Music and Speech Analysis Using the '*Bach*' Scale Filter-Bank

A Thesis
Submitted for the Degree of
Master of Science
in the Faculty of Engineering

By
G. Ananthakrishnan

Department of Electrical Engineering
Indian Institute of Science
Bangalore - 560 012
India

April 2007

Dedicated to

My Father, Mother and Aparna

Acknowledgements

Before I start thanking everybody and everything around me, i'd like to tell you how I usually spent my day at the Institute. Figure 0.1, gives you an idea about, what sort of a life I had at IISc.

Yes, my life WAS as confusing as the Figure 0.1. But a lot of people helped me 'figure' my way out! Some gave me advice, some gave me opportunity, some gave me empathy and some were 'just there' for me when I needed them. The two and a half years that I spent at IISc were fantastic, and I want to thank every one of the trees, that I have tried to show in Figure 0.1 in vain, for just being where they were. Every spot in IISc will remain etched in my mind for some incident, some joke, some idea or some brilliant moment with my friends on campus.

So, let me start with thanking my guide and my mentor, Prof. A. G. Ramakrishnan, who motivated me to take up music as my research topic, a field that is very close to my heart. A marriage of music and signal processing is definitely the most delicious combination of anything I've had in my life (including all the Belgian and Swiss chocolates). I'd also like to thank all the professors who taught me, especially Prof. K. R. Ramakrishnan, Prof. M. N. Murthy and Prof. T. V. Srinivas, who not only imparted knowledge, but also left me with a permanent thirst for exploring the unknown.

My lab-mates, Thot, Pati, Raj, Praveen and Lakshmish in the first year, and later Ranjani, Farshad, Suresh, Ritesh, Amrik and Arvind gave me great company in heated discussions, dejection on failures, celebration on success and of course the late night movies. I must thank them for me being able to spend the '9' hours working (ok, supposed work) and more importantly the 2 hours of sleep in the lab (thanks for not disturbing me). I must

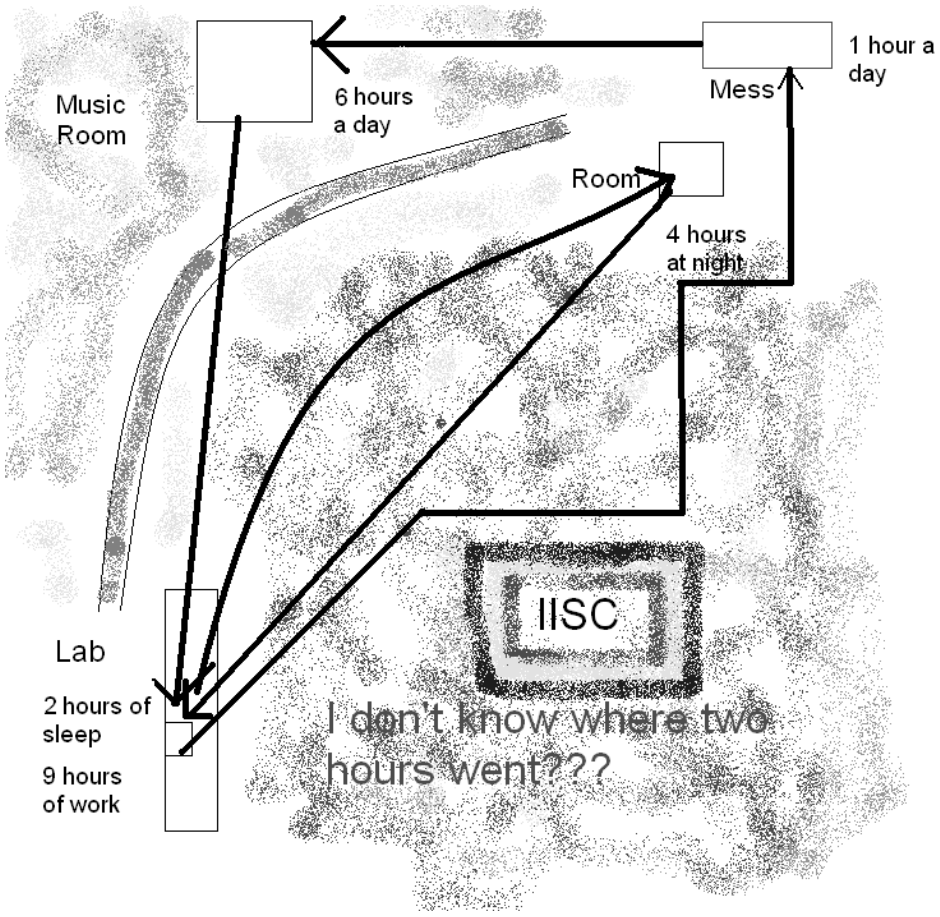

Figure 0.1: A usual day at the Institute

specially thank Ranjani who helped with running some of the simulations and tabulating the data.

Now I turn to the music room, where I definitely had spent more time than I ought to have. I want to thank Mata Priteshwari, Mariamma Temple Nirmala, Great Great K.R.S.P, Vinod aba, Sandeep Samadhana, Prashant, Yanni Karthik, Anand, 'Five star' Anando, Aditidi, Rumki, Pradeep, Vivek, Reuben, DDey, Anirban, Garima, Mahua, wawww.arunkumarer.com, Sheron, Sujitda, JJ, Jyothi, Aashia, Prashanthi, Karthik sir, Kalyani, Deepak, Emti, Niran-

jan..... wow!! what a list and i'm sure i've missed a lot of people!! Life spent with these guys was awesome!! These were the moments that relaxed me and gave me the impetus to overcome failures and try again and again till I succeeded. I'd specially like to mention Pritesh, Vinod, Sathya, Karthik and Prashant who were really great friends to me and were always 'just there' for me.

I spent time in my room and the mess just because I had to sleep and eat. But they helped survive in IISc, so thanks to them too. Prachi, Tarun, Piyush and Kripa and of course, Pritesh, gave me good company and a lot of laughter during my super-fast meals at the mess.

Finally i'd like to thank my family: Mom, Dad, Aparna, Pati, Bhargavi, Mama, Tata and Ammama for supporting me always and believing in me.

Abstract

The aim of this thesis is to define a perceptual scale for the 'Time-Frequency' analysis of music signals. The equal tempered '*Bach*' scale is a suitable scale, since it covers most of the genres of music and the error is equally distributed for each semi-tone. However, it may be necessary to allow a tolerance of around 50 cents or half the interval of the *Bach* scale, so that the interval can accommodate other common intonation schemes. The thesis covers the formulation of the *Bach* scale filter-bank as a time-varying model. It makes a comparative study with other commonly used perceptual scales. Two applications for the *Bach* scale filter-bank are also proposed, namely automated segmentation of speech signals and transcription of singing voice for query-by-humming applications.

Even though this filter-bank is suggested with a motivation from music, it could also be applied to speech. A method for automatically segmenting continuous speech into phonetic units is proposed. The results, obtained from the proposed method, show around 82% accuracy for the English and 85% accuracy for the Hindi databases. This is an improvement of around 2 - 3% when the performance is compared with other popular methods in the literature. Interestingly, the *Bach* scale filters perform better than the filters designed for other common perceptual scales, such as *Mel* and *Bark* scales.

'Musical transcription' refers to the process of converting a musical rendering or performance into a set of symbols or notations. A query in a 'query-by-humming system' can be made in several ways, some of which are singing with words, or with arbitrary syllables, or whistling. Two algorithms are suggested to annotate a query. The algorithms are designed to be fairly robust for these various forms of queries. The first algorithm is a frequency selection based method. It works on the basis of selecting the most likely frequency components at

any given time instant. The second algorithm works on the basis of finding time-connected contours of high energy in the 'Time-Frequency' plane of the input signal. The time domain algorithm works better in terms of instantaneous pitch estimates. It results in an error of around 10 - 15%, while the frequency domain method results in an error of around 12 - 20%.

A song rendered by two different people will have quite a few different properties. Their absolute pitches, rates of rendering, timbres based on voice quality and inaccuracies, may be different. The thesis discusses a method to quantify the distance between two different renderings of musical pieces. The distance function has been evaluated by attempting a search for a particular song from a database of a size of 315, made up of songs sung by both male and female singers and whistled queries. Around 90 % of the time, the correct song is found among the top five best choices picked.

Thus, the *Bach* scale has been proposed as a suitable scale for representing the perception of music. It has been explored in two applications, namely automated segmentation of speech and transcription of singing voices. Using the transcription obtained, a measure of the distance between renderings of musical pieces has also been suggested.

Contents

List of Tables

List of Figures

Nomenclature

Symbols	: **Definitions**
$\aleph$	: Set of Natural numbers
M	: Number of band-pass filters or the length of the frequency vectors
k	: Time instant (no. of samples)
T	: Total number of samples
F_s	: Sampling frequency
$base$	: arbitrary starting frequency for the filter-bank
n	: The relative '*Bach* frequency
$s(k)$	: Input speech or audio signal
$h_n(k)$	: the coefficients of the n^{th} filter of the filter-bank
$F_n(k),\ F_k(n)$	: 2-D feature vector representation
$f_c(n)$	: Centre frequency of the n^{th} filter
$S_{scale}(n)$	: Difference in frequency between the n^{th} filter and $(n+1)^{th}$ filter for an arbitrary frequency '*scale*'
f	: frequency in Hz
$scale(f)$	: function converts frequency f in Hz to an arbitrary frequency '*scale*'
$f(scale),\ f$	: function converts frequency in an arbitrary frequency '*scale*' to f in Hz
$mel(f)$	: function converts frequency f in Hz to '*Mel*' frequency
$z(f)$	: function converts frequency f in Hz to '*Bark*' frequency
$e(f)$	: function converts frequency f in Hz to '*ERB*' frequency

f_{scale}	: Frequency in the arbitrary '*scale*'
f_{mel}	: Frequency in the '*Mel*' scale
f_z	: Frequency in the '*Bark*' scale
f_e	: Frequency in the '*ERB*' scale
f_{Bach}, n	: Frequency in the relative '*Bach*' scale
$Bach(f)$	: function converts frequency f in Hz to relative '*Bach*' frequency
$B_{scale}(f)$	: function finds the Bandwidth of a filter with centre frequency f Hz for an arbitrary frequency '*scale*'
$B_{mel}(f)$	: function finds the Bandwidth of a filter with centre frequency f Hz for '*Mel*' scale
$B_z(f)$	: function finds the Bandwidth of a filter with centre frequency f Hz for '*Bark*' scale
$B_e(f)$	: function finds the Bandwidth of a filter with centre frequency f Hz for '*ERB*' scale
$B_{bachL}(f)$	: function finds the Bandwidth of a filter with centre frequency f Hz for '*Bach* linear' scale
$B_{bachN}(f)$	: function finds the Bandwidth of a filter with centre frequency f Hz for '*Bach* non-linear' scale
$f_b(n)$	: Bandwidth of the n^{th} filter
$N(n)$	: Number of coefficients used to construct the n^{th} filter
$E(f)$	: Equal loudness pre-emphasis at f Hz frequency at 40db level
W, w	: length of speech in 'seconds' and 'number of samples' as tolerance for error
$DF(k)$	: Distance function defined at sample k for classes X and Y defined on either side of the k
R, r	: length of speech in 'seconds' and 'number of samples' used for peak detection
EDM	: Mean Euclidean Distance
$EDML$	: Log Mean Euclidean Distance
$NEDML$	: Normalized Log Mean Euclidean Distance
KLD	: Kullback-Leibler distance
ISD	: Itakura-Saito distance

MD	: Mahalanobis distance
MPB	: Matched Phoneme Boundary
$\sigma^2(k)$	: Spectral variance at sample k
SF	: Spreading Function
K	: Region for selecting a peak corresponding to a sinusoidal component
T_{MN}	: Threshold for tone masking noise
T_{MT}	: Threshold for tone masking tone
$\overline{S_k}$	: Vector containing indexes of significant sinusoids
$\overline{FS_k}$	: Vector containing corresponding amplitudes of significant sinusoids
I_k	: The index of the highest amplitude among $F_k(n) \forall n$
$F_I(k)$	: The value of $F_k(I_k)$
HS_k	: Harmonically richest set for the 'k^{th}' time sample
$\widetilde{\lambda_{auto}}(k)$	: Pitch contour estimate for for the 'k^{th}' time sample
Ψ	: Tolerance allowed for note onset timing in secs
ψ	: Tolerance allowed for note onset timing in number of samples
ϑ	: Note onset estimate based on change in pitch
ξ	: Note onset estimate based on change in spectral variance
ζ	: Note onset due to change in Difference Normalized Spectral Variance
Θ	: Threshold used for calculating ζ
NSV	: Normalized spectral variance
$DNSV$	: First Difference Normalized spectral variance
$\widehat{\overline{A}}$	: Operator on vector $\overline{A}$ with a boolean output
$N_o(k)$	: Is 1 if 'k' is a note onset, 0 otherwise
$\lambda_{auto}(k)$, $\overline{\lambda_{auto}}$	: Final pitch contour estimate for for the 'k^{th}' time sample
τ	: Threshold for Detecting 2-D energy contours
N_c	: Number of Energy contours
$\lambda_i(k)$	: i^{th} energy contour $\forall k \in [1, T]$
$\Gamma_X(i)$	: Representation of the i^{th} note
Υ_{start}	: represents the note onset time (sec) of the i^{th} note

λ_{start}	: the starting relative frequency of the i^{th} note
λ_{end}	: the ending relative frequency of the i^{th} note
$\lambda_{man}(k)$, $\overline{\lambda_{man}}$	: Pitch contour obtained by Manual transcription
μ_X	: The mean of the pitch contour $\overline{\lambda_X}$
%E	: Percentage error of generated pitch contour with respect to manually generated pitch contour
%D	% Deletions
%I	% Insertions
%T	% Correct transcription
%D	% Correct silence detection regions
$SP_X(i)$	: Starting point of i^{th} phrase of rendering X
$EP_X(i)$	: Ending point of i^{th} phrase of rendering X
T_X	: Number of samples in rendering X
$\widehat{E_X(k)}$	: Energy contour of rendering $X \forall k \in [1, T_X]$
$D(i, j)$	: Distance between phrases or renderings i and j
F_{ft}	: High resolution Fourier transform of the signal
ω	: Angular frequency
$[RL_{base}(n), RH_{base}(n)]$	: The range of the n^{th} filter in angular frequency
$S_{ft}(base)$	: The sum contribution of magnitude of the Fourier transform of the signal towards the filter-bank designed for *base* frequency
tr	: Time resolution in seconds

Chapter 1

Introduction

1.0.1 History of TF Representations

Speech and audio signals have engaged the interest of man since time immemorial. The Indian grammarians, as early as the 7th century CE had tried to categorize and study speech signals. Later, Sir Isaac Newton recognized the relation between vowel qualities and resonances. Kratzenstein in 1779 and von Kempelen in 1791 [1], were among the first to devise mechanical resonators, which made successful imitations of the various phones used in speech. Later, Bell [2] formalized the study of speech in the form of the universal phonetic language. The Bell research labs contributed to the field of electronic speech synthesis in a big way. The first most significant contribution was the 'VODER' [3] of Homer Dudley in 1939, which was later modified and suitably enhanced using complete band-compression systems, based on the principles of speech analysis and synthesis. The most notable of these was the 'Vocoder' [1], which spawned a subfield of communication engineering.

Practically every aspect of speech communication has been greatly benefitted from the widespread application of short-time analysis methods. Systems for speech recognition, synthesis and coding, segment the speech into short intervals, analyzing each segment of speech under the implicit assumption that the signal is stationary over the interval [4]. The work horse of most linguistic experts and speech and audio engineers has been the spectrogram (illustrated in figure 1.1) ever since its invention by Bell laboratories in 1942. It is true that the audio signal spectrogram contains a great deal of detailed information about speech waves. Its real value, however, lies in the organization of that information into a 'picture'.

The 'Pattern Playback' [5], developed by Frank Cooper in 1950, worked like the inverse

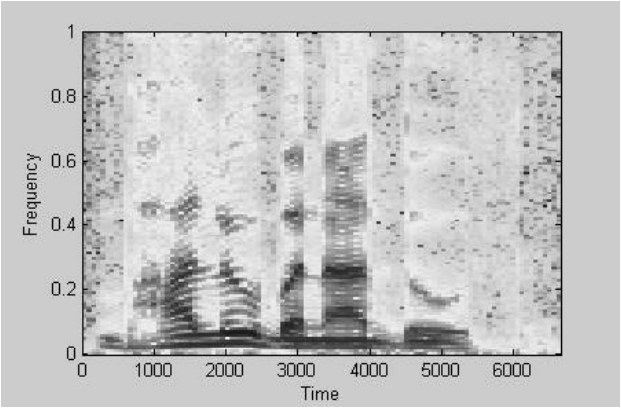

Figure 1.1: Spectrogram of the Hindi sentence '/b^{h}a:lu: bana: du:t̲/'

of a sound spectrograph, i.e. it had a mechanism to convert a spectrograph into a corresponding speech like sound signal. Instead of real spectrograms, it was also possible to use fake spectrograms painted by hand. By means of perception experiments performed with signals produced in this way, it was possible to obtain a series of new cognitions about the perceptual role of various details in the spectra of speech signals.

However, the Time-Frequency (TF) trade-off of the spectrogram is a well-known drawback of short-time analysis. Using a short window for good time resolution reduces the frequency resolution, and vice-versa. Another interesting drawback is that the frequency response of the vocal tract is very different when the glottis is open than from the response when it is closed. There is a large variation in the short-time energy of the speech waveform within a single pitch period during the closure. Thus, frequency analysis using a window duration in the order of a single pitch period will produce significantly different results depending on the location within that duration. Using a window that encompasses several pitch periods will effectively smooth out these variations, but will also blur transient events, such as plosives.

Besides the trade-off in resolution, there is an underlying inaccuracy of short-time analysis methods for non-stationary signals. While the spectrogram gives an accurate estimate of the instantaneous frequency of a pure tone, it has been shown by Smits [6] that amplitude modulation causes a bias in the peak of the spectrogram away from the true instantaneous frequency. Using the peak of a spectrogram to measure the instantaneous frequency or

equivalently the sweep rate of a formant transition will result in a biased estimate, underestimating the sweep rate. [7]

In order to overcome these shortcomings in analyzing non-stationary signals, several TF approaches have been suggested. Broadly, they are categorized into three, namely TF distributions, statistical or parametric and perceptual models. Wigner distribution, Choi-Williams distribution, CK distribution, and Cohen-Posch distribution are some of the commonly used TF distributions for analyzing audio and speech signals [7]. Among statistical/parametric methods, HMM based methods [8], time-varying AR [9], ARMA models [10], modeling the non-stationarity using Kalman filter representation [11], and AM-FM decomposition [12] are the common approaches.

A great deal of TF research has been motivated by human signal interpretation. Since the task at hand is audio and speech analysis, human hearing may provide insight into defining a potentially useful speech representation. The first of the perceptual scales, namely, the 'Mel' scale [13] was proposed by Stevens in 1936. It is a subjective scale for the measurement of pitch constructed from determinations of the half-value of pitches at various frequencies. The corresponding filter-bank is roughly described as linear at low frequencies, with fixed bandwidth, and logarithmic at higher frequencies, with constant-Q bandwidth. Mel spectrograms and Mel cepstral co-efficients [14] have been used widely.

Another perceptual scale, the Bark scale [15], is based on the critical bandwidth of masking. It has been reasoned that the progressive reduction in bandwidth of a masking noise will reduce the threshold of a pure tone located at the center of the band only when the noise components actually effective in masking the tone begin to be eliminated. The break should occur at the critical bandwidth. Reducing the bandwidth to less than critical should lower the masked threshold by an amount equal in decibels to the reduction in total power; increasing bandwidth beyond critical width should leave the threshold unchanged. Filter-banks constructed with critical bandwidths have also been popular.

Perceptual properties have also been incorporated into short-time parametric speech processing techniques. One such approach is perceptual linear predictive (PLP) analysis [16]. In PLP, the speech spectrum is first pre-emphasized and transformed to a critical band

spectrum and compressed using a cube-root operation to approximate the intensity-loudness relationship [17]. The resulting perceptual spectrum is then inverse-transformed to the auto-correlation domain, and then the Yule-Walker equations are solved for the all-pole model of the perceptual spectrum. One potential advantage of PLP is that fewer parameters are needed to model the speech spectrum due to the data reduction inherent in the perceptual representation. An approach to the robustness in a noisy environment, inspired by certain perceptual phenomena, is known as relative spectra (RASTA) [18]. In this technique, speech parameters are band-pass filtered in time. RASTA provides reduced sensitivity to additive noise and channel effects.

1.0.2 Theory of Music Semi-tones

In case of music signals, the most apparent pitch based perceptual property is the division of an octave into 12 semi-tones. This is valid for almost all genres of music. (There are genres which use 19, 22 or 31 semi-tones per octave). The oldest tuning intonation, namely the Pythagorean intonation, used the system of pure fifths. This tuning was used in the medieval era. It had several short-comings, namely the 'dissonant third' and 'wolf fifth' [19]. The '*Just*' or the 'Helmholtz' tuning for the chromatic scale [20], developed in the Renaissance, was based on consonance of the notes. This tuning used the fact that if the interval between two notes is a ratio of small integers (such as 2/1, 3/2, or 4/3), they sound good together - they are consonant rather than dissonant. The '*Just*' intonation used these ratios. But, though some intervals of notes sound perfectly consonant, some other intervals were quite dissonant. Besides, musicians could not play a particular song at any key they wished to.

The great 17^{th} century musician J. S. Bach implemented the '*equal tempered scale*' [21], which allowed musicians greater flexibility. It is a compromise tuning scheme. The equal tempered system uses a constant frequency multiple of $2^{(1/12)}$ or 1.059 between two consecutive notes of the chromatic scale. This ratio is equal to 100 cents in the perceptual scale. Hence, in spite of playing a musical piece in any key of choice, the dissonance is kept below 1% as shown in table 1.1 [21]. The '*Bach*' scale or the '*Equal tempered intonation*' is the most common scale in all modern Western and Indian music. The modern music pieces

played with any other intonation method would sound rather dissonant, but medieval music pieces played with a '*Bach*' intonation sound passably consonant

Table 1.1: Comparison between just intonation and equal tempered intonation

Name	Ratio in 12 Equal Tempered Intonation	Just Intonation Ratio	Percentage Difference
Unison	1.0	1/1	0.00%
Minor Second	1.059	16/15	-0.68%
Minor third	1.189207	6/5	-0.91%
Major third	1.259921	5/4	+0.79%
Perfect fourth	1.334840	4/3	+0.11%
Diminished fifth	1.414214	7/5	+1.02%
Perfect fifth	1.498307	3/2	-0.11%
Minor sixth	1.587401	8/5	-0.79%
Major sixth	1.681793	5/3	+0.90%
Minor seventh	1.781797	16/9	+0.23%
Major seventh	1.887749	15/8	+0.68%
Octave	2.000000	2/1	0.00%

In the context of Indian Classical Music, the interval between two chromatic notes is divided into 1, 2 or 3 '*shruthis*' giving a total of 22 '*shruthis*'. Each *shruthi* corresponds to approximately 50 cents in the perceptual scale. It must be noted that the octave is divided into 22 '*shruthis*' only for tuning purposes. For a given '*raaga*' only 12 specifically tuned notes are available per octave. Table 1.2 shows the intonation scheme for the 22 '*shruthis*' of Indian classical music [22].

We can observe from table 1.2 that they are not evenly spaced. The difference between each of the 22 '*Shruthis*' and the closest equal tempered scale is always less than equal to 20 cents.

