Analysis of Voice Signals for the Harmonics-to-Noise Crossover Frequency

LUCAS LEÓN OLLER

KTH Computer Science and Communication

Master of Science Thesis
Stockholm, Sweden 2008
Analysis of Voice Signals for the Harmonics-to-Noise Crossover Frequency

LUCAS LEÓN OLLER

Master’s Thesis in Music Acoustics (30 ECTS credits)
School of Electrical Engineering
Royal Institute of Technology year 2008
Supervisor at CSC was Sten Ternström
Examiner was Sten Ternström

TRITA-CSC-E 2008:103
ISRN-KTH/CSC/E--08/103--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden
URL: www.csc.kth.se
Analysis of Voice Signals for the Harmonics-to-Noise Crossover Frequency

Abstract

The harmonics-to-noise ratio (HNR) has been used to assess the behaviour of the vocal fold closure. The objective of this work was to find a particular harmonics-to-noise crossover frequency (HNF), where the harmonic components of the voice drop below the noise floor, and use it as an indicator of the vocal fold insufficiency. As the frequency range of interest grows, the different irregularities present in the voice make its computation more difficult. Different approaches to the problem were tested in order to derive a method that is robust to all such irregularities. The limitations of each approach are discussed.

Analys av brusinnehållet i röstsignaler vid höga frekvenser

Sammanfattning

Kvoten mellan deltoner och brus (HNR) har ibland använts för att kvantifiera graden av stämbandsslutning. Målet för föreliggande arbete var att för röstsignaler bestämma en brytfrekvens (HNF) där denna kvot blir lika med ett; som ett skalärt mått på stämbandsfunktion. Denna frekvens ligger ofta ovanför det frekvensområde som normalt har studerats i röster, vilket gör att standard-metoderna inte alltid är tillämpliga. Flera olika analysmetoder har prövats, och deras respektive begränsningar diskuteras.
Acknowledgments

I would like to thank both Kungliga Tekniska Högskolan (KTH) and Universidad Politècnica de Catalunya (UPC) for making this experience and work possible.

My supervisor, Sten Ternström, deserves special attention for his patience and guidance during all this months. Thank you. I am also in debt with the entire Department of Speech, Music and Hearing for making my time here not only useful in an academic point of view, but also a great experience.

I would also like to thank my whole family, Manuel, Teresa, Ana, Yago and Maria for all the kinds of support they have always given me.

As well, thanks to all the people that, being aware or not, have made this work possible: Virgilijus Uloza, for the database with disordered voices. The staff in charge of the The Mathworks MatlabCentral, for saving me a lot of time with really helpful functions. Mike Brookes from the Department of Electrical & Electronic Engineering at the Imperial College, for his wonderful Matlab toolbox for speech processing. And last but not least, Dan Ellis at the Columbia University, for all his incredibly helpful and well-commented Matlab routines.
## Contents

**Acknowledgments** ................................................................. i

**List of Figures** ................................................................. iv

1. **Introduction** ................................................................. 1

2. **Theoretical background in voice** ........................................ 3
   2.1. Physiology of Voice Production ........................................ 3
   Vocal Folds ................................................................. 3
   Vocal Tract ................................................................. 4
   Source-filter theory ....................................................... 5
   Voice perturbations ....................................................... 6
   Jitter and Shimmer ....................................................... 6
   Flutter ................................................................. 6
   2.2. High frequency content ............................................... 8
   2.3. HNR ................................................................. 8

3. **Database acquisition** ...................................................... 10
   3.1. Recording conditions .................................................. 10
   3.2. Used equipment ....................................................... 10
   3.3. Used database ....................................................... 11
   3.4. Unhealthy voice samples ............................................. 11

4. **Analysis** ........................................................................... 12
   Software ................................................................. 12
   4.1. Preliminary considerations ............................................ 12
   4.1.1. Analysis of the irregularities .................................... 12
   Analysis window. Trade between fundamental frequency variation and window length ................................................. 12
   Experiments on time/frequency balance .................................. 14
   Results and plots ....................................................... 14
   Conclusions ................................................................. 15
4.2. Analysis tools ................................................. 17
    Cepstral analysis ........................................... 17
    Fundamental period extraction algorithms ................. 18
        Autocorrelation ......................................... 18
        Period separation algorithm ......................... 19
    Peak detection algorithms ............................... 20
        "Rubber band" Technique .................................. 20
4.3. Approaches and results .................................. 21
    4.3.1. Cepstrum-based signal decomposition ............... 22
        Procedure .................................................. 22
        Results and conclusions ............................... 23
    4.3.2. Noise floor estimation: consecutive period subtraction .... 24
    4.3.3. Cepstrum-based HNF detection algorithm .......... 25
        Results and conclusions ............................... 26
    4.3.4. Period normalization ............................... 27
        Results and conclusions ............................... 27
5. Conclusions ............................................... 31
6. Further work ............................................... 32
    FDA algorithm + shimmer reduction ....................... 32
    Research on the frequency structure of the noise ....... 32
    Algorithms robust to severe voice disorders? ............ 32
Bibliography ................................................. 33
Bibliography ................................................. 33
A. Matlab Code ............................................... 34
    A.1. Autocorrelation-based T0 extraction ................. 34
    A.2. "Rubber band" peak detection algorithm .............. 36
        Rubber band .............................................. 36
        Minmax .................................................... 37
        Deltatrain ............................................... 38
    A.3. Period separation and normalization ................... 39
    A.4. Cepstrum-based HNF detection ......................... 42
    A.5. Harmonic envelope and noise floor extraction ........ 44
List of Figures

2.1. Location of the vocal folds and function diagram. Image extracted from [2] – used with permission. ................................. 3
2.2. Vocal tract and nasal tract. Image extracted from [12] ............... 4
2.4. Jitter and shimmer. Image extracted from [17] – used with permission. 6
2.5. Evolution of $F_0$ during 1 second. ($F_0 = 124$ Hz) ...................... 7
2.6. Spectrum of speech flutter. The peak around $F_0/2$ is due to the jitter. ($F_0 = 153$ Hz). [15] – used with permission. ...................... 7
2.7. Spectrum of a voiced sound: linear and logarithmic scales .......... 9

4.1. Magnitude spectrum of a generic window .................................. 13
4.2. Real speech signal (red) and synthesized signal (blue) for a window length of $10 \cdot T_0$ .......................................................... 15
4.3. Sweep of deterministic timing patterns (left) and of deterministic amplitude patterns (right). See text below. Image extracted from [8] ... 16
4.4. Power-spectral density of an impulse train with random jitter. ......... 17
4.5. Real cepstrum of a vowel ....................................................... 18
4.6. Autocorrelation-based $T_0$ extraction ........................................ 19
4.7. $T_0$ extraction ................................................................. 20
4.8. Example of the application of the rubber band algorithm .............. 21
4.9. Spectrum of a windowed segment of a vowel sample (red) and noise floor+harmonic estimation (blue) ................................. 23
4.10. Harmonic component reconstruction, comparison of window lengths. Blue 5$T_0$, Green 4$T_0$. ......................................................... 24
4.11. Averaged spectra (black), noise floor estimation before improvement (red), noise floor estimation after improvement (green), single spectrum (yellow) ......................................................... 25
4.12. Period subtraction estimation (blue) and cepstrum-based estimation (red) ................................................................. 26
4.13. Energy of the anti-transformed first rahmonic for the averaged spectra (green) and for a single spectrum (blue). See related Figure below. .... 26
4.14. Analyzed spectra. In the upper spectrum the red part indicates the point where the slope change in the rahmonic energy occurs. The lower spectrum is the result of averaging all the spectra of the windowed signal segments. There is no clear slope change in this case. 29

4.15. Normalized (green) vs unnormalized spectrum 30

4.16. Noise floor and harmonic envelope 30
1. Introduction

So far, little research has been done on the high frequency region of the human voice. Still, it is known to be important from the point of view of perception, so it definitely conveys information. Most of the speech research has been carried by the telephony industry, and has always stayed below 5 kHz due to their requirements; intelligibility has been their main target, and it is not seriously affected by filtering out the highest two octaves of the voice spectrum. There are several fields lacking further research on this topic, like speech synthesis, medical voice diagnosis, music production or new-era digital telephony. This thesis is part of a project whose goal is to increase our knowledge of the acoustical fine structure of the highest octaves of the spectrum of human vocal sounds.

The particular aim of this work is to explore the whole voice spectrum in order to derive a method to measure the harmonics-to-noise crossover frequency (HNF) of a given voice. This parameter defines the frequency at which the harmonic content of voiced sounds drops below the noise component, and it is closely related to the precision of the vocal folds behaviour. The success in finding this frequency would give voice clinicians a non-invasive method for estimating the efficacy of the glottal closure, which is an indicator of the health of the voice production mechanism.

A common measurement for this kind of assessment is the harmonics-to-noise ratio (HNR). It provides an indication of the overall periodicity of the voice signal by quantifying the ratio between the periodic (harmonic part) and aperiodic (noise) components. Unfortunately, this parameter is usually measured as an overall characteristic of the signal, and not as a function of frequency, so it cannot be used to find the HNF. In recent years, several research articles have been published on the measurement of this parameter, but none of the existing methods meets the requisites needed for the calculation of the crossover frequency: the frequency-based methods stay below 5 kHz, and the time-based methods do not provide the frequency distribution of the HNR and are therefore not suitable for our purpose. This work tries to extend the usage of the conventional methods to the whole spectrum in order to find the HNF, which is supposed to reside between 5 and 20 kHz.

