph to a unique color; it is also characterized by a different amplitude value with respect to the others. This allows an immediate identification of the various audio types and of their position in the analyzed audio clip. Other details of graphs formatting will not be given here.

The only important thing left to say is that this process returns three JPEG files as the final result (standard Matlab function *saveas* is used to save the obtained graphs on a .jpg file). These files are all put in the same folder containing the classifier and the .dat files related to the previously extracted low-level features:

1. the first .jpg file is a graphical representation of the classification results stored in vector *CLASSIFICATION_NORMALIZED*, storing the classification results on which has been applied the normalization algorithm. Its name is “Off-line classification results”;
2. the second file is the graph of the vector *CLASSIFICATION_FINAL*. This file is named “Off-line NO NORMALIZED results”;
3. the third file is a graph representing the three point vectors related to music, noise and speech audio parts (respectively named *points_MUSIC*, *points_NOISE* and *points_SPEECH*) identified into the analyzed audio clip (see also Paragraph 4.5.3). Also these vectors have been previously scaled on the x axis so that each point of the graph represents a second of the audio clip. Each vector is easily

distinguishable from the others because it has a unique color. The name of this file is “Points noise-speech-music”.

In figures 4-11, 4-12 and 4-13 below there is an example for each of the three output JPEG files created by Matlab function *OffLineClassifier*, all related to the same audio clip (whose length is of 4 minutes).

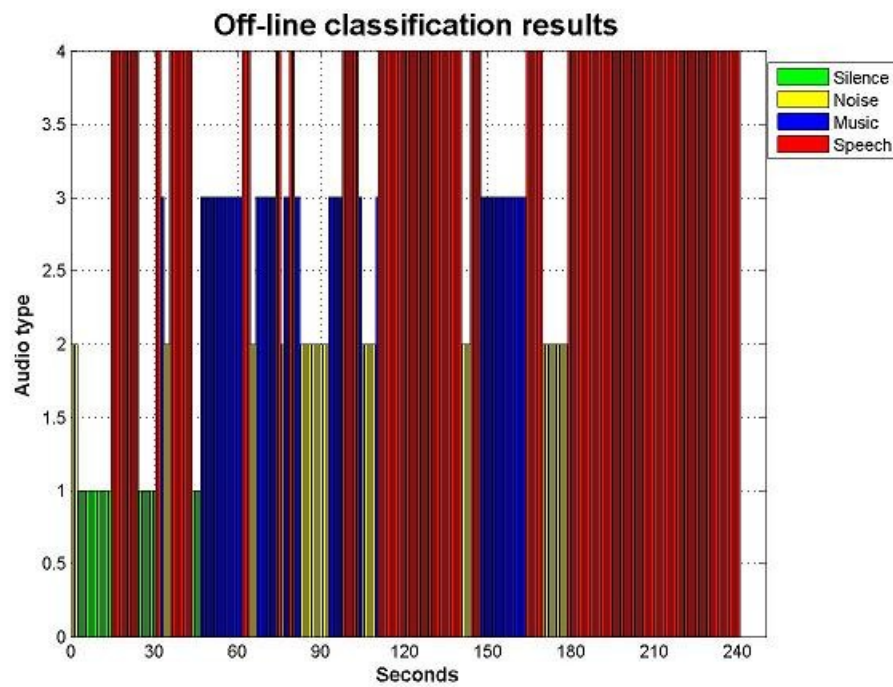


Figure 4-11: First output file of function *OffLineClassifier*: “Off line classification results.jpg”

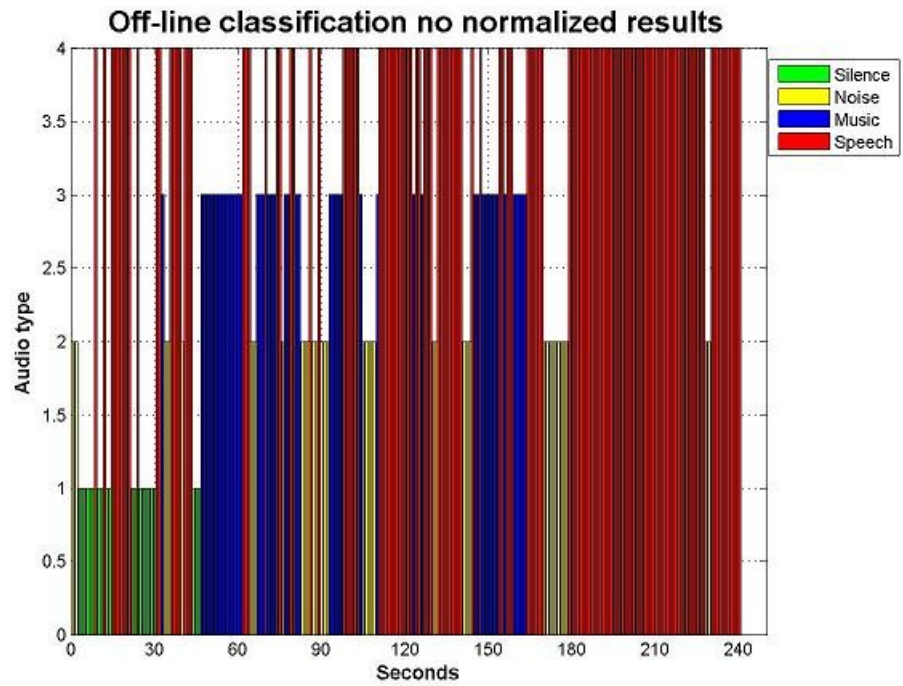


Figure 4-12: Second output file of function OffLineClassifier: "Off-line NO NORMALIZED results.jpg"

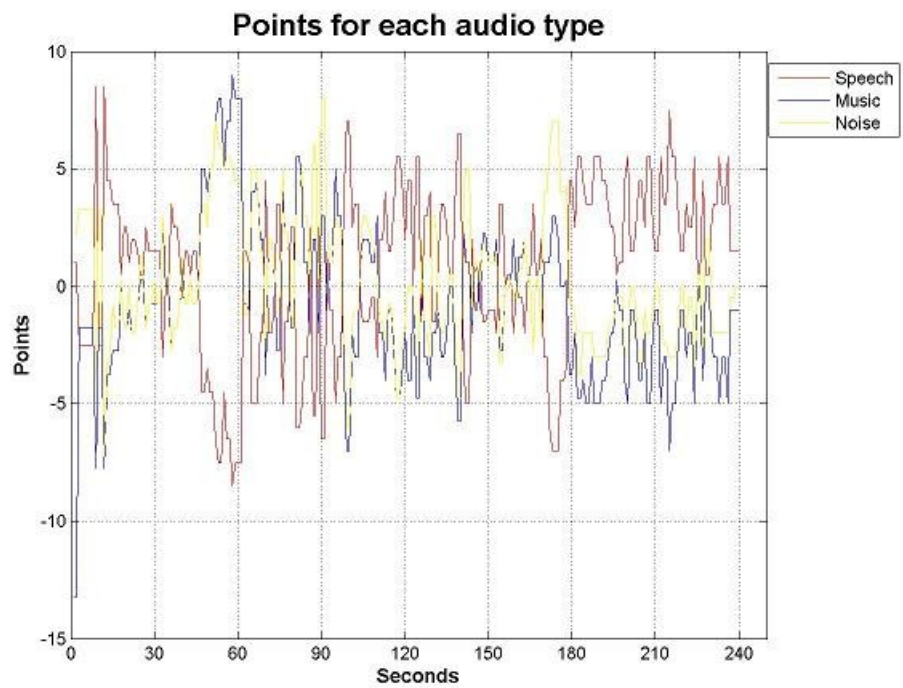


Figure 4-13: Third output file of function OffLineClassifier: "Points noise-speech-music.jpg"

4.6.2 Real-time classifier

Results plotting of the real-time classifier is almost identical to that of the off-line one. The only difference is that in this case the function performing this operation is separated from *RealTimeClassifier*, the one performing all classification tasks. The motivation that led to the separation of the two tasks into different functions was the exigency to prove that classifier could be able to satisfy real-time constraints. To do so, it had necessarily to be able to work and return its results independently from other functional blocks, like for example the plotting one.

The name of the function performing plotting of classification results (normalized and not) and of point vectors related to noise, speech and music is *PlotResults*, and it is stored in file “PlotResults.m”.

The output of the function is identical to that of *OffLineClassifier* in the off-line case (the three graphs already analyzed in paragraph 4.6.1); the only difference is the input of the function, that is identical to the output of *RealTimeClassifier*. The four input variables (whose meaning has been already explained in Paragraph 4.5.3.2) the function receives are:

1. *CLASSIFICATION_RESULT* ;
2. *points* ;
3. *DURATION* ;
4. *NumberOfFrames*.

The reason why the first two variables are necessary is obvious: they constitute the data set that has to be plotted; *DURATION* is necessary for the calculation of the scaling step; *NumberOfFrames* constitutes the upper limit of the for cycle regulating the normalization process and has to be passed to *PlotResults* for this reason.

4.7 Summary

The two figures below show the whole processing flow related to both the off-line

classifier (Figure 4-14) and the real-time one (Figure 4-15).

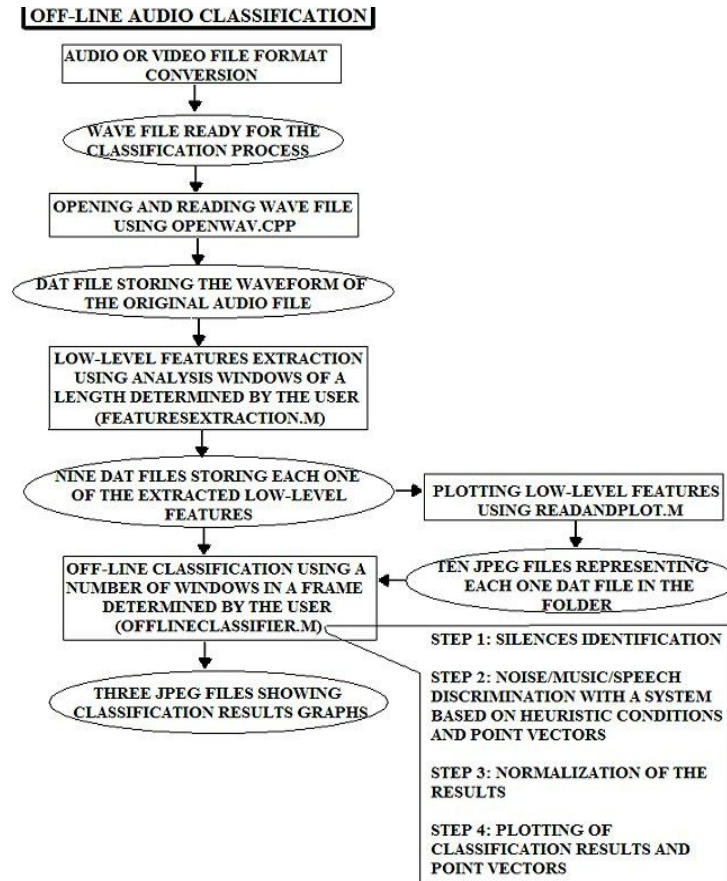


Figure 4-14: Flow chart representing all the working steps of the off-line audio classifier

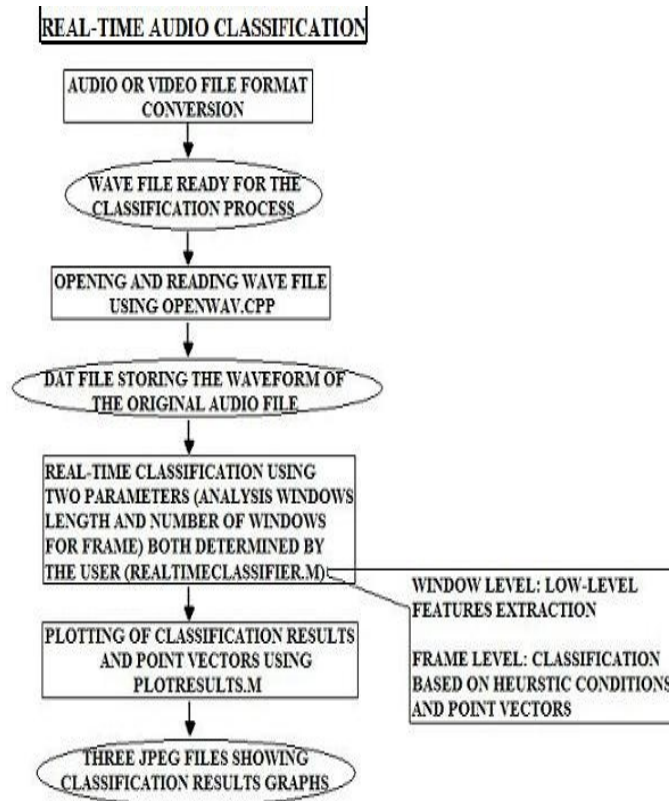


Figure 4-15: Flow chart representing all the working steps of the real-time audio classifier

5 Evaluations

5.1 Preliminary considerations

In this paragraph will be described all the preliminary steps that it is necessary to perform before being able to make a good evaluation of the results obtained using both off-line and real-time classifiers.

5.1.1 Manual classification and its limits

The first thing we do here is to provide an answer for a very important question: how is it possible to find the results the two classifiers *should* return for a perfect classification accuracy (the concept of classification accuracy will be better described in Paragraph 5.1.2)? *Manual classification* is the answer, but at this point it is very important to underline the fact that also manual classification is not at all an operation that returns perfect results.

Usually bigger studies on audio and video classification (having some economic aid or involving a group of researchers) use more than one person to evaluate the audio (or video) content for each part of the audio clip they want to segment in sub parts having some common property. This because manual classification is sometimes simple, but in some other cases it can be very subjective depending on the person that is listening (watching) the audio (video) clip. In the case of the present two classifiers, the most frequent thing it can happen that could lead to manual classification errors is the fact that often (especially when the audio clip is extracted from a film or a TV show) there is more than one overlapped audio type, so that it can become very difficult to decide which is the one dominating the others. This problem is evident in the last part of the audio clip we will use as an example for all the rest of this paragraph. As it can be seen from Table 5-1, manual classification says “speech” from second 41 until the end of the

audio clip (second 60). In reality, this part is also characterized by music and noise, whose presence is felt particularly during some long pauses of talker's voice. For this reason, another person listening the same part of the audio clip could decide to label it as “noise”, or “music”.

In the present project, manual classification has been performed by only one person (the author). This means that there has been only one person deciding the labels to assign to each second of the audio clip; this fact surely constitutes a limitation for the goodness of the material with which both classifiers' results have to be compared.

