UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

SINGING VOICE DETECTION IN POLYPHONIC MUSIC

Martín Rocamora

Supervisor: Alvaro Pardo

A thesis submitted in partial fulfillment for the
degree of Master in Electrical Engineering

Universidad de la República
Facultad de Ingeniería, Instituto de Ingeniería Eléctrica
Departamento de Procesamiento de Señales

August, 2011

For all that music gives us.
For the spiritual and the mundane.
Its ability to bring and to
transcend the here and now.
The connection with life and beyond.

To Ernesto, Pablo and Luis

For Natalia

Abstract

When listening to music most people are able to distinguish the sound of different musical instruments, though this ability may require some training. However, when it comes to the singing voice, anyone can easily recognize it from the other several instruments of a musical piece. This dissertation deals with the automatic detection of singing voice in polyphonic music recordings. It is motivated by the idea that the automatic identification of the segments of a song containing vocals would be a helpful tool in music content processing research and related applications. In addition, the efforts on building such a tool could contribute to some extent to sound perception understanding and its emulation by machines.

Two different computer systems are developed in this work that process a polyphonic music audio file and produce in return labels indicating time intervals when singing voice is present. Each of them corresponds to a different conceptual approach.

The first one is a pattern recognition system based on acoustic features computed from the audio sound mixture and can be regarded as the standard solution. A significant effort has been put into its improvement by considering different acoustic features and machine learning techniques. Results indicate that it seems rather difficult to surpass certain performance bound by variations on this approach.

For this reason, a novel way of addressing the singing voice detection problem was proposed, that involves the separation of harmonic sound sources from the polyphonic audio mixture. This is based on the hypothesis that sounds could be better characterized after being separated, which would provide an improved classification. A non-traditional time-frequency representation was implemented, devised for analysing non-stationary harmonic sound sources such as the singing voice. Besides, a polyphonic pitch tracking algorithm was proposed, which tries to identify and follow the most prominent harmonic sound sources in the audio mixture. Classification performance indicates that the proposed approach is a promising alternative, in particular for not much dense polyphonies where singing voice can be correctly tracked. As an outcome of this work an automatic singing voice separation system is obtained with encouraging results.

Acknowledgements

It goes without saying that the work reported in this document is a collaborative effort and would not have been possible without the help of many people. I am really lucky to have had such great advisors, colleagues and friends during all this time. At the risk of unfair omission, I want to express my gratitude to them.

First I would like to express my deepest gratitude to my supervisor Alvaro Pardo for his endless support, his enthusiasm for my work and his friendship. I would like to thank the financial support of the Comisión Académica de Posgrados and Comisión Sectorial de Investigación Científica, Universidad de la República. Thanks to the examination committee: Luiz W. P. Biscainho, Paulo A. A. Esquef and Luis Weruaga, for their valuable review and suggestions to improve this document. I am grateful to all my fellow workers at the Electrical Engineering Institute and at the Electroacoustic Music Studio. It is difficult to mention just a few, but I would like to thank Juan Martín López, Guillermo Carbajal and Juan Cardelino.

I would like to express my gratitude and my admiration to my dearest friends Ernesto López and Pablo Cancela. Their attitude towards research improved and influenced this thesis to great extent. My deepest gratitude to Luis Jure for his friendship and advice, and for everything I have learned working with him. Thanks also to the other members of the Audio Processing Group, Nacho Irigaray and Haldo Sponton, because of their support and friendship. I am very honoured to work with all of them.

I had the privilege to spend a great and fruitful time during an internship at the Music Technology Group (MTG), Universitat Pompeu Fabra, Barcelona. I would like to thank all the members of the MTG, and especially my office mates, Cyril Laurier, Joan Serrà, Enric Guauss, César Alonso and Emilia Gómez for the discussions and insights at the beginning of this work. My immense gratitude to my supervisor Perfecto Herrera for his knowledge and kind generosity devoting much time to my work during my stay.

I am especially thankful to my dearest friend Diego Azar for his inspiring music and for giving me the chance to work together and learn a lot while having fun.

Finally, to my family. Thanks to Mariana, Larisa, Rodrigo, María, Graciela and Guillermo. My deep gratitude to my mother for his support and love. I am grateful to my father to whom I owe my love for music and my interest in computers. A especial thanks to my beloved brother who is an essential support in my life. And to my dearest love Natalia, I am grateful for everything, including the miracle of life, Juana and Lorenzo.

Martín Rocamora

Contents

1 Introduction

1.1 Problem statement

When listening to music most people are able to distinguish the sound of different musical instruments. This ability may require some training, such as the one that is acquired just by carefully listening to music. However, when it comes to the singing voice, anyone can easily recognize it from the other several instruments of a musical piece, and even more, it usually becomes the focus of our attention. This fact should not be surprising, since our auditory system is particularly oriented towards oral communication through voice. Furthermore, singing voice usually conveys much information in music, because it commonly expounds the main melody, contains expressive features and carries lyrics.

This dissertation deals with the automatic detection of singing voice in polyphonic music recordings, i.e. where there are several simultaneous acoustic sources. It aims to build a computer system intended to process a music audio file and produce in return labels indicating time intervals when singing voice is present, such as the ones depicted in Figure 1.1. This requires the implementation of software that explicitly deals with those distinctive acoustic features that allow to discriminate between singing voice and other musical instruments; even when combined in complex sound mixtures where some of these characteristics may be obscured by the presence of several concurrent sounds. Yet, despite the fact that musical instruments recognition is quite a simple task for musically trained people, we still do not possess a complete understanding of which are the processes involved in our perception of timbre. In addition, the singing voice is one of the most complex musical instruments, able to produce a huge amount of different sounds and expressive nuances (for a single person and between different singers, music styles, languages, etc.). All of this makes the automatic detection of singing voice in polyphonic sound mixtures a very challenging problem. One of the most troublesome characteristics of the problem is the variability of music, both with regards to the singing performance and to the accompaniment. In order to limit the scope of the work we focus on western popular music, which is in fact a very broad category anyway. For this reason, clarifications are given throughout this dissertation, such as the type of music a given algorithm is appropriate for, or which singing styles or musical instruments are problematic for a certain technique.

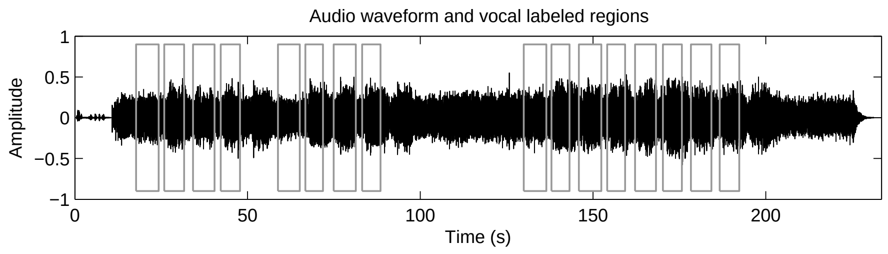


FIGURE 1.1: Audio waveform and manually labeled vocal regions for a popular song.

1.2 Context and motivation

Much research in the area of audio signal processing over the last years has been devoted to music content retrieval, that is the extraction of musically meaningful content information by analyzing an audio recording. The automatic analysis of either an individual audio file or large music collections makes it possible to tackle diverse music related problems and applications, from computer aided musicological studies [1] to automatic music transcription [2] and recommendation [3]. Not surprisingly, several research works deal with the singing voice, such as singer identification [4], singing voice separation [5], singing voice melody transcription [6], query by humming [7], lyrics transcription [8], among others. This kind of research would probably benefit from a reliable segmentation of a song into singing voice fragments. Furthermore, singing voice segments of a piece are valuable information for music structure analysis [9], as can be inferred from the vocal labels in Figure 1.1.

This thesis work is motivated by the idea that the automatic identification of the segments of a song containing vocals would be a helpful tool in music content processing research and related applications. In addition, we hope that the efforts on building such a tool could contribute to some extent to sound and music perception understanding by humans and its emulation by machines.

1.3 Description of the present work

This work explores the existing methods to tackle the problem of singing voice detection in polyphonic music and implements the most common approach, namely a statistical pattern recognition system based on acoustic features computed directly from audio sound mixtures. A study of the acoustic descriptors reported to be used for the problem is conducted, which involves their comparison under equivalent conditions. Different classifiers are applied as well, and their parameters finely tuned. In this way, an estimate of the achievable performance that can be expected following this approach is obtained. This work indicates that variations on a system of this kind provide no improvement on singing voice detection performance, which raises the question of whether the approach might suffer from the *glass ceiling* effect that has been ascertained in other similar music information retrieval problems [10]. This calls for new paradigms for addressing the problem.

For this reason, a different strategy is proposed, which involves the extraction of harmonic sound sources from the mixture and their individual classification. This is based on the hypothesis that sound sources could be better characterized after being isolated in a way which is not feasible when dealing with the audio mixture. The proposed sound sources separation technique comprises a non-traditional time-frequency analysis in order to capture typical fluctuations of the singing voice, and algorithms for multiple fundamental frequency estimation and tracking. Most of this work finds application beyond this particular research problem. Some features are proposed to exploit information on pitch evolution and prominence, that proved to be informative about the class of the extracted sounds. This information is combined with traditional spectral power distribution features to perform the classification. Obtained results indicate the proposed approach is a promising alternative, in particular for not much dense polyphonies where singing voice can be correctly tracked. However, the sound sources separation introduces new challenges and further work is needed in order to improve the performance results on more complex music and to overcome its main limitations. As an outcome of this work an automatic singing voice separation system is obtained with very encouraging results. Both of the studied approaches for singing voice detection that were implemented within this thesis work are outlined in Figure 1.2.

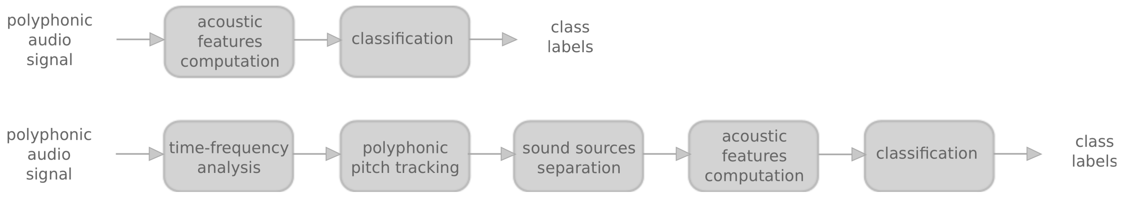


FIGURE 1.2: Block diagrams of both approaches implemented within this thesis work for addressing the problem of singing voice detection in polyphonic music.

1.4 Thesis outline

The remainder of this dissertation is organized as follows. Chapter 2 is a survey of the existing approaches for addressing the singing voice detection problem in polyphonic music. Related research fields and problems are identified and acoustic characteristics of the singing voice are mentioned. Then the type of solutions proposed is described, considering the kind of features and classification methods. Finally a summary of the most relevant research work on the subject is given, which determines the most common approach to the problem, and highlights in the end recent work which seems enlightening.

Then the dissertation is divided into two parts which correspond to the different approaches implemented. Part I comprises the study and development of a classification system based on acoustic features computed directly from the polyphonic audio mixture. Chapter 3 is devoted to describing concepts, methods, tools and data that are used along the process of developing the pattern recognition system. Then, Chapter 4 explores most of the features reported to be used for the singing voice detection problem. Different techniques are applied in Chapter 5 for feature selection, extraction and combination. Based on this study the most appropriate set of features is selected for the following development steps. In Chapter 6 various automatic classification algorithms are studied and their parameters are finely tuned. The performance of the different algorithms is compared for the classification of polyphonic music. As a result, a singing voice detection system is obtained which follows the classical pattern recognition approach.

Part II corresponds to the proposed approach based on sound source separation and classification. It includes in Chapter 7 a description of the time-frequency representation techniques developed in our research group intended to overcome limitations of classical tools and to take advantage of a-priori knowledge of music signals, in particular of the singing voice. Then, in Chapter 8 a polyphonic pitch tracking technique is proposed that makes use of tools presented in Chapter 7 for building a pitch salience representation and performs temporal integration of local pitch candidates based on an unsupervised clustering method. In this way, pitch contours of the more prominent harmonic sound sources in the analysed audio signal are obtained. Chapter 9 is devoted to describing how the identified harmonic sources are extracted from the audio mixture and to the process of classifying each of the isolated sounds. Some new features are proposed based on pitch related information and they are used in conjunction with spectral power distribution features as the input for a classification system of isolated sound sources.

Finally, Chapter 10 summarizes the implemented approaches and compares them with different testing databases. The dissertation ends with a critical discussion on the present work, the main conclusions and some of the more relevant ideas for future work.

Further information is provided at: http://iie.fing.edu.uy/∼rocamora/mscthesis/

2

Survey of the existing approaches to singing voice detection

2.1 Introduction

To address the singing voice detection problem the knowledge on research fields such as classification of musical instruments [11, 12] and speech processing [13, 14] are of particular relevance. The former studies the ability to distinguish different musical instruments. Singing voice detection can be considered a particular case of musical instrument classification in complex mixtures, so many features used in this field may be useful for characterizing vocal and non-vocal segments of a song. Given the similarities between speech and singing voice it is reasonable to apply techniques and descriptors used to segment and recognize speech to singing voice problems. Speech/music discrimination, singing voice separation and singer identification are closely related problems, as many systems developed to perform these tasks try to identify those fragments of the audio file containing vocals.

Although singing voice resembles speech to a certain extent there are significant differences between them that need to be taken into account. To sing a melody line with lyrics it is usually necessary to stretch the voiced sounds and shrink the unvoiced sounds to match note durations.[1] For this reason, singing voice is more than 90% voiced, whereas speech is only approximately 60% voiced [15]. The majority of the singing voice energy falls between 200 Hz and 2000 Hz [16], but in speech the unvoiced sounds are more common and tend to raise this energy limit up to 4000 Hz. Speech has a characteristic energy modulation peak around the 4-Hz syllabic rate usually considered as an evidence of its presence in automatic speech processing [17]. With regards to pitch, in natural speech the fundamental frequency slowly drifts down with smooth changes. The fundamental frequency contour of the singing voice compared to speech tends to be more piece-wise constant with abrupt changes in between, following the pitch of notes. However, pitch variations are also very common in a vocal music performance since they are used by the singers to convey different expressive intentions and to stand out from the accompaniment. Besides this, speech pitch is normally between 80 to 400 Hz whereas singing has a wider pitch range that can reach 1400 Hz in a soprano singer [18, 19]. Moreover, singing voice is highly harmonic, that means that the partials of the sound are located at multiples of the fundamental frequency. Additionally, a known feature of operatic singing is the presence of an additional formant (resonance of the vocal tract), called the singing formant, in the frequency range of 2000 to 3000 Hz, that enables the voice to stand out from the accompaniment [20]. However, the singing formant does not exist in many other types of singing such as the ones in pop or rock.

2.2 Previous work

The common procedure adopted for partitioning a song into vocal and non-vocal segments is to extract features from audio signal frames (nearly stationary block of samples) at each few tens of milliseconds, and then to classify them as pertaining to one of the possible classes using a threshold method or a statistical classifier.

Threshold methods tend to be simple but need descriptors that clearly discriminate between classes in order to be successful. The methods proposed to classify audio frames into vocal and non-vocal apply a threshold on only one descriptor [21, 22] or compute different descriptors and apply a set of thresholds on them [23, 24]. On the other hand, statistical classifiers are trained using accompanied singing voices and pure instrumentals and can learn complex boundaries between classes combining several descriptors. This has the drawbacks of a certain amount of time spent on training and the difficulty of obtaining ground truth annotations for this purpose. In the singing voice detection problem, finding the exact boundaries over the entire song can be time-consuming, and sometimes could be difficult in case of slow decays or masking. Moreover, special attention has to be paid to ensure generalization beyond the training set avoiding overfitting.

[1]Vocal sounds are usually divided by speech researchers in voiced and unvoiced. The vibration of the vocal folds produces quasi-periodic sound referred to as voiced. Those vocal sounds generated by the turbulence of air against the lips or tongue (such as the consonants “s” or “f”) are known as unvoiced and their waveform appears random though with some limited spectral shaping [14].

Several statistical classifiers have been explored to address the problem of singing voice detection, such as Hidden Markov Models (HMM) [25, 26], Gaussian Mixture Models (GMM) [27, 28], Artificial Neural Networks (ANN) [29, 30] and Support Vector Machines (SVM) [21, 31, 32].

The short-term classification of each signal frame considers only local information so it is prone to errors, and the classification obtained is typically noisy, changing from one class to the other. For this reason, usually long-term information is introduced, by smoothing the classification [27] or by partitioning the song into segments (much longer than frames) and assigning the same class to the whole segment. This partitioning of the song is performed based on tempo [26, 31], chord change [21] or spectral (timbre) change [28]. More global temporal aspects of a song are also taken into account by modeling the song structure (intro, verse, chorus, etc) by means of an HMM [26].

With regards to descriptors, research on musical instruments classification has demonstrated the importance of temporal and spectral features [11, 12], and the speech processing field has contributed to well known techniques (such as Linear Prediction) to compute voice signal attributes [13, 14]. A broad group of descriptors has been used for the purpose of singing voice detection. Singing voice carries the main melody and the lyrics of a popular song, so it is usually one of the most salient instruments of the mixture. Therefore, vocal frames can be identified by an energy increase of the signal, and energy or power descriptors are often used [22, 23, 26, 30, 31]. The timbre of different instrument sounds is partially dictated by their spectral envelope and when a new sound enters a mixture it usually introduces significant spectral changes. Among the descriptors computed to represent the spectral energy distribution are Mel Frequency Cepstral Coefficients (MFCC) [27, 28, 31], Linear Prediction Coefficients (LPC) (warped LPC or perceptually derived LPC) [16, 25, 29, 31], Log Frequency Power Coefficients (LFPC) [26], and spectral Flux, Centroid and Roll-Off [23, 30]. The delta coefficients (i.e. derivatives, see equation 4.1) of the previous features or of their variances are also used to capture temporal information [29]. As stated previously, singing voice is highly harmonic, thus the harmonicity of the signal, usually computed as an Harmonic Coefficient (HC), is used as a clue for singing voice detection [16, 23, 33].

Regarding harmonicity a pre-processing technique was proposed to attenuate the non-vocal harmonic sounds in the mixture [22, 26]. After estimating the key of the acoustic musical signal, an inverse comb filter is applied to attenuate those harmonic patterns originated from the pitch notes in the key. The pitch contours of the harmonic sounds of the accompaniment tend to be more steady compared to those of the singing voice (that usually exhibit vibrato and intonation). Therefore, the non-vocal harmonic sounds are more attenuated than the vocal ones after filtering.

The following is a summary of the most relevant research work on singing voice detection, which is outlined in table 2.1. To our knowledge, the first work that focused and described the problem of locating the singing voice segments in music signals was

Table 2.1: *Previous work chart.*

Reference	Descriptors	Classifiers
Berenzweig and Ellis [25]	Posterior Probabilities, PLP	ANN trained for speech
Chou and Gu [33]	Harmonic coef., 4Hz modulation, MFCC	GMM
Kim and Whitman [16]	Harmonicity	Threshold
Zhang [23]	Harmonic coef., Energy, ZCR, Flux	Threshold
Maddage et al. [31]	LPC, Spectral Power (MFCC, ZCR)	SVM (ANN, GMM)
Maddage et al. [21]	Twice iterated Fourier Transform	Threshold
Tzanetakis [30]	Centroid, Roll-off, Flux, Energy, Pitch	Logistic, ANN (SVM, Tree)
New et al. [26]	Log-Frequency Power Coefficients	Multi-Model HMM (song)
Shenoy et al. [22]	Frequency sub-bands energy	Threshold
Tsai and Wang [27]	MFCC	2 GMM (vocal, non-vocal)
Li and Wang [28]	MFCC	4 GMM models

[25]. Posterior probability features and their statistics are derived from an ANN trained on a database of phonemes to work with speech. An HMM with two states is used to discriminate singing from accompaniment. Close in time, an approach for singing detection in speech/music discrimination is described in [33]. It employs a set of features that comprises MFCC, HC, 4-Hz modulation and energy based features to train a GMM.

Following these early works, new other approaches for singing detection were proposed. Artist classification is improved in [29] voice segments are detected with a multi-layer perceptron that is fed with Perceptual LPC (PLPC), plus deltas and double deltas (see equation 4.1). The vocal detected segments provide an improvement in the artist classification task. An harmonic sound detector is presented in [16] to identify vocal regions of an audio signal for singer identification. It works under the hypothesis that most harmonic sounds correspond to regions of singing. By means of an inverse comb filter bank, the signal is attenuated and the Harmonicity is computed as the ratio between the total energy in a frame over the energy of the most attenuated signal. The automatic singer identification system described in [23] identifies the starting point of the singing voice in a song using energy features, zero-crossing rate (ZCR), HC and Spectral Flux and performs classification with a set of thresholds. In [31] a study which aims to establish if there are significant statistical differences between vocal music, instrumental music and mixed vocal and instruments is described. The set of descriptors contains LPC, LPC derived Cepstrum, MFCC, Spectral Power, Short Time Energy and ZCR. Performance of an SVM is shown to be superior to that of an ANN or a GMM.

Subsequent research explored other descriptors and classification approaches. A technique for singing voice detection is proposed in [21] in which musical signals are segmented into beat-length frames using a rhythm extraction algorithm. The Fast Fourier Transform (FFT) of each frame is calculated, and after filtering to a narrow bandwith containing mostly voice, another FFT is applied (Twice Iterated Composite Fourier Transform, TICFT). Based on a threshold over the energy of the TICFT, singing voice frames are separated from instrumental frames. Performance is improved by some frame-correction rules based on chord pattern changes. In [30] singing voice detection is performed by a bootstrapping process that consists in manually annotating a few random fragments of the song being processed to train a classifier. The feature set includes Mean Relative Energy, and Mean and Standard Deviation of Spectral Centroid, Roll-off, Flux and Pitch. Different classifiers are tested, being Logistic Regression and ANN those which performed best. In the work described in [26], a key estimation and an inverse comb filtering is performed to attenuate the harmonic sounds of the accompaniment. The LFPC computed after this process show an energy distribution in which the vocal segments have relatively higher energy values in the higher frequency bands. Classification is done with a multi-model HMM that models the different sections of a typical song structure (intro, verse, chorus, bridge, or outro). A classification refinement is provided by a verification step based on classification confidence and an automatic bootstrapping process (similar to that proposed in [30]). The work in [22] also addresses the singing voice segmentation problem by estimating the key of the song in order to perform an inverse comb filtering to attenuate the harmonic sounds of the accompaniment. Singing voice is retained after filtering due to vibrato and intonation. The energy in different frequency sub-bands is computed and the highest energy frames are classified as vocal.

Most recent works use MFCC feature vectors and classifiers such as GMM as the standard machine learning approach to singing voice detection. A system of this type for vocal/non-vocal classification is used in [27, 34, 35] with application to to blind clustering of music, singing language identification and singer recognition, respectively. The class of each frame is hypothesized according to log-likelihoods and the final decision is made at homogeneous segments. In [5, 28] the problem of singing voice separation from music accompaniment is addressed, and singing voice detection is performed following this same approach. The audio is partitioned by detecting large spectral changes. Segments are classified in a similar way as described before, by considering the log-likelihoods of all the frames within a segment.

Certain improvements were reported in recent work by attempting a better temporal integration of the posterior probabilities yielded by the classifiers, by means of an HMM in [32] and by an autoregressive moving average filtering (ARMA) in [36].

An interesting approach proposed lately involves trying to explicitly capture vibrato or frequency modulation features of the singing voice [24, 37, 38]. This is usually performed by sinusoidal modeling followed by the analysis of modulations for each of the identified partial tracks. However, the authors themselves report that the vocal detection performance is inferior than that of the standard pattern recognition approach [24, 38].

Of special relevance are the results reported in [39] on the influence of music accompaniment on low-level features, when addressing the classification of audio segments into rap and other types of singing. Some features that yield useful results when applied on isolated vocal tracks are not able to preserve information about the vocal content when mixed with background music. In addition, the performance of a standard machine learning classifier depends on the background music of the training and testing data. The authors suggest that it is quite probable that the classifier is in fact considering phenomena present in the accompaniment that occur in correlation with the vocal characteristics. This indicates that trying to derive information on a particular sound source based on features computed from the audio mixture can be misleading.

Also of particular interest to the present research is the study on music perception described in [40]. An experiment was conducted on subjects to examine how the length of a sound excerpt influences the ability of the listener to identify the presence of a singing voice in a short musical excerpt. The results indicate that subjects perform well above chance even for the shortest excerpt length (100 ms). This suggest that there is information on the presence of vocals in music recordings for such short temporal segments. In addition, it is reported that transitions between notes appear to benefit listenersṕerceptual ability to detect vocals.

Part I

Detection based on spectral features from audio sound mixtures

3 Methods

In the first part of this dissertation a pattern recognition system is designed to tackle the singing voice detection problem. It is based on acoustic features computed directly from audio mixtures. The development follows the typical steps for building a supervised pattern recognition system. This chapter introduces a few concepts, methods and databases that are used along this process.

3.1 Design of a classification system

The basic stages involved in the design of a classification system [41] are described in the following. Note that they are not independent, but strongly related. Based on the results from one step it may be necessary to revisit an earlier task in order to improve the overall performance of the system. Moreover, some methods combine several stages, for instance classification and features selection in a single optimization loop.

Features generation Involves computing measurable quantities from the available data. The features generation is detailed in Chapter 4.

Features selection and extraction Is the process of identifying those features (or transformations of them) that make the classes most different from each other, i.e. maximize inter-class distances. It also deals with selecting the number of features to use. This is described in Chapter 5.

Classifier design Implies learning a decision boundary between classes in the feature space based on a certain optimality criterion and the available data. This is the task covered in Chapter 6.

Performance evaluation This is the task of assessing performance, in order to evaluate how different methods work and compare to each other, as well as estimating performance on new data not used for training. The following section (3.2) elaborates on this.

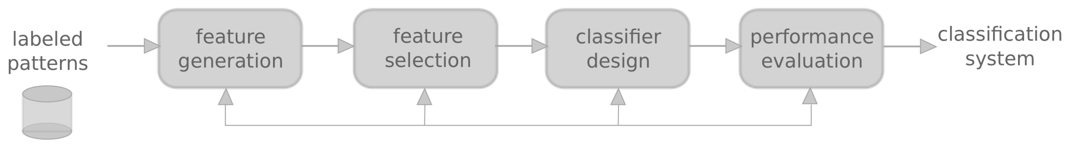


FIGURE 3.1: Stages involved in the development of a classification system.

From a statistical point of view the classification problem can be stated as follows. Consider the two-class case, where w_1 and w_2 are the classes, and x represents the feature vector of an unknown pattern that shall be classified. Suppose we can compute the conditional probabilities $P(w_i|x)$ $i = 1, 2$, referred as *posteriori class probabilities*, which represent the probability that the pattern belongs to the class w_i given that the observed feature vector is x. The pattern is classified to w_1 if $P(w_1|x)$ is greater than $P(w_2|x)$, or to w_2 otherwise. Using the Bayes' rule $P(w_i|x)$ can be expressed as,

$$P(w_i|x) = \frac{p(x|w_i)P(w_i)}{p(x)},$$

where $P(w_i)$ are the *a priori class probabilities*, $p(x|w_i)$ are the *class-conditional probability* density functions (also referred as *likelihood function* of w_i with respect to x) and $p(x)$ is the probability density function of x, that can be computed as $p(x) = \sum_{i=1}^{2} p(x|w_i)P(w_i)$. Thus, ignoring $p(x)$ because it is the same for both classes, the classification rule can be formulated as $p(x|w_1)P(w_1) \gtrless p(x|w_2)P(w_2)$.

The class-conditional probabilities $p(x|w_i)$ describe the distribution of the feature vectors in each of the classes and can be estimated from the available training data. The a priori class probabilities may be known or can also be derived from the training data.

In the singing voice detection problem, the priors are dependent on the application, for instance, they will be very different if we aim at classifying short audio clips for a music recommender or if we are segmenting a whole song into vocal/non-vocal regions. Even in this latter case, priors may vary significantly for different music styles, as it is shown in Figure 3.2. Therefore, in the present work we considered that a priori probabilities are equal, that is, $P(w_1) = P(w_2) = 1/2$, being conscious that the classification performance can be improved if this is adjusted for a particular application or music material. It is interesting to note that the database of songs from mainstream commercial music used in [24, 32] for singing voice detection is well balanced (50.3% vocal and 49.7% of non-vocal).

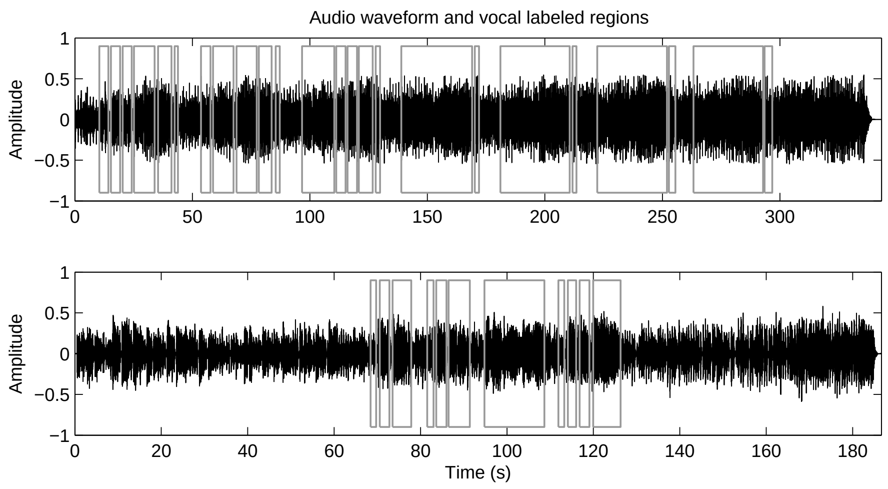


FIGURE 3.2: Two labeled songs that exhibit different a priori class probabilities (*Tangled up in blue* by Bob Dylan and *This Year's Kisses* by Billie Holiday).

3.2 Performance evaluation

In the design of a classification system there are some methodological issues to consider regarding the assessment of performance. One of them has to do with performance comparison of different classification algorithms and feature sets. It is necessary to try to ensure that the empirical performance differences obtained are not due to random effects in the estimation of performance (e.g. selection of a particular dataset), but from real substantive differences between the classification methods or feature sets compared. Another problem has to do with predicting the achievable performance of a classification system against new data (different from that used for training), i.e. its ability to generalize. Using the rate of success (or failure) of the system on the training data as a performance estimate is definitely a bad idea, since it is likely to be very optimistic. When the amount of data available for designing the classifier is very large there is no objection, the classification model is built with a large training dataset and performance is estimated with another large data set. However, in a real problem the amount of labeled data available is generally scarce (and usually obtaining them is expensive and involves the participation of experts). This raises the following trade-off: first in order to build a good classifier many of the available data should be used for training, while on the other side to get a reliable performance estimate as much data as possible must be booked for testing.

In the machine learning methodology the available data is usually subdivided into *training*, *validation* and *test* set.[1] The *training* data is used for learning the parameters of the classifier by minimizing a certain error function. The *validation* set is used to tune the parameters and to compare different classifier configurations or techniques. Finally, since the resulting classifier may be overffited to the validation data, an independent performance estimate is provided by the *test* set to assess the generalization ability of the fully-specified classifier.

Different alternatives to deal with a reduced data set have been proposed. One of them is to retain a certain portion of the available data at random only to estimate performance (*holdout method*). Care must be taken in order the data to be representative, so sampling is usually done *stratified*, i.e. ensuring that each class is adequately represented in the training and test set. A more general form of mitigating the bias produced by the particular selection of training and test sets is to repeat the process several times. In each iteration using different random samples of the available data to train and test. The error obtained in the different iterations is averaged to estimate the performance. A variant of this procedure, called *cross-validation* is to divide the total data set into a fixed number of partitions (folds) and repeat the process of training and validation using each of the partitions to evaluate the performance while the other is used for training. In this way each available data instance is used once for evaluation. Usually the partitioning is done stratified, resulting in a stratified cross validation. While the best way to evaluate performance with a reduced data set is still a controversial issue, stratified cross validation (using 10 partitions) is one of the most widely accepted methods [42]. To reduce the influence of the choice of such partitions, when looking for a more accurate estimate the cross-validation process is usually repeated (for instance, repeating 10 times the stratified cross validation with 10 partitions).

To contrast a classification algorithm or a feature set with another, one could simply compare the performance estimate obtained for each of them. However, it is desirable to establish whether the performance differences are not due solely to the particular dataset in which the estimate is based on. One way to address this problem is to construct several data sets of the same size for testing and to get an estimate of the performance of the algorithms on each of them (for example using cross validation). Each experiment returns an independent performance estimate. It is of interest to establish whether the mean of these estimates for a given algorithm is significantly higher or lower than the average for another. This can be done using a statistical tool such as the Student test or t-test. Considering the difference between estimates has a Student distribution, given a certain significance level (e.g. 95 %), the test determines whether the average difference is significantly different from zero (null hypothesis) by verifying if it exceeds the confidence intervals. When the amount of data is limited and the data partition process is repeated several times the estimates are not independent. In this case, a variant called corrected resampled t-test [43] should be used.

[1]The terms *validation* and *test* are often confused, and *testing* is used as a synonym for the latter.

Another useful tool for visualizing and assessing performance of classifiers is the Receiver Operating Characteristics (ROC) graph [44]. Consider a two-classes problem, namely a positive class and a negative class. Some classification methods produce a discrete label corresponding to each class, while others produce a continuous output to which different thresholds may be applied to predict class membership. Given an instance to be classified and the output of the classification system there are four different alternatives. If the instance is positive, it may be classified as positive which counts as a *true positive*, or it may be classified as negative which counts as a *false negative*. In the same way, if the instance is negative, a *true negative* is produced if it is correctly classified and a *false positive* if it is wrongly classified. Thus, the *true positive rate* (tpr) and the *false positive rate* (fpr) are defined as,

$$tpr = \frac{\text{true positive}}{\text{total positive}} \quad fpr = \frac{\text{false positive}}{\text{total false}}.$$

In a ROC graph true positive rate is plotted on the Y axis and false positive rate is plotted in the X axis, as depicted in Figure 3.3. A classifier that outputs only a class label produces a single point in the ROC graph space (discrete classifier). The location of this point indicates how the classifier balances the trade-off between high true positive rate and low false positive rate. The upper left point corresponds to perfect classification, but in a real situation increasing the true positive rate inevitably produces some false positives. If the output of the classifier is continuous, each threshold value produces a different point in the ROC graph space, and if the threshold is varied from $-\infty$ to $+\infty$ a continuous curve can be traced. Random guess corresponds to the diagonal line $y = x$.

The decision on where to establish the threshold determines the operating point of the classifier, and this may depend on the problem, typically on the cost of false positives or false negatives. Notice from the continuous curves on Figure 3.3 that depending on the operating point one classifier may be better than the other. To compare different classifiers the ROC curve can be reduced to a single value representing the expected performance by means of the area under ROC curve (AUC).[2] Its value is between 0 and 1, given that it is the portion of the area of the whole unit square, and a realistic classifier should have a AUC greater than 0.5 which corresponds to random guess.

In the singing voice detection problem the optimal operating point may depend on the particular application. For instance, if the detected segments are to be used for singer recognition it is desirable to have the minimum number of false positives, not to mislead the recognition, even if this implies a moderate true positive rate since only a small number of reliable detections would be enough for the identification.

[2]The AUC has an important statistical property: the AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. [44]

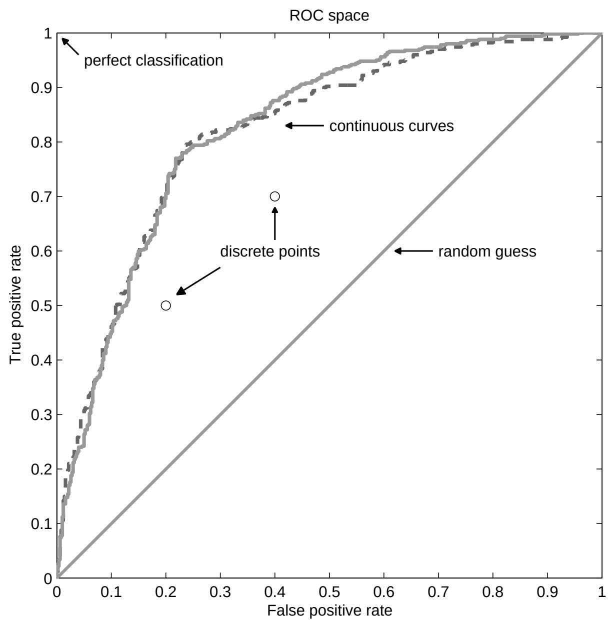


FIGURE 3.3: ROC space plot. Perfect classification point and random guess line are depicted. Two points are indicated corresponding to two different discrete classifiers. The curves for two different continuous output classifiers show that one classifier can be better than the other depending on the operating point.

3.3 Audio datasets

Independent datasets of popular music recordings were used for training, validation and testing for both approaches to the singing voice detection problem implemented in this dissertation. The characteristics of the dataset used in each case are described in due course throughout the document. However, appendix B provides further details an serves as an unified reference for all the audio databases mentioned in this thesis work.

In the following, the databases used in Part I of this dissertation are introduced. The training database [TRAIN1] was built by automatically extracting short audio excerpts from music recordings and manually classifying them into vocal and non-vocal. Three different excerpt-length sets were constructed of 0.5, 1 and 3 seconds length. A vocal excerpt is discarded if it does not contain vocal sounds for at least more than 50% of its length. The non-vocal excerpts do not have any aurally identifiable vocal sound. All different-length sets contain 500 instances of each class. The music utilized belongs to a music genres database comprising alternative, blues, classical, country, electronic, folk, funk, heavy-metal, hip-hop, jazz, pop, religious, rock and soul. In addition, approximately 25% of pure instrumental and a cappella music was also added.

A validation database, that consists of 63 fragments of 10 seconds that were manually annotated, was used to select the fragment length and to evaluate some post-processing strategies. This is described in our preliminary study reported in [45] and some of these results are used in the development of the classification system. Music was extracted from Magnatune[3] recordings belonging to similar genres as the ones used for training (in fact the same ones except for classical and religious). A subset of this database, referred as [VALID2], was also used for validating the second approach proposed in this dissertation which is described in Part II, but with a reduced set of musical genres.

In Chapter 6 a comparison of different classifiers and sets of features is performed using a validation set [VALID1]. This database comprises 30 songs by *The Beatles* from the albums *Abbey Road* and *A hard day's night* that were manually labeled.

Once the features and the classifier were set and its parameters finely tuned for both classification approaches proposed in this dissertation, an independent evaluation was conducted on two different test databases [TEST1] and [TEST2] that are introduced in Chapter 10.

3.4 Software tools

The following is a brief summary of the main software tools used and developed in the course of this dissertation research work, some of which are available online.

The computation of acoustic features was implemented in `Matlab`, and different parts of the code were based on functions provided by [46].

The manual labeling of audio files was done using `wavesurfer`[4]. Different audio editing tasks were performed with `Audacity`[5]. Several scripts were implemented in `bash` and `python` that make use of audio processing tools such as `sox`[6] and `snack`[7].

The simulations on features extraction and selection, as well as on the classification methods were performed using the `Weka`[8] software platform [42].

The classification systems developed to process polyphonic music files run in `Matlab` to deal with the audio files and compute the acoustic descriptors and perform the classification by invoking a program coded in `Java` that imports `Weka` software classes.

The time-frequency analysis techniques and the pitch tracking algorithm were implemented in `Matlab` and `C` code.

[3]`http://magnatune.com/`
[4]`http://www.speech.kth.se/wavesurfer/`
[5]`http://audacity.sourceforge.net/`
[6]`http://sox.sourceforge.net/`
[7]`http://www.speech.kth.se/snack/`
[8]`http://www.cs.waikato.ac.nz/ml/weka/`

4 Audio features generation

4.1 Introduction

In the context of this thesis work a study of the acoustic descriptors reported to be used for the singing voice detection problem was conducted, in which they were compared under equivalent conditions. This chapter describes the feature generation process for each of them.

Based on the existing literature the following descriptors were implemented. Most common features used in previous work are different ways of characterizing spectral energy distribution. Within this category we considered: Mel Frequency Cepstral Coefficients (MFCC), Perceptually derived LPC (PLPC) and Log-Frequency Power Coefficients (LFPC). Another feature implemented is the Harmonic Coefficient (HC), which provides a degree of harmonicity of the signal frame. Pitch was also included, being the only non-spectral feature reported that was considered relevant (4Hz modulation is appropriate for speech but not for singing [33], zero-crossing rate (ZCR) strongly correlates with the Spectral Centroid [12] and other power or energy features can be regarded as variants of spectral descriptors such as LFPC). In addition, a general purpose musical instruments classification feature set was built, including Spectral Centroid, Roll-off, Flux, Skewness, Kurtosis and Flatness. Apart from that, other sources of information that could provide new clues for singing voice detection were explored. Unfortunately none of them was considered significantly relevant compared to other existing features, even when combined in heterogeneous feature sets, and were eventually discarded (though they are briefly reported in Section 4.2.7).

4.2 Descriptors computation

Audio signal is processed in frames of 25 ms using a Hamming window and a hop size of 10 ms. Considering that the majority of energy in the singing voice falls between 200 Hz and 2000 Hz [16], and based on performance results obtained in simulations, the frequency bandwidth for the analysis is set to 200 Hz to 16 kHz.

4.2.1 Mel Frequency Cepstral Coefficients

The Cepstrum analysis is widely used in speech processing, mainly because of its usefulness on separating the representation of the voice excitation from the vocal tract filter [47, 48]. This is achieved by turning the filtering in the frequency domain into an addition by using the logarithm as

$$S(f) = X(f)\,H(f), \quad \log(|S(f)|^2) = \log(|X(f)|^2) + \log(|H(f)|^2)$$

where $S(f)$ and $X(f)$ are the spectra of the speech signal and that of the voice excitation respectively, and $H(f)$ is the vocal tract filter frequency response. Then, the real cepstrum is obtained by the inverse Fourier Transform, thus yielding a time domain variable τ

$$c(\tau) = \mathrm{FT}^{-1}\left[\log(|S(f)|^2)\right] = \mathrm{FT}^{-1}\left[\log(|X(f)|^2) + \log(|H(f)|^2)\right].$$

A common set of features derived from the discrete time Cepstrum analysis are the Mel Frequency Cepstral Coefficients (MFCC), which characterize the magnitude of the spectrum using a reduced number of coefficients. Lower coefficients tend to represent the envelope of the spectrum, while higher ones describe finer detail of the spectrum [47]. A mapping from the linear frequency scale to the Mel scale [49] is performed in order to better approximate humansáuditory, since it has less resolution at high frequencies and a finer resolution at low frequencies. The mapping can be computed according to,

$$f_{\text{mel}} = 2595 \log_{10}\left[\frac{f}{700} + 1\right].$$

A comparison of the different frequency scales used in the features generation process, including the Mel scale, is shown in Figure 4.3. Apart from the speech processing field, MFCC features are also commonly used in music information retrieval (e.g. musical instruments recognition [50, 51]). A classical study of their usefulness on music modeling can be found in [52]. The use of this particular frequency scale instead of other approximately logarithmic spacing seems not to be so critical, and slightly better results have been reported using Octave Scale Cepstral Coefficients instead [9].

The implementation of MFCC is based on [46] and derives for each signal frame 13 coefficients from 40 mel scale frequency bands. The computation process is outlined in Figure 4.2. An FFT is applied to the signal frame and the squared magnitude spectrum is obtained. After that, the magnitude spectrum is processed by a filter bank, which is depicted in Figure 4.3, whose center frequencies are spaced according to the mel scale. Then, signal power on each band is computed and the logarithm is taken. The elements of these vectors are highly correlated so a Discrete Cosine Transform (DCT) is applied, which transforms back to a time domain, and dimensionality is further reduced by retaining only the lowest order coefficients. An example of the evolution of lower order MFCC coefficients for a music excerpt is shown in Figure 4.1.

4.2.2 Perceptually derived Linear Prediction

Voice modeling by the Linear Prediction technique is typically carried out by considering the vocal audio signal $s(n)$ as the output of an all-pole system, i.e. an autoregressive model. The actual signal sample value is predicted by linear combination of past samples,

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + e(n)$$

where a_k are the linear prediction coefficients, p is the order of the model and $e(n)$ is the prediction error, also known as the residual. A common approach for determining the linear prediction coefficients is to minimize the total quadratic prediction error, $E = \sum_n e^2(n)$. This can be solved efficiently using methods such as the Levinson-Durbin recursion. The obtained coefficient characterize the all-pole filter that describes the spectral envelope of the analyzed signal frame.