An important reason to study this particular frequency is to prove if it is independent of the vocal tract configuration, unlike the HNR. The overall value of the HNR of the signal varies because different vocal tract configurations involve different amplitudes for the harmonics, as it will be explained in section 2.1. This independence would make the HNF a useful non-invasive method for estimating the efficacy of the glottal closure, since the signal could be recorded outside the mouth. As well, this parameter uses a very simple metric that requires only little knowledge about signal processing for voice clinicians to understand.

Due to the lack of true periodicity of the voice this task has proved impossible using conventional methods as they are described in the different articles. In this work, sensible adaptations of the original methods have been made in order to fulfill our task. The limitations and possibilities of all of them are discussed. According to the results, the main limitation of the different methods resides in the small cycle-to-cycle perturbations that cause the aperiodicity...
of the voice. These irregularities affect the main frequency analysis tool – the Fourier transform – in a way that make it impossible to distinguish harmonics from noise. Different ways around this problem are also discussed.

As a consequence of the pursuit of HNF, several efforts have been made to separate the periodic (deterministic) from the aperiodic (stochastic) part of the signal. Some of the results could be of great interest in other applications, such as instrument modeling. Still, as this is not the main goal of this work, the results are not discussed.
2. Theoretical background in voice

This part contains only a brief explanation of the fundamental parts of the voice production mechanism in order to be able to understand the rest of the work. For further detail of all the components that form this mechanism see the next references [13, 17]. For a more singing-related point of view see [14].

2.1. Physiology of Voice Production

Vocal Folds

The vocal folds are located in the larynx and they are an essential part of the human speech mechanism. During speech, the vibration of the vocal folds converts the air flow from the lungs into a sequence of short flow pulses. This pulse train, the glottal flow, provides the excitation signal of voiced speech sounds.

To oscillate, the vocal folds are brought near enough together such that air pressure builds up beneath the larynx. The folds are pushed apart by this increased subglottal pressure, with the inferior part of each fold leading the superior part (see Figure 2.1). The rush of air vibrates the folds and generates...
a buzz-like sound that is shaped by the resonating cavities – the mouth, nose and sinuses – as will be explained further on in this section.

The vocal fold vibration cannot be directly measured with non-invasive methods because the larynx is not accessible during phonation. However, there are several indirect means of studying the function of the vocal folds. One of the methods is called inverse filtering, which is used to estimate the pulsating air flow produced by the vibrating vocal folds mathematically. This method is not reliable up to 20 kHz. At a minimum, only the speech signal captured with a microphone is required for the procedure. Electroglottography is another voice analysis method. It measures conductance variations across the neck of the subject during speech. This yields information about vocal fold vibration because the varying amount of contact between the vocal folds causes conductance fluctuation. Inverse filtering and electroglottography are non-invasive methods: they do not require surgery or internal examination of the body, nor do they prevent the subject from speaking normally.

Vocal Tract

The vocal tract can be defined as the airway used in the production of speech, especially the passage above the larynx (see Figure 2.1), including the pharynx, mouth, and, in a broad definition, the nasal cavities.

The tract has several resonances – i.e., the air in the tract vibrates more readily at some frequencies than others. You can vary these resonances by moving tongue and lips, and this variation has a lot to do with the different speech sounds produced, as it will be seen in the section below.
Source-filter theory

The acoustic properties of vowels have traditionally been described on the basis of the source-filter theory [4]. In this theory the vocal folds are the source and the vocal tract is the filter (see Figure 2.3): the vibration of the vocal folds produces a varying air flow which may be treated as a periodic source (A). This source signal is input to the vocal tract (B). The tract behaves like a time-varying filter in that its response is different for different frequencies. It is variable because, by changing the position of your tongue, jaw etc you can change that frequency response. The position of the maxima (enhanced frequencies) of this filter define the formants. The input signal and the vocal tract, together with the radiation properties of the mouth, face and external field, produce a sound output (C). Vowels are perceived and classified on the basis of the two lowest formant frequencies of the vocal tract.

![Source-filter theory](image)

The Figure above is very helpful to understand the principles of the source-filter theory, but there is an important part missing. The vocal folds are not only the source of the periodic component, but also the main source of aerodynamic turbulent noise in vocalic sounds. This noise is also shaped by the vocal tract filter, and it is therefore also present in the output signal. For modal glottal configuration, i.e., utterance of vowels, there is only airflow during a part of the cycle, at least for healthy vocal folds. Of course, this type of noise is more present in disordered voices where, for instance, there is an obstacle that does not let the vocal folds close completely. This type of noise can be expected to increase towards high frequencies.

The decay of the harmonic envelope of the spectrum of healthy vocal folds should be around -12 dB/decade. The decay rate is especially dependent on the closure efficacy of the vocal folds (frames 7 and 8 in Figure 2.1). The relative energy of the harmonics decreases more steeply when the closure is imprecise or incomplete [4].

Both the harmonic decay rate and the turbulent noise are directly related to the vocal folds behaviour, and therefore the efficacy of their closure. Both the HNR and the HNF are completely defined by these two factors. The steeper the harmonics decrease the lower the HNR and HNF, as for increasing turbulent noise.
Voice perturbations

Voiced speech signals exhibit repeating deterministic pattern as well as random perturbations. For the deterministic aspect, some asymmetry in the vocal folds can cause them to meet the same way only every two or three cycles, causing the periodicity to be achieved every second or third cycle, respectively. That would result in the appearance of a subharmonic frequency. The stochastic component consists of random variations in timing and amplitude of the signal. Small perturbations are called jitter and shimmer, but there are as well larger perturbations or trends that can be observed during the utterance of sustained vowels, flutter. These random variations in frequency and amplitude make the speech signal non-periodic. Despite the non-periodicity, the term “quasi-periodic” was coined to describe the voiced speech sounds, as the the deviations from periodicity are small. Still, they are big enough to make its analysis a tough task, as will be explained in section 4.1.1.

Jitter and Shimmer

In speech and voice science jitter and shimmer are used to describe the short-term – cycle-to-cycle – variations in fundamental frequency and amplitude, respectively (see Figure 2.4). Again, no mathematical definition of these concepts has been universally accepted, probably due to the freedom to establish a criteria. Nevertheless, they are very useful to describe the fluctuations of the voice signal in a qualitative way. In a signal synthesis context, jitter and shimmer are given as a percentage that represents the maximum deviation from a nominal frequency or amplitude. In real speech signals this is not so easy, since there is no clear nominal frequency or amplitude.

![Image of Jitter and Shimmer](image.png)

Figure 2.4. Jitter and shimmer. Image extracted from [17] – used with permission

Flutter

Flutter is characterized by the superimposition of frequency and amplitude oscillations on the voice signal. It causes periodic fluctuations in the fundamental frequency ($F_0$) and intensity in the singing voice. These pulsations typically occur at a rate of 3 to 7 Hz and with a $F_0$ modulation extent of ±1 semitone (100
cents or approx. 6%). This perturbations are observable in a much larger time scale than jitter, but they have to be taken into account for the analysis. They are fast enough to not be perceived as pitch changes, but they are necessary for the voice to sound natural. In Figure 2.5 the evolution of $F_0$ in time can be seen. Jitter is cannot be appreciated because it happens in a much shorter scale.

![FUNDAMENTAL FREQUENCY VARIATION](image1)

Figure 2.5. Evolution of $F_0$ during 1 second. ($F_0 = 124$ Hz)

In Figure 2.6, from [16], the spectrum of the flutter signal can be observed. This type of perturbation can be simulated filtering white noise with a second-order resonant low-pass filter. The result is the Figure above.

![Flutter spectrum](image2)

Figure 2.6. Spectrum of speech flutter. The peak around $F_0/2$ is due to the jitter. ($F_0 = 153$ Hz). [15] – used with permission.
2.2. High frequency content

There is very few publications on the content of the voice spectrum above the seventh octave (see Table 2.1, extracted from [15]). The frequency ranges used for this definition of octaves are not exactly as the ones used in music, but it is useful for us to define them that way, since they represent the human audible range (20-20000 Hz).

<table>
<thead>
<tr>
<th>Octave</th>
<th>Frequencies</th>
<th>Vocal significance (general)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20-40 Hz</td>
<td>- not vocal -</td>
</tr>
<tr>
<td>1</td>
<td>40-80 Hz</td>
<td>-not vocal -</td>
</tr>
<tr>
<td>2</td>
<td>80-160 Hz</td>
<td>male fundamental $F_0$</td>
</tr>
<tr>
<td>3</td>
<td>160-320 Hz</td>
<td>female fundamental $F_0$</td>
</tr>
<tr>
<td>4</td>
<td>320-640 Hz</td>
<td>first formant $F_1$</td>
</tr>
<tr>
<td>5</td>
<td>640-1250 Hz</td>
<td>$F_1$-$F_2$</td>
</tr>
<tr>
<td>6</td>
<td>1.25-2.5 kHz</td>
<td>$F_2$-$F_3$</td>
</tr>
<tr>
<td>7</td>
<td>2.5-5 kHz</td>
<td>$F_3$-$F_5$</td>
</tr>
<tr>
<td>8</td>
<td>5-10 kHz</td>
<td>distinct modes; audible to most</td>
</tr>
<tr>
<td>9</td>
<td>10-20 kHz</td>
<td>lumped modes; audible to some</td>
</tr>
</tbody>
</table>

Table 2.1. Octave bands of the voice spectrum. [15] – used with permission.