Moreover, manual classification was performed using a simple program called *Winamp* (produced by Microsoft). This program was used to read WAV files and to decide the label to assign to each second of the audio clip. Temporal sensibility of this program is one second, so sometimes there could be some trouble trying to find the exact moment of a transition between an audio type and another. The situation just mentioned surely determines a rise in the imperfection of the results provided by manual classification.

The unity measure chosen to partition the analyzed audio clips is the second; for example, an audio clip whose length is 4 minutes will be divided in 240 parts, each labeled with an audio type. One of the reasons of this choice (the second reason will be explained in paragraph 5.1.2) is the fact that it could reduce the impact of the small precision of the program used to perform the manual classification.

Apart from these considerations regarding the limits of manual classification, the process in itself consist simply in listening the audio clip and then subdividing it depending on the identified audio type. For each of the obtained sub parts it has to be reported the starting and the finishing second. All this information is stored in a DAT file called SUMMARY_audiofilename.dat; this file is then saved in the folder containing also the WAV file and all the programs projected for classification purposes and described in Chapter 4. An example showing the structure of this file is given below:

Music: 0:00 - 0:37 [s]

Noise: 0:37 - 0:40 [s]

Speech: 0:40 - 1:00 [s]

This example is related to a WAV file whose length is 1 minute; the same file will be used as a starting point for all considerations that will be made in Paragraph 5.1.

5.1.2 Choice of parameters to describe classifier quality

The main parameter that has to be evaluated for both classifiers is the overall *classification accuracy*, that can be evaluated dividing the number of audio parts correctly classified for the total number of parts in which the audio clip is divided. In other words, on count matches existing between manual and automatic classification results, and then on divide this sum for the length of the audio clip in seconds (because it has been chosen to assign to each audio part a length of one second). An example of classification accuracy referred to the same audio clip whose summary has been presented in Paragraph 5.1.1 is shown in Table 5-1 below.

Second of the audio clip	Manual classification results	Off-line automatic classification results	Match?
0:01	Music	<i>Music</i>	YES
0:02	Music	<i>Music</i>	YES
0:03	Music	<i>Music</i>	YES
0:04	Music	<i>Music</i>	YES
0:05	Music	<i>Music</i>	YES
0:06	Music	<i>Music</i>	YES
0:07	Music	<i>Music</i>	YES
0:08	Music	<i>Music</i>	YES
0:09	Music	<i>Music</i>	YES
0:10	Music	<i>Music</i>	YES
0:11	Music	<i>Music</i>	YES
0:12	Music	<i>Music</i>	YES
0:13	Music	<i>Noise</i>	NO
0:14	Music	<i>Noise</i>	NO

0:15	Music	<u>Noise</u>	NO
0:16	Music	<u>Noise</u>	NO
0:17	Music	<u>Noise</u>	NO
0:18	Music	<u>Noise</u>	NO
0:19	Music	<u>Noise</u>	NO
0:20	Music	<u>Noise</u>	NO
0:21	Music	<u>Noise</u>	NO
0:22	Music	<u>Music</u>	YES
0:23	Music	<u>Music</u>	YES
0:24	Music	<u>Music</u>	YES
0:25	Music	<u>Music</u>	YES
0:26	Music	<u>Music</u>	YES
0:27	Music	<u>Music</u>	YES
0:28	Music	<u>Music</u>	YES
0:29	Music	<u>Music</u>	YES
0:30	Music	<u>Music</u>	YES
0:31	Music	<u>Music</u>	YES
0:32	Music	<u>Music</u>	YES
0:33	Music	<u>Music</u>	YES
0:34	Music	<u>Music</u>	YES
0:35	Music	<u>Noise</u>	NO
0:36	Music	<u>Noise</u>	NO
0:37	Noise	<u>Noise</u>	YES
0:38	Noise	<u>Speech</u>	NO
0:39	Noise	<u>Speech</u>	NO
0:40	Noise	<u>Speech</u>	NO
0:41	Speech	<u>Speech</u>	YES
0:42	Speech	<u>Speech</u>	YES
0:43	Speech	<u>Speech</u>	YES
0:44	Speech	<u>Speech</u>	YES
0:45	Speech	<u>Music</u>	NO
0:46	Speech	<u>Music</u>	NO
0:47	Speech	<u>Noise</u>	NO
0:48	Speech	<u>Noise</u>	NO
0:49	Speech	<u>Noise</u>	NO
0:50	Speech	<u>Noise</u>	NO

0:51	Speech	<u>Noise</u>	NO
0:52	Speech	<u>Noise</u>	NO
0:53	Speech	<u>Music</u>	NO
0:54	Speech	<u>Music</u>	NO
0:55	Speech	<u>Music</u>	NO
0:56	Speech	<u>Music</u>	NO
0:57	Speech	<u>Music</u>	NO
0:58	Speech	<u>Speech</u>	YES
0:59	Speech	<u>Speech</u>	YES
1:00	Speech	<u>Speech</u>	YES
Number of correctly classified parts			33
Total number of seconds in the audio clip			60
Classification accuracy (CA)			33 / 60 = 0,55 → 55%

Table 5-1: An example of accuracy evaluation referred to the off-line classification of an audio clip of one minute (parameters: 1) number of samples for one window = 8192 → window length = 0.184 s; 2) number of windows for one frame = 10; 3) frame length = 1.84 s)

Classification accuracy depends on two other parameters set by user before the proper classification process begins; these are:

- *Length of a window in samples and number of windows for frame* in the off-line case;
- *Length of a window in seconds and number of windows for frame* in the real-time classifier.

In the off-line classifier, the number of samples for window will be translated each time in its correspondent measure in seconds. This will be done using the inverse of the equation used in the real-time classifier (function *RealTimeClassifier*) to calculate the number of samples of a window starting from its length in seconds:

$$\mathbf{Window\ length\ [s]} = \mathbf{Window\ length\ [samples] / Sample\ Rate\ [Hz = 1 / s]}$$

This translation is necessary to allow the use of a unique unity measure (seconds) for both the classifiers (this is the second motivation for choosing seconds as the unity measure for results evaluation; the first can be found in paragraph 5.1.1). In this way, it will be possible to evaluate the accuracy of the two classifiers for frames of the same or

different lengths and, among those characterized by the same length, how this parameter changes varying the combination of the two components of this length; these components are:

- *features window length*, expressed in seconds;
- *frame length*, expressed in windows.

For example, two frames characterized by the same length in seconds (let's say 2 seconds) can be composed by 10 windows of 0.2 s each, 4 windows of 0.5 s each, and so on. The general equation that rules this concept is:

$$\mathbf{Frame\ length\ [s]} = \mathbf{Window\ length\ [s]} * \mathbf{Frame\ length\ [windows]}$$

After seeing how classification accuracy varies changing the length of each analysis window and the number of windows for each frame, the combinations of these parameters giving the better results will be taken and used for further analysis. In particular, it will be performed an *error rate* analysis for each audio type. This means dividing the number of parts (seconds in our case) of an audio type that have been misclassified for the total number of parts of that particular audio type in the audio clip. For example, in Table 5-1, the error rate related to audio type “speech” can be evaluated in this way:

1. counting the number of speech parts that have not been correctly recognized by the classifier; in this case parts from 0:45 to 0:57 → **13**;
2. counting the total speech parts of the audio clip in “Manual classification results” column; in this case parts from 0:41 to 1:00 → **20**;
3. dividing the result of point 1 for that of point 2; in this case $\mathbf{ER}_{\text{speech}} = 13 / 20 = 0,65 = \mathbf{65\ \%}$.

Another parameter that can tell something about the functioning of the two classifiers projected in the present work is the error rate related to the probability an audio part of one specific type has to be confused with another audio type by the automatic classifier (*relative error rate*). For example, relative error rate related to the probability a “speech” frame has to be confused with a “noise” one can be evaluated in the following

way:

1. counting the number of speech parts that have been misclassified as noise; in this case parts from second 47 to second 52 → **6**;
2. counting the total number of speech parts that have not been correctly recognized by the classifier; in this case parts from 0:45 to 0:57 → **13**;
3. dividing the result of point 1 for that of point 2; in this case $\mathbf{RER}_{\text{speech-noise}} = 6 / 13 = 0,462 = \mathbf{46,2\%}$.

This parameter will be calculated for all the 12 possible misclassification combinations between audio types (the number of these combinations will decrease of 3 units for each missing audio type in a particular audio clip): silence-speech; silence-noise; silence-music; speech-noise; speech-music; speech-silence; noise-speech; noise-music; noise-silence; music-speech; music-noise; music-silence. The sum of the three combinations related to a same audio type will obviously be equal to 100%, for the way Relative Error Rate is evaluated.

Error rate and relative error rate will be useful to understand what are the major limits of the two classifiers and to try to discuss some possible solutions that could be applied in future works to overcome these limits.

The last parameter that will be evaluated is the *normalization improvement* (NI): this parameter describe how classification results get better or worse applying on them the normalization algorithm described in Paragraph 4.6.1. It can be calculated subtracting the classification accuracy obtained for the no normalized results for the same parameter calculated for the normalized ones. Its value will be positive in the case normalized results are nearer to manual classification ones (this situation would mean that normalization process improves performances of the classifier), negative if they are worse.

5.1.3 Computational time evaluation for real-time classifier

A very important parameter for the real-time classifier is the time it needs to automatically classify a frame; this time should be considerably lower than the duration of the same frame. To evaluate if our classifier is able to fulfill this aspect, computational time required by Matlab function *RealTimeClassifier* to classify a frame will be calculated for different frame lengths. This will be done using standard Matlab functions *tic* and *toc*, whose functioning is very simple: it is only necessary to put *tic* at a certain point of the Matlab script, and *toc* at another point. While program is running, this simple operation makes the program calculate the temporal interval elapsed between *tic* and *toc*. The output of the combined use of these two functions is displayed as quick as possible on Matlab command prompt:

Elapsed time is x seconds

If *tic* and *toc* are in the middle of a cycle (for example a *while* or a *for* cycle), this output will be repeated for each round of the cycle; if *toc* is inserted in a condition (for example, an *if*), time computation between *tic* and *toc* will stop only when this condition will be satisfied by the program. In other words, *tic* and *toc* can be considered as normal instructions: they are activated when (and if) the execution of the program flow passes from them.

In our case, it seems particularly useful putting *tic* just before the beginning of the outer *for* cycle and *toc* at the end of the same cycle in *RealTimeClassifier* (see Paragraph 4.5.3.2 for details). That is exactly what it will be done. In this way, it will be evaluated the time classifier needs to fulfill all its tasks for the entire frame; then, being *NumberOfFrames* one of the output parameters of function *RealTimeClassifier*, it will be possible to evaluate the average time classifier needs to work on a single frame using the following expression:

**Medium Frame Processing Time [s] = Whole Clip Processing Time [s] * Number
of Frames**

Results of this process, as well as the evaluation of all parameters listed in Paragraph

5.1.2, will be provided in Paragraph 5.2.

5.1.4 Data set used to evaluate the work of both classifiers

The data set consists of three audio clips taken from three different files characterized by different audio qualities. Two of these three files are quite famous movies, and the third can be defined as a “documentary on music”. Length of the two films extracts is 3 minutes each, while the documentary is 4 minutes long, for a total of 10 minutes. In reality, more than two hours of audio taken from these and others films have been analyzed and classified; however, these three audio clips can be considered sufficient to demonstrate the fact that both off-line and real-time classifier can be applied on different genres with different audio qualities, without changing considerably the obtained results in terms of classification accuracy.

In table 5-2 there is a description of the salient characteristics related to the three audio clips used for both off-line and real-time classifier accuracy evaluation.

Audio clip extracted from	Length (in seconds)	Audio quality	Total number of seconds for each audio type (manual classification)
No Country For Old Men (movie)	180	Good	Speech → 114 Noise → 66
Sin City (movie)	180	Medium	Music → 97 Noise → 4 Speech → 79
Good Copy Bad Copy (documentary on piracy in music)	240	Bad	Noise → 17 Silence → 10 Speech → 173

Table 5-2: Main features of the three audio clip employed for the evaluation of both the off-line and the real-time classifier

5.2 Results evaluation

All the parameters will be calculated for:

- all the three audio clips previously described in Paragraph 5.1.4;
- different lengths of the analysis window used to extract low-level features and of the length of each frame on which the classification algorithm is applied.

5.2.1 Classification accuracy

The first and the most important parameter we are going to evaluate for both the off-line and the real-time classifiers is the *Classification Accuracy* (CA).

The results obtained for this parameter are all listed in Table 5-3 below.

Frame length	Analysis window length	Windows for frame	CA off-line classifier			CA real-time classifier			Average CA
			NCFOM	SC	GCBC	NCFOM	SC	GCBC	
1.3 s	0.093 s	14	71,1%	72,8%	78,3%	74,5%	71,1%	77,9%	74,3%
1.39 s	0.139 s	10	82,8%	75,6%	82,1%	73,3%	60,0%	76,3%	75,0%
1.49 s	0.093 s	16	83,3%	72,2%	82,9%	78,9%	67,8%	77,9%	77,2%
1.49 s	0.185 s	8	77,8%	74,4%	67,5%	70,0%	60,6%	68,9%	69,9%
1.85 s	0.093 s	20	80,6%	81,1%	84,2%	81,7%	81,1%	74,6%	80,6%
1.85 s	0.185 s	10	75,0%	81,1%	77,5%	71,7%	71,7%	77,1%	75,7%
1.85 s	0.372 s	5	63,9%	62,8%	50,0%	45,6%	52,2%	37,9%	52,1%
1.95 s	0.139 s	14	84,4%	75,6%	82,9%	81,1%	76,1%	75,8%	79,3%
2.23 s	0.185 s	12	92,2%	79,4%	82,5%	68,9%	72,2%	79,6%	79,1%
2.6 s	0.185 s	14	82,8%	81,7%	80,4%	88,9%	75,0%	74,6%	80,6%
Average CA			79,4%	75,7%	76,8%	73,5%	68,8%	72,1%	

Table 5-3: Classification accuracy (CA) evaluated for the three audio clips varying the length of the frame and of the analysis window. NCFOM stands for “No Country For Old Men”, SC stands for “Sin City” and GCBC is “Good Copy Bad Copy”. The three best results obtained for each audio clip are evidenced in bold

The first consideration that it can be done watching results in Table 5-3 is that *off-line classifier obtains better results than real-time one*. This happens for all the three analyzed audio clips and with almost all the combinations of length of the analysis window and number of windows in one frame. Watching at the average classification accuracy obtained by the two classifiers (last row of the table) it is possible to notice

that off-line classifier is characterized by an higher overall precision than the real-time one for all the audio clips; the difference between the two classifiers ranges from the 4,7% of “Good Copy Bad Copy” to the 6,9% of “Sin City”. This difference was highly predictable; it can be explained by the fact that real-time classification algorithm doesn't use Root Mean Square and normalized measures related to low-level features, and so it is less precise than the off-line one.

