Departament de Teoria
del Senyal i Comunicacions de Terrassa

An Automatic Audio Segmentation System for Radio Newscast

Final Project

ADVISOR

Professor Ignasi Esquerra

STUDENT

Vincenzo Dimattia

March 2008

Preface

The work presented in this thesis has been carried out at the Department of Signal Theory and Communications (TSC) at the UPC Terrassa. The thesis is the final requirement for obtaining the Master degree in Telecommunication Engineering. The work has been supervised by the professor Ignasi Esquerra.

CONTENTS

Abstract vi

CHAPTER 1: INTRODUCTION 1

1.1 Audio Retrieval Systems 1

1.1.1 Automatic Broadcast News Transcription 2

1.2 Segmentation Approaches 2

1.2.1 Audio Classification 2

1.2.1 Speaker Change Detection 3

1.3 Audio feature extraxtion 3

1.4 Mpeg-7 Audio Descriptors 4

1.5 CLAM 5

1.6 XML 6

1.7 Project Objective 7

1.8 Project Overview 7

CHAPTER 2: AUDIO FEATURE EXTRACTION USING CLAM 8

2.1 The Project File 8

2.2 The Schema File 9

2.3 High-level descriptors 11

2.4 Low-level Descriptors 11

2.5 Descriptors Pool File 11

2.6 High-level Descriptors 12

2.7 Low-Level Descriptors 12

2.8 Showcase 15

2.9 Loading a Project 15

2.10 The Schema and the dynamic GUI 15

2.11 Viewing Song Properties 16

2.12 Editing Low-level Descriptors 16

2.13 Editing High-level Descriptors 17

2.14 Viewing Associated Schema 18

CHAPTER 3:SPECTRAL DESCRIPTOR EXTRACTION WITH `ClamExtractorExample` 19

3.1 Mean 19

3.2 Geometric Mean 20
3.3 Energy 22
3.4 Centroid 23
3.5 Flatness 25
3.6 Magnitude Kurtosis 27
3.7 Low Freq Energy Relation 28
3.8 MaxMagFreq 30
3.9 Spread 30
3.10 Magnitude Skewness 32
3.11 Roll off 33
3.12 Spectral Slope 34
3.13 High Frequency Content 36
3.14 Cepstrum 37
3.14.1 Mel Frequency Cepstral Coefficient 39

CHAPTER 4: AUDIO DATABASE AND MANUAL SEGMENTATION 43
4.1 WaveSurfer Tool for the Manual Segmentation 44
4.2 Data Base Description 46
4.3 MPEG-7 Multimedia Description Schemes 49
4.4 Organization of MDS tools 49
4.4.1 Content Description 50
4.4.2 Basic Elements 51
4.5 Automatic generation of an XML Document for the description of the Segments 52
4.6 Example of a XML Document for the Description of the Segments 56

CHAPTER 5: AUTOMATIC AUDIO SEGMENTATION 59
5.1 Feature extraction 59
5.2 Model-Based Segmentation 61
5.3 Metric-Based Segmentation 61
5.4 Hybrid Segmentation 62
5.5 Decoder-Guided Segmentation 62
5.6 Model-Selection-Based Segmentation 62
5.7 Model Selection Criteria 63
5.8 Change Detection via BIC 64

5.9 Detecting Multiple Changing Points Spread 67

CHAPTER 6: EXPERIMENTS AND RESULTS 68

6.1 Evaluation Method 69

6.2 Alternative Measure 69

6.3 Segmentation System 70

6.4 First Experiment 71

6.5 Second Experiment 77

6.6 Third Experiment 79

6.7 Conclusion 82

CHAPTER 7: CONCLUSION 84

7.1 Future Works 85

REFERENCES 86

Abstract

Current web search engines generally do not enable searches into audio files. Informative metadata would allow searches into audio files, but producing such metadata is a tedious manual task. Tools for automatic production of metadata are therefore needed. This project describes the work done on the development of an automatic audio segmentation system which can be used for this metadata extraction. In this work the radio newscast are divided into segments in which there is only one speaker. Audio features used in this project include Mel Frequency Cepstral Coefficients. This feature was extracted from audio files that were stored in a WAV format, using CLAM. Model-Selection-Based segmentation is used to segment audio signals using this feature.

In order to evaluate and improve the performance of the segmentation system a manual segmentation is performed and two different evaluation metrics was computed implementing a matlab code.

The segments obtained by the manual segmentation were then described in an MPEG-7 compliant XML document. While the segments of the automatic segmentation could be put in input to the classification system implemented by Giuseppe Dimattia in the same laboratory in his thesis [71].

CHAPTER 1
INTRODUCTION

The amount of audio available in different databases on the Internet today is immense. Successful access to such large amounts of data requires efficient search engines. Traditional web search engines, such as Google, are often limited to text and image indexing, thus many multimedia documents, video and audio, are excluded from these classical retrieval systems. Even systems that do allow searches for multimedia content, like AltaVista and Lycos, only allow queries based on the multimedia filename, nearby text on the web page containing the file, and metadata embedded in the file such as title and author. This might yield some useful results if the metadata provided by the distributor is extensive. Producing this data is a tedious manual task, and therefore automatic means for creating this information is needed.

Today many radio stations provide entire news or talk shows in form of podcasts or streaming services, with general headlines of the content. In general no detailed index of the file is provided, which makes it time consuming to look for the part of interest.

The optimal division of radio shows, such as broadcast news, debate programs, and music programs, should be based on the topics covered in each part. Such a division requires that it is possible extract topics and the parts related to this topic. Topic detection requires transcription of the speech parts of the audio, but adding audio cues would aid in retrieving coherent segments.

Adding audio cues is done by segmenting the audio based on the characteristics of the audio stream. The segments generated can then be summarized on basis of the type of audio.

- Music clips would be described by genre and artist.
- Speech summaries naturally and would consist of identities of speakers and a transcription of what is said.

1.1 Audio Retrieval Systems

Research in audio retrieval has mainly been focused on music and speech. Music retrieval has focused on quantifying different characteristics of the music such as mood, beat, and other characteristics to classify genre or find similarities between songs. This area is not the focus of this thesis and will not be covered further.

Approaches to spoken document retrieval have included automatic broadcast news transcription and other speech retrieval systems.

1.1.1 Automatic Broadcast News Transcription

Broadcast news transcription system has been heavily researched and is considered to be the most demanding assignment in speech recognition, as the speakers and conditions vary much in the course of a news show. The speakers range from anchor speaking in an ideal environment to reporters speaking from noisy environments on a non-ideal telephone line. Non-native English speakers make the problem even more challenging. The shows often use music between different parts of the show and in the background of speakers.

These systems therefore often include segmentation preprocessor that removes the non-speech parts and finds segments with homogenous acoustical characteristics. In this way the speech recognizers can be adjusted to the differing acoustical environments. Solving the task has required advances in both preprocessing and speech decoding.

The performances of these systems are evaluated in annual Rich Transcription workshops arranged by [1]. The participants include commercial groups such as IBM and
BBN and academical groups such as LIMSI and CU-HTK, whose systems are described in [2] and [3], respectively.

1.2 Segmentation Approaches

The segmentation approaches used in the systems mentioned above and in other retrieval systems covers a wide range of methods. In general the methods can be divided into two groups: audio classification and change detection. These approaches have different attributes that qualify them for different uses as presented below.

1.2.1 Audio Classification

As mentioned above the typical first part of a speech retrieval system concerns identifying different audio classes. The four main classes considered are speech, music, noise, and silence but depending on the application more specific classes such as noisy speech, speech over music, and different classes of noise, have been considered.

The task of segmenting or classifying audio into different classes has been implemented using a number of different schemes. Following the paper by [4] a multitude of approaches have

been proposed. Two aspects that must be considered are feature and classification model selection.

Different features have been proposed, based on different observations on the characteristics that separate speech, music and other possible classes of audio. The features are generally divided on basis of the time horizon they are extracted.

The simplest features proposed include time domain and spectral features. Time domain features typically represent a measure of the energy or zero crossing counts.

Cepstral coefficients have been used with great success in speech recognition systems, and subsequently have shown to be quite successful in audio classification tasks as well [2]. Other features have also been proposed based on psychoacoustic observations, see e.g., [5].

The other aspect to be considered is the classification scheme to use. A number of classification approaches have been proposed, that can be divided into rule-based and model-based schemes. The rule-based approaches use some simple rules deducted from the properties of the features. As these methods depend on thresholds, they are not very robust to changing conditions, but may be feasible for real-time implementations.

Model-based approaches have included Maximum A Posteriori (MAP) classifiers, Gaussian Mixture Model (GMM), K-nearest-neighbor (K-NN), and linear perceptrons. Another approach in this context is to model the time sequence of features, or the probability of switching between different classes. Hidden Markov Models (HMM) take this into account.

1.2.2 Speaker Change Detection

Another approach to identify homogenous audio segments could be done by performing event detection. Approaches to speaker change detection can be divided into supervised and unsupervised methods. If the number of speakers and identities are known in advance, supervised models for each speaker can be trained, and the audio stream can be classified accordingly. If the identities of the speakers are not known in advance unsupervised methods must be employed.

I speak about the speaker change detection in detail in the 5^{th} chapter.

1.3 Audio Feature Extraction

Feature extraction is the process of converting an audio signal into a sequence of feature vectors carrying characteristic information about the signal. These vectors are used as basis

for various types of audio analysis algorithms. It is typical for audio analysis algorithms to be based on features computed on a window basis. These window based features can be considered as short time description of the signal for that particular moment in time.

Feature extraction is a very important issue to get optimal results in this application. Extracting the right information from the audio increases the performance of the system and decrease the complexity of subsequent algorithms.

Generally applications require different features enhancing the characteristics of the problem.

A wide range of audio features exist for classification tasks. These features can be divided into two categories: time domain and frequency domain features.

In the frequency domain, spectral descriptors are often computed from the Short Time Fourier Transform (STFT). By combining this measurement with perceptually relevant information, such as accounting for frequency and temporal masking, one can produce an auditory spectrogram which can then be used to determine the loudness, timbre, onset, beat and tempo, and pitch and harmony [6]. In addition to spectral descriptors, there also exist temporal descriptors, which are composed of the audio waveform and its amplitude envelope, energy descriptors, harmonic descriptors, derived from the sinusoidal harmonic modeling of the signal, and perceptual descriptors, computed using a model of the human hearing process. The items listed here are examples of low-level audio descriptors (LLD), which are used to depict the characteristics of a sound. Examples of spectral descriptors include the spectral centroid, spread, skewness, kurtosis, slope, decrease, rolloff point, and variation. Harmonic descriptors include the fundamental frequency, noisiness, and odd-to-even harmonic ratio. Finally, perceptual descriptors include Mel-Frequency Cepstral Coefficients (MFCC), loudness, sharpness, spread, and roughness [7]. From these LLD's a higher-level representation of the signal can be formed.

1.4 Mpeg-7 Audio Descriptors

The Moving Picture Experts Group (MPEG) is a working group of ISO/IEC in charge of the development of standards for digitally coded representation of audio and video [8]. Until now, the group has produced several standards:

The MPEG-1 standard is used e.g. for Video CDs and also defines several layers for audio compression, one of which (layer 3) is the very popular MP3 format. The MPEG-2 standard is another standard for video and audio compression and is used e.g. in DVDs and digital TV broadcasting.

MPEG-4 is a standard for multimedia for the fixed and mobile web. MPEG-7 defines the Multimedia Content Description Interface and is the standard for description and search of audio and visual content. MPEG-21 defines the Multimedia Framework. The MPEG-7 standard, part 4 [9], describes a number of low level audio descriptors as well as some high-level description tools. The five defined sets for high-level audio description are partly based on the low-level descriptors and are intended for specific applications (description of audio signature, instrument timbre, melody, spoken content as well as for general sound recognition and indexing) and will not be further considered here.

The low-level audio descriptors comprise 17 temporal and spectral descriptors, divided into seven classes. Some of them are based on basic waveform or spectral information while others use harmonic or timbral information.

In the next section I speak about CLAM, a full-fledged software framework for research and application development in the Audio and Music Domain, it extracts most of the Mpeg7 low level descriptors but most of them has been review against the standard and some of them are not what they are intended (notably Spectral Kurtosis and Skew).

1.5 CLAM

CLAM stands for C++ Library for Audio and Music. It offers a conceptual model as well as tools for the analysis, synthesis and processing of audio signals. CLAM should include all utilities needed in a Sound Processing Project it is Easy to use and adapt to any kind of need and Platform Independent in fact you can compile it under GNU/Linux, Windows and Mac platforms. The framework is publicly distributed and licensed under the GPL [13].

The CLAM framework is cross-platform. All the code is ANSI C++ and it is regularly compiled under GNU/Linux, Windows and Mac OSX using the GNU C++ compiler but also the Microsoft compiler.

CLAM offers a processing kernel that includes an infrastructure and processing and data repositories.

In that sense, CLAM is both a black-box and a white-box framework [14]. It is black-box because already built-in components included in the repositories can be connected with minimum programmer effort in order to build new applications. And it is white-box because the abstract classes that make up the infrastructure can be easily derived to extend the framework components with new processes or data classes.

Finally, it also includes a number of tools that allow the user to transparently use system-level services such as audio and MIDI input/output or even GUI components in any operating system.

Apart from the infrastructure and the repositories, which together make up the CLAM *processing kernel* CLAM also includes a number of tools that can be necessary to build an audio application [15].

Any CLAM Component can be stored to XML. Furthermore, Processing Data and Processing Configurations make use of a macro-derived mechanism that provides automatic XML support without having to add a single line of code [12].

1.6 XML

XML [10] is a text based format to represent hierarchical data. XML uses named tags enclosed between angle brackets to mark the begin and the end of the hierarchical organizers, the XML elements. Elements contain other elements, attributes and plain content.

The power of XML is that you can adapt your own tags (elements) and tag attributes (attributes) in order to describe your own data. Another important advantage of the XML format is that it is structured and human-readable. For these reasons XML is starting to spread rapidly as a multimedia description language (see MPEG7 language [11], for instance). On the downsides, its main inconvenience is that, because it is a textual format, it is very inefficient both in size and in loading/storing speed. The XML specification defines the concepts of well-formedness and validity.

An XML document is well-formed if it has a correct nesting of tags. In order for a document to be valid, it must conform to some constraints expressed in its document type definition (DTD) or its associated XML Schema.

On the other hand XML-Schema [12] is a definition language for describing the structure of an XML document using the same XML syntax and it is bound to replace the existing DTD language. The purpose of a schema is to define a class of XML documents by using particular constructs to constrain their structure: datatypes, elements and their content, attributes and their values. Schemas are written in regular XML and this allows users to employ standard XML tools instead of specialized applications.

1.7 Project Objective

The main goal of this project is to design a segmentation system that is able to detect audio events in radio newscasts and that produces audio segments that could be put as input of a classification system Mpeg-7 compliant.

In the first part of this project we are interested in making experiments on the low feature extraction using CLAM from some audio sample of radio newscasts in order to investigate their possible applications in the audio event detection. While the last part is dedicated to the segmentation system, which will segment a radio newscast audio stream into homogeneous regions using one of the feature analysed.

1.8 Project Overview

The remainder of this thesis is organized into the following chapters:

- Chapter 2 presents the CLAM framework and the CLAM Music Annotator.
- Chapter3 present same examples of the spectral descriptors extract with CLAM Music Annotator.
- Chapter 4 describes the audio database, the tool for manual segmentation and how create automatically an MPEG-7 Compliant XML document for the description of the Segments.
- Chapter 5 describes the automatic audio segmentation.
- Chapter 6 shows the experiments that I made on the GRR database, the result and the final system evaluation.
- Chapter 7 describes the final system evaluation.

CHAPTER 2
AUDIO FEATURE EXTRACTION USING CLAM

A useful implementation of the CLAM framework is the CLAM Music Annotator [18]. This tool allows one to analyze and visualize a piece of music. The annotator extracts LLD's, as well as high-level features like the roots and modes of chords, note segmentation, and song structure. CLAM's Annotator can be customized to extract any set of features based on an XML description schema definition. For example, one could create a general schema that extracts a wide range of LLD's, or a specific schema could be designed for chord extraction.

In terms of visualization, the annotator includes a Tonnetz visualization for tone correlation display and a key space, courtesy of Jordi Bonada and Emilia Gomez at the University of Pompeu Fabre, for major and minor chord correlation.

The CLAM Music Annotator makes use of different XML files in order to relate with the outside world. All previously generated information is input to the program through XML files and the result of the editing process is also dumped into those files.

The Project File contains a pointer to a Schema File and another one to a Song List File. The Song List File contains a list of Audio Files and Descriptors Pool Files.

In the following sections we will detail the content of each of those files.

2.1 The Project File

The Project file is an XML file with the "\.pro" extension. It simply contains the name of the extractor and the path to the Schema File. It also contains a list of Sound file. A Sound file is simply the path to an existing sound file. This sound file can be in virtually any format, including PCM encoded files such as WAVs or AIFFs or compressed formats such as MP3 or OGG. On the other hand, the Descriptors File has a pointer to the descriptors related to that particular sound file. If omitted, the program will simply add the "\.pool" extension to the sound file name.

In the listing 2.1 is shown an example of the song list file.

```
- <Project>
  - <Description>
      <h1>Example project for the CLAM Descriptors Extractor</h1> <p> <em>This is a generated project to test the CLAM music
      annotator.Descriptors have been generated by the program <tt>ClamExtractorExample</tt> provided with the annotator.You
      could use that extractor as example to build your own one.</em></p> <h2>Description:</h2> <p> In a real project this
      documentation may be useful for example to give instructions aboutthe annotation task or the aspects to take into account for
      the review.</p> <h2>Reference descriptors:</h2> <p> You can take some descriptors as reference such as the energy. Consider
      that such descriptors are computed every 2048 samples so their precission may be not as accurate as it seems. </p>
    </Description>
    <Schema>CLAMDescriptors.sc</Schema>
    <Extractor >ClamExtractorExample  </Extractor >
    <PoolSuffix>.pool</PoolSuffix>
  - <Songs>
    - <Song>
      - <SoundFile>
          /home/vincenzo/Scrivania/WavTestCLAM/11-Anchorwoman-61-training.wav
        </SoundFile>
      </Song>
    - <Song>
      - <SoundFile>
          /home/vincenzo/Scrivania/WavTestCLAM/Anchorman2-36-training.wav
        </SoundFile>
      </Song>
    - <Song>
      - <SoundFile>
          /home/vincenzo/Scrivania/WavTestCLAM/11-MaleTel-52-training.wav
        </SoundFile>
      </Song>
    - <Song>
      - <SoundFile>
          /home/vincenzo/Scrivania/WavTestCLAM/12-Music-146-okMin.wav
        </SoundFile>
      </Song>
    - <Song>
      - <SoundFile>
          /home/vincenzo/Scrivania/WavTestCLAM/ThemeMusic-165-okMin.wav
        </SoundFile>
      </Song>
    </Songs>
  - <Views>
    - <View>
        <Type>BarGraph</Type>
        <AttributeScope >Frame </AttributeScope >
        <AttributeName >MelFrequencyCepstrumCoefficients  </AttributeName >
      </View>
    </Views>
  </Project>
```

Listing 2.1: Sample SongList file

2.2 The Schema File

The Schema File contains the list of all the different descriptors that later will be loaded from a Descriptors File. In some cases it also gives their type and range of expected values. Although this File is a regular XML

File with the "\.sc" extension, in many senses it mimics the purpose and syntax of an XML Schema format [12].

The Schema File is actually divided into two different sections. The first one defines the Schema for high-level descriptors while the second one defines the schema for low-level descriptors (see example in listing 2.2).

A high-level descriptor is considered to be any descriptor that has a whole song scope and is unique within this scope. This kind of descriptor can be of any type. On the other hand a low-level descriptor has a Frame scope and can only take floating point values. In this thesis I don't care about high level descriptors.

```
- <DescriptionScheme>
  - <Uri>
      descriptionScheme:www.iua.upf.edu:clam:dummyTest-0.90
    </Uri>
  - <Attributes>
      <Attribute name="Artist" scope="Song" type="String" />
      <Attribute name="Title" scope="Song" type="String" />
    - <Attribute name="Genre" scope="Song" type="Enumerated" >
        <EnumerationValues>Dance Classic Jazz Rhythm&Blues Folk </EnumerationValues>
      </Attribute>
      [...]
    - <Attribute name="SimilarityGroup" scope="StructuralPart" type="Enumerated" >
        <EnumerationValues>A B C D E F G H I J </EnumerationValues>
      </Attribute>
    - <Attribute name="Frames" scope="Song" type="FrameDivision" >
        <ChildScope>Frame </ChildScope>
      </Attribute>
    - <Attribute name="Mean" scope="Frame" type="Float">
      + <Documentation></Documentation>
      </Attribute>
      <Attribute name="GeometricMean" scope="Frame" type="Float"/>
      <Attribute name="Energy" scope="Frame" type="Float"/>
      <Attribute name="Centroid" scope="Frame" type="Float"/>
      <Attribute name="Moment2" scope="Frame" type="Float"/>
      <Attribute name="Moment3" scope="Frame" type="Float"/>
      <Attribute name="Moment4" scope="Frame" type="Float"/>
      <Attribute name="Moment5" scope="Frame" type="Float"/>
      <Attribute name="Moment6" scope="Frame" type="Float"/>
      <Attribute name="Flatness" scope="Frame" type="Float"/>
      <Attribute name="MagnitudeKurtosis" scope="Frame" type="Float"/>
      <Attribute name="MaxMagFreq" scope="Frame" type="Float"/>
      <Attribute name="LowFreqEnergyRelation" scope="Frame" type="Float"/>
      <Attribute name="Spread" scope="Frame" type="Float"/>
      <Attribute name="MagnitudeSkewness" scope="Frame" type="Float"/>
      <Attribute name="Rolloff" scope="Frame" type="Float"/>
      <Attribute name="Slope" scope="Frame" type="Float"/>
      <Attribute name="HighFrequencyContent" scope="Frame" type="Float"/>
    - <Attribute name="MelFrequencyCepstrumCoefficients" scope="Frame" type="FloatArray">
      - <BinLabels>
          MFCC1 MFCC2 MFCC3 MFCC4 MFCC5 MFCC6 MFCC7 MFCC8 MFCC9 MFCC10 MFCC11
          MFCC12
        </BinLabels>
      </Attribute>
    </Attributes>
</DescriptionScheme>
```

Listing 2.2: Sample Annotator Schema File

We will now see how the schema is defined both for high-level and low-level descriptors.

2.3 High-level descriptors

As already mentioned a high-level descriptor has a unique value for a whole song or sound source. It can be of any of the following types: floating point number (“\Float"); integer number (“\Int”); string (“\String”); or value set restricted strings (“\Enum”).

A high-level descriptor is therefore defined by giving its “\Name” and its “\Type”. In case the type is a number, an optional range of valid values may be given (“\iRange” in case of integer values and “\fRange” in case of floating point values). See the HLD section in listing 2.2.

2.4 Low-level Descriptors

A low-level descriptor is in any case a vector of floating point values where each value refers a particular frame. In this case we only need to define the name of the descriptors. Therefore the low-level descriptors section of the schema is simply a list of low-level descriptors names (see again listing 2.2).

2.5 Descriptors Pool File

This is an XML file with the “\.pool” extension that contains all the values, both for the high-level and low-levels descriptors. The content must observe the restrictions given in the related Schema or else it will not be validated.

Every song on the project has its own descriptors file. Descriptions may be generated by any third-party application by providing a proper schema, though it is much easier to generate it from within the CLAM framework. In this case, the Descriptors file is directly the XML representation of a CLAM Descriptors Data Pool. Any extraction algorithm using them may dump its results in such format without having to worry about formatting issues.

As in the Schema, a Descriptors file is divided into two sections: one for the high-level descriptors and another one for the low-level descriptors (see listing 2.3). Note that in the Descriptors file this difference is explicit by the existence of two different “\Scopes”, one with the name “\Song” and size=1 (there is only one song for each song) and the other one with the name \Frame" and size=432 (in this case there are 432 frames in the wav).

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<DescriptorsPool>

 <ScopePool name="Song" size="1">
  [...]
 <ScopePool name="Frame" size="432">
  [...]
 </ScopePool>

</DescriptorsPool>
```

Listing 2.3: Sample Descriptors file

I will now explain how high and low-level descriptors are stored.

2.6 High-level Descriptors

In the Song scope we basically see a list of AttributePool elements. In any case each of those elements has an attribute with the name of the particular descriptor and its content is the content of the descriptor. Note that the type of the descriptor is implicitly resolved from the schema and must therefore not be given in the Descriptors File (see HLD description in listing 2.4). Finally, segmentation information is also included in the high-level description. This descriptor must not be given in the schema as it is always supposed to be available. When including segmentation marks in the description you must give their size (i.e. how many segmentation marks are available) and the list of positions in number of samples. Those song level values are just dummy values generated to demonstrate Annotator capabilities to adapt different kinds of data.

```
- <ScopePool name="Song" size="1">
    <AttributePool name="Artist">Unknown Artist</AttributePool>
    <AttributePool name="Title">Unknown Title</AttributePool>
  - <AttributePool name="Genre">
      <Enumerated>Folk</Enumerated>
    </AttributePool>
    [...]
    <AttributePool name="SimilarityGroup"/>
  </ScopePool>
```

Listing 2.4: Sample High-level Description

2.7 Low-Level descriptors

The low-level descriptors section of the Descriptors file is also a list of AttributePool elements where for each element we must define its name and a list of values. Note that in this

case we must not give the size of each attribute because this is already defined by the size of the "\Frame" scope. Therefore these vectors must all have as many elements as defined in the scope (432 in the example given in listing 2.5).