Table 1.2: 22 '*shruthis*' of Indian classical music

Name of '*shruthi*'	Frequency ratio	Frequency (cents)	Closest Frequency in Equal tempered scale (cents)
'*Shadja*'	1/1	0	0
'*Ekashruthi Rishabha*'	256/243	90	100
'*Dvishruthi Rishabha*'	16/15	112	100
'*Trishruthi Rishabha*'	10/9	182	200
'*Chatusshruthi Rishabha*'	9/8	204	200
'*Shuddha Gandhara*'	32/27	294	300
'*Sadharana Gandhara*'	6/5	316	300
'*Antara Gandhara*'	5/4	386	400
'*Chyutha Madhyama Gandhgara*'	81/64	408	400
'*Suddha Madhyama*'	4/3	498	500
'*Tivra Suddha Madhyama*'	27/20	520	500
'*Prati Madhyama*'	45/32	590	600
'*Chyuta Panchama Madhyama*'	729/517 or 64/45	610	600
'*Panchama*'	3/2	702	700
'*Ekashruthi Daivatha*'	128/81	792	800
'*Dvishruthi Daivatha*'	8/5	814	800
'*Trishruthi Daivatha*'	5/3	884	900
'*Chatusshruthi Daivatha*'	27/16	906	900
'*Shuddha Nishada*'	16/9	996	1000
'*Kaisiki Nishada*'	9/5	1018	1000
'*Kakali Nishada*'	15/8	1088	1100
'*Chyuta Shadja Nishada*'	243/128	1110	1100
'*Tara Shadja*'	2/2	1200	1200

1.0.3 Motivation for '*Bach*' Scale Representation and Applications of the Representation

The aim of this thesis is to define a perceptual scale for 'Time-Frequency' analysis of music signals. The equal tempered '*Bach*' scale seems the most suitable scale, since it covers most

of the genres of music and the error is equally distributed for each semi-tone. However, it may be necessary to allow a tolerance of around 50 cents or half the interval of the '*Bach*' scale, so that the interval can accommodate the other common intonation schemes. The thesis covers the formulation of the '*Bach*' scale filter-bank and discusses two applications of the filter-bank; one in Automated Segmentation of Speech Signals and the other in Transcription of Singing Voices. Further, the thesis discusses a method to quantify the distance between two different renderings of a musical piece.

Chapter 2 develops the formulation of the time-frequency representation obtained from the '*Bach*' scale filter-bank along with its properties. It must be noted that the representation is a shift from the usual short-time stationary model of speech and audio signals. The chapter also shows a comparative study between the proposed '*Bach*' scale and other perceptual based scales.

Even though this filter-bank is built with a motivation which lies in music, it could also be applied in speech applications. Chapter 3 shows an application of the '*Bach*' filters in automated speech segmentation. It suggests a method for automatically segmenting continuous speech into phonetic units. The method which employs a time-varying model to represent speech, shows slightly better performance as compared to previously suggested methods. It is also shown in Chapter 3 that the '*Bach*' scale performs comparably, if not better than other perceptual scales defined in the context of speech signals.

Chapter 4 deals with the automated transcription of singing voices. Two algorithms are suggested and both show encouraging results. The transcription is compared against manually transcribed data. The algorithms are designed so that the transcription is reasonably robust for the possible ways a query can be made by a user, for example, humming, singing with words and whistling. Another important feature of the algorithm is that only relative pitch is obtained as compared to an absolute pitch. In the context of a Query by Humming system, this is acceptable, since queries will have a pitch which is different from the original rendition of the musical piece.

The same song rendered by two different people will have quite different properties. They may have a different absolute pitch, different rates of rendering, different timbres based on

voice quality and different inaccuracies. So, in order to find distances between two different renderings, the two renderings need to be normalized based on the absolute values of pitch and rate of rendering. By pitch extraction, the variation in timbre and inaccuracies in pitch are taken care of. However, the varying rates are not compensated for. In order to address this, a dynamic distance measure needs to be calculated. Chapter 5 defines a distance function to find the distance between any two renderings of musical pieces.

Chapter 2

Formulation of the Filter-bank

This chapter deals with formulation of a generalized TF representation considering a time-varying model of the speech signal. The formulation has been adapted for the '*Mel*', '*Bark* and '*Equivalent Rectangular Bandwidth*' perceptual frequency scales. The '*Bach*' scale formulation has also been proposed and a comparative analysis has been made with the above scales.

2.1 Time-Frequency Representation

The formulation of the TF representation is similar to the Filter-Bank Summation (FBS) method [23]. The speech signal is filtered by a bank of 'M' band-pass filters, each shifted in frequency by a fixed factor. So we have 'M' filtered versions of the same speech signal. Consider the'n^{th}' such version of the signal. The energy around the 'n^{th}' frequency component of the signal around a time instant 'k' will be equal to the 'k^{th}' output energy of the 'n^{th}' filter-bank.

$$F_k(n) = F_n(k) = abs(h_n(k) \bigotimes s(k)) \tag{2.1}$$

where $s(k)$ is the input speech signal. $h_n(k)$ is the band-pass filter designed around centre frequency 'n'. The symbol $\bigotimes$ represents linear filtering. The feature vectors, $F_k(n) \forall k \;\; 1 \leq k \leq T$ and $F_n(k) \forall n \;\; 1 \leq n \leq M$, are the two 2-D representations of the signal $s(k)$. The two representations will be used interchangeably henceforth. It must be noted here that this representation of a time-series (signal $s(k)$) is a time-varying representation as against a short time representation, since every single time sample has a unique frequency representation. The first filter is centered around a '*base*' frequency, which is an arbitrary starting frequency.

Usually, it is in the range of 20 to 80 Hz, since the spectral information of speech and audio signals starts in that region. The filter-bank is only an analysis filter-bank and not a perfect reconstruction one. So, the strict condition of perfect reconstruction, the 'FBS' constraint [23], is not applicable.
Assuming 'm' filters per octave, we can calculate 'M', the maximum number of filters, starting at the '*base*' frequency and with a sampling frequency of 'F_s' Hz

$$M = m * log_2\left(\frac{F_s}{2 * base}\right) \tag{2.2}$$

Since we are looking at a musical perceptual scale, namely the 12 semi-tonal equal tempered *Bach* scale, we take $m = 12$. In order to make a just comparison with the other perceptual scales, we assume the same parameters of m and *base* for all scales, e.g. for $m = 12$, F_s= 11000 Hz and $base = 55$ Hz, we obtain $M = 79$.

The centre frequencies $(f_c(n) \forall n \quad 1 \leq n \leq M)$ of the filters of an arbitrary scale are obtained by uniformly dividing the respective frequency scales by M.

$$S_{scale}(n) = scale(f_c(n)) - scale(f_c(n-1)) \tag{2.3}$$

where '$S_{scale}(n)$' is the shift in the frequency for every next filter and the function *scale* converts the frequency in Hz to the particular scale.

2.2 Comparison of Scales

Filter-banks have been designed for four auditory scales namely the '*Mel*' or the Radio scale, the '*Critical Band Rate*' scale (CRB) or '*Bark*' scale, '*Equivalent Rectangular Band*' (ERB) rate scale and the '*Bach*' scale. The first filter for all the banks is shifted by '*base*' frequency and number of filters per octave is m.

2.2.1 Mel Scale

The approximation of the experimental data for *Mel* scale is given by Beranek [24],

$$mel(f) = 1127 * ln\left(1 + \frac{f}{700}\right) \tag{2.4}$$

where f is the frequency in Hz. The relation between *Mel* scale and the frequency in Hz is shown in figure 2.1. The inverse relation is given by

$$f(mel) = 700 * \left(e^{\frac{mel}{1127}} - 1 \right) \tag{2.5}$$

The shift for every next filter is given by

$$S_{mel} = \frac{mel(F_s/2) - mel(base)}{M} \tag{2.6}$$

To calculate the actual centre frequencies of the filters,

$$\begin{aligned} f_c(1) &= mel(base) \\ f_c(2) &= f(mel(f_c(1)) + S_{mel}) \\ &\vdots \\ f_c(n) &= f(mel(f_c(n-1)) + S_{mel}) \end{aligned} \tag{2.7}$$

The bandwidth formulation is given by

$$B_{mel}(f) = \frac{700 * \left(e^{\frac{mel(f) + S_{mel}}{1127}} - e^{\frac{mel(f) - S_{mel}}{1127}} \right)}{2} \tag{2.8}$$

Figure 2.2 shows the bandwidth plot for the *Mel* scale.

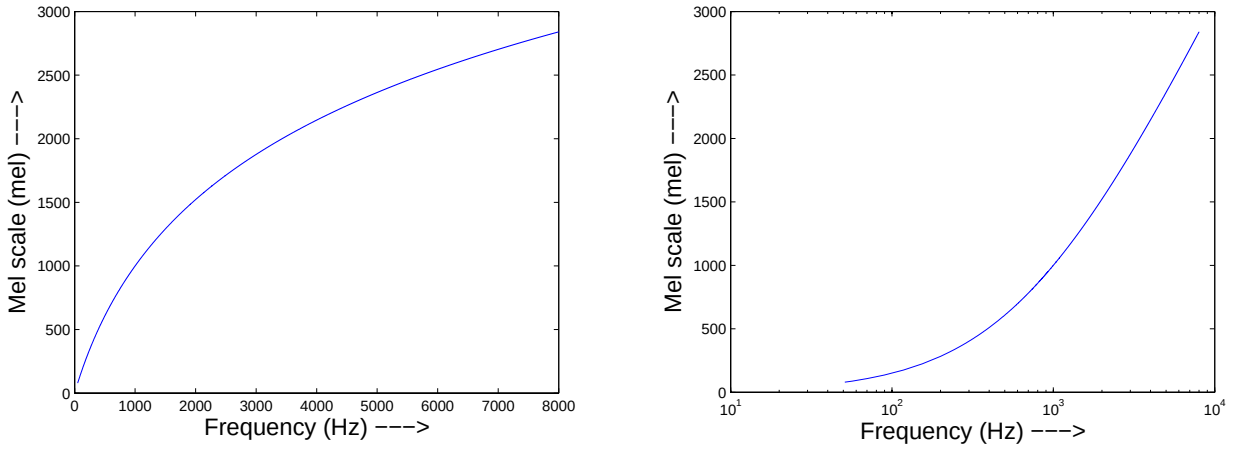

Figure 2.1: The *Mel* scale as (a) linear and (b) semilog plots

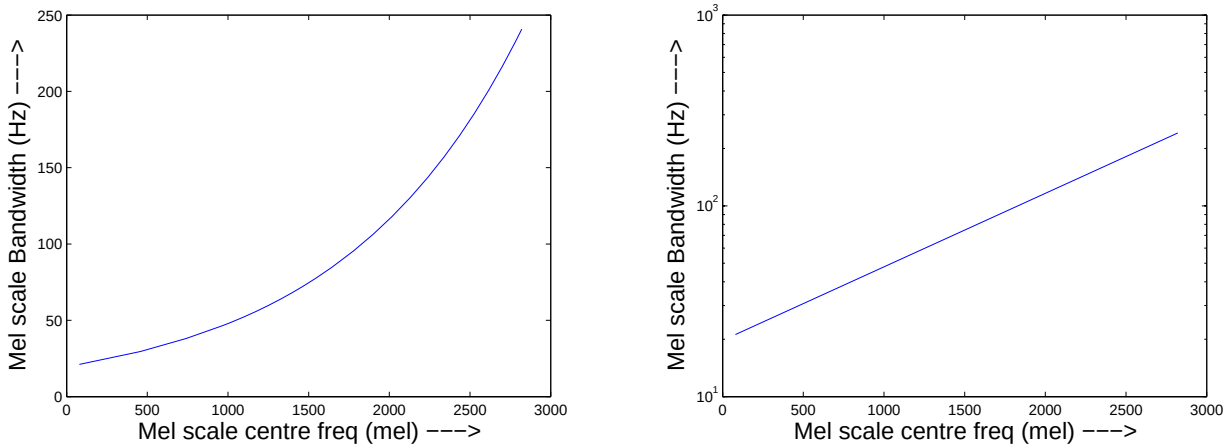

Figure 2.2: The bandwidth of *Mel* scale as (a) linear and (b) semilog plots

2.2.2 Bark Scale

The approximation of the experimental data for *Bark* scale is given by Beranek [24],

$$z(f) = \frac{26.81}{1 + \frac{1960}{f}} - 0.33 \tag{2.9}$$

where f is the frequency in Hz. The relation between the *Bark* scale and the frequency in Hz is shown in figure 2.3. The inverse relation is given by

$$f(z) = \frac{1960}{\frac{26.81}{(z + 0.33)} - 1} \tag{2.10}$$

The shift for every next filter is given by

$$S_z = \frac{z(F_s/2) - z(base)}{M} \tag{2.11}$$

To calculate the centre frequencies of the filters,

$$\begin{aligned} f_c(1) &= z(base) \\ f_c(2) &= f(z(f_c(1)) + S_z) \\ &\vdots \\ f_c(n) &= f(z(f_c(n-1)) + S_z) \end{aligned} \tag{2.12}$$

The bandwidth formulation is given by

$$B_{Bark}(f) = \frac{52548}{z(f)^2 - 52.56 * z(f) + 690.39} \tag{2.13}$$

Figure 2.4 shows the bandwidth plot for the *Bark* scale.

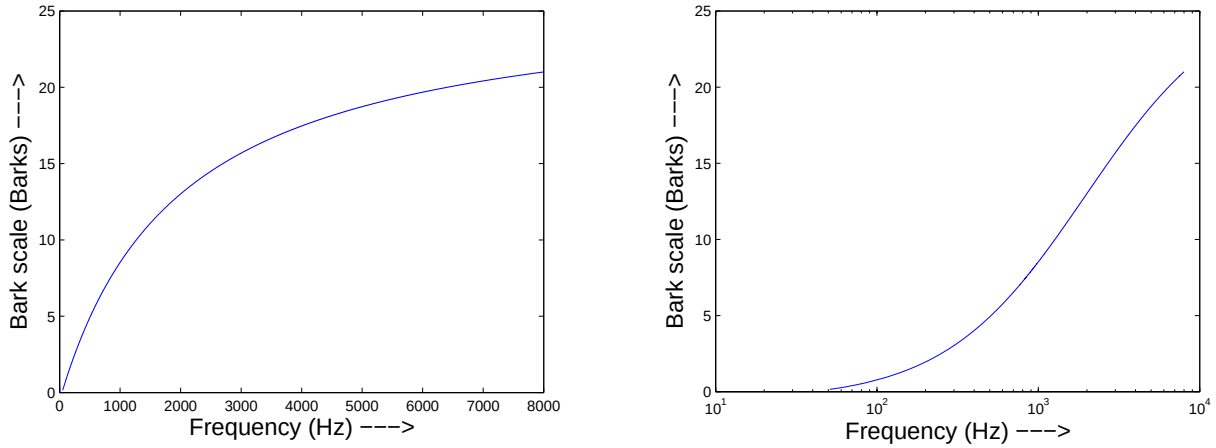

Figure 2.3: The *Bark* scale as (a) linear and (b) semilog plots

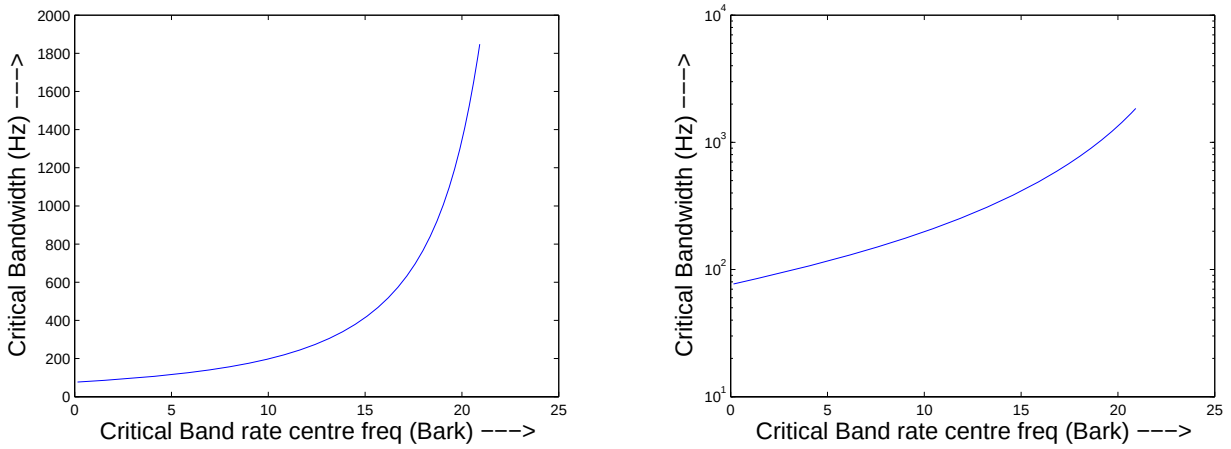

Figure 2.4: The bandwidth of *Bark* scale as (a) linear and (b) semilog plots

2.2.3 Equivalent Rectangular Bandwidth (ERB) Scale

The approximation of the experimental data for *ERB* scale is given by Moore [25],

$$e(f) = 11.17 * ln\left(\frac{f + 312}{f + 14675}\right) + 43.0 \tag{2.14}$$

where f is the frequency in Hz. The relation between ERB scale and the frequency in Hz is shown in figure 2.5. The shift for every next filter and the centre frequencies of the filters are calculated as in the previous cases.
The bandwidth formulation is given by

$$B_e(f) = 6.23 * 10^{-6} * f^2 + 9.393 * 10^{-2} * f + 28.52 \tag{2.15}$$

Figure 2.6 shows the bandwidth plot for the ERB scale.

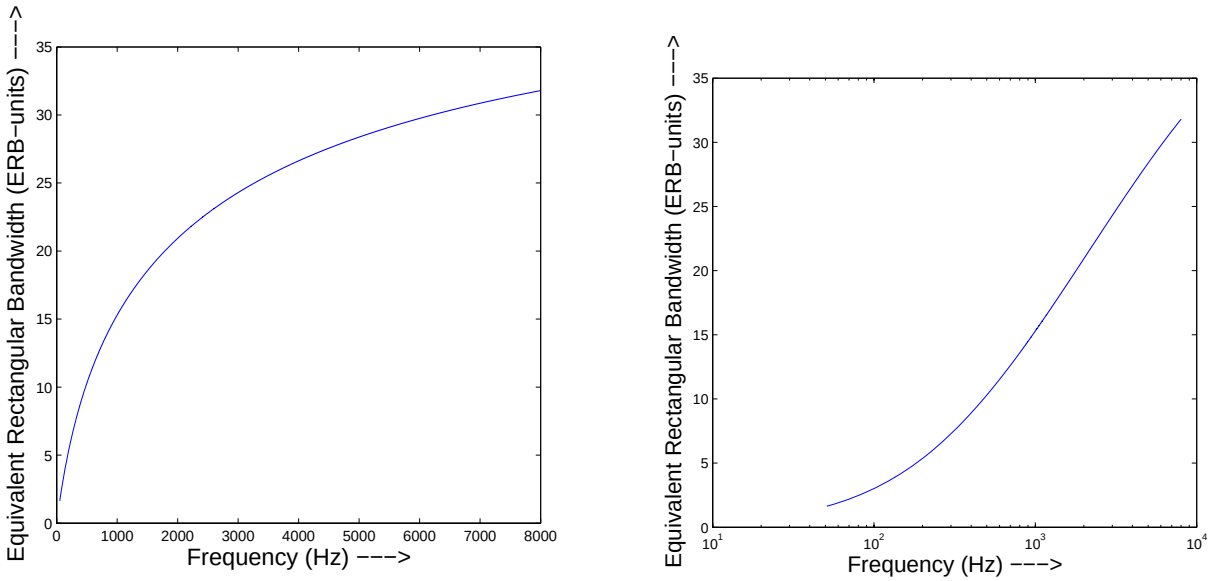

Figure 2.5: The ERB scale as (a) linear and (b) semilog plots

2.2.4 Bach Scale

The formulation of the relative $Bach$ scale which is a constant Q scale (Q = 17) [26] is as follows

$$Bach(f) = 12 * log_2\left(\frac{f}{base}\right) \tag{2.16}$$

where f is the frequency in Hz. The relation between the relative $Bach$ scale and the frequency in Hz is shown in figure 2.7. The inverse relation is given by

$$f(Bach) = base * 2^{\frac{Bach}{12}} \tag{2.17}$$

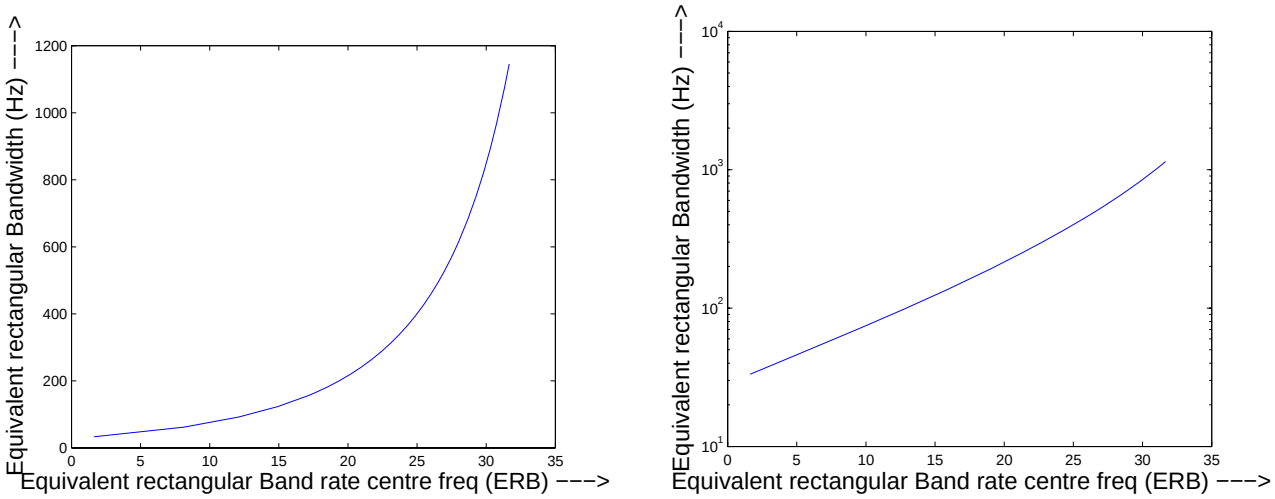

Figure 2.6: The bandwidth of *ERB* scale as (a) linear and (b) semilog plots

The centre frequency of the n^{th} filter can directly be calculated by,

$$f_c(n) = base * 2^{\frac{n}{12}} \tag{2.18}$$

There are two possible bandwidth formulations

1. **Linear with respect to the centre frequency :** In this formulation, the bandwidth of the n^{th} filter is given by

$$B_{bachL}(n) = \frac{base * \left(2^{\frac{n+1}{2}} - 2^{\frac{n-1}{2}}\right)}{2} \tag{2.19}$$

   Figure 2.9 shows the bandwidth plot for the *Bach* scale in the linear formulation. This formulation is the most suitable for music signals because the bandwidth is equal to 50 cents in perceptual frequency or half the interval between semi-tones.

2. **Non-linear with respect to the centre frequency :** The formulation gives an exponential relation between the bandwidth and the centre frequency. In this formulation, the bandwidth of the n^{th} filter is given by

$$B_{bachN}(n) = base * \left(2^{(2^{\frac{(n-1)*log_2\left(\frac{M-12}{12}\right)}{M}} - 1)}\right) \tag{2.20}$$

   Figure 2.9 shows the bandwidth plot for the *Bach* scale in the non-linear formulation.