The voice can show harmonic energy up to 20 kHz, but the last two octaves of the voice spectrum contain only a very small fraction of the total energy. Above 5 kHz the signal power can be around 50 dB below the dominant peak. In Figure 2.7 it can be appreciated how narrow the two last octaves are in the logarithmic scale. This scale represents more accurately how the human hearing works. These are the two main reasons why the telephony industry has focused its research below that frequency. The intelligibility of the signal is not seriously affected when filtering out the last two octaves.

For more detail in the description of the last two octaves of the spectrum, see [15].

2.3. HNR

The definition of the harmonics-to-noise ratio has only been established qualitatively. There is no mathematical definition of this parameter that has been universally accepted, and this has been a double-edged sword. On one hand, this has allowed to establish a criteria of our own for its measurement. On the other hand, this lack of definition has led to a proliferation of different methods to find this parameter, each of which gives different HNR values for the same signals. This makes it impossible to compare different computations of the HNR. When two measures of the same method are observed, the results can be compared, but it is not possible to compare results between different methods.

Most of the articles this work has been based on are closely related to the medical field, since the HNR has been used to assess vocal fold behaviour. So far, all the voice diagnosis tools that compute this parameter using a frequency approach
have stayed below 5 kHz. These tools have been able to show a correlation between the HNR and vocal fold disorders. Still, as it has been mentioned before, the HNF is more closely related to the efficacy of the vocal fold closure due to its independence from the vocal tract configuration [3]. Moreover, its simple metric would be a very simple for voice clinicians to understand, without the need of deep knowledge in signal processing.

The aim of this work is not only to compute the overall value of the HNR, but to do it in a way that allows us to compute it as a function of frequency in order to calculate the harmonics-to-noise crossover frequency. Not all the existing methods to calculate the HNR give that flexibility, and thus they have not been thoroughly tested. The methods to calculate this parameter can be classified in two different types: frequency-based and time-based. Some authors consider the cepstral-based methods a different type [10], but in the end they require a frequency-based estimation, so that third type would not be not technically different. See [10] for a wide number of references of the different methods.
3. Database acquisition

There are several databases available to researchers, but none of them meet the requisites necessary to carry out this work. Basically there are only few that where recorded with a sampling rate high enough to work with the frequency content of the last two octaves. In the end, a database was collected by Sten Ternström in collaboration with the University of York. See [15] for more information about the recordings.

3.1. Recording conditions

There are basically three requisites that our high fidelity recordings needed to meet:

1. Anechoic room: the recordings had to be done in an anechoic room in order to prevent the room acoustics to interfere with the features of the raw voice signals as it comes from the mouth.
2. Sampling rate above 40 kHz: in order to analyze the spectrum up to 20 kHz at least twice the sampling frequency is needed. To follow standards that almost any sound card could work with the final choice was 44.100 Hz.
3. Dynamic Range: the recordings were made using 16-bit resolution. The low power of the two last octaves requires the maximum dynamic range available with 16 bits (ideally 96 dB). Low-noise amplifiers were used.

3.2. Used equipment

— Neumann KM140 (Cardioid)
— Line Audio CM3 (Cardioid)
— DPA 4066C (Omnidirectional)
— External sound card RME Fireface 400; MOTU Traveler.
3.3. Used database

From all the acquired database, the samples of 5 sustained vowels were used. The samples are of at least 5 seconds each, although only one second samples were used throughout this work. It is important to mention that the 8 subjects (4 males and 4 females) had singing experience. As a first approach to the search for the HNF it was more logical to use non-disordered voices.

3.4. Unhealthy voice samples

A set of 10 examples of sustained vowel [a] were used to test the algorithms (provided by V. Uloza). The files contained the diagnosis of the laryngeal diseases, as well as the pictures of patients’ larynges. The dynamic range of this recordings was not good enough to appreciate the any frequency content above 14 kHz, but the recordings helped to test the algorithms. Some of the voices had severe disorders that crashed some of the algorithms, although there was only little time for the testing.
4. Analysis

Software

All digital signal processing, programming, and data visualization has done using Matlab (The MathWorks) and Soundswell’s tools. Matlab versions 7.3 (R2006b), and 7.4 (R2007a) were used. Custom scripts, functions, and user interfaces were developed for the purposes of the work. Soundswell version 4.0 was used exclusively for data visualization.

4.1. Preliminary considerations

Before starting to analyze the voice signal it is necessary to be aware of the voice characteristics that can interfere with our analysis tools. Small fluctuations in frequency, amplitude, and waveshape are always present, even in fully healthy voices, due to the internal body “noises”. The larynx is sensitive to the fluctuations of all surrounding body mechanisms, such as the respiratory, lymphatic, vascular and other transport systems [17]. In this work only segments of voiced speech – sustained vowels – are analyzed, and therefore only the irregularities that belong to this type of speech are mentioned here.

4.1.1. Analysis of the irregularities

Understanding how the different irregularities of the voice affect our analysis tools is essential. The main tool that will be used throughout the whole study is the discrete version of the Fourier Transform (DFT), since most of the tested methods are carried out at some point in the frequency domain. The VHF content of the signal is of most importance to our objective, and it is also the part of the signal that is most affected by these perturbations.

Analysis window. Trade between fundamental frequency variation and window length

Frequency analysis is subject to the Heisenberg uncertainty principle: the analysis window determines the trade-off between frequency resolution and time localization. The choice of the window type is based on three of its spectral characteristics (see Figure 4.1):
1. Main lobe width (bins or Hz): this parameter depends on the window shape, its length and the number of FFT points.
2. Main-to-sidelobe level (dB): this parameter depends exclusively on the window shape, and it is independent of the length of the window.
3. Side lobe roll-off rate (dB/octave): this parameter describes the decay rate of the side lobes with frequency.

Figure 4.1. Magnitude spectrum of a generic window

The ideal case would be to have a narrow main lobe and a very low sidelobe level to reduce leakage between DFT channels or bins. The Hanning window has been chosen and used throughout this work. It has a rather wide main lobe, but in compensation the main to sidelobe level is large (around 31.5 dB), and the decay of the lobes is very fast (60 dB/decade). The choice of this window has proved to be good enough for our purposes.

To make a correct choice of the window length it is necessary to have an idea of the speed of the voice perturbations, because that is what will set the limits, specially in terms of frequency deviations. If we take the evolution of the fundamental frequency across time \( F_0(t) \), we are interested in \( \left| \frac{dF_0}{dt} \right|_{\text{max}} \), because the faster the variations, the shorter the window will have to be, in order not to lose resolution in the highest octaves. Deviations in frequency affect the output of the DFT in a more limiting way, as discussed in [8]. This sort of deviations cause the high-frequency harmonics to blur until they are hardly or not distinguishable from noise at all.

There are mainly two factors pushing in different directions: on one hand, the amount of blurring is proportional to the length of the analysis window, so that leads to the choice of the shortest possible window. On the other hand, as the window length decreases the width of the harmonic increases, losing frequency resolution. To be able to estimate the noise floor it must be made sure that the power between the harmonics is due to the noise, and not the window residuals.
In order to find the best balance between time/frequency resolution a series of experiments have been carried out.

**Experiments on time/frequency balance**

In order to assess this balance properly a number of test signals should be synthesized, using different parameters for frequency perturbations. Afterward, real speech signals (with its different intrinsic parameters) could be tested, to confirm the extracted conclusions. Instead, and due to time limitations, a more direct approach has been used.

First, a set of representative voice samples is needed. Representative means that they should not be trained singer voices, or totally unhealthy ones. The procedure the following:

1. Calculate a first rough estimation of the fundamental period $T_0$ and its fluctuations, using an autocorrelation-based method (see A.1). The fundamental period length is used as a reference. Since it is the cycle-to-cycle perturbations that are causing the blurring of the signal, it is a sensible approach to measure the window length in terms of number of periods.
2. Compute the spectrogram of the vowel sample for each period length $L_w = k \cdot T_0$, where $k$ is an integer number between two and twenty.
3. Compute the spectrogram of a synthesized vowel with the same pitch, with steady harmonics. The same window lengths are used.
4. The results are plotted, comparing the results of the real and the synthesized signal two-by-two for every different window length. This way, the evolution of how the voice perturbations are affecting the spectrum can be appreciated.

**Results and plots**

In order to get a signal that could be representative, the results of two samples have been chosen. The first one is a low-pitched /u/ vowel, like in zoo, uttered by a man. The reason to pick a low-pitched vowel is because the harmonics are closer together, and it is therefore more difficult to obtain a good resolution. The second sample belongs to the database mentioned in section 3.3. It is a female, and the vowel is an /e/, like in men. This sample has been randomly chosen among all the available samples.

