Audio Segmentation and Classification

Abdillahi Hussein Omar

Kgs. Lyngby 2005

Preface

The work presented in this thesis has been carried out at the Intelligent Signal Processing Group, at the Institute of Informatics and Mathematical Modelling, IMM, Technical University of Denmark. The thesis is the final requirement for obtaining the degree of Master of Science in Engineering. The work has been supervised by Associate Professor Jan Larsen and co-supervised by Ph.D student Peter Ahrendt. I would like to thank my supervisor for the good ideas, and Peter Ahrendt for the weekly discussions and the useful comments on my results. I would also like to thank my family, who has been very understanding all the time.

Kgs Lyngby, February 28, 2005

_____

Abdillahi Hussein Omar, s971356

Abstract

This project describes the work done on the development of an audio segmentation and classification system. Many existing works on audio classification deal with the problem of classifying known homogeneous audio segments. In this work, audio recordings are divided into acoustically similar regions and classified into basic audio types such as speech, music or silence. Audio features used in this project include Mel Frequency Cepstral Coefficients (MFCC), Zero Crossing Rate and Short Term Energy (STE). These features were extracted from audio files that were stored in a WAV format. Possible use of features, which are extracted directly from MPEG audio files, is also considered. Statistical based methods are used to segment and classify audio signals using these features. The classification methods used include the General Mixture Model (GMM) and the k- Nearest Neighbour (*k*-NN) algorithms. It is shown that the system implemented achieves an accuracy rate of more than 95% for discrete audio classification.

Keywords: *audio content analysis, segmentation, classification, GMM, k-NN, MFCC, ZCR, STE and MPEG*

Contents

List of Figures

Chapter 1

Introduction

Audio signals which include speech, music and environmental sounds are important types of media. The problem of distinguishing audio signals into these different audio types is thus becoming increasingly significant. A human listener can easily distinguish between different audio types by just listening to a short segment of an audio signal. However, solving this problem using computers has proven to be very difficult. Nevertheless, many systems with modest accuracy could still be implemented.

Audio segmentation and classification have applications in wide areas. For instance, content based audio classification and retrieval is broadly used in the entertainment industry, audio archive management, commercial music usage, surveillance, etc. There are many digital audio databases on the World Wide Web nowadays; here audio segmentation and classification would be needed for audio searching and indexing. Recently, there has been a great deal of interest in monitoring broadcast news programs, in this case classification of speech data in terms of speaker could help in efficient navigation through broadcast news archives.

Like many other pattern classification tasks, audio classification is made up of two main sections: a signal processing section and a classification section. The signal processing part deals with the extraction of features from the audio signal. The various methods of time-frequency analysis developed for processing audio signals, in many cases originally developed for speech processing, are used. The classification part deals with classifying data based on the statistical information extracted from the signals.

Two different classifiers, k-Nearest Neighbour(k-NN) and General Mixture model (GMM), were trained and tested to classify audio signals into music, speech and silence. The audio features used for classification were the Mel Frequency Cepestral Coefficients(MFCC), Zero Crossing Rates(ZCR) and Short Time Energy(STE). And for segmentation purposes Root Mean Square(RMS) features were used. The figure (figure1.1) below shows segmentation and classification of audio into three classes.

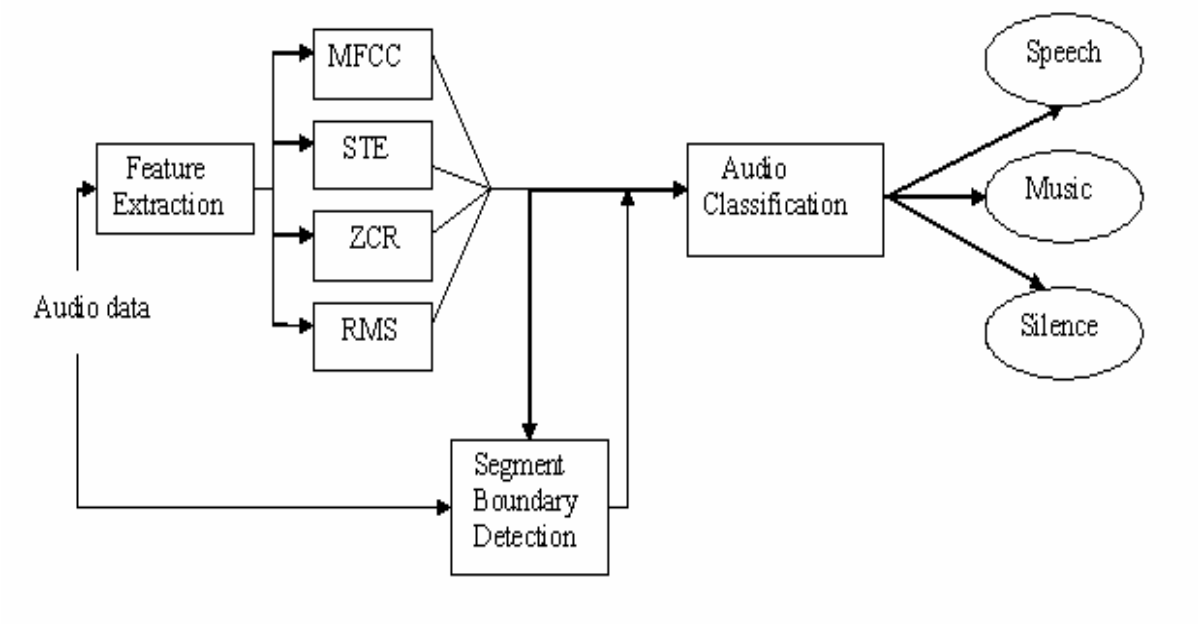

Figure 1. 1 Segmentation and classification of audio data.

1.1 Project Objective

The main goal of this project was, initially to design a system that would be able to classify audio signals into music or speech. The classification task was further to be extended to include audio signals other than speech and music. And finally the system was to be modified so that the audio signal is partitioned first into homogeneous segments and then classified. Audio characterisation based on MPEG subband level data was also to be explored.

1.2 Previous Work

Research in the field of audio segmentation and classification has gained a lot of attention these past few years. Various methods for audio discrimination have been proposed for the needs of different applications. These approaches can be categorised into those that deal with problem of classifying discrete audio samples that contain only one type of sound, and those that deal with the problem of classifying audio samples that contain different sound types. In the following paragraphs some of these methods along with their performances are presented.

E. Scheier and M.Slaney [6] used 13 features for audio classification and made experiments with different classification algorithms. In their work a correct classification rate of 98.6% is reported for a 2.4 sec window. Khaled El-Maleh *et al* [2] also proposed a speech/music classification technique based on the line spectral frequencies(LSFs). Two different classification methods (a quadratic Gaussian and nearest neighbour) were used in their work. Based on LSF features and the quadratic Gaussian classifier , which was modified to make decisions over a one second window, a correct classification rate of 90.7% was obtained. When LSF features were used in conjunction with ZCR, a performance of 94.8% is reported. Lie Lu *et el* in their work on audio classification and segmentation have used features such as high zero crossing rate, low short time energy ratio (LSTER) and spectrum flux (SF).

Hugo Meindo and J. Neto [3] in their work on audio segmentation, classification, and clustering have used symmetric Kullback-Liebler, KL2, for audio segmentation. The KL2 is calculated over 12^{th} order PLP coefficients extracted from an audio signal. The same features were used for the purpose of Speech/non-speech classification. For analysis window of 0.5 seconds a correct classification rate of around 92.6% is reported. In [5] a speech/music discriminator based on RMS and Zero-crossings is presented. Here, a correct classification rate of about 95% is obtained. Tong Zhang and Kuo [12] proposed a system that classifies audio recordings into basic audio types using simple audio features such as the energy function, average zero crossing rate and spectral peak track. An accuracy rate of more than 90% for audio classification is reported. G. Tzanetakis and P. Cook [4] presented a general methodology for temporal segmentation based on multiple features.

1.2 Project overview

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

- Chapter 2 describes feature extraction: which features have been extracted from the audio and how they are extracted is explained.

- Chapter 3 describes the classifiers : the different classification methods used in this project are outlined.

- Chapter 4 presents the segmentation method, the procedure for detecting changes based on the audio characteristic is explained.

- Chapter 5 describes digital audio formats: existing audio formats are discussed, one example of perceptually coded formats is explained and finally a tool for direct conversion of MP3 audio files into feature space is discussed.

- Chapter 6 shows experimental results, the type of data used for training and test and finally the results obtained and their implications are discussed.

Chapter 2

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.

The performance of a set of features depends on the application. The design of descriptive features for a specific application is hence the main challenge in building audio classification systems. A wide range of audio features exist for classification tasks. These features can be divided into two categories: time domain and frequency domain features. The Features considered in this chapter are: Mel Frequency Cepstral coefficient (MFCC), zero crossing rates and short time energy.

2.1 Short term features

Speech signals are considered to be slowly time varying signals. A speech signal over a period of say between 10 to 100ms has a characteristic which is fairly stationary. Over a

longer period of time, however, the signal characteristics alter to reflect the changes in the speech sounds being spoken. Although music has a larger dynamic range than speech, like speech its characteristics over a short period of time remain stationary. This notion leads to a variety of short-time processing techniques in which short segments of audio signal are isolated and processed as though they were short segments from a sustained audio with fixed properties. This process of segmenting audio signals in to frames is repeated, usually periodically, as often as required. Normally these short segments, which are sometimes referred to as analysis frames, overlap one another. After processing is done on each frame, a single number or a set of numbers may be obtained. Hence, such processing results in a new time dependant sequence which can serve as representation of the audio signal.

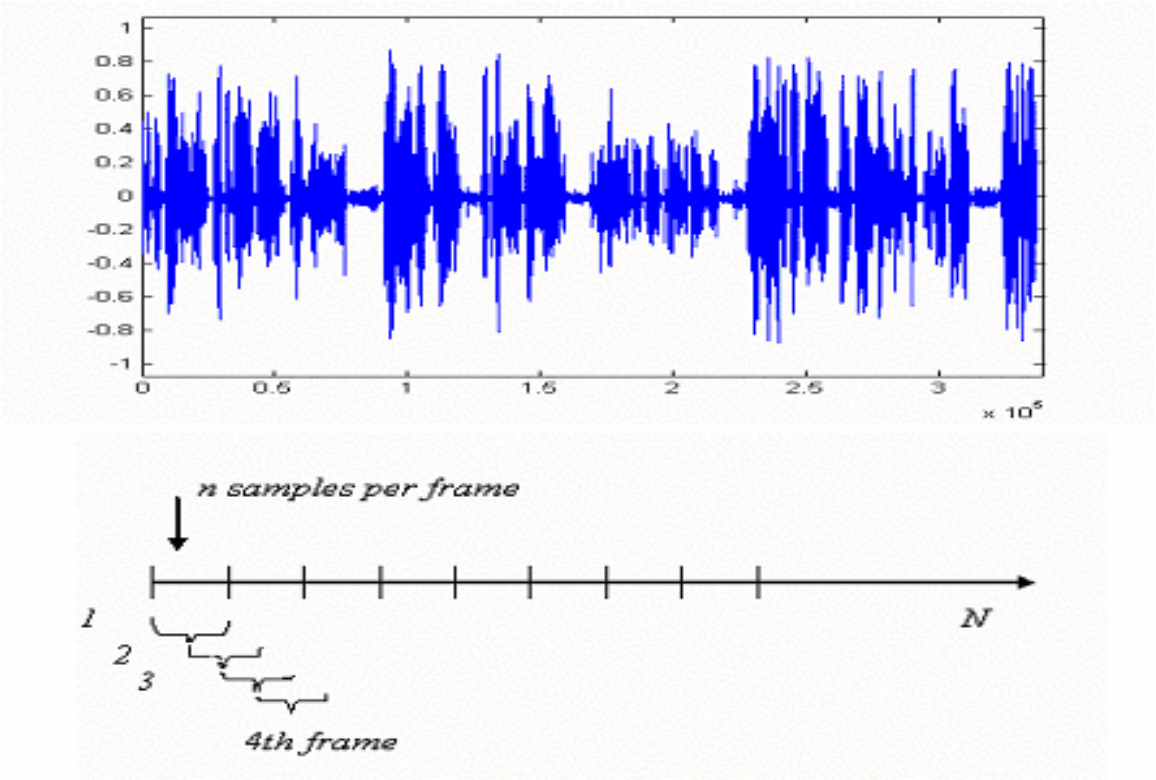

Figure 2. 1 Plot of an audio signal together with a plot that shows how the audio signal could be segmented into overlapping frames.

2.1.1 Short Time Energy

The energy *E* of a discrete time signal *x(n)* is defined by the expression (2.1). For many audio signals such a measurement is of less importance, since it gives little information about time dependent characteristics of such signals.

$$E = \sum_{n=-\infty}^{\infty} x^2(n) \qquad (2.1)$$

As mentioned earlier, the amplitude of an audio signal varies with time. A convenient representation that reflects these amplitude variations is the short time energy of the signal. In general, the short time energy is defined as follows

$$E_m = \sum_n \left[x(n)\, w(m-n)\right]^2 \qquad (2.2)$$

The above expression can be rewritten as

$$E_m = \sum_n x(n)^2\, h(m-n) \qquad (2.3)$$

where $h(m) = w^2(m)$

In the above expression the term $h(m)$ is interpreted as the impulse response of a linear filter. The nature of short time energy representation is determined by the choice of the impulse response, $h(m)$. The bandwidth of the hamming window, as it can be seen in the two figures below, is twice the bandwidth of a rectangular window of the same length. Moreover the hamming window results in a much higher attenuation outside the bandwidth when compared to the rectangular window. However, in both cases, increasing the length of the window decreases the bandwidth. For speech signals, the duration of a pitch period varies from 20 samples, at a sampling rate of around 10 KHz, for a high pitch female or a child, up to 250 samples for a very low pitch male. With this in mind, for a 10 KHz sampling rate a practical choice of the window length is on the order of 100 to 200 samples.

Short term energy is used in different audio classification problems. In speech signals, it provides a basis for distinguishing voiced[1] speech segments from unvoiced ones. In the case of a very high quality speech, the short term energy features are used to distinguish speech from silence.

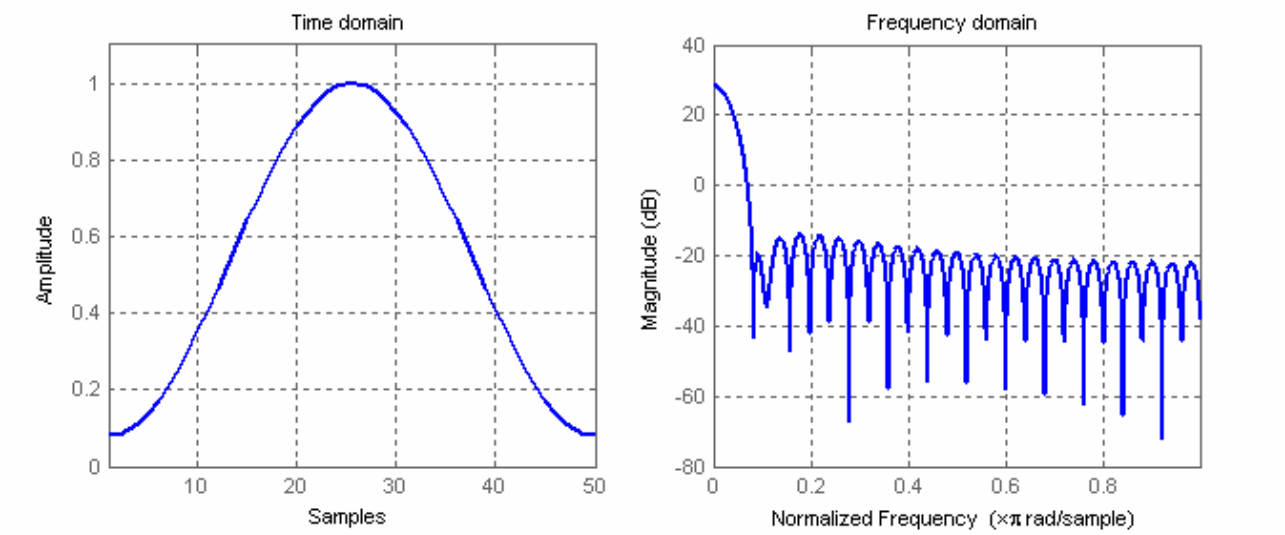

Figure 2. 2 50- point Hamming window

[1] Based on their mode of excitation, speech sounds can be classified as: voiced, unvoiced and plosive sounds. Voiced sounds are produced by forcing air through the glottis with the tension of the vocal cords adjusted so that they vibrate in a relaxation oscillation. Unvoiced sounds are produced by forming constrictions at some point in the vocal tract, and forcing air through the constriction at a high velocity enough to produce turbulence.