Another important consideration is related to the length of the analysis window used for the extraction of the low level features. It can be seen watching the columns of Table 5-3 that, *using frames of equal length in seconds, better results are often obtained if shorter analysis windows are used*. For example compare the two upper or the three lower data rows in Table 5-4 below, that is an extract taken from the results shown in Table 5-3. It shows that almost all the times we keep fixed the total frame length increasing the length of each analysis window, global classification accuracy decreases.

Frame length	Analysis window length	Windows for frame	CA off-line classifier			CA real-time classifier			Average CA
			NCFOM	SC	GCBC	NCFOM	SC	GCBC	
1.49 s	<u>0.093 s</u>	16	83,3%	72,2%	82,9%	78,9%	67,8%	77,9%	<u>77,2%</u>
1.49 s	0.185 s	8	77,8%	74,4%	67,5%	70,0%	60,6%	68,9%	69,9%
1.85 s	<u>0.093 s</u>	20	80,6%	81,1%	84,2%	81,7%	81,1%	74,6%	<u>80,6%</u>
1.85 s	0.185 s	10	75,0%	81,1%	77,5%	71,7%	71,7%	77,1%	75,7%
1.85 s	0.372 s	5	63,9%	62,8%	50,0%	45,6%	52,2%	37,9%	52,1%

Table 5-4: Extract taken from Table 5-3, that shows how both classifiers improve their performance if using shorter analysis windows. The two exceptions to this rule are evidenced in bold. The two combinations returning the best classification results are underlined. NCFOM stands for “No Country For Old Men”, SC stands for “Sin City” and GCBC is “Good Copy Bad Copy”

On the other hand it can be stated with a certain confidence level that the *number of windows belonging to the same frame affect in a limited way classification accuracy*. Moreover, the length of each frame in which the audio clip is divided, affects in some way the accuracy of the classification process. For example this can be seen from figures 5-1, 5-2 and from Table 5-5 that, keeping the same analysis window and increasing the number of windows belonging to a single frame, obtained results that

follow a generic trend of grow but with a lot of exceptions. On the other hand, it can be noticed from Table 5-2 how frames shorter than 1.5 seconds or longer than 3 seconds (results of this case are not shown) don't return significantly good classification results with respect to frame lengths between 1.85 and 2.6 seconds; this can be considered the optimal working range of both the off-line and the real-time classifier.

Frame length	Analysis window length	Windows for frame	CA off-line classifier			CA real-time classifier			Average CA
			NCFOM	SC	GCBC	NCFOM	SC	GCBC	
1.49 s	0.185 s	8	77,8%	74,4%	67,5%	70,0%	60,6%	68,9%	69,9%
1.85 s	0.185 s	10	75,0%	81,1%	77,5%	71,7%	71,7%	77,1%	75,7%
2.23 s	0.185 s	12	92,2%	79,4%	82,5%	68,9%	72,2%	79,6%	79,1%
2.6 s	0.185 s	14	82,8%	81,7%	80,4%	88,9%	75,0%	74,6%	80,6%

Table 5-5: Extract taken from Table 5-3, that shows how both classifiers performance is affected in a limited way by the number of windows belonging to one single frame. NCFOM stands for “No Country For Old Men”, SC stands for “Sin City” and GCBC is “Good Copy Bad Copy”

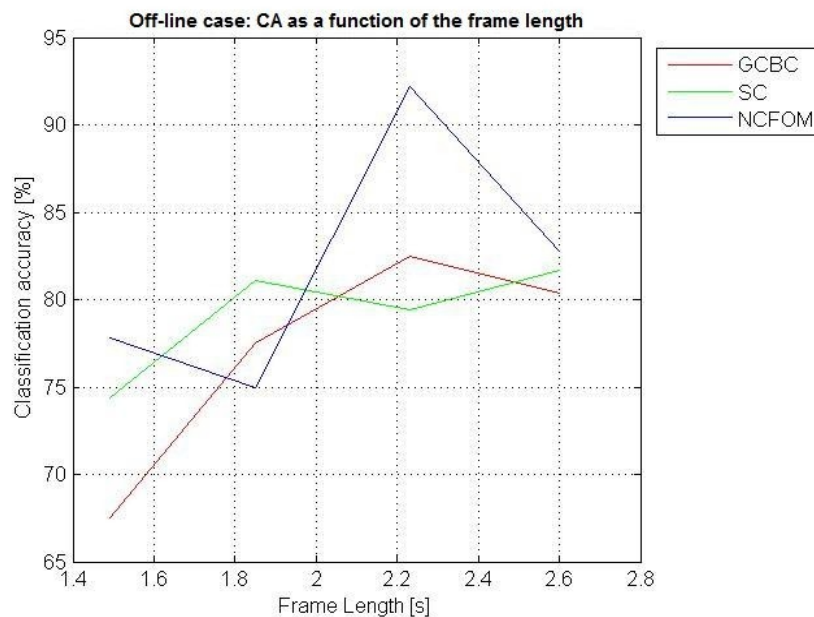


Figure 5-1: Graph showing how CA changes in the off-line case increasing the number of analysis windows in a frame and keeping the same analysis window length (0,185 s)

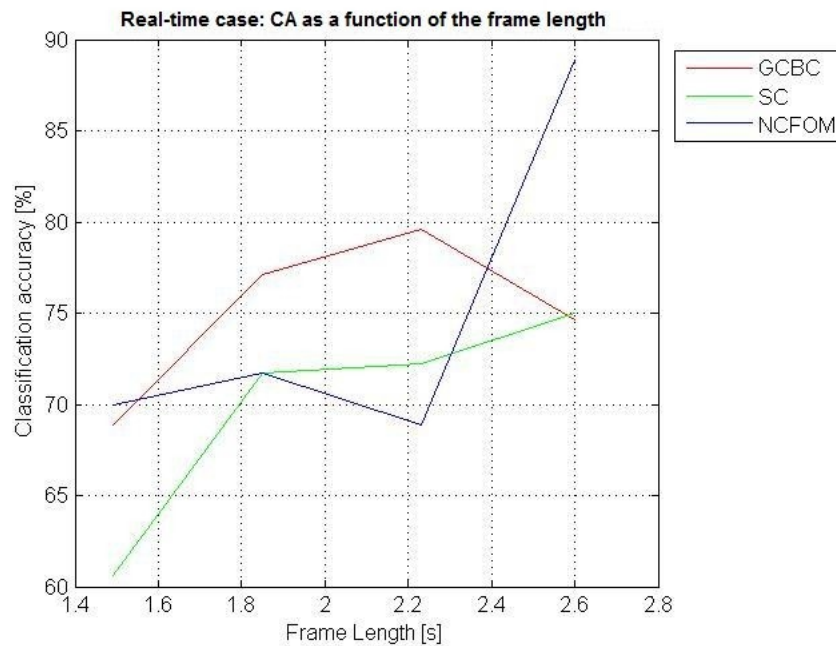


Figure 5-2: Graph showing how CA changes in the real-time case increasing the number of analysis windows in a frame and keeping the same analysis window length (0,185 s)

5.2.2 Error rate

The second parameter whose obtained results will be discussed is the Error Rate (ER) calculated for each of the four audio types – speech, music, noise, silence – in which both classifiers try to divide the analyzed audio clip.

Table 5-6 lists the values obtained by this parameter for speech parts. As it is possible to notice looking at this table, results in this case are generally quite good. This because speech (with silence) is probably the audio type that can be identified and recognized in the easiest way. Almost optimal results are obtained using a frame of 1,85 seconds, constituted by 20 analysis windows each lasting 0.093 seconds. On the other hand, very bad results (more than 60% misclassified speech frames on average) are obtained using a long analysis window (0,392 seconds).

The best results are obtained using the off-line classifier on the audio clip with the higher quality level, and this consideration was highly expected. A strange thing is the fact that the clip characterized by the worse quality level reaches an higher level of speech identification than the medium one and, furthermore, obtains better results with

the real-time classification algorithm than with the other one. This last consideration can be explained by the fact that clips with a bad audio quality tend to assume values not very uniform for the presence of background noise throughout all the length of the audio clip; this constitutes an help for the identification of speech true positives, but leads also to the presence of a quantity of false positives, as it is also possible to notice analyzing the other tables of this paragraph and those of the Relative Error Rate in paragraph 5.2.3.

Frame length	Analysis window length	Windows for frame	ER _{speech} off-line classifier			ER _{speech} real-time classifier			Average ER _{speech}
			NCFOM	SC	GCBC	NCFOM	SC	GCBC	
1.3 s	0.093 s	14	19,2%	18,8%	14,5%	24,8%	22,5%	4,1%	17,3%
1.39 s	0.139 s	10	11,2%	22,5%	11,0%	24,8%	30,0%	7,6%	17,9%
1.49 s	0.093 s	16	9,6%	25,0%	7,6%	10,4%	32,5%	2,3%	14,6%
1.49 s	0.185 s	8	12,0%	40,0%	32,0%	25,6%	30,0%	22,7%	27,0%
<u>1.85 s</u>	<u>0.093 s</u>	<u>20</u>	4,0%	1,3%	0,6%	12,8%	6,3%	0,6%	<u>4,3%</u>
1.85 s	0.185 s	10	13,6%	20,0%	16,3%	24,8%	18,8%	6,4%	16,7%
1.85 s	0.372 s	5	30,4%	67,5%	56,4%	67,2%	76,3%	72,7%	61,8%
<u>1.95 s</u>	<u>0.139 s</u>	<u>14</u>	7,2%	25,0%	0,0%	3,2%	7,5%	0,0%	<u>7,2%</u>
2.23 s	0.185 s	12	4,8%	18,8%	0,0%	32,8%	20,0%	0,0%	12,7%
2.6 s	0.185 s	14	9,6%	13,8%	3,5%	2,4%	12,5%	3,5%	7,6%
Average ER_{speech}			12,2%	25,3%	14,2%	22,9%	25,6%	12,0%	

Table 5-6: Error Rate (ER) related to speech frames and evaluated for the three audio clips varying the length of the frame and of the analysis window. The two best results obtained for each audio clip are evidenced in bold. The two combinations returning the best classification results are underlined.

The situation concerning the recognition of musical parts in an audio clip is worse than the speech one. Here, the quality of the considered audio clip plays an important role. As it can be noticed looking at Table 5-7 below, the percentage of misclassified music frames in “Sin City” (medium audio quality) is significantly lower (around 35%) than “Good Copy Bad Copy” (bad audio quality) both with the off-line and the real-time classifiers. Good results are obtained especially when using longer analysis windows.

Also in this case off-line classifier works almost ever better than real-time one, but

with some important exceptions, such as in the case of frames that last 1,85 seconds using 20 analysis windows for the clip characterized by medium audio quality (30,2% obtained by real-time classifier against 34,4% of the off-line one).

Frame length	Analysis window length	Windows for frame	ER _{music} off-line classifier			ER _{music} real-time classifier			Average ER _{music}
			NCFOM	SC	GCBC	NCFOM	SC	GCBC	
1.3 s	0.093 s	14	-	32,3%	55,0%	-	35,4%	67,5%	47,6%
1.39 s	0.139 s	10	-	24,0%	50,0%	-	46,9%	60,0%	45,2%
1.49 s	0.093 s	16	-	29,2%	55,0%	-	29,2%	77,5%	47,7%
<u>1.49 s</u>	<u>0.185 s</u>	<u>8</u>	-	13,5%	52,5%	-	45,8%	47,5%	<u>39,8%</u>
1.85 s	0.093 s	20	-	34,4%	57,5%	-	30,2%	100,0%	55,5%
1.85 s	0.185 s	10	-	15,6%	52,5%	-	35,4%	57,5%	40,3%
<u>1.85 s</u>	<u>0.372 s</u>	<u>5</u>	-	10,4%	42,5%	-	26,0%	35,0%	<u>28,5%</u>
1.95 s	0.139 s	14	-	22,9%	67,5%	-	35,4%	90,0%	54,0%
2.23 s	0.185 s	12	-	19,8%	70,0%	-	33,3%	72,5%	48,9%
2.6 s	0.185 s	14	-	19,8%	72,5%	-	33,3%	82,5%	52,0%
Average ER_{music}			-	22,2%	57,5%	-	35,4%	69,0%	

Table 5-7: Error Rate (ER) related to music frames and evaluated for the three audio clips varying the length of the frame and of the analysis window. The two best results obtained for each audio clip are evidenced in bold. The two combinations returning the best classification results are underlined.

Noise is the most difficult audio type to identify and classify correctly, as already underlined in paragraph 3.2.3. However, results obtained by the two classifiers described in the present work are quite encouraging in this sense. In fact, with the exception of the real-time classifier working on the lower audio quality clip, noise is often identified correctly, especially in the clip “No Country For Old Men”. Like music, also in noise classification the best results are obtained using a longer analysis window, but in a lot of cases also short windows work well in this sense. For example, using in a frame 20 analysis windows of 0,093 seconds each, the error rate related to noise identification is of approximately 40% on average.

Frame length	Analysis window length	Windows for frame	ER _{noise} off-line classifier			ER _{noise} real-time classifier			Average ER _{noise}
			NCFOM	SC	GCBC	NCFOM	SC	GCBC	
1.3 s	0.093 s	14	50,9%	75,0%	22,2%	25,5%	0,0%	100,0%	45,6%
1.39 s	0.139 s	10	30,9%	75,0%	11,1%	30,9%	75,0%	100,0%	53,8%
1.49 s	0.093 s	16	32,7%	50,0%	27,8%	41,8%	100,0%	100,0%	58,7%
1.49 s	0.185 s	8	45,5%	25,0%	11,1%	40,0%	75,0%	88,9%	47,6%
<u>1.85 s</u>	<u>0.093 s</u>	<u>20</u>	54,5%	0,0%	61,1%	30,9%	0,0%	94,4%	<u>40,2%</u>
1.85 s	0.185 s	10	50,9%	75,0%	16,7%	36,4%	50,0%	100,0%	54,8%
<u>1.85 s</u>	<u>0.372 s</u>	<u>5</u>	49,1%	75,0%	16,7%	23,6%	0,0%	38,9%	<u>33,9%</u>
1.95 s	0.139 s	14	34,5%	50,0%	66,7%	54,5%	75,0%	100,0%	63,5%
2.23 s	0.185 s	12	14,5%	75,0%	61,1%	27,3%	50,0%	88,9%	52,8%
2.6 s	0.185 s	14	34,5%	75,0%	38,9%	30,9%	75,0%	100,0%	59,1%
Average ER_{noise}			39,8%	57,5%	33,3%	34,2%	50,0%	91,1%	

Table 5-8: Error Rate (ER) related to noise frames and evaluated for the three audio clips varying the length of the frame and of the analysis window. The two best results obtained for each audio clip are evidenced in bold. The two combinations returning the best classification results are underlined.