Even with extremely short windows, the high part of the harmonics begins to blur. With a window length around $4T_0$, the harmonics are clear enough to find their position in frequency, but the noise floor is totally masked by the windowing effect: the mainlobe centered in every harmonic collides with the surrounding harmonics. At that point, only in some signals it is possible to see harmonics at the end of the spectrum, and that kind of voices are not common. Even in those voices, the position of the harmonics that can be seen in the last two octaves is affected by nearby subharmonics, as will discussed in the experiment conclusions section.

In Figure 4.2 an example of the vowel /e/ can be seen, compared with a synthesized signal of the same pitch. In this case, with a window length of $10^4T_0$, it

---

1. The spectrogram function in Matlab gives as an output the PSDs (Power Spectral Densities) of the windowed segments of the signal.
is possible to see the noise floor between the lowest harmonics, but it is already impossible to distinguish the harmonics above 10 kHz. A visual criteria has been used to decide if the region between harmonics was due to the real noise floor or due to the windowing effect. In the synthesized signal it is obvious that the harmonics collide because of the windowing effect, since there is no noise. If it can be assured that the region between harmonics in the real signal is 10 dB above the window collision in the synthesized one, it can decided that the noise floor has been reached.

Figure 4.2. Real speech signal (red) and synthesized signal (blue) for a window length of $10 \cdot T_0$

Conclusions

The main conclusion of this experiment is that it is not possible to have a window length that allows us to reach the noise floor and at the same time have enough resolution in the two last octaves of the spectrum. Only in some voices the harmonics are steady enough to use a window length above four times the fundamental period. If this work needs to be of some use in clinical voice diagnosis at best signals with perturbations similar to these ones can be expected.

Normally, the windowing phenomenon is used to reduce random noise, because it involves averaging. But in this case, this same averaging effect masks completely the harmonics in the last octaves. If we visualize a real-time line spectra, the harmonics swing in time. The problem is that a small variation in the fundamental frequency of the signal would be amplified by 100 in the $100-th$ harmonic, so the harmonics at the end of the spectrum swing faster. The result is that the averaging effect masks them so that they are not distinguishable from noise.

The noise floor in the spectrogram, even with a long window, is not only formed by random noise. There are lots of subharmonics in the signal, due to the frequency and amplitude perturbations in the noise, that are as well averaged by the window length. To this respect, a recent article published by N. Malyska
and T. F Quatieri [8] analyzes how the amplitude and frequency perturbations separately affect the harmonic spectrum. A couple of examples of deviations in the fundamental period and amplitude are shown in Figure 4.3. The effects of the period deviations will become clear in the results of one of the peak detection algorithms under study (see section ).

Figure 4.3. Sweep of deterministic timing patterns (left) and of deterministic amplitude patterns (right). See text below. Image extracted from [8]

This series of experiments use a synthesized periodic impulse train where different values of jitter and shimmer are used. In the case of jitter, every second impulse is shifted by 0.5 ms in 0.1-ms increments (Y₁ to Y₅). The filled triangles indicate the heights of the harmonic components, unfilled triangles indicate the heights of the subharmonic components, and the grey contours shape the line spectra. In the case of shimmer, every second impulse is scaled from 0.9 to 0.2 in increments of 0.2. It can be appreciated that the jitter phenomenon has worse results in the spectrum. A real case of the combination of these two phenomena can be seen in Figure 4.8.

Another interesting conclusion in this article is their prediction of the shape of the power-spectral density of an impulse train with random frequency perturbations exclusively (see Figure 4.4). It is hard for me to know if all the assumptions they use to get this result are correct, but their conclusion fits well with the results of this experiment. A deeper study would be necessary, but it is worth mentioning. In summary, a signal of this type would have mainly three components in the spectrum (other kinds of noise are ignored here): a frequency-flat contribution, a series of low-pass filtered harmonics and a noise floor with high-pass characteristics.

If we observe the power-spectrum densities of the signals analyzed in the experiment, the noise floor between the low-frequency harmonics requires a long
window to be reached. The explanation given in the article does not have to be the main reason of this behaviour, but it definitely contributes. There are other reasons: the subharmonics do not affect the lower harmonics so much, because they contain much more energy. It is also important to mention that the noise generated at the vocal folds is also expected to be weak at low-frequency regions, as it is mentioned in section 2.1.

Another consequence of the length of the analysis window is that the amplitude of the harmonics grows as the window length grows. As well, the harmonics narrow, because of the windowing effects mentioned above. The problem is that if the computation of the HNR is frequency-based, the height of the harmonics is extremely important. If the HNR is going to be used to compare different signals analyzed with different window lengths, a correction factor for the window length must be found. So far, the only proposed solution to this problem is to normalize the period length of all the analyzed signals to the same length, to be able to use the same window length for all of them. This solution has not been tried extensively.

The overall conclusion of this experiment is that there are two possibilities to find the HNF of a voice signal:

1. Pre-process the signal to get rid of the cycle-to-cycle perturbation in order to be able to use the Fourier transform.
2. A type of analysis that does not require signal windowing.

4.2. Analysis tools

Cepstral analysis

Apart from the Fourier Transform, there is basically another tool that is commonly used in speech analysis and processing: the cepstrum. The cepstrum is defined as the inverse Fourier transform of the logarithm of the magnitude spectrum:

$$ \int_{-\infty}^{\infty} \log |S(f)| e^{j2\pi f t} df $$
In the cepstral domain, the independent variable is measured in seconds, and it is referred as quefrency. Figure 4.5 shows the real cepstrum for a vowel sample. The quefrency domain can be split in three different regions. In the lowest part (0-0.002 s, red in the figure) there is a broader peak corresponding to the slow variations caused by the formant structure, i.e., the vocal tract characteristics. The rest of the quefrency region can be attributed to the excitation part. The first cepstral peak (called rahmonic in the quefrency domain) defines the second region, marked in green in the figure. This rahmonic appears around the period length in quefrency, and it can be attributed to the harmonic or periodic part. The rest of the cepstral coefficients can belong to the random part of the signal. As it can be appreciated in the figure, small rahmonics can appear at twice and three times the period length, but the energy contribution due to these peaks is insignificant in comparison with the first rahmonic [1].

![Figure 4.5. Real cepstrum of a vowel](image)

**Fundamental period extraction algorithms**

**Autocorrelation**

This method is not the most accurate, but it gives a good estimate of the fundamental period. It is used to choose the window length in the first place, and as the first rough estimate required by the rubber band algorithm (see next section). The autocorrelation is calculated for every windowed segment of the signal in the time domain. The autocorrelation maximum is equivalent to the fundamental period (in samples). Another algorithm is used to improve this estimation [5], getting a values that improve the precision below the sample level.

The evolution of the fundamental period is calculated for every windowed segment. The same windowed segments that will be used in the spectrogram. This estimation will be used in order to have a good estimate of the position of the
first harmonic in every spectrum. The problem is that it is not possible to detect cycle-to-cycle variations.

![Fundamental period variation](image)

**Figure 4.6. Autocorrelation-based $T_0$ extraction**

### Period separation algorithm

The aim of this method is to separate every period individually. The algorithm is based in a method for high-precision fundamental frequency extraction [18]. The previous algorithm is used to get a first estimation of the fundamental frequency, and then the signal is low-pass filtered with a cut-off frequency of 1.5-$F_0$. The resulting signal is very close to a sinusoidal signal, and the zero-crossings are directly related to its frequency. The position of the negative-to-positive zero crossings is calculated ($I_N(i)$ is the position of the i-th zero crossing), to have an approximation of the length of every cycle. After that, the algorithm calculates the length of every period following the following steps:

Assuming $P(1)$, the position of the start of the first period, the minimum between $I_N(1)$ and $I_N(2)$. $\text{sig}(n)$ is the discrete signal.

For $i = 2$ to $Nc + 1$

1. Initial guess of the period length: $P(i) = I_N(i) + (P(i - 1) - I_N(i - 1))$

2. Set upper and lower limits to search for the final period: (PERC is a value between 5-15% to set the size of the search range)

$$J_j = P(i) - \text{PERC}(P(i) - P(i - 1))$$

3. Find $j$ that minimizes ($J_{\text{min}}$):

$$\text{ERR}(j) = \frac{1}{j - P(i - 1)} \sum_{k=P(j-1)}^{P(i-1)} (\text{sig}(k + \lfloor j - P(i - 1) \rfloor) - \text{sig}(k))^2$$

19
4. $P(i) = J_{\text{min}}$

The precision of this algorithm can be improved further by using fractional delay filters. Still, the results are good, as will be proved in section 4.3.2.

**Peak detection algorithms**

These algorithms are a basic tool to measure the HNR, and due to the blurring effect at high frequencies this irregularities of the voice the task is not simple. Almost every method requires at some point the use of a peak detection algorithm. After testing some peak detection algorithm found in different Matlab databases, it was decided to write the code from scratch.