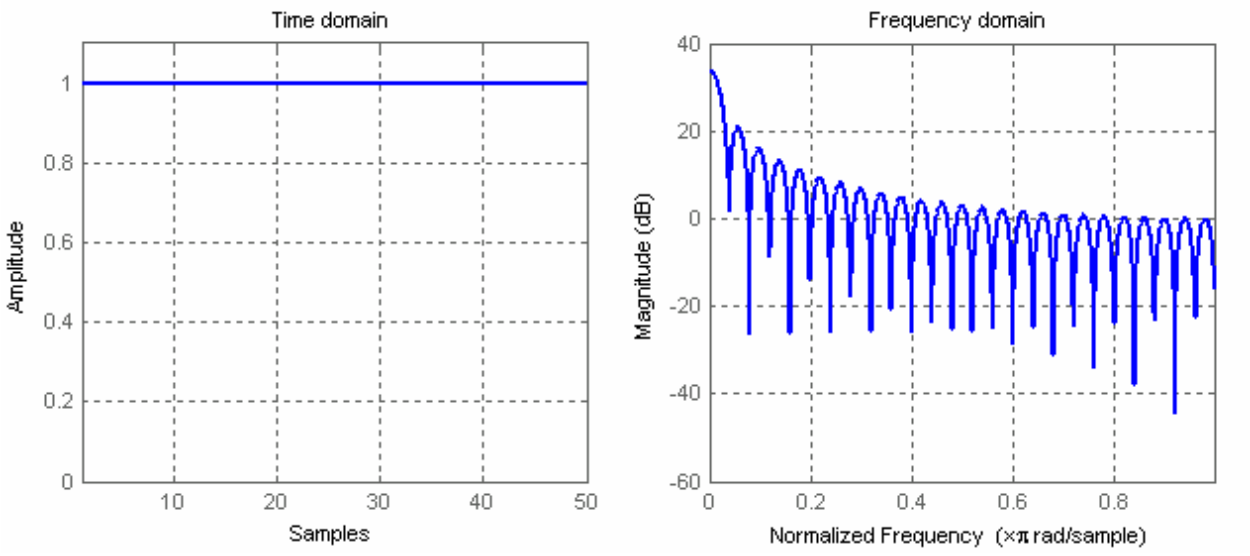

Figure 2. 3 50-point rectangular window

2.1.2 Zero Crossing Rates

In the case of discrete time signals, a zero crossing is said to occur if there is a sign difference between successive samples. The rate at which zero crossings happen is a simple measure of the frequency content of a signal. For narrow band signals, the average zero crossing rate gives a reasonable way to estimate the frequency content of the signal. But for a broad band signal such as speech, it is much less accurate. However, by using a representation based on the short time average zero crossing rate, rough estimates of spectral properties can be obtained. The expression for the short time average zero crossing rate is shown below. In this expression, each pair of samples is checked to determine where zero crossings occur and then the average is computed over *N* consecutive samples.

$$Z_m = \sum_n \left| sign[x(n)] - sign[x(n-1)] \right| w(m-n) \qquad (2.4)$$

where the sign function is $sign[x(m)] = \begin{cases} 1 & x(m) \geq 0 \\ -1 & x(m) \angle 0 \end{cases}$,

$$w(m) = \begin{cases} \frac{1}{2N} & 0 \le m \le N-1 \\ 0 & otherwise \end{cases}$$ and $x(n)$ is the time domain signal for frame *m.*

Zero crossing rate has been proven to be useful in characterising different audio signals and has been popularly used in speech/music classification problems. Variations of the zero crossing rate have also been used in some audio classification systems. In[1], it is suggested that a variation of the ZCR- the high zero-crossing rate ratio(*HZCRR*), to be more discriminative than the exact value of ZCR.

2.1.3 Mel Frequency Cepstral Coefficient

MFCCs are short term spectral based features. MFCC features are frequently used by many researchers for speech recognition and it has also been shown in [13] that MFCC works well in music/ speech classification problem. A block diagram showing the steps taken for the computing MFFCs can be seen in figure 2.4. Each step in this process of creating Mel Frequency Cepstral Coefficients is motivated by computational or perceptual considerations.

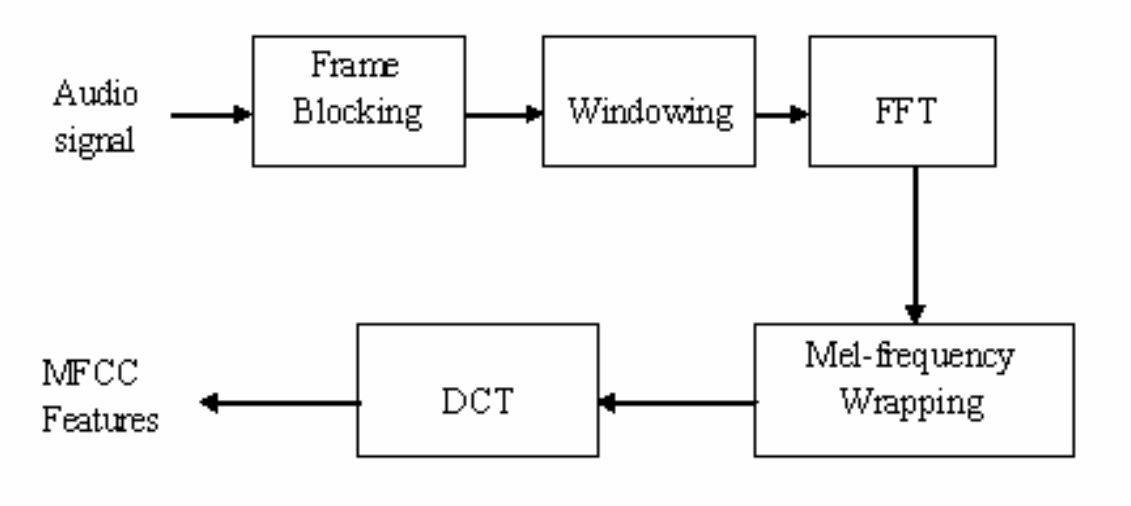

Figure 2. 4 Block diagram showing the steps for computing MFCCs

The first step in this process is to block a continuous audio signal into frames. The purpose here is to model small sections of the audio signal that are statistically stationary. Each frame consists of *n* samples with adjacent frames separated by *m* samples. The following frame starts *m* samples after the first sample and overlaps it by *(n - m)* samples. In a similar way the third frame starts *m* samples after the second frame and overlaps it by *(n - m)* samples. Typical values for *n* and *m* are 256 and 100 respectively.

The next step is to use a window function on each individual frame in order to minimise discontinuities at the beginning and end of each frame. Typically the window function used is the Hamming window and has the following form:

$$w(n) = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), & 0 \le n \le (N-1) \\ 0 \qquad\qquad , \textit{otherwise} \end{cases} \qquad (2.5)$$

Given the above window function and assuming that there are *N* samples in each frame, we will obtain the following signal after windowing.

$$y(n) = x(n)\,w(n) \quad , \quad 0 \le n \le (N-1) \qquad (2.6)$$

The next step is the process of converting each frame of *N* samples from the time domain to the frequency domain. Here we will take the Discrete Fourier Transform of each frame. We use the FFT algorithm, which is computationally efficient, to implement the DFT. As the amplitude of the spectrum is much more important than the phase, we will retain only the amplitude spectrum. The Discrete Fourier Transform on the set of *N* samples is defined as follows [15].

$$X(k) = \sum_{n=0}^{N-1} x(n)\, e^{\frac{-j2\pi kn}{N}} \quad , \quad k = 0,1,2,\ldots\ldots\ldots,(N-1) \qquad (2.7)$$

The next step is the transformation of the real frequency scale to the mel frequency scale. A mel is a unit of measure of *perceived pitch or frequency* of a tone [14]. The mel- frequency is based on the nonlinear human perception of the frequencies of audio signals. It is a linear frequency spacing below 1KHz and logarithmic above this frequency. By taking the *pitch* of the 1 KHz tone as a reference and assigning it 1000 mels, and then making some test measurements based on human perception of audio signals, it is possible to drive a model

for an approximate mapping of a given real frequency to the mel frequency scale. The following is an approximation of the mel frequency based on such experiments.

$$Mel(f) = 2595 \log_{10}(1 + \frac{f}{700}), \tag{2.8}$$

Where f is the physical frequency in *Hz* and *Mel* is the perceived frequency in *mels*.

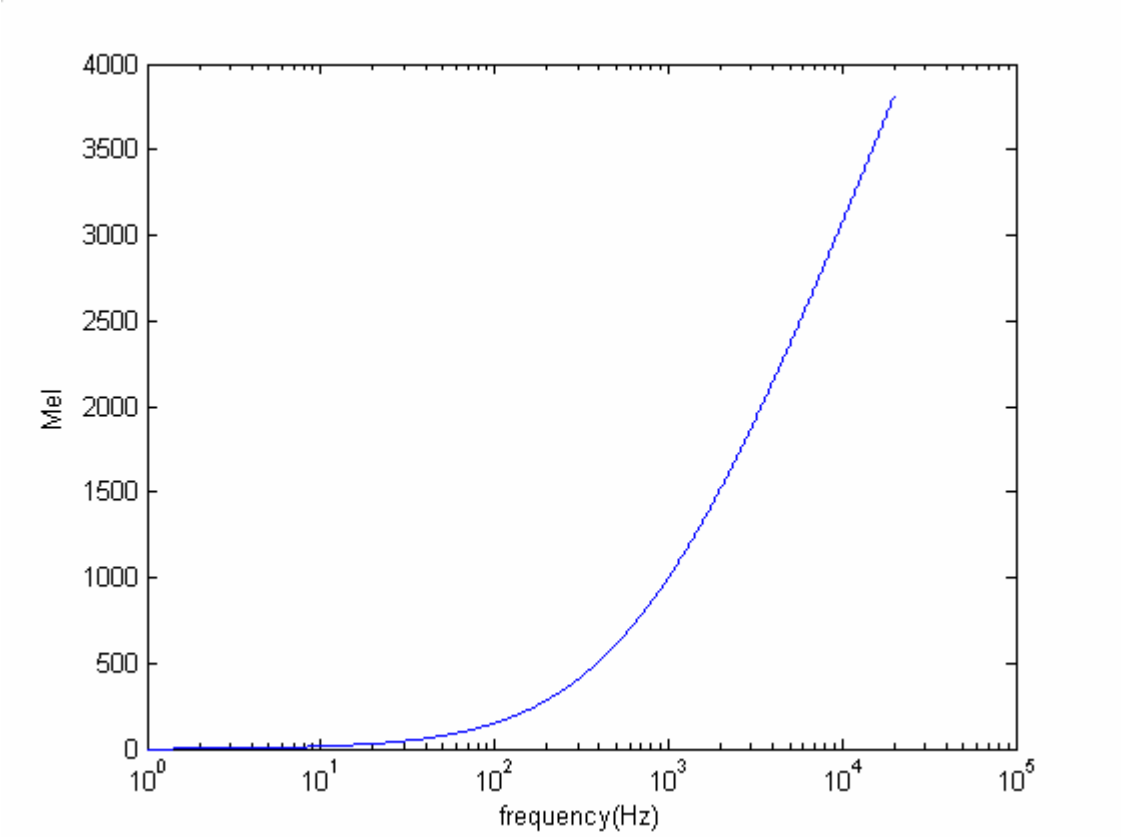

Figure 2. 5 Mel scale mapping

The idea of a perceptual frequency scale has led to the investigation of the benefits of using a frequency axis that is warped to correspond to the mel scale. One of the techniques used

to obtain the new frequency axis is to use a filter bank. Here, one filter for each desired mel frequency component is designed. Usually the filters have a triangular band pass frequency response and are evenly spaced on the mel scale. Since the filter bank is applied in the frequency domain, the modified spectrum of the signal thus consists of the *output power* of these filters when the input is the signal obtained at the previous step. It has been found that the perceived loudness of audio signals to be approximately logarithmic and hence the logarithm of the power spectrum is taken. Figure (2.6) shows a plot of mel spaced filter bank.

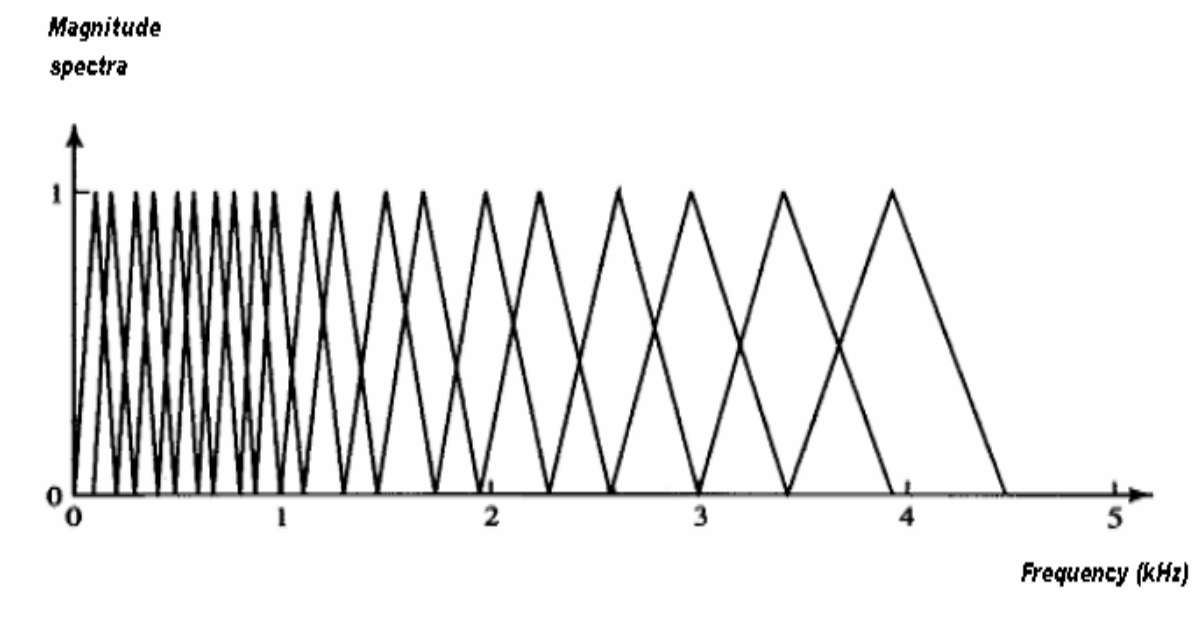

Figure 2. 6 Frequency response of a mel-spaced filterbank

The next step is the final stage of the Mel Frequency Cepstral feature construction. In this stage, the log mel spectrum is converted back to the time domain and the result is the Mel Frequency Cepstral Coefficients. The components of the mel spectral vectors calculated are highly correlated. In order to reduce the number of the parameters, some transform, which decorrelates their components, is applied to these vectors. Theoretically, the Karhunen-Loeve (KL) works well for this purpose. However, the KL is very complex and since the DCT can still end in good results, the DCT is frequently used. After the DCT, some 13 cepstral features are obtained for each frame.

Lets now denote the mel spectral coefficients obtained in the previous step as $\hat{S}_k$, $k = 1,2,....,K$, then we can calculate the MFCCs as follows

$$\hat{C}_n = \sum_{k=1}^{K} \hat{S}_k * \cos\left[n * \left(k - \frac{1}{2} \right) * \frac{\pi}{K} \right], \quad n = 1,2,....,K \qquad (2.9)$$

2.2 Summary

In this chapter the main audio features used in speech/music classification systems have been considered. MFCC features are frequently used by many researchers for speech recognition and others have also used MFCC features in music genre classification problems. Short term energy is widely used in different audio classification schemes. Zero crossing rate, being a simple measure of the frequency content of a signal, is also used to distinguish between different audio types.

Chapter 3

Audio Classification

In this chapter, the problem of classifying the extracted audio features into one of a number of audio classes is considered. The basic classification task can be considered as a process where a previously unknown input data is assigned to a class $C \in \{C_1, C_2, ..., C_n\}$. Such assignments are made by establishing and applying a decision rule; for example, a simple decision rule could be the assignment of a new data sample to a class whose mean it is closest to in feature space.

Classification algorithms are divided into supervised and unsupervised algorithms. In a supervised classification, a labelled set of training samples is used to "train" the algorithm whereas in the case of an unsupervised classification the data is grouped into some clusters without the use of labelled training set. Parametric and nonparametric classification is another way of categorizing classification algorithms. The functional form of the probably density of the feature vectors of each class is known in parametric methods. In non parametric methods on the other hand, no specific functional form is assumed in advance, instead the probability density is rather approximated locally based on the training data.

3.1 The Gaussian classifier

The Gaussian classifier is an example of a parametric classifier. This classifier is based on

the assumption that feature vectors of each class obey a multidimensional Gaussian distribution. In the training stage, estimates of the parameters (mean and covariance) of the Gaussian probability density functions of each class are computed using the training data. In the classification stage the input vector is mapped to the class with the largest likelihood. In *d*-dimensions, the probability density function is written as:

$$p(X) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(X-\mu)^{'}\ \Sigma^{-1}\ (X-\mu)\right\} \qquad (3.1)$$

where Σ is a $d \times d$ covariance matrix and the mean μ is a *d* dimensional vector.

3.2 The K Nearest Neighbour classifier

The K nearest neighbour classifier is an example of a non parametric classifier. The basic algorithm in such classifiers is simple. For each input feature vector to be classified, a search is made to find the location of the K nearest training examples, and then assign the input to the class having the largest members in this location. Euclidean distance is commonly used as the metric to measure neighbourhood. For the special case of K=1 we will obtain the nearest neighbour classifier, which simply assigns the input feature vector to the same class as that of the nearest training vector. The Euclidean distance between feature vectors $X = \{x_1, x_2, ..., x_n\}$ and $Y = \{y_1, y_2, ..., y_n\}$ is given by:

$$d = \sqrt{\sum_{i=1}^{n}(x_i - y_i)} \qquad (3.2)$$

The KNN algorithm, as mentioned earlier, is very simple yet rather powerful, and used in many applications. However, there are things that need to be considered when KNN classifiers are used. The Euclidean distance measure is typically used in the KNN algorithm. In some cases, use of this metric might result in an undesirable outcome. For instance, in cases where several feature sets (where one feature set has relatively large values) are used as a combined input to a KNN classifier, the KNN will be biased by the larger values. This leads to a very poor performance. A possible method for avoiding this problem would be to normalise the feature sets.

In Figure 3.1, an example of a three class classification task is shown. The aim is to use the KNN classifier for finding the class of an unknown feature **X**. As it can be seen in the

figure, of the closest neighbours (K=5 neighbours) four belong to *class a* and only one belongs to *class b* and hence **X** is assigned to *class a*.