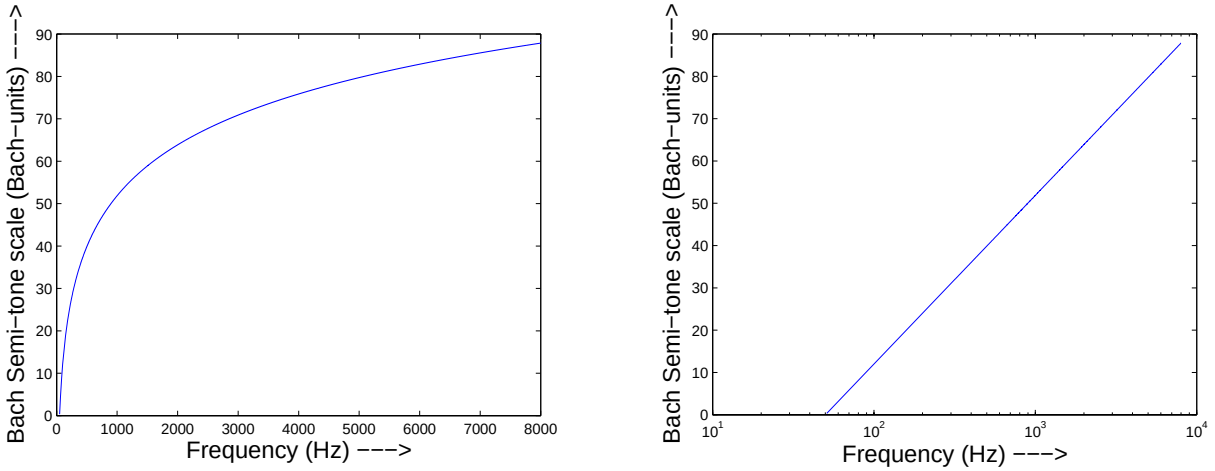

Figure 2.7: The *Bach* scale in (a) linear and (b) semilog plots

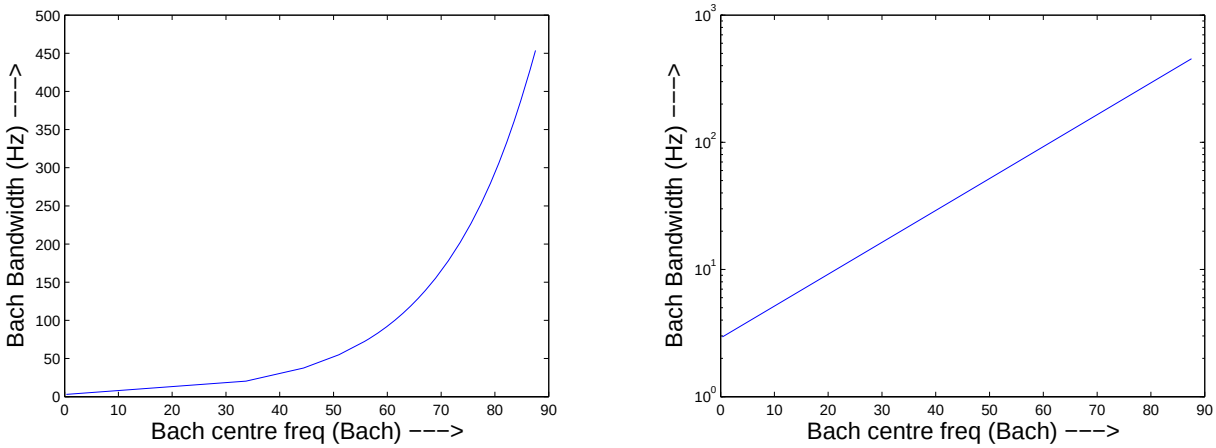

Figure 2.8: The bandwidth of *Bach* scale in the linear formulation as (a) linear and (b) semilog plots

Figure 2.10 compares the values of centre frequencies in Hz with respect to the Bach scale, while figure 2.11 compares the bandwidths in each scale.

2.3 Construction of Filters

The filters designed are lag-windows obtained by the standard Blackman-Tukey spectral estimation method [27]. The design objective is to reduce the leakage incurred by the window $\overline{h_n}(k)$ as much as possible given the bandwidth $f_b(n)$. $f_b(n)$ is the bandwidth of the

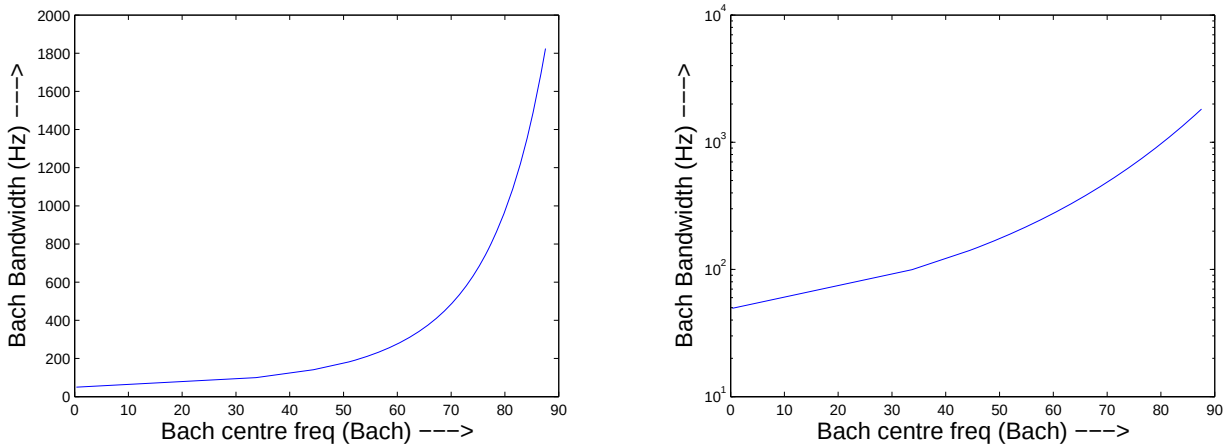

Figure 2.9: The bandwidth of *Bach* scale in the non-linear formulation as (a) linear and (b) semilog plots

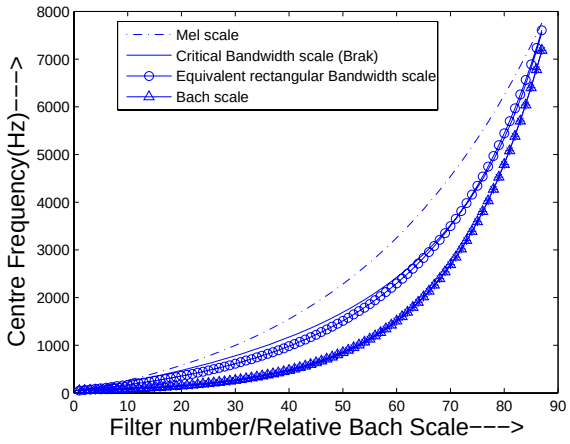

Figure 2.10: The comparison of the centre frequencies of different scales

n^{th} filter of an arbitrary scale. $\overline{h_n}(k)$ is considered a sequence whose DTFT $\overline{H_n}(\omega)$ is the squared magnitude of another sequence $\overline{v_n}(k)$ so that the solution of the constructed window is positive semi-definite.

$$\overline{h_n(k)} = \sum_{j=0}^{N(n)-1} \overline{v_n}(j)\overline{v_n^*}(j-k) \tag{2.21}$$

where $N(n)$ is the number of coefficients allowed for the n^{th} filter. The objective can be reformulated to that of minimizing the relative energy in the side-lobes or maximizing the

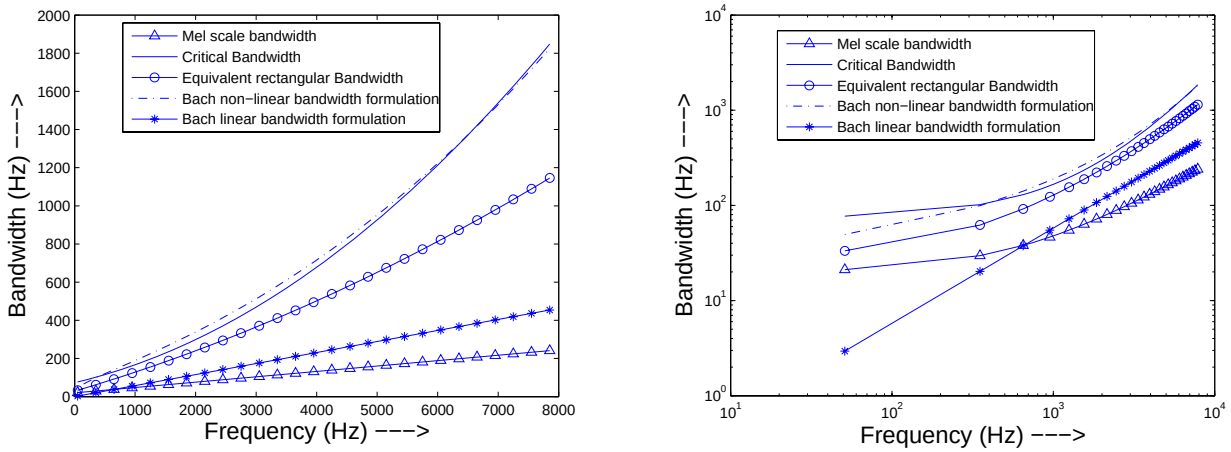

Figure 2.11: The comparison of the bandwidths of different scales as (a) linear and (b) semilog plots

relative energy in the main lobe. Let us assume the parameter β, such that,

$$\beta = \frac{f_b(n)}{2} \tag{2.22}$$

Thus, we have to find

$$\max_{\omega} \left\{ \frac{\int_{-\beta\pi}^{\beta\pi} \overline{H_n}(\omega) d\omega}{\int_{-\pi}^{\pi} \overline{H_n}(\omega) d\omega} \right\} \tag{2.23}$$

The solution boils down to finding the eigen-vector associated with the maximum eigen value of the matrix with elements

$$\gamma_{m,n} = \beta * sinc[(m-n) * \beta * \pi] \tag{2.24}$$

where

$$sinc(x) = \begin{cases} = 0 & \text{if } x = 1 \\ = \frac{\sin x}{x} & \text{otherwise} \end{cases} \tag{2.25}$$

The number of filter coefficients, $N(n)$, used to generate the n^{th} filter is determined by

$$N(n) = 2 * round\left(\frac{1}{f_b(n)}\right) \tag{2.26}$$

From figures 2.12 and 2.11, we can see that the time resolution is poor for lower frequencies but better for higher frequencies. So we get the paradoxical ability to get better time

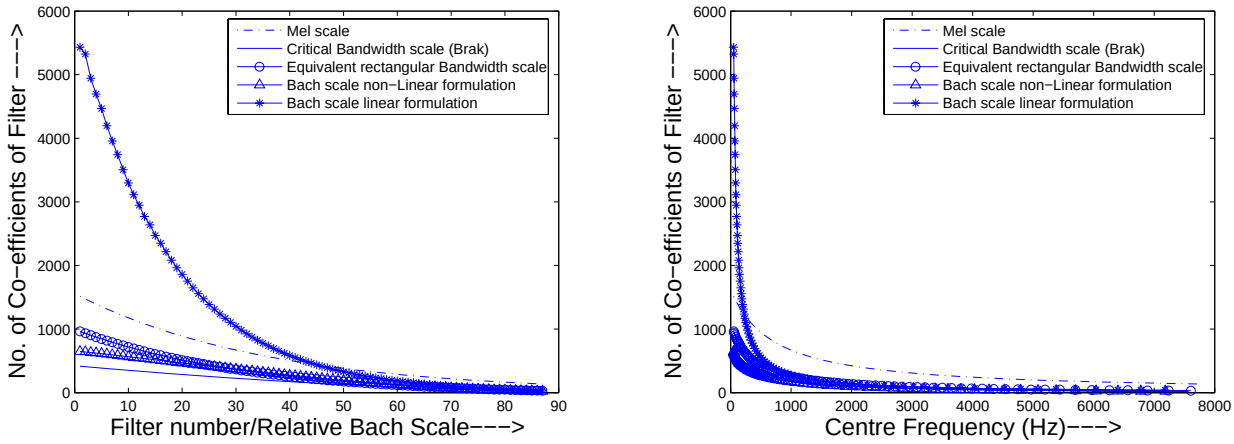

Figure 2.12: The comparison of the no. of coefficients of filters for different scales as (a) as a function of relative Bach value and (b) as a function of frequency in Hz

resolution for higher frequencies and better frequency resolution for lower frequencies. From equation 2.26, we also infer that the filter coefficients are real, symmetric and finite, so the phase responses are linear.

The final set of filters $h_n(k)$ are obtained by modulating the low-pass filters $\overline{h_n}(k)$ with the corresponding centre frequency value $f_c(n)$

$$\widetilde{h_n}(k) = \overline{h_n}(k) * e^{\frac{-2\pi j * k * f_c(n)}{F_s}} \tag{2.27}$$

where j is the square root of -1.

The magnitude responses of the set of filters constructed by the *Bach* linear and non-linear scales are shown in figures 2.13 and 2.14 respectively. We can observe that the filters in the linear formulation are narrow band as compared to the ones in the non-linear formulation. There is a considerable overlap of the filter pass-band frequencies in the non-linear formulation. The filters are almost triangular with minimal overlap in the linear formulation. Thus, the linear formulation is more suitable for music signals. However, because of the better time resolution of the filters in the non-linear formulation, they may be useful for speech signals.

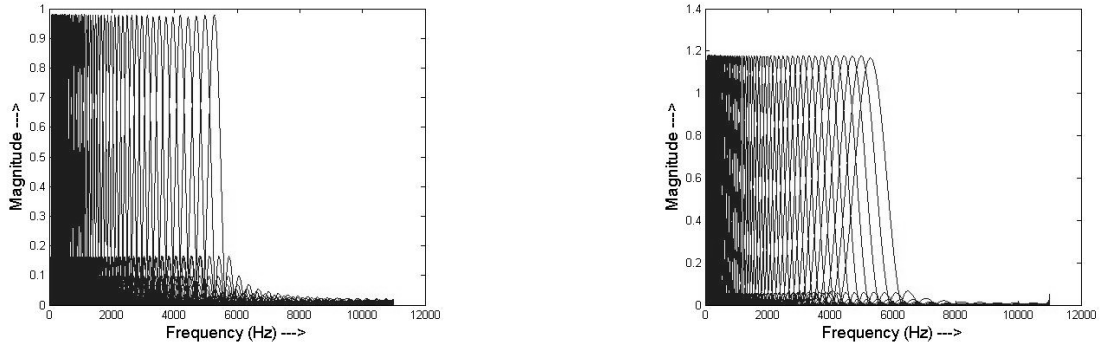

Figure 2.13: The magnitude responses of the filters of the *Bach* scale (a) linear formulation (b) non-linear formulation with $base = 55Hz$ and $F_s = 11000$ Hz

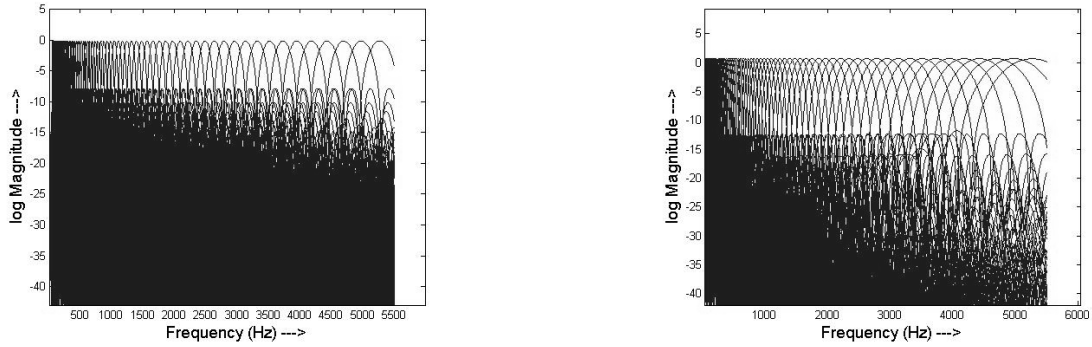

Figure 2.14: The log magnitude responses of the filters of the *Bach* scale (a) linear formulation (b) non-linear formulation with $base = 55Hz$ and $F_s = 11000$ Hz

2.3.1 Equal Loudness Pre-emphasis

The function $E(f)$, is an approximation of the non-equal sensitivity of the human ear given by [16]. It simulates the sensitivity of hearing at approximately the 40db level.

$$E(f) = \frac{((2\pi * f/F_s)^2 + 56.8 * 10^6) * (2\pi * f/F_s)^4}{((2\pi * f/F_s)^2 + 6.3 * 10^6)^2 * ((2\pi * f/F_s)^2 + 0.38 * 10^9) * ((2\pi * f/F_s)^6 + 9.58 * 10^{26})} \tag{2.28}$$

Each of the windows is pre-emphasized by the equation 2.28

$$\widehat{h_n}(k) = \widetilde{h_n}(k) * E(f_c(n)) \tag{2.29}$$

Finally, the last operation is the cubic-root amplitude compression

$$h_n(k) = (\widehat{h_n}(k))^{0.33} \tag{2.30}$$

This operation is an approximation to the power law of hearing [13] and simulates the nonlinear relation between the intensity of sound and its perceived loudness. Figures 2.15 and 2.16 shows the pre-emphasized windows. Though the emphasis seems heavily biased towards the higher frequencies from the magnitude response in Figure 2.15, the log magnitude responses in Figure 2.16, show the real picture, because it imitates the response of the human ear more closely. So we can see that only a slight bias is given to the higher frequencies. Now the filter-bank $h_n(k) \forall n \;\; 1 \leq n \leq M$ is ready to be used in various applications such as speech and music signal processing.

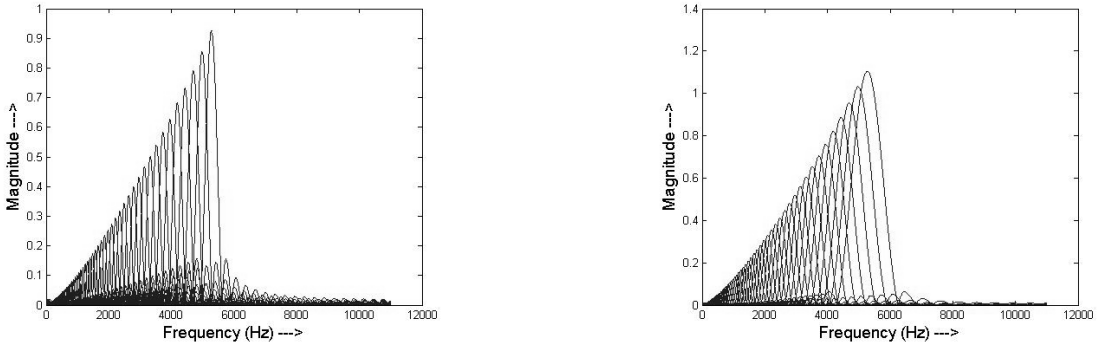

Figure 2.15: Pre-emphasized magnitude responses of the filters of the *Bach* scale (a) linear formulation (b) non-linear formulation with $base = 55Hz$ and $F_s = 11000$ Hz

2.4 Bach-o-gram

The feature space, $F_n(k) \forall n \;\; 1 \leq n \leq M$, obtained from equation 2.1 can be plotted as the TF representation of the signal $s(k)$. Similar to the spectrogram, a Bach-o-gram can be plotted as shown in figure 2.17. We can see the significant formants of the signal are spread over the wider area of the spectrum, rather than concentrated in the lower areas as in a spectrogram. The formants are related to each other logarithmically.

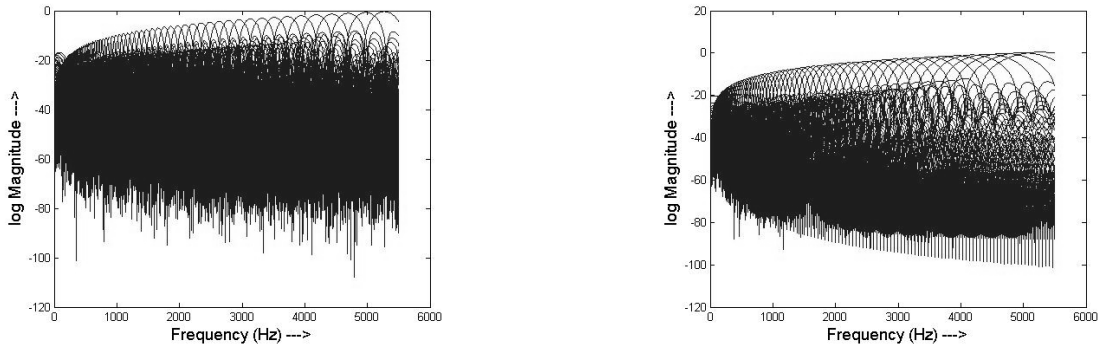

Figure 2.16: Pre-emphasized log magnitude responses of the filters of the *Bach* scale (a) linear formulation (b) non-linear formulation

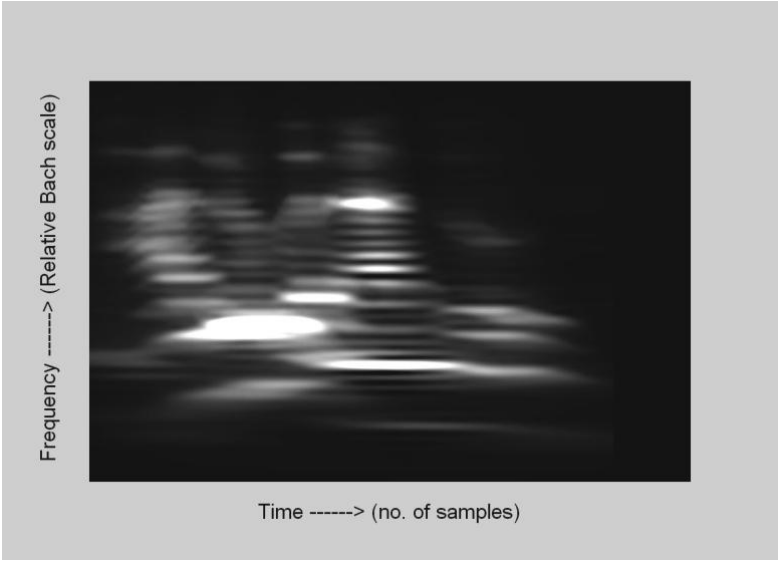

Figure 2.17: The Bach-o-gram of a sample hindi sentence '/b^{h}a:lu: bana: du:t̲/'

Chapter 3

Language Independent Automated Speech Segmentation

3.1 Introduction and Motivation

Current speech technology development strongly relies on corpus-based methodologies and therefore on the availability of good speech corpora. For a corpus to be really useful for the development of speech recognizers or speech synthesizers, apart from the speech itself, it should contain information about its contents (labeling) and about the time alignment between the labels and speech (segmentation).

Phones are usually considered the smallest units of speech, by the concatenation of which any other speech unit (syllable, word, phrase, etc.) can be built. Not surprisingly, phonetic segmentation and labeling are very desirable and useful in a speech corpus. The most precise way to obtain this information is manually. However, manual phonetic labeling and (particularly) segmentation are very costly and require much time and effort.

In speech recognition systems, post processing based on language models, tends to smooth errors caused by improper segmentation. However, this liberty cannot be taken for concatenative synthesis based Text-to-Speech (TTS) systems. In the development of both concatenative acoustic unit inventories, and prosodic models, it is usual to select single examples instead of relying on an average language model. Thus, a segmentation error will produce an audible error in the synthetic voice [28]. This need for extremely precise segmentation has led speech synthesis to rely on manual segmentation for years. During the last few years, however, the need to develop new voices and languages quickly and with high quality (which

frequently implies large inventories) has evoked the interest in automatic segmentation techniques, which do not require large amounts of training data.

Finally, it is worth mentioning that some researchers believe that speech recognition can benefit from more precise segmentation in training or decoding [29].