"Rubber band" Technique

The spectrogram is seldom well-defined above 10 kHz, but even if the harmonics are not visible, they are supposed to be, because of their relationship with the fundamental frequency. The idea of this technique is rather simple. Once the fundamental frequency ($F_0$), has been estimated, the location of the $i$-th harmonic should be $f_i = iF_0$. The analogy with a rubber-band becomes clear when we think how the small variations in the fundamental frequency affect the locations of the harmonics. If the variation in $F_0$ is $\Delta F_0$ then the variation in the $i$-th harmonic will be $i\Delta F_0$, similarly to how equally spaced dots drawn on a rubber band behave when it is stretched. If the value of $F_0$ at every instant was exactly known, the problem of peak detection would be solved. The problem lies in the accuracy of the estimation of $F_0$. 

![Fundamental period variation](image)
The algorithm needs a rough estimation of the fundamental frequency to start working, and it is calculated using the autocorrelation-based method. It generates a train of pulses following the harmonic distribution, separated \( F_0 \) samples. The width of the pulses is calculated taking into account the window type, length and the number of FFT points. For the Hanning window this is: \( 4 \frac{N_{FFT}}{L_w} \), where \( L_w \) is the window length and \( N_{FFT} \) is the number of FFT points. The objective is to maximize the product between the logarithm of the power spectral density of the signal under study and this train of pulses. If the resolution of the high frequency harmonics is very poor it is recommended to use the linear scale instead.

The results of this algorithm prove that the harmonics peaks appreciated in the spectrum of the signal are not equally spaced. The problem is that the peaks in the spectrum are not just due to the harmonic component, but a sum of the harmonic component, subharmonics and noise, and that can make the position of the peak shift. In Figure 4.8 it can be seen how the some peaks seem to be a little “left-overestimated” while others are totally “right-overestimated”. The experiments carried out in [8] fit with this results.

![Figure 4.8. Example of the application of the rubber band algorithm](image)

4.3. Approaches and results

Most of the approaches that have been tested in this work are based on articles extracted from different publications. In all of them the frequency range under study stays below 8 kHz. It is 5 kHz in most of the cases. One of the main goals of this work is to use the proposed methods and extend them up to 20 kHz, or at least discuss the feasibility.
Ideally, we would like to be able to separate the deterministic and the stochastic part of the signal. The deterministic part consists of the sum of sinusoidal components that form the harmonics. The stochastic part would be fully described by its power spectral density. In that case, the measurement of the HNR would be straightforward, as it will be discussed further in this section.

4.3.1. Cepstrum-based signal decomposition

There are several ways of using the information provided by the cepstrum to estimate the HNR [10, 11, 1]. The lowest part of the quefrency (0-0.001 s) domain is used to estimate the spectrum of the noise floor, and the first harmonic is used to obtain a clean harmonic spectrum, to subsequently use a frequency-based estimation of the HNR:

\[
HNR = 10 \log_{10} \left\{ \frac{\sum N_{FFT/2} |S_i|}{\sum N_{FFT/2} |N_i|} \right\}
\]

Where \(S_i\) are the harmonic values extracted from the power spectrum and \(N_i\) are noise values at the harmonic locations extracted from the noise floor estimation.

Procedure

The procedure to extract the noise floor as it is described in [11] has to be adapted to meet the requirements of our particular task. It can be summarized in 4 steps:

1. Compute the spectrogram (8192-point FFT) of the signal using a Hanning analysis window (\(L_w = 4T_0\)) with 50% overlap between segments.
2. The resulting spectra are averaged, to obtain a single 8192-point FFT.
3. Compute the cepstrum of the averaged spectra.
4. Filter the cepstrum to select the first 88 cepstral coefficients (equivalent to 0.002 seconds), and transform them back to the frequency domain.

The Fourier transform of the low-pass liftered cepstrum provides a baseline that is used as a noise floor estimation.

The procedure to obtain the harmonic part is similar, only the choice of the cepstral coefficient varies. The first harmonic peak is assumed to be around 0.001 s wide, and the rest of the cepstrum is zero-padded (set to zero). The transformation of the peak back into the frequency domain is a very clean harmonic signal.
Results and conclusions

This approach is suitable for the aim of this work. This method was originally designed for signals sampled at 10 kHz, so the harmonic reconstruction was not needed because the harmonics below 5 kHz are powerful enough. The window length used in its description is too large to have good resolution in the high part of the spectrum for our signals, so a four period-long window is being used instead.

![Figure 4.9. Spectrum of a windowed segment of a vowel sample (red) and noise floor+harmonic estimation (blue)](image)

The influence of the window length in the reconstruction of the harmonic part plays a very important role in this approach. In Figure 4.10 the anti-transformation of the rahmonic peak can be seen, using two different windows. The green curve corresponds to a four period window, and the blue one to a five period window. As the analysis windows grows, the lower harmonics gain amplitude and the higher harmonics lose amplitude. This curves correspond to the harmonic power spectral density (linear scale) as a function of frequency, so it is actually a way to plot the HNR as a function of frequency. Here it can be seen how the harmonic content is weaker at some regions but then it recovers again. That means that there is probably more than one crossover frequencies.

The noise floor estimate of this method is a little overestimated at lower frequencies. An improvement to get a better estimate of the noise floor is to compute the cepstrum of the spectral estimates and then average the different noise floor estimations. The problem of this solution is that the noise averaging effect on the is lost too. The resulting estimation is closer to the real noise floor in a large frequency range, specially at low frequencies, where the original method tends to overestimate the noise floor. The improvement, though, does not seem very significant (see Figure 4.11).
The noise floor extracted using the cepstrum-based method seems to be the spectrum of the filter (vocal tract) rather than an estimation of the noise floor, as mentioned in [11, 10]. This method seems provide an estimate of where the harmonics are “leaning”, but that does not have to be the real noise floor. The conclusion is that this estimation of the noise floor also contains the subharmonics due to the jitter and shimmer phenomena, and is therefore wrong.

In the case of a sawtooth signal with additive Gaussian noise this algorithm performed well. The difference between the real overall HNR and the calculated one is below 2 dB. The difference is that averaging the different spectra in this case really reduces the noise, while in real speech cases averaging only leads to the blurring of the highest harmonics. As well, in this case there are no subharmonics.

### 4.3.2. Noise floor estimation: consecutive period subtraction

This method assumes that the signal consists of two components: a harmonic component which is the periodic pattern that repeats from cycle to cycle, and an additive noise component which produces the cycle-to-cycle differences. Therefore, subtracting two consecutive periods (the voice signal features should not change much from period to period) only the noise component would remain [7, 6].

The period separation algorithm A.3 has been used to separate the signal. Every period has been resampled to the maximum length, using sinc interpolation. After the subtraction of every two consecutive periods, every resulting signal has been Fourier transformed, and the spectra have been averaged.
This algorithm seems to work better than the cepstrum-based one, providing a less overestimated noise. It is important to mention that a better precision in the selection of the periods would improve the quality of this estimation. However, this method only provides an estimation of the noise floor, and it is not enough to calculate the HNR or HNF.

4.3.3. Cepstrum-based HNF detection algorithm

This is a whole different approach for finding the crossover frequency. The theoretical background of this method is the following: the harmonics usually drop the noise below floor in the two last octaves (5-20 kHz). As it has been explained in section , the harmonics in the quefrency domain represent the periodic component of the signal. So, by calculating the cepstrum only of the last audible octave, there should be very little energy in the harmonics. If now the 8th octave is included, that energy should increase, because the range under study contains definitely more harmonic energy from the periodic component. As the range used for the calculation of the cepstrum approaches the lowest octaves, the growth of the harmonics should accelerate.

If the same procedure is repeated, increasing the frequency range in 50 Hz every iteration, at some point, the range is going to contain harmonics that are above the noise floor level, and then the energy of the harmonics will start to faster. That point would be the harmonics-to-noise crossover frequency. The energy of the harmonics is calculated by transforming them back to the frequency domain and calculating the sum of the square values of the power spectral density.
Results and conclusions

The energy contained in the rahmonics grows as expected when the range is increased. Plotting the energy of the rahmonics as a function of the number of iteration, it can be seen that the growth is exponential for most signals, but it is not possible to assure it for every signal. Both logarithmic and linear scale have been checked. This method has been applied to averaged spectra and single spectra, to compare the behaviour of the algorithm in both cases. In some spectra the energy growth has two very differentiated slopes. As it can be seen in the figure below.
In the case of the single spectrum, there is clearly a frequency below which the harmonic content starts to be more powerful. Still, this frequency is too low, and there is clearly harmonic content above this point, so it is definitely not the HNF. The more usual pattern of the rahmonic energy growth can be seen in the green curve of Figure 4.13.

Again, the influence of the window is very important. As the window length increases, the rahmonic energy grows more slowly in the first iterations, and faster in the last ones. This makes sense, because as the high frequency blurs, the low harmonics gain amplitude and resolution. This method seems to detect regularity in the spectrum curve, so it is again limited by the voice perturbations.

This approach deserves more research. The best alternative would be to transform them back to the frequency domain for every iteration, and see the evolution in that domain, instead of just calculating the energy. This approach has a clear advantage: it does not require any precise pitch extraction algorithm, or any peak detection algorithm.

4.3.4. Period normalization

In order to reduce the influence of the window length in the analysis, the voice signal could be pre-processed to reduce its irregularities as much as possible. The aim of this method is to get rid of the cycle-to-cycle fundamental period variations, since they seem to be the most restrictive in terms of the length of the window.