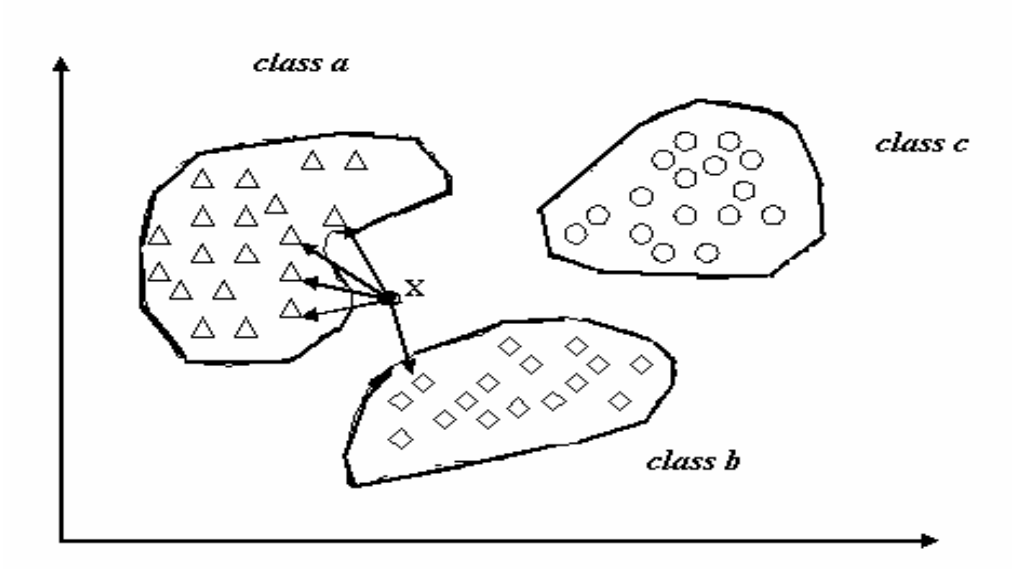

Figure 3. 1 The K nearest neighbourhood rule (K=5)

Some of the disadvantages of the K nearest neighbour classifiers are:

- Need the entire feature vectors of all training data when a new vector feature is to be classified, and hence large storage requirements.
- The classification time is longer when compared to some other classifiers.

The K nearest neighbour classifiers have some qualities that are important such as

- It requires no training and this is helpful especially when a new training data is added.
- Uses local information and hence can learn complex functions without needing to represent them explicitly.

3.3 The GMM classifier

The General Mixture Model (GMM) classifier is a type of classifier which combines the advantages of parametric and non parametric methods. As the name indicates, the density function is a form of density function known as mixture model. A brief description of the classifier is given in the following paragraphs.

Given a *d*-dimensional vector ***X,*** a Gaussian mixture density is a weighted sum of *M* component densities and can be written as equation (3.3). The number *M* of components is treated as a parameter of the model and is typically much less than the number *N* of data points.

$$p(X/\mu,\Sigma,p)=\sum_{j=1}^{M}p_j b_j(X) \qquad (3.3)$$

where $p_j \le 0,\ j=1,2,....,M$ are the mixture weights, with $\sum_{j=1}^{M}p_j=1$ and $b_j(X)$ are the component densities each with mean vector μ_j and covariance matrix $\sum_j$.

$$b_j(X)=\frac{1}{(2\pi)^{d/2}\left|\Sigma_j\right|^{1/2}}\exp\left\{-\frac{1}{2}\left(X-\mu_j\right)'\ \Sigma_j^{-1}\left(X-\mu_j\right)\right\}$$

For the Gaussian mixture model given in equation (3.3), the mixture density is parameterised by the mean vectors, covariance matrices and mixture weights from all component densities.

$$\theta=\left\{p_j,\mu_j,\Sigma_j\right\},\qquad j=1,2,...M$$

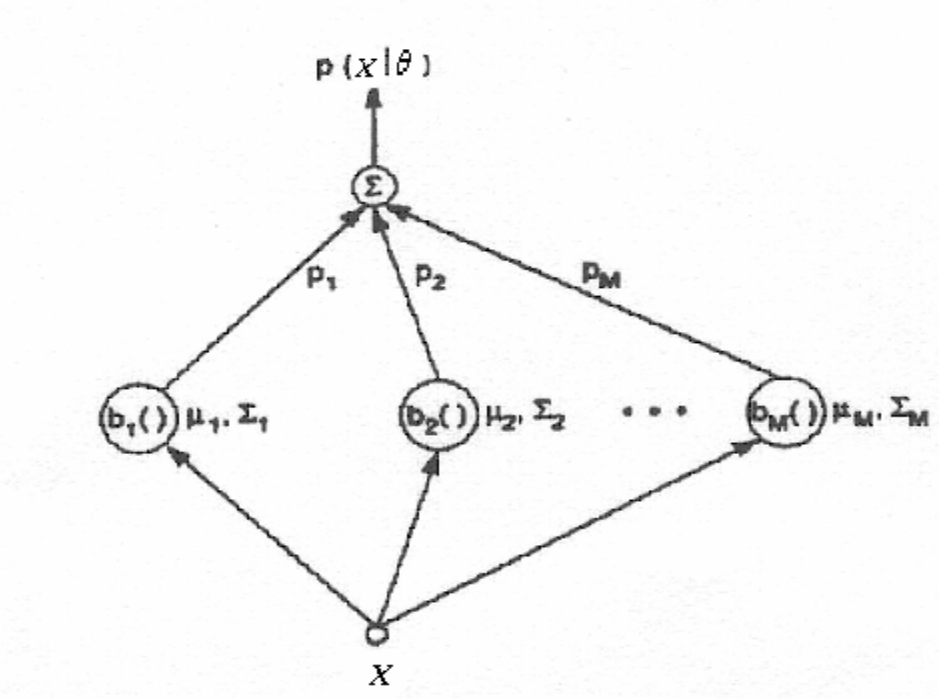

Figure 3. 2 Representation of an *M* component mixture model

General Mixture Models can assume many different forms, depending on the type of covariance matrices. The two mostly used are the full and diagonal covariance matrices. When the type of the covariance matrix is diagonal, the number of parameters that need to be optimised are reduced. This constraint on the matrices reduces the modelling capability and it might be necessary to increase the number of components. However, in many applications this compromise has proven worthwhile.

For audio classification, the distribution of the feature vectors extracted from a particular audio class is modelled by a mixture of *M* weighted multidimensional Gaussian distributions. Given a sequence of feature vectors from an audio class, maximum likelihood of the parameters are obtained using the iterative Expectation Maximization (EM) algorithm. The basic idea of the EM algorithm is, beginning with an initial model θ , to estimate a new model $\theta^{'}$, such that $p(X/\theta') \geq p(X/\theta)$. The new model then becomes the initial model for the next iteration. The process is continued until some convergence threshold is reached. The class of an unknown audio sample can then be obtained with the log likelihood ratio. By assuming equal prior for each class, points in feature space for which the likelihood is relatively high are classified as belonging to that class. The log-likelihood ratio for speech/music classification can be expressed as follows:

$LR = \frac{p(X/\theta_M)}{p(X/\theta_S)}$, where $p(X/\theta_M)$ is the likelihood given that the feature vector X is from music class and $p(X/\theta_S)$ is the likelihood given that the feature X is from speech class. The likelihood ratio in log domain is given as : $LLR = \log\left(\frac{p(X/\theta_M)}{p(X/\theta_S)}\right)$. If the value of LLR is greater than 0, then the unknown audio sample belongs to the music class otherwise it belongs to the speech class.

3.4 Summary

In this chapter different classification algorithms have been discussed. The classification algorithms were categorized into parametric and non-parametric methods. The k-nearest neighbour classifier is a simple yet powerful classification method. However, the classification time is longer when compared to some other classifiers, and requires storage of the entire training vectors. The general mixture model requires estimation of the parameters of a model and hence is computationally complex. Contrary to the k-NN, the GMM does not require storage of training vectors and is much faster.

Chapter 4

Audio segmentation

Systems that are designed for classifying audio signals usually take segmented audios rather than raw audio data as input. In order to get segmented audios from a given audio stream that contains different types of sounds, boundaries between the different audio types have to be marked. The process of detecting, if there is any change in the characteristics, the boundaries in an audio signal is referred as segmentation. Changes in audio signal characteristics such as the entrance of a guitar solo or a change from spoken words to music are some examples of segmentation boundaries.

Temporal segmentation, contrary to classification, does not interpret data and hence can be more easily modelled using mathematical techniques. Several approaches to segmentation, based on different features, have been proposed. In [4], a general methodology based on multiple features is explained. In their work, the basic features used include features such as RMS, Zero-crossings and Spectral Flux and the actual features used are the means and variances taken over one second windows. In [5], segmentation is implemented based on the mean signal amplitude distribution. In this project the method described in [5] is implemented.

4.1 Root Mean Square of audio signals

For a short audio signal (frame) consisting of N samples, the amplitude of the signal measured by the Root Mean Square is described by equation (4.1). RMS is a measure of the loudness of an audio signal and since changes in loudness are important cues for new sound events it can be used in audio segmentation. In this project the distribution of the RMS features are used to detect boundaries between speech and music signals. The method for detecting boundaries is based on the dissimilarity measure of these amplitude distributions. In figures (4.1) and (4.2) below plots of the RMS together with histograms for music and speech signals are shown.

$$A = \sqrt{\left(\sum_{i=1}^{N} x^2(i) \right)} \qquad (4.1)$$

Given a discrete audio signal x, the signal is split into short non overlapping frames and the Root Mean Square is calculated for each frame. The window size is chosen to a certain value based on the application. In our implementation the window size is set to 512 samples, i.e. with a sampling frequency of 22050 Hz, these windows are approximately 23ms long.

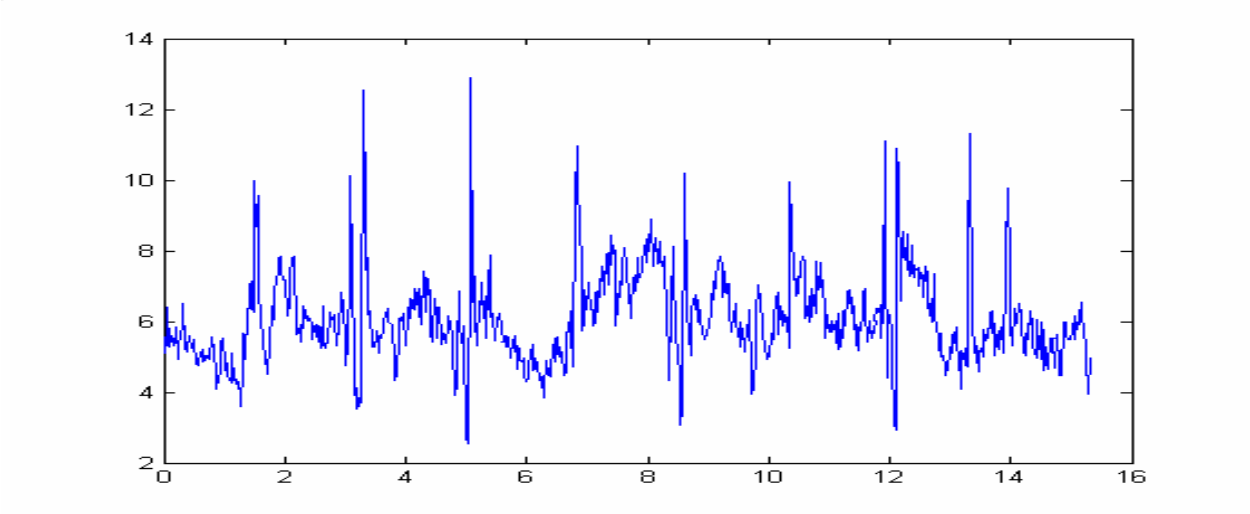

Figure 4. 1 The RMS of a music signal

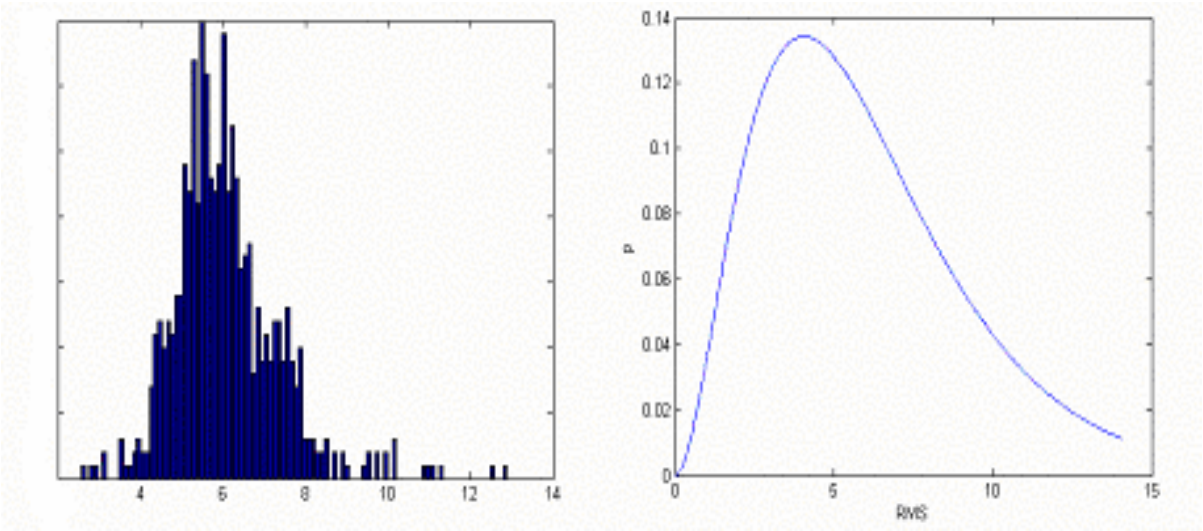

Figure 4. 2 Histogram of the RMS of the music signal together with the distribution

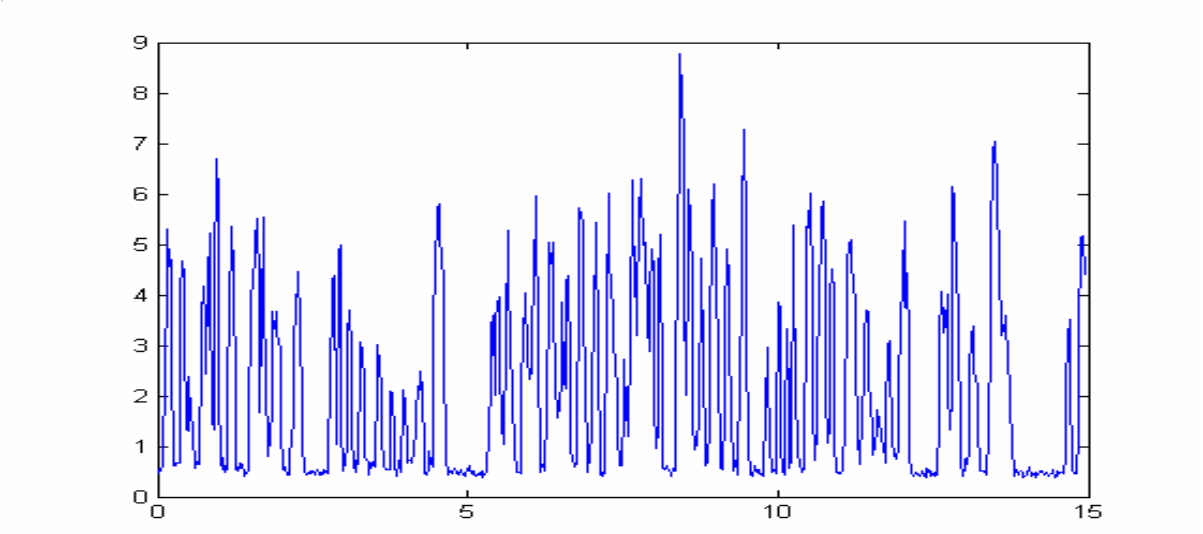

Figure 4. 3 The RMS of a speech signal

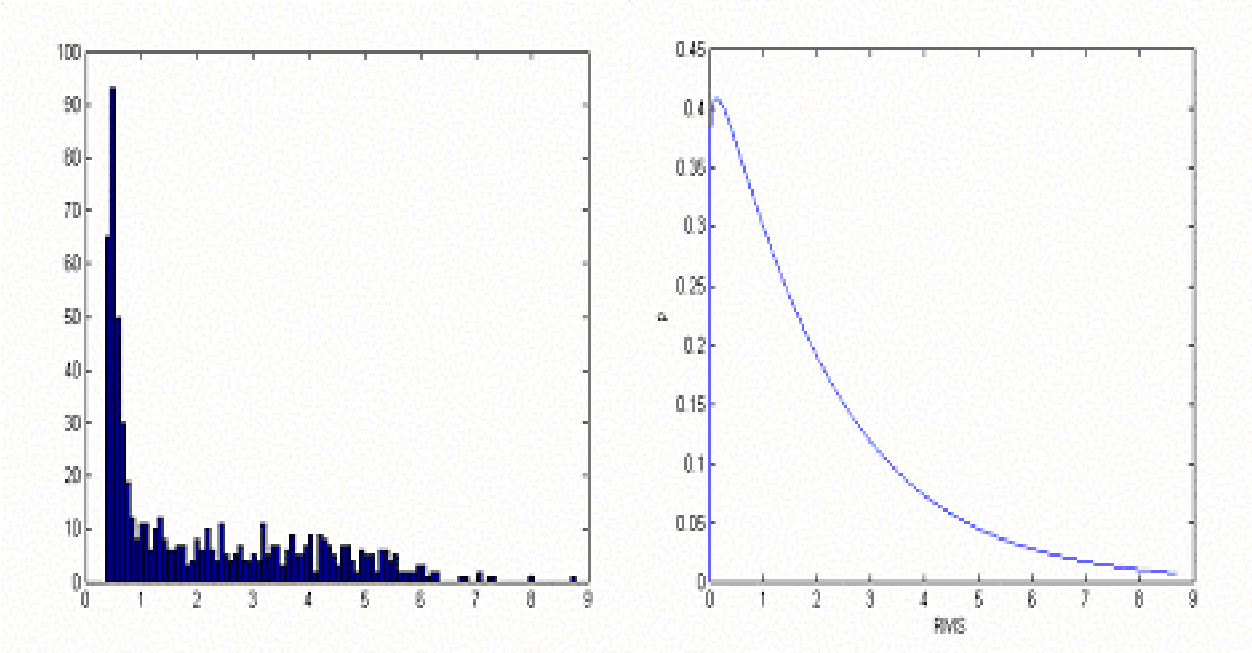

Figure 4. 4 Histogram of the amplitude of a speech signal together with the amplitude distribution

4.2 Transition detection

As mentioned earlier, in segmentation the aim is to find if there is any important difference between the distributions of two consecutive windows and thereby a transition from one audio type to another. This is done by measuring the distance between the distributions of a pair of windows at a time. Various acoustic distance measures have been defined to evaluate similarities between two adjacent windows. The feature vectors in each of the two adjacent windows are assumed to follow some probability densities and the distance is defined by the dissimilarity of these two densities. Some of the similarity measures frequently used in audio segmentations are: the Kullback-Liebler (KL), Mahalanobis distance, and Bhattacharyya distance. In [3] the symmetric Kullback-Liebler is used to evaluate acoustic similarity. The similarity measure discussed in this project is based on the Bhattacharyya distance given by the following equation

$$\rho(p_1, p_2) = \int \sqrt{p_1(x)\, p_2(x)}\, dx \qquad (4.2)$$

Since measuring distances between distributions is costly, models which are appropriate for the distributions are found and the task is reduced to the estimation of the parameters of the model. As seen in figures 4.2 and 4.4 the *chi- squared* distribution fits both music and speech amplitude distribution well and hence, it is used in our segmentation task. The generalised *chi- squared* distribution is defined by the probability density function shown in Equation (4.3).

$$p(x) = \begin{cases} \dfrac{x^a e^{-bx}}{b^{(a+1)}\Gamma(a+1)} & x \geq 0 \\ 0 & x < 0 \end{cases} \qquad (4.3)$$

The parameters a , b are related to the mean and the variance of the Root Mean Square as follows.