Silence is usually correctly identified by both classifiers, obtaining better results when shorter frames are used. This because a long frame can be a limit to the precision of the classifier; in other words, the longer a frame is, higher is the probability for that frame to contain a no silence part, and to incorrectly classify the whole frame. Also for this parameter, off-line classifier works better than the real-time one; this because the former uses two low-level features to identify silences, against the one adopted by the latter.

Frame length	Analysis window length	Windows for frame	ER _{silence} off-line classifier			ER _{silence} real-time classifier			Average ER _{silence}
			NCFOM	SC	GCBC	NCFOM	SC	GCBC	
<u>1.3 s</u>	<u>0.093 s</u>	<u>14</u>	-	-	0,0%	-	-	10,0%	<u>5,0%</u>
1.39 s	0.139 s	10	-	-	20,0%	-	-	20,0%	20,0%
1.49 s	0.093 s	16	-	-	10,0%	-	-	10,0%	10,0%
<u>1.49 s</u>	<u>0.185 s</u>	<u>8</u>	-	-	0,0%	-	-	10,0%	<u>5,0%</u>
1.85 s	0.093 s	20	-	-	30,0%	-	-	30,0%	30,0%
1.85 s	0.185 s	10	-	-	20,0%	-	-	30,0%	25,0%

1.85 s	0.372 s	5	-	-	30,0%	-	-	30,0%	30,0%
1.95 s	0.139 s	14	-	-	20,0%	-	-	40,0%	30,0%
2.23 s	0.185 s	12	-	-	30,0%	-	-	40,0%	35,0%
2.6 s	0.185 s	14	-	-	50,0%	-	-	50,0%	50,0%
Average ER_{silence}			-	-	21,0%	-	-	27,0%	

Table 5-9: Error Rate (ER) related to silence frames and evaluated for the three audio clips varying the length of the frame and of the analysis window. The two best results obtained for each audio clip are evidenced in bold. The two combinations returning the best classification results are underlined.

Both classifiers work very well in the field of speech and silence recognition, a little worse for noise and music fragments. However, there are some exceptions to this rule; for example, using longer windows results in a slight decrease of the error rate both in music and noise recognition, but also in a big increase of this parameter in speech recognition. An idea to improve the overall performances would be that of combining the benefits obtained by the two different lengths, using longer windows for noise and music identification, and shorter ones for speech discrimination. This solution would be complicated when working with real-time constraints, but can be applied quite easily in the off-line case.

5.2.3 Relative error rate

Relative Error Rate (RER) constitutes an important parameter, because it is useful to discover what are the audio types that more likely generate confusion into the classification algorithm.

Table 5-10 shows speech RER; it seems evident from the results that errors are more likely to occur with noise than other audio types. Only using a longer analysis window (0,372 seconds) speech frames are confused more with music than with noise ones. On the other hand, real time classifier never classifies a speech frame as a silence one, while some errors in this sense occur applying the off-line classifier on the highest quality audio clip (“No Country For Old Men”) and, rarely, on the lowest quality clip (“Good Copy Bad Copy”). Sin City's speech-music RER is very often higher than the

same parameter evaluated for the other two audio clips; this because in some parts there is soft music overlapped with speech, and these parts were almost all the times labeled as “speech” during the manual classification process.

Frame length	Analysis window length	Windows for frame	Parameter	RER _{speech} off-line classifier			RER _{speech} real-time classifier		
				NCFO M	SC	GCBC	NCFO M	SC	GCBC
1.3 s	0.093 s	14	RER _{speech-music}	4,0%	46,7%	24,0%	3,2%	44,4%	14,3%
			<u>RER_{speech-noise}</u>	76,0%	53,3%	24,0%	96,8%	55,6%	85,7%
			RER_{speech-silence}	20,0%	0,0%	52,0%	0,0%	0,0%	0,0%
1.39 s	0.139 s	10	RER _{speech-music}	7,1%	100,0%	42,1%	16,1%	50,0%	30,8%
			<u>RER_{speech-noise}</u>	92,9%	0,0%	57,9%	83,9%	50,0%	69,2%
			RER_{speech-silence}	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
1.49 s	0.093 s	16	RER _{speech-music}	0,0%	45,0%	15,4%	0,0%	61,5%	0,0%
			<u>RER_{speech-noise}</u>	100,0%	55,0%	84,6%	100,0%	38,5%	100,0%
			RER_{speech-silence}	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
1.49 s	0.185 s	8	RER _{speech-music}	0,0%	31,20%	56,4%	21,9%	70,8%	76,9%
			<u>RER_{speech-noise}</u>	60,0%	68,8%	25,5%	78,1%	29,2%	23,1%
			RER_{speech-silence}	40,0%	0,0%	18,2%	0,0%	0,0%	0,0%
1.85 s	0.093 s	20	RER_{speech-music}	0,0%	0,0%	0,0%	0,0%	40,0%	0,0%
			<u>RER_{speech-noise}</u>	20,0%	100,0%	100,0%	100,0%	60,0%	100,0%
			RER _{speech-silence}	80,0%	0,0%	0,0%	0,0%	0,0%	0,0%
1.85 s	0.185 s	10	RER _{speech-music}	0,0%	31,2%	71,4%	0,0%	80,0%	45,5%
			<u>RER_{speech-noise}</u>	52,9%	68,8%	28,6%	100,0%	20,0%	54,5%
			RER_{speech-silence}	47,1%	0,0%	0,0%	0,0%	0,0%	0,0%
1.85 s	0.372 s	5	<u>RER_{speech-music}</u>	26,3%	59,3%	85,6%	59,5%	95,1%	93,6%
			RER _{speech-noise}	36,8%	40,7%	14,4%	40,5%	4,9%	6,4%
			RER_{speech-silence}	36,8%	0,0%	0,0%	0,0%	0,0%	0,0%
1.95 s	0.139 s	14	RER _{speech-music}	0,0%	20,0%	-	0,0%	0,0%	-
			<u>RER_{speech-noise}</u>	100,0%	80,0%	-	100,0%	100,0%	-
			RER_{speech-silence}	0,0%	0,0%	-	0,0%	0,0%	-
2.23 s	0.185 s	12	RER _{speech-music}	0,0%	60,0%	-	4,9%	75,0%	-

			<u>RER_{speech-noise}</u>	100,0%	40,0%	-	95,1%	25,0%	-
			RER_{speech-silence}	0,0%	0,0%	-	0,0%	0,0%	-
2.6 s	0.185 s	14	RER _{speech-music}	0,0%	27,3%	100,0%	0,0%	0,0%	83,3%
			<u>RER_{speech-noise}</u>	25,0%	72,7%	0,0%	100,0%	100,0%	16,7%
			RER_{speech-silence}	75,0%	0,0%	0,0%	0,0%	0,0%	0,0%

Table 5-10: Relative Error Rate (RER) related to speech frames and evaluated for the three audio clips varying the length of the frame and of the analysis window. Lowest RER for each combination is evidenced in bold, while the highest is underlined

In table 5-11 below all the values related to music RER for all the three audio clips are listed. As speech, also music is almost never confused with silence by both classifiers. A different behavior characterizes $RER_{music-speech}$ and $RER_{music-noise}$, depending on the analyzed audio clip; in fact, while in “Sin City” music is most of the times confused with noise, in “Good Copy Bad Copy” those musical frames that are not recognized correctly by both classifiers are very often labeled as “speech”. This because in the former music is more harmonic (being made principally of two instruments – trumpet and piano) while in the latter it is harder and it is accompanied by a vocal presence in some points.

Frame length	Analysis window length	Windows for frame	Parameter	RER _{music} off-line classifier			RER _{music} real-time classifier		
				NCFO M	SC	GCBC	NCFO M	SC	GCBC
1.3 s	0.093 s	14	RER _{music-speech}	-	3,2%	90,9%	-	14,7%	66,7%
			<u>RER_{music-noise}</u>	-	96,8%	9,1%	-	85,3%	33,3%
			RER_{music-silence}	-	0,0%	0,0%	-	0,0%	0,0%
1.39 s	0.139 s	10	RER _{music-speech}	-	0,0%	90,0%	-	6,7%	79,2%
			<u>RER_{music-noise}</u>	-	100,0%	10,0%	-	93,3%	20,8%
			RER_{music-silence}	-	0,0%	0,0%	-	0,0%	0,0%
1.49 s	0.093 s	16	RER _{music-speech}	-	3,6%	72,7%	-	21,4%	71,0%
			<u>RER_{music-noise}</u>	-	96,4%	27,3%	-	78,6%	29,0%
			RER_{music-silence}	-	0,0%	0,0%	-	0,0%	0,0%
1.49 s	0.185 s	8	RER _{music-speech}	-	0,0%	81,0%	-	4,5%	100,0%
			<u>RER_{music-noise}</u>	-	100,0%	14,3%	-	95,5%	0,0%
			RER_{music-silence}	-	0,0%	4,8%	-	0,0%	0,0%
1.85 s	0.093 s	20	RER _{music-speech}	-	3,0%	95,7%	-	13,8%	67,5%

			<u>RER_{music-noise}</u>	-	97,0%	4,3%	-	86,2%	32,5%
			RER_{music-silence}	-	0,0%	0,0%	-	0,0%	0,0%
1.85 s	0.185 s	10	RER _{music-speech}	-	0,0%	95,2%	-	11,8%	91,3%
			<u>RER_{music-noise}</u>	-	100,0%	4,8%	-	88,2%	8,7%
			RER_{music-silence}	-	0,0%	0,0%	-	0,0%	0,0%
1.85 s	0.372 s	5	RER _{music-speech}	-	0,0%	76,5%	-	8,0%	92,9%
			<u>RER_{music-noise}</u>	-	100,0%	23,5%	-	92,0%	7,1%
			RER_{music-silence}	-	0,0%	0,0%	-	0,0%	0,0%
1.95 s	0.139 s	14	RER _{music-speech}	-	0,0%	92,6%	-	5,9%	63,9%
			<u>RER_{music-noise}</u>	-	100,0%	7,4%	-	94,1%	36,1%
			RER_{music-silence}	-	0,0%	0,0%	-	0,0%	0,0%
2.23 s	0.185 s	12	RER _{music-speech}	-	10,5%	85,7%	-	6,3%	75,9%
			<u>RER_{music-noise}</u>	-	78,9%	14,3%	-	93,7%	24,1%
			RER_{music-silence}	-	10,5%	0,0%	-	0,0%	0,0%
2.6 s	0.185 s	14	<u>RER_{music-speech}</u>	-	26,3%	96,6%	-	9,4%	75,8%
			RER _{music-noise}	-	73,7%	3,4%	-	90,6%	24,2%
			RER_{music-silence}	-	0,0%	0,0%	-	0,0%	0,0%

Table 5-11: Relative Error Rate (RER) related to music frames and evaluated for the three audio clips varying the length of the frame and of the analysis window. Lowest RER for each combination is evidenced in bold, while the highest is underlined.

Like music and speech, also noise is almost never misclassified as “silence” by neither off-line nor real-time classifier. This confirms the goodness of the projected silence identification algorithm; the only exception to these considerations comes from the clip extracted from “No Country For Old Men”, because some of its noise parts, labeled as “noise” during the manual classification process, are characterized by sounds that are almost inaudible for a neglectful listener. On the other hand, the parameter named $RER_{noise-speech}$ obtains high (bad) values for all the three analyzed audio clips. In particular, the clip extracted from the movie “Sin City” has only 4 seconds labeled as “noise” in the manual classification; this noise, however, can be easily confused with speech for its characteristics. As it is possible to see also from Table 5-12, noise is very rarely confused with music by both classifiers: this fact demonstrates that the heuristic conditions used to distinguish between these two audio types (a task that very often is

not so simple to accomplish) work with a good efficiency level during the classification process.

Frame length	Analysis window length	Windows for frame	Parameter	RER _{noise} off-line classifier			RER _{noise} real-time classifier		
				NCFOM	SC	GCBC	NCFOM	SC	GCBC
1.3 s	0.093 s	14	<u>RER</u> _{noise-speech}	39,3%	100,0%	100,0%	14,3%	-	61,1%
			RER _{noise-music}	25,0%	0,0%	0,0%	85,7%	-	38,9%
			RER _{noise-silence}	35,7%	0,0%	0,0%	0,0%	-	0,0%
1.39 s	0.139 s	10	<u>RER</u> _{noise-speech}	41,2%	100,0%	100,0%	41,2%	100,0%	55,6%
			RER _{noise-music}	52,9%	0,0%	0,0%	58,8%	0,0%	44,4%
			RER _{noise-silence}	5,8%	0,0%	0,0%	0,0%	0,0%	0,0%
1.49 s	0.093 s	16	<u>RER</u> _{noise-speech}	44,4%	100,0%	100,0%	21,7%	50,0%	72,2%
			RER _{noise-music}	55,6%	0,0%	0,0%	78,3%	50,0%	27,8%
			RER _{noise-silence}	0,0%	0,0%	0,0%	0,0%	0,0%	0,0%
1.49 s	0.185 s	8	RER _{noise-speech}	20,0%	100,0%	50,0%	31,8%	100,0%	25,0%
			RER _{noise-music}	44,0%	0,0%	50,0%	68,2%	0,0%	75,0%
			<u>RER</u> _{noise-silence}	36,0%	0,0%	0,0%	0,0%	0,0%	0,0%
1.85 s	0.093 s	20	<u>RER</u> _{noise-speech}	56,7%	-	100,0%	76,5%	-	70,6%
			RER _{noise-music}	6,7%	-	0,0%	23,5%	-	29,4%
			RER _{noise-silence}	36,7%	-	0,0%	0,0%	-	0,0%
1.85 s	0.185 s	10	<u>RER</u> _{noise-speech}	14,3%	100,0%	100,0%	35,0%	100,0%	44,4%
			RER _{noise-music}	25,0%	0,0%	0,0%	65,0%	0,0%	55,6%
			RER _{noise-silence}	60,7%	0,0%	0,0%	0,0%	0,0%	0,0%
1.85 s	0.372 s	5	<u>RER</u> _{noise-speech}	3,7%	100,0%	66,7%	0,0%	-	85,7%
			RER _{noise-music}	33,3%	0,0%	33,30%	100,0%	-	14,3%
			RER _{noise-silence}	63,0%	0,0%	0,0%	0,0%	-	0,0%
1.95 s	0.139 s	14	<u>RER</u> _{noise-speech}	68,4%	100,0%	100,0%	73,3%	100,0%	55,6%
			RER _{noise-music}	31,6%	0,0%	0,0%	26,7%	0,0%	38,9%
			RER _{noise-silence}	0,0%	0,0%	0,0%	0,0%	0,0%	5,6%
2.23 s	0.185 s	12	<u>RER</u> _{noise-speech}	75,0%	100,0%	90,9%	53,3%	100,0%	62,6%
			RER _{noise-music}	0,0%	0,0%	0,0%	46,7%	0,0%	31,3%

			RER_{noise-silence}	25,0%	0,0%	9,1%	0,0%	0,0%	6,3%
2.6 s	0.185 s	14	<u>RER_{noise-speech}</u>	52,6%	100,0%	85,7%	70,6%	100,0%	61,1%
			RER _{noise-music}	0,0%	0,0%	14,3%	29,4%	0,0%	38,9%
			RER_{noise-silence}	47,4%	0,0%	0,0%	0,0%	0,0%	0,0%

Table 5-12: Relative Error Rate (RER) related to noise frames and evaluated for the three audio clips varying the length of the frame and of the analysis window. Lowest RER for each combination is evidenced in bold, while the highest is underlined.