Some psychoacoustic concepts are introduced in the Perceptually derived Linear Prediction Coding (PLPC) [53] analysis technique that make it more consistent with human hearing in comparison with conventional LPC analysis. The PLPC coefficients are obtained by a critical band power integration of the signal, following the Bark scale [54]. This scale gives an analytical approximation to the *critical bands* of hearing, which are frequency regions in which neighbouring frequencies interact. The mapping is computed as [46],

$$f_{\text{bark}} = 6 \text{ asinh}\left(\frac{f}{600}\right)$$

and is shown in Figure 4.3. The critical band power integration is followed by equal loudness weighting and intensity to loudness conversion (cubic root of sound intensity) [53]. Finally, a conventional Linear Prediction analysis is applied. The process is outlined

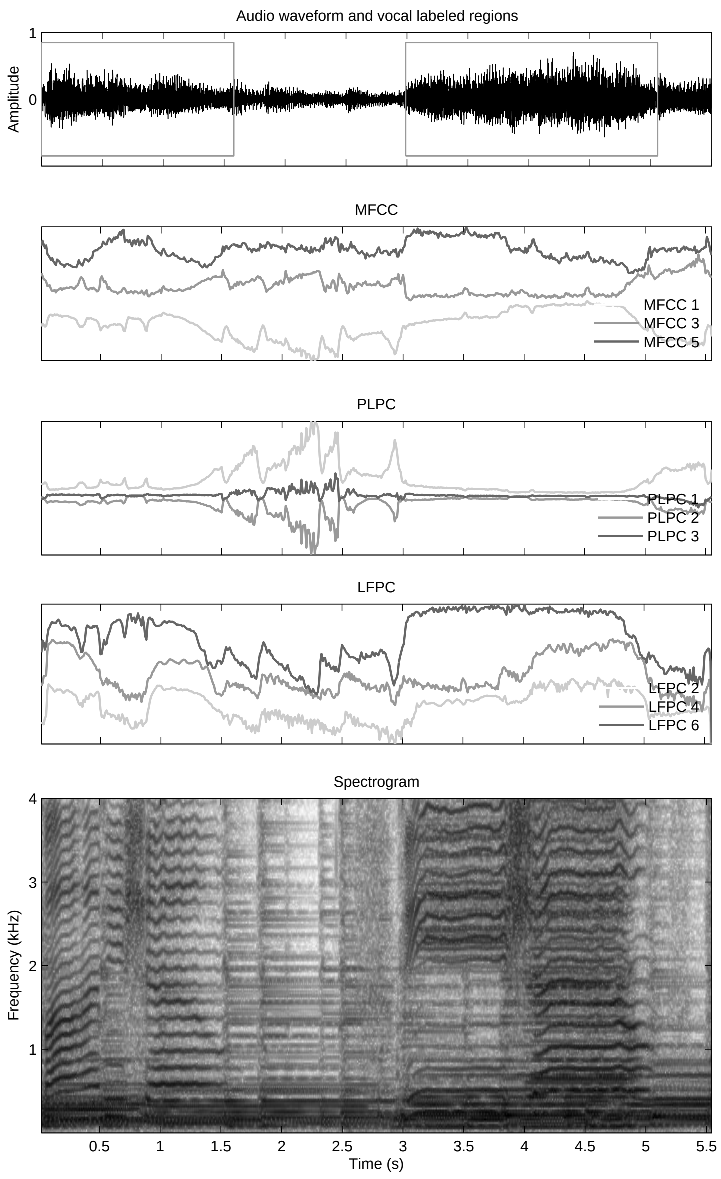


FIGURE 4.1: Example of the MFCC, PLPC and LFPC features aimed at describing spectral energy distribution for a music audio excerpt. It can be noticed how the evolution of the features is related to the changes on the spectral content of the signal.

in Figure 4.2 and is implemented based on [46] using a model order of 12. An example of PLP coefficients evolution for a music audio excerpt is given in Figure 4.1.

4.2.3 Log-Frequency Power Coefficients

A simple way of describing spectral power distribution can be achieved by means of the Log-Frequency Power Coefficients (LFPC). The computation process is outlined in Figure 4.2. The signal frame is passed through a bank of 12 band-pass filters spaced logarithmically, and the coefficient of each band is obtained by computing the power of

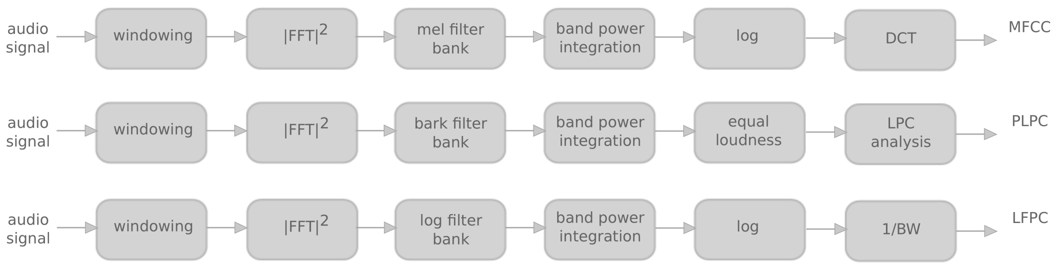


FIGURE 4.2: Schematic block diagrams for MFCC, PLPC and LFPC computation.

the band divided by the band bandwidth and expressed in decibels [26], as follows

$$\mathrm{LFPC}_m = 10\ \log_{10}\frac{S_m}{N_m}, \quad S_m = \sum_{k=f_{m-1}}^{f_m} |X(k)|^2, \quad m = 1, 2, \dots, 12$$

where m corresponds to the band number, S_m is the power of the band, N_m is the number of spectral components in the band, $X(k)$ is the k^{th} spectral component of the signal frame, and f_m are the indexes of the band boundaries corresponding to frequencies spaced logarithmically from 200Hz to 16kHz as represented in Figure 4.3. The evolution of LPFC features for a music audio example is shown in Figure 4.1.

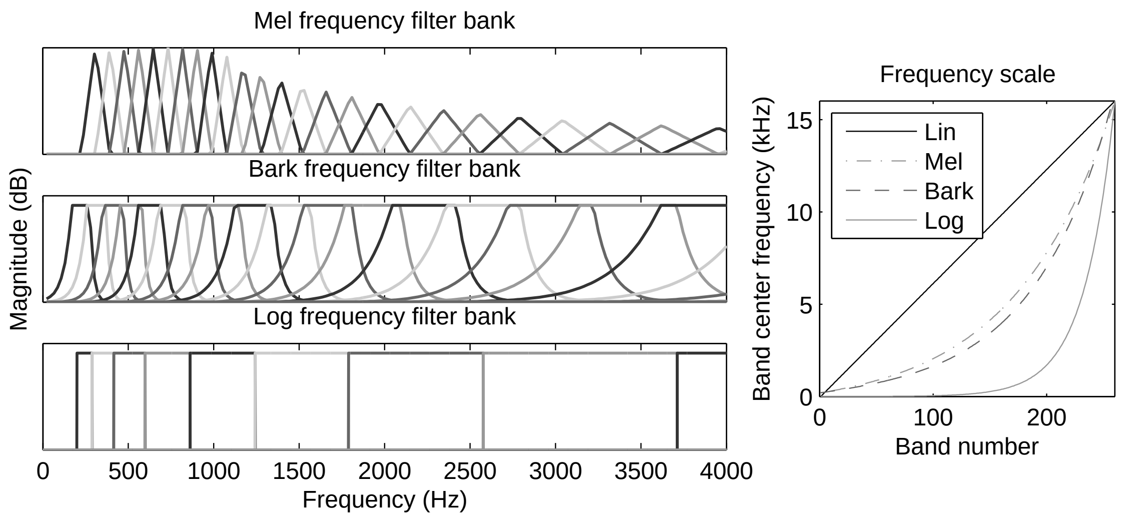


FIGURE 4.3: Frequency scales and filterbanks for MFCC, PLPC and LFPC features.

4.2.4 Harmonic Coefficient

Implementation of HC follows the procedure described in [33], where the temporal and spectral autocorrelation of the signal frame are computed (TA and SA respectively), and the HC is obtained as the maximum of the sum of the autocorrelation functions, $\mathrm{HC} = \max_\tau [\,\mathrm{TA}(\tau) + \mathrm{SA}(\tau)\,]$, where τ is the temporal delay. Temporal and spectral autocorrelation functions are calculated as

$$\mathrm{TA}(\tau) = \frac{\sum_{n=1}^{N-\tau} \bar{x}_t(n)\ \bar{x}_t(n+\tau)}{\sqrt{\sum_{n=1}^{N-\tau} \bar{x}_t^2(n) \sum_{n=1}^{N-\tau} \bar{x}_t^2(n+\tau)}},\ \mathrm{SA}(\tau) = \frac{\sum_{k=1}^{\frac{M}{2}-k_\tau} \bar{X}_t(k)\ \bar{X}_t(k+k_\tau)}{\sqrt{\sum_{k=1}^{\frac{M}{2}-k_\tau} \bar{X}_t^2(k) \sum_{k=1}^{\frac{M}{2}-k_\tau} \bar{X}_t^2(k+k_\tau)}}$$

where $x_t(n)$ is a signal frame of N samples, $X_t(k)$ is the magnitude spectrum of the signal frame computed by an M point FFT, $\bar{x}_t(n)$ and $\bar{X}_t(k)$ are the zero mean versions of $x_t(n)$ and $X_t(k)$, and $k_\tau = \frac{M}{\tau f_s}$ is the frequency bin index that corresponds to the time delay τ. Figure 4.4 shows an example of the behaviour of the HC feature for a synthetic audio signal whose degree of harmonicity varies over time. Although the computed value shows a good correlation with the harmonicity of the spectrum, in a real polyphonic music sound it becomes too noisy. Thus, it proved to be of little help for singing voice detection as reported in our work [45], even when combined with other features.

4.2.5 Spectral descriptors set

Spectral Flux, Roll-off, Centroid, Skewness, Kurtosis and Flatness are implemented based on [12]. The Spectral Flux (SFX) is a measure of local spectral change, and it is computed as the spectral difference of two consecutive frames,

$$\mathrm{SFX}_t = \sum_{k=1}^{\frac{M}{2}} \left(\hat{X}_t(k) - \hat{X}_{t-1}(k)\right)^2$$

where $\hat{X}_t(k)$ is the energy normalized magnitude spectrum of the signal frame. Spectral Roll-off is computed as the frequency index R below which the majority of the spectral energy is concentrated ($\gamma = 0.85$ is used),

$$\sum_{k=1}^{R} X_t(k)^2 \leq \gamma \sum_{k=1}^{\frac{M}{2}} X_t(k)^2.$$

The rest of the spectral descriptors consider the spectrum as a distribution, whose values are the frequencies and whose probabilities are the normalized spectral amplitudes, and compute measures of the distribution shape. The Spectral Centroid is the barycenter or center of gravity of the spectrum and is computed as,

$$\mathrm{SC} = \frac{\sum_{k=1}^{\frac{M}{2}} k\ X_t(k)}{\sum_{k=0}^{\frac{M}{2}-1} X_t(k)}.$$

The behaviour of this feature is depicted in Figure 4.4 for a test signal. The Skewness is a measure of the asymmetry of a distribution around its mean, while the Kurtosis is a measure of the flatness of a distribution around its mean. They are computed as the

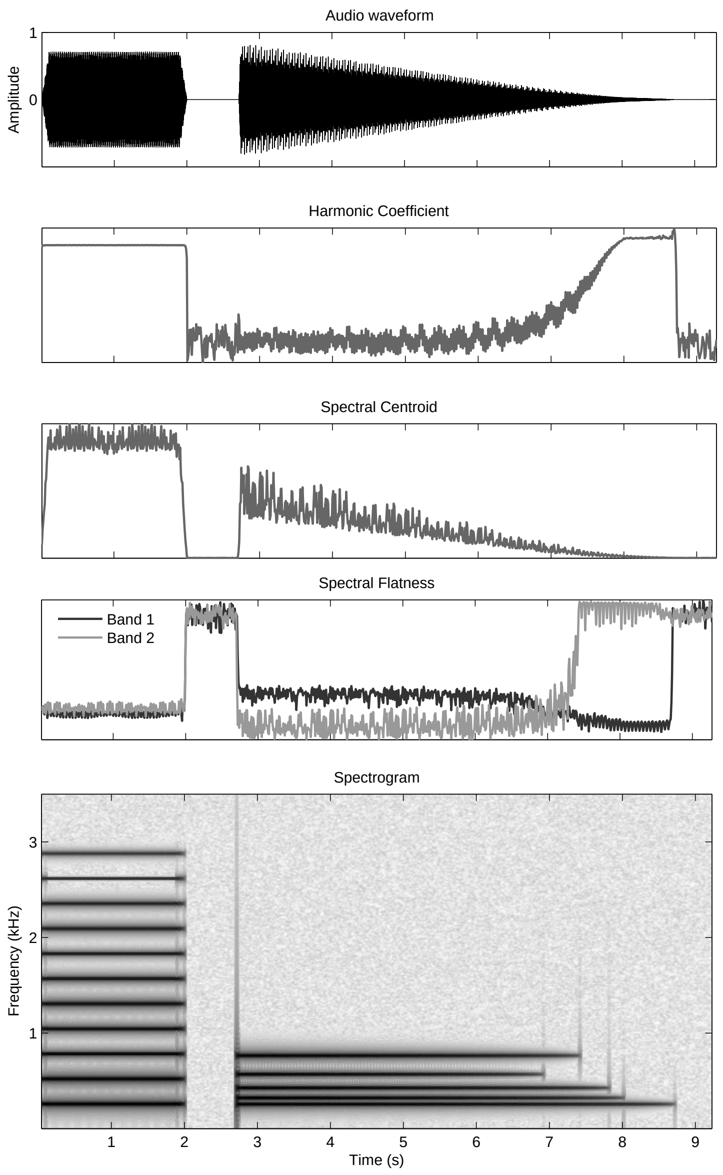


FIGURE 4.4: Synthetic test signal comprising an harmonic stationary sound followed by an inharmonic bell-like sound in which partials gradually vanish and evolves into a single sinusoidal component. Harmonic Coefficient, Spectral Centroid and Spectral Flatness features are depicted along with the test signal to illustrate their behaviour.

normalized moments of order 3 and 4 respectively by,

$$\mathrm{SS} = \frac{\sqrt{\frac{M}{2}}\sum_{k=1}^{\frac{M}{2}}(X_t(k)-\tilde{X}_t)^3}{\left(\sum_{k=1}^{\frac{M}{2}}(X_t(k)-\tilde{X}_t)^2\right)^{\frac{3}{2}}}, \quad \mathrm{SK} = \frac{\frac{M}{2}\sum_{k=1}^{\frac{M}{2}}(X_t(k)-\tilde{X}_t)^4}{\left(\sum_{k=1}^{\frac{M}{2}}(X_t(k)-\tilde{X}_t)^2\right)^{2}} - 3.$$

where $\tilde{X}_t$ is the mean value of the magnitude spectrum $X_t(k)$. The Spectral Flatness is a measure of how flat or similar to white noise the spectrum is. It is computed as the ratio of the geometric mean to the arithmetic mean of the spectrum,

$$\mathrm{SF} = \frac{\sqrt[\frac{M}{2}]{\prod_{k=1}^{\frac{M}{2}} X_t(k)}}{\frac{1}{\frac{M}{2}} \sum_{k=1}^{\frac{M}{2}} X_t(k)}.$$

A low value indicates a tonal signal, while for a noisy signal the value is close to 1. It is typically computed for several frequency bands. We used the following four overlapped frequency bands: 200-1000, 800-2500, 2000-3500 and 2500-5000 Hz. The first two bands are depicted in Figure 4.4 for an audio test signal.

4.2.6 Pitch estimation

Reliable pitch estimation in polyphonic music signals remains up to the moment a very challenging problem, so for the sake of simplicity a classical technique for fundamental frequency estimation in monophonic audio signals is applied [55].[1] This has the drawback of an unreliable estimation, prone to octave errors and noisy due to the many sounds present at the same time.

The implemented algorithm uses the Difference Function (DF), a variation of the auto-correlation function that calculates the difference between the signal frame and a delayed version of it,

$$DF(\tau) = \sum_{n=1}^{N-\tau} \left(x(n) - x(n+\tau)\right)^2 .$$

For a periodic signal this function is zero at values of τ multiples of the signal period, while in case of quasi periodic signal it has minimums close to zero. The pitch of the signal is estimated as the inverse of the delay value of the first minimum.

4.2.7 Other explored features

Apart from studying the features reported in the literature, some new sources of information to derive audio descriptors were also explored in this work. Unfortunately, none of them yielded promising results in our preliminary simulations and were discarded in further stages of the system development. Anyway, they are briefly described in the following, including some arguments on why they were abandoned.

In order to provide new clues for singing voice detection efforts were devoted to develop descriptors that could capture vocal sound characteristics. An spectral feature that characterizes voice signals is the presence of formants, resonances of the vocal tract that correspond to peaks of the spectral envelope. Note however, that other musical instruments also exhibit formants, though they are not time-varying as in the human voice.

[1] Pitch estimation in polyphonic music signals is tackled within this thesis work in Chapter 8.

With regards to pitch, vibrato is a very distinctive feature of singing voice compared to speech [19], being a periodic fluctuation of pitch at a frequency of 3 to 10 Hz. However, it is an expressive device that is used only in some singing styles and not present in all sung utterances, apart from the fact that many other musical instruments can produce it. Besides, singing voice is more than 90% voiced (where as speech is only approximately 60% voiced) [15], since to sing a melody line with lyrics it is usually necessary to stretch voiced sounds and shrink unvoiced sounds to match notes durations. For this reason, a high the degree of audio waveform periodicity was regarded as a possible indication of vocal sounds.

Apart from that, based on the observation that in an stereo recording the leading voice is generally located at the middle of the stereo image, panning (i.e. azimuthal position) information was considered as another clue for singing voice detection. Finally, trying to further exploit spectral information, relations between spectral bands were taken into account. To describe spectral energy distribution Band Energy Ratios coefficients (BERC) were considered, which were previously used in music classification [56] and auditory scene recognition [57]. Motivated by the idea that the amount of energy in a given frequency band could be correlated to the energy in other band, as it happens with formant locations for a given phoneme, Band Loudness Intercorrelations (BLI) were also computed.

By means of the same pitch estimation algorithm described in section 4.2.6, another feature called Voicing was derived, that indicates how periodic or voiced the signal in a frame is. It corresponds to the value of the minimum of the difference function used to estimate pitch [58], and proved to be a bit less noisy than the pitch estimation. To detect vibrato a simple algorithm was developed that takes a piece-wise stable pitch contour, computes the FFT of each stable fragment and looks for a prominent peak in the range of 3 to 10 Hz. Both features are based on a pitch estimation algorithm intended for monophonic sounds, therefore their usefulness was quite limited.

In order to build a formants-related descriptor the spectral envelope of an audio frame was estimated by means of a warped frequency LP analysis [16], which provides a better suited representation to resolve low frequency formants. The amplitude and frequency of the first 4 peaks of the envelope were used as acoustic features. The main drawback of this approach is that in complex sound mixtures voice formants are often obscured by the presence of other musical instruments thus resulting in meaningless features.

A description of the panning information of an audio mixture was achieved by considering the Panning coefficients proposed in [59]. An energy-weighted histogram of the stereo image is parametrized with a set of cepstral coefficients. The moderate performance obtained together with the constraint of stereo audio led us to disregard this feature.

Implementation of BER divides the signal into 24 bark bands and calculates the energy of each band. BER coefficients are obtained as the ratio of each band energy over the global signal energy. Similarly, in BLI implementation the signal is divided into 24 bark bands and power on each band is obtained, followed by equal loudness weighting and intensity to loudness conversion. Considering an audio fragment composed of several frames, a vector of loudness coefficients is obtained for each bark band. Correlation coefficients for each pair of band loudness coefficient vectors are computed. BLI coefficients correspond to the correlation coefficients weighted by the contribution of the bands to the total loudness. A relatively high number of correlation coefficients is obtained (276 coefficients for 24 bark bands). Therefore, a feature selection based on Principal Components Analysis (PCA) was applied. Results provided by this reduced set were one of the best within the new tested features, though lower than the obtained for other classical and more simple descriptors such as MFCC or LFPC. In addition, no improvement on performance was obtained by combining BLI-PCA features with other kind of descriptors. For these reasons, the BLI features were not further considered.

4.3 Temporal integration

As stated previously, the audio signal is processed in overlapped short-term analysis windows, called frames, over which the signal can be considered stationary. Thus far, features are computed from frames so they describe the audio signal at each 10 ms. Many music information retrieval systems do not take into account the temporal properties of the signal over several frames [60]. Instead, they are based on the assumption that the features values in different frames are statistically independent and perform classification of individual frames ignoring the evolution of characteristics over time. This approach has been called *bag of frames*, in analogy of the *bag of words* applied in text data that considers distribution of words without preserving their organization in phrases [61]. Although being a rather predominant paradigm, it is known to be suboptimal and the existence of a *glass ceiling* of this approach in particular applications has been pointed out [10]. Therefore, some researchers have explored different alternatives to take advantage of the information carried by the temporal evolution of features. This is commonly referred as *temporal integration*, which can be defined as the process of combining several different feature observations in order to make a classification [60].

Two kinds of temporal integration of features have been identified in current research, namely *early* and *lately* integration [60]. The former is the process of combining all the short-time feature vectors into a single new feature vector that characterizes the signal at a higher time scale. Typically this is done by computing first statistical moments (i.e. mean, variance, skewness) of the features over a sequence of frames, called segment or fragment. In addition, derivatives of frame level features are used to capture information about temporal evolution. Other strategies include the approximation of temporal dynamics of features by means of autoregressive models [62], or vector quantization [63].

The later type of integration implies considering temporal information in the classification scheme. This can be done by combining successive classifier decisions, typically at a short-time scale, and smoothing them (e.g. by ARMA filtering [36]) or computing the product of the posterior probabilities of each class [27] to derive a classification at a longer time scale. There are also some approaches that handle the temporal integration directly by means of the classification scheme, such as in Hidden Markov Models [64].

In this work we follow an early integration approach and explore some simple lately integration strategies based on posterior probabilities provided by the classifier.[2] To take into account descriptors information over several consecutive frames, an audio segment of fixed length is considered, and the first statistical moments are computed: mean, median, standard deviation, skewness and kurtosis. Different segment lengths are tested, namely 0.5, 1 and 3 seconds, in order to establish the optimal temporal horizon for this early integration. Tests conducted on different classifiers and sets of features which are reported in our preliminary study [45] indicate that 1 second length is the most appropriate from the ones considered and was adopted henceforth. We also explored other ways of setting the segment length based on homogeneous spectral characteristics and on a grid derived from rhythm structure, but without obtaining a relevant performance increase.

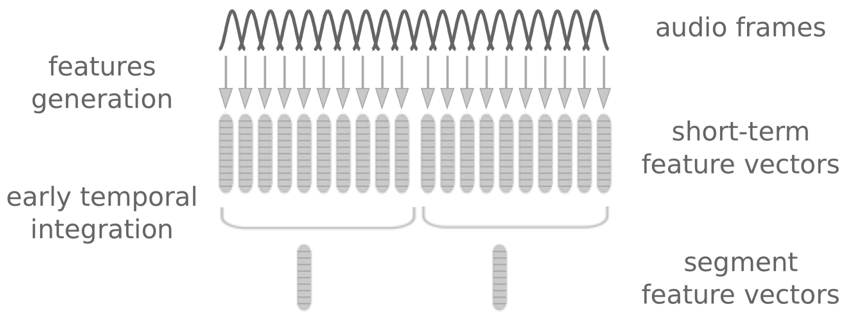


FIGURE 4.5: Schematic diagram of early temporal integration of features.

Additionally, in order to capture temporal information, deltas (i.e. derivatives) and double deltas are computed for each descriptor coefficient and the same statistical measures are calculated. The computation involves a simple approximation to a linear slope using an odd length window as follows [12, 46], where M is set to 2 for a five points window,

$$\Delta c[n] = \frac{\sum_{m=-M}^{M} m\, c[n+m]}{\sum_{m=-M}^{M} m^2}. \tag{4.1}$$

Double deltas are computed likewise, substituting $c[n]$ by $\Delta c[n]$ in the above equation.

[2]The lately temporal integration strategies are described in Chapter 6 section 6.4 and in [45].

5

Feature extraction and selection

5.1 Introduction

Building a classification system based on an extensive set of features is generally not the most appropriate approach. Some of these features may be redundant or irrelevant, or even may be not consistent with the problem. This can mislead the machine learning algorithm and is not efficient from a computational point of view. Additionally, if the number of features is very large it may become difficult to interpret the derived classification model. This is why a detailed analysis of the available features, involving selection and transformation, is an important part in the design of a classification system.

As part of this thesis work, a preliminary study was conducted on the usefulness of different kinds of features previously used in the singing voice detection problem and reported in [45]. This chapter does not reproduces that article, but deepens and extends the techniques applied for comparison, selection and combination of features. All the following experiments are done using the [TRAIN1] database (see appendix B).

5.1.1 Feature extraction and selection

In order to tackle a classification problem it may be useful to transform the original features into new ones in order to better reveal the structure of the distribution of patterns and provide the classifier with more separable classes. Applying this type of transformation is called `feature extraction` and often leads to an improvement of classification performance. In turn, using a feature extraction scheme (such as Principal Components Analysis, PCA) can also be useful to estimate the discriminating power of each of the new descriptors to facilitate the selection of features.

Feature selection aims to eliminate those useless descriptors and provide a better subset to be used for learning. There are several approaches to do this, that can be categorized in the following way [12, 42]. First of all, many learning algorithms, such as decision trees, are devised in order to determine which attributes are most appropriate to discriminate among classes. This approach, in which feature selection is part of the classification algorithm is called `embedding`. Another approach, called `filtering`, involves filtering the feature set based solely on assessments on the characteristics of the data available for learning. Therefore, the obtained group of descriptors is independent of the classification algorithm ultimately used to tackle the classification problem at hand. However, it is also possible to evaluate a subset of features by the performance obtained for the same classification scheme to be used afterwards. This approach, called `wrapping` is the most appropriate in theory but involves a high computational cost, since each performance estimation involves training the classifier and processing test data.

Regardless of which of the above mentioned feature selection approaches is applied, the evaluation of features can be done individually or in subsets [42]. In the first case the goal is to establish the discriminating power of each feature on its own. The subset of features used for classification is constructed by ranking the whole set according to their discriminating ability and selecting the first ones. This methodology is computationally efficient but has the drawback that the selected set may be redundant. In order to prevent the selected features to be correlated, it is important to evaluate them together. To do that, the attribute space is traversed by building different subsets and an assessment of each subset is performed (either by filtering or wrapping) to select the most appropriate. The number of possible subsets grows exponentially with the number of features so an exhaustive search may be too costly or impractical. For this reason non-exhaustive searching algorithms are applied that aim to achieve a feature set as closer to the optimum as possible.

5.1.2 Searching the feature space to build subsets

One of the simplest ways to search the feature space to build subsets is called `forward selection`. It starts with the empty set and the features are added one by one. Every feature that is not in the subset yet it is tentatively added and the resulting subset is evaluated. Then, only the feature that yields the best result is effectively included in the subset and the process is repeated. The search ends when none of the characteristics considered is able to overcome the result of the subset of the previous step. This guarantees to find a local optimal set, but not necessarily global. The search can also be done in reverse, i.e. to start with the full feature set and to remove them one by one, what is called `backward elimination`. In both cases it is common to introduce some additional conditions to favor the selection of small sets, such as a certain amount of performance increase. Backward elimination generally produces larger sets and obtains better performance than forward selection [42]. This is because the performance measures are only estimates and if one of them is overly optimistic it may stop the selection process early, with an insufficient number of features in the case of forward selection and too many features in the case of backward elimination. The incremental selection however is less computationally expensive and may be more appropriate to better understand the classification model at the expense of a small performance reduction.

The `Best First` algorithm is a slightly more sophisticated and effective variant of this type of selection. The search does not end when the performance assessment no longer increases, but it keeps an ordered record of previously evaluated subsets so as to revisit them when a certain number of consecutive expansions does not provide a performance improvement. The search continues from those better ranked previously considered subsets, so it is less prone to get stuck in a local optimal subset. It can operate in forward or backward mode and considers all possible subsets if not restricted in any way, for example by limiting the number of not-improving subsets to be expanded.

5.1.3 Acoustic feature sets considered

Based on the results of our preliminary study [45], two of the originally implemented features where not taken into account, namely Harmonic Coefficient (HC) and Pitch. The study confirms that HC is not able to discriminate singing voice sounds from other harmonic musical instruments so it proved to be of little help, even when combined with other kind of features. The poor performance obtained with the Pitch descriptors, due to the utilization of a monophonic fundamental frequency estimation, also led to the decision of discarding them for the classification task. In addition, based on the reported results (see section 4.3) the audio fragment length, comprising several signal frames, is set to 1 second. All the remaining kinds of descriptors described in Chapter 4 were considered for selection and combination. The total number of features for each category is presented in Table 5.1. This includes the statistical measures within the audio fragment (mean, median, standard deviation, skewness and kurtosis). For the spectral power distribution features, i.e. MFCC, PLPC, and LFPC, first and second order derivatives are also considered (equation 4.1).

TABLE 5.1: *Number of features for each category of descriptors.*

	MFCC	LFPC	SPEC	PLPC	BERC
# features	195	180	45	195	120

In the following sections, the feature selection and extraction techniques applied to the problem are described. All simulations were performed using the Weka[1] software [42].

5.2 Individual feature selection

Two different individual feature selection methods were applied, namely `information gain` and `principal components analysis`. The former makes use of the same feature selection principle typically embedded in classification trees. The latter involves a linear transformation for feature extraction. In both cases the selected subset is built considering the best ranked features. The next two sections give a brief description of each of them and present the results obtained.

Performance is estimated following a wrapping approach, as the 10-fold cross validation (CV) classification rate on the training data for a Support Vector Machine (SVM)[2]. This classifier is selected for performance estimation since it yields one of the best results in the study of classification methods presented in Chapter 6.

5.2.1 Information gain

Imagine we are given a training set of patterns for a two-class problem (A and B), and one of the available features is a binary one (0 and 1). For the feature to be helpful in the classification, the two partitions in which the classifier divides the training set should be as pure as possible regarding class labels. A measure of purity commonly used for this purpose, called `information`, has its roots in information theory. The information associated with the feature represents the expected amount of information that would be needed to specify whether a new instance should be classified as belonging to class A or B, given that the feature value is known. It is computed as the `entropy`, denoted H, of the resulting partitions and it is measured in bits.

Consider that the total number of training patterns is m and that they are divided into partitions of m_1 and m_2 elements according to the value of the binary feature, such that $m = m_1 + m_2$. Let a_0 and b_0 be the number of the m patterns of each class in the original set. Assume that they are distributed to each new partition as $m_1 = a_1 + b_1$ and $m_2 = a_2 + b_2$. The information provided by the binary feature can be computed as,

$$\mathrm{H}([a_1, b_1], [a_2, b_2]) = \frac{m_1}{m}\mathrm{H}([a_1, b_1]) + \frac{m_2}{m}\mathrm{H}([a_2, b_2])$$

[1] http://www.cs.waikato.ac.nz/ml/weka/

[2] Trained using the Sequential Minimal Optimization method (SMO). See [42].

where $H([a_i, b_i]) = -\frac{a_i}{m_i}\log(\frac{a_i}{m_i}) - \frac{b_i}{m_i}\log(\frac{b_i}{m_i})$. Thus, `information gain` is obtained by comparing this value to the entropy of the original set,

$$\Delta H = H([a, b]) - H([a_1, b_1], [a_2, b_2])$$

where $H([a, b]) = -\frac{a_0}{m_0}\log(\frac{a_0}{m_0}) - \frac{b_0}{m_0}\log(\frac{b_0}{m_0})$.

This technique must be extended to deal with a numeric attribute, instead of a binary one. The same criterion used during the formation of decision trees can be applied for this purpose, that is, recursively splitting the training set based on the attribute's values [65]. Finally, the usefulness of the attribute is assessed as the purity of the resulting partitions. A stopping criterion must be defined for the recursive splitting, since there is no point in further dividing partitions that are pure enough. This can be done by means of the `Minimum Description Length` (MDL), which aims at minimizing the amount of information needed to specify the class labels of the data given the splits, plus the information required to encode each splitting point.

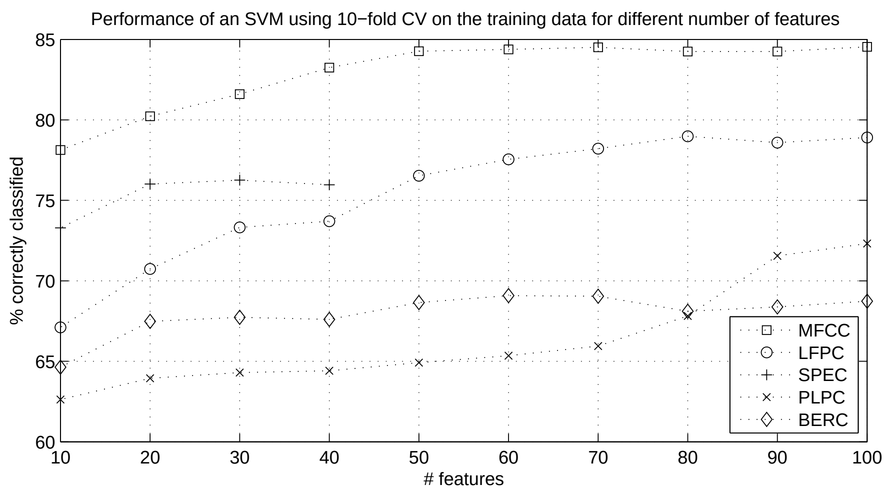


FIGURE 5.1: Percentage of correctly classified instances for a SVM classifier using 10-fold CV on the training set, while varying the number of features retained from the information gain ranking.

Results obtained for the different feature categories using the information gain selection technique are presented in Figure 5.1. Performance of the subset tends to improve as the number of features is increased. However, for some categories it is possible to reduce the number of features while maintaining or even increasing performance.

5.2.2 Principal components analysis

In principal component analysis (PCA) a new coordinate system is built to maximize the variance of the data in the direction of the new axes. The goal is to represent data in a way that best describes the variation in a sum-squared error sense. The first axis is selected in the direction of the maximum variance of the data. The following is located so it is perpendicular to the former and such that maximizes the data variance along the selected direction. This process is repeated, in each step the new axis is perpendicular to the former and maximizes the variance of the data in the axis direction.

The implementation is straightforward. The data covariance matrix is computed and its eigenvectors are obtained. These vectors indicate the directions of maximum variance and are therefore the coordinates of the new space. The eigenvalue related to each vector indicates the variance along that axis. The new coordinates can be sorted according to the percentage of the total variance they concentrate. Often, most of the variance is described by a reduced set of the new coordinates, suggesting a lower dimension subspace for the data. The selection of features involves keeping only those coordinates that account for most of the variance of the data and discarding the rest. In this way, useless features are likely to be ignored. The approach is unsupervised, which means that class labels are not considered for the analysis. For this reason, although the transformation is beneficial for filtering useless features, it does not guarantee the new coordinates to be helpful for discriminating among classes.[3]

The extraction and feature selection using principal component analysis was studied for the different categories of descriptors. Figure 5.2 shows the classification performance of an SVM using the different sets of features, while varying the percentage of cumulative variance used for selection. It also depicts the resulting number of features in each set.

It can be noticed that in order to keep the 95% of the variance, a small percentage of the total number of features is required (BER 34%, PLPC 45%, SPEC 47%, LFPC 48% and MFCC 65%). This suggest that an important part of the original features could be discarded. Considering this reduced set, performance only decreases for the LFPC and PLPC categories, 4.4 % and 2.3 % respectively. As the cumulative variance is reduced performance tends to be lower, although this behavior is not strictly monotone and presents some exceptions such as the MFCC case.

5.3 Selection of subsets of features

5.3.1 Correlation-based subset selection

To avoid redundant features in the selected subset, the correlation-based approach looks for a group of features in which its elements exhibit strong correlation with the class label, but also have low intercorrelation among them [66]. To do that, the information

[3]Linear Discriminant Analysis (LDA) is a supervised technique also based on eigenvectors that seeks a new coordinate system in which classes are better separable.

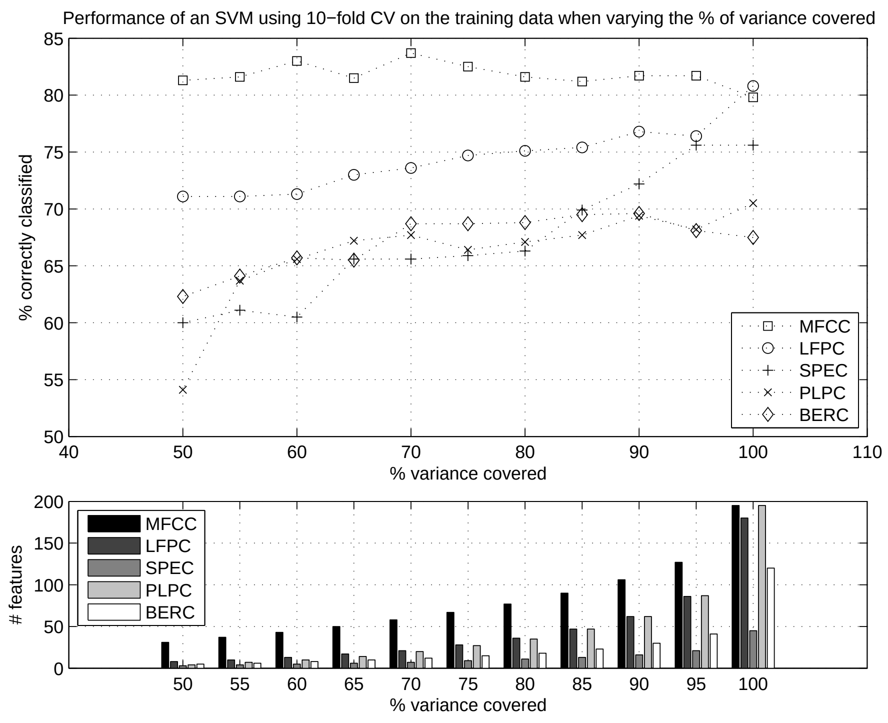


FIGURE 5.2: Percentage of correctly classified instances for an SVM using 10-fold CV on the training set, while varying the percentage of variance used for feature selection. At the bottom, the number of features for the obtained subsets is depicted.

gain or `mutual information` between two features X and Y is defined as

$$\Delta H(Y, X) = H(Y) - H(Y|X) = H(X) + H(Y) - H(X, Y).$$

From the above equation it can be noticed that mutual information is symmetrical. To compute entropy it is necessary to discretize features that take continuous values, for which a procedure similar to that described in section 5.2.1 is applied. That is, using the information gain principle for splitting the data and MDL as stopping criterion [65]. Given the above definition, mutual information has a bias towards features which take more values. To compensate for that, the `symmetric uncertainty` coefficient is defined as the normalized information gain,

$$U(X, Y) = 2\,\frac{\Delta H(X, Y)}{H(X) + H(Y)} = 2\,\frac{H(X) + H(Y) - H(X, Y)}{H(X) + H(Y)}.$$

The evaluation of a subset of features S is carried by the following heuristic measure,

$$M_S = \frac{\sum_j U(X_j, C)}{\sqrt{\sum_j \sum_i U(X_j, X_i)}} \quad \forall i, j \in S.$$

The numerator is the average correlation between the subset elements and the class label C, indicating the ability to predict the class through these features. On the other hand, the denominator is the average intercorrelation between features of the subset, which can be regarded as a measure of redundancy. Thus, sets with features strongly correlated with the class label and poorly correlated with each other are preferred.

Correlation based feature selection was applied to the problem, using the Best First algorithm for searching the feature space in forward selection and backward elimination mode. The results for the different categories of descriptors are presented in Table 5.2. It can be noticed that for most categories forward selection and backward elimination yield very similar results, indicating that there is no justification for the additional computational burden of backward elimination. Anyway for LFPC the difference is noticeable in favour of backward elimination. It is also noted that the selection of MFFC features produces a slightly greater subset for backward elimination than for forward selection, which agrees with the differences mentioned in section 5.1.2.

forward	MFCC		LFPC		SPEC		PLPC		BERC	
# features	195	40	180	42	45	13	195	31	120	16
% correct	81.6	83.4	78.9	73.9	75.6	74.5	70.9	70.9	69.5	67.0
backward	MFCC		LFPC		SPEC		PLPC		BERC	
# features	195	42	180	42	45	13	195	31	120	16
% correct	81.6	83.5	78.9	75.2	75.6	74.5	70.9	70.9	69.5	67.1

Table 5.2: Correlation based feature selection results using the Best First algorithm for searching the attribute space in forward selection and backward elimination configurations. Percentage of correctly classified instances for an SVM using 10-fold CV on the training data is reported for the original and the reduced set of features, as well as the number of features in each case.

5.3.2 Wrapping selection

The wrapping feature selection approach was applied by means of an SVM classifier using 10-fold CV on the training set to evaluate the subsets of features, and the Best First algorithm using forward selection to explore the feature space. The results are summarized in Table 5.3. The number of features and the percentage of correctly classified instances for the original and the reduced sets are reported. For most categories this selection allows for the least number of features while maintaining a very good performance, close to the maximum achieved by other approaches.

	MFCC		LFPC		SPEC		PLPC		BERC	
# features	195	11	180	24	45	9	195	11	120	12
% correct	81.6	82.9	78.9	79.3	75.6	75.8	70.9	65.3	69.5	70.1

Table 5.3: Results for the wrapping feature selection approach. Performance is assessed using 10-fold CV on the training set for an SVM classifier. The Best First algorithm is used to search the feature space. The results reported include performance and number of features for the original an reduced data sets.

5.3.3 Selection of homogeneous groups of features

Another selection procedure was carried out motivated by practical considerations concerning the inclusion of the selected features in the final system. As stated previously, the original feature sets include means, medians, standard deviations, skewness and kurtosis of the estimated coefficients and their derivatives. However, it would be interesting to relieve the system of computing whole groups of descriptors (for example, all second order derivatives) if their contribution is not very relevant. However, this is not generally ensured when applying automatic selection techniques.

For this reason, a procedure similar to backward elimination is applied, that is, starting with the full set and deleting a whole group of attributes one at a time, trying to leave out as much as possible without reducing classification performance significantly. Since the number of groups within each category is small an exhaustive search of combinations is feasible.

Results obtained by this approach are depicted in table 5.4. The MFCC set contains only the medians and standard deviations and includes the Δ coefficients (equation 4.1). The LFPC set also contains only medians and standard deviations, but includes Δ and $\Delta\Delta$ coefficients. The SPEC set comprises: mean and median of the Centroid, mean, median and standard deviation of the Roll-off, mean and kurtosis of the Skewness, mean and median of the Kurtosis, and mean, median and standard deviation of the Flatness. The PLPC set contains median and skweness and includes Δ coefficients. Finally, the BERC set comprises only standard deviations.

	MFCC		LFPC		SPEC		PLPC		BERC	
# features	195	52	180	72	45	21	195	52	120	24
% correct	81.6	85.0	78.9	79.3	75.6	76.2	70.9	71.3	69.5	69.0

Table 5.4: Results of the selection based on backward elimination of whole homogeneous groups of features. The number of features as well as 10-fold CV performance on the training data for an SVM are reported for the original and reduced sets.

As a result of this selection it is likely that some features are redundant or even irrelevant. For this reason, it is interesting to apply one of the automatic selection methods presented above as the next step. To do that, the wrapping approach is chosen since it yields the best overall selection in the previous experiments. Table 5.5 presents the results of applying the wrapping approach to the reduced sets, using 10-fold CV on the training set for an SVM classifier and the Best First search (forward and backward). It can be noticed that it is possible to further reducing the number of features while maintaining or even slightly improving performance.

forward	MFCC		LFPC		SPEC		PLPC		BERC	
# features	52	19	72	13	21	10	52	17	24	9
% correct	85.0	85.1	79.3	77.6	76.2	77.4	71.3	73.4	69.0	69.9
backward	MFCC		LFPC		SPEC		PLPC		BERC	
# features	52	38	72	54	21	15	52	29	24	11
% correct	85.0	85.2	79.3	80.3	76.2	76.7	71.3	73.0	69.0	69.8

TABLE 5.5: Wrapping feature selection on the reduced sets previously obtained. Number of features and performance estimation for the original and obtained features sets.

5.4 Feature selection results

Comparing all feature selection techniques it can be seen that the wrapping approach was one of the most effective (see section 5.3.2). In addition, the best results were obtained by removing homogeneous subgroups of features and then applying the wrapping selection approach (see section 5.3.3). This indicates that reducing the number of features improves the performance of the automatic selection algorithm. When the number of descriptors is not sufficiently small compared to the number of training patterns, it is more likely that spurious relationships that tend to mislead the selection algorithm arise.

For MFCC, SPEC, PLPC and BERC categories of Table 5.5, the sets obtained by backward elimination seem more appropriate given the limited number of features and the performance results. For the LFPC features, however, it is not clear whether the best set is the one obtained by forward selection, which involves very few features, or by backward elimination, which yields better performance (see Table 5.5), or even if it is more convenient to use the whole set obtained by the wrapping approach (see Table 5.3). The disadvantage of the latter is that it is very heterogeneous, which makes it less practical from an implementation point of view.

An alternative for better comparing the feature sets is to use a statistical test, as mentioned in Chapter 3 section 3.2. To do this, 10-fold stratified CV is repeated 10 times, thus obtaining 100 performance estimates for each set of features. Since the estimates are not independent a corrected resampled t-test is used [43]. The results of the comparison indicate that there is no statistical evidence to conclude that the smaller set is significantly worse than the others (using a significance level of 95%). Based on this result the smallest LFPC set is selected for the study on combination of descriptors presented in section 5.5.