$a = \dfrac{\mu^2}{\sigma^2} - 1$ and $b = \dfrac{\sigma^2}{\mu^2}$.

According to equation(4.2) the similarity measure has a value that lies between zero and one. For completely identical distributions a value of one is obtained and on the other end a value of zero is obtained for two completely different distributions. The value $(1-\rho)$ is chosen to interpret the characteristics of the two windows to be compared. For two Root Mean Square distributions that are described by *chi–squared* distribution the similarity measure can be written as a function of the parameters a and b as shown in equation(4.4) below.

$$\rho(p_1, p_2) = \frac{\Gamma(\frac{a_1 + a_2}{2} + 1)}{\sqrt{\Gamma(a_1+1)\Gamma(a_2+1)}} \frac{2^{\left(\frac{a_1+a_2}{2}\right)+1} b_1^{\left(\frac{a_2+1}{2}\right)} b_2^{\left(\frac{a_1+1}{2}\right)}}{(b_1+b_2)^{\left(\frac{a_1+a_2}{2}\right)+1}} \qquad (4.4)$$

Based on the above equation, a possible transition with in a given frame k can be found. Hence for each window k a value $D(k)$ which gives a probable transition within that window is computed as follows.

$$D(k) = 1 - \rho(p_{k-1}, p_{k+1}) \qquad (4.5)$$

The above function emphasises on the fact that, a single change within a frame k implies a difference in the characteristics of the two immediate neighbours, frames $k-1$ and $k+1$. But

for an instantaneous change within a frame, the neighbouring frames *k-1* and *k+1* will have similar characteristics and therefore the factor $\rho(p_1, p_2)$ will have a value close to one whereas the value obtained for $D(k)$ will be small. Any change from speech to music or very large changes in volume, such as change from audible sound to silence, locally maximizes the $D(k)$. These changes can be detected by setting a suitable threshold. Since large changes are expected in neighbouring frames of a change frame, some sort of filtering and normalisation is needed. Using the Equation(4.6), we can compute the normalised distance.

$$D_{norm}(k) = \frac{D(k)V(k)}{D_{\max}(k)} \tag{4.6}$$

In the above equation, the variable $V(k)$ denotes the positive difference of $D(k)$ from the mean of the neighbouring frames and in the case of a negative difference it is set to zero. The maximal value of the distances in the same neighbourhood of the examined frame is given by $D_{\max}(k)$. In this project two frames after and two frames before the current one are chosen as neighbourhood. By setting a threshold, it is possible to find the local maxima of $D_{norm}(k)$ and at the end we can detect the change candidate frame. In some cases (for example when the threshold is small) false detection would be easily generated leading to over segmentation of the audio signal. However, in the case where segmentation is followed by classification, over segmentation does not lead to serious problems.

4.3 Summary

In this chapter the method for finding the time of transition between one audio type to another in long audio recordings have been presented. The method is based on the probability distribution of the root mean square features of music and speech audio signals. The Bhattacharyya distance is used as a similarity measure.

Chapter 5

Perceptually coded audio

These days, digital audio is available in many different formats. Some of the common audio formats are: WMA, MP3, Pulse Code Modulation (PCM), etc. However due to bandwidth, the most interesting formats are the perceptually coded formats.

The purpose of this chapter is to explore existing audio content analysis approaches in compressed form. Specifically, the kinds of information accessible in an MPEG-1 compressed audio stream and how to determine features from these are examined. Before the MPEG-1 standard is examined, it is a good idea first to look at the way humans perceive sound. The following is a brief introduction on this topic.

5.1 Perceptual Coding

In audio, video and speech coding the original data are analog signals that have been transformed into the digital domain using sampling and quantization. The signals are intended to be stored or transmitted with a given fidelity, not necessarily without any distortions. Optimum results are typically obtained using a combination of removal of data which can be reconstructed and the removal of data that are not important.

In the case of speech coding a model of vocal tract is used to define the possible signals that can be generated in the vocal tract. Very high compression ratios can be achieved by considering parameters that describe the actual speech signal. For generic audio coding

however, this method leads only to a limited success. This is due to the fact that other audio signals such as music signals have no predefined method of generation. Hence, source coding is not a practical approach to generic coding of audio signals.

Perceptual coding is different from source coding in that the emphasis is on the removal of only the data that are irrelevant to the auditory system. The main question in perceptual coding is therefore: How can data be removed while keeping distortion from being audible. Answers to this question can be obtained from Psychoacoustics. Psychoacoustics describes the relationship between acoustic events and the resulting audio sensation. Some relevant concepts about Psychoacoustics are given in the following.

Critical bands are important notions in Psychoacoustics. The concept of critical bands is related to the processing and propagation of audio signals in human auditory system. Several experimental results have revealed that the inner ear in humans behaves as a bank of bandpass filters which analyse a broad spectral range in subbands, called critical bands, independently from others. A perceptual unit of frequency, Bark, has been introduced and is related to the width of a single bandwidth. A commonly used transformation to this scale of hearing is given by the following relation.

$$b = 13 \times \arctan(0.76 \times 10^{-3} \times f) + 3.5 \times \arctan\left[\left(\frac{f}{7500}\right)^2\right]$$

where b and f denote the frequency in Barks and Hertz respectively.

Masking is another concept in Psychoacoustics used to describe the effect by which a fainter, but distinctly audible signal, the 'maskee', becomes inaudible when relatively louder signal, the 'masker', occurs simultaneously. This phenomenon is fundamental for audio coding standards. Masking depends both on the frequency composition of both the masker and the maskee as well as their variation with time. Masking in frequency domain plays an important role and hence is applied very often. In general, the masking effect is dependant on the intensities of the masker and the maskee tones as well as their frequencies. This relation is best described in the frequency domain by the masking curves defined for maskers of given intensity and frequency. All components that lay below these curves are masked and therefore become inaudible. Figure 5.1 shows an example of masking curves computed versus frequency in Barks.

As already mentioned before, because of the masking effects the human ear is able to perceive only a part of the audio spectrum. In perceptual audio coding therefore, a perceptual coder is used for computation of masking thresholds and bit allocation is

performed in a way that avoids bits to be wasted representing sounds that would not be perceived.

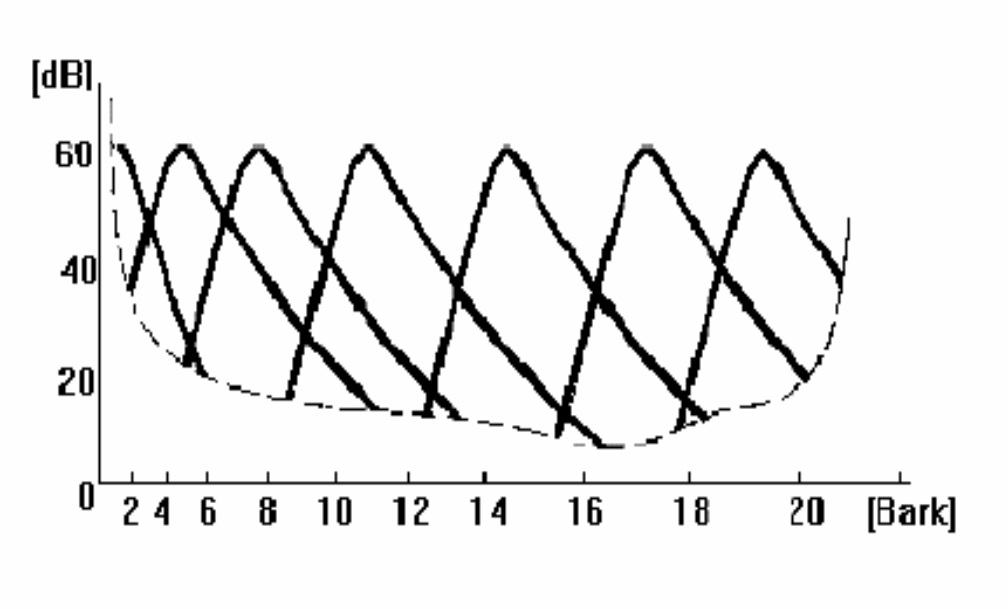

Figure 5. 1 Plot of masking curves as function of Bark frequency

5.2 MPEG Audio Compression

MPEG audio compression algorithm is the standard for digital compression of high fidelity audio. Unlike source model based coders, the MPEG audio compression technique makes use of the perceptual limitations of the human auditory system. Much of the compression results from the removal of perceptually irrelevant audio parts. As mentioned earlier, removal of perceptually irrelevant audio parts results in inaudible distortions. Based on this method, the MPEG audio can compress any signal meant to be heard by humans. MPEG audio offers divers audio coding standards :

- MPEG-1 denotes the first phase of MPEG standard . It was designed to fit the demands of many applications including digital radio and live transmission of audio via *ISDN*. MPEG-1 audio consists of three operating modes called layers: Layer 1, Layer 2 and Layer 3. Layer 1 forms the basic algorithm whereas layer 2 and Layer 3 are rather extensions that use the basic algorithm found in Layer 1. The

compression performance gets better for each successive layer but at a cost of greater encoder complexity.

- MPEG-2 denotes the second phase of MPEG standard. The main application area for MPEG-2 is digital television. It consists of two extensions to MPEG-1 audio.

    - Coding of Multichannel audio signals. The multichannel extension is done in a back ward compatible way allowing MPEG-1 decoders to reproduce a mixture of all available channels.

    - Coding at lower sampling frequencies: sampling frequencies of 16 kHz, 22.05 kHz and 24 kHz is added to the sampling frequencies supported by the MPEG-1.

In the following, a short description of the coding methods for the three MPEG-1 layers is given.

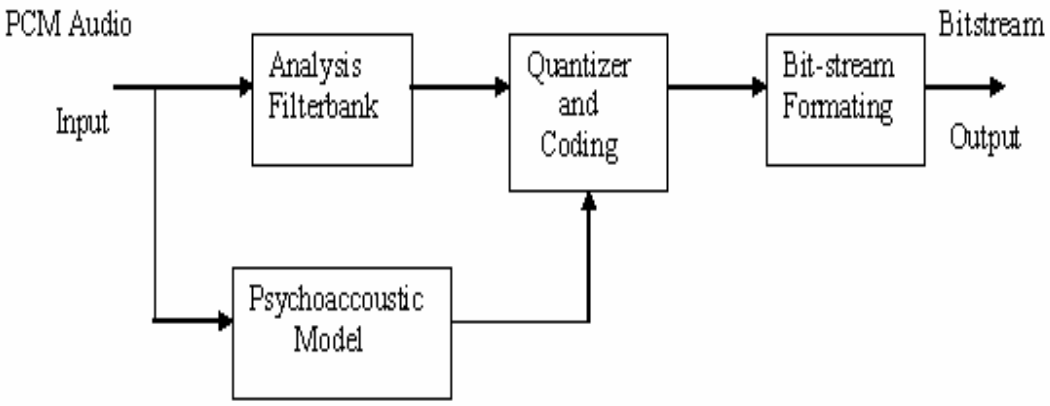

Figure 5. 2 Block diagram of MPEG encoding

In Layer 1 an Layer 2, the coding method consists of a segmentation part to format the data into blocks, a basic polyphase filter bank, a psychoacoustic model to determined the desired bit allocation and a quantization part.

The polyphase filterbank is used to compute 32 frequency band magnitudes (subband values). The filter bank used in MPEG-1 uses a 511-tap prototype filter. Polyphase filter

structures are computationally very efficient and are of moderate complexity. However, the filters are equally spaced and hence the frequency bands do not correspond well to the critical band partition. The impulse response of each subband is obtained by multiplication of the impulse response of a single prototype lowpass filter, by a modulating function which shifts the lowpass response to the appropriate frequency range.

In the quantisation process, blocks of decimated samples are formed and divided by a scale factor so that the sample of largest magnitude is unity. In Layer 1, blocks of 12 samples are formed in each subband and each block is assigned one bit allocation. There are 32 blocks, each with12 samples, representing 32 × 12 audio samples. In Layer 2 , in each subband a 36 sample superblock is formed of three consecutive blocks of 12 samples. There is one bit allocation for each 36-sample superblock. All the 32 blocks, each with 36 samples represent a total of 32 × 36 audio samples. As in Layer 1 a scale factor is calculated for each 12 sample block.

Layer 2 provides additional coding of the scale factor. Depending on the importance of the changes between the three scale factors, one, two or all three scale factors are transmitted along with a 2-bit scale factor select information.

For each subband, there are three main types of information to be transmitted.

- Bit allocation : it tells the decoder the number of bits used to code each subband sample. In Layer 1 there are four bits used to transmit the bit allocation for each subband whereas in Layer 2 the number of bits used vary depending on the total bit rate and sampling rate.

- Scale factor: it is a multiplier that sizes the samples to make full use of the range of the quantizer. The computation of scale factor is performed every 12 subband samples. Six bits are allocated for each scale factor. To recover the quantised subband value, a decoder multiplies the decoded quantiser output with the scale factor.

- Subband samples: The subband samples are transmitted using the word - length defined by the bit allocation for each subband.

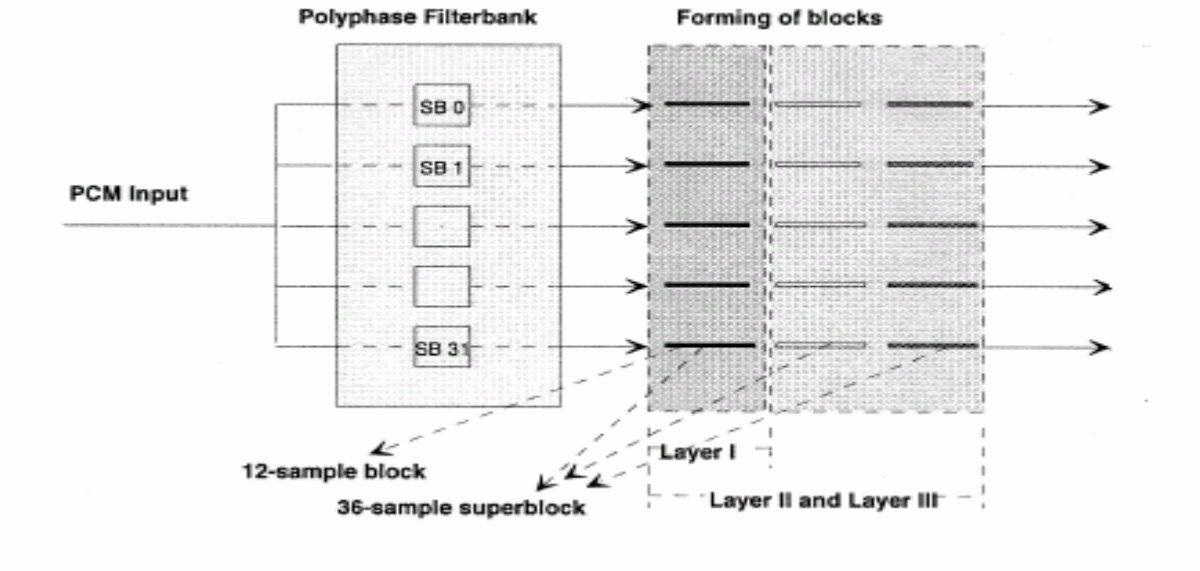

Figure 5. 3 Subband blocks in MPEG encoding

Layer 3 combines some of the features of Layer 1 and Layer 2 with additional coding features. In Layer 3 the output of the filter bank in Layer 1 and Layer 2 is further processed with a Modified Discrete Cosine Transform. This results in subdivision of each polypahse filter output into eighteen finer subbands. In contrast to the two other layers the subband values are encoded in groups of 18 subband samples. A block can be regarded as either consisting of 18 values in each of 32 subbands or of one value in each of 576 subbands depending on whether one accesses the filterbank outputs or the MDCT outputs.