Silence, as it is possible to see from table 5-13 below, appears only in one of the three analyzed audio clips: “Good Copy Bad Copy”. When it is not correctly recognized, silence is always confused by both classifiers with speech: this situation can be easily explained considering that there is a speech part immediately after the silence one; furthermore, the partition of the audio clip in intervals of a fixed length surely helps the interpretation of an half-silence-half-speech frame as a fully speech one.

Frame length	Analysis window length	Windows for frame	Parameter	RER _{silence} off-line classifier			RER _{silence} real-time classifier		
				NCFOM	SC	GCBC	NCFOM	SC	GCBC
1.3 s	0.093 s	14	<u>RER_{silence-speech}</u>	-	-	-	-	-	100,0%
			RER _{silence-music}	-	-	-	-	-	0,0%
			RER _{silence-noise}	-	-	-	-	-	0,0%
1.39 s	0.139 s	10	<u>RER_{silence-speech}</u>	-	-	100,0%	-	-	100,0%
			RER _{silence-music}	-	-	0,0%	-	-	0,0%
			RER _{silence-noise}	-	-	0,0%	-	-	0,0%
1.49 s	0.093 s	16	<u>RER_{silence-speech}</u>	-	-	100,0%	-	-	100,0%
			RER _{silence-music}	-	-	0,0%	-	-	0,0%
			RER _{silence-noise}	-	-	0,0%	-	-	0,0%
1.49 s	0.185 s	8	<u>RER_{silence-speech}</u>	-	-	-	-	-	100,0%
			RER _{silence-music}	-	-	-	-	-	0,0%
			RER _{silence-noise}	-	-	-	-	-	0,0%
1.85 s	0.093 s	20	<u>RER_{silence-speech}</u>	-	-	100,0%	-	-	100,0%
			RER _{silence-music}	-	-	0,0%	-	-	0,0%
			RER _{silence-noise}	-	-	0,0%	-	-	0,0%
1.85 s	0.185 s	10	<u>RER_{silence-speech}</u>	-	-	100,0%	-	-	100,0%

			$RER_{\text{silence-music}}$	-	-	0,0%	-	-	0,0%
			$RER_{\text{silence-noise}}$	-	-	0,0%	-	-	0,0%
1.85 s	0.372 s	5	<u>$RER_{\text{silence-speech}}$</u>	-	-	100,0%	-	-	100,0%
			$RER_{\text{silence-music}}$	-	-	0,0%	-	-	0,0%
			$RER_{\text{silence-noise}}$	-	-	0,0%	-	-	0,0%
1.95 s	0.139 s	14	<u>$RER_{\text{silence-speech}}$</u>	-	-	100,0%	-	-	100,0%
			$RER_{\text{silence-music}}$	-	-	0,0%	-	-	0,0%
			$RER_{\text{silence-noise}}$	-	-	0,0%	-	-	0,0%
2.23 s	0.185 s	12	<u>$RER_{\text{silence-speech}}$</u>	-	-	100,0%	-	-	100,0%
			$RER_{\text{silence-music}}$	-	-	0,0%	-	-	0,0%
			$RER_{\text{silence-noise}}$	-	-	0,0%	-	-	0,0%
2.6 s	0.185 s	14	<u>$RER_{\text{silence-speech}}$</u>	-	-	100,0%	-	-	100,0%
			$RER_{\text{silence-music}}$	-	-	0,0%	-	-	0,0%
			$RER_{\text{silence-noise}}$	-	-	0,0%	-	-	0,0%

Table 5-13: Relative Error Rate (RER) related to silence frames and evaluated for the three audio clips varying the length of the frame and of the analysis window. Highest RER for each combination is underlined.

It is possible to summarize results obtained for the various combinations linked to relative error rate stating that both classifiers – off-line and real-time – very often confuse speech and music labeling them as “noise”; on the other hand, noise is most of the times considered as “speech”. This suggests that a possible direction to improve performances of both classifiers could be adding or modifying (making them more selective) the heuristic conditions used to distinguish this audio type from the others, especially speech.

5.2.4 Normalization improvement

The fourth parameter we are going to evaluate is the so-called *normalization improvement* (Normalization Improvement – NI) due to the normalization algorithm applied over the classification results. As it is possible to see from Table 5-14, this parameter does not follow a well-defined trend varying the length of each frame and of

the analysis windows used for the extraction of the low-level features (rows of the table); anyway, the best results for this parameter are obtained using the off-line classifier using frames formed by 12 analysis windows of 0.185 seconds each, while longer analysis windows (0.372 seconds each) give back quite bad results, in one case (considering the audio clip extracted from “Good Copy Bad Copy”) even negative. Furthermore, from the same table it is possible to see that the normalization process is more effective considering the better quality audio track (“No Country For Old Men”) than the other two.

The most important thing to say about the normalization process is the fact that it makes rise both classifiers' performances in terms of classification accuracy, acting in the same way on the results produced by both the off-line and the real-time classifier (average NI value is greater than 5% in both cases).

Frame length	Analysis window length	Windows for frame	NORMALIZATION IMPROVEMENT off-line classifier			Average NORMALIZATION IMPROVEMENT
			NCFOM	SC	GCBC	
1.3 s	0.093 s	14	4,4%	6,7%	4,1%	5,1%
1.49 s	0.093 s	16	6,1%	5,0%	3,7%	4,9%
<u>1.85 s</u>	<u>0.093 s</u>	<u>20</u>	8,4%	7,8%	5,0%	<u>7,1%</u>
1.85 s	0.372 s	5	7,2%	0,6%	-0,8%	2,3%
<u>2.23 s</u>	<u>0.185 s</u>	<u>12</u>	12,8%	4,4%	6,7%	<u>8,0%</u>
			NORMALIZATION IMPROVEMENT real-time classifier			
<u>1.39 s</u>	<u>0.139 s</u>	<u>10</u>	7,7%	6,7%	5,0%	<u>6,5%</u>
1.49 s	0.185 s	8	7,2%	2,3%	6,4%	5,3%
1.85 s	0.185 s	10	4,5%	5,6%	8,3%	6,1%
1.95 s	0.139 s	14	5,5%	6,7%	0,4%	4,2%
2.6 s	0.185 s	14	13,3%	1,7%	2,1%	5,7%

Table 5-14: Normalization Improvement (NI) evaluated, for both classifiers and for each of the three analyzed audio clips, varying the length of the frames and of the analysis windows in which they are divided. The three best results obtained for each audio track are evidenced in bold, the three combinations with the highest NI are underlined

From the above considerations, it is possible to state that the normalization process is very useful for the overall classification process; as explained in paragraph 4.6.1, a

normalization algorithm has been employed that does not modify in a consistent way the obtained results, doing so only in the case there is a frame labeled as a certain audio type positioned in the middle of two frames classified as another (and common) audio type. This suggests that an algorithm modifying in a stronger way the classified frames would reasonably result in a further improvement of the performances of both classifier; this algorithm could, for example, operate on groups larger than three frames or impose conditions less strict or particular with respect to those used in the implemented classification algorithm.

5.2.5 Computational time evaluation

The last parameter we are going to evaluate is the *computational time needed by the real-time classifier to extract low-level features and to perform the classification relatively to a single frame*. The values of this parameter, that do not consider time necessary to read the samples belonging to a frame from the WAV file that stores them, show clearly that real-time classifier is, for its structural simplicity, perfectly able to fulfill its tasks in an amount of time a lot lower than the upper limit imposed by its real-time constraints. This limit, according to heuristic valuations based on the consideration that it would be necessary, in a practical case, to convert for each frame the file from its original WAV format and to read its samples, can be fixed to a third of the whole frame length. The projected real-time classifier needs on average 1/40 of the frame length to classify it (this ratio tends to grow, using shorter frames, up to a maximum of 1/32 in the case of frames of 1.3 seconds each for the audio clip “No Country For Old Men”). In table 5-15 below are listed all values of this parameter for all the three analyzed audio clips and for various combinations of the variables “frame length” and “analysis window length”.

Frame length	Analysis window length	Windows for frame	Medium Processing Time for one minute of the audio clip (real-time classifier)			Medium Frame Processing Time (real-time classifier)		
			NCFOM	SC	GCBC	NCFOM	SC	GCBC
1.3 s	0.093 s	14	1,82 s	1,56 s	1,52 s	39,2 ms	33,6 ms	32,9 ms

1.39 s	0.139 s	10	1,64 s	1,55 s	1,68 s	37,9 ms	35,8 ms	38,9 ms
1.49 s	0.093 s	16	1,59 s	1,56 s	1,55 s	39,3 ms	38,6 ms	38,3 ms
1.49 s	0.185 s	8	1,45 s	1,37 s	1,81 s	35,6 ms	33,6 ms	44,4 ms
1.85 s	0.093 s	20	1,52 s	1,52 s	1,48 s	46,9 ms	47,1 ms	45,5 ms
1.85 s	0.185 s	10	1,32 s	1,32 s	1,23 s	40,3 ms	40,6 ms	37,9 ms
1.85 s	0.372 s	5	1,44 s	1,42 s	1,36 s	44,5 ms	43,9 ms	41,8 ms
1.95 s	0.139 s	14	1,53 s	1,51 s	1,45 s	49,5 ms	48,8 ms	46,9 ms
2.23 s	0.185 s	12	1,3 s	1,26 s	1,23 s	47,6 ms	46,2 ms	45,1 ms
2.6 s	0.185 s	14	1,27 s	1,26 s	1,21 s	54,4 ms	54,2 ms	51,8 ms

Table 5-15: Computational time needed by real-time classifier to perform classification related to the whole audio track (left column) and to one of its frames (right column; medium value evaluated dividing the parameter shown in the left column for the total number of frames in which the analyzed audio track is subdivided)

Therefore, a great esteem of the projected real-time classifier lies in the fact that it is really able to return its results complying with the rules of a real-time contest, irrespective of its obtained results in terms of classification accuracy.

6 Conclusions and future works

The present document has described the process that led to the realization of two audio classifiers: the first off-line and the other able to work in real-time contests, and so useful to classify for example a video in streaming. This is not the first attempt made in history to automatically classify a video using audio features: a lot of studies and researches have been performed in the last 15 years in this sense; from a strictly applicative point of view, the using of both of them would require in most cases their inclusion in more complex systems able to segment a video with information related to the audio that accompanies it. Anyway, it seems important to underline the fact that both classifiers are able to fulfill their tasks starting from whatever file WAV even without a complete classification system incorporating them.

Both classifiers, after receiving in input a WAV file (that most of the time is the result of a format conversion from another audio or video file originally of another format), are able to segment it in parts of variable length depending on the audio type identified each time; these audio types are four: speech, music, noise and silence. To perform this task it has been necessary to extract some low-level audio features, some of them related to time domain, the majority related to frequency domain: energy function of the audio signal, Zero Crossing Rate, Root Mean Square, spectral energy distribution related to 4 frequency sub bands, Spectral Centroid and Roll-Off Point; these features (except RMS, that is not used in the real-time classifier) are all employed in both the classifiers described in the present work. Then, the control of the processing flow passes to the classification algorithm; this can be divided into two functional parts: a first part having the task to locate all silence frames in the audio clip and mark them in some way, while the second part has to distinguish all the frames that have not been classified as silence by the first one into noise, music and speech using:

- heuristic conditions applied over statistical and arithmetical features (local maximum, local minimum, variance and arithmetic mean) calculated starting from

the low-level features listed above;

- some point vectors (one for each of the audio types over which this part of the algorithm works, so three in this case), whose length is equal to the number of frames in which the whole audio clip is divided; starting from the points each audio type obtains for each frame, the classification algorithm decides what audio type has to be assigned to that frame.

The system that has been described during the present work constitutes an absolute novelty if we consider the previous works having the same objective of automatically segmenting a video clip using audio features. Furthermore, it's worth underlining the fact that (another novelty) both classifiers, while used as separate blocks with respect to the input data format conversion system, give user the possibility to choose the length of: 1) the analysis windows over which audio low-level features are extracted; 2) the frames, groups of analysis windows that constitute the unity measure at which both classifiers are able to assign a particular audio type.

Results obtained by both classifiers appear to be encouraging: classification accuracy related to off-line classifier is on average around 77%, while real-time classifier reaches an accuracy higher than 72%. These results are (especially in the off-line case) lower than those obtained by audio classifiers projected in the recent past, some of them being able to achieve a classification accuracy even higher than 90%. Such difference can be explained with the consideration that those classifiers were most of the times (all the times in the real-time case) able to discriminate only two audio types: speech and music. Both classifiers presented in this work, however, are characterized by some significant structural properties that, if developed, could bring to very good results:

1. introduction of instruments capable to locate and identify also noise frames (and not only music and speech ones) in the audio track; this consideration not only explains the worse performances of both classifiers with respect to others designed in the past years, but at the same time also opens a new road to follow for applications that, inserted in a segmentation video contest, cannot at all achieve their tasks without a good noise identification algorithm (let's think, for example, at a system able to automatically extract highlights related to a generic sport event; is

there a better way to perform this task not passing for the analysis of the crowd reactions that happen during this event?);

2. both classifiers are characterized by an extremely high scalability considering more than one point of view: it would be easy to add or subtract audio low-level features helping the classification process; to add or modify statistical and mathematical properties calculated from them or the heuristic conditions applied over these properties; to make the algorithm become more specific, making it able not only to distinguish the four audio types listed above, but also to locate more and more specific audio classes (for example, it would be possible to distinguish speech parts depending on the speaker, or on the gender or age of the speaker, and so on);

3. the most significant consideration deals with the real-time classifier only: it can return its classification results in a time that is on average 40 times lower than the length of the analyzed frame; so, this classifier could be potentially made “heavier”, adding new audio low-level features, statistical and mathematical properties and heuristic conditions to make rise its overall performances, and it would go on respecting the temporal limits imposed by a real-time contest.