Finally it is interesting to compare the performance of all the categories using the same statistical test, to try to determine if any of them is significantly better than the others. Figure 5.3 and Table 5.6 show the results of corrected resampled t-test for 10 repetitions of 10-fold CV to compare all categories of descriptors. With a significance level of 95%

the test indicates that there is significant statistical evidence to consider the MFCC set above all others.

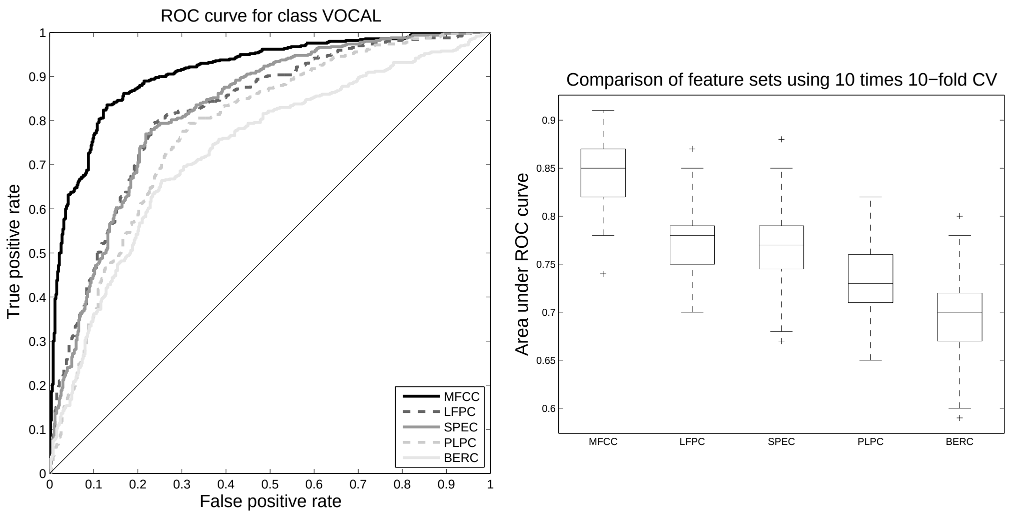


Figure 5.3: Comparison of feature sets by means of ROC curves and Box plots of the Area Under Curve (AUC). Data is obtained by 10 repetitions of 10 fold-CV, i.e. there are 100 estimations for each feature set.

	MFCC	LFPC	SPEC	PLPC	BERC
# features	19	13	10	17	9
mean % correctly classified	84.6	77.4	77.0	72.9	69.7
standard deviation	3.32	3.43	3.77	3.99	4.16

Table 5.6: Comparison of features sets by means of a corrected resampled t-test for 10 repetitions of a 10-fold CV. For a significance level of 95% the MFCC set is superior to the rest.

An interesting effect of the selection is that with a reduced set of features it becomes much easier to try to visualize the distribution of patterns in the feature space and to intuitively establish how separable the classes are. Individual assessment of features using information gain was applied to determine the most discriminative MFCC descriptors. Then graphical representations of the distribution of training patterns were built. Figure 5.4 shows one of these representations for three of the most discriminative MFCC features. While it is possible to identify regions where one class is predominant there is clearly a significant overlap. In the study of classifiers reported in Chapter 6 some results of classification using only these features suggest performance figures of about 75%.

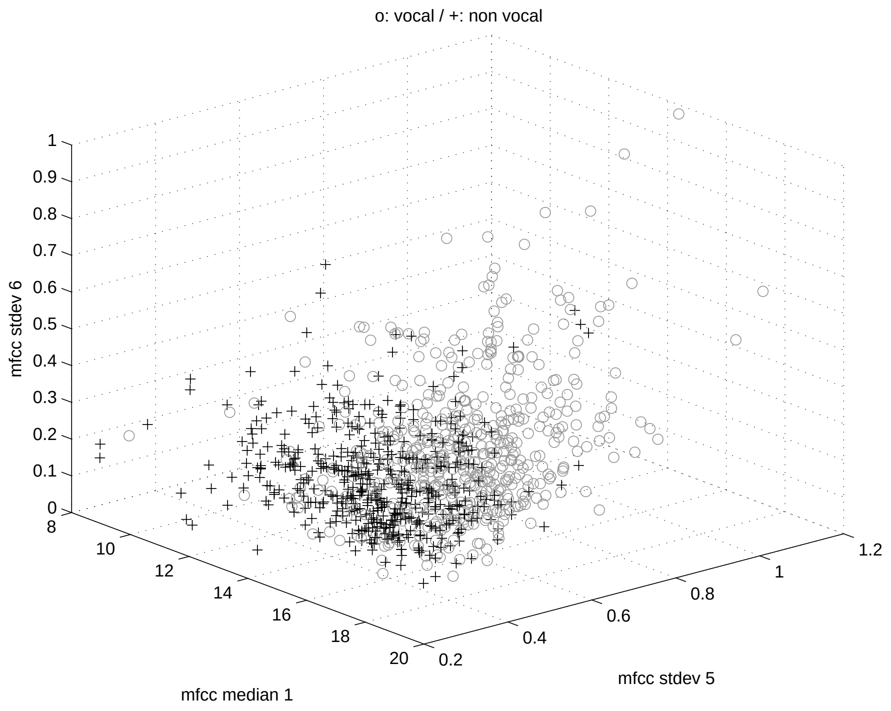


FIGURE 5.4: Distribution of the training patterns for three of the most discriminative features of the MFCC category. Significant overlap between classes can be appreciated.

5.5 Combination of feature sets

Combining different categories of descriptors (MFCC, LFPC, etc.) can in theory improve the performance of a classification algorithm because it takes into account different sources of information. As a prelude to the combination a study was conducted to identify redundancy and potential complementarity between different categories of descriptors, comparing the training patterns that are misclassified in each case. For each category an SVM classifier was built, the training patterns were classified and errors were recorded. Then we determined how many of the misclassified patterns are correctly classified by using another set of features. Table 5.7 shows the number of errors in common and not in common among different feature categories. This analysis indicates that the combination of characteristics can be helpful.

This information was also used to identify outliers in the training database. It was determined that there are 13 training patterns out of 1000 that are incorrectly classified by using any of the feature sets, i.e. errors in common to all categories. Audio fragments were aurally inspected and although some special cases were identified (e.g. low prominent voices and interfering sounds such as loud drums), it was considered that none of these cases was a true outlier, so they were not removed from the database.

New feature sets were built with combination of the available categories trying to exploit their potential benefits. Performance was estimated as percentage of correctly classified

	MFCC	LFPC	SPEC	PLPC	BERC
MFCC	148	73 / 75	80 / 68	68 / 80	67 / 81
LFPC	73 / 146	219	103 / 116	96 / 123	90 / 129
SPEC	80 / 143	103 / 120	223	97 / 126	85 / 138
PLPC	68 / 199	96 / 171	97 / 170	267	80 / 187
BERC	67 / 226	90 / 203	85 / 208	80 / 213	293

Table 5.7: Comparison of the number of errors in common and not in common (common / not-in-common) between the different feature categories on the training database [TRAIN1] which contains 1000 patterns. For instance, using the MFCC features yields 148 errors, but 75 of these patterns are correctly classified by the LFPC set (the remaining 73 are errors in common to both categories).

instances for an SVM using 10-fold CV on the training data set. Only a very few of these sets performed better than the MFCC features alone, and by a reduced margin. Table 5.8 shows the percentage of correctly classified instances and the number of features for the combinations that outperform the MFCC set.

% correctly classified	# features	categories
85.1	19	MFCC
85.8	29	MFCC SPEC
85.4	38	MFCC SPEC BERC
85.5	42	MFCC LFPC SPEC
85.6	46	MFCC SPEC PLPC
85.4	49	MFCC SPEC PLPC BERC
86.5	51	MFCC LFPC SPEC BERC
85.5	59	MFCC LFPC SPEC PLPC
86.9	68	MFCC LFPC SPEC PLPC BERC

Table 5.8: Results for the combinations of features that outperform the MFCC set alone. Performance estimation is obtained as the percentage of correctly classified instances for an SVM using 10-fold CV.

% correctly classified	σ	# features	categories
84.6	3.32	19	MFCC
85.4	3.02	29	MFCC SPEC
86.2	2.92	51	MFCC LFPC SPEC BERC
86.5	2.97	68	MFCC LFPC SPEC PLPC BERC

Table 5.9: Mean performance and standard deviation for each set obtained by repeating 10 times the 10-fold CV on the training set. Also the number of features is presented. For a significance level of 95% a corrected resampled t-test indicates there is no statistical evidence to consider one set significantly better than the others.

In order to determine whether these differences are significant, 10-fold CV was repeated 10 times and a corrected resampled t-test was applied, only for the three best performing combinations and the MFCC set. In Table 5.9, the mean value of the correctly classified percentage, the standard deviation and the number of features is presented for each set. For a significance level of 95% the test indicates there is no statistical evidence to consider one set significantly better than the others. Additionally, it is reasonable to assume that a smaller number of features favours generalisation, so the MFCC set is the one selected for the study of classifiers presented in Chapter 6.

5.6 Discussion

The process of designing the classification system included the study of feature selection and extraction techniques, for the features reported to be used in the problem. Descriptors were evaluated individually and in subgroups. Individual assessment of features provided information to identify the most discriminative ones. But these sets were not the most suitable because they may contain redundant features and could overlook fruitful feature combinations. The wrapping feature selection approach proved to be the most appropriate. In this case the assessment of a subset of features is done based on the performance of the classifier to be used in the final system. Unfortunately this approach is computationally demanding since the estimation of performance involves training and evaluating a classifier at each iteration of the cross validation.

Additionally, a method was introduced that seeks to reduce the number of features by discarding entire groups of descriptors if their contribution is not so relevant. This selection scheme produced very good results in terms of performance and number of features. Besides, a wrapping selection algorithm was applied to the reduced sets obtained by this method. Reducing the number of features compared to the number of training patterns provides a more reliable automatic feature selection which can achieve better performance. The results obtained seem to confirm this hypothesis, since performance was slightly increased for smaller feature sets (see Table 5.5).

Combination of features did not yield very encouraging results. Although the estimated performance slightly increased, there is no evidence to consider combined sets better than the MFCC category alone. This is probably because all categories of descriptors are derived from different kinds of spectral information. Combining truly different sources of information would likely be important to be able to increase performance. It would also be important to carefully study what type of information are the MFCC features capturing.

6

Classification

6.1 Classifiers

Different machine learning techniques exploit different aspects of the training data. For this reason, a technique can be very appropriate for a given problem and fail in face of another. For the purpose of designing a classifier for the problem at hand, various automatic classification algorithms were studied, their parameters were adjusted and their performance was compared. In this chapter the studied techniques and the obtained results are presented. All simulations were performed using the Weka[1] software [42] and the [TRAIN1] training database (see appendix B).

6.1.1 Decision trees

Decision trees are based on a tree structure where each node represents a decision on some attribute and each leaf corresponds to a class. Given a certain pattern to be classified, the tree structure is followed according to the decisions on each node until a leaf is reached. To construct a decision tree a feature is selected to be placed at the root and a decision associated with this feature is formulated. Then the process repeats recursively for each of the branches that arise. The principles used for the selection of features and to determine the associated decisions were discussed in Chapter 5, section 5.2.1. The idea is that each branch divides the training patterns that reach a node into

[1]http://www.cs.waikato.ac.nz/ml/weka/

sets as pure as possible in terms of the class to which they belong. This can be evaluated considering the information gain associated with each decision.

Figure 6.1 shows a decision tree for the problem of singing voice detection built with only 3 of the most discriminative MFCC features. Each node has an associated threshold on a feature and each leaf of the tree corresponds to a class. The number of training patterns of each class that reach a leaf is specified. The class label assigned to a leaf can be set based on the majority. An advantage of this classification scheme is that the obtained model can be easily understood and provides clues to the most discriminative features and values that distinguish between different classes. A restriction of this kind of trees, based on a binary decision over a single feature, is that class boundaries derived from a decision are necessarily parallel to the axes of the feature space.

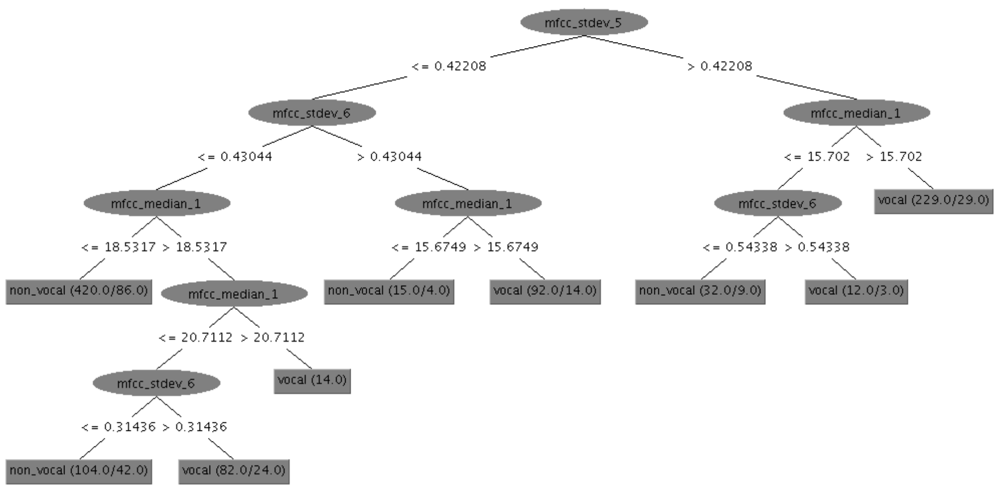


Figure 6.1: Decision tree obtained using only the three most discriminative MFCC features. Class label for each leaf as well as the number of training patterns from each class involved are depicted. It is easy to verify that the classification rate on the training set is 78.9% (211 errors out of 1000 patterns). However, performance estimation using 10-fold CV on the training set is 73.1%.

Decision trees that are constructed in this way are usually overfitted to the training data, so pruning techniques are used to favor generalization. An alternative to perform the pruning is to build the entire tree and then remove some of the branches, which is known as `postpruning` or `backward pruning`. Another alternative is to try to decide during the construction process when to stop the generation of branches, which is called `prepruning`. The latter strategy is attractive from a computational point of view. However, most tree learning algorithms adopt the backward pruning, as it is very difficult to predict whether two attributes that individually do not have great discriminative capacity, work very well when combined.

An implementation of the popular C4.5 algorithm was adopted for the experiments. In this algorithm two postpruning methods are utilized, namely `subtree replacement` and `subtree raising`. The former replaces a subtree by a single leaf, while in the latter a subtree is replaced by a subtree of a lower level. After comparing error estimates for a subtree and its replacement, it could decided whether it is appropriate or not to do the pruning. Ideally, the error estimation should be done with an independent data set that has not been used for training. To hold out a portion of the data for this testing has the drawback that the obtained tree is trained using fewer data than available. On the other hand, using the training data to do this would lead to no pruning at all.

The C4.5 algorithm uses a heuristic based on the training data to estimate the error rate. The idea is to consider that each node is replaced by the class of the majority of the N patterns that reach the node and to count the number of errors E. Assuming that these patterns are generated by a Bernoulli process of parameter p and q (the probability of each class), $f = E/N$ is the observed error rate while q corresponds to the real error rate (probability of the minority class in the node). Given a certain confidence level, the empirical error rate can be used to estimate a confidence interval for the real error rate. Since the empirical error rate is obtained from the training data, the upper limit of the confidence interval is considered as a pessimistic estimate of the real error rate. Therefore, for a given confidence level c the error estimation e is obtained as [42]

$$\mathrm{P}\left[\frac{E/N - q}{\sqrt{q(1-q)/N}} > z\right] = c, \quad e = \frac{f + z^2/(2N) + z\sqrt{f/N - f^2/N + z^2/(4N^2)}}{1 + z^2/N}.$$

The performance of the C4.5 algorithm was studied by varying its parameters, for the MFCC feature set. One of the parameters is the confidence level c used in pruning. The starting value recommended by the authors is $c = 25\%$, but it is interesting to reduce it to produce a more drastic pruning and check if it increases generalization ability. The other important parameter is the minimum number m of training patterns in each branch for a node not being removed. The default value is $m = 2$, but it should be increased if the data is noisy [42].

An exhaustve grid search was conducted for different values of both parameters, using as performance indicator the classification rate estimated by 10-fold cross validation. For the confidence level c, the search range was set from 1% to 50% with a step of 1%, while the value of m is varied between 1 and 5. Table 6.1 shows the results for three different configurations corresponding to no-pruning, default parameters and best achieved values. Differences in performance are of little significance, since they correspond for instance to a variation of 5 patterns out of 1000 when comparing no-pruning with the best configuration.

	no pruning	$c = 0.25$, $m = 2$	$c = 0.17$, $m = 4$
% classification rate	75.4	75.6	75.9

Table 6.1: Comparative performance indicators for different configurations of the C4.5 algorithm using the MFCC feature set.

6.1.2 K-nearest neighbour (k-NN)

In the method known as `nearest neighbour rule`, each new pattern is compared to the available training patterns using a distance metric and it is assigned the closest pattern class. This simple rule tends to work relatively well and can easily represent arbitrary nonlinear decision boundaries (unlike the decision trees). No explicit model is derived from the training data but the data itself. However, there are several practical problems in this approach. One of them is that the nearest neighbour search is slow for large datasets, unless an efficient data structure is used (such as *kD-tree* [42]). Another drawback is that noisy or unreliable training patterns produce noticeable performance decrease, since they lead to systematic errors. A common way to tackle this problem is to consider the k nearest neighbours and assign the majority class. Intuitively, the noisier the training patterns the greater should be the value of k. To determine it cross-validation is usually applied, which is computationally expensive but effective.

The distance metric can incorporate knowledge about the problem, i.e. to establish what it means to be near or far in terms of available features. It is not always easy to draw such a distance and, in their absence, the Euclidean distance is often used, or other distances that involve powers other than two (e.g. Manhattan). Higher powers increase the influence of large distances over small ones. One problem with this approach is that all features have the same influence on the distance calculation. However, in most problems some characteristics are more important than others and there are even irrelevant characteristics. This results in a classification approach very sensitive to noisy features. One may however modify the distance calculation by adding weights to each characteristic as a way of regulating its contribution. The idea is that these weights can be learned from the training data. Consider two patterns x and y, in an n-dimensional space of features. In the case of the Euclidean distance, weights $w_1, w_2, \dots w_n$ are introduced in each dimension,

$$d_w = \sqrt{w_1(x_1 - y_1)^2 + w_2(x_2 - y_2)^2 + \cdots + w_n(x_n - y_n)^2}.$$

To learn the weight values, an update rule can be applied for each training pattern that is classified. If the distance between the pattern to be classified and the nearest neighbour is considered, the difference $d = |x_i - y_i|$ is a measure of the contribution of each individual attribute to the decision. If this difference is small, its contribution is positive, while if it is large the contribution is negative. The update rule can then be based on this difference and whether the classification is correct or not. If the classification is correct weights increase, being decreased otherwise. The magnitude of the increase is determined from the difference of each feature, being higher if the difference is small and smaller if the difference is large. After updating the weights are usually normalized.

The technique of nearest neighbours using the Euclidean distance was applied for the MFCC feature set. Two different ways of learning weights are also considered. In one case, the magnitude of the increase is $1 - |x_i - y_i|$, and in the other case it is $1/|x_i - y_i|$. The optimal value for the number k of nearest neighbours was obtained by cross validation. Figure 6.2 shows the classification rate estimated by 10 repetitions of 10-fold CV for different distances by varying k. It can be noted that the best performance is obtained for values of k below 10 and that in these cases the use of distance weights is advantageous though marginal. The best performance estimate value obtained is 80.6% and corresponds to $k = 3$, for both updating methods.

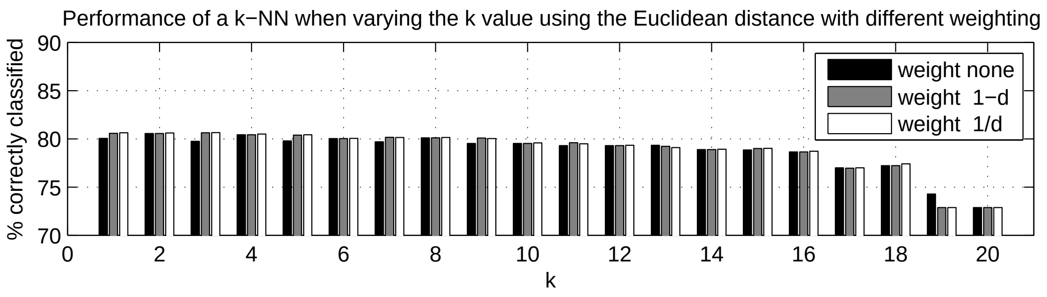


FIGURE 6.2: Classification rate of the MFCC feature set obtained by 10 repetitions of 10-fold CV by varying the number of neighbours k. Results are reported for the Euclidean distance without weighting and when learning weights by an updating factor of $1 - |x_i - y_i|$ and $1/|x_i - y_i|$.

It is interesting to note that when using a less refined feature set, the performance rate falls abruptly. For example, for the MFCC feature set of homogeneous groups (see section 5.3.3), performance rate is 76.8 % and 76.1 % for $k = 3$ and $k = 4$ respectively (no weighting). This shows how noisy features have an adverse effect on performance. Moreover, in the extreme cases where only 3 of the most discriminative MFCC features are used, performance is 70.4% and 70.7% for $k = 3$ and $k = 4$ respectively (no weighting).

6.1.3 Artificial neural networks

An `artificial neural network` (ANN) is constructed by interconnecting nonlinear processing units called neurons or perceptrons. The perceptron, which can be considered the simplest type of neural network, is a linear combination of entries followed by a nonlinear unit which produces the output. With this scheme the obtained decision boundaries are hyperplanes, thus allowing properly classifying linearly separable problems. However, there are many problems where a linear discriminant is not enough. Fortunately, simple algorithms are available that by combining several nonlinear units in a hierarchical structure can learn nonlinear decision boundaries from the training data.

The `multilayer perceptron` (MLP) is a very popular type of neural network that consists of a layer of input nodes, one or more inner layers of perceptrons and an outer layer which generates the output. There is no feedback between layers so that this type of network is called `feed-forward`, i.e. the input is propagated in one direction to generate the output. Usually each element of a layer is connected with each of the elements of the next layer, but the network may be partially connected. Figure 6.3 shows the diagram of a multilayer perceptron built with only three of the most discriminative MFCC features. The number of input nodes is determined by the dimension of the feature space, while the number of nodes in the outer layer corresponds to the desired dimension for the output.[2] Therefore, the design of a multilayer perceptron involves the determination of the number of hidden layers, the number of neurons in the hidden layers and the non-linear function to be used. The weights of the linear combination of inputs to each neuron are determined in the training process.

The number of layers and hidden neurons defines the network topology. It is important to note that in a network of this type, the output layer implements linear discriminants but on a space in which entries were mapped in a nonlinear fashion. The expressive power of the network is determined by the capacity of the hidden units to map the entries so that the classes are linearly separable for the output layer. An important theoretical result in this respect is the `universal approximation theorem`, which states that a multilayer perceptron with one hidden layer is sufficient to approximate any continuous function between the inputs and outputs [67, 68]. This ensures that there is a solution but does not indicate the number of internal neurons or whether a single hidden layer is optimal from a practical point of view. Therefore, the topology of the network is set heuristically, in some cases incorporating knowledge about the problem. While the process of optimizing the weights may be more manageable with two or more hidden layers [67], it has been found empirically that these networks are more likely to fall into a local minimum [68]. If the problem has no specific reasons to use multiple hidden layers, often a single hidden layer is used and the number of internal neurons is determined by maximizing an estimate of the classification rate [68].

The standard method for training is called `backpropagation`, which is based on the `gradient descent` technique and the algorithm is an extension of LMS. For each neuron i it is necessary to establish the coefficient w_{ij} for each entry j and an independent term b_i. The total number of weights is given by the dimension of the observation space n_O, the number of classes n_C, the number of inner layers and the number of hidden neurons n_H. In the case of a network of only one inner layer the number of weights N_w is,

$$N_w = n_O\, n_H + n_C\, n_H + n_H + n_C.$$

[2]It is common to assign an output neuron for each class, although a problem of two classes could use a single output neuron and assign the ends of the range of output values to each class.

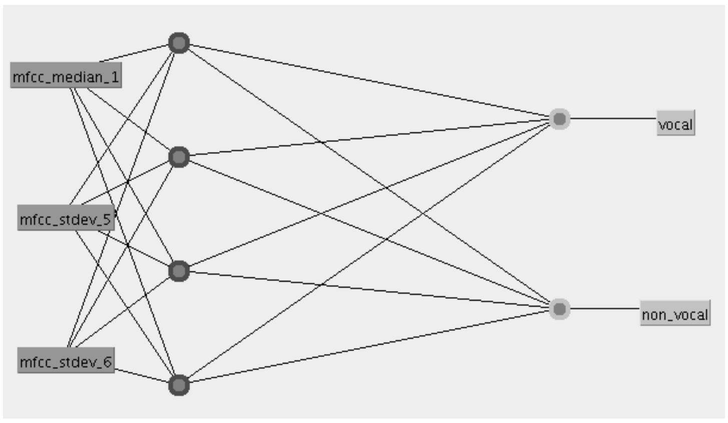


FIGURE 6.3: Multilayer perceptron built on only three of the most discriminative MFCC features. The number of input nodes is the number of features, and there is an output neuron for each class. The number of internal neurons and the number of hidden layers was set arbitrarily. In this network 26 coefficients have to be determined. The classification rate on the training set is 77.0% and using 10-fold CV is 76.4%.

A multilayer perceptron with a single hidden layer was designed and trained by back-propagation. The first step was to set the optimal number of hidden neurons. To do this, we compared the performance of a fully trained network (number of epochs $n_e = 500$) by varying the number of internal neurons. A usual rule of thumb is to limit the number of weights to one tenth of the training patterns available, which provides a bound for the number of internal neurons. In this case the number of training patterns is 1000 so $N_w^o = 100$, the number of classes is $n_C = 2$ and the number of observations is $n_O = 19$ for the MFCC feature set. Therefore, the maximum number of hidden neurons would be $n_H^o = 4$. Taking this into account the number of hidden neurons was varied between 1 and 10. Figure 6.4 shows the performance rate estimated by 10 repetitions of 10-fold CV when varying the number of internal neurons. The best performance (83.6%) is obtained for a single hidden neuron. This may be somewhat unexpected given that the number of hidden neurons determines the capacity of the network to nonlinearly map the entries. However, by increasing the number of internal neurons it also increases the ability of the network to overfitting the training data and thus reducing performance due to lack of generalization. Interestingly, a corrected resampled t-test states that at $n_H = 5$ the network performance is significantly lower than for $n_H = 1$, which seems to be in accordance with the estimated number of maximum number of hidden neurons.

After selecting the network topology the performance of the multilayer perceptron was assessed when varying the number of epochs to try to determine the optimum value. Figure 6.5 shows the performance rate obtained by 10 repetitions of 10-fold CV when varying the number of epochs n_e from 1 to 100, for a single internal neuron ($n_H = 1$). For comparison purposes results are also included for two and three internal neurons ($n_H = 2$ and $n_H = 3$). Maximum performance (83.6%) is achieved for the multilayer perceptron with only one internal neuron and a number of epochs $n_e = 60$.

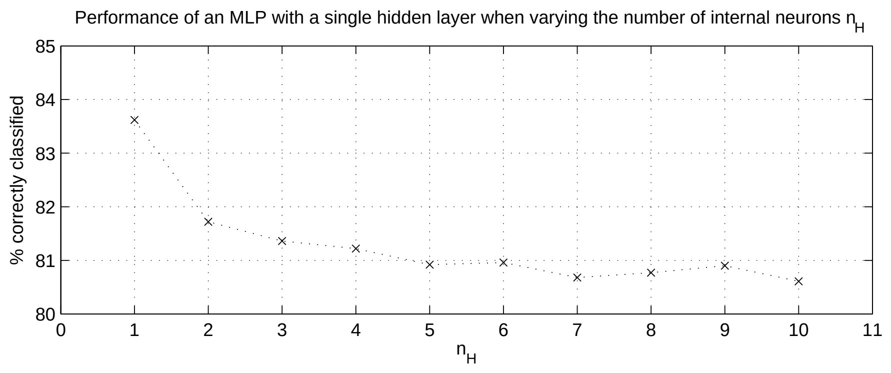


FIGURE 6.4: Performance rate obtained by 10 times repetition of 10-fold CV on the training data when varying the number of hidden neurons of a MLP with a single inner layer. These results correspond to a number of epochs of $n_e = 500$ (similar results are obtained for $n_e = 1000$).

It can be seen from these results that performance is reduced for the networks with two and three internal neurons when increasing the number of epochs, which indicates they are overfitting the training data. However, the network of a single neuron does not seem to suffer from this effect and performance does not drop significantly. This seems to indicate the features are not able to correctly separate the classes so overfitting has to be avoided by restricting the expressive power of the network to a minimum.

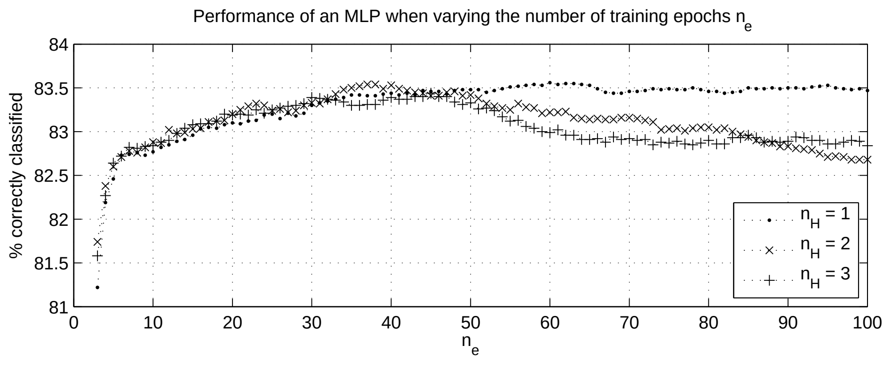


FIGURE 6.5: Performance of an MLP network with a single inner layer when varying the number of training epochs n_e from 1 to 100 (first two values are omitted). Results are reported for networks with one, two and three hidden neurons ($n_H = 1$, 2 and 3). Performance is estimated by 10 repetitions of 10-fold CV.

6.1.4 Support vector machines

The `support vector machine` (SVM), is one of the best known examples of the so-called `kernel methods`. The idea is to represent the patterns in a high dimensional space and to use therein the dot product as a distance measure. It is intended that a problem which is not linearly separable in the original feature space could become so in the new space. The power of the approach is that the mapping can be devised such that the dot product in the high dimension space could be calculated from simple operations on the input patterns without performing the explicit mapping between both spaces (known as the *kernel trick*). This allows non-linear formulations of any algorithm that can be described in terms of dot products (e.g. kernel PCA).

A kernel can be considered as a function that given two patterns returns a real number that characterizes their similarity [69]. This means that for x and x' belonging to the set of patterns $\mathcal{X}$, the kernel k is such that,

$$k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R},\ (x, x') \rightarrow k(x, x').$$

A usual type of similarity measure is the inner product, $k(x, x') = \langle x, x' \rangle$. When the inner product between two patterns is small it means that they are far away, whereas if it is large patterns are similar. A general approach to define the similarity measure is to carry out a mapping Φ (typically non-linear) in order to represent the patterns in a new space $\mathcal{H}$ that supports an inner product, $\Phi : \mathcal{X} \rightarrow \mathcal{H},\ x \rightarrow \mathrm{x} = \Phi(x)$. Thus, the similarity measure can be defined through the inner product in $\mathcal{H}$ as,

$$k(x, x') = \langle \mathrm{x}, \mathrm{x}' \rangle = \langle \Phi(x), \Phi(x') \rangle.$$

The freedom in the choice of Φ enables to find a more appropriate representation of the patterns for a given problem and to define new measures of similarity and learning algorithms. Furthermore, by carefully selecting Φ, the inner product in $\mathcal{H}$ can be calculated without the need for explicitly computing the mapping. For instance, consider a mapping of a pattern x that comprises all possible products of its coordinates $[x]_i$, in the form,

$$\Phi : \mathcal{X} = \mathbb{R}^2 \rightarrow \mathcal{H} = \mathbb{R}^3,\ \Phi(x) = \Phi([x]_1, [x]_2) \rightarrow ([x]_1^2, [x]_2^2, \sqrt{2}[x]_1[x]_2).$$

Given such Φ, the inner product in $\mathcal{H}$ is,

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle = [x]_1^2[x']_1^2 + [x]_2^2[x']_2^2 + 2[x]_1[x]_2[x']_1[x']_2 = \langle x, x' \rangle^2.$$

In this way, to assess the similarity between x and x' in $\mathcal{H}$ we are only interested in the inner product and this can be calculated directly from the patterns in $\mathcal{X}$ such as $k(x, x') = \langle x, x' \rangle^2$. To illustrate the usefulness of such mapping is worth noting that for a two class not linearly separable problem in $\mathbb{R}^2$, in which the decision boundary is an ellipse, the mapping Φ yields a distribution of patterns in $\mathbb{R}^3$ that are linearly separable by the plane obtained transforming the original boundary.

Selecting the appropriate kernel for a particular problem is the most important issue in kernel methods. Among the most commonly used are the `Polynomial kernels`, which are of the form $k(x, x') = \langle x, x' \rangle^d$, as in the example above. This type of kernel can take into account high-order statistics of the data (by considering the products between features) avoiding the explosion in computational time and memory of other machine learning methods such as polynomial classifiers [69]. Another type of widely used kernel is the `Radial Basis Function` (RBF), which can be expressed as $k(x,x') = f(d(x,x'))$, where f is a function of some metric of distance d between two patterns. The metric typically involves the inner product as $d(x,x') = |x-x'| = \sqrt{\langle x-x', x-x'\rangle}$. A very popular kernel of this type is the one in which the function f is Gaussian, that is $k(x,x') = e^{-\gamma\langle x-x', x-x'\rangle^2}$.

The SVM classification is based on the idea of finding an hyperplane in $\mathcal{H}$ in order to build a decision function $f(\mathrm{x})$ that can distinguish between training patterns of the two distinct classes, of the form

$$\langle \mathrm{w}, \mathrm{x}\rangle + \mathrm{b} = 0 \quad \text{with} \quad \mathrm{w} \in \mathcal{H},\ b \in \mathbb{R}$$
$$f(\mathrm{x}) = \mathrm{sign}(\langle \mathrm{w}, \mathrm{x}\rangle + \mathrm{b}).$$

Assuming a problem of two linearly separable classes, of all the hyperplanes that correctly divide the training patterns, there is only one *optimal hyperplane* that has a maximum margin of separation to any training pattern. This hyperplane can be found by maximising the distance to all the training patterns $\mathrm{x}_i = \Phi(x_i),\ x_i \in \mathcal{X}$,

$$\max_{\mathrm{w}\in\mathcal{H},\ \mathrm{b}\in\mathbb{R}} \min\{|\mathrm{x}-\mathrm{x}_i| \ / \ \mathrm{x} \in \mathcal{H}, \langle \mathrm{w}, \mathrm{x}\rangle + \mathrm{b} = 0,\ i = 1, \ldots, m\}.$$

There are theoretical arguments that support the generalisation ability of this solution (see [69], section 7.2). For instance, if new test patterns are built from the training patterns by adding a certain amount of noise bounded by the margin, the classification of these new patterns will be correct. Besides, small disruptions in the hyperplane parameters (w and b) do not change the classification of training patterns.

Additionally, the problem of finding the optimal hyperplane is attractive from a computational point of view, because it can be solved as a quadratic programming problem for which there are efficient algorithms. The vector w is perpendicular to the hyperplane, and it can be seen geometrically that the margin is inversely proportional to the norm of w. The optimisation problem can be formulated as the minimization of the norm of w subject to the restriction that the training patterns are correctly classified,

$$\begin{cases} \displaystyle\min_{\mathrm{w}\in\mathcal{H},\ \mathrm{b}\in\mathbb{R}} \tau(\mathrm{w}) = \frac{1}{2}|\mathrm{w}|^2 \\ y_i(\langle \mathrm{w}, \mathrm{x}_i\rangle + b) \geq 1 \ \ \forall i = 1, \ldots, m \end{cases}$$

where y_i is 1 for the patterns of one class and -1 for the patterns of the other class. When solving this optimisation problem it can be seen that only the training patterns that meet the constraint as an equality are involved in the solution. These patterns are called `support vectors`, given that they support the decision boundary. The remaining patterns can be discarded, which coincides with the intuitive idea that the hyperplane is completely determined only by the training patterns closest to it.

The SVM classification is illustrated in Figure 6.6 for a two class linear separable problem. The optimal hyperplane which satisfies the maximum margin is depicted. The margin is inversely proportional to the norm of w. The highlighted patterns are the support vectors which define the decision boundary.

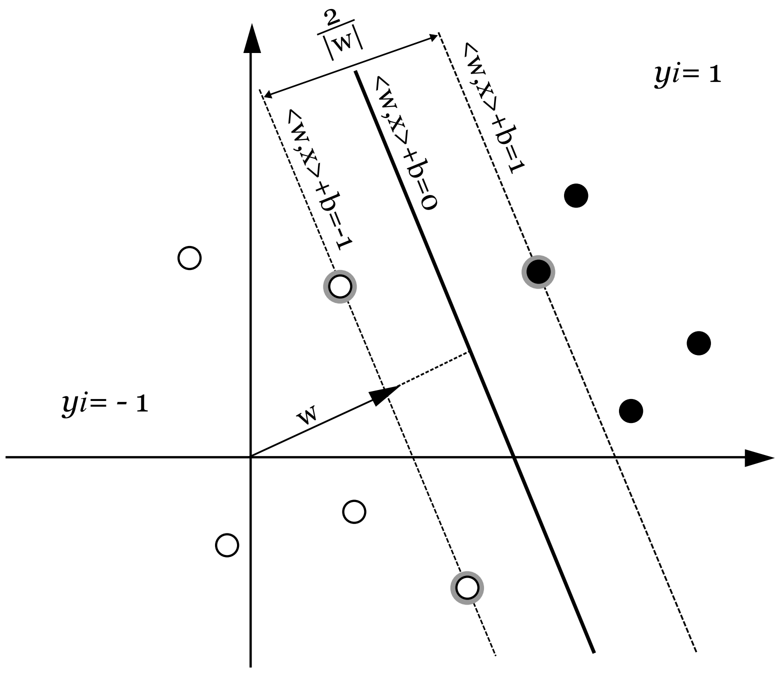


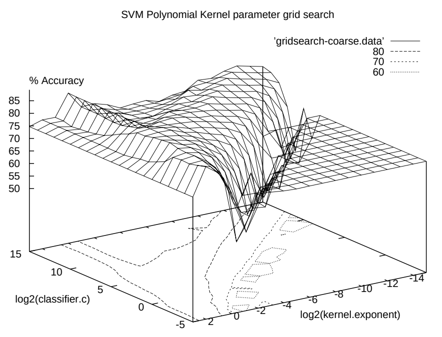
Figure 6.6: Diagram of an SVM classifier for a linear separable two class problem. The optimal hyperplane which maximises the margin and the support vectors are depicted. The vector w is perpendicular to the hyperplane.

In practice, there may not be an hyperplane able to thoroughly separate classes, for example when they overlap or there are some outliers. To allow some patterns violate the proposed solution, slack variables ξ_i are introduced for each training pattern x_i, that relax the constraints as follows

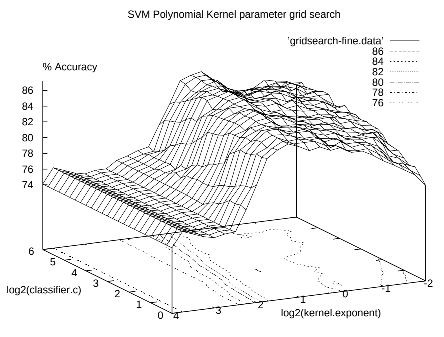
$$\begin{cases} \min\limits_{\mathrm{w}\in\mathcal{H},\ \mathrm{b}\in\mathbb{R},\ \xi\in\mathbb{R}^m} \tau(\mathrm{w},\xi) = \frac{1}{2}|\mathrm{w}|^2 + C\sum_{i=1}^{m}\xi_i \\ y_i(\langle \mathrm{w},\mathrm{x}_i\rangle + b) \geq 1-\xi_i \ \ \forall i = 1,\dots,m \\ \xi_i \geq 0 \ \ \forall i = 1,\dots,m. \end{cases}$$

By allowing the ξ_i variables to be large enough constraints can always be fulfilled, so the constant $C > 0$ is introduced, which penalises the growth of the ξ_i variables in the function to be minimised. This constant then regulates the trade-off between margin maximisation and training error minimisation.

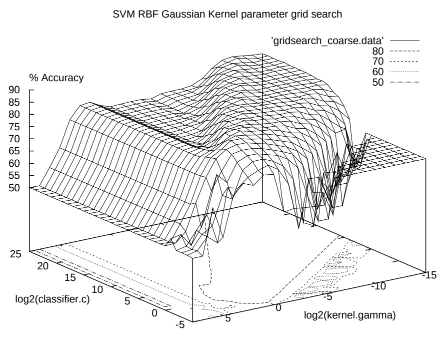
It is necessary to determine the value of the penalty factor C and the kernel parameters to fully specify the model and solve the optimization problem. This can be done by performing an exhaustive search of parameters restricted to a grid of values, choosing the best set based on the estimated cross-validation performance on the training data [70]. To reduce computational cost a coarse search can be performed at first, and then a refinement can be carried out in the area of interest.



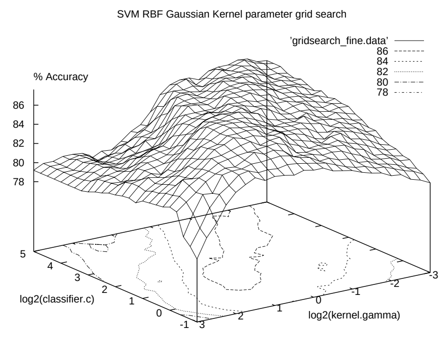
(a) SVM Polynomial Kernel - coarse grid



(b) SVM Polynomial Kernel - fine grid



(c) SVM Gaussian RBF Kernel - coarse grid



(d) SVM Gaussian RBF Kernel - fine grid

FIGURE 6.7: Grid search of the optimal parameters for an SVM classifier using Polynomial and Gaussian RBF kernels. An initial coarse grid search is performed followed by a refinement in the optimal region. Plots show a cross-validation estimate of the `Accuracy = true positives / (true positives + false positives)`. Grid parameter values are exponential as suggested in [70]. Coarse grid values for the Polynomial kernel are $C = 2^{-5}, 2^{-4}, \ldots, 2^{15}$ and $d = 2^{-15}, 2^{-14}, \ldots, 2^{3}$, and for the Gaussian RBF kernel $C = 2^{-5}, 2^{-4}, \ldots, 2^{25}$ and $\gamma = 2^{-15}, 2^{-14}, \ldots, 2^{8}$. Fine grid values for the Polynomial kernel are $C = 2^{0}, 2^{0.25}, \ldots, 2^{6}$ and $d = 2^{-2}, 2^{-1.75}, \ldots, 2^{2}$, and for the Gaussian RBF kernel are $C = 2^{-1}, 2^{-0.75}, \ldots, 2^{5}$ y $\gamma = 2^{-3}, 2^{-2.75}, \ldots, 2^{3}$. The optimal parameters for the Polynomial kernel are $C = 2^{2.5}$ and $d = 2^{1}$, and for the Gaussian RBF kernel $C = 2^{3.25}$ and $\gamma = 2^{-1}$, for a classification rate of 85.8% and 86.4% respectively estimated by 10-fold CV on the training data.

SVM classifiers were constructed for the problem of singing voice detection using Polynomial and Gaussian RBF kernels. The parameters are the factor C and exponent d in the case of Polynomial kernel, and the factor γ in the case of Gaussian RBF kernel. To determine the most appropriate pair of values in each case, an initial search on a coarse grid is performed to determine an optimum point and then the search is refined around that point. Figure 6.7 shows the performance estimate obtained by cross validation on the training data in each case. The best performance reached is 86.4% for the Gaussian RBF kernel and 85.8% for the Polynomial kernel.

6.2 Comparison of classifiers

In order to establish whether the classification schemes have significant performance differences, a corrected resampled t-test was performed for 10 repetitions of 10-fold CV on the training set. Optimal parameters for each classifier were selected based on results reported in previous sections. Table 6.2 presents the obtained results. For a significance level of 95 % the statistical test indicates that the SVM classifier is significantly superior to the other classification schemes. Figure 6.8 shows a comparison of the classifiers by means of ROC curves and box plots for the 10 repetitions of 10-fold CV.