```
- <ScopePool name="Frame" size="432">
   + <AttributePool name="Mean"></AttributePool>
   + <AttributePool name="GeometricMean"></AttributePool>
   + <AttributePool name="Energy"></AttributePool>
   + <AttributePool name="Centroid"></AttributePool>
   + <AttributePool name="Moment2"></AttributePool>
   + <AttributePool name="Moment3"></AttributePool>
   + <AttributePool name="Moment4"></AttributePool>
   + <AttributePool name="Moment5"></AttributePool>
   + <AttributePool name="Moment6"></AttributePool>
   + <AttributePool name="Flatness"></AttributePool>
   + <AttributePool name="MagnitudeKurtosis"></AttributePool>
   + <AttributePool name="MaxMagFreq"></AttributePool>
   + <AttributePool name="LowFreqEnergyRelation"></AttributePool>
   + <AttributePool name="Spread"></AttributePool>
   + <AttributePool name="MagnitudeSkewness"></AttributePool>
   + <AttributePool name="Rolloff"></AttributePool>
   + <AttributePool name="Slope"></AttributePool>
   + <AttributePool name="HighFrequencyContent"></AttributePool>
   + <AttributePool name="MelFrequencyCepstrumCoefficients" size="20"></AttributePool>
   </ScopePool>
</DescriptorsPool>
```

Listing 2.5: Sample Low-level Description

2.8 Showcase

The application has been developed within the CLAM framework, using qt [19] for the graphical user interface. Figure 2.1 is a capture of the whole interface running on UBUNTU, although its look is virtually the same in any of the major platforms and graphical environments (Windows, Mac OSX, GNOME...).

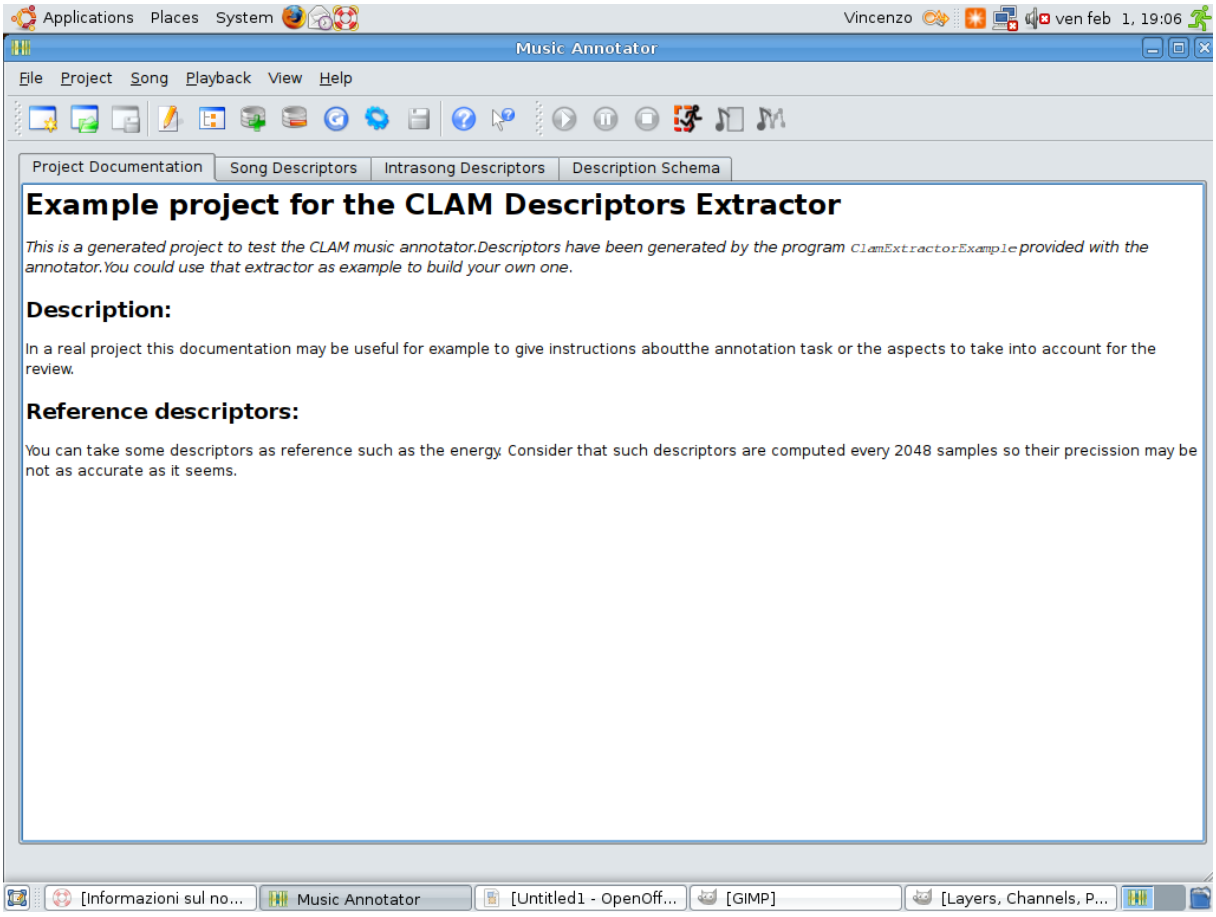


Figure 2.1: The CLAM Music Annotator GUI

2.9 Loading a Project

Once the program is started, the first thing that we must do is to load a project file. This project file will have a pointer to the Song List and the Schema files. Once loaded, the GUI is reconfigured and the list of songs and related descriptions is available (see Figure 2.2).

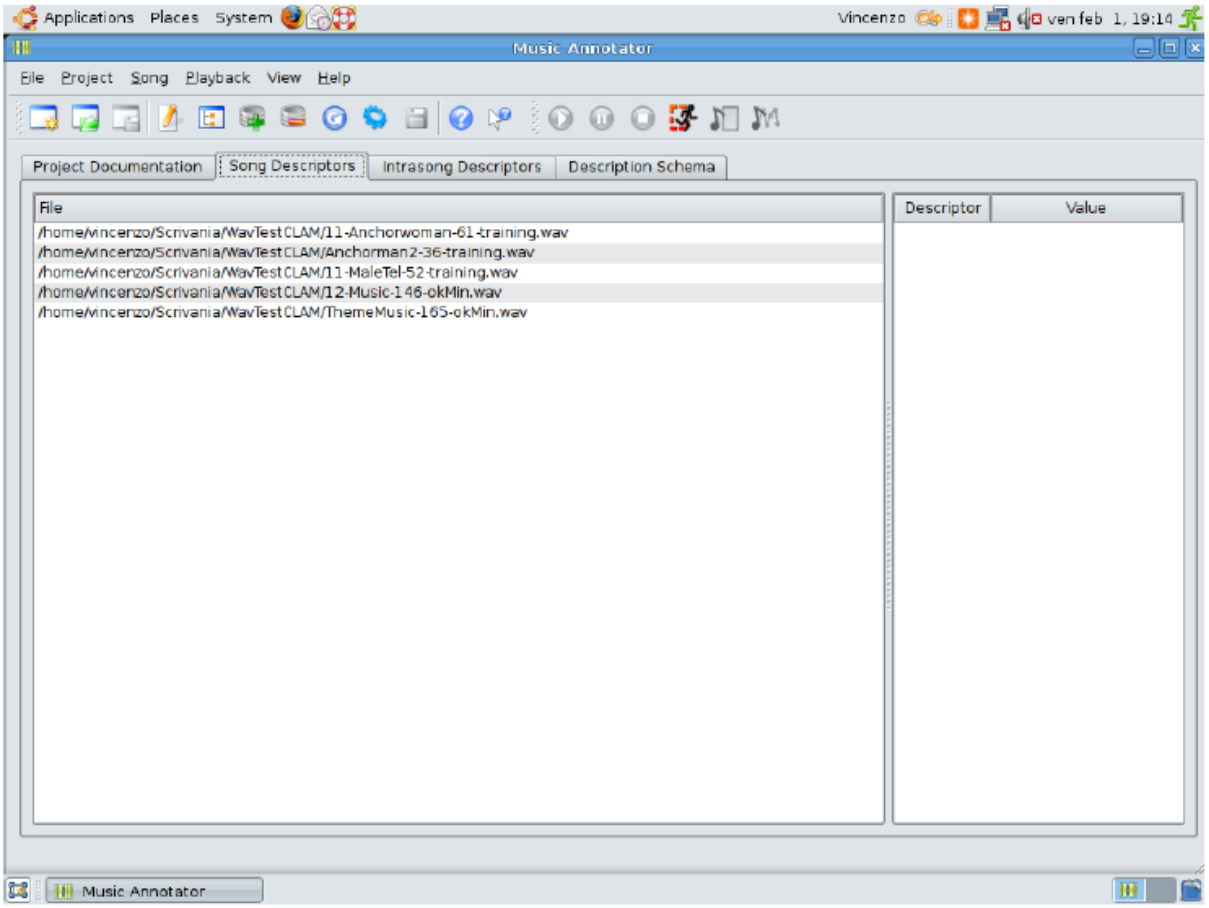


Figure 2.2: The CLAM Music Annotator GUI song list

2.10 The Schema and the dynamic GUI

One of the most important features in the CLAM Annotator is its ability to dynamically adapt the GUI. The GUI shows the descriptors according to the Schema that is loaded with the Project.

In the case of low-level descriptors, the amount and label of each of the tabs corresponds to the schema. And in the case of high-level descriptors, the schema defines the label and also the kind of editing widget that is shown.

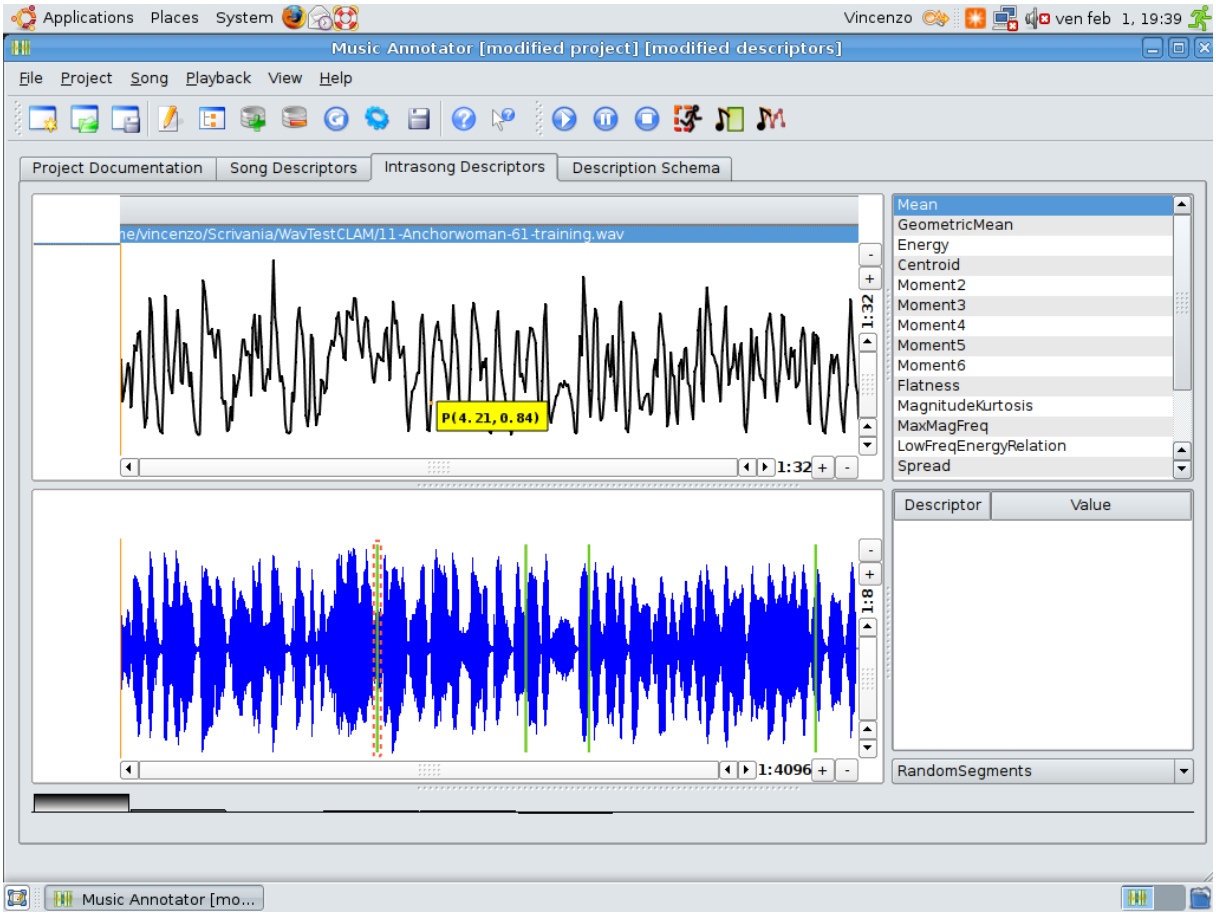


Figure 2.3: The low-level descriptors

2.11 Viewing Song Properties

Once a song is selected from the Song List, the audio file and the descriptors are loaded. After this loading process finishes the waveform including segmentation marks is available on the lower-left, the low-level descriptors are shown on the upper-right, and the high-level descriptors are on the lower-right. The user can listen to the sound file and start the edition process.

Low level descriptors view and segmentation view are synchronized in respect zoom, horizontal scroll and cursor position. That feature makes easy to take segmentation editing decisions taking into account low level features values.

2.12 Editing Low-level Descriptors

Low-level descriptors are represented by equidistant connected points that you can drag to change its Y value. Each point represents the value for the descriptor in a given frame.

Because point to point edition may be hard, some convenient edition modes such trim or draw are provided.

2.13 Editing High-level Descriptors

The edition of a high-level description adapts on the "\type" of the descriptor as defined in the project's schema. Figure 2.4 shows how integer and float descriptors may be edited by a slider that uses the range given in the schema, while enumerated value descriptors can be selected with a drop-down list widget with the allowed values.

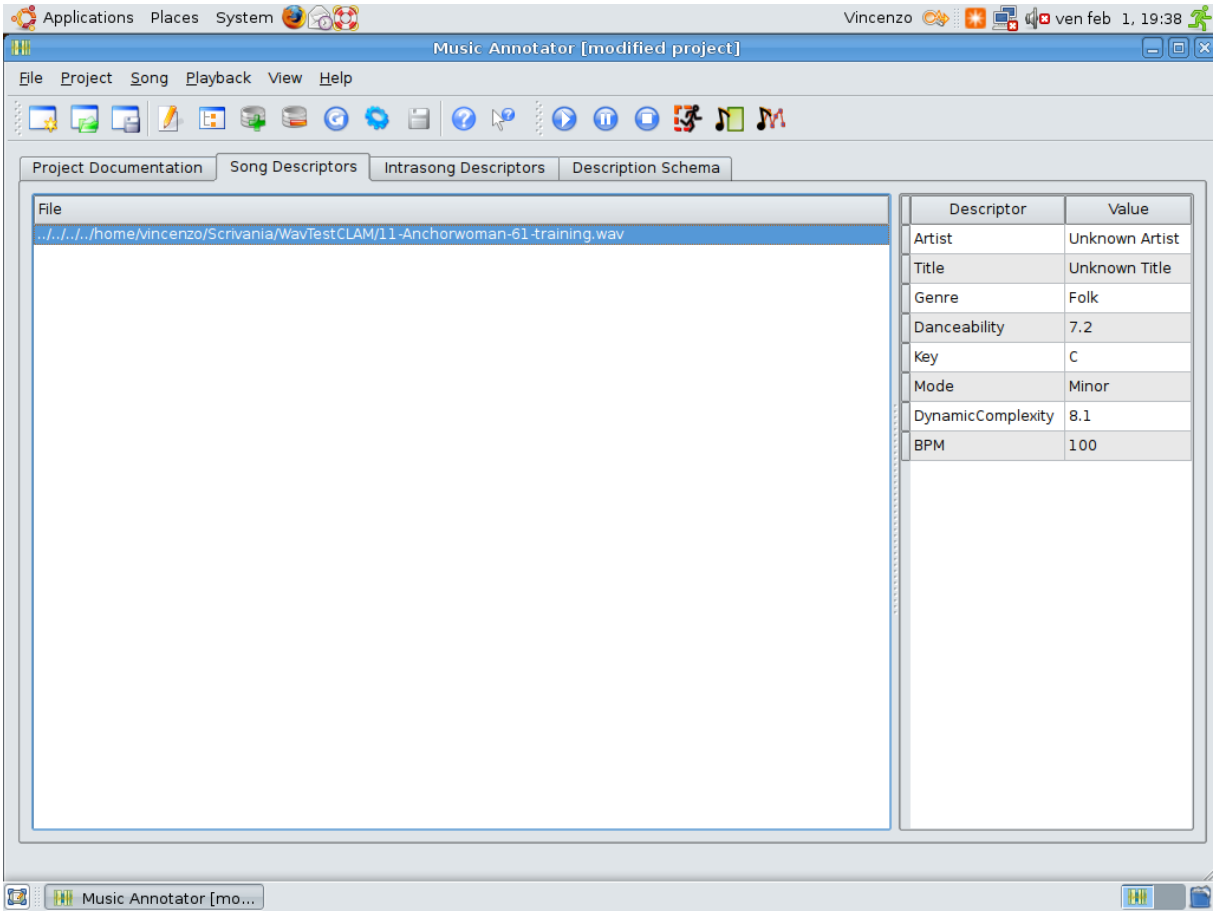


Figure 2.4: The high-level descriptors

Regular strings use a simple text box widget were the user can enter free text.

2.14 Viewing Associated Schema

The Schema file contains the list of all the different descriptors (see Figure 2.5). The Schema file is actually divided into two different sections. The first one defines the Schema for high-level descriptors while the second one defines the schema for low-level descriptors.

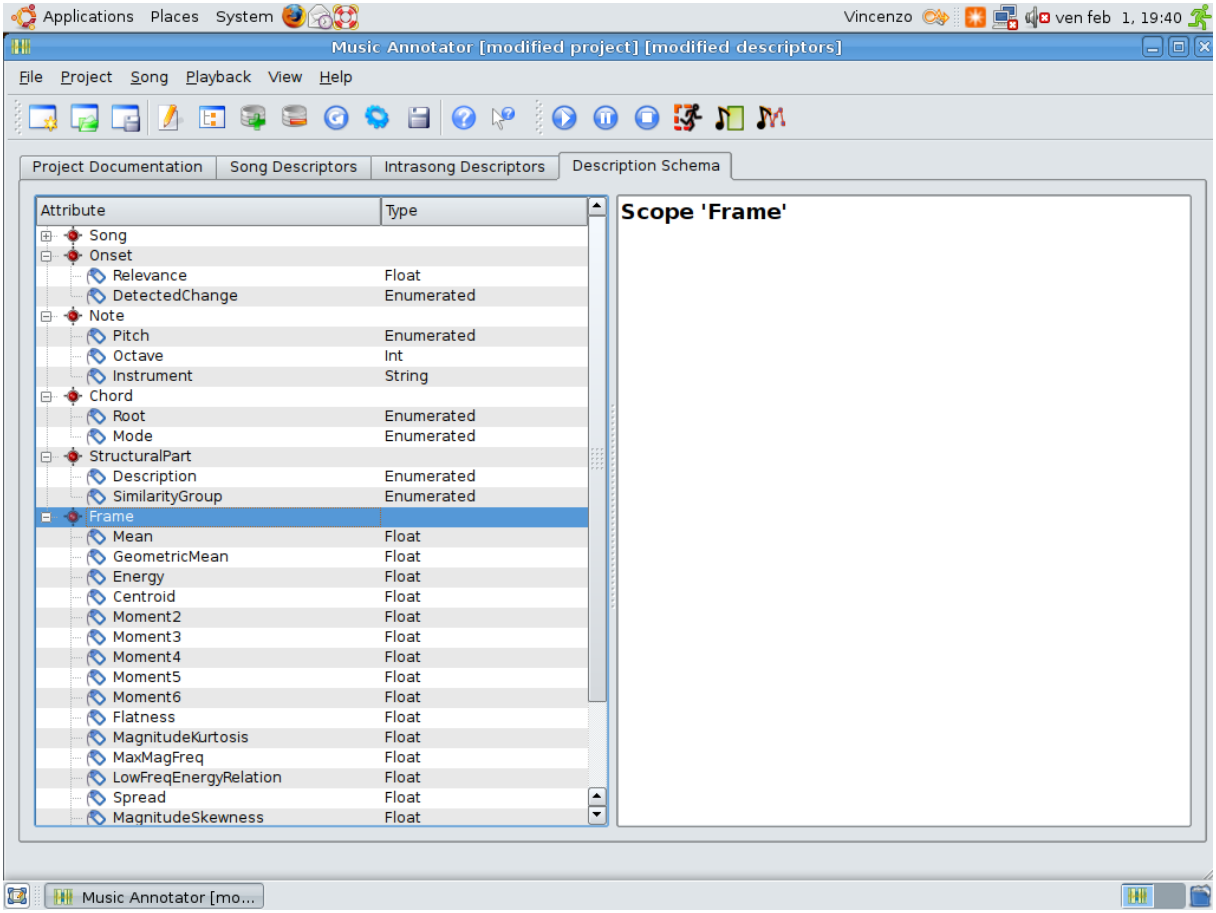


Figure 2.5: Visualization of the Description Schema

CHAPTER 3
SPECTRAL DESCRIPTOR EXTRACTION WITH `ClamExtractorExample`

In this chapter the features will be described and illustrated through an example. The features are extracted with `ClamExtractorExample` from speech and music samples of the Giornale Radio RAI (GRR) data base that I will describe in the next chapter.

The type of signals we are dealing with, namely speech and music, are so called quasi-stationary signals. This means that they are fairly stationary over a short period of time.

This encourages the use of features extracted over a short time period.

In this project the audio used is in 44,100 kHz, 16 bit, PCM wave format. The audio is partitioned into frames of 1023 samples.

To obtain the graph of the descriptors I write a Matlab code and I used XMLTree an XML toolbox for Matlab [46] in order to allow Matlab to read the descriptors from the XML CLAM files.

3.1 Mean

This descriptor is the spectral power mean value. It is calculated by CLAM computing [58]:

$$Mean(X) = \frac{\sum x_i}{Size\,(X)} \qquad (3.1)$$

In the listing 3.1 is reported its definition in Stats.hxx.

```
U GetMean()
{
        if (mData->Size()<=0) return U(.0);
        //FirstOrder* first;
        return GetMoment(FirstOrder);
}
```

Listing 3.1: Definition at line 216 of file Stats.hxx.

Figure 3.1(a) shows 7 seconds of male speech with the corresponding mean (Figure 3.1(b)) and 3.1(c) shows 7 seconds of Theme music speech with the mean (Figure 3.1 (d)). In the figure 3.1 we can see that the mean follows the envelope of the corresponding signal.

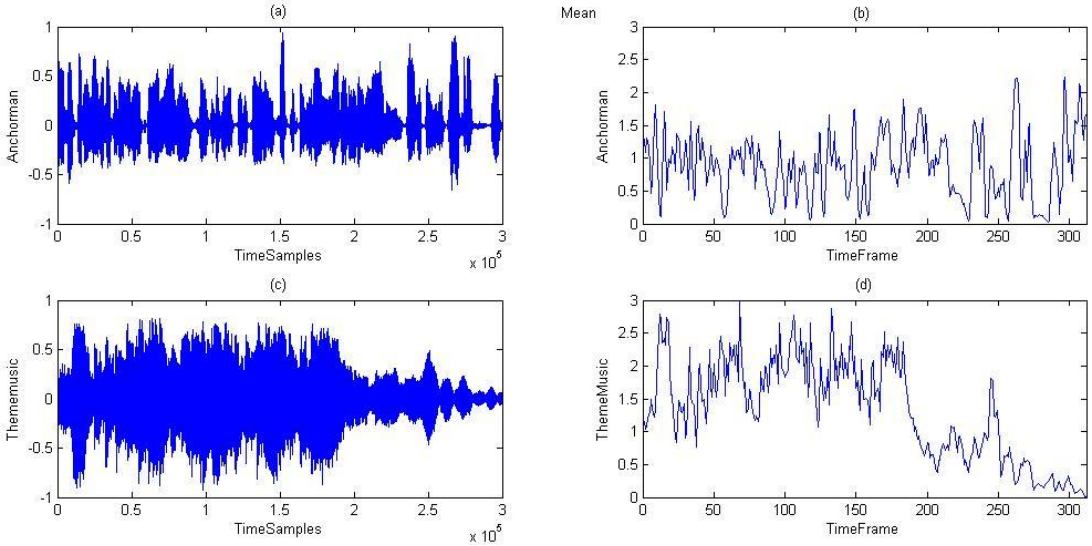


Figure 3.1: (a) Anchorman speech signal of 7 s; (b) Mean of the Anchorman speech; (c)Theme music signal of 7 s; (d) Mean of the Theme Music

3.2 Geometric Mean

This feature is the geometric mean of the spectral power values sequence [58]. The geometric mean of a sequence $\{a_i\}_{i=1}^{n}$ is defined by

$$G(a_1, \dots, a_n) \equiv \left(\prod_{i=1}^{n} a_i\right)^{1/n} \qquad (3.2)$$

Thus,

$$G(a_1, a_2) = \sqrt{a_1 a_2} \qquad (3.3)$$

$$G(a_1, a_2, a_3) = \sqrt{a_1 a_2 a_3} \ , \qquad (3.4)$$

and so on.

Computing this measurement over long sequences of small real numbers poses a numerical problem. To avoid this, computation of Geometric mean is restricted to Log scale Spectral Power Distributions since this allows changing the product for a summation. This measure is expressed in dBs.

The geometric mean gives the mean magnitude order. It converges with the mean when all the values x_i are closer.

$$GeometricMean(X) = (\prod x_i)^{\frac{1}{Size\ (X)}} \quad (3.5)$$

In order to make the computation cheap, for easy computation, logarithms are used.

$$\log(GeometricMean(X)) = \frac{\sum \log_e x_i}{Size\ (X)} \quad (3.6)$$

In the listing 3.2 is reported the definition in Stats.hxx of the function that computes the Geometric Mean.

```
U GetGeometricMean()
{
        return mGeometricMean(*mData);
}
```

Listing 3.2: Definition at line 433 of file Stats.hxx

Figure 3.2(a) shows 10 seconds of male speech with the corresponding geometric mean in Figure 3.2(b) and 3.2(c) shows 10 seconds of music signal with the geometric mean in Figure 3.2(d). In the figure 3.2 we can see that the geometric mean follows the envelope of the corresponding signal.

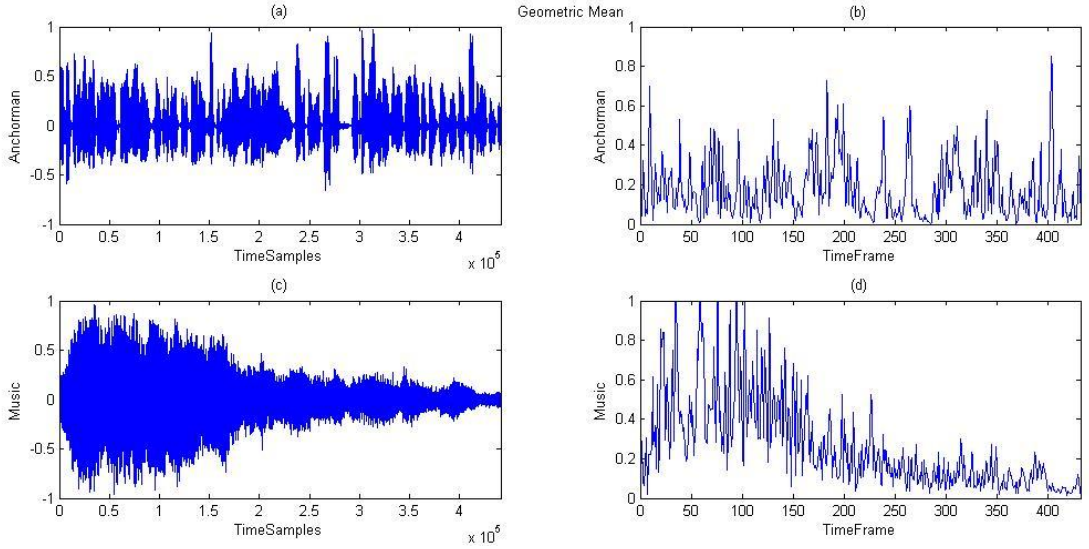


Figure 3.2: (a) Anchorman speech signal of 10 s; (b) Geometric Mean of the Anchorman speech; (c) Music signal of 10 s; (d) Geometric Mean of the Music

3.3 Energy

This descriptor is another simple feature that has been used in various formats in audio applications. The energy is the squared sum of spectral power distribution values. It is defined as the total energy in a frame [58]:

$$Energy(X) = \sum x_i^2 \qquad (3.7)$$

In the listing 3.3 is reported the definition in Stats.hxx of the function that computes the Energy.