The earliest attempts at automated segmentation were using the spectrogram of the signal and counting the number of zero-crossings in a region of speech [30],[31]. Van Hemert used the intra frame correlation measure [32] between spectral features to obtain the segments. Statistical modelling (AR, ARMA) [33] has also been used. HMM based automated phonetic segmentation [34] requires a great amount of training, but provides excellent results. The most popularly used feature vector based methods are the Spectral Transition Measure (STM) and the Maximum Likelihood (ML) segmentation methods [35]. Another method called A-LCR, which uses average level-crossings has been suggested by Sarkar [36]. Natarajan and Murthy [37] have used group delay functions to segment speech into syllable like units. All the cited methods do not need training except the HMM based algorithms.

Segmentation methods can be divided into two main categories, namely implicit segmentation and explicit segmentation [32]. Implicit segmentation splits up the utterance into phonetic segments without explicit information such as the phonetic transcription. It uses an implicit definition for a segment, like spectral stability or number of zero-crossings. Explicit segmentation methods split up the utterance into segments that are explicitly defined by phonetic transcription. Explicit methods are not language independent, since the phoneme inventory and phonetic reference patterns are required. In general, the start-up time for a new TTS system can be minimized by initially using an implicit method of segmentation and then going onto an explicit method, with manual intervention.

The motivation to develop a *language independent, automated, implicit* segmentation technique is to minimize the time required in porting a TTS system available in a particular language to another language. This is a necessity in an environment like in India, where multiple languages are in vogue, while standard data is unavailable for most of the languages.

Chapter 2 developed a time-varying TF representation of a signal $s(k)$. Several filter-

banks were discussed, based on various perceptual scales. This chapter endevours to come up with scheme to perform segmentation of the speech signal using the proposed TF representations, and to make a comparative performance analysis of the various perceptual scales.

3.2 Two Class Problem

Speech is considered as a sequence of quasi-stationary units : phones. Segmentation should ideally segregate the signal into such quasi-stationary units. However, due to co-articulation effects, the boundaries are not clearly defined.

Consider the k^{th} speech sample from $s(k)$. The feature vectors obtained from equation 2.1 are $F_n(k)$. The filter-bank could be based on any of the scales that are discussed in Chapter 2. Consider a length of 'W' seconds, which is equal to w number of samples. Where

$$w = \frac{W}{F_s} \tag{3.1}$$

The samples from $[s(k-w)$ to $s(k)]$ can be considered one class with corresponding feature vectors $F_n(k-w)$ to $F_n(k)$. Say these belong to class 'X'. The samples $s(k)$ to $s(k+w)$ with feature vectors $F_n(k)$ to $F_n(k+w)$ belong to class 'Y'. The choice of 'W' is made empirically as demonstrated in Figure 3.2

Several distance functions (DF) can be defined which denote the dissimilarity between the two classes. For every sample $k \in [1, T]$ of $s(k)$, a corresponding $DF(k)$ can be found using the distance measures described in section 3.3, where T is the total length of the signal $s(k)$.

By the definition of a phone, the dissimilarity or distance between the classes X and Y on either side of the k^{th} sample, should be maximum when k is on a phone boundary. The segment boundary is attributed to the point of maximum difference between the two regions of a sample of speech, which corresponds to a peak in the distance function, as shown in Figure 3.1. The intensity of the peak is not relevant for segmentation; the mere presence denotes a phone boundary. The peak detection, is not merely a local maxima detection problem. This is because of the presence of noise in the $DF(k)$, leading to several false

positives. So it may be necessary to determine the maximum value in a region R, rather than just the local maxima. It is achieved by the following method

$$\begin{aligned} &\text{If} \quad DF(k) = \max_{i = k-r : k+r} (DF(i)) \\ &\quad \text{then} \quad Peak(k) = 1 \\ &\text{else} \\ &\quad Peak(k) = 0 \\ &\text{end if} \end{aligned} \tag{3.2}$$

where R is chosen empirically as demonstrated in Figure 3.3 and

$$r = \frac{R}{F_s} \tag{3.3}$$

3.3 Distance Functions

Let us consider a two class problem where 'X' and 'Y',

$$\begin{aligned} X &= \{\overline{x_1}, \overline{x_2}, \overline{x_3}, \ldots, \overline{x_w}\} \\ Y &= \{\overline{y_1}, \overline{y_2}, \overline{y_3}, \ldots, \overline{y_w}\} \end{aligned} \tag{3.4}$$

are the two classes with means $\overline{\mu_X}$ and $\overline{\mu_Y}$.
$\overline{x_i}$ and $\overline{y_i}$ are M dimensional vectors, samples of class X and Y, respectively.

$$\overline{\mu_X} = \frac{\sum_{i=1}^{w} \overline{x_i}}{w} \quad \overline{\mu_Y} = \frac{\sum_{i=1}^{w} \overline{y_i}}{w} \tag{3.5}$$

The following distance measures can be formulated :-

1. **Euclidean Distance of Mean Features (EDM)** :

$$EDM(X,Y) = \|\overline{\mu_X} - \overline{\mu_Y}\| \tag{3.6}$$

2. **Euclidean Distance of Mean Log Features (EDML)** :

$$\overline{\ell\mu_X} = \frac{\sum_{i=1}^{w} \log_{10} \overline{x_i}}{w} \quad \overline{\ell\mu_Y} = \frac{\sum_{i=1}^{w} \log_{10} \overline{y_i}}{w} \tag{3.7}$$

$$EDML(X,Y) = \|\overline{\ell\mu_X} - \overline{\ell\mu_Y}\| \tag{3.8}$$

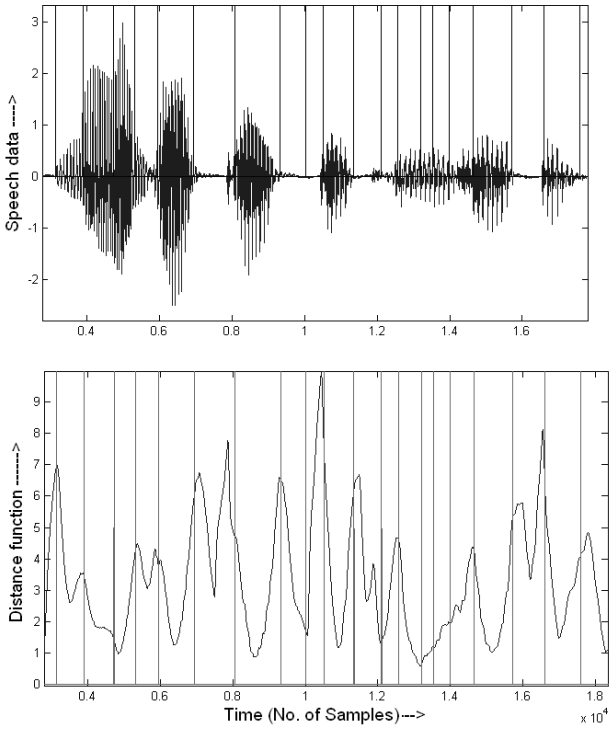

Figure 3.1: (a) A sample speech waveform (b) A Distance function (EDM) plotted over $k \in [1, T]$. The vertical lines correspond to the manual segments.

3. **Normalized Euclidean Distance of Mean Log Features (NEDML)** :

$$\upsilon_X = \|\overline{\ell\mu_X}\| \quad \upsilon_Y = \|\overline{\ell\mu_Y}\| \tag{3.9}$$

$$NEDML(X,Y) = \frac{(\overline{\ell\mu_X} - \overline{\ell\mu_Y})^T(\overline{\ell\mu_X} - \overline{\ell\mu_Y})}{\upsilon_X \upsilon_Y} \tag{3.10}$$

4. **Kullback-Leibler Distance (KLD)** [38] : It is an asymmetric distance function. In case of discrete random variables, p_X and p_Y are the 'probability mass functions' of classes X and Y, respectively.

$$KLD(X,Y) = \left\| \sum_{j=-\infty}^{+\infty} \left(p_X(j) * log_e \left(\frac{p_X(j)}{p_Y(j)} \right) \right) \right\| \tag{3.11}$$

5. **Itakura-Saito Distance (ISD)** [4] : It is a symmetric distance function defined for discrete random variables, where p_X and p_Y are the 'probability mass functions' of classes X and Y, respectively.

$$ISD(X,Y) = \left\| \sum_{j=-\infty}^{+\infty} \left(\frac{p_X(j)}{p_Y(j)} - \log_e \left(\frac{p_X(j)}{p_Y(j)} \right) - 1 \right) \right\| \tag{3.12}$$

6. **Mahalanobis Distance (MD) [39]** : The MD between μ_X and μ_Y is defined for covariance matrix C_{XY}, assuming that X and Y have the same distribution.

$$MD(X,Y) = \sqrt{(\overline{\mu_X} - \overline{\mu_Y})^T C_{XY}^{-1} (\overline{\mu_X} - \overline{\mu_Y})} \tag{3.13}$$

3.4 Comparative Analysis

3.4.1 Definition of Accuracy

The quality of automated segmentation is evaluated by comparing the output with manually segmented databases. If an automated segment boundary falls within ±20 ms of a manually segmented boundary, then it is considered to be a 'Matched Phoneme Boundary' (MPB). If more than one automated segment boundary falls within ±20 ms of a manual boundary or no manual boundary is found within ±40 ms of an automated boundary, then such boundaries are considered to be 'Insertions' (Ins). On the other hand, if no unique automated boundary is found within ±40 ms of a manual boundary, then it is considered a 'Deletion' (Del).

3.4.2 Description of Data

The results are obtained for 100 sentences of English data from the (F_s = 16000 Hz) TIMIT database for both male and female speakers. The data has a Signal to Noise Ratio (SNR) of 36 dB. 100 sentences of Hindi and Tamil data have also been segmented. The Hindi and Tamil data have a sampling frequency (F_s) of 44.1 KHz and an SNR of 30 dB. The data available for Hindi and Tamil are only that of a male voice.

3.4.3 Comparison with Other Methods

The following methods have been used as a comparative basis to study the proposed method.

1. ML Segmentation [35] using MFCC with a symmetric lifter $(1 + A * sin(1/2(n/L))$, where (A = 4, L is the MFCC dimension = 16).

2. Spectral Transition measure (STM) [35] using feature vector and lifter combination.

3. Average level crossing rate method (A-LCR), as described in [36], using non-uniform level allocation.

Table 3.1 compares the performances of the proposed method and the other standard methods. The proposed method does marginally better in terms of 'MPB' percentage. However, the standard methods use information such as the number of phonemes and location of silences as information in order to obtain the correct phoneme boundaries. The proposed method using *EDML* and a *Bach Linear* scale filter-bank gets comparatively better results without using such information.

Table 3.1: Comparison between various segmentation methods on the TIMIT database

Segmentation Method Used	%MPB	%Del	%Ins
ML [35]	80.8	19.2	18.8
STM [35]	70.1	29.9	25.2
A-LCR [36]	79.8	20.2	24.2
EDML with *Bach(Lin)* scale	**82.5**	22.3	18.9

3.4.4 Comparison between Various Audio Scales Using the *EDML* Distance Measure

The results presented in Table 3.2 are for filter-banks with 79 filters and the *base* frequency is taken to be 55 Hz. The filters are designed for F_s = 11 KHz. The speech samples are re-sampled to 11 KHz before applying the algorithm. The value of 'W' and 'R' are taken as 40 ms and 15 ms, respectively, as seen in Figures 3.2 and 3.3.

From Table 3.2, we can see that the *Bach linear* and the *Bach non-linear* scales perform comparably, if not marginally better than the *Mel* or *Bark* scales. We can, however, see a significant difference in the number of false inserted phone boundaries between the *Mel* and *Bark* scales as against the *Bach* scales. However, it can be noted that the number of deletions of the boundaries are higher in case of the *Bach linear* case.

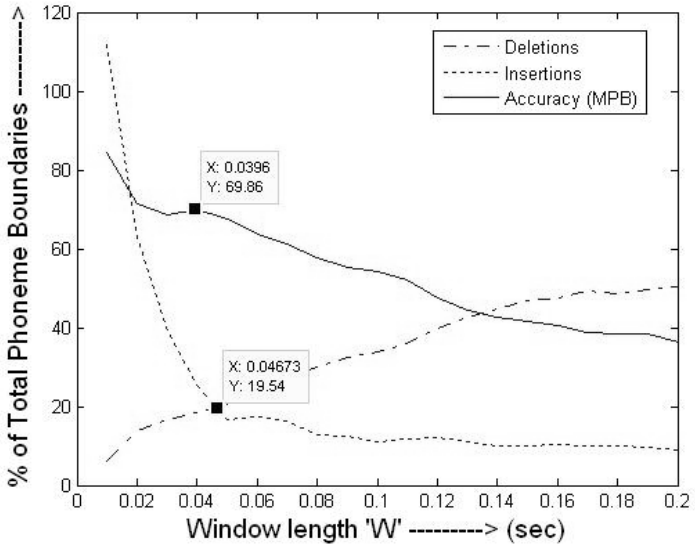

Figure 3.2: The analysis of the effect of '*W*' on the %Deletions, %Insertions and accuracy of the segmentation algorithm for 20 Hindi sentences.

Table 3.2: Comparison of segmentation performances of the various filter-bank scales for TIMIT database

Audio frequency scale	%MPB	%Del	%Ins
Mel (Eqn. 2.7 and 2.8)	78.1	16.5	68 .1
Bark (Eqn. 2.12 and 2.13)	78.1	17.9	50.1
ERB (Eqn. 2.14 and 2.15)	76.3	18.9	52.4
Bach (Lin) (Eqn. 2.18 and 2.19)	**82.5**	22.3	**18.9**
Bach (Non-Lin) (Eqn. 2.18 and 2.20)	79.3	**17.4**	20.4

3.4.5 Comparison of Performances of Different Distance Functions

The results presented in Table 3.3 compare the performances of the various distance measures described in section 3.3 for the Hindi database. They are all calculated using the *Bach linear* scale filter-bank formulation. It can be observed that *EDML* and *NEDML* perform better than the other distance functions both in terms of accuracy as well as % Deletions and

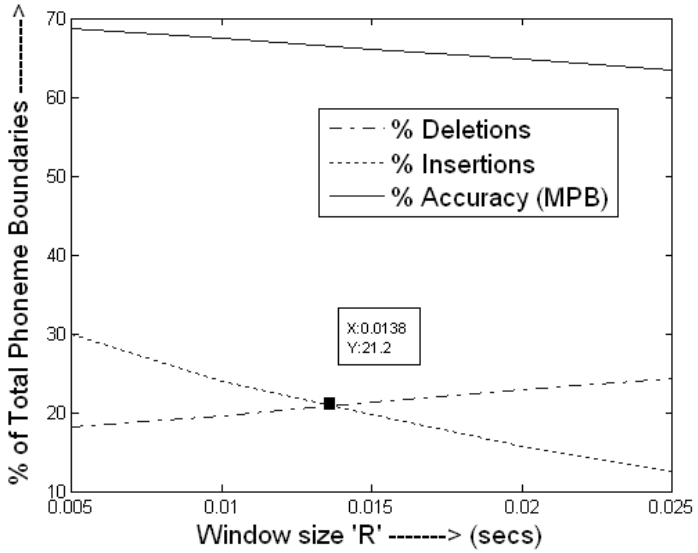

Figure 3.3: The analysis of the effect of 'R' on the %Deletions, %Insertions and accuracy of the segmentation algorithm for 20 Hindi sentences.

Insertions. Figure 3.4 shows a comparison between the segment boundaries detected by the *EDML*, *EDM* and the *NEDML* functions.

3.4.6 Comparison for Different Languages

Table 3.4 compares the segmentation performance of the proposed method using the *Bach Linear* scale formulation on different languages using the *EDML* distance function. It can be seen that the performance for the three languages namely English, Hindi and Tamil are comparable.

3.5 Comparative Studies for Phoneme Classes

Phonemes from most Indian languages can be divided into 10 classes namely 'vowels' (VV), 'nasalized vowels' (NV), 'diphthongs' (DD), 'stop consonants' (SC), 'aspirated stop consonants' (AS), 'fricatives' (FF), 'nasals' (NN), 'glides' (GG), '/r/ and /R/' phonemes (RR)

Table 3.3: Comparison of performances between the various distance functions on the 'Hindi' database

Distance function	%MPB	%Del	%Ins
EDML	**86.6**	3.2	**21.4**
EDM	80.6	4.5	25.6
NEDML	84.7	**2.5**	39.8
KLD	41.6	21.4	46.3
ISD	69.9	15.5	34.0
MD	35.2	35.2	37.3

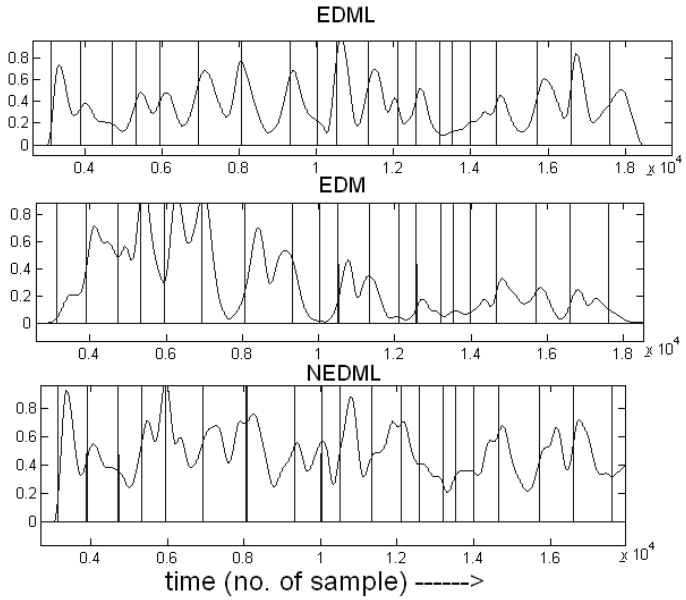

Figure 3.4: The plots of *EDML*, *EDM* and *NEDML* against time. The vertical lines indicate the actual phone boundaries

and silence (SS). A study has been made to compare the performance of each of the distance functions mentioned in section 3.3 for individual phoneme classes.

Table 3.5 shows the comparative %Insertions for individual phoneme classes by the var-

Table 3.4: Comparison of the performance of *EDML* using *Bach Linear* filter-bank for various languages

Language	%MPB	%Del	%Ins
English (TIMIT)	82.5	22.3	18.9
Hindi	86.6	3.2	21.4
Tamil	81.9	15.3	23.7

ious distance functions. The distance function with the best performance for each phoneme class is highlighted. This study is done using the *Bach Linear* filter-bank for 100 Hindi sentences. Similarly, Table 3.6 shows the comparative %Deletions for individual phoneme classes by the various distance functions.

Table 3.5: Comparison of %Insertions for various distance functions grouped by phoneme class

Phoneme Class	EDML	EDM	NEDML	KLD	ISD	MD
VV	**16.9**	29.7	30.3	54.6	41.8	57.6
NV	54.5	54.4	**50**	54.5	63.6	54.5
DD	**52.3**	133.3	100	64.5	80.6	92.5
SC	**17.4**	12.7	44.7	45.3	36.0	23.8
AS	42.1	36.8	55.2	57.8	**28.9**	34.2
FF	**22.8**	41.4	38.5	82.8	52.8	55.7
NN	11.3	**6.3**	12.6	35.4	22.7	11.3
GG	**5.7**	13.0	15.9	15.9	11.5	21.7
RR	5.0	7.5	11.3	7.5	**3.7**	5.6
SS	3.8	**0**	21.1	34.6	19.2	17.3

As can be seen from Tables 3.5 and 3.6, various distance functions perform to a varying degree of accuracy for different phoneme classes.

Table 3.6: Comparison of %Deletions for various distance functions grouped by phoneme class

Phoneme Class	EDML	EDM	NEDML	KLD	ISD	MD
VV	7.65	**5.4**	9.0	12.5	13.1	30.8
NV	27.2	**9.1**	18.1	18.1	22.7	18.1
DD	**0**	4.7	**0**	5.1	7.7	6.1
SC	9.3	16.8	**8.1**	23.8	11.0	46.5
AS	18.4	15.7	**5.2**	28.9	7.9	44.7
FF	**10.0**	**10.0**	**10.0**	**10.0**	17.1	41.4
NN	3.8	10.1	**2.5**	27.8	17.7	21.5
GG	**10.1**	13.0	**10.1**	26.0	31.8	31.8
RR	7.5	**3.7**	5.0	40.5	21.5	41.7
SS	**14.2**	47.6	**14.2**	**14.2**	11.9	33.3

3.6 Comparative Studies for Filters from Individual Octaves

It is also interesting to study the effect of finding the *EDML* function only for filters from single octaves. It is obvious that when considering some octaves, the performance would be better than for other octaves.

In fact, it can be observed from Table 3.7, that when considering the fourth octave alone, the %MPB is better than when considering all the octaves together. However, the %Insertions is higher than, when we combine all the octaves. Thus, combining all the octaves gives the best compromise over accuracy and insertions. However, study of how different octaves perform for different phoneme classes is also interesting. Tables 3.8 and 3.9 show the comparative %Inserions and %Deletions, respectively, for individual phoneme classes by the various octaves.

We can see that the mid and lower octaves perform the best for vowels, stop consonants and nasals, while the higher mid and higher octaves perform better for fricatives, glides and silence regions. Thus it may be possible to exploit this information, when the phonemic

Table 3.7: Comparison of performances between filters for individual octaves on the 'Hindi' database using *EDML* distance measure

Octave	%MPB	%Del	%Ins
Octave 1 (50 - 100 Hz)	72.7	4.6	54.7
Octave 2 (100 - 200 Hz)	71.3	4.4	37.1
Octave 3 (200 - 400 Hz)	81.3	3.0	29.9
Octave 4 (400 - 800 Hz)	**86.8**	2.9	29.3
Octave 5 (800 - 1600 Hz)	84.8	2.7	35.3
Octave 6 (1600 - 3200 Hz)	85.4	2.3	39.0
Octave 7 (3200 - 6400 Hz)	85.6	**1.8**	50.1
All (50 - 6400 Hz)	86.6	3.2	**21.4**

Table 3.8: Comparison of %Insertions for filters for individual octaves grouped by phoneme class

Phoneme Class	Oct 1	Oct 2	Oct 3	Oct 4	Oct 5	Oct 6	Oct 7	All
VV	62.6	43.7	34.9	26.5	32.3	41.9	65.4	**19.9**
NV	108.0	79.2	59.1	49.4	52.5	71.7	100.0	**43.9**
DD	138.0	109.5	85.7	66.6	95.2	90.4	123.8	**52.3**
SC	53.9	29.3	**21.7**	33.5	39.9	37.3	39.4	30.2
AS	84.7	71.0	68.1	76.7	77.1	79.0	76.0	**57.4**
FF	75.4	51.3	44.4	52.3	70.3	71.2	69.0	**24.7**
NN	37.6	34.7	23.1	21.4	26.3	26.2	28.1	**11.3**
GG	28.3	21.2	17.2	16.0	20.4	21.0	30.4	**11.2**
RR	12.3	9.9	12.0	8.9	8.4	9.4	13.4	**5.5**
SS	20.1	3.1	2.9	2.7	4.8	4.4	2.9	**2.1**

sequence is available. In order to improve the accuracy it may be possible to give variable weights to the octaves, based on the phonemic group that the phoneme to be segmented falls under.