	% correct	σ	parameters
Tree	75.4	4.88	$c = 0.17$, $m = 4$
k-NN	77.2	4.28	$k = 3$, weighting: $1 - d$
MLP	83.6	3.30	$n_H = 1$, $n_e = 60$
SVM	85.9	3.17	Gaussian RBF, $C = 2^{3.25}$, $\gamma = 2^{-1}$

Table 6.2: Performance comparison estimated by 10 repetitions of 10-fold CV on the training set using best parameters for each classifier. A corrected resampled t-test shows that for a significance level of 95% the SVM classifier is superior to the rest.

6.3 Evaluation

An evaluation was conducted to assess the performance of the system in face of new data. A validation dataset [VALID1] was constructed, independent of the training set, which consists of 30 music files manually labeled, corresponding to the albums *Abbey Road* and *A Hard Day's Night* of The Beatles.

To perform the automatic classification the audio file is divided into consecutive one-second fragments. Within each fragment descriptors are computed for 25 ms frames every 10 ms and statistical measures are obtained. Finally, each fragment is classified into vocal or non-vocal.

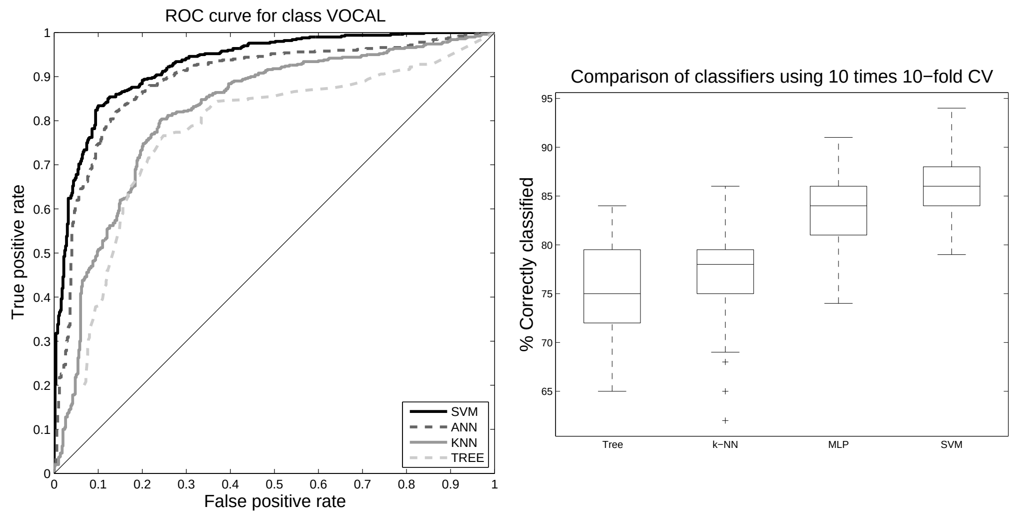


Figure 6.8: Comparison of classifiers by means of ROC curves and Box plots of the percentage of correctly classified instances. Data is obtained by 10 repetitions of 10-fold CV, i.e. there are 100 estimations for each feature set.

Then the automatic classification is compared with manual labels by calculating two performance measures. The first one is the percentage of time in which the manual and automatic classification coincide. This measure is an indicator of the performance that can be reached for a real system in the classification of music files. However, because the audio file is arbitrarily fragmented, it is possible for a fragment to lie over a transition between vocal and non-vocal, so the classification into one of the two classes does not make much sense. For this reason, the other measure of performance involves discarding those fragments containing a transition between classes and accounting for misclassified fragments. This is consistent with the training process in which each audio clip corresponds to a single class. The database contains a total of 4598 fragments of one second of which 1273 are located on a label transition, so that the evaluation is done based on a total of 3325 fragments.

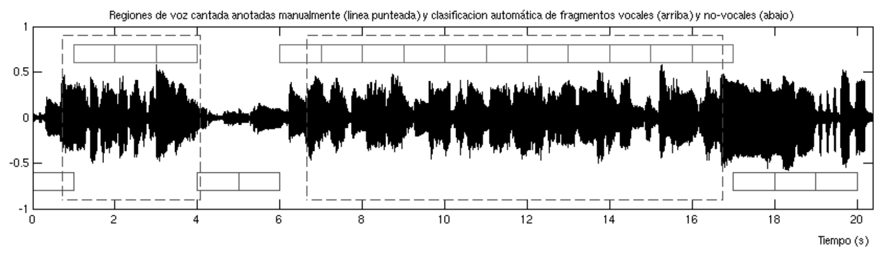


Figure 6.9: Example of automatic classification. Dotted line indicates manually labeled vocal regions. The audio excerpt is processed into one second length fragments that are classified as vocal (above) and non-vocal (below). Percentage of time in which the manual and automatic classification coincide is 93.6%. There are 4 fragments, from a total of 20, that lie over a class transition. All the remaining are correctly classified.

Figure 6.9 is an example of the automatic classification for a 20-second excerpt of one of the files in the database. When comparing the manual labeling with the automatic classification it is observed that despite the fact this is an example in which the system seems to operate correctly, the differences between boundaries of both classifications produce a performance rate of 93.6%. If fragments in which there is a transition between classes are discarded, all remaining fragments are correctly classified.

Different configurations of the classification system were evaluated on this dataset. First, all the previously studied classification techniques were applied to the MFCC feature set used along this chapter. Then other feature sets were also considered. One of them is the obtained by applying the selection of homogeneous groups of features to the MFCC category, as described in section 5.3.3, from which the above mentioned MFCC feature set is derived. Another set is the best performing combination of features obtained in section 5.5 that includes all categories of descriptors. For these two sets, the investigated classifiers were applied, an SVM with Gaussian RBF kernel and an MLP trained by backpropagation. The optimal parameters of the SVM were determined as in section 6.1.4 and the MLP was constructed with a single internal neuron and trained for 100 epochs.

classifier	# errors	% correct	% performance	# features
SVM	675	79.7	73.5	19, MFCC
MLP	649	80.5	74.5	19, MFCC
k-NN	714	78.5	73.0	19, MFCC
Tree	797	76.0	71.0	19, MFCC
SVM	560	83.2	76.6	52, MFCC
MLP	605	81.8	75.5	52, MFCC
SVM	564	83.0	76.7	68, MFCC LFPC SPEC PLPC BERC
MLP	546	83.6	77.3	68, MFCC LFPC SPEC PLPC BERC

Table 6.3: Classification results on the validation dataset for different configurations of the system. The number of errors corresponds to the number of misclassified fragments and this is also expressed as the fragment classification performance (out of 3325 single-class fragments). Classification performance is also reported as the percentage of time in which the manual and automatic classification coincide.

Firstly, it is interesting to compare the results of the evaluation for the smaller MFCC feature set, to the performance estimates obtained using 10-fold CV on the training data (see table 6.2). It is noted that the decision tree and nearest neighbour rule estimates are quite consistent with the results. However, the percentages of correctly classified fragments for the SVM classifier and that of the MLP differ considerably from the estimates based on the training data. This may be due to the ability of these classifiers to overfit the trainig data. It may also happen that the training database is not sufficiently representative of the problem. More simulations are needed to establish the causes of these differences.

Since the reduced MFCC feature set is derived from a more complete MFCC set, it is interesting to compare the results of the evaluation in both cases. In the feature selection process it was estimated, based on the training data, that an SVM classifier using a set of 19 MFCC features has comparable performance with that of the complete set of 52 MFCC features (see Table 5.5), but evaluation results show a noticeable difference in favor of the more complete set. In this case, reducing the number of features does not seem to favor the generalization ability of the classifier but leads to decreasing classification performance. On the other hand, it appears that there are no substantial differences in overall performance of the 52 MFCC features set compared to the other sets that combine the different categories of descriptors. Therefore, the MFCC feature set seems more appropriate because it is computationally less demanding. It should be noted in conclusion that the performance obtained using SVM and MLP are very similar, except for the most comprehensive MFCC set.

6.4 Discussion

Various automatic classification techniques were studied, in a search for the most appropriate parameter configuration for each of them. Using a decision tree based on the C4.5 algorithm yields a classification model that can easily be interpreted because it indicates the most discriminative features and threshold values that can distinguish between classes. However, performance is significantly lower than those of the other classification schemes studied, both in the estimation based on the training data and in the evaluation. On the other hand, in the case of the nearest neighbour technique, while it follows a simple classification approach, its performance significantly exceeds that of the decision tree and the assessment on the validation data shows the performance is not located well below the other classifiers. It was found that within the studied classification techniques, the multi-layer perceptron and the support vector machine are the most powerful. Both techniques, unlike the previous ones, allow the construction of nonlinear decision boundaries to discriminate between classes. The multilayer perceptron makes it by combining basic blocks that implement linear discriminants into a hierarchical structure and thus achieving a non-linear mapping of the input features through the hidden layers. The support vector machine uses a nonlinear mapping of the input features into a high-dimensional space and then seeks for a linear discriminant in that new space. As for the results obtained with these techniques there is no substantial evidence to prefer one over the other (although an SVM may have some theoretical advantage over MLP in terms of generalization).

Finally it is interesting to assess the performance of the developed system in the real situation of music classification. To do this the percentage of time in which the manual and automatic classification coincide can be used as a performance indicator (see table 6.3). The evaluation results indicate that there is some room for improvement.

In our study reported in [45], some post-processing strategies were considered to improve classification performance of the system based on classification confidence and contextual information. The first one is devised for a fragment that lies over a transition between vocal and non-vocal. For this reason, if a frame has a low classification confidence (based on probability estimates for each class) it is subdivided into two new fragments and each of them is re-classified. In case of a transition each new fragment could be classified to a different class (see figure 6.10). Additionally, two simple context rules are proposed. If a low probability fragment is surrounded by elements of the same class re-classification is avoided. On the other hand, if one of the half size fragments produced by re-classification is surrounded by elements of the other class, it is deleted. Although all of these post-processing proved to be advantageous (see results reported in [45]), their contribution to performance is quite marginal.

Other strategy could be to partition the audio file into uniform sections with regards to the spectral characteristics for the purpose of avoiding the classification of fragments which contain transitions between classes. It could be also interesting to synchronize audio fragments grid to the temporal music structure, using musical pulse such as tatum as a time reference. If transition between classes happen to be in accordance with metrical pulse, this could favor fragments to be homogeneous. Both of the above mentioned approaches were explored, the former using a spectral onset detection technique [71] and the latter by means of a tempo estimation algorithm [72] (using Beatroot[3] software), but none of them provided a relevant performance increase.

It seems that it is difficult to improve classification performance by attempting variations on this standard pattern classification approach. It is necessary to deepen the study of more specific descriptors able to capture distinctive features of the singing voice and to explore different approaches to address the problem.

[3] `http://www.eecs.qmul.ac.uk/~simond/beatroot/index.html`.

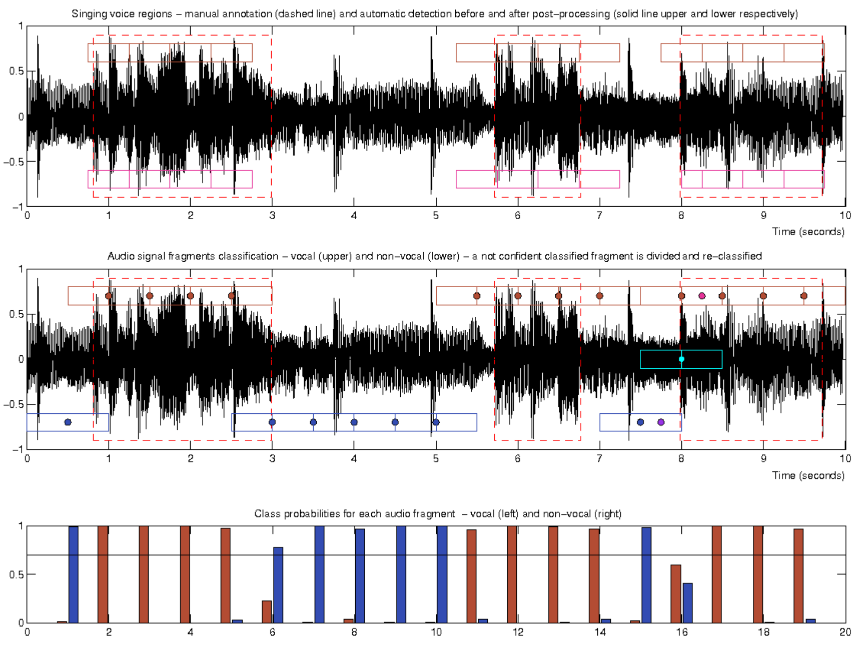


FIGURE 6.10: Post-processing based on classification confidence. Audio frames in this example are half overlapped. An audio frame with low probability estimates for each class is subdivided into two new fragments and each of them is re-classified. In this example, fragment centered on second 8 lies over a transition between non-vocal and vocal and its probability estimates fall below a threshold of 0.7. The fragment is divided and each new segment is correctly classified. Classification is improved after post-processing (from 84.2% to 86.5%) as it is shown in the manual annotation versus automatic detection comparison at the top.

Part II

Harmonic sound sources extraction and classification

7
Time frequency analysis

The alternative approach proposed in this dissertation for singing voice detection involves the separation of the harmonic sound sources from an audio mixture and their individual classification. For this purpose it is reasonable to look for a signal analysis that concentrates the energy of each component in the time-frequency plane as much as possible. In this way, the interference between sound sources would be minimized and they could be extracted from the mixture by some kind of time-frequency filtering. This chapter is devoted to describing the time-frequency analysis techniques that were applied in this work in order to improve the time-frequency representation of music audio signals with an emphasis on properly capturing singing voice pitch fluctuations.

The work described in this chapter was developed in collaboration with Pablo Cancela and Ernesto López, and was originally reported in two international conference papers [73, 74].[1] The herein description reproduces some passages of the original articles and also includes modifications and additions. This is done in order to put the work in the context of this dissertation.

[1]Part of this work was done in the context of the research project "Estudio y aplicación de técnicas de representación tiempo-frecuencia al procesamiento de audio", 2010-2011, supported by Comisión Sectorial de Investigación Científica, Universidad de la República.

7.1 Time frequency representation of music audio signals

Most real signals (for instance, music audio signals) are non-stationary by nature. Moreover, usually an important part of the information of interest has to do with the non stationarity (beginning and end of events, modulations, drifts, etc). For this reason, the development of time-frequency (TF) representations for the analysis of signals whose spectral content varies in time is an active field of research in signal processing [75]. The TF representations are commonly adapted to the signal in order to enhance significant events so as to facilitate the detection, estimation or classification. An alternative goal is to obtain a sparse representation for compression or denoising. In some cases the elements of the sparse representation become associated with salient features of the signal thus also providing feature extraction [76].

The Short Time Fourier Transform (STFT) [77] is the standard method for time-frequency analysis. This representation is appropriate under the assumption that the signal is stationary within the analysis frame. In addition, time-frequency resolution is constant in the STFT. However, for the analysis of music signals a non uniform tiling of the time-frequency plane is highly desired. Higher frequency resolution is needed in the low and mid frequencies where there is a higher density of harmonics. On the contrary, frequency modulation (typical of the singing voice rapid pitch fluctuations) calls for improved time resolution in higher frequencies. Different multi-resolution time-frequency alternatives to the STFT have been proposed such as the Constant-Q Transform (CQT) [78] or representations based on the Wavelet Transform [79].

The precise representation of frequency modulated signals, like singing voice, is a challenging problem in signal processing. Many time-frequency transforms can be applied for this purpose. The most popular quadratic time-frequency representation is the Wigner-Ville Distribution (WVD), which offers good time-frequency localization but suffers from interfering cross-terms [80]. Several alternatives were proposed to attenuate the interferences such as the Smoothed Pseudo WVD and other Cohen class distributions [80], but with the side effect of resolution loss due to the smoothing. A different approach to perform the analysis is considering the projection over frequency modulated sinusoids (chirps), in order to obtain a non-Cartesian tiling of the time-frequency plane that closely matches the pitch change rate. Among the chirp-based transforms, the Chirplet Transform [81] and the Fractional Fourier Transform [82] involve the scalar product between the signal and linear chirps (linear FM), and can reach optimal resolution for a single component linear chirp. However, many sounds present in music (e.g. voice) have a harmonic structure, and these transforms are not able to offer optimal resolution simultaneously for all the partials of a harmonic chirp (harmonically related chirps). In the case of harmonic signals, the Fan Chirp Transform (FChT) [83] is better suited as it provides optimal time-frequency localization in a “fan” geometry. The FChT can be considered as a time warping followed by a Fourier Transform, which enables an efficient implementation using the FFT. Although many of these techniques were applied to speech [84], the use of time-frequency representations other than the STFT for music analysis remains rather scarce [76, 85] and in particular the FChT to the best of our knowledge has almost not been explored for this purpose, except for very few works [86, 87].

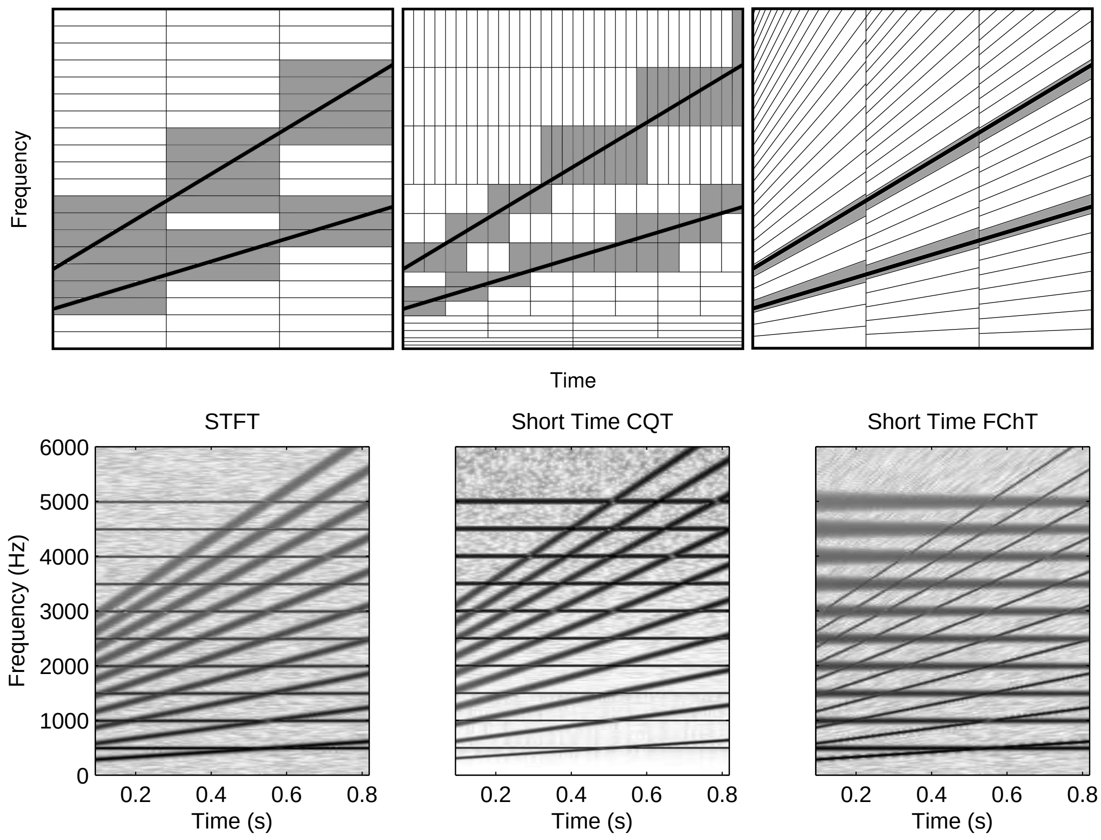


Figure 7.1: Above: Time-frequency tiling sketch for the STFT, the Short Time CQT and the Short Time FChT (from left to right). The resulting resolution for a harmonic chirp with two components is depicted. Below: Analysis of the sum of a synthetic stationary harmonic signal and a harmonic chirp, for each time-frequency representation.

In the course of this thesis work, two different time-frequency representations were studied and further developed, namely the Constant-Q Transform and the Fan Chirp Transform. Existing efficient algorithms for multi-resolution spectral analysis of music signals were reviewed [88, 89], and compared with a novel proposal based on the IIR filtering of the FFT [73]. The proposed method, apart from its simplicity, shows to be a good compromise between design flexibility and reduced computational effort. With regards to the Fan Chirp Transform, a formulation and an implementation were devised to be computationally manageable, and which enable the generalization of the FChT for the analysis of non-linear chirps [74]. Besides, the combination with a constant Q Transform was explored in order to build a multi-resolution FChT.

Figure 7.1 shows a comparison of the time-frequency tiling for the STFT, the Short-Time CQT (STCQT) and the Short-Time FChT (STFChT). The resulting resolution for a harmonic chirp with two components is depicted. The figure also shows the analysis of the sum of a synthetic stationary harmonic signal and a harmonic chirp, for each time-frequency representation. Note the improved time-frequency localization of the harmonic chirp for the STCQT and the STFChT, at the expense of a poorer resolution for the stationary harmonic signal. The STFChT is tuned to obtain optimal resolution for the harmonic chirp, as described in section 7.3. A comparison of the different time-frequency representations applied to a music audio excerpt which is used as an example through this and the following chapter is presented in Figures 7.2 and 7.3. The following sections describe the Constant Q Transform and the Fan Chirp Transform respectively.

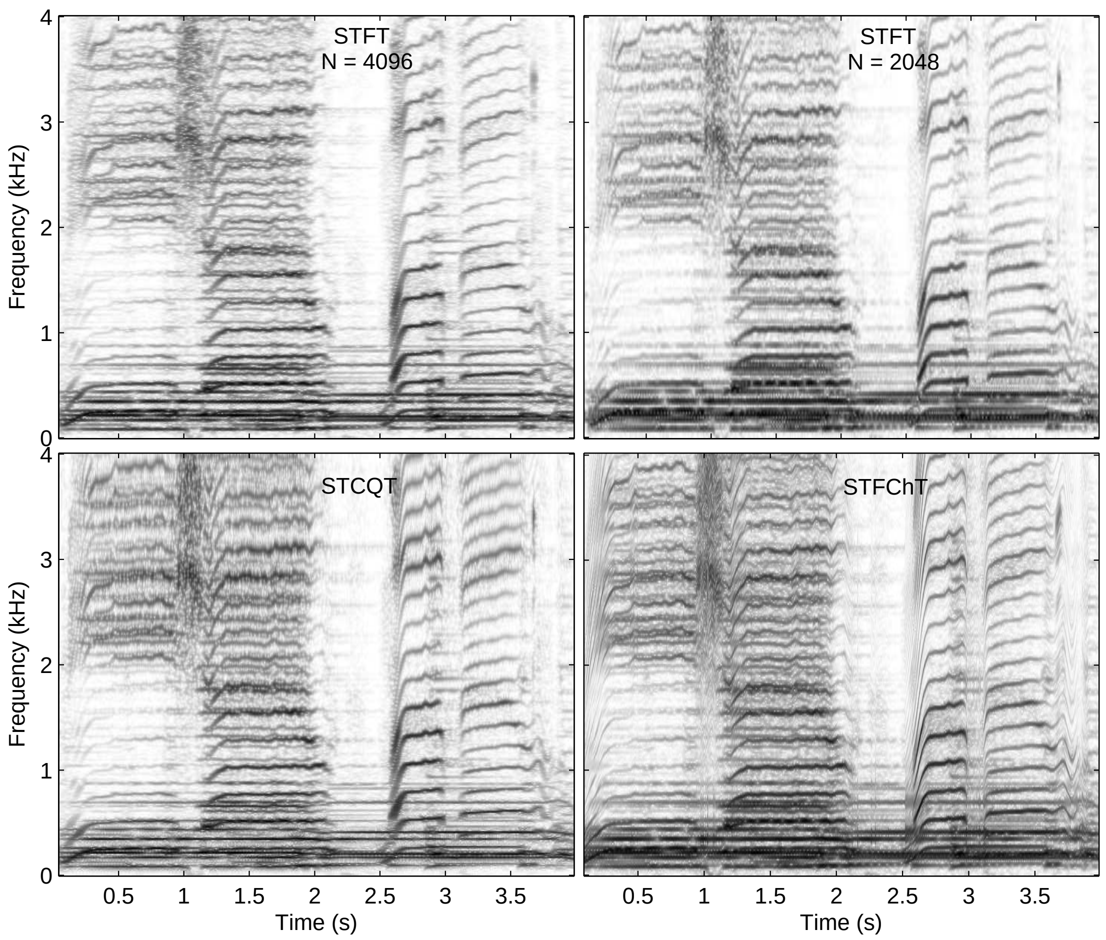


Figure 7.2: Comparison of time-frequency representations for an audio excerpt (which is used throughout the document) of the music file "pop1.wav" from the MIREX [90] melody extraction test set. It consists of three simultaneous prominent singing voices in the first part followed by a single voice in the second part, and a rather soft accompaniment without percussion. The representations depicted are: Spectrograms for window length of 4096 and 2048 samples at $f_s = 44100$ Hz, a Short Time CQT for a Q value corresponding to 34 cycles for each frequency within the analysis window and a Short Time FChT tuned to the most prominent harmonic source in each time window frame. Note the improved time-frequency resolution for the most prominent singing voice in the latter representation.

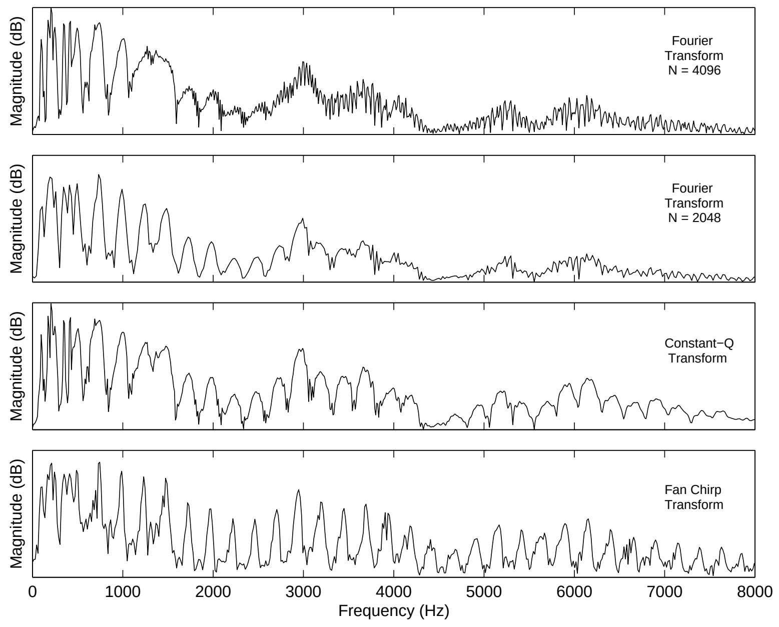


FIGURE 7.3: Comparison of frequency representations for a frame of the audio excerpt of Figure 7.2 at time instant $t = 2.66s$. The prominent singing voice has a very high pitch change rate at this instant. This produces a blurry representation of the strongly non-stationary higher harmonics in the Fourier Transform. The representation of these harmonics is improved with the CQT because of the use of shorter time windows in high frequency. The FChT exhibits the most clear harmonic peak structure.

7.2 Constant Q Transform

7.2.1 Existing methods

Several proposals have been made to circumvent the conventional linear frequency and constant resolution of the DFT. The constant-Q transform (CQT) [78] is based on a direct evaluation of the DFT but the channel bandwidth Δf_k varies proportionally to its center frequency f_k (being k the bin index), in order to keep constant its quality factor $Q = f_k/\Delta f_k$. Considering that $\Delta f_k = fs/N[k]$, the quality factor becomes, $Q = f_k/(fs/N[k])$, and it can be kept constant if the length $N[k]$ of the window function $w_k[n]$ varies inversely with frequency. This implies that always Q cycles for each frequency are analyzed. The expression for the kth spectral component of the CQT is,[2]

$$X^{cq}[k] = \frac{1}{N[k]} \sum_{n=0}^{N[k]-1} w_k[n]x[n]e^{-j2\pi Qn/N[k]}. \tag{7.1}$$

Direct evaluation of the CQT is very time consuming, but fortunately an approximation can be computed efficiently taking advantage of the FFT [88]. In the original formulation center frequencies are distributed geometrically, to follow the equal tempered scale used in Western music, in such a way that there are two frequency components for each

[2]A normalization factor $1/N[k]$ must be introduced since the number of terms varies with k.

musical note (although higher values of Q provide a resolution beyond the semitone). However, other frequency bin spacing can also be used, such as constant separation [73].

Various approximations to a constant-Q spectral representation have also been proposed. The bounded-Q transform (BQT) [91] combines the FFT with a multirate filterbank. Octaves are distributed geometrically, but within each octave, channels are equally spaced, hence the log representation is approximated but with a different number of channels per octave. Note that the quartertone frequency distribution, in spite of being in accordance with Western tuning, can be too scattered if instruments are not perfectly tuned, exhibit inharmonicity or are able to vary their pitch continuously (e.g. glissando or vibrato). Recently a new version of the BQT with improved channel selectivity was proposed in [92] by applying the FFT structure but with longer kernel filters, a technique called Fast Filter Bank. An approach similar to the BQT is followed in [93] as a front-end to detect melody and bass line in real recordings. Also in the context of extracting the melody of polyphonic audio, different time-frequency resolutions are obtained in [89] by calculating the FFT with different window lengths. This is implemented by a very efficient algorithm, named the Multi-Resolution FFT (MR FFT), that combines elementary transforms into a hierarchical scheme.

7.2.2 IIR filtering of the spectrum (IIR CQT)

Previous multi-resolution analysis methods are generally based on applying time windows of different lengths. Multiplying the signal frame with a time window corresponds to convolving the spectrum of the signal with the spectrum of the window. This is equivalent to filtering the spectrum of the signal. Thus, variable windowing in time can also be achieved applying an IIR filterbank in the frequency domain. Let us define the k^{th} filter as a first order IIR filter with a pole p_k, and a zero z_k, as,

$$Y_k[n] = X[n] - z_k X[n-1] + p_k Y_k[n-1] \tag{7.2}$$

Its Z transform is given by,

$$H_{f_k}(z) = \frac{z - z_k}{z - p_k}.$$

Here, $H_{f_k}(z)$ evaluated in the unit circle $z = e^{j\tau}$ represents its time response, with $\tau \in (-\pi, \pi]$ being the normalized time within the frame. A different time window for each frequency bin is obtained by selecting the value of the k^{th} bin as the output of the k^{th} filter.

The design of these filters involves finding the zero and pole for each k such that $w_k(\tau) = |H_{f_k}(e^{j\tau})|$, where $\tau \in (-\pi, \pi]$ and $w_k(\tau)$ is the desired window for the bin k. When a frame is analyzed, it is desirable to avoid discontinuities at its ends. This can be achieved by placing the zero in $\tau = \pi$, that is $z_k = -1$. If one is interested in a symmetric window, i.e. $w_k(\tau) = w_k(-\tau)$, the pole must be real. Considering a causal realization of the filter,

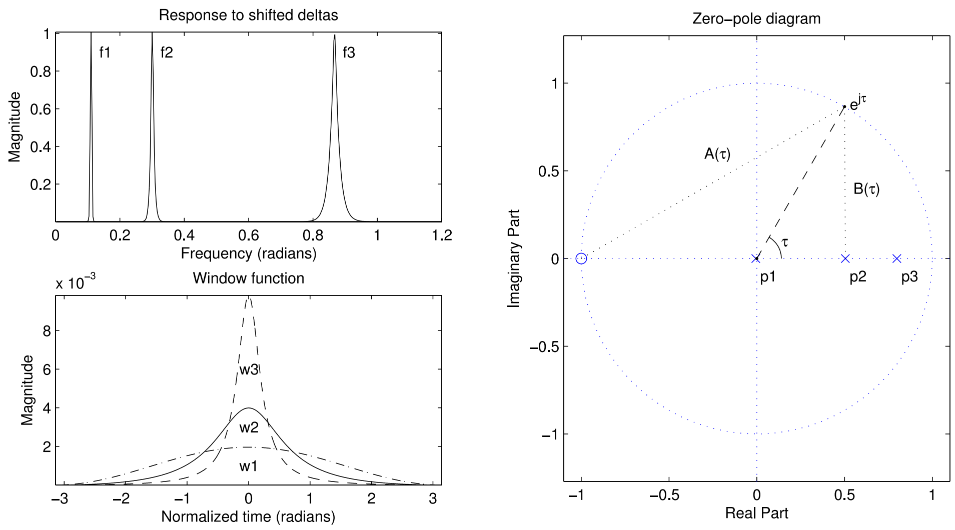


FIGURE 7.4: Zero-Pole diagram and IIR filters responses for three different input sinusoids of frequencies $f_1 = 0.11$, $f_2 = 0.30$ and $f_3 = 0.86$ radians.

p_k must be inside the unit circle to assure stability, thus $p_k \in (-1, 1)$. Figure 7.4 shows the frequency and time responses for the poles depicted in the zero-pole diagram.

This IIR filtering in frequency will also distort the phase, so a forward-backward filtering should be used to obtain a zero-phase filter response. Then, the set of possible windows that can be represented with these values of p_k is,

$$w_k(\tau) = \frac{(1-p_k)^2}{4}\left[\frac{A(\tau)}{B(\tau)}\right]^2 = \frac{(1-p_k)^2(1+\cos\tau)}{2(1+p_k^2-2p_k\cos\tau)} \tag{7.3}$$

where $A(\tau)$ and $B(\tau)$ are the distances to the zero and the pole, as shown in Figure 7.4, and $g_k = (1-p_k)^2/4$ is a normalization factor[3] to have 0 dB gain at time $\tau = 0$, that is, $w_k(0) = 1$.

While this filter is linear and time invariant (in fact frequency invariant[4]) a different time window is desired for each frequency component. Computing the response of the whole bank of filters for the entire spectrum sequence and then choosing the response for only one bin is computationally inefficient. For this reason, a Linear Time Variant (LTV) system, that consists in a Time Varying (TV) IIR filter, is proposed as a way to approximate the filterbank response at the frequency bins of interest.

A direct way of approximating the IIR filterbank is by a first order IIR of the form of equation 7.2, but in which the pole varies with frequency ($p = p[n]$),

$$Y[n] = X[n] + X[n-1] + p[n]Y[n-1]. \tag{7.4}$$

[3]This normalization factor can be calculated from the impulse response evaluated at $n = 0$, or by the integral of the time window function.

[4]Note that we use the usual time domain filtering terminology in spite of the fact that filtering is performed in the frequency domain.

```
p = design_poles(NFFT,Q);
X = fft(fftshift(s));
Y'(1) = X(1);
for n = 2:NFFT/2
    Y'(n) = X(n-1) + X(n) + p(n)Y'(n-1);
end
Y(n) = Y'(NFFT/2);
for n = NFFT/2-1:-1:1
    Y(n) = Y'(n+1) + Y'(n) + p(n)Y(n+1);
end
```

TABLE 7.1: Pseudocode of the TV IIR filter. First, the poles and normalization factor are designed given the number of bins (NFFT) and the Q value. Then the FFT of the signal frame s is computed after centering the signal at time 0. Finally the forward-backward TV IIR filtering is performed for that frame.

With an appropriate design, it reasonably matches the desired LTV IIR filterbank response, and its implementation has low computational complexity. In the proposed approach the first step is to design an IIR filterbank that accomplishes the constant Q behavior. Then, a TV IIR filter is devised based on the poles of the filterbank. A simple and effective design of the TV IIR filter consists in choosing for each frequency bin the corresponding pole of the IIR filterbank, that is $p[n] = p_k$, with $k = n$. For low frequencies a constant Q would imply a window time support longer than the frame time, so in practice it becomes necessary to set a limit. Finally a fine tuning is performed to improve the behaviour of the TV IIR filter in order to effectively obtain a constant Q value. Figure 7.5 shows a detail of the poles design for low frequencies. The reader is referred to the original paper [73] for further details on the design process.

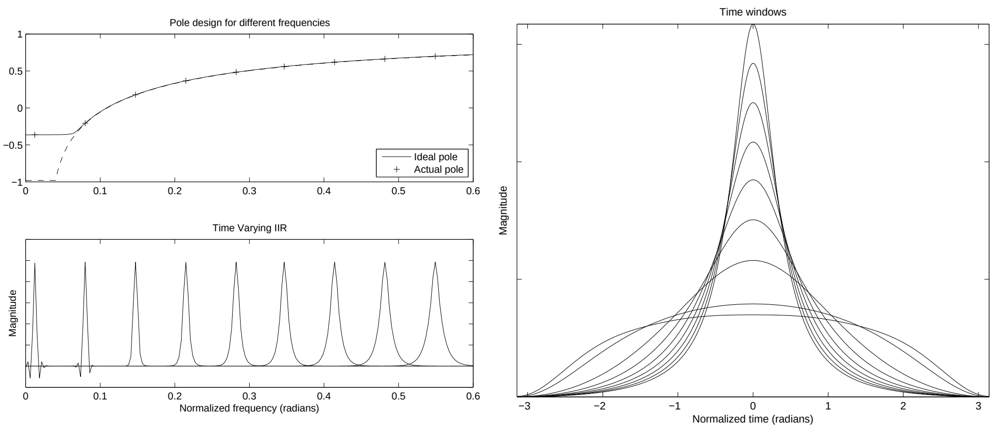


FIGURE 7.5: Detail of poles design at different low frequencies. Pole location for the ideal and actual design. Impulse response and windows of the TV IIR.

The implementation is rather simple, as can be seen in the pseudocode of Table 7.1. A function to design the poles is called only once and then the forward-backward TV IIR filtering is applied to the DFT of each signal frame. The proposed IIR filtering applies a window centered at time 0, so the signal frame has to be centered before the transform.

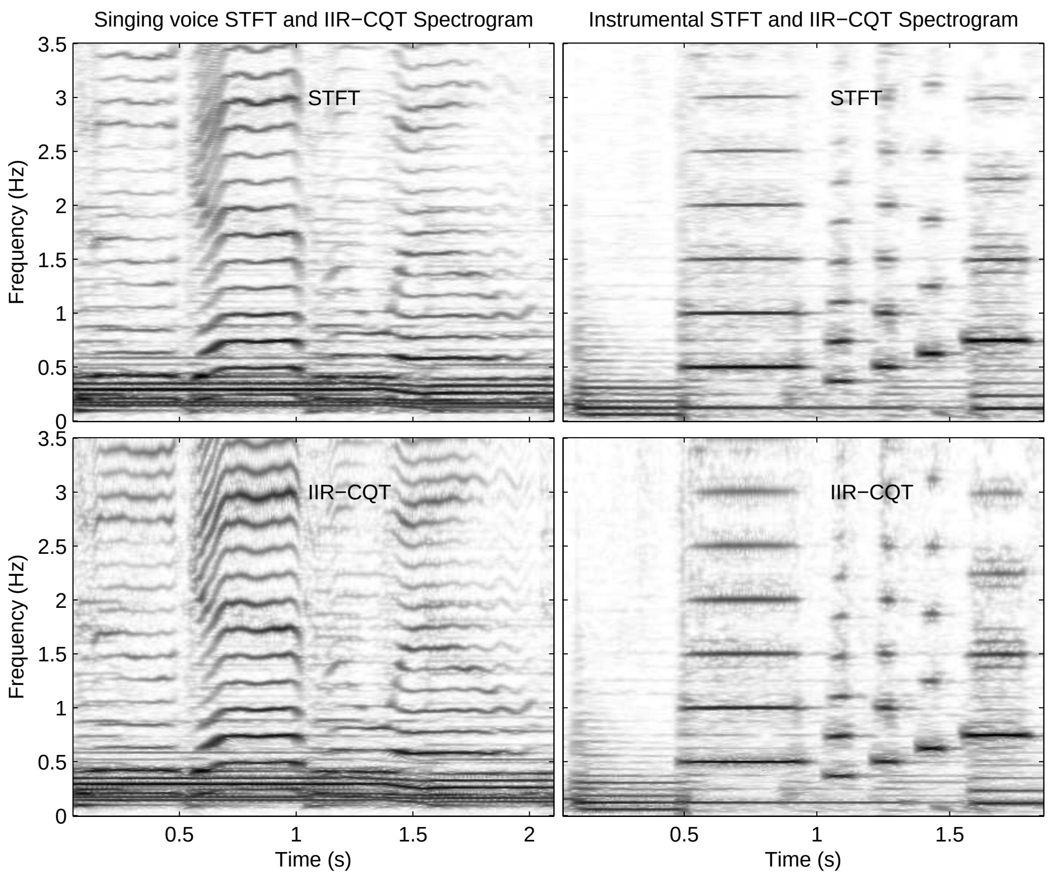


FIGURE 7.6: STFT and IIR CQT for two audio excerpts, one with a leading singing voice and the other, instrumental music.

Finally, two examples of the IIR CQT analysis of polyphonic music are shown in Figure 7.6 compared to conventional spectrograms. As expected for a CQT, singing voice partials with high frequency slope are sharper in the IIR CQT than in the spectrogram. This improved time resolution in high frequencies also contributes to define more precisely the note onsets, as can be seen in the second example (e.g. the bass note at the beginning). Moreover, in the low frequency band, where there is a higher density of components, the IIR CQT achieves a better discrimination. At the same time, frequency resolution for the higher partials of notes with a steady pitch is deteriorated.

In our work [73] the proposed method for computing a constant Q spectral transform is compared with two existing techniques. It shows to be a good compromise between the flexibility of the efficient CQT and the low computational cost of the MR FFT. Taking into account that it was used in the spectral analysis of music with encouraging results[5] [94] and that its implementation is rather simple, it seems to be a good spectral representation tool for audio signal analysis algorithms.

[5]The method is part of the spectral analysis front-end of a melody extraction algorithm submitted by Pablo Cancela to the MIREX Audio Melody Extraction Contest 2008, performing best on Overall Accuracy. Evaluation procedure and results are available at http://www.music-ir.org/mirex/2008/index.php/Audio_Melody_Extraction.

7.3 Fan Chirp Transform

7.3.1 Formulation

In our work [74], the proposed definition of the FChT is,

$$X(f,\alpha) \triangleq \int_{-\infty}^{\infty} x(t)\ \phi'_\alpha(t)\ e^{-j2\pi f \phi_\alpha(t)} dt, \tag{7.5}$$

where $\phi_\alpha(t) = (1 + \frac{1}{2}\alpha t)\,t$, is a time warping function, and α is a parameter called *chirp rate*. This was formulated independently from the original work [83], so the properties are slightly different as will be indicated later. Notice that by the change of variable $\tau = \phi_\alpha(t)$, the formulation becomes,

$$X(f,\alpha) = \int_{-\infty}^{\infty} x(\phi_\alpha^{-1}(\tau))\ e^{-j2\pi f \tau} d\tau, \tag{7.6}$$

which can be regarded as the Fourier Transform of a time warped version of the signal $x(t)$, and enables an efficient implementation based on the FFT. The goal pursued is to obtain a precise representation of linear chirp signals of the form $x_c(t,f) = e^{j2\pi f \phi_\alpha(t)}$. Considering a limited analysis time support, the analysis basis is $\Gamma = \{\gamma_k\}_{k\in\mathbb{Z}}$, $\gamma_k = \phi'_\alpha(t)\ e^{j2\pi \frac{k}{\breve{T}}\phi_\alpha(t)}$, $t \in \left[\phi_\alpha^{-1}(-\frac{\breve{T}}{2}), \phi_\alpha^{-1}(\frac{\breve{T}}{2})\right]$. The inner product of the chirp and a basis element results in,

$$\begin{aligned}\left\langle x_{ch}(t, 2\pi\tfrac{l}{\breve{T}}), \gamma_k \right\rangle &= \frac{1}{\breve{T}} \int_{\phi_\alpha^{-1}(-\frac{\breve{T}}{2})}^{\phi_\alpha^{-1}(\frac{\breve{T}}{2})} \phi'_\alpha(t)\ e^{j2\pi \frac{l-k}{\breve{T}} \phi_\alpha(t)} dt \\ &= \frac{1}{\breve{T}} \int_{-\frac{\breve{T}}{2}}^{\frac{\breve{T}}{2}} e^{j2\pi \frac{l-k}{\breve{T}}\tau} d\tau = \delta[l-k], \end{aligned} \tag{7.7}$$

which denotes that only one element of the basis concentrates all the energy of the chirp. Note that the limits of integration include an integer number of cycles of the chirp, in the warped and the original time interval.

In [83] the basis are designed to be orthonormal, in order to obtain perfect reconstruction directly from the analysis basis. However, its response to a chirp of constant amplitude is not represented by a single element. It is important to note that when the signal is windowed the orthogonality disappears so as the perfect reconstruction. In a similar way, the result given by equation 7.7 does not hold anymore. To that end, it is worth defining a more appropriate goal, that is what kind of response would be desirable for a time limited chirp. The approach proposed in our work [74] permits to achieve a delta convolved with the Fourier Transform of a well-behaved analysis window. This motivates the above definition of the analysis basis Γ and the application of the analysis window to the time warped signal (which also differs from [83]). Then, the proposed

FChT for a time limited support is,

$$X_w(f,\alpha) = \int_{-\infty}^{\infty} x(t)\; w(\phi_\alpha(t))\; \phi'_\alpha(t)\; e^{-j2\pi f \phi_\alpha(t)} dt \tag{7.8}$$

where $w(t)$ stands for a time limited window, such as Hann.