```
T GetEnergy()
{
        return mEnergy(*mData);
}
```

Listing 3.3: Definition at line 411 of file Stats.hxx

The speech signal is composed of altering voiced and unvoiced sounds and silence periods. These unvoiced and silence periods carry less energy than the voiced sounds. Thus, the Energy values for speech will have a large variation. This can also be seen in figure 3.3(a), where the same speech signal is shown and the corresponding Energy values are shown in Figure 3.3 (b).

In figure 3.3 (b) we can see that the voiced speech parts give larger energy values than the unvoiced and silence periods.

Because of the pitched nature of music the Energy of music is more constant and larger than speech. This can be seen in figure 3.3(c), where the music signal is shown with the corresponding Energy values in figure 3.3(d). As shown in the same figure is clear that the pitched parts of the music give high Energy values.

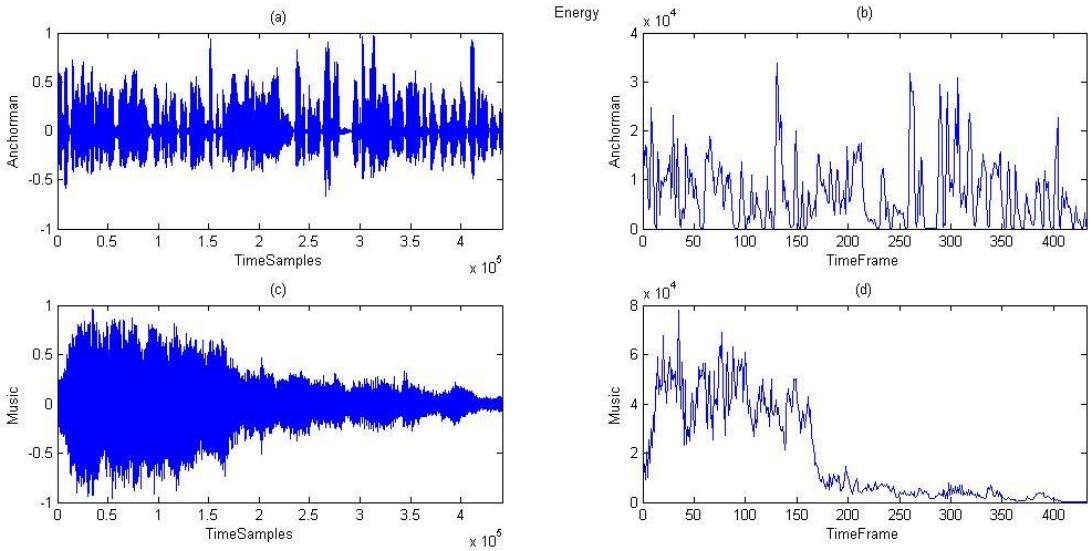


Figure 3.3: (a) Anchorman speech signal of 10 s; (b) Energy of the Anchorman speech signal; (c) Music signal of 10 s; (d) Energy of the Music signal

3.4 Centroid

The Centroid is the frequency where the center of mass of the spectral power distribution lies. This measure is expressed in Hz. The Centroid of a function returns the position i around which higher values are concentrated [58].

$$Centroid(X) = \frac{\sum i \cdot x_i}{x_i} \quad (3.8)$$

How we can see in the listing 3.4 whenever the Mean(X) is less than 1e-7, then it will return the mid position

$$\frac{Size\,(X)-1}{2} \quad (3.9)$$

```
U GetCentroid()
{
        return GetCenterOfGravity(FirstOrder);
        if (mCentroid.HasValue()) return mCentroid;
        unsigned N = mData->Size();
        U mean = GetMean();
        if (mean < 1e-7 )
        {
                mCentroid = U(N-1)/2;
                return mCentroid;
        }
        U centroid=0.0;
        for (unsigned i = 0; i < N; i++)
        {
                centroid += (abs?Abs((*mData)[i]):(*mData)[i]) * (i+1);
        }
        mCentroid=centroid/mean/U(N) - 1;
        return mCentroid;
}
```

Listing 3.4: Definition at line 241of file Stats.hxx.

Spectral Centroid is the "balancing point" of the spectral power distribution. It is an indicator as to whether the power spectrum is dominated by low or high frequencies and can be regarded as an approximation of the perceptual sharpness of the signal [48].

In the Figure 3.4(b) and in Figure 3.4(d) we can see that the telephone reporter has a lower centroid respect to the anchorman centroide because in the telephone speech signal the higher frequencies are cut by the communication channel. It is clear comparing Figure 3.4 (a) and 3.4 (b).

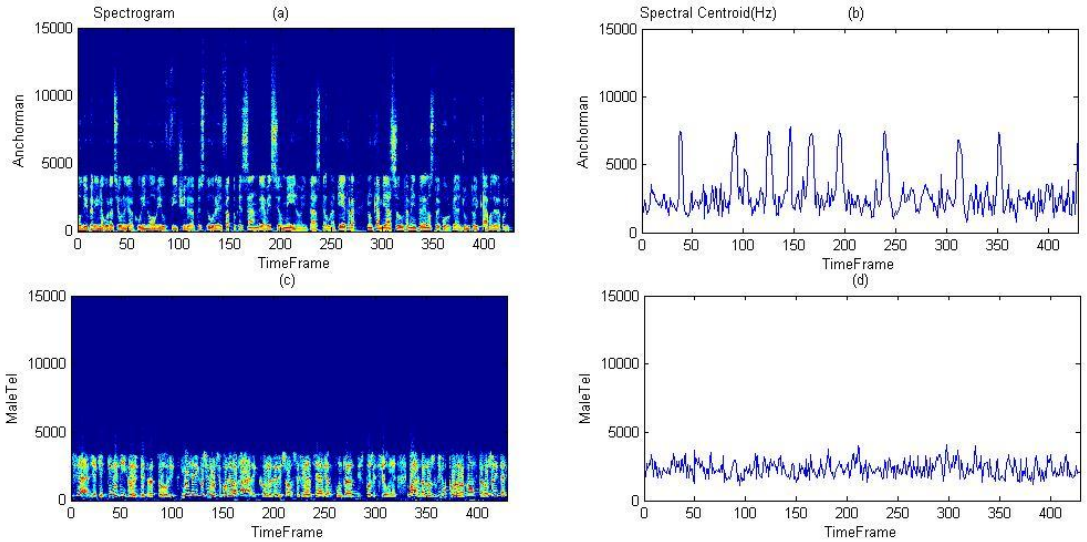


Figure 3.4: (a) Top 50dB of the Anchorman speech signal spectrogram of 10 s; (b) Spectral Centroid of the Anchorman speech signal; (c) Top 50dB of the Male Telephone signal spectrogram of 10 s; (d) Spectral Centroid of the Male Telephone signal

Many kinds of music involve percussive sounds which, by including high-frequency noise, push the spectral mean higher. In addition, excitation energies can be higher for music than

for speech, where pitch stays in a fairly low range [47]. But from Figure 3.5 (a) it is clear that the theme music of the GRR of our example has a spectrum dominated by lower frequencies, in fact if we compare the Figure 3.5(b) and 3.4(b) we can notice that the theme music has a centroid lower than the anchorman centroid. Speech is usually limited in frequency to about 8 kHz whereas music can extend through the upper limits of the ear's response at 20 kHz. In general, most of the signal power in music waveforms is concentrated at lower frequencies [52].

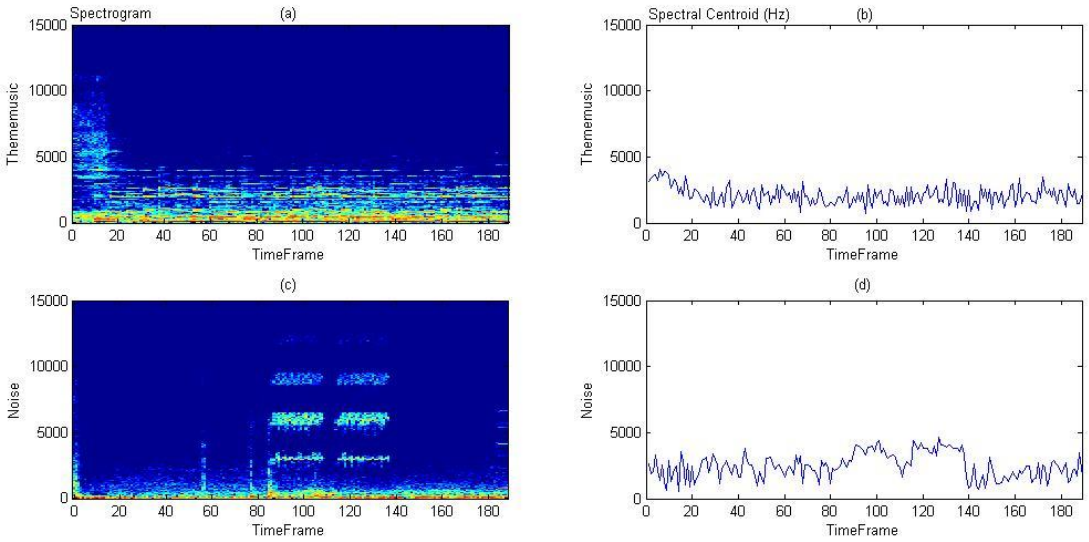


Figure 3.5: (a) Top 50dB of the theme music signal spectrogram of 4 s; (b) Spectral centroid of the theme music signal; (c) Top 50dB of the noise signal spectrogram of 4 s; (d) Spectral centroid of the noise signal

3.5 Flatness

The spectrum flatness reflects the flatness properties of the power spectrum. The flatness is the relation among the geometric mean and the arithmetic mean [58].

$$Flatness(X) = \frac{GeometricMean\ \ (X)}{Mean\ (X)} \quad (3.10)$$

How we can see in the listing 3.5 the function that computes this descriptor is the GetFlatness. When the mean is lower than 1e-20, it is set at 1e-20 and when the geometric mean is lower than 1e-20, it is set at 1e-20

```
U GetFlatness()
{
        U mean = GetMean();
        U geometricMean = GetGeometricMean();
        if (mean<1e-20) mean=TData(1e-20);
        if (geometricMean<1e-20 ) geometricMean=TData(1e-20);
        return geometricMean/mean;
}
```

Listing 3.5: Definition at line 552 of file Stats.hxx.

The Spectral Flatness is used in order to determine the noiselike or tonelike nature of the signal. In practice the use of the Spectral Flatness is useful to estimate the tonality of the signal [49].

A flat spectrum shape corresponds to a noise or an impulse signal. Hence, high flatness values are expected to reflect noisiness. On the contrary, low values may indicate a harmonic structure of the spectrum. From a psycho-acoustical point of view, a large deviation from a flat shape (i.e. a low spectral flatness measure) generally characterizes the *tonal* sounds [10].

The Figure 3.6 shows that the Scream signal flatness is higher than the Song spectral flatness in fact the scream signal is more impulsive respect to the Song signal.

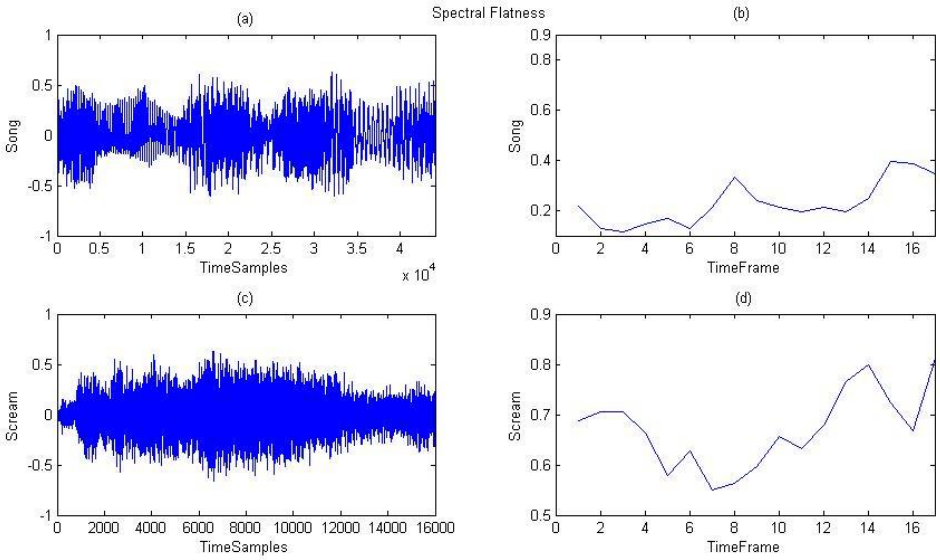


Figure 3.6: (a) 44,100 kHz song signal of 1 s; (b) Spectral flatness of the song signal; (c) 16 khz scream signal of 1 s; (d) Spectral flatness of the scream signal.

3.6 Magnitude Kurtosis

The Kurtosis of a distribution gives an idea of the degree of pickness of the distribution [58].

$$Kurtosis(X) = \frac{\sum((x_i - Mean\ (X)^4)}{(\sum(x_i - Mean\ (X))^2)^2} \quad (3.11)$$

Typical values:

- A normal distribution of x_i values has a kurtosis near to 3.
- A constant distribution has a kurtosis of $\frac{-6(n^2+1)}{5(n^2-1)} + 3$

Singularities and solutions:

- Constant functions: Currently returns 3 although it is not clear that it should be the right one, and it can vary on future implementations.

In the listing 3.6 is reported the definition in Stats.hxx of the function that computes the Magnitude Kurtosis.

```
U GetKurtosis()
{
        return mKurtosis(*mData,GetCentralMomentFunctor<2>(),GetCentralMomentFunctor<4>(),true);
}
```

Listing 3.6: Definition at line 383 of file Stats.hxx

Mixtures of speech signals have a kurtosis lower than the kurtosis values of the individual speech signals [50].

The Figure 3.7 confirm this, in fact male with female speech signal has a kurtosis values lower than the kurtosis values of the anchorwoman speech signal.

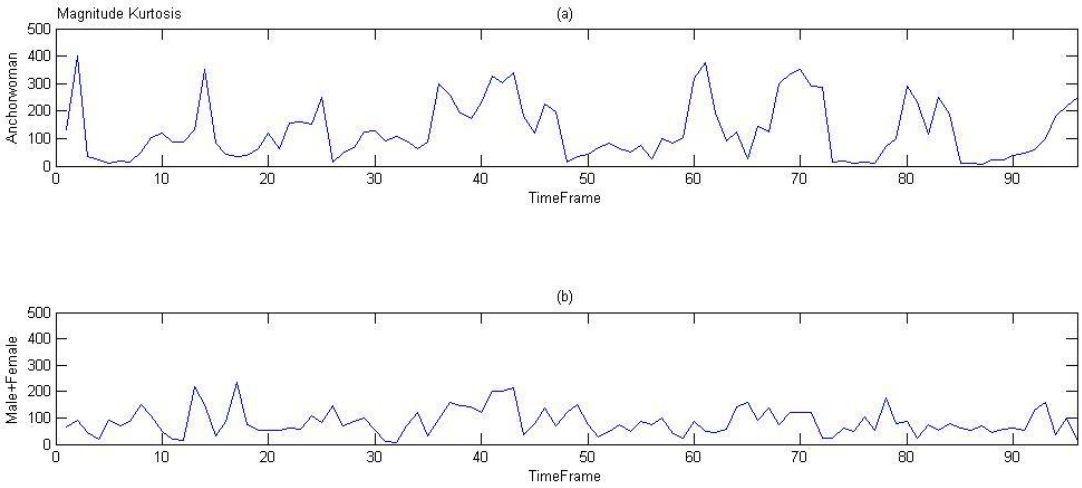
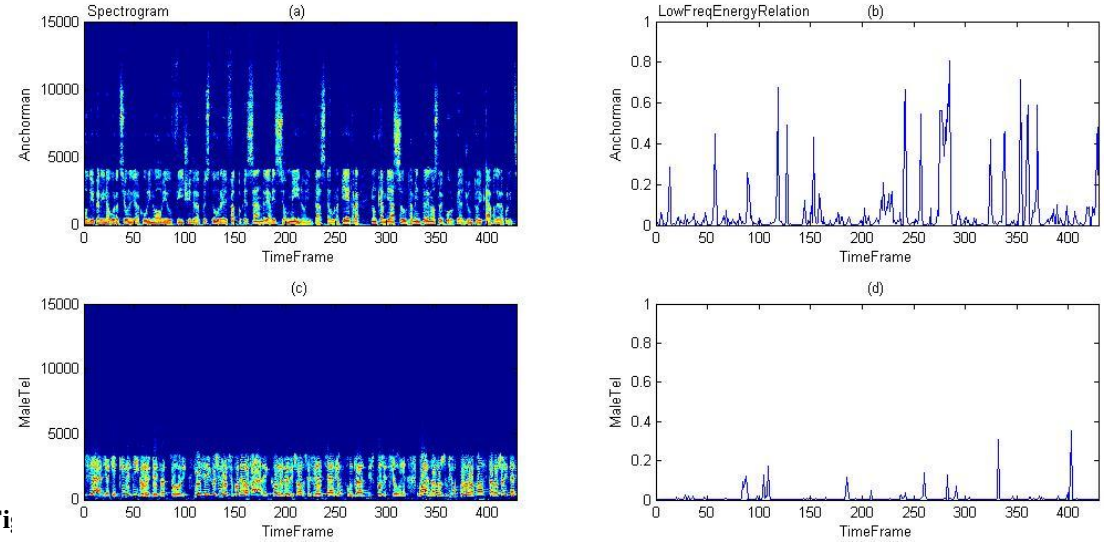


Figure 3.7: (a) Spectral Kurtosis Magnitude of Anchorwoman speech signal of 2 s; (b) Spectral Kurtosis Magnitude of Male with Female speech signal of 2 s

3.7 LowFreqEnergyRelation

This descriptor is the ratio between the energy over 0-100 Hz band and the whole spectrum energy. To avoid singularities while keeping descriptor continuity, when the whole spectrum energy drops bellow 10^ {-4}, such value is considered as whole spectrum energy. 0-100 Hz is a very narrow band so this feature is used to spot bass sounds. Speech is in general composed of altering voiced and unvoiced sounds. In between words small silence periods occur. The voiced sounds in speech are the sounds where a pitch can be found. Unvoiced sounds on the other hand have a structure that resembles noise. Figure 3.8(a) shows 10 seconds of top 50dB of the anchorman speech signal spectrogram. The Figure 3.8(b) shows low LowFreqEnergyRelation values where voiced sounds are present and high LowFreqEnergyRelation values where unvoiced sounds are present. The altering voiced and unvoiced sounds in speech give the LowFreqEnergyRelation values a relatively large variation.



Fi of the anchorman speech signal; (c) Top 50dB of the male telephone speech signal spectrogram of 10 s; (d) LowFreqEnergyRelation of the male telephone speech signal

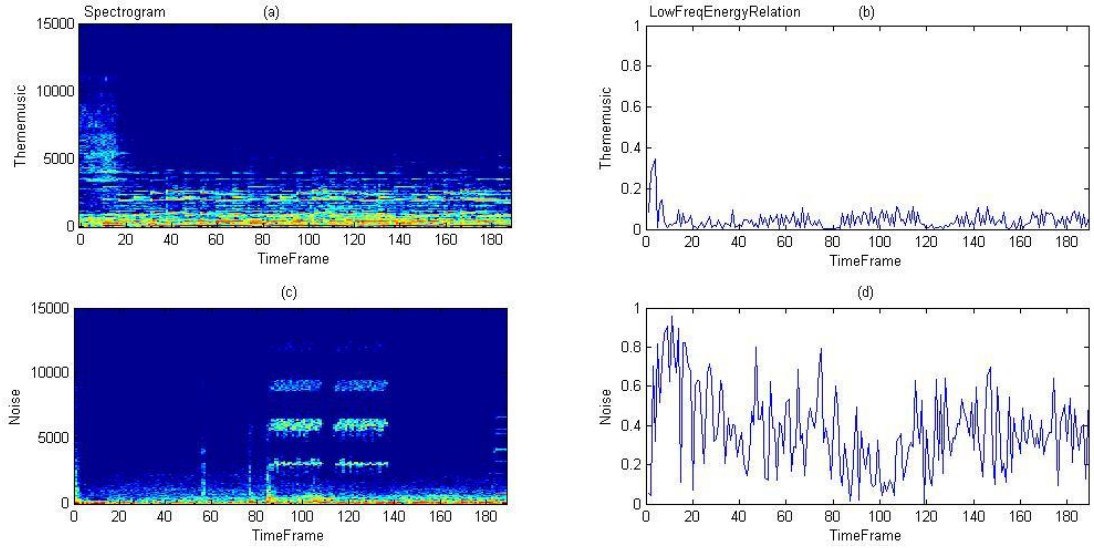


Figure 3.9: (a) Top 50dB of the theme music signal spectrogram of 4 s; (b) LowFreqEnergyRelation of the theme music signal; (c) Top 50dB of the noise signal spectrogram of 4 s; (d) LowFreqEnergyRelation of the noise signal

In general music is more pitched than speech. This is caused by the clear tones made by the instruments. Figure 3.9 (a) shows 4 seconds of top 50dB of the theme music signal spectrogram. The figure 3.9 (b) shows the LowFreqEnergyRelation does not have as many

peaks as the speech-signal. This gives a smaller variation of LowFreqEnergyRelation. The Figure 3.9 (d) shows that the LowFreqEnergyRelation of the environmental noise has a large variation.

3.8 MaxMagFreq

This descriptor gives the frequency where there is the maximum of the spectral amplitude. The Figure 3.10(a) shows high MaxMagFreq values where voiced sounds are present and low Max values where unvoiced sounds are present. The altering voiced and unvoiced sounds in speech give the MaxMagFreq values a relatively large variation. We can make the same considerations for the figure 3.10(b) but in this case the maximum frequency is 3 kHz, which is the maximum frequency of the telephone channel. The figure 3.10 (c) shows the MaxMagFreq do not have as many peaks as the speech-signal. Comparing figure 3.10 (a) and 3.10 (b) we can observe that the MaxMagFreq values of Music signal are lower than MaxMagFreq values of the speech signal, in fact speech is a higher frequency signal respect to music signal.

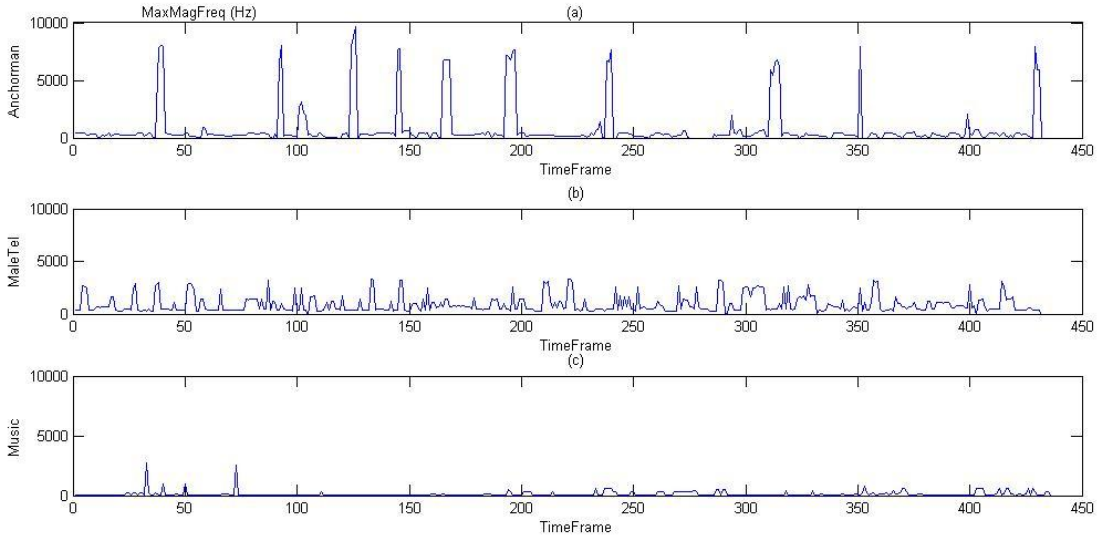


Figure 3.10: (a) MaxMagFreq of Anchorman speech signal of 10 s; (b) MaxMagFrq of Male Telephone speech signal of 10 s; (c) MaxMagFReq of Music signal of 10 s.

3.9 Spread

The spectral spread is the variation of the spectrum around its mean value. The spectral spread is computed from the second order moment.

The program computes and returns the Spread around the Centroid [58].

$$Spread(Y) = \frac{\sum_{i=0}^{N-1}(Centroid\ (Y) - x_i)^2 y_i}{\sum_{i=0}^{N-1} y_i} \quad (3.12)$$

The spread gives an idea on how much the distribution is not concentrated over the distribution centroid. Taking the array as a distribution and the values being probabilities, the spread would be the variance of such distribution.

Significant values are the following:

- For a full concentration on a single bin: 0.0
- For two balanced diracs on the extreme bins

$$Spread(BalancedDiracsDistribution) = \frac{N^2}{4}$$

- For a uniform distribution the spread it's:

$$Spread(UniformDistribution) = \frac{(N-1)(N+1)}{12}$$

Singularities and solution:

- How we can see in the listing 3.7 when $\sum y_i$ is less than 1e-14 it returns the uniform distribution formula above.
- Centroid NaN silence NaN is solved inside GetCentroid
- When Centroid is less than 0.2, 0.2 is taken as the centroid value.

```
U GetSpread()
{
        const unsigned N = mData->Size();
        const Array<T> & data = *mData;
        const U centroid = GetCentroid();

        // Compute spectrum variance around centroid frequency
        TData variance = 0;
        TData sumMags  = 0;
        for (unsigned i=0; i<N; i++)
        {
                U centroidDistance = i - centroid;
                centroidDistance *= centroidDistance;
                variance += centroidDistance * data[i];
                sumMags  += data[i];
        }
        // NaN solving: Silence is like a plain distribution
        if (sumMags < 1e-14) return U(N+1) * (N-1) / 12;

        return variance / sumMags;
}
```

Listing 3.7: Definition at line 298 of file Stats.hxx.

The spectral spread describes whether the spectrum is widely spread out or concentrated around its centroid. Noise-like sounds should have a wider spectrum compared to voiced

sounds such as speech [51]. The Figure 3.11 shows that this descriptor potentially enables discriminating between pure-tone and noise-like sounds, in fact, in this figure we can see that the song spectral spread is lower than the spectral spread of the song signal.

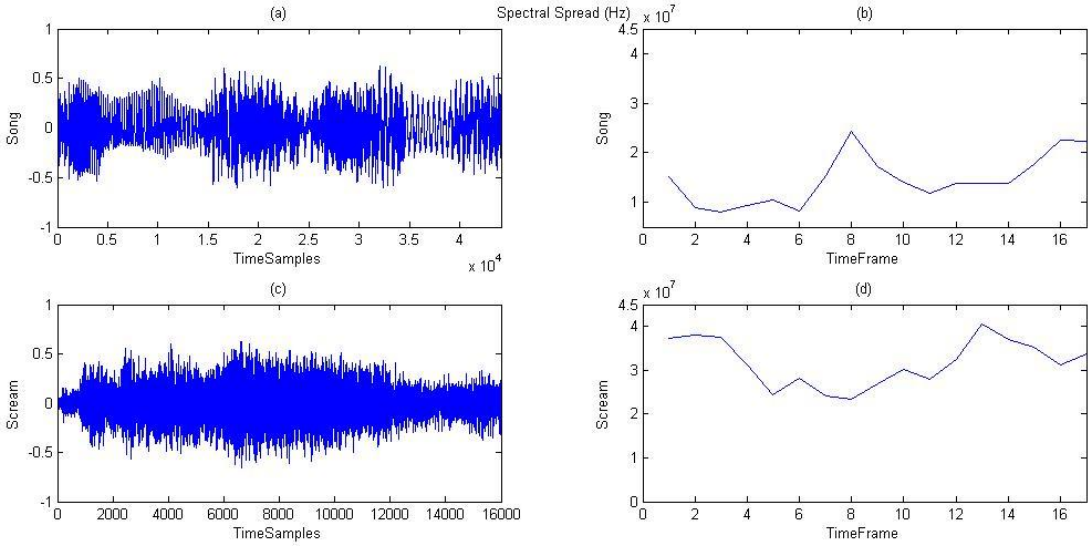


Figure 3.11: (a) 44,100 kHz Song signal of 1 s; (b) Spectral Spread of the Song signal; (c) 16 kHz Scream signal of 1 s; (d) Spectral Spread of the Scream signal

3.10 Magnitude Skewness

The Skewness of a distribution gives an idea of the asymmetry of the variance of the values [58].

$$Skewness(X) = \frac{\sum((x_i - Mean\,(X))^3)}{(\sum(x_i - Mean\,(X))^2)^{\frac{3}{2}}} \quad (3.13)$$

Typical values:

- This function returns greater positive values when there are more extreme values above the median than below.
- Returns negative values when there are more extreme values below the median than above.
- Returns zero when the distribution of the x_i values around the Median is equilibrated.

Singularities and solutions:

- Constant functions: Currently returns NaN but, in the future, it should return 0 because it can be considered an equilibrated function.

In the listing 3.8 is reported the definition in Stats.hxx of the function that computes the Magnitude Skewness.