5.3 MPEG audio processing

In audio segmentation and classification tasks, dealing with compressed audios has a number of advantages. The following advantages can be mentioned.

- Has smaller computational and storage requirements than the uncompressed audio processing.
- Long audio streams can be dealt with.
- Some of the audio signal analysis carried out during encoding can be utilized.

Because of these advantages, it is highly desirable to use audio data that is directly obtained from compressed audios.

Features that can be used in many audio processing algorithms can be directly extracted from the MPEG audio streams. In MPEG encoded audio there are two types of information that can be used as a basis for further audio content analysis: the information embedded in the header-like fields (fields such as bit allocation, scale factors) and the encoded subband values.

As we have already seen, the scale factors in the header-like fields carry information about the maximum level of the signal in each subband. This information could be, for instance used in silence detection tasks. The bit allocation field stores the dynamic range of subband values. Whereas the scale factor selection field stores how the loudness changes on three subsequent groups.

Almost all compressed domain audio analysis techniques use subband values as starting point for feature calculations. In MPEG-1 audio, the subband values are not directly accessible and hence some degree of decoding is required. However, the reconstruction of PCM samples, which is the most time consuming step in decoding, is avoided since the subband values will still be in compressed domain.

The subband values in Layer 1 and Layer 2 can be approximated directly using the quantised values in an encoded frame. However, since this values are normalised by the scale factor in each of the 32 subbands, to arrive at the subband values encoded in the file, denormalizing the quantised values is needed.

As already stated, in Layer 3 there are 576 subband values. To extract the magnitudes of these bands from a given file, it is necessary to decode the quantised samples. Furthermore, the scalefactors need to be readjusted and quantisation has to be reversed. This results in the 576 MDCT coefficients. It is also possible to further decode the MDCT coefficients and thereby obtain the 32 Polyphase Filterbank coefficients.

In the following some of the features that can be extracted from an MPEG-1audio are presented. In figure 5.4 below, the structure of MPEG-1 audio is shown. In this figure the subband values are denoted by $S_j(i)$ where j is the subband number and lies in the interval [0 I-1]. The value of I varies depending on the layer type.

Features can be computed either on the subband resolution level, on the block resolution level or on the frame resolution level. In the following the method for feature extraction is based on the work of Silvia Pfeiffer and Thomas Vincent [8] .

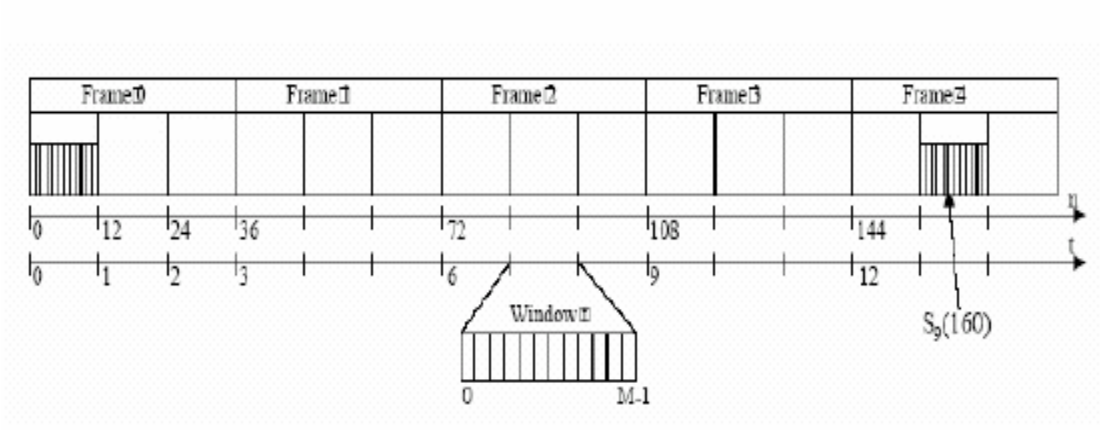

Figure 5. 4 Structure of MPEG-1(Layer 2) audio

In figure 5.4 each frame is made up of three blocks. The window size is denoted by M and the time position with in a window by m, where $0 \leq m \leq M$-1. The window number while going over a file is denoted by t and is related to the time position within a file. Depending on the choice of resolution for analysis, a subband value at window position m can be accessed. If for example, a block is chosen as a window size (non-overlapping) and a subband value resolution level is chosen for feature calculation, the subband value (in a Layer 2 block), $S_9(160)$ will be in window number t=13 at position m=4.

Different methods for feature extraction based on the subband values are proposed by different researchers. In [9], for an MPEG audio frame, a root mean squared subband vector is calculated for the frame as:

$$G(i) = \sqrt{\frac{\sum_{t=1}^{18}\left(S_t(i)^2\right)}{18}} \quad , \; i = 1, 2, \ldots, 32 \qquad (5.1)$$

The resulting G is a 32-dimensional vector that describes the spectral content of the sound for that frame. Based on the above equation the following features are further calculated:

Centriod: The centriod is defined as the balancing point of the vector and can be calculated as follows

$$C = \frac{\sum_{i=1}^{32} i\, G(i)}{\sum_{i=1}^{32} G(i)} \tag{5.2}$$

Rolloff : This is defined as the value of R below which 85% of the magnitude distribution is concentrated.

$$\sum_{i=1}^{R} G(i) = 0.85 \sum_{i=1}^{32} G(i) \tag{5.3}$$

RMS : The root mean square is a measure of the loudness .

$$RMS = \sqrt{\frac{\sum_{i=1}^{32} \left(G(i)^2\right)}{32}} \tag{5.4}$$

Energy Features: When calculating energy features in the compressed domain, the results are closer approximations of perceptual loudness. This can be attributed to the fact that the subband values have been filtered by the psychoacoustic model and thus the influence of inaudible frequencies is reduced. A generalized formula for signal energy is given by.

$$E(t) = \frac{1}{I\ M} \sum_{i=0}^{I-1} \sum_{m=0}^{M-1} S_i^{\ 2}\ (Mt+m)\ h(M-1-m) \tag{5.6}$$

where $h(m)$ is the window function

In [8], the scalefactors in Layer-1 and Layer-2 are used for a block resolution subband energy measure. The scale factors are the maximum value of the sequence of the subband values within a block.

$$scf_i(t) = \max\left(\left|S_i(Mt+m)\right| : 0 \le m \le M-1\right), \quad 0 \le i \le I-1 \tag{5.7}$$

signal magnitude : In[8], sum of the scalefactors are used for a fast approximation of the signal magnitude.

5.4 Summary

This chapter has focused upon two main areas: perceptual audio coding and feature extraction from perceptually encoded audio files. In order to understand the MPEG audio coding algorithms a brief introduction on human perception of audio signals has been given. The kinds of information accessible in an MPEG-1 compressed audio recordings and how to determine features from these information is examined. The advantages of using features extracted from MPEG-1 audio for classification purposes have been also highlighted.

Chapter 6

Experimental results and Conclusions

In this chapter, the methods used to implement the system for discriminating audio signals will be described in details. Moreover the experimental results obtained together with some comments will be presented. The chapter is split into the following sub sections: data description, feature extraction, segmentation and classification.

6.1 Description of the audio data

The audio files used in the experiment were randomly collected from the internet and from the audio data base at IMM. The speech audio files were selected from both Danish and English language audios, and included both male and female speakers. The music audio samples were selected from various categories and consist of almost all musical genres. These files were in different formats (MP3, aif, wav, etc) and in order to have a common format for all the audio files and to be able to use them in matlab programs, it was necessary to convert these files to a wav format with a common sampling frequency. For this purpose the windows audio recorder was used and the recorded audio files were finally stored as 22050 Hz, 8 bit, mono audio files.

The recorded audio files were further partitioned into two parts: the training set and the test set. This was important since each audio file was intended to be used only once,

either for training or for testing a classifier. The training vectors correspond to 52566 frames for speech and 73831frames for music files.

Audio type	Number of files	Average length	Total length
Speech	45	15 sec.	675 sec.
Music	55	15 sec.	825 sec.

Table 6.1 Training data

Audio type	Number of files	Average length	Total length
Speech	30	15 sec.	450 sec.
Music	30	15 sec.	450 sec.
silence	30	15 sec.	450 sec.
Music + speech	10	120 sec.	1200 sec.

Table 6.2 Test data

6.2 Feature extraction

Feature extraction has already been mentioned in the previous chapters. Here, it is focused on how features are extracted from the row audio data and how they are used in classification and segmentation modules. MFCC, Zero-crossing rate and Short time energy features are used in the classification part whereas RMS is the only feature used in the segmentation part.

6.2.1 MFCC features

In order to extract MFCC features from the row audio signal, the signal was first partitioned into short overlapping frames each consisting of 512 samples. The overlap size was set to half the size of the frame. A Hamming window was then used to window each frame to avoid signal discontinuities at the beginning and end of each frame. A time series of MFCC vectors are then computed by iterating over the audio file resulting in thirteen coefficients per frame. The actual features used for classification task were the means of the MFCCs taken over a window containing 15 frames. Furthermore only six out of the thirteen coefficients were used. In this way a very compact data set was created. The following figures show plots of the speech and music signals as a function of time together with

their respective MFCCs . Note that there are significant changes in the upper part of the MFCC plots, whereas the lower parts seem to remain relatively unchanged. Therefore, for speech and music signals one can neglect the lower part of the MFCCs without losing any important information.

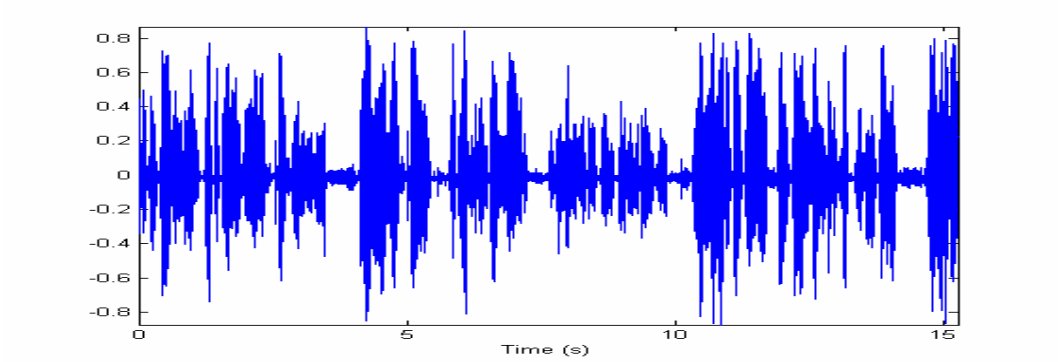

Figure 6.1 Plot of a speech signal as function of time

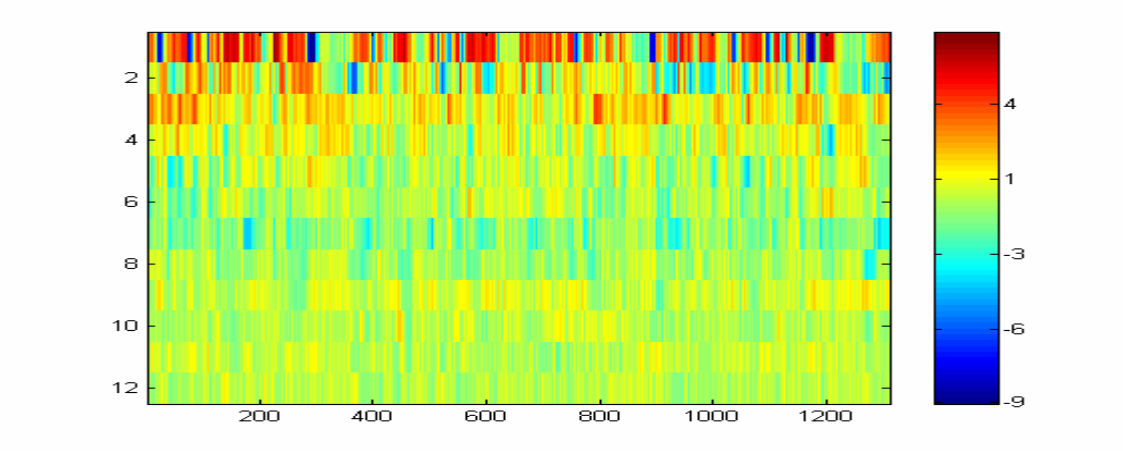

Figure 6. 2 Plot of the MFCCs for the speech signal

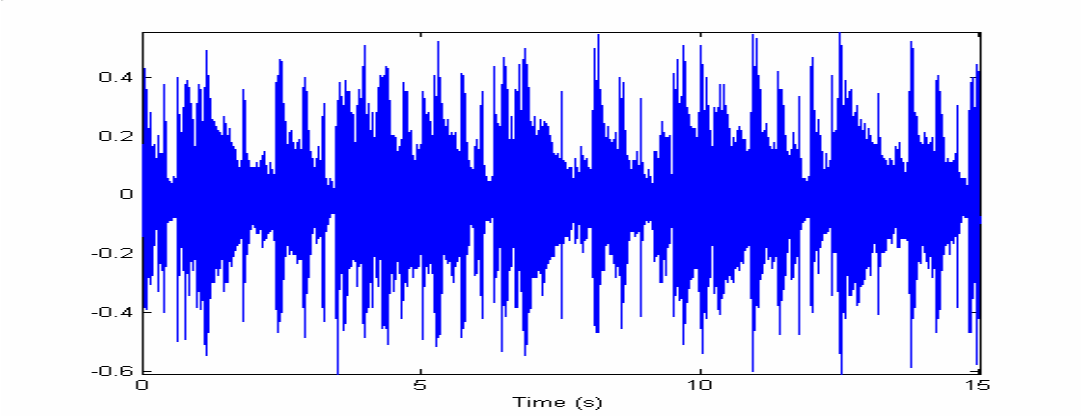

Figure 6. 3 Plot of a music signal as function of time

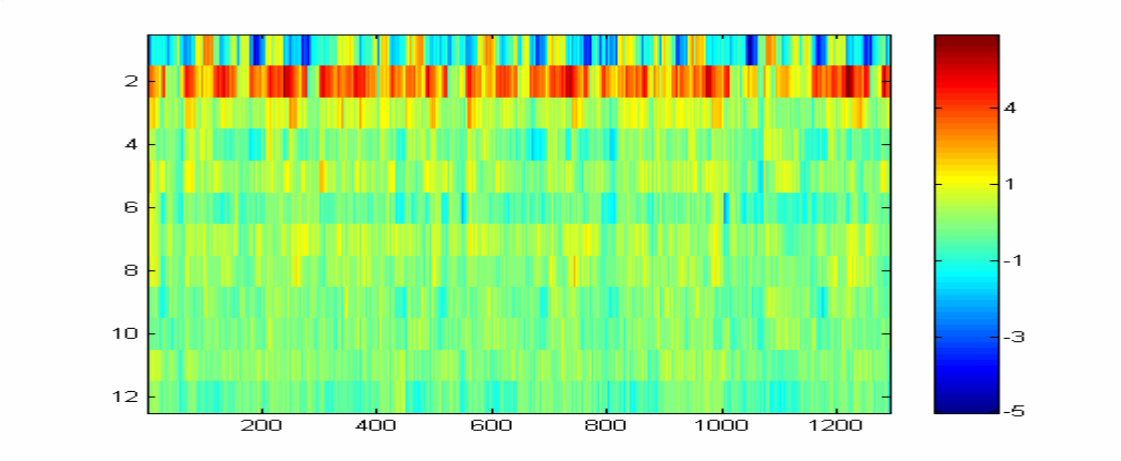

Figure 6. 4 Plot of the MFCCs for the music signal

6.2.1 The STE and ZCR features

Since these features were intended to be used either in conjunction with the MFCCs or independently, it was necessary to split the audio signal so that the length of these features were the same as the length of the MFCCs. Hence, the partition of the audio signal into overlapping windows was exactly the same as in the case of the MFCC features. The Short Time Energies and the Zero-crossing rates were extracted from such windows, one from each window. The actual features used for the classification task were the means taken over a window containing 15 frames. The following figures show plots of STE and ZCR for both music and speech signals.