Table 3.9: Comparison of %Deletions for filters for individual octaves grouped by phoneme class

Phoneme Class	Oct 1	Oct 2	Oct 3	Oct 4	Oct 5	Oct 6	Oct 7	All
VV	2.2	2.2	2.4	2.8	2.2	1.8	**0.8**	3.3
NV	5.5	4.5	4.0	6.5	3.0	2.0	**1.0**	6.0
DD	4.7	4.7	**0**	**0**	**0**	**0**	**0**	**0**
SC	6.4	3.9	2.5	2.2	2.6	2.7	**2.1**	2.3
AS	5.9	6.6	3.6	3.7	**2.3**	2.6	3.0	4.3
FF	6.1	5.0	2.3	2.9	2.1	**1.2**	1.6	3.8
NN	6.3	7.8	3.0	1.9	2.0	1.9	2.7	**1.3**
GG	7.2	9.5	5.5	4.7	5.7	4.9	**3.6**	5.9
RR	4.6	6.5	5.3	1.5	2.2	**1.2**	1.3	1.7
SS	4.8	3.7	**3.5**	5.3	7.8	5.1	5.1	5.3

3.7 Conclusions and Future Work

As we can see from Table 3.1, the proposed method using a time varying model with a *Bach linear* filter-bank formulation, performs relatively better than the other methods suggested for language independent automated segmentation. This result is surprising, especially since many a research has shown that the perception of intonation in speech signals is closer to the ERB scale [40], than the logarithmic '*Bach*' scale, as in the case of music signals. The *EDML* function performs the best among the suggested distance functions. The proposed method requires minimum training and is language independent.

However, in order to be able to use it for segmentation of the database of a new language, the method is not accurate enough. To improve the accuracy of the segmentation, it is necessary to use explicit information such as the phonetic transcription, if available. In the context of TTS systems, the phonetic transcription is available. Training can be conducted on the basis of the knowledge of the spectral and temporal structure of individual phones, or phoneme classes. Tables 3.5 and 3.6 show that it is possible to exploit the various distance functions for different phoneme classes and Tables 3.8 and 3.9 show the same for different octaves.

Future work may also include comparative study of *Bach* scale with respect to other scales in various other applications of speech processing such as, recognition and coding. It may be useful to derive *Bach* cepstral coefficients and *Bach* linear predictive coefficients, in order to have a fair comparison between *Bach* scale and other perceptual scales.

Chapter 4

Singing Voice Transcription

4.1 Introduction and Motivation

'Musical transcription' refers to the process of converting a musical rendering or performance into a set of symbols or notations. The notations may contain information about the pitch, duration, stress and timbre of the musical rendering. But in the context of query-by-humming systems, only pitch and timing information are important, since the quality of rendering should not be taken into account.

Several approaches to pitch tracking are reviewed in [41], [42] and [43]. However, for singing voices, a reliable conversion of pitch estimates to a symbolic notation has proven to be a challenging problem [44]. This is because a typical vocal rendition contains both inaccuracies in pitch and timing. We perceive the pitch of a song as discrete values corresponding to the musical semi-tones. But the pitch estimates of the existing pitch tracking algorithms are usually continuous functions of frequency. Assigning them semi-tonal values, as perceived by human beings, has proven to be non-trivial.

It is possible to convert pitch estimates of only the voiced regions to a symbolic notation. But, for a good quality transcription, it also becomes necessary to assign suitable pitch values to the unvoiced regions, though the pitch for unvoiced regions is usually undefined. Thus, we require a pitch tracking algorithm, which returns discrete frequency values, and is defined for all time instants.

Pitch extraction from singing voice has traditionally been viewed from the perspective of

absolute pitch of the voice. However, for the query- by-humming applications, the absolute pitch is not of as much importance as that of relative pitch, since the queries may not have the same absolute pitch as the original rendering of the song. There are other issues related to hummed queries too. The query may or may not contain words, or perhaps could be a whistle. Often, the change in pitch is not always associated with a new note. This is because of the presence of 'pitch-bends' or 'glides' in singing voice and whistling. Clarisse *et al.* [45] have proposed a method which first determines note segments from a humming input and then assigns a note value for each segmented note region. Viitaniemi *et al.* [46] have introduced a musical-key estimation model and used a probabilistic model to infer note values from raw pitch estimates. However, both these systems fail to consider the dynamic nature of singing voices and don't make allowances for glides or pitch-bends. They also lack robustness for various forms of queries.

This chapter tries to provide two pitch tracking algorithms, which are reasonably robust for the possible ways a query can be made by a user. The algorithms obtain a relative pitch contour as against an absolute pitch contour. It must be noted that the output of the automated transcription is compared with manually transcribed Music Instrument Digital Interface (MIDI) format files, i.e. manual transcription of the actual sample of singing voice (or whistle). So the manually transcribed MIDI files contain many notations of glides or pitch-bends where the pitch changes from one semi-tone to the other, without it being referred to as a new note. This is an approach, different from the usual trend of comparing the automated transcription result with the MIDI file of the original sound track.

Brown [26] suggested a fundamental frequency detection algorithm based on a constant Q-transform. The algorithms discussed are special case of the above method (Q = 17), treating the signal as a time varying signal as against short time stationary. The two algorithms are: the frequency selection based algorithm described in section 4.2 and the time domain selection based algorithm described in Section 4.3.

4.2 Frequency Selection Based Algorithm

The *Bach* linear scale filter-bank, designed in Chapter 2, has a higher frequency resolution for lower frequencies and higher time resolution for higher frequencies. The fundamental

frequency for human voices is usually in the range of 50 to 500 Hz. A band-pass filter designed to track the fundamental frequency needs a high number of filter coefficients and is sluggish in nature (poor time resolution). A filter designed in the range of the 3^{rd} harmonic usually has the optimum compromise between time and frequency resolution. Besides, for complex tones with fundamental frequency in the range 100 to 400 Hz, the dominant region lies around the third, fourth, and fifth harmonics [47]. The 3^{rd} harmonic is usually (though not always) one of the significant harmonic peaks, which makes it another good reason to track the same. The input to the algorithm is the 2-D vector $F_k(n) \forall n \;\; 1 \leq n \leq M, \; \forall k \;\; 1 \leq k \leq T$, obtained from equation 2.1 using the *Bach* linear formulation. We propose that the *Bach* linear scale filter-bank models the human perception of music. The block-diagram of the algorithm is shown in Figure 4.1. The descriptions of the individual blocks are presented in the following sub-sections.

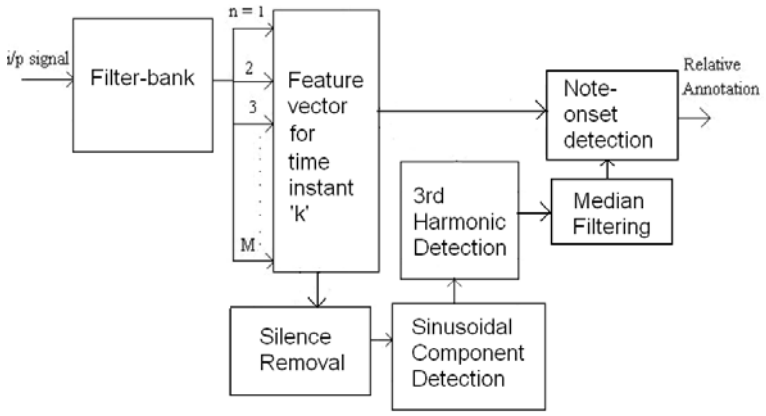

Figure 4.1: Block diagram of the frequency selection based algorithm

4.2.1 Silence Removal

It is assumed that the background noise is non-sinusoidal in nature, which implies that the power spectrum will not be spiky and will have a low instantaneous spectral variance. This aspect is used to roughly determine the difference between silence and non-silence regions. So, if the instantaneous spectral variance, $(\sigma^2(k))$, is below a certain threshold, then it is considered as silence. This threshold is related to the Signal to Noise Ratio (SNR) of the

signal. The instantaneous spectral variance is calculated as

$$\sigma^2(k) = \frac{1}{M-1}\sum_{n=1}^{M}(F_n(k) - \breve{F}(k))^2 \tag{4.1}$$

where

$$\breve{F}(k) = \sum_{n=1}^{M}\frac{F_n}{M} \tag{4.2}$$

The algorithm for detecting silence is as follows.

Algorithm 4.1. *: Silence removal*

$\sigma^2_{max} = 20 * log_{10}\left(\max_{k=1\ to\ T}\sigma^2(k)\right)$
for $k = 1$ to T
  if $\sigma^2_{max} - 20 * log_{10}(\sigma^2(k)) > SNR$
    $s(k)$ is a region of silence
  else
    $s(k)$ is a non-silence region
  end if
end for

The voiced region of the signal may be interspersed with several unvoiced regions, belonging to stop consonants and fricatives. But, due to the sluggish nature of the lower frequency filters, the instantaneous spectral variance does not drop below the threshold. So this method is fairly successful as demonstrated in Tables 4.2, 4.3 and 4.4.

4.2.2 Estimation of Significant Sinusoidal Components (Tones)

The frequency energy vector, $F_k(n)$, for every time instant k contains the relative frequency energies around the particular time instant. However, in the presence of a dominant frequency, the other frequencies at that time instant are masked. This masking is associated with critical band-width. So, the *Bach* frequencies, 'f_{Bach}', need to be converted to the equivalent critical band-width frequencies 'f_z' (*Bark* scale).

$$z(f_{Bach}) = \frac{26.81}{1 + \frac{1960}{base * 2^{\frac{f_{Bach}}{12}}}} - 0.33 \tag{4.3}$$

The above equation is obtained by combining equations 2.9 and 2.17. The relative *Bach* frequency 'n' is a sinusoid or tone if,

$$\begin{aligned} F_k(n) &= \max_{i=n-K \text{ to } n+K} F_k(i) \qquad \text{and} \\ F_k(n) &\geq 7db + \min_{i=n-K \text{ to } n+K} F_k(i) \end{aligned} \tag{4.4}$$

where

$$K = \begin{cases} 2, & \text{for} \quad 1 \leq n \leq \log_2 \frac{5.5*10^3}{base} \\ 3, & \text{for} \quad \log_2 \frac{5.5*10^3}{base} \leq n \leq \log_2 \frac{11*10^3}{base} \\ 6, & \text{for} \quad \log_2 \frac{11*10^3}{base} \leq n \leq M \end{cases} \tag{4.5}$$

M is the total number of filters as defined in Equation 2.2. The Spreading Function (SF) [48] is defined as

$$SF(i,j) = \begin{cases} 17 * dz - 0.4 * F_k(j) + 11 & \text{for} \quad -3 \leq dz < -1 \\ (0.4 * F_k(j) + 6) * dz & \text{for} \quad -1 \leq dz < 0 \\ -17 * dz & \text{for} \quad 0 \leq dz < 1 \\ (0.15 * F_k(j) - 17) * dz - 0.15 * F_k(j) & \text{for} \quad 1 \leq dz \leq 8 \end{cases} \tag{4.6}$$

SF depends on the location of the masked frequency, 'i' (*Bach*), the masker location, 'j' (*Bach*), the power spectrum, '$F_k(j)$', and the difference between 'i' and 'j' in *Bark* (Equivalent Critical Bandwidth). The quantity 'dz' is defined as

$$dz = z(i) - z(j) \tag{4.7}$$

where the function 'z' is defined in Equation 4.3. The threshold (T_{MN}) [48] for tone masking noise is

$$T_{MN}(i,j) = F_k(j) - 0.175 * z(j) + SF(i,j) - 2.025(dB\,SPL) \tag{4.8}$$

and for tone masking tone, the threshold (T_{MT}) is as follows [48].

$$T_{MT}(i,j) = F_k(j) - 0.275 * z(j) + SF(i,j) - 6.025(dB\,SPL) \tag{4.9}$$

Algorithm 4.2, given below, finds the indexes of the significant sinusoids (tones), $\overline{S_k}$, and their corresponding amplitudes, $\overline{FS_k}$, at time instant 'k' given the feature vector $F_k(n) \forall n \;\; 1 \leq n \leq M$. The initial masker, '$I_k$', is taken as the frequency with the highest amplitude, i.e. $F_k(I_k)$.

Algorithm 4.2. *: Significant sinusoid detection*

$F_k(I_k) = \max_{n=1 \text{ to } M}(F_k(n))$
$I_k = \arg\max_{n=1 \text{ to } M}(F_k(n))$
Initialize $ind = 1$ and $j = I_k$
for $i = j - 1$ to $-1 : 1$ and $j + 1$ to M
  $T_{MN} = F_k(j) - 0.175 * z(j) + SF(i, j) - 2.025(dB\,SPL)$
  $T_{MT} = F_k(j) - 0.275 * z(j) + SF(i, j) - 6.025(dB\,SPL)$
  if $F_k(i) > T_{MN}(i, j)$
    $\overline{S_k}(ind) = i$ and $\overline{FS_k}(ind) = F_k(i)$
    ind++
    if $F_k(i) > T_{MN}(i, j)$
      $j = i$
    end if
  end if
end for

4.2.3 3^{rd} Harmonic Detection

The detected significant sinusoids have to be grouped according to their possible harmonics. The method used is similar to the one used by Hermes [40]. We are taking into consideration the first seven harmonics, which are usually the important ones in a human voice or whistle. Due to the non-linear nature of the *Bach* frequency scale, the harmonic values are related to the fundamental frequency, 'f_0', as shown in Table 4.1. The array, 'H', of harmonic differences is defined as

$$H = \{0, 12, 19, 24, 28, 31, 34\} \tag{4.10}$$

For a group of significant sinusoids, all potential harmonically related sub-groups are identified. This procedure is illustrated with the following example.

Example 1: Consider a group of $\overline{S_k} = \{7, 12, 19, 23, 26, 31, 35, 38, 42\}$ with corresponding $\overline{FS_k} = \{0.02, 0.03, 0.05, 0.02, 0.07, 0.06, 0.04, 0.02, 0.03\}$. The group is ordered to obtain the ordering matrix, G_k, as shown below

Table 4.1: The first seven harmonics and their corresponding *Bach* values

1^{st} harmonic	-	f_0
2^{nd} harmonic	1^{st} overtone	$f_0 + 12$
3^{rd} harmonic	2^{nd} overtone	$f_0 + 19$
4^{th} harmonic	3^{rd} overtone	$f_0 + 24$
5^{th} harmonic	4^{th} overtone	$f_0 + 28$
6^{th} harmonic	5^{th} overtone	$f_0 + 31$
7^{th} harmonic	6^{th} overtone	$f_0 + 34$

$$G_k = \begin{pmatrix} 7 & 19 & 26 & 31 & 35 & 38 & 0 \\ 12 & 0 & 31 & 0 & 0 & 0 & 0 \\ 19 & 31 & 38 & 0 & 0 & 0 & 0 \\ 23 & 0 & 42 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 38 & 42 & 0 & 0 \end{pmatrix} \tag{4.11}$$

'L' is the total number of harmonically related sub-groups, which is 5 in the above example. $I(k) = 26$ and $F_I(k) = 0.07$.

There are two ways to find the most likely 3^{rd} harmonic as demonstrated in Algorithms 4.3 and 4.4 below. Let $\widetilde{\lambda_{auto}}(k)$ be the the pitch estimate for the 'k^{th}' time sample. Usually, the two estimates turn out to be the same. In case they are different, we consider the estimate that falls within the same octave of the pitch estimate of the previous instant, i.e. $\widetilde{\lambda_{auto}}(k-1)$.

Algorithm 4.3. *: Detection of most significant peak*

```
for p = 1 to L
    for q = 1 to 7
        if G_k(p,q) = I(k)
            λ̃_auto(k) = G_k(p,q) − H(q) + 19 ··· H is obtained from Equation 4.10
        end if                                  ··· 19 corresponds to the 3rd harmonic
    end for                                     in Bach scale
end for
```

Using Algorithm 4.3, $\widetilde{\lambda_{auto}}(k) = 26$ for Example 1.

Algorithm 4.4. *: Detection of harmonically richest set*

$$HS_k = \arg \max_{p=1 \text{ to } L} \left(\sum_{q=1}^{7} F_k(G_k(p,q)) \right) \tag{4.12}$$

$$\widetilde{\lambda_{auto}}(k) = G_k(HS_k, 3) \tag{4.13}$$

where HS_k is the harmonically richest set. In this case too, $\widetilde{\lambda_{auto}}(k) = 26$ for Example 1.

4.2.4 Median Filtering

The pitch estimate, $\widetilde{\lambda_{auto}}(k)$, is then median filtered in order to avoid some spikes created during the transitions of the pitch.

$\lambda_{auto}(k),\ \forall k\ \ 1 \leq k \leq T$, is the final pitch contour generated by the frequency selection based algorithm. Figure 4.2 shows the pitch contour generated by the algorithm for a sample rendering of a song. We can observe from the figure that the output of the algorithm contains discrete values of frequencies (corresponding to the semitones) and is defined even for the unvoiced parts of the rendering.

4.2.5 Note Onset Detection

Note onset detection is a non-trivial problem and is usually the source of most of the transcription errors. There are three possible types of note onsets.

1. Note onset associated with a change in frequency
2. Note onset associated with a change in energy
3. Note onset associated with a change in syllable being pronounced

A change in pitch may not always be associated with a note onset, because of the presence of glides or pitch-bends in the singing voice. Thus, a three pronged approach to estimating the note onsets is suggested. It is elaborated in Algorithms 4.5, 4.6 and 4.7.

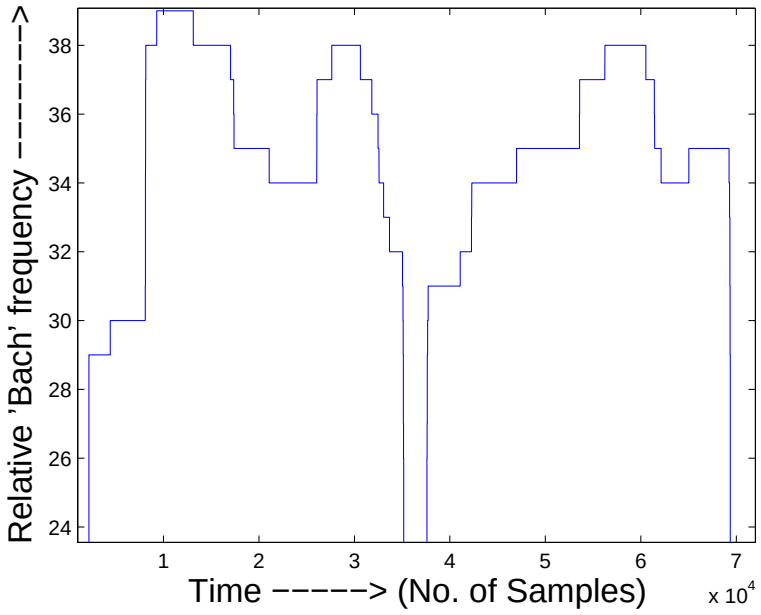

Figure 4.2: A typical contour generated by the algorithms in Section 4.2

The parameter, 'Ψ', is the allowed tolerance for note onset timing. 'ψ' is the actual number of samples obtained as shown below.

$$\psi = \frac{\Psi}{F_s} \tag{4.14}$$

Algorithm 4.5. *: Onset estimate based on change in pitch*

flag = OFF
for $k = 1$ *to* T
  if $\widetilde{\lambda_{auto}}(k-\psi)$ to $\widetilde{\lambda_{auto}}(k-1) = \widetilde{\lambda_{auto}}(k)$ and flag = OFF
    $C_b = k$
      $\cdots$ time sample before change in pitch
  end if
  if $\widetilde{\lambda_{auto}}(k-\psi)$ to $\widetilde{\lambda_{auto}}(k-1) \neq \widetilde{\lambda_{auto}}(k)$ or flag = ON
    flag = ON
  end if
  $\vartheta(k) = 0$
      $\cdots$ variable to denote onset due to change in pitch

if $\widetilde{\lambda_{auto}}(k-\psi)$ to $\widetilde{\lambda_{auto}}(k-1) = \widetilde{\lambda_{auto}}(k)$ or flag = ON
  $C_a = k - \psi$
    $\cdots$ time sample after change in pitch
  $\vartheta(\frac{C_b+C_a}{2}) = 1$
    $\cdots$ Onset due to change in pitch at $k = \frac{C_b+C_a}{2}$
  flag = OFF
 end if
end for

$\vartheta(k) = 1$ denotes the estimate for note onsets based on change in pitch.

Algorithm 4.6. *: Onset estimate based on change in instantaneous spectral variance*

for $k = 1$ *to* T
 if $\sigma^2(k) = \min_{p=k-\psi \text{ to } k+\psi}(\sigma^2(p))$
  $\xi(k) = 1$
    $\cdots$ Onset due to change in instantaneous spectral variance
 else
  $\xi(k) = 0$
 end if
end for

$\xi(k) = 1$ denotes the estimate for note onsets based on variations in spectral energy. Often, there may be a note onset even though the pitch does not change. There is just a change in energy associated with the two consecutive notes with the same pitch. Change in instantaneous spectral energy is a good measure to detect this.

Normalized Spectral Variance (NSV) is defined as

$$NSV(k) = \frac{1}{M-1}\sum_{n=1}^{M}\left(\frac{F_n(k) - \breve{F}_n(k)}{\breve{F}_n(k)}\right)^2 \tag{4.15}$$

where $\breve{F}_n(k)$ is obtained from Equation 4.2. NSV normalizes the instantaneous spectral variance with respect to the variations in amplitude. So, the NSV is independent of the

energy associated with the spectrum, and just depends on the change in spectral content. The first derivative (or difference) is called the Difference Normalized Spectral Variance ($DNSV$).

$$DNSV(k) = NSV(k) - NSV(k-1) \tag{4.16}$$

Figure 4.3 shows

(a) the plot of NSV for an artificially generated signal containing two sinusoids with different frequencies.
(b) the corresponding plot of $DNSV$.
(c) the spectrogram of the artificially generated signal.

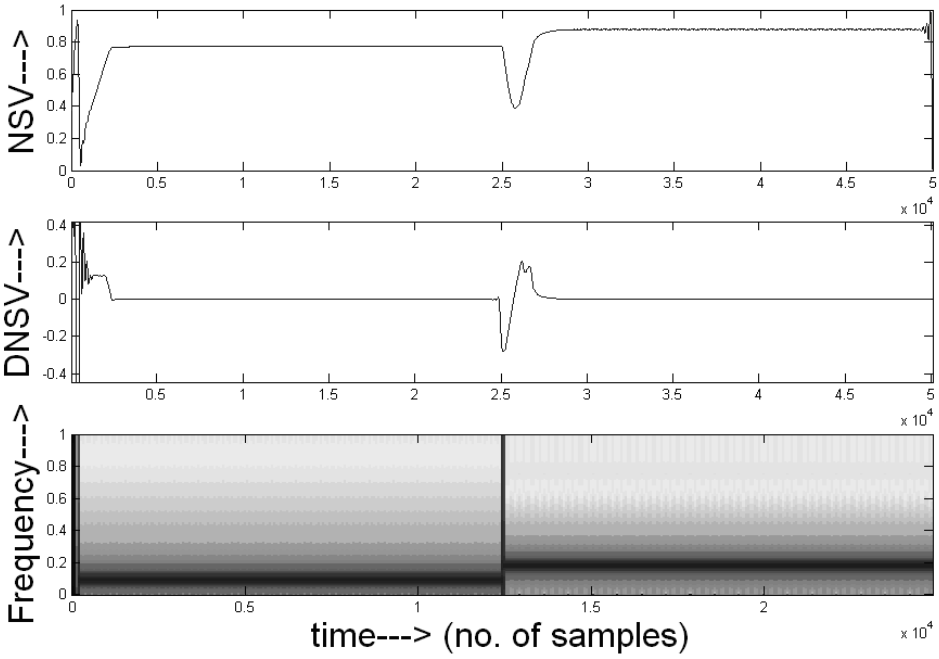

Figure 4.3: The plot of (a) Normalized Spectral Variance (NSV), (b) first difference of NSV and (c) the spectrogram of a synthesized sample, having two sinusoids ($F_s = 11000$Hz)

The note onset estimate, $\zeta(k)$, obtained based on the $DNSV$, is shown in Algorithm 4.7 given below.