```
U GetSkew()
{
        return mSkew(*mData,mStdDev,GetCentralMomentFunctor<3>(),true);
}
```

Listing 3.8: Definition at line 355 of file Stats.hxx

As we can see in the Figure 3.12 this feature has a higher value for speech than for music in fact the speech signal has more energy above the median, whereas the music has a spectrogram more equilibrated around the median.

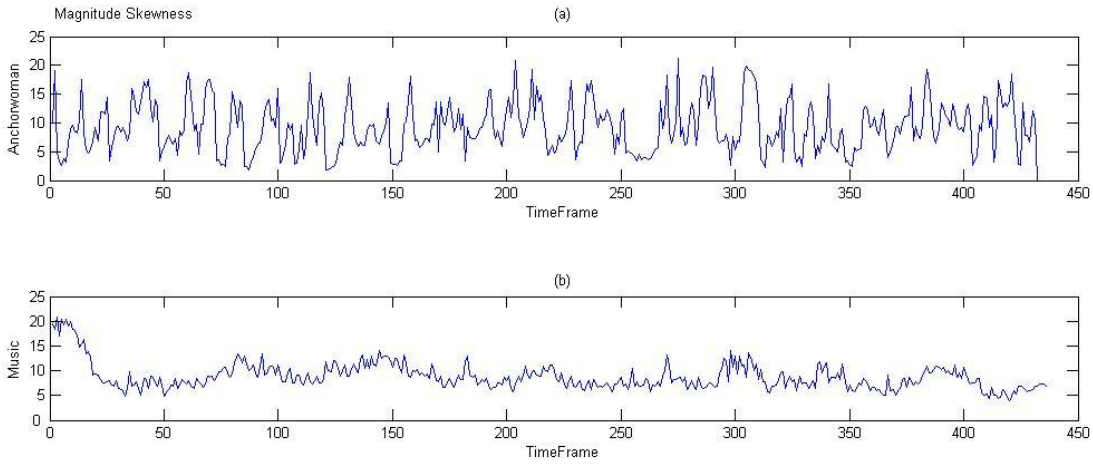


Figure 3.12: (a) Spectral magnitude skewness of anchorwoman speech signal of 10 s; (b) Spectral magnitude skewness of music signal of 10 s

3.11 Spectral Roll-Off

The spectral roll-off point is the frequency value so that the 85% of the spectral energy is contained below it [58, 53].

For silences this is 0Hz. Measured in Hz.

$$Rolloff \Big/ \sum_{f=0}^{RollOff} a_f^2 = 0.85 \times \sum_{f=0}^{Spectral\ \ Range} a_f^2 \qquad (3.14)$$

Other studies have used roll-off frequencies computed with other ratios, e.g. 92% in [54] or 95% in [55].

The roll-off is a measure of spectral shape useful for distinguishing voiced from unvoiced speech. This is confirmed by the Figure 3.13 and 3.14. We can notice, comparing Figure 3.14

(d) and 3.14 (c), that we have the peaks in the roll-off in correspondence of the peaks of the spectrum at the same time frame.

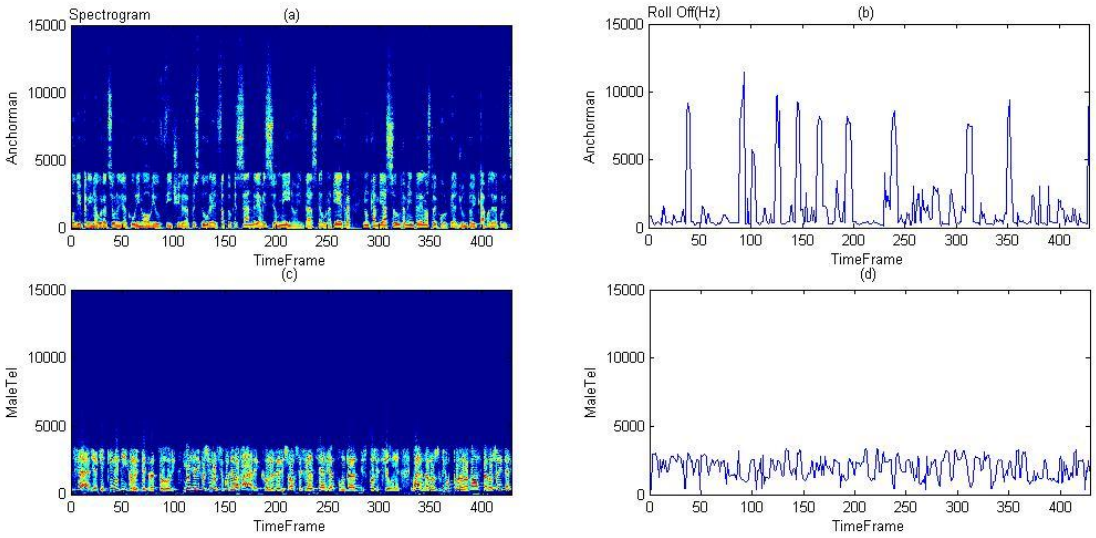


Figure 3.13: (a) Top 50dB of the anchorman speech signal spectrogram of 10 s; (b) Spectral Roll-Off of the anchorman speech signal; (c) Top 50dB of the male telephone speech signal spectrogram of 10 s; (d) Spectral Roll-Off of the male telephone speech signal.

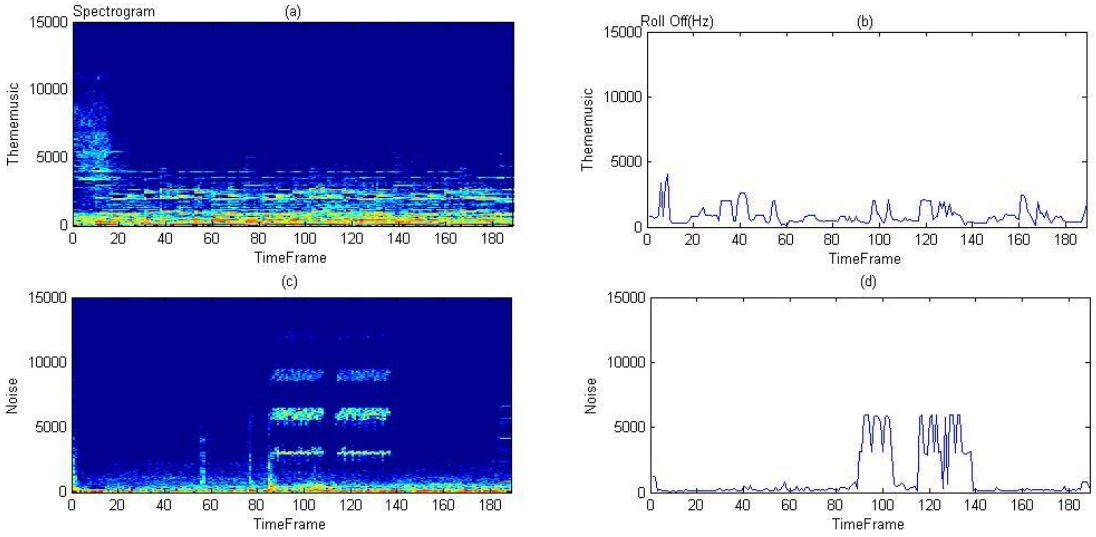


Figure 3.14: (a) Top 50dB of the theme music signal spectrogram of 4 s; (b) Spectral Roll-Off of the theme music signal; (c) Top 50dB of the noise signal spectrogram of 4 s; (d) Spectral Roll-Off of the noise signal

3.12 Spectral Slope

The spectral slope represents the amount of decreasing of the spectral magnitude [58].

The slope gives an idea of the mean pendent on the array:

- Less than zero means that is decreasing
- More than zero means that is increasing
- Zero means that any tendency is the dominant

The Slope is defined as:

$$\frac{1}{\sum x_i} \frac{N\sum i x_i - \sum i \sum x_i}{N\sum i^2 - (\sum i)^2} \qquad (3.15)$$

The formula (3.15) can be transform into one depending on the Centroid which is already calculated in order to obtain other stats:

$$6\frac{2Centroid\ -N+1}{N(N-1)(N+1)} \qquad (3.16)$$

The slope is relative to the array position index. If you want to give to the array position a dimensional meaning, (p.e. frequency or time) then you should divide by the gap between array positions. For example GetSlope/BinFreq for a FFT or GetSlope*SampleRate for an audio.

In the listing 3.9 is reported the definition in Stats.hxx of the function that computes the Spectral Roll-Off.

```
U GetSlope()
{
        // TODO: Sums where Y is used can be taken from Mean and Centroid

        const TSize size  = mData->Size();

        // \sum^(i=0)_(N-1)(x_i)
        const TData sumY = GetMean()*size;
        // \sum^(i=0)_(N-1)(i x_i)
        const TData sumXY = GetCentroid()*GetMean()*size;
        // \sum^(i=0)_(N-1)(i)
        const TData sumX = (size-1)*size/2.0;
        // \sum^(i=0)_(N-1)(i^2)
        const TData sumXX = (size-1)*(size)*(size+size-1)/6.0;

        //TData num = size*sumXY - sumX*sumY;
        // = size Centroid Mean size - (size-1)(size)(size)Mean/2
        // = size^2 mean (Centroid - (size-1)/2)
        //num = size*size*GetMean()*(GetCentroid()-(size-1)/2.0);

        // size*sumXX - sumX*sumX =
        // = size (size-1) size (size+size-1)/6 - (size-1)^2(size)^2/4
        // = size^2 ( (size-1)(size+size-1)/6 - (size-1)^2/4 )
        // = size^2 (size-1)( (size+size-1)/6 - (size-1)/4 )
        // = size^2 (size-1)( (4*size-2) - (3*size-3) )/12
        // = size^2 (size-1) (size+1)/12

        //TData denum = (size*sumXX - sumX*sumX)*sumY;
        // = size mean size^2 (size-1) (size+1) / 12
        // = size^3 mean (size-1) (size+1) / 12
        //denum = size*size*size * GetMean() * (size-1) * (size+1) /12.0;

        // return num/denum;
        // = size^2 mean (Centroid - (size-1)/2) / (size^3 mean (size-1) (size+1) / 12)
        // = (Centroid - (size-1)/2) / (size (size-1) (size+1) /12)
        // = ( 12*centroid - 6*size + 6 ) / ( size (size-1) (size+1) )
        // = 6 (2*centroid - size + 1)) / ( size (size-1) (size+1) )
        return 6*(2*GetCentroid() - size + 1) / (size * (size-1) * (size+1));

}
```

Listing 3.9: Definition at line 49r of file Stats.hxx

3.13 High Frequency Content

This descriptor is defined as the sum of the squared spectrum magnitude multiplied by the wave number of the bin. It is pretty similar to the derivative of the energy, or a high pass filter, which gives higher values for high frequency content. It is very useful in locating high frequencies. This can be confirmed comparing the Figure 3.15 (a) and (b).

This feature can be utilized also to distinguish male and female, in fact as shows the Figure 3.16 the anchorman high frequency content is lower than high frequency content of anchorwoman. In fact anchorwoman speech signal, respect to the anchorman signal, has more energy at high frequency.

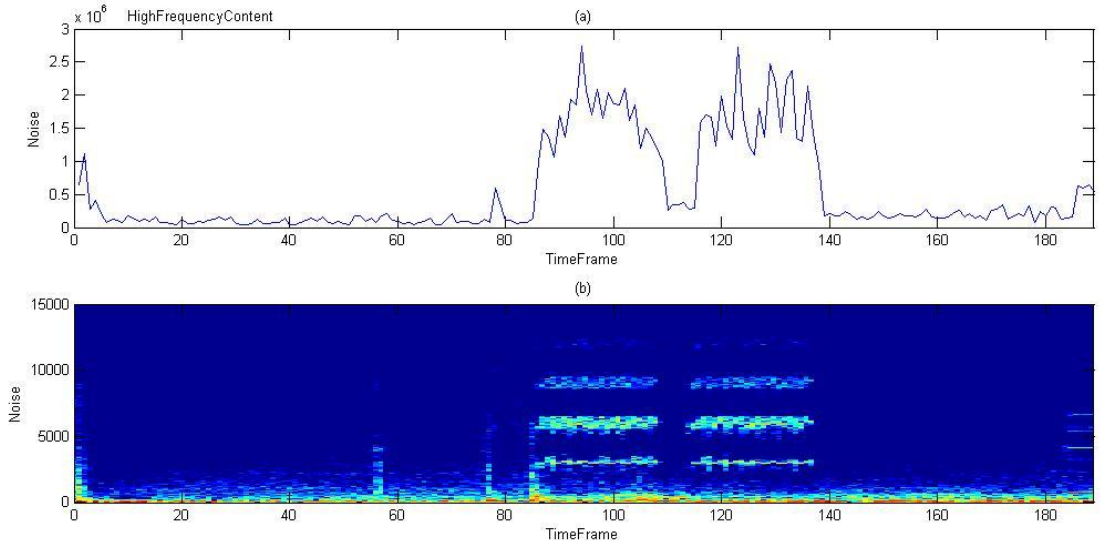


Figure 3.15: (a) High frequency content of the noise signal; (b) Top 50dB of the noise signal spectrogram of 4 s

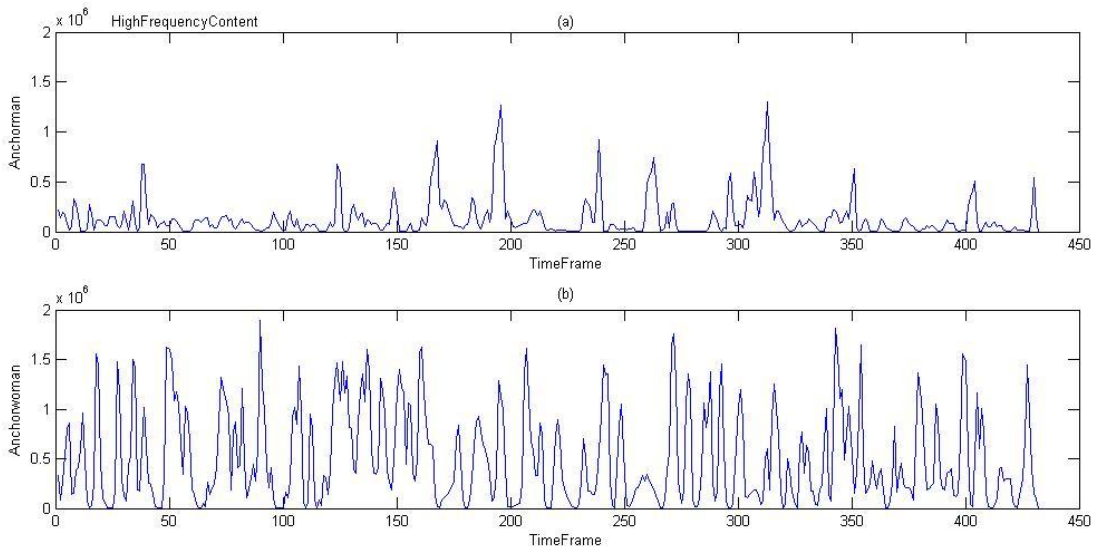


Figure 3.16: (a) High frequency content of anchorman speech signal of 10 s; (b) High frequency content of anchorwoman speech signal of 10 s.

3.14 Cepstrum

Initially cepstral analysis was introduced in conjunction with speech recognition, as a way to model the human articulatory system as described below. Later the features have shown useful in speaker recognition as well as other audio applications such as audio classification and music summarization.

As described in [56] the speech signal is composed of a quickly varying part e (n) (excitation sequence) convolved with a slowly varying part θ (n) (vocal system impulse response):

$$s(n) = e(n) * \theta(n) \qquad (3.17)$$

The convolution makes it difficult to separate the two parts, therefore the cepstrum is introduced. The cepstrum is defined as:

$$c_s(n) = IDFT\{\log|DFT\{s(n)\}|\} \qquad (3.18)$$

Where DFT is the Discrete Fourier Transform and IDFT is the Inverse Discrete Fourier Transform. By moving the signal to the frequency-domain the convolution becomes a multiplication:

$$S(k) = E(k)\,\Theta(k) \qquad (3.19)$$

Further, by taking the logarithm of the spectral magnitude the multiplication becomes an addition:

$$\begin{aligned} \log|S(k)| &= \log|E(k)\Theta(k)| \\ &= \log|E(k)| + \log|\Theta(k)| \\ &= C_e(k) + C_\theta(k) \end{aligned} \qquad (3.20)$$

IDFT is linear and therefore works individually on the two components:

$$\begin{aligned} c_s(n) &= IDFT\{C_e(k) + C_\theta(k)\} \\ &= IDFT\{C_E(k)\} + IDFT\{C_\theta(k)\} \\ &= c_e(n) + c_\theta(n) \end{aligned} \qquad (3.21)$$

The domain of the signal $c_s(n)$ is called the frequency-domain. Figure 3.17 shows the speech signal transformation process.

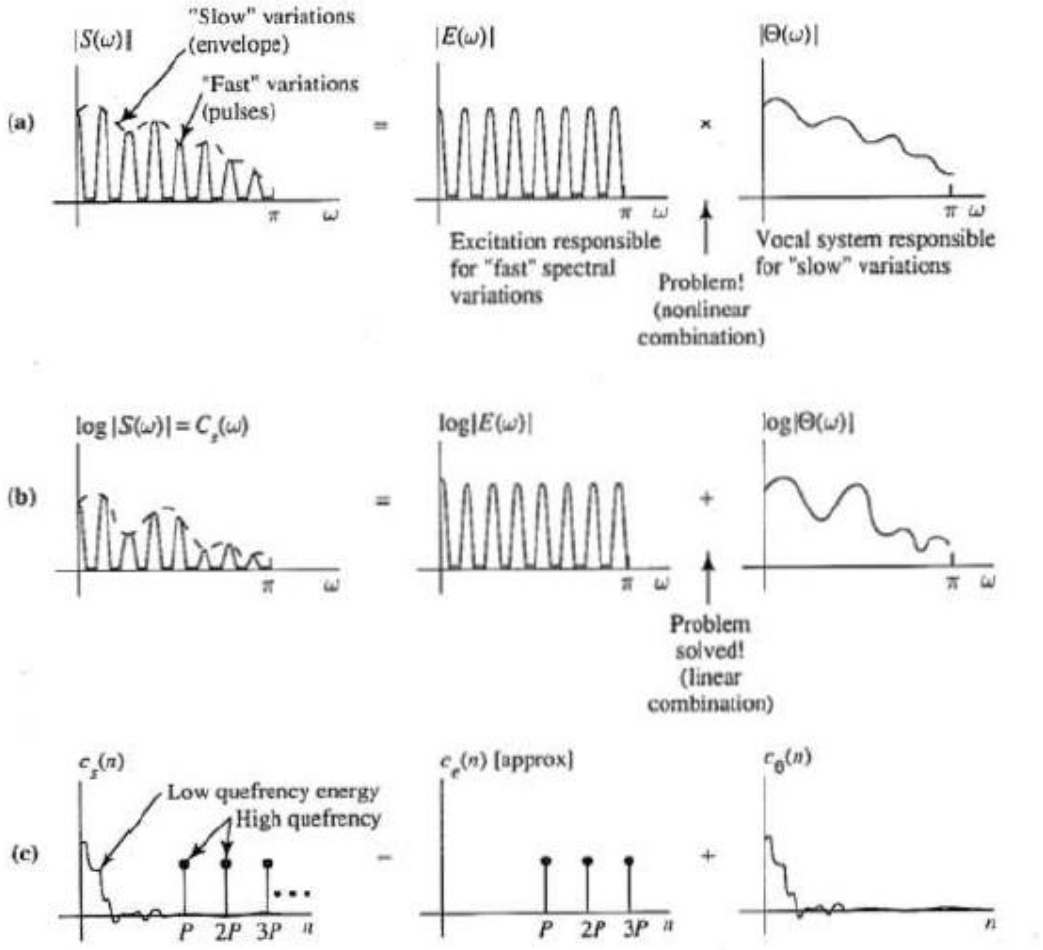


Figure 3.17: shows how the signal is composed of a slowly varying envelopePart convolved with quickly varying excitation part. Figure taken from [56].

3.14.1 Mel-frequency Cepstral Coefficients

The MFCCs are the most popular cepstrum-based audio features, even though there exist other types of cepstral coefficients [57], like the *linear prediction cepstrum coefficient* (LPCC), extracted from the linear prediction coefficient (LPC). MFCC is a perceptually motivated representation defined as the cepstrum of a windowed short-time signal. A non-linear *mel*-frequency scale is used, which approximates the behaviour of the auditory system. The mel is a unit of pitch (i.e. the subjective impression of frequency).

The *mel scale* is a scale of pitches judged by listeners to be equal in distance one from another. The reference point between this scale and normal frequency measurement is defined by equating a 1000 Hz tone, 40 dB above the listener's threshold, with a pitch of 1000 mels. Below about 500 Hz the mel and hertz scales coincide; above that, larger and larger intervals

are judged by listeners to produce equal pitch increments. The MFCCs are based on the extraction of the signal energy within critical frequency bands by means of a series of triangular filters whose centre frequencies are spaced according to the mel scale.

The nonlinear mel scale accounts for the mechanisms of human frequency perception, which is more selective in the lower frequencies than in the higher ones.

The extraction of MFCC vectors is depicted in Figure 3.18.

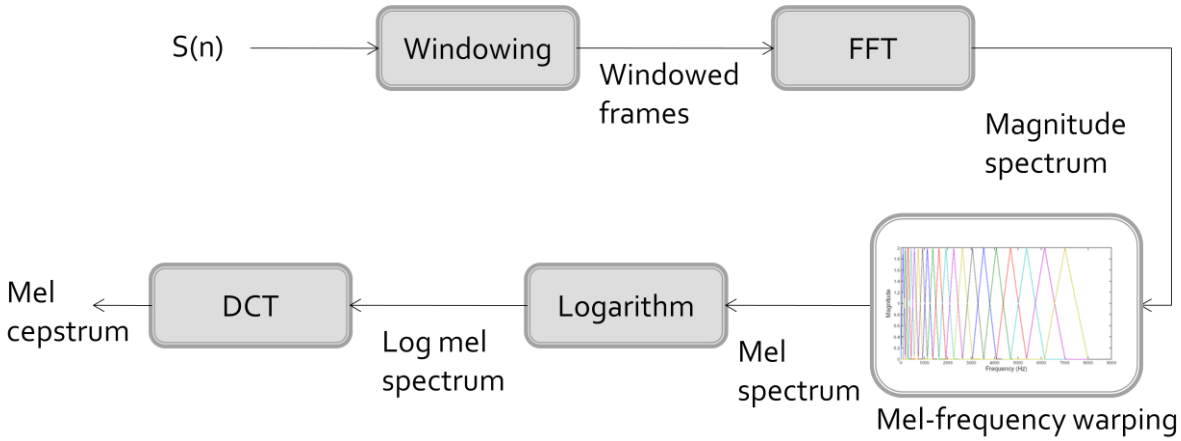


Figure 3.18: MFCC Calculation

The input signal s(n) is first divided into overlapping frames of N_w samples.

In order to minimize the signal discontinuities at the borders of each frame a windowing function is used, such as the Hamming function defined as:

$$w(n) = \frac{1}{2}\left\{1 - \cos\left[\frac{2\pi}{N_w}\left(n + \frac{1}{2}\right)\right]\right\} \qquad (0 \le n \le N_w - 1) \qquad (3.22)$$

An FFT is applied to each frame and the absolute value is taken to obtain the magnitude spectrum. The spectrum is then processed by a mel-filter bank; the log-energy of the spectrum is measured within the pass-band of each filter, resulting in a reduced representation of the spectrum. The cepstral coefficients are finally obtained through a Discrete Cosine Transform (DCT) of the reduced log-energy spectrum.

In this project the ClamExtractorExample employs a filter bank of 20 filters and a high frequency cut-off of 11250Hz in order to compute a 20 MFCC for each frame.

Figure 3.20 (b) shows 20 MFCCs for 10 seconds of speech and figure 3.20(a) shows 20 MFCCs for 10 seconds of music. The speech shows a large variation in the coefficients. This is due to the altering voiced/unvoiced/silence structure in speech. These different structures have different spectral characteristics, which are reflected in the MFCCs. The MFCCs for the music seems to be much more structured and do not show the same variation in the coefficients.

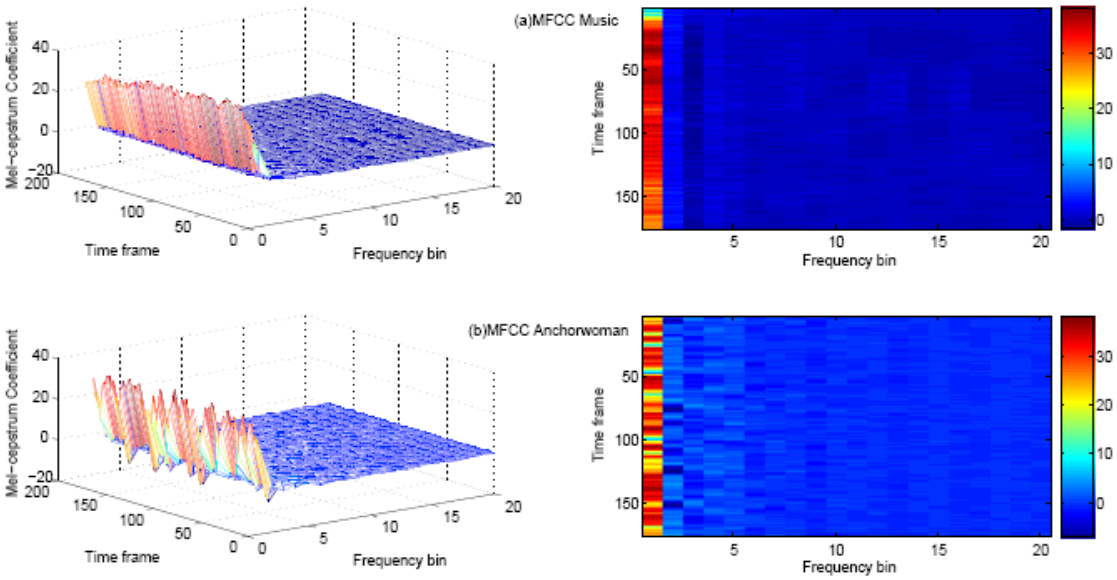


Figure 3.20: (a) MFCC of 10 s of music signal; (b) MFCC of 10 s of anchorwoman speech signal.

Figure 3.21 (b) shows 20 MFCCs for 10 seconds of anchorman speech signal and figure 3.21 (a) shows 20 MFCCs for 10 seconds of male telephone signal. As we can see in the figure 3.8(a) and 3.8(b) these speech signals have different spectral characteristics, which are reflected in the MFCCs.

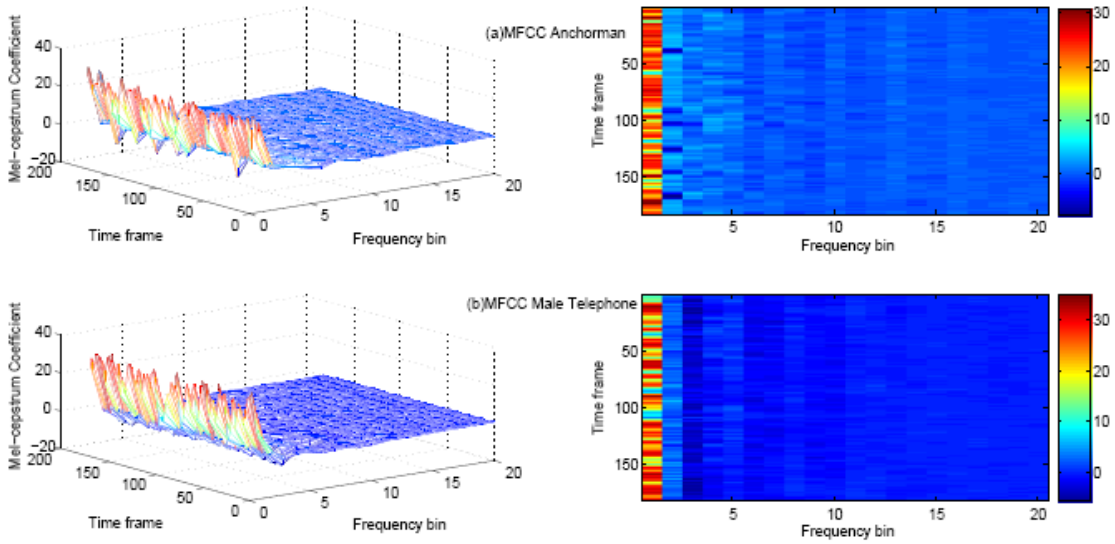


Figure 3.21: (a) MFCC of 10 s of anchorman speech signal; (b) MFCC of 10 s of male telephone speech signal.

CHAPTER 4

AUDIO DATABASE AND MANUAL SEGMENTATION

For the audio segmentation a data base of speech and music was collected. The speech contains a wide range of different radio newscast. The samples are chosen to reflect many different kinds of typical speech, ranging from anchor speakers in almost perfect conditions to conversations between multiple speakers. Also, narrow-band telephone interviews are present.

Some of the speech clips contain speech from reporters speaking from noisy environments.

15 Giornali Radio RAI (GRR) was recorded from the RAI web page http://www.radio.rai.it/grr/ with Freecoder, a software tools for recording internet audio.

The sources of the clips are listed in table 4.1. The clips were collected in November and December 2007. The GRR are chosen in order to have the same anchor for the same GRR.

Number	GR	Date	Time	Duration (min)	Anchor
1	RAI GR1	10/11/2007	13.00	30	Anchorman1: G. Trevisi Anchorman2: A. Biciocchi
2	RAI GR1	11/11/2007	13.00	30	Anchorman1: G. Trevisi Anchorman2: A. Biciocchi
3	RAI GR1	17/11/2007	13.00	30	Anchorman1: G. Trevisi Anchorman2: A. Biciocchi
4	RAI GR1	26/11/2007	13.00	30	Anchorman1: G. Trevisi Anchorman2: A. Biciocchi
5	RAI GR1	02/12/2007	13.00	30	Anchorman1: G. Trevisi Anchorman2: A. Biciocchi
6	RAI GR2	05/11/2007	19.30	28	Anchorwoman1: L. Scardini Anchorwoman2: V.Montanari
7	RAI GR2	08/11/2007	19.30	28	Anchorwoman1: L. Scardini Anchorwoman2: A. Fiori
8	RAI GR2	12/11/2007	19.30	28	Anchorwoman1: L. Scardini Anchorwoman2: A. Fiori
9	RAI GR2	14/11/2007	19.30	28	Anchorwoman1: L. Scardini Anchorwoman2: A. Fiori
10	RAI GR2	21/11/2007	19.30	28	Anchorwoman1: L. Scardini Anchorwoman2: A. Fiori
11	RAI GR3	06/11/2007	8.45	20	Anchorwoman: A. Pizzato
12	RAI GR3	07/11/2007	8.45	20	Anchorwoman: A. Pizzato
13	RAI GR3	08/11/2007	8.45	20	Anchorwoman: A. Pizzato
14	RAI GR3	09/11/2007	8.45	20	Anchorwoman: A. Pizzato
15	RAI GR3	10/11/2007	8.45	20	Anchorwoman: A. Pizzato

Table 4.1: List of the GR wav files present in the data base

The GRRs are then segmented manually, using WaveSurfer [59], into different classes.

4.1 WaveSurfer Tool for the Manual Segmentation

WaveSurfer is an Open Source tool for sound visualization and manipulation [59]. WaveSurfer has a simple and logical user interface that provides functionality in an intuitive way and which can be adapted to different tasks. It can be used as a stand-alone tool for a wide range of tasks in speech research and education. Typical applications are speech/sound analysis and sound annotation/transcription. WaveSurfer can also serve as a platform for more advanced/specialized applications. This is accomplished either through extending the WaveSurfer application with new custom plug-ins or by embedding WaveSurfer visualization components in other applications.

As shown in Figure 4.1 when WaveSurfer is first started, it contains an empty sound. You can load a sound file from disk.

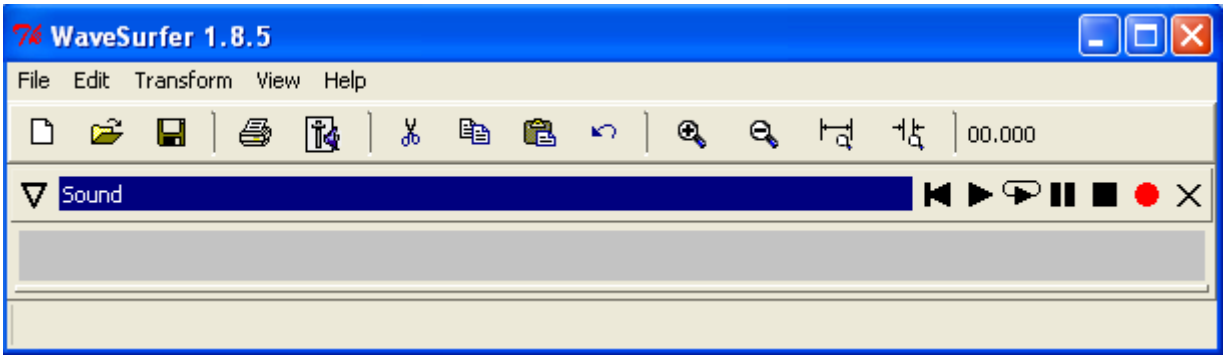


Figure 4.1: WaveSurfer Interface

To allow for different sophisticated tasks WaveSurfer gives the possibility of adding panes. A pane is a window stacked on top of the WaveBar that can contain for example a waveform, a spectrogram, a pitch-curve, a time axis or a transcription or something else. Unlike the WaveBar, a pane will not necessarily display the whole sound. Rather it will display a portion of the sound that is specified in the WaveBar. Think of the WaveBar as an overview and the pane as a variable magnifying glass.

WaveSurfer can read a number of sound file formats including WAV, AU, AIFF, MP3, CSL, and SD. It can also save files in several formats, including WAV, AU, and AIFF. There are separate plug-ins to handle Ogg/Vorbis and NIST/Sphere files. WaveSurfer can be used to visualize and analyze sound in several ways. The standard analysis plug-in can display Waveform, Spectrogram, Pitch, Power or Formant panes.

WaveSurfer has many facilities for transcribing sound files. Transcription is handled by a dedicated plug-in and it's associated pane type. In the Figure 4.2 is shown WaveSurfer interface when the the configuration Transcription is chosen.

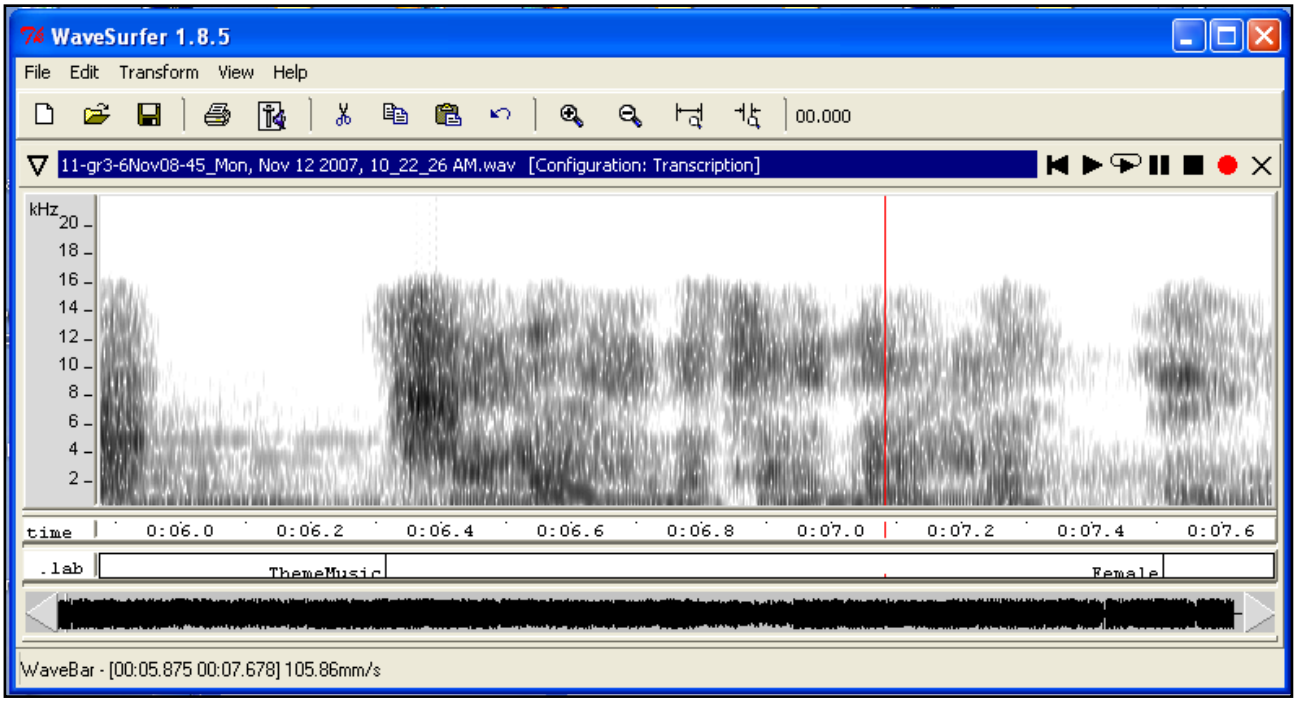


Figure 4.2: WaveSurfer interface when the transcription configuration is chosen.

The properties-dialog can be used to specify which label file that should be displayed in a transcription pane. Unicode characters are supported in order to keep the binary versions small. The transcription plug-in is used in combination with format handler plug-ins which handles the conversion between file formats and the internal format used by the transcription plug-in. The standard popup menu has additional entries for transcription panes. *Popup | Load Transcription* and *Popup | Save Transcription* are used to load and save transcription files.

In the Listing 4.1 is shown an example of a lab file that WaveSurfer has as output.