The first step is to separate every period of the signal, using the period separation algorithm (see section 4.2). The next step is to normalize the periods to the same length. The longest period is used as a reference, in order to resample all the other periods without losing frequency information, using sinc interpolation. Interpolating samples is equivalent to increasing the sample rate, but reducing the number of samples involves reducing the sample rate, and therefore the useful bandwidth. After normalizing the periods are concatenated to rebuild the signal. The result is a signal where the cycle-to-cycle variations have been removed.

The concatenation is done simply by putting the resampled periods one after the other. This is not a very safe way of doing it, because clicks might appear in the resulting signal (abrupt jumps of the signal).

Results and conclusions

In Figure 4.15 a window of 10 times the fundamental period has been used to calculate the spectrogram of both the normalized and original signal. The difference between the two spectra is huge. There is also a small misalignment between them, due to the addition of frequency information after resampling.

It is difficult to say how this normalization affects the HNR of the signal, but it makes clear that the cycle-to-cycle variations of the signal are set the main limitation for the choice of the window length. The normalization of the signal...
does not seem to affect the shape of the noise floor, and now large analysis windows can be used. The estimation of the noise floor is now simple: first, the rubber band peak detector is used to find the harmonics. The second step consists of averaging the noise samples between the harmonics, and then fit a curve that contains all those points. It is also possible to calculate the harmonic envelope of the signal (see Figure 4.16).

This method has not been used to measure the HNR, because the normalization might well affect the properties of the signal. Since the small jump in the signal occurs exactly after the same number of samples, i.e., the normalized period length, it makes sense that clear peaks appear at that particular frequency. On the other hand, in some way this method gets rid of the irregularities that we want to measure. At least it can be used to obtain a good estimation of the noise level, and then continue analyzing with the original signal.

This method just reduces the period/frequency variations, but it could be improved by using FDA (Functional Data Analysis) to reduce the amplitude variations [7]. FDA uses a more complex non-linear normalization that includes expansion or compression of the time scale in order to get a better match of the periods in amplitude and phase.
Figure 4.14. Analyzed spectra. In the upper spectrum the red part indicates the point where the slope change in the rahmonic energy occurs. The lower spectrum is the result of averaging all the spectra of the windowed signal segments. There is no clear slope change in this case.
Figure 4.15. Normalized (green) vs unnormalized spectrum

Figure 4.16. Noise floor and harmonic envelope
5. Conclusions

Only the methods that involve some kind of signal pre-processing seem to be suitable for the measure of the HNF. The perturbations present in the raw voice signal make it impossible to estimate the HNR as a function of frequency or the HNF using the Fourier transform directly, even for healthy voices. There is no analysis window length that allows to make a estimation of the noise floor level and locate the harmonics positions for the whole spectrum at the same time. Moreover, the noise floor reached at low frequencies for long windows seems to contain more subharmonic power due to jitter and shimmer than power from the noise generated at the vocal folds.

The only approach involving signal pre-processing in this work is the concatenation of normalized periods. The results of the simple concatenation used in the algorithm show already a big difference between the pre-processed and the raw signal. A consequence of this simple reduction of the voice perturbations is that the window length can be increased further. There is still a limit, but after pre-processing the voice signal, all the previously discarded methods can be considered again, since the dependence of the window length has been reduced as well.

It is difficult to predict how signal pre-processing can affect the value of the HNF, but regardless of this, it is the only alternative to be able to compute it, so it deserves some research.

The window length is a very important parameter in all the methods, even after pre-processing. The main problem is that all of them require at some point the use of the Fourier transform to compute the HNR as a function of frequency, and thus the window length is always a factor in the equation. This gives two options: find a correction factor that compensates for the different window lengths, or find a way to use always the same window length. In order the maximum possible period length (for $F_0 = 50$ Hz for example) should be used to normalize all the periods. This would involve interpolating all the signals, but the noise added in the interpolation should be less than the deviation caused by using different window lengths.

In the end it seems that there can be more than one crossover frequency, because in some voices, the harmonic part of the voice disappears in some parts of the spectrum, but then it recovers again. The assessment of the vocal fold behaviour can not be reduced to a simple frequency value as expected, but the crossover frequencies can still be used to define different regions in the spectrum. A ratio between harmonic regions to harmonic-free regions could be a good indicator of the function of the vocal folds.
6. Further work

FDA algorithm + shimmer reduction

The next step in signal pre-processing would be to try to use a better algorithm in order to reduce the perturbations in the voice signal. For this purpose, the next step would be to use Functional Data Analysis (FDA) [7, 6]. This articles use a sophisticated nonlinear time normalization to find the deterministic part of the signal. After normalizing the periods to the same length, the algorithm finds a time wrapping function that uses compression and expansion of the time scale in order to reduce the differences between periods. This algorithm uses a time-based approach to calculate the overall HNR value, but a small adaption could make it useful for our purpose: instead of calculating the time-based HNR after finding the time wrapping functions, the wrapped periods could be concatenated to have a non-linearly normalized signal. An algorithm to reduce shimmer could also be used to reduce the perturbations even more.

Research on the frequency structure of the noise

There is little knowledge about the expected frequency structure of the noise present in voiced signals. More research on this topic is necessary. It would be helpful to find out the influence of the subharmonic power in the estimation of the noise floor provided by the different methods tested in section 4. As well, more knowledge about the structure of the noise would give us more tools to isolate the stochastic and deterministic components of the voice signal.

Algorithms robust to severe voice disorders?

In this work little testing has been done with disordered voices. This is of major importance, because in the end the search for the HNF will have to be done in this kind of voices. For a first approach the use of healthy voices is the most sensible, but some research in this direction is important to find the limitations the algorithms. The problem is that most algorithms for pitch extraction, or period recognition are not totally robust to voice perturbations. If the signal has severe perturbations, the pitch extraction algorithm for instance will start double or triple the period estimation, due to the non-similarity between consecutive periods.
Bibliography


Appendix A
Matlab Code

A.1 Autocorrelation-based \( T_0 \) extraction

```matlab
function \[x2,T0,T0real,Lw,Nw\] = AUTOCORR(x,fs,OF,Np)
% AUTOCORR Gives an estimation of the evolution of the fundamental
% period \( T_0 \). The value of \( T_0 \) is updated every \( Np \cdot T_0 \)
% samples/seconds (which is also the size of
% \( Lw \) (analysis window length)
%
% \[x2,T0,T0real,Lw,Nw\] = AUTOCORR(x,fs,OF,Np);
%
% INPUT:
% x => signal from wav
% fs => samling frequency
% OF => overlapping factor (in %)
% Np => number of periods for window length (Lw).
%
% OUTPUT:
% x2 => adecuated input signal (shorter, to avoid strange
% things at the beginning and end of the file)
% T0 => \( T_0 \) estimation (sample precision)
% T0real => \( T_0 \) estimation (below sample precision.
% See Acknowledgements
% Lw => window length (\( Np \cdot \text{estimated}_{T0} \))
% Nw => Number of windows that fit in the new signal \( x2 \)
%
% ACKNOWLEDGEMENTS
% http://www.phon.ucl.ac.uk/courses/spsci/matlab/lect10.html
% "Algorithm to find a period time, interpolated between the
% sampling instants", STL-QPSR 1/1991)
%
%%%% FIRST ESTIMATION OF PITCH
maxT=ceil(fs/50);                 % minimuam speech Fx at 50Hz
minT=floor(fs/500);                 % maximum speech Fx at 500Hz
r=xcorr(x,maxT,'coeff');
len=length(r);
if rem(len,2)==0    %EVEN
    r=r(len/2+1:end);
else                %ODD
    r=r((len+1)/2:end);
end
[rmax,Tx]=max(r(minT:maxT));
T0=Tx+minT-2;
```

34
Window length (Lw) is Np times the period length
corr=1.02; % correction factor for higher T0
Lw=10*round((Np*T0*corr)/10);

NOVERLAP=Lw*OF; % OF% of overlapping
HOP=Lw-NOVERLAP;

Window-by-window ANALYSIS
Nw=floor( (length(x)-Lw)/HOP +1 );
%Nw = Maximum number of windows that fit in the window length
x2=x(HOP+1:HOP*(Nw-2));
%OBS! ADECUATE SIGNAL LENGTH (remove zeros added by Soundswell Editor)
Nw=(length(x2)-Lw)/HOP+1;

X=zeros(Lw,Nw);

for i=1:Nw
    for j=1:Lw
        X(j,i)=x2((i-1)*HOP+j);
    end
end

% AUTOCORRELATION for every WIDOW to check f0 evolution (windows overlap, to smooth evolution)
T0=zeros(Nw,1);
for i=1:Nw
    r=xcorr(X(:,i),maxT,'coeff');
    len=length(r);
    r=r((len+1)/2:end);