Algorithm 4.7. *: Onset detection based on change in $DNSV$*

for $k = 1$ to T
  if $DNSV(k) = \min_{p=k-\psi \; to \; k+\psi}(DNSV(p))$
    if $DNSV(k) < \Theta$
      $\zeta(k) = 1$
        $\cdots$ Onset due to change in $DNSV$
  else
    $\zeta(k) = 0$
  end if
end for

$\zeta(k) = 1$ denotes the estimate for note onsets based on $DNSV$. The threshold, 'Θ', is calculated as

$$\Theta = \frac{\min_{k=1 \; to \; T}(DSNV(k))}{10^{\frac{-SNR}{20}}} \tag{4.17}$$

Often, the syllable pronounced while singing is only a vowel, i.e. a consonant does not precede or follow the vowel. A note produced with such a syllable does not have a drop in the instantaneous spectral variance, since the energy remains more or less constant. However, this change is picked up by a minima in the $DNSV$.

Since there are various ways in which note onsets can occur, it is necessary to come up with a strategy of detecting all these possible variations of note onset occurrences. Whether a region, Ψ, around time sample k contains a note onset or not, is decided by the operator $\widehat{\overline{A}}$, which maps the vector $\overline{A}$, to a boolean value as shown below.

For vector $\overline{A}$ of length $1 \leq i \leq P$

$$\begin{array}{ll} \text{if} & \sum_{i=1}^{P} \overline{A}(i) > 1 \\ & \widehat{\overline{A}} = 1 \\ \text{else} & \widehat{\overline{A}} = 0 \\ \text{end if} & \end{array} \tag{4.18}$$

The final decision about whether a note onset has occurred or not is made by the Algorithm 4.8. It employs the operator in Equation 4.18 on $\overline{\vartheta(p)}$, $\overline{\xi(p)}$ and $\overline{\zeta(p)}$, obtained from Algorithms 4.5, 4.6 and 4.7, respectively.

Algorithm 4.8. *: Final decision on note onset detection*

for $k = 1$ to T
  for vectors $\overline{\zeta(p)}$, $\overline{\xi(p)}$, $\overline{\vartheta(p)}$ where vector $\overline{p} = k - \Psi$ to $k + \Psi$
    $N_o(k) = (\widehat{\overline{\vartheta}} \wedge \widehat{\overline{\zeta}}) \vee (\widehat{\overline{\xi}} \wedge \widehat{\overline{\zeta}})$
      $\cdots N_o(k) = 1$ denotes note onset at k
  end for
end for

Figure 4.4 shows an example of note onset detection.

4.3 Time Domain Selection Based Algorithm

The second algorithm works in the time domain. It works on the basis of finding connected 2-D contours of high energy in the 'Time-Frequency' space, $F_n(k)$. Figure 4.5 shows the basic blocks of the algorithm. The silence removal (Algorithm 4.1), note onset detection (Sub-section 4.2.5) and median filtering (Sub-section 4.2.4) blocks are the same as in Section 4.2. The two blocks 'detect connected 2-D energy-contours' and 'detect harmonic energy-contours' are explained in Sub-sections 4.3.1 and 4.3.2 respectively.

4.3.1 Detection of Connected 2-D Energy-Contours

The 2-D 'TF' space is spanned by $F_n(k) \forall n \quad 1 \leq n \leq M, \ \forall k \quad 1 \leq k \leq T$. What this algorithm does, is to estimate those high energy 2-D contours that are connected in k axis, i.e. in time. The Algorithm 4.9 demonstrates how these contours are estimated. It is based on finding the most significant peak, $(\widehat{x}, \widehat{y})$, in the 2-D feature vector plane and tracing areas that are linked to this peak. Then, this region is subtracted from the original feature vector plane, $F_k(n)$. This process is continued till there is no peak in the entire feature vector

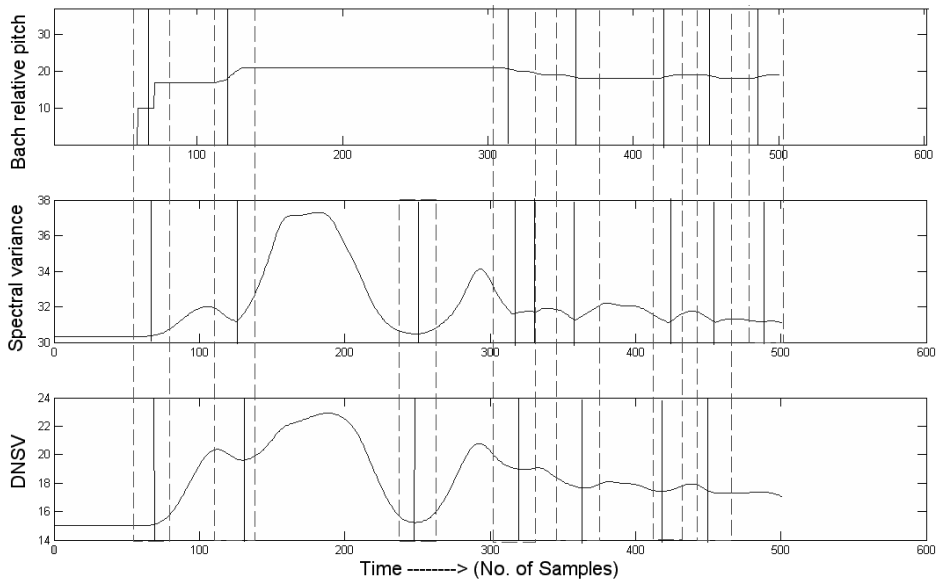

Figure 4.4: Example of note onset detection. The dashed lines indicate note onsets. The vertical solid lines indicate note onsets estimated by each of Algorithms 4.5, 4.6 and 4.7, respectively

plane. Each contour is represented by $\widehat{\lambda_i}(k) \forall k \quad 1 \leq k \leq T$, and N_c is total number of such energy-contours obtained. Figure 4.6 shows how energy-contours are extracted from a sample rendering of a song for which $N_c = 3$.

Algorithm 4.9. *: 2-D Connected Energy-contour detection*

while $\forall\ k \ \ 1 \leq k \leq T, n \ \ 1 \leq n \leq M$ there $\exists F_k(n) > \tau$

$\cdots$ where τ is the threshold

$i = 0$

$\{\widehat{x}, \widehat{y}\} = \arg \max_{k=1\ to\ T} \left(\arg \max_{n=1\ to\ M} F_k(n) \right)$

$F_{\widehat{x}}(\widehat{y} - K)\ to\ F_{\widehat{x}}(\widehat{y} + K) = 0$

$\cdots$ where K is obtained as in equation 4.5

for $k = \widehat{x} + 1$ to T

$y = \arg \max_{n=\widehat{y}-1 \; to \; \widehat{y}+1} (F_k(n))$

$\widehat{\lambda}_i(x) = y$ $\cdots$ where $\widehat{\lambda}_i(k) \forall k \;\; 1 \leq k \leq T$ is the i^{th} energy- contour

$\widehat{y} = y$

$F_k(\widehat{y} - K) \; to \; F_k(\widehat{y} + K) = 0$

end for

for $k = \widehat{x} - 1$ to 1

$y = \arg \max_{n=\widehat{y}-1 \; to \; \widehat{y}+1} (F_k(n))$

$\widehat{\lambda}_i(x) = y$ $\cdots$ where $\widehat{\lambda}_i(k) \forall k \;\; 1 \leq k \leq T$ is the i^{th} energy- contour

$\widehat{y} = y$

$F_k(\widehat{y} - K) \; to \; F_k(\widehat{y} + K) = 0$

end for

end while

$N_c = i$ $\cdots$ where N_c is the total number of energy-contours

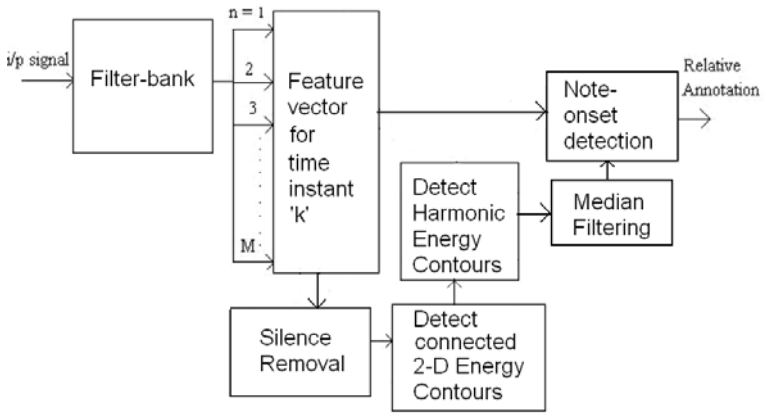

Figure 4.5: Block diagram of the time domain selection based algorithm

4.3.2 Detection of Harmonically Related Energy-Contours

We have N_c number of energy-contours. They may or may not be harmonically related to each other. This sub-section deals with trying to predict which of the contours are harmonically related. In order to achieve this, relations of the harmonic components to the fundamental are obtained, as shown in Table 4.1. The algorithm is as given below.

Algorithm 4.10. *: Detecting harmonically related connected energy-contours*

for $i = 1$ to N_c

$\widehat{\lambda_{sum}}(i) = \sum_{k=1}^{T} \left(F_k(\widehat{\lambda_i}(k)) \right)$

for $j = 1$ to N_c

$\widehat{\lambda_{sum}}(j) = \sum_{k=1}^{T} \left(F_k(\widehat{\lambda_j}(k)) \right)$

$\forall k \;\; 1 \leq k \leq T, \; \widehat{\lambda_{diff}}(k) = \widehat{\lambda_i}(k) - \widehat{\lambda_j}(k)$

$\widehat{\lambda_{mean}} = \sum_{k=1}^{T} \widehat{\lambda_{diff}}(k)$

if $\widehat{\lambda_{mean}} \in H$

$\cdots$ where H is obtained from equation 4.10

$\widehat{\lambda_i}(k) = \dfrac{\left(\widehat{\lambda_i}(k) + \widehat{\lambda_j}(k) + \widehat{\lambda_{mean}} \right)}{2}$

$\widehat{\lambda_{sum}}(i) = \dfrac{\widehat{\lambda_{sum}}(j) + \widehat{\lambda_{sum}}(i)}{2}$

end if

end for

end for

$\lambda_{auto}(k) = \widehat{\lambda_{imax}}(k)$ such that $imax = \arg \max_{i=1 \; to \; N_c} (\widehat{\lambda_{sum}}(i))$

The difference between two energy-contours is obtained as $\widehat{\lambda_{diff}}(k) \forall k \;\; 1 \leq k \leq T$. If its mean, $\widehat{\lambda_{mean}}$, belongs to one of the values indicated in the Table 4.1, then the two contours are harmonically related. The two contours are averaged to obtain a new contour. The total energy of the new contour, $\widehat{\lambda_{sum}}$, is calculated. Finally, the contour with the highest $\widehat{\lambda_{sum}}$, is considered as the '*Pitch Contour*', i.e. $\lambda_{auto}(k) \forall k \;\; 1 \leq k \leq T$. A typical contour, generated by the Algorithms 4.9 and 4.10, is shown in Figure 4.7.

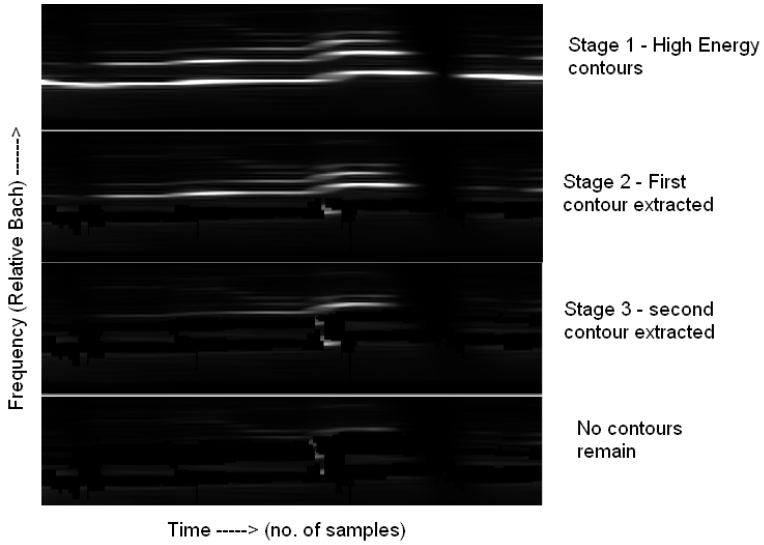

Figure 4.6: An example showing how contours are extracted by Algorithm 4.9 for a sample rendering of a song

4.4 Representation of the Transcription

The two methods of transcription to relative *Bach* scale, and the note onset detection algorithms, complete the transcription requirements of a query made by a user to a query-by-humming system. Typically, this transcription is compared to the transcription of existing songs in the database and the closest matches are chosen to be displayed to the user. However, a direct comparison of the transcription of a rendering to that of another rendering will result in quite a poor performance. The reason for that is the variations in duration, absolute pitch and additional notes or deletion of notes in the rendering. In order to compensate for this sort of variations in rendering, it is necessary to obtain a useful representation.

A suggested representation for an arbitrary rendering, 'X', is as follows. The pitch contour is converted into a set of 3 variables:

$$\Gamma_X(i) = \{\Upsilon_{start}, \lambda_{start}, \lambda_{end}\} \;\; \forall i \;\; 1 \leq i \leq T_N \tag{4.19}$$

where Υ_{start}, λ_{start} and λ_{end} represent the note onset time (seconds), starting relative fre-

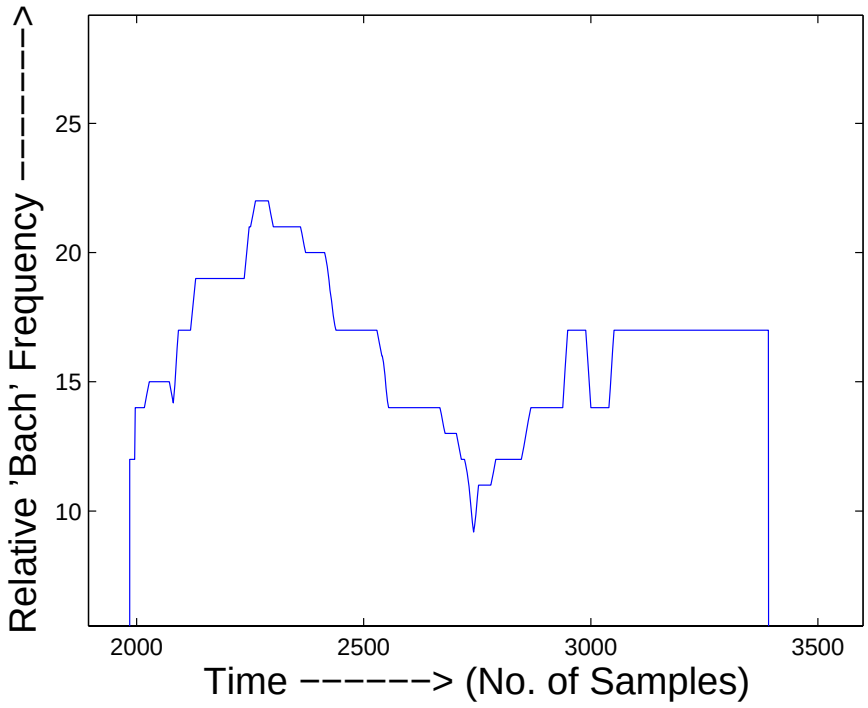

Figure 4.7: A typical contour generated by the Algorithms 4.9 and 4.10

quency, and the ending relative frequency of the i^{th} note, respectively. In the case of glides or pitch bends, λ_{start} and λ_{end} would be different, and in the case of regular notes, they would be the same. T_N is the total number of note onsets detected.

4.5 Evaluation and Results

The algorithm was tested for 110 renderings from 12 male and 10 female singers. The renderings included singing with words, humming with /m/ and /n/ (nasal) phones, and whistling. The recording was done with a SNR of 25 dB and sampling rate (F_s) of 11 KHz. The transcription for the above renderings was done by trained musicians.

Figure 4.8 shows

1. the instantaneous spectral variance of a singing voice signal
2. the corresponding DNSV and

3. the pitch contour (λ_{man}) of the signal obtained by converting manual transcription to relative *Bach* frequencies.

Figure 4.9 shows

1. the pitch contour generated by the algorithm in Section 4.2 and
2. the pitch contour of the manually transcribed file converted into the relative *Bach* frequencies.

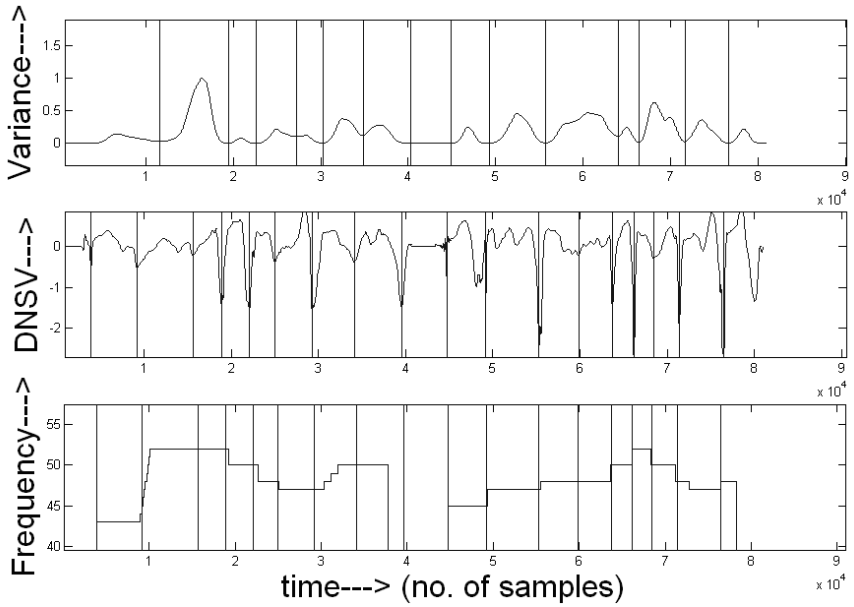

Figure 4.8: The plot of (a) σ^2, (b) $DNSV$ and (c) manually annotated frequencies of the singing voice sample. The vertical lines in the figures indicate the output of the segmentation (note onsets) by Algorithms 4.6, 4.7 and 4.5, respectively. ($F_s = 11000$Hz)

There are two ways of analyzing the error of the algorithm. The first method does not take into account the onset timings and finds the fidelity with which the pitch is tracked. The percentage error (%E) (which is a slight modification of the evaluation method used by

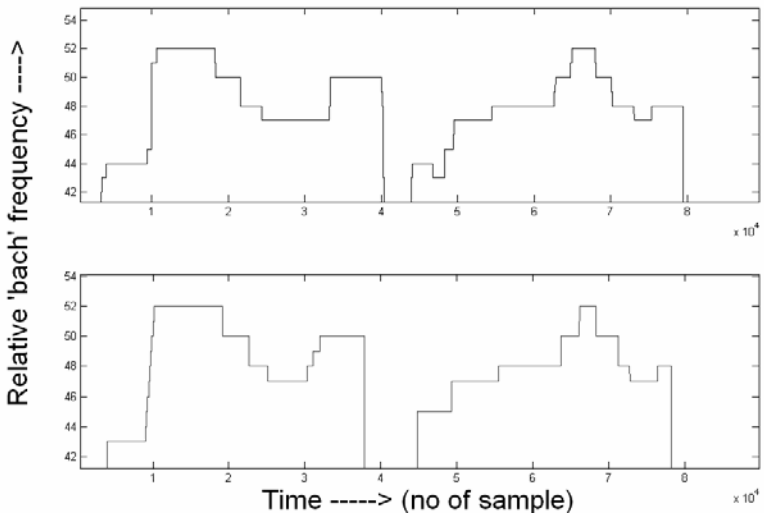

Figure 4.9: (a) The pitch contour generated by the algorithm in Section 4.2, (b) the pitch contour obtained by converting manual transcription to relative *Bach* frequencies (F_s = 11000Hz)

[44]) is calculated as follows

$$\%\mathrm{E} = \frac{\sum_{k=1}^{T} \| \left(\lambda_{auto}(k) - \lambda_{man}(k) - \mu_{auto} + \mu_{man}\right) \|}{\sum_{k=1}^{T} \| \left(\lambda_{man}(k) - \mu_{man}\right) \|} \tag{4.20}$$

where $\lambda_{man}(k),\ \forall k\ \ 1 \leq k \leq T$, is the pitch contour obtained by converting the manually transcribed data to relative *Bach* frequencies. μ_{auto} and μ_{man} are the means of the pitch contours and are calculated for an arbitrary pitch representation $\Gamma_X$as follows.

$$\mu_X = \frac{1}{T_N} \sum_{i=1}^{T_N} \Gamma_X(i, 2) \tag{4.21}$$

Γ_X is obtained as explained in Equation 4.19.

The second method employs the use of the onset timings. The following measures can be defined.

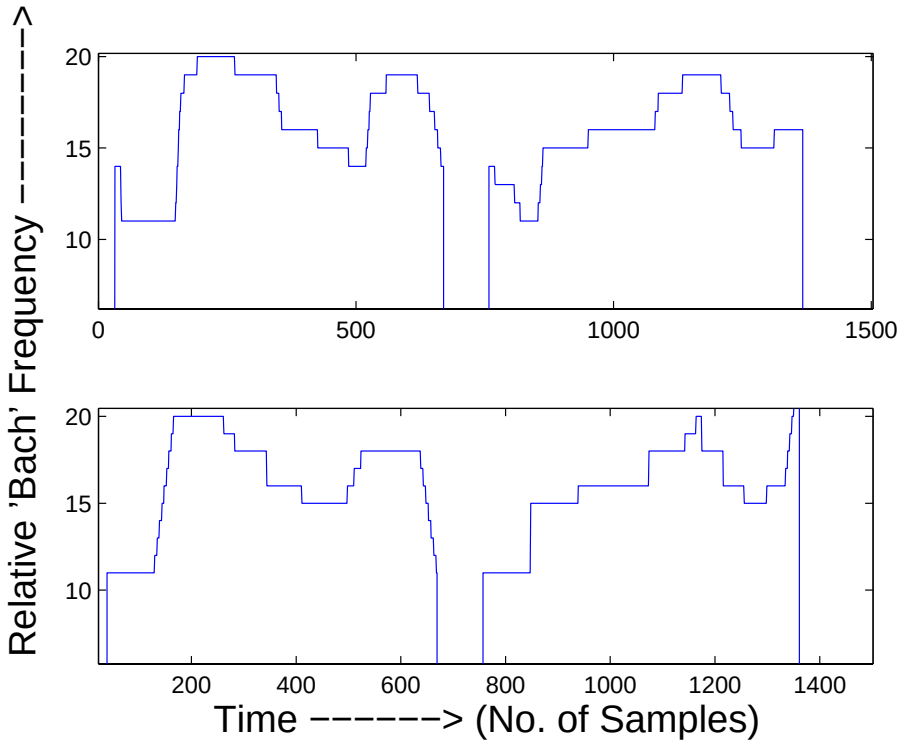

Figure 4.10: (a) The pitch contour generated by the algorithm in Section 4.3, (b) the pitch contour obtained by converting manual transcription to relative *Bach* frequencies (F_s = 220Hz)

- If a unique automated note onset is detected within 100ms (Ψ) of a manual note onset, then it is considered as a correct detection of onset (%O).
- If there is no automated onset within 100ms of a manual note onset, then it is considered as a deleted onset (%D).
- If there is no manual note onset within 100 ms of an automated note onset, then it is considered as an inserted onset(%I).
- If the value of the automated relative *Bach* frequency for a correct onset is equal to that of the manually transcribed one, then it is considered as correct transcription (%T).
- If the silence regions are detected within 100 ms of an actual silence region, then it is considered a correct detection of the silence region (%S). This quantity evaluates the

efficacy of Algorithm 4.1

The results of the 110 test files are shown in Tables 4.2, 4.3 and 4.4. For the tested data, the results are encouraging. Audio reconstruction of the transcribed data was played back to the singers, who approved of the transcription quality. It could be said that the algorithm is robust for query-by-humming applications since the error rate is more or less constant for the different possible methods of queries. From Tables 4.2, 4.3 and 4.4, we can see that the proposed algorithm is more consistent than the standard pitch tracking algorithm (based on autocorrelation function of the time signal) used in a commercial software called PRAAT [49]. The pitch values obtained from the PRAAT algorithm are rounded off to the nearest *Bach* value and compared with manually annotated relative pitch values. The optimal settings used in the PRAAT software were as follows.