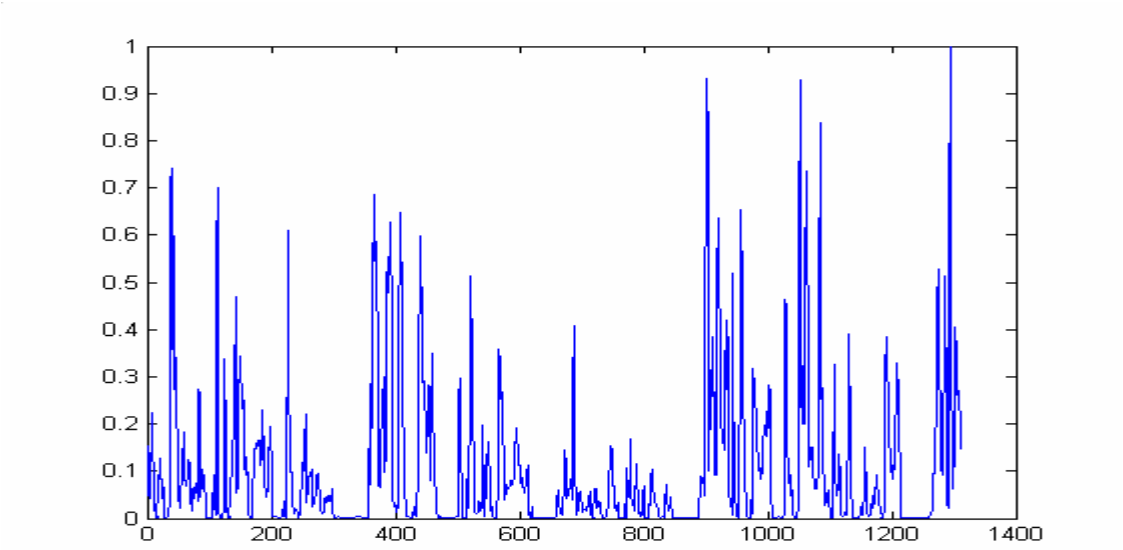

Figure 6. 5 STE for speech signal

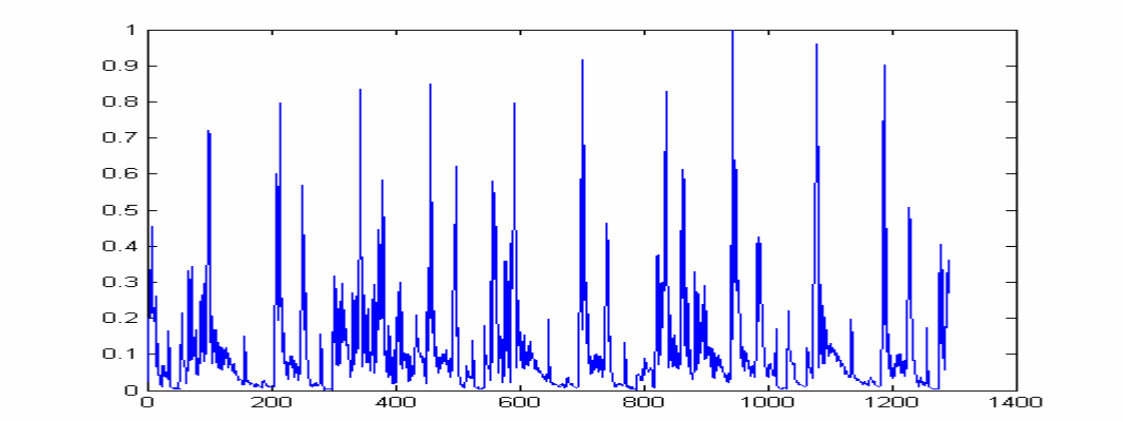

Figure 6. 6 STE for music signal

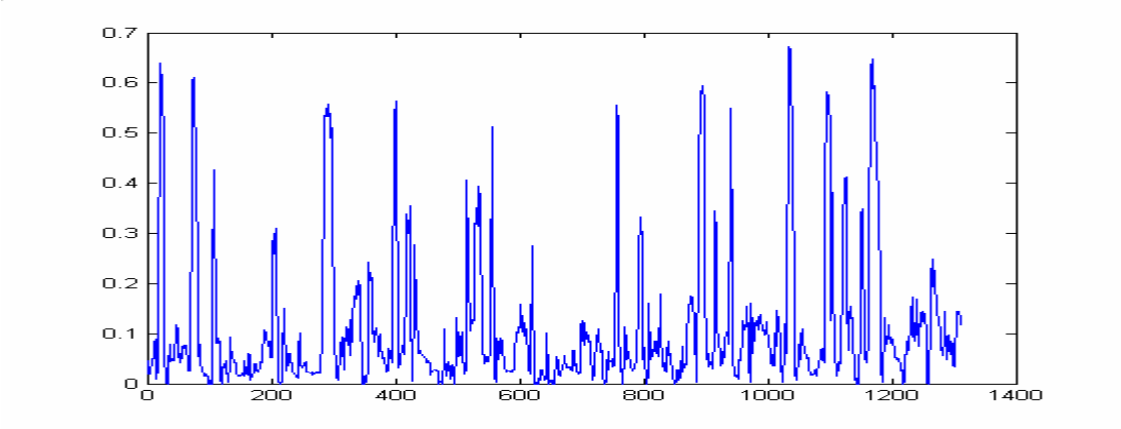

Figure 6. 7 ZCR for speech signal

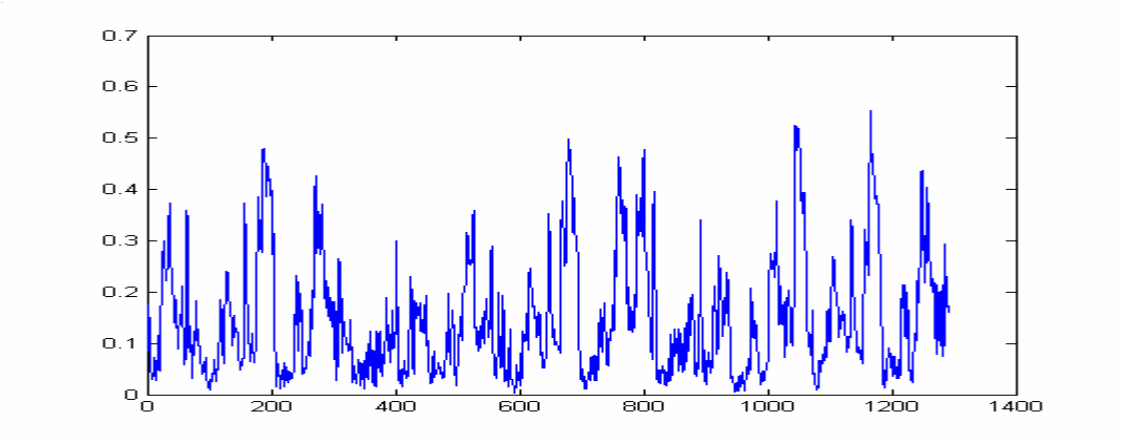

Figure 6. 8 ZCR for music signal

6.2.1 The RMS feature

Although the RMS is somewhat related to the short time energy, it is often used as a measure of the loudness of audio signals and therefore a unique feature to segmentation. Since this feature was used alone for another task, the audio signal was split in a rather different way. The audio signal was first partitioned into short non overlapping frames each consisting of 512 samples. The Root Mean Square was computed by iterating over the audio file based on the amplitude equation shown on page 25 and a single RMS value is obtained for each frame. The following figures show plots of RMS for both music and speech signals.

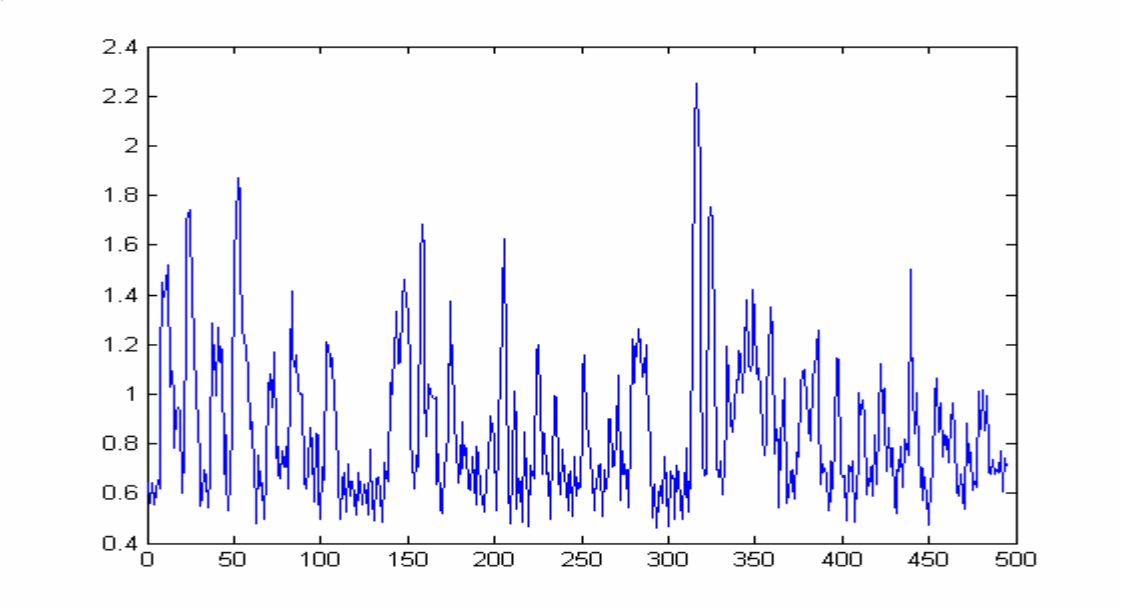

Figure 6. 9 RMS of a speech signal

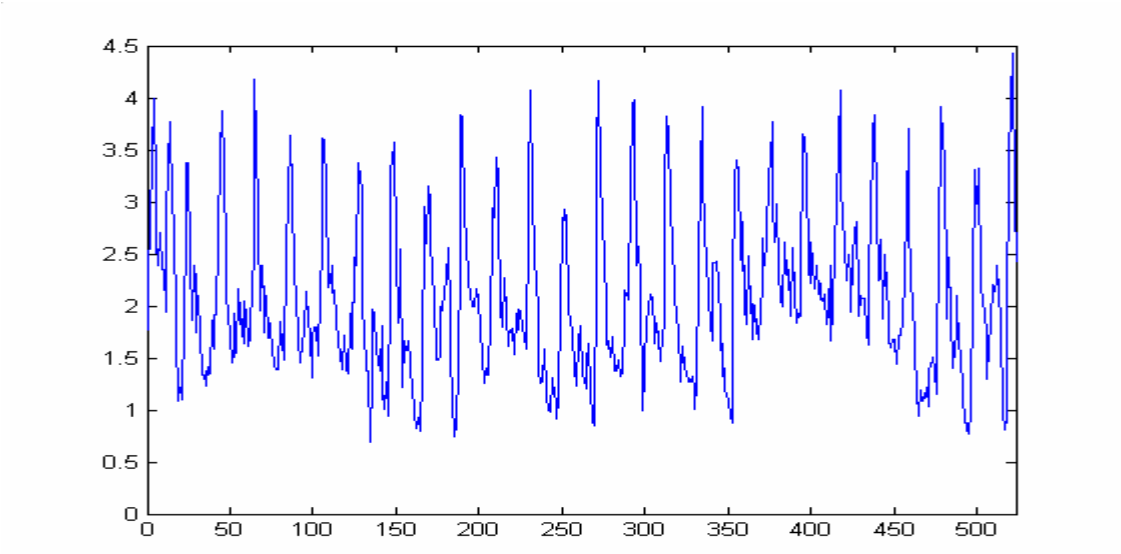

Figure 6. 10 RMS of a music signal

6.3 Classification

The classification task was done with each of the classifiers: the Generalised Mixture Model and the k-Nearest Neighbour classifier discussed in Chapter 3. Each classifier was trained on a set of labelled examples and then tested on other cases whose true

classification was known in advance but not given to the classifier. Experiments were done for two classification tasks: 2-class task, where the classifier classifies the audio into music or speech and 3-class task, where the audio signal is classified either into music, speech or silence. For experimentation, the features were either used alone or in conjunction with the others. In this way the effect of each feature set on the classification task can be observed.

The classifiers were used to classify each frame into music, speech or others. The features were then blocked into a one second long segments. A global decision was made for the entire block by choosing the class that appeared most frequently. The experiment was done in two different ways: to begin with discrete homogeneous audio signals were used as input to the classifier and then audio recordings containing different audio types were presented to the classifier.

It is always a good idea to use each data example only once in a single analysis. If the same data was used to train and test a classifier, the test would not give a good indication of how we might expect the classifier to perform on new data. Considering the amount of time and effort invested in acquiring, labelling and processing data, it can be tempting to reuse data. But any such reuse is likely to cause overestimation of the accuracy of the classifier and spoil the usefulness of the experiment. In order to compare the two classifiers fairly, the same training set and the same test set were used for both the GMM and the k-NN classifier.

Obtaining data in a form that is suitable for learning is often costly and learning from such data may also be costly. The costs associated with creating a useful data include the cost of transforming the raw data into a suitable form , labelling the data and storing it. The costs associated with learning from the data involve the time it takes to learn from the data. Given this costs, it is always a good idea to limit the size of the training set. Hence, a common question asked at the beginning of many audio classification tasks is: what length of the audio data should I use for training ? In machine learning there are two basic observations:

- The computational cost of learning a model increases as a function of the size of the training data and

- The performance of a model has diminishing improvements as a function of the size of the training data.

The curve describing the performance as a function of the size of the training data is called the learning curve. In order to find an optimal size of a training set, the nearest neighbour classifier was chosen. By using different sizes of the total training data the error rate was calculated. For each size of the training set chosen, the experiment was repeated five times (randomly chosen sets) and the average results were taken. The following figure shows the learning curve obtained when the nearest neighbour classifier was used.

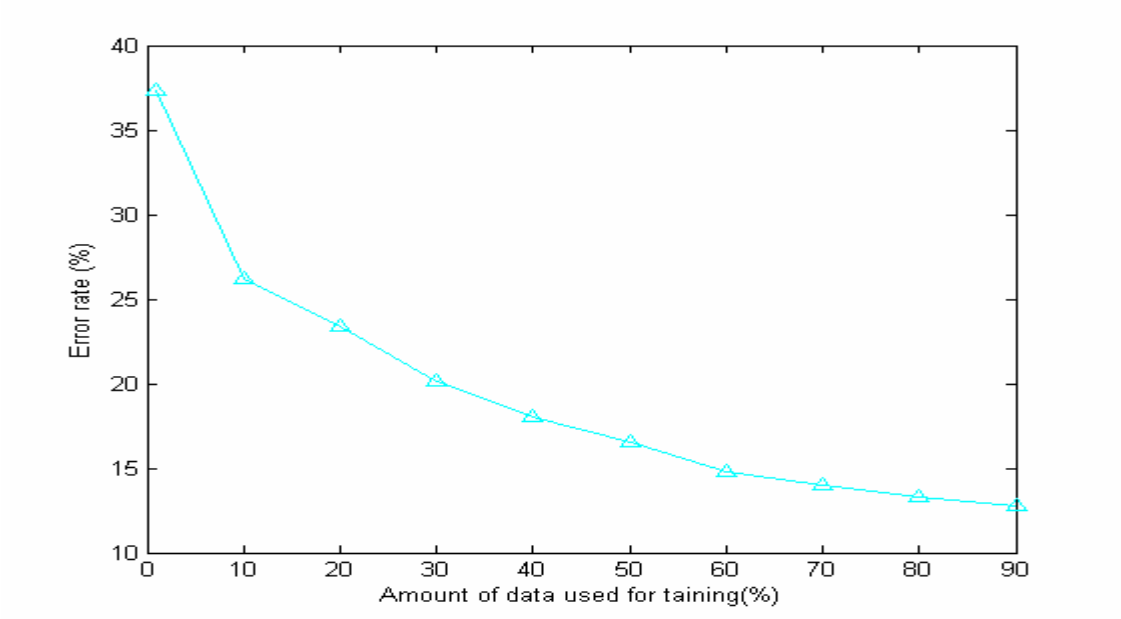

Figure 6. 11 The learning curve

6.3.1 k-NN

The *k*-Nearest neighbour classification method is a lazy learning, local classification algorithm. As mentioned earlier the basic algorithm is simple. For each new data point to be classified, the k nearest training samples are located. The class label which has the most members in the set of k nearest points is assigned to the new data point. In this project the Euclidean distance measure is used as the similarity measure. Several simulations were done using different *k* (k=1,3,5 and 10) values and the smallest *k* value (k=5) that worked well was chosen for testing the results. In order to see the effect of each feature on the performance, different combinations of the features were used as an input to the classifier.

The tables below show the confusion matrices for the k-NN classifier. Each row in the matrices correspond to the true class of the data and each column correspond to the class predicted by the classifier. The value of K for all the simulations was set to five.

	Music	Speech
Music	**89.1**	10.9
Speech	8.4	**91.6**

Table 6.3 Confusion matrix when MFCC features were the only inputs.

	Music	Speech
Music	**93.01**	6.99
Speech	5.49	**94.51**

Table 6.4 Confusion matrix when the inputs were MFCC and STE features.

	Music	Speech
Music	**90.61**	9.39
Speech	7.47	**92.53**

Table 6.5 Confusion matrix when the inputs were the MFCC and ZCR.

	Music	Speech
Music	**93.67**	6.33
Speech	5.71	**94.29**

Table 6.6 Confusion matrix when the inputs were MFCC, ZCR and STE features.

From the results obtained the following observations can be made. The MFCC features used as an input, alone, result in an overall correct classification rate of 90.3%. When the MFCC features were used in conjunction with the short time energy and the Zero Crossing rate the overall classification rate gets better and is around 93.98%. The same is true when MFCC feature are used together with short time energy features. However, when the input to the classifier was a combination of MFCC features and zero crossing

rate only little improvement in the overall correct classification rate was seen. We conclude therefore that the MFCC features in conjunction with the short time energy alone can with a good classification rate be used for a speech/music discrimination.

It is worth to mention that the features used in the simulations were pre-processed in order to avoid classifier bias. As mentioned in chapter 3, the use of Euclidean distance measure can affect the performance of the classifier when two or more feature sets were used at one time. In many cases each feature set can be normalised as follows.

$\mu_i = \frac{1}{N}\sum_{i=1}^{N} x_i$, where μ is the mean

$\delta^2{}_i = \frac{1}{N}\sum_{i=1}^{N}\left(x_i - \mu_i\right)^2$, where δ^2 is the variance. The normalised feature vector is then given as:

$\hat{X} = \frac{X - \mu}{\delta}$, where δ is the standard deviation

Generally most of the (frame) misclassifications that occur happen to be in the case of music samples, and particularly music samples that contain classical music. This could be attributed to the fact that in many classical music samples there are a number of silence frames and that this frames might be classified as speech features. Also, dividing the test features into segments and classifying each segment yields much better classification results than when frame classification was done.