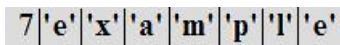
So, both classifiers described during the present work can be considered a first step for more than one research direction. In fact, as it can be noticed from all the considerations made above, they would be employed as the core-unit of bigger audio or video segmentation systems, this systems having a generic purpose or being more specific in different fields; for example, they could be specialized in recognizing various musical genres, or crowd's reactions in specific events, and then inserted in bigger commercial applications.

APPENDIXES

A. WAV format description

WAV (or *WAVE*), short for Waveform Audio Format, is a Microsoft and IBM audio file format. It is Windows' native file format for storing digital uncompressed audio data. It has become one of the most widely supported digital audio file formats on the PC due to the popularity of Windows and the huge number of programs written for this platform. Almost every modern program that can open and/or save digital audio files is able to support this file format, making it both extremely useful and a fundamental requirement for software developers to understand.

Since WAV is native for Windows and therefore for Intel processors, all data valued are stored in Little-Endian (least significant byte first order). WAV files (whose extension is *.wav*) may contain strings of text for specifying cue point labels, notes, etc. Strings are stored in a format where the first byte specifies the number of following ASCII text bytes in the string. The following bytes are the ASCII character bytes that make up the text string (see figure 4.2).



```
7|'e'|'x'|'a'|'m'|'p'|'l'|'e'
```

Figure A-1: Wave string format example

The most common WAV format (it exists a variant containing compressed audio) contains uncompressed audio in the *LPCM* format. Pulse-Code Modulation (PCM) is a digital representation of an analog signal where the magnitude of the signal is sampled regularly at uniform intervals, then quantized to a series of symbols in a numeric (usually binary) code. Linear PCM is a particular method of PCM which represents an audio waveform as a sequence of amplitude values recorded at a sequence of times. These values are directly proportional to the amplitude (for this the term “linear” is used; other methods of PCM use different relations between samples and amplitude

such as the logarithmic one).

WAV files use the standard *RIFF* (Resource Interchange File Format) structure, which groups file content into separate chunks, each containing its own header and data bytes. The chunk header specifies type and size of the chunk data bytes. This organization method allows programs that do not use (or recognize) particular types of chunk to easily skip over them and continue processing following known chunks. Certain types of chunk may contain sub-chunks. Header contains a lot of useful information about the audio file, such as its size in bytes in the *data* subchunk, that stores all data of the WAV file; sample rate, number of channels (1 or 2), bits per sample and compression code, in the *fmt* subchunk. Structure of a WAV file can be seen in figure 4.3, where are also reported for each header field its length and its offset from the beginning of the file.

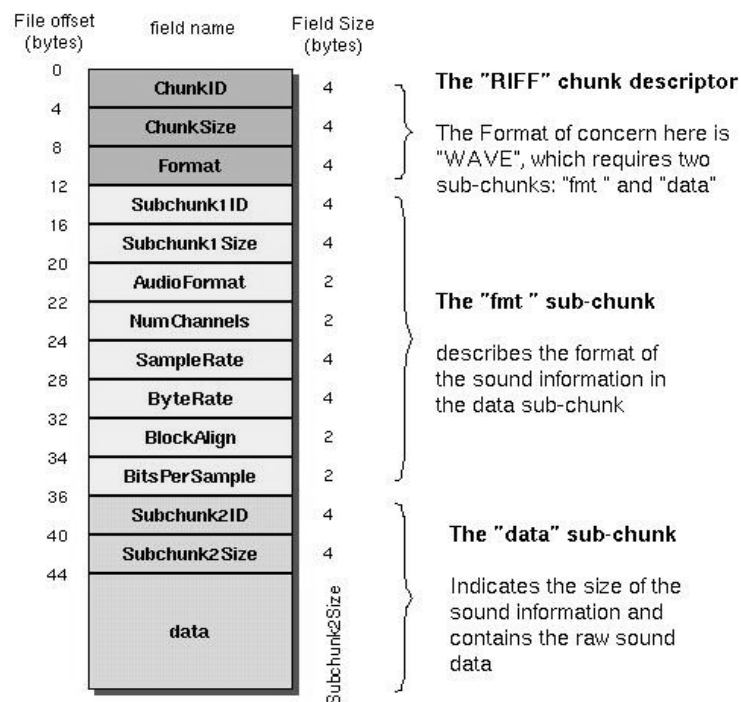


Figure A-2: The canonical WAV file format

B. MATLAB implementation

Feature name	MATLAB implementation	Notes
<u>ZCR</u>	$ZCR = (1 / (2 * WindowSamples)) * \sum(\text{abs}(\text{sign}(\text{window}) - \text{sign}(\text{window2})))$	<ul style="list-style-type: none"> •window2 is equal to window but with its values shifted of one position to the right •abs is a MATLAB function able to compute the module of a number •sign returns 1 if the argument is positive or 0, -1 if it is negative
<u>Energy</u>	$ENERGY = \sum(\text{window}.^2) / \text{EnergyDenominator}$	<ul style="list-style-type: none"> •EnergyDenominator = WindowSamples * WindowSamples is calculated once and for all at the beginning of the program to reduce required computational time
<u>Spectral Centroid</u>	$CENTROID = \text{CentroidNumerator} / \text{FFTsum}$	<ul style="list-style-type: none"> •CentroidNumerator is calculated using the following cycle: for $i = 1 : \text{length}(\text{FFT})$ CentroidNumerator = CentroidNumerator + (FrequencyBin(i) * FFT(i)) •FFTsum = sum(FFT)
<u>RMS</u>	$RMS = \sqrt{\text{FFTQuadraticSum} / \text{WindowSamples}}$	<ul style="list-style-type: none"> •sqrt is a MATLAB function that returns the square root of its argument •FFTQuadraticSum = sum(FFT.^2)
<u>N-th frequency sub band energy</u>	$\text{BandNEnergy} = \sum(\text{FFT}(\text{LowerBandNLimit} : \text{UpperBandNLimit}).^2) / \text{FFTQuadraticSum}$	<ul style="list-style-type: none"> •LowerBandNLimit and UpperBandNLimit are calculated once and for all at the beginning of the program to reduce required computational time. For example, UpperBand1Limit for N = 1 is calculated as follows: •while FrequencyBin(i) < 1000 •i = i + 1 •end •UpperBand1Limit = i
<u>Roll-off point</u>	<pre>for i = 1 : length(FFT) partialsumROLLOFF = partialsumROLLOFF + FFT(i) if partialsumROLLOFF >= 0.95 * FFTsum ROLLOFF = FrequencyBin(i) break</pre>	

Table B-1: MATLAB instructions used for the extraction of each low-level feature related to one analysis window

MATLAB implementation	Notes
<pre> for FrameIndex = 1 : NumberOfFrames for WindowIndex = 1 : FrameLength % FEATURES EXTRACTION FROM % THE WINDOW WindowIndex end % CLASSIFICATION OF THE FRAME % FrameIndex end </pre>	<p>FrameLength is the number of windows contained in a frame. It is one of the input variables of function RealTimeClassifier, so it is user that decides its value</p>

Table B-2: Structure of RealTimeClassifier.m. The “for” cycle containing the extraction of the features for each window in one frame is evidenced in bold

MATLAB implementation	Notes
<pre> for k = 1 : length(datFiles) filename = datFiles(k).name; if strncmpi('ENERGY', filename, 6) ENERGYvector = load(filename) % GROUPING DATA READ FROM DAT % FILE % APPLICATION OF HEURISTIC % CONDITIONS OVER EACH FRAME end end </pre>	<ul style="list-style-type: none"> • <i>datFiles = dir (*.dat')</i> <i>dir</i> is a standard MATLAB function returning the list of the specified files in the specified directory to an m-by-1 structure. One of the fields of this structure is <i>name</i>, containing the name of the file selected by <i>dir</i> (stored in this case in variable <i>filename</i>). • <i>strncmpi('str1', 'str2', n)</i> returns 1 if the first <i>n</i> characters of the strings <i>str1</i> and <i>str2</i> are the same except for case, and 0 otherwise. In this case this command will open the DAT file storing energy information previously computed by <i>FeaturesExtraction</i> • <i>ENERGYvector = load(filename)</i>. Standard MATLAB function <i>load</i> copies in <i>ENERGYvector</i> all data contained in file whose name is <i>filename</i>

Table B-3: Structure of OffLineClassifier.m. The “if” condition that recognizes a DAT file storing low-level information is evidenced in bold

MATLAB implementation	Notes
<pre> if strncmpi('Energy', filename, 6) while j < length(ENERGYvector) </pre>	<p><i>GroupENERGY</i> stores low-level features (in this example energy) related to a frame. So, it</p>

<pre> k = 1 for i = j : j + GROUPLEN - 1 if i <= length(ENERGYvector) groupENERGY(k) = ENERGYvector(i); k = k + 1; end end maxgroup(index) = max(groupENERGY); mingroup(index) = min(groupENERGY); averagegroup(index) = mean(groupENERGY); index = index + 1; j = j + GROUPLEN; end % CLASSIFICATION BASED ON HEURISTIC % CONDITIONS AND POINT VECTORS end </pre>	<p>is possible to compute the heuristic features related to a frame using standard functions that MATLAB offers. In the example, <i>maxgroup</i> is the energy local maximum related to a frame, <i>mingroup</i> the local minimum and <i>averagegroup</i> corresponds to the arithmetic mean.</p>
--	--

Table B-4: Structure of OffLineClassifier.m. The “while” cycle in which grouping of low-level features and evaluation of related heuristic features are performed is evidenced in bold

	<u>Silence identification algorithm</u>	<u>Music/speech/noise discrimination algorithm</u>
MATLAB implementation	<pre> if strncmpi('Energy', filename, 6) while j < length(ENERGYvector) if mean(groupENERGY) < 5 SIL_ENERGY(s) = index; s = s + 1; end end end if strncmpi('RMS', filename, 3) while j < length(ENERGYvector) if mean(groupRMS) < 0.2 SIL_RMS(s) = index; s = s + 1; end end end end </pre>	<pre> if strncmpi('Energy', filename, 6) while j < length(ENERGYvector) end for i = 1 : NUMBERGROUPS if mingroup(i) > 200 p_NOISE(i) = p_NOISE(i) + 2 p_MUSIC(i) = p_MUSIC(i) + 2 p_SPEECH(i) = p_SPEECH(i) - 2 elseif mingroup(i) > 100 p_MUSIC(i) = p_MUSIC(i) + 1 p_SPEECH(i) = p_SPEECH(i) - 1 p_NOISE(i) = p_NOISE(i) + 0.5 end end for i = 1 : NUMBERGROUPS if CLASSIFICATION_GLOBAL(i)==0 pts(1) = p_MUSIC(i); end </pre>

	<pre> for i = 1 : length(SIL_ENERGY) for j = 1 : length(SIL_RMS) if SIL_ENERGY(i)==SIL_RMS(j) CLSFS(SIL_ENERGY(i)) = 1 end end end </pre>	<pre> pts(2) = p_SPEECH(i); pts(3) = p_NOISE(i); aux = max(pts); if aux == pts(1) CLSFS(i) = 3 elseif aux == pts(2) CLSFS(i) = 4 else CLSFS(i) = 2 end end end </pre>
<p>Notes</p>	<ul style="list-style-type: none"> • <i>SIL_ENERGY</i> is a vector whose positions are filled with the indexes of each frame that satisfies silence condition related to energy; <i>SIL_RMS</i> is its equivalent for RMS • <i>CLSFS</i> is the vector storing classification results for each frame • Variables names are in some cases different from those actually used in the project 	<ul style="list-style-type: none"> • <i>NUMBERGROUPS</i> is a variable storing the number of frames in which the audio clip has been divided • <i>mingroup</i> is a vector of length <i>NUMBERGROUPS</i> storing the local minimum (related to energy in the example) for each frame of the audio clip • <i>p_MUSIC</i> is a vector of length <i>NUMBERGROUPS</i> storing in each position points gained by audio type “music” for a particular frame. The same is for <i>p_NOISE</i> with noise and <i>p_SPEECH</i> for speech • <i>pts</i> is an auxiliary vector used to store the points gained by each audio type in the i-th frame • Variables names are in some cases different from those actually used in the project

Table B-5: Structure of OffLineClassifier.m. Structure of both silence identification and music-speech-noise discrimination algorithms is evidenced in bold

MATLAB implementation	Notes
<pre> for FrameIndex = 1 : NumberOfFrames for WindowIndex = 1 : FrameLength % FEATURES EXTRACTION FROM % WINDOW WindowIndex end % CLASSIFICATION OF THE FRAME % FrameIndex end </pre>	<ul style="list-style-type: none"> • <i>NumberOfFrames</i> = <i>ceil (NumberOfWindows / FrameLength)</i> <i>ceil</i> is a standard MATLAB function that returns the biggest integer smaller than the argument of the same function • <i>NumberOfWindows</i> is calculated at the very beginning of the function as: <i>NumberOfWindows</i> = <i>ceil (L / WindowSamples)</i> Here, <i>L</i> is the length of the vector (<i>data</i>) containing the samples previously read from the file .dat

	<ul style="list-style-type: none"> • $WindowSamples = round (SampleRate * WindowLengthinSeconds)$ $round$ is a standard MATLAB function returning the integer closest to its argument and $SampleRate$ is the first number read from the file .dat containing also the waveform
--	--

Table B-6: Structure of RealTimeClassifier.m. The “for” cycle performing the classification frame by frame is evidenced in bold

MATLAB implementation	Notes
<pre> for i = 1 : NUMBERGROUPS oneahead = i + 1; onebehind = i - 1; if (aux(i) ~= aux(oneahead)) & (aux(oneahead) == aux(onebehind)) aux(i) = aux(oneahead); end end </pre>	<p><i>NUMBERGROUPS</i> is a variable representing the number of the frames in which the audio clip had been previously subdivided</p>

Table B-7: Normalization algorithm applied on an auxiliary vector (*aux*)

MATLAB implementation	Notes
<pre> for i = 1 : NUMBERGROUPS for j = j : round (i * step) CLASSIFICATION_FINAL(j) = CLASSIFICATION_GLOBAL(i); CLASSIFICATION_NORMALIZED(j) = aux(i); end j = round (i * step); end </pre>	<p>$step = DURATION / NUMBERGROUPS$, where $DURATION$ had been previously calculated as the result of the division between the total number of samples in the WAV file and the sample rate of the same file, the latter put in the first position of the DAT file storing also the waveform</p>