    [rmax,Tx]=max(r(minT:maxT));
    T0(i)=Tx+minT-2;
end

ESTIMATION IMPROVEMENT
T0real=zeros(Nw-1,1); %OBS!! This vector has one less value
for i=1:Nw-1
    Np=floor(Lw/T0(i)); %number of periods in the current window (i)
    T0real_temp=0; % since there is more than one period per window, the corrected T0 is averaged.
    for p=1:Np
        x_w_p=(p-1)*T0(i)+(i-1)*HOP;
        for j=1:T0(i)
            d(j).=5* ( x2(j+1 + x_w_p)-x2(j + x_w_p) + ...
                       x2(T0(i)+j+1 + x_w_p)-x2(T0(i)+j + x_w_p) )
            e(j)= x2(j + x_w_p) - x2(T0(i)+j + x_w_p);
        end
        deltaT=sum(d.*e)/sum(d.^2);
        T0real_temp(p)=T0(i)+deltaT;
    end
    T0real(i)=mean(T0real_temp);
end
A.2 “Rubber band” peak detection algorithm

Rubberband

```matlab
function [f0c,dt,fi_ini,meas_new]=rubberband(f0f,Ptest,width)
% rubberband  Calculates the product between Ptest and a deltatrain
% of fundamental frequency between
% [f0f*.95 - f0f*.1.05] and then selects the maximum,
% to give the best f0 estimation (f0c). The increment
% between iterations is defined by 'inc'.
%
% [f0c,dt,fi_ini,meas_new]=rubberband(f0f,Ptest,width)
% INPUT:
% f0f     => first fundamental frequency estimation (in Hz)
% Ptest   => power spectrum to fit (linear if harmonics are not very
% clear, dB if they are)
% width   => Width of the harmonics (depends on window type and
% size) taken into account for the calculation
% * For Hanning: width = 2*(NFFT/Lw); Actually,
% the harmonic contains 4*(NFFT/Lw) samples
%
% OUTPUT:
% f0c       => final fundamental frequency estimation (in Hz)
% dt        => deltatrain: separation f0c,
% width of pulses = 2*width+1
% fi_ini    => contains locations of the START of every pulse, so
% the i-th pulse goes from fi_ini(i) to
% fi_ini(i)+2*width+1
% meas_new  => Debug information. Contains the evolution of the
% product between dt and Ptest, for every iteration
%
% ACKNOWLEDGEMENTS
% Sten Ternström for the idea

Ptest=Ptest+1000; % to make it positive (if working in dB)
f0n(1)=f0f*.95;   % to make sure the algorithm should start
                 % increasing f0.

[dt,fi_ini]=deltatrain(f0n(1),width);               %create delta train
meas_old=sum(Ptest.*dt);                           %multiply and sum
meas_new(1)=meas_old;

inc=0.0005;  %frequency increase

k=2;
while (f0n(k-1)<f0f*1.05)
  f0n(k)=f0n(k-1)+inc;
  [dt,fi_ini]=deltatrain(f0n(k),width);
  meas_new(k)=sum(Ptest.*dt);
end
```

```matlab
Ptest=Ptest+1000; % to make it positive (if working in dB)
f0n(1)=f0f*.95;   % to make sure the algorithm should start
                 % increasing f0.

[dt,fi_ini]=deltatrain(f0n(1),width);               %create delta train
meas_old=sum(Ptest.*dt);                           %multiply and sum
meas_new(1)=meas_old;

inc=0.0005;  %frequency increase

k=2;
while (f0n(k-1)<f0f*1.05)
  f0n(k)=f0n(k-1)+inc;
  [dt,fi_ini]=deltatrain(f0n(k),width);
  meas_new(k)=sum(Ptest.*dt);
end
```
k=k+1;
end

[Y0,I0]=max(meas_new);
f0c=f0n(I0);
[dt,fi_ini]=deltatrain(f0c,width);

MinMax

function [dtH, dtN, I_MAX, I_N]=minmax(Ptest, Ts, width)
% MinMax Alternativa to rubberband function. This function is
% designed for spectra of the normalized signal, and uses
% the fact that all the period have the same length (Ts)
% to avoid calling the rubberband function. After
% using deltatrain function to obtain a rough estimate
% of the harmonics' locations, this algorithm searches
% for the maxima inside the delta train width to %
% generate a new delta train with more accurate values.

[dtH, dtN, I_MAX, I_N]=minmax(Ptest, Ts, width)

% INPUT:
% Ptest => SIGNAL LOG SPECTRUM
% Ts => Length of the resampled period, i.e., T0ref from
% NORMALIZE.m
% width => Width of the harmonics (depends on window type and
% size) taken into account for the calculation
% * For Hanning: width = 2*(NFFT/Lw); Actually, the
% harmonic contains 4*(NFFT/Lw) samples

% OUTPUT:
% dtH => train of pulses centered on I_MAX values (harmonic
% locations). The width of the pulses is 2*width.
% dtN => train of pulses centered on I_N values (noise
% region locations). The width of the pulses is one.
% I_MAX => contains locations of the harmonics' maxima.
% I_N => contains locations of the noise floor regions.

fs=44100;
NFFT=8191;
max_samples=(NFFT+1)/2;

f0f=fs/Ts;
width=ceil(width);

[dt,fi_ini]=deltatrain(f0f,NFFT,width);

f0s=NFFT*(f0f/fs);
Nharm=floor(max_samples/f0s);
Fh=zeros(2*width,Nharm);
for i=1:Nharm
    for j=1:2*width           %% CHECK WIDTH VALUE !!!
        if (fi_ini(i)+j-1)<max_samples+1
            Fh(j,i)=Ptest(fi_ini(i)+j-1);
        end
    end
end

% RESELECt MAXIMA
for i=1:Nharm
    [Y_MAX(i),I_MAX_Fh]=max( Fh(:,i) );
    if (fi_ini(i)+I_MAX_Fh-1)<max_samples+1
        I_MAX(i)=fi_ini(i)+I_MAX_Fh-1;
    end
end

% NEW IMPROVED DELTATRAIN (not equally spaced pulses)
dtH=zeros(max_samples,1);
for i=1:length(I_MAX)
    for j=-width:width
        if (I_MAX(i)+j)<max_samples+1
            dtH(I_MAX(i)+j)=1;
        end
    end
end

%%% MINIMA
for i=1:length(I_MAX)-1
    I_N(i)=round((I_MAX(i)+I_MAX(i+1))/2);
end
dtN=zeros(max_samples,1);
dtN(I_N)=1;
dtN(1:7)=1;
dtN(end-6:end)=1;

Deltatrain

function [dt,fi_ini]=deltatrain(f0f,width)
%   deltatrain  Generates a train of pulses with a distance of
%               f0f (after converting to samples) between them.
%               
%   [dt,fi_ini]=deltatrain(f0f,width)
%   
%   INPUT:
%   f0f   => distance between the center of the pulses (in Hz)
%   width => Width of the pulses.
%   
%   OUTPUT:
%   dt    => deltatrain: separation f0c,
%          width of pulses = 2*width+1
fs=44100;
NFFT=8191;
max_samples=(NFFT+1)/2;
width=floor(width);
f0s=(f0f/fs)*NFFT;
Npulse=floor(max_samples/f0s);  %number of deltas/harmonics that fit
dt=zeros(max_samples,1);
for i=1:Npulse
  fi=round(i*f0s);
  if fi-width>0, fi_ini(i)=fi-width; else fi_ini(i)=1; end
  for j=-width:width
    if ((fi+j)<max_samples+1)&&(fi+j)>0
      dt(fi+j)=1;
    end
  end
end

A.3 Period separation and normalization

Requires SincResample.m function, that can be found at MatlabCentral

function [PerP,PerR,xr,T0pp,FOUT,Imin]=NORMALIZE(yini)
% NORMALIZE Uses the "period separation algorithm" to return a
% version of the input signal cycle-to-cycle T0
% perturbations are gone (xr).
% Includes execution of AUTOCORR.m
% [PerP,PerR,xr,T0pp,FOUT,Imin]=NORMALIZE(yini)
% INPUT:
% yini => signal from wav
% OUTPUT:
% PerP => Matrix max(T0)x(number of periods): Contains all
% the periods of the signal (except first and last)
% with its original size. Outside period length NaN
% values are used.
% PerR => Matrix ref(T0)x(number of periods): Contains all
the periods of the signal (except first and last) with the reference period length (maximum).

xr => Normalized signal. Concatenation of the normalized periods contained in PerR
T0pp => Vector containing the length of every period
FOUT => Highest sampling frequency due to the normalization (the shortest period provides the highest sampling frequency due to the interpolation).
Imin => Location of in the PerP or PerR of the shortest period

ACKNOWLEDGEMENTS for WM Method (period separation algorithm)
Inro R. Titze and Haixiang Liang
"Comparison of F0 Extraction Methods for High-Precision Voice Perturbation Measurements", Journal of Speech and Hearing Research,
year = 1993
volume = 36
number = 6
pages = 1120-1133

MatlabCentral "The MathWorks"’s open exchange community where SincResample.m was found.