Consider the case of a signal composed of L harmonically related linear chirps, i.e. a harmonic chirp, $x_{hc}(t, f_0, L) = \sum_{k=1}^{L} e^{j2\pi k f_0 \phi_\alpha}$. All components share the same fan chirp rate α, so applying the appropriate warping ϕ_α delivers constant frequency harmonically related sinusoidal components. The FChT representation therefore shows a sharp harmonic structure as it is composed of deltas convolved with the Fourier Transform of the window. This situation is illustrated in Figure 7.7 for a harmonic chirp.

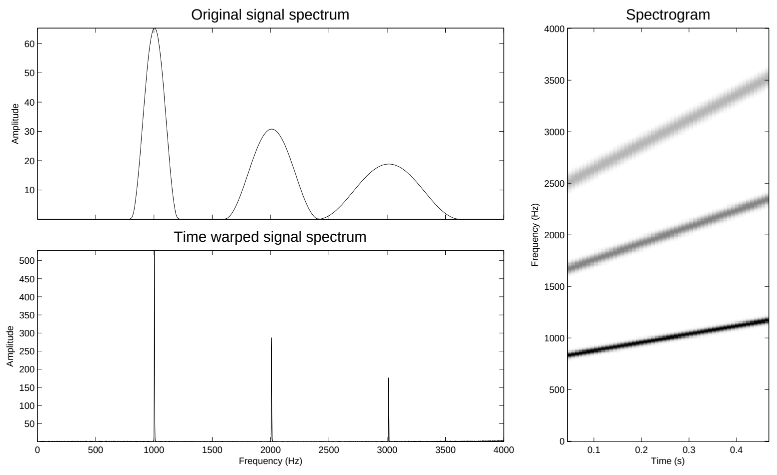


FIGURE 7.7: Analysis of the harmonic chirp of three components depicted in the spectrogram on the right. Appropriately warping the signal produces a harmonic signal of constant frequency, which corresponds to an optimally concentrated spectrum.

7.3.2 Discrete time implementation

As stated before, the FChT of a signal $x(t)$ can be computed by the Fourier Transform of the time warped signal $\breve{x}(t) = x\left(\phi_\alpha^{-1}(t)\right)$, where

$$\phi_\alpha^{-1}(t) = -\frac{1}{\alpha} + \frac{\sqrt{1+2\alpha t}}{\alpha}. \tag{7.9}$$

This warping function transforms linear chirps of instantaneous frequency $\nu(t) = (1+\alpha t)\, f$ into sinusoids of frequency $\breve{\nu}(t) = f$. In practice, the original signal is processed in short time frames. In order to properly represent it with its warped counterpart, temporal warping is implemented by adopting the following criteria. After the time warping, the frequency of the resulting sinusoid is the frequency of the linear chirp at the center of the analysis window. Besides, the amplitude value of the warped signal remains unchanged in the central instant of the window. This implies that the duration of the original signal and the warped signal may be different, something that is not imposed in [83].

The temporal warping is implemented by non-uniform resampling of the finite length discrete signal frame $x[n]$. An equally-spaced grid is constructed in the warped time domain. To compute the sample corresponding to time instant $\breve{t}_m$ of the warped signal, it is necessary to evaluate $x[n]$ at time instant $t_m = \phi_\alpha^{-1}(\breve{t}_m)$. As this instant may not coincide with a sampling time, the evaluation must be done using some interpolation technique. Time warping process is illustrated in Figure 7.8. The last step of the FChT is to apply an analysis window to the time warped signal and compute the DFT.

Another consideration regarding the implementation, is that time warping design is performed numerically based on relative instantaneous frequency functions. More precisely, the design begins with the selection of the warping instantaneous frequency $f_r[n]$ for each sample. Then, the function $\phi[n]$ is obtained by numerical integration of $f_r[n]$. Finally the function $\phi^{-1}[n]$, needed to compute the resampling times, is obtained by numerical inversion. This allows the implementation of arbitrary warping functions instead of only linear warpings.

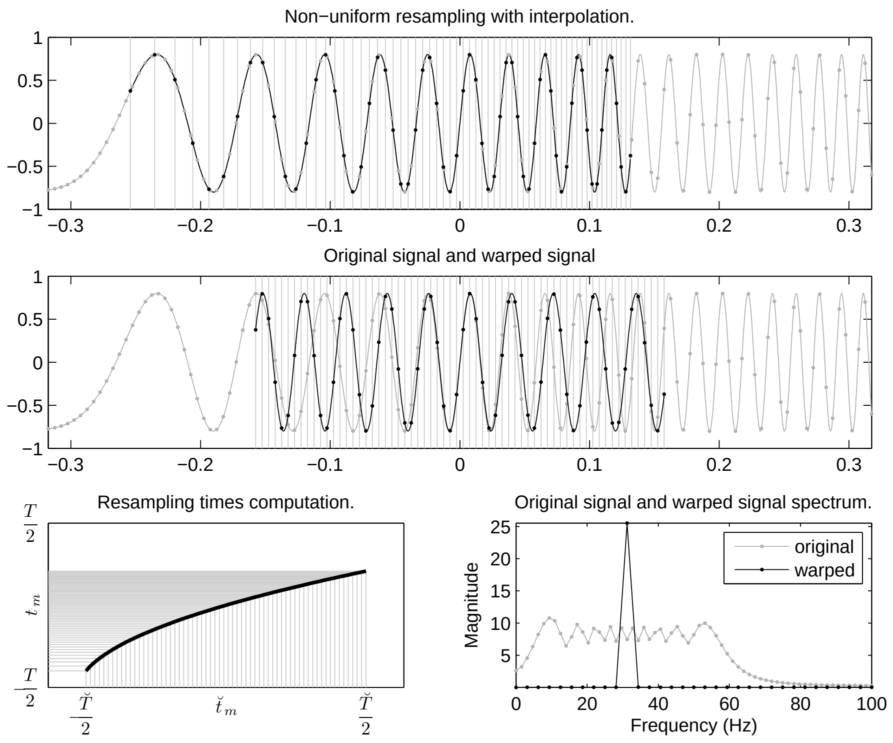


FIGURE 7.8: Illustration of the warping process. A sinusoid is obtained by appropriately warping a linear chirp. Note that the central time instant remains the same and the time supports are different. The FChT of the linear chirp shows a sharp high peak.

7.3.3 Fan Chirp Transform for music representation

In a practical situation, real signals such as speech or music sounds can be assimilated to harmonically related linear chirps only within short time intervals, where the evolution of frequency components can be approximated by a first order model. This suggests the application of the FChT to consecutive short time signal frames, so as to build a time-frequency representation as a generalization of the spectrogram [83]. In the monophonic case, a single value of the fan chirp rate α that best matches the signal pitch variation rate should be determined for each frame. Selecting the fan chirp rate is the key factor to obtain a detailed representation using the FChT. Different approaches could be followed, such as predicting the pitch evolution and estimating α as the relative derivative of the pitch [83]. Figure 7.9 shows the comparison of the STFT and the STFChT for a monophonic synthetic harmonic signal that exhibits a sinusoidal frequency modulation. The fan chirp rate α value for each frame was selected as the pitch change rate, which is known in this case.

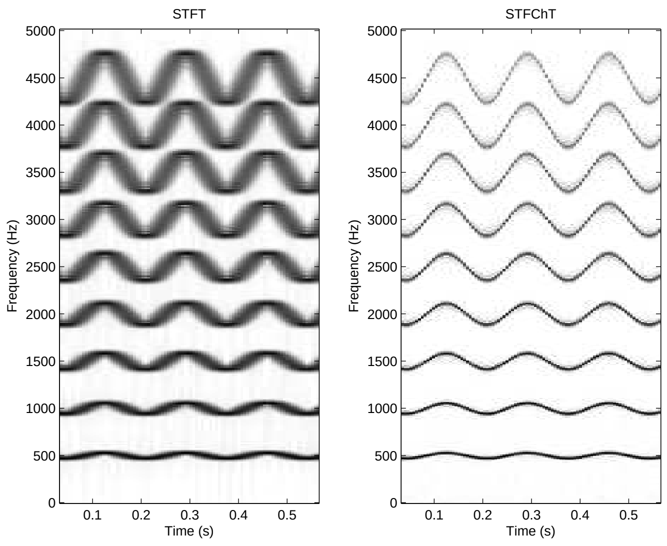


Figure 7.9: Comparison of the STFT and the STFChT for the analysis of a synthetic harmonic signal with a sinusoidal frequency modulation that emulates a vibrato. For the computation of the FChT in each frame the actual pitch change rate of the signal, which is known, was used to set the fan chirp rate α value.

In the polyphonic case, there is no single appropriate value of α, because the multiple harmonic sounds present are likely to change their fundamental frequency (f_0) differently within the analysis frame. For this reason, a multi-dimensional representation for each frame seems better suited in this case, consisting in several FChT instances with different α values. A given FChT is tuned to represent one of the harmonic sounds with reduced spectrum spread, whereas poorly representing the remaining ones. This is illustrated in Figure 7.10 for the synthetic signal used before (Figure 7.1), consisting of a harmonic stationary signal and a harmonic chirp.

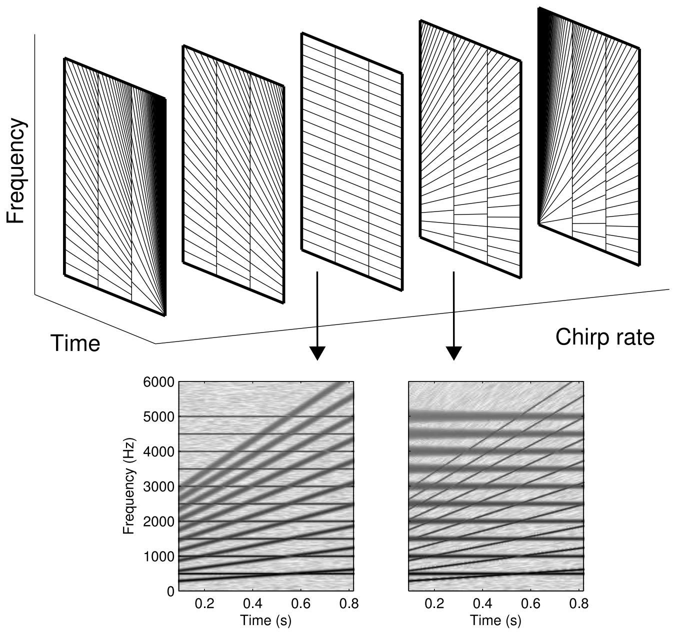


Figure 7.10: Multi-dimensional representation used for polyphonic signals. In every frame several FChT instances are selected, which better represent each single harmonic sound. A given FChT provides an accute representation for the source to which it is tuned, but the other sources are likely to be diffuse.

The selection of a reduced set of α values for each frame that produce the better representation of each sound present, can be tackled by means of sinusoidal modeling techniques as in [86]. In our work [74] a straightforward exhaustive approach is adopted, that consists in computing for each audio frame several FChT instances whose α values are in a range of interest. Then, using this multidimensional representation a dense (f_0, α) plane is constructed and the best chirp rates are selected based on pitch salience (see Chapter 8, section 8.1.5). This makes use of the fact that the energy of a harmonic source is more concentrated for the FChT instance whose α value better matches its corresponding pitch change rate (see Figure 7.11). Hence for this representation the harmonics of the source have higher amplitudes compared to any other FChT instance. In addition, the pitch salience computation from the FChT produces itself a detailed representation of the melodic content of the signal, that can be useful in several applications. This is described in detail in the following chapter.

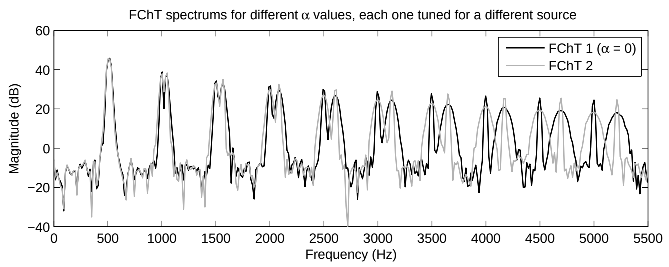


Figure 7.11: Comparison of the FChT instances of Figure 7.10 at time $t = 0.6$ s. It can be noticed how the harmonics have higher amplitudes for the correct α value.

8 Pitch tracking in polyphonic audio

Multiple fundamental frequency (f0) estimation is one of the most important problems in music signal analysis and constitutes a fundamental step in several applications such as melody extraction, sound source identification and separation. There is a vast amount of research on pitch tracking in audio, often comprising an initial frame by frame f0 estimation followed by formation of pitch contours exploiting estimates continuity over time. Techniques such as dynamic programming, linear prediction, hidden Markov models, among many others (see [95] for a review), were applied to the temporal tracking.

This chapter describes the algorithm developed in the context of this thesis work for pitch tracking in polyphonic audio, which is particularly oriented towards the singing voice. The local pitch estimation technique was devised in collaboration with Pablo Cancela and Ernesto López and was reported in an international conference paper [74]. A pitch salience representation for music analysis called F0gram was proposed, based on the Fan Chirp Transform. The representation provides a set of local fundamental frequency candidates together with a pitch change rate estimate for each of them. The pitch tracking algorithm that performs temporal integration of local pitch candidates is based on unsupervised clustering of F0gram peaks. The technique benefited from fruitful discussions with Pablo Cancela and was reported in a regional conference paper [96]. The herein description reproduces some passages of the original articles and also includes modifications and additions. This is done in order to put the work in the context of this dissertation.

8.1 Local pitch estimation based on the FChT

8.1.1 Pitch salience computation

The aim of pitch salience computation is to build a continuous function that gives a prominence value for each fundamental frequency in a certain range of interest. Ideally it shows pronounced peaks at the positions corresponding to the true pitches present in the signal frame. This detection function typically suffers from the presence of spurious peaks at multiples and submultiples of the true pitches, so some sort of refinement is required to reduce this ambiguity. A common approach for pitch salience calculation is to define a fundamental frequency grid, and compute for each frequency value a weighted sum of the partial amplitudes in a whitened spectrum [97]. A method of this kind was formulated in our work [74] according to the log-spectrum gathering proposed in [84], and is described in the following. The whole process of local pitch estimation and tracking is illustrated by a single audio example whose spectrogram is depicted in Figure 8.1, and was introduced in the previous chapter (Figure 7.2).

8.1.2 Gathered log-spectrum (GlogS)

The salience of a given fundamental frequency candidate f_0 can be obtained by gathering the log-spectrum at the positions of the corresponding harmonics as [84]

$$\rho_0(f_0) = \frac{1}{n_H} \sum_{i=1}^{n_H} \log |X(if_0)|, \tag{8.1}$$

where $X(f)$ is the spectrum of a signal frame and n_H is the number of harmonics that are supposed to lie within the analysis bandwidth. Linear interpolation along samples of the discrete log-spectrum is applied to estimate the values at arbitrary frequency positions. The logarithm provides better results compared to the gathering of the linear spectrum. This makes sense, because the logarithm function can be regarded as a kind of whitening in order to make the pitch salience computation more robust against formant structure and noise. With this respect, it is interesting to note that a p-norm with $0 < p < 1$ is also appropriate and shows similar results. Note that this seems coherent with the use of more robust norms which is advocated in sparsity research. Therefore, the actual implementation is $\log(\gamma|X(if_0)| + 1)$ which adds the flexibility to custom the norm applied by means of the γ parameter[1].

[1]Higher values of γ tend to a 0-norm while lower values tend to a 1-norm. All the results reported correspond to $\gamma = 10$.

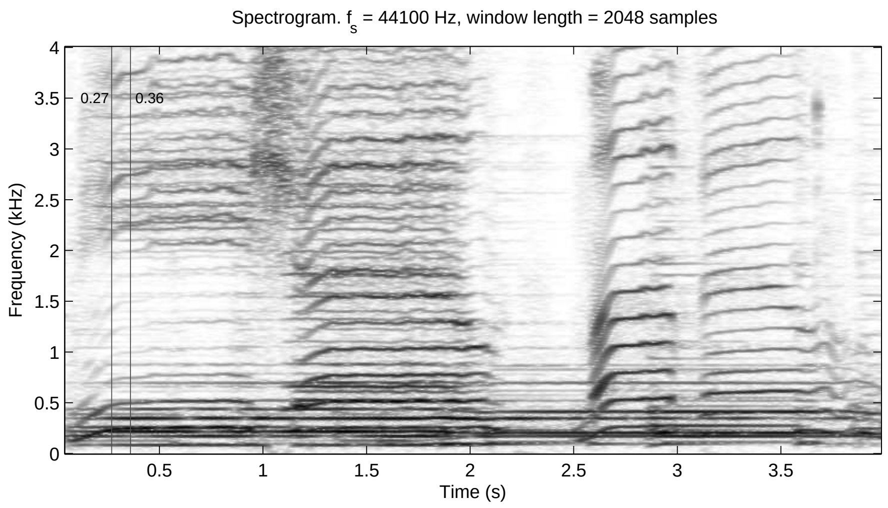


FIGURE 8.1: Spectrogram of the audio excerpt introduced in Figure 7.2 from the MIREX melody extraction test set. This clip is used throughout the chapter to illustrate different steps of the pitch tracking algorithm. It consists of three simultaneous prominent singing voices in the first part followed by a single voice in the second part, and a rather soft accompaniment without percussion. Two time instants are indicated corresponding to Figure 8.2 and Figure 8.4.

8.1.3 Postprocessing of the gathered log-spectrum

The harmonic accumulation shows peaks not only at the position of the true pitches, but also at multiples and submultiples (see Figure 8.2). To handle the ambiguity produced by multiples, the following simple non-linear processing is proposed in [84]

$$\rho_1(f_0) = \rho_0(f_0) - \max_{q \in \mathbb{N}} \rho_0(f_0/q). \tag{8.2}$$

This is quite effective in removing pitch candidates multiples of the actual one (as can be seen in Figure 8.2). When dealing with monophonic signals this suppression is enough. If pitch estimation is obtained as the position of the maximum of $\rho_1(f_0)$, $\hat{f}_0 = \arg\max \rho_1(f_0)$, submultiple spurious peaks do not affect the estimation because their amplitude is necessarily lower than for the true pitch. However, in the polyphonic case, submultiple peaks should also be removed. For this reason, the detection function is further processed to remove the (k-1)-th submultiple according to,

$$\rho_2(f_0) = \rho_1(f_0) - a_k\, \rho_1(kf_0) \tag{8.3}$$

where a_k are attenuation factors. From the simulations conducted it turned out that removing only the first and second submultiples ($k = 2$ and $k = 3$), for an f_0 range of four octaves, is commonly sufficient for melodic content visualization and melody detection (see Figure 8.2). For a single ideal harmonic sound, it can be shown that the attenuation factor value for removing the first submultiple $a_2 = 1/2$ (see Appendix C). However, it can also be shown that the variance of $\rho_0(f_0)$ is proportional to fundamental frequency

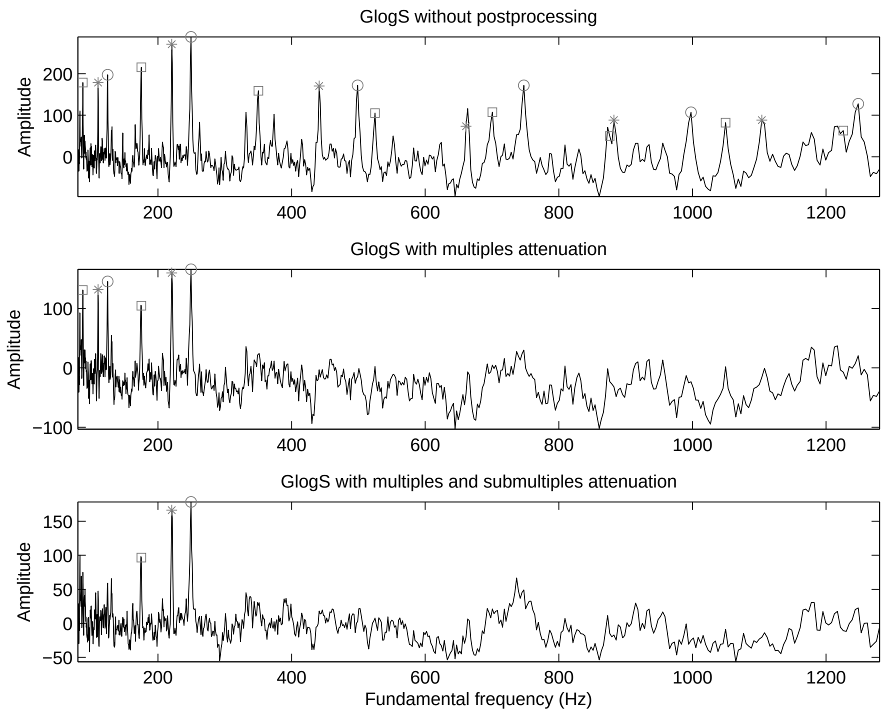


FIGURE 8.2: Normalized gathered log spectrum and the postprocessing stages for a frame of the audio excerpt at $t = 0.36$s. In the frame there are three prominent simultaneous singing voices and a low accompaniment. Positions of each corresponding f_0, multiples and first submultiple are also depicted. Only the first submultiple is attenuated in this example.

(see [84] appendix B). In practice a true pitch peak can be unnecessarily attenuated due to the large variance at its multiple, so a more conservative attenuation factor is preferred. Slightly better results were experimentally obtained over polyphonic music for $a_2 = 1/3$ and $a_3 = 1/6$, which are the values used for the reported results.

8.1.4 Normalization of the gathered log-spectrum

The variance increase with f_0 is an undesired property. It leads to unbalanced frequency regions in melodic content visualization and to incorrect detections when pursuing melody extraction. For this reason, the last step in pitch salience computation is to normalize $\rho_2(f_0)$ to zero mean and unit variance. To do this, the mean and the variance of $\rho_2(f_0)$ are collected at each f_0 for every frame from a music collection (the complete RWC Popular Music Database [98] was used for this purpose). Each one of these statistics are then approximated by a second-order polynomial, as illustrated in Figure 8.3. The polynomials evaluated at each f_0 are the model used to obtain a normalized gathered log-spectrum $\overline{\rho}_2(f_0)$. The fundamental frequency grid used is logarithmically spaced with 192 points per octave, from E_2 extending over 4 octaves.

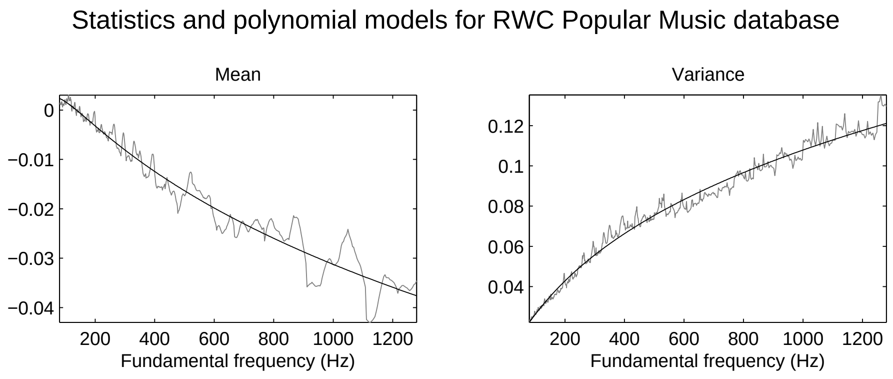


Figure 8.3: Gathered log spectrum normalization model.

8.1.5 Fan chirp rate selection using pitch salience

As early mentioned, the α values that best represent the different harmonic sounds in a signal frame are selected by means of pitch salience. Several FChT instances are computed for each frame using different α values. For each FChT a normalized gathered log spectrum is calculated as described above, so as to build a dense pitch salience plane $\overline{\rho}_2(f_0, \alpha)$. See Figure 8.4 for an example of this dense pitch salience plane. Given a sound source of fundamental frequency $\hat{f}_0$, the energy of its harmonics is more concentrated at the FChT instance corresponding to the best matching α value $\hat{\alpha}$. Therefore, the value of $\overline{\rho}_2(\hat{f}_0, \hat{\alpha})$ is the highest among the different available α values. For this reason, a different α value is selected for each f_0 in the grid, giving a single pitch salience value for each f_0 (see Figure 8.4). Thus, the reduced set of α values can be selected according to their corresponding pitch salience.

8.1.6 Pitch visualization: F0gram

The pitch salience function of each frame obtained as described in the previous section is used to build an F0gram that shows the temporal evolution of pitch for all the harmonic sounds of a music audio signal, as can be seen in Figure 8.5. Note that even in the case that two sources coincide in time and frequency they can be correctly represented if their pitch change rate is different, which is observable at time $t = 0.23$s. It can also be seen the precise pitch contour evolution obtained, even for severe pitch fluctuations. Additionally, the gathered log spectrum normalization provides a balanced contrast of the F0gram, without spurious noticeable peaks when no harmonic sound is present. A drawback of the selected pitch salience function is that it tends to underestimate low frequency harmonic sounds with a small number of prominent partials. For a sound of this kind, the summation of the power spectrum in equation 8.1 is mainly determined by a few number of harmonic positions, but as n_H is high the resulting accumulated value will be over attenuated. This is the case for the accompaniment in the selected example, that only appears when no singing voice is present. Therefore, the F0gram is

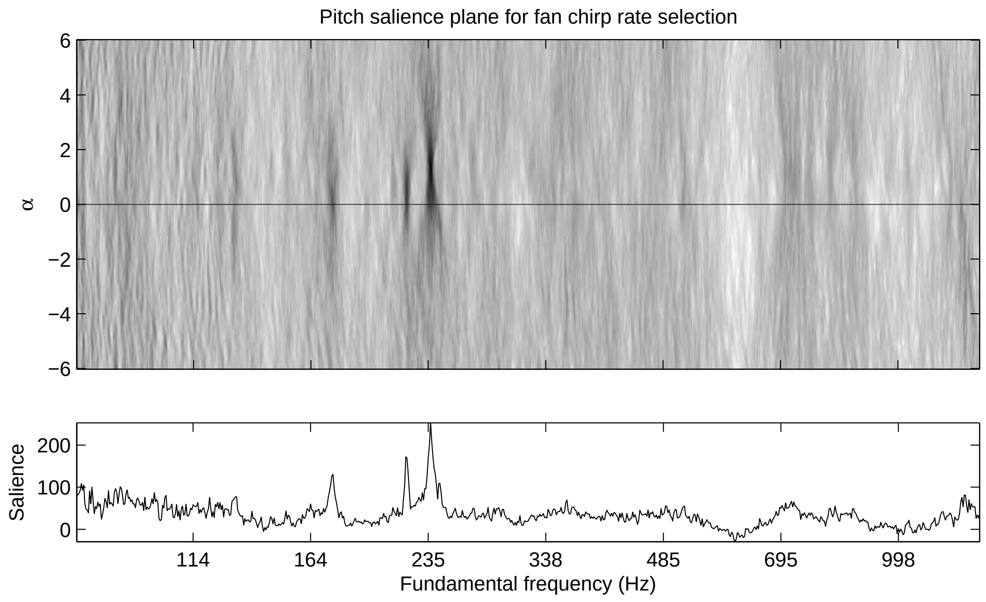


FIGURE 8.4: Pitch salience plane $\overline{\rho}_2(f_0, \alpha)$ for a frame of the audio excerpt at $t = 0.27$s. Prominent salience peaks (darker regions) can be distinguished corresponding to the three singing voices. Note that two of them are located approximately at $\alpha = 0$ and one at $\alpha = 1.3$. This indicates that two of the voices are quite stationary within the frame while the other is increasing its pitch. The maximum pitch salience value for each f_0 is also depicted.

somehow tailored for the singing voice, which usually exhibit a high number of salient harmonics. This kind of visualization tool can be useful itself for analyzing performance expressive features such as glissando, vibrato, and pitch slides, that turn out clearly distinguishable.

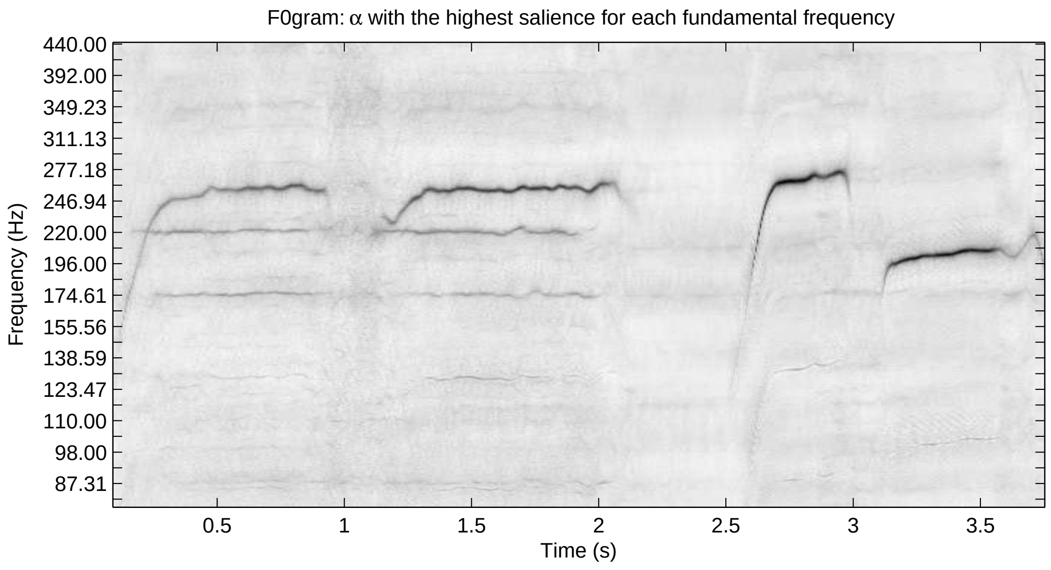


FIGURE 8.5: Example of melodic content visualization for the selected audio excerpt. The pitch contour of the three simultaneous singing voices followed by a single voice can be clearly appreciated. Note that the f_0 grid extends far beyond the frequency limit of this graph (approximately up to E_6).

8.2 Pitch tracking by clustering local frequency estimates

The proposed technique for pitch contours formation does not involve a classical temporal tracking algorithm. Instead the pitch tracking is performed by unsupervised clustering of F0gram peaks (which are depicted in Figure 8.6 for the audio example). The spectral clustering method [99] is selected for this task as it imposes no assumption of convex clusters, thus being suitable for filiform shapes such as pitch contours. The pitch change rate estimates provided by the FChT analysis play an important role in the definition of similarity between pitch candidates. The clustering is carried out within overlapped observation windows corresponding to several signal frames. Then contours are formed by simply joining clusters that share elements. This short-term two-stage processing proved to be more robust than aiming a straightforward long-term clustering.

There are very few applications of spectral clustering for tracking a sound source. Blind one-microphone separation of two speakers is tackled in [100] as a segmentation of the spectrogram. A method is proposed to learn similarity matrices from labeled datasets. Several grouping cues are applied such as time-frequency continuity and harmonicity. A simple multiple pitch estimation algorithm is part of the feature extraction. The mixing conditions are very restrictive (equal strength and no reverberation). Performance is assessed through a few separation experiments.

Clustering of spectral peaks is applied in [101], for partial tracking and source formation. Connecting peaks over time to form partials and grouping them to form sound sources is performed simultaneously. The problem is modeled as a weighted undirected graph where the nodes are the peaks of the magnitude spectrum. The edge weight between nodes is a function of frequency and amplitude proximity (temporal tracking) and a harmonicity measure (source formation). Clustering of peaks across frequency and time is carried out for windows of an integer number of frames ($\sim$ 150 ms) using a spectral clustering method. Clusters from different windows are not connected for temporal continuation. The two more compact clusters of each window are selected as the predominant sound source.

8.2.1 Spectral Clustering

The goal of clustering can be stated as dividing data points into groups such that points in the same cluster are similar and points in different clusters are dissimilar. An useful way of representing the data is in the form of a similarity graph, each vertex corresponding to a data point. Two vertices of the graph are connected if their similarity is above certain threshold, and the edge between them is weighted by their similarity value. In terms of the graph representation, the aim of clustering is to find a partition of the graph such that different groups are connected by very low weights whereas edges within a group have high weights.

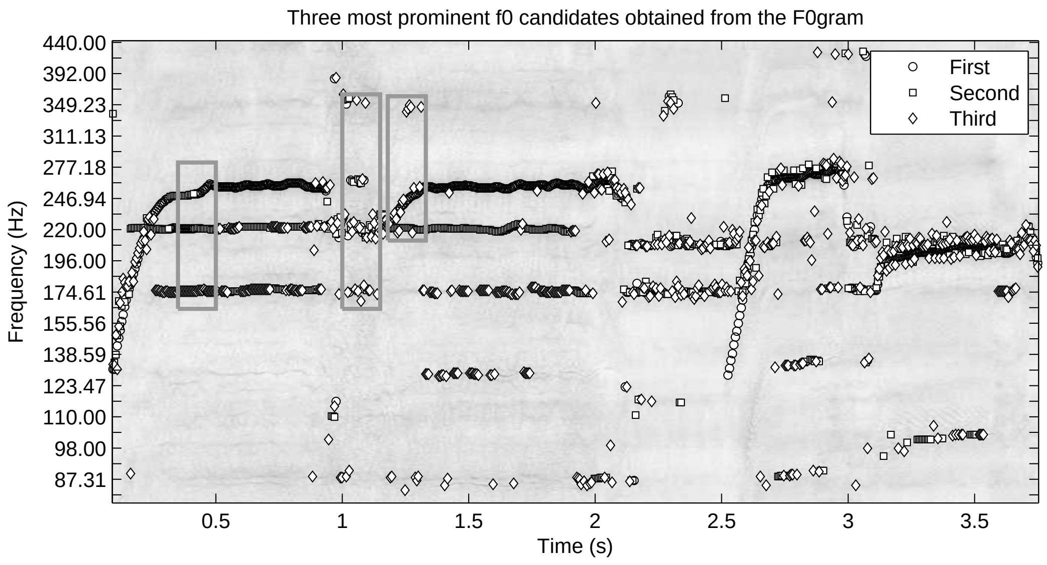


FIGURE 8.6: F0gram and the three most prominent f0 candidates for each signal frame. There are three regions highlighted corresponding to the local clustering examples analyzed in Figure 8.9.

The simplest way to construct a partition is to solve the mincut problem [102]. Given a number of clusters k, it consists in finding a partition $A_1, \ldots, A_k$ that minimises

$$\mathrm{cut}(A_1, \ldots, A_k) = \frac{1}{2} \sum_{i=1}^{k} W(A_i, \bar{A}_i), \tag{8.4}$$

where $W(A,B) = \sum_{i \in A, j \in B} w_{ij}$ is the sum of weights of vertices connecting partitions A and B, and $\bar{A}$ stands for the complement of A. This corresponds to finding a partition such that points in different clusters are dissimilar to each other. The problem with this approach is that it often separates one individual vertex from the rest of the graph. An effective way of avoiding too small clusters is to minimize the Ncut function [99],

$$\mathrm{Ncut}(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{\mathrm{cut}(A_i, \bar{A}_i)}{\mathrm{vol}(A_i)}, \tag{8.5}$$

where $\mathrm{vol}(A) = \sum_{i \in A} d_i$ is the sum of the degree of vertices in A. The degree of a vertex is defined as $d_i = \sum_{j=1}^{n} w_{ij}$, so $\mathrm{vol}(A)$ measures the size of A in terms of the sum of weights of those edges attached to their vertices. The Ncut criterion minimises the between-cluster similarity (in the same way as mincut), but it also implements a maximisation of the within-cluster similarities. Notice that the within-cluster similarity can be expressed as $W(A,A) = \mathrm{vol}(A) - \mathrm{cut}(A, \bar{A})$ [102]. In this way the Ncut criterion implements both objectives: to minimise the between-cluster similarity, if $\mathrm{cut}(A, \bar{A})$ is small, and to maximise the within-cluster similarity, if $\mathrm{vol}(A)$ is large and $\mathrm{cut}(A, \bar{A})$ is small.

The mincut problem can be solved efficiently. However with the normalization term introduced by Ncut it becomes NP hard. Spectral clustering is a way to solve relaxed versions of this type of problems. Relaxing Ncut leads to the normalized spectral clustering algorithm. It can be shown [102] that finding a partition of a graph with n vertices into k clusters by minimizing Ncut, is equivalent to finding k indicator vectors $h_j = (h_{1j}, \dots, h_{nj})'$ with $j = 1, \dots, k$ of the form,

$$h_{ij} = \begin{cases} 1/\mathrm{vol}(A_j) & \text{if vertex } v_i \in A_j \\ 0 & \text{otherwise.} \end{cases} \tag{8.6}$$

In this way, the elements of the indicator vectors point out to which cluster belongs each graph vertex. This problem is still NP hard, but can be relaxed by allowing the elements of the indicator vectors to take, instead of two discrete values, any arbitrary value in $\mathbb{R}$. The solution to this relaxed problem corresponds to the first k generalized eigenvectors of

$$(D - W)\,u = \lambda\,D\,u, \tag{8.7}$$

where D is an n by n diagonal matrix with the degrees of the graph vertices $d_1, \dots, d_n$ on the diagonal, and $W = (w_{ij})_{i,j=1\dots n}$ is the matrix of graph weights.

The vectors u of the solution are real-valued due to the relaxation and should be transformed to discrete indicator vectors to obtain a partition of the graph. To do this, each eigenvalue can be used in turn to bipartition the graph recursively by finding the splitting point such that Ncut is minimized [99]. However, this heuristic may be too simple in some cases and most spectral clustering algorithms consider the coordinates of the eigenvectors as points in $\mathbb{R}^k$ and cluster them using an algorithm such as k-means [102]. The change of representation from the original data points to the eigenvector coordinates enhances the cluster structure of the data, so this last clustering step should be very simple if the original data contains well defined clusters. In the ideal case of completely separated clusters the eigenvectors are piecewise constant so all the points belonging to the same cluster are mapped to exactly the same point.

Finally, the algorithm can be summarized as [102],

```
input:  similarity matrix $S \in \mathbb{R}^{nxn}$,
        number of clusters k
steps:

   1. build a similarity graph using matrix $S$
   2. compute the unnormalized Laplacian of the graph $L = (D - W)$
   3. compute the first $k$ generalized eigenvectors of $(D - W)\,u = \lambda\,D\,u$
   4. consider the eigenvectors $u_1, \dots, u_k$ as columns of a matrix $U \in \mathbb{R}^{nxk}$
   5. consider the vectors $y_i \in \mathbb{R}^k$ $i = 1, \dots, n$ corresponding to the rows of $U$
   6. cluster the points $(y_i)$ in $\mathbb{R}^k$ with k-means into clusters $C_1, \dots, C_k$
output:  clusters $A_1, \dots, A_k$ / $A_i = \{j \mid y_j \in C_i\}$.
```

8.2.2 Pitch contours formation

In order to apply the spectral clustering algorithm to the formation of pitch contours several aspects must be defined. In particular, the construction of the graph involves deciding which vertices are connected and which are not. Then, a similarity function has to be designed such that it induces meaningful local neighbours. Finally, an effective strategy has to be adopted to estimate the number of clusters. In what follows, each of these issues are discussed and the proposed algorithm is described.

8.2.2.1 Graph construction

Constructing the similarity graph is not a trivial task and constitutes a key factor in spectral clustering performance. Different alternatives exist for the type of graph, such as k-nearest neighbor, ϵ-neighborhood or fully connected graphs, which behave rather differently. Unfortunately, barely any theoretical results are known to guide this choice and to select graph parameters [102]. A general criterion is that the resulting graph should be fully connected or at least should contain significantly fewer connected components than the clusters to be detected. Otherwise, the algorithm will trivially return connected components as clusters.

To include information on temporal proximity a local fixed neighborhood is defined, such that f_0 candidates at a certain time frame are connected only to candidates in their vicinity of a few frames (e.g. two neighbor frames on each side). In this way the graph is in principle fully connected, as can be seen in Figure 8.7, and the resulting connected components are determined by similarity between vertices. Two candidates distant in time may nevertheless belong to the same cluster by their similarity to intermediate peaks. Note that in Figure 8.7 only one neighbor frame on each side is taken into account to link peaks. In this case, if a peak is missing the given contour may be disconnected. For this reason, a local neighbourhood of two or three frames on each side is preferred. Similarity of not connected components is set to zero, so a sparse similarity matrix is obtained.

In addition, a contour should not contain more than one f_0 candidate per frame. To favour this, candidates in the same frame are not connected. Specifying cannot-link constraints of this type is a common approach for semi-supervised clustering [103]. However, this does not strictly prohibit two simultaneous peaks to be grouped in the same cluster if their similarity to neighbor candidates is high. For this reason, clusters should be further processed to detect this situation and select the most appropriate candidate in case of collisions.

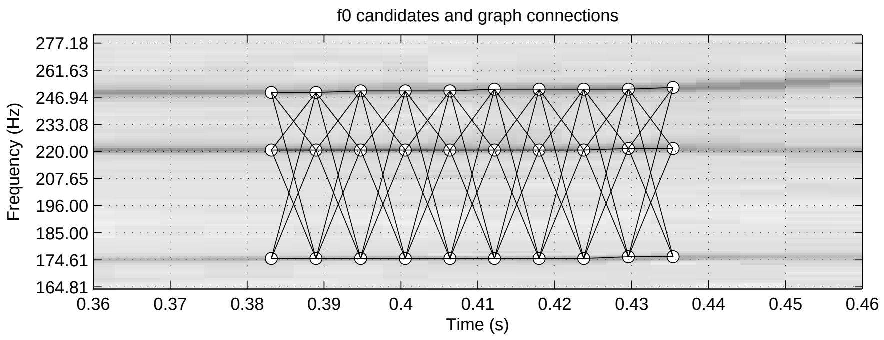


FIGURE 8.7: Graph connections considering only one neighbor frame on each side for an observation window of 10 frames. The resulting graph is fully connected.

8.2.2.2 Similarity measure

To define a similarity measure between F0gram peaks it is reasonable to base it on the assumption of slow variation of pitch contours in terms of fundamental frequency and pitch salience (as defined in section 8.1).

Fundamental frequency distance between two graph vertices v_i and v_j may be better expressed in a logarithmic scale, that is, as a fraction of semitones. To do this, the pitch value of a vertex is expressed as the corresponding index in the logarithmically spaced grid used for pitch salience computation.[2] Then, this can be converted to a similarity value $s_{f0}(v_i, v_j) \in (0, 1]$ using a Gaussian radial basis function

$$s_{f0}(v_i, v_j) = e^{-\frac{d^2_{f0}(v_i,v_j)}{\sigma^2_{f0}}} \tag{8.8}$$

where $d_{f0}(v_i, v_j) = |f0_i - f0_j|$ stands for pitch distance and σ_{f0} is a parameter to define the width of local neighborhoods. In a similar way, a similarity function can be defined to account for salience proximity. To combine both similarity functions they can be multiplied, as in [101].

Although this approach for combination was implemented and proved to work in several cases, the similarity measures have some shortcomings. Pitch based similarity is not able to discriminate contours that intersect. In this case, salience may be useful but it also has some drawbacks. For instance, points that are not so near in frequency and should be grouped apart, may be brought together by their salience similarity. This suggest the need for a more appropriate way of combining similarity values.

A significant performance improvement was obtained by combining the pitch value of the candidates and the chirp rates provided by the FChT. The chirp rate can be regarded as a local estimation of the pitch change rate. Thus, the pitch value of the next point in the contour can be predicted as $\overrightarrow{f0}^k_i = f0^k_i\,(1 + \alpha^k_i\,\Delta t)$, where $f0^k_i$ and α^k_i are the pitch and

[2]In which a 16th semitone division is used (192 points per octave).

chirp rate values, i and k are the candidate and frame indexes respectively, and Δt is the time interval between consecutive signal frames. Figure 8.8 depicts most prominent f0 candidates and their predictions for a short region of the example. Note that there are

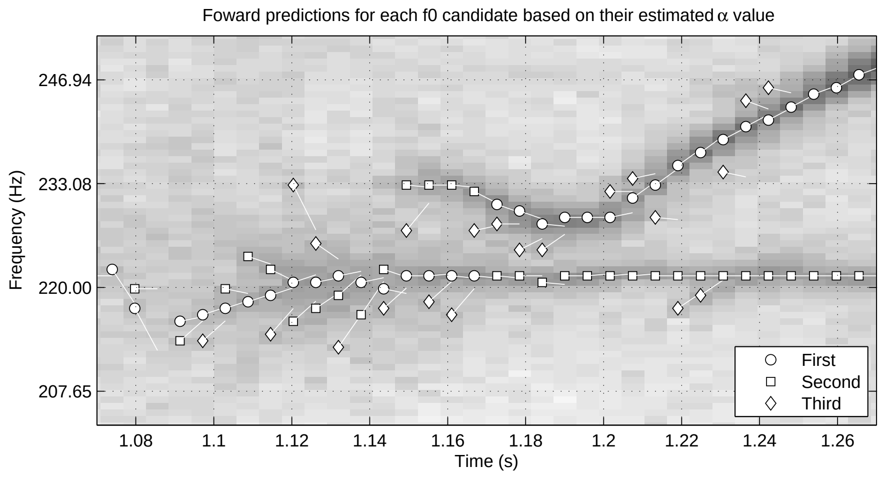


FIGURE 8.8: Forward predictions of the three most prominent f0 candidates for a short interval of the example. Although the clusters seem to emerge quite defined, spurious peaks may mislead the grouping. The situation can be improved if a backward prediction is also considered.

some spurious peaks in the vicinity of a true pitch contour whose estimate lie close to a member of the contour and can lead to an incorrect grouping. A more robust similarity measure can be obtained by combining mutual predictions between pitch candidates. This is done by computing for each candidate also a backward prediction $\overleftarrow{f0}$ in the same way as before. Then, distance among two candidates v_i^k and v_j^{k+1} is obtained by averaging distances between their actual pitch values and their mutual predictions,

$$d_{f0}(v_i^k, v_j^{k+1}) = \frac{1}{2}\Big[|f0_i^k - \overleftarrow{f0}_j^{k+1}| + |\overrightarrow{f0}_i^k - f0_j^{k+1}|\Big] \tag{8.9}$$

Using this mutual distance measure the similarity function is defined as in Equation (8.8). Additionally, the same reasoning can be extended to compute forward and backward predictions for two or three consecutive frames. These similarity values are used as graph weights for candidates in their temporal proximity.