6.3.2 GMM

Unlike the k-NN classification method the GMM method requires determining the parameters of the model based on the training set. The GMM classifier was implemented by first estimating the probability density functions of the features under the two possible conditions, music or speech, based on the training set. A new test set is then classified according to the likelihood ratio, that is the ratio of the values of the pdfs of the two classes at that point. The pdfs of the two data sets were estimated by fitting a General Mixture Model. The Gaussian means were first initialised by using the k-means clustering and then the model is refined using the Expectation Maximisation algorithm. Equal prior likelihoods were assumed for each class and the decision rule was that points in the feature space for which one pdf was larger, were classified as belonging to that class.

The following tables show the simulation results obtained using the GMM classifier. The number of clusters was fixed to 20 and the number of iterations was set to 5. The simulation was made five times and the average was taken.

	Music	**Speech**
Music	**85.22**	**14.78**
Speech	**0.44**	**99.56**

Table 6.7 The features used were the MFCCs.

	Music	**Speech**
Music	**89.78**	**10.22**
Speech	**0.22**	**99.78**

Table 6.8 The features used were the MFCC and STE features.

	Music	Speech
Music	**85.65**	**14.35**
Speech	**0.00**	**100.00**

Table 6.8 The features used were the MFCC and ZCR features.

	Music	Speech
Music	**91.30**	**8.70**
Speech	**0.00**	**100.00**

Table 6.9 The features used were the MFCC, STE and ZCR features.

Although the results obtained in this case showed similar tendencies as in the case of the K-nearest neighbour classifier, the correct classification rate was even better. When the MFCC features were used in conjunction with the short time energy and zero crossing rate, a correct classification rate of around 95.65% was obtained. This result was the best result among the classification results obtained from both the GMM classifier and the KNN classifiers. A correct classification rate of about 94.78% was obtained for the case when MFCC in conjunction with the Short Time Energy features were used. However, for the case where the input was a combination of MFCC and ZCR features, the classification rate was 92.83% , which is almost the same as when pure MFCC features were used.

6.3.3 Comparison of the classification results

Now that we have used both the k-nearest neighbour classifier and the general mixture model classifier, it would be interesting to make some kind of comparison between these classifiers and also see the effect the features have on the outcome. The Table below shows the classification results obtained for the two classifiers with different feature combinations.

Features	**k-NN (k=5) Accuracy (%)**	**GMM Accuracy (%)**
MFCC	90.35	92.39
MFCC + ZCR	91.57	92.83
MFCC + STE	93.76	94.78
MFCC + ZCR+ STE	93.98	95.65

Table 6.9 Accuracy testing results for the speech/music classifier

The classification results obtained from using a general mixture model classifier and a k-nearest neighbour classifier demonstrate the effect of the classification algorithms. The general mixture model classifier seemed to have a far better correct classification rate than k-nearest neighbour classifier (around 2%). In both cases, adding more features to the MFFC features showed a positive effect on the outcome, although using the MFCC in conjunction with STE resulted in a relatively higher classification rate than when MFCC features were used in conjunction with the zero crossing rates.

6.3.4 Classification into three classes

In the following experiment, the GMM classifier was used to classify audio signals into music, speech and silence. The same training set (as in the case of the two class task) was used to train the GMM, and a new set of 30 silence audio files was added to the existing 60 audio files used in the speech/music classification task. In the training stage, only two classes were used to "train" the classifier. In the classification stage, any data that did not belong to any of the two classes was to be classified as silence. The test was repeated 5 times and the average is shown in the table below. Note that the results obtained in this case cannot be compared with the results obtained in music/speech classification task, since in this classification task, audio signals other than music and speech were taken into consideration.

	Music	**Speech**	**Silence**
Music	**80.0**	11.74	8.26
Speech	0.66	**99.34**	0.00
Silence	1.31	0.00	**98.69**

Table 6.10 3-class task, the features used were the MFCCs only.

Correct classification rate of around 92.65% was obtained for this task. The number of music frames that had been classified as silence seemed to be a bit higher relative to the number of speech frames that had been classified as silence. A reasonable explanation to this observation could be that in speech, silences between two spoken sentences are less than 0.5 seconds, whereas silences in music are usually greater than 0.5 seconds.

6.3.5 Classification of audios that contain different audio types

Lets look at the case where the input to the classifiers was an audio stream that contained speech and music. The same procedure as that of a discrete input case was followed here. To start with, each audio frame (~23ms) was classified into one of the two/three audio classes. Decisions over a one second (~43 frames) window was then made, by choosing the class that appeared most frequently. In applications where the interest lies in extracting one audio type from an audio stream, this would not be a good solution for many reasons. One good reason could be that the beginning and end of one audio type cannot be determined with some accuracy.

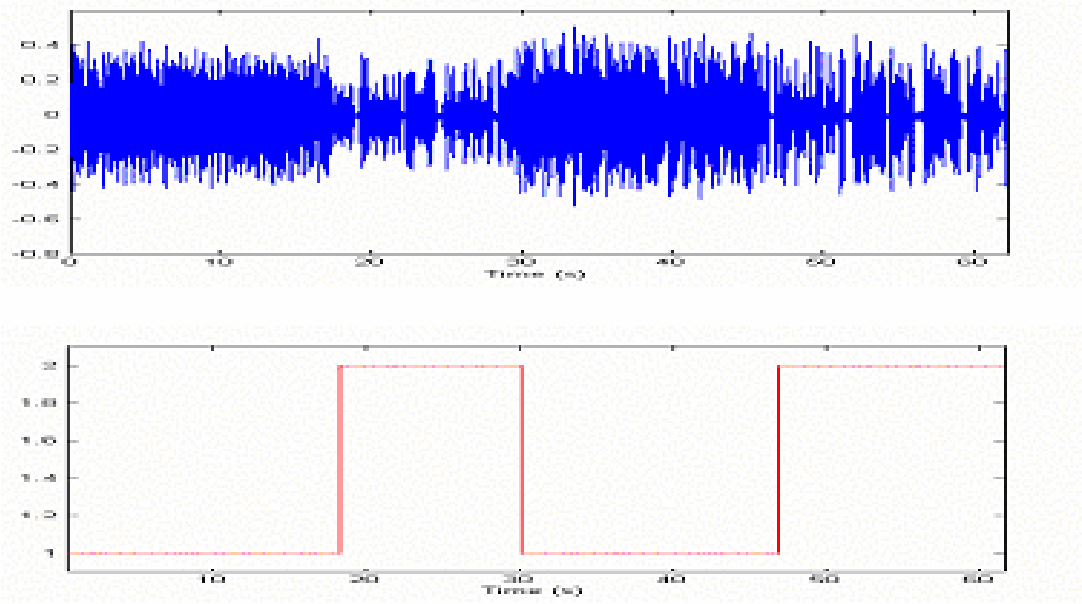

Figure 6. 12 Classification of an audio stream into two classes, 2 stands for speech while one is for music

In the above example, no frame misclassification was observed, hence the smooth regions on the figure. But in most cases (as in figure 6.13) there will be some frames that are misclassified and this could be another reason to why this method is not optimal. The following figure shows classification of the same audio signal using the k-nearest neighbour classifier. As it can be seen in the figure one or two music windows had been classified as speech.

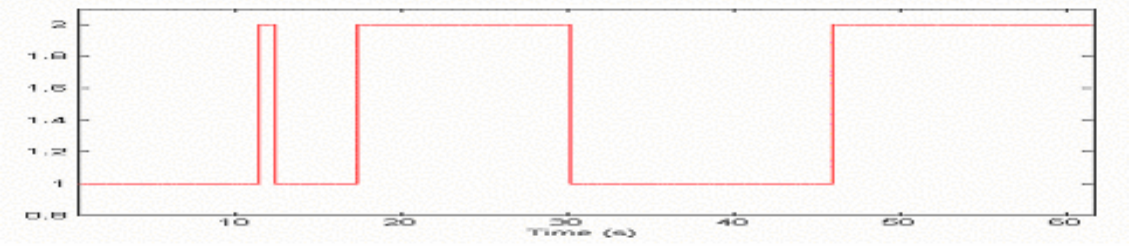

Figure 6. 13 Classification of an audio stream into two classes using a k-NN Classifier

6.4 Audio Segmentation

As mentioned in chapter 2 the main aim for segmentation is to partition the input audio signal into acoustically similar audio segments. In order to divide an input audio signal into similar regions, the method explained in chapter 4 was followed. The expression for the Root Mean Square was used for computing the RMS feature for non-overlapping frames each containing 512 samples. About 43 consecutive frames were then joined to form windows that have an approximate length of one second. For each window the probability density function was then found from the computed feature vectors. These pdfs were then used as a basis for similarity measure, i.e. how similar two consecutive windows were. If there was a strong similarity between these two windows, then no boundary was assumed to exist and on the other hand if the two windows were more dissimilar then a transition was considered. In the following, a summary of the steps followed is shown.

- Split the audio stream into frames each containing 512 samples.
- For each frame compute the root mean square value.
- Group consecutive frames to form windows each of length one second.
- Calculate the mean and variance on each window.
- Find the similarity measure.

- For each window *j* calculate a value *D(j),* that gives a possibility of a transition within that window.

- Calculate locally normalised distances, $D_{norm}(j)$.

Some experimental results for finding transition windows for audio recordings that had been obtained from a local radio station are shown below. As it can be seen in the figure the value of *D(j)* will be relatively small if the windows *(j-1)* and *(j+1)* are similar and on the other hand the value *D(j)* will approach one, if the two windows are not similar. Since the method implemented is not only sensitive to any changes from music to speech and vice versa but also large changes in volume, there can be some false transitions in cases such as, changes from silence to audible sound. These changes can be filtered out with a suitable threshold. Large values of *D* can also be expected around a change window and hence some normalisation is required.

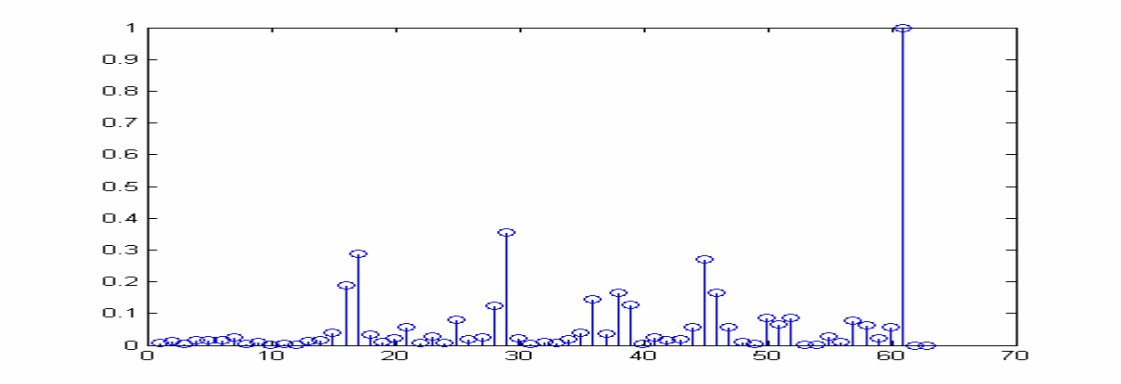

Figure 6. 14 Plot of *D(j)* as a function of time

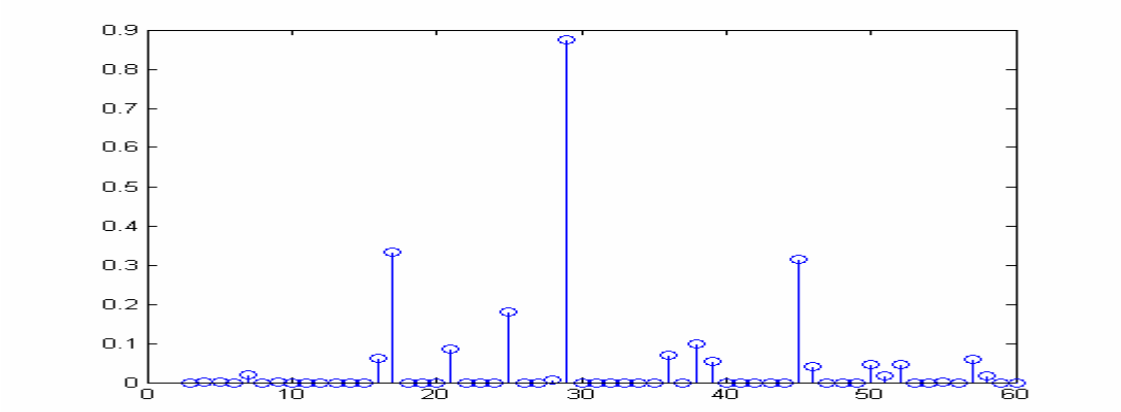

Figure 6. 15 Plot of $D_{norm}(j)$ as a function of time

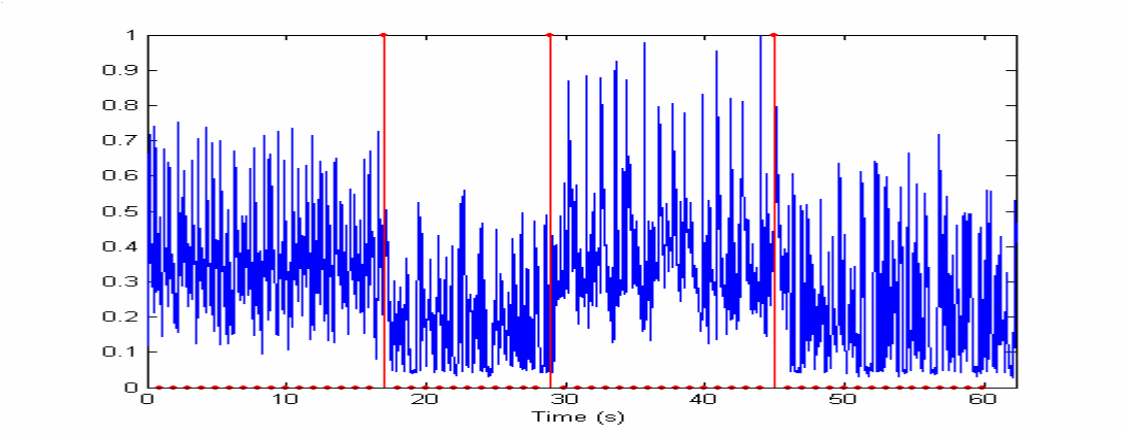

Figure 6. 16 Detected audio transitions together with the RMS

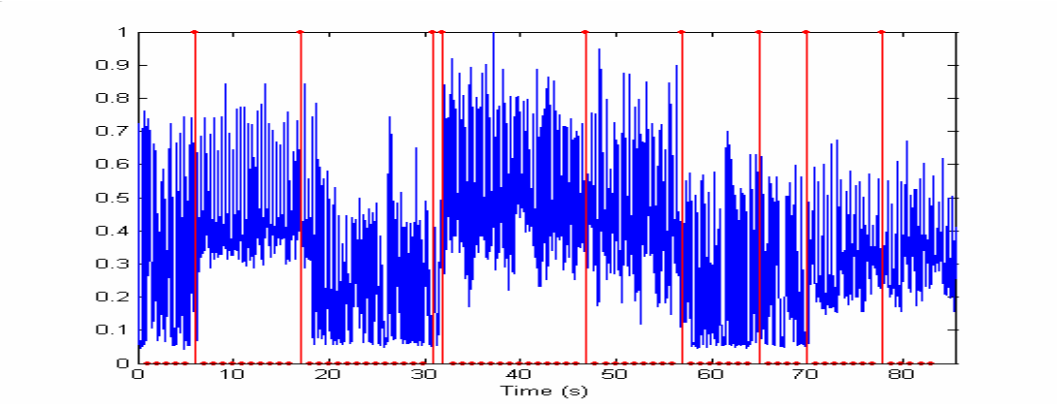

Figure 6. 17 Another segmentation example, where one or more false transition detections are shown

In section (6.3.5), audio signals have been classified either into music or speech using one of the two classifiers. It has been pointed out that the aforementioned method was not that optimal. Furthermore, partition of audio into several homogeneous regions was demonstrated in the previous section. In this section, the aim is to combine the classification method with the segmentation procedure. This would in fact reduce the number of misclassified windows, and hence an improvement in the performance. This method was implemented and tested on several audio signals that contained music and speech. Using this method, it was possible to extract only one type of audio and save it in a file. However, since the time of transition was not precisely detected a small part of a neighbouring window was included. These could be minimised if the window size was decreased by half. The following figures demonstrate the classification of audio signal

into the two classes. Speech audio is marked by the red colour while music audio is marked by the blue colour.

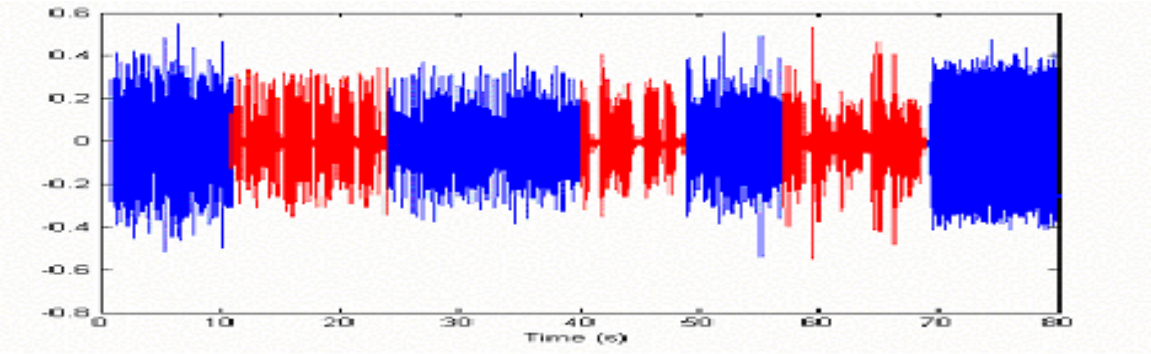

Figure 6. 18 classification into speech and music

6.5 Summary

In this chapter the actual implementation of the system discussed in chapters one to five has been presented. How the different classification algorithms, discussed in chapter 3, were trained and tested using the different features extracted from audio signals, that have been stored in a WAV format, is explained. Comparison of the two classification methods and the effect of each feature set on the classification results is presented. Segmentation of audio recording based on the method explained in chapter four has been also presented. And finally implementation of a system that combines the segmentation algorithm with the classification algorithm is presented.