```
0.0000000 0.4425000 Silence
0.4425000 0.8650000 Female
0.8650000 1.1450000 Silence
1.1450000 1.3575000 Female
1.3575000 1.5150000 Silence
1.5150000 1.6700000 Female
1.6700000 1.9950000 Silence
1.9950000 2.2525000 Female
2.2525000 2.4200000 Silence
2.4200000 3.5900000 Female
3.5900000 4.1650000 Silence
4.1650000 6.3150000 ThemeMusic
6.3150000 7.5100000 Female
7.5100000 11.3925000 Male+Music
```

Listing 4.1: Piece of a lab file

WaveSurfer allows splitting sound on labels, but every label should have a different name so we wrote a matlab code that modify the lab file, produced with WaveSurfer, adding an increasing number to each name label.

4.2 Data Base Description

In order to have an idea of the amount of the seconds of each class I wrote a matlab code that reads the file lab of WaveSurfer and analysing it returns for each class the total seconds.
In the following tables are reported the amount of seconds for each class.

CLASSES	DURATION (h:m:s:ms)
Speech	4:48:27:231
Music	0:16:12:139
Silence	0:11:23:472
Other	1:17:9:121
Total	6:33:11:963

Table 4.2: Duration of the classes: Speech, Music, Silence, Other

CLASSES	DURATION (h:m:s:ms)
Male	3:4:22:348
Female	1:44:4:883
Total	4:48:27:231

Table 4.3: Duration of the subclasses Male and Female of the class Speech

CLASSES	DURATION (h:m:s:ms)
Anchorman	0:20:15:627
Male	1:37:7:879
MaleSpot	0:5:20:382
MaleTel	1:1:38:459
Total	3:4:22:348

Table 4.4: Duration of the subclasses: Anchorman, Male, MaleSpot, MaleTel of the class Male

CLASSES	DURATION (h:m:s:ms)
Anchorwoman	0:40:21:574
Female	0:56:51:503
FemaleSpot	0:3:41:117
FemaleTel	0:3:10:688
Total	1:44:4:883

Table 4.5 Total duration of the wav files of the subclasses: Anchorwoman, Female, FemaleSpot, FemaleTel of the class Female

CLASSES	DURATION (h:m:s:ms)
Anchorman1	0:10:8:774
Anchorman2	0:10:6:853
Total	0:20:15:627

Table 4.6 Total duration of the wav files of the subclasses: Anchorman1, Anchorman2 of the class Anchorman.

CLASSES	DURATION (h:m:s:ms)
Anchorwoman1	0:13:36:137
Anchorwoman2	0:8:54:940
Anchorwoman3	0:1:38:625
Anchorwoman	0:16:11:873
Total	0:40:21:574

Table 4.7 Total duration of the wav files of the subclasses: Anchoworman1, Anchoworman2, Anchorwoman3, and Anchorwoman of the class Anchorwoman.

CLASSES	DURATIONS (h:m:s:ms)
OtherMusic	0:3:29:218
ThemeMusic	0.12:42:921
Total	0:16:12:139

Table 4.8: Total duration of the wav files of the subclasses: OtherMusic, ThemeMusic of the class Music.

CLASSES	DURATIONS (h:m:s:ms)
Speech+Music	0:56:37:544
Speech+Music	0:18:20:299
Environmental	0:2:11:278
Total	1:17:9:121

Table 4.9: Total duration of the wav file of the subclasses: Speech+Music, Speech+Noise, Environmental of the class Other.

In order to obtain a compact view of the available audio materials for each class the matlab code also returns the pie diagrams, reported in figure 4.4 and 4.11, that represent statistical information of the data base created.

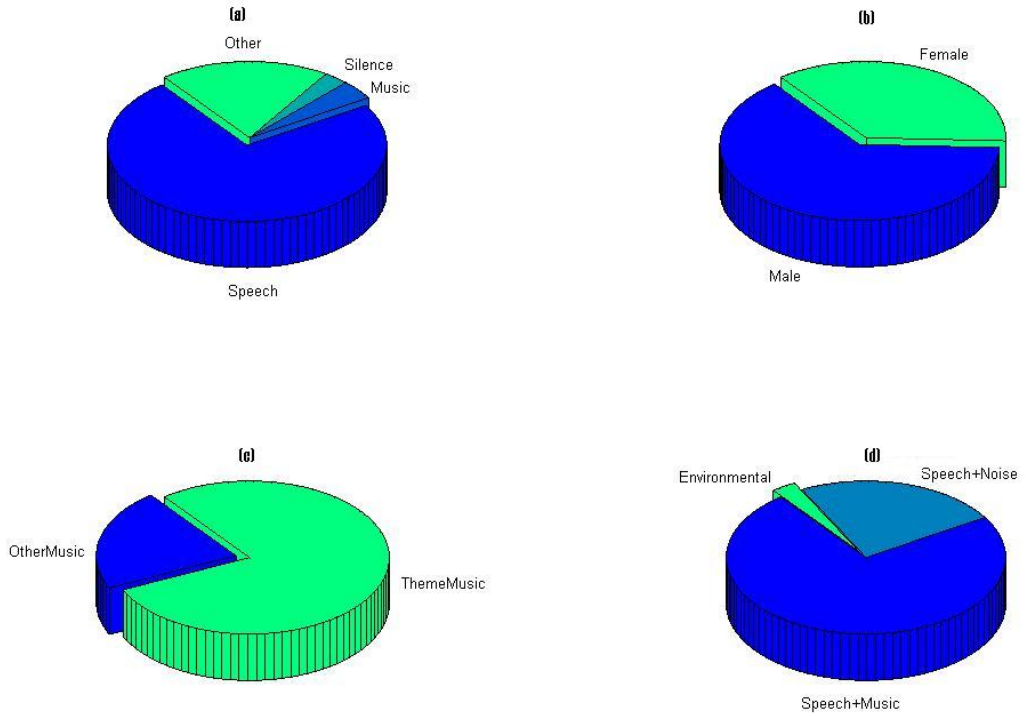


Figure 4.3: a) Pie diagram of the classes: Speech, Music, Other, Silence; b) Pie diagram of the subclasses Male, Female of the class Speech; c) Pie diagram of the subclasses OtherMusic, ThemeMusic of the class Music; d) Pie diagram of the subclasses Speech+Noise ,Speech+Music, Enviromental of the class Other.

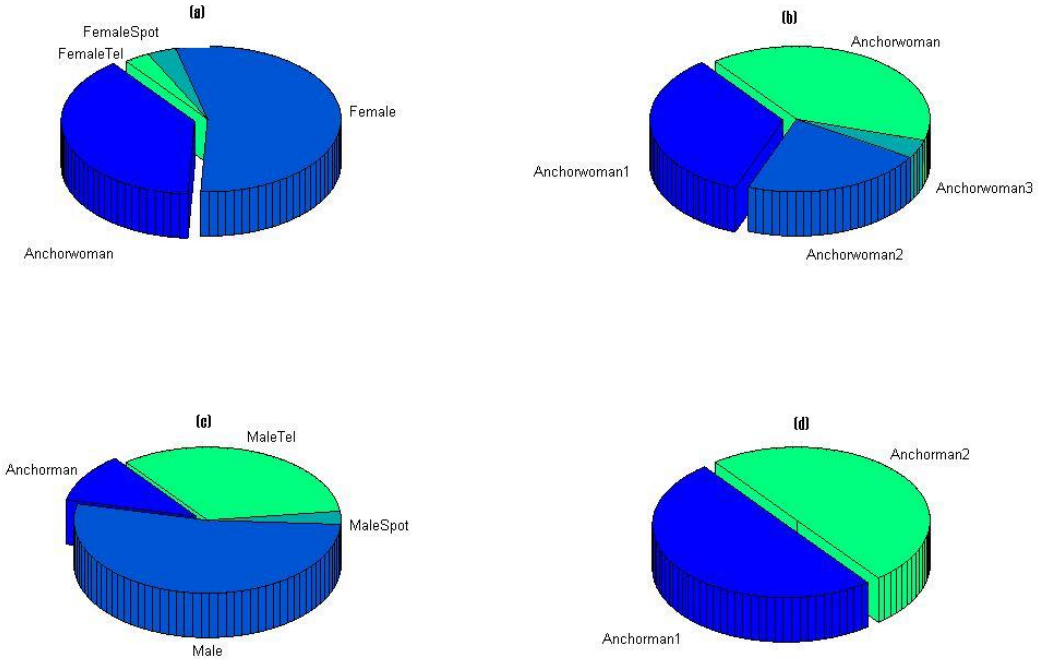


Figure 4.4: a) Pie diagram of the subclasses: Anchorwoman, Female, FemaleSpot, FemaleTel of the class Female; b) Pie diagram of the subclasses Anchorwoman1, Anchorwoman2, Anchorwoman3, Anchorwoman of the class Anchorwoman; c) Pie diagram of the subclasses Anchorman, Male ,MaleSpot, MaleTel of the class Male; d) Pie diagram of the subclasses Anchorman1, Anchorman2, of the class Anchorman.

After the manual segmentation with WaveSurfer I implemented a matlab code that converts the file lab in an XML document MPEG-7 compliant. In the following sections I describe the MPEG-7 Multimedia Descriptor Schemes (MDSs), which are metadata structures for describing and annotating audio-visual (AV) content. The DSs provide a standardized way of describing in XML the important concepts related to AV content description and content management in order to facilitate searching, indexing, filtering, and access.

4.3 MPEG-7 Multimedia Description Schemes

MPEG-7 Multimedia Description Schemes are defined using the MPEG-7 Description Definition Language (DDL), which is based on the XML Schema Language, and are instantiated as documents or streams. The resulting descriptions can be expressed in a textual form (i.e., human readable XML for editing, searching, filtering) or compressed binary form (i.e., for storage or transmission). The goal of the MPEG-7 standard is to allow interoperable searching, indexing, filtering and access of audio-visual (AV) content by enabling interoperability among devices and applications that deal with AV content description. MPEG-7 describes specific features of AV content as well as information related to AV content management. Overall, the standard specifies four types of normative elements: Descriptors, Description Schemes (DSs), a Description Definition Language (DDL), and coding schemes. The MPEG-7 Descriptors are designed primarily to describe low-level audio or visual features. On the other hand, the MPEG-7 DSs are designed primarily to describe higher-level AV features.

4.4 Organization of MDS tools

MPEG-7 Multimedia Description Scheme (MDS) comprises the set of Description Tools (Ds and DSs) dealing with multimedia entities. MDS contains, among others, which we can see in figure 4.5, the following areas; Basic Elements, Content Management and Content Description [60].

- *Basic Elements*: address specific needs of audiovisual content description, such as the description of time, persons, places and other textual annotation.
- *Content Management*: describes different aspects of creation and production of the process such as: title, creators, locations, dates and media information of the audiovisual content (storage media, coding format and compression) to adjust to different network environments.
- *Content Description*: describes the structure, segmentation of the content and semantics (entities, events, relationships) of the audiovisual content. Thus, it allows attaching audio,

video, annotation and content management to the multimedia segments, to depict them in detail.

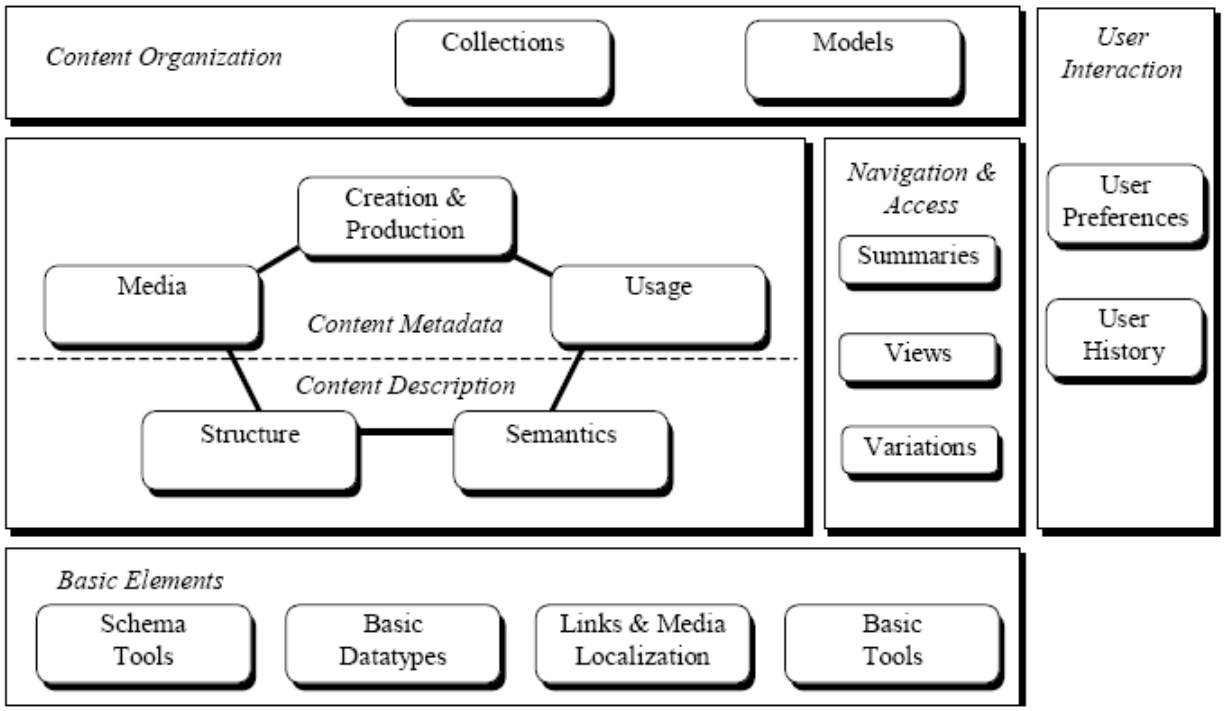


Figure 4.5: Overview of the MPEG-7 Multimedia DSs. Figure from [61]

As the target of this section of my thesis is obtain an XML MPEG-7 document, which describes for each GR all the segments manually compute, and their temporal location in the wav file, I'm interested only in Content Description and Basic Elements.

4.4.1 Content Description

The description schemes for content description describe the Structure (region, video frames and audio segments) and Semantics (objects, events and abstract notions). The functionality of each of these classes of description schemes is given as follows:

- *Structural aspects*: description schemes describe the multimedia content from the viewpoint of its structure. The description is built around the notion of Segment description scheme that represents a spatial, temporal or spatial temporal portion of the multimedia content. The Segment description scheme can be organized into a hierarchical structure to produce a Table of Content for accessing or an Index for searching the multimedia content. The Segments can be further described on the basis of perceptual features using MPEG-7 Descriptors for color, texture, shape, motion, audio features and so forth, as well as semantic information using Textual Annotations.

- *Conceptual aspects*: description schemes describe the multimedia content from the viewpoint of real-world semantics and conceptual notions. The Semantic description schemes involve entities such as objects, events, abstract concepts and relationship. The Segment description schemes and Semantic description schemes are related by a set of links that allows the multimedia content to be described on the basis of both content structure and semantics together.

In this project I am interesting only to give a structural description of the segments and so I used the Segment description schemes as we can see in the listing 4.2

```
- <Audio xsi:type="AudioSegmentType">
  + <MediaTime>
  - <TemporalDecomposition>
    + <AudioSegment>
    + <AudioSegment>
    + <AudioSegment>
    [...]
    + <AudioSegment>
    + <AudioSegment>
    </TemporalDecomposition>
  </Audio>
```

Listing 4.2: AudioSegmentType

4.4.2 Basic Elements

As we can see from figure 4.5 the set of description tools called the MPEG-7 *Basic Elements* is subdivided into three groups: Schema tools, Basic datatypes, Link and localization tools, and Basic tools.

- *Basic data types*, which represent mathematical constructs useful for multimedia description, such as matrices and vectors.
- *Linking and localization tools*, which are used to specify references within description, to link MPEG-7 description to media, to identify and locate media and to describe time.
- *Basic tools* that address common aspects of multimedia content description. This includes graphs for structuring complex multimedia content descriptions, text annotations, descriptions of people and places, specifications of effective response and description ordering.
- *Schema tools* compared with other basic elements, Schema tools have a different functionality because they not target the description of the content but are used to create valid descriptions and to manage them.

In the table 4.10 I report the four groups with their description tool

Schema tools	*Basic datatypes*	*Link & localization tools*	*Basic tools*
Base types	Integer, Real	References	Graphs & relations
Root element	Matrices, Vectors	Media Locators	Textual annotation
Top-level tools	Region, Country	Time	Classification schemes
Multimedia content entity tools	Currency		Terms
Package tool			Agents
Description metadata tool			Places
			Affective description
			Ordering key

Table 4.10: Overview of the MPEG-7 Basic Elements

4.5 Automatic generation of an XML Document for the description of the Segments

To create MPEG-7 descriptions of any multimedia content, the first requirement is to build a wrapper for the description using the *Schema Tools,* which should have the header information reported in the listing 4.3.

```
- <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001 Mpeg7-2001.xsd">
   [...]
  </Mpeg7>
```

Listing 4.3: Root element

The root type provides metadata about the description as well as information that is common to the description, such as the language of the text and the convention for specifying time. The root element provides a choice of elements for creating either a complete description or a description unit, which are defined as follows:

- *Complete Description*: describes multimedia content using the top-level types. For example, the description of an image is a complete description.
- *Description Unit*: describes an instance of a D, DS, or header. A description unit can be used to represent partial information from a complete description. For example, the description of a shape or color is a description unit.

In this thesis as we are interested to the description of the entire GRR I choose the complete description that describes multimedia content using the top-level types of the *Schema Tools.* The top-level types are used in complete descriptions to describe multimedia content and metadata related to content management. Each top-level type contains the description tools that are relevant

for a particular description task. For describing multimedia content entities such as audio, we have to use the top-level type ContentEntityType, as shows in listing 4.4.