Still remains to be set the value of σ_{f0}, which plays the role of determining the actual value assigned to points in the vicinity and to outlying points. Self tuning σ_{f0} for each pair of data points was tested based on the distance to the k-th nearest neighbor of each point, as proposed in [104]. This approach can handle clusters with different scales, but when applied to this particular problem it frequently groups noisy peaks far apart from each other. It turned out that, given the filiform shape of clusters that are to be detected, a fixed value for σ_{f0} was more effective. Since pitch predictions become less reliable as the time interval grows, a more restrictive value for σ_{f0} is used for measuring similarity to points at the second and third consecutive frames (reported results correspond to $\sigma_{f0}^1 = 0.8$, $\sigma_{f0}^2 = 0.4$).

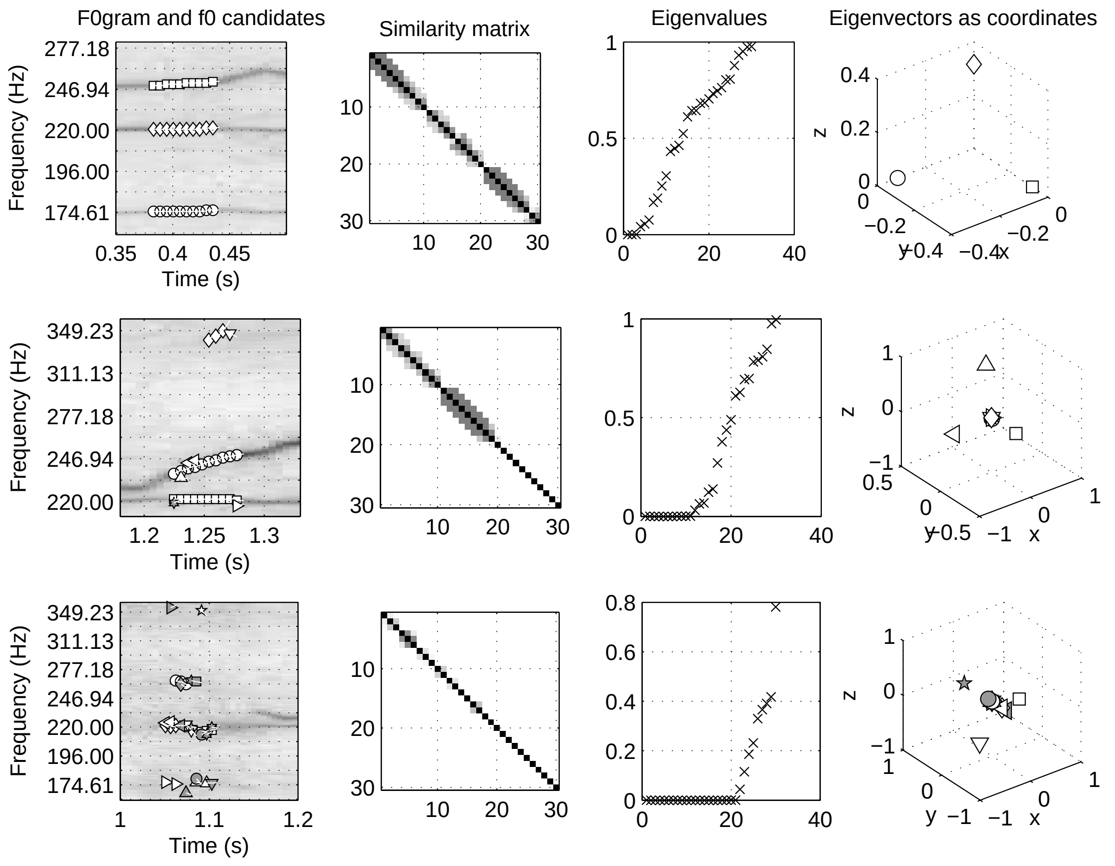


Figure 8.9: Local clustering of f0 candidates for three short time intervals of the audio example. Similarity matrix, eigenvalues and eigenvectors as coordinates are depicted. *First:* Three well-defined clusters can be identified in the data, as well as the corresponding bands in the similarity matrix. The multiplicity of eigenvalue zero coincides with the number of connected components. All members of a cluster are mapped to the same point in the transformed space. *Second:* Example with two true pitch contours and several spurious peaks. The two corresponding bands in the similarity matrix can be appreciated. Multiplicity of eigenvalue zero not only indicates the relevant connected components but also isolated points. The true contours are correctly identified by the algorithm and spurious peaks tend to be isolated. *Third:* Example where there are no harmonic sources present. Multiplicity of eigenvalue zero is high and almost all peaks are considered as a single cluster.

Figure 8.9 shows three different examples of the local clustering. The examples selected depict different situations to illustrate the behaviour of the technique, namely three well-defined clusters (one for each f0 candidate), two clusters and spurious peaks, and a region with no harmonic sources where clusters should not arise. An observation window of 10 signal frames is used and the three most prominent F0gram peaks are considered. A neighborhood of two frames on each side is used. Similarity matrix is sorted according to the detected clusters, producing a sparse band diagonal matrix, where clusters can be visually identified as continuous bands.

8.2.2.3 Determination of the number of clusters

Automatically determining the number of clusters is a difficult problem and several methods have been proposed for this task [103]. A method devised for spectral clustering is the eigengap heuristic [102]. The goal is to choose the number L such that all eigenvalues $\lambda_1, \ldots, \lambda_L$ are very small, but λ_{L+1} is relatively large. Among the various justifications for this procedure, it can be noticed that in the ideal case of L completely disconnected components, the Laplacian[3] graph has as many eigenvalues zero as there are connected components, and then there is a gap to the next eigenvalue.

This heuristic was implemented, but it sometimes failed to detect the correct number of cluster (e.g. when clusters are not so clear there is no well-defined gap). The following iterative strategy gave better results. It consist in firstly estimating the number of connected components using the multiplicity of eigenvalue zero by means of a restrictive threshold. Then, the compactness of the obtained clusters is evaluated. To do this, different measures were tested and a threshold on the sum of distances to the centroid in the transformed space was selected. As mentioned before, in case of completely separeted connected components all members of the same cluster are mapped to a single point in the transformed space. For this reason, the detection of poor quality clusters showed not to be too sensitive to the actual value used for thresholding. Each of the not compact clusters is further divided until all the obtained clusters conform to the threshold. This is done repeatedly by running k-means only to points in the cluster, starting with $k = 2$ for a bipartition and incrementing the number of desired clusters until the stop condition is met. This strategy tends to isolate each spurious peak as a single cluster (see Figure 8.9), what in turn favours to ignore them in the formation of pitch contours.

8.2.2.4 Filtering simultaneous members

Despite the introduction of cannot-link constraints some clusters can occasionally contain more than one member at the same time instant. The best f0 candidate can be selected based on pitch distance to their neighbors. This approach was explored but difficulties were encountered for some particular cases. For instance, when a contour gradually vanishes F0gram peaks are less prominent, their pitch change rate estimate is less reliable and spurious peaks appear in the nearby region. Therefore, under the assumption of slow variation of contour parameters, salience similarity was introduced as another source of information. To do this, the most prominent peak of the cluster is identified and the cluster is traversed in time from this point in both directions, selecting those candidates whose salience is closest to the already validated neighbors.

[3]Which was defined in the algorithm formulation as $L = (D - W)$ (see section 8.2.1).

8.2.2.5 Formation of pitch contours

The above described local clustering of f0 candidates has to be extended to form pitch contours. Increasing the length of the observation window showed not to be the most appropriate option. The complexity of the clustering is increased for longer windows, since a higher number of clusters inevitably arise mainly because of spurious peaks. Additionally, the computational burden grows exponentially with the number of graph vertices. Thus, an observation window of 10 signal frames was used in the reported simulations ($\sim$ 60 ms) as a good balance of the trade-off.

Performing a local grouping makes it necessary to link consecutive clusters for pitch contours continuation. Neighboring clusters in time can be identified based on the similarity among their members. A straightforward way to to this is by performing local clustering on overlapped observation windows and then grouping clusters that share elements. Figure 8.10 shows the clustering obtained using half overlapped observation windows for the two previously introduced examples which contain valid contours.

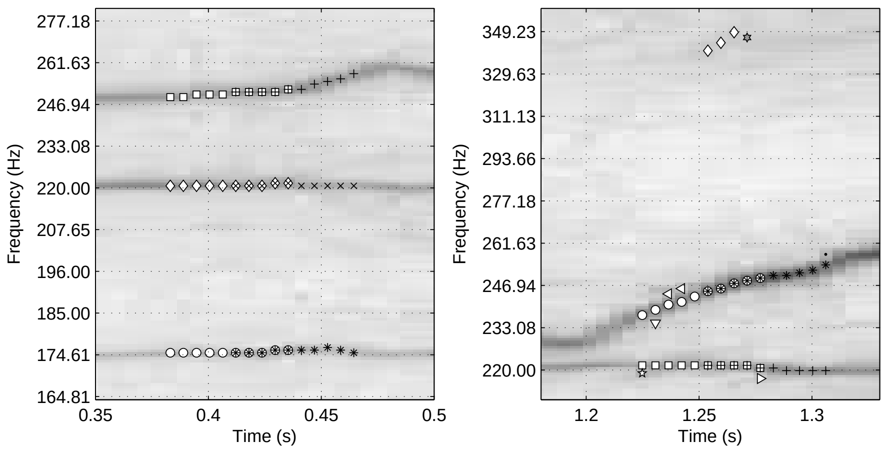


FIGURE 8.10: Examples of clustering using half overlapped observation windows. The pitch contours are correctly continued since several of their members are shared.

8.2.3 Evaluation and results

The contours obtained by applying the proposed algorithm to the audio excerpt example are depicted in Figure 8.11. The three most prominent peaks of the F0gram are considered for pitch tracking. Several issues can be noted from these results. Firstly, the main contours present are correctly identified, without the appearance of spurious detections when no harmonic sound is present (e.g. around $t = 1.0$ s). The example shows that many sound sources can be tracked simultaneously with this approach. No assumption is made on the number of simultaneous sources, which is only limited by the number of pitch candidates considered. The total number of contours and concurrent voices at each time interval is derived from the data.

It can also be seen that the third voice of the second note (approximately in the range $t = 1.0 - 2.0$ s) is only partially identified by two discontinued segments. Because of the low prominence of this contour some of the pitch candidates appear as secondary peaks of the more prominent sources instead of representing the third voice. This situation can be improved by increasing the number of prominent peaks considered, as depicted in Figure 8.12 where ten pitch candidates where used for contour computation.

Apart from that, there are the three short length contours detected in the interval $t = 2.1 - 2.5$ s that seem to be spurious. However, when carefully inspecting the audio file it turned out that they correspond to harmonic sounds from the accompaniment, that stand out when singing voices remain silent. Although these contours have a very low salience they are validated because of their structure and the absence of prominence constrains. It will depend on the particular problem in which the algorithm is used if these contours may be better filtered out based on salience.

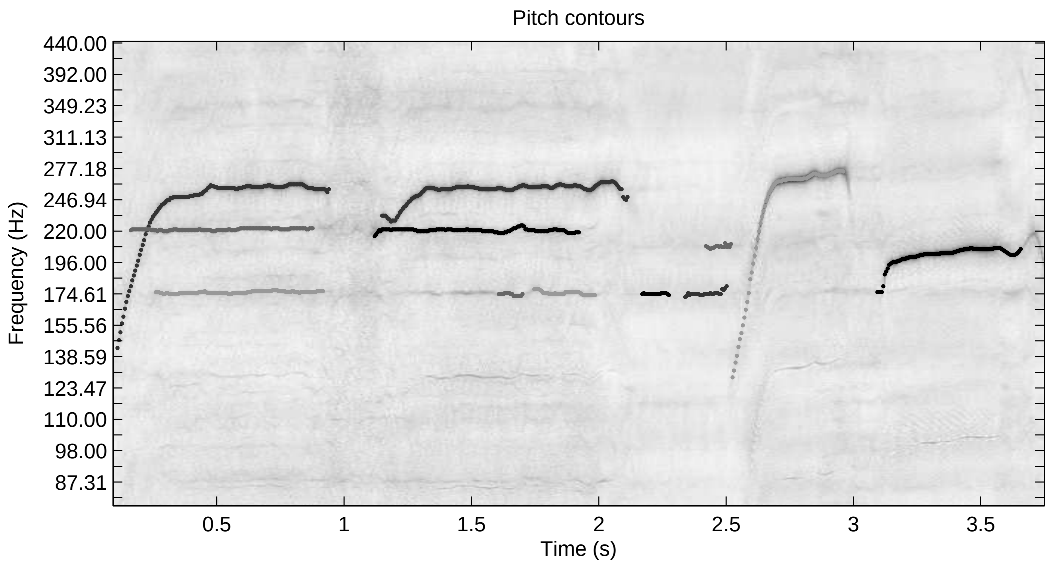


FIGURE 8.11: Pitch contours for the audio example obtained by considering the three most prominent F0gram peaks.

A melody detection evaluation was conducted following a procedure similar to the one applied in [74]. The vocal files of the 2004-2005 MIREX melody extraction test set, which is a publicly labeled database available from `http://www.music-ir.org/mirex/`, were considered. It comprises 21 music excerpts for a total duration of 8 minutes.

The three most prominent F0gram peaks were selected as pitch candidates to form contours using the herein described algorithm. All the identified pitch contours were considered as main melody candidates and the ones that better match the labels were used to assess performance. Only those frames for which the melody was present according to the labels were taken into account to compute the following evaluation measure

$$\text{score}(f_0) = \min\{1, \max\{0, (\text{tol}_\text{M} - \Delta f_0)/(\text{tol}_\text{M} - \text{tol}_\text{m})\}\},$$

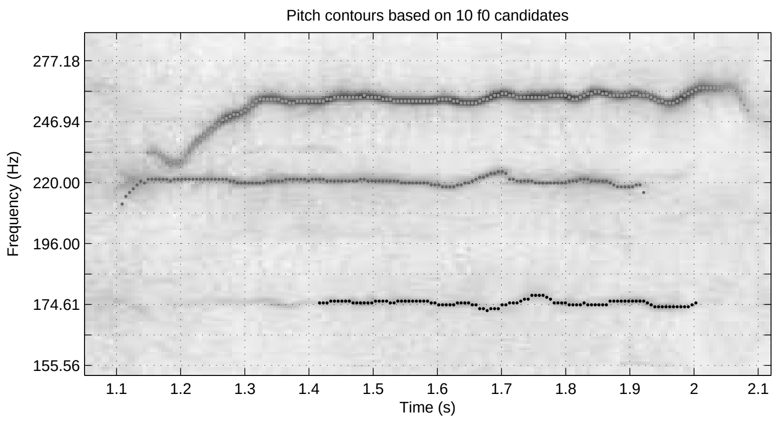


FIGURE 8.12: Results obtained when using ten f0 candidates to form pitch contours instead of only three as in Figure 8.11. Note that the third voice contour is better identified, though the initial portion is still missing.

where $\Delta f_0 = 100|f_0 - f_0^{gt}|/f_0^{gt}$ is the relative error between the pitch contour value and the ground truth, and the tolerances $\mathrm{tol_M}$ and $\mathrm{tol_m}$ correspond to 3% and 1% respectively. This represents a strict soft thresholding.

The performance obtained in this way is compared to an equivalent evaluation that considers F0gram peaks as main melody estimates without performing any type of grouping into contours (as reported in [74]). Grouping the F0gram peaks into contours involves the determination of the onset time and duration of each contour, necessarily leaving some time intervals without melody estimation. This is avoided when isolated F0gram peaks are considered as main melody estimates, since for every melody labeled frame there is always a pitch estimation. Therefore, this performance measure can be considered as a best case.

Results of the evaluation are presented in Table 8.1. Two different values are reported for the pitch contours formation corresponding to a single run of the k-means algorithm and 10 repetitions. When the clusters in the transformed space are not well defined the k-means algorithm can get stuck in a local minima. This situation can be improved if several executions are performed but with different sets of initial cluster centroid positions and the best performing solution is returned (i.e. lowest centroid distances). It can be noticed that the k-means repetition consistently gives a slight performance increase.

In addition, precision and recall values are reported. Precision is computed as the mean score value of the estimations within the 3% threshold. Remaining frames are considered not recalled items, as well as melody labeled frames for which there is no pitch contour.

When visually inspecting the results for individual files it turned out that most melody labeled regions for which there were no estimated contours correspond to low salience portions of the F0gram (for instance, when a note vanishes). It seems that labels are produced from monophonic files containing only the vocal melody and when mixed into a polyphonic track some regions are masked by the accompaniment. Figure 8.13 shows a detail of the current example where this situation can be appreciated. In order to take this into account the evaluation was repeated but ignoring low prominent melody frames. To do this a salience estimation was obtained for each labeled frame by interpolating the F0gram values. Then a global threshold was applied to discard those frames whose salience was below 30% of the F0gram maximum value (26% of the total frames).

TABLE 8.1: Results for the melody detection evaluation. The pitch contours are obtained from the three most prominent f0 candidates. An evaluation using F0gram peaks (1st to 3rd) without tracking is also reported.

F0gram	no salience threshold			30% salience threshold		
peaks	score	precision	recall	score	precision	recall
1	83.38	96.93	86.03	97.22	99.01	98.19
1-2	88.24	97.59	90.42	99.20	99.59	99.61
1-3	90.33	97.91	92.26	99.61	99.74	99.87
Pitch	no salience threshold			30% salience threshold		
contours	score	precision	recall	score	precision	recall
1 k-means	81.99	90.37	84.77	96.69	98.27	97.63
10 k-means	83.21	90.38	85.53	97.20	98.40	98.15
frames	100%			74%		

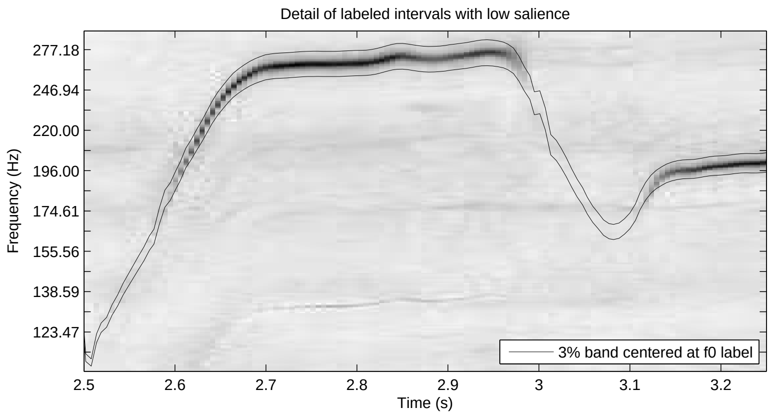


FIGURE 8.13: Some melody labeled regions of the example exhibit a very low salience (2.5-2.6 and 3.0-3.1 s).

The performance of the pitch contours formation by itself is quite encouraging. However, it decreases considerably compared to the values obtained before grouping F0gram peaks. The gap is reduced by restricting the evaluation to the most prominent peaks, which seems to confirm that low salience regions are troublesome for the algorithm. Visually inspecting the estimations for individual files gives the idea that most pitch contours are correctly identified. However, the evaluation results indicate the algorithm seems not to take full advantage of the information given by the F0gram peaks. Blindly relying on estimated α values no matter their corresponding salience is probably the most important shortcoming of the proposed algorithm.

8.2.4 Discussion and conclusions

In this chapter a novel way of performing pitch tracking in polyphonic audio was described. The technique is based on a local pitch estimation, called F0gram, that makes use of the Fan Chirp Transform. This representation aims at capturing pitch fluctuations and is particularly suited for the singing voice. Then contours are constructed by clustering local f0 estimates. The grouping is performed by applying a Spectral Clustering method since it can handle filiform shapes such as pitch contours. The proposed similarity measure takes advantage of the pitch change rate estimate provided by the FChT based F0gram. The determination of the number of clusters is tackled by an iterative approach, where the number of connected components is taken as an initial estimate and not compact enough clusters are further divided into an increasing number of groups. This strategy tends to isolate each spurious peak in single clusters, which are then ignored in the formation of pitch contours. Clustering is carried out for overlapped observation windows of a few hundred milliseconds and clusters from different time windows are linked if they share elements. In this way, groups that exhibit a coherent geometric structure emerge as pitch contours while the others are discarded.

The clustering approach to the tracking problem seems appealing because the solution involves the joint optimization of all the pitch contours present in a given time interval. Therefore, many sound sources can be tracked simultaneously and the number of contours and simultaneous sources can be automatically derived from the data. This differs from most classical multiple f0 tracking techniques in which each source is tracked in turn. In addition, the algorithm is unsupervised and relies on a few set of parameters. The influence of each parameter has not been fully assessed and the determination of optimal values should be tackled in future work. Preliminary results indicate that performance is not too sensitive to a particular setup configuration of some of them (e.g. number of candidates, k-means repetitions), but as it would be expected the values for σ_{f0} have to be set with more care. It is important to notice that the algorithm has low computational cost given that efficient algorithms exist for solving generalized eigenvector problems as well as for the k-means step [105]. Results of a melody detection evaluation indicate the introduced technique is promising for pitch tracking and can effectively distinguish most singing voice pitch contours. There is still some room for improvement.In particular, other sources of information should be included in the similarity measure in order to take full advantage of the local pitch candidates. The estimation of the pitch change rate is less reliable for low salience peaks. This could be taken into account when computing similarity, for example by adjusting the σ_{f0} value in accordance with the salience of the candidate.

9 Harmonic sounds extraction and classification

Thus far, the early stages of the alternative approach proposed for singing voice detection have been described. In particular, how the audio signal is analysed using the Fan Chirp Transform and the way in which the harmonic sounds present are pitch tracked. This chapter focuses on the remaining stages of the method. Firstly, the process of extracting the sounds corresponding to each of the identified pitch contours from the polyphonic audio mixture is presented. Then it describes and analyses the features that are computed from the extracted signals and how they are classified as being singing voice or not.

9.1 Sound source extraction

Recall from Chapters 7 and 8, that the FChT analysis of a signal frame produces a precise spectral representation in which the energy of a harmonic source is very concentrated, in the case that the chirp rate α closely matches the true pitch rate of the source. This was illustrated for synthetic harmonic sources in Chapter 7 (see Figures 7.7 to 7.11). In polyphonic music audio signals the accurate TF representation of pitched sounds can be exploited to separate harmonic sources from the audio mixture. A comparison of the FChT and the DFT spectra for a signal frame of polyphonic music is presented in Figure 9.1. The audio file is *pop1.wav* from the MIREX melody extraction test set, (which was already used in Chapters 7 and 8) and serves as an example throughout this section to illustrate the sound source extraction process. The α value for the FChT analysis is tuned for the pitch change rate of the most prominent singing voice. The ideal location of harmonic positions for this source is indicated with vertical lines. As can be noticed, the FChT spectrum shows a much concentrated representation than the DFT for the prominent singing voice.

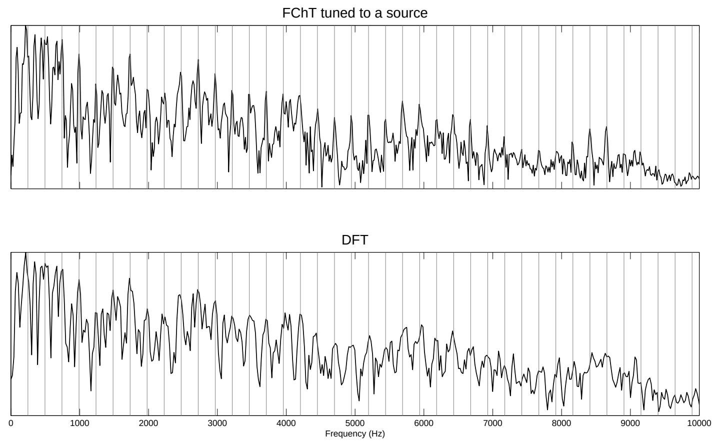


FIGURE 9.1: Comparison of the FChT and DFT spectra of a signal frame of polyphonic music. The α value is selected to match the pitch change rate of the most prominent sound source which is a singing voice. The location of the ideal harmonic positions is indicated with vertical lines.

The filtering process applied to extract a harmonic source is performed in the FChT spectral domain and is described in the following. Firstly, the harmonic ideal frequencies are computed, i.e. $f_n = nf_0,\ n = 1 \ldots N$, based on the fundamental frequency value f_0, which is known since the source has been pitch tracked. The maximum harmonic frequency f_N and its corresponding harmonic number N is limited by the analysis bandwidth that is set to 10 kHz for the FChT computation. Then the closest frequency bin k_n of the FChT is identified for each harmonic frequency f_n, such that $k_n = \arg(\ \min\{\text{abs}(f_i - f_n)\})$, where f_i corresponds to each analysis frequency. The filtering can be done straightforwardly by constructing an FChT spectrum retaining only those k_n frequency bins from the original spectrum, followed by inverse FFT to obtain the time domain signal $\breve{x}(t)$. However, since the frequency of the harmonic may not perfectly coincide with the frequency of the bin and given that a Hann window was applied to the warped signal frame (see equation 7.8), the energy of the frequency component is slightly spread over the neighbouring bins. For this reason, the filtered FChT spectrum is constructed by considering the k_n frequency bin plus one additional bin on each side. That is, for each harmonic frequency f_n three bins are retained, $\{k_n$-1, k_n, k_n+1$\}$. Notice that the filtering could also be done by estimating the frequency, amplitude and phase of each spectral harmonic component by means of sinusoidal modeling techniques [106] and synthesizing the time domain signal by directly evaluating complex exponentials. However, the above described method was preferred for simplicity and efficiency.

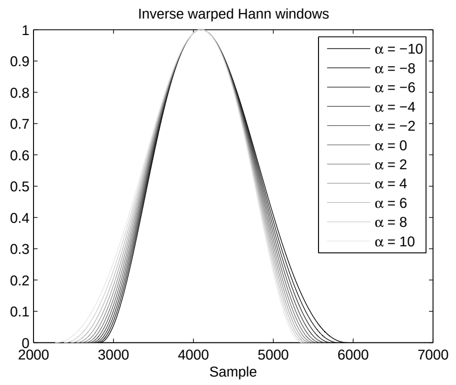


FIGURE 9.2: Inverse warped Hann windows for different α values.

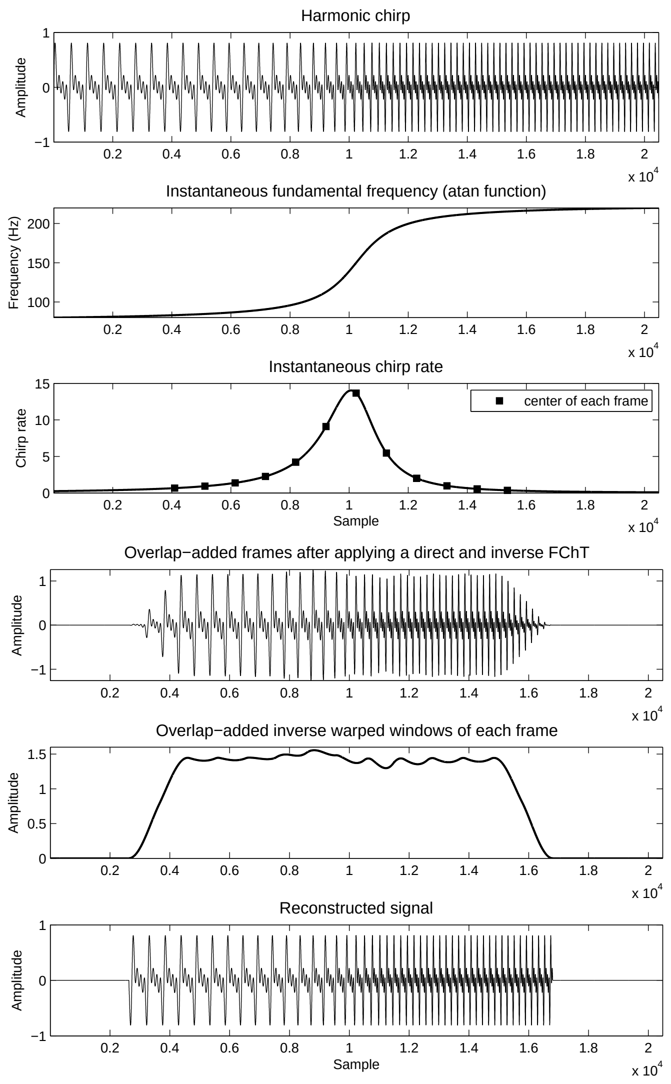


FIGURE 9.3: Overlap add process for the forward and inverse FChT.

Given that the FChT spectrum of the signal $x(t)$ is obtained by computing the FFT of the time warped signal $\breve{x}(t) = x\left(\phi_{\alpha}^{-1}(t)\right)$, where $\phi_{\alpha}^{-1}(t)$ was defined in equation 7.9, then the time domain signal yielded by the filtering process after the inverse FFT is a time warped signal. Therefore, the next step is to apply the inverse warping to the signal frame using the warping function $\phi_\alpha(t)$ (introduced in equation 7.5),

$$\phi_\alpha(t) = (1 + \frac{1}{2}\,\alpha\, t)\, t$$

where the α value is known since it was determined in the FChT analysis. Regarding the discrete time implementation it is important to notice the following. Recall from section 7.3.2 that the finite length discrete signal frame $\breve{x}[m]$ is known in an equally spaced grid of time instants $\breve{t}_m$ in the warped time domain. The inverse warping is implemented by non-uniform resampling of $\breve{x}[m]$. An equally-spaced grid of time instants t_n is constructed in the unwarped time domain, and the corresponding time instants in the warped time domain are determined according to $\breve{t}_n = \phi_\alpha(t_n)$. Thus, the discrete signal $\breve{x}[m]$ has to be evaluated at time instants $\breve{t}_n$. As time instants $\breve{t}_n$ may not coincide with sampling times $\breve{t}_m$, the evaluation is performed using an interpolation technique. A linear interpolation is applied in the current implementation since it is computationally efficient and yields sufficiently accurate results. However, the approximation error can be reduced by using a better interpolation technique (e.g. splines).

Another issue that has to be taken into account is the inverse warping of the time limited window $w(t)$ used in the FChT analysis. Since the windowing process is applied after the time warping in the FChT computation, when performing the inverse warping of a windowed signal frame the resulting window, i.e. the amplitude envelope, will be distorted. Moreover, the type of distortion is different for each chirp rate α value, given that the inverse warping function is different. This is illustrated in Figure 9.2 for the Hann window. For this reason, when performing the classical overlap-add method for reconstructing a sliding window processed signal the amplitude envelope differs from the original one. This situation is depicted in Figure 9.3, for a harmonic chirp of constant amplitude whose fundamental frequency varies from 80 to 220 Hz according to the arctan function. In order to eliminate the unwanted amplitude variations the inverse warped window functions are computed for each signal frame. Then, they are combined using overlap-add to obtain the resulting global amplitude envelope. Finally, this envelope is used for inverse weighting the signal thus yielding the correct amplitude values, as shown in Figure 9.3.

The source extraction process is illustrated in Figure 9.4 for the most prominent singing voice of the *pop1.wav* audio file from the MIREX melody extraction test set. The fundamental frequency and the chirp rates are derived from the available labels. This is done to assess the source filtering process without the influence of errors that could be introduced by the pitch tracking algorithm. The residual is obtained by subtracting in the time-domain the reconstructed main singing voice from the original audio mixture.

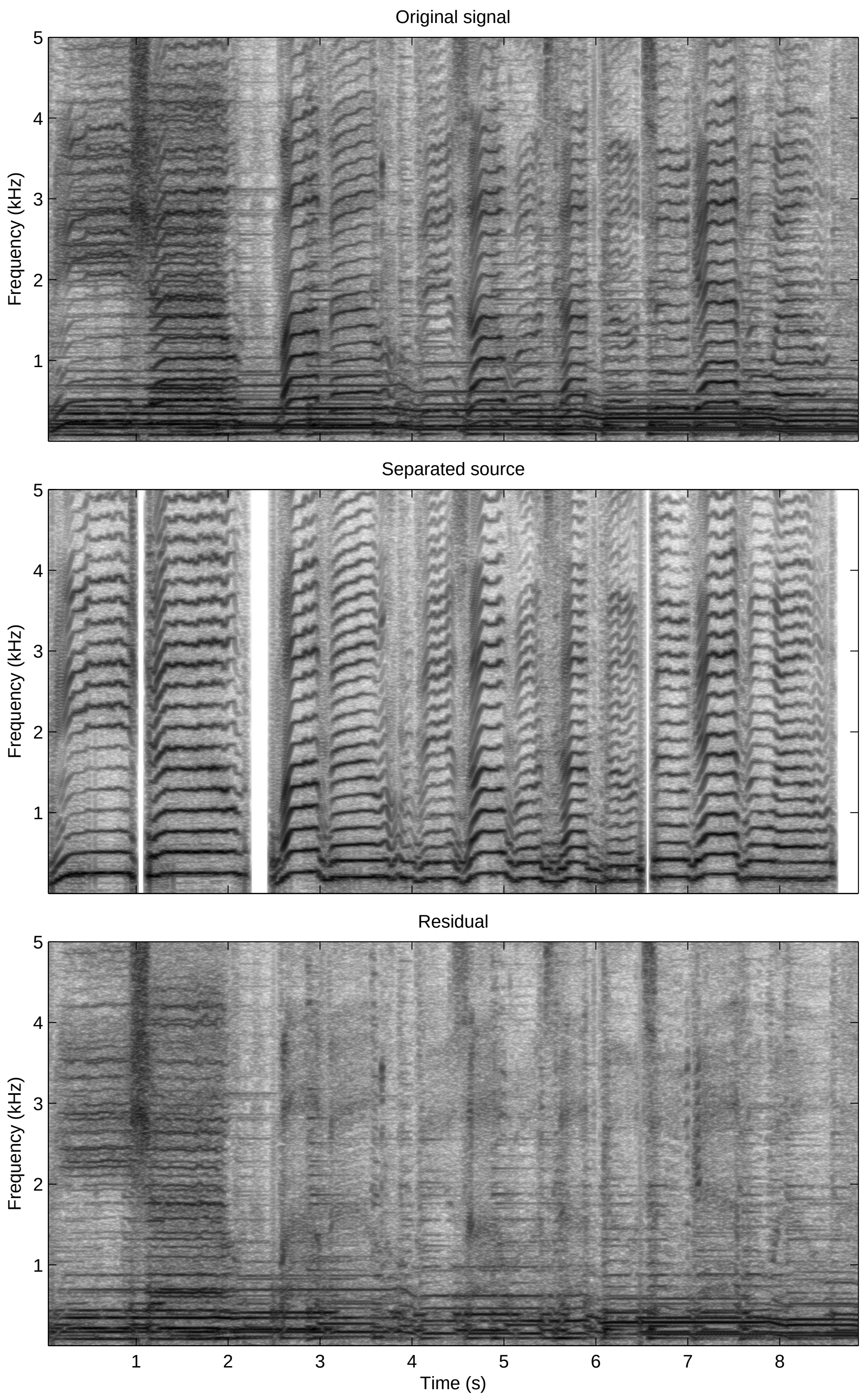


FIGURE 9.4: STFT-based spectrograms of a source separation example for an excerpt of the *pop1.wav* audio file from the MIREX melody extraction test set. The fundamental frequencies and the chirp rates for the analysis are derived from the available labels.

9.2 Features from extracted signals

Once the sounds are isolated from the audio mixture different features are computed in order to grasp those distinctive characteristics that may enable singing voice discrimination. The Mel Frequency Cepstral Coefficients (MFCC, see section 4.2.1) are considered, due to their ability to describe spectral power distribution and because they yielded the best results on the study presented in the first part of this dissertation. Besides, some acoustic features are proposed to describe continuous variations of pitch frequently found in a vocal performance.

It is important to notice that an implicit feature which characterizes extracted signals is its harmonicity, given the pitch computation and sound extraction process. Additionally, as it has been previously noted (see section 8.1.6), the pitch salience function favours harmonic sounds with a relatively high number of partials. This tends to highlight singing voice sounds which typically have a lot of prominent harmonics, over harmonic musical instruments whose spectral envelope may have a steeper decaying slope (e.g. a bass).

9.2.1 Mel Frequency Cepstral Coefficients

MFCC features are obtained in a similar way as described in section 4.2.1, but applied to the extracted sound instead of the audio mixture. In this way, they are supposed to better described spectral power distribution of individual sound sources in comparison to the global spectral characterization they provide when considering the original audio.

The parameters are the same as used in the first part of this dissertation. Signal frame length is set to 25 ms, using a Hamming window and the hop size is 10 ms. For each signal frame 40 mel scale frequency bands are used, from which 13 coefficients are obtained. The only difference is the analysis bandwidth which is set to 10 kHz (instead of 16 kHz) because this is the limit used for the FChT analysis.

Temporal integration of coefficients is also done in an early integration fashion, by computing statistical measures (median and standard deviation) of the frame-based coefficients within the whole pitch contour. Based on the results of the first part of this dissertation, first order derivatives (Δ, equation 4.1) are also included.

9.2.2 Pitch related features

A singer is able to vary the pitch of his voice continuously, within a note and during a transition between different pitches. In a musical piece, pitch variations are used by the performer to convey different expressive intentions and to stand out from the accompaniment. This is by no means an exclusive feature of the singing voice, since there are many other musical instruments capable of the same behaviour. However, in a typical music performance where singing voice takes part as a leading instrument, for instance a popular song, continuous modulations of its fundamental frequency are of common use. In addition, the accompaniment frequently comprises fixed-pitch musical instruments, such as piano or fretted strings. Thus, low-frequency variations of a pitch contour are considered as an indication of singing voice. Nevertheless, since other musical instruments can also produce such modulations, this feature shall be combined with other sources of information for proper detection.

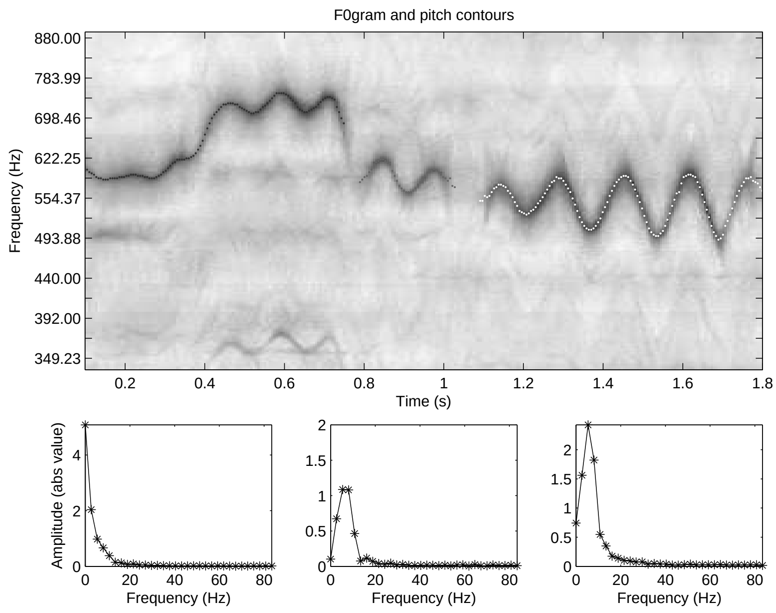


FIGURE 9.5: Example of vocal notes with vibrato and low frequency modulation. The audio file is an extract from the *opera-fem4.wav* file from the MIREX melody extraction test set. The summary spectrum $c[\hat{k}]$ is depicted at the bottom for each contour.

One of the most notable expressive vocal features is *vibrato*, which consist in a periodic modulation of pitch at a rate of around 6 Hz [107] (see Figure 9.5 for an example). It was previously used as a clue to detect singing voice, with a relative success [24, 37]. However, it is an expressive device not always found, being present mainly in the steady state of sustained sounds. Other pitch fluctuations take place in a typical vocal performance, such as continuous glissando or portamento, which is a slide between two pitches. In [108] different common variations in singing voice pitch are studied, in order to translate the continuous output of a pitch tracker (micro-intonation) to the actual sequence of notes performed (macro-intonation). Apart from the above mentioned pitch variations, others are identified, such as a device called *spike*, which is a monotonically increase followed by a monotonically decrease of pitch, often encountered in short note repetitions or as ornamental notes [108].

In order to describe the pitch variations, the contour is regarded as a time dependent signal and the following procedure is applied. The pitch values f_0 are represented in a logarithmic scale, that is a fraction of semitones (using the same 16th semitone division

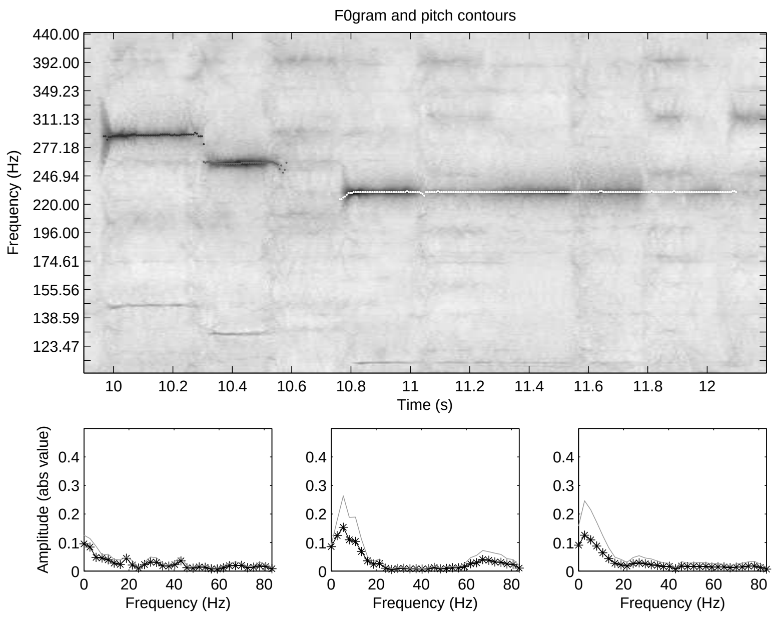


FIGURE 9.6: Example of saxophone notes without low frequency modulation. The audio file is an excerpt from the file *jazz1.wav* from the MIREX melody extraction test set. The summary spectrum $\hat{c[k]}$ is depicted at the bottom for each contour. The maximum spectrum $c[k]_{\max}$ is also shown in solid line. Note that for the second and third notes it is higher than $\hat{c[k]}$ (since both contours exhibit some fluctuations at the beginning or at the end) and it is attenuated by the median spectrum $c[k]_{\text{med}}$.

grid used for pitch tracking, see section 8.2.2.2). After removing the DC component by subtracting the mean value, the contour is processed by a sliding window and a spectral analysis is applied to each signal frame. The spectral analysis is performed by the Discrete Cosine Transform (DCT), according to,

$$c[k] = w[k] \sum_{n=1}^{N} f_0[n] \cos \frac{\pi(2n-1)(k-1)}{2N}, \quad k = 1 \dots N, \quad w[k] = \begin{cases} \frac{1}{\sqrt{N}} & k = 1 \\ \sqrt{\frac{2}{N}} & 2 \leq k \leq N \end{cases}$$

Very similar results are obtained using the DFT, though the DCT is preferred since it allows for the analysis of components at intermediate DFT frequency bins. The frame length is set to $\sim$ 180 ms which corresponds to $N = 32$ time samples, so as to have at least two frequency components in the range of a typical vibrato. The frames are highly overlapped, using a hop size of $\sim$ 20 ms, i.e. 4 samples.