% Call to AUTOCORR, for T0 estimation and yini adecuation (see AUTOCORR.m)
fs=44100;
Np=4;
OF=.5;  % 50% overlapping
[y,T0,T0real,Lw,Nw] = AUTOCORR(yini,fs,OF,Np);

f0m=fs/mean(T0real);
[b,a] = butter(8,f0m*1.5/(fs/2),’low’);
%[b,a] = butter(8,f0s(4)*1.5/max_samples,’low’);
yf=filter(b,a,y);

thresh = 0;
len = length(yf);
zc = (yf >= thresh) - (yf < thresh);

%count = sum((zc(1:N-1) - zc(2:N)) ~= 0);
zc2=(zc(1:len-1) - zc(2:len));
k=1;
%cross2=zeros(length(zc),1);
for i=1:length(zc2)
    if zc2(i)==2
        cross(k)=i;         %POSITION BEFORE CROSING (to negative)
        %cross2(i)=.07;     %DELTATRAIN every cycle
        k=k+1;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%   LOCATION OF MINIMUMS as first T0 estimate   %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
PERC=0.02;
Pmin=zeros(2,1);

[Y,I]=min(y(cross(1):cross(2)));
Pmin(1)=I+cross(1)-1;

T0m=round(mean(T0real));
for i=2:length(cross)
Pmin(i)=Pmin(i-1)+T0m;
    if Pmin(i)>length(y), break, end
    J1=floor(Pmin(i)-PERC*(Pmin(i)-Pmin(i-1)));
    J2=ceil(Pmin(i)+PERC*(Pmin(i)-Pmin(i-1)));
    if J2>length(y), break, end
    [Y,I]=min(y(J1:J2));
Pmin(i)=I+J1-1;
end

%%%%%%% Pmin INTERPOLATION? CODE HERE
%%%%%%% END Pmin INTERPOLATION

T0pp=Pmin(2:end)-Pmin(1:(end-1)); % SKIPS first and last INCOMPLETE CYCLES
PerP=NaN(max(T0pp),length(T0pp));
for i=1:length(T0pp) %number of cycles
    for j=1:max(T0pp)
        if j<T0pp(i)+1
            if (Pmin(i)+j-1)<length(y)+1
                PerP(j,i)=y(Pmin(i)+j-1);
            else
                PerP=PerP(:,1:i-1);
            end
        end
    end
end

%RESAMPLE EVERY PERIOD to the same (maximum) length
Tref=max(T0pp);
%Tref=min(T0pp); %for minimum instead
PerR=NaN(Tref,size(PerP,2));

FOUT=1;
for i=1:size(PerP,2)
    [PerR(:,i), FOUTnew]=SincResample(PerP(1:T0pp(i),i), Tref, fs);
    if FOUTnew < FOUT
        FOUT=FOUTnew;
        Imin=i;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% RECONSTRUCTION AFTER NORMALIZING %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
xr=PerR(:,1);
for i=2:size(PerR,2)
xr=[xr;PerR(:,i)];
A.4 Cepstrum-based HNF detection

Requires Curve Fitting Toolbox, spec2cep.m and cep2spec.m routines, which can be found at http://labrosa.ee.columbia.edu/matlab/rastamat/.

function HNF_DETECTION(Ptest,peaki1,peaki2,peaki3,peaki4)
% HNF_DETECTION      Calculates and plots the evolution in the 
%                      energy of the first rahmonic, jumping by 
%                      50 samples every iteration.
% %
% HNF_DETECTION(Ptest,peaki1,peaki2,peaki3,peaki4)
% % INPUT:
% % Ptest  => power spectrum to fit (dB to give high harmonics 
%     more weight)
% % peakiX  => Location of the X-th rahmonic peak in samples
% %
% ACKNOWLEDGEMENTS
% % Sten Ternström for the idea
% % Dan Ellis at “Columbia University” for the routines
% % spec2cep.m and cep2spec.m

Rwidth=22;       %rahmonic total width:
                  % 1ms (quefrency) x fs = 2*22 samples
fs=44100;
NFFT=8191;
max_samples=(NFFT+1)/2;

init=700;           % number of samples for the first iteration
                    % (from end-700 to end)
finit=4000;         % number of samples for the penultimate iteration
                    % (last iteration includes whole spectrum)
cepstra=zeros(max_samples,(finit/init+1)/50+1);
Ptest2=zeros(max_samples,(finit/init+1)/50);
k=1;
for D = init:50:finit
    for i=1:length(Ptest)
        if i>length(Ptest)-D-1
            Ptest2(i,k)=Ptest(i);
        else
            Ptest2(i,k)=Ptest(end-D);
        end
    end
    cepstra(:,k) = spec2cep(Ptest2(:,k), max_samples);
    k=k+1;
end
wavplay(xr,fs);
end
Ptest2(:,end)=Ptest;
 cepstra(:,end) = spec2cep(Ptest, max_samples);

ff2=linspace(0,fs/2,max_samples);
figure(1)
for i = 1:size(cepstra,2)
    drawnow
    subplot(211)
    plot(ff2,10*log10(Ptest),'-k',ff2,10*log10(Ptest2(:,i)),'r--')
    ylim([-150 -50])

    subplot(212)
    plot(abs(cepstra(:,i)))
    ylim([-1 50])
    xlim([1 1000])
    % WAITING TIMER
    t1=clock;t2=clock;
    while etime(t2, t1)<0.2
        t2=clock;
    end
    % ***********************************************************
    %           ENERGY in Rahmonic
    % ***********************************************************
    if peaki1~=0, Peak1s=cep2spec([zeros(peaki1-Rwidth-1,1);...
        cepstra(peaki1-Rwidth:peaki1+Rwidth,i);...
        zeros(max_samples-(peaki1+Rwidth),1)],max_samples); end
    if peaki2~=0, Peak2s=cep2spec([zeros(peaki2-Rwidth-1,1);...
        cepstra(peaki2-Rwidth:peaki2+Rwidth,i);...
        zeros(max_samples-(peaki2+Rwidth),1)],max_samples); end
    if peaki3~=0, Peak3s=cep2spec([zeros(peaki3-Rwidth-1,1);...
        cepstra(peaki3-Rwidth:peaki3+Rwidth,i);...
        zeros(max_samples-(peaki3+Rwidth),1)],max_samples); end
    if peaki4~=0, Peak4s=cep2spec([zeros(peaki4-Rwidth-1,1);...
        cepstra(peaki4-Rwidth:peaki4+Rwidth,i);...
        zeros(max_samples-(peaki4+Rwidth),1)],max_samples); end
    if peaki1~=0, Epeak1(i)= sum(Peak1s.^2); else Epeak1(i)=0; end
    if peaki2~=0, Epeak2(i)= sum(Peak2s.^2); else Epeak2(i)=0; end
    if peaki3~=0, Epeak3(i)= sum(Peak3s.^2); else Epeak3(i)=0; end
    if peaki4~=0, Epeak4(i)= sum(Peak4s.^2); else Epeak4(i)=0; end
    EpeakT(i)=Epeak1(i)+Epeak2(i)+Epeak3(i)+Epeak4(i);
end
figure(2),
plot(EpeakT)
A.5 Harmonic envelope and noise floor extraction

Requires Curve Fitting Toolbox (MATLAB) and normalized signal (see section B.3).

```matlab
function [Nfloor, Henv]=fitting(Ptestt, Ts, width)
% fitting    Calculates the position of the harmonics and of the
% noise floor regions to provide a harmonic envelope
% curve and a noise floor estimation. It is designed
% to work with the normalized signal (xr output
% from NORMALIZE.m)
% [Nfloor, Henv]=fitting(Ptestt, Ts, width)
%
% INPUT:
% Ptestt => power spectrum to fit (dB to give high harmonics
% more weight)
% Ts     => Length of the period. resampled period, i.e., T0ref from
% NORMALIZE.m
% width  => Width of the harmonics (depends on window type and size)
%          taken into account for the calculation
%          * For Hanning: width = 2*(NFFT/Lw); Actually, the
%            harmonic contains 4*(NFFT/Lw) samples
%
% OUTPUT:
% Nfloor    => Noise floor estimation
% Henv      => Envelope of the harmonics

NFFT=8191;
max_samples=(NFFT+1)/2;

% SHAPE PRESERVING
f = fittype('pchipinterp');
% types: cflibhelp interpolant
% similar: f=fittype('splineinterp');

% From now on the code can be used for a loop (more than one Ptestt)
[dtH, dtN, I_MAX, I_N]=minmax(Ptestt, Ts, width);
dtHMAX=zeros(max_samples,1);
dtHMAX(I_MAX)=1;
PNm=zeros(max_samples,1);
PNm(1:7)=Ptestt(1:7);
PNm(end-6:end)=Ptestt(end-6:end);

% AVERAGE samples between harmonics to get a better
% estimate of the noise floor
for j=1:length(I_N)
    if (I_MAX(j)+1)-width > I_MAX(j)+width
        PNm(I_N(j))=mean( Ptestt(I_MAX(j)+width:I_MAX(j+1)-width) );
    else

44
PNm(I_N(j)) = Ptestt(I_N(j)) * dtN(I_N(j),i);
end
end

outliersN = excludedata(tt',dtN,'indices',find(PNm==0));
outliersH = excludedata(tt',dtMAX,'indices',find(dtMAX==0));

sN = fitoptions('Exclude', outliersN,'Method', 'PchipInterpolant');
sH = fitoptions('Exclude', outliersH,'Method', 'PchipInterpolant');

cfunN = fit(tt',PNm,f,sN);
cfunH = fit(tt',Ptestt,f,sH);

Nfloor = feval(cfunN,tt);
Henv = feval(cfunH,tt);