- Pitch range (Hz) is 55 to 550 Hz
- Optimize for Intonation method
- Maximum number of pitch candidates are 15
- Silence Threshold is 0.03
- Voicing Threshold is 0.45
- Octave cost is 0.01
- Octave jump cost is 0.35
- Voiced-unvoiced cost is 0.14

We can see that the percentage error increases drastically for whistling. The robustness can be attributed to the use of the *Bach* filters, and the algorithm which estimates the relative frequency as opposed to the absolute fundamental frequency.

The note onset detection leads to more errors in transcription than the pitch tracking. Insertion or deletion of note onsets cause the highest loss in the quality of transcription. The number of inserted note onsets is high for singing with words, while the number of deleted note onsets is high for whistling. The time-contour based algorithm performs marginally better than the frequency selection based algorithm.

$\%E$ - Error rate for pitch tracking

$\%O$ - Correct detection of onset

$\%D$ - Deleted onset

$\%I$ - Inserted onset

$\%T$ - Correct transcription

$\%S$ - Correct detection of silence

Table 4.2: Performance evaluation of the algorithm for singing with syllables or words for both male and female voices (60 files)

%E for pitch detection using PRAAT	%E for freq. selection based method in Section 4.2	%E for time -contour based method in Section 4.3	%O	%D	%I	%T	%S
23.3 %	12.2 %	8.2 %	83.4 %	13.4%	31.2%	79.9%	83.1 %

Table 4.3: Performance evaluation of the algorithm for singing with nasals /m/ and /n/ for both male and female voices (25 files)

%E for pitch detection using PRAAT	%E for freq. selection based method in Section 4.2	%E for time -contour based method in Section 4.3	%O	%D	%I	%T	%S
15.8 %	14.3 %	9.2 %	84.1%	18.8%	21.3%	79.1%	81.2%

4.6 Conclusion and Future Work

Thus, an algorithm for transcribing human singing voice into relative pitch contours has been suggested. It is shown to be robust for most forms of human queries. The pitch tracking shows sufficiently low error rates. Although the fundamental frequency detection algorithm in PRAAT is extremely accurate, transcription to musical semitones is rather poor. Earlier

Table 4.4: Performance evaluation of the algorithm for whistling (25 files)

%E for pitch detection using PRAAT	%E for freq. selection based method in Section 4.2	%E for time -contour based method in Section 4.3	%O	%D	%I	%T	%S
66.4 %	18.2 %	10.4 %	83.3%	17.2%	8.5%	76.3%	72.3%

efforts at transcribing voices show, that one needs a higher level of learning based on musicological knowledge [44], to improve the transcription accuracy. We can see that even though the algorithms presented are not as accurate as the pitch detecting algorithm in PRAAT, they do a better job at transcription to musical semitone notes, without any musicological learning. However, the note detection needs improvement in order to have a really robust system in place. A representation scheme suitable for query-by-humming systems has also been suggested in this chapter.

The validity of the suggested algorithm remains to be tested on a larger database. It may be worthwhile to extend these algorithms to include polyphonic music pieces, predominantly containing human singing voices. A robust note onset detection algorithm is required to make the quality of transcription even better.

Chapter 5

A Strategy to Find Distances between Two Musical Renderings

5.1 Introduction and Motivation

Most of the problems associated with query-by-humming systems are not to do with a poor quality of transcription, but more to do with a poor representation of the transcription. The features suggested by Shakra [50] include interval sequence, melodic contour and note duration sequence. However, as noted by Sorsa [51], the errors included in the search-keys generated by the users are the main cause of failures of such query-by-humming systems.

The techniques described in [52], [53] and [54] treat the query and tune as sequences of pitch-duration tuples, and use dynamic programming to match a query to a tune. That is, the match is described as the minimum number of deletions, insertions or replacements of notes required to transform the query into the tune excerpt. This is commonly called 'Dynamic Time Warping' (DTW). This is analogous to matching notes in a musical score. The disadvantage of this scheme is that it fails to take into account the time-based nature of music. Francu [55] suggested a metric to quantify distance between renderings, but he assumes that the two renderings are at around the same pace or tempo.

However, the following problems are common in queries. They need to be addressed in detail.

1. Query may be at a different pace or tempo as compared to that of the rendering of the same song in the database.

2. Query is usually at a different absolute pitch as compared to that of the rendering in the database.

3. Deletions and insertions of notes are extremely common occurrences in queries.

4. The query may consist of phrases which are not in the same chronology (or sequence) as that in the rendering available in the database.

5. The tempo may not remain constant over a query.

6. The absolute pitch may not remain constant over a query.

5.2 Features Required for Obtaining Distance

5.2.1 Phrase

Songs, especially those rendered vocally, are made of phrases. There is usually a silence or a pause between two phrases, which is utilized for catching one's breath. We can define a 'phrase' as a contiguous unit of a song between two pauses. As noted in [51], longer search keys usually perform worse than shorter search keys, because the tempo and the pitch tend to change over a long query, even for trained singers. Another reason for the failure is a difference in the sequence of the phrases.

In order to overcome these problems, it is suggested that phrases may be used as the unit for finding the distance between two renderings. Usually, phrases are not long enough to bring about a noticeable variation in tempo or pitch. An algorithm is presented below, which can detect the start and end of a phrase in a rendering, $X(k) \forall k \;\; 1 \leq k \leq T_X$. T_X is the length of rendering X.

Algorithm 5.1. *: Phrase detection*

$P_{flag} = \text{OFF}$ $\cdots$ P_{flag} identifies phrase or silence region
$i = 0$
for $k = 1$ to T_X
    if $P_{flag} = \text{OFF}$

if $20 * log_{10}\left(\max_{k=1\ to\ T_X} \sigma^2(k)\right) - 20 * log_{10}(\sigma^2(k)) > SNR$
and $\sigma^2(k+\psi) > 2 * \sigma^2(k)$
$P_{flag} = \text{ON}$
$SP_X(i) = k \cdots SP_X(i)$ indicates the start of i^{th} phrase
end
else
if $20 * log_{10}\left(\max_{k=1\ to\ T_X} \sigma^2(k)\right) - 20 * log_{10}(\sigma^2(k)) > SNR$
and $\sigma^2(k+\psi) < 0.5 * \sigma^2(k)$
$P_{flag} = \text{OFF}$
$EP_X(i) = k \ \cdots EP_X(i)$ indicates the end of i^{th} phrase
$i = i + 1$
end
end if
end for
$Ł_X = i$ $\cdots Ł_X$ is the total number of phrases for the rendering X

Here, ψ is defined as in Equation 4.14.

5.2.2 Energy Profile

The energy profile, $\widehat{E_X}(k)$, is defined for a pitch contour, $\lambda_X(k) \forall k \ \ 1 \leq k \leq T_X$, of a rendering X.

$$\forall k \ \ 1 \leq k \leq T_X \ \ \widehat{E_X}(k) = F_k(\lambda_X(k)) \tag{5.1}$$

$\lambda_X(k)$ is obtained from either Algorithm 4.2 or 4.3. $F_k(n)$, $\forall k \ \ 1 \leq k \leq T_X \ \forall n \ \ 1 \leq n \leq M$, is the feature vector obtained as in Equation 2.1. The motivation for using this feature is giving greater importance to those regions in the pitch contour that are more reliable in their transcription.

Thus, the phrase information $SP_X(i)$ and $EP_X(i) \ \forall i \ \ 1 \leq i \leq Ł_X$, the energy profile $\widehat{E}(k) \ \forall k \ \ 1 \leq k \leq T_X$ and the pitch contour $\lambda_X(k) \ \forall k \ \ 1 \leq k \leq T_X$ are needed to frame the distance function.

5.3 Finding the Distance between Two Renderings

The strategy for a query-by-humming should be formulated in such a way that even if a few extra matches come up, the correct matches should not be lost. So, the proposed distance function finds the minimum possible distance between two renderings, rather than the average distance.

The distance function, defined for pitch contour vectors $\overline{\lambda_X}$, $\overline{\lambda_Y}$ and Energy profile vectors $\overline{\widehat{E_X}}$, $\overline{\widehat{E_Y}}$, respectively, is as follows.

$$D(X,Y) = (\overline{\lambda_X} - \overline{\lambda_Y})^T * \overline{\widehat{E_X}} * \overline{\widehat{E_Y}^T} * (\overline{\lambda_X} - \overline{\lambda_Y}) \tag{5.2}$$

Thus, it finds the Euclidian distance between the pitch contours weighting it against the energy profiles. More importance is given to the areas of the pitch contour with higher energy values. In order to implement this distance function for actual renderings, the following algorithm has to be implemented.

We define a function, $resample(\overline{A}, fac)$, which re-samples vector $\overline{A}$ by a factor 'fac'. The distance is found between renderings X, $\forall k \;\; 1 \leq k \leq T_X$, and Y, $\forall k \;\; 1 \leq k \leq T_Y$, as follows

Algorithm 5.2. *: Algorithm to implement the distance function for two renderings*

for $i = 1$ to Ł$_X$
  for $j = 1$ to Ł$_Y$
    $fac = \dfrac{EP_Y(j) - SP_Y(j)}{EP_X(i) - SP_X(i)}$
    $\overline{\lambda_X} = \lambda_X\,(\forall k \in [SP_X(i), EP_X(i)]) - \mu_X$
      $\cdots$ μ_X is obtained from equation 4.21 in the interval $[SP_X(i), EP_X(i)]$
    $\overline{\lambda_X} = resample(\overline{\lambda_X}, fac)$
    $\overline{\lambda_Y} = \lambda_Y\,(\forall k \in [SP_Y(j), EP_Y(j)]) - \mu_Y$
      $\cdots$ μ_Y is obtained from equation 4.21 in the interval $[SP_Y(j), EP_Y(j)]$
    $\overline{\widehat{E_X}} = \widehat{E_X}(\forall k \in [SP_X(i), EP_X(i)])$

$$\overline{\widehat{E_X}} = resample(\overline{\widehat{E_X}}, fac)$$
$$\overline{\widehat{E_Y}} = \widehat{E_Y}(\forall k \in [SP_Y(i), EP_Y(i)])$$
$$D(i,j) = (\overline{\lambda_X} - \overline{\lambda_Y})^T * \overline{\widehat{E_X}} * \overline{\widehat{E_Y}}^T * (\overline{\lambda_X} - \overline{\lambda_Y})$$

end for

end for

$$Dist(X,Y) = \frac{1}{Ł_X} \sum_{i=1}^{Ł_X} \left(\min_{j=1\ to\ Ł_Y} (D(i,j)) \right)$$
$$Dist(Y,X) = \frac{1}{Ł_Y} \sum_{j=1}^{Ł_Y} \left(\min_{i=1\ to\ Ł_X} (D(i,j)) \right)$$

Thus, the distance function, $Dist(X,Y)$, is normalized against number of phrases.

5.4 Results and Discussion

In order to evaluate the results, 9 singers (5 female and 4 male) were asked to sing the same set of 35 Hindi and English songs. Thus, for each song, 9 renderings or versions are available. Each rendering has a different pace (tempo) and different absolute pitch. Some of the renderings are sung with words or syllables like /la/, /ta/ or /da/, while other are hummed with nasals like /m/ and /n/, while some of them are whistled. The sequence of phrases too is altered in some renderings.

A comparison is made between the proposed method and a standard method using Dynamic Time Warping (DTW) of the sequence of notes $(\Gamma_X(i) \forall i \ \ 1 \leq i \leq T_N)$, obtained from Equation 4.19. The DTW is performed on the series of $\frac{\lambda_{start}}{\mu_X}$, where μ_X is obtained from Equation 4.21. Equal penalty is given for insertion, deletion and substitution of notes.

Every song in this database of 315 songs was compared to the remaining songs in the database, and their distance measures found by the proposed method and the DTW based method. Obviously, the pair with the minimum distance is the best match. If the nearest match for a particular rendering is a different rendering (or version) of the same song, then it is considered a correct match. Table 5.1 shows what percentage of correct matches are obtained among the first 10 nearest neighbours and Table 5.2 shows the same results among

the 5 nearest neighbours. As we can see, the proposed method performs better than the standard DTW algorithm, and gives over 95 percent accuracy for at least one in the best 10 matches. It shows that the proposed method is robust for the various forms of queries and also for rendering of altered phrases.

Table 5.1: Percentage of songs having 'N' number of correct matches among the first 10 neighbours

N	%Songs with 'N' correct matches by proposed method	%Songs with 'N' correct matches by DTW
1	96.4 %	77.7 %
2	73.6 %	63.4 %
3	44.7 %	45.9 %
4	30.2 %	34.3 %
5	15.7 %	24.3 %

Table 5.2: Percentage of songs having 'N' number of correct matches among the first 5 neighbours

N	%Songs with 'N' correct matches by proposed method	%Songs with 'N' correct matches by DTW
1	89.3 %	61.4 %
2	44.6 %	31.4 %
3	20.3 %	14.2 %

5.5 Conclusions

A strategy for measuring distance between two renderings is proposed along with a distance measure. It has been evaluated on a database of 315 songs of both male and female voices and whistle queries. The results shown are encouraging. The correct song is picked up among the top 5 best choices around 90% of the time. The database consists of queries with varying absolute pitch, tempo (pace) and also varying sequence of phrases. Further testing needs to be done on a larger database.

Chapter 6

Concluding Remarks

A Time-Frequency representation of an audio signal using the *Bach* scale filter-bank has been proposed. The *Bach* scale tries to model the human perception of music. The TF representation is a time-varying representation, since every time instant is represented by a unique set of frequency features.

When applied in automated speech segmentation, the TF representation using *Bach* filters performs better than existing methods. This is rather surprising, because the *Bach* scale does not emulate the performance of the human ear in any way. But, the *Bach* scale probably gives us an idea about how humans interpret frequencies. Two frequencies, that fall within the scope of a semitone, may not be distinguished, even though the difference may be perceived. This opens up a wide scope of research, which deals with discerning between the ability to perceive and choosing to perceive. However, adequate studies have to be performed in other applications, like speech recognition, unit selection for TTS and speech coding, before coming to any conclusion about the effectiveness of the *Bach* scale in speech.

The two pitch tracking algorithms proposed in this thesis are robust for most types of queries made to a query-by-humming system. Though the pitch tracking algorithms have sufficiently low error rates, the problem of good transcription lies in the detection of note onsets. The problem is a complex one, because there are different ways in which a note onset can occur. The three step approach proposed in this thesis alleviates the problem to some extent. So, it is possible to handle most of the ways in which a note onset can occur.

In most pitch tracking algorithms, it is necessary to specify a range of frequencies where the fundamental frequency should lie. The range is usually different for male and female voices and whistling. The proposed algorithms do not need the range of frequencies as a parameter, because they try to track only the relative pitch. In fact, we don't need the absolute pitch of the rendering in query-by-humming applications, since it is unlikely that the singer will render the song in the same absolute pitch of the original sound track, in any case. This property has been exploited by the proposed algorithms to achieve robustness. So, even though the actual pitch may be different, the pitch estimates in relation with the other pitch values in the contour are consistent. The better performance of the proposed algorithms when compared to similar pitch tracking methods may be attributed to the use of *Bach* scale filter-banks.

A representation of the transcription, suited for query-by-humming applications, has been suggested. This representation, though useful, is not efficient for matching renderings of a song. This is because of a high number of deletions and insertions of note onsets by the music transcription process. So, a new distance measure based on the energy profiles of the renderings has been suggested. It has also been demonstrated to have a better performance, than the method using DTW on the note onsets.

The transcription of a query is just one aspect of the problem in query-by-humming systems. The second part is the more difficult one, which involves transcription of polyphonic music. Only when we have a robust system that can transcribe the lead track of a polyphonic music, can we build a successful query-by-humming system. This is an extremely difficult problem because of the wide variety of music present. Machine learning algorithms, which can predict the likely pitch contours, would fail to be general and robust for different types of music. So, it is necessary to first develop a non data-driven system which is fairly consistent over a large class of songs. It is possible to adapt the proposed algorithms to track pitch in polyphonic music. Future work should be directed towards this objective.

Appendix A

Selection of the '*base*' Frequency

According to the formulation of the '*Bach*' filter-bank in Sub-section 2.2.4, the '*base*' frequency can be arbitrarily chosen to be between 50 and 80 Hz. The remaining filters are designed accordingly. However, selection of the *base* frequency depends on the signal itself and is crucial in the pitch tracking algorithms. This is because, the singer may sing at an arbitrary absolute pitch. In order to solve the problem, five filter-banks with different *base* frequencies are designed. Table A.1 shows the range of frequencies each filter in the filter-bank covers, for each of the following five *base* frequencies, namely 53, 54, 55, 56 and 57 Hz. It shows the ranges of only the first 10 filters of the filter-bank. Actually M such ranges are obtained, where M is obtained from Equation 2.2.

We can see that the ranges covered are very similar for the set of filters with $base = 53$ and $base = 56$, and similarly for $base = 54$ and $base = 57$. The second filter with $base = 53$ Hz, has the same range as the first filter with $base = 56$ and so on. Thus, it is sufficient to have to select from the three *base* frequencies, namely 54, 55 and 56 Hz. The range for filter-banks with other *base* frequencies will be covered by one of these three set of filter-banks. Selection among these three *base* frequencies is another problem altogether. Several methods could be suggested. One among them is stated as below.

Algorithm A.1. *: Selection of* 'base' *frequency*

1. Select a non-silent region of the signal.

2. Find a high resolution Fourier Transform of the signal, i.e. $F_{ft}(\omega)$, where $\omega \in [0, \pi]$

Table A.1: Five base frequencies, and the corresponding ranges covered by the first ten filters

Filter Number	Range for *base*=53 Hz	Range for *base*=54 Hz	Range for *base*=55 Hz	Range for *base*=56 Hz	Range for *base*=57 Hz
1	51.5 - 54.5	52.4 - 55.6	53.4 - 56.6	54.4 - 57.6	55.4 - 58.6
2	54.5 - 57.8	55.6 - 58.9	56.6 - 60.0	57.6 - 61.0	58.6 - 62.1
3	57.8 - 61.2	58.9 - 62.4	60.0 - 63.5	61.0 - 64.7	62.1 - 65.8
4	61.2 - 64.9	62.4 - 66.1	63.5 - 67.3	64.7 - 68.5	65.8 - 69.8
5	64.9 - 68.7	66.1 - 70.0	67.3 - 71.3	68.5 - 72.6	69.8 - 73.9
6	68.7 - 72.8	70.0 - 74.2	71.3 - 75.5	72.6 - 76.9	73.9 - 78.3
7	72.8 - 77.1	74.2 - 78.6	75.5 - 80.0	76.9 - 81.5	78.3 - 83.0
8	77.1 - 81.7	78.6 - 83.3	80.0 - 84.8	81.5 - 86.3	83.0 - 87.9
9	81.7 - 86.6	83.3 - 88.2	84.8 - 89.9	86.3 - 91.5	87.9 - 93.1
10	86.6 - 91.7	88.2 - 93.5	89.9 - 95.2	91.5 - 96.9	93.1 - 98.7

3. For $base = 54$, 55 and 56 Hz, find

$$S_{ft}(base) = \sum_{n=1}^{M} \frac{1}{RH_{base}(n) - RL_{base}(n)} \int_{RL_{base}(n)}^{RH_{base}(n)} F_{ft}(\omega)\, d\omega \tag{A.1}$$

where

$$RL_{base}(n) = 2 * \pi * base * 2^{\frac{n}{12}} - \frac{base * 2^{\frac{n}{12}} - base * 2^{\frac{n-1}{12}}}{2 * F_s} \tag{A.2}$$

and

$$RH_{base}(n) = 2 * \pi * base * 2^{\frac{n}{12}} + \frac{base * 2^{\frac{n+1}{12}} - base * 2^{\frac{n}{12}}}{2 * F_s} \tag{A.3}$$

4. Select the *base* frequency corresponding to the largest $S_{ft}(base)$.

5. Use the filter-bank designed with the best *base* frequency.

Figure A.1 shows a typical high resolution fourier transform of a song rendering.

The effectiveness of the algorithm for five different song renderings is demonstrated in Table A.2. The error rate is calculated for the time-contour based method (Section 4.3). $\%E$, as defined in Sub-section 4.5, is usually the lowest for the highest $S_{ft}(base)$. However, it can also be seen that the error for filter-banks with different *base* frequencies does not vary by more than 1 to 2 %. So, the variation in the *base* frequency may not cause too much of a difference to the pitch tracking algorithm.

Table A.2: $S_{ft}(base)$ for various *base* frequencies and corresponding $\%E$ (defined in Subsection 4.5)

Parameters	Song 1	Song 2	Song 3	Song 4	Song 5
$S_{ft}(54)$	2712	1160	2561	1403	315.9
$S_{ft}(55)$	2717	1159	2542	1400	316.2
$S_{ft}(56)$	2715	1158	2562	1404	316.3
% Errors					
$\%E$ for $base = 54$	8.3	9.2	10.2	7.3	8.1
$\%E$ for $base = 55$	5.3	9.4	10.3	7.5	7.3
$\%E$ for $base = 56$	6.5	9.5	10.1	7.1	7.3

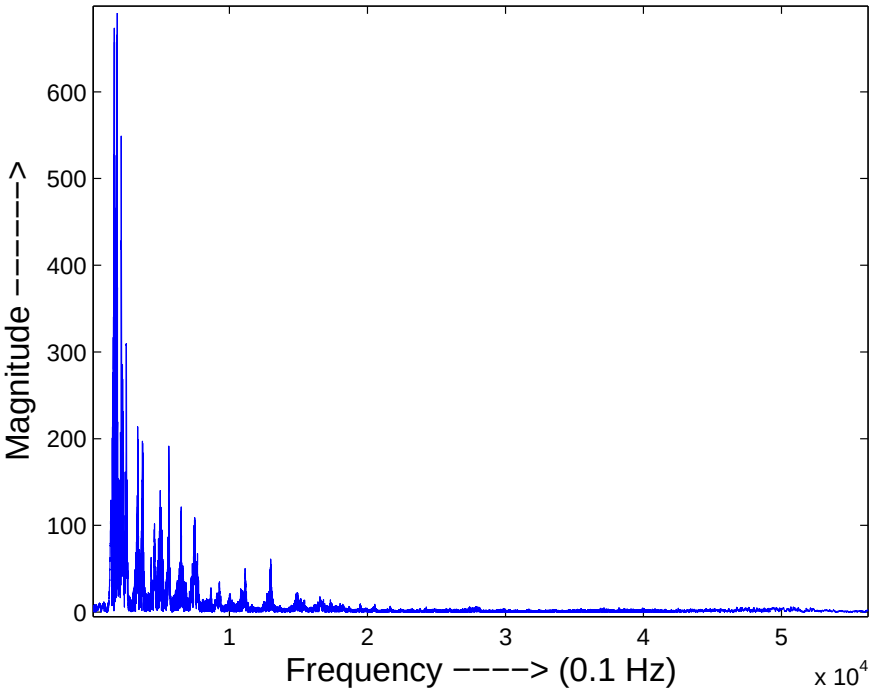

Figure A.1: High resolution Fourier transform of a sample rendering of a song

Appendix B

Determination of Time Resolution of the Human Ear at Various Frequencies

The number of coefficients for a filter in a filter-bank is determined from Equation 2.26, for the formulation given in Chapter 2. However, there is a perceptual way of determining the number of coefficients required for a particular filter. This depends on the time resolution available to the human ear at a particular frequency. An experiment was set up in order to determine this time resolution at various frequencies.