After analysing all the frames, the $c[k]^i$ coefficients are summed up in a single spectrum $\tilde{c}[k]$ as follows. Since we are interested in high values in low frequency, the maximum absolute value for each frequency bin is taken, namely $\hat{c}[k]$. However, it was observed that this over estimates frames with high low energy values that occasionally arise in noisy contours due to tracking errors. Thus, the median of the absolute value of each frequency bin $\bar{c}[k]$ is also considered and both spectra are combined as

$$\tilde{c}[k] = \frac{\hat{c}[k] + \bar{c}[k]}{2}, \quad \text{where} \begin{cases} \hat{c}[k] = \max\limits_i \{|c[k]^i|\} \\ \bar{c}[k] = \operatorname*{median}\limits_i \{|c[k]^i|\}. \end{cases}$$

Examples of the behaviour of the pitch fluctuations analysis are provided in Figures 9.5 to 9.8. It is important to notice that Figure 9.8 is a counterexample of the proposed features, because a non-vocal instrument exhibits low frequency modulations. Then, two features are derived from this spectrum. The low frequency power (LFP) is computed as the sum of absolute values up to 20 Hz ($k = k_L$). Since well-behaved pitch contours do not exhibit prominent components in the high frequency range, a low to high frequency power ratio is considered (PR), which tries to exploit this property,

$$\text{LFP} = \sum_{k=1}^{k_L} \tilde{c}[k], \qquad \text{PR} = \frac{\text{LFP}}{\sum_{k_L+1}^{N} \tilde{c}[k]}. \tag{9.1}$$

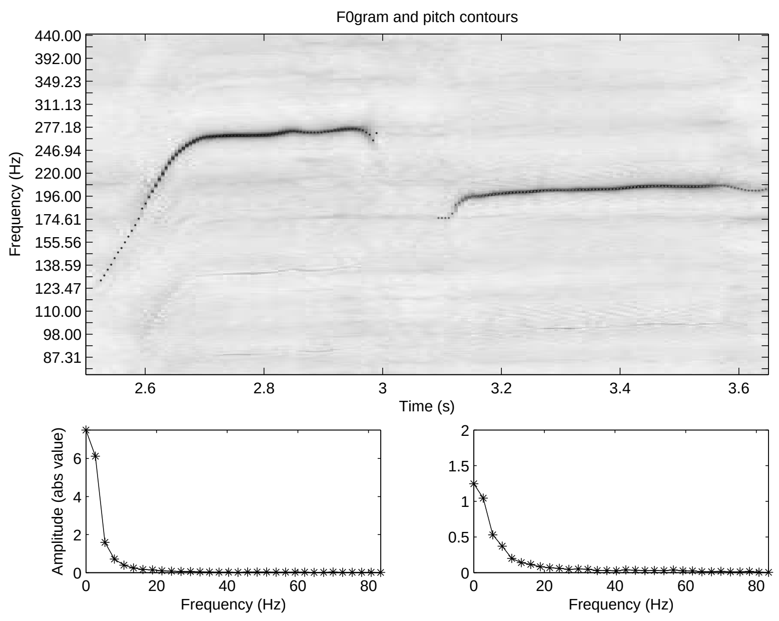


FIGURE 9.7: Example of two vocal notes without vibrato but with low frequency modulation. The audio file is an excerpt from the file *pop1.wav* from the MIREX melody extraction test set. The summary spectrum $c\hat{[k]}$ is depicted at the bottom for each contour.

In addition, two pitch-related features are also considered apart from the above described. One of them is simply the extent of pitch variation, which is computed as,

$$\Delta f_0 = \max(f_0[n]) - \min(f_0[n]),$$

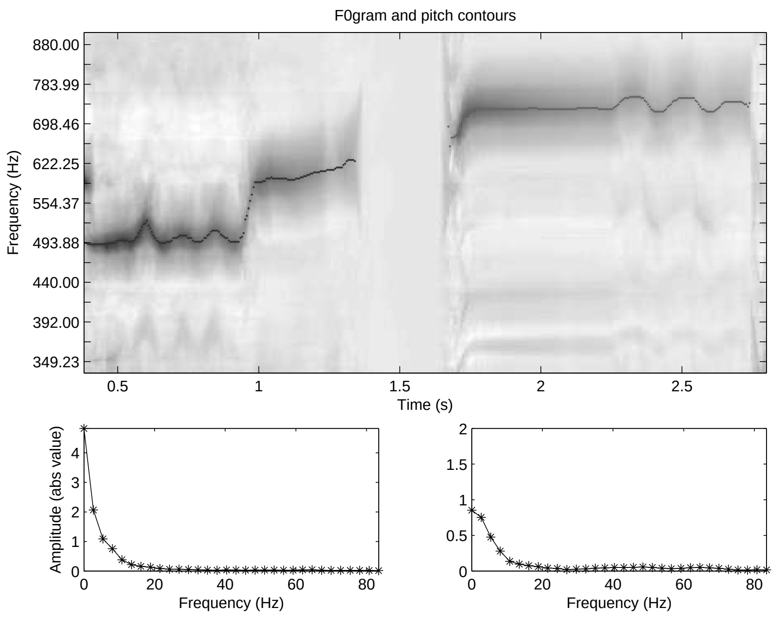


FIGURE 9.8: Example of two guitar notes of an audio file from the training database described in section 9.3.1. It serves as a counterexample of the proposed features, since a non-vocal instrument exhibit low frequency modulations. Pitch modulation descriptors must be combined with other source of information for proper classification.

where $f_0[n]$ are the pitch values of the contour (frequency values in a logarithmic scale). The other is the mean value of pitch salience along the contour, that is,

$$\text{Salience} = \text{mean}\{\rho(f_0[n])\}.$$

This gives an indication of the prominence of the sound source, but it also includes some additional information. As previously noted, pitch salience computation favours harmonic sounds with high number of harmonics, such as the singing voice. Besides, following [74], a *pitch preference* weighting function was introduced in salience computation to highlight most probable values for a singing voice. To do this, salience is weighted by a Gaussian function centered at MIDI note 60 (C4) and with a standard deviation of an octave and a half. These values were selected considering the singing voice main melody pitch distribution of two labeled databases: the singing voice files of the MIREX melody extraction test set and the RWC Popular Music database [98]. To favour generalization the standard deviation was tripled, as shown in Figure 9.9.

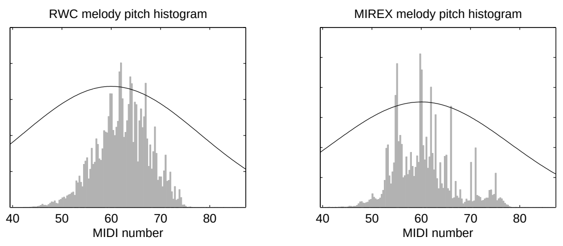


FIGURE 9.9: Pitch preference function (mean = 60, stdev = 18) and singing voice melody pitch histogram for RWC Popular and MIREX melody extraction test data.

9.3 Classification methods

9.3.1 Training database

A database of isolated vocal and non-vocal sounds has to be used for training the classifier. Initially the training dataset constructed for the first part of this dissertation was considered for this purpose, i.e. the 1000 one second length audio segments. The idea was to apply the source extraction front-end to each audio segment and to build the training database with the obtained sounds. This would be desirable because it enables a comparison of the singing voice detection approaches using the same training data. However, some problems were encountered. First of all, given that the music fragments are polyphonic audio mixtures, vocal labeled instances may contain several harmonic pitch contours, some of which could be non-vocal sounds from the accompaniment. For this reason, after applying the source extraction front-end to each of these vocal audio segments the resulting sounds have to be aurally inspected to decide if they shall be classified as vocal or not, which is time consuming. This was attempted, notwithstanding, but a second problem was encountered. The short length of the audio segments constrains the performance of the source extraction front-end. For instance when a pitch contour is only present in a relatively small portion of the fragment (e.g. due to the presence of unvoiced sounds) the pitch tracking sometimes fails. Therefore, a rather significant number of the training examples should be inevitably discarded, preventing the comparison of the approaches with the same training patterns. All of this motivated the construction of a new training database.

An interesting property of the sound source extraction classification approach is that monophonic audio clips can be used for training, that is music in which either a single singing voice or other musical instrument takes part. This constitutes an important advantage over the classical polyphonic audio approach. There is a lot of monophonic music material available, from a cappella and instrumental solo performances [98], to musical instruments databases [109, 110] and multi-track recordings [111]. Moreover, collecting such a database requires much less effort than manually labeling songs.

Following the above approach an audio training database was built based on more than 2000 audio files, comprising singing voice on the one hand and typical musical instruments found in popular music on the other. Instead of musical instruments databases, in which each note in the chromatic scale is recorded along the whole register of the instrument, more natural performances where preferred such as a cappella music, multi-track recordings, and short commercial audio clips intended for video and music production.[1] This is because musical instruments databases recorded note by note usually lack the expressive nuances found in a real music performance, for instance some of the typical pitch fluctuations previously described.

[1]Such as the ones found at http://www.producerloops.com/.

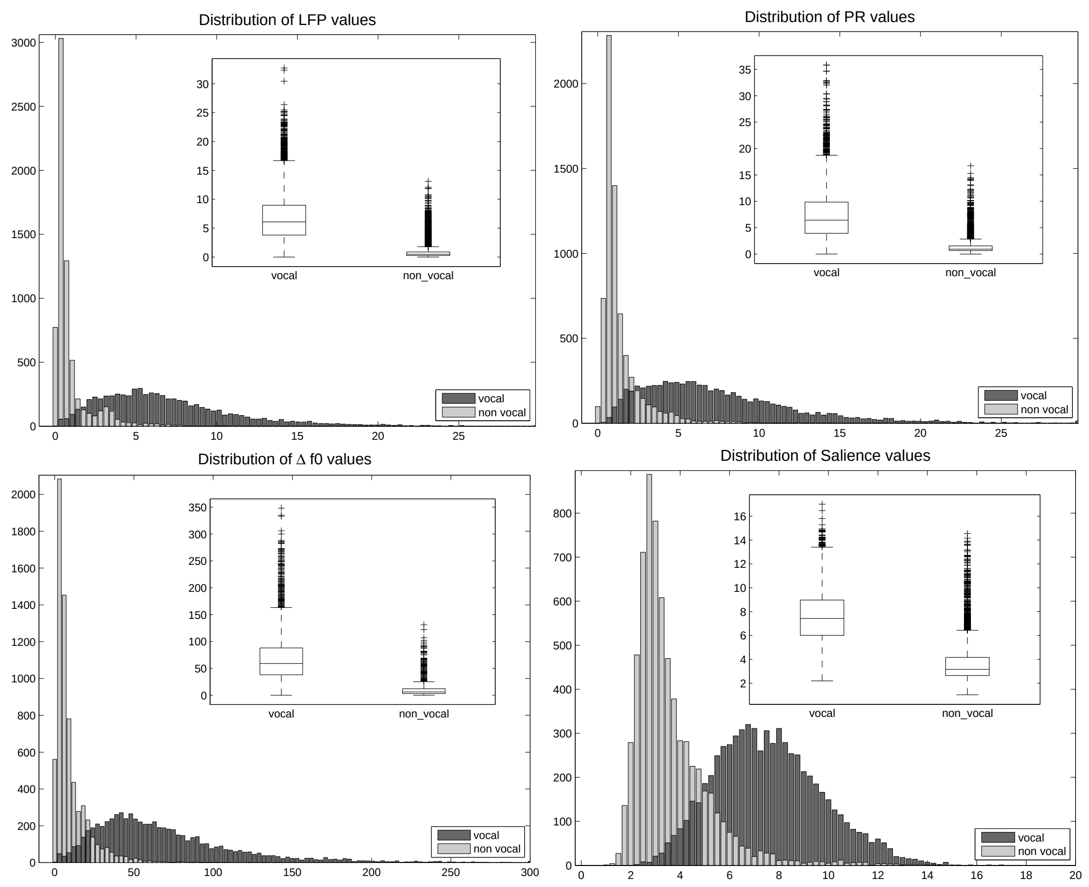


FIGURE 9.10: Histograms and box plots of pitch-related features on the training data.

The procedure for building the database involves the FChT analysis followed by pitch tracking and sound source extraction. Most important parameters for this front-end are: 10 kHz analysis bandwidth, 21 chirp rate α values in the interval [-5,5] and a 16th semitone f_0 grid extending over four octaves (from E_2 to E_6). After source extraction, MFCC and pitch-related features are computed on the synthesized harmonic sources.

When inspecting the resulting extracted sounds it turned out that several of them exhibit very low energy, almost inaudible in some cases. This is because unlike the polyphonic case there are no interfering sounds in the training examples. Hence the signal to noise ratio is high, enabling the tracking algorithm to detect even very low prominent sounds. Therefore, some of the resulting contours were discarded based on their energy, as they were not considered valid training examples. For this purpose a threshold on the median of the first MFCC coefficient was applied, since it describes the overall energy of an audio frame. Furthermore, too short extracted sounds were also discarded, by means of a threshold of $\sim$ 200 ms. This was done in order to be able to perform the pitch variations analysis described in section 9.2.2 for all the instances of the training database. Finally a database of 13598 audio sounds was obtained, with a mean duration of 0.57 second and a standard deviation of 0.51 second, where vocal/non-vocal classes are exactly balanced (50% of each).

9.3.2 Classifiers and training

First of all, it is interesting to assess the discriminating ability of the proposed pitch-related features on the training patterns. For this purpose, histograms and box-plots are presented in Figure 9.10 for all the pitch-related features considered. Although it was already stated that these features have to be used with other sources of information for proper classification, it seems they are informative about the class of the sound. In addition, simple classification experiments were conducted to further study the information provided. Using only these pitch-related features two simple classifiers, namely an SVM operating as a linear discriminant and a k-NN, are trained and tested using 10-fold CV on the training data. The linear discriminant is obtained by a Polynomial Kernel SVM with a kernel exponent $d = 1$ and a penalty factor $C = 1$ (see section 6.1.4). The number of nearest neighbors is set to $k = 5$ after selection by cross-validation. Results are reported on Table 9.1. The performance is quite encouraging, though an estimate based on training data despite the cross-validation is probably over optimistic.

SVM classifier (linear discriminant)

		classified as	
		vocal	non-vocal
class	vocal	6376	423
	non-vocal	439	6360

correctly classified 93.7%

k-NN classifier (k=5)

		classified as	
		vocal	non-vocal
class	vocal	6474	325
	non-vocal	432	6367

correctly classified 94.4%

Table 9.1: Percentage of correctly classified instances and confusion matrix obtained by 10-fold CV on the training dataset for each classifier using only the pitch-related features. The rows of the matrix correspond to the actual class of the audio sound and the columns indicate the classifier hypothesis.

A feature selection experiment was conducted on the pitch-related features using the Correlation Based feature selection method and the Best First searching algorithm (see section 5.3.1). The best resulting subset of features is the complete set, indicating all features provide some sort of information according to this selection criteria. In the case of the MFCC features, no selection was attempted in order to use the same set of features that reported the best results for the singing voice detection approach described in the first part of this dissertation. The only differences are the analysis bandwith, constrained by the sound source extraction front-end to 10 kHz, and the elimination of the first MFCC coefficient which was used for filtering the training database and is a measure of the overall energy of an audio frame of the sound. As it happened when analysing polyphonic audio segments (see Figure 5.4), classes seem to be rather overlapped in the training patterns by considering three of the most discriminative MFCC features, as shown in Figure 9.11.

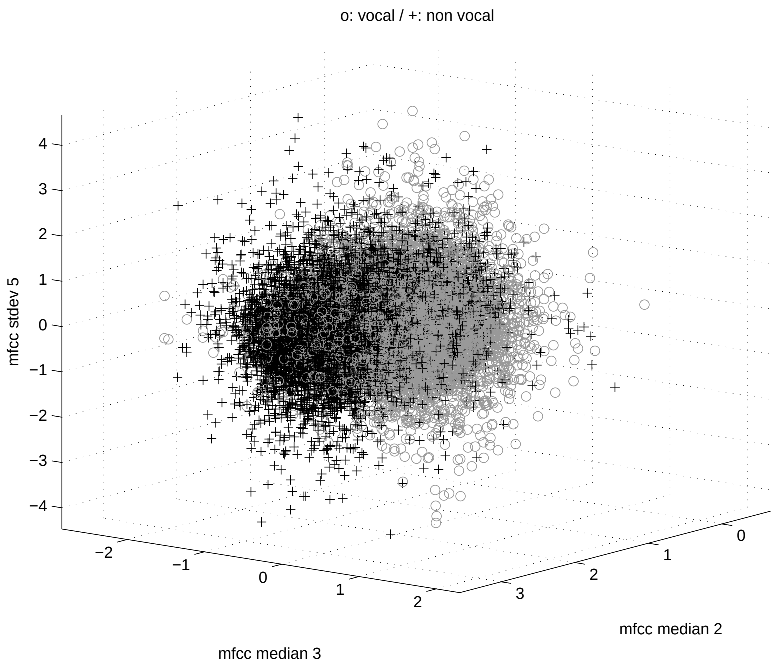


FIGURE 9.11: Distribution of the training patterns for three of the most discriminative features of the MFCC category. Significant overlap between classes can be appreciated.

An SVM classifier with a Gaussian RBF Kernel, $k(x, x') = e^{-\gamma \langle x-x', x-x' \rangle^2}$, was selected for further classification experiments. This type of classifier was one of the best in performance in the evaluations conducted in Chapter 6. Optimal values for the γ kernel parameter and the penalty factor C were selected by grid-search, in a similar way as described before (see Figure 6.7), for each set of features.

Two different sets of features were considered: the MFCC features alone, and with the addition of the pitch-related features. Performance estimated by 10-fold CV on the training set is presented in Table 9.2 as well as the classifier parameters.

MFCC (50 features)

		classified as	
		vocal	non-vocal
class	vocal	6671	128
	non-vocal	123	6676

correctly classified 98.2%

classifier parameters: $\gamma = 2$, $C = 4$

MFCC + Pitch (54 features)

		classified as	
		vocal	non-vocal
class	vocal	6740	59
	non-vocal	38	6761

correctly classified 99.3%

classifier parameters: $\gamma = 1$, $C = 16$

TABLE 9.2: Percentage of correctly classified instances and confusion matrices obtained by 10-fold CV on the training dataset for the MFCC set and with the addition of pitch-related features using an SVM classifier. The rows of each matrix correspond to the actual class of the audio sound and the columns indicate the classifier hypothesis.

Some remarks can be made on these results. Firstly, that the performance is encouraging, though it has to be taken with caution since it is based on training data. Then, that the pitch-related features seem to contribute to the discrimination between classes, specially considering that an increase of 1% is more relevant in such a high performance level. Finally, that the confusion matrix is well balanced in either case, indicating there seems not to be a bias in classification.

9.3.3 Classification of polyphonic music

The training database consists of isolated monophonic sounds, but the goal of the system is to classify polyphonic music. In order to do that, the following procedure is performed. First of all, the sound source extraction front-end is applied. This comprises the time-frequency analysis by means of the FChT, the polyphonic pitch tracking and the resynthesis of the sound corresponding to each of the identified contours. After that, MFCC and pitch-related features are computed on every extracted sound and they are classified to either being vocal or not.

A time interval of the polyphonic audio file is labeled as vocal if any of the identified pitch contours it contains is classified as vocal. This is shown in the classification examples of Figures 9.12 and 9.13. When manual labeling a musical piece, very short pure instrumental regions are usually ignored. For this reason, the automatic labels are further processed and two vocal regions are merged if they are separated by less than a certain amount of time, which was experimentally set to 500 ms. Note, however, that this sometimes introduces unwanted false positives, as shown in Figure 9.15. It is important to notice that manually generated labels include unvoiced sounds (i.e. not pitched, such as fricative consonants). Although this type of sounds are shrunk when singing, this constitute a systematic source of errors of the proposed approach that should be tackled by also modeling unvoiced singing voice sounds.

The SVM Gaussian RBF kernel classifier trained only on MFCC features, which was presented in Table 9.2, is used for testing classification performance on polyphonic music. It is also important to assess the impact of the inclusion of pitch-related features on classifying polyphonic music. In order to do that the following issue has to be taken into account. The analysis of pitch fluctuations described in section 9.2.2 imposes a certain minimum contour length, otherwise non existing information would be drawn. For this reason a threshold of $\sim$ 200 ms is applied and pitch fluctuations analysis is avoided for shorter segments. In those cases only Δf_0 and Salience are used as pitch-related features. Therefore, two different classifiers are trained to include pitch features (along with the MFCC coefficients), one which considers all of them, i.e. LFP, PR, Δf_0 and Salience, and the other which handles only the latter two. When processing polyphonic music the classification system applies the corresponding classifier based on pitch contour length.

In the following section an evaluation is presented which compares this system to the one which uses only MFCC features for polyphonic music classification.

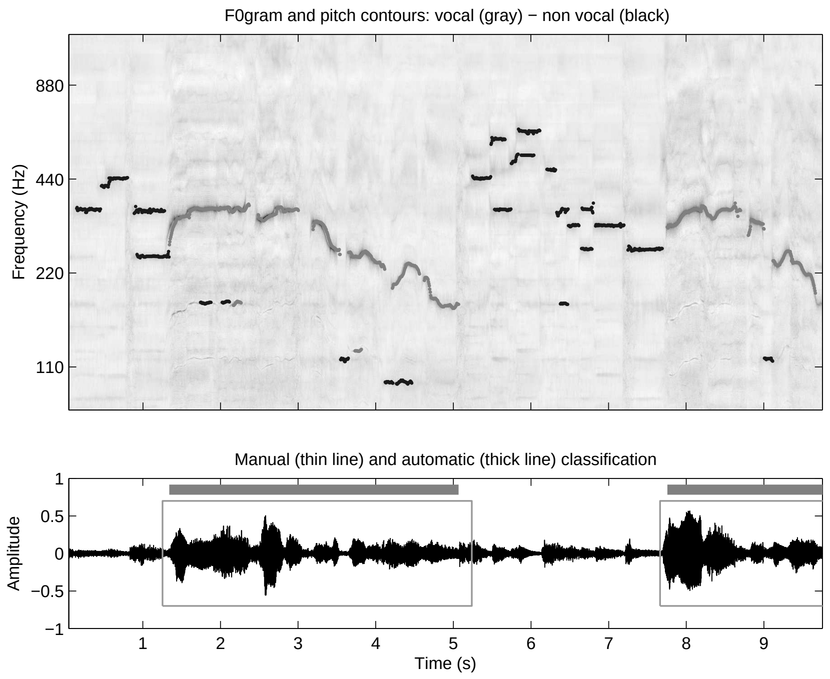


FIGURE 9.12: Example of automatic classification for a Blues song excerpt from the testing dataset. It comprises singing voice, piano, bass and drums. Singing voice notes are correctly distinguished from piano and bass notes that also appear in the F0gram.

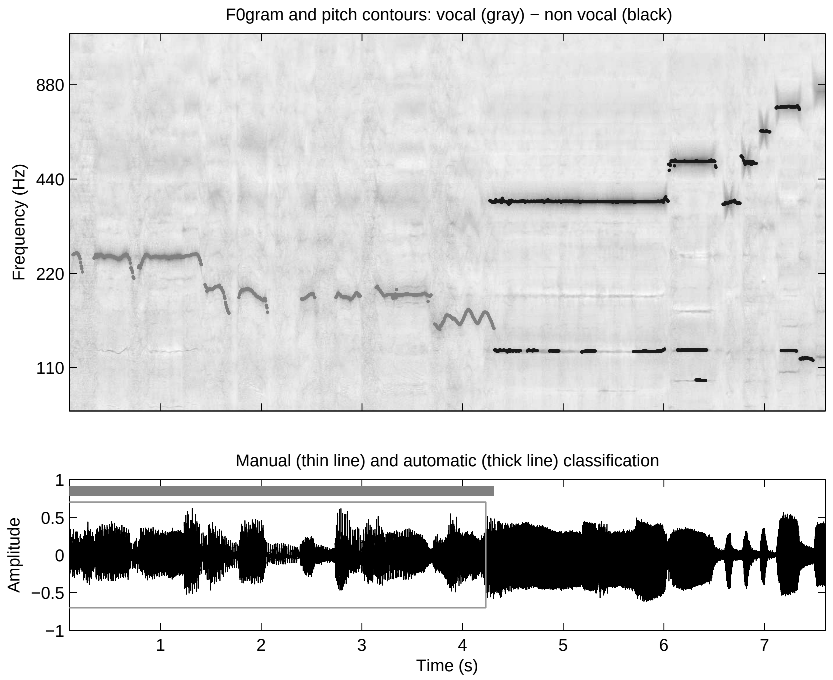


FIGURE 9.13: Example of automatic classification for a fragment of the song *For no one* by The Beatles. A singing voice in the beginning is followed by a French horn solo. There is a rather soft accompaniment which is almost not present in the F0gram.

9.4 Evaluation and results

An evaluation was conducted to estimate the performance of the classification approach applied on polyphonic music audio files, and to assess the usefulness of the proposed pitch-related features. For this purpose a validation database of 30 audio fragments of 10 seconds length was utilized [VALID2], a subset of the data used for validation in our preliminary work [45]. Music was extracted from Magnatune[2] recordings and files are distributed among songs labeled as: blues, country, funk, pop, rock and soul. A few musical genres were discarded, such as heavy-metal or electronica, because of the high density of sources and the ubiquity of prominent noise-like sounds, what makes the pitch tracking rather troublesome.

Only the three most prominent local f_0 candidates were considered for pitch tracking. The configuration of the sound source extraction front-end was the same as for building the training database (see section 9.3.1). Pitch contours shorter than $\sim$ 90 ms were discarded and no threshold on the salience or energy of the extracted sounds was applied.

Classification results are presented in Table 9.3 for each set of features using an SVM classifier. Performance is measured as the percentage of time in which the manual and automatic classification coincides. The addition of the information related to pitch produces a noticeable performance increase in the overall results, as well as for almost every file of the database, as shown in Figure 9.14.

	correctly classified
MFCC (50 features)	71.3%
MFCC + Pitch (54 features)	78.0%

TABLE 9.3: Classification performance for both set of features measured as the percentage of time in which the manual and automatic classification coincides.

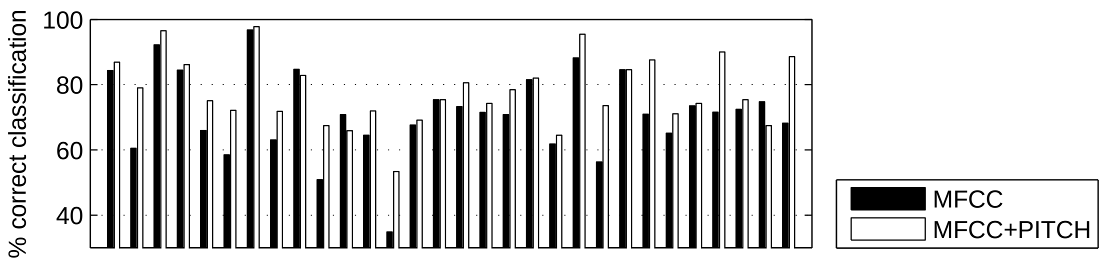


FIGURE 9.14: Classification performance for each file of the database. The addition of the pitch-related features provides a performance increase for almost every file.

[2] http://magnatune.com/

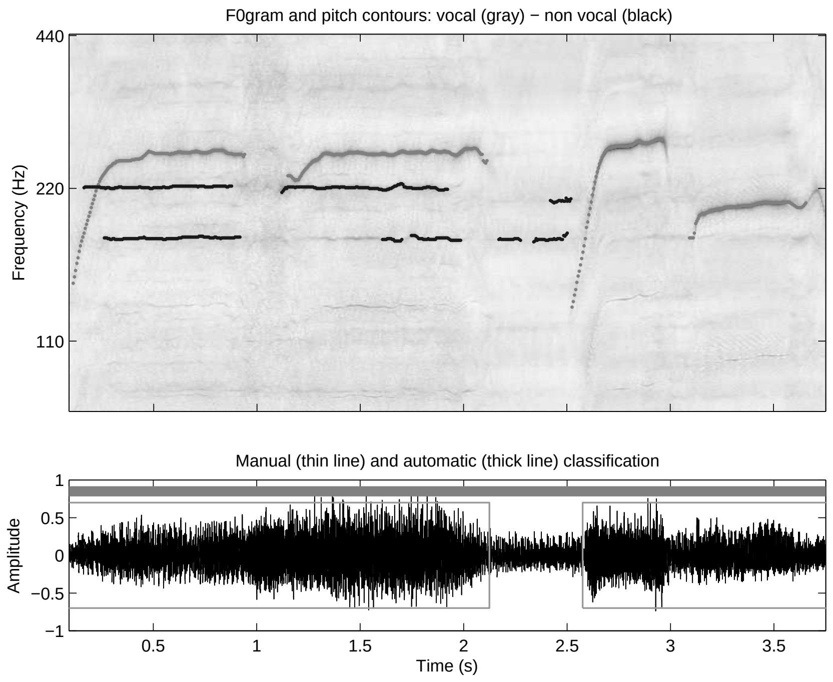


FIGURE 9.15: Example of automatic classification for an excerpt of the file *pop1.wav* from the MIREX melody extraction set. Note that since close vocal regions are merged when producing the automatic labels the short pure instrumental interval is ignored.

The impact of the addition of the pitch-related features is also shown in the example of Figure 9.15 for the file *pop1.wav* from the MIREX melody extraction test set (used for illustrating the pitch tracking algorithm in Chapter 8). It consists of three simultaneous prominent singing voices in the first part followed by a single voice in the second part, and a rather soft accompaniment without percussion. The leading singing voice is correctly identified but the backing vocals are wrongly classified as instrumental sounds. This is not very surprising since they exhibit almost no pitch modulations and are arguably intended to be part of the accompaniment of the leading voice.

Another example of the effect of the pitch-related features is presented in Figure 9.16 for a Jazz audio excerpt from the testing database. It consists of a saxophone solo with piano, bass and drums. The classification using only MFCC features produces several false positives. This situation is improved by the addition of pitch-related features, though some notes are still misclassified. In order to improve the proposed system this type of examples has to be carefully studied in order to understand the underlying causes for the remaining classification errors.

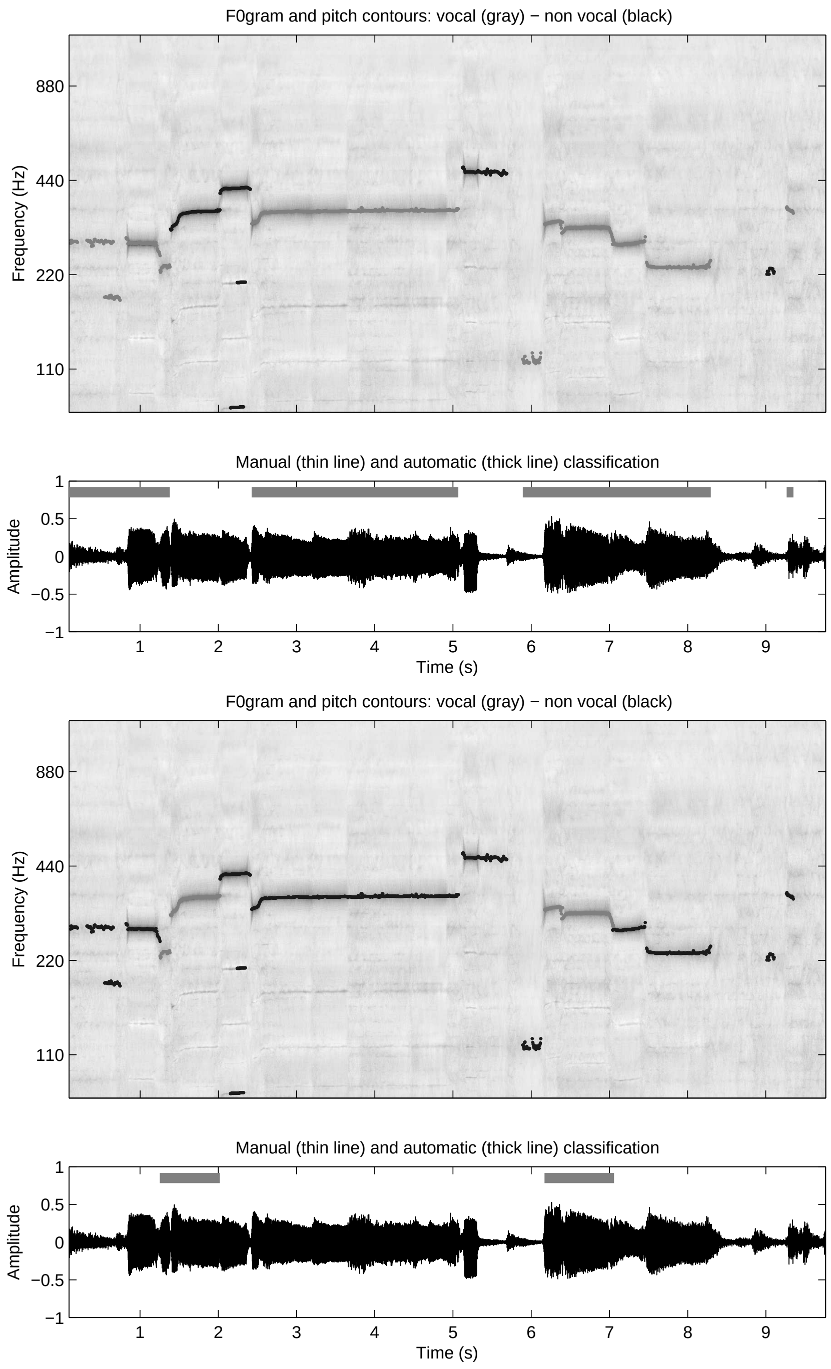


FIGURE 9.16: Example of automatic classification of a Jazz audio excerpt, with a leading saxophone, piano, bass and drums. The classification using only MFCC features (above) produces several false positives, some of which are removed by introducing the pitch-related features (below).

9.5 Discussion and conclusions

A front-end for harmonic sound sources extraction from polyphonic music was presented, that makes use of the time-frequency analysis methods and the polyphonic pitch tracking algorithms described in the two previous chapters. The extracted sounds are then classified as being vocal or not. For this purpose the classical MFCC features, which reported the best results in the study conducted in the first part of this dissertation, are applied. In addition, some new features are proposed intended to capture characteristic of typical singing voice pitch contours. A database of monophonic sounds was built and their corresponding acoustic features were computed in order to train different classifiers for the singing voice detection problem. The usefulness of the pitch-related features was assessed on the training dataset as well as on polyphonic music files. Results obtained indicate that these features provide additional information for singing voice discrimination.

The are significant differences between the estimated performance on the training data and that obtained when testing the system on polyphonic music. This is to be expected for many reasons. First of all, the performance measure is different. For the training data it accounts for the amount of correct and false classifications given isolated sounds. In the case of polyphonic music the performance measure is computed as the percentage of time in which the manual and automatic labels coincide. This introduces many others sources of error. The proposed source extraction approach ignores unvoiced sounds which are nonetheless manually labeled in the polyphonic music examples. In addition, pitch tracking errors are a critical point in performance. As it was observed in the evaluation presented in Chapter 8 section 8.2.3, low prominent sounds are troublesome for the algorithm and when a note gradually vanishes the pitch tracking may stop prematurely. Other kind of situations were also observed, such as superpositions or continuations of notes from different instruments, for instance when a singing voice is followed by an instrumental solo and the same pitch is used in the transition (such as in the example of Figure 9.13 but with a coincident pitch). This type of problems should be taken into account when forming pitch contours, and could be improved by considering timbre information in the pitch tracking algorithm, as proposed in [112].

In addition, there is probably other reason for the high performance results obtained by 10-fold CV on the training database. Given the process of building the database there is some redundancy, in the sense that a certain instrument or voice timbre is represented by several sounds. This makes the 10-fold CV performance estimation less reliable since the training folds are likely to contain very similar instances to that of the testing fold in each iteration. Although this seems to be a shortcoming of the performance estimation method rather than a problem of the training database, a more compact set would be desirable in terms of training time and data storage. Selecting a small representative subset of a large dataset without degrading classification performance is not a trivial task that can be tackled by means of data condensation techniques [113]. Anyway, the training database should be enriched and broadened in order to increase its generalization ability. This would also call for data condensation to manage an even larger dataset.

More experiments are needed to better assess the performance of the proposed technique and the influence of each processing step and parameter. For example, only the three most prominent local fundamental frequency candidates where considered in the reported results. This assumes the singing voice is almost always the most prominent sound, because as it was observed in the pitch tracking simulations (see the example of Figure 8.11) some local fundamental frequency candidates tend to appear as secondary peaks of the most prominent sounds instead of representing the less prominent ones. A performance increase would be expected by considering more local fundamental frequency candidates, but this should be tested.

In addition, other source of information should be taken into account in order to devise acoustic features for singing voice detection. With regards to this, the proposed approach enables the exploration of features that are commonly applied in monophonic sounds classification and could not be addressed in polyphonic sound mixtures.

An interesting property of the proposed singing voice detection approach is that it not only delivers labels but also isolated sounds from the polyphonic mixture. As a by product of this research an automatic singing voice extraction system is obtained. The extracted sounds identified as singing voice can be subtracted from the polyphonic mixture providing a residual signal with the remaining sounds. This produces very good results in several cases, mainly in low polyphonies where singing voice can be tracked appropriately. This has many applications in music processing. Two automatic singing voice separation examples are shown in Figures 9.17 and 9.18, for audio excerpts previously introduced. More examples including the audio files are provided at http://iie.fing.edu.uy/∼rocamora/mscthesis/.

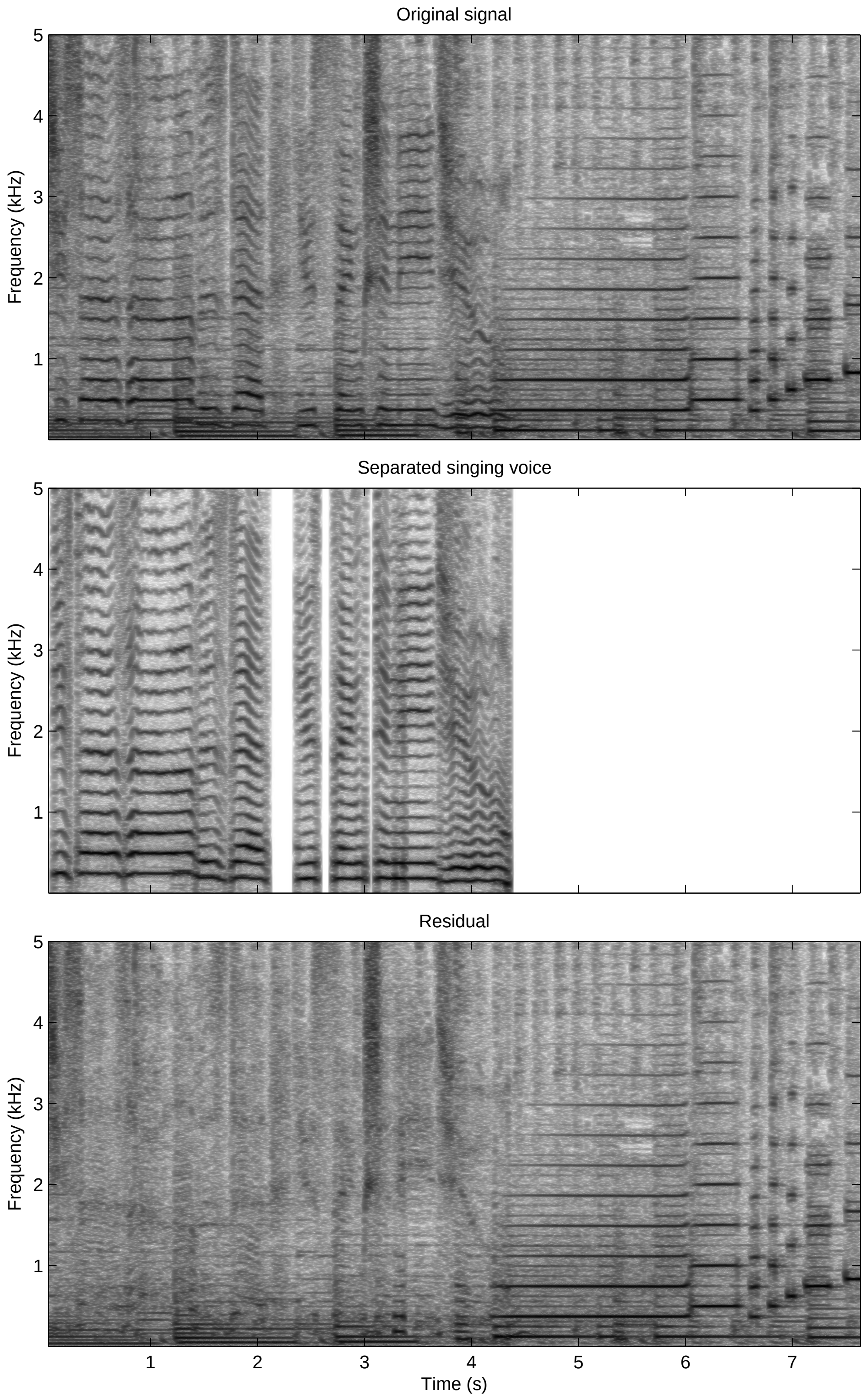


FIGURE 9.17: Automatic singing voice separation example for a fragment of the song *For no one* by The Beatles. A singing voice in the beginning is followed by a French horn solo. There is a rather soft accompaniment where bass notes are clearly visible.

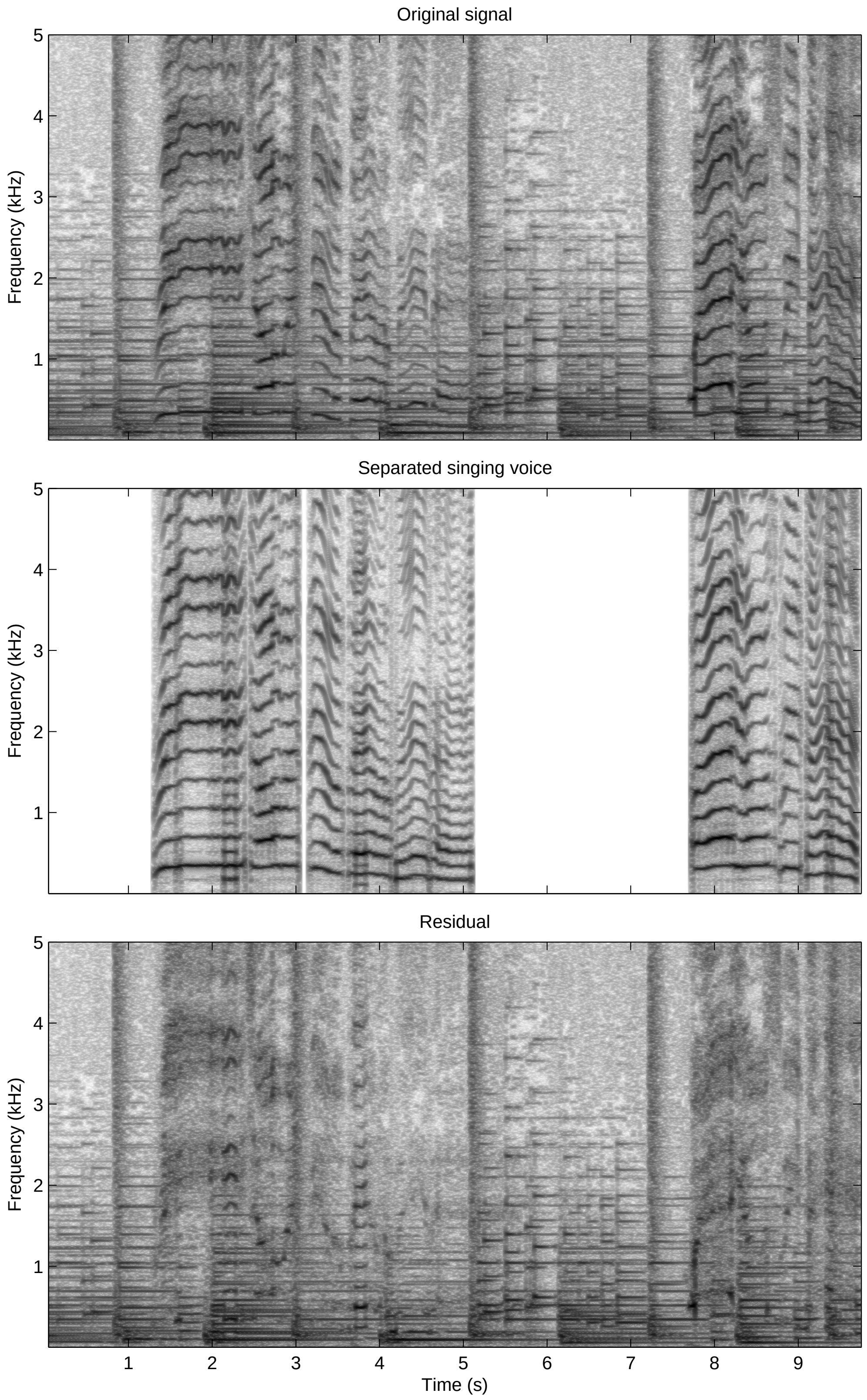


FIGURE 9.18: Automatic singing voice separation example for a fragment of a Blues song excerpt which was introduced in Figure 9.12. Apart from the singing voice, piano, bass and drums are also present.

10
Discussion and conclusions

10.1 Summary and comparison of the proposed approaches

This dissertation tackled the problem of singing voice detection in polyphonic music. Based on a review of the existing literature the most standard approach to address the problem was implemented, that is, a classical pattern recognition system based on acoustic features computed from the polyphonic music signal. Most of those features are different ways of describing the spectral power distribution of the audio signal, thus characterizing sound mixtures of musical instruments. The signal is processed in frames and acoustic features are computed for each of them. The audio file is divided into short segments which are classified as either being vocal or non-vocal, based on statistical measures of the feature values within the segment. From the study on the features reported to be used for the problem Mel Frequency Cepstral Coefficients (MFCC) were selected for the implementation of the system. In addition a Support Vector Machine (SVM) was the classifier adopted from the ones analysed, being trained on a database of polyphonic audio segments of each class.