```
- <Description xsi:type="ContentEntityType">
  + <MultimediaContent xsi:type="AudioType">
  </Description>
```

Listing 4.4: ContentEntityType

In order to give an identifier to each segment and to describe its temporal location we have to use the References tool of the linking and localization tools.

The Reference data type provides three basic reference mechanisms:

- *Idref*: References a description element within the same description document. The target is identified by its ID attribute, which is unique within the description document.
- *xpath* : References a description element using a subset of XML Path Language(XPath). An XPath expression identifies a reference target by its position within the description tree.
- *href*: References a description or description element using a Uniform Resource Identifier (URI). Unlike the idref and xpath mechanism, href references can refer to an element in another description diocument.

In this work I have use the first mechanism as show in listing 4.5

```
<AudioSegment id="S178">
```

Listing 4.5: *Idref*

MPEG-7 can represent two different kinds of time: (1) media time, which is time measured or stored within the media and (2) generic time, which is time measured in the world. Both and world time use the same representation, except that the data types for world time also contain time zone (TZ) information [63]. Here I describe only the media time data types.

The MPEG-7 media time data types are compatible with time specification used in common multimedia formats such as MPEG-4 and are based on the ISO 8601 standard. The media time data types represent time periods using a start time point (mediaTimePoint data type) and a duration (mediaDuration data type). The mediaTimePoint data type uses the following syntax:

-YYY-MM-DThh:mm:ss:nFN

which includes the year (Y), month(M), days(D), a separator T, hours(h), minutes(m) and seconds (s), 1/N is a fraction of one second and n the number of those fractions.

For example the MediaTimePoint in the listing 4.6 indicates 19 minutes 59 seconds and 957 milliseconds.

```
<MediaTimePoint>T00:19:59:957F1000</MediaTimePoint>
```

Listing 4.6: MediaTimePoint

The mediaDuration data types uses the following format:

(-) PnDTnHnMnSnNnF

In this format, each part of the duration contains a count followed by a letter indicating the unit being counted: P is the separator indicating the start of a duration, days are indicated by D, T separates time from days, H indicates hours, M minutes, S seconds and the subsecond part uses the same representation as mediaTimePoint, with N indicating the counted fractions and F the franctions of one second.

For example the MediadDuration in the listing 4.7 indicates 2 seconds and 267 milliseconds.

```
<MediaDuration>PT0M2S267N1000F</MediaDuration>
```

Listing 4.7: MediaDuration

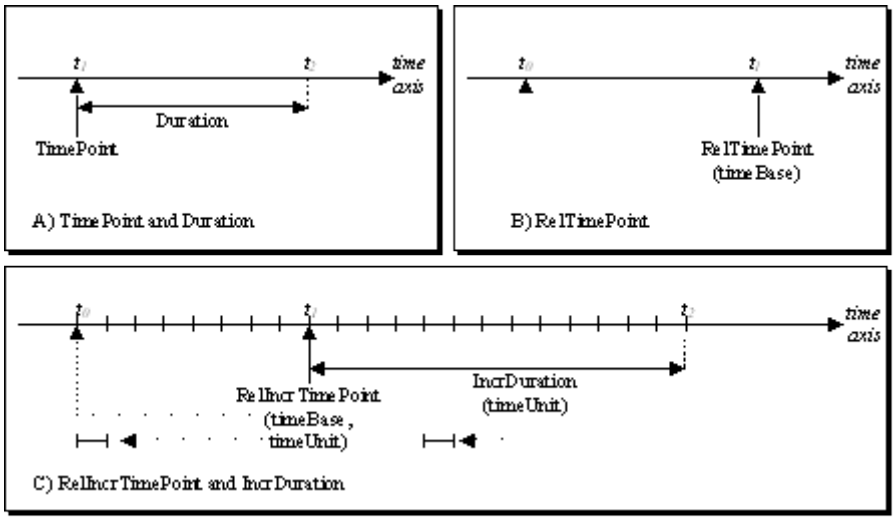


Figure 4.6: Kinds of media time representation

On the top of the mediaDuration and mediaTimePoint data types, MPEG-7 builds three kinds of media time representation:

- Simple Time: The basic representation of an absolute time point (figure 4.6a).

- Relative time: Specifies a media time point relative to a time base. Useful if a media segment, such as a story in a news sequence, is placed dynamically. To update the story's description, only the time base (t_0) needs to be changed (figure 4.6b).
- Incremental time: Specifies a time period by counting predefined time units (figure 4.6c)

In order to indicate to each segment its label I used the text annotation tool of the basic tool (see table 4.10), as show in listing 4.8.

```
<TextAnnotation>
  <FreeTextAnnotation>MaleSpot</FreeTextAnnotation>
</TextAnnotation>
```

Listing 4.8: TextAnnotation

4.6 Example of a XML Document for the Description of the Segments

As I said earlier we implemented a matlab code that converts the lab files into a XML document MPEG-7 compliant.

In the code we employed an XML utility in order to create the node of the XML document. In the listing 4.2 we reported the XML document created from the lab file of a GR3.

```
<?xml version="1.0" encoding="utf-8" ?>
- <Mpeg7 xmlns="urn:mpeg:mpeg7:schema:2001" xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:mpeg:mpeg7:schema:2001 Mpeg7-2001.xsd">
  - <Description xsi:type="ContentEntityType">
    - <MultimediaContent xsi:type="AudioType">
      - <Audio xsi:type="AudioSegmentType">
        - <MediaTime>
            <MediaTimePoint>T00:00:00</MediaTimePoint>
            <MediaDuration>PT20M13S598N1000F</MediaDuration>
          </MediaTime>
        - <TemporalDecomposition>
          - <AudioSegment id="S01">
            - <TextAnnotation>
                <FreeTextAnnotation>Silence</FreeTextAnnotation>
              </TextAnnotation>
            - <MediaTime>
                <MediaTimePoint>T00:00:00</MediaTimePoint>
                <MediaDuration>PT0M0S443N1000F</MediaDuration>
              </MediaTime>
            </AudioSegment>
          - <AudioSegment id="S02">
            - <TextAnnotation>
                <FreeTextAnnotation>Female</FreeTextAnnotation>
              </TextAnnotation>
            - <MediaTime>
                <MediaTimePoint>T00:00:00:443F1000</MediaTimePoint>
                <MediaDuration>PT0M0S423N1000F</MediaDuration>
              </MediaTime>
            </AudioSegment>

          [...]

           - <AudioSegment id="S180">
            - <TextAnnotation>
                <FreeTextAnnotation>MaleSpot</FreeTextAnnotation>
              </TextAnnotation>
            - <MediaTime>
                <MediaTimePoint>T00:20:04:217F1000</MediaTimePoint>
                <MediaDuration>PT0M0S455N1000F</MediaDuration>
              </MediaTime>
            </AudioSegment>
           - <AudioSegment id="S181">
            - <TextAnnotation>
                <FreeTextAnnotation>Silence</FreeTextAnnotation>
              </TextAnnotation>
            - <MediaTime>
                <MediaTimePoint>T00:20:04:672F1000</MediaTimePoint>
                <MediaDuration>PT0M8S925N1000F</MediaDuration>
              </MediaTime>
            </AudioSegment>
          </TemporalDecomposition>
        </Audio>
      </MultimediaContent>
    </Description>
  </Mpeg7>
```

Listing 4.9: Example of a XML Document MPEG-7 compliant created from a lab file of a GR3

The code is composed by four functions: the main function reads the lab file and controls if already exists an xml document for this lab file if yes then calls the function CreateXML passing it the entire duration of the GR, the starting point of the first segment and its duration.

The function CreateXML starting from the root element create all the nodes of the XML three as istance of the class docNode using the method createElement and sets their attributes using the method setAttribute. After that puts all the nodes in the correct position in the three using the method appendChild. This function ends with the writing of the wrapper and the description of the first segment in the XML document.

For the second segment the XML document already exists and so instead to call the function CreateXML calls the function addXMLAudioSegment, with the same parameters, which creates all the nodes necessary for the description of a segment in the same way of the function CreateXML and append this node in XML three of the document created early. This function ends rewriting the XML document with the new XML three. The code works in the same way for the other segments till the end of the lab file.

In order to test our XML document we validated it using the NIST MPEG-7 Validation Service [65]. In the listing 4.10 I report the output of the validation.

NIST MPEG-7 Validation Result ...