Table B-8: Scaling algorithm applied on classification results

REFERENCES

- [1] S. Smoliar, H. Zhang, "Content-Based video indexing and retrieval," *IEEE Multimedia Magazine*, vol. 1, no. 2, pp. 62 - 72, Summer, 1994
- [2] J. Foote, "Content-based retrieval of music and audio", *Proc.SPIE*, vol. 3229, pp. 138-147, 1997
- [3] E. Wold, T. Blum, J. Wheaton, "Content-based classification, search and retrieval of audio", *IEEE Multimedia*, vol. 3, no. 3, pp. 27-36, 1996
- [4] H. Sundaram, S.F. Chang, "Video scene segmentation using video and audio features", *IEEE International Conference on Multimedia and Expo*, vol. 2, pp. 1145-1148, Jul.30 - Aug.2, 2000
- [5] H. Jiang, T. Lin, H. Zhang, "Video segmentation with the assistance of audio content analysis", *IEEE International Conference on Multimedia and Expo*, vol. 3, pp. 1507-1510, Jul.30 - Aug.2, 2000
- [6] J.D. Broesch, D. Stranneby, W. Walker, "Digital Signal Processing: Instant access", Butterworth-Heinemann, p. 3, 2008
- [7] S.W. Smith, "The scientist and engineer's guide to digital signal processing", California Technical Pub., pp. 3-4, 1997
- [8] A. Hanjalic, L. Xu, "Affective video content representation and modeling", *IEEE Transactions on Multimedia*, vol. 7, no. 1, pp. 143-154, 2005
- [9] M. Roach, J. Mason, "Recent trends in video analysis: a taxonomy of video classification problems", *Proceedings of the International Conference on Internet and Multimedia Systems and Applications*, 2002
- [10] R. Fablet, P. Bouthemy, "Motion-based feature extraction and ascendant hierarchical classification for video indexing and retrieval", *3rd Int. Conf. on visual Information Systems, VISual'99*, Amsterdam, 1999
- [11] D. Dori, "Cognitive image retrieval", *Int conf. Pattern Recognition*, vol. 1, pp. 42-45, 2000
- [12] M. Flickner, "Query by image and video content: the QBIC system," *Computer*, vol. 28, no. 9, pp. 23-32, 1995
- [13] S-F. Chang, W. Chen, H.J. Meng, H. Sundaram, D. Zhong, "A fully automated content based video search engine supporting spatio-temporal queries", *IEEE Trans. CSVT*, vol. 8, no. 5, pp. 602-615, 1998

- [14] Y. Yongsheng, L. Ming, "A Survey on content based video retrieval", http://www.cs.ust.hk/faculty/dimitris/COMP530/video_survey.pdf
- [15] M.A. Smith, T. Kanade, "Video skimming and characterization through the combination of image and language understanding", in *Proc. IEEE International Workshop on Content-Based Access of Image and Video Database*, pp. 61–70, 1998
- [16] P. Aigrain, H. Zhang, D. Petkovic, "Content-based representation and retrieval of visual media: A state of the art review", *Multimedia Tools and Applications*, vol. 3, pp. 179–202, 1996
- [17] T. Kikukawa and S. Kawafuchi, "Development of an automatic summary editing system for the audio-visual resources", *Transactions on Electronics and Information*, vol. J75, pp. 204–212, 1992
- [18] A. Hampapur, R. Jain, T. Weymouth, "Digital video segmentation", *ACM Multimedia '94 Proceedings*, vol. A2, pp. 357–364, 1994
- [19] H. Zhang, A. Kankanhalli, S. Smoliar, "Automatic partitioning of full-motion video", *Multimedia Systems*, vol. 1, pp. 10–28, 1993
- [20] A. Akutsu, Y. Tonomura, H. Hashimoto, Y. Ohba, "Video indexing using motion vectors", in *SPIE Visual Communication and Image Processing*, vol. 1818, pp. 1522–1530, 1992
- [21] R. Zabih, J. Miller, and K. Mai, "A feature-based algorithm for detecting and classifying production effects", *Multimedia Systems*, vol. 7, pp. 119–128, 1999
- [22] A. M. Alattar, "Detecting and compressing dissolve regions in video sequences with a dvi multimedia image compression algorithm", in *IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 13–16, 1993
- [23] B. Günsel, A. M. Tekalp, "Content-based video abstraction", *IEEE International Conference on Image Processing*, vol. 3, pp. 128–132, 1998
- [24] H. Zhang, J. Wu, D. Zhong, S. Smoliar, "An integrated system for content-based video retrieval and browsing", *Pattern Recognition*, vol. 30, pp. 643–658, 1997
- [25] J. Vermaak, P. Peraz, M. Gangnet, A. Blake, "Rapid summarisation and browsing of video sequences", *British Machine Vision Conference*, vol. 1, pp. 424–433, 2002
- [26] W. Wolf, "Key frame selection by motion analysis", *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, pp. 1228–1231, 1996
- [27] D. Pye, N. Hollinghurst, T. Mills, K. Wood, "Audiovisual segmentation for content based retrieval", *International conference on spoken language processing (ICSLP 98)*, Sydney, 1998

- [28]J.M.Gauch, S.Gauch, S.Bouix, X.Zhu, “Real time video scene detection and classification”, *Information Processing and Management*, vol. 35, pp. 401–420, 1999
- [29]S.Pfeiffer, S.Fischer, W.Effelsberg, “Automatic audio content analysis”, *Proc. 4th ACM Int. Conf. Multimedia*, pp. 21–30, 1996
- [30]D.Kimber, L.Wilcox, “Acoustic segmentation for audio browsers”, *Proc. Interface Conf.*, Sydney, 1996
- [31]S.Srinivasan, D.Petkovic, D.Ponceleon, “Toward robust features for classifying audio in the CueVideo system”, *Proc. 7th ACM Int. Conf. Multimedia*, pp. 393–400, 1999
- [32]S.Chen, P.Gopalakrishnan, “Speaker, environment and channel change detection and clustering via the bayesian information criterion”, *DARPA Proc. Speech Recognition Workshop*, 1998
- [33]C.H.Wu, C.H.Hsieh, “Multiple change-point audio segmentation and classification using an mdl-based gaussian model”, *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, pp. 647–657, 2006
- [34]H.Harb, L.Chen, J.Auloge, “Speech/music/silence and gender detection algorithm”, *Proceedings of the 7th International conference on Distributed Multimedia Systems DMS01*, 2001
- [35]M.Seigler, U.Jain, B.Raj, R.Stern, “Automatic segmentation, classification, and clustering of Broadcast news audio”, *Proc. of the DARPA speech recognition workshop*, 1997
- [36]F.Kubala, H.Jin, S.Matsoukas, L.Nguyen, R.Schwartz, J.Makhoul, “The 1996 Bbn Byblos Hub-4 Transcription System”, 1996
- [37]J.Saunders, “Real-time discrimination of broadcast speech/music”, *Proc. ICASSP’96*, vol. II, Atlanta, pp. 993–996, 1996
- [38]E.Scheirer, M.Slaney, “Construction and evaluation of a robust multifeature music/speech discriminator”, *Proc. ICASSP’97*, vol. II, pp. 1331–1334, 1997
- [39]T.Zhang, C.J.Kuo, “Audio content analysis for online audiovisual data segmentation and classification”, *IEEE Transactions on Speech and Audio Processing*, vol. 9, n. 4, pp. 441–457, 2001
- [40]C.Panagiotakis, G.Tziritas, “A speech/music discriminator based on RMS and zero-crossings”, *IEEE Transactions on Multimedia*, vol. 7, pp. 155–166, 2005
- [41]J.Ajmera, I.McCowan, H.Bourlard, “Speech/music segmentation using entropy and dynamism features in a HMM classification framework”, *Speech Communication*, vol. 40, pp. 351–363, 2003

- [42]N.Casagrande, D.Eck, B.Kigl., “Frame-level audio feature extraction using AdaBoost”, *Proc. of ISMIR 2005*, pp. 345-350, London, 2005
- [43]J.E.Muñoz-Exposito, S.Garcia-Galán, N.Ruiz-Reyes, P.Vera-Candeas and F. Rivas-Peña, “Speech/Music discrimination using a single Warped LPC-based feature”, *Proc. of ISMIR 2005*, pp. 614-617, London, 2005
- [44]A.Pikratis, T.Giannakopoulos, S.Theodoridis, “A speech/music discriminator of radio recordings based on dynamic Programming and bayesian networks”, *IEEE Transactions on Multimedia*, vol. 10 (5), pp. 846-857, 2008
- [45]J.Huang, Z.Liu, Y.Wang, “Integration of audio and visual information for content-based video segmentation”, *Proc. IEEE Conf. Image Processing*, 1998
- [46]M.R.Naphade, T.Kristjansson, B.Frey, “Probabilistic multimedia objects (MULTI-JECTS): a novel approach to video indexing and retrieval in multimedia systems”, *Proc. IEEE Conf. Image Processing*, Chicago, 1998
- [47]J.S.Boreczky L.D.Wilcox, “A hidden Markov model framework for video segmentation using audio and image features”, *Proc. Int. Conf. Acoustics, Speech, Signal Processing '98*, pp. 3741–3744, 1998
- [48]L.Gu, H.Jiang, H.Zhang, W.Qi, X.Chen, “Integrating visual, audio and text analysis for news video”, 2000
- [49]W.Cooley, J.W.Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, pp. 297–301, 196

GLOSSARY

AAC	Advanced Audio Coding (filename extension .aac) – lossy compression audio format designed to be the successor of MP3. It generally achieves better sound quality than MP3 at many bit rates
ADC	Analog to Digital Converter
AIFF	Audio Interchange File Format (filename extension .aiff) – uncompressed audio format commonly used by Apple PCs (Macintosh)
ALAC	Apple Lossless Audio Codec (filename extension .m4a) – lossless compression audio format owned by Apple Inc. and used on Macintosh
ANN	Artificial Neural Network – computational model that consists of an interconnected group of artificial neurons and processes information in an adaptive way
ATRAC	Adaptive TRansform Acoustic Coding (filename extension .aa3) – lossy compression audio format developed by Sony. It exists also a lossless version: <i>ATRAC Advanced Lossless</i>
ASR	Automatic Speech Recognizer
DAC	Digital to Analog Converter
DFT	Discrete Fourier Transform
DSP	Digital Signal Processing
FFT	Fast Fourier Transform
FLAC	Free Lossless Audio Codec (filename extension .flac) – lossless compression audio format
GMM	Gaussian Mixture Model – probabilistic model for density estimation using a Gaussian distribution
HMM	Hidden Markov Model – statistical model in which the system being modelled is assumed to be a <i>Markov process</i> (the likelihood of a given future state, at any given moment, depends only on its present state, and not on any past state) with unobserved state
JPEG	Joint Photographic Experts Group – committee and image lossy compression method introduced by the same committee (filename extension .jpg). It is the most common image format used by digital

	cameras and other photographic image capture devices and the most common format for storing and transmitting photographic images on the World Wide Web
KNN	K-Nearest Neighbors – algorithm that classifies objects based on closest training examples in the feature space; type of instance-based learning
I/O	Input/Output
LPCM	Linear Pulse Code Modulation
MFCCs	Mel-Frequency Cepstrum Coefficients
MP3	MPEG-1 Audio Layer 3 (filename extension .mp3) – digital audio encoding format using a lossy compression algorithm. <i>De facto</i> standard of audio compression for the transfer and playback of music on digital audio players
MPEG	Moving Picture Experts Group – organization that sets standards for video and audio compression
NLP	Natural Language Processing – field of computer science concerned with the conversion of information from computer databases into readable human language and vice versa
PC	Personal Computer
PCM	Pulse Code Modulation
RIFF	Resource Interchange File Format
RMS	Root Mean Square
RP	Roll-off Point
SC	Spectral Centroid
SNR	Signal-Noise Ratio
STE	Short Time Energy
WAV	Waveform Audio Format
WLPC	Warped Linear-Predictive Coding
WMA	Windows Media Audio (filename extension .wma) – Microsoft lossy audio data compression technology. It exists also a lossless version: <i>WMA Lossless</i>
ZCR	Zero Crossing Rate

ANALISIS EN VIVO DE LA BANDA SONORA

Resumen

La realización práctica del proyecto aquí descrito se llevó a cabo en la Facultad de Telecomunicaciones de la Escuela Politécnica Superior (Universidad Autónoma de Madrid), bajo la supervisión del profesor José María Martínez. Este trabajo busca una solución al problema de la segmentación automática de un vídeo utilizando informaciones extraídas de la banda audio que le acompaña. En este contexto, encuentran su lugar los dos clasificadores de audio que serán presentados durante el trabajo de tesis: el primero está diseñado para trabajar en un ambiente donde el tiempo de procesamiento necesario para realizar la clasificación tiene que respetar las limitaciones impuestas por un contexto de tiempo real, mientras que el segundo puede hacerlo en un contexto off-line. Otro de los objetivos del presente proyecto es identificar los requisitos (en términos de velocidad, complejidad computacional y rendimiento) que necesariamente diferencian a los dos clasificadores dependiendo del ámbito en el cual trabajan.

Los dos clasificadores comienzan de un fichero WAV (obtenido convirtiendo el formato del audio o del video que se quiere analizar) para dividir el clip en fragmentos dependientes del tipo audio identificado por cada cuadro. Al final del proceso de clasificación, el clip está dividido en fragmentos pertenecientes a 4 tipos de audio diferentes: voz, música, ruido y silencio. El algoritmo de clasificación se divide en dos partes: la primera diputada a la identificación de los cuadros de silencio, mientras que la segunda parte distingue a aquellos que no han sido clasificados como silencios en música, sonido y voz. Ambas estas partes utilizan una serie de condiciones heurísticas aplicadas a algunas medidas estadísticas (varianza, media aritmética, máximo y mínimo local) determinadas a su vez por las propiedades de audio de bajo nivel energía temporal de la señal de audio, la tasa de cruces por zero (ZCR), la distribución espectral de energía en 4 bandas de frecuencia, el valor cuadrático medio de la señal (RMS), el centroide espectral y el punto de rolloff. Además, solamente en la segunda parte del

algoritmo, para cada cuadro se asignan puntos (almacenados en un vector) a cada clase de audio cada vez que una condición es verificada .

Los resultados de ambos clasificadores son satisfactorios, con un promedio de precisión de la clasificación de 77% para el clasificador off-line y del 72% para el real-time, que además cumple los vínculos temporales; este resultado es muy prometedor en perspectiva de eventuales proyectos futuros que siguen el mismo patrón de clasificación propuesto en este trabajo de tesis.