Among the various possible research directions to follow in order to improve this basic approach, the separation of sound sources from the audio mixture was selected as a promising option. This was based on the hypothesis that sound sources could be better characterized after being isolated from the sound mixture, and this in turn would provide an improved classification performance. Therefore, in the second approach implemented for addressing the problem, a harmonic sound sources extraction front-end was proposed to deal with polyphonic music. A non-traditional time-frequency representation was implemented, namely the Fan Chirp Transform (FChT), which is devised for analysing non-stationary harmonic sound sources. This is done to better representing the singing voice, which typically exhibits rapid pitch fluctuations. Then, a polyphonic pitch tracking algorithm, which tries to identify and follow the most prominent harmonic sound sources in the audio mixture, was proposed. This comprises the computation of a pitch salience function which provides local fundamental frequency candidates and their temporal integration into pitch contours by means of an unsupervised clustering technique. The identified sources are retained in the FChT spectral domain and resynthesized. On these extracted sounds the classical MFCC features are computed, as well as some new features that are proposed to exploit information on pitch evolution and prominence. Finally, the extracted sounds are classified using an SVM trained on a database of monophonic sounds.

Some final evaluation experiments were conducted to compare the performance of the proposed approaches, which is reported in Figure 10.1. The test data used comprises four different sets of labeled polyphonic music. Each classification approach was validated during its development with different audio files (sections 6.3 and 9.4). Both of these validation datasets are included in the evaluation experiments, namely [VALID1] and [VALID2], which are the 30 songs from the albums *Abey Road* and *A Hard Day's Night* by The Beatles and the 30 audio fragments of 10 seconds length from Magnatune recordings, respectively. In addition, a dataset of 46 songs [TEST1], which was used for testing in our preliminary study [45], is considered. Music in this set comprises 7 different singers and includes folk, hip-hop, pop and rock. The last set [TEST2] is a group of 16 songs from the Jamendo[1] free music sharing website, that was used for testing in [32] and is kindly provided by the authors.[2]

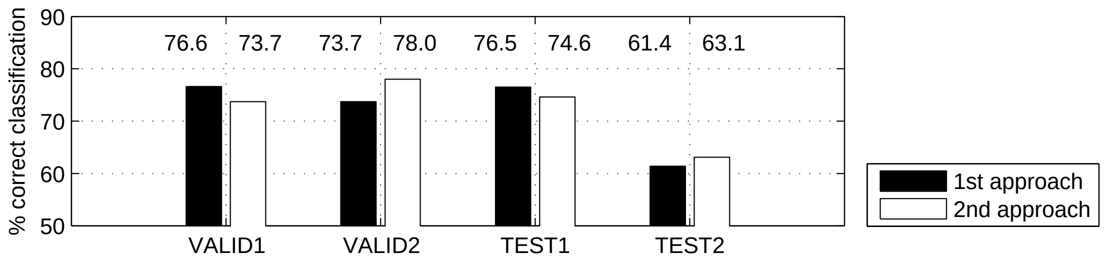


Figure 10.1: Performance of the proposed systems for each dataset, measured as the percentage of time in which the manual and automatic classification coincides.

The obtained results indicate the performance of both systems is similar, though minor differences are observed depending on the testing dataset. For example, the source extraction approach shows the best achieved performance on the [VALID2] database, which is probably related to the fact that some musical genres where avoided in this

[1]http://www.jamendo.com
[2]http://www.mathieuramona.com/wp/data/jamendo

set to favour the correct pitch tracking of harmonic sound sources. In the case of [VALID1] and [TEST1] databases the first approach yields better results but by a small margin. By inspecting the differences in performance for each song it turned out that some audio files included in both sets are evidently problematic for the source separation approach, mainly regarding the degree of polyphony. For example considering only three fundamental frequency candidates is clearly insufficient for pitch tracking the multiple and duplicated singing voices present in *Carry that weight* or *Because* by The Beatles, in which none of them is necessarily the leading one thus produce a constant alternation of the local pitch hypothesis between different sources. Other difficult examples include *High on Rebellion* by Patti Smith, in which besides several singing voices there are loud and distorted electric rhythm guitar sounds during the whole song. Besides, the system is not well suited for the speech-like singing style of hip-hop, in which notes are generally short and with a high number of unvoiced sounds. In brief, as it would be expected, the source separation approach is appropriate for low polyphonies in which a leading singing voice is identified. In those cases, as it seems to be confirmed by the results on the [VALID2] dataset, it can improve the results of the classical approach. However, further experiments are needed to be able to provide more evidence on this.

The lowest performance are obtained with the [TEST2] dataset, where the source separation approach yields a slightly better performance. The difference compared to the other testing databases is probably due to the type of music it contains or it may be related to the fact that this set is coded in OGG Vorbis at 112 kbps which is a relatively low quality. A more rigorous study on the databases and the performance on the individual audio files is needed to account for the observed differences. In addition, it is difficult to compare these results with the ones reported in other research work because there is no standard dataset or evaluation procedure, but this should be tackled in future work.

10.2 Critical discussion on the present work

In this work the most standard existing approach for singing voice detection was studied and a considerable effort has been put into the improvement of its performance by considering different features and classifiers. A study was conducted in which almost all the different features reported to be used for the problem were gathered and compared under equivalent conditions to assess its usefulness, applying various techniques for feature selection, extraction and combination. As a result, the MFCC features were identified as the most appropriate for the singing voice detection problem. A preliminary report on this study [45] served as a reference for other research works on the subject [24, 38, 39] which tried to improve on the results provided by the MFCC features without success. This preliminary study was later extended yielding analogous results, as reported in Chapter 5. In a similar way, various classification methods were applied to the problem and their parameters finely tuned, without obtaining a significant performance increase. In addition, although different attempts were made no other type of features were obtained that proved to be appropriate for the problem. In short, relatively encouraging results are obtained with this simple pattern recognition system, but it turned out to be difficult to go beyond the achieved performance by attempting variations on the same approach, which calls for a new paradigm for addressing the problem. This kind of *glass ceiling* effect was observed in other similar music information retrieval problems [10].

Anyway, many aspects of the standard approach for singing voice detection were not tackled within this work and surely deserve further attention. For instance, the temporal integration of frame level features is performed in a very simple way and there is some room for improvement in better modeling the temporal evolution of features. Besides, it would be important to shed some light on what type of information relevant for this particular problem are the MFCC features capturing, beyond the evident description of the spectral power distribution. This may lead to some improvements on the computation of specifically devised acoustic features.

A novel way of addressing the singing voice detection problem was proposed within this dissertation, which involves the separation of harmonic sound sources from the polyphonic audio mixture. Although the results of some tests are encouraging, a drawback of this approach is that it introduces new challenges and is prone to different sources of error that have a negative impact on the overall performance (e.g. pitch tracking errors). On the other hand, it provides a better insight of possible opportunities for improvement, since errors tend to be more interpretable than for the classical pattern recognition approach (e.g. deficiency of the time-frequency analysis, shortcomings of the pitch salience function, pitch tracking algorithm limitations). More work is needed to better analyse those sources of errors. In particular it is necessary to design experiments that discriminate between pitch tracking errors and wrong classifications (such as the ones in Figure 9.16). Another contribution of the present work is a simple set of pitch related features that seem to be appropriate for the singing voice detection problem, but that should be further developed.

It is interesting to note that the computational effort of the second approach is several orders of magnitude higher than for the first one. This was not considered very relevant in this research but it should be tackled in future work. The most heavy task of the source extraction front-end is the computation of the several FChT instances, and although its implementation is rather efficient there are many alternatives to improve it further.

10.3 Conclusions and future work

One of the main conclusions of the present work is that no improvements on singing voice detection performance are easily obtained by variations on the usual pattern recognition system based on acoustic features computed from audio sound mixtures. In addition, it was observed that the approach proposed herein which involves the harmonic sound sources separation followed by their classification seems a feasible alternative, though much work is needed to improve performance results and solve its main limitations.

An advantage of the sound source separation approach is that it enables the application of other acoustic features for describing isolated sounds that are otherwise obscured in the polyphonic sound mixture. In this sense, the estimation of formants in the extracted sounds is an interesting direction for future work. A distinctive characteristic of the singing voice is that formants are changed continuously by modulations of the vocal tract resonances. Some preliminary experiments were conducted on this by using recently proposed spectral envelope estimation techniques [114] and several challenging practical issues arise. Firstly, the validation of spectral peaks as true formants is not a trivial task [115]. Secondly, temporal tracking formant locations is somehow troublesome, considering that they dynamically emerge and disappear [116]. Finally, appropriate ways of using this information for singing voice detection have to be devised.

Above all, there is a need for a thorough study on the distinctive features that human beings exploit in order to recognize singing voice sounds. This implies taking into account psychoacoustic and cognitive information which was not yet considered, and would surely demand an important research effort. The work reported in [40] is the kind of music perception experiment that should be conducted, but which has almost not been addressed up to the moment.

As an interesting outcome of this research work an automatic singing voice separation system is obtained which yields very promising results, mainly in low polyphonies where singing voice can be tracked appropriately. The improvement of this tool and its application to different music scenarios are the most appealing ideas for future research.

A

Author's related publications

In the following we provide a list of publications associated with this dissertation in which its author has participated. Electronic versions of all of them are available from http://iie.fing.edu.uy/publicaciones/.

1. **Authors:** Martín Rocamora, Pablo Cancela

   Title: Pitch tracking in polyphonic audio by clustering local fundamental frequency estimates

   Conference: Brazilian AES Audio Engineering Congress, 9th. São Paulo, Brazil

   Year: 2011

   Related to chapter: 8

   Abstract: A novel way of performing pitch tracking by means of clustering local fundamental frequency (f0) candidates is described. The technique is based on an existing pitch salience representation for polyphonic music called F0gram which relies on the Fan Chirp Transform. The grouping is performed by applying a Spectral Clustering method, since it can handle filiform shapes such as pitch contours. The approach seems appealing since many sound sources can be tracked simultaneously and the number of contours and sources is derived from the data. Results of a melody detection evaluation indicate the introduced method is promising, despite that various aspects of the technique deserve further work.

2. **Authors:** Pablo Cancela, Ernesto López, Martín Rocamora

   Title: Fan chirp transform for music representation

   Conference: Int. Conf. on Digital Audio Effects, 13th. DAFx-10. Graz, Austria

   Year: 2010

   Related to chapter: 7 and 8

   Abstract: In this work the Fan Chirp Transform (FChT), which provides an acute representation of harmonically related linear chirp signals, is applied to the analysis of pitch content in polyphonic music. The implementation introduced was devised to be computationally manageable and enables the generalization of the FChT for the analysis of non-linear chirps. The combination with the Constant Q Transform is explored to build a multi-resolution FChT. An existing method to compute pitch salience from the FChT is improved and adapted to handle polyphonic music. In this way a useful melodic content visualization tool is obtained. The results of a frame based melody detection evaluation indicate that the introduced technique is very promising as a front-end for music analysis.

3. **Authors:** Pablo Cancela, Martín Rocamora, Ernesto López

   Title: An efficient multi-resolution spectral transform for music analysis

   Conference: Int. Society for Music Information Retrieval Conf., 10th. ISMIR 2009. Kobe, Japan

   Year: 2009

   Related to chapter: 7

   Abstract: In this paper we focus on multi-resolution spectral analysis algorithms for music signals based on the FFT. Two previously devised efficient algorithms (efficient constant-Q transform [1] and multiresolution FFT [2]) are reviewed and compared with a new proposal based on the IIR filtering of the FFT. Apart from its simplicity, the proposed method shows to be a good compromise between design flexibility and reduced computational effort. Additionally, it was used as a part of an effective melody extraction algorithm.

4. **Authors:** Martín Rocamora, Perfecto Herrera

   Title: Comparing audio descriptors for singing voice detection in music audio files

   Conference: Brazilian Symposium on Computer Music, 11th. São Paulo, Brazil

   Year: 2007

   Related to chapter: 4, 5 and 6

   Abstract: Given the relevance of the singing voice in popular western music, a system able to reliable identify those portions of a music audio file containing vocals would be very useful. In this work, we explore already used descriptors to perform this task and compare the performance of a statistical classifier using each kind of them, concluding that MFCC are the most appropriate. As an outcome of our study, an effective statistical classification system with a reduced set of descriptors for singing voice detection in music audio files is presented. The performance of the system is validated using independent datasets of popular music for training, validation and testing, reaching a classification performance of 78.5% on the testing set.

B

List of audio databases

The following is a unified list of all the audio databases used within this thesis work.

1. **Identifier:** [TRAIN1]

   Source: Developed during this thesis work

   Related to chapter: 5 and 6

   Format: 44,100 Hz, 16 bits, WAV

   Description: It was built by automatically extracting short audio excerpts from music recordings and manually classifying them into vocal and non-vocal. Three different excerpt-length sets were constructed of 0.5, 1 and 3 seconds length. A vocal excerpt is discarded if it does not contain vocal sounds for at least more than 50% of its length. The non-vocal excerpts do not have any aurally identifiable vocal sound. All different-length sets contain 500 instances of each class. Music utilized belongs to a music genres database comprising alternative, blues, classical, country, electronic, folk, funk, heavy-metal, hip-hop, jazz, pop, religious, rock and soul. In addition, approximately 25% of pure instrumental and a cappella music was also added. The music genres database was provided by the Music Technology Group, Pompeu Fabra, Barcelona.

2. **Identifier:** [TRAIN2]

   Source: Developed during this thesis work

   Related to chapter: 9

   Format: 44,100 Hz, 16 bits, most WAV and some MP3 at different coding rates

   Description: Database based on more than 2000 monophonic audio files, comprising singing voice on the one hand and typical musical instruments found in popular music on the other, comprising a cappella music, multi-track recordings, and commercial audio clips from different sources. The proposed sound source extraction front-end was applied to each file and some of the sounds obtained were discarded based on energy and duration. Finally a database of 13598 audio sounds was obtained where vocal/non-vocal classes are exactly balanced (50% of each).

3. **Identifier:** [VALID1]

   Source: Developed during this thesis work

   Related to chapter: 9 and 10

   Format: 44,100 Hz, 16 bits, Ogg Vorbis 128 kbps

   Description: The 30 songs corresponding to the albums *Abbey Road* and *A Hard Day's Night* by The Beatles where manually labeled for vocal/non-vocal regions.

4. **Identifier:** [VALID2]

   Source: Provided by the Music Technology Group, Pompeu Fabra, Barcelona

   Related to chapter: 6 and 10

   Format: 44,100 Hz, 16 bits, MP3 256 kbps

   Description: A set of 30 audio fragments of 10 seconds length manually annotated, which is a subset of the data used for validation in our preliminary work [45]. Music was extracted from Magnatune recordings and files are distributed among songs labeled as: blues, country, funk, pop, rock and soul.

5. **Identifier:** [TEST1]

   Source: Provided by Sira Ferradáns from the Music Technology Group, Pompeu Fabra, Barcelona

   Related to chapter: 10

   Format: 44,100 Hz, 16 bits, MP3 192 kbps

   Description: Manually labeled database of 46 popular songs performed by 7 different singers in the musical genres of folk, hip-hop, pop and rock.

6. **Identifier:** [TEST2]

   Source: Provided by Mathieu Ramona [32]

   Related to chapter: 10

   Format: 44,100 Hz, 16 bits, OGG Vorbis 112 kbps

   Description: The test set of the complete database was selected, comprising 16 songs. It is available from `http://www.mathieuramona.com/wp/data/jamendo`. Description provided by the author: "We have collected a set of 93 songs with Creative Commons license from the Jamendo free music sharing website, which constitute a total of about 6 hours of music. The files are all from different artists and represent various genres from mainstream commercial music. Each file has been manually annotated by the same person with high precision, with the Transcriber free software, developed for the annotation of speech transcription. The jamendo audio files are coded in stereo Vorbis OGG 44,1 kHz with 112 kbps bitrate. The files are divided into three non-overlapping sets, a training set, a validation set and a testing set, composed respectively of 61, 16 and 16 songs."

7. **Identifier:** [MIREX melody extraction test set]

   Source: Provided by MIREX Evaluation Campaign [90]

   Related to chapter: 8

   Format: 44,100 Hz, 16 bits, WAV

   Description: The files containing vocals from the MIREX melody extraction test set were selected. It comprises 21 music excerpts for a total duration of 8 minutes and is publicly available from: `http://www.music-ir.org/mirex/`.

C

Submultiple peak attenuation

The aim of this appendix is to establish the value of the attenuation factor a in equation 8.3, in order to supress the spurious peak of $\rho_1(f_0)$ at $F_0/2$. The same idea can be used to derive the attenuation value at other submultiples.

Following the same reasoning used in [84] appendix B, consider a single harmonic sound source of fundamental frequency F_0, n_H harmonics and log-energy $\overline{\log X^2}$.

Computing $\rho_0(f_0)$ for multiples and submultiples of F_0 gives,

$$
\begin{aligned}
\rho_0(kF_0) &\backsimeq \overline{\log X^2} \\
\rho_0(F_0) &= \overline{\log X^2} \\
\rho_0(F_0/2) &= \frac{\overline{\log X^2}}{2} + \frac{1}{2n_H}\sum_{i=1}^{n_H} \log \eta_i(\tfrac{2i-1}{2}F_0) \\
\rho_0(F_0/4) &= \frac{\overline{\log X^2}}{4} + \frac{1}{4n_H}\sum_{i=1}^{n_H} \log[\eta_i(\tfrac{4i-3}{4}F_0) + \eta_i(\tfrac{4i-2}{4}F_0)\ldots \\
&\quad + \eta_i(\tfrac{4i-1}{4}F_0)]
\end{aligned}
$$

where η_i are i.i.d. random variables representing the samples of the spectral noise. The multiple attenuation results in,

$$\begin{aligned}
\rho_1(kF_0) &\backsimeq 0 \\
\rho_1(F_0) &= \frac{\overline{\log X^2}}{2} - \frac{1}{2n_H} \underbrace{\sum_{i=1}^{n_H} \log \eta_i(\tfrac{2i-1}{2}F_0)}_{r_1} \\
\rho_1(F_0/2) &= \frac{\overline{\log X^2}}{4} + \frac{1}{4n_H} \overbrace{\sum_{i=1}^{n_H} \log \eta_i(\tfrac{2i-1}{2}F_0)} \\
&\quad - \frac{1}{4n_H} \underbrace{\sum_{i=1}^{n_H} \log \eta_i(\tfrac{4i-3}{4}F_0)}_{r_2} - \frac{1}{4n_H} \underbrace{\sum_{i=1}^{n_H} \log \eta_i(\tfrac{4i-1}{4}F_0)}_{r_3}.
\end{aligned}$$

The submultiple attenuation evaluated at $F0/2$ is then,

$$\begin{aligned}
\rho_2(F_0/2) &= \rho_1(F_0/2) - a\rho_1(F_0) \\
\rho_2(F_0/2) &= \frac{\overline{\log X^2}}{2}\left(\frac{1}{2} - a\right) + \frac{r_1}{2n_H}\left(\frac{1}{2} + a\right) - \frac{r_2 + r_3}{4n_H}
\end{aligned}$$

Considering $E\{r_1\} = E\{r_2\} = E\{r_3\} = n_H \log N_0$, where N_0 stands for the noise power (see [84] appendix B),

$$E\left\{\rho_2(F_0/2)\right\} = \frac{\overline{\log X^2}}{2}\left(\frac{1}{2} - a\right) + \frac{\log N_0}{2}\left(\frac{1}{2} + a\right) - \frac{\log N_0}{2}.$$

Therefore the value of a to suppress the peak at $F_0/2$ is $a = 1/2$.

Bibliography

[1] Daniel Leech-Wilkinson. *The Changing Sound of Music: Approaches to Studying Recorded Musical Performance*. Published online through the Centre for the History and Analysis of Recorded Music (CHARM), London, 2009. ISBN 1-897791-21-6. URL http://www.charm.rhul.ac.uk/studies/chapters/intro.html.

[2] A. Klapuri and M. Davy, editors. *Signal Processing Methods for Music Transcription*. Springer, New York, 2006. ISBN 0-387-30667-6.

[3] O. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2008.

[4] H. Fujihara, M. Goto, T. Kitahara, and H.G. Okuno. A modeling of singing voice robust to accompaniment sounds and its application to singer identification and vocal-timbre-similarity-based music information retrieval. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(3):638 –648, March 2010. ISSN 1558-7916. URL http://dx.doi.org/10.1109/TASL.2010.2041386.

[5] Yipeng Li and DeLiang Wang. Singing voice separation from monaural recordings. In *Proceedings of the 7th International Conference on Music Information Retrieval, ISMIR 2006, Victoria, Canada, 8-12 October*, pages 176–179, 2006.

[6] Matti Ryynänen and Anssi Klapuri. Transcription of the singing melody in polyphonic music. In *Proceedings of the 7th International Conference on Music Information Retrieval, ISMIR 2006, Victoria, Canada, 8-12 October*, pages 222–227, 2006.

[7] Xiao Wu, Ming Li, Jian Liu, Jun Yang, and Yonghong Yan. A top-down approach to melody match in pitch contour for query by humming. In *International Conference of Chinese Spoken Language Processing*, December 2006.

[8] A. Mesaros and T. Virtanen. Automatic recognition of lyrics in singing. *EURASIP Journal on Audio, Speech, and Music Processing*, 2010, 2010. URL http://dx.doi.org/10.1155/2010/546047.

[9] N.C. Maddage. Automatic structure detection for popular music. *Multimedia, IEEE*, 13(1):65 – 77, January-March 2006. ISSN 1070-986X. URL http://dx.doi.org/10.1109/MMUL.2006.3.

[10] J.J Aucouturier and F. Pachet. Improving timbre similarity: How high is the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1(1), 2004.

[11] K. D. Martin. *Sound-Source Recognition: A Theory and Computational Model*. PhD thesis, MIT. Cambridge, MA, 1999.

[12] P. Herrera, A. Klapuri, and M. Davy. Automatic Classification of Pitched Musical Instrument Sounds. In A. Klapuri and M. Davy, editors, *Signal Processing Methods for Music Transcription*, pages 163–200. Springer, New York, 2006. ISBN 0-387-30667-6.

[13] L. R. Rabiner and R. W. Schafer. *Digital Processing of Speech Signals*. Prentice Hall, New Jersey, 1978.

[14] T. Chilton. Speech analisys, 1999. School of Electronic and Physical Sciences, University of Surrey.

[15] P. R. Cook. *Identification of control parameters in an articulatory vocal tract model, with applications to the synthesis of singing*. PhD thesis, Stanford Univ., Stanford, CA, 1990.

[16] Y. E. Kim and B. P. Whitman. Singer identification in popular music recordings using voice coding features. In *Proceedings of the 3rd International Conference on Music Information Retrieval, ISMIR 2002, Paris, France, October 13-17*, pages 164–169, 2002.

[17] E. Scheirer and M. Slaney. Construction and evaluation of a robust multifeature speech/music discriminator. In *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, volume 2, pages 1331–1334, April 1997. URL http://dx.doi.org/10.1109/ICASSP.1997.596192.

[18] Y. Li and D. L. Wang. Separation of singing voice from music accompaniment for monaural recordings, 2005. Technical Report OSU-CISRC-9/05-TR61, Department of Computer Science and Engineering, The Ohio State University, Columbus, Ohio, USA.

[19] D. Gerhard. Pitch-based acoustic feature analysis for the discrimination of speech and monophonic singing. *Journal of the Canadian Acoustical Assosiation*, 30(3): 152–153, 2002.

[20] J. Sundberg. *The science of the singing voice*. De Kalb, Il., Northern Illinois University Press, 1987.

[21] N.C. Maddage, Kongwah Wan, Changsheng Xu, and Ye Wang. Singing voice detection using twice-iterated composite fourier transform. In *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on*, volume 2, pages 1347–1350, June 2004. URL http://dx.doi.org/10.1109/ICME.2004.1394478.

[22] A. Shenoy, Y. Wu, and Y. Wang. Singing voice detection for karaoke application. In *Visual Communications and Image Processing 2005, Proc. of SPIE*, volume 5960, 2005. URL http://dx.doi.org/10.1117/12.631645.

[23] Tong Zhang. Automatic singer identification. In *Proceedings of the 2003 International Conference on Multimedia and Expo - Volume 2*, ICME '03, pages 33–36, Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7803-7965-9. URL http://dl.acm.org/citation.cfm?id=1170745.1171614.

[24] L. Regnier and G. Peeters. Singing voice detection in music tracks using direct voice vibrato detection. In *Proceedings of the 2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP '09, pages 1685–1688, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-1-4244-2353-8.

[25] A.L. Berenzweig and D.P.W. Ellis. Locating singing voice segments within music signals. In *Applications of Signal Processing to Audio and Acoustics, 2001 IEEE Workshop on the*, pages 119–122, 2001. URL http://dx.doi.org/10.1109/ASPAA.2001.969557.

[26] Tin Lay Nwe, Arun Shenoy, and Ye Wang. Singing voice detection in popular music. In *Proceedings of the 12th annual ACM international conference on Multimedia*, MULTIMEDIA '04, pages 324–327, New York, NY, USA, 2004. ISBN 1-58113-893-8. URL http://doi.acm.org/10.1145/1027527.1027602.

[27] W.-H. Tsai and Hsin min Wang. Automatic singer recognition of popular music recordings via estimation and modeling of solo vocal signals. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(1):330–341, January 2006. ISSN 1558-7916. URL http://dx.doi.org/10.1109/TSA.2005.854091.

[28] Yipeng Li and DeLiang Wang. Separation of singing voice from music accompaniment for monaural recordings. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(4):1475–1487, May 2007. ISSN 1558-7916. URL http://dx.doi.org/10.1109/TASL.2006.889789.

[29] Adam Berenzweig, Daniel P. W. Ellis, and Steve Lawrence. Using Voice Segments to Improve Artist Classification of Music. In *Proceedings of the AES International Conference on Virtual, Synthetic, and Entertainment Audio*, Espoo, Finland, June 2002. URL http://www.ee.columbia.edu/~dpwe/pubs/aes02-aclass.pdf.

[30] G. Tzanetakis. Song-specific bootstrapping of singing voice structure. In *Multimedia and Expo, 2004. ICME '04. 2004 IEEE International Conference on*, volume 3, pages 2027–2030, June. URL http://dx.doi.org/10.1109/ICME.2004.1394662.

[31] N. C. Maddage, C. Xu, and Y. Wang. A SVM-based classification approach to musical audio. In *Proceedings of the 4th International Conference on Music Information Retrieval, ISMIR 2003, Baltimore, Maryland, USA, October 26-30*, page 2, 2003.

[32] M. Ramona, G. Richard, and B. David. Vocal detection in music with support vector machines. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 1885–1888, April 2008. URL http://dx.doi.org/10.1109/ICASSP.2008.4518002.

[33] Wu Chou and Liang Gu. Robust singing detection in speech/music discriminator design. In *Proceedings of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP '01. Salt Lake City, Utah, USA, May 7-11*, volume 2, pages 865–868, 2001. URL http://dx.doi.org/10.1109/ICASSP.2001.941052.

[34] W. H. Tsai, H. M. Wang, D. Rodgers, S. S.Cheng, and H. M. Yu. Blind clustering of popular music recordings based on singer voice characteristics. In *Proceedings of the 4th International Conference on Music Information Retrieval, ISMIR 2003, Baltimore, Maryland, USA, October 26-30*, page 7, 2003.

[35] W. H. Tsai and H. M. Wang. Towards automatic identification of singing language in popular music recordings. In *Proceedings of the 5th International Conference on Music Information Retrieval, ISMIR 2004, Barcelona, Spain, October 10-14*, pages 373–380, 2004.

[36] Hanna Lukashevich, Matthias Gruhne, and Christian Dittmar. Effective Singing Voice Detection in Popular Music Using ARMA Filtering. In *Proceedings of the 10th International Conference on Digital Audio Effects, DAFx-07, Bordeaux, France, September 10-15*, pages 165–168, 2007.

[37] Hélène Lachambre, Régine André-Obrecht, and Julien Pinquier. Singing voice characterization in monophonic and polyphonic contexts. In *Proceedings of the 17th European Signal Processing Conference, EUSIPCO-2009, Glasgow, Scotland, August 24-28*, pages 1344–1348, 2009.

[38] Markaki M., Holzapfel A., and Stylianou Y. Singing voice detection using modulation frequency features. In *Proceedings of ISCA Tutorial and Research Workshop on Statistical and Perceptual Audition (SAPA), Brisbane, Australia*, page 4, September 2008.

[39] D. Gartner and C. Dittmar. Vocal characteristics classification of audio segments: An investigation of the influence of accompaniment music on low-level features. In *Machine Learning and Applications, 2009. ICMLA '09. International Conference on*, pages 583 –589, December 2009. URL http://dx.doi.org/10.1109/ICMLA.2009.40.

[40] F. Vallet and M. McKinney. Perceptual constraints for automatic vocal detection in music recordings. In *Proceedings of the 3rd Conference on Interdisciplinary Musicology (CIM 07), Tallinn, Estonia, August 15-19*, 2007. URL http://www.uni-graz.at/~parncutt/cim07/.

[41] Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Third Edition*. Academic Press, Inc., Orlando, FL, USA, 2006. ISBN 0123695317.

[42] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[43] Claude Nadeau and Yoshua Bengio. Inference for the generalization error. *Machine Learning, Kluwer Academic Publishers*, 52(3):239–281, 2003. ISSN 0885-6125. URL http://dx.doi.org/10.1023/A:1024068626366.

[44] Tom Fawcett. ROC Graphs: Notes and Practical Considerations for researchers. Tech Report HPL-2003-4, improved version, HP Labs, 1985. URL http://home.comcast.net/~tom.fawcett/public_html/papers/ROC101.pdf.

[45] Martín Rocamora and Perfecto Herrera. Comparing audio descriptors for singing voice detection in music audio files. In *Proceedings of the 11th Brazilian Symposium on Computer Music, São Paulo, Brazil, September 1-3*, pages 187–196, 2007. URL http://iie.fing.edu.uy/publicaciones/2007/RH07.

[46] Daniel P. W. Ellis. PLP and RASTA (and MFCC, and inversion) in Matlab, 2005. URL http://www.ee.columbia.edu/~dpwe/resources/matlab/rastamat/.

[47] A.V. Oppenheim and R.W. Schafer. From frequency to quefrency: a history of the cepstrum. *Signal Processing Magazine, IEEE*, 21(5):95 – 106, sept. 2004. ISSN 1053-5888. URL http://dx.doi.org/10.1109/MSP.2004.1328092.

[48] Alan V. Oppenheim and Ronald W. Schafer. *Digital Signal Processing*. Prentice Hall, 1975. ISBN 0132146355.

[49] Stanley Smith Stevens, J Volkmann, and E B Newman. A scale for the measurement of the psychological magnitude pitch. *Journal of the Acoustical Society of America*, 8(3):185–190, 1937. URL http://dx.doi.org/10.1121/1.1901999.

[50] J.C. Brown. Computer identification of musical instruments using pattern recognition with cepstral coefficients as features. *Journal of The Acoustical Society of America*, 105:1933–1941, 1999. URL http://dx.doi.org/10.1121/1.426728.

[51] A. Eronen and A. Klapuri. Musical instrument recognition using cepstral coefficients and temporal features. In *Proceedings of the 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP '00. Istanbul, Turkey, June 05-09*, volume 2, pages II753 –II756, 2000. URL http://dx.doi.org/10.1109/ICASSP.2000.859069.

[52] B. Logan. Mel frequency cepstral coefficients for music modeling. In *Proceedings of the 1st International Conference on Music Information Retrieval, ISMIR 2000, Plymouth, Massachusetts, USA, October 23-25*, page 11, 2000.

[53] Hynek Hermansky. Perceptual linear predictive (PLP) analysis of speech. *Journal of The Acoustical Society of America*, 87(4):1738–1752, 1990. URL http://dx.doi.org/10.1121/1.399423.

[54] Hugo Fastl and Eberhard Zwicker. *Psychoacoustics: Facts and Models (Springer Series in Information Sciences)*. Springer, Berlin, Germany, third edition, 2006. ISBN 3540231595.

[55] Alain de Cheveigné and Hideki Kawahara. YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111 (4):1917–1930, 2002. URL http://dx.doi.org/10.1121/1.1458024.

[56] M. F. McKinney and J. Breebaart. Features for audio and music classification. In *Proceedings of the 4th International Conference on Music Information Retrieval, ISMIR 2003, Baltimore, Maryland, USA, October 26-30*, page 8, 2003.

[57] Vesa Peltonen, Juha Tuomi, Anssi Klapuri, Jyri Huopaniemi, and Timo Sorsa. Computational auditory scene recognition. In *Proceedings of the 2002 IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP '02. Orlando, Florida May 13-17*, volume 2, pages II–1941 –II–1944, 2002. URL http://dx.doi.org/10.1109/ICASSP.2002.5745009.

[58] M. Ryynänen. Probabilistic modelling of note events in the transcription of monophonic melodies. Master's thesis, Tampere University of Technology, Department of Information Technology, 2004.

[59] J. Janer. Stereo image description of audio mixtures with the panning coefficients, 2007. MTG-Technical Report, Universitat Pompeu Fabra, Barcelona.

[60] C. Joder, S. Essid, and G. Richard. Temporal integration for audio classification with application to musical instrument classification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(1):174 –186, January 2009. ISSN 1558-7916. URL http://dx.doi.org/10.1109/TASL.2008.2007613.

[61] Jean-Julien Aucouturier, Boris Defreville, and Francois Pachet. The bag-of-frames approach to audio pattern recognition: A sufficient model for urban soundscapes but not for polyphonic music. *Journal of The Acoustical Society of America*, 122 (2):881–891, 2007. URL http://dx.doi.org/10.1121/1.2750160.

[62] A. Meng, P. Ahrendt, J. Larsen, and L.K. Hansen. Temporal feature integration for music genre classification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 15(5):1654 –1664, july 2007. ISSN 1558-7916. URL http://dx.doi.org/10.1109/TASL.2007.899293.

[63] Shlomo Dubnov and Xavier Rodet. Timbre recognition with combined stationary and temporal features. In *Proceedings of International Computer Music Conference, Ann Arbor, USA*, pages 102–108, 1998.

[64] A. Eronen. Musical instrument recognition using ICA-based transform of features and discriminatively trained HMMs. In *Proceedings of the 7th International Symposium on Signal Processing and Its Applications, ISSPA 2003, Paris, France, July 1-4*, volume 2, pages 133–136, 2003. URL http://dx.doi.org/10.1109/ISSPA.2003.1224833.

[65] U. M. Fayyad and K. B. Irani. On the handling of continuous-valued attributes in decision tree generation. *Machine Learning, Kluwer Academic Publishers*, 8: 87–102, 1992.

[66] M. Hall. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, Hamilton, NZ: Waikato University, Department of Computer Science, 1998.

[67] S. Haykin. *Adaptive filter theory*. Prentice Hall, 3er ed, 1996.

[68] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2nd ed, 2000.

[69] B. Scholkopf and A.J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. MIT Press, 1st ed, 2001. ISBN 0262194759.

[70] C. Hsu, C. Chang, and C. Lin. A practical guide to support vector classification. *Department of Computer Science, National Taiwan University*, 2007. Online web resource: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

[71] Chris Duxbury, Juan Pablo Bello, Mike Davies, Mark Sandler, and Mark S. Complex domain onset detection for musical signals. In *Proceedings of the 6th International Conference on Digital Audio Effects, DAFx-03, London, England, September 8-11*, page 4, 2003.

[72] Simon Dixon. Automatic extraction of tempo and beat from expressive performances. *Journal of New Music Research*, 30:39–58, 2001.

[73] P. Cancela, M. Rocamora, and E. López. An efficient multi-resolution spectral transform for music analysis. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009. Kobe, Japan, October 26-30*, pages 309–314, 2009.

[74] P. Cancela, E. López, and M. Rocamora. Fan chirp transform for music representation. In *Proceedings of the 13th International Conference on Digital Audio Effects, DAFx-10, Graz, Austria, September 6-10*, pages 330–337, 2010. URL http://iie.fing.edu.uy/publicaciones/2010/CLR10.

[75] P. Flandrin. *Time-Frequency/Time-scale Analysis*. Wavelet Analysis and Its Applications. Academic Press, 1999.

[76] S. A. Abdallah and M. D. Plumbley. Unsupervised analysis of polyphonic music by sparse coding. *IEEE Transactions on Neural Networks*, 17(1):179–196, January 2006. ISSN 1045-9227. URL http://dx.doi.org/10.1109/TNN.2005.861031.

[77] D. Gabor. Theory of communication. *Journal I.E.E.*, 93(26):429–457, 1946.

[78] J. C. Brown. Calculation of a constant Q spectral transform. *Journal of the Acoustical Society of America*, 89(1):425–434, 1991. URL http://dx.doi.org/10.1121/1.400476.

[79] A. Ruedin and J. Vuletich. A novel basis for analysis of music. In *XIII Reunión de Trabajo en Procesamiento de la Información y Control, Rosario, Argentina, September 16-18*, pages 275–280, 2009.

[80] L. Cohen. Time-frequency distributions: a review. *Proceedings of the IEEE*, 77 (7):941–981, 1989.

[81] S. Mann and S. Haykin. The chirplet transform: physical considerations. *IEEE Transactions on Signal Processing*, 41(11):2745–2761, 1991.

[82] L. B. Almeida. The fractional fourier transform and time-frequency representations. *IEEE Transactions on Signal Processing*, 42(11):3084 – 3091, 1994.

[83] Luis Weruaga and Márian Képesi. The fan-chirp transform for non-stationary harmonic signals. *Signal Processing*, 87(6):1504–1522, June 2007.

[84] M. Képesi and L. Weruaga. Adaptive chirp-based time-frequency analysis of speech signals. *Speech Communication*, 48(5):474–492, 2006.

[85] C. Kereliuk and P. Depalle. Improved hidden Markov model partial tracking through time-frequency analysis. In *Proceedings of the 11th Conference on Digital Audio Effects, DAFx-08, Espoo, Finland, September 1-4*, pages 111–116, 2008.

[86] M. Bartkowiak. Application of the fan-chirp transform to hybrid sinusoidal+noise modeling of polyphonic audio. In *16th European Signal Processing Conference, EUSIPCO 2008, Lausanne, Switzerland, August 25-29*, page 5, 2008.

[87] P. Zhao, Z. Zhang, and X. Wu. Monaural speech separation based on multi-scale fan-chirp transform. In *Proceedings of the 2008 IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP '08. Las Vegas, USA, March 30 - April 4*, pages 161–164, 2008.

[88] Judith C. Brown and Miller S. Puckette. An efficient algorithm for the calculation of a constant Q transform. *Journal of The Acoustical Society of America*, 92(5): 2698–2701, 1992. URL http://dx.doi.org/10.1121/1.404385.

[89] Karin Dressler. Sinusoidal extraction using and efficient implementation of a multi-resolution FFT. In *Proceedings of the 9th International Conference on Digital Audio Effects, DAFx-06, Montreal, Quebec, Canada, September 18-20*, pages 247–252, 2006.

[90] J. S. Downie. The music information retrieval evaluation exchange (2005–2007): A window into music information retrieval research. *Acoustical Science and Technology*, 29(4):247–255, 2008.

[91] K. L. Kashima and B. Mont-Reynaud. The bounded-Q approach to time-varying spectral analysis. Technical Report STAN-M-28, Stanford University, 1985.

[92] Filipe C. C. B. Diniz, Iuri Kothe, Sergio L. Netto, and Luiz W. P. Biscainho. High-Selectivity Filter Banks for Spectral Analysis of Music Signals. *EURASIP Journal on Advances in Signal Processing*, 2007. URL http://dx.doi.org/10.1155/2007/94704.

[93] Masataka Goto. A Real-time Music Scene Description System: Predominant-F0 Estimation for Detecting Melody and Bass Lines in Real-world Audio Signals. *Speech Communication (ISCA Journal)*, 43(4):311–329, 2004. URL http://dx.doi.org/10.1016/j.specom.2004.07.001.

[94] P. Cancela. Tracking melody in polyphonic audio. MIREX 2008. In *Music Information Retrieval Evaluation eXchange*, 2008. URL http://www.music-ir.org/mirex/abstracts/2008/AudioMelodyExt_pcancela.pdf.

[95] A. de Cheveigné. Multiple F0 estimation. In D. Wang and G. Brown, editors, *Computational Auditory Scene Analysis: Principles, Algorithms and Applications*, pages 45–79. IEEE / Wiley, 2006.

[96] Martín Rocamora and Pablo Cancela. Pitch tracking in polyphonic audio by clustering local fundamental frequency estimates. In *Proceedings of the 9th Brazilian AES Congress on Audio Engineering , São Paulo, Brazil, May 17-19*, pages 80–87, 2011. URL http://iie.fing.edu.uy/publicaciones/2011/RC11/.

[97] A. Klapuri. A method for visualizing the pitch content of polyphonic music signals. In *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009. Kobe, Japan, October 26-30*, pages 615–620, 2009.

[98] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. RWC Music Database: Popular, Classical, and Jazz Music Databases. In *Proceedings of the 3rd International Conference on Music Information Retrieval, ISMIR 2002, Paris, France, October 13-17*, pages 287–288, 2002.

[99] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000. ISSN 0162-8828. URL http://dx.doi.org/10.1109/34.868688.

[100] F. R. Bach and M. I. Jordan. Learning spectral clustering, with application to speech separation. *Journal of Machine Learning Research*, 7:1963–2001, 2006.

[101] M. Lagrange, L. G. Martins, J. Murdoch, and G. Tzanetakis. Normalized cuts for predominant melodic source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):278–290, February 2008. ISSN 1558-7916. URL http://dx.doi.org/10.1109/TASL.2007.909260.

[102] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing, Springer Netherlands*, 17:395–416, 2007. ISSN 0960-3174. URL http://dx.doi.org/10.1007/s11222-007-9033-z.

[103] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters, Elsevier Science Inc.*, 31(8):651–666, June 2010. ISSN 0167-8655. URL http://dx.doi.org/10.1016/j.patrec.2009.09.011.

[104] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems 17*, pages 1601–1608. MIT Press, 2004.

[105] Greg Hamerly. Making k-means even faster. In *Proceedings of the 2010 SIAM International Conference on Data Mining (SDM 2010), Colombus, Ohio, USA, April 29 - May 1*, pages 130–140, 2010.

[106] Mathieu Lagrange and Sylvain Marchand. Estimating the instantaneous frequency of sinusoidal components using phase-based methods. *Journal of the Audio Engineering Society*, 55(5):385–399, 2007. URL http://www.aes.org/e-lib/browse.cfm?elib=14165.

[107] Perfecto Herrera and Jordi Bonada. Vibrato extraction and parameterization in the spectral modeling synthesis framework. In *Proceedings of the 1st Digital Audio Effects Workshop, DAFx98. Barcelona, Spain, November 19-21*, 1998.

[108] Emanuele Pollastri. Some considerations about processing singing voice for music retrieval. In *Proceedings of the 3rd International Conference on Music Information Retrieval, ISMIR 2002. Paris, France, October 13-17*, pages 285–286, 2002.

[109] The University of Iowa Musical Instrument Samples. Free available musical instruments recordings. URL http://theremin.music.uiowa.edu/MIS.html.

[110] Masataka Goto, H Hashiguchi, T Nishimura, and R Oka. RWC Music Database: Music Genre Database and Musical Instrument Sound Database. In *Proceedings of the 4th International Conference on Music Information Retrieval, ISMIR 2003, Baltimore, Maryland, USA, October 26-30*, pages 229–230, 2003.

[111] ccMixter. Creative Commons user provided multitrack songs and short audio clips. URL http://ccmixter.org/.

[112] Maarten van der Seijs. Improvements on time-frequency analysis using time-warping and timbre techniques. Master's thesis, Delft University of Technology, 2011.

[113] Pabitra Mitra, C.A. Murthy, and Sankar K. Pal. Density-based multiscale data condensation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:734–747, 2002. ISSN 0162-8828. URL http://doi.ieeecomputersociety.org/10.1109/TPAMI.2002.1008381.

[114] F. Villavicencio, A. Röbel, and X. Rodet. Improving LPC spectral envelope extraction of voiced speech by true-envelope estimation. In *Proceedings of the 2006 IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP '06. Toulouse, France. May 14-19*, 2006. URL http://dx.doi.org/10.1109/ICASSP.2006.1660159.

[115] Chanwoo Kim, Kwang-deok Seo, and Wonyong Sung. A robust formant extraction algorithm combining spectral peak picking and root polishing. *EURASIP Journal on Advances in Signal Processing*, 2006:33–33, January 2006. ISSN 1110-8657. URL http://dx.doi.org/10.1155/ASP/2006/67960.

[116] Douglas OShaughnessy. Formant estimation and tracking. In Jacob Benesty, M.Mohan Sondhi, and Yiteng(Arden) Huang, editors, *Springer Handbook of Speech Processing*, pages 213–228. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-49125-5. URL http://dx.doi.org/10.1007/978-3-540-49127-9_11.