Chapter 7

Conclusion and future work

The aim of this project was to design a system that could be used to segment an audio signal into similar regions and then classify these regions into music, speech and silence audio classes. The project could be considered as a combination of two tasks; a segmentation task and a classification task. Classification algorithms were used either independently with a given audio segment or in combination with the segmentation algorithm.

Features extracted from music and speech signals (in WAV format) were used in the two tasks. Three feature sets were used to train and test two different classifiers, the General Mixture Model classifier and the k-Nearest Neighbour classifiers, to classify audio signals, and only one feature set was used to partition audio into similar regions. Nearly all the audio files used in this project had been obtained from the internet. The majority of these audio files were in MP3 format and it was necessary to convert them to WAV format. Thus, the process for extracting audio feature showed to be very time consuming. It would have been very advantageous if the system was designed to take in audio in MP3 format. This could have had two effects on the system; the need for converting one audio format to another would have been avoided, and features would have been extracted directly from the encoded data. The two classifiers were trained and tested with the same training and test sets. With each classifier, four experiments were run with different combinations of the

feature sets. The General Mixture Model classifier showed a better classification performance in all cases. The best correct classification result, which was more than 95%, was obtained when all the feature sets were combined and used as an input to a GMM classifier. In addition, the GMM was able to classify a long audio file in relatively shorter time when compared to the k-Nearest Neighbour classifier. However, in GMM classifiers there was a higher degree of variation in the classification results of the same audio segment.

The segmentation algorithm was based on the root mean square features. This feature is usually used in audio segmentation since changes in loudness are important cues for new audio events. The segmentation algorithm was implemented and tested on different audio files and was able to detect the transition frame in most cases. However, the method is incomplete since the segment limits could not be specified within some degree of accuracy. In most of the simulations considered, where a long audio file was segmented and classified, there has been cases where auditory verification of the boundaries indicated that part of the preceding segment was included within boundaries of the current segment.

The system implemented worked well on classifying any type of music and speech segments with a correct classification rate of 95.65% for one second windows. The system also worked reasonably well for segmenting audio signals into similar classes. Some improvement in the segmentation method used is however required.

There are many things that could be done in the future. The segmentation algorithm could be modified to detect the transition point with an accuracy of 30ms, and also to automatically set the threshold for finding the local maxima of the normalised distance measure. More training data could be used in the classification part. The system could be trained to include other classes other than music, speech and silence. Further classifications into different music genre or identifying a speaker are also other possibilities.

A Maaate

Maaate[1] is a C++ toolkit that enables audio content analysis on compressed audio files. It is designed to support MPEG1/2 Layer 1,2 and 3 audio files. It makes the subband samples, and other preprocessed features as well as the file format specific fields accessible. It also allows content analysis functions such as silence detection to work on the extracted features.

Maaate is implemented in C++ using standard template library. In order to separate different functionalities and provide simple Application Program Interfaces (APIs), Maaate is designed in tiers.

Tier 1 deals with parsing of MPEG audio streams and offers access to the encoded fields. The most important class in this tier is the MPEGfile class. Tier 2 offers two generic data containers that can be used by the analysis modules. The SegmentData and the SegmentTable classes provide the data containers. Tier 2 also provides a module interface to plugin analysis routines that are stored in dynamically loaded libraries.

Tier 1 consists of the following classes

MPEGfile : contains The API to open an MPEG audio file and process the audio frames.

Header : contains the code to parse and access MPEG audio frame headers.

AllLayers : contains code that all three layers need for parsing one MPEG audio frame. The AllLayers class is an abstract class and *as such* only instances of its subclasses (Layer 1,2 and 3) can be created.

Layer1-Layer3 : are subclasses of AllLayers and contain layer specific code.

MDecoder : provides a simple API to use for playback applications where decoding to PCM into a buffer is required.

As we have already seen in chapter 3, an MPEG audio file is made up of a sequence of audio frames. Each frame has a header which contains information about the type of data that is encoded in the frame. Based on this information, the length of the data encoded in the frame can be calculated and the data can be parsed. At the API, one frame at a time may be parsed and encoded data requested. The encoded data in Layer 1 and Layer 2 are similar whereas the encoded data in Layer 3 is different from both Layers.

A module is a collection of related functions that provide a broader functionality. Modules that analyze the content of an MPEG audio file collect information from several frames and compute a more abstract information. Some example of modules are described below.

-Feature extraction modules are modules that make use of the tier 1 field access functions and store their results in one of the containers provided by tier 2. Feature extraction modules include modules such as spectral flux , spectral centroid and Energy modules.

-Feature analysis modules are modules that use the extracted features for further analysis. These modules make use of filled (features extracted) containers and store their results in another container.

-Content analysis modules calculate higher level information using feature extraction and analysis modules. Such modules usually call for other modules to manipulate their results, which again may be stored in the relevant containers.

A module is an instance of the Module class, which also provides functions to get information on the instantiated module, handle input and output parameters, check constrains on parameters and call the module functions. The apply function of a module takes a list of parameters as an input and produces, as a result of its processing, a list of parameters. To setup the environment under which the apply function will work, other functions are required. The following is a description of the functions found within a module and callable at the module interface:

- init (required) : sets up the basic information of the module such as its name, description, and the input and output parameter specification.
- default(required) : sets default values for input parameters and returns the input parameter list.
- suggest (optional): takes an input parameter list, suggests parameter values based on information provided by other parameters, and changes constrains of input parameters as required.
- reset (optional) : provides the possibility to reset a module
- apply (required) : takes a list of parameters as an input, performs analysis and returns a list of output parameters.
- destroy (optional) clears memory allocated within the module and deletes parameter specification.

A parameter is an instance of the ModuleParam class. In an application, the parameters are handled in the following way: the list of parameter specifications for input and output is set up by the init function. Thereafter, the application sets up an input parameter list by calling the default function. The application may then change the input parameter values as it requires. It is then possible to call the suggest function which will fill in necessary parameter values and constraints and perform sanity checks. And finally the application may call the apply-function to check whether the parameter are within a specific range of allowed values.

The allowed data types for parameters are either basic types or complex types and are all listed in the type MaaateType. The basic type for parameters include boolean, integer, real and string types. The Complex types for parameters are: a pointer to an opened audio file, a pointer to a segment data structure and a pointer to a segment table.

The following is a list of audio features that can be extracted using the Maaate audio toolkit. Plots of some features extracted from music and speech files (mp2) are also shown in the figures below.

Pre-processed features:

- Normalise Subband Energies
- NormalisedSubband Value Energies
- Subband Scalefactors(Channel 0)
- Subband Mean
- Subband RMS
- Subband Values (Channel 0)
- Subband Values (Mean over channels)
- Subband Values (RMS over channels)

Energy features:

- Band Energy
- Signal Energy
- Signal Magnitude
- Sum of Scalefactors

Bandwidth features:

- Bandwidth
- Signal Bandwidth
- Significant Subbands

Spectral energy statistics:

- Band Energy Ratio
- Central Moment
- Spectral Centroid
- Spectral Flux
- Spectral Rolloff

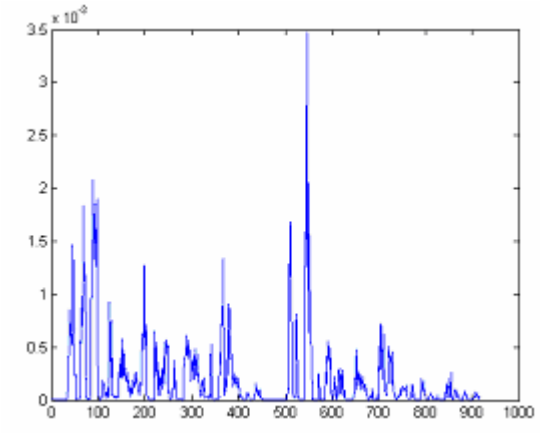

Figure 1a Signal Energy for speech

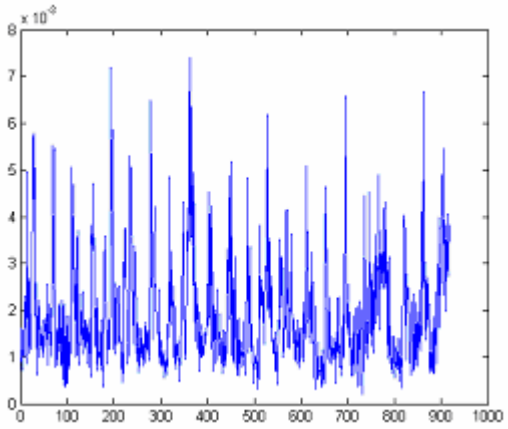

Figure 1b Signal Energy for music

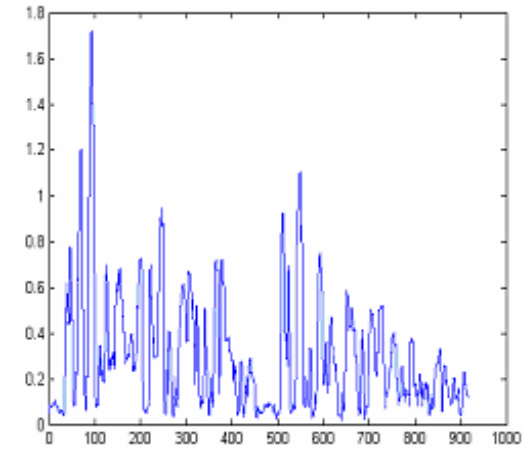

Figure 2 Sum scalefactors for speech

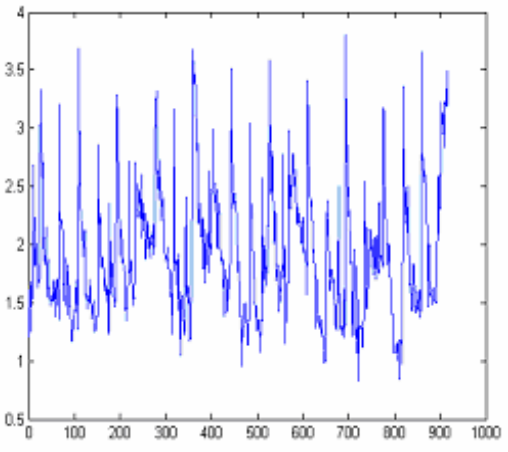

Figure 2b Sum scalefactors for music

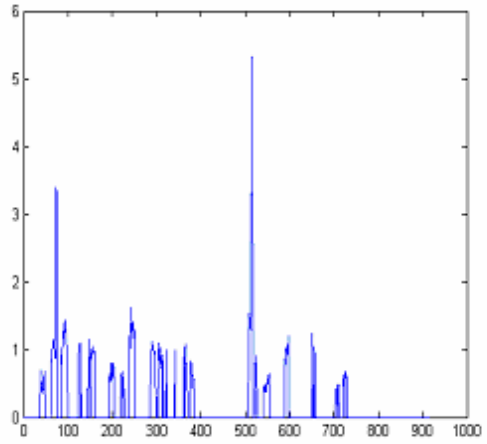

Figure 3a Spectral centroid for speech

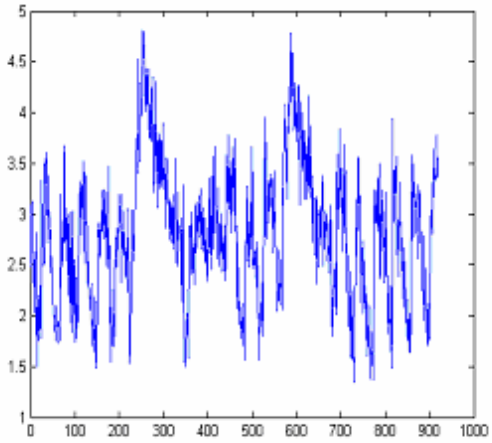

Figure 3b Spectral centroid for Music

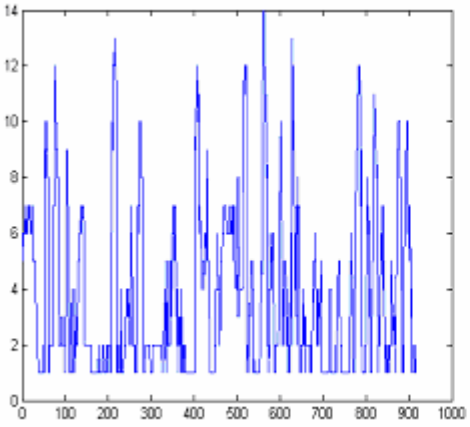

Figure 4a Roll off for speech

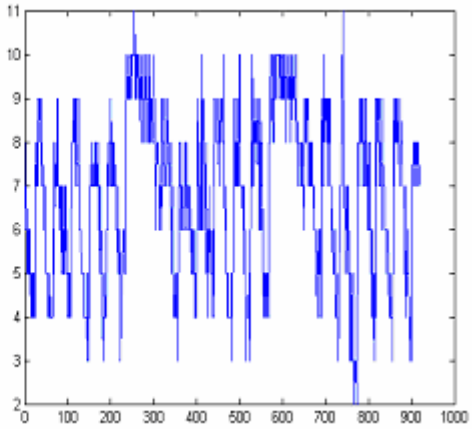

Figure 4b Roll off for music

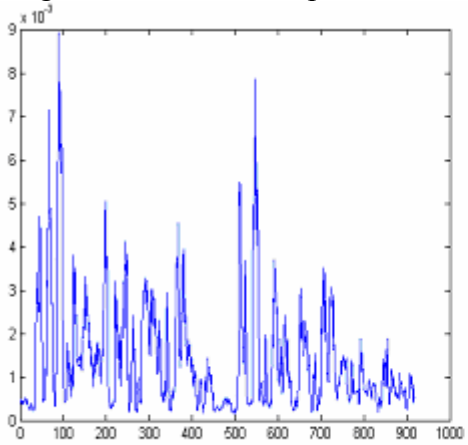

Figure5a Signal magnitude for speech

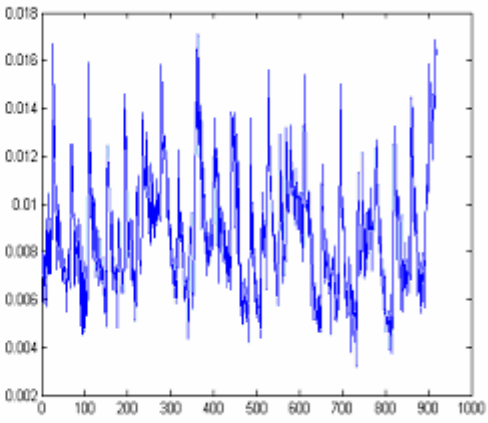

Figure5b Signal magnitude for audio

7. References

[1] Lie Lu, Hong-Jiang Zhang and Hao Jiang. "Content analysis for audio classification and segmentation". *IEEE Transactions on speech and audio processing,* vol.10, no.7, October 2002

[2] K. El-Maleh, M. Klein, G. Petrucci and P. Kabal , " Speech/Music discrimination for multimedia applications," *Proc. IEEE Int. Conf. on acoustics, Speech, Signal Processing* (Istanbul), pp. 2445-2448, June 2000

[3] H. Meindo and J.Neto, " Audio Segmentaion, Classification and Clustering in a Broadcast News Task" , *in Proceedings ICASSP 2003*, Hong Kong, China, 2003.

[4] G. Tzanetakis and P. Cook, " Multifeature audio segmentation for browsing and annotation," *Proc.1999 IEEE workshop on applications of signal processing to Audio and Acoustics*, New Paltz, New York, Oct17-20, 1999.

[5] C. Panagiotakis and G.Tziritas " A Speech/Music Discriminator Based on RMS and Zero-Crossings". *IEEE Transactions on multimedia,* 2004.

[6] E. Scheirer and M. Slaney, " Construction and evaluation of a robust multifeature speech/music discriminator, " in *Proc. ICASSP '97*, Munich, Germany, 1997, , pp. 1331-1334.

[7] Davis Pan, "A Tutorial on MPEG/Audio Compression,". IEEE Multimedia Vol. 2, No. 7, 1995, pp. 60-74.

[8] Silvia Pfeiffer and Thomas Vincent "Formalisation of MPEG-1 compressed domain audio features", Technical Report No.01/196, CSIRO Mathematical and Information Sciences, Dec. 2001.

[9] G. Tzanetakis and P. Cook, " Sound analysis using MPEG compressed audio", *Proc. IEEE Intl. Conf. on acoustics, Speech, Signal Processing,* ICASSP, 2000

[10] D. Pan, " A tutorial on MPEG/audio compression," *IEEE Multimedia*, vol. 2, No.2, 1995, pp.60-74.

[11] Christopher M. Bishop, *Neural Networks for Pattern Recognition* , Oxford University Press, 1995

[12] Tong Zhang and C.C. Jay Kuo, "Heuristic Approach for Generic Audio Data Segmentation and Annotation," *ACM Multimedia (1),* 1999, pp 67-76.

[13] Beth Logan, " Mel Frequency Cepstral Coeffcients for Music Modelling," *in international Symposium on Music information retrieval*, October 2000.

[14] John R. Deller, Jr., John H.L. Hansen and John G. Proakis, *Discrete-Time Processing of Speech Signals,* IEEE Inc. 2000.

[15] John G. Proakis and Dimitris G. Manolakis, *Digital Signal Processing principles, algorithms and applications,* Prentice-Hall, Inc, 1996.

[16] L.R. Rabiner and R.W.Schafer, *Digital Processing of speech signals*, Prentice-Hall, 1978.

[17] MPEG Maaate. http://www.cmis.csiro.au/Maaate/