```
<Mpeg7Validation file="11-gr3_labels.xml">
  <Total warning="0" error="0" fatal="0"/>
  <Processed element="1094" attribute="368" time="25092ms"/>
</Mpeg7Validation>
```

Listing 4.10: NIST MPEG-7 Validation Result

CHAPTER 5
AUTOMATIC AUDIO SEGMENTATION

Segmenting audio data into speaker-labeled segments is the process of determining where speakers are engaged in a conversation (start and end of their turn). This finds application in numerous speech processing tasks, such as speaker-adapted speech recognition, speaker detection and speaker identification.

Example applications include speaker segmentation in TV broadcast discussions or radio broadcast discussion panels.

In [20], distance-based segmentation approaches are investigated. Segments belonging to the same speaker are clustered using a distance measure that measures the similarity of two neighboring windows placed in evenly spaced segments of time intervals. The advantage of this method is that it does not require any a priori information. However, since the clustering is based on distances between individual segments, accuracy suffers when segments are too short to describe sufficiently the characteristics of a speaker.

In [21], a model-based approach is investigated. For every speaker in the audio recording, a model is trained and then an HMM segmentation is performed to find the best time-aligned speaker sequence. This method places the segmentation within a global maximum likelihood framework. However, most model based approaches require a priori information to initialize the speaker models.

Similarity measurement between two adjacent windows is based on a comparison of their parametric statistical models. The decision of a speaker change is performed using a model-selection-based method [22, 23] called the Bayesian information criterion (BIC). This method is robust and does not require thresholding. In [24, 25] it is shown that a hybrid algorithm, which combines metric-based and model-based techniques, works significantly better than all other approaches.

5.1 Feature extraction

The performance of the segmentation depends on the feature representation of audio signals. Discriminative and robust features are required, especially when the speech signal is corrupted by channel distortion or additive noise. Various features have been proposed in the literature:

• Mel-frequency cepstrum coefficients (MFCCs): one of the most popular sets of features used to parameterize speech is MFCCs. As outlined in Chapter 3, these are based on the human auditive system model of critical frequency bands. Linearly spaced filters at low frequencies and logarithmically at high frequencies have been used to capture the phonetically important characteristics of speech.

• Linear prediction coefficients (LPCs)[26]: the LPC-based approach performs spectral analysis with an all-pole modeling constraint. It is fast and provides extremely accurate estimates of speech parameters.

• Linear spectral pairs (LSPs) [27]: LSPs are derived from LPCs. Previous research has shown that LSPs may exhibit explicit differences in different audio classes. LSPs are more robust in noisy environments.

• Cepstral mean normalization (CMN) [28]: the CMS method is used in speaker recognition to compensate for the effect of environmental conditions and transmission channels.

• Perceptual linear prediction (PLP) [29]: this technique uses three concepts from the psychophysics of hearing to derive an estimate of the auditory spectrum:

- ✓ critical-band spectral resolution,
- ✓ equal loudness curve
- ✓ the intensity–loudness power law

The auditory spectrum is then approximated by an autoregressive all-pole model. A fifth-order all-pole model is effective in suppressing speaker-dependent details of the auditory spectrum. In comparison with conventional linear predictive (LP) analysis, PLP analysis is more consistent with human hearing.

• RASTA-PLP [30]: the word RASTA stands for *RelAtive SpecTrAl technique*. This technique is an improvement on the traditional PLP method and incorporates a special filtering of the different frequency channels of a PLP analyzer. The filtering is employed to make speech analysis less sensitive to the slowly changing or steady-state factors in speech.

The RASTA method replaces the conventional critical-band short-term spectrum in PLP and introduces a less sensitive spectral estimation.

• Principal component analysis (PCA): PCA transforms a number of correlated variables into a number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.

• MPEG-7 audio spectrum projection (ASP).

5.2 Model-Based Segmentation

In model-based segmentation, a set of models for different acoustic speaker classes from a training corpus is defined and trained prior to segmentation. The incoming speech stream is classified using the models. The segmentation system finds the best time-aligned speaker sequence by maximum likelihood selection over a sliding window. Segmentation can be made at the locations where there is a change in the acoustic class. Boundaries between the classes are used as segment boundaries. However, most model-based approaches require a priori information to initialize the speaker models.

5.3 Metric-Based Segmentation

The metric-based segmentation task is divided into two main parts: speaker change detection and segment clustering.

First, the speech signal is split into smaller segments that are assumed to contain only one speaker. Prior to the speaker change detection step, acoustic feature vectors are extracted. Speaker change detection measures a dissimilarity value between feature vectors in two consecutive windows. Consecutive distance values are often low-pass filtered. Local maxima exceeding a heuristic threshold indicate segment boundaries.

Various speaker change detection algorithms differ in the kind of distance function they employ, the size of the windows, the time increments for the shifting of the two windows, and the way the resulting similarity values are evaluated and thresholded. The feature vectors in each of the two adjacent windows are assumed to follow some probability density (usually Gaussian) and the distance is represented by the dissimilarity of these two densities. Various similarity measures have already been proposed in the literature for this purpose.

The metric-based method is very useful and very flexible, since no or little information about the speech signal is needed a priori to decide the segmentation points. It is simple and applied without a large training data set. Therefore, metric-based methods have the advantage of low computation cost and are thus suitable for real-time applications. The main drawbacks are:

- ✓ It is difficult to decide an appropriate threshold
- ✓ Each acoustic change point is detected only by its neighbouring acoustic information.

- ✓ To deal with homogeneous segments of various lengths, the length of the windows is usually short (typically2 seconds). Feature vectors may not be discriminative enough to obtain robust distance statistics.

5.4 Hybrid Segmentation

Hybrid segmentation is a combination of metric-based and model-based approaches. A distance-based segmentation algorithm is used to create an initial set of speaker models. Starting with these, model-based segmentation performs more refined segmentation.

The hybrid segmentation can be divided into seven modules: silence removal, feature extraction, speaker change detection, segment-level clustering, speaker model training, model-level clustering and model-based resegmentation using the retrained speaker models.

5.5 Decoder-Guided Segmentation

The input audio stream can be first decoded; then the desired segments can be produced by cutting the input at the silence locations generated from the decoder [31, 32]. Other information from the decoder, such as the gender information, could be utilized in the segmentation [32].

5.6 Model-Selection-Based Segmentation

The segmentation methods earlier described, according to [22], are not very successful in detection the acoustic changes present in the data. The decoder-guided segmentation only places boundary at silence locations, which in general has no direct connection with the acoustic changes in the data. Both the model-based segmentation and the metric-based segmentation rely on thresholding of measurements which lack stability and robustness. Besides, the model-based segmentation does not generalize to unseen acoustic conditions.

In this thesis, I'm interested in detecting change in speaker identity in the radio news casting.

The input audio stream can be modelled as a Gaussian process in the cepstral space. I use the same maximum likelihood approach presented in [22] in order to detect turns of a Gaussian process; the decision of a turn is based on the *Bayesian Information Criterion* (BIC), a model selection criterion in the statistics literature.

In this chapter I first describe the model selection criterions, the maximum likelihood approach for acoustic change detection explained in [22] and then I describe how I

implemented this algorithm and I present experiments on the data base that I have described in the fourth chapter.

5.7 Model Selection Criteria

The challenge of model identification is to choose one from among a set of candidate models to describe a given data set. Candidates of a series of models often have different numbers of parameters. It is evident that when the number of parameters in the model is increased, the likelihood of the training data is also increased. However, when the number of parameters is too large, this might cause the problem of overtraining. Further, model-based segmentation does not generalize to acoustic conditions not presented in the model.

Several criteria for model selection have been introduced in the literature, ranging from non-parametric methods such as cross-validation to parametric methods such as the BIC [34]. The BIC permits the selection of a model from a set of models for the same data: this model will match the data while keeping complexity low. Also, the BIC can be viewed as a general change detection algorithm since it does not just take into account prior knowledge of speakers.

BIC is a likelihood criterion penalized by the model complexity: the number of parameters in the model. In detail, let $X = \{x_i : i = 1, \dots, N\}$ be the data set we are modelling; let $M = \{M_i : i = 1, \dots, K\}$ be the candidates of desired parametric models. Assuming we maximize the likelihood function separately for each model M, obtaining, say $L(X, M)$. Denote #(M) as the number of parameters in the model M. The BIC criterion is defined as:

$$BIC(M) = \log L((X, M)) - \lambda \frac{1}{2} \#(M) \times \log(N) \qquad (5.1)$$

Where the penalty weight Λ=1. The BIC procedure is to choose the model for which the BIC criterion is maximized.

BIC is closely related to other penalized likelihood criterions such as AIC [35] and RIC [36]. One can vary the penalty weight Λ in (5.1), although only Λ = 1 corresponds to the definition of BIC.

5.8 Change Detection via BIC

In this section, I describe the maximum likelihood approach for acoustic change detection based on the BIC criterion suggested by the authors of [22].

Denote $X = \{x_i \in R^d, i = 1, \dots, N\}$ as the sequence of cepstral vectors extracted from the entire audio stream; assume x is drawn from an independent multivariate Gaussian process:

$$x_i \sim N(\mu_i, \sum_i) \qquad (5.2)$$

where μ_i is the mean vector and $\sum_i$ is the full covariance matrix.

Instead of making a local decision based on the distance between two adjacent sliding windows of fixed size, [22] applied the BIC to detect the change point within a window.

The maximum likelihood ratio between H0 (no speaker turn) and H1 (speaker turns at time i) applied to the GLR is then defined by:

$$R_{BIC}(i) = \frac{N_X}{2}\log|\sum_X| - \frac{N_{X_1}}{2}\log\left|\sum_{X_1}\right| - \frac{N_{X_2}}{2}\log\left|\sum_{X_2}\right| \qquad (5.3)$$

where $\sum_X, \sum_{X_1}, \sum_{X_2}$ are the covariance matrices of the complete sequence, the subset $X1 = \{x_1, \dots, x_i\}$ and the subset $X2 = \{x_{i+1}, \dots, x_{N_X}\}$ respectively.

N_X, N_{X_1}, N_{X_2} are the number of acoustic vectors in the complete sequence, sub-set X_1 and sub-set X_2.

The speaker turn point is estimated via the maximum likelihood ratio criterion as:

$$\hat{t} = \arg\max_i R_{BIC}(i) \qquad (5.4)$$

On the other hand, we can view the hypothesis testing as a problem of model selection. We are comparing two models: one models the data as two Gaussians; the other models the data as just one Gaussian. The difference between the BIC values of these two models can be expressed as

$$BIC(i) = R(i) - \Lambda P \qquad (5.5)$$

Where the likelihood ratio R (i) is defined in (5.3), the penalty

$$P = \frac{1}{2}\left(d + \frac{1}{2} d\,(d+1)\right) \log N \qquad (5.6)$$

And the penalty weight $\Lambda = 1$; d is the dimension of the space. Thus if (5.5) is positive, the model of two Gaussians is favoured. Thus we decide there is a change if

$$\{\max_i BIC(i)\} > 0 \qquad (5.7)$$

It is clear that the m.l.e. of the changing point also can be expressed as

$$\hat{t} = \arg\max_i BIC(i) \qquad (5.8)$$

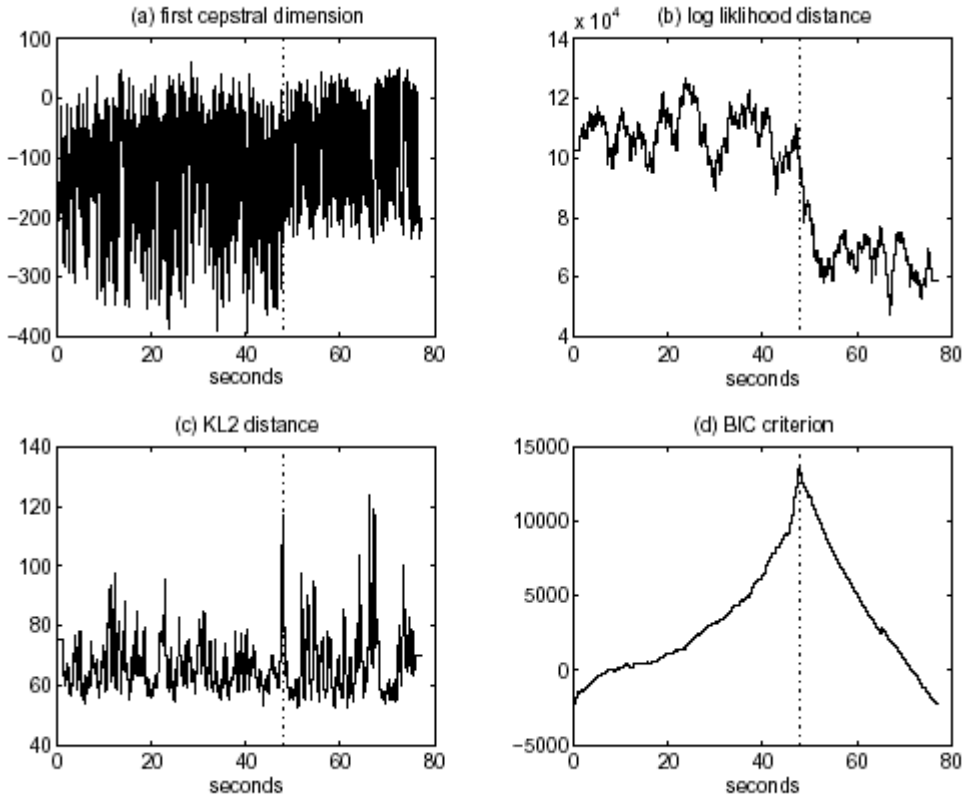


Figure 5.1 Detecting one changing point

Comparing with the metric-based segmentation described in the first sections of this chapter, the BIC procedure according to the author of [22] has the following advantages:

- *Robusteness* [37, 38] proposed to measure the variation at location i as the distance between a window to the left and a window to the right; typically the window size is

short, e.g. two seconds; the distance can be chosen to be the log likelihood ratio distance [39] or the KL distance. According to [22], such measurements are often noisy and not robust, because it involves only the limited samples in two short windows. In contrast, the BIC criterion is rather robust, since it computes the variation at time i utilizing all the samples. Figure 5.2 [22] shows an example which indicates the robustness of the Chen and Gopalakrishnan's BIC procedure. Panel (a) plots the first dimension of the cepstral vectors of a speech signal of 77 s which contains two speakers; the dotted line indicates the location of the change. One can clearly notice the changing behaviour around the changing point. Panel (b) shows the log likelihood distance: it attains local maximum at the location of a change; however, it has several maxima which do not correspond to any changing points; it also seems rather noisy. Similarly Panel (c) shows the KL2 distances [37] between two adjacent sliding windows of 100 frames: there is a sharp spike at the location of the change; however, there are several other spikes which do not correspond to any changing points. Panel (d) displays the BIC criterion; it clearly predicts the changing point.

- *Thresholding-free* Chen and Gopalakrishnan's BIC procedure is able to automatically performs model selection, whereas [37] is based on thresholding. As shown in Figure 5.1 (b) and (c), it is difficult to set a thresholding level to pick the changing points. Figure 5.1(d) indicates there is a change since the BIC value at the detected changing point is positive.
- *Optimaticality* Chen and Gopalakrishnan's BIC procedure is derived from the theory of maximum likelihood and model selection. The (5.8) converges to the true changing point as the sample size increase.

But because of the growing window, Chen and Gopalakrishnan's BIC scheme suffers from high computation costs, especially for audio streams that have many long homogeneous segments [40].

How I have said earlier BIC is supposed to have the advantage of not having any thresholding but this is only true if $\Lambda=1$ or if there a systematic way to find the optimal value of Λ. In absence of this, Λ is an implicit threshold embedded into the penalty term. This fact has been mentioned in previous work and was also noticed during my experiments as discussed later. In [41], it was mentioned that the threshold found using BIC principle (with $\Lambda=1$) yielded significantly worse results compared to the best possible threshold selection. In [42], the value

of Λ used was different than 1. In [43] a development dataset was used to find the optimal value of this parameter.

During my experiments we noticed that higher values of Λ result in a higher threshold, and thus ignore many genuine speaker changes. A lower value, on the other hand, results in many false alarms.

5.9 Detecting Multiple Changing Points

[22] propose the following algorithm to sequentially detect the changing points in the Gaussian process x:

1. Initialize the interval [a, b]: a = 1; b = 2.
2. Detect if there is one changing point in [a, b] via BIC.
3. if (no change in [a, b])
    a. let b = b+1;

   else
    b. let t be the changing point detected;
    c. set a = t+1; b = a +1

   end
4. go to 2.

By expanding the window [a, b], the final decision of a change point is made based on as much data points as possible. This can be more robust than decisions based on distance between two adjacent sliding windows of fixed sizes [37], though the Chen and Gopalakrishnan's approach is more costly.

The algorithm that I used is very similar to the one presented in [33]. I implemented this algorithm in Matlab starting from the Alexander Haubold's algorithm [45] and basically maintaining his approach. In his algorithm there are two BIC evaluation levels: on top of first level (coarse) BIC evaluation there is a second level (fine) BIC evaluation.

In the next chapter I describe in detail the algorithm that I used in my experiments on the GRR database.

CHAPTER 6

EXPERIMENTS AND RESULTS

As I said in the previous chapter I implemented the algorithm, descript in the section 5.9, in Matlab. The program requires as input the audio wav file and the CLAM XML file. In order to interface Matlab with CLAM I needed an XML parser which was able to extract the values of the descriptors from the CLAM XML file. For this purpose I used XMLTree an XML toolbox for Matlab [46].

The speaker change detection was performed using 20-dimensional Mel-cepstral vectors with a frame size of 1023.

I made experiments with 5 GRRs (Giornale Radio Rai). As we can see in the Tab 6.1 the GRRs have duration of 20-30 minutes and my pc hasn't enough memory to allow CLAM to extract the descriptors of these file. For this reason I created a matlab code, which splits each GRR into segments of 10 minutes with an overlap of 1 minute. Then I put the paths of these segments in the Project file of CLAM music Annotator, as descript in 3.2.1, in this way I obtained the XML documents containing the low level descriptors descript in the chapter 4. As commented in [38], it is very hard to come up with a standard for analyzing the errors in segmentation since segmentation can be very subjective; even two people listening to the same speech may segment it differently. Nevertheless, I analyze the performance of my detection by comparing with the hand – segmentation precedent made with WaveSurfer. For the Evaluation I used the same lab files, which I used for database creation, but I had to segment it and to create a matlab code that deletes all the segments that has duration inferior to 1s.

In order to compare the two lab files I used two different evaluation methods that I describe in the next section.

1	GR3	06/11/2007	**20 min**
2	GR3	07/11/2007	**20 min**
3	GR1	17/11/2007	**30 min**
4	GR2	08/11/2007	**28 min**
5	GR2	12/11/2007	**28 min**

Table 6.1: GRR

6.1 Evaluation Method

In a change detection system there are two types of errors. The first type takes place when a true change is not spotted and is called recall (R) while the second type happens when the system detects a change that does not actually exists and is called precision (P).

Following the approach used in [66] I implemented a matlab code that takes as input the two lab file and computes precision P; recall R and F-measure F.

Precision is defined as the proportion of detected transitions that are relevant. Recall is defined as the proportion of relevant transitions detected.

Thus, if B = {relevant transitions}, C = {detected transitions} and A = B ∩ C, from the above definition,

$$P = A/C \qquad (6.1)$$

$$R = A/B \qquad (6.2)$$

$$F = 2\,P * R/\,(P + R) \qquad (6.3)$$

A parameter w determines how far two boundaries can be apart but still count as one boundary. A typical value is from 0.5 s to 3 s, i.e., all boundaries within the range of w seconds before to w seconds after a boundary b are seen as identical to b. Figure 6.1 shows the effect of w: black boundaries in the upper panel count as hits, red ones as false alarm.

In the example in figure 6.1 precision is 3/6 and recall ¾.

This metrics go from zero (bad performance) to 1 (good performance).

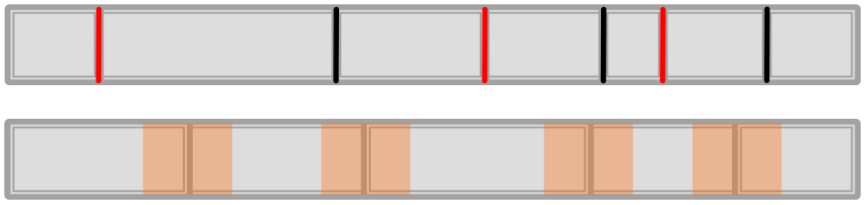


Figure 6.1: Boundary evaluation; top: detected boundaries, bottom: true boundaries

6.2 Alternative Measure

In [67] is used another performance measure, so I also computed P_{am}, R_{am} and F_{am}. P_{am} and R_{am} correspond to [67]'s 1-f and 1-m, respectively, and are calculated as follows: Considering the measurement M [computed segmentation] as a sequence of segments S^i_M, and the ground truth G likewise as segments S^j_G, we compute a directional Hamming distance

d_{GM} by finding for each S^i_M the segment S^j_G with the maximum overlap, and then summing the difference,

$$d_{GM} = \sum_{S^i_M} \sum_{S^k_G \neq S^j_G} \left| S^i_M \cap S^k_G \right| \qquad (6.4)$$

where $|\cdot|$ denotes the duration of a segment. We normalize d_{GM} by the track length dur to give a measure of the missed boundaries. Similarly, we compute d_{MG}, the inverse directional Hamming distance, and a similar normalized measure d_{MG}/L of the segment fragmentation.
Then ,

$$P_{am} = 1 - \frac{d_{MG}}{dur} \qquad (6.5)$$

$$R_{am} = 1 - \frac{d_{GM}}{dur} \qquad (6.6)$$

$$F_{am} = 1 - 2 * R_{am} * \frac{P_{am}}{R_{am} + P_{am}} \qquad (6.6)$$

The main advantage of the alternative measures is that they somehow reflect how much the two segmentations differ from each other: If a boundary b from computed segmentation is apart more than w from the corresponding one in the ground truth b_0, it does not count for P or R, regardless of how far they are apart (since b doesn't belongs to A). In contrast, P_{am} and R_{am} will rise depending on the distance between b and b_0 since these measures are not based on the boundaries directly but rather on (overlapping) segments between them.
Also this metrics go from zero (bad performance) to 1 (good performance).

6.3 Segmentation System

In the figure 6.2 is reported the segmentation system scheme that I used to make the experiment that I will describe in the next sections.

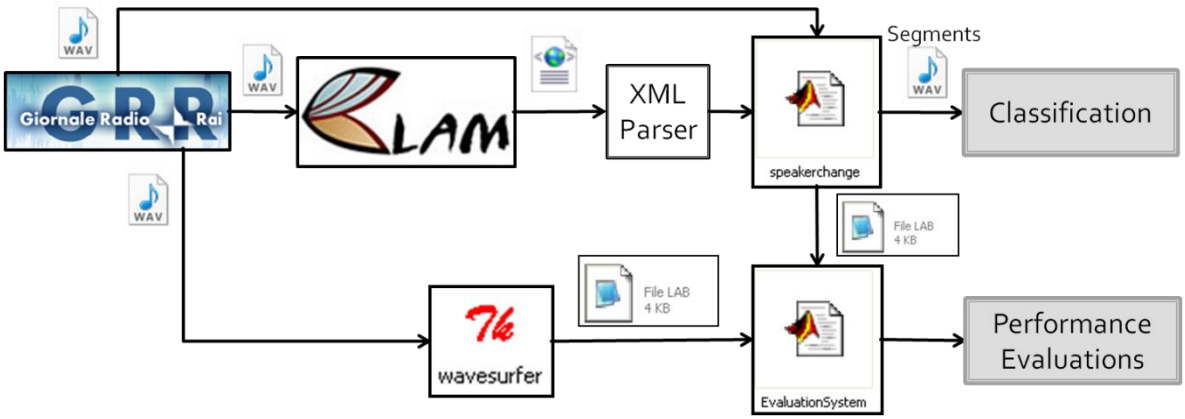


Figure 6.2: Segmentation system scheme.

6.4 First Experiment

In the first experiment I used the following algorithm:

1. Initialize the interval [a, b]:
   a = 1 b = 2*BICEVALSTEP
2. Detect if there is one changing point in [a, b] evaluating a coarse BIC every 16 frames.
3. Let be i_{MAX} the index of the maximum positive value in the BIC vector.
   if (i_{MAX} exists && i_{MAX} < length(BIC) - BICEVALTRAILBUFFER)
      a. Find the change point in [a, b] evaluating a fine BIC every frame.
      b. Set a = b – BICEVALSTEP; b = b+BICEVALSTEP.
   else
      Set b = b + BICEVALSTEP.
   end
4. Go to 2

The values like BICEVALSTEP and BICEVALTRAILBUFFER are also optimized for good performance as well as keeping the computational complexity reasonable. In my experiments BICEVALSTEP was chosen to correspond to 3,68 s of speech.

The [a, b] interval is the interval where we are assuming that there is at most one changing point. At the second point of the algorithm a BIC vector of fix(b-a/16) elements is created and a changing point is detected if there is a maximum positive value in the top of the BIC vector (i_{MAX} < length(BIC) - BICEVALTRAILBUFFER) otherwise any changing point is detected and the interval [a, b] increase. If there is a maximum we compute a fine BIC every frame and then a is set to b-BICEVALSTEP. The figure 6.3 shows how became the [a, b] interval if the maximum value is in the red zone.