Introducción

1. Motivaciones y objetivos

En un periodo histórico dominado por Internet y los datos multimedia, en el cual más y más personas se acercan y utilizan con cada vez mayor competencia tecnologías digitales sofisticadas, se hace necesario introducir herramientas nuevas y innovadoras que permitan manejar con facilidad y eficiencia esta enorme cantidad de información en circulación. Desde la difusión de los ordenadores personales hasta tecnologías introducidas más recientemente (como el Blackberry o el teléfono móvil), la tendencia en esta dirección parece ser la de tratar de unificar el flujo de información de diversos tipos (sobre todo textuales, audio y vídeo) en un único flujo de información multimedia. En esta situación en rápida evolución, análisis y clasificación automática de secuencias de vídeo son tareas necesarias para un gran número de aplicaciones, tales como las dedicadas a la indexación y búsqueda de video dentro de las bases de datos multimedia, así como las especializadas en edición de vídeo, y muchas otras. La investigación en este ámbito se ha centrado principalmente en la utilización de la información visual; por ejemplo, una de las técnicas utilizadas se basa en el uso de las estadísticas derivadas de las imágenes (histograma de color, descriptores de textura o de forma), que luego se utilizan para describir el contenido de los cuadros.

Solamente en los últimos años, varios grupos de investigación han comenzado a centrar su atención en el análisis de la banda audio, con el propósito de segmentar la

secuencia vídeo. Los resultados de esta nueva línea de investigación parecen prometedores. En un primer nivel semántico muy general, el principio en que se basa este audio análisis se puede aclarar con el siguiente ejemplo: el sonido que acompaña a un vídeo en un evento deportivo será seguramente muy diferente de lo que se asocia con un noticiero. Además, diferentes generos deportivos se caracterizan a menudo por los ruidos de fondo muy diferentes. Así pues, un buen algoritmo basado en un análisis cuidadoso de la banda de audio puede ser capaz de permitir la búsqueda (por ejemplo, dentro de una base de datos de vídeo) de todos los vídeos relacionados con un partido de baloncesto o de fútbol. Siempre hablando de la semántica de un vídeo es posible, en principio, buscar todas las partes de un cualquier vídeo que podrían ser de interés para un usuario genérico; por ejemplo, en un partido de fútbol, todos los goles o las acciones importantes. Esta tarea puede ser realizada usando un buen sistema de clasificación de audio que sea capaz de buscar dentro de un vídeo todas aquellas secciones con algunas específicas propiedades acústicas; estas propiedades tienen que estar de alguna manera conectadas a un fragmento que sea de particular interés para ese video y no otros. Por ejemplo, el público que grita en un estadio puede ser considerado por el clasificador como símbolo de un recurso importante para la economía del juego o incluso un gol.

Otra línea de investigación que se ha abierto en los últimos años se basa en un análisis que involucra tanto a las imágenes en movimiento cuanto al sonido que las acompaña; este método intenta aprovechar las ventajas de ambos tipos de análisis para mejorar aún más la calidad de la clasificación automática. Por esta razón, varios modelos que se han propuesto integran la utilización de propiedades de bajo nivel tanto de audio que de vídeo.

El alcance del proyecto presentado en este documento se limita exclusivamente al análisis de la banda de audio. Su objetivo principal es el de obtener resultados que sean utilizables en un sistema de alto nivel, capaz de realizar la búsqueda y la indexación del material de video contenido en una base de datos. Estos resultados consisten esencialmente en la clasificación de la banda de audio, de acuerdo con los valores de sus propiedades acústicas de bajo nivel, en 4 tipos de audio diferentes: voz, música, ruido y silencio. Este trabajo de clasificación se llevará a cabo en dos contextos diferentes: off-line y real time. La hipótesis que subyace a este proyecto es la siguiente: la clasificación

en tiempo real es sin duda destinada a ser menos sofisticada que la off-line, esto sucede debido a las limitaciones temporales impuestas por el contexto; el punto real de esta cuestión consiste en ver como el contexto de trabajo en tiempo real puede limitar la selectividad del algoritmo utilizado para realizar la clasificación de la banda de audio.

Otro de los objetivos que el presente trabajo se propone satisfacer es contribuir al avance de los métodos de segmentación de audio tanto respecto a la complejidad computacional tanto a la velocidad de ejecución del algoritmo utilizado, especialmente en el caso del clasificador que trabaja en tiempo real. La mayoría de los algoritmos utilizados en los proyectos realizados en el pasado en este ámbito obtienen los mejores resultados solamente en los casos de clasificación automática más simples, es decir, los algoritmos que tienen como objetivo identificar en la banda de audio solamente las clases de “voz” y “música”. Otra contribución que este proyecto quiere aportar – en el progreso de las técnicas de clasificación automática de un vídeo analizando solamente la banda de audio que le acompaña – consiste en un esquema de clasificación de la señal de audio adaptable a diferentes duraciones tanto de los cuadros como de la ventana de análisis, buscando un compromiso entre la precisión del análisis y la calidad de la clasificación.

Es evidente como un sistema de clasificación de buena calidad requiere una buena base de datos con la cual pueda desempeñar sus funciones; en el ámbito de la clasificación de audio, estos datos son casi siempre un conjunto más o menos grande de propiedades de la señal de audio a lo largo del tiempo o en el dominio de la frecuencia. Este concepto es válido tanto en el caso off-line como en el de real-time. En este proyecto se utilizarán las siguientes propiedades de audio de bajo nivel para el clasificador que trabaja fuera de conexión: la función que expresa como varía la energía de la señal de audio a través del tiempo, la tasa de cruces por cero (ZCR), la distribución espectral de energía en 4 bandas de frecuencia, el valor cuadrático medio de la señal (RMS), el centroide espectral y el punto de rolloff. El clasificador en tiempo real utiliza todas las propiedades mencionadas anteriormente, excepto el Root Mean Square.

La extracción de las propiedades de audio de bajo nivel es el primer paso fundamental para la realización de un buen clasificador de audio; el segundo paso consiste en un algoritmo que, a partir de tales propiedades, sea capaz en un primer momento de

reconocer los cuadros de silencio, y luego de clasificar correctamente las partes restantes en música, voz y ruido.

2. Estructura de trabajo

El resto del documento está organizado en capítulos; en cada uno se analizará un pasaje en el flujo de trabajo que ha caracterizado a la realización de los dos clasificadores audio. La siguiente lista consta de una breve presentación para cada uno de los capítulos que componen el presente documento.

- En el capítulo 2 ("Estado del arte") se proporciona la historia del análisis de audio digital. Este capítulo comienza con un apartado que analiza las etapas fundamentales en la historia del procesamiento de señales digitales (DSP) y, a continuación, se centra en los últimos años, durante los cuales muchos modelos han sido propuestos por varios grupos de investigación para segmentar un vídeo utilizando informaciones extraídas de las imágenes en movimiento o de la banda de audio, o ambas, y para clasificar de alguna manera los fragmentos así obtenidos.
- El capítulo 3 ("Diseño") comienza con una breve descripción de las características básicas de cada clase de audio - voz, música, ruido y silencio - que los dos clasificadores quieren identificar en cada clip analizado. Luego continúa con una descripción detallada de todas las propiedades de bajo nivel extraídas de la banda de audio, y las relaciona con la información proporcionada para discriminar entre las cuatro clases audio; estas propiedades se utilizan durante la posterior fase de clasificación. El capítulo termina con un breve análisis de la idea principal que será la base para la realización del algoritmo de clasificación.
- En el capítulo 4 ("Desarrollo") se describen y explican en detalle todos los pasos que llevaron al desarrollo de los dos clasificadores de audio, en aplicación de las ideas producidas durante la fase de diseño, con una atención especial a la descripción de los diferentes algoritmos implementados.
- Los resultados y la evaluación del trabajo de ambos clasificadores con un análisis de sus cualidades positivas y sus puntos críticos, se presentan en el capítulo 5 ("Integración, pruebas y resultados").

- En el último capítulo ("Conclusiones") se hace un balance conclusivo y se asumen las posibles líneas de investigación a seguir en la misma dirección para el trabajo futuro.

Conclusiones

En el presente documento se ha descrito el proceso que condujo a la creación de dos clasificadores audio: el primero off-line y el segundo capaz de trabajar en tiempo real (este último conveniente para ambientes como los vídeos en streaming), cada vez más comunes y populares en el mundo del Internet. Estos clasificadores son parte de una cadena de estudios y de investigación que se comenzó hace más de 15 años; en una perspectiva aplicativa, su utilización práctica requeriría, en la mayoría de los casos, la inclusión de uno de los dos clasificadores en sistemas más complejos, que puedan segmentar un vídeo a partir de la información obtenida de la banda de audio que lo acompaña. Merece la pena subrayar como los dos clasificadores realizados sean capaces de trabajar a partir de cualquier fichero audio de formato WAV.

Ambos clasificadores, recibido en entrada un fichero WAV (que a su vez se puede haber extraído de un fichero audio o vídeo de cualquier otro formato), son capaces de dividirlo en partes de longitud variable en función del tipo audio cada vez identificado; los tipos reconocidos son cuatro: voz, música, ruido y silencio. Para llevar a cabo el proceso de clasificación ha sido necesario extraer algunas propiedades de audio de bajo nivel, algunas relacionadas con el dominio del tiempo, la mayoría con el de la frecuencia: energía, ZCR, RMS, distribución de energía espectral en 4 bandas de frecuencia, centroide espectral y punto de rolloff. El algoritmo de clasificación de audio realizado se puede dividir en dos partes funcionales: la primera identifica los fragmentos de silencio en el clip de audio, la segunda distingue los cuadros restantes en voz, música y ruido utilizando:

- una serie de condiciones heurísticas aplicadas sobre herramientas estadísticas (máximo y mínimo local, varianza y media aritmética) obtenidas a su vez a partir

de las propiedades de bajo nivel antes mencionadas; y

- tres vectores a puntos (uno para cada tipo de audio que esta parte del algoritmo mira a identificar), que forman la base de datos que el algoritmo de clasificación utiliza para elegir a qué tipo asignar cada cuadro en el clip de audio en consideración.

El sistema implementado no tiene precedentes, teniendo en cuenta los previos trabajos con el mismo propósito. Hay que destacar también que los dos clasificadores dan al usuario la posibilidad de elegir la longitud de: 1) la ventana de análisis con respecto a la que se calculan las propiedades de audio de bajo nivel; y 2) el número de ventanas de análisis por cuadro; el cuadro constituye la unidad temporal sobre la cual ambos clasificadores evalúan el audio tipo.

Los resultados obtenidos por los dos clasificadores son alentadores: el clasificador off-line obtiene una precisión media de clasificación alrededor del 77%, mientras que el clasificador real-time obtiene un resultado un poco más modesto, con una precisión ligeramente superior al 72%. Estos resultados son (sobre todo en el caso off-line) “peores” que los obtenidos por clasificadores de audio realizados en trabajos anteriores (capaces, en algunos casos, de lograr una precisión superior al 90%), pero esta diferencia se explica con el hecho de que la mayoría de los clasificadores proyectados en el pasado (especialmente en el caso real-time) pueden casi siempre discriminar solamente dos clases audio: voz y música. Por otra parte, los dos clasificadores realizados muestran, sin embargo, algunas características estructurales significativas que, si se sigue investigando en esta dirección, podrían dar resultados satisfactorios:

1. Introducción de herramientas con las cuales es posible identificar fragmentos de ruido (además de voz y música) en la banda de audio; esta consideración explica la disminución en el rendimiento general de los dos clasificadores de audio respecto a otros proyectos similares, pero al mismo tiempo abre la puerta a aplicaciones nuevas que, en un contexto de segmentación de vídeo, no pueden prescindir de un buen sistema de identificación del ruido. Por ejemplo, ¿qué mejor manera de lograr en un sistema que tiene la tarea de extraer los aspectos más destacados de una secuencia de vídeo perteneciente a un cualquiera evento deportivo, si no estudiando los gritos, aplausos y silbidos de la multitud que le

apoya personalmente?

2. Ambos algoritmos tienen una buena escalabilidad: se puede fácilmente agregar o quitar propiedades de audio de bajo nivel o añadir a propiedades estadísticas extraídas de estas o a las condiciones heurísticas impuestas a estos datos. Así, sería fácilmente posible hacer el algoritmo más específico, ósea capaz no solamente de distinguir en la banda de audio los cuatro tipos antes mencionados, sino también un mayor número de clases (tales como la subdivisión de las partes habladas en función del sexo, o de la edad del hablante);
3. El dato más importante se refiere al clasificador que trabaja en tiempo real, capaz de devolver los resultados de la clasificación para cada cuadro en un tiempo medio igual aproximadamente a $1/40$ de su duración; por lo tanto este clasificador podría potencialmente ser "cargado" por nuevas propiedades de audio de bajo nivel, herramientas estadísticas y condiciones heurísticas, sin dejar de respetar las limitaciones de tiempo impuestas por un contexto de trabajo en tiempo real.

Los dos clasificadores descritos en el presente documento pueden legítimamente considerarse el primer paso para varias líneas de investigación. Como se desprende de las consideraciones que se acaban de hacer, de hecho, pueden ser explotados como la unidad central de eventuales sistemas de segmentación (audio o video) genéricos, o ampliados a lo largo de una o más direcciones. Por ejemplo, podrían ser especializados en el reconocimiento de diferentes géneros musicales o de las reacciones de la multitud que asiste a un determinado tipo de evento, para luego ser colocados en aplicaciones comerciales.

PRESUPUESTO

1) Ejecución Material

- Compra de ordenador personal (Software incluido)..... 1.300 €
- Alquiler de impresora láser durante 6 meses..... 50 €
- Material de oficina..... 150 €
- Total de ejecución material..... 1.500 €

2) Gastos generales

- 16 % sobre Ejecución Material..... 240 €

3) Beneficio Industrial

- 6 % sobre Ejecución Material..... 90 €

4) Honorarios proyecto

- 640 horas a 15 € / hora 9600 €

5) Material fungible

- Gastos de impresión..... 30 €
- Encuadernación..... 50 €

6) Subtotal del presupuesto

- Subtotal Presupuesto..... 11510 €

7) I.V.A. aplicable

- 16% Subtotal Presupuesto.....1841,6 €

8) Total presupuesto

- Total Presupuesto..... 13351,6 €

Madrid, Diciembre de 2009

El Ingeniero Jefe de Proyecto

Fdo.: Francesco Alfeo

Ingenier Superior de Cine y Media

PLIEGO DE CONDICIONES

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de contribuciones a técnicas de segmentación de video utilizando la banda sonora. En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.

Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor económico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.
2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.
3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.
4. La obra se realizará bajo la dirección técnica de un Ingeniero Superior de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.
5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.
6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometidos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea beneficiosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y director de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y administración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4% del presupuesto y la provisional del 2%.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entenderá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpreta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denominado en la actualidad "Presupuesto de Ejecución de Contrata" y anteriormente llamado "Presupuesto de Ejecución Material" que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presente trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.
2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.
3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.
4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.
5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.
6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.
7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.
8. Si la modificación no es aceptada, por el contrario, la empresa consultora declinará toda

responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime oportuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.