B.0.1 Motivation

The design of the experimental setup to determine the time resolution of the human ear, is based on the unique ability of the human ear to perceive 'beats', when two sounds with very close frequencies occur together. Beats are nothing but a result of the fluctuations in amplitude, as shown in Figure B.1. When two frequencies f_1 and f_2 are heard together, then the effect is the same as amplitude modulation with a carrier frequency $f_c = f_1 + \frac{f_2 - f_1}{2}$ (assume $f_1 < f_2$), modulated by a frequency of $f_m = \frac{f_2 - f_1}{2}$, with a modulation index of ∞.

When $f_2 - f_1$ is very small, the ear cannot perceive a difference in the frequency if the two occur one after the other. However, if heard together, beats are perceived. We know that the perceived time period of these beats is $\frac{1}{f_2 - f_1}$. As the difference between f_1 and f_2 keeps increasing, the rate at which the beats are heard, keeps increasing. We start hearing a sound with some roughness. However, below a particular time period, we stop hearing

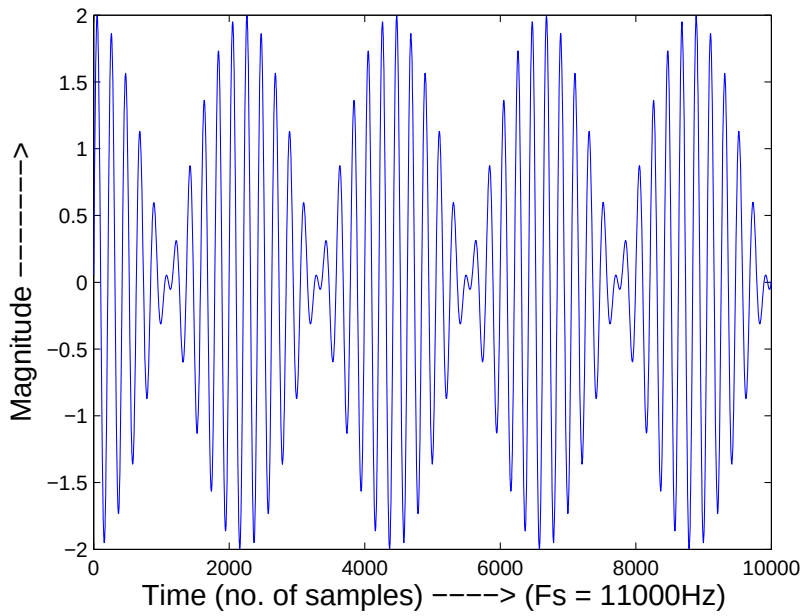

Figure B.1: The effect of hearing two frequencies, $f_1 = 100$ Hz and $f_2 = 110$ Hz, together

beats or roughness, and we can hear two separate smooth frequencies. In terms of amplitude modulation, we can say that there is a limit to the perception of the modulation frequency.

What is interesting is, that the limit to the perception of the modulation frequency depends on the carrier frequency itself. In other words, the minimum time period of beats that can be heard depends on the frequencies themselves. This limit is taken as the threshold for time resolution of the human ear. In order to do a systematic study on this perceptual phenomenon, an experiment was conducted.

B.0.2 Experimental Setup

Five users were provided with hi-fi headphones having a flat frequency response. Two random frequencies with a small difference were selected to be played together. These frequencies were normalized to equal loudness at 40 db [16]. The user was told to respond with '1' if he/she heard beats or roughness and with '0' if he/she heard a smooth sound, with two separate frequencies. If the user responded with '1' initially, the difference between the frequencies was increased gradually till he responded with '0'. The difference of frequencies

was noted when he responded with '0', and the inverse of the difference is taken as the time resolution of the human ear at the mean of the two frequencies. Each user was asked to repeat the experiment for 50 such frequency pairs. Figure B.2 shows the data obtained by the experiment.

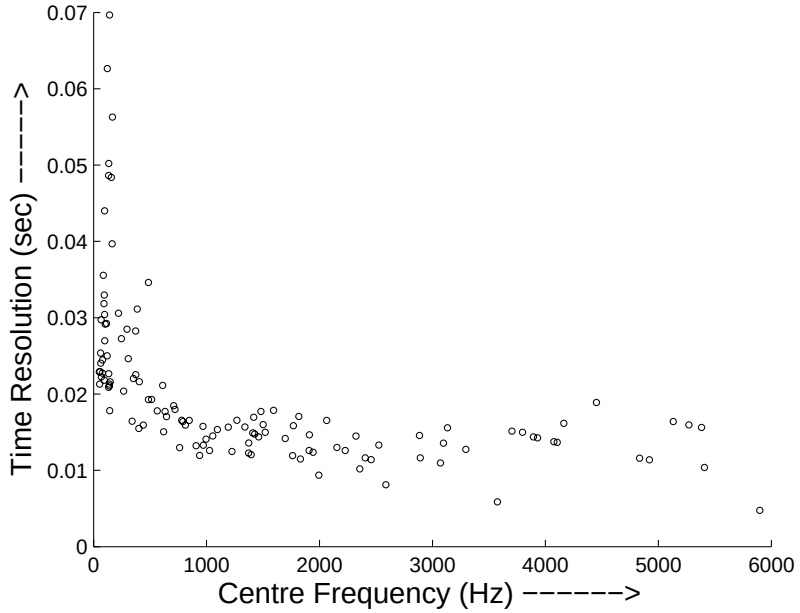

Figure B.2: The experimental values obtained for time resolution

B.0.3 Discussion of Results

A power function was fit into the experimental data, and the best fit was for function

$$tr(f) = 0.6876 * f^{-0.00794} - 0.6328 \tag{B.1}$$

where 'tr' is the time resolution in seconds, as a function of f, the frequency in Hz.

Figure B.3 shows the fit of the Equation B.1. Figures B.4, B.5, B.6 and B.7 compare the experimental data with the length of the filters obtained from various frequency scales. Typically, it is considered that the roughness is not heard outside the critical bandwidth of hearing. However, the experimental data shows that perception of roughness stops much

before the edge of the critical bandwidth. The non-linear formulation of the *Bach* scale has the best approximation at lower frequencies and the *Mel* scale has the best approximation at higher frequencies.

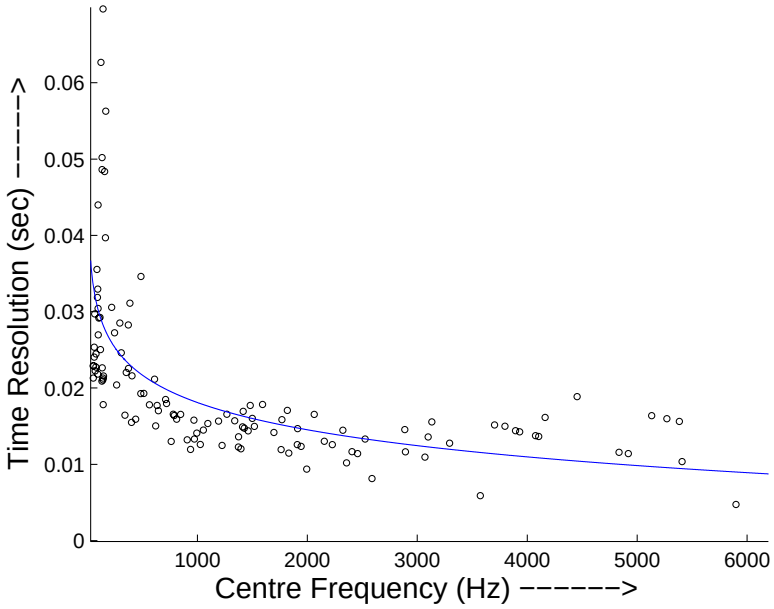

Figure B.3: The power law approximation fit the data in Figure B.2

B.0.4 Conclusion

The experiment for obtaining the time resolution of the human ear using the minimum limit of the perception of beats shows fairly interesting results. A better fit for the obtained experimental data may be proposed. The experimental data shows the possibility of proposing a new perceptual scale based on this property. More detailed perceptual experiments need to be performed in order to explore this phenomenon.

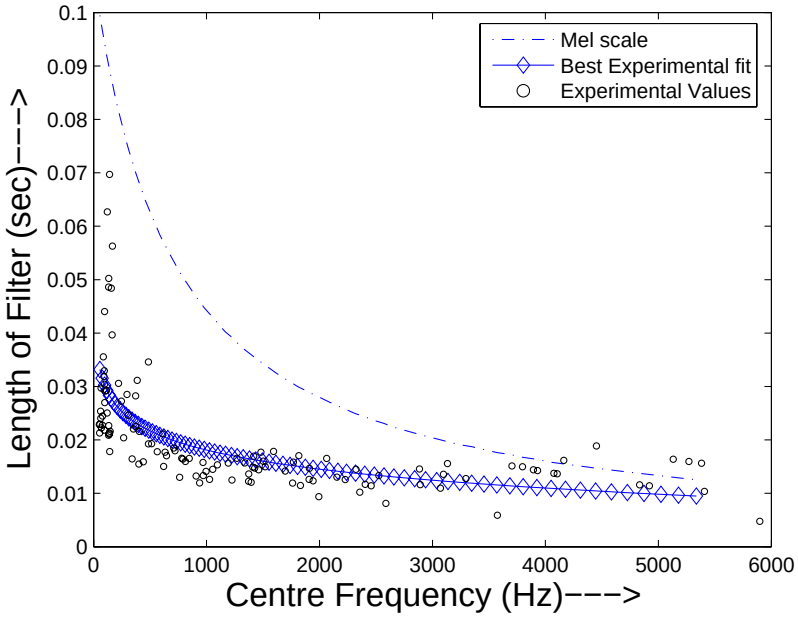

Figure B.4: Comparison between the length of the filter proposed for *Mel* scale formulation and the experimental data

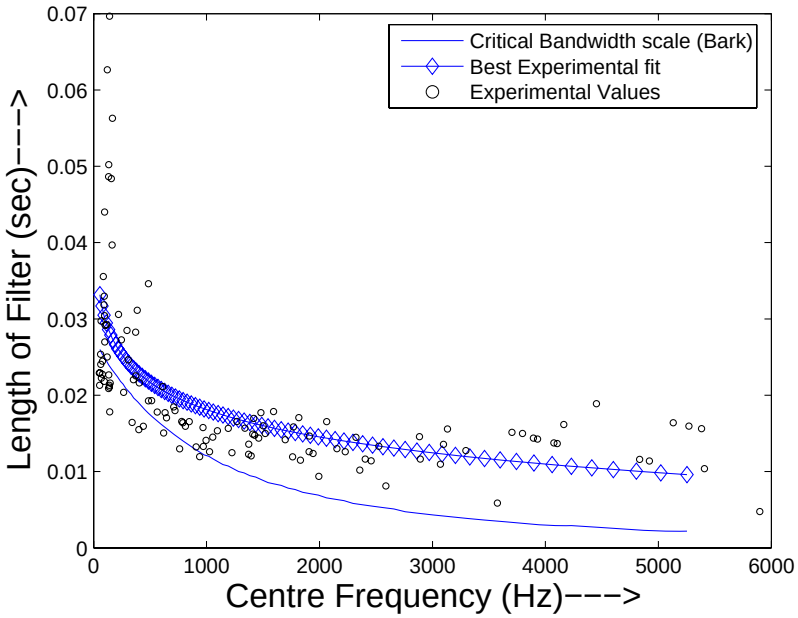

Figure B.5: Comparison between the length of the filter proposed for *Bark* scale formulation and the experimental data

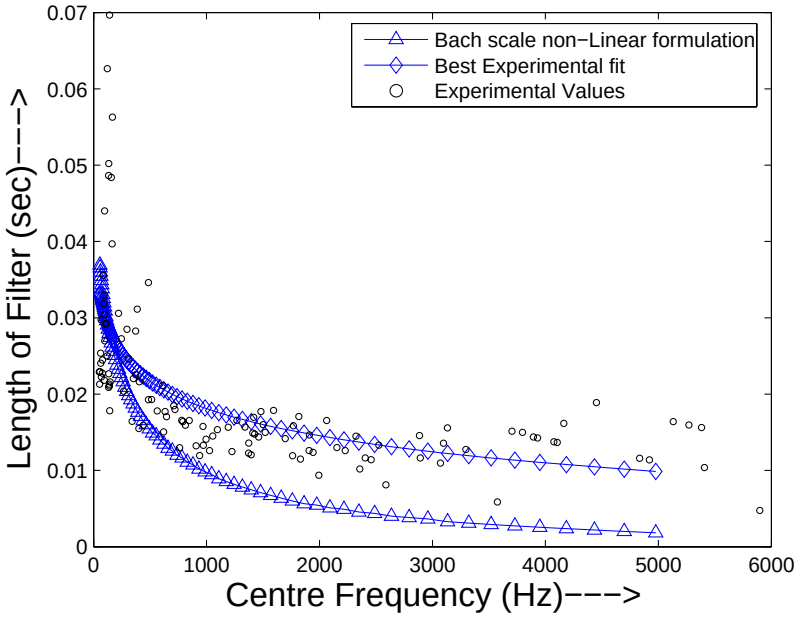

Figure B.6: Comparison between the length of the filter proposed for *Bach non-linear* scale formulation and the experimental data

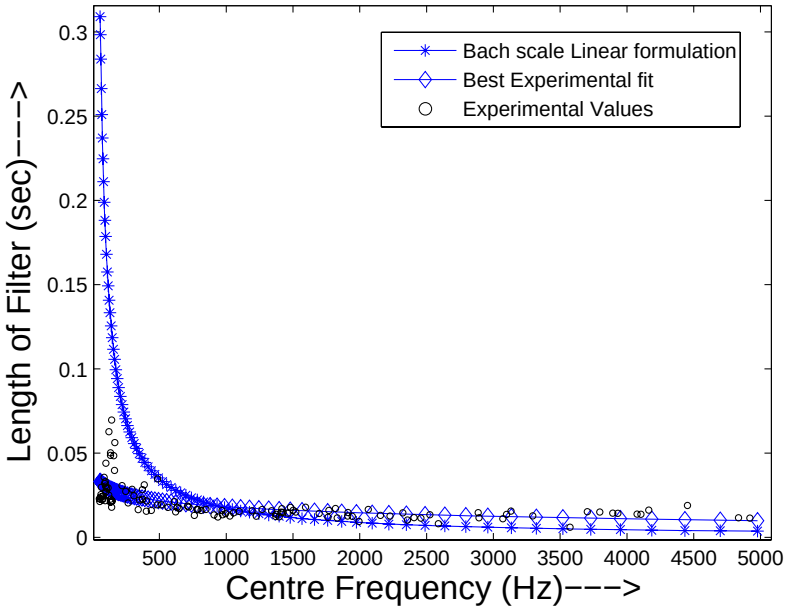

Figure B.7: Comparison between the length of the filter proposed for *Bach linear* scale formulation and the experimental data

References

[1] J. L. Flanagan. Voices of men and machines. *J. of the Acoustical Society of America*, 51(1):1375–1387, 1972.

[2] Alex Melville Bell. *Visible Speech: the Science of Universal Alphabetics*. Simpkin, Marshal and Co., London, 1867.

[3] D. Klatt. Review of text-to-speech conversion for english. *J. of the Acoustical Society of America*, 82(3), September 1987.

[4] L. Rabiner and B. H. Juang. *Fundamentals of Speech Recognition*. Pearson Education Press, 1993.

[5] F. S. Cooper. Spectrum analysis. *J.of the Acoustical Society of America*, 22:761–762, 1950.

[6] R. Smits. Accuracy of quasistationary analysis of highly dynamic speech signals. *J. Acoust. Soc. Amer.*, 96(6):3401–3415, 1994.

[7] James W. Pitton, Kuansan Wang, and Biing-Hwang Juang. Time-frequency analysis and auditory modeling for automatic recognition of speech. *Proc. of IEEE*, 84(9):1199–1215, 1996.

[8] O. Ghitza. and M. M. Sondhi. Hidden markov models with templates as nonstationary states: An application to speech recognition. *Computer Speech and Language*, 7(2):101–119, 1993.

[9] O. Kakusho. and M. Yanagida. Hierarchical ar model for time varying speech signals. *Proc. ICASSP*, pages 1295–1298, 1982.

[10] Y. Grenier. Time-dependent arma modeling of nonstationary signals. *IEEE Trans. Acoust. Speech Signal Process.*, 3(1):899–911, 1983.

[11] K. Nathan, Y. Lee, and H. Silverman. A time-varying analysis method for rapid transmitions in speech. *IEEE Trans. Signal Process.*, 39:815–824, 1991.

[12] G. Fant. *Acoustic Theory of Speech Production*. 2nd ed. Mouton, New York, 1970.

[13] S.S. Stevens, J. Volkmann, and E.B. Newman. A scale for the measurement of the psychological magnitude pitch. *J. Acoust. Soc. Amer.*, 8(3):185–190, 1937.

[14] S. Davis and P. Mermelstein. Comparison of parametric representation for monosyllable word recognition in continuously spoken sentences. *IEEE Trans. Acoust. Speech Signal Process*, ASSP-28:357–366, 1980.

[15] H. Fletcher and W. A. Munson. Relation between loudness and masking. *J. Acoust. Soc. Amer.*, 9(1):1–10, 1937.

[16] H. Hermansky. Perceptual linear predictive (plp) analysis for speech. *J. Acoust. Soc. Amer.*, 87(4):1738–1752, 1990.

[17] H. Fletcher and W. A. Munson. Loudness, its definition, measurement and calculation. *J. Acoust. Soc. Amer.*, 5(65):82–108, 1933.

[18] H. Hermansky and N. Morgan. Rasta processing of speech. *IEEE Trans. Speech Audio Process.*, 2(4):578–589, 1994.

[19] Gerald Abraham. In *The Concise Oxford History of Music*. Oxford University Press, 1979.

[20] Donald Jay Grout and Claude V. Palisca. In *A History of Western Music*. J. M. Dent and Sons Ltd, 1960.

[21] Owen H. Jorgensen. In *Tuning: Containing The Perfection of Eighteenth-Century Temperament, The Lost Art of Nineteeth-Century Temperament, and the Science of Equal Temperament, Complete with Instructions for Aural and Electronic Tuning*, East Lansing, 1991. Michigan State University Press.

[22] R. Rangaramanuja Ayyangar. *History of South Indian Music from Vedic Times to the Present*. Aryabhushan Press, Pune, 1972.

[23] Thomas F. Quatieri. *Discrete Time Speech Signal Processing*. Pearson Education Inc., New Delhi, India, 2004.

[24] L.L. Beranek. *Acoustic Measurements*. Wiley, New York, 1988.

[25] B.C.J. Moore and B.R. Glasberg. Suggested formulae for calculating auditory-filter bandwidths and excitation patterns. *J. Acoust. Soc. Am.*, 74(3):750–753, 1983.

[26] J. C. Brown. Calculation of a constant q spectral transform. *J. Acoust. Soc. Amer*, 89(9):425–434, 1991.

[27] Petre Stoica and Randolph L. Moses. *Introduction to Spectral Analysis*. Prentice Hall, New Jersey, 1993.

[28] S. Cox, R. Brady, and P. Jackson. Techniques for accurate automatic annotation of speech waveforms. *Proc. the International Conference on Spoken Language Processing*, 5:1947–1950, 1998.

[29] C. D. Mitchell, M. P. Harper, and L. H. Jamieson. Using explicit segmentation to improve 'hmm' phone recognition. *Proc. of ICASSP*, 1:229–232, 1995.

[30] D.R. Reddy. Segmentation of speech sounds. *J.Acoust.Soc.Am.*, 40(2):307–312, 1966.

[31] James R. Glass and Victor W. Zue. Multi level acoustic segmentation of continuous speech. *Proc. of ICASSP*, pages 429–432, 1988.

[32] Jan P. van Hemert. Automatic segmentation of speech. *IEEE Trans. on Signal Proc.*, 39(4):1008–1012, April, 1991.

[33] R. Andre-Obrecht. Automatic segmentation of continuous speech signals. *Proc. ICASSP-Tokyo*, pages 2275–2278, 1986.

[34] D.T. Toledano, L.A. Hernandez Gomez, and L.V. Grande. Automatic phonetic segmentation. *IEEE Trans. Speech and Audio Proc.*, 11(6):617–625, 2003.

[35] T. Svendsen and F.K. Soong. On the automatic segmentation of speech signals. *Proc. ICASSP-Dallas*, pages 77–80, 1987.

[36] Anindya Sarkar and T.V. Srinivas. Automatic speech segmentation using average level crossing rate information. *Proc. of ICASSP*, pages I397–I400, 2005.

[37] T. Nagarajan and Hema A. Murthy. Group delay based segmentation of spontaneous speech into syllable-like units. *EURASIP Journal of Applied Signal Processing*, 17:2614–2625, 2004.

[38] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley and Sons, New York, 1991.

[39] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley and Sons, New York, 2002.

[40] D. J. Hermes and J. C. von Gestel. The frequency scale of speech intonation. *J. Acoust. Soc. Amer*, 90.

[41] S. Furui and M. M. Sondhi. *Advances in Speech signal processing*. Marcel Dekker, Inc., New York, 1991.

[42] W. B. Kleijn and K. K. Paliwal. *Speech Coding and Synthesis*. Elsevier Science, New York, 1995.

[43] A. de Cheveigne and H. Kawahara. Yin, a fundamental frequency estimator for speech and music. *J. Acoust. Soc. Am.*, 111:1917–1930, 2002.

[44] Matti P. Ryynnen and Anssi P. Klapuri. Modelling of note events for singing transcription. *Proc. Workshop on Statistcal and Perceptual Audio Processing, SAPA*, 2004.

[45] L. P. Clarisse, J. P. Martens, M. Lesaffre, B. De Baets, H. De Meyer, and M. Leman. An auditory model based transcriber of singing sequences. *Proc. 3rd International Conference on Music Information Retrieval, ISMIR*, 2002.

[46] T. Viitaniemi, A. Klapuri, and A. Eronen. A probabilistic model for the transcription of single-voice melodies. *Proc. of the 2003 Finnish Signal Processing Symposium (FINSIG)*, pages 59–63, 2003.

[47] R. J. Ritsma. Frequencies dominant in the perception of the pitch of complex sounds. *J. Acoust. Soc. Am*, 42:191–198, 1967.

[48] X. Huang, A. Acero, and H.W. Hon. *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*. Prentice Hall, New Jersey, 2001.

[49] Paul Boersma and David Weenink. Praat: A system for doing phonetics by computer. *www.praat.org*, 2001.

[50] Ismail Shakra, Gustavo Frederico, and Abdulmotaleb El Saddik. Building a melody retrieval system. *Proc. IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems*, pages 83–87, 2004.

[51] T. Sorsa and J. Huopaniemi. Melodic resolution in music retrieval. *Proceedings of the Second Annual International Symposium on Music Information Retrieval, ISMIR*, pages 33–34, 2001.

[52] A. Ghias, J. Logan, D. Chamberlin, and B.C. Smith. Query by humming: musical information retrieval in an audio database. *Proc. Third International Conference on Multimedia, San Francisco*, pages 231–236, 1995.

[53] M. A. Raju and P. Rao. Building a melody retrieval system. *Proc. of National Conference on Communications, IITB*, 2002.

[54] R. J. McNab, L. A. Smith, I. H. Witten, C. L. Henderson, and S. J. Cunningham. Toward the digital music library: tune retrieval from acoustic input. *Proc. of Digital Libraries*, pages 11–18, 1996.

[55] C. Francu and C. G. Nevill-Manning. Distance metrics and indexing strategies for a digital library of popular music. *Proc. IEEE International Conference on Multimedia and Expo, ICME*, 2:889–892, 2000.