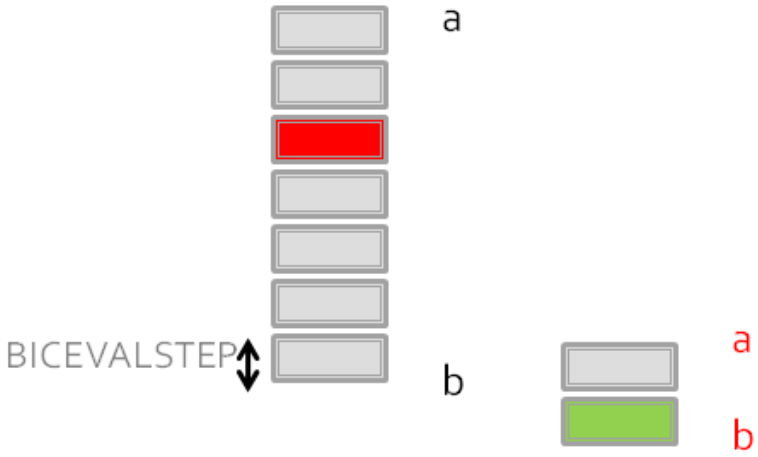


Figure 6.3: [a, b] Interval setting.

In this experiment BICEVALSTEP is set to 16 frames; BICEVALTRAILBUFFER is set to 160 frames and λ is set to 1.

In the following figures I report the evaluation parameters for each 10 minutes segment of GRR in order to evaluate the performance of this algorithm.

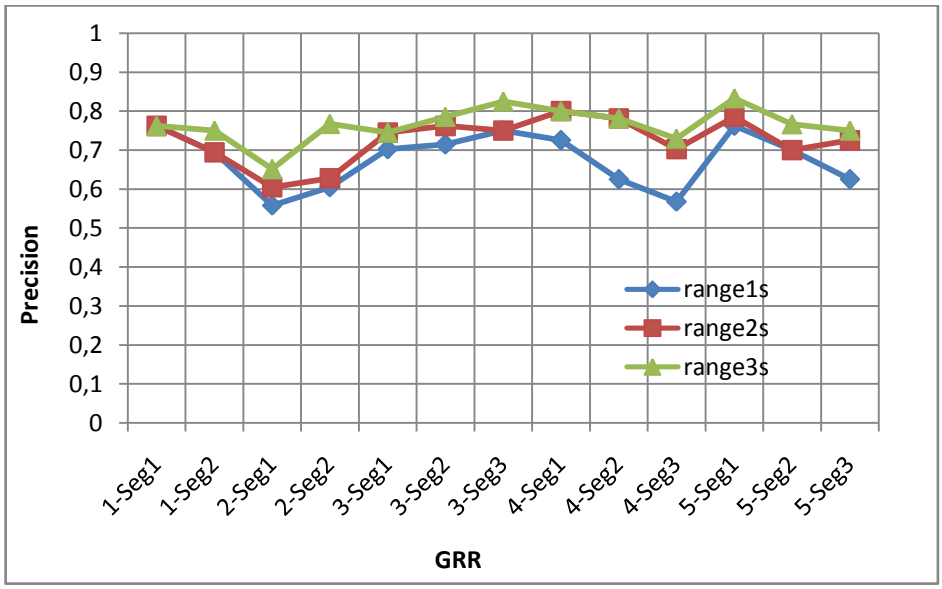


Figure 6.4: Precision of the first experiment varying the range

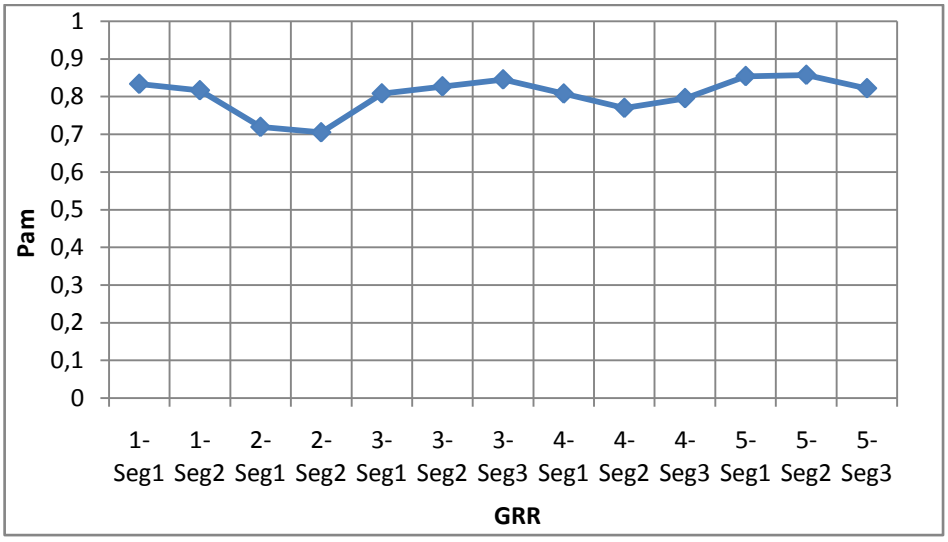


Figure 6.5: Pam of the first experiment

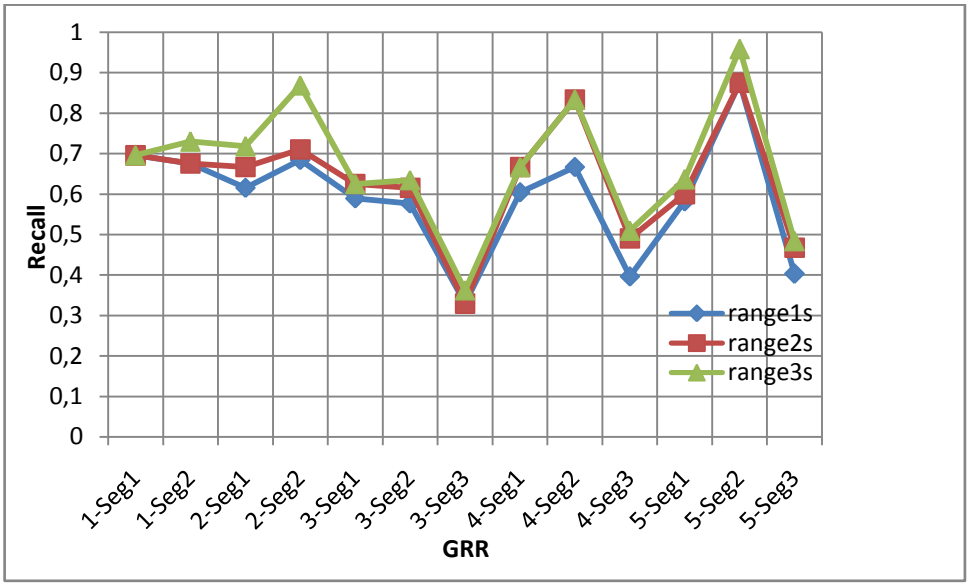


Figure 6.6: Recall of the first experiment varying the range

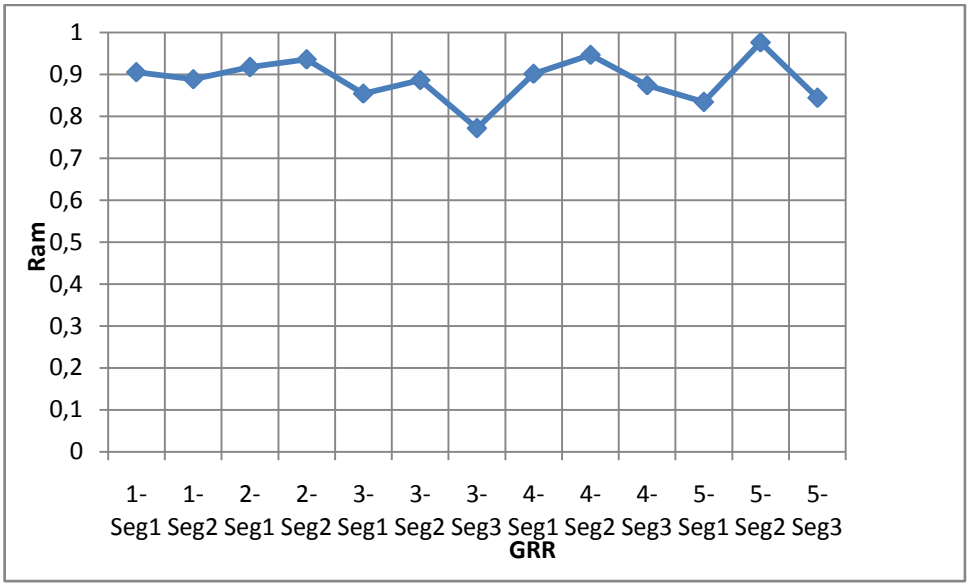


Figure 6.7: Ram of the first experiment

In the figure 6.6 can be noted that the recall values have a large variation this is due to the different characteristics of the 3 segments of GRR, in fact we have the worst result in the third one segments in which there are commercials while we have a good recall in the seconds segments in which there are only news.

Comparing figure 6.6 and 6.4 we can note that the precision values haven't a large variation as the recall values this means that the number of false alarm are quite the same for all segments.

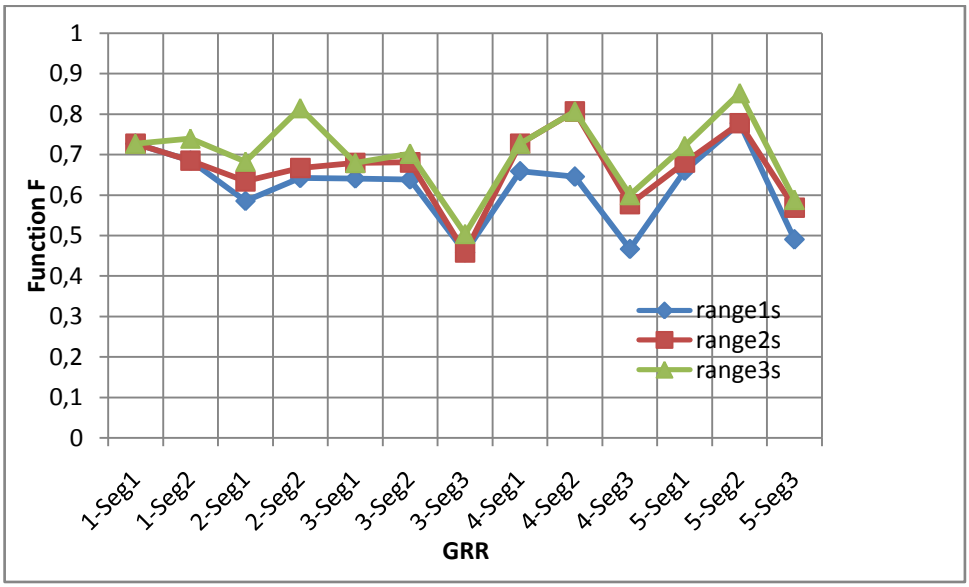


Figure 6.8: F function of the first experiment varying the range

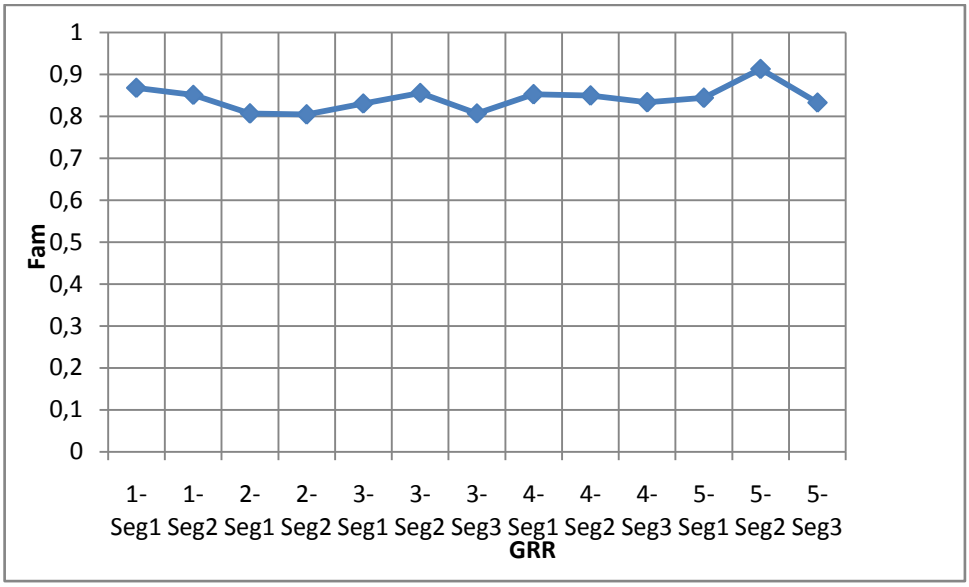


Figure 6.9: Fam of the first experiment

Comparing the figure 6.8 and 6.9 we can see that mean F_{am} is higher than mean F this is due to the "binaryness" of P and R.

In the Figure 6.9 it should be noted that the performance of the algorithm is rather stable.

In the Table 6.1 I report for each segment of 10 minutes of GRR the number of segment calculated with a manual segmentation and the number of the segments that are calculated automatically.

GRR	# Truth Segments	# Predicted Segments Λ=1
1-GR3-Seg1	47	43
1-GR3-Seg2	38	37
2-GR3-Seg1	40	44
2-GR3-Seg2	39	44
3-GR1-Seg1	57	48
3-GR1-Seg2	53	43
3-GR1-Seg3	92	41
4-GR2-Seg1	49	41
4-GR2-Seg2	31	33
4-GR2-Seg3	54	38
5-GR2-Seg1	56	43
5-GR2-Seg2	25	31
5-GR2-Seg3	63	41

Table 6.2: Number of the segments for each GRR

I calculated also manually the performance of the automatic audio segmentation of the 5-GR2-Seg2 comparing the 2 file lab. In the Table 6.2 are reported the number of the false alarm and the number of the missed detection.

GRR	# False Alarm	# Missed Detection
5-GR2-Seg2	**9**	**3**

Table 6.3: Number of the false alarm and missed detection for the 5-GR2-Seg3

Can be noted that the number of the false alarm and the missed detection reflect the values of precision and recall.

We can compute the number of false alarm and the missed detection in the Figure 6.9 where are reported the sequence of segments calculated manually and automatically. We can see that

the missed segments are very short. In fact as emphasized in [22] the accuracy of the BIC procedure depends on the detectabilities of the true changing points.

Let $T = \{t_i\}$ be the true changing points; the detectability can be defined as

$$D(t_i) = min(t_i - t_{i-1} + 1, t_{i+1} - t_i + 1) \qquad (6.7)$$

When the detectability is low, the current changing point is often missed.

In this example the detectabilities of the missed segments is very low.

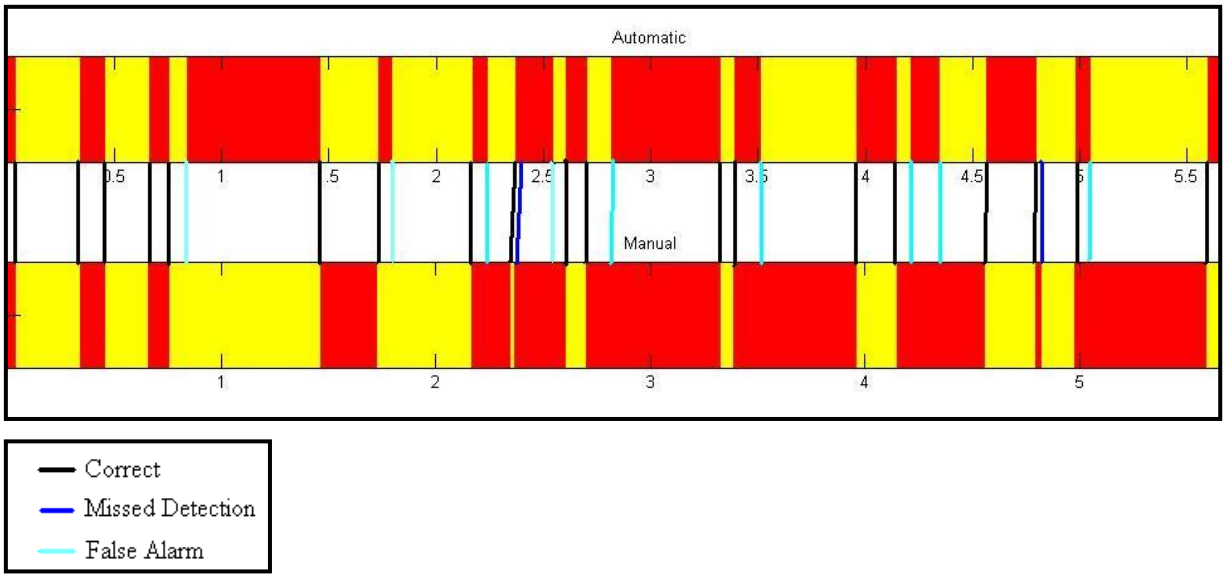


Figure 6.9: Comparison between manual and automatic segmentation

6.5 Second Experiment

In the second experiment I used the same algorithm of the first experiment but I set BICEVALTRAILBUFFER to ten frames in order to have more segments detected in fact in this experiment a maximum could be in the upper half part of the BIC vector while in the first experiment the maximum should be in the first four positions in the BIC vector to be consider as a changing point. In this experiment I also made a tuning of λ. In fact in the literature BIC is supposed to have the advantage of not having any thresholding. However, this is only true if λ =1 or if there a systematic way to find the optimal value of λ. In absence of this, λ is an implicit threshold embedded into the penalty term.

This fact has been mentioned in previous work. In [68], it was mentioned that the threshold found using BIC principle (with λ=1) yielded significantly worse results compared to the best possible threshold selection. In [69], the value of λ used was different than 1.0. In [70] a development dataset was used to find the optimal value of this parameter.

So I computed the number of the segments detected tuning λ and then I chose tree different values of λ according to the type of segment: if it was a news segment or a commercial segment. A systematic way to select the value of λ could be to split the GRR in input into 10 m segments and then to select λ according to the GRR and to the number of segment. But in this way this system is thresholding-free only in the case of the GRRs.

In the table 6.3 are reported the values of λ and the number of the detected segments. In this experiment I chose the value of λ corresponding to the number of the segments detected in the yellow area in the table 6.3.

GRR	*# T Seg*	*# P Seg Λ=0.9*	*# P Seg Λ=1*	*# P Seg Λ=1.1*	*# P Seg Λ=1.2*	*# P Seg Λ=1.3*	*# P Seg Λ=1.4*
1-GR3-Seg1	**47**	74	68	55	**45**	42	40
2-GR3-Seg1	**40**	75	63	52	**42**	35	34
3-GR1-Seg2	**53**	74	68	59	**52**	44	41
4-GR2-Seg2	**31**	62	48	40	**33**	26	25
5-GR2-Seg2	**25**	71	48	37	29	26	**25**
3-GR1-Seg1	**57**	80	72	**61**	51	51	51
4-GR2-Seg1	**49**	74	62	**51**	44	43	40
5-GR2-Seg1	**56**	77	70	**57**	54	54	51
3-GR1-Seg3	**92**	**74**	71	66	62	58	53
4-GR2-Seg3	**54**	72	64	**48**	43	39	39
5-GR2-Seg3	**63**	86	78	**58**	50	45	40
1-GR3-Seg2	**38**	71	62	50	41	**39**	36
2-GR3-Seg2	**39**	83	78	58	46	42	**41**

Table 6.4: Tuning of λ in the second experiment.

This approach should give a minor variation to the performance metrics. In fact the figures 6.10 and 6.11 show that the variation of the function F and Fam is decreased.

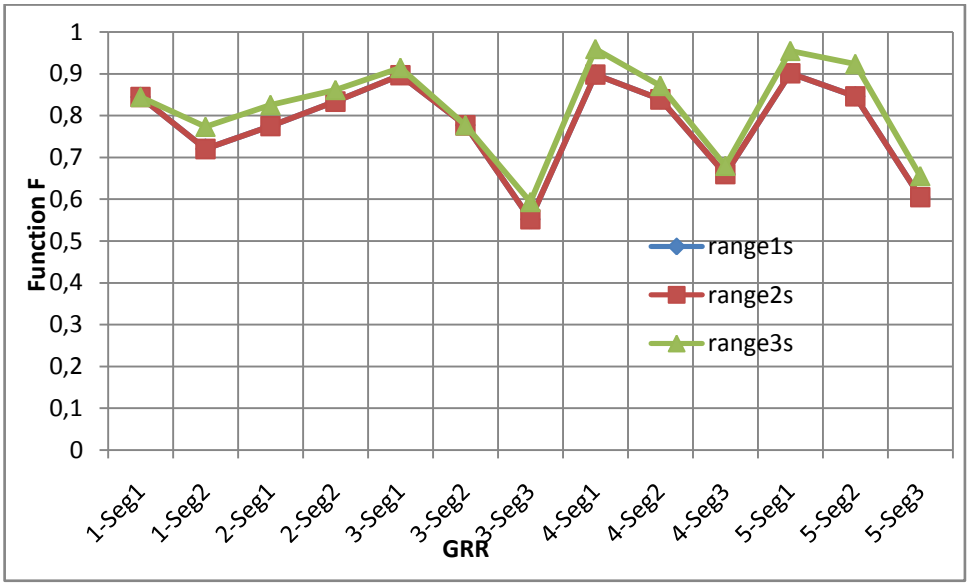


Figure 6.10: F function of the second experiment varying the range.

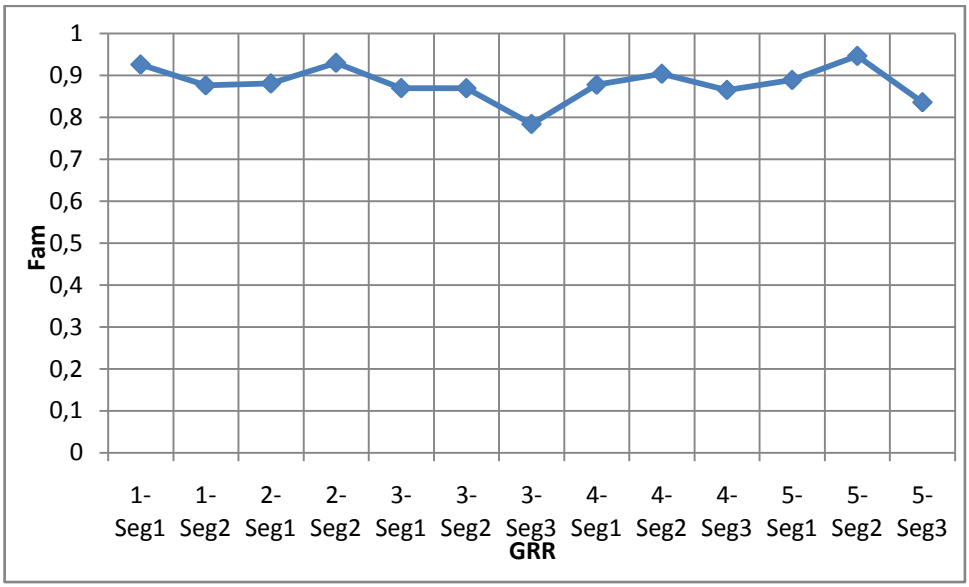


Figure 6.11: Fam of the second experiment

6.6 Third Experiment

In the third experiment I changed the third point of the algorithm that I used in the previous experiments. The algorithm in this experiment runs as follows:

1. Initialize the interval [a, b]:
   a = 1 b = 2BICEVALSTEP
2. Detect if there is one changing point in [a, b] evaluating a coarse BIC every 16 frames.
3. Let be i_{MAX} the index of the maximum positive value in the BIC vector.
   if (i_{MAX} exists && i_{MAX} < length(BIC) - BICEVALTRAILBUFFER)
      a. Find the change point in [a, b] evaluating a fine BIC every frame.
      b. Set $a = \left[b - \left(\frac{b-1}{BICEVALSTEP}\right) - (fix\left(\frac{i_{MAX}+16}{BICEVALSTEP}\right) + 1)\right] BICEVALSTEP$

         b = b + BICEVALSTEP
   else
      Set b = b + BICEVALSTEP.
   end
4. go to 2

So in this experiment if there is a changing point: 'a' is set to the beginning of the next BICEVALSTEP and not to the last BICEVALSTEP.

The figure 6.12 shows how became the [a, b] interval if the changing point was in the red zone.

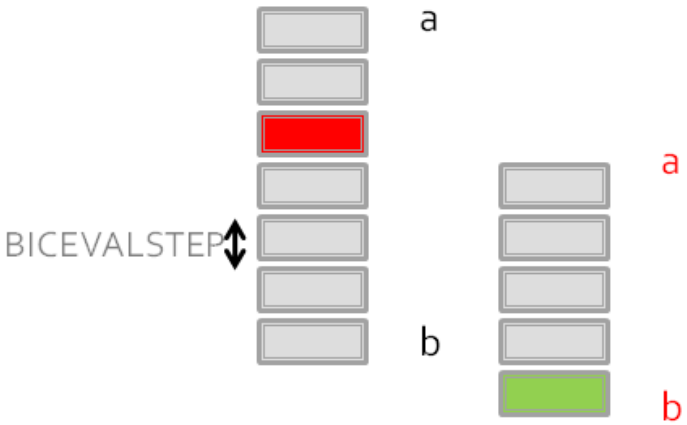


Figure 6.12: [a, b] interval setting in the third experiment.

In this experiment BICEVALSTEP is set to 160 frames and BICTRAILBUFFER is set to 10 frames. We made also in this experiment the λ tuning in order to choose the optimal value of λ.

In the table 6.4 there are the number of the detected segments varying λ. In this case we can choose the same value of λ for all the segments. In fact I chose the value of λ corresponding to the number of detected segments in the yellow area.

GRR	*# T Seg*	*# P Seg Λ=1*	*# P Seg Λ=1.1*	*# P Seg Λ=1.2*	*# P Seg Λ=1.3*	*# P Seg Λ=1.4*	*# P Seg Λ=1.5*
1-GR3-Seg1	**47**	67	57	**48**	45	42	41
2-GR3-Seg1	**40**	64	50	**43**	34	34	34
3-GR1-Seg2	**53**	68	59	**51**	45	42	42
4-GR2-Seg2	**31**	52	43	**34**	28	27	25
5-GR2-Seg2	**25**	46	35	29	26	**25**	23
3-GR1-Seg1	**57**	77	61	**56**	54	54	52
4-GR2-Seg1	**49**	68	53	**45**	44	42	38
5-GR2-Seg1	**56**	70	59	58	**57**	54	51
3-GR1-Seg3	**92**	**78**	74	68	64	58	52
4-GR2-Seg3	**54**	66	**48**	45	40	40	38
5-GR2-Seg3	**63**	78	**60**	49	47	43	40
1-GR3-Seg2	**38**	61	52	42	**40**	37	36
2-GR3-Seg2	**39**	78	57	49	44	43	**39**

Table 6.5: λ tuning in the third experiment

In the figures 6.13 and 6.14 it should be noted that the performance of the algorithm is rather stable also using one value of λ for all segments.

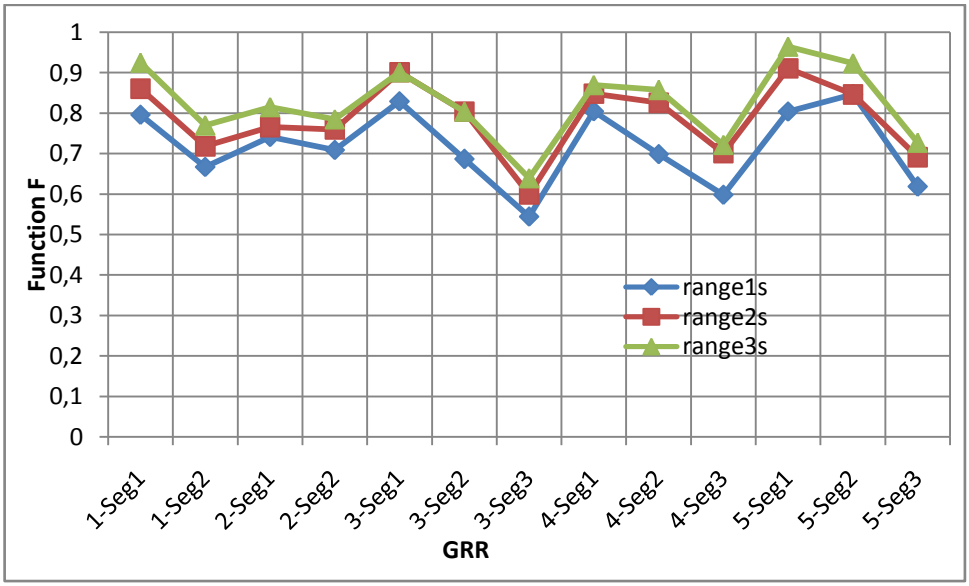


Figure 6.13: F function of the third experiment varying range.

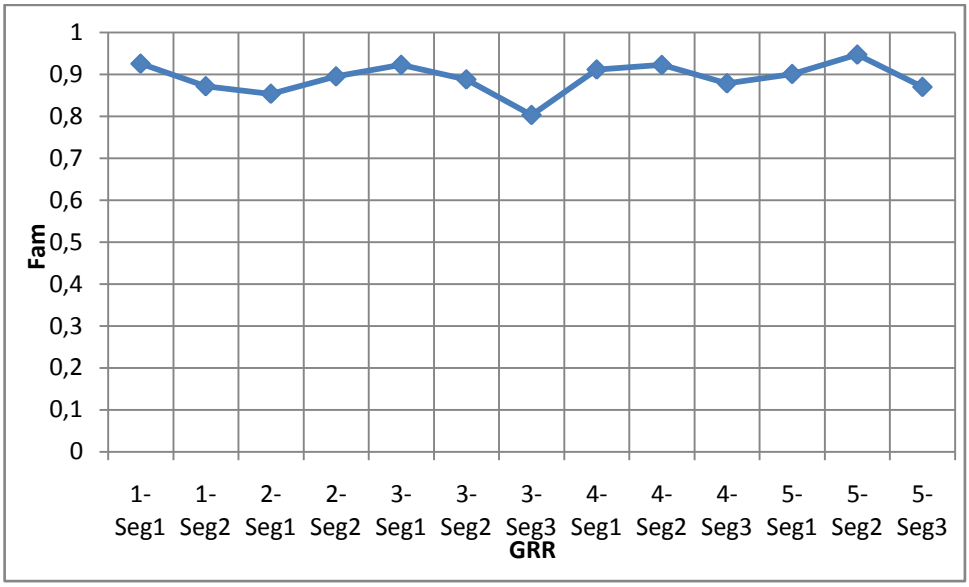


Figure 6.14: Fam of the third experiment.

6.7 Conclusion

In the figure 6.15 and 6.16 are reported the F function and Fam of the tree experiment. This graphs show that the performance of the third experiment is more stable than the performance of the other experiment, and moreover in the third experiment we have only one value of λ for all segments.

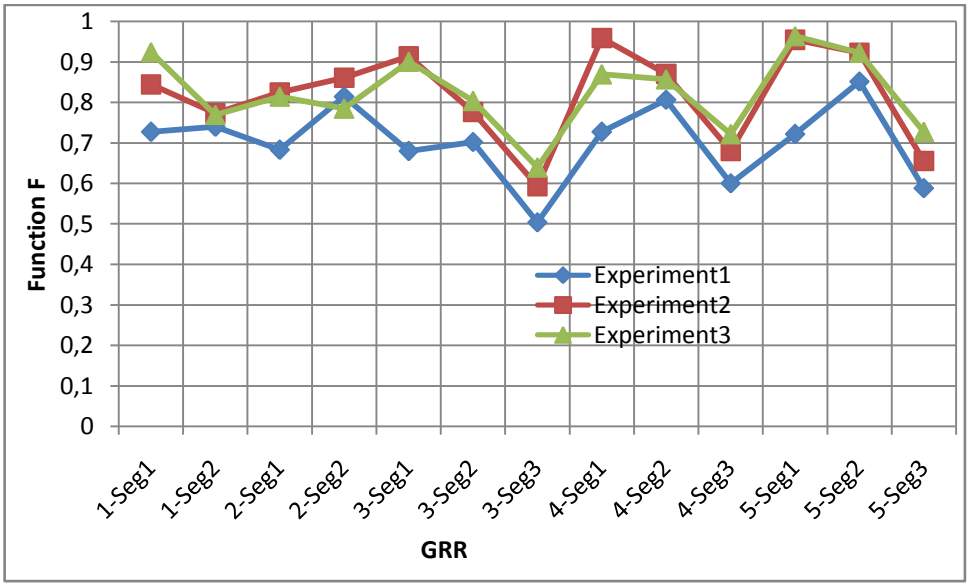


Figure 6.15: F function of the three experiments with range=3s.

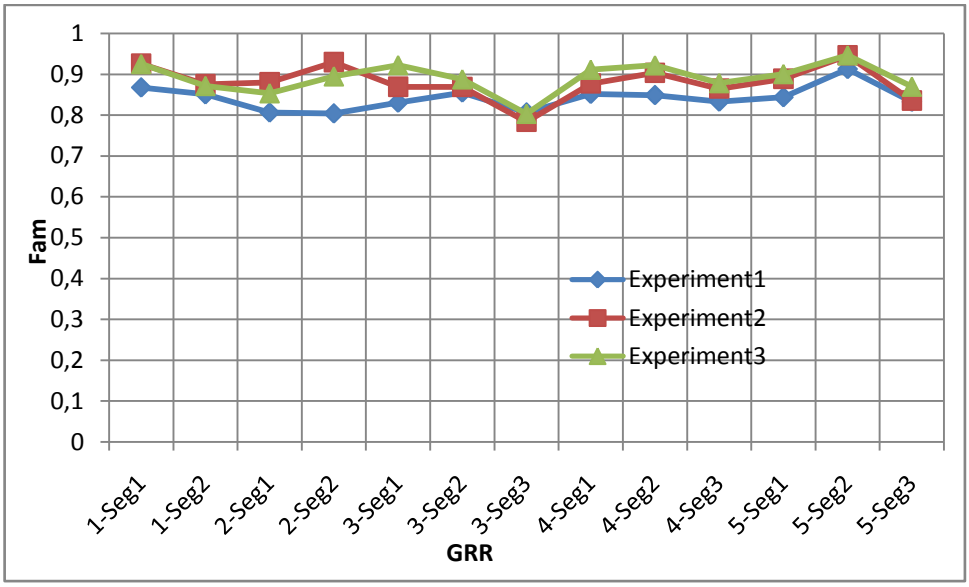


Figure 6.16: Fam of the three experiments.

Summarizing:

- In the first experiment the recall metric has a too large variation.
- In the second experiment the recall variation is decreased, but we have tree different values of lambda.
- In the third experiment we reach good performance using only one value of lambda.

So the third experiment on the GRR database show that the last algorithm can successfully detect speaker change in the GRR wav file.

The output of this system could be the input of the classification system implemented by Giuseppe Dimattia in his thesis [71].

CHAPTER 7

CONCLUSION

I am very pleased with the amount of research and testing that took place toward the completion of this project. I believe I achieved the goals of the project as well.

In this thesis a segmentation system was developed in order to segment a continuous audio stream in terms of acoustically homogenous regions. In fact it was able to detect audio events in a radio newscast audio stream. The experiments conducted on some samples of the GRR database that was created demonstrated that the spectral descriptors extracted using CLAM music annotator was able to distinguish music and speech; male and female; anchor and telephone reporter; one speaker and more speakers. After these experiments the MFCC descriptor was chosen to segment the GRR audio stream. The segmentation system based on BIC was implemented in Matlab and an XML parser was employed to allow the communication between CLAM and Matlab.

The evaluation of output of algorithms is, at least in the research phase, as important as the algorithm itself. If the evaluation procedure does not produce useful and applicable performance numbers any effort to optimize an algorithm becomes futile. In fact you would not know whether algorithm A performs better than algorithm B.

Thus, I decided to devote a significant part of my work to my evaluation system.

It is very handy if there is a simple-to-use procedure that automatically produces both optical appealing and informative reports. This makes it possible to have rapid feedback loops in a phase where you adjust the large number of degrees of freedoms, i.e., algorithm parameters, to get an optimal result. I implemented two matlab codes in order to evaluate the performance of the segmentation system. The first one is based on boundary it computes recall; precision and the F function that is a mean of the other two metrics. The second one is based on the Directional Hamming Distance and is based on overlap and not on boundaries. These have as input the two lab file one obtained by an hand-segmentation and the second one is the output of the segmentation code. The manual segmentation is performed using WaveSurfer that allows the manual annotation and that has as output a lab file. A matlab code that converts these file lab into XML MPEG-7 compliant document was also implemented. In order to make experiments 15 GRRs was collected. In the first experiment the weigh parameter lambda is set to 1. In the second experiment a tunning of lambda was performed and I choose three different value of lambda according the contents of the GRR

In the last experiment I modified the algorithm and I also implemented a lambda tunning but this time we could choose one value for lambda for all kind of segments and the performances of the segmentation system are improved.

The outputs of the segmentation system are the segments that are ready to be the input of classification system. In the figure 7.1is represented the entire system implemented in this thesis.

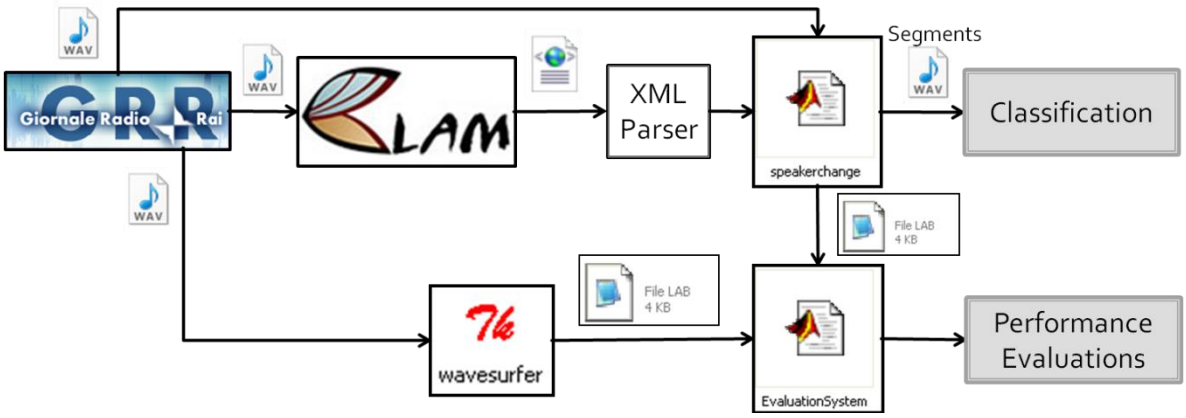


Figure 7.1: Segmentation System.

7.1 Future Works

In this thesis for the extraction of the low level descriptors I employed the ClamExtractorExample in the future one could start from this extractor and develop an own one modifying some parameters.

Another future work could be changing the matlab code in oder to segment the audio stream basing on another spectral features extracted using CLAM or on the Low Level Descriptors MPEG-7 compliant and then using the evaluation system could compare the different segmentation system.

One could consider also integration of this work into CLAM code base. Integration into a big existing project is often appreciated.

Another direction for future research could be modifying this system in order to make it real time.

REFERENCES

[1] NIST. Rich transcription task. http://nist.gov/speech/tests/rt/, 2005.

[2] J.-L. Gauvain, L. Lamel, and G. Adda. The limsi broadcast news transcription system. Speech Communication, 37(1-2):89–108, 2002. ISSN 0167-6393.

[3] S. E. Johnson, P. Jourlin, K. S. Jones, and P. C. Woodland. Information retrieval from unsegmented broadcast news audio. International Journal of Speech Technology, 4:251–268, 2001.

[4] J. Saunders. Real-time discrimination of broadcast speech/music. In IEEE International Conference on Acoustics, Speech, and Signal Processing. (ICASSP'96). Conference Proceedings., volume 2, pages 993–996, Atlanta, GA, USA, May 1996. IEEE.

[5] M. F. McKinney and J. Breebaart. Features for audio and music classification. Proc. of ISMIR, pages 151–158, 2003.

[6] T. Jehan. Creating Music by Listening. PhD thesis, Massachusetts Institute of Technology, 2005.

[7] G. Peeters and X. Rodet. Automatically selecting signal descriptors for sound classification. In Proceedings of the International Computer Music Conference, G¨oteborg, Sweden, 2002.

[8] http://www.chiariglione.org/mpeg/

[9] Information technology - Multimedia content description interface - Part 4: Audio. ISO/IEC 15938-4:2002. International Organisation for Standardisation, 2002.

[10] world wide web consortium (w3c)'s xml homepage: http://www.w3.org/xml/.

[11] B.S. Manjunath, P. Salembier, and T. Sikora, editors. Introduction to MPEG 7: Multimedia Content Description Language. John Wiley and Sons, Ltd, West Sussex, England, 2002.

[12] World Wide Web Consortium (W3C)'s XML-Schema homepage, http://www.w3.org/XML/Schema.

[13] Free Software Foundation. Gnu general public license (gpl) terms and conditions. http://www.gnu.org/copyleft/gpl.html.

[14] D. Roberts and R. Johnson. Evolve Frameworks into Domain-Specific Languages. In Procedings of the 3rd International Conference on Pattern Languages for Programming, Monticelli, IL, USA, September 1996.

[15] http://www.iua.upf.edu/mtg/publications/97b3cd-DEA-2006-Arumi.pdf

[16] D. Roberts and R. Johnson. Evolve Frameworks into Domain-Speci_c Languages. In Procedings of the 3rd International Conference on Pattern Languages for Programming, Monticelli, IL, USA, September 1996.

[17] D. Garcia and X. Amatrian. XML as a means of control for audio processing, synthesis and analysis. In Proceedings of the MOSART Workshop on Current Research Directions in Computer Music, Barcelona, Spain, 2001.

[18] Amatriain, X., Massaguer, J., Garcia, D., and Mosquera, I. (2005). The clam annotator: A cross-platform audio descriptors editing tool. In Proceedings of 6th International Conference on Music Information Retrieval, London, UK.

[19] J. Blanchette and M. Summer_eld. C++ GUI Programming with QT 3. Pearson Education, 2004.

[20] Gish H. and Schmidt N. (1994) "Text-Independent Speaker Identification", *IEEE Signal Processing Magazine*, pp. 18–21.

[21] Wilcox L., Chen F., Kimber D. and Balasubramanian V. (1994) "Segmentation of Speech Using Speaker Identification", *Proceedings ICASSP 1994*, Adelaide, Australia, April.

[22] Chen S. and Gopalakrishnan P. (1998) "Speaker Environment and Channel Change Detection and Clustering via the Bayesian Information Criterion", *DARPA Broadcast News Transcription and Understanding Workshop 1998*, Lansdowne, VA, USA, February.

[23] Delacourt P. and Welekens C. J. (2000) "DISTBIC: A Speaker-Based Segmentation for Audio Data Indexing", *Speech Communication*, vol. 32, pp. 111–126.

[24] Kemp T., Schmidt M., Westphal M. and Waibel A. (2000) "Strategies for Automatic Segmentation of Audio Data", *Proceedings ICASSP 2000*, Istanbul, Turkey, June.

[25] Kim H.-G. and Sikora T. (2004a) "Automatic Segmentation of Speakers in Broadcast Audio Material", *IS&T/SPIE's Electronic Imaging 2004*, San Jose, CA, USA, January.

[26] Rabiner L. R. and Schafer R. W. (1978) *Digital Processing of Speech Signals*, Prentice Hall (Signal Processing Series), Englewood Cliffs, NJ.

[27] Kabal P. and Ramachandran R. (1986) "The Computation of Line Spectral Frequencies Using Chebyshev Polynomials", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-34, no. 6, pp. 1419–1426.

[28] Furui S. (1981) "Cepstral Analysis Technique for Automatic Speaker Verification", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-29, pp. 254–272.

[29] Hermansky H. (1990) "Perceptual Linear Predictive (PLP) Analysis of Speech", *Journal of the Acoustical Society of America*, vol. 87, no. 4, pp. 1738–1752.

[30] Hermansky H. and Morgan N. (1994) "RASTA Processing of Speech", *IEEE Transactions on Speech and Audio Processing*, vol. 2, no. 4, pp. 578–589.

[31] P. Woodland, M. Gales, D. Pye and S. Young. " The Development of the 1996 HTK broadcast news transcription system". Proceedings of the Speech Recognition Workshop, pp 73-78,1997.

[32] F. Kubala et al., "The 1996 BBN Byblos Hub-4 transcription system", Proceedings of the Speech Recognition Workshop, pp 90-93, 1997.

[33] http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=1757&objectType=file

[34] G. Shwarz, "Estimating the dimension of a model", The Annals of Statistics, vol. 6, pp 461-464, 1978.

[35] H. Akaibe, "A new look at the statistical identification model". IEEE Trans. Auto. Control, vol 19, pp 716-763, 1974.

[36] D. Foster and E. George, "The risk inflation factor in multiple linear regression". Technical Report, Univ. of Texas, 1993.

[37] M. Siegler, U. Jain, B. Ray and R. Stern, "Automatic segmentation, classification and clustering of broadcast news audio". Proceedings of the Speech Recognition Workshop, pp 97-99, 1997.

[38] H. Beigi and S. Maes, "Speaker, channel and environment change detection", Proceedings of the World Congress on Automation, 19998.

[39] H. Gish and N. Schmidt, "Text-independent speaker identification". IEEE Signal Processing Magazine, pp 18-21, Oct. 1994.

[40] Hyoung-Gook Kim, Nicolas Moreau, Thomas Sikora, "MPEG-7 Audio and Beyond: Audio Content Indexing and Retrieval" Wiley, October 2005

[41] T. Kemp, M. Schmidt, M. Westphal, and A. Waibel, "Strategies for automatic segmentation of audio data," in *Proc. ICASSP*, vol. 3, 2000, pp. 1423–1426.

[42] A. Tritschler and R. Gopinath, "Improved speaker segmentation and segments clustering using the Bayesian information criterion," *Eurospeech*, pp. 679–682, 1999.

[43] P. Delacourt and C. J. Wellekens, "DISTBIC: A speaker based segmentation for audio data indexing," *Speech Commun.*, vol. 32, pp. 111–126, Sept. 2000.

[44] Jitendra Ajmera, Iain McCowan, and Hervé Bourlard, *Fellow, IEEE* Robust Speaker Change Detection ieee signal processing letters, vol. 11, no. 8, august 2004

[45]http://www.aquaphoenix.com/research/matlab

[46]http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=1757&objectType=file

[47] E. Scheirer and M. Slaney, "Construction and evaluation of a robust multifeature speech/music discriminator," *IEEE Transactions on Acoustics, Speech and Signal Processing (ICASSP'97)*, pp. 1331–1334, 1997.

[48] B. S. Manjunath Philippe Salembier Thomas Sikora " Introduction to MPEG-7 Multimedia Content Description Interface" John Wiley & Sons, LTD

[49] Johnston, J. D. 1988. "Transform Coding of Audio Signals Using Perceptual Noise Criteria," IEEE Journal of Selected Areas in Communication, Vol. 6, pp. 314-323.

[50] LeBlanc, J. and de Leon, P., "Speech Separation by Kurtosis Maximization," *Proc. IEEE ICASSP 1998*, pp. 1029-1032, 1998.

[51] Panu Korpipää , Miika Koskinen , Johannes Peltola , Satu-Marja Mäkelä , Tapio Seppänen, Bayesian approach to sensor-based context awareness, Personal and Ubiquitous Computing, v.7 n.2, p.113-124, July 2003

[52] John Saunders. Real time discrimination of broadcast speech music. In Proc. 1996 ICASSP. pages 993-996.1996

[53] Tzanetakis G. and Cook P. (2002) "Musical Genre Classification of Audio Signals",*IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293 302.

[54] Li D., Sethi I. K., Dimitrova N. and McGee T. (2001) "Classification of General Audio Data for Content-based Retrieval", *Pattern Recognition Letters, Special Issue on Image/Video Indexing and Retrieval*, vol. 22, no. 5.

[55] Wang Y., Liu Z. and Huang J.-C. (2000) "Multimedia Content Analysis Using Both Audio and Visual Cues", *IEEE Signal Processing Magazine*, vol. 17, no. 6, pp. 12–36.

[56] J. R. Deller, J. G. Proakis, and J. H. L. Hansen. Discrete-time Processing of Speech Signals. Prentice Hall, New Jersey, 1993. ISBN 0023283017.

[57] Angelini B., Falavigna D., Omologo M. and De Mori R. (1998) "Basic Speech Sounds, their Analysis and Features", in *Spoken Dialogues with Computers*, pp. 69–121, Academic Press, London.

[58]http://clam.iua.upf.edu/doc/CLAM-devel-doxygen/classCLAM_1_1SpectralDescriptors.html

[59] http://www.speech.kth.se/wavesurfer/

[60] Van Beek, P., A.B., Heuer, J., Martinez, J., Salembier, P., Smith, J. and Walker T. MPEG-7: Multimedia Description Schemes, ISO/IEC FDIS 15938-5:2001, INternational Standard Document, 2001.

[61] MPEG-7: Overview of MPEG-7 Description Tools, José M. Martínez, Copyright © 2002 IEEE. Reprinted from IEEE Computer Society, July-September 2002.

[62] ISO 8601, Representations of dates and times, 1988-06-15.

[63] Manjunath, Salembier, Sikora, "Introduction to MPEG-7 Multimedia Content Description Interface" John Wiley & Sons, LTD

[64] ISO/IEC FDIS 15938-5:2003, INternational Standard Document, 2003

[65] http://m7itb.nist.gov/M7Validation.html

[66] W. Chai. Automated analysis of musical structure. PhD thesis, Massachusetts Institute of Technology, MA, USA, September 2005.

[67] S. Abdallah, M. Sandler, C. Rhodes, and M. Casey. Using duration models to reduce fragmentation in audio segmentation. Machine Learning, 65(2):485–515, 2006.

[68] T. Kemp, M. Schmidt, M. Westphal, and A. Waibel, "Strategies for automatic segmentation of audio data," in *Proc. ICASSP*, vol. 3, 2000, pp. 1423–1426.

[69] A. Tritschler and R. Gopinath, "Improved speaker segmentation and segments clustering using the Bayesian information criterion," *Eurospeech*, pp. 679–682, 1999.

[70] P. Delacourt and C. J. Wellekens, "DISTBIC: A speaker based segmentation for audio data indexing," *Speech Commun.*, vol. 32, pp. 111–126, Sept. 2000.

[71] Giuseppe Dimattia "An Automatic Audio Classification System for Radio Newscast" TSC UPC Terrassa